TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue square with white letters

Description automatically generated

**BÁO CÁO ĐỀ TÀI**

**Môn Thị giác máy tính**

Đề tài : Mô hình CNN đơn giản với TensorFlow/Keras

ứng dụng vào phân loại sản phẩm thời trang.

Giảng viên hướng dẫn: ThS. Nguyễn Thị Nguyệt

Lớp: 21CĐP2

Sinh viên thực hiện

|  |  |
| --- | --- |
| Đàm Minh Khuê | 2155020046 |
| Đinh Tuệ Minh | 2155020062 |
| Nguyễn Lê Cẩm Nhung | 2155020072 |
| Nguyễn Thị Thanh Tâm | 2155020082 |

Hà Nội, 03/2025

**Mục lục**

[Tổng quan đề tài 3](#_Toc193962610)

[1. Đặt vấn đề 3](#_Toc193962611)

[2. Mục tiêu nghiên cứu 4](#_Toc193962612)

[3. Phạm vi nghiên cứu 4](#_Toc193962613)

[4. Cơ sở lý thuyết 5](#_Toc193962614)

[4.1. Tổng quan về Computer Vision 5](#_Toc193962615)

[4.2. Mô hình CNN (Convolutional Neural Network) 7](#_Toc193962616)

[4.3. TensorFlow và Keras 15](#_Toc193962617)

[4.4. Tập dữ liệu Fashion-MNIST 18](#_Toc193962618)

[Xây dựng mô hình 21](#_Toc193962619)

[Kết quả thực nghiệm 33](#_Toc193962620)

[1. Kết quả huấn luyện 33](#_Toc193962621)

[2. Cải thiện mô hình 39](#_Toc193962622)

[3. So sánh với các mô hình khác 46](#_Toc193962623)

[Đánh giá và kết luận 48](#_Toc193962624)

[1. Đánh giá kết quả 48](#_Toc193962625)

[3. Hướng khắc phục và phát triển trong tương lai 50](#_Toc193962626)

[Tài liệu tham khảo 52](#_Toc193962627)

# Tổng quan đề tài

## Đặt vấn đề

Trong thời đại công nghệ số phát triển mạnh mẽ, việc xử lý và phân tích hình ảnh đã trở thành một lĩnh vực nghiên cứu quan trọng, đặc biệt trong các ứng dụng thực tế như thương mại điện tử, an ninh, y tế và giải trí. Việc tự động nhận diện và phân loại hình ảnh không chỉ giúp tiết kiệm thời gian mà còn nâng cao độ chính xác trong quá trình ra quyết định. Học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Network - CNN), đã chứng minh được hiệu quả vượt trội trong các bài toán nhận diện và phân loại hình ảnh.

Fashion-MNIST là một tập dữ liệu gồm 70,000 hình ảnh sản phẩm thời trang, được phát triển nhằm thay thế tập dữ liệu MNIST truyền thống, cung cấp một bài toán phân loại hình ảnh có độ phức tạp cao hơn. Các sản phẩm trong tập dữ liệu này không chỉ đa dạng về kiểu dáng mà còn có sự khác biệt về chất liệu, phong cách, tạo ra những thách thức mới trong việc phân loại chính xác.

Cùng với sự bùng nổ của thương mại điện tử, nhu cầu về các hệ thống tự động hóa để phân loại và gợi ý sản phẩm cho người tiêu dùng ngày càng gia tăng. Một mô hình phân loại chính xác và hiệu quả có thể giúp các doanh nghiệp tối ưu hóa quy trình vận hành, cải thiện trải nghiệm khách hàng và nâng cao năng suất. Vì vậy, nghiên cứu này tập trung vào việc phát triển một mô hình CNN đơn giản nhưng hiệu quả nhằm phân loại hình ảnh sản phẩm thời trang, với mục tiêu không chỉ đạt độ chính xác cao mà còn có tiềm năng ứng dụng thực tế.

## Mục tiêu nghiên cứu

* Phát triển mô hình CNN: Nghiên cứu sẽ thiết kế và triển khai một mô hình CNN với kiến trúc hợp lý nhằm tối ưu hóa khả năng phân loại hình ảnh. Chúng tôi sẽ điều chỉnh các tham số như số lượng lớp, kích thước bộ lọc, và kỹ thuật tăng cường dữ liệu để cải thiện hiệu suất mô hình.
* Đánh giá hiệu suất mô hình: Một phần quan trọng của nghiên cứu là đánh giá hiệu suất của mô hình đã phát triển. Chúng tôi sẽ sử dụng các chỉ số như độ chính xác (accuracy), độ nhạy (sensitivity), độ đặc hiệu (specificity) và thời gian huấn luyện để đánh giá khả năng phân loại của mô hình trên tập dữ liệu kiểm tra.
* So sánh với các phương pháp khác: Nghiên cứu sẽ không chỉ dừng lại ở việc phát triển mô hình CNN mà còn thực hiện so sánh hiệu suất với các phương pháp phân loại hình ảnh khác, bao gồm các mô hình học sâu khác và các kỹ thuật truyền thống. Điều này sẽ giúp xác định ưu điểm và nhược điểm của từng phương pháp.
* Rút ra kết luận: Dựa trên các kết quả thu được, nghiên cứu sẽ rút ra các kết luận về hiệu quả của mô hình CNN trong việc phân loại hình ảnh sản phẩm thời trang. Chúng tôi cũng sẽ đưa ra các khuyến nghị cho việc cải thiện mô hình trong tương lai và ứng dụng trong thực tiễn.

## Phạm vi nghiên cứu

Tập dữ liệu: Nghiên cứu sẽ tập trung vào việc sử dụng tập dữ liệu Fashion-MNIST, bao gồm 70,000 hình ảnh sản phẩm thời trang được phân thành 10 lớp. Chúng tôi sẽ chỉ xem xét các hình ảnh trong tập huấn luyện và tập kiểm tra, không mở rộng ra các tập dữ liệu khác.

Mô hình học sâu: Nghiên cứu chủ yếu sẽ tập trung vào việc phát triển và tối ưu hóa mô hình CNN. Các kiến trúc khác như mạng nơ-ron hồi tiếp (RNN) hay mạng nơ-ron đối kháng (GAN) sẽ không được xem xét trong nghiên cứu này, nhằm đảm bảo sự tập trung vào việc cải thiện kỹ thuật phân loại hình ảnh.

Giới hạn của nghiên cứu: Nghiên cứu sẽ không xem xét các yếu tố như ảnh hưởng của các yếu tố bên ngoài đến sự phân loại, chẳng hạn như ánh sáng, góc chụp, và chất lượng hình ảnh. Chúng tôi sẽ chỉ tập trung vào việc phát triển một mô hình CNN hiệu quả trên tập dữ liệu đã cho mà không tiến hành thu thập dữ liệu mới.

## Cơ sở lý thuyết

### Tổng quan về Computer Vision

Computer Vision (Thị giác máy tính) là một lĩnh vực con của trí tuệ nhân tạo (AI) tập trung vào việc cho phép máy tính và hệ thống phân tích, hiểu và xử lý hình ảnh và video. Mục tiêu chính của Computer Vision là mô phỏng khả năng thị giác của con người, giúp máy tính nhận diện và phân loại các đối tượng trong hình ảnh, từ đó thực hiện các nhiệm vụ khác nhau như phân tích, nhận diện và theo dõi.

Lịch sử và phát triển:

* Khởi đầu: Computer Vision bắt đầu từ những năm 1960 với các nghiên cứu cơ bản về nhận diện hình ảnh. Những công trình đầu tiên thường chỉ xử lý các hình ảnh đơn giản và có độ phân giải thấp.
* Tăng trưởng nhanh chóng: Từ đầu thế kỷ 21, với sự phát triển của công nghệ máy tính và lượng dữ liệu lớn (big data), Computer Vision đã đạt được nhiều thành tựu đáng kể. Sự xuất hiện của mạng nơ-ron sâu (Deep Neural Networks) đã cách mạng hóa lĩnh vực này.

Computer Vision tập trung nghiên cứu nhằm phục vụ các mục đích như:

* Nhận diện hình ảnh (Image Recognition): xác định các đối tượng có trong một bức ảnh hoặc video, phân loại hình ảnh thành các nhóm ví dụ như “hình ảnh có động vật” hay “hình ảnh có người”...
* Xử lý ảnh (Image Processing): thay đổi kích thước, chuyển đổi màu sắc, lọc nhiễu, làm mịn… hoặc biến đổi không gian như thay đổi góc độ, phóng to thu nhỏ ảnh hoặc thay đổi vị trí của đối tượng trong ảnh để cải thiện khả năng nhận diện
* Nhận diện đối tượng (Object Detection): xác định sự hiện diện của đối tượng và cả vị trí của nó trong ảnh. Sử dụng các kỹ thuật như YOLO (You Only Look Once) và Faster R-CNN được ứng dụng trong nhận diện đối tượng.
* Nhận diện khuôn mặt (Face Recognition): là quá trình nhận diện khuôn mặt người trong ảnh hoặc video. Các phương pháp thường được sử dụng bao gồm Eigenfaces, Fisherfaces và Deep Learning (DNN-based)...
* Phân vùng hình ảnh (Segmentation): là quá trình chia một hình ảnh thành các phần nhỏ dựa trên các đặc điểm giống nhau như màu sắc hoặc kết cấu… Có 2 loại phân vùng chính đó là Semantic Segmentation và Instance Segmentation.
* Semantic Segmentation: Phân loại mỗi pixel trong hình ảnh thuộc về một lớp cụ thể, chẳng hạn như "chó", "mèo", "bầu trời",...
* Instance Segmentation: Ngoài phân loại mỗi pixel, còn cần phân biệt các đối tượng giống nhau trong hình ảnh, ví dụ như phân biệt giữa các con chó khác nhau trong cùng một hình ảnh.

Hệ thống thị giác máy tính sử dụng công nghệ trí tuệ nhân tạo (AI) để bắt chước khả năng của não người trong việc nhận biết đối tượng và phân loại đối tượng. Các nhà khoa học máy tính đào tạo máy tính nhận biết dữ liệu hình ảnh bằng cách nhập khối lượng lớn thông tin. Thuật toán máy học (ML) xác định các kiểu mẫu thông thường trong những hình ảnh hoặc video này và áp dụng kiến thức đó để xác định chính xác những hình ảnh chưa biết. Ví dụ: nếu máy tính xử lý hàng triệu hình ảnh ô tô, chúng sẽ bắt đầu xây dựng kiểu mẫu nhận dạng và có thể phát hiện chính xác phương tiện trong một hình ảnh. Thị giác máy tính sử dụng các công nghệ như được đưa ra dưới đây.

* Deep Learning (Học sâu): sử dụng các mạng nơ-ron nhân tạo với nhiều lớp (còn gọi là mạng nơ-ron sâu). Mục tiêu của deep learning là học các biểu diễn dữ liệu phức tạp thông qua các lớp nơ-ron, giúp máy tính tự động nhận diện và phân tích các mẫu trong dữ liệu mà không cần phải lập trình chi tiết từng bước.
* CNNs (Mạng nơ-ron tích chập): Mạng nơ-ron tích chập (CNN) sử dụng hệ thống ghi nhãn để phân loại dữ liệu hình ảnh và hiểu toàn bộ hình ảnh. Mạng nơ-ron này phân tích hình ảnh dưới dạng điểm ảnh và cung cấp cho mỗi điểm ảnh một giá trị nhãn. Giá trị được nhập vào để thực hiện một phép toán gọi là phép nhân chập và đưa ra các dự đoán về hình ảnh. Giống như một người cố gắng nhận biết một đối tượng ở khoảng cách xa, CNN trước tiên sẽ xác định đường nét và hình dạng đơn giản trước khi điền vào các chi tiết bổ sung như màu sắc, hình thức bên trong và kết cấu. Cuối cùng, nó lặp lại quy trình dự đoán qua nhiều lần lặp để nâng cao độ chính xác.
* Generative Adversarial Networks (GAN): là một loại mô hình học sâu dùng để tạo ra các hình ảnh giả hoặc phân tích các đặc trưng của hình ảnh trong các ứng dụng như chỉnh sửa ảnh, tạo ảnh...

