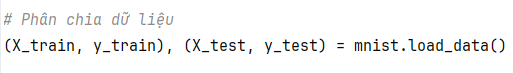
**BÀI TẬP 5**

**XÂY DỰNG MÔ HÌNH PHÂN LỚP**

**TẬP DỮ LIỆU ẢNH CHỮ SỐ VIẾT TAY MNIST**

**--------**

1. **Mô hình không có rút trích đặc trưng:**
   1. **Phân chia dữ liệu:**



**Hình. Phân chia dữ liệu**

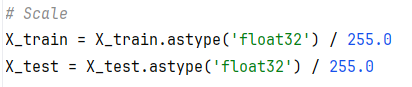
- Thư viện *minist* chứa toàn bộ dữ liệu MINIST, gồm hình ảnh của các chữ số viết tay từ 0 đến 9 với 60,000 hình ảnh huấn luyện và 10,000 hình ảnh kiểm tra.

- Hàm **load\_data()** trả về 2 tuple lần lượt là:

+ (X\_train, y\_train): Chứa hình ảnh của các chữ số viết tay và nhãn tương ứng trong tập huấn luyện.

+ (X\_test, y\_test): Chứa hình ảnh của các chữ số viết tay và nhãn tương ứng trong tập kiểm tra.

* 1. **Chuẩn hóa dữ liệu:**



**Hình. Chuẩn hóa dữ liệu**

Do dữ liệu hình ảnh ban đầu là các giá trị nguyên từ 0 đến 255, các giá trị này sẽ được chuẩn hóa về khoảng từ 0 đến 1 bằng cách chia cho 255 để giúp mô hình huấn luyện hiệu quả hơn.

* 1. **Thêm chiều kênh màu:**



**Hình. Thêm chiều kênh màu**

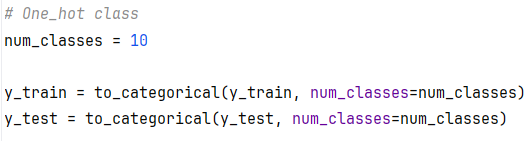
Bộ dữ liệu MNIST chỉ có 1 kênh màu, nhưng yêu cầu đầu vào của CNN thường là các ảnh nhiều kênh. Hàm **expand\_dims()** cho phép thêm một chiều mới vào cuối mảng.

Kết quả:

|  |  |
| --- | --- |
| Trước khi thêm | Sau khi thêm |

**Bảng. Kết quả trước và sau khi thêm kênh màu**

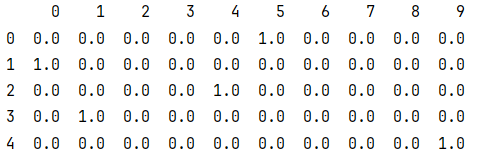
* 1. **One-hot encoding cột nhãn:**



**Hình. Mã hóa cột nhãn**

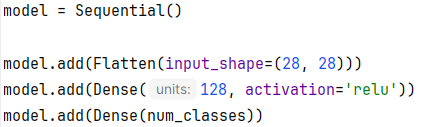
Hàm **to\_categorical()** từ thư viện *keras* giúp chuyển các nhãn từ dạng số nguyên thành dạng one\_hot encoding.

Kết quả:



**Hình. 5 hàng đầu tiên của y\_train**

* 1. **Xây dựng mô hình:**



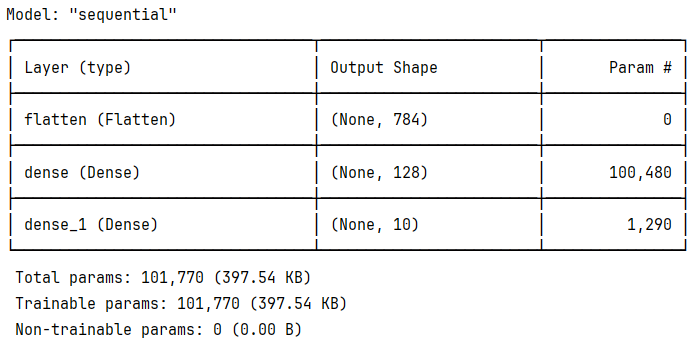
**Hình. Mô hình mạng đơn giản**

- Mô hình được khởi tạo theo dạng “tuần tự” (Sequential). Mô hình này bao gồm các tầng xếp chồng lên nhau theo thứ tự.

