BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

****

**BÀI TẬP LỚN MÔN HỌC SÂU**

**PHÂN LOẠI ẢNH TỪ BỘ DỮ LIỆU CIFAR-10**

**BẰNG PYTORCH**

NGÀNH: KHOA HỌC MÁY TÍNH

CHUYÊN NGÀNH: TRÍ TUỆ NHÂN TẠO VÀ KHOA HỌC DỮ LIỆU

SINH VIÊN: **ĐỖ MINH HOÀNG**

MÃ LỚP: 124211

HƯỚNG DẪN: **PGS. TS. NGUYỄN MINH TIẾN**

HƯNG YÊN – 2024

**NHẬN XÉT**

**Nhận xét của giảng viên hướng dẫn:**

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

.......................................................................................................................................

**GIẢNG VIÊN HƯỚNG DẪN**

*(Ký và ghi rõ họ tên)*

LỜI CAM ĐOAN

Em xin cam đoan bài tập lớn “Phân loại ảnh từ bộ dữ liệu CIFAR-10 bằng PyTorch” là kết quả thực hiện của bản thân em dưới sự hướng dẫn của thầy Nguyễn Minh Tiến.

Những phần sử dụng tài liệu tham khảo trong bài tập lớn đã được nêu rõ trong phần tài liệu tham khảo. Các kết quả trình bày trong bài tập lớn và chương trình xây dựng được hoàn toàn là kết quả do bản thân em thực hiện.

Nếu vi phạm lời cam đoan này, em xin chịu hoàn toàn trách nhiệm trước khoa và nhà trường.

*Hưng Yên, ngày … tháng … năm…..*

Sinh viên

…………………………………..

LỜI CẢM ƠN

Để có thể hoàn thành bài tập lớn này, lời đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Khoa học máy tính, Khoa Công nghệ thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện bài tập lớn môn học này.

Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Minh Tiến đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

Em cũng xin chân thành cảm ơn tất cả các Thầy, các Cô trong Trường đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết, quý báu để giúp em thực hiện được bài tập lớn này.

Mặc dù em đã có cố gắng, nhưng với trình độ còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hy vọng sẽ nhận được những ý kiến nhận xét, góp ý của các Thầy cô về những kết quả triển khai trong bài tập lớn này.

Em xin trân trọng cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 10](#_Toc179587843)

[1.1 Lý do chọn đề tài 10](#_Toc179587844)

[1.2 Mục tiêu của đề tài 11](#_Toc179587845)

[1.2.1 Mục tiêu tổng quát 11](#_Toc179587846)

[1.2.2 Mục tiêu cụ thể 11](#_Toc179587847)

[1.3 Giới hạn và phạm vi của đề tài 11](#_Toc179587848)

[1.3.1 Đối tượng nghiên cứu 11](#_Toc179587849)

[1.3.2 Phạm vi nghiên cứu 11](#_Toc179587850)

[1.4 Nội dung thực hiện 12](#_Toc179587851)

[1.5 Phương pháp tiếp cận 12](#_Toc179587852)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 13](#_Toc179587853)

[2.1 Tìm hiểu về PyTorch 13](#_Toc179587854)

[2.2 Mô hình CNN – Convolution Neural Network 16](#_Toc179587855)

[2.2.1 Convolutional layer 17](#_Toc179587856)

[2.2.2 Lớp gộp - Pooling layer 20](#_Toc179587857)

[2.2.3 Fully connected layer 21](#_Toc179587858)

[2.3 Mô hình VGG16 22](#_Toc179587859)

[CHƯƠNG 3: PHÂN TÍCH VÀ XỬ LÝ DỮ LIỆU 24](#_Toc179587860)

[3.1 Thông tin dữ liệu 24](#_Toc179587861)

[3.1.1 Cấu trúc bộ dữ liệu 24](#_Toc179587862)

[3.1.2 Đặc điểm 24](#_Toc179587863)

[3.1.3 Ứng dụng 25](#_Toc179587864)

[3.1.4 Ưu và nhược điểm 25](#_Toc179587865)

[3.2 Tiền xử lý dữ liệu 25](#_Toc179587866)

[3.2.1 Chuẩn hóa (Normalization) 25](#_Toc179587867)

[3.2.2 Biến đổi dữ liệu (Data Augmentation) 26](#_Toc179587868)

[3.2.3 Chuyển hoá dữ liệu sang Tensor 26](#_Toc179587869)

[3.2.4 Chia dữ liệu cho tập Validation 26](#_Toc179587870)

[3.2.5 Chia lô và xáo trộn 27](#_Toc179587871)

[3.2.6 Kết quả sau khi tiền xử lý 27](#_Toc179587872)

[CHƯƠNG 4: XÂY DỰNG MÔ HÌNH 29](#_Toc179587873)

[4.1 Thêm thư viện 29](#_Toc179587874)

[4.2 Xây dựng các hàm hỗ trợ 30](#_Toc179587875)

[4.2.1 Training\_step 30](#_Toc179587876)

[4.2.2 Validation\_step 31](#_Toc179587877)

[4.2.3 Validation\_epoch\_end 31](#_Toc179587878)

[4.2.4 Epoch\_end 32](#_Toc179587879)

[4.2.5 Accuracy 32](#_Toc179587880)

[4.3 Xây dựng mô hình CNN truyền thống 32](#_Toc179587881)

[4.4 Xây dựng mô hình với VGG16 34](#_Toc179587882)

[4.5 Kết quả sau của hai mô hình 35](#_Toc179587883)

[4.5.1 Kết quả mô hình CNN truyền thống 35](#_Toc179587884)

[4.5.2 Kết quả mô hình VGG16 35](#_Toc179587885)

[4.6 Demo chương trình bằng streamlit 37](#_Toc179587886)

[KẾT LUẬN 38](#_Toc179587887)

[TÀI LIỆU THAM KHẢO 39](#_Toc179587888)

**DANH MỤC HÌNH ẢNH**

[*Hình 2.2: Convolutional Neural Network (CNN) 16*](#_Toc179587889)

[*Hình 2.3: Phép toán tích chập 17*](#_Toc179587890)

[*Hình 2.4: Filter Matrix 17*](#_Toc179587891)

[*Hình 2.5: Feature Map 18*](#_Toc179587892)

[*Hình 2.6: Một số bộ lọc – kernal khác 18*](#_Toc179587893)

[*Hình 2.7: Bước nhảy – Stride 19*](#_Toc179587894)

[*Hình 2.8: Padding 19*](#_Toc179587895)

[*Hình 2.9: Hàm ReLU 20*](#_Toc179587896)

[*Hình 2.10: Pooling layer 21*](#_Toc179587897)

[*Hình 2.11: Fully connected layer 21*](#_Toc179587898)

[*Hình 2.12: Mô hình VGG16 22*](#_Toc179587899)

[*Hình 2.13: Cấu trúc VGG16 23*](#_Toc179587900)

[*Hình 3.1: Hình ảnh trước và sau khi tiền xử lý 27*](#_Toc179587901)

[*Hình 3.2: Batch trước và sau khi được tiền xử lý 28*](#_Toc179587902)

[*Hình 4.1: Accuracy và loss funciton model CNN 35*](#_Toc179587906)

[*Bảng 4.3: Tổng hợp độ đo trên mô hình CNN 35*](#_Toc179587907)

[*Hình 4.2: Accuracy và loss funcition model VGG16 36*](#_Toc179587908)

[*Hình 4.3: Demo đơn giản bằng streamlit 37*](#_Toc179587910)

**DANH MỤC BẢNG BIỂU**

[*Bảng 3.1: Tổng quan về example và class 28*](#_Toc179587950)

[*Bảng 4.1: Cấu trúc các lớp của mô hình CNN 33*](#_Toc179587951)

[*Bảng 4.2: Mô hình VGG16 34*](#_Toc179587952)

[*Bảng 4.3: Tổng hợp độ đo trên mô hình CNN 35*](#_Toc179587953)

[*Bảng 4.4: Tổng hợp độ đo trên mô hình VGG16 36*](#_Toc179587954)

**DANH MỤC CÁC THUẬT NGỮ**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Từ viết tắt | Cụm từ tiếng anh | Diễn giải |
| 1 | CNN | Convolutional Neural Network | Mạng nơ-ron tích chập |
| 2 | conv | Convolutional | Tích chập |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# TỔNG QUAN VỀ ĐỀ TÀI

## Lý do chọn đề tài

Với sự phát triển không ngừng của khoa học công nghệ, đặc biệt là các thiết bị được hỗ trợ công nghệ xử lý ảnh ngày càng hiện đại và được sử dụng phổ biến trong đời sống con người đã làm cho lượng thông tin thu được bằng hình ảnh ngày càng tăng và phổ biến. Theo đó, lĩnh vực xử lý hình ảnh cũng được chú trọng phát triển, ứng dụng rộng rãi trong đời sống xã hội hiện đại. Không chỉ dừng lại ở việc chỉnh sửa, tăng chất lượng hình ảnh mà với công nghệ xử lý ảnh hiện nay chúng ta có thể giải quyết các bài toán nhận dạng chữ viết, nhận dạng dấu vân tay, nhận dạng khuôn mặt, nhận dạng vật thể… Và đặc biệt phát hiện và phân loại ảnh.

