**TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA**

**KHOA CÔNG NGHỆ THÔNG TIN**


# ĐỒ ÁN MÔN HỌC

## MÔN HỌC: MẠNG NƠ-RON

**ĐỀ TÀI:**

## TÌM HIỂU NGÔN NGỮ PYTHON, THƯ VIỆN PYTORCH VÀ VIẾT ỨNG DỤNG PHÂN LỚP HÌNH ẢNH

## (BIRD, CAT, FROG, HORSE)

**TP. HỒ CHÍ MINH, THÁNG 08 NĂM 2020**

**TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA**

**KHOA CÔNG NGHỆ THÔNG TIN**


# ĐỒ ÁN MÔN HỌC

## MÔN HỌC: MẠNG NƠ-RON

**ĐỀ TÀI:**

## TÌM HIỂU NGÔN NGỮ PYTHON, THƯ VIỆN PYTORCH VÀ VIẾT ỨNG DỤNG PHÂN LỚP HÌNH ẢNH

## (BIRD, CAT, FROG, HORSE)

**Nhóm báo cáo:**

**Nguyễn Tiểu Phụng**

**Huỳnh Đức Anh Tuấn**

**Giảng viên hướng dẫn:**

**Th.s Ngô Thanh Tú**

**TP. HỒ CHÍ MINH, THÁNG 08 NĂM 2020**

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy cô giảng viên trong khoa Công nghệ thông tin trường Đại học Trần Đại Nghĩa. Và đặc biệt là thầy Thạc sĩ Ngô Thanh Tú – giảng viên học phần "Mạng Nơ-ron" đã tận tình hướng dẫn, truyền đạt kiến thức và kỹ năng cần thiết để em có thể hoàn thành đồ án môn học này.

Tuy nhiên, trong quá trình tìm hiểu và nghiên cứu đề tài, do kiến thức chuyên ngành và thời gian còn hạn chế em vẫn còn nhiều thiếu sót trong quá trình tìm hiểu, thực hiện, đánh giá và trình bày về đề tài. Rất mong được sự quan tâm, góp ý của các thầy cô và giảng viên bộ môn để đồ án môn học của em được hoàn chỉnh hơn.

Xin chân thành cảm ơn!

# MỞ ĐẦU

## 1. Lý do chọn đề tài

- Deep Learning là một thuật toán dựa trên một số ý tưởng 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 nhiều trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên.
- Hiện nay rất nhiều các bài toán nhận dạng, phân loại sử dụng deep learning để giải quyết do deep learning có thể giải quyết các bài toán với số lượng, kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống.
- Convolutional Neutal Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được nhưng hệ thống thông minh với độ chính xác cao. Trong bài báo cáo này, chúng em tập trung nghiên cứu về "Mạng Nơ-ron nhân tạo" cũng như ý tưởng phân lớp ảnh dựa trên mô hình CNNs (Image Classification). Và áp dụng để xây dựng ứng dụng phân lớp ảnh "bird", "cat", "frog" và "horse".

## 2. Cấu trúc đồ án

- Chương 1: Tìm hiểu ngôn ngữ lập trình Python
- Chương 2: Tìm hiểu mạng nơ-ron và thư viện Pytorch
- Chương 3: Xây dựng ứng dụng

# MỤC LỤC

# CHƯƠNG 1: TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON

## 1.1. Giới thiệu ngôn ngữ lập trình Python

### 1.1.1. Lịch sử phát triển

- Ngôn ngữ Python được Guido van Rossum tạo ra và được phát hành lần đầu tiên vào tháng 2 năm 1991.
- Python khá giống Perl, Ruby, Scheme, Smalltalk và Tcl.
- Python được phát triển trong một dự án mã mở do một tổ chức phi lợi nhuận Python Software Foundation quản lý.
- Python được phát triển để chạy trên nền Unix. Nhưng theo thời gian, nó đã "bành trướng" sang mọi hệ điều hành từ MS-DOS đến MAC OS, OS/2, Windows, Linux và một số điều hành khác thuộc họ Unix.
- Python là ngôn ngữ bậc cao (high-level), có hình thức sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cho phép người sử dụng viết mã với số lần gõ phím tối thiểu.
- Python cũng là một trong những ngôn ngữ phổ biến nhất thế giới.
- Python không phải được đặt theo tên của con rắn thần Python trong thần thoại Hy Lạp đâu. Rossum là fan của một sê-ri chương trình hài cuối những năm 1970, và cái tên “Python” được lấy từ tên một phần trong sê-ri đó “Monty Python’s Flying Circus”.

### 1.1.2. Phiên bản

Bảng 1.1 các phiên bản Python đã phát hành

| **Phiên bản** | **Ngày phát hành** |
|---|---|
| Python 1.0 (bản phát hành chuẩn đầu tiên)<br>Python 1.6 (Phiên bản 1.x cuối cùng) | 01/1994<br>05/09/2000 |

| Bảng 1.1 các phiên bản Python đã phát hành | |
|---|---|
| **Phiên bản** | **Ngày phát hành** |
| Python 2.0 (Giới thiệu list comprehension)<br>Python 2.7 (Phiên bản 2.x cuối cùng) | 16/10/2000<br>03/07/2010 |
| Python 3.0 (Loại bỏ cấu trúc và mô-đun trùng lặp) | 03/12/2008 |
| Python 3.3 | 2012 |
| Python 3.5 (hỗ trợ cho các byte và bytearray) | 13/09/2015 |
| Python 3.6(cải tiến đáng kể trong thư viện tiêu chuẩn) | 23/12/2016 |
| Python 3.7(Cải tiến mô hình dữ liệu Python) | 27/06/2018 |
| Python 3.8 (Được bổ sung nhiều tính năng mới) | 14/10/2019 |
| Python 3.9 (Loại bỏ hết các tính năng dùng tương thích ngược ở bản 2x) | 27/04/2020 |

### 1.1.3. Một số điểm khác nhau giữa phiên bản 3x và 2x

#### 1.1.3.1. Sự khác biệt ở hàm PRINT

- Ở 2x print ‘a’,’b’ dễ gây hiểu lầm với kiểu dữ liệu Tuple khi ta truyền nhiều đối số vào parantheses.
- Ở phiên bản 3x hàm print dùng ().

| Python 2x | Python 3x |
|---|---|
| print 'Hello, World!'<br>print ('Hello, World!')<br>>>> Hello, Word!<br>>>> Hello, Word! | print ('Hello, World!')<br>print 'Hello, World!'<br>>>> Hello, World!<br>print 'Hello, World!<br>SyntaxError: invalid syntax |

#### 1.1.3.2. Toán tử DIV

- Ở Python 2x toán tử chia (/) có một ý nghĩa mơ hồ cho các đối số 'số': nó trả về sàn của kết quả toán học của phép chia nếu các đối số là kiểu ints hoặc long, nhưng nó trả về một xấp xỉ hợp lý của kết quả phân chia nếu các đối số là kiểu float hoặc phức. Vấn đề này lại được giải quyết ở bản 3x.

| Python 2x | Python 3x |
|---|---|
| print '3 / 2 =', 3 / 2 | print ('3 / 2 =', 3 / 2) |
| print '3 // 2 =', 3 // 2 | print ('3 // 2 =', 3 // 2) |
| print '3 / 2.0 =', 3 / 2.0 | print ('3 / 2.0 =', 3 / 2.0) |
| print '3 // 2.0 =', 3 // 2.0 | print ('3 // 2.0 =', 3 // 2.0) |
| >>>3 / 2 = 1 | >>>3 / 2 = 1.5 |
| >>>3 // 2 = 1 | >>>3 // 2 = 1 |
| >>>3 / 2.0 = 1.5 | >>>3 / 2.0 = 1.5 |
| >>>3 // 2.0 = 1.0 | >>>3 // 2.0 = 1.0 |

#### 1.1.3.3. Kiểu STRING mặc định

- Python 2x có các kiểu str () thuộc kiểu ASCII, riêng biệt unicode (), nhưng không có kiểu byte.
- Python 3x có các chuỗi Unicode (utf-8) và 2 lớp byte: byte và bytearrays.

| Python 2x | Python 3x |
|---|---|
| print type(unicode('Chuỗi này giống kiểu str ở Python 3')) | print('strings are now utf-8 \u03BCnico\u0394é!') |
| >>> <type 'unicode'> | >>> strings are now utf-8 μnicoΔé! |

| | |
|---|---|
| print type(b‘Đây giống như một chuỗi str do không có kiểu byte ở 2x’)<br>>>> <type 'str'> | print(sys.version,' has', type(b' bytes for storing data'))<br>>>> 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] has <class 'bytes'> |
| print '2 chuỗi này' + b'giống nhau'<br>>>> 2 chuỗi này giống nhau | print('note that we cannot add a string' + b'bytes for data')<br>>>> Traceback (most recent call last):<br>File "<stdin>", line 1, in <module><br>TypeError: can only concatenate str (not "bytes") to str |

#### 1.1.3.4. Loại bỏ hàm XRANGE() ở bản 3x

- Việc sử dụng xrange() rất phổ biến trong Python 2.x để tạo một đối tượng có thể lặp lại.
- Ở python 3x đã loại bỏ hàm XRANGE() thay vào đó là sử dụng hàm RANGE(). Nhưng hàm range() ở bản 3x đã được tích hợp cơ chế ‘lazy avaluation’ cơ chế giải phóng bộ nhớ đã chiếm dụng của xrange() (bản 2x).

#### 1.1.3.5. Xử lý ngoại lệ

- Trong Python 3x yêu cầu xử dụng từ khóa as để xử lý ngoại lệ.

| Python 2x | Python 3x |
|---|---|
| try:<br>let_us_cause_a_NameError<br>except NameError, err:<br>print err, '--> our error message'<br>>>>name<br>'let_us_cause_a_NameError' is not defined --> our error message | try:<br>let_us_cause_a_NameError<br>except NameError as err:<br>print(err, '--> our error message')<br>>>>name 'let_us_cause_a_NameError' is not defined --> our error message |

#### 1.1.3.6. Banker’s Round

- Python 3 đã áp dụng cách làm tròn số thập phân chuẩn mới hiện nay khi kết quả là hòa (0,5) ở các chữ số có nghĩa cuối cùng. Bây giờ, trong Python 3, số thập phân được làm tròn đến số chẵn gần nhất.
- Mặc dù đó là một sự bất tiện cho tính di động của mã, nhưng nó được cho là cách làm tròn tốt hơn so với làm tròn cũ vì nó tránh được sự thiên vị đối với số lượng lớn.

| Python 2x | Python 3x |
|---|---|
| round(15.5) | round(15.5) |
| >>> 16.0 | >>>16 |
| round(16.5) | round(16.5) |
| >>>17.0 | >>>16 |

#### 1.1.3.7. Ordering Comparisons

- Python 3.0 đã đơn giản hóa các quy tắc của toán tử so sánh:
- Các toán tử so sánh thứ tự ( < , <= , > = , > ) đưa ra một ngoại lệ TypeError khi các toán hạng không có thứ tự tự nhiên có ý nghĩa. Do đó, các biểu thức như 1 < " , 0 > Không có hoặc len <= len không còn hợp lệ và ví dụ: Không < Không làm tăng TypeError thay vì trả về Sai .

### 1.1.4. Đặc điểm của Python

- Python là ngôn ngữ lập trình hướng đối tượng, bậc cao, mạnh mẽ. Ngoài ra, học Python là khá đơn giản và dễ dàng. Python cũng là một ngôn ngữ thông dịch, tức là ngôn ngữ không cần phải biên dịch ra file chạy mà đọc code đến đâu thì chạy

đến đó. Khi chạy lệnh Python ta sẽ có một giao diện dòng lệnh giống của Unix, có thể chạy từng dòng code ngay trực tiếp tại đây.

- Python có rất nhiều ưu điểm để khiến cho các nhà lập trình web yêu thích và sử dụng nó cho ngôn ngữ lập trình đầu tiên của mình đó là:
    + Đơn giản: Cú pháp đơn giản giúp cho người lập trình dễ dàng đọc và tìm hiểu.
    + Tốc độ: Python có tốc độ xử lý nhanh hơn so với ngôn ngữ PHP.
    + Tương tác: Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.
    + Chất lượng: Thư viện có tiêu chuẩn cao, Python có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.
    + Thuận tiện: Python được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay.
    + Mở rộng: Với tính năng này, Python cho phép người lập trình có thể thêm hoặc tùy chỉnh các công cụ nhằm tối đa hiệu quả có thể đạt được trong công việc.
    + Có trên tất cả các nền tảng hệ điều hành từ UNIX, MS – DOS, Mac OS, Windows và Linix và các OS khác thuộc họ Unix.
    + Tương thích mạnh mẽ với Unix, hardware, thirt-party software với số lượng thư viện khổng lồ (400 triệu người sử dụng)
    + Python với tốc độ xử lý cực nhanh, python có thể tạo ra những chương trình từ những script siêu nhỏ tới những phần mềm cực lớn như Blender 3D.

## 1.2. Hướng dẫn cài đặt bằng Anaconda/Miniconda

### 1.2.1. Giới thiệu Ananconda/Miniconda và cài đặt

- Anaconda là một Distribution miễn phí và mã nguồn mở của Python và R giúp đơn giản hóa việc cài đặt, quản lý và triển khai packages (numpy, scipy, tensorflow, ...).

- Anaconda phục vụ cho nhiều mục đích, đặc biệt trong Data Science (Khoa học dữ liệu), Machine learnig (Máy học), Big Data (Dữ liệu lớn), Image Processing (Xử lý ảnh), ...
- Anaconda hiện nay đã có hơn 20 triệu người dùng và hơn 7500 packages khoa học dữ liệu dành cho Windows, Linux và MacOS.
- Trong khi đó Spyder là 1 trong những IDE (môi trường tích hợp dùng để phát triển phần mềm) tốt nhất cho data science và quang trọng hơn là nó được cài đặt khi bạn cài đặt Anaconda.