- Tầng *Flatten* giúp chuyển đổi các ảnh đầu vào kích thước 28x28 thành vector 1D.

- Tầng *Dense* đầu tiên là tầng kết nối đầy đủ với 128 nơ ron và sử dụng hàm kích hoạt “ReLU”. Tầng *Dense* thứ hai, cũng là tầng cuối cùng của mạng, là tầng đầu ra của mô hình với số lượng lớp là 10 (0 9).

- Dữ liệu được đưa vào mô hình này không trải qua bước rút trích đặc trưng mà trực tiếp đưa vào huấn luyện.



**Hình. Kiến trúc mô hình**

- Các hình ảnh có kích thước 28x28 được đưa vào tầng *flatten* sẽ được làm phẳng thành vecto có kích thước 28 \* 28 = 784. Tầng này không có tham số học do nó chỉ thực hiện chuyển đổi dữ liệu đầu vào mà không có thực hiện phép tính toán nào liên quan đến trọng số hay bias.

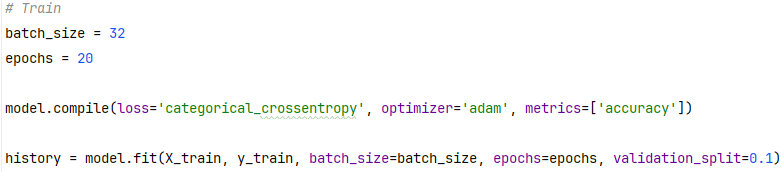
- Tầng *dense* nhận dữ liệu từ tầng trước (flatten) với kích thước là 784 sẽ thực hiện các tính toán và cho ra vecto có kích thước 128 (do tầng này có 128 nơ ron). Do có tính toán nên tầng này có số lượng tham số như sau:

Số lượng tham số = (Số lượng đầu vào + 1) \* số nơ ron = (784 + 1) \* 128 = 100,480.

- Tầng *dense\_1* cũng nhận dữ liệu từ tầng *dense* với kích thước là 128 sẽ cho vecto có kích thước là 10 (MNIST có 10 lớp). Số lượng tham số = (128 + 1) \* 10 = 1,290.

- Tổng tham số của mô hình là: 100,480 + 1,290 = 101,770.