Xây dựng một mô hình có thể nhận diện và phân loại hình ảnh như: chó, mèo, ô tô, tàu thuyến, máy bay… Mô hình này sẽ giúp ích cho việc tự động hóa nhiều quy trình khác nhau và cung cấp thông tin hữu ích từ dữ liệu hình ảnh. Cụ thể trong y học, mô hình có thể chẩn đoán bệnh, giúp xác định các vùng bất thường trong hình ảnh chụp từ máy MRI, CT hay X-quang hay trong công nghệ ô tô tự động mô hình sẽ giúp cho ô tô nhận diện vật cản từ đó đưa ra các phương hướng xử lý.

Với mong muốn xây dựng một mô hình có thể đảm đương các yêu cầu trên nên từ đó em đã chọn xây dựng mô hình “Phân loại hình ảnh”. Ở đây em sẽ sử dụng bộ dữ liệu “CiFar-10” để từ đó có thể hiểu hơn về mô hình và có các bước cải tiến trong quá trình xây dựng project sau này.

Mục tiêu của đề tài

### Mục tiêu tổng quát

* Từ bộ dữ liệu CiFar-10 nhận diện và dự đoán được các nhãn của ảnh.
* Tốc độ xử lý nhanh, chính xác.
* Tránh bị overfitting hay underfitting.

### Mục tiêu cụ thể

Mô hình phân loại hình ảnh cho tệp dữ liệu CIFAR cần đạt được mục tiêu sau:

* Mô hình phải nhanh, chính xác cao.
* Đáp ứng yêu cầu hình ảnh.
* Mô hình phải phân loại được các hình ảnh.

## Giới hạn và phạm vi của đề tài

### Đối tượng nghiên cứu

* Đối tượng nghiên cứu: mô hình phân loại hình ảnh
* Khách thể nghiên cứu:
  + Những mô hình phân loại hình ảnh;
  + Quy trình hoạt động của 1 mô hình học máy;
  + Các app, trang web có tích hợp mô hình phân loại ảnh.

### Phạm vi nghiên cứu

* Phạm vi không gian: tại các app, web có tích hợp mô hình phân loại ảnh
* Phạm vi thời gian: Trong 4 tháng vừa qua
* Ý nghĩa khoa học: Tổng hợp các kiến thức, quy trình để tạo 1mô hình phân loại.
* Thực tiễn của đề tài: Sau này có thể áp dụng để chuẩn đoán bệnh cho người, động vật thực vật, trong phân loại thực phẩm...

## Nội dung thực hiện

* Thu thập dữ liệu ban đầu phục vụ cho việc pre-train
* Lựa chọn pre-trained model (mô hình được đào tạo trước)
* Xử lý bộ dữ liệu ảnh và ảnh đầu vào để nhận dạng
* Tích hợp hệ thống nhận dạng ảnh vào hệ thống chẩn đoán nhãn của đối tượng trong ảnh

## Phương pháp tiếp cận

Nghiên cứu về các phương pháp, công cụ, công nghệ được sử dụng để thực hiện đề tài. Lựa chọn những công nghệ tối ưu và hiện đại để bắt kịp xu thế phát triển không ngừng của công nghệ hiện nay.

Sau khi đã nắm bắt được công nghệ và hướng đi, bắt tay vào lập kế hoạch triển khai dự án, đưa ra những công việc cụ thể cần thực hiện. Tiến hành tìm hiểu các thuật toán, phương pháp, tính khả thi cũng như bất khả thi của những công việc cần thực hiện.

Tiếp theo đi phân tích, tiến hành xây dựng ứng dụng, từng bước hoàn chỉnh, ghép nối dự án thành sản phẩm hoàn chỉnh.

Khi đã xây dựng hoàn chỉnh mã nguồn, tiến hành thử nghiệm, chỉnh sửa, cập nhật sai sót.

Lựa chọn nền tảng triển khai hệ thống ứng dụng chạy trên web hay ứng dụng desktop sao cho thuận tiện nhất.

Cuối cùng là tổng kết và đưa ra hướng phát triển trong tương lai.

# CƠ SỞ LÝ THUYẾT

## Tìm hiểu về PyTorch

Pytorch là framework được phát triển bởi Facebook. Đây là một ông lớn về công nghệ đầu tư rất nhiều nguồn lực cho việc phát triển Trí tuệ nhân tạo. Pytorch được phát triển với giấy phép mã nguồn mở do đó nó tạo được cho mình một cộng đồng rất lớn.Một cộng đồng lớn đồng nghĩa với nhiều tài nguyên để học và các vấn đề của bạn có thể đã có ai đó giải quyết và chia sẻ với cộng đồng. Pytorch cùng với Tensorflow và Keras là một trong những framework phổ biến được sử dụng trong các bài toán về Deep Learning hiện nay. Đặc biệt, trong các lĩnh vực nghiên cứu, hầu như các tác giả đều sử dụng pytorch để triển khai bài toán của mình. Pytorch cho thấy lợi thế của nó trong lĩnh vực nghiên cứu bởi việc rất dễ dàng để bạn debug và visualize, ngoài ra nó theo cơ chế Dynamic Graphs cho phép giảm thời gian huấn luyện mô hình.

Điều đáng chú ý là PyTorch đã áp dụng một cải tiến của Chainer được gọi là phân biệt tự động ở chế độ đảo ngược . Về cơ bản, nó giống như một máy ghi âm ghi lại các thao tác đã hoàn thành và sau đó phát lại để tính toán độ dốc. Điều này làm cho PyTorch tương đối đơn giản để gỡ lỗi và thích ứng tốt với một số ứng dụng nhất định như mạng thần kinh động. Nó phổ biến cho việc tạo mẫu vì mỗi lần lặp lại có thể khác nhau.

PyTorch đặc biệt phổ biến với các nhà phát triển Python vì nó được viết bằng Python và sử dụng chế độ thực thi háo hức bắt buộc, được xác định theo từng lần chạy của ngôn ngữ đó, trong đó các hoạt động được thực thi khi chúng được gọi từ Python. Khi mức độ phổ biến của ngôn ngữ lập trình Python vẫn tiếp tục, một cuộc khảo sát đã xác định sự tập trung ngày càng tăng vào các nhiệm vụ AI và machine learning, đồng thời, áp dụng nhiều hơn PyTorch liên quan. Điều này làm cho PyTorch trở thành một lựa chọn tốt cho các nhà phát triển Python mới làm quen với deep learning và thư viện các khóa học deep learning dựa trên PyTorch ngày càng phát triển. API vẫn nhất quán từ các bản phát hành đầu tiên, có nghĩa là mã này tương đối dễ hiểu đối với các nhà phát triển Python có kinh nghiệm.

Thế mạnh đặc biệt của PyTorch là tạo mẫu nhanh và các dự án nhỏ hơn. Tính dễ sử dụng và tính linh hoạt của nó cũng khiến nó được cộng đồng học thuật và nghiên cứu yêu thích.

