Exercise 10 - Convolutional Neural Network

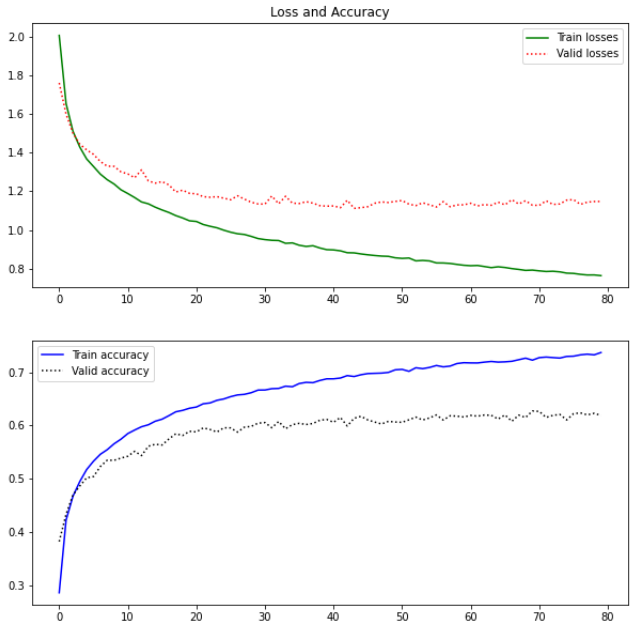
Đặng Linh Anh

January 2021

3. (Optional) Dựa vào file "2.FashionMNIST-CNN-onlyConv.ipynb", áp dụng CNN cho Cifar10 data

Lời giải

- Sau khi tuning các giá trị `kernel_size, strides, filters` cùng với việc sử dụng các initializer, activation và optimizer tốt nhất lần lượt là He Normal, Relu, Adam, ta được model với hơn 20,000 tham số, kết quả của quá trình train được thể hiện như sau:

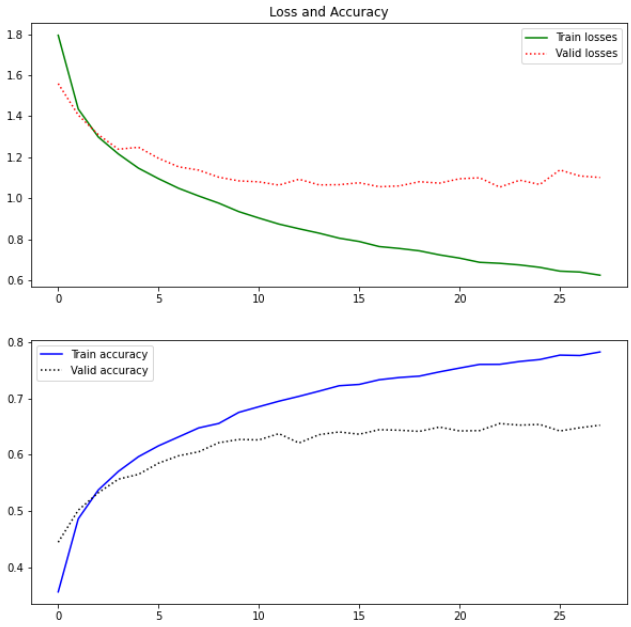


Hình 1: Huấn luyện model sử dụng mạng CNN trên dataset Cifar-10

Accuracy cho tập test: 61.87%

Link Notebook: CIFAR10

- Sử dụng layer concatenate với input sau 2 layers đầu, sau đó lặp lại kiến trúc của model đầu, kết quả nhận được trong quá trình train:



Hình 2: Huấn luyện model sử dụng mạng CNN trên dataset Cifar-10 có sử dụng concatenation

Ta thấy, accuracy của model được cải thiện: 64.15%

Link Notebook: CIFAR10 USING CONCATENATION

4. (Optional) Các bạn tìm hiểu sơ qua các cách lấy feature trong computer vision như sobel, local binary pattern; để thấy được ý nghĩa của các filter trong CNN.

i. **Lọc Sobel**: Lọc Sobel là cách tính xấp xỉ đạo hàm bậc nhất theo các hướng x và y, thường được dùng để xác định biên của ảnh.

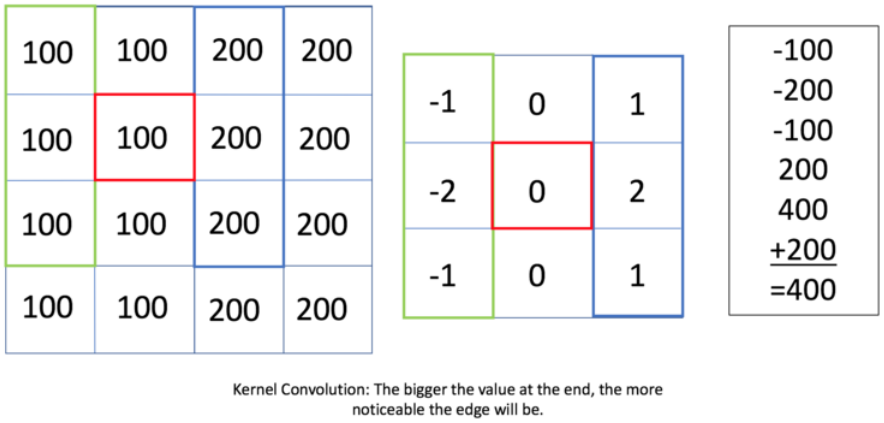
$$dst = \Delta src = \frac{\partial src}{\partial x} + \frac{\partial src}{\partial y}$$

Ma trận lọc theo hướng x và y lần lượt là:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Ví dụ dưới đây minh họa cách lọc Sobel xác định biên của ảnh.