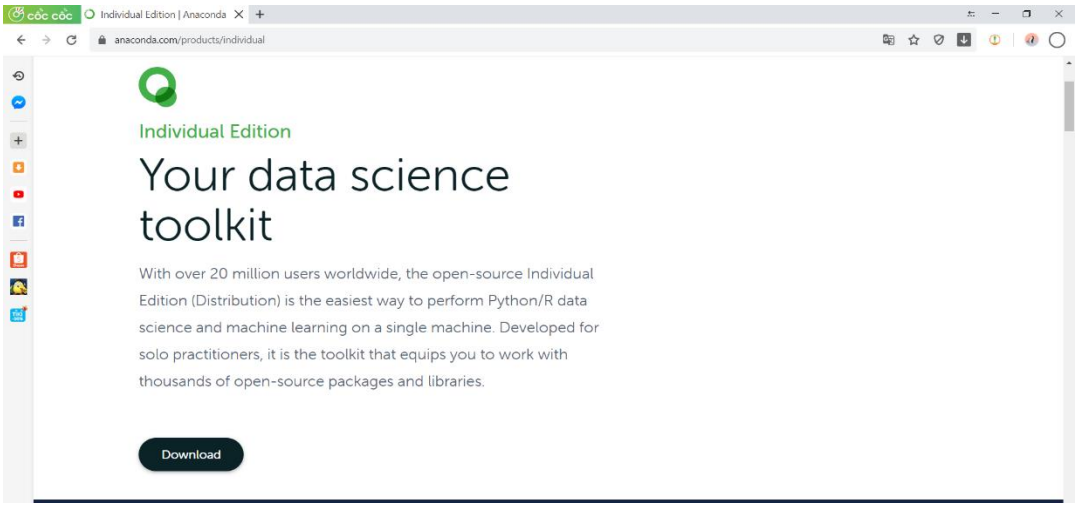
### 1.2.2. Download Anaconda/Miniconda và hướng dẫn cài đặt

#### 1.2.2.1. Các bước cài đặt

- Yêu cầu phần cứng và phần mềm:
  + Hệ điều hành: Win 7, Win 8/8.1, Win 10, Red Hat Enterprise Linux/CentOS 6.7, 7.3, 7.4, and 7.5, and Ubuntu 12.04+.
  + Ram tối thiểu 4GB.
  + Ổ cứng trống tối thiểu 3GB để tải và cài đặt.

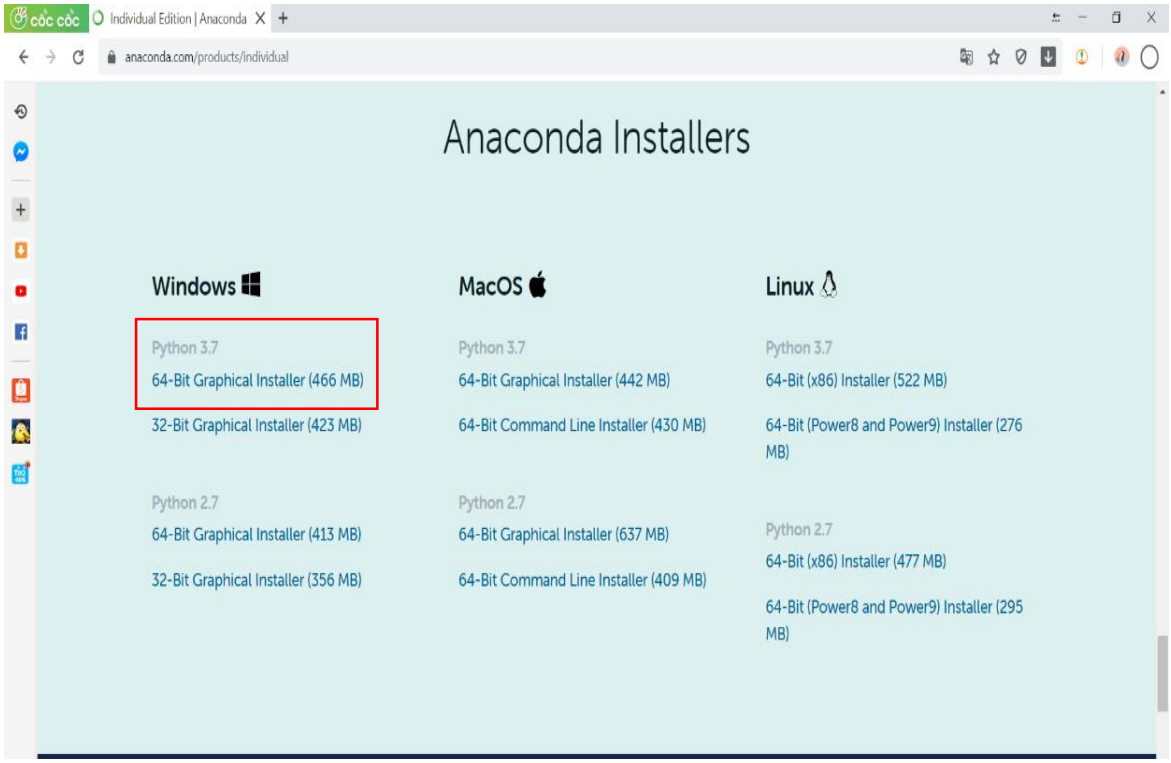
**Cài đặt:**

- Bước 1: truy cập vào trang web https://www.anaconda.com/



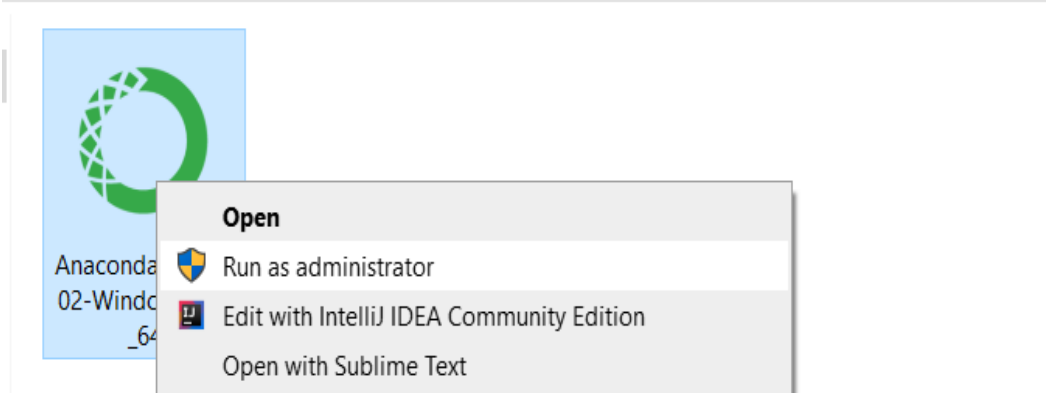

Hình 1.1 Trang chủ ananconda.com

+ Chọn xuống mục download: và chọn tải phiên bản thích hợp, ở đây em chọn hệ điều hành windows bản python 3.7 và 64-bit Graphical.



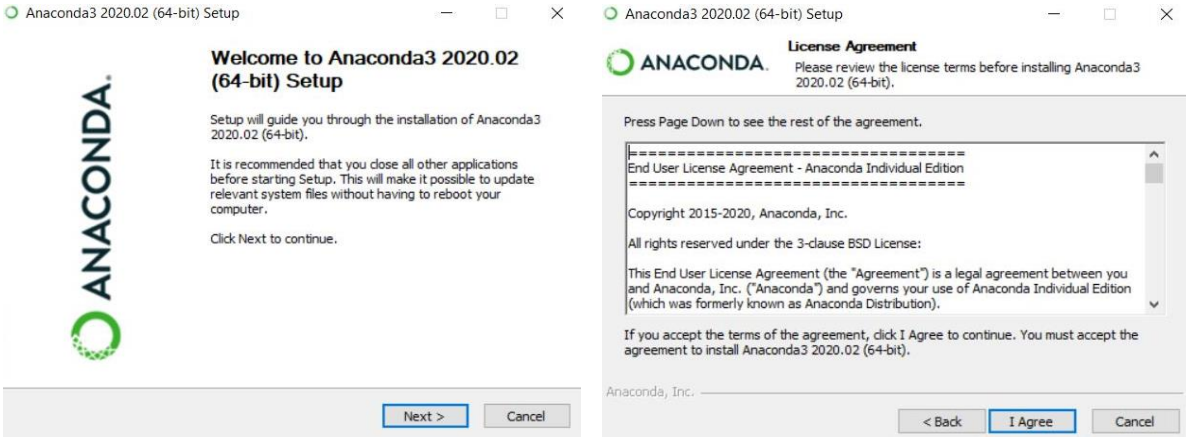

Hình 1.2 Download bản cài đặt

- Bước 2: Chạy file cài đặt với quyền admin



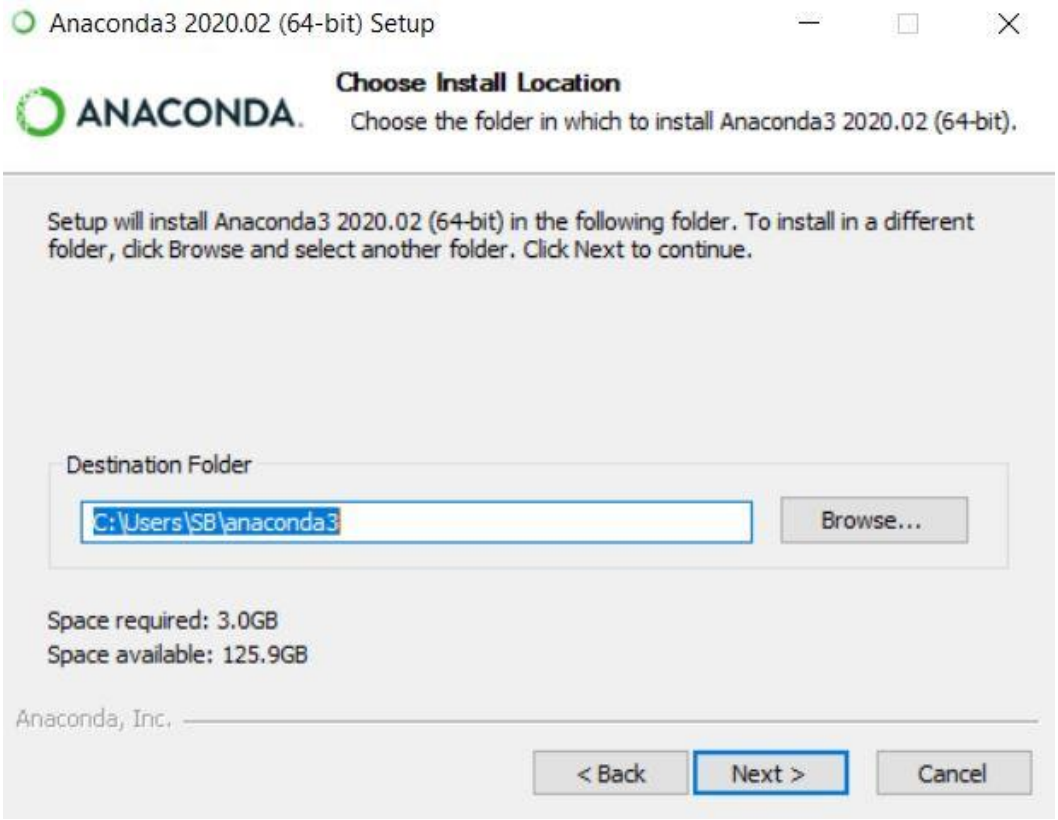

Hình 1.3 Tiến hình cài đặt

- Bước 3: Chấp nhập các yêu cầu thiết lập và tiến hình cài đặt



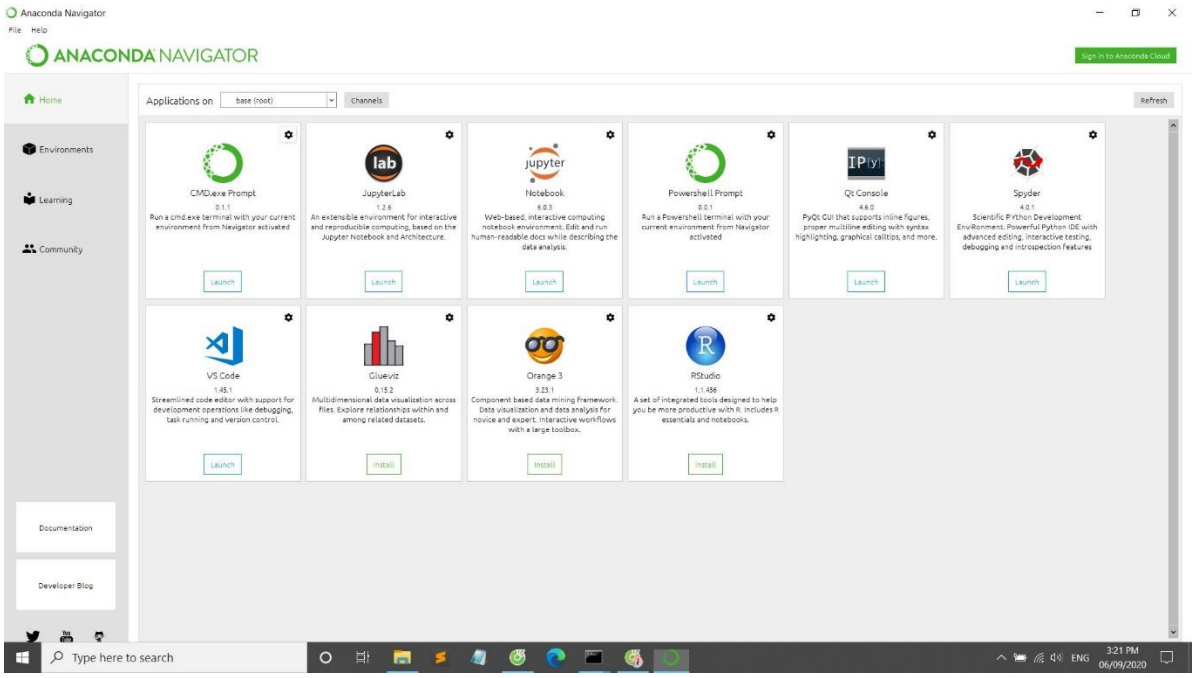

Hình 1.4 Giao diện cài đặt

- Bước 4: Chọn tài khoản và vị trí để cài đặt. Ở đây em chọn recommended và ổ C:/