Khác với xử lý hình ảnh sử dụng các thuật toán để chỉnh sửa ảnh, bao gồm làm sắc nét, làm mịn, lọc hoặc tăng cường, thị giác máy tính không làm thay đổi hình ảnh, mà thay vào đó hiểu những gì nó nhìn thấy và thực hiện một tác vụ, chẳng hạn như gắn nhãn. Trong một số trường hợp, ta có thể sử dụng xử lý hình ảnh để sửa đổi hình ảnh sao cho hệ thống thị giác máy tính có thể hiểu rõ hơn về hình ảnh đó. Trong các trường hợp khác, thị giác máy tính được sử dụng để xác định hình ảnh hoặc các phần của một hình ảnh rồi sau đó sử dụng xử lý hình ảnh để sửa đổi hình ảnh.

Ứng dụng của Thị giác máy tính:

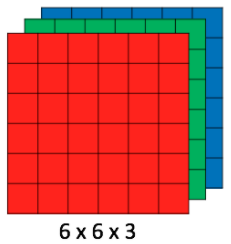
* Nhận diện khuôn mặt: Được sử dụng trong an ninh và các ứng dụng mạng xã hội để nhận diện và phân loại người dùng.
* Phân tích y tế: Giúp bác sĩ phân tích hình ảnh y khoa như MRI, CT scans để phát hiện bệnh lý.
* Xe tự lái: Cung cấp khả năng nhận diện đối tượng, đọc biển báo và theo dõi lộ trình.
* Thương mại điện tử: Cải thiện trải nghiệm người dùng thông qua nhận diện sản phẩm và phân loại hình ảnh.

### Mô hình CNN (Convolutional Neural Network)

#### Giới thiệu chung.

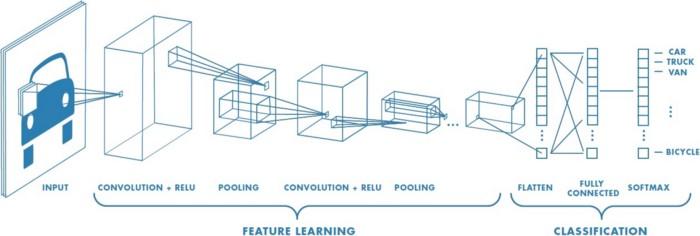
Convolutional Neural Network (CNN) là một loại mô hình mạng nơ-ron nhân tạo chính trong lĩnh vực nhận dạng hình ảnh và thị giác máy tính. CNN được phát triển dựa trên nguyên tắc hoạt động của vùng thị giác trong bộ não con người, giúp xử lý và nhận dạng các đặc trưng trong hình ảnh.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).



Hình 1: Mảng ma trận RGB 6x6x3

Về kỹ thuật, mô hình CNN để huấn luyện và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.



Hình 2: Luồng CNN xử lý hình ảnh hoàn chính

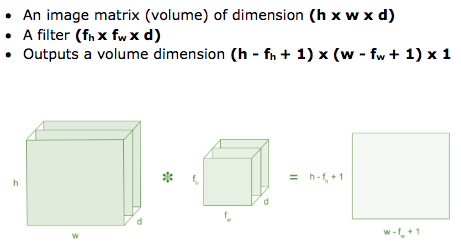
Mô hình CNN được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

* Nhận dạng hình ảnh: Phân loại và nhận dạng khuôn mặt, biển số xe, chữ viết tay.
* Thị giác máy tính: Phát hiện vật thể, theo dõi đối tượng trong video.
* Y tế: Phân tích hình ảnh y khoa, chẩn đoán bệnh từ ảnh X-ray, MRI.
* Xe tự hành: Nhận diện làn đường, biển báo giao thông.
* Xử lý hình ảnh: Deepfake, biên tập ảnh thông minh.

#### Cấu trúc của CNN

* Lớp tích chập (Convolutional Layer) : Là thành phần chính của CNN, giúp trích xuất đặc trưng từ hình ảnh. Mỗi lớp tích chập sử dụng các bộ lọc (kernel) nhỏ trượt qua ảnh để tính giá trị của từng vùng con.

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 vuông 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.



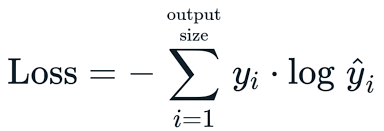
* Hàm kích hoạt (Activation Function): Hàm kích hoạt thường dùng là ReLU (Rectified Linear Unit), giúp giảm tính phi tuyến tính và tăng khả năng huấn luyện.
* Lớp giảm kích thước (Pooling Layer): Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu 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 giá trị lớn nhất trong mỗi vùng (thường dùng kích thước 2x2 hoặc 3x3)) hoặc Average Pooling (Lấy giá trị trung bình của mỗi vùng.)

* Lớp kết nối đầy đủ (Fully Connected Layer): Là lớp cuối cùng, kết nối toàn bộ các đặc trưng đã trích xuất để dự đoán kết quả.
* Lớp Softmax: Khi mô hình cần phân loại nhiều lớp, Softmax chuyển đầu ra thành dạng xác suất.

#### Quá trình huấn luyện CNN

* Tiền xử lý dữ liệu: Chuẩn hóa ảnh, tăng cường dữ liệu.
* Truyền dữ liệu qua mạng (Forward Pass): Truyền dữ liệu qua các lớp tích chập, pooling, fully connected. Các lớp convolutional sẽ sử dụng các bộ lọc để quét qua hình ảnh, tạo ra các bản đồ đặc trưng (feature maps). Sau khi các đặc trưng được trích xuất qua các lớp, lớp fully connected sẽ giúp dự đoán kết quả (ví dụ: phân loại hình ảnh thành các lớp).
* Tính toán hàm mất mát (loss function): So sánh dự đoán với nhãn thực tế. Hàm mất mát được sử dụng để đo độ sai lệch giữa dự đoán của mô hình và giá trị thực tế (label). Một trong những hàm mất mát phổ biến trong CNN là Cross-Entropy Loss cho các bài toán phân loại.



Trong đó:

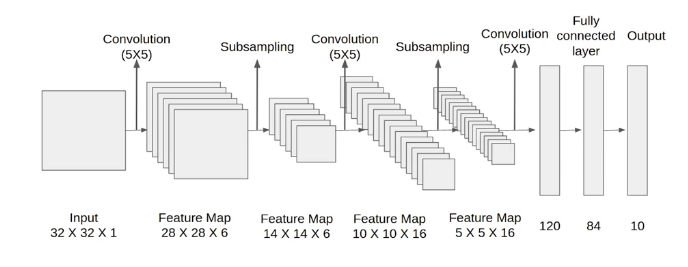
* output size: Số lượng lớp.
* y(i)​: Nhãn thực tế (ground truth) của lớp i.
* y^(i): Xác suất dự đoán của lớp i từ mô hình.
* Lan truyền ngược (Backpropagation): Đây là một quá trình quan trọng trong việc huấn luyện mạng nơ-ron. Sau khi tính toán được hàm mất mát, thuật toán backpropagation sẽ tính toán gradient của hàm mất mát đối với các trọng số của các lớp trong mạng. Dựa trên gradient, thuật toán Gradient Descent sẽ điều chỉnh các trọng số của mạng để giảm thiểu hàm mất mát. Các phương pháp cải tiến của gradient descent như Stochastic Gradient Descent (SGD), Adam, hoặc RMSprop được sử dụng để cập nhật các trọng số một cách hiệu quả hơn.
* Lặp lại quá trình huấn luyện (epochs): Quá trình forward pass, tính toán hàm mất mát, backpropagation và cập nhật trọng số sẽ được lặp đi lặp lại qua nhiều vòng (epochs). Mỗi lần qua tất cả dữ liệu huấn luyện được gọi là một epoch. Sau mỗi epoch, mô hình sẽ được kiểm tra trên bộ dữ liệu kiểm tra (validation set) để đánh giá hiệu suất và điều chỉnh các tham số học.

#### Các mô hình CNN nổi bật

* LeNet-5: Mô hình CNN đầu tiên.

Lenet-5 là một trong những mô hình được đào tạo trước sớm nhất do Yann LeCun và những người khác đề xuất vào năm 1998, trong bài báo nghiên cứu Gradient-Based Learning Applied to Document Recognition. Họ đã sử dụng kiến ​​trúc này để nhận dạng các ký tự viết tay và in bằng máy.

Kiến trúc của Lenet-5: Mạng có 5 lớp với các tham số có thể học được và do đó được đặt tên là Lenet-5. Nó có ba bộ lớp tích chập với sự kết hợp của nhóm trung bình. Sau lớp tích chập và nhóm trung bình, chúng ta có hai lớp được kết nối đầy đủ. Cuối cùng, một bộ phân loại Softmax phân loại hình ảnh thành các lớp tương ứng.

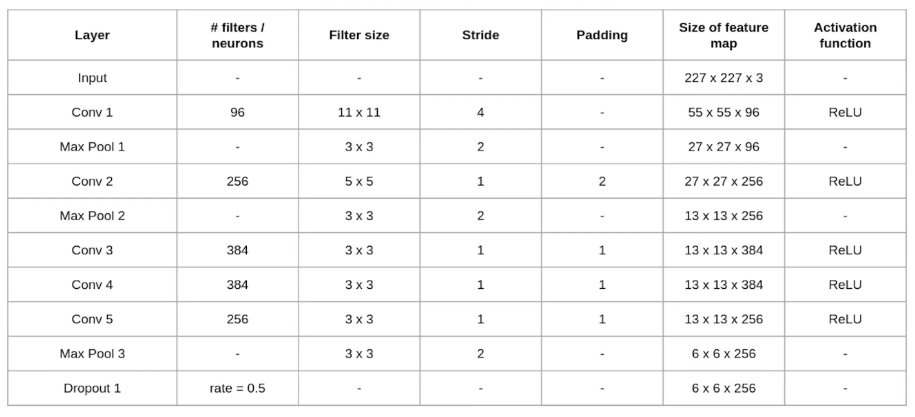


Hình 3: Kiến trúc của mô hình LeNet-5

* AlexNet: Mô hình CNN đầu tiên thắng giải ImageNet.

AlexNet là một cột mốc quan trọng trong thị giác máy tính, chứng minh sức mạnh của kiến ​​trúc sâu trong nhận dạng hình ảnh. Với tám lớp, 62,3 triệu tham số và các cải tiến như kích hoạt và bỏ qua ReLU, nó đã đặt nền tảng cho các mô hình AI hiện đại. Bài viết này đi sâu vào kiến ​​trúc của AlexNet và tác động lâu dài của nó đối với deep learning.

Một điều cần lưu ý ở đây là vì Alexnet là một kiến ​​trúc sâu, nên tác giả đã giới thiệu phần đệm để ngăn kích thước của bản đồ đặc điểm giảm đáng kể. Đầu vào của mô hình này là hình ảnh có kích thước 227x227x3.



Các lớp tích chập và max-pooling là các khối xây dựng cơ bản của AlexNet. Các lớp này trích xuất các tính năng và giảm kích thước không gian, cho phép xử lý hiệu quả trong khi vẫn giữ lại thông tin hình ảnh quan trọng.



Cuối cùng, chúng ta có lớp kết nối đầy đủ cuối cùng hoặc lớp đầu ra với 1000 nơ-ron vì chúng ta có 10000 lớp trong tập dữ liệu. Hàm kích hoạt được sử dụng ở lớp này là Softmax. Đây là kiến ​​trúc của mô hình Alexnet. Nó có tổng cộng 62,3 triệu tham số có thể học được.

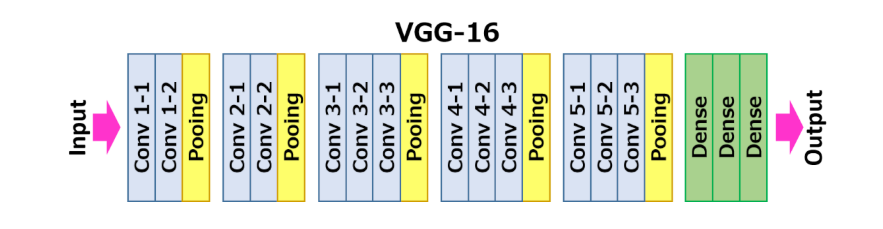
* VGGNet: Sâu hơn, dùng nhiều lớp tích chập.

VGG-16 là một trong những mô hình được đào tạo trước phổ biến nhất để phân loại hình ảnh. Được giới thiệu tại Hội nghị ILSVRC 2014 nổi tiếng, đây là và vẫn là MÔ HÌNH đánh bại ngay cả ngày nay. Được phát triển tại Visual Graphics Group tại Đại học Oxford, VGG-16 đã đánh bại AlexNet khi đó là tiêu chuẩn và nhanh chóng được các nhà nghiên cứu và ngành công nghiệp áp dụng cho các nhiệm vụ phân loại hình ảnh của họ.