Các nhà phát triển của Facebook đã làm việc chăm chỉ để cải thiện các ứng dụng hiệu quả của PyTorch. Các bản phát hành gần đây đã cung cấp các cải tiến như hỗ trợ công cụ trực quan hóa TensorBoard của Google và biên dịch đúng lúc. Nó cũng đã mở rộng hỗ trợ cho ONNX (Trao đổi mạng thần kinh mở), cho phép các nhà phát triển kết hợp với các khung hoặc thời gian chạy deep learning hoạt động tốt nhất cho ứng dụng của họ.

Một số thuộc tính quan trọng của PyTorch là:

- Có một cộng đồng rộng lớn và sôi động tại cộng đồng PyTorch.org với các tài liệu và hướng dẫn tuyệt vời. Các diễn đàn đang hoạt động và hỗ trợ.

- Nó được viết bằng Python và được tích hợp với các thư viện Python phổ biến như NumPy để tính toán khoa học, SciPy và Cython để biên dịch Python sang C để có hiệu suất tốt hơn. Vì cú pháp và cách sử dụng của nó tương tự như Python nên PyTorch tương đối dễ học đối với các nhà phát triển Python.

- Nó được hỗ trợ tốt bởi các nền tảng đám mây lớn.

- Ngôn ngữ kịch bản, được gọi là TorchScript, rất dễ sử dụng và linh hoạt khi ở chế độ háo hức. Đây là chế độ thực thi bắt đầu nhanh, trong đó các thao tác được thực thi ngay lập tức khi chúng được gọi từ Python, nhưng cũng có thể được chuyển sang mô hình đồ thị để tăng tốc và tối ưu hóa trong môi trường thời gian chạy C++.

- Nó hỗ trợ CPU, GPU và xử lý song song cũng như đào tạo phân tán. Điều này có nghĩa là công việc tính toán có thể được phân phối giữa nhiều lõi CPU và GPU và việc đào tạo có thể được thực hiện trên nhiều GPU trên nhiều máy.

- PyTorch hỗ trợ các biểu đồ tính toán động, cho phép thay đổi hành vi mạng trong thời gian chạy. Điều này mang lại lợi thế linh hoạt lớn so với phần lớn các khung máy học vốn yêu cầu mạng thần kinh phải được xác định là đối tượng tĩnh trước khi chạy.

- PyTorch Hub là kho lưu trữ các mô hình được đào tạo trước có thể được gọi, trong một số trường hợp chỉ với một dòng mã.

- Các thành phần tùy chỉnh mới có thể được tạo dưới dạng lớp con của lớp Python tiêu chuẩn, các tham số có thể dễ dàng được chia sẻ với các bộ công cụ bên ngoài như TensorBoard và các thư viện có thể dễ dàng được nhập và sử dụng nội tuyến.

- PyTorch có một bộ API được đánh giá cao có thể được sử dụng để mở rộng chức năng cốt lõi.

- Nó hỗ trợ cả “chế độ háo hức” để thử nghiệm và “chế độ biểu đồ” để thực hiện hiệu suất cao.

- Nó có một bộ sưu tập lớn các công cụ và thư viện trong các lĩnh vực từ thị giác máy tính đến học tập tăng cường.

- Nó hỗ trợ giao diện front-end thuần túy C++ quen thuộc với các lập trình viên Python và có thể được sử dụng để xây dựng các ứng dụng C++ hiệu suất cao.



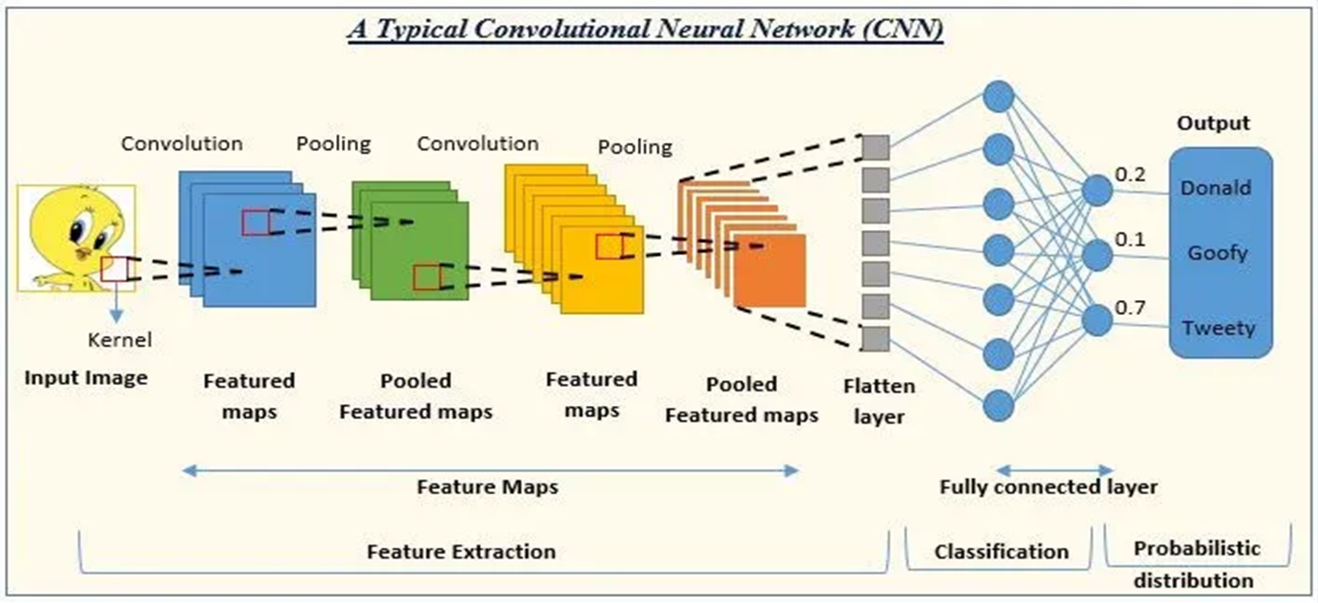
Hình 2.1: PyTorch

## Mô hình CNN – Convolution Neural Network

Convolutional Neural Network (CNN) là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng Computer Vision. CNN được dùng trong nhiều bài toán như nhận dạng ảnh, phân tích video, ảnh MRI, hoặc cho các bài của lĩnh vực xử lý ngôn ngữ tự nhiên, và hầu hết đều giải quyết tốt các bài toán này.

CNN là một kiến trúc mạng neuron rất thích hợp cho các bài toán mà dữ liệu là ảnh hoặc video. Có hai loại layer chính trong CNN: Convolutional layer và Pooling layer.

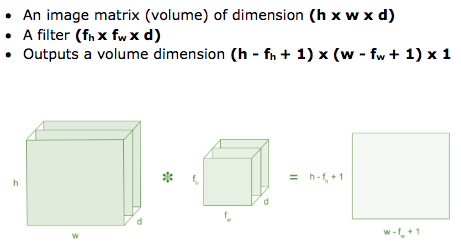
Để hình dung rõ hơn về cấu trúc cũng như cách hoạt động của một mô hình convolution neural network ta hãy nhìn xuống ảnh 2.2 dưới đây:



Hình 2.2: Convolutional Neural Network (CNN)

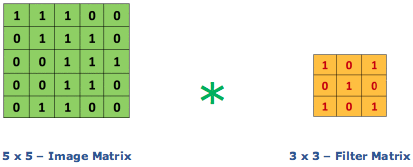
### Convolutional layer

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.



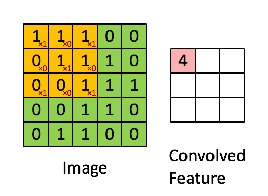
Hình 2.3: Phép toán tích chập

Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.



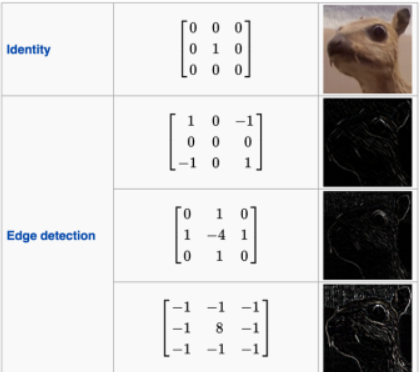
Hình 2.4: Filter Matrix

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



Hình 2.5: Feature Map

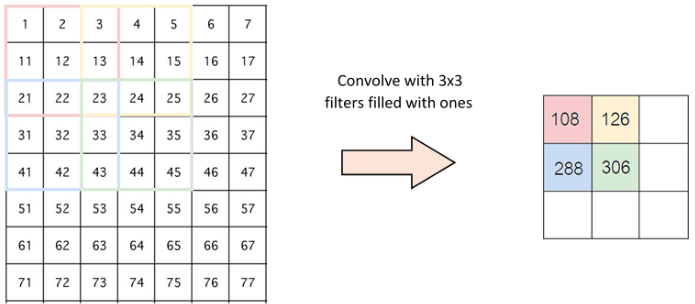
Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.