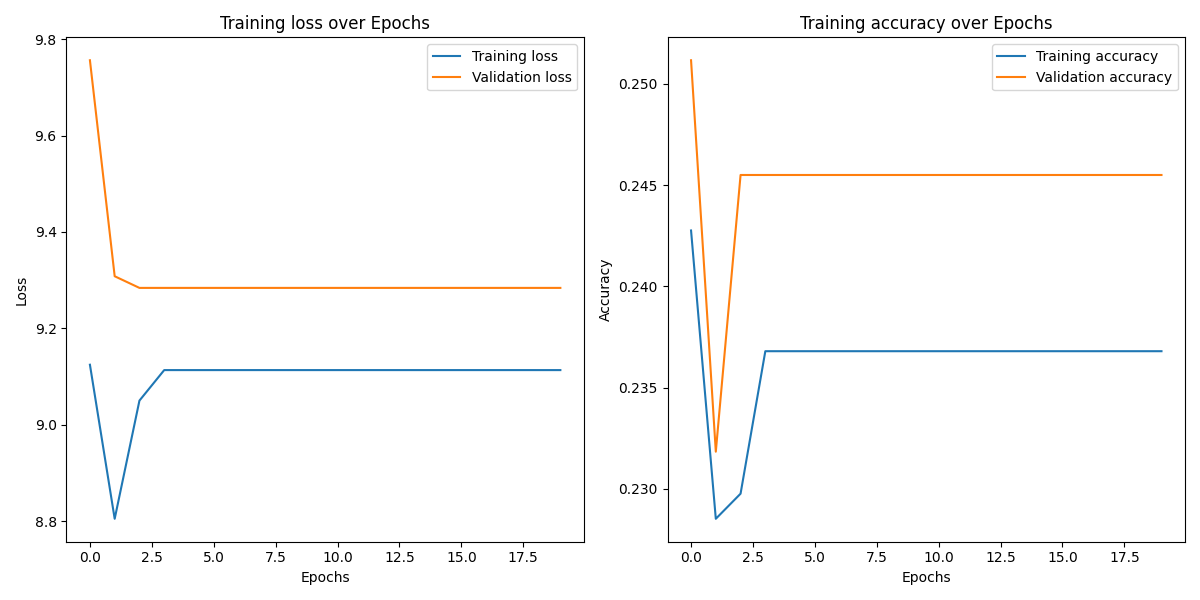
* 1. **Huấn luyện mô hình:**



**Hình. Huấn luyện mô hình**

Mô hình trước khi huấn luyện sẽ được biên dịch để bắt đầu quá trình học bằng cách kết nối mô hình với các hàm và thuật toán cần thiết để thực hiện cập nhật trọng số.

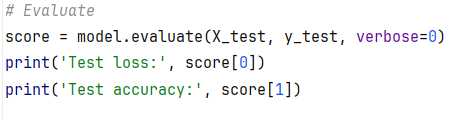
Mô hình sau đó sẽ được huấn luyện qua 20 epochs với mỗi lần cập nhật trọng số sẽ được thực hiện trên 32 mẫu dữ liệu. Trong quá trình này, 10% dữ liệu huấn luyện (validation\_split=0.1) sẽ được sử dụng để đánh giá hiệu quả mô hình khi huấn luyện.



**Hình. Kết quả huấn luyện**

Ở biểu đồ “Training loss over Epochs”, đường biểu diễn loss gần như phẳng sau vài epoch đầu tiên, cho thấy mô hình không cải thiện qua các epoch. Còn ở biểu đồ “Traning accuracy over Epochs”, tương tự biểu đồ trước, độ chính xác gần như không đổi sau các epoch đầu. Hai biểu đồ trên cho thấy mô hình có hiệu suất kém, cần cải thiện thêm như thêm các lớp tích chập để học đặc trưng cục bộ từ hình ảnh.

* 1. **Đánh giá mô hình:**



**Hình. Đánh giá mô hình**

Mô hình được đánh giá trên tập kiểm tra với kết quả như sau:



**Hình. Kết quả đánh giá**

Kết quả cho thấy độ mất mát trên tập dữ liệu là khá cao và độ chính xác là khoảng 23%. Điều này thể hiện mô hình có dấu hiệu overfitting do kiến trúc mô hình quá đơn giản, không đủ khả năng học các đặc trưng của dữ liệu.

1. **Mô hình có rút trích đặc trưng (CNN):**
   1. **Chuẩn bị dữ liệu:**

Các bước chuẩn bị dữ liệu được thực hiện theo các bước tương tự như ở phần 1, bao gồm việc tải dữ liệu MNIST, chuyển đổi dữ liệu sang định dạng phù hợp, mã hóa nhãn và chia dữ liệu thành tập đào tạo và kiểm tra. Với dữ liệu vừa chuẩn bị, tiến hành xây dựng mô hình CNN để phân loại. Mô hình này sẽ được thiết kế để tận dụng khả năng của mạng nơ-ron tích chập trong việc trích xuất đặc trưng từ hình ảnh và phân loại chúng thành các lớp khác nhau, đồng thời đạt được độ chính xác cao trong việc phân loại hình ảnh.