Hình 4: Kiến trúc của mạng VGG-16

Sơ đồ trực quan hơn của Mô hình VGG-16.



* ResNet: Giải quyết vấn đề gradient vanishing.

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và thậm chí đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế nó còn đứng vị trí đầu tiên trong cuộc thi ILSVRC and COCO 2015 với ImageNet Detection, ImageNet localization, Coco detection và Coco segmentation.Hiện tại thì có rất nhiều biến thể của kiến trúc ResNet với số lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,...Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp nhất định.

A diagram of a diagram

AI-generated content may be incorrect.

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

* EfficientNet: Mô hình CNN tối ưu hiệu suất.

EfficientNet là kiến ​​trúc Mạng nơ-ron tích chập (CNN) sử dụng phương pháp chia tỷ lệ hợp chất để chia tỷ lệ độ sâu, chiều rộng và độ phân giải đồng đều, mang lại độ chính xác cao với hiệu quả tính toán.

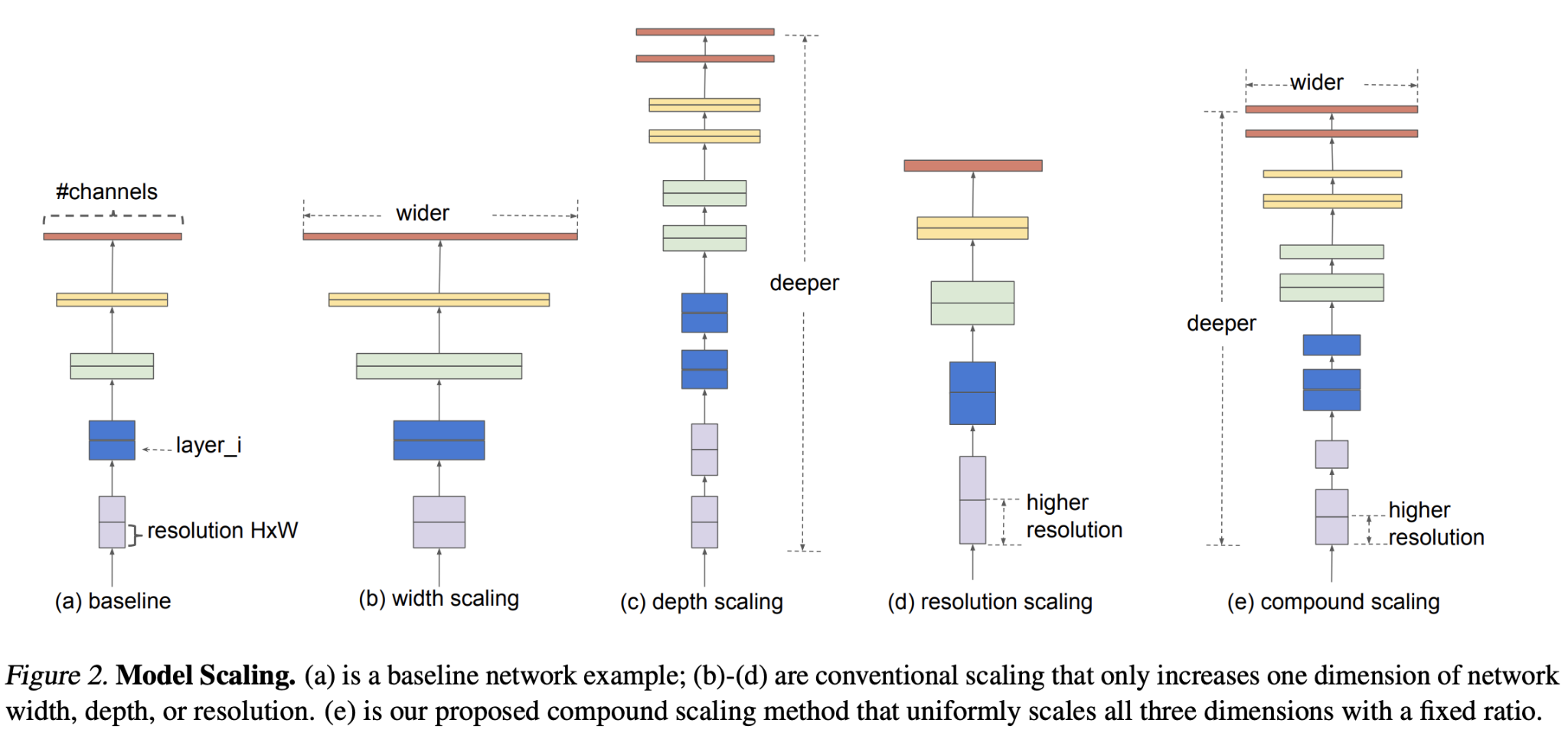
Như ta đã biết, có ba kích thước tỷ lệ của CNN: depth, width, and resolution:

* Depth là độ sâu của mạng tương đương với số lớp trong đó.
* Width là độ rộng của mạng. Ví dụ: một thước đo chiều rộng là số kênh trong lớp Conv
* Resolution là độ phân giải hình ảnh được chuyển đến CNN.

EfficientNet là một kiến ​​trúc mạng nơ-ron tích chập và phương pháp mở rộng quy mô thống nhất tất cả các chiều của độ sâu/chiều rộng/độ phân giải bằng cách sử dụng hệ số hợp chất. EfficientNet sử dụng hệ số hợp chất để mở rộng quy mô thống nhất chiều rộng, chiều sâu và độ phân giải của mạng theo cách có nguyên tắc.

Phương pháp mở rộng quy mô hợp chất được chứng minh bằng trực giác rằng nếu hình ảnh đầu vào lớn hơn, thì mạng cần nhiều lớp hơn để tăng trường tiếp nhận và nhiều kênh hơn để nắm bắt các mẫu chi tiết hơn trên hình ảnh lớn hơn.

EfficientNet cũng truyền tốt và đạt được độ chính xác tiên tiến trên CIFAR-100 (91,7%), Flowers (98,8%) và 3 tập dữ liệu học chuyển giao khác, với số lượng tham số ít hơn theo cấp số nhân.



### TensorFlow và Keras

#### TensorFlow

##### TensorFlow là gì?

TensorFlow là một framework mã nguồn mở được phát triển bởi Google nhằm hỗ trợ việc xây dựng và triển khai các mô hình học máy và học sâu. TensorFlow cung cấp các API mạnh mẽ cho phép người dùng thực hiện các tác vụ phức tạp, từ việc huấn luyện mô hình đến triển khai trên nhiều nền tảng khác nhau, bao gồm máy chủ, thiết bị di động và web.

##### Các component của Tensorflow

* Tensor: Tên của TensorFlow được đưa ra trực tiếp là nhờ vào framework cốt lõi của nó: Tensor. Trong TensorFlow, tất cả các tính toán đều liên quan tới các tensor. 1 tensor là 1 vector hay ma trận của n-chiều không gian đại diện cho tất cả loại dữ liệu. Tất cả giá trị trong 1 tensor chứa đựng loại dữ liệu giống hệt nhau với 1 shape đã biết (hoặc đã biết 1 phần). Shape của dữ liệu chính là chiều của ma trận hay mảng. 1 tensor có thể được bắt nguồn từ dữ liệu input hay kết quả của 1 tính toán. Trong TensorFlow, tất cả các hoạt động được tiến hành bên trong 1 graph – biểu đồ. Biểu đồ là 1 tập hợp tính toán được diễn ra liên tiếp. Mỗi operation được gọi là 1 op node (operation node) và được kết nối với nhau.
* Graph: TensorFlow sử dụng framework dạng biểu đồ. Biểu đồ tập hợp và mô tả tất cả các chuỗi tính toán được thực hiện trong quá trình training. Biểu đồ cũng mang rất nhiều lợi thế:
  + Nó được làm ra để chạy trên nhiều CPU hay GPU, ngay cả các hệ điều hành trên thiết bị điện thoại.
  + Tính di động của biểu đồ cho phép bảo toàn các tính toán để bạn sử dụng ngay hay sau đó. Biểu đồ có thể được lưu lại để thực thi trong tương lai.
  + Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau. 1 tensor có 1 node và 1 edge. Node mang operation toán học và sản xuất các output ở đầu cuối. Các edge giải thích mối quan hệ input/output giữa các node.

#### Keras

##### Giới thiệu Keras

Keras là một API cấp cao cho TensorFlow, giúp đơn giản hóa quá trình xây dựng, huấn luyện và đánh giá các mô hình học sâu. Keras hỗ trợ nhiều loại mô hình, bao gồm:

* Sequential Model: Dễ dàng xây dựng các mô hình tuyến tính bằng cách xếp chồng các lớp.
* Functional API: Cho phép tạo ra các mô hình phức tạp hơn với nhiều đầu vào và đầu ra, hỗ trợ việc chia sẻ các lớp và tạo ra các mô hình phân nhánh.

Keras cũng hỗ trợ nhiều phương pháp tối ưu hóa, hàm mất mát (loss functions), và các kỹ thuật chuẩn hóa, giúp người dùng dễ dàng tối ưu hóa mô hình của mình và cải thiện độ chính xác.

Tính năng nổi bật của Keras

* Dễ sử dụng: Giao diện thân thiện và dễ hiểu giúp người dùng dễ dàng xây dựng và tinh chỉnh mô hình.
* Tính năng mở rộng: Hỗ trợ tích hợp với TensorFlow, giúp tận dụng sức mạnh của TensorFlow cho các tác vụ phức tạp hơn.
* Cộng đồng mạnh mẽ: Keras có một cộng đồng lớn và sôi động, cung cấp nhiều tài liệu và hỗ trợ cho người dùng

##### Các thành phần chính của Keras

Mô hình (Model): Keras hỗ trợ hai cách để xây dựng mô hình:

* Sequential API: Dễ sử dụng, các lớp được xếp chồng lên nhau theo thứ tự.
* Functional API: Linh hoạt hơn, hỗ trợ các mô hình phức tạp như mạng có nhiều đầu vào/đầu ra.

Các lớp chính (Layers): Một số lớp quan trọng trong Keras:

* Dense: Lớp kết nối đầy đủ.
* Conv2D: Lớp tích chập 2D dùng trong xử lý ảnh.
* LSTM, GRU: Các lớp dành cho mô hình tuần tự.
* Dropout: Giảm overfitting bằng cách vô hiệu hóa một số nơ-ron.

Loss function và Optimizer

* Loss function: Dùng để đánh giá mức độ sai lệch giữa dự đoán và giá trị thực tế (ví dụ: categorical\_crossentropy, mse).
* Optimizer: Các thuật toán tối ưu hóa như SGD, Adam, RMSprop giúp cập nhật trọng số trong quá trình huấn luyện.

#### Quy trình xây dựng mô hình với TensorFlow và Keras

Bước 1: Chuẩn bị dữ liệu

* Tiền xử lý dữ liệu (chuẩn hóa, chia tập dữ liệu).
* Augmentation (tăng cường dữ liệu để cải thiện khả năng tổng quát hóa).

Bước 2: Xây dựng mô hình

* Sử dụng Sequential API hoặc Functional API.
* Xác định số lớp, số nơ-ron, và hàm kích hoạt.

Bước 3: Compile mô hình

* Chọn Loss function, Optimizer, và Metrics.

Bước 4: Huấn luyện mô hình

* Sử dụng model.fit() để huấn luyện mô hình trên tập dữ liệu.
* Theo dõi quá trình học thông qua các chỉ số như accuracy, loss.

Bước 5: Đánh giá và tinh chỉnh mô hình

* Kiểm tra mô hình trên tập kiểm tra.
* Điều chỉnh hyperparameter nếu cần.

#### Ứng dụng của TensorFlow và Keras

TensorFlow và Keras có mặt trong nhiều lĩnh vực khác nhau:

* Thị giác máy tính: Nhận diện hình ảnh, phân loại đối tượng (dùng CNN).
* Xử lý ngôn ngữ tự nhiên (NLP): Dịch máy, chatbot, phân tích cảm xúc.
* Y tế: Chẩn đoán bệnh từ ảnh X-quang, MRI.
* Tài chính: Phát hiện gian lận, dự đoán giá chứng khoán.

TensorFlow và Keras là những công cụ mạnh mẽ trong lĩnh vực trí tuệ nhân tạo và học sâu. Nhờ tính linh hoạt và dễ sử dụng, chúng đã trở thành tiêu chuẩn trong nghiên cứu và ứng dụng thực tế. Việc thành thạo TensorFlow và Keras sẽ giúp phát triển các mô hình AI hiệu quả, ứng dụng vào nhiều lĩnh vực khác nhau.