Hình 2.6: Một số bộ lọc – kernal khác

#### Bước nhảy – Stride

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2



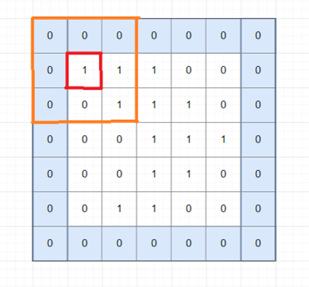
Hình 2.7: Bước nhảy – Stride

#### Đường viền – Padding

Padding là một kỹ thuật được sử dụng để bảo toàn kích thước không gian của hình ảnh đầu vào sau các hoạt động tích chập trên bản đồ đặc điểm. Padding bao gồm việc thêm các pixel bổ sung xung quanh đường viền của bản đồ đặc điểm đầu vào trước khi tích chập.

Có thể thực hiện điều này theo hai cách:

* Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).
* Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.



Hình 2.8: Padding

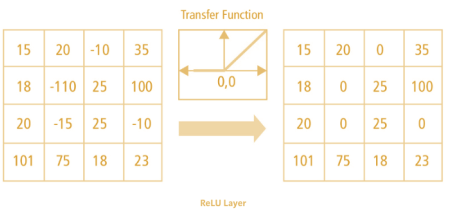
#### Hàm kích hoạt ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là:

ƒ (x) = max (0, x).

Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.

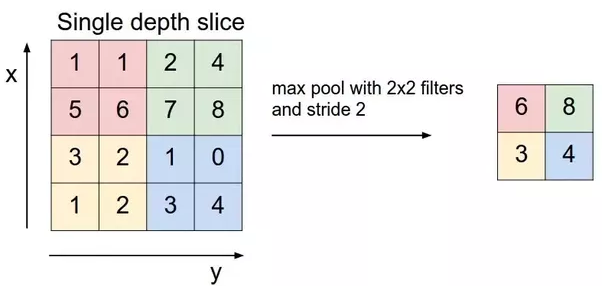
Có 1 số hà phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.



Hình 2.9: Hàm ReLU

### Lớp gộp - Pooling layer

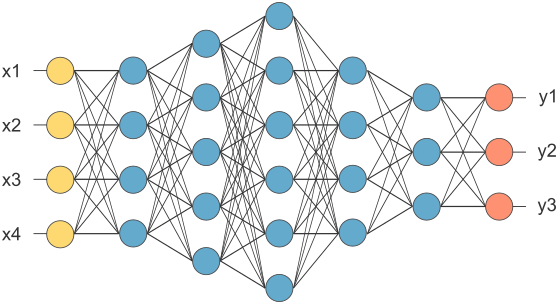
* Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:
* Max Pooling
* Average Pooling
* Sum Pooling
* Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling



Hình 2.10: Pooling layer

### Fully connected layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra.



Hình 2.11: Fully connected layer

## Mô hình VGG16

VGG16 là mô hình mạng nơ ron tích chập được đề xuất bởi K. Simonyan và A. Zisserman từ Đại học Oxford trong bài báo “Mạng chuyển đổi rất sâu để nhận dạng hình ảnh quy mô lớn”. Mô hình này đạt được độ chính xác trong thử nghiệm top 5 là 92,7% trong ImageNet, đây là bộ dữ liệu gồm hơn 14 triệu hình ảnh thuộc 1000 lớp. Đó là một trong những mẫu nổi tiếng được gửi tới ILSVRC-2014. Nó cải thiện AlexNet bằng cách thay thế các bộ lọc có kích thước hạt nhân lớn (lần lượt là 11 và 5 ở lớp chập thứ nhất và thứ hai) bằng nhiều bộ lọc có kích thước hạt nhân 3×3 lần lượt. VGG16 đã được đào tạo trong nhiều tuần và đang sử dụng GPU NVIDIA Titan Black.



Hình 2.12: Mô hình VGG16

Đầu vào của lớp cov1 có kích thước cố định là hình ảnh 224 x 224 RGB. Hình ảnh được truyền qua một chồng các lớp chập (đối lưu), trong đó các bộ lọc được sử dụng với trường tiếp nhận rất nhỏ: 3×3 (là kích thước nhỏ nhất để nắm bắt khái niệm trái/phải, lên/xuống, giữa ). Ở một trong các cấu hình, nó cũng sử dụng các bộ lọc tích chập 1×1, có thể được coi là một phép biến đổi tuyến tính của các kênh đầu vào (tiếp theo là phi tuyến tính). Bước tích chập được cố định ở 1 pixel; phần đệm không gian của đối lưu. đầu vào của lớp sao cho độ phân giải không gian được giữ nguyên sau khi tích chập, tức là phần đệm là 1 pixel cho đối lưu 3 × 3. các lớp. Việc gộp không gian được thực hiện bởi năm lớp gộp tối đa, tuân theo một số đối lưu.  các lớp (không phải tất cả các lớp chuyển đổi đều được theo sau bằng cách gộp tối đa). Việc gộp tối đa được thực hiện trên cửa sổ 2 × 2 pixel, với bước 2.

Ba lớp được kết nối đầy đủ (FC) tuân theo một chồng các lớp tích chập (có độ sâu khác nhau trong các kiến ​​trúc khác nhau): hai lớp đầu tiên có 4096 kênh mỗi lớp, lớp thứ ba thực hiện phân loại ILSVRC 1000 chiều và do đó chứa 1000 kênh (một cho mỗi kênh). lớp học). Lớp cuối cùng là lớp soft-max. Cấu hình của các lớp được kết nối đầy đủ là giống nhau trong tất cả các mạng.

Tất cả các lớp ẩn đều được trang bị tính phi tuyến tính chỉnh lưu (ReLU). Cũng cần lưu ý rằng không có mạng nào (ngoại trừ một mạng) chứa Chuẩn hóa phản hồi cục bộ (LRN), việc chuẩn hóa như vậy không cải thiện hiệu suất trên tập dữ liệu ILSVRC nhưng dẫn đến tăng mức tiêu thụ bộ nhớ và thời gian tính toán.



Hình 2.13: Cấu trúc VGG16

# PHÂN TÍCH VÀ XỬ LÝ DỮ LIỆU

## Thông tin dữ liệu

CIFAR-10 (Canadian Institute For Advanced Research - 10 classes) được phát triển bởi Alex Krizhevsky, Geoffrey Hinton và các cộng sự tại University of Toronto. Mục đích là dùng để huấn luyện và kiểm thử các mô hình phân loại hình ảnh, đặc biệt là các mô hình mạng nơ-ron sâu.

### Cấu trúc bộ dữ liệu

* Số lượng ảnh: Tổng cộng có 60.000 ảnh màu, trong đó có:
* 50.000 ảnh cho tập huấn luyện (training set).
* 10.000 ảnh cho tập kiểm thử (test set).
* Kích thước ảnh: Mỗi ảnh có kích thước 32x32 pixels, với 3 kênh màu (RGB).
* Số lớp (classes): Bộ dữ liệu gồm 10 lớp (categories), mỗi lớp đại diện cho một đối tượng cụ thể. Các lớp này bao gồm:
* (0) Máy bay (airplane)
* (1) Ô tô (automobile)
* (2) Chim (bird)
* (3) Mèo (cat)
* (4) Hươu (deer)
* (5) Chó (dog)
* (6) Ếch (frog)
* (7) Ngựa (horse)
* (8) Tàu thuỷ (ship)
* (9) Xe tải (truck)

### Đặc điểm

Độ phân giải thấp: Các ảnh đều có kích thước nhỏ 32x32, giúp mô hình học và đánh giá nhanh chóng, nhưng cũng đặt ra thử thách cho việc nhận diện các đặc trưng chi tiết.