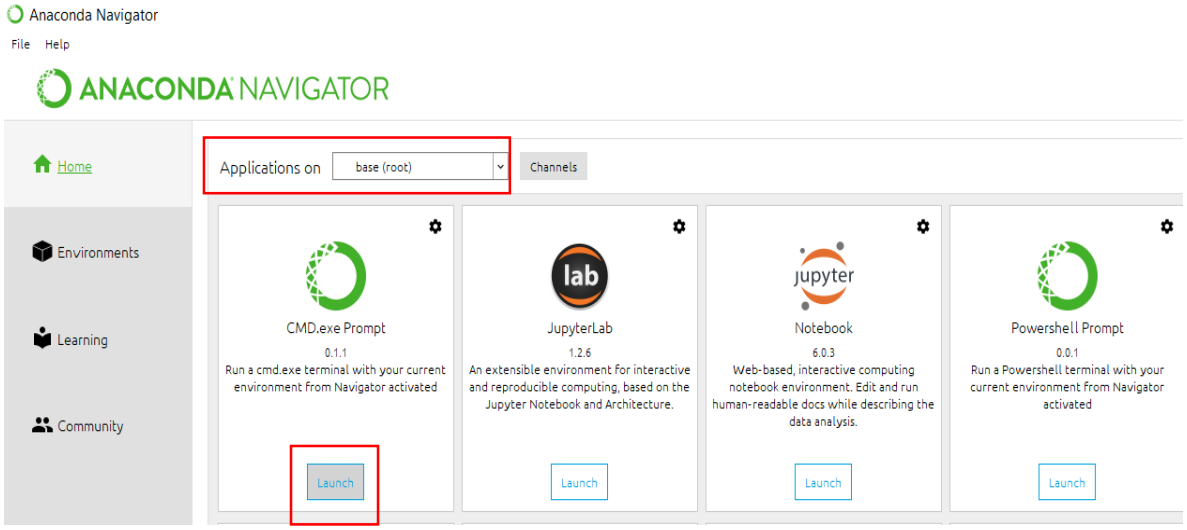

Hình 1.5 Chọn tài khoản và vị trí để cài đặt

- Bước 5: Giao diện Anaconda khi cài đặt xong



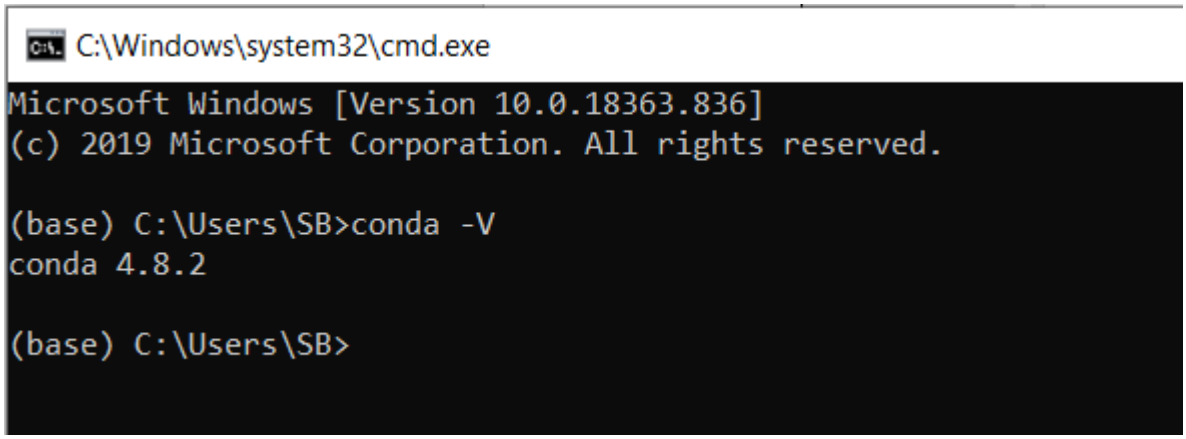

Hình 1.6 Giao diện của phần mền Anaconda

- Bước 6: Kiểm tra lại phiên bản phần mềm Anaconda

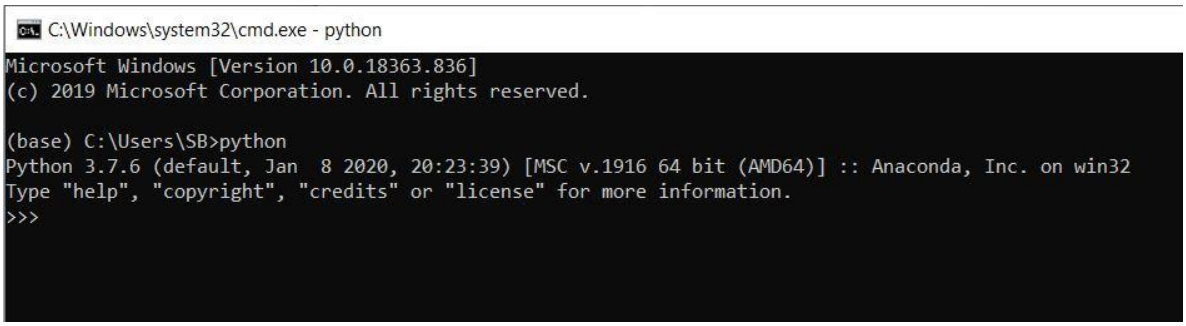



Hình 1.7 Giao diện home của Anaconda

+ Chọn Launch ở Environments base(root) để mở CMD ở môi trường conda để kiểm tra lại phiên bản cài đặt bằng lệnh: ***conda -V***




Hình 1.8 Kiểm tra phiên bản conda

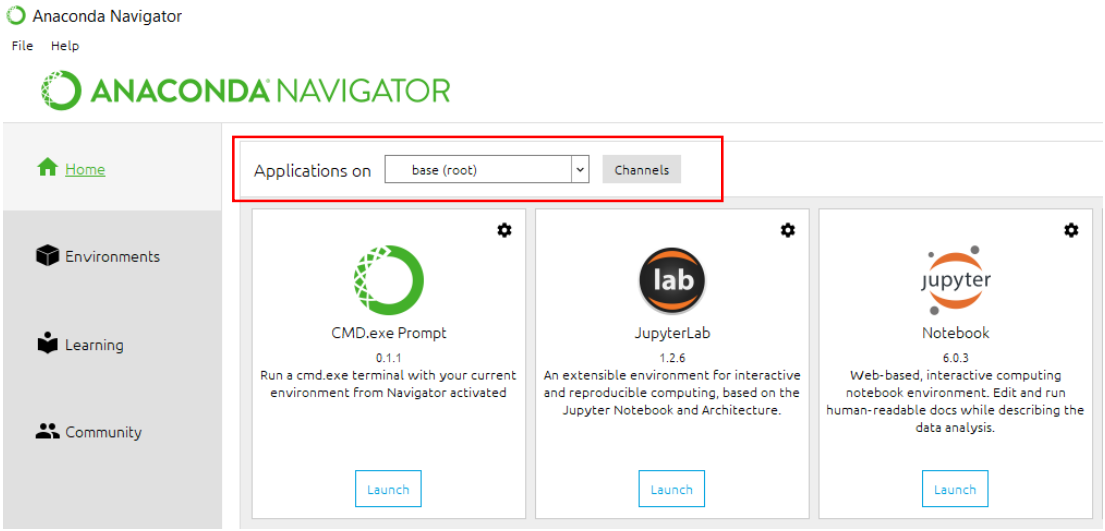



Hình 1.9 Kiểm tra phiên bản python

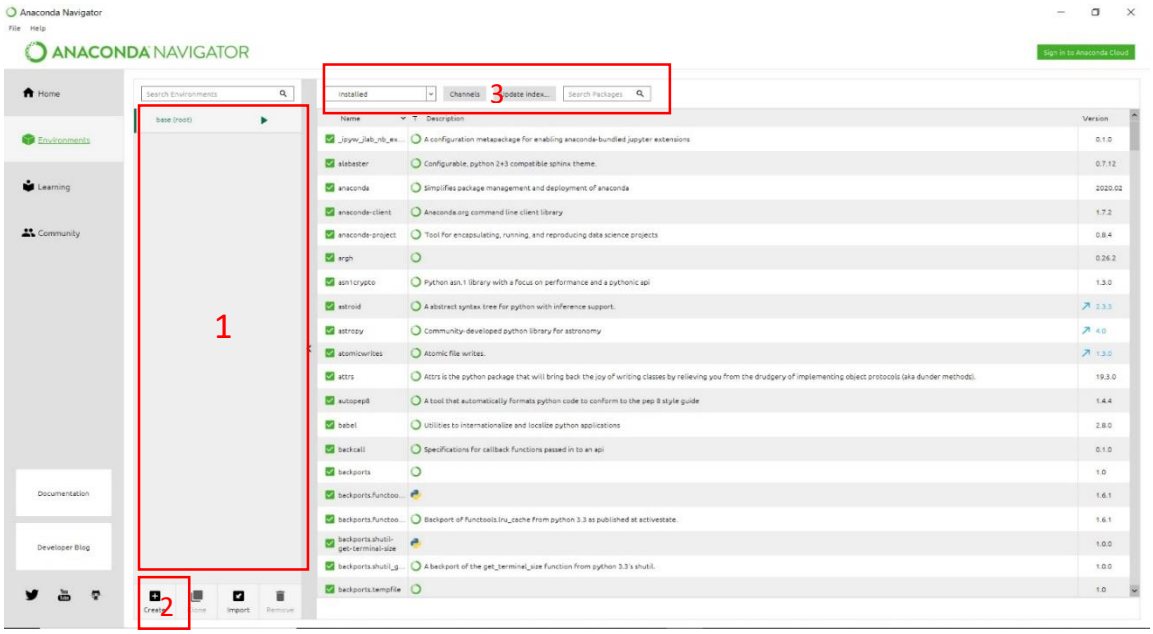
#### 1.2.2.2. Quản lý môi trường

- Với Anaconda có nhiều packages khoa học phụ thuộc vào các phiên bản cụ thể của các packages khác. Các nhà khoa học dữ liệu thường sử dụng nhiều phiên bản của nhiều package và sử dụng nhiều môi trường để phân tách các phiên bản khác nhau này.
- Chương trình dòng lệnh (command-line program conda) vừa là trình quản lý các package vừa là trình quản lý môi trường (environment manager). Điều này giúp các nhà khoa học dữ liệu đảm bảo rằng mỗi phiên bản của mỗi package có tất cả các phụ thuộc mà nó yêu cầu và hoạt động chính xác.

- Anaconda Navigator cung cấp cho người dùng một giao diện đồ họa để quản lý các environment (môi trường) và package. Ta sẽ có environment mặc định là base (root) chứa các package cơ bản.
- Ở ngăn giao diện Home là nơi quản lý các Application (ứng dụng) tại một environment (trong vòng đỏ).

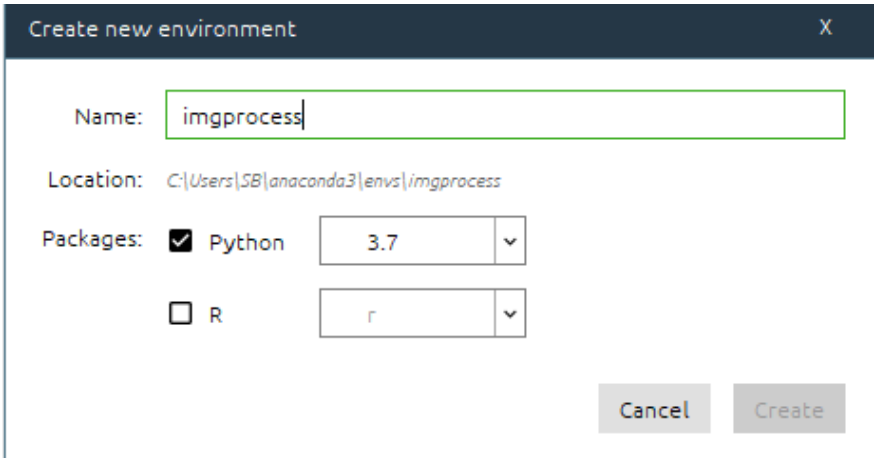



Hình 1.10 Giao diện environment

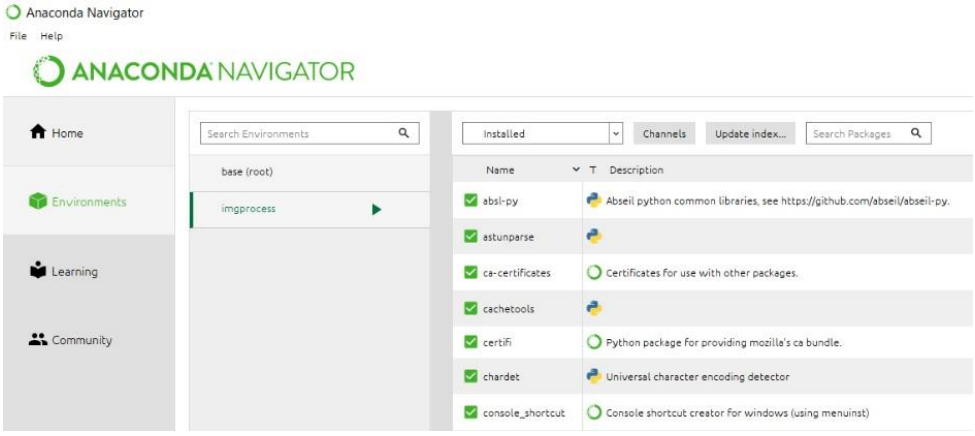



Hình 1.11 Các ngăn của giao diện Environments

- Vùng số 1 là danh sách các environment ta đã tạo.
- Vùng số 2 là nút để tạo environment mới, sau nhấp chuột vào ta sẽ có giao diện như hình 1.12. Chúng ta chọn version của Python và đặt tên cho môi trường.
- Tương tự Clone là để sao chép một bản environment với các package giống một environment đã tạo. Import dùng để tạo environment bằng file có sẵn. Remove để xóa environment.
- Vùng thứ 3 dùng để tìm kiếm và cài đặt các package trong environment bạn đã chọn trong vùng thứ nhất.
- Ví dụ tạo mới một environment imgprocess.




Hình 1.12 Tạo mới một environment imgprocess

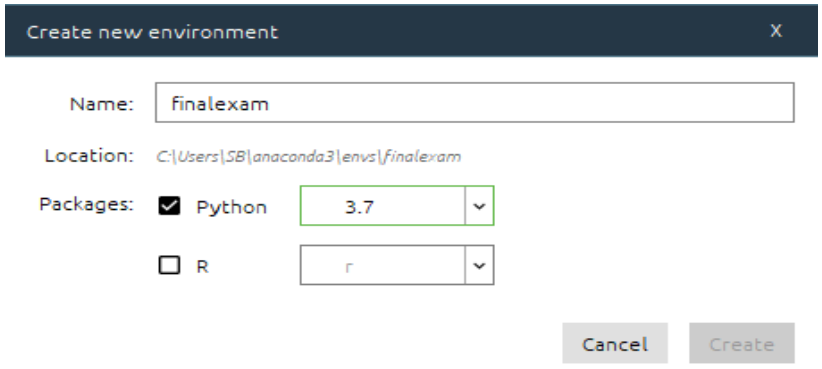



Hình 1.13 Environment imgprocess đã được tạo thành công

### 1.2.3. Hướng dẫn cài thêm thư viện bằng conda

#### 1.2.3.1. Các thư viện sử dụng

- Tạo một environment finalexam để cài các thư viện hỗ trợ cho CHƯƠNG 3.