### Tập dữ liệu Fashion-MNIST

#### Giới thiệu tập dữ liệu

Fashion-MNIST là một tập dữ liệu được phát triển bởi Zalando, một nền tảng thương mại điện tử nổi tiếng trong lĩnh vực thời trang. Tập dữ liệu này được công bố lần đầu vào năm 2017 như một sự thay thế cho tập dữ liệu MNIST, vốn nổi tiếng trong cộng đồng học máy nhưng đã trở nên quen thuộc và kém thách thức hơn cho các mô hình học sâu hiện đại. Fashion-MNIST nhằm mục đích cung cấp một bài toán phân loại hình ảnh thú vị hơn, với các đặc điểm và thử thách thực tế hơn so với nhận diện chữ số viết tay.

Fashion-MNIST chứa tổng cộng 70,000 hình ảnh, trong đó 60,000 hình ảnh dành cho tập huấn luyện và 10,000 hình ảnh dành cho tập kiểm tra. Các hình ảnh trong tập dữ liệu này đều là hình ảnh màu xám (grayscale) với kích thước tiêu chuẩn là 28x28 pixel. Mỗi hình ảnh được mã hóa từ 0 đến 255, trong đó 0 biểu thị cho màu đen và 255 cho màu trắng, cho phép dễ dàng xử lý và phân tích.

Việc sử dụng Fashion-MNIST trong nghiên cứu và phát triển mô hình học máy không chỉ giúp cải thiện khả năng nhận diện của các thuật toán mà còn hỗ trợ các nhà nghiên cứu trong việc thử nghiệm và tối ưu hóa các phương pháp mới mà không cần phải thu thập dữ liệu hình ảnh từ thực tế, điều này có thể tốn thời gian và nguồn lực đáng kể.



Hình 5: Bộ hình ảnh của tập dữ liệu Fashion-MNIST

#### Cấu trúc và định dạng của tập dữ liệu

Tập dữ liệu Fashion-MNIST được tổ chức theo định dạng tương tự như MNIST, giúp các nhà phát triển dễ dàng áp dụng các mô hình đã được huấn luyện trước đó. Cấu trúc của tập dữ liệu bao gồm hai phần chính: tập huấn luyện và tập kiểm tra.

* Tập huấn luyện (Training set): Bao gồm 60,000 hình ảnh và nhãn tương ứng. Mỗi hình ảnh được lưu trữ dưới dạng ma trận 2 chiều với kích thước 28x28 pixel, tạo thành một ma trận 4 chiều với định dạng (60,000, 28, 28, 1), trong đó 1 đại diện cho kênh màu xám.
* Tập kiểm tra (Test set): Bao gồm 10,000 hình ảnh và nhãn tương ứng, với cùng kích thước như tập huấn luyện. Tương tự, tập kiểm tra được lưu trữ dưới dạng ma trận 4 chiều với định dạng (10,000, 28, 28, 1).

Cấu trúc này giúp đơn giản hóa quá trình nhập dữ liệu vào mô hình học sâu, nhờ vào tính đồng nhất trong cách tổ chức và định dạng dữ liệu. Các nhãn trong tập dữ liệu được lưu trữ dưới dạng mảng 1 chiều, nơi mỗi nhãn tương ứng với loại sản phẩm thời trang mà hình ảnh mô tả.

Ví dụ về cách tổ chức dữ liệu:

* X\_train: Ma trận 4 chiều chứa hình ảnh huấn luyện với kích thước (60,000, 28, 28, 1).
* y\_train: Mảng 1 chiều chứa nhãn của các hình ảnh huấn luyện với kích thước (60,000,).
* X\_test: Ma trận 4 chiều chứa hình ảnh kiểm tra với kích thước (10,000, 28, 28, 1).
* y\_test: Mảng 1 chiều chứa nhãn của các hình ảnh kiểm tra với kích thước (10,000,).

#### Các lớp trong tập dữ liệu

Fashion-MNIST phân loại các hình ảnh thành 10 lớp sản phẩm thời trang khác nhau, mỗi lớp đại diện cho một loại mặt hàng cụ thể:

* T-shirt/top: Lớp này bao gồm các loại áo phông, áo sơ mi, và áo thun. Đây là một trong những loại sản phẩm phổ biến nhất trong ngành thời trang.
* Trouser: Lớp này chứa các loại quần, bao gồm cả quần dài và quần short.
* Pullover: Áo len hoặc áo khoác nhẹ thường được sử dụng trong thời tiết lạnh. Đây là một sản phẩm thời trang quan trọng trong mùa đông.
* Dress: Váy là một trong những loại trang phục được ưa chuộng, đặc biệt trong các dịp lễ hội và sự kiện.
* Coat: Áo khoác là sản phẩm không thể thiếu trong tủ quần áo mùa lạnh, giúp giữ ấm và tạo phong cách thời trang.
* Sandal: Dép xỏ ngón và các loại sandal khác thường được sử dụng trong mùa hè. Đây là lựa chọn phổ biến cho thời trang đi biển.
* Shirt: Áo sơ mi dài tay, thường được mặc trong các sự kiện trang trọng hoặc công sở.
* Sneaker: Giày thể thao, mang lại sự thoải mái và phong cách cho người dùng, đặc biệt trong các hoạt động thể thao và thường ngày.
* Bag: Túi xách, được sử dụng để đựng đồ cá nhân và là một phần quan trọng của thời trang.
* Ankle boot: Bốt cổ thấp, thường được sử dụng trong mùa thu và đông, tạo nên phong cách trẻ trung và hiện đại.

Chương "Tổng quan đề tài" đã trình bày những nội dung quan trọng làm nền tảng cho nghiên cứu về ứng dụng Thị giác Máy tính trong nhận diện hình ảnh, cụ thể là sử dụng mô hình CNN (Convolutional Neural Network) để phân loại dữ liệu Fashion-MNIST với sự hỗ trợ của TensorFlow và Keras.

Nhìn chung, chương này cung cấp kiến thức nền tảng vững chắc cho việc nghiên cứu và triển khai mô hình nhận diện hình ảnh trong các chương tiếp theo.

# Xây dựng mô hình

Để xây dựng và tối ưu hóa mô hình nhận diện trang phục một cách hiệu quả, nhóm đã thực hiện một quy trình làm việc chi tiết trên môi trường Google Colab. Quy trình này bao gồm nhiều bước quan trọng, từ việc chuẩn bị dữ liệu, thiết kế kiến trúc mô hình, huấn luyện, đánh giá hiệu suất, cho đến tối ưu hóa để cải thiện độ chính xác.

Dưới đây là các bước cụ thể mà nhóm đã thực hiện trong suốt quá trình nghiên cứu và phát triển mô hình nhận diện trang phục:

1. Cài đặt thư viện

Cài đặt TensorFlow trên Google Colab để cung cấp các hàm và module cần thiết để xây dựng, huấn luyện và đánh giá mô hình.

**pip install tensorflow**

Sau đó cài đặt các thư viện cần dùng như sau:

**import tensorflow as tf**

**from tensorflow.keras.datasets import fashion\_mnist**

**from tensorflow.keras.utils import to\_categorical**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense, Flatten, Dropout**

**from tensorflow.keras.layers import Conv2D, MaxPooling2D**

**import matplotlib.pyplot as plt**

* Thư viện TensorFlow (đặt biệt danh là tf), công cụ mạnh mẽ để xây dựng và huấn luyện mô hình học máy.
* Tập dữ liệu fashion\_mnist chứa ảnh quần áo, dùng để huấn luyện mô hình nhận dạng ảnh.
* Hàm to\_categorical để chuyển đổi nhãn (ví dụ: "áo", "quần") sang dạng số mà mô hình học máy hiểu được [0..1].
* Mô hình Sequential để cấu trúc mạng nơ-ron theo kiểu xếp chồng các lớp.
* Các loại lớp cho mạng nơ-ron:
  + Dense, Flatten, Dropout: Các lớp cơ bản, thường dùng trong nhiều mô hình.
  + Conv2D, MaxPooling2D: Các lớp chuyên biệt cho xử lý ảnh.
* Thư viện Matplotlib (đặt biệt danh là plt) để tạo biểu đồ và hiển thị hình ảnh, giúp quan sát dữ liệu và kết quả.

1. Tiền xử lý dữ liệu

* Tải và cài đặt bộ dữ liệu Fashion\_MNIST để chia nhãn như sau:

**(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()**

* + train\_images và train\_labels: chứa 60.000 ảnh và nhãn dùng để huấn luyện.
  + test\_images và test\_labels: chứa 10.000 ảnh và nhãn dùng để đánh giá mô hình.
* Kiểm tra kích thước của dữ liệu:

**# Kiểm tra hình dạng dữ liệu và phạm vi giá trị**

**print(train\_images.shape)**

**print("Minimum value: {}, Maximum Value:{}".format(train\_images.max(), train\_images.min()))**

**train\_images[0]**

Ảnh có kích thước (28,28) và giá trị pixel trong khoảng 0-255.

* Chuẩn hoá dữ liệu

**train\_images = train\_images.reshape((train\_images.shape[0], 28, 28, 1)).astype('float32') / 255**

**test\_images = test\_images.reshape((test\_images.shape[0], 28, 28, 1)).astype('float32') / 255**

* + Đưa ảnh về dạng **(28,28,1)** (thêm kênh màu) và chuẩn hóa pixel về **[0,1]**.

**print("Before:",train\_labels)**

**train\_labels = to\_categorical(train\_labels)**

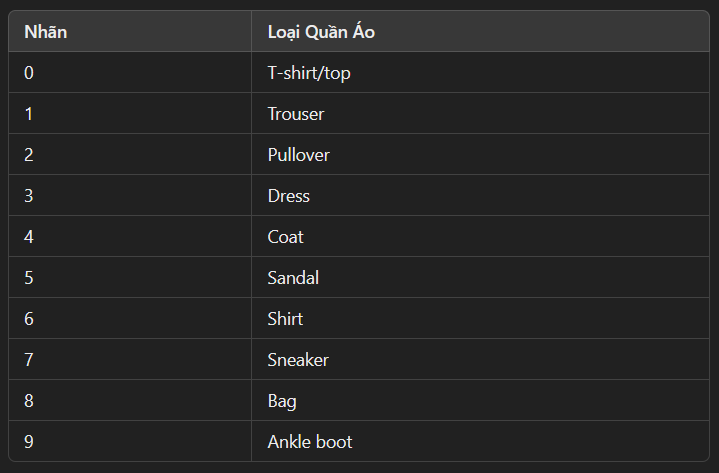
**test\_labels = to\_categorical(test\_labels)**

**print("After:", train\_labels)**

* + Biến đổi nhãn từ số nguyên **0-9** thành vector one-hot **(10 giá trị)**.

**labels\_dict = {0: "T-shirt/top", 1: "Trouser", 2: "Pullover", 3: "Dress",4: "Coat",5: "Sandal",6: "Shirt",7: "Sneaker",8: "Bag",9: "Ankle boot"}**

* + Tạo ánh xạ từ số **(0-9)** sang loại quần áo.