Hình 3: Minh họa bộ lọc Sobel

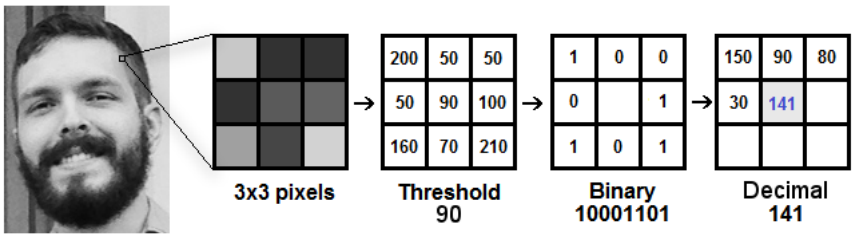
Mở rộng cho các filters trong CNN: Theo cách hiểu đơn giản, model sau các quá trình train sẽ học và thay đổi các weights một cách độc lập để đạt được loss nhỏ dần. Tuy nhiên, các weight trong từng filter luôn thay đổi theo cách nào đó để phép convolution trích xuất được đặc trưng có ích cho bài toán. Lấy ví dụ cho bài toán detect khuôn mặt, ở những layer đầu tiên, việc của các filter là trích xuất ra các đặc trưng cơ bản từ đường biên (lines) đến góc (corners). Tiếp theo đó, feature map đó sẽ cho phép mô hình học bằng việc trích xuất ra những đặc trưng phức tạp hơn như mắt, mũi, miệng, tai,...Để detect được khuôn mặt, mô hình cần xác định được các vị trí tương đối của các bộ phận đã nhận diện được trước đó với nhau, do đó, các filter sau đó sẽ làm nhiệm vụ kết nối những đặc trưng đó.

ii. **Local Binary Pattern**: Local Binary Pattern (LPB) là phương pháp trích xuất đặc trưng liên quan đến việc hình thành số nhị phân từ cách so sánh các giá trị neighbor với giá trị ngưỡng (giá trị trung tâm). LBP có 4 tham số:

- Bán kính (Radius): là bán kính giới hạn xét đến xung quanh cell trung tâm, được dùng trong xây dựng LBP dạng đường tròn. Thường được cài đặt là 1.
- Neighbors: Số lượng cell trong đường tròn giới hạn, tham gia tính toán cho LBP. Càng nhiều số lượng neighbors, phép tính càng nặng và phức tạp. Thường được cài đặt là 8.
- Kích thước X: Số lượng cell xét theo trục X. Thường được cài đặt là 8.
- Kích thước Y: Số lượng cell xét theo trục Y. Thường được cài đặt là 8.

Các bước tính toán LBP:

- Trong một window 3×3, so sánh các giá trị pixel neighbors với pixel trung tâm (giá trị ngưỡng, $v_{center} = 90$). Ta có được ma trận với các giá trị boolean.
- Nối các giá trị neighbors thành một số nhị phân (10001101)
- Chyển số nhị phân này sang dạng thập phân, gán giá trị này vào pixel trung tâm, ta được ma trận output của phương pháp LBP.



Hình 4: Minh họa phương pháp LBP trong bài toán nhận diện khuôn mặt

Nhận xét:

Phương pháp này lấy được giá trị đặc trưng rất tốt cho mỗ window của ảnh. Mỗi window với các distribution khác nhau sẽ cho ra số nhị phân khác nhau, từ đó tạo ra giá trị pixel trung tâm khác nhau. Ở các bài toán nhận diện, trước khi CNN ra đời, LBP được kết hợp với các thuật toán machine learning cổ điển đã cho ra kết quả khá tốt (thông thường được train trên các histogram của ảnh).

5. (Optional) Các bạn suy nghĩ ý nghĩa của việc dùng filter (1x1).

Lời giải

Với kernel có kích thước $1 \times 1 \times n_{channels}$, khi thực hiện convolution, các giá trị pixel ở các channels được cộng lại với nha. Vì vậy, co thể nói, phép 1×1 convolution tính toán trên từng pixel của ảnh, không tương tác được với các neighbors. Tuy nhiên, việc này cho phép các channels được tương tác với nhau, thường được dùng để giảm chiều sâu của feature map đồng thời tăng thời gian tính toán.

Ví dụ so sánh 2 kiến trúc:

```
input (256 depth) -> 1x1 convolution (64 depth) -> 4x4 convolution (256 depth)
```

```
input (256 depth) -> 4x4 convolution (256 depth)
```

Từ thực nghiệm ta nhận thấy, kiến trúc dưới chậm hơn đáng kể so với kiến trúc trên. Layer `1x1 conv` còn được gọi là layer bottleneck, cho phép giảm chiều sâu đầu vào để tối ưu tính toán cho layer sau (`4x4 conv`), đồng thời trích xuất được một lượng feature đáng kể dọc theo chiều sâu của feature map.