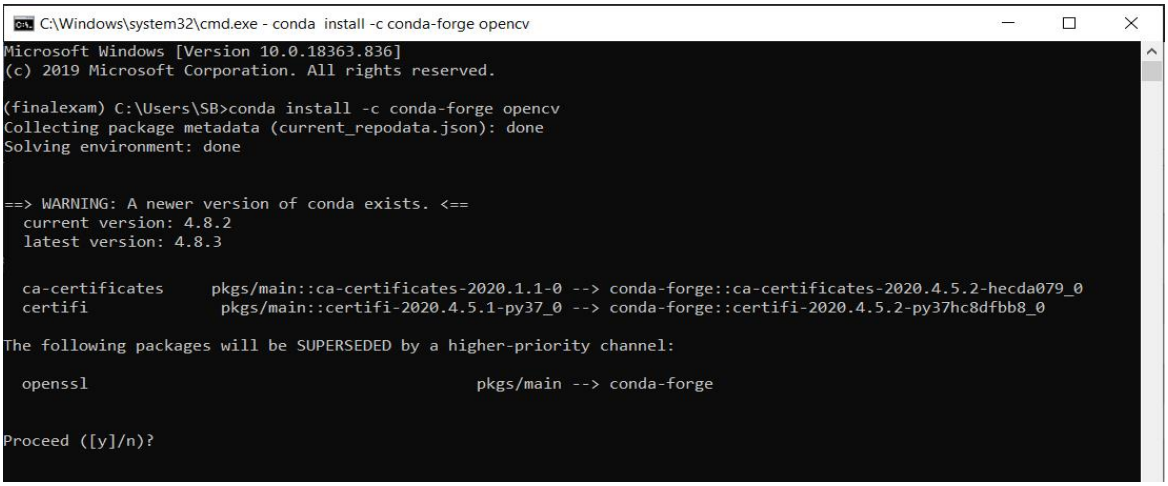



Hình 1.14 Create environment finalexam

- Các thư viện cần cài đặt thêm như:
  + OpenCV
  + Numpy
  + PILLOW

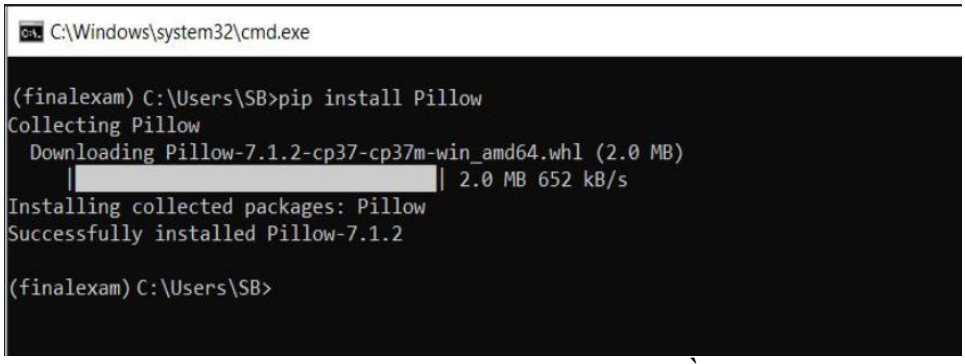
#### 1.2.3.2. Cài đặt bằng dòng lệnh

- Mở CMD.exe Prompt
- Thư viện openCV: dùng lệnh ***conda install -c conda-forge opencv***

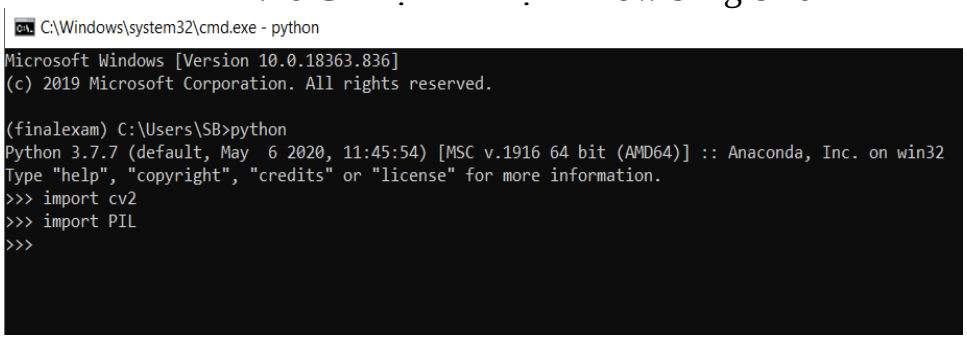



Hình 1.15 Cài thư viện opencv trên env fianlexam bằng cmd

- Thư viện PILLOW: dùng lệnh ***pip install Pillow***



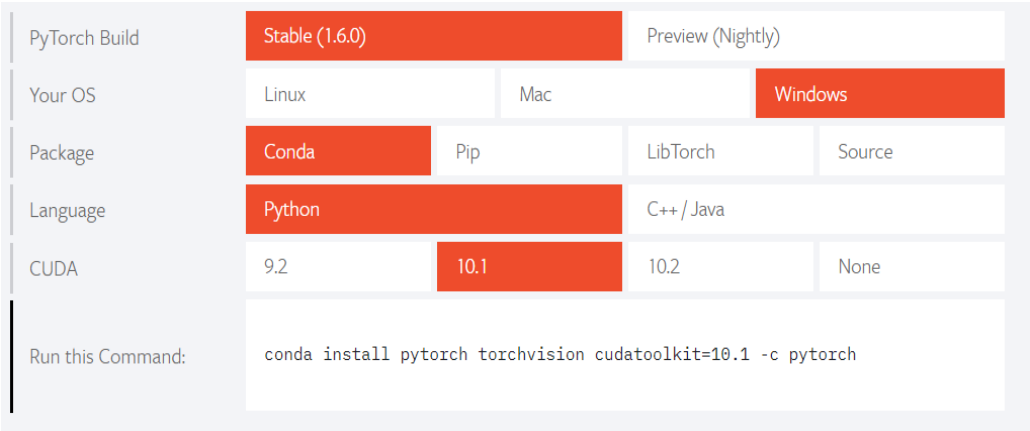

Hình 1.16 Cài đặt thư viện Pillow bằng cmd




Hình 1.17 Kiểm tra lại các thư viện đã cài đặt

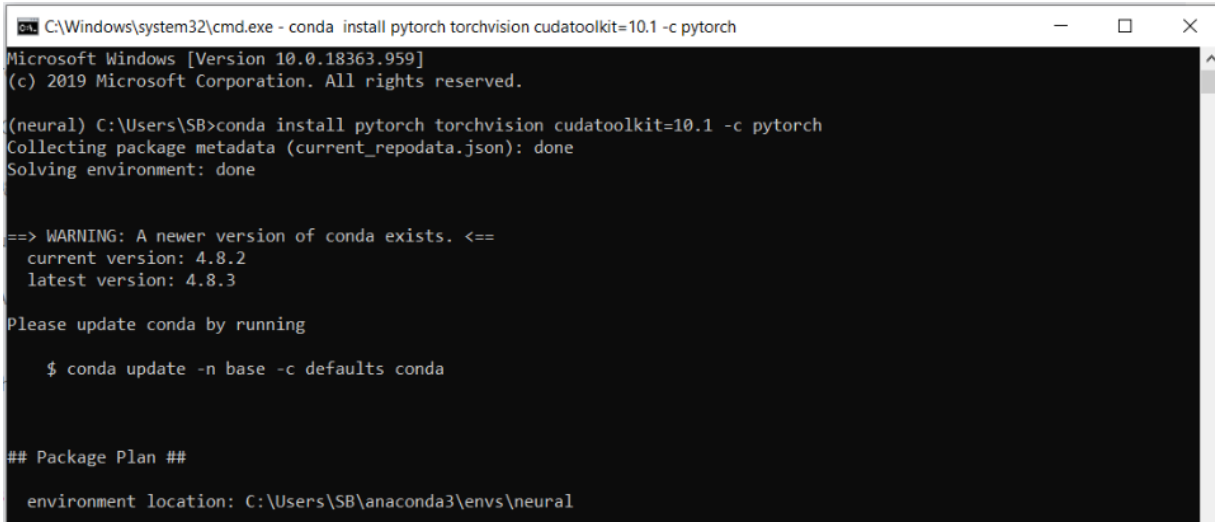
- Thư viện pytorch:

+ Truy cập vào trang chủ https://pytorch.org/
+ Lựa chọn các gói hỗ trợ phù hợp (Sử dụng bản 10.1).




Hình 1.18 Các điều kiện cài đặt Pytorch

+ Mở CMD.exe Prompt với environment neural
+ Thư viện pytorch: dùng lệnh conda install pytorch torchvision cudatoolkit=10.1 -c pytorch

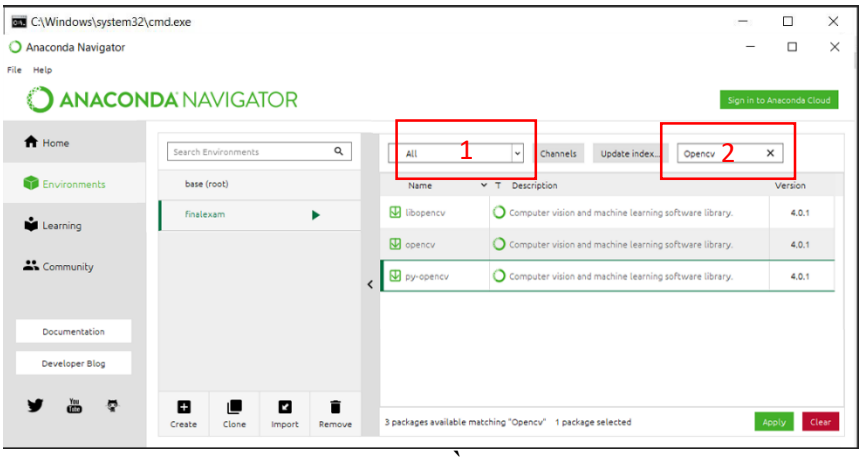



Hình 1.19 Cài thư viện Pytorch trên env neural bằng cmd

### 1.2.3.3. Cài đặt bằng anaconda-navigator

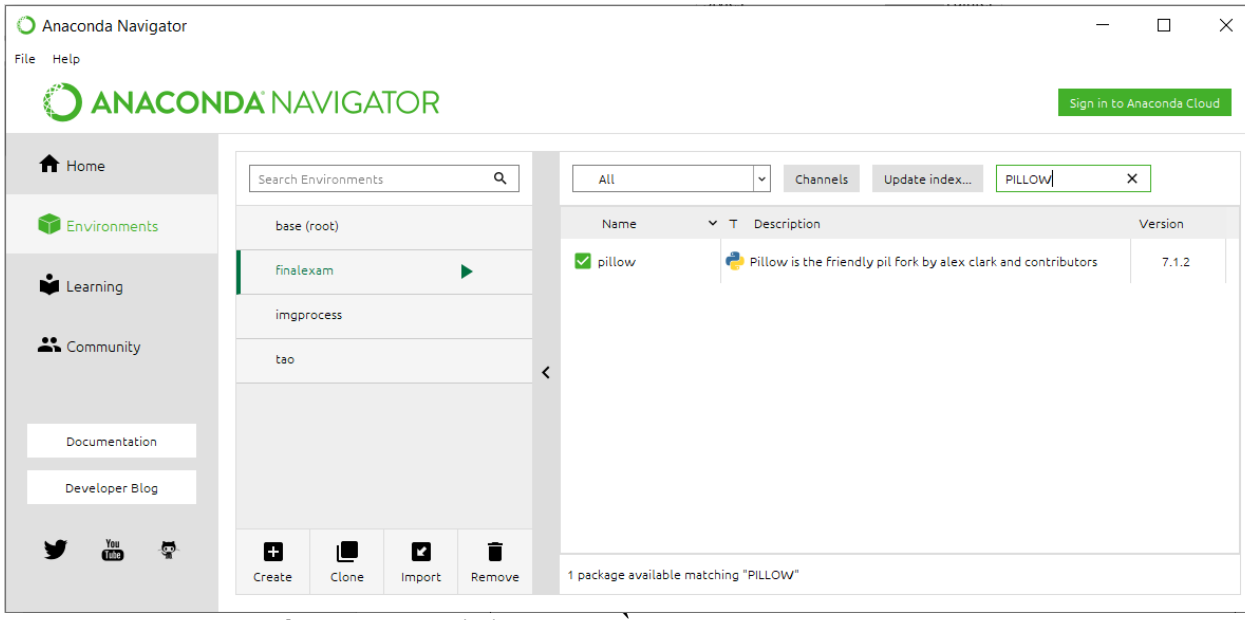
- Để cài thư viện với giao diện anaconda-navigator đầu tiên ta chọn môi trường cần cài thư viện (finalexam).
    + Ở vùng số 1 chọn All, vùng số 2 gõ tên thư viện cần cài đặt vào.
    + Chọn tick các package cần cài đặt và chọn Apply.

Cài thư viện OpenCV bằng giao diện anaconda-navigator




Hình 1.20 Cài đặt opencv bằng ananconda-navigator

Cài thư viện Pillow bằng giao diện anaconda-navigator




Hình 1.21 Cài đặt PIL bằng anaconda-navigator

# CHƯƠNG 2: TÌM HIỂU MẠNG NƠ RON VÀ THƯ VIỆN PYTORCH

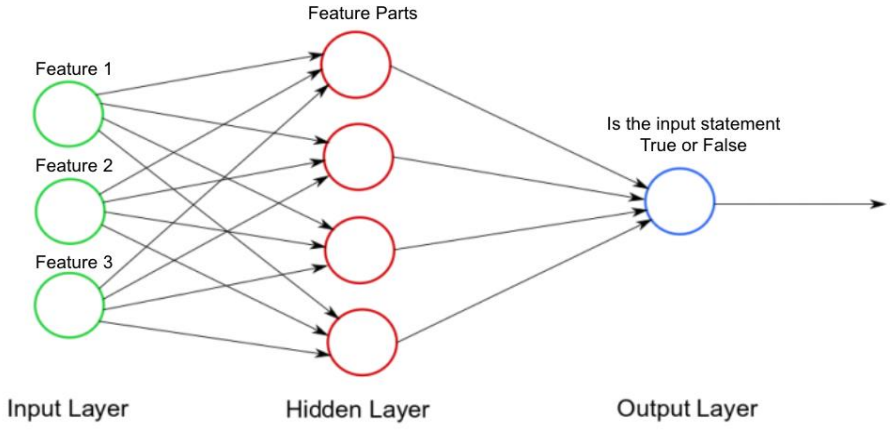
## 2.1. Giới thiệu về mạng nơ-ron

### 2.1.1. Giới thiệu về Machine learning

- Machine Learning là một tập con của AI. Theo định nghĩa của Wikipedia, Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed".
- Nói đơn giản, Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.
- Machine Learning thường được chia thành học có giám sát, trong đó máy tính học bằng ví dụ từ dữ liệu được gắn nhãn và học không giám sát, trong đó các máy tính nhóm các dữ liệu tương tự và xác định chính xác sự bất thường.

### 2.1.2. Lịch sử phát triển của mạng nơ-ron nhân tạo – ANN

- Mạng nơ-ron nhân tạo (Neural Network) là một hệ thống các chương trình và cấu trúc dữ liệu mô phỏng cách vận hành của não người. Mỗi nơ-ron trong mạng nơ-ron là một hàm toán học lấy dữ liệu thông qua đầu vào, biến đổi dữ liệu đó thành dạng dễ điều chỉnh hơn và sau đó phun ra thông qua đầu ra.