Hình 6: Bảng ánh xạ nhãn quần áo

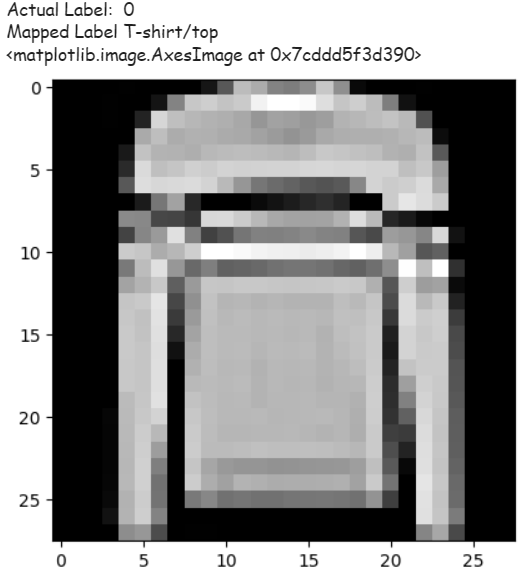
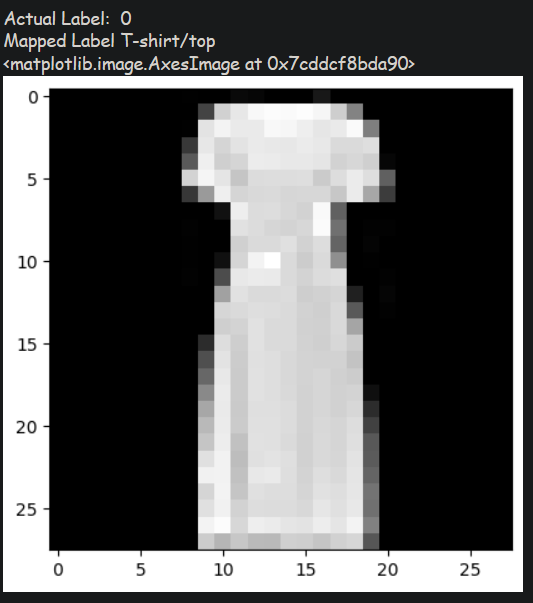
* Trực quan hoá dữ liệu

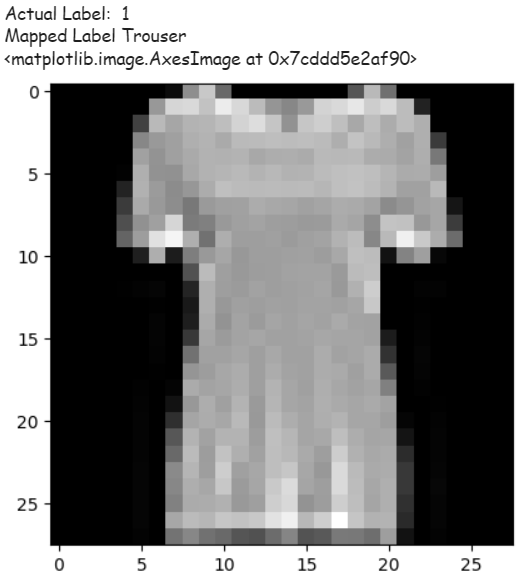
Trực quan hóa hình ảnh từ tập dữ liệu training bằng lệnh imshow() của thư viện matplotlib để xem một trong những hình ảnh có trong tập dữ liệu.

#Có thể thay số labels kiểm tra các dữ liệu khác

**print("Actual Label: ", train\_labels[50].argmax())**

**print("Mapped Label", labels\_dict[train\_labels[50].argmax()])**

**plt.imshow(train\_images[50], cmap='gray')**

A black and white image of a boat

AI-generated content may be incorrect.

Hình 7, 8, 9, 10: Trực quan dữ liệu quần áo trong Fashion-MNIST

1. Tạo kiến trúc mô hình

Sử dụng API mô hình tuần tự (Sequential model API) để tạo một mô hình CNN đơn giản và xây dựng mô hình qua các kỹ thuật:

* Tăng cường kiến trúc mô hình

**from tensorflow.keras.layers import BatchNormalization**

**model = Sequential([**

**Conv2D(64, (3, 3), activation='relu', input\_shape=(28, 28, 1)),**

**BatchNormalization(),**

**MaxPooling2D((2, 2)),**

**Conv2D(128, (3, 3), activation='relu'),**

**BatchNormalization(),**

**MaxPooling2D((2, 2)),**

**Flatten(),**

**Dense(128, activation='relu'),**

**BatchNormalization(),**

**Dropout(0.8), # Bạn có thể thử điều chỉnh tỉ lệ dropout (ví dụ: 0.4)**

**Dense(10, activation='softmax')**

**])**

**model.summary()**

* + Conv2D → Lớp tích chập để trích xuất đặc trưng ảnh.
  + Batch Normalization → Chuẩn hóa giúp mô hình ổn định hơn.
  + Max Pooling 2D → Lấy đặc trưng chính bằng cách giảm kích thước.
  + Flatten → Chuyển dữ liệu về vector 1 chiều.
  + Dense(128, activation='relu') → Lớp Fully Connected để phân loại.
  + Dropout(0.8) → Ngăn overfitting bằng cách bỏ ngẫu nhiên 80% nơ-ron.
  + Dense(10, activation='softmax') → Lớp đầu ra với 10 nhãn.
* Tăng cường dữ liệu: Tạo ra các biến thể của ảnh huấn luyện (xoay/phóng/dịch chuyển), giúp mô hình học được các đặc trưng đa dạng hơn và giảm hiện tượng overfitting.

**from tensorflow.keras.preprocessing.image import ImageDataGenerator**

**# Tạo đối tượng ImageDataGenerator với các biến đổi**

**datagen = ImageDataGenerator(**

**rotation\_range=10, # Xoay ảnh ngẫu nhiên trong khoảng ±10 độ**

**zoom\_range=0.3, # Phóng to/thu nhỏ ảnh**

**width\_shift\_range=0.3, # Dịch chuyển ảnh theo chiều ngang**

**height\_shift\_range=0.3 # Dịch chuyển ảnh theo chiều dọc**

**)**

**# Áp dụng lên tập huấn luyện**

**datagen.fit(train\_images)**

* Sử Dụng Callbacks (Early Stopping và Model Checkpoint)

**from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint**

**# Dừng huấn luyện sớm nếu loss trên validation không cải thiện**

**early\_stop = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)**

**# Lưu lại mô hình tốt nhất**

**checkpoint = ModelCheckpoint('best\_model.h5', monitor='val\_loss', save\_best\_only=True)**

* + Early Stopping giúp tránh overfitting bằng cách dừng huấn luyện khi mô hình không cải thiện sau số epoch chỉ định tại chỉ số patience (ví dụ: patience=5).
  + ModelCheckpoint lưu lại phiên bản mô hình có hiệu suất tốt nhất dựa trên chỉ số loss trên tập validation.

1. Biên dịch mô hình

Sử dụng lệnh model.compile() để cấu hình quy trình học trước khi huấn luyện mô hình. Đây là nơi ta xác định loại hàm mất mát (loss function), trình tối ưu hóa (optimizer) và các số liệu được mô hình đánh giá trong quá trình huấn luyện và thử nghiệm.

**from tensorflow.keras.optimizers import Adam**

**from tensorflow.keras.callbacks import ReduceLROnPlateau**

**# Sử dụng Adam với learning rate nhỏ hơn**

**optimizer = Adam(learning\_rate=0.0005)**

**model.compile(optimizer=optimizer,**

**loss='categorical\_crossentropy',**

**metrics=['accuracy'])**

**# Tạo callback để giảm learning rate khi loss không cải thiện**

**lr\_scheduler = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=3)**

* Ở đây mô hình dùng bộ tối ưu Adam từ thư viện Keras của TensorFlow. Adam là một thuật toán tối ưu hóa phổ biến được sử dụng trong các mô hình học sâu. Nó được biết đến với hiệu quả và khả năng điều chỉnh tốc độ học cho từng tham số.
* Đặt learning\_rate (tốc độ học) thành 0.0005 sẽ kiểm soát mức độ điều chỉnh trọng số của mô hình trong mỗi bước huấn luyện. Tốc độ học nhỏ hơn giúp mô hình hội tụ đến giải pháp tốt hơn nhưng có thể yêu cầu thời gian huấn luyện lâu hơn.
* model.compile(...): Dòng này cấu hình mô hình cho việc huấn luyện.
  + optimizer=optimizer: Nó chỉ định bộ tối ưu Adam mà chúng ta vừa tạo.
  + loss='categorical\_crossentropy' : Điều này đặt hàm mất mát thành 'categorical\_crossentropy', phù hợp cho các bài toán phân loại nhiều lớp.
  + metrics=['accuracy']: Điều này yêu cầu mô hình theo dõi độ chính xác như một chỉ số hiệu suất trong quá trình huấn luyện.

1. Huấn luyện mô hình

Tiến hành huấn luyện mô hình với batch\_size = 128 và epochs = 10. Trong đó:

* Batch size: số lượng dữ liệu Mini-Batch Gradient Descent sử dụng trong 1 lần để cập nhật tham số
* Epoch: 1 epoch là một lần duyệt qua hết các dữ liệu trong tập huấn luyện.
* Huấn luyện mô hình với datagen.flow() để sử dụng ảnh đã tăng cường.

**batch\_size = 128**

**history = model.fit(**

**datagen.flow(train\_images, train\_labels, batch\_size=batch\_size),**

**validation\_data=(test\_images, test\_labels),**

**epochs=10, # Bạn có thể tăng số epoch nếu cần**

**callbacks=[early\_stop, lr\_scheduler, checkpoint]**

**)**

1. Kiểm tra độ chính xác

Sau khi huấn luyện, ta sẽ kiểm tra độ chính xác của mô hình với quy chuẩn sau:

* Với test\_loss: giá trị càng gần 0 càng tốt
* Với test\_acc: Thường được tính theo thang điểm từ 0 đến 1 (hoặc 0% đến 100%)

Với bài toán Fashion MNIST:

* Accuracy > 0.90 (90%) được xem là rất tốt
* Accuracy từ 0.85-0.90 (85-90%) là khá tốt
* Accuracy < 0.80 (80%) có thể cần cải thiện thêm

**test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)**

**print("Test Loss: ", test\_loss)**

**print("Test Accuracy: ", test\_acc)**

**training\_loss = history.history['loss']**

**training\_accuracy = history.history['accuracy']**

Để thấy rõ độ chính xác hơn qua từng epochs, ta sẽ tạo biểu đồ thể hiện quá trình với thư viện matplotlib

**# Create subplots for loss and accuracy**

**plt.figure(figsize=(10, 10))**

**# Loss subplot**

**plt.subplot(1, 2, 1)**

**plt.plot(training\_loss, label='Tổn thất')**

**plt.xlabel('Epochs')**

**plt.ylabel('Loss')**

**plt.legend()**

**# Accuracy subplot**

**plt.subplot(1, 2, 2)**

**plt.plot(training\_accuracy, label='Độ chính xác')**

**plt.xlabel('Epochs')**

**plt.ylabel('Accuracy')**

**plt.legend()**

**plt.show()**

* Khởi tạo hình vẽ với kích thước 10x10
* Vẽ biểu đồ tổn thất (Loss)
  + plt.subplot(1, 2, 1): Tạo subplot đầu tiên (bên trái) trong lưới 1 hàng, 2 cột.
  + plt.plot(training\_loss, label='Tổn thất'): Vẽ đường biểu diễn tổn thất theo số epoch.
  + plt.xlabel('Epochs'): Gán nhãn trục x là số epoch.
  + plt.ylabel('Loss'): Gán nhãn trục y là giá trị tổn thất.
  + plt.legend(): Hiển thị chú thích (label).
    - Nếu đường tổn thất giảm dần → Mô hình đang học tốt.  
      Nếu tổn thất dao động mạnh hoặc không giảm → Mô hình có thể đang bị overfitting hoặc chưa hội tụ.
* Vẽ biểu đồ độ chính xác (Accuracy)
  + plt.subplot(1, 2, 2): Tạo subplot thứ hai (bên phải) để hiển thị độ chính xác.
  + plt.plot(training\_accuracy, label='Độ chính xác'): Vẽ đường biểu diễn độ chính xác theo số epoch.
  + plt.xlabel('Epochs'): Nhãn trục x là số epoch.
  + plt.ylabel('Accuracy'): Nhãn trục y là độ chính xác.
  + plt.legend(): Hiển thị chú thích (label)

Nếu độ chính xác tăng dần → Mô hình đang học hiệu quả.  
Nếu độ chính xác không tăng hoặc giảm → Cần điều chỉnh siêu tham số như learning rate, số epoch, batch size,...