Đa dạng: Mỗi lớp có 6.000 ảnh, với 5.000 ảnh cho huấn luyện và 1.000 ảnh cho kiểm thử. Các ảnh được lựa chọn đa dạng về góc độ, ánh sáng, và màu sắc, giúp mô hình có thể học được nhiều đặc trưng khác nhau.

### Ứng dụng

Học sâu (Deep Learning): CIFAR-10 thường được sử dụng để huấn luyện các mô hình mạng nơ-ron tích chập (Convolutional Neural Network - CNN), làm cơ sở để nghiên cứu và phát triển các thuật toán học máy.

Benchmark: Bộ dữ liệu này được sử dụng như một benchmark để so sánh hiệu năng của các mô hình khác nhau trong việc phân loại hình ảnh.

### Ưu và nhược điểm

#### Ưu điểm

* Dễ sử dụng: CIFAR-10 là một bộ dữ liệu cơ bản và dễ sử dụng, phù hợp cho người mới bắt đầu học về học máy và học sâu.
* Phổ biến: Được nhiều nghiên cứu sử dụng, điều này giúp dễ dàng so sánh và đánh giá kết quả của các mô hình khác nhau.

#### Nhược điểm

* Độ phân giải thấp: Ảnh có kích thước nhỏ, điều này gây khó khăn khi phân tích các đặc trưng chi tiết của đối tượng.
* Không đủ phức tạp: Đối với các bài toán thực tế, CIFAR-10 không đủ phức tạp để đánh giá khả năng của các mô hình nâng cao. Trong những trường hợp này, người ta thường chuyển sang bộ dữ liệu ImageNet.

## Tiền xử lý dữ liệu

### Chuẩn hóa (Normalization)

Các giá trị pixel của ảnh ban đầu nằm trong khoảng [0, 255], nhưng để mô hình học tốt hơn, ta cần chuẩn hóa chúng về khoảng [0, 1] hoặc [-1, 1]. Điều này giúp các giá trị dữ liệu được phân phối ổn định và hỗ trợ quá trình tối ưu hóa.

Trong CIFAR-10, thường sử dụng chuẩn hóa để đưa giá trị pixel về khoảng [-1, 1] bằng cách áp dụng 1trong 2 công thức dưới đây:

* transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
* transforms.Normalize(mean=[0.4914, 0.4822, 0.4465], std=[0.2470, 0.2435, 0.2616])

Các giá trị mean và std này là trung bình và độ lệch chuẩn của bộ dữ liệu CIFAR-10.

### Biến đổi dữ liệu (Data Augmentation)

Để cải thiện hiệu suất của mô hình và tránh tình trạng overfitting, biến đổi dữ liệu (Data Augmentation) là một kỹ thuật rất hữu ích. CIFAR-10 có số lượng ảnh tương đối nhỏ, do đó cần tăng tính đa dạng của dữ liệu bằng cách thực hiện một số phép biến đổi ngẫu nhiên:

* Random Horizontal Flip: Lật ngẫu nhiên ảnh theo chiều ngang, giúp mô hình học được nhiều đặc trưng hơn từ cùng một ảnh nhưng ở các hướng khác nhau.
* **Random Crop**: Cắt ngẫu nhiên một vùng từ ảnh với một tỉ lệ xác định. Việc này giúp mô hình tập trung vào các chi tiết khác nhau của đối tượng.
* **Color Jitter**: Thay đổi độ sáng, độ tương phản, độ bão hòa, hoặc màu sắc để làm phong phú thêm sự đa dạng của dữ liệu.

### Chuyển hoá dữ liệu sang Tensor

Sau khi thực hiện các bước biến đổi, dữ liệu cần được chuyển đổi thành **Tensor**, vì các thư viện học sâu như PyTorch làm việc với các đối tượng dạng tensor. Ta có thể dung lệnh dưới đây để biến đổi:

* transforms.ToTensor()

### Chia dữ liệu cho tập Validation

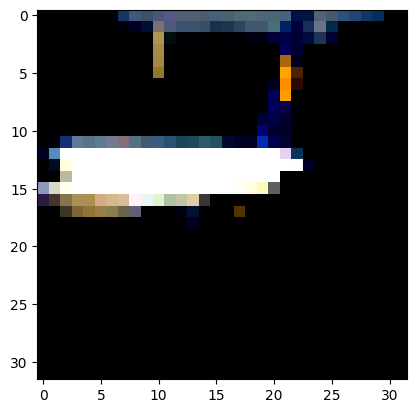
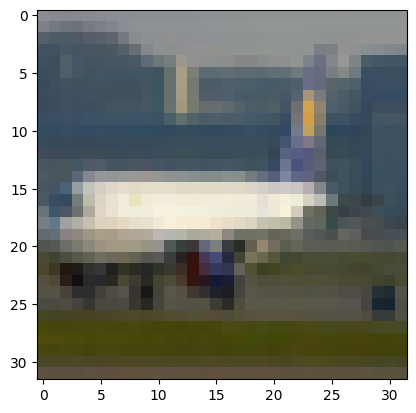
CIFAR-10 mặc định không có tập xác thực (validation set). Bạn có thể tạo tập này bằng cách tách một phần từ training set, ở đây ta sẽ tách 10% từ tập huấn luyện để làm tập xác thực. Điều này giúp đánh giá độ chính xác của mô hình trong quá trình huấn luyện mà không dùng đến tập kiểm thử (test set).

### Chia lô và xáo trộn

* Chia lô (Batching): Dữ liệu cần được chia thành các lô (batches) để huấn luyện một cách hiệu quả. Trong bài này ta sẽ đặt kích thước batch là 128. Việc chia lô giúp tăng tốc quá trình huấn luyện bằng cách xử lý nhiều mẫu cùng lúc.
* Xáo trộn (Shuffling): Xáo trộn dữ liệu trước mỗi epoch để đảm bảo mô hình không bị học theo thứ tự của dữ liệu. Việc này giúp mô hình học tốt hơn và tránh overfitting.

### Kết quả sau khi tiền xử lý

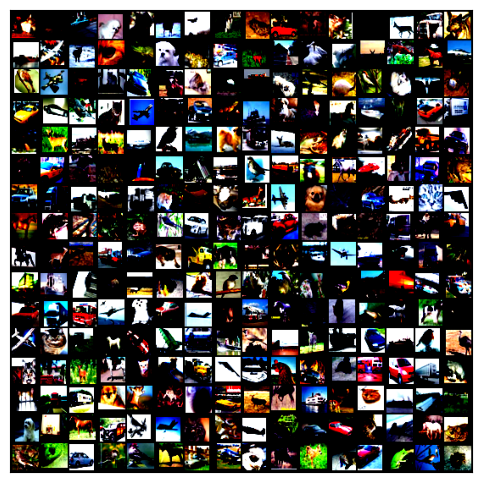
Dưới đây là hình ảnh trước và sau khi tiền xử lý



Hình 3.1: Hình ảnh trước và sau khi tiền xử lý

* Chúng ta có thể thấy rằng sau khi được tăng cường tuy nhìn bằng mắt thường thì chỉ phần nào đoán được nó là chiếc máy bay, nhưng trong máy học đây là đặc trưng khiến mô hình có thể nhận diện tốt hơn.
* Các phần nền không liên quan đã được làm tối đi và hiện lên chỉ có chiếc máy bay và đường nét của nó.

Batch trước và sau khi tiền xử lý



Hình 3.2: Batch trước và sau khi được tiền xử lý

* Khi nhìn vào batch ta sẽ có cái nhìn tổng quan hơn về lô ảnh sẽ được train. Kết quả sẽ rất là khả thi cho mô hình mà ta sẽ xây dựng sắp tới.
* Sau khi đã tiền xử lý xong suôi ta đã có đầy đủ các tệp để xây dựng mô hình. Dưới đây sẽ là cái nhìn tổng quan về số lượng example cho các tệp

Bảng 3.1: Tổng quan về example và class

|  |  |  |
| --- | --- | --- |
|  | Example | Class |
| Train | 45000 | 10 |
| Validation | 5000 | 10 |
| Test | 10000 | 10 |

# XÂY DỰNG MÔ HÌNH

## Thêm thư viện

Đầu tiên ta sẽ đi qua về một số thư viện mà ta sẽ sử dụng

Os: Thư viện này thuộc Python chuẩn và được sử dụng để tương tác với hệ điều hành. Nó giúp thực hiện các thao tác như đọc hoặc liệt kê các file, xử lý đường dẫn, và thực hiện các thao tác hệ thống khác. Thư viện này dùng để liệt kê danh sách các thư mục và file trong thư mục dữ liệu (os.listdir(data\_dir)).