Hình 2.1 Mô hình mạng nơ-ron cơ bản

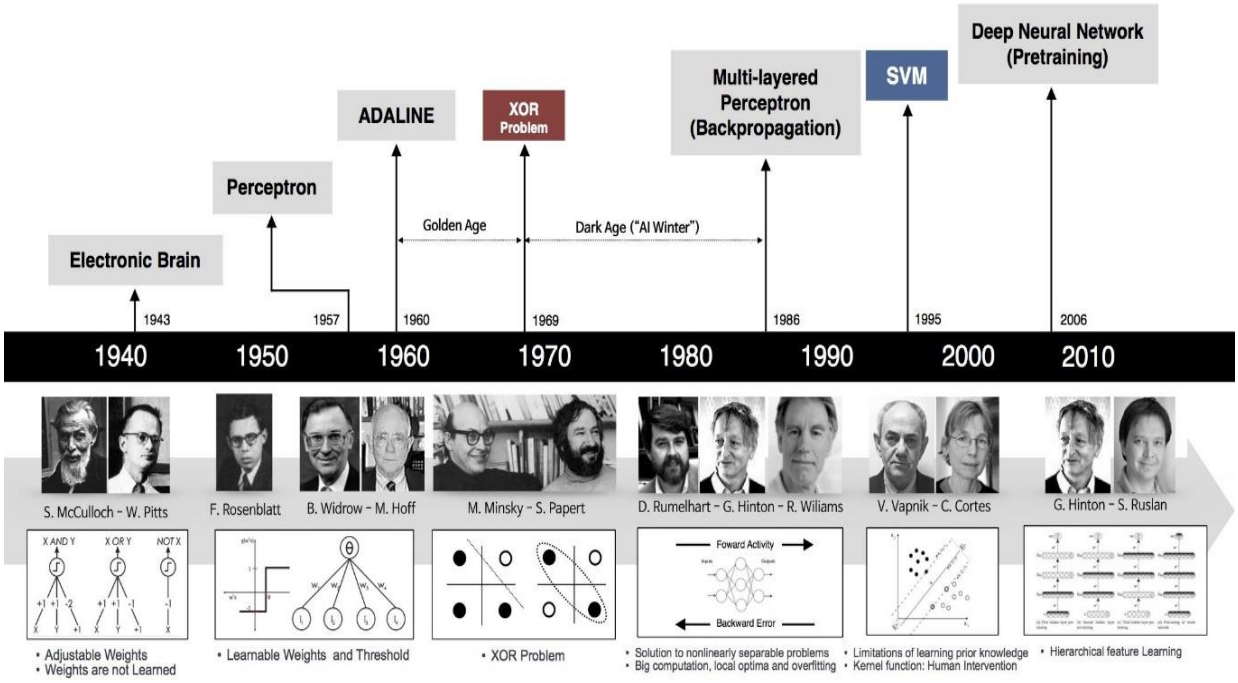
- Khái niệm mạng nơ-ron được bắt đầu vào cuối thập kỷ 1800 khi người ta cố gắng mô tả hoạt động của trí tuệ con người. Ý tưởng này bắt đầu được áp dụng cho các mô hình tính toán từ mạng Perceptron.
- Đầu thập kỷ 1950 Friedrich Hayek là người đầu tiên khẳng định ý tưởng về trật tự tự phát trong não xuất phát từ các mạng phân tán gồm các đơn vị đơn giản (nơ-ron).
- Cognitron (1975) là một mạng nơ-ron đa tầng thời kỳ đầu với một thuật toán huấn luyện. Các chiến lược thần kinh khác nhau sẽ khác nhau về cấu trúc thực sự của mạng và các phương pháp thiết lập trọng số cho các kết nối.
- Giữa những năm 1980, xử lý phân tán song song (parallel distributed processing) trở nên một chủ đề thu hút được nhiều quan tâm dưới cái tên connectionism.
- Ngày nay, không chỉ dừng lại ở mức nghiên cứu lý thuyết, các nghiên cứu ứng dụng mạng nơron để giải quyết các bài toán thực tế được diễn ra ở khắp mọi nơi. Các ứng dụng mạng nơron ra đời ngày càng nhiều và ngày càng hoàn thiện hơn. Điển hình là các ứng dụng: xử lý ngôn ngữ (Language Processing), nhận dạng ký tự (Character Recognition), nhận dạng tiếng nói (Voice Recognition), nhận dạng mẫu (Pattern Recognition), xử lý tín hiệu (Signal Processing), Lọc dữ liệu (Data Filtering), …

### 2.1.3. Lịch sử phát triển của Deeplearning

- Deep Learing là nhóm thuật toán nhỏ của Machine Learning lấy ý tưởng dựa trên Neural Network (mạng neuron) của con người. Deep Learning thường yêu cầu lượng dữ liệu lớn và nguồn tài nguyên sử dụng nhiều hơn các phương pháp thông thường, tuy nhiên cho độ chính xác cao hơn.
- Những dấu mốc quan trọng trong lịch sử phát triển của DeepLearning.
- Perceptron (60s) - Perceptron là một thuật toán supervised learning giúp giải quyết bài toán phân lớp nhị phân, được khởi nguồn bởi Frank Rosenblatt năm

1957 trong một nghiên cứu được tài trợ bởi Văn phòng nghiên cứu hải quân Hoa Kỳ (U.S Office of Naval Research – từ một cơ quan liên quan đến quân sự).

- MLP và Backpropagation ra đời (80s)- Năm 1986, Geoffrey Hinton cùng với hai tác giả khác xuất bản một bài báo khoa học trên Nature với tựa đề "Learning representations by back-propagating errors". Trong bài báo này, nhóm của ông chứng minh rằng neural nets với nhiều hidden layer (được gọi là multi-layer perceptron hoặc MLP) có thể được huấn luyện một cách hiệu quả dựa trên một quy trình đơn giản được gọi là backpropagation.
- Năm 2006 - neural networks với nhiều hidden layer được đổi tên thành deep learning.




Hình 2.2 Các dấu móc quan trọng trong quá trình phát triển Deeplearning

### 2.1.4. Một số thư viện Deeplearning nổ tiếng hiện nay

- TensorFlow là một nguồn thư viện phần mềm mở cho tính toán số học sử dụng dữ liệu từ đồ thị.

+ Các nút biểu đồ đại diện cho các hoạt động toán học, trong khi các cạnh của biểu đồ đại diện cho các mảng dữ liệu đa chiều (tenxơ) giữa các đồ thị.

+ Kiến trúc linh hoạt này cho phép bạn triển khai tính toán cho một hoặc nhiều CPU hoặc GPU trong máy tính để bàn, máy chủ hoặc thiết bị di động mà không cần viết lại các đoạn mã.

- PyTorch là một gói Python cung cấp hai tính năng cấp cao:

  + Tính toán với các dữ liệu số học (như NumPy) với khả năng tăng tốc GPU mạnh mẽ.

  + Mạng lưới thần kinh sâu được xây dựng trên một hệ thống autograd dựa trên băng.

  + Bạn có thể sử dụng lại các gói Python yêu thích của mình như NumPy, SciPy và Cython để mở rộng PyTorch khi cần.

- MX Apache MXNet (incubating) là một khung học tập sâu được thiết kế với tính hiệu quả và tính linh hoạt cao.

  + MXNet cho phép người dùng kết hợp chương trình biểu tượng và bắt buộc để tối đa hóa hiệu quả và năng suất.

  + MXNet chứa một bộ lập lịch phụ thuộc động để tự động song song cả các hoạt động mang tính tượng trưng và các hoạt động bắt buộc.

- TFlearn là một thư viện học tập sâu mô-đun và minh bạch được xây dựng trên đỉnh của Tensorflow. TFlearn được thiết kế để cung cấp API cấp cao hơn cho TensorFlow nhằm tạo điều kiện và tăng tốc các thử nghiệm, trong khi vẫn hoàn toàn minh bạch và tương thích với nó.
- Distributed Keras là một khung học tập sâu phân tán được xây dựng dựa trên Apache Spark và Keras, tập trung vào các thuật toán tối ưu hóa phân tán "tiên tiến". Distributed Keras đã thiết kế khung theo cách mà một trình tối ưu hóa phân

tán mới có thể được thực hiện dễ dàng, do đó cho phép người dùng tập trung vào nghiên cứu.

- Microsoft Cognitive Toolkit (Previously CNTK) là bộ công cụ học tập sâu thống nhất mô tả các mạng thần kinh là một chuỗi các bước tính toán thông qua biểu đồ có hướng.
- Chainer là một khung công tác mã nguồn mở độc lập dựa trên Python cho các mô hình học sâu. Chainer cung cấp một phương tiện linh hoạt, trực quan và hiệu suất cao để thực hiện đầy đủ các mô hình học tập sâu, bao gồm các mô hình tiên tiến như mạng thần kinh tái phát và bộ mã hóa tự động đa dạng.
- DLib là bộ công cụ C ++ hiện đại chứa các thuật toán và công cụ học máy để tạo phần mềm phức tạp trong C ++ để giải quyết các vấn đề trong thế giới thực.
- DeepLearning4J là một phần của Skymind Intelligence Layer, cùng với ND4J, DataVec, Arbiter và RL4J. Nó là một thư viện mạng thần kinh phân tán, mã nguồn mở, được cấp phép của Apache 2.0 được viết bằng Java và Scala.

### 2.1.5. Các khái niệm cơ bản trong mạng nơron

- Batch size:

  + Là tổng số mẫu huấn luyện có trong một min-batch.Giống như số epoch, batch size là một siêu tham số không có quy tắc nào.

  + Điều quan trọng cần lưu ý là kích thước batch bị ảnh hưởng bởi các siêu tham số khác, chẳng hạn như tốc độ học, vì vậy sự *kết hợp* của các siêu tham số này cũng quan trọng như chính kích thước batch.

- Epoch:

  + Một Epoch được tính là khi đã đưa tất cả dữ liệu vào mạng neural network 1 lần.

  + Trong mỗi lần kết nối mạng, thông số được cập nhật và đường cong chuyển từ trạng thái vừa đủ đến tối ưu, đến quá mức. Không có quy tắc nào để

chọn số epoch - đây là một siêu thông số phải được xác định trước khi bắt đầu training.

- Learning rate:

  + Learning rate là một siêu tham số kiểm soát mức độ thay đổi mô hình để đáp ứng với sai số ước tính mỗi khi trọng số của mô hình được cập nhật.

  + Việc chọn learning rate là một thách thức vì giá trị quá nhỏ có thể dẫn đến quá trình luyện tập lâu dài có thể gặp khó khăn, trong khi giá trị quá lớn có thể dẫn đến việc học một bộ trọng lượng dưới tối ưu quá nhanh hoặc quá trình luyện tập không ổn định.

- Momentum:

  + Các đợt chạy ngắn với giá trị xung lượng 0,99, 0,97, 0,95 và 0,9 sẽ nhanh chóng hiển thị giá trị tốt nhất cho động lượng.

  + Sử dụng momentum theo chu kỳ cùng với phạm vi LR kiểm tra ổn định sự hội tụ khi sử dụng các giá trị tốc độ learning rate lớn hơn một hằng số momentum không.

- Iterations: Iterations là số lượng batchs cần để hoàn thành 1 epoch.
- Perceptron: Một perceptron hay hiểu theo tiếng việt là một tế báo thần kinh là đơn giản là một hàm toán học nhận đầu vào từ một hoặc nhiều số, thực hiện các phép toán và trả về kết quả đầu ra.
- Activation functions:

  + Hàm kích hoạt hay activation functions được sinh ra với mục đích bẻ gãy sự tuyến tính của mạng nơ ron.

  + Các hàm này có thể hiểu đơn giản như một bộ lọc để quyết định xem thông tin có được đi qua nơ ron hay không.

- Sigmoid function:

+ Sigmoid có thể được coi là một hàm được sử dụng để làm trơn dữ liệu và nó là một hàm khả vi.

+ Sigmoid rất hữu ích để chuyển đổi bất kỳ giá trị nào thành xác suất và có thể được sử dụng để phân lớp nhị phân - binary classification.

- Artificial neural network (ANN):

+ Artificial neural network (ANN) có thể coi là một tập hơn của các perceptron và các hàm kích hoạt. Các perceptron đơn lẻ sẽ được kết hợp với nhau thành các lớp ẩn hidden layers hay units.

+ Các lớp ẩn sử dụng các hàm kích hoạt phi tuyến ánh xạ các lớp đầu vào thành các lớp đầu ra trong một không gian có số chiều thấp hơn và được gọi chung là mạng nơ ron nhân tạo.

- Gradient descent:

+ Đây là một giải thuật sử dụng trong tối ưu trong không gian nhiều chiều - multidimensional optimization.

+ Mục đích của nó là có thể đạt đến tối ưu toàn cục - global maximum

## 2.2. Giới thiệu về thư viện Pytorch

### 2.2.1. Lịch sử phát triển và các phiên bản của Pytorch

- PyTorch là 1 thư viện Python-based hỗ trợ tạo ra các DeepLearning models và sử dụng chúng cho các ứng dụng khác nhau. PyTorch không chỉ là 1 thư viện DeepLearning, mà chính là 1 package về tính toán khoa học.
- Phần core của PyTorch (libtorch) được viết bằng C++ (high-performance C++ API). Điều này còn làm cho quá trình chuyển đổi từ R&D sang Production dễ dàng hơn.
- PyTorch, tương tự như Python, nó được thiết kế tập trung vào tính dễ sử dụng và thậm chí người dùng có kiến thức lập trình rất cơ bản cũng có thể sử dụng nó trong các dự án có liên quan đến Deep Learning.

- Một số phiên bản gần đây của pytorch

| Bảng 1 – Các phiên bản gần đây của pytorch | | |
|---|---|---|
| Thứ tự | Phiên bản | Tương thích |
| 1 | v1.6.0 | CUDA 9.2, 10.1, 10.2, CPU only |
| 2 | v1.5.0 | CUDA 9.2, 10.1, 10.2, CPU only |
| 3 | v1.4.0 | CUDA 9.2, 10.1, CPU only |
| 4 | v1.2.0 | CUDA 9.2, 10.0, CPU only |
| 5 | v1.1.0 | CUDA 9.0, 10.0, CPU only |

### 2.2.2. Một số ưu điểm của pytorch

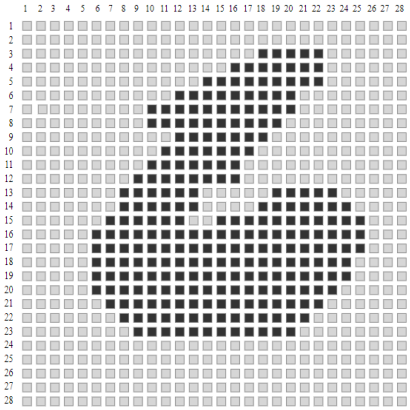
- PyTorch sử dụng đồ thị tính toán động.
- PyTorch sử dụng thực thi không đồng bộ của Python để thực hiện song song dữ liệu
- Bản chất Pythonic: PyTorch là Pythonic, có nghĩa là các nhà phát triển Python sẽ cảm thấy thoải mái hơn khi viết mã với PyTorch so với các khung học sâu khác.
- Dễ học: Giống như ngôn ngữ Python, PyTorch được coi là tương đối dễ học hơn so với các khuôn khổ học sâu khác. Lý do chính là do cú pháp dễ dàng và trực quan của nó.
- Cộng đồng mạnh: Mặc dù PyTorch là một khung công tác tương đối mới hơn, nó đã phát triển một cộng đồng các nhà phát triển chuyên dụng rất nhanh chóng.
- Dễ dàng gỡ lỗi: PyTorch được tích hợp sâu với Python nên có thể dễ dàng sử dụng nhiều công cụ gỡ lỗi Python với nó.
- PyTorch có thể sử dụng TensorBoard để ghi lại các mô hình và chỉ số PyTorch trong giao diện người dùng TensorBoard. Vô hướng, hình ảnh, biểu đồ, đồ thị và hình ảnh nhúng đều được hỗ trợ cho các mô hình PyTorch và tensors.