1. Thực nghiệm

Sau khi đánh giá độ chính xác, ta sẽ cho mô hình đưa ra đánh giá các dữ liệu có sẵn trong thư viện qua một mẫu kiểm tra. Cụ thể:

* Gọi hàm dự đoán với tập mẫu hình ảnh:

**predictions = model.predict(test\_images)**

* Tạo một mẫu kiểm tra chứa các hình bất kì, đưa ra dự đoán và so sánh với kết quả thực tế
* Ví dụ đưa ra mẫu kiểm tra 8x8 hình, bắt đầu từ ảnh c= 20

**# Create a axb grid of subplots**

**fig, axes = plt.subplots(8, 8, figsize=(10, 10))**

**# Assuming you have defined labels\_dict, predictions, and test\_labels earlier**

**c = 20**

**for i in range(8):**

**for j in range(8):**

**# Dự đoán của mô hình cho ảnh thứ c**

**prediction = predictions[c].argmax()**

**# Kết quả thực tế (lấy chỉ số nhãn từ one-hot encoding)**

**actual = test\_labels[c].argmax()**

**# Hiển thị ảnh (ảnh xám)**

**axes[i, j].imshow(test\_images[c], cmap='gray')**

**# Hiển thị dự đoán và kết quả thực tế**

**axes[i, j].set\_xlabel(f"Pred: {labels\_dict[prediction]}\nActual: {labels\_dict[actual]}")**

**axes[i, j].set\_xticks([]) # Ẩn trục x**

**axes[i, j].set\_yticks([]) # Ẩn trục y**

**c += 1**

**plt.tight\_layout(pad=3.0) # Điều chỉnh khoảng cách giữa các subplot**

**plt.show()**

* Trả về số lần đoán đúng/sai và phần trăm dự đoán chính xác

**# Chuyển đổi one-hot encoding của test\_labels về dạng số nguyên**

**actual\_labels = test\_labels.argmax(axis=1)**

**# Lấy nhãn dự đoán của mô hình**

**predicted\_labels = predictions.argmax(axis=1)**

**# Tính số lần dự đoán đúng và sai**

**correct\_predictions = (predicted\_labels == actual\_labels).sum()**

**wrong\_predictions = len(actual\_labels) - correct\_predictions**

**# Tính tỷ lệ phần trăm chính xác**

**accuracy = (correct\_predictions / len(actual\_labels)) \* 100**

**# In kết quả**

**print(f"Số lần dự đoán đúng: {correct\_predictions}")**

**print(f"Số lần dự đoán sai: {wrong\_predictions}")**

**print(f"Độ chính xác của mô hình trên tập kiểm tra: {accuracy:.2f}%")**

Sau khi hoàn tất việc cài đặt và huấn luyện, chạy tất cả dòng mã theo các bước để chạy mô hình và đưa ra dự đoán. Có thể tinh chỉnh các chỉ số như epoch, dropout, thêm các kỹ thuật tăng cường,... để tăng độ chính xác của dự đoán của mô hình.

**Kết luận**

Mô hình CNN cơ bản với TensorFlow/Keras với tập dữ liệu Fashion-MNIST được xây dựng trên Google Colab theo các bước sau:

1. Cài đặt và Import Thư viện: Chuẩn bị môi trường làm việc với TensorFlow và các thư viện cần thiết.
2. Tiền xử lý dữ liệu: Tải dữ liệu Fashion MNIST, kiểm tra kích thước và giá trị pixel, sau đó chuẩn hóa dữ liệu và chuyển đổi nhãn sang one-hotencoding.
3. Xây dựng mô hình CNN: Thiết kế kiến trúc mô hình sử dụng các lớp Conv2D, MaxPooling2D, BatchNormalization, Dense, Dropout… nhằm trích xuất đặc trưng và phân loại hình ảnh. Sử dụng ImageDataGenerator để tạo ra các biến thể của ảnh huấn luyện, giúp mô hình học được các đặc trưng đa dạng và giảm overfitting. Áp dụng EarlyStopping và ModelCheckpoint để tối ưu quá trình huấn luyện, tự động điều chỉnh learning rate và lưu lại mô hình tốt nhất.
4. Biên dịch và huấn luyện mô hình: Cấu hình optimizer (Adam với learning rate thấp), hàm mất mát (categorical\_crossentropy) và các số liệu (accuracy) để huấn luyện mô hình trên dữ liệu đã tăng cường.
5. Đánh giá mô hình: Đo lường hiệu suất của mô hình trên tập kiểm tra, trực quan hóa quá trình huấn luyện qua biểu đồ loss và accuracy.
6. Dự đoán và trực quan hóa kết quả: Sử dụng mô hình đã huấn luyện để dự đoán các ảnh kiểm tra, hiển thị kết quả dự đoán so với nhãn thực qua lưới các subplot, và tính toán độ chính xác tổng quát.

Qua các bước này, mô hình không chỉ được xây dựng và huấn luyện một cách hiệu quả mà còn được đánh giá trực quan và tổng quát, giúp dễ dàng nhận diện các điểm cần cải thiện.

# Kết quả thực nghiệm

## Kết quả huấn luyện

Sau khi xây dựng và huấn luyện mô hình CNN cơ bản với tập dữ liệu Fashion-MNIST, mô hình cơ bản đã đưa ra dự đoán với độ chính xác khá cao. Cụ thể như sau

* Tăng cường kiến trúc mô hình với các lớp:

**model = Sequential([**

**Conv2D(64, (3, 3), activation='relu', input\_shape=(28, 28, 1)),**

**BatchNormalization(),**

**MaxPooling2D((2, 2)),**

**Conv2D(128, (3, 3), activation='relu'),**

**BatchNormalization(),**

**MaxPooling2D((2, 2)),**

**Flatten(),**

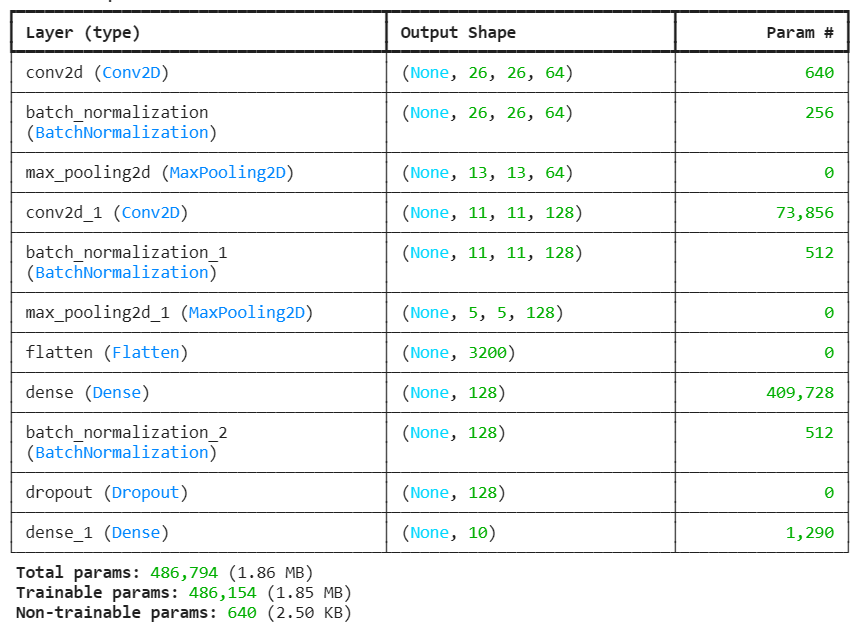
**Dense(128, activation='relu'),**

**BatchNormalization(),**

**Dropout(0.8),**

**Dense(10, activation='softmax')**

**])**

* Hai lớp tích chập (Conv2D) kết hợp với BatchNormalization và MaxPooling2D để trích xuất và giảm kích thước đặc trưng từ ảnh.
* Lớp Flatten để chuyển đổi dữ liệu từ dạng ma trận sang vector.
* Lớp Dense (fully connected) với BatchNormalization và Dropout nhằm giảm overfitting và học các mối quan hệ phức tạp trong dữ liệu.
* Lớp đầu ra với 10 neuron sử dụng softmax để phân loại ảnh vào 10 lớp của Fashion MNIST.  
  

Hình 11: Bảng tham số của cấu trúc CNN từ bộ mô hình số 1

Mô hình CNN này có cấu trúc hai khối Conv2D + MaxPooling2D, sau đó Flatten và hai lớp Dense (một ẩn, một đầu ra).

Số lượng tham số (~487k) là vừa phải, phù hợp cho việc huấn luyện trên các bộ dữ liệu cỡ trung bình.

Lớp Dense ẩn với 128 neuron chiếm phần lớn tham số, thể hiện khả năng học biểu diễn phi tuyến mạnh mẽ sau khi ảnh đã được trích xuất đặc trưng ở phần CNN.

BatchNormalization và Dropout được dùng để tăng ổn định và giảm overfitting, giúp mô hình có khả năng tổng quát tốt hơn.

* Tăng cường dữ liệu:

**datagen = ImageDataGenerator(**

**rotation\_range=10, # Xoay ảnh ngẫu nhiên trong khoảng ±10 độ**

**zoom\_range=0.3, # Phóng to/thu nhỏ ảnh**

**width\_shift\_range=0.3, # Dịch chuyển ảnh theo chiều ngang**

**height\_shift\_range=0.3 # Dịch chuyển ảnh theo chiều dọc**

**)**

* rotation\_range=10: Cho phép xoay ảnh một cách ngẫu nhiên trong khoảng ±10 độ.
* zoom\_range=0.3: Cho phép phóng to hoặc thu nhỏ ảnh ngẫu nhiên trong khoảng 30%. Điều này có nghĩa là kích thước của ảnh có thể thay đổi từ 70% đến 130% so với kích thước gốc.
* width\_shift\_range=0.3: Dịch chuyển ảnh theo chiều ngang một cách ngẫu nhiên trong khoảng ±30% chiều rộng của ảnh.
* height\_shift\_range=0.3: Dịch chuyển ảnh theo chiều dọc một cách ngẫu nhiên trong khoảng ±30% chiều cao của ảnh.
* Số lượng epochs: 10

**batch\_size = 128**

**history = model.fit(**

**datagen.flow(train\_images, train\_labels, batch\_size=batch\_size),**

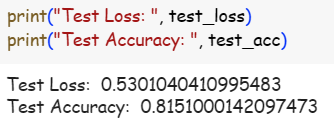
**validation\_data=(test\_images, test\_labels),**

**epochs=10,**

**callbacks=[early\_stop, lr\_scheduler, checkpoint]**

**)**

* Chỉ số độ chính xác và tổn thất của mô hình lần lượt là Test Accuracy: 0.8151000142097473 và Test Loss: 0.5301040410995483



* Biểu đồ của chỉ số qua từng epochs



Hình 12: Biểu đồ Tổn thất và Độ chính xác mô hình qua từng epochs của bộ mô hình số 1

* Kết quả dự đoán một mẫu kiểm tra 8x8, bắt đầu từ hình 20:



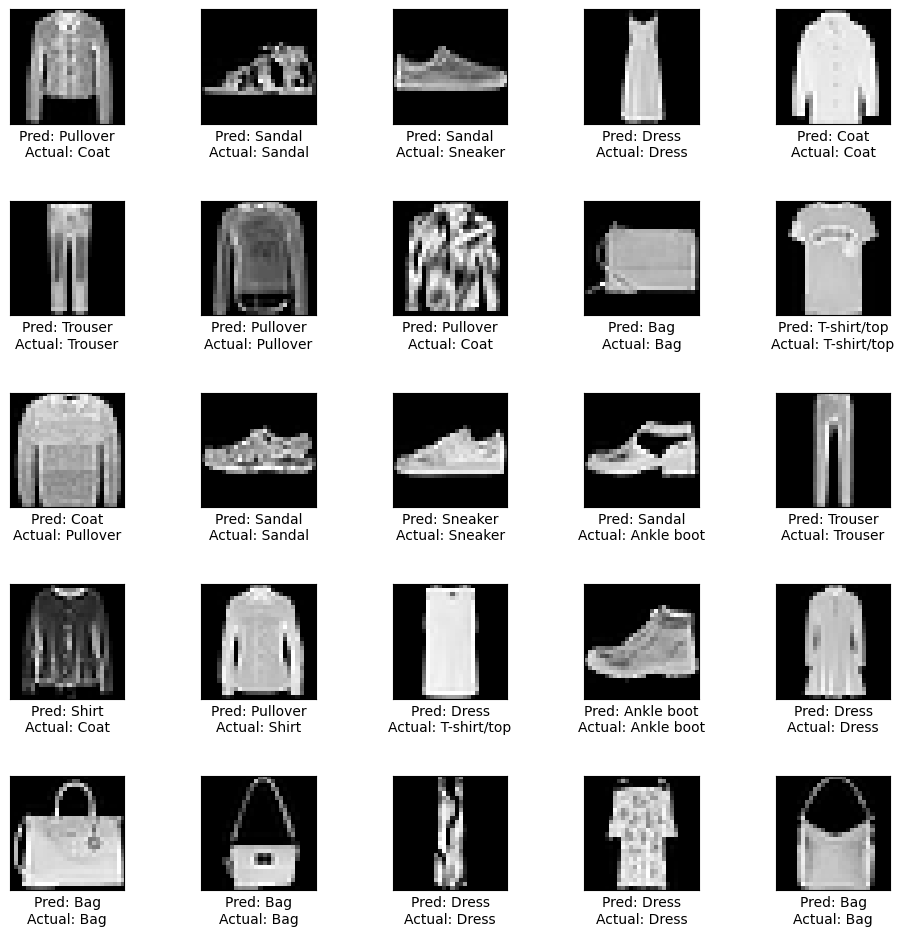
Hình 13: Kết quả dự đoán của mô hình số 1 cỡ 8x8

Kết quả dự đoán thu thập được như sau:

Số lần dự đoán đúng: 8151

Số lần dự đoán sai: 1849

Độ chính xác của mô hình trên tập kiểm tra: 81.51%

* Kết quả dự đoán một mẫu kiểm tra 5x5, bắt đầu từ hình 10:  
  

Hình 14: Kết quả dự đoán của mô hình số 1 cỡ 5x5

Số lần dự đoán đúng: 7664

Số lần dự đoán sai: 2336

Độ chính xác của mô hình trên tập kiểm tra: 76.64%

**Đánh giá kết quả:**

* Test Accuracy: ~80%: Độ chính xác này ở mức trung bình, chưa thực sự tốt cho bài toán Fashion MNIST. Ta có thể thấy ở biểu đồ Độ chính xác (Accuracy) ở bên phải:
  + Độ chính xác tăng tương ứng với sự giảm dần của tổn thất, bắt đầu từ mức dưới 0.4 (40%) và tăng lên khoảng 0.75 (75%) sau 9 epoch.
  + Đường accuracy chưa có dấu hiệu bão hòa hoàn toàn (vẫn đang tăng) nên mô hình có thể tiếp tục cải thiện nếu huấn luyện thêm (trong trường hợp không bị overfitting).