Tarfile: Thư viện này cũng thuộc Python chuẩn, được sử dụng để xử lý các file tar (nén .tar, .tar.gz). Nó giúp mở, đọc, và giải nén các file tar. Thư viện này có thể được sử dụng để mở và giải nén file dữ liệu nếu cần thiết (trong trường hợp dữ liệu của bạn được cung cấp dưới dạng file tar).

Torch: Đây là thư viện cốt lõi của PyTorch, một thư viện mã nguồn mở cho học sâu. PyTorch được sử dụng rộng rãi để xây dựng và huấn luyện các mô hình học sâu.

Torch.nn: Đây là mô-đun cung cấp các lớp và hàm cần thiết để định nghĩa các mạng nơ-ron. Nó bao gồm các lớp như Linear, Conv2d, ReLU, và nhiều lớp khác để xây dựng mạng.

Torch.nn.functional: Cung cấp các hàm hoạt động (activation functions, loss functions, v.v.) mà không cần khởi tạo như lớp nn. Ví dụ như F.relu(), F.cross\_entropy().

Torch.utils.data.DataLoader: Đây là lớp giúp bạn tạo các batches từ bộ dữ liệu. Nó hỗ trợ việc xáo trộn dữ liệu, chia thành các lô (batches), và giúp tăng tốc độ trong quá trình huấn luyện.

DataLoader(): Sử dụng để lấy các lô dữ liệu từ tập huấn luyện, kiểm thử hoặc xác thực, giúp mô hình huấn luyện với từng phần của dữ liệu thay vì toàn bộ cùng một lúc.

Torch.utils.data.random\_split: Hàm này giúp bạn chia tập dữ liệu thành các phần nhỏ hơn, ví dụ như chia bộ dữ liệu thành training set và validation set một cách ngẫu nhiên.

Torchvision: Đây là thư viện mở rộng của PyTorch hỗ trợ xử lý dữ liệu hình ảnh. Nó cung cấp các bộ dữ liệu phổ biến, các lớp và hàm tiền xử lý dữ liệu, giúp cho việc làm việc với hình ảnh trở nên dễ dàng.

Torchvision.datasets.ImageFolder: Là một lớp tiện ích giúp tạo bộ dữ liệu từ cấu trúc thư mục. Mỗi thư mục con tương ứng với một lớp của bộ dữ liệu và các hình ảnh trong thư mục đó là các ví dụ của lớp đó.

Torchvision.datasets.utils.download\_url: Hàm này hỗ trợ tải dữ liệu từ internet và lưu trữ tại một vị trí cục bộ. Đây là cách tiện lợi để tải các bộ dữ liệu lớn từ các nguồn trực tuyến.

Torchvision.transforms.ToTensor: Đây là một phép biến đổi thuộc torchvision.transforms. Phép biến đổi này chuyển đổi dữ liệu ảnh từ định dạng PIL Image hoặc NumPy array sang dạng torch.Tensor và chuẩn hóa giá trị pixel từ khoảng [0, 255] về khoảng [0, 1].

Để việc train model thuận lợi hơn thì trước hết ta sẽ đi xây dựng các hàm hỗ trợ cho việc train model, cùng với đó là sẽ xử lý về thiết bị. Ở đây là sẽ kiểm tra xem thiết bị có GPU không, nếu như có GPU ta sẽ gói dữ liệu thành các gói để di chuyển vào bộ nhớ của GPU

Tóm lại :

* PyTorch và torchvision là hai thư viện quan trọng cho việc xây dựng, huấn luyện và đánh giá các mô hình học sâu, đặc biệt là xử lý dữ liệu ảnh.
* Các thư viện os và tarfile được sử dụng để thao tác với hệ thống file và giải nén các bộ dữ liệu, trong khi torch và torchvision giúp xử lý dữ liệu, định nghĩa mạng, và huấn luyện mô hình.

## Xây dựng các hàm hỗ trợ

### Training\_step

Mục đích: Xử lý một lô (batch) dữ liệu trong quá trình huấn luyện mô hình.

Các bước thực hiện:

* Nhận batch dữ liệu (images, labels) từ tập huấn luyện.
* Thực hiện dự đoán (out = model(images)) với mô hình được cung cấp (model).
* Tính loss giữa dự đoán và nhãn thực (loss = nn.CrossEntropyLoss()(out, labels)), sử dụng hàm mất mát Cross Entropy Loss cho bài toán phân loại.

Kết quả: Hàm này trả về giá trị loss, cho phép cập nhật trọng số của mô hình dựa trên gradient.

### Validation\_step

Mục đích: Xử lý một batch dữ liệu trong quá trình validation để đánh giá hiệu suất của mô hình.

Các bước thực hiện:

* Nhận batch dữ liệu (images, labels) từ tập validation.
* Thực hiện dự đoán (out = model(images)).
* Tính loss giữa dự đoán và nhãn thực (loss = nn.CrossEntropyLoss()(out, labels)).
* Tính độ chính xác (accuracy) của dự đoán (acc = self.accuracy(out, labels)).

Kết quả: Trả về một từ điển (dict) chứa giá trị validation loss và validation accuracy cho batch đó. Cụ thể:

### Validation\_epoch\_end

Mục đích: Tổng hợp kết quả từ các bước validation trong một epoch.

Các bước thực hiện:

* Nhận đầu vào là một danh sách outputs - mỗi phần tử trong danh sách này là kết quả từ validation\_step của mỗi batch trong epoch.
* Tính trung bình loss của tất cả các batch trong epoch (epoch\_loss = torch.stack(batch\_losses).mean()).
* Tính trung bình accuracy của tất cả các batch trong epoch (epoch\_acc = torch.stack(batch\_accs).mean()).

Kết quả: Trả về một từ điển chứa validation loss và validation accuracy trung bình của epoch đó.

### Epoch\_end

Mục đích: Hiển thị kết quả sau khi kết thúc mỗi epoch trong quá trình huấn luyện.

Các bước thực hiện:

* Nhận các thông tin gồm: epoch hiện tại và kết quả (result) từ quá trình huấn luyện và validation của epoch đó.
* In ra kết quả gồm train\_loss, val\_loss, và val\_acc cho epoch hiện tại.

Vai trò: Hỗ trợ theo dõi quá trình huấn luyện, giúp bạn biết được hiệu suất của mô hình theo thời gian.

### Accuracy

Mục đích: Tính toán độ chính xác (accuracy) của dự đoán so với nhãn thực.

Các bước thực hiện:

* torch.max(outputs, dim=1): Lấy giá trị lớn nhất và chỉ số tương ứng (ở đây là predictions) theo chiều dim=1, nghĩa là dọc theo số lớp (đối với phân loại). Điều này giúp xác định lớp mà mô hình dự đoán cho mỗi ảnh.
* So sánh các dự đoán (preds) với nhãn thực (labels) và tính tỷ lệ chính xác (torch.sum(preds == labels).item() / len(preds)).
* torch.tensor(): Chuyển kết quả tỷ lệ chính xác thành dạng tensor.

Kết quả: Trả về giá trị accuracy dưới dạng tensor, cho biết tỷ lệ dự đoán đúng so với tổng số mẫu.

## Xây dựng mô hình CNN truyền thống

Layer (type): Tên và loại lớp (layer) trong mô hình. Ví dụ Conv2d là lớp tích chập 2D (Convolutional layer), ReLU là hàm kích hoạt (Rectified Linear Unit), MaxPool2d là lớp pooling tối đa 2D, và Linear là lớp kết nối đầy đủ (fully connected layer).

Output Shape: Kích thước đầu ra của mỗi lớp. -1 biểu thị batch size. Số này sẽ thay đổi khi bạn đưa batch có kích thước cụ thể vào mô hình.