* 1. **Xây dựng mô hình:**



**Hình. Mô hình mạng CNN**

- Dữ liệu đầu vào được đưa về kích thước 28x28 với một kênh màu (ảnh đen trắng).

- Mô hình được khởi tạo bằng hàm **Sequential()**, bao gồm các tầng xếp chồng tuần tự lên nhau.

- Thêm tầng *Input* đầu tiên vào mô hình, yêu cầu dữ liệu đầu vào phải có kích thước 28x28x1.

- Conv2D(32, kernel\_size=(3, 3), activation=‘relu’): Dòng này thêm lớp tích chập đầu tiên vào mô hình với 32 bộ lọc (filter). Mỗi bộ lọc có kích thước 3x3. Hàm kích hoạt được sử dụng là hàm ReLU.

- MaxPooling2D(pool\_size=(2, 2)): Thêm một tầng pooling với kích thước 2x2. Tầng này giúp giảm kích thước của đặc trưng bằng cách chọn giá trị lớn nhất trong mỗi vùng 2x2, từ đó giảm số lượng tham số và tính toán.

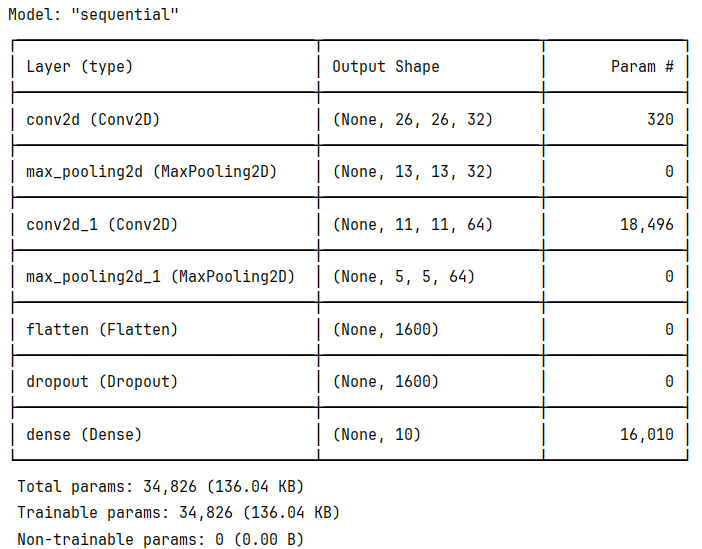
- Conv2D(64, kernel\_size=(3, 3), actionvation=‘relu’): Thêm tầng tích chập thứ hai vào mô hình với 64 bộ lọc, kích thước cũng là 3x3, sử dụng hàm kích hoạt ReLU.

- MaxPooling2D(pool\_size=(2, 2)): Thêm một tầng pooling với kích thước 2x2, tương tự tầng MaxPooling ở trên.

- Flatten(): Tầng này giúp chuyển đổi dữ liệu từ hình ảnh thành vecto để đưa vào tầng Dense.

- Dropout(0.5): Tầng này giúp bỏ qua một số nơ-ron trong quá trình huấn luyện để giảm thiểu hiện tượng overfitting.

- Dense(num\_classes, activation=‘softmax’): Tầng cuối cùng của mô hình luôn là tầng Dense có số nơ-ron bằng số lớp đầu ra của bài toán (num\_classes=10). Tầng này sử dụng hàm *softmax* để chuyển đổi các giá trị đầu ra thành xác suất cho từng lớp.



**Hình. Kiến trúc mô hình**

- Từ dữ liệu đầu vào có kích thước 28x28x1, khi đi qua tầng tích chập đầu tiên Conv2D(32, kernel\_size=(3, 3), activation=‘relu’), dữ liệu sau quá trình rút trích đặc trưng sẽ có kích thước:

Kích thước đầu ra sẽ là (output\_size, output\_size, batch\_size) = (26, 26, 32).