Thông thường, một mô hình CNN tốt cho Fashion MNIST có thể đạt:

* Độ chính xác > 90% là rất tốt
* Độ chính xác 85-90% là khá tốt
* Độ chính xác < 80% (như trường hợp này) cần cải thiện
* Test Loss: ~0.5: Chỉ số tổn thất này hơi cao, cho thấy mô hình còn khá nhiều sai số trong dự đoán, giá trị loss càng gần 0 càng tốt.

Điều đó cũng thể hiện ở biểu đồ Tổn thất (Loss) ở bên trái:

* + Đường tổn thất bắt đầu ở mức khá cao (xấp xỉ 1.8–2.0) và giảm dần qua các epoch.
  + Đến epoch thứ 9, tổn thất còn khoảng 0.5–0.6, cho thấy mô hình đang học tốt và dần thu hẹp sai số giữa dự đoán và nhãn thực tế.
  + Việc tổn thất liên tục giảm là dấu hiệu tích cực, cho thấy mô hình tiếp tục hội tụ.

Nhận xét tổng quan:

* Mô hình học ổn định: Tổn thất giảm liên tục và độ chính xác tăng dần cho thấy mô hình đang học đúng hướng, không có dấu hiệu dao động mạnh.
* Còn dư địa cải thiện: Với độ chính xác ở mức ~75% sau 9 epoch, mô hình có thể tiếp tục được huấn luyện lâu hơn, hoặc áp dụng thêm các kỹ thuật (tăng cường dữ liệu, điều chỉnh kiến trúc, tinh chỉnh hyperparameters) để đẩy cao độ chính xác.
* Chưa xuất hiện overfitting rõ rệt: Chưa có biểu hiện loss tăng trở lại hoặc accuracy giảm, tuy nhiên để kết luận chính xác cần so sánh thêm với đường validation loss/accuracy (nếu có).

Tóm lại, mô hình đang trong giai đoạn học tích cực, giảm tổn thất và tăng độ chính xác một cách đều đặn. Nếu muốn đạt kết quả cao hơn, ta có thể tăng số epoch, áp dụng regularization, hoặc thử các kiến trúc CNN sâu hơn tùy vào mục tiêu và tài nguyên tính toán.

## Cải thiện mô hình

Để tăng độ chính xác, nhóm đã áp dụng một số phương pháp để tăng độ chính xác của mô hình, cụ thể:

* Tăng cường kiến trúc mô hình với các lớp:
  + Lớp tích chập: Ở khối đầu tiên, mô hình bổ sung thêm một lớp Conv2D nữa với 64 bộ lọc ngay sau lớp BatchNormalization. Sau đó mới áp dụng MaxPooling2D. Việc thêm lớp Conv2D thứ hai giúp mô hình có khả năng trích xuất các đặc trưng phức tạp hơn từ ảnh trước khi giảm kích thước đặc trưng.
  + Dropout: Tỉ lệ dropout giảm xuống 0.5, cho phép mô hình giữ lại nhiều thông tin hơn so với tỉ lệ 0.8.

**model = Sequential([**

**Conv2D(64, (3, 3), activation='relu', input\_shape=(28, 28, 1)),**

**BatchNormalization(),**

**Conv2D(64, (3, 3), activation='relu'),**

**MaxPooling2D((2, 2)),**

**Conv2D(128, (3, 3), activation='relu'),**

**BatchNormalization(),**

**MaxPooling2D((2, 2)),**

**Flatten(),**

**Dense(128, activation='relu'),**

**BatchNormalization(),**

**Dropout(0.5),**

**Dense(10, activation='softmax')**

**])**

* Tăng cường dữ liệu: Zoom, Width, Height Shift Range = 0.1: Áp dụng các biến đổi nhỏ hơn (tương đương ±10% thay vì ±30%) giúp mô hình thấy được sự thay đổi nhẹ của ảnh.

**datagen = ImageDataGenerator(**

**rotation\_range=10, # Xoay ảnh ngẫu nhiên trong khoảng ±10 độ**

**zoom\_range=0.1, # Phóng to/thu nhỏ ảnh**

**width\_shift\_range=0.1, # Dịch chuyển ảnh theo chiều ngang**

**height\_shift\_range=0.1 # Dịch chuyển ảnh theo chiều dọc**

**)**

* + Số lượng epochs: tăng số lượng epochs từ 10 → 15

So sánh giữa hai bộ mô hình như sau:

|  |  |  |
| --- | --- | --- |
|  | Bộ mô hình số 1 | Bộ mô hình số 2 |
| Tăng cường kiến trúc mô hình | Hai lớp Conv2D  Dropout(0.8) | Ba lớp Conv2D  Dropout(0.5) |
| Tăng cường dữ liệu | Zoom, Width, Height Shift Range = 0.3 | Zoom, Width, Height Shift Range = 0.1 |
| Số lượng epochs: | 10 | 15 |

Bảng 1: Bảng so sánh giữa nội dung của hai bộ mô hình

Kết quả thu được như sau:

* Khi tăng cường kiến trúc dữ liệu:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

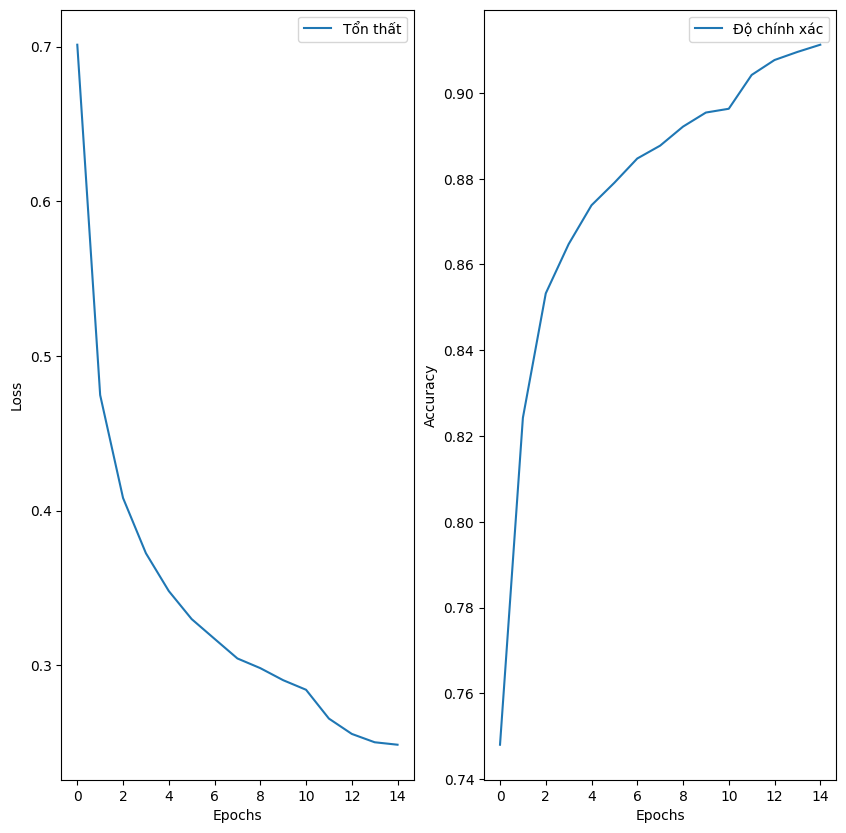
AI-generated content may be incorrect.

Hình 15: Kết quả tham số của cấu trúc mô hình CNN tăng cường của mô hình số 2

* + Ở **mô hình cũ**, khối đầu tiên có thể chỉ có 1 Conv2D (64 filters) rồi MaxPooling2D.
  + Ở **mô hình mới**, khối đầu tiên có 2 Conv2D (64 filters) liên tiếp trước khi pooling, tạo ra thêm ~36.928 tham số.
  + Ngoài ra, kích thước output shape sau mỗi Conv2D/Pooling có thể thay đổi chút ít, nhưng nhìn chung sự khác biệt chính vẫn là lớp Conv2D bổ sung.

Mô hình có thêm một lớp tích chập giúp trích xuất nhiều đặc trưng hơn ở khối đầu, thường dẫn đến khả năng học tốt hơn (nhưng cũng đòi hỏi tài nguyên tính toán cao hơn). Nếu không có đủ dữ liệu hoặc không dùng các kỹ thuật regularization (như Dropout, BatchNormalization), mô hình nhiều tham số hơn có thể dễ overfit. Nhiều tham số hơn đồng nghĩa thời gian train mỗi epoch có thể dài hơn một chút, đặc biệt trên CPU.

* Khi tăng cường dữ liệu: với **Zoom, Width, Height Shift Range = 0.1:**
  + Giữ lại nhiều thông tin gốc hơn, giảm nguy cơ tạo ra các biến dạng quá mức, phù hợp nếu dữ liệu của bạn đã có độ đa dạng tương đối hoặc khi bạn muốn mô hình học những đặc trưng chính xác mà không bị nhiễu do quá nhiều biến đổi.
  + Nếu dữ liệu huấn luyện hạn chế về sự đa dạng, mức tang cường nhẹ có thể không đủ để cải thiện khả năng tổng quát hóa của mô hình.
* Tăng số epoch sẽ làm tăng lần huấn luyện của mô hình, từ đó cũng tăng độ chính xác của dự đoán:



Hình 16: Biểu đồ Tổn thất và Độ chính xác dự đoán sau mỗi epoch của mô hình số 2

* Đồ thị bên trái - Tổn thất (Loss):
  + Trục y: giá trị loss (từ 0.3 đến 0.7).
  + Trục x: số epoch (từ 0 đến 14).
  + Đường màu xanh thể hiện loss giảm dần theo thời gian.
  + Loss giảm mạnh trong các epoch đầu (từ 0.7 xuống 0.4). Sau đó giảm chậm dần và ổn định ở khoảng 0.25.

Xu hướng giảm này cho thấy mô hình đang học tốt hơn.

* Đồ thị bên phải - Độ chính xác (Accuracy):
  + Trục y: độ chính xác (từ 0.74 đến 0.90 tương đương 74% đến 90%).
  + Trục x: số epoch (từ 0 đến 14).
  + Đường màu xanh thể hiện accuracy tăng dần theo thời gian.
  + Accuracy tăng nhanh trong các epoch đầu (từ 75% lên 85%). Sau đó tăng chậm và đạt khoảng 90% ở cuối quá trình.

Điều đó cho thấy mô hình đã cải thiện hơn do tổn thất giảm đều và ổn định, độ chính xác tăng đều và đạt mức khá cao (90%). Mô hình cũng không có dấu hiệu overfitting vì Loss không tăng trở lại và Accuracy không giảm ở các epoch cuối.

Ta có thể dừng training ở epoch 14 vì cả loss và accuracy đã bắt đầu ổn định. Không có nhiều cải thiện đáng kể ở các epoch cuối. Kết quả này tốt hơn nhiều so với kết quả kiểm tra độ chính xác ~80% trước đó.