- Autograd: Tính đạo hàm tự động của pytorch các module autograd cung cấp tự động cho tất cả các hoạt động trên tensors. Nó là một khung làm việc được xác định theo từng lần chạy.

### 2.2.3. Ví dụ mẫu Pytorch Examples

Giới thiệu về bộ ví dụ mẫu PyTorch Examples và ví dụ phân lớp ký tự số viết tay MNIST.

- MNIST chứa 70.000 hình ảnh các chữ số viết tay: 60.000 cho đào tạo và 10.000 cho thử nghiệm.
- Mỗi ảnh MNIST có kích thước 28 * 28 = 784 pixel. Mỗi pixel được cung cấp dưới dạng một số từ 0-255 cho biết mật độ của nó. Để giữ cho mọi thứ đơn giản, chúng tôi sẽ bỏ qua mật độ (thang màu xám) và coi mỗi pixel là BẬT hoặc TẮT (đen trắng).



Hình 2.3 Mô hình một ký tự trong bộ dữ liệu MNIST

Hình 2.4 Hình ảnh dữ liệu dùng để kiểm tra

- Huấn luyện mạng.

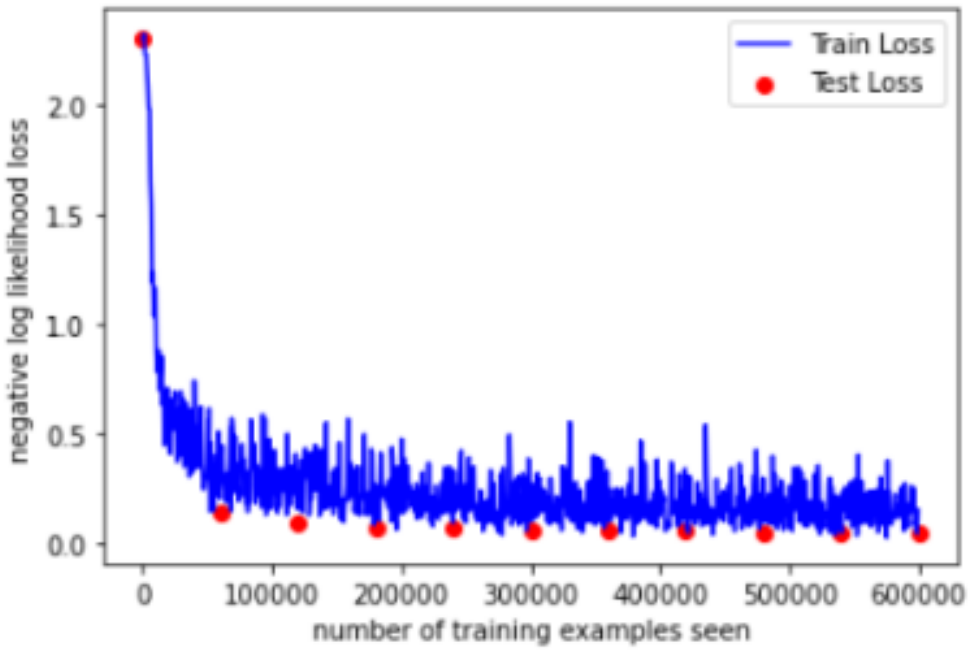
```
Train Epoch: 10 [51840/60000 (86%)]     Loss: 0.085315
Train Epoch: 10 [52480/60000 (87%)]     Loss: 0.172823
Train Epoch: 10 [53120/60000 (88%)]     Loss: 0.085768
Train Epoch: 10 [53760/60000 (90%)]     Loss: 0.212456
Train Epoch: 10 [54400/60000 (91%)]     Loss: 0.036676
Train Epoch: 10 [55040/60000 (92%)]     Loss: 0.139395
Train Epoch: 10 [55680/60000 (93%)]     Loss: 0.063721
Train Epoch: 10 [56320/60000 (94%)]     Loss: 0.069036
Train Epoch: 10 [56960/60000 (95%)]     Loss: 0.162455
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.047492
Train Epoch: 10 [58240/60000 (97%)]     Loss: 0.130600
Train Epoch: 10 [58880/60000 (98%)]     Loss: 0.159305
Train Epoch: 10 [59520/60000 (99%)]     Loss: 0.206158

Test set: Avg. loss: 0.0409, Accuracy: 9868/10000 (99%)
```

Hình 2.5 Kết quả huấn luyện dữ liệu

- Đánh giá hiệu suất của mô hình mạng.

  + Kết quả huấn luyện cho biết độ sai lệch trung bình là 0.0409% và độ chính xác là 99% ở lần huấn luyện thứ 10.

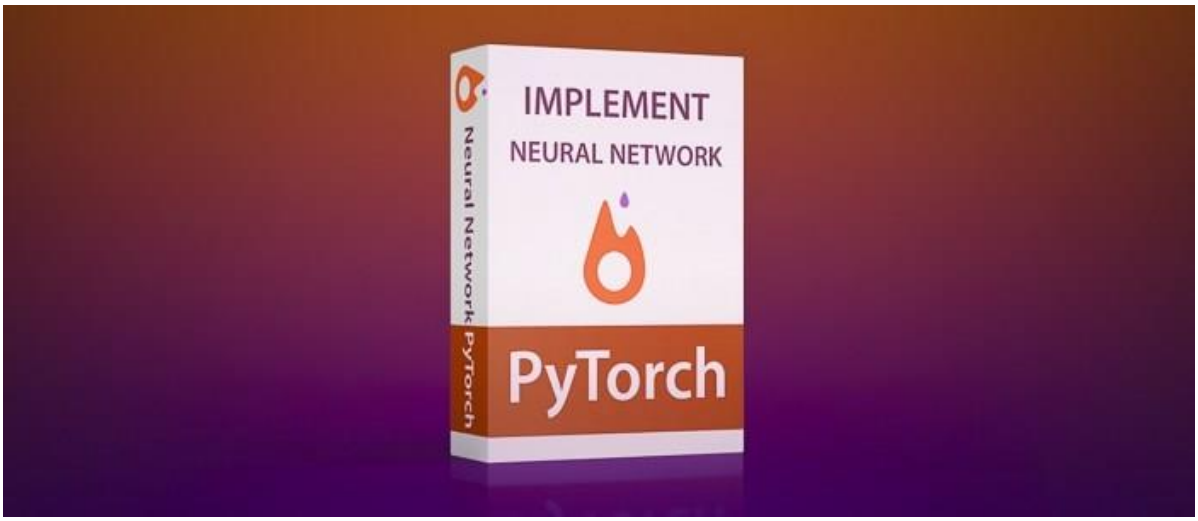



Hình 2.6 Biểu đồ đường cong đào tạo

- Kiểm tra:


Hình 2.7 Kết quả dự đoán

### 2.2.4. Một số hỗ trợ cho người học của Pytorch




Hình 2.8 Implement Neural Network using PyTorch

- Các tài liệu hướng dẫn ở trang chủ của Pytorch: https://pytorch.org/
- Có các ví dụ hướng dẫn chi tiết.
- Cộng đồng mạnh: Mặc dù PyTorch là một khung công tác tương đối mới hơn, nó đã phát triển một cộng đồng các nhà phát triển chuyên dụng rất nhanh chóng. Mang lại khả năng debug dễ dàng hơn theo hướng interactively, rất nhiều nhà

nghiên cứu và engineer đã dùng cả pytorch và tensorflow đều đánh giá cáo pytorch hơn trong vấn đề debug và visualize.

+ Hỗ trợ tốt dynamic graphs.
+ Được phát triển bởi đội ngũ Facebook.
+ Kết hợp cả các API cấp cao và cấp thấp.

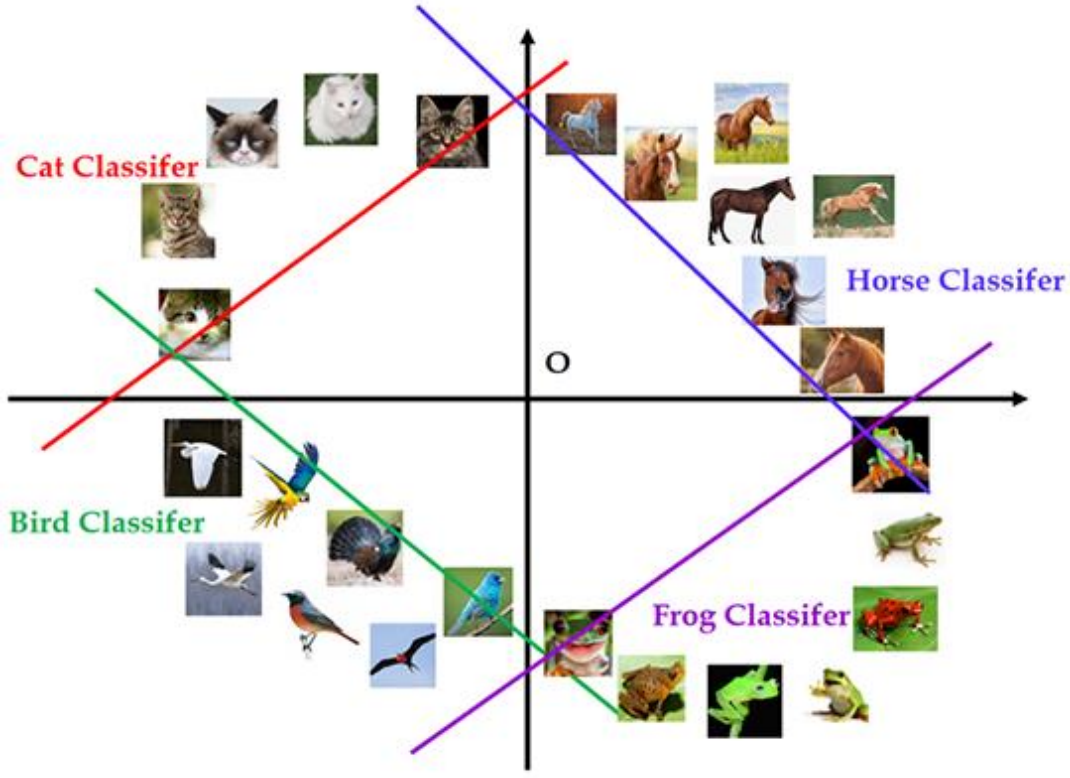
- Trong số những framework hỗ trợ deeplearning thì pytorch là một trong những framework được ưa chuộng nhiều nhất (cùng với tensorflow và keras), có lượng người dùng đông đảo, cộng đồng lớn mạnh.
- Vào năm 2019 framework này đã vươn lên vị trí thứ 2 về số lượng người dùng trong những framework hỗ trợ deeplearning (chỉ sau tensorflow). Đây là package sử dụng các thư viện của CUDA và C/C++ hỗ trợ các tính toán trên GPU nhằm gia tăng tốc độ xử lý của mô hình. 2 mục tiêu chủ đạo của package này hướng tới là:

+ Thay thế kiến trúc của numpy để tính toán được trên GPU.
+ Deep learning platform cung cấp các xử lý tốc độ và linh hoạt.

# CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

## 3.1. Nêu bài toán

- Bài toán phân lớp là quá trình phân lớp một đối tượng dữ liệu vào một hay nhiều lớp đã cho trước nhờ một mô hình phân lớp (model). Mô hình này được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có gán nhãn (hay còn gọi là tập huấn luyện). Quá trình phân lớp là quá trình gán nhãn cho đối tượng dữ liệu.
- Bằng những kiến thức đã biết thì yêu cầu đặt ra là với từng dữ liệu chúng ta phải xem xét và phân lớp chúng vào những lớp khác như thế nào và phải đảm bảo độ chính xác.
- Như vậy, nhiệm vụ của bài toán phân lớp là cần tìm một mô hình phân lớp để có thể xác định được dữ liệu được truyền vào là thuộc phân lớp nào trong 4 lớp "Bird", "Cat", "Frog", "Horse".

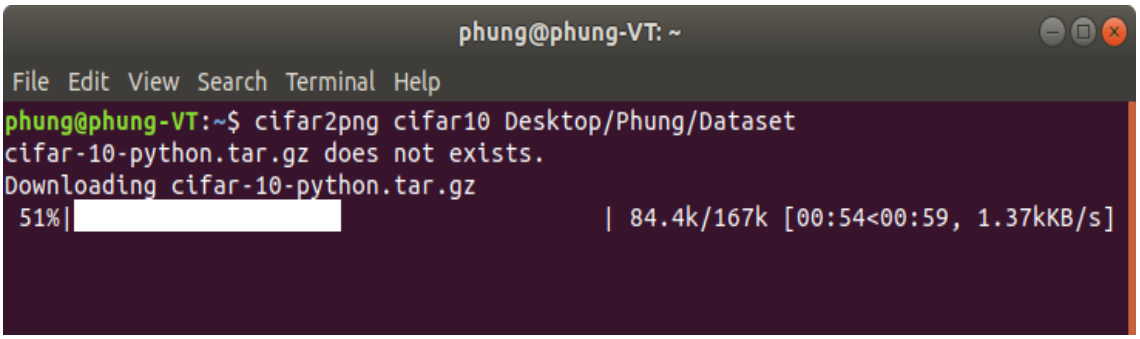



Hình 3.1 Sơ đồ mô phỏng 4 lớp "Bird", "Cat", "Frog", "Horse"

## 3.2. Chuẩn bị dữ liệu

- Bộ dữ liệu CIFAR-10 bao gồm 60000 hình ảnh màu 32x32 trong 10 lớp, với 6000 hình ảnh cho mỗi lớp. Có 50000 hình ảnh dùng để đào tạo và 10000 hình ảnh dùng cho việc kiểm tra.
- Tập dữ liệu thô được chia thành 5 lô đào tạo và một lô thử nghiệm, mỗi lô có 10000 hình ảnh. Lô thử nghiệm chứa chính xác 1000 hình ảnh được chọn ngẫu nhiên từ mỗi lớp. Các lô đào tạo chứa các hình ảnh còn lại theo thứ tự ngẫu nhiên, nhưng một số lô đào tạo có thể chứa nhiều hình ảnh từ lớp này hơn lớp khác. Giữa chúng, các lô đào tạo chứa chính xác 5000 hình ảnh từ mỗi lớp.
- Để chuẩn bị dữ liệu cho việc xây dựng ứng dùng thì, từ bộ dữ liệu CIFAR10 thô cần phải được chuyển đổi về bộ dữ liệu hình ảnh và được phân theo từng lớp với 2 nhóm dùng để huấn luyện và kiểm tra.
- Ở đây 4 bộ dữ liệu cần dung trong 10 bộ dữ liệu của CIFAR10 là: “Bird”, “Cat”, “Frog”, “Horse”.
- Các bước thực hiện:

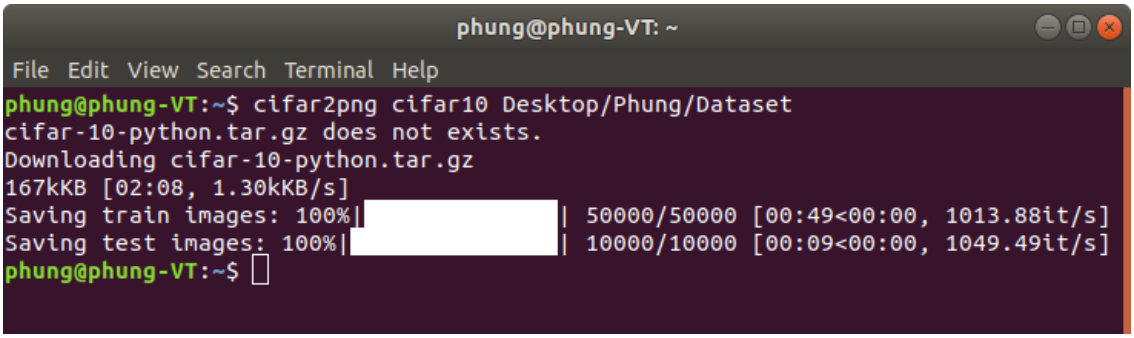
+ Bước 1 download bộ dữ liệu về và giải nén:




Hình 3.2 Quá trình tải và giải nén bộ dữ liệu CIFAR10

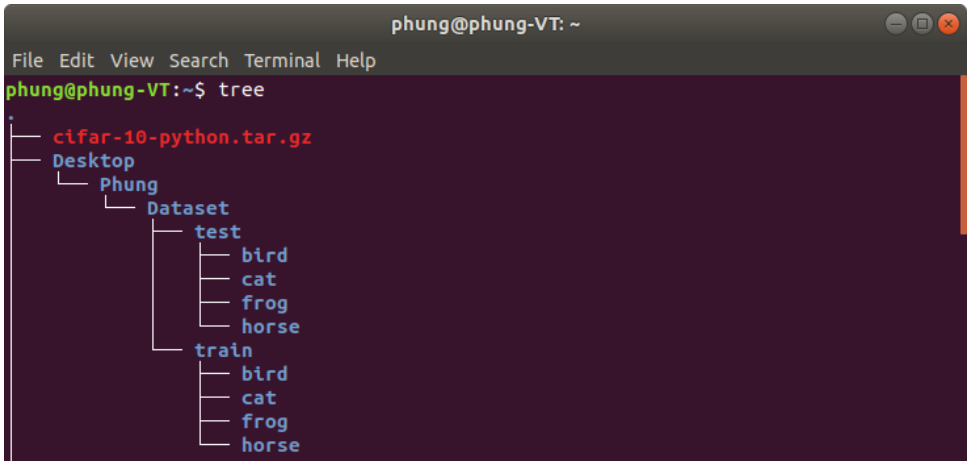
- Bộ dữ liệu sẽ được download và giải nén tự động nhờ vào các thư viện hỗ trợ như Pytorch, requests.
- Dựa vào các Label list mà bộ dữ liệu sẽ được giải nén thành 5 lô dùng để đào tạo huấn luyện “data_batch_1” đến “data_batch_5” và một lô dùng cho việc kiểm tra “test_batch”.

+ Phân loại và chuyển đổi:

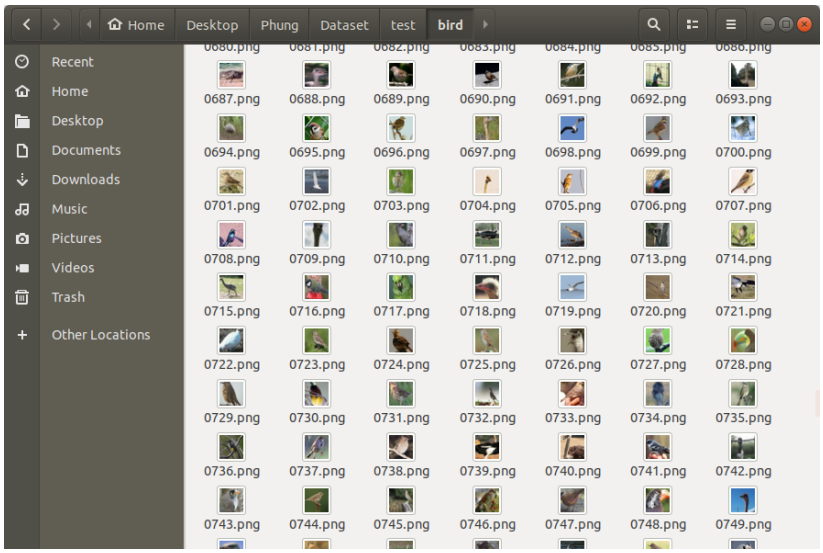



Hình 3.3 Quá trình chuyển đổi thành hình ảnh

+ Lựa chọn bộ dữ liệu cần dùng gồm 4 lớp “Bird”, “Cat”, “Frog”, “Horse”



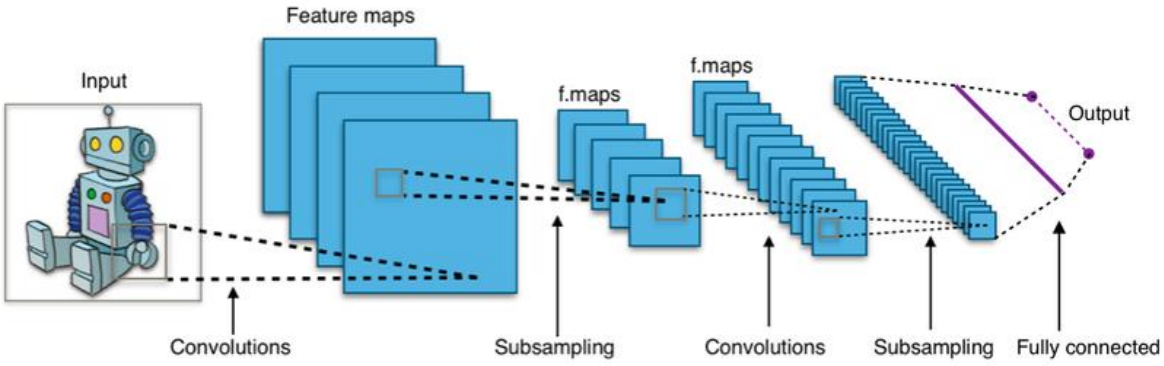

Hình 3.4 Cây thư mục của bộ dữ liệu




Hình 3.5 Hình ảnh dùng cho việc kiểm tra của bộ dữ liệu về “Bird”

## 3.3. Phương pháp lựa chọn đề tài

- CNN (Convolutional Neural Network) là một Structure rất phổ biến và quen thuộc trong Deep Learning. CNN được ứng dụng nhiều trong Computer Vision, Recommender System, Natural Language Processing, ...
- Với Convolutional Neural Network, đây là một deep neural network architectures. Hiểu đơn giản, nó cũng chính là một dạng Artificial Neural Network, một Multiplayer Perceptron nhưng mang thêm 1 vài cải tiến, đó là Convolution và Pooling.



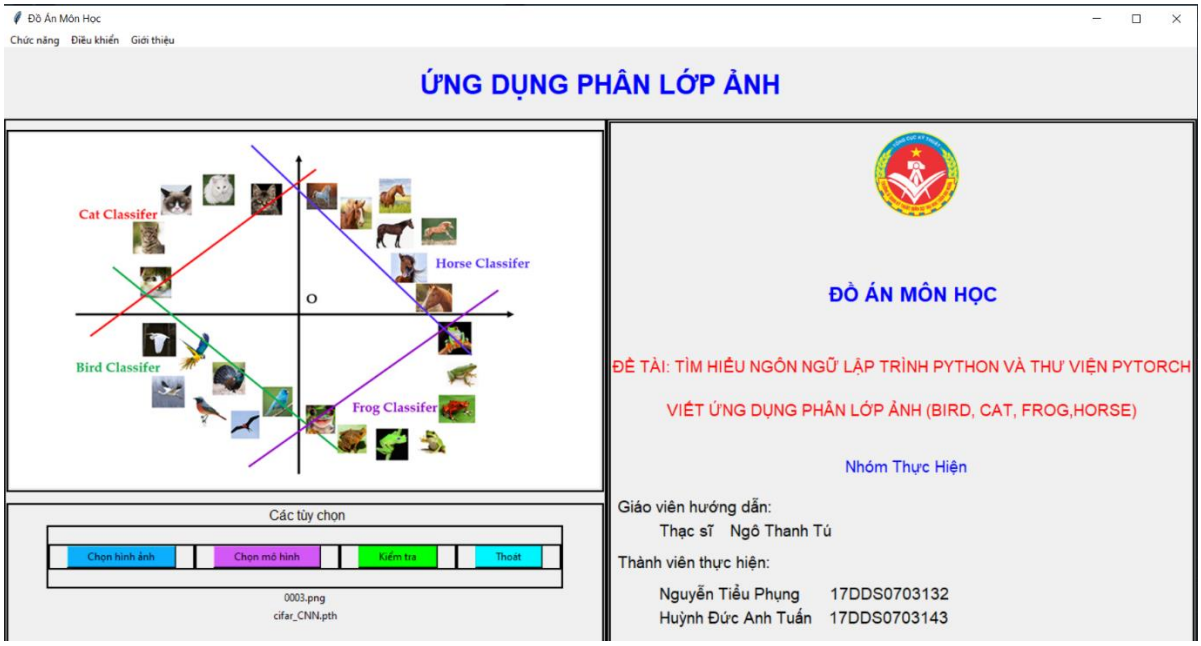

Hình 3.6 Mô hình Convolutional Neural Network

- Trong mạng neural network, mạng CNN là 1 phương thức rất hay được sử dụng để nhận dạng hình ảnh, phân loại ảnh, nhận diện đối tượng, nhận diện khuôn mặt, ...
- CNN thực hiện phân loại ảnh bằng các bước nhận ảnh đầu vào, xử lý và phân loại nó dưới dạng các nhãn. Máy tính nhìn nhận dữ liệu đầu vào như 1 mảng các pixel dựa trên độ phân giải của ảnh.
- Dựa vào nó máy tính nhìn nhận ảnh dưới dạng h x w x d (h: height, w: width, d: dimension). 1 ảnh 6x6x3 nghĩa là có 3 kênh màu (RGB) còn ảnh 4x4x1 là grayscale image.
- Với những đặc điểm trên thì CNN rất phù hợp cho việc nghiên cứu và xây dựng một mô hình ứng dụng phân lớp hình ảnh với các dữ liệu ở bộ dữ liệu CIFAR10.

## 3.4. Giao diện và các chức năng của ứng dụng

- Bước 1: Load và chuẩn hóa các tập dữ liệu dùng để huấn luyện và kiểm tra.

+ Khi giao diện chính được khởi động thì sẽ tự động kích hoạt việc huấn luyện các tập dữ liệu. Khi các phím chức năng chuyển sang màu xanh thì việc huấn luyện hoàn tất.

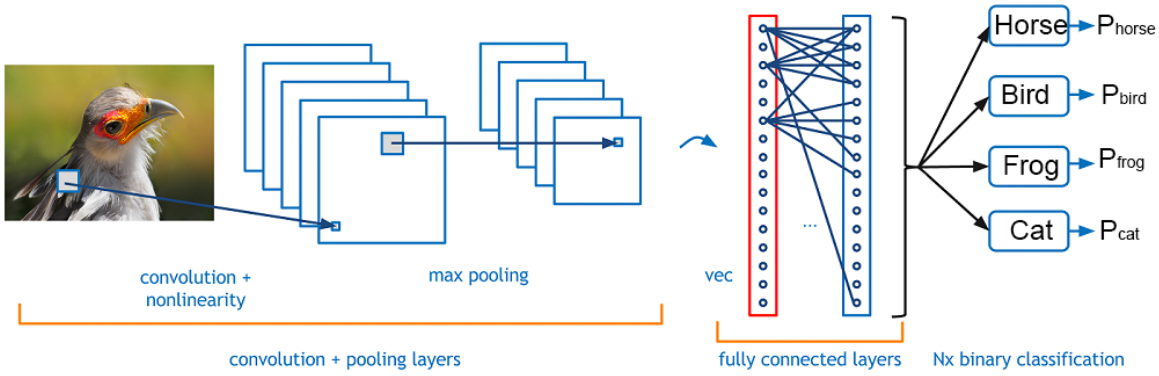



Hình 3.7 Giao diện màn hình chính của ứng dụng

- Bước 2: Xác định mô hình mạng nơ-ron.

+ CNN bao gồm tập hợp các lớp cơ bản bao gồm: convolution layer + nonlinear layer, pooling layer, fully connected layer. Các lớp này liên kết với nhau theo một thứ tự nhất định.

+ Thông thường, một ảnh sẽ được lan truyền qua tầng convolution layer + nonlinear layer đầu tiên, sau đó các giá trị tính toán được sẽ lan truyền qua pooling layer, bộ ba convolution layer + nonlinear layer + pooling layer có thể được lặp lại nhiều lần trong network. Và sau đó được lan truyền qua tầng fully connected layer và softmax để tính sác xuất ảnh đó chứa vật thể gì.




Hình 3.8 Mô hình mạng nơ-ron CNN

- Bước 3: Xác định hàm mất mát.