Ví dụ, đầu ra của lớp đầu tiên Conv2d-1 là [batch\_size, 32, 32, 32], tức là đầu ra của lớp này có 32 kênh và kích thước không gian của ảnh là 32x32.

Param #: Số lượng tham số trong mỗi lớp. Conv2d có tham số bao gồm các bộ lọc (filters) và bias (bộ trọng số và thiên vị). Ví dụ, lớp Conv2d-1 có 896 tham số. Điều này được tính từ số bộ lọc (32 bộ lọc) nhân với kích thước bộ lọc (3 kênh đầu vào \* 3x3 kích thước bộ lọc) cộng với 32 bias (32 \* (3 \* 3 \* 3) + 32 = 896).

Bảng 4.1: Cấu trúc các lớp của mô hình CNN

|  |  |  |
| --- | --- | --- |
| Layer (type) | Output Shape | Param # |
| Conv2d-1 | [-1, 32, 32, 32] | 896 |
| ReLU-2 | [-1, 32, 32, 32] | 0 |
| Conv2d-3 | [-1, 64, 32, 32] | 18496 |
| ReLU-4 | [-1, 64, 32, 32] | 0 |
| MaxPool2d-5 | [-1, 64, 16, 16] | 0 |
| Conv2d-6 | [-1, 128, 16, 16] | 73856 |
| ReLU-7 | [-1, 128, 16, 16] | 0 |
| Conv2d-8 | [-1, 128, 16, 16] | 147584 |
| ReLU-9 | [-1, 128, 16, 16] | 0 |
| MaxPool2d-10 | [-1, 128, 8, 8] | 0 |
| Conv2d-11 | [-1, 256, 8, 8] | 295168 |
| ReLU-12 | [-1, 256, 8, 8] | 0 |
| Conv2d-13 | [-1, 256, 8, 8] | 590080 |
| ReLU-14 | [-1, 256, 8, 8] | 0 |
| MaxPool2d-15 | [-1, 256, 4, 4] | 0 |
| Flatten-16 | [-1, 4096] | 0 |
| Linear-17 | [-1, 1024] | 4195328 |
| ReLU-18 | [-1, 1024] | 0 |
| Dropout-19 | [-1, 1024] | 0 |
| Linear-20 | [-1, 512] | 524800 |
| ReLU-21 | [-1, 512] | 0 |
| Dropout-22 | [-1, 512] | 0 |
| Linear-23 | [-1, 10] | 5130 |

## Xây dựng mô hình với VGG16

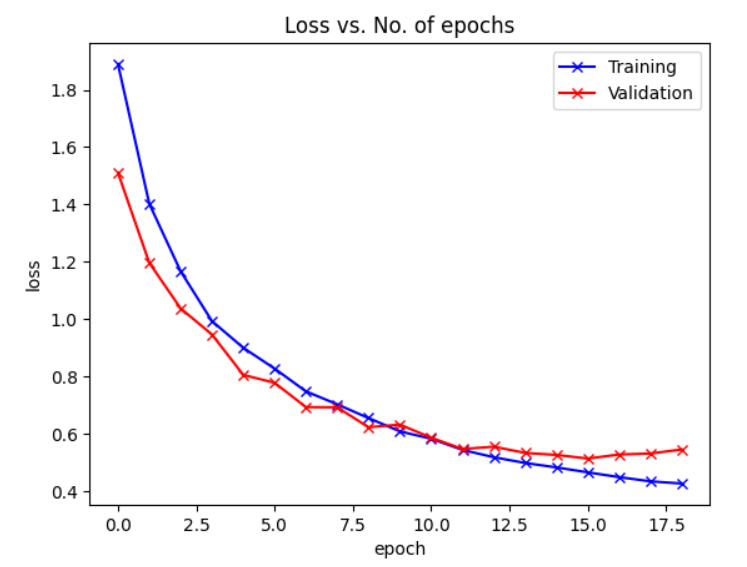
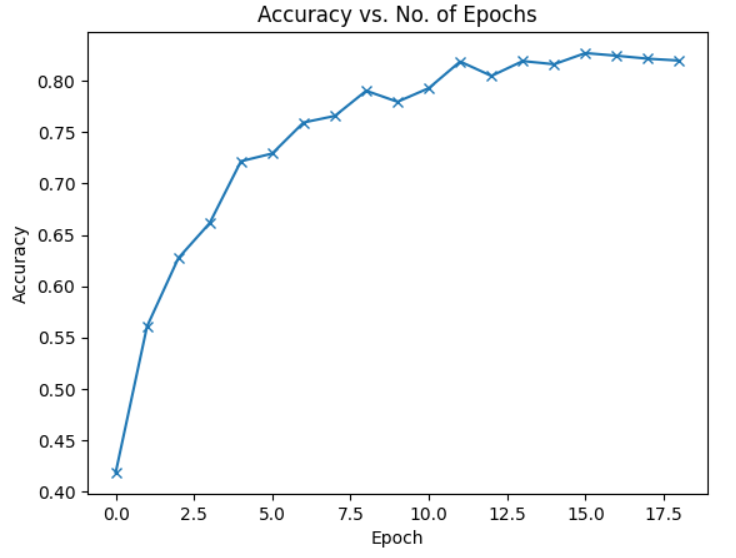
Bảng 4.2: Mô hình VGG16

|  |  |
| --- | --- |
| Layer | Input Shape -> Output Shape |
| Conv2d + ReLU | Input: [3, 224, 224] -> Output: [64, 224, 224] |
| Conv2d + ReLU | Input: [64, 224, 224] -> Output: [64, 224, 224] |
| MaxPool2d | Input: [64, 224, 224] -> Output: [64, 112, 112] |
| Conv2d + ReLU | Input: [64, 112, 112] -> Output: [128, 112, 112] |
| Conv2d + ReLU | Input: [128, 112, 112] -> Output: [128, 112, 112] |
| MaxPool2d | Input: [128, 112, 112] -> Output: [128, 56, 56] |
| Conv2d + ReLU | Input: [128, 56, 56] -> Output: [256, 56, 56] |
| Conv2d + ReLU | Input: [256, 56, 56] -> Output: [256, 56, 56] |
| Conv2d + ReLU | Input: [256, 56, 56] -> Output: [256, 56, 56] |
| MaxPool2d | Input: [256, 56, 56] -> Output: [256, 28, 28] |
| Conv2d + ReLU | Input: [256, 28, 28] -> Output: [512, 28, 28] |
| Conv2d + ReLU | Input: [512, 28, 28] -> Output: [512, 28, 28] |
| Conv2d + ReLU | Input: [512, 28, 28] -> Output: [512, 28, 28] |
| MaxPool2d | Input: [512, 28, 28] -> Output: [512, 14, 14] |
| Conv2d + ReLU | Input: [512, 14, 14] -> Output: [512, 14, 14] |
| Conv2d + ReLU | Input: [512, 14, 14] -> Output: [512, 14, 14] |
| Conv2d + ReLU | Input: [512, 14, 14] -> Output: [512, 14, 14] |
| MaxPool2d | Input: [512, 14, 14] -> Output: [512, 7, 7] |
| Linear + ReLU | Input: [512\*7\*7] -> Output: [4096] |
| DropOut | No shape change |
| Linear + ReLU | Input: [4096] -> Output: [4096] |
| DropOut | No shape change |
| Linear | Input: [4096] -> Output: [10] |

## Kết quả sau của hai mô hình

### Kết quả mô hình CNN truyền thống

Nhìn vào biểu đồ thể hiện Accuracy qua từng Epochs ta nhận thấy rằng accuracy đã được tăng dần theo từng epochs những đến epochs 19 thì lại có xu hướng giảm, và ở đây Early stopping đã dừng việc train lại khi không thấy mô hình có dấu hiệu tốt lên.