Tham số của tầng này được tính là:

Params = (kernel\_width \* kernel\_height \* input\_channel) \* batch\_size

= (3 \* 3 \* 1) \* 32 = 320.

- Tầng MaxPooling(pool\_size=(2, 2)) có nhiệm vụ giảm chiều dữ liệu, cụ thể:

Tầng này chỉ đơn giản là giảm chiều dữ liệu, nghĩa là không có sự tính toán trọng số và bias, nên sẽ không có tham số. Params = 0.

- Tầng Conv2D(64, kernel\_size=(3, 3), activation=‘relu’) nhận đặc trưng đầu vào có kích thước (26, 26, 32), sau quá trình rút trích đặc trưng, dữ liệu đầu ra của tầng này dựa vào công thức tính toán trên có kích thước là (11, 11, 64). Tham số của tầng này dựa trên công thức tính tham số phía trên là 18,496.

- Tầng MaxPooling(pool\_size=(2, 2)) đưa dữ liệu đầu vào kích thước (11, 11, 64) thành (5, 5, 64).

- Tầng Flatten() giúp chuyển đổi đầu vào thành một vecto phẳng. Kích thước đầu vào của tầng này là 5x5x64, vậy kích thước đầu ra sẽ là 5\*5\*64 = 1600. Tầng này không cập nhật trọng số và bias nên không có tham số cần học.

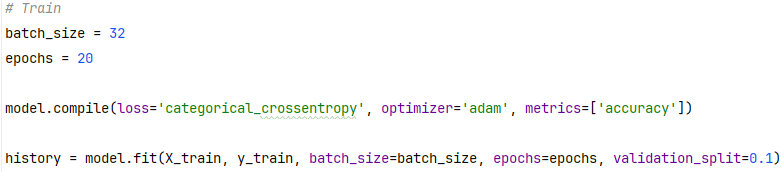
- Tầng Dropout() chỉ đơn giản giúp mô hình giảm hiện tượng overfitting nên không có thay đổi gì ở dữ liệu hay tính toán trọng số.

- Tầng Dense(10, activation=‘softmax’): Tầng này chuyển đổi dữ liệu đầu vào thành xác suất cho mỗi lớp đầu ra của bài toán, nên kích thước đầu ra của mô hình là 10. Tham số cần học được tính như sau:

Params = (kích thước đầu vào + 1) \* số lớp = (1600 + 1) \* 10 = 16,010.

- Tổng tham số cho mô hình này là: 320 + 18,496 + 16,010 = 34,826.

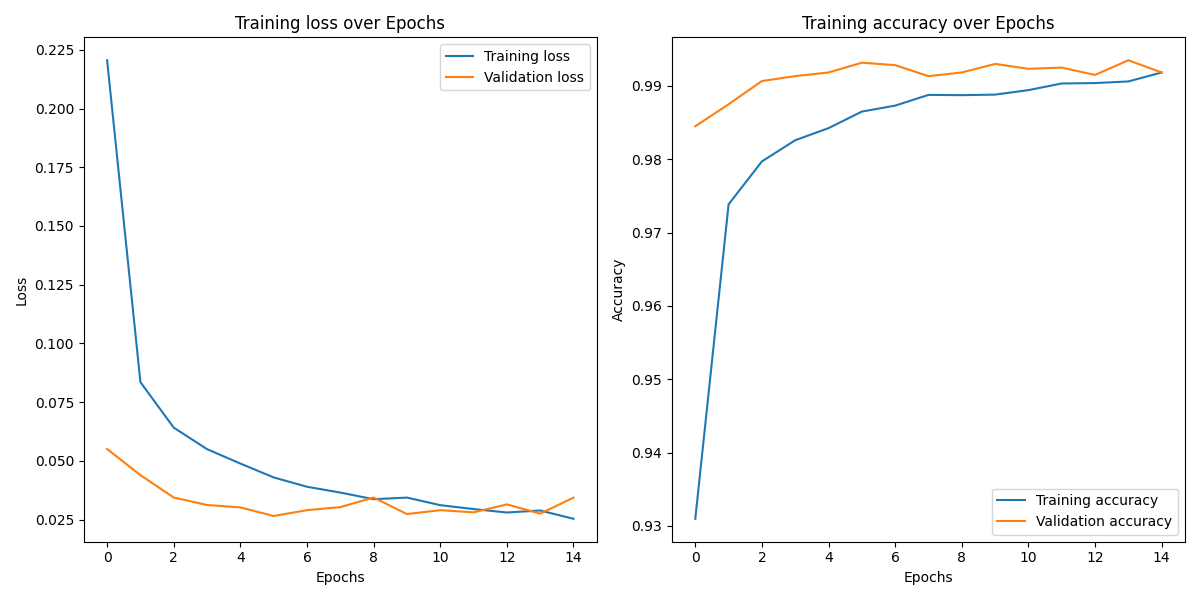
* 1. **Huấn luyện mô hình:**



**Hình. Huấn luyện mô hình**

Tương tự mô hình phần 1, trước khi huấn luyện, mô hình sẽ được biên dịch để bắt đầu quá trình học bằng cách kết nối mô hình với các hàm và thuật toán cần thiết để thực hiện cập nhật trọng số.

Mô hình sau đó sẽ được huấn luyện qua 20 epochs với mỗi lần cập nhật trọng số sẽ được thực hiện trên 32 mẫu dữ liệu. Trong quá trình này, 10% dữ liệu huấn luyện (validation\_split=0.1) sẽ được sử dụng để đánh giá hiệu quả mô hình khi huấn luyện.



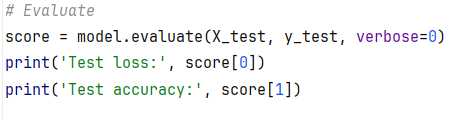
**Hình. Kết quả huấn luyện**

Ở biểu đồ “Training loss over Epochs”, *Training loss* (đường màu xanh) giảm mạnh và dần tiến về 0, thể hiện mô hình học tốt trên tập huấn luyện. Còn *Validation loss* (đường màu cam) cũng giảm mạnh ở những epoch đầu nhưng sau đó giữ ở mức khá ổn định, điều này có thể giải thích rằng mô hình bắt đầu overfitting nhẹ.

Ở biểu đồ “Training accuracy over Epochs”, *Training accuracy* (đường màu xanh) tăng nhanh và đạt gần như tuyệt đối, nghĩa là mô hình gần như học được quy luật từ dữ liệu huấn luyện hiệu quả. Còn *Validation accuracy* (đường màu cam) cũng tăng nhanh và duy trì ở mức khá cao.

Biểu đồ này cho thấy mô hình đạt hiệu suất cao hơn khi dữ liệu trải qua quy trình rút trích đặc trưng của hình ảnh, giúp mô hình học được các đặc trưng từ mỗi hình ảnh tốt hơn. Do đó, kết quả ở mô hình này (CNN) sẽ tốt hơn mô hình không trải qua rút trích đặc trưng mà trực tiếp đưa vào huấn luyện.

* 1. **Đánh giá mô hình:**



**Hình. Đánh giá mô hình**

Mô hình được đánh giá trên tập kiểm tra với kết quả như sau:



**Hình. Kết quả đánh giá**

Kết quả cho thấy mô hình có khả năng dự đoán tốt và độ lệch nhỏ giữa các giá trị dự đoán và thực tế trên tập kiểm tra. Độ chính xác 99,18% là một kết quả khá ấn tượng, chứng tỏ mô hình có hiệu suất tốt trên dữ liệu chưa từng thấy.

**---HẾT---**