+ Vì loss function đo đạc chênh lệch giữa y và y^, nên không lạ gì nếu ta nghĩ ngay đến việc lấy hiệu giữa chúng:

$$L(\hat{y}, y) = \hat{y} - y$$

+ Tuy nhiên hàm này lại không thỏa mãn tính chất không âm của một loss function. Ta có thể sửa nó lại một chút để thỏa mãn tính chất này. Ví dụ như lấy giá trị tuyệt đối của hiệu:

$$L(\hat{y}, y) = |\hat{y} - y|$$

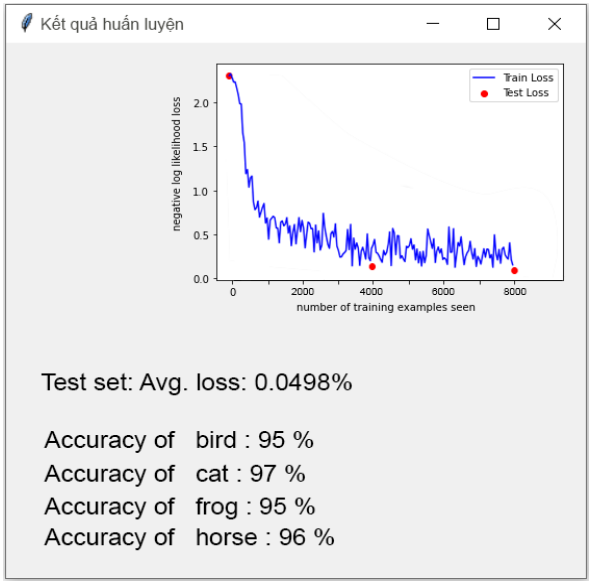
Loss function này không âm nhưng lại không thuận tiện trong việc cực tiểu hóa, bởi vì đạo hàm của nó không liên tục (nhớ là đạo hàm của f(x) = |x| bị đứt quãng tại x = 0) và thường các phương pháp cực tiểu hóa hàm số thông dụng đòi hỏi phải tính được đạo hàm. Một cách khác đó là lấy bình phương của hiệu:

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

+ Khi tính đạo hàm theo y^, ta được $\nabla L = 0.5 \times 2 \times (y\hat{} - y) = y\hat{} - y$. Các bạn có thể thấy rằng hằng số 1/2 được thêm vào chỉ để cho công thức đạo hàm được đẹp hơn, không có hằng số phụ. Loss function này được gọi là **square loss**. Square loss có thể được sử dụng cho cả regression và classification, nhưng thực tế thì nó thường được dùng cho regression hơn

- Bước 4: Huấn luyện mạng trên tập dữ liệu huấn luyện.

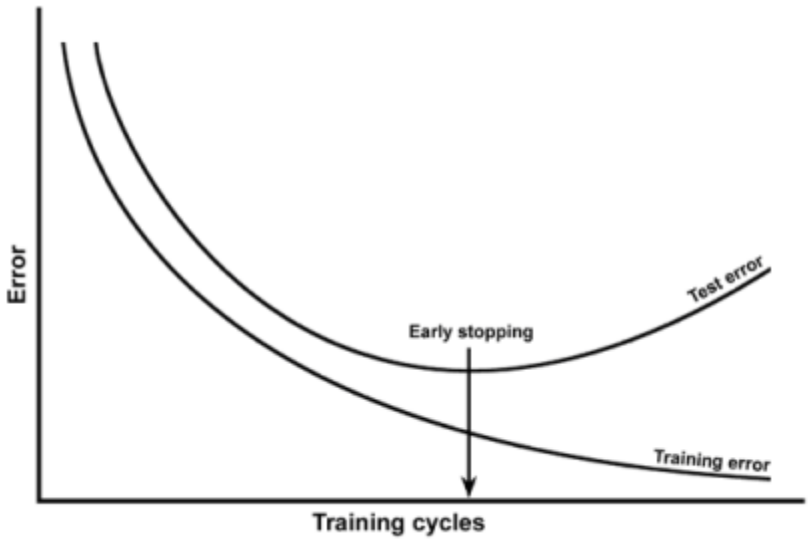
+ Việc huấn luyện dữ liệu sẽ dựa trên 4 bộ dữ liệu train của các lớp “Bird”, “Cat”, “Frog”, “Horse”.




Hình 3.9 Kết quả thông số huấn luyện

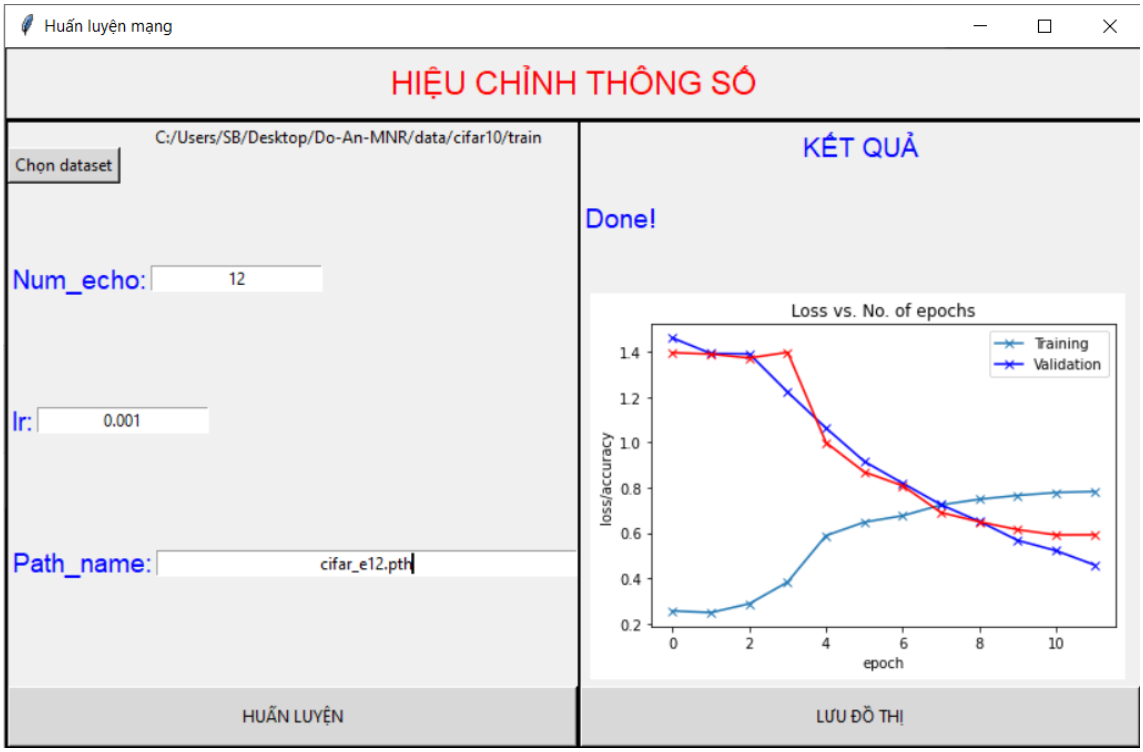
+ Kết quả huấn luyện cho thấy độ chính xác của 4 bộ dữ liệu điều trên 90%.

+ Kết quả của hàm mất mát rất thấp chỉ khoảng 0.0498%.




Hình 3.10 Mô hình đường cong training and validation losses

+ Nhìn chung thì cả tổn thất về đào tạo và xác nhận dường như giảm theo thời gian. Nếu việc đào tạo mô hình đủ lâu thì sẽ nhận thấy rằng tổn thất đào tạo tiếp tục giảm, trong khi tổn thất xác thực ngừng giảm và thậm chí bắt đầu tăng sau một thời điểm nhất định.
+ Hiện tượng này được gọi là overfitting , và nó là không. 1 tại sao nhiều mô hình học máy cho kết quả khá khủng khiếp trên dữ liệu trong thế giới thực. Điều này xảy ra bởi vì mô hình, trong một nỗ lực để giảm thiểu tổn thất, bắt đầu học các patters là duy nhất đối với dữ liệu đào tạo, đôi khi thậm chí ghi nhớ các ví dụ đào tạo cụ thể. Bởi vì điều này, mô hình không tổng quát hóa tốt cho dữ liệu chưa từng thấy trước đó.

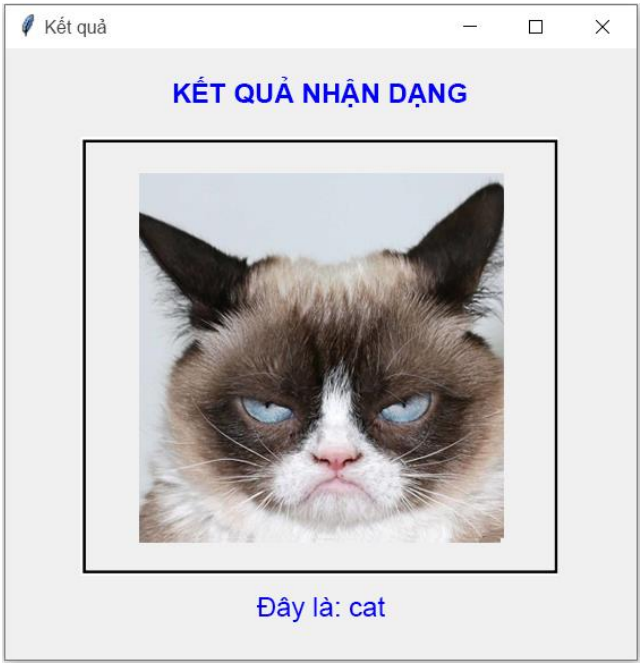



Hình 3.11 Giao diện diện huấn luyện mô hình mạng

+ Num_echo: Số lần đưa tất cả dữ liệu vào mạng để thực hiện việc huấn luyện.
+ Lr: tỷ lệ học của mô hình.

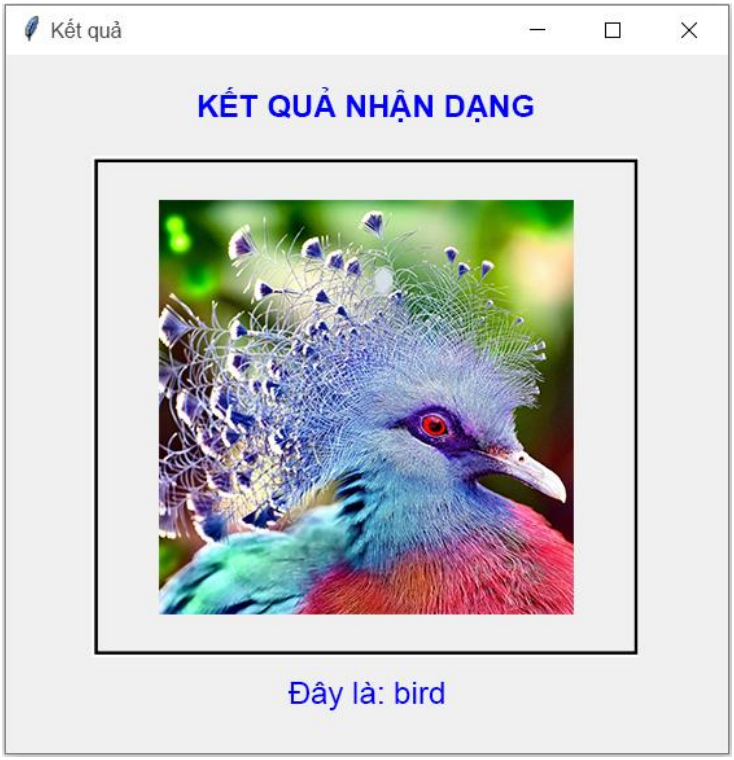
- Bước 5: Kiểm tra mạng trên tập dữ liệu kiểm tra.

Sau khi đã tìm được mô hình phân lớp và hoàn tất việc huấn luyện, thì ở bước này chúng ta sẽ đưa vào các dữ liệu mới để kiểm tra độ chính xác trên mô hình phân lớp.

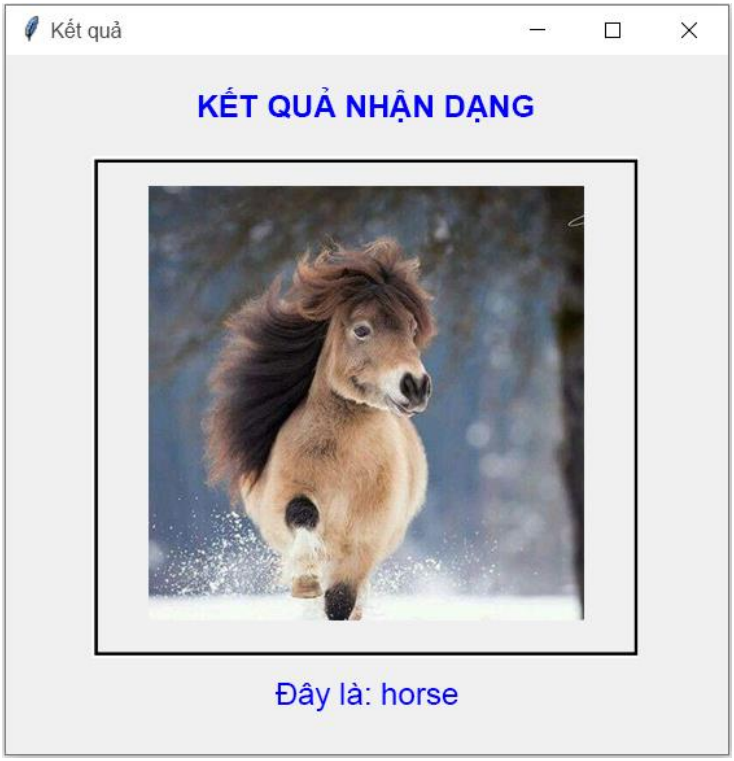



Hình 3.12 Kết quả phân lớp "cat"


Hình 3.13 Kết quả phân lớp "frog"




Hình 3.14 Kết quả phân lớp “bird”




Hình 3.15 Kết quả phân lớp “horse”

## 3.5. Đánh giá

- Ưu điểm

+ Ứng dụng phân loại hình ảnh dựa trên phương pháp CNNs đã đạt được một số điểm như sau:
+ Nắm được vấn đề cơ bản về deep learning và mạng nơ-ron.
+ Sử dụng tốt các thư viện hỗ trợ như Pytorch, tkinter, pillow để xử lý hình ảnh và dữ liệu, cũng như sử dụng ngôn ngữ lập trình python.
+ Nắm được các đặt điểm của bài toán phân lớp.

- Khuyết điểm

+ Tốc độ huấn luyện tập dữ liệu còn hạn chế.
+ Cấu trúc ứng dụng còn đơn giản.

## 3.6. Hướng phát triển của bài toán

- Nâng cao hiệu quả của chương trình, mở rộng số lượng các lớp nhận dạng nhiều hơn với tập dữ liệu CIFAR100.
- Phát triển chương trình thành module phần cứng. Có khả năng tương thích với các thiết bị quan sát như camera…
- Nghiên cứu theo hướng một ứng dụng cụ thể như: nhận diện phương tiện giao thông, nhận dạng các loại đồ vật, hàng hóa.

# TÀI LIỆU THAM KHẢO


[ 1 ] Trang chủ của Pytorch: https://pytorch.org/

[ 2 ] Tài liệu hỗ trợ của Pytorch: https://pytorch.org/tutorials/

[ 3 ] Blog Machine Learning Cơ Bản của T.s Vũ Hữu Tiệp: https://machinelearningcoban.com/

[ 4 ] MNIST Handwritten Digit Recognition in PyTorch: https://nextjournal.com/

[ 5 ] Y. LeCun and Y. Bengio.“Convolutional networks for images, speech, and time series.” In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.

[ 6 ] Kenji Doi “Convert CIFAR-10 and CIFAR-100 datasets into PNG images”

[ 7 ] “Convert CIFAR-10 and CIFAR-100 datasets into PNG images” by Dan Clark, KDnuggets.