Thực nghiệm cũng cho ra kết quả khả quan hơn:

* Với mẫu kiểm tra 8x8, bắt đầu từ hình 20



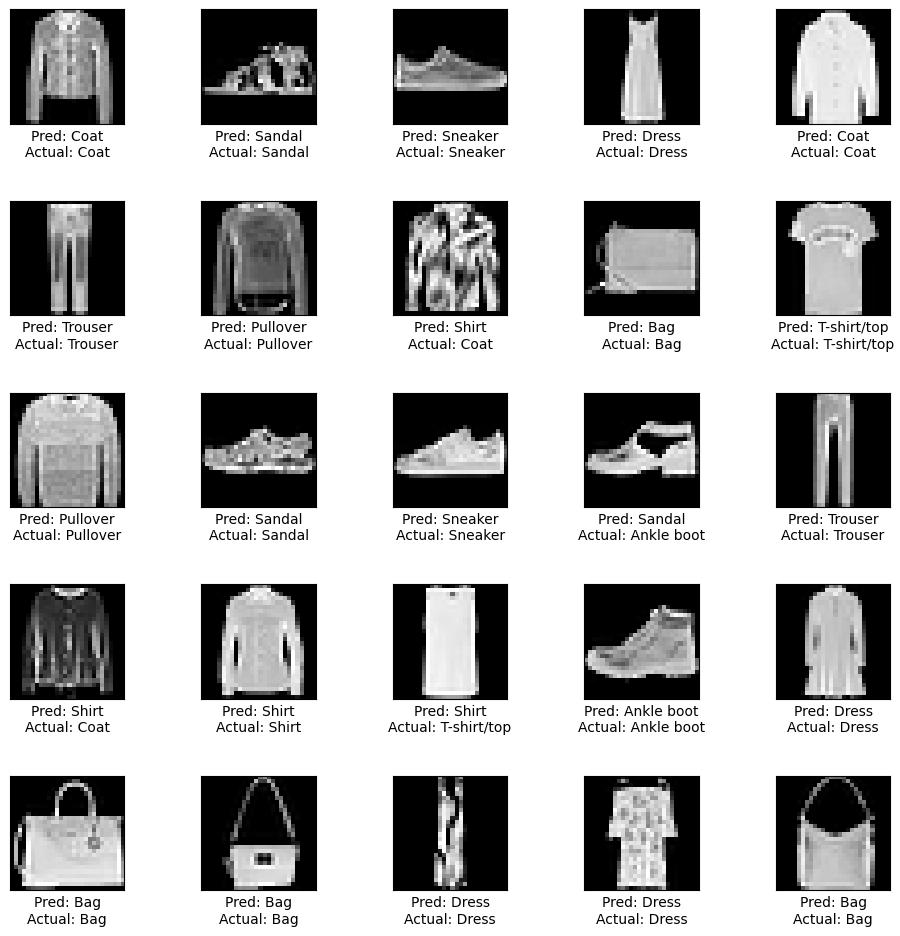
Hình 17: Kết quả dự đoán của mô hình số 2 cỡ 8x8

Số lần dự đoán đúng: 9109

Số lần dự đoán sai: 891

Độ chính xác của mô hình trên tập kiểm tra: 91.09%

* Với mẫu kiểm tra 5x5, bắt đầu từ hình 10



Hình 18: Kết quả dự đoán của mô hình số 2 cỡ 5x5

Số lần dự đoán đúng: 9109

Số lần dự đoán sai: 891

Độ chính xác của mô hình trên tập kiểm tra: 91.09%

So sánh giữa hai bộ mô hình kiểm tra:

|  |  |  |
| --- | --- | --- |
|  | Bộ kiểm tra số 1 | Bộ kiểm tra số 2 |
| Số epoch huấn luyện | 10 | 15 |
| Lượng tham số sử dụng | ~400.000 | ~500.000 |
| Độ chính xác của mô hình | ~80%: độ chính xác trung bình | ~90%: độ chính xác tốt |

Bảng 2: Bảng so sánh kết quả giữa 2 bộ kiểm tra

Kết luận:

Kết quả cho thấy mô hình đã được cải thiện đáng kể, với tổn thất giảm xuống mức thấp và độ chính xác đạt gần 90%. Quá trình huấn luyện diễn ra ổn định, chưa có dấu hiệu overfitting nghiêm trọng. Có thể thử tăng số epochs để xem mô hình có thể học tốt hơn không hoặc thử nghiệm với các kỹ thuật tăng cường khác nhau, điều chỉnh learning rate hoặc batch size và xem xét confusion matrix để phân tích các lớp bị nhầm lẫn nhiều nhất.

## So sánh với các mô hình khác

Bên cạnh mô hình CNN cơ bản còn có các mô hình khác thường được sử dụng trên tập dữ liệu Fashion MNIST:

* Mô hình Perceptron đa lớp (MLP - Multi-Layer Perceptron)

Mô hình chỉ sử dụng các lớp Dense (Fully Connected) không có lớp tích chập (Convolutional Layers). Yêu cầu chuyển đổi ảnh 2D thành vector 1D. MLP Đơn giản, dễ triển khai và hoạt động tốt trên dữ liệu dạng bảng. Tuy nhiên, mô hình này không tận dụng được cấu trúc không gian của ảnh (không giữ lại đặc trưng hình dạng, biên, họa tiết) và độ chính xác cũng thấp hơn CNN trên tập dữ liệu hình ảnh.

MLP đạt độ chính xác khoảng 88-90% trên tập Fashion MNIST. CNN cơ bản có thể đạt 92-95%, tùy vào số lượng lớp và cách tối ưu.

* Mô hình CNN sâu hơn (Deep CNN)

Nếu tăng số lớp Conv2D và MaxPooling2D, mô hình có thể trở thành một CNN sâu hơn với hiệu suất tốt hơn.

* Nhiều lớp Conv2D (tích chập) với số lượng filters cao hơn.
* Batch Normalization để giúp tối ưu hóa quá trình huấn luyện.
* Global Average Pooling thay vì Flatten để giảm số lượng tham số.

Ưu điểm của Deep CNN là bắt được nhiều đặc trưng chi tiết hơn từ hình ảnh và hoạt động tốt hơn trên tập dữ liệu lớn. Khi tăng số lượng lớp thì mô hình yêu cầu nhiều tài nguyên tính toán hơn và dễ bị overfitting nếu không có dropout hoặc regularization.

So sánh kết quả:

* CNN cơ bản có thể đạt ~92-94% độ chính xác.
* CNN sâu hơn có thể đạt ~96-98% nhưng cần thời gian huấn luyện lâu hơn.
* ResNet-50 hoặc MobileNet (Mạng nơ-ron tích chập tiền huấn luyện - Pretrained Models)

Mô hình Pretrained (Sử dụng Transfer Learning) sử dụng mô hình như ResNet-50, MobileNet, VGG16 đã được huấn luyện trước trên tập dữ liệu ImageNet. Chỉ cần thay thế phần fully connected (Dense Layer) để phù hợp với Fashion MNIST. Từ đó mô hình có thể đạt độ chính xác cao trên Fashion MNIST với dữ liệu ít hơn đồng thời được tối ưu hóa tốt, tận dụng kiến thức từ tập dữ liệu lớn. Với kích thước mô hình lớn hơn, nó sẽ yêu cầu nhiều tài nguyên tính toán và không phải lúc nào cũng hiệu quả trên các bài toán nhỏ hơn.

**So sánh kết quả:**

* ResNet-50 hoặc MobileNet có thể đạt ~98-99% độ chính xác trên Fashion MNIST.
* Tuy nhiên, mô hình lớn hơn có thể gây quá tải nếu không có GPU mạnh.

Với các mô hình phổ biến trên, ta có thể rút ra một bảng so sánh sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Mô hình | Độ chính xác trên Fashion MNIST | Số tham số | Yêu cầu tài nguyên |
| MLP (Dense Only) | ~88-90% | Trung bình | Ít |
| CNN cơ bản (mô hình hiện tại) | ~92-94% | Trung bình | Vừa |
| CNN sâu hơn (Deep CNN) | ~96-98% | Cao | Cao |
| ResNet-50 / MobileNet | ~98-99% | Rất cao | Rất cao |

Bảng 3: Bảng so sánh mô hình CNN cơ bản với các mô hình khác

Mô hình hiện tại (CNN cơ bản) đã tốt hơn MLP nhưng có thể cải thiện bằng cách tăng số lớp tích chập hoặc sử dụng kỹ thuật tối ưu hóa. Nếu có GPU mạnh thì có thể thử dùng mô hình ResNet hoặc MobileNet sẽ giúp đạt độ chính xác cao hơn.

# Đánh giá và kết luận

## Đánh giá kết quả

Mô hình được xây dựng dựa trên kiến trúc CNN, bao gồm các lớp tích chập (Conv2D) với các kích thước bộ lọc tăng dần, sau đó là các lớp pooling (MaxPooling2D) nhằm giảm kích thước đặc trưng và giảm số tham số. Sau giai đoạn tích chập, dữ liệu được chuyển sang dạng vector thông qua lớp Flatten và đưa vào các lớp Dense để thực hiện phân loại. Ngoài ra, dự án đã áp dụng các kỹ thuật như Dropout để giảm hiện tượng overfitting.

* Tiền xử lý dữ liệu: Dữ liệu ảnh được chuẩn hóa về khoảng giá trị [0, 1] và định dạng lại thành kích thước (28, 28, 1) phù hợp với đầu vào của mô hình CNN. Việc sử dụng dữ liệu Fashion MNIST với 60.000 ảnh huấn luyện và 10.000 ảnh kiểm tra đảm bảo tính đa dạng và đủ kích thước để huấn luyện mô hình hiệu quả.
* Tăng cường dữ liệu: Để tăng cường tính tổng quát của mô hình, bạn đã tích hợp các biến thể của dữ liệu thông qua ImageDataGenerator với các phép biến đổi như xoay, phóng to, và dịch chuyển ảnh. Điều này giúp mô hình học được nhiều đặc trưng đa dạng hơn và giảm thiểu overfitting.
* Biên dịch và Huấn luyện: Mô hình được biên dịch sử dụng thuật toán Adam với hàm mất mát (categorical\_crossentropy) phù hợp với bài toán phân loại đa lớp. Quá trình huấn luyện được theo dõi qua số epoch và batch size hợp lý, cùng với các callback như ReduceLROnPlateau, EarlyStopping và ModelCheckpoint nhằm tối ưu hóa quá trình huấn luyện và lưu lại mô hình tốt nhất.
* Đánh giá

Qua quá trình huấn luyện, mô hình thể hiện sự hội tụ với việc giảm dần giá trị loss và tăng dần độ chính xác trên tập huấn luyện cũng như tập kiểm tra. Biểu đồ loss và accuracy cho thấy mô hình đang học hiệu quả và có khả năng tổng quát tốt.

So với các mô hình khác (ví dụ như MLP hay các kiến trúc CNN phức tạp hơn), mô hình đạt được độ chính xác cao (trong khoảng 92-94% đối với CNN cơ bản). Nếu áp dụng thêm các cải tiến như tăng cường dữ liệu, Batch Normalization, L2 regularization và tối ưu hyperparameters, hiệu suất có thể được nâng cao thêm.

Việc sử dụng tăng cường dữ liệu cùng với các kỹ thuật regularization đã góp phần giảm thiểu overfitting. Mô hình cho thấy sự cân bằng giữa khả năng học sâu từ dữ liệu huấn luyện và khả năng dự đoán chính xác trên tập kiểm tra.  
Nhìn chung, mô hình CNN cơ bản đã trích xuất được các đặc trưng hình ảnh một cách hiệu quả. Sử dụng các kỹ thuật tiên tiến như tăng cường dữ liệu, dropout và các callbacks đã tối ưu hóa quá trình huấn luyện.

1. **Những hạn chế gặp phải**

Trong quá trình nghiên cứu và triển khai mô hình CNN để phân loại hình ảnh sản phẩm thời trang, nhóm đã gặp phải một số hạn chế và khó khăn sau:

* Hạn chế về tài nguyên tính toán
  + Việc huấn luyện mô hình CNN đòi hỏi tài nguyên tính toán lớn, đặc biệt khi làm việc với các tập dữ liệu lớn hơn hoặc mô hình phức tạp hơn.
  + Khi chạy trên máy tính cá nhân không có GPU, thời gian huấn luyện lâu hơn, ảnh hưởng đến quá trình thử nghiệm và tinh chỉnh mô hình.
* Giới hạn của tập dữ liệu Fashion-MNIST
  + Dù Fashion-MNIST là tập dữ liệu phổ biến, nhưng nó chỉ chứa hình ảnh grayscale (