Hình 4.1: Accuracy và loss funciton model CNN

Để có cái nhìn tổng quan hơn ta có thể xem độ đo cho loss và accuracy trên từng tệp dữ liệu ở bảng 4.2 dưới đây

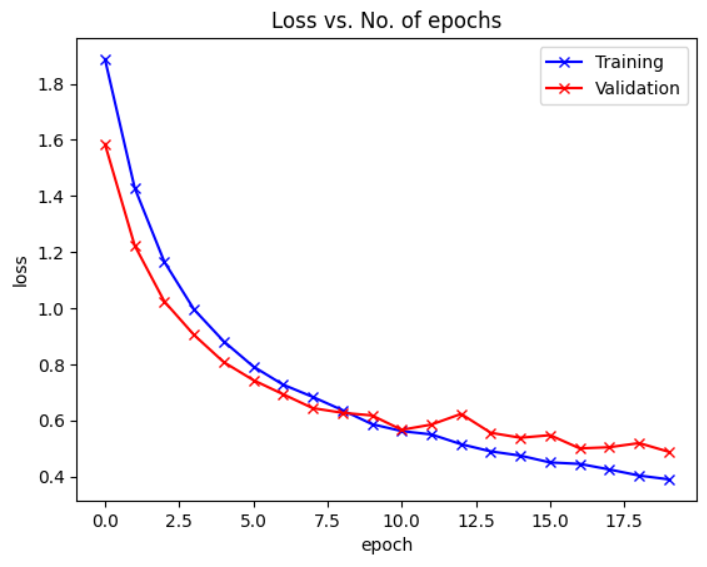
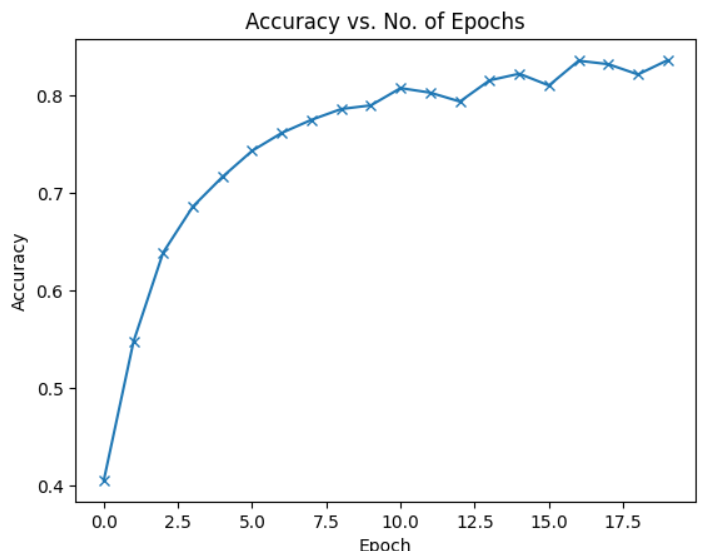
Bảng 4.3: Tổng hợp độ đo trên mô hình CNN

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Train | 0.4031 | 0.8626 |
| Validation | 0.5029 | 0.8373 |
| Test | 0.4871 | 0.8428 |

### Kết quả mô hình VGG16

Cũng tương tự như mô hình CNN, những điều đặc biệt là xu hướng của mô hình đang tốt hơn trông thấy. Về accuracy thì đang có xu hương tang dần sau mỗi epochs, loss của train và validiton thì cũng đang có xu hướng giảm dần qua từng epochs.

Ở biều đồ loss funcition cũng thể hiện rằng model đang học tốt và không có dấu hiệu bị overfitting



Hình 4.2: Accuracy và loss funcition model VGG16

Để có cái nhìn tổng quan hơn ta cũng có thể xem độ đo cho loss và accuracy cho từng tệp dữ liệu ở bảng 4.3 dưới đây

Bảng 4.4: Tổng hợp độ đo trên mô hình VGG16

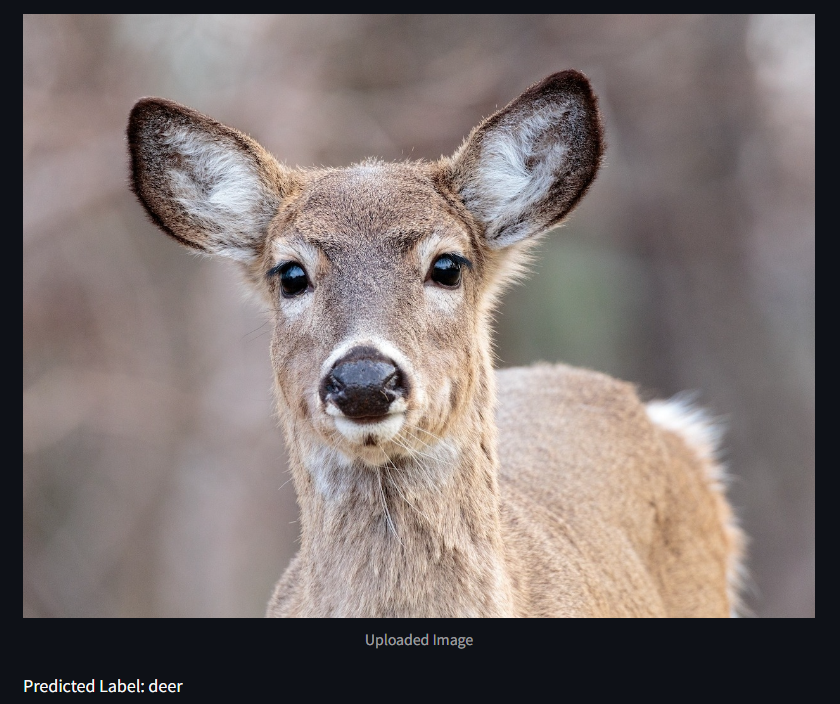
|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Train | 0.26 | 0.85 |
| Validation | 0.35 | 0.87 |
| Test | 0.35 | 0.88 |

Đối với mô hình VGG16 thì mô hình cũng đã cải thiện độ chính xác, có lẽ mô hình sẽ tăng thêm độ chính xác nếu chúng ta tăng số lượng epochs lên. Cùng với đó là loss funcition cũng sẽ giảm bởi xu hưởng của mô hình đang có là tăng độ chính xác và giảm loss.

## Demo chương trình bằng streamlit

Chúng ta cũng có thể xây dựng demo đơn giản với streamlit. Các bước thực hiện như sau:

* B1: Tải công cụ tên là "Streamlit" để xây dựng giao diện đơn giản cho ứng dụng.
* B2: Sử dụng mô hình đã huấn luyện
* B3: Xây dựng giao diện người dung
* B4: Chạy ứng dụng
* B5: Tải hình ảnh và nhận kết quả



Hình 4.3: Demo đơn giản bằng streamlit

# KẾT LUẬN

* **Đã đạt được**

- Độ chính xác cao: Mô hình CNN đã đạt được độ chính xác cao trong việc phân loại 10 đối tượng khác nhau, nhờ vào khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh.

- Hiểu được cách thức hoạt động của mạng neural network, hiểu được cách thức hoạt động của các lớp convolutional, các hàm kích hoạt như ReLU, Sigmoid

- Biết thêm về các phương thức tăng cường ảnh.

- Khả năng tổng quát: Các mô hình CNN có khả năng tổng quát tốt khi được huấn luyện trên tập dữ liệu lớn và đa dạng, cho phép phân loại các đối tượng đã học trong điều kiện ánh sáng và môi trường khác nhau.

* **Mặt hạn chế**

- Yêu cầu dữ liệu lớn: Mô hình CNN thường yêu cầu một lượng lớn dữ liệu để huấn luyện hiệu quả. Việc thu thập và dán nhãn cho đối tượng có thể gặp khó khăn.

- Khả năng chống chịu kém: Mô hình CNN có thể gặp khó khăn trong việc phân loại đối tượng trong các điều kiện khó khăn như mưa, sương mù, hoặc ánh sáng kém, dẫn đến tỷ lệ nhận diện thấp hơn.

- Overfitting: Nếu không được điều chỉnh đúng cách, mô hình CNN có thể bị overfitting, tức là học quá kỹ các đặc trưng của dữ liệu huấn luyện, làm giảm khả năng tổng quát khi áp dụng trên dữ liệu mới.

- Kiến thức và công nghệ: Chưa thể đưa mô hình tích hợp vào cammera để thử nghiệm trong thực tế.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1]. | LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. |
| [2].  [3]. | Ho Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.  Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. arXiv preprint arXiv:1801.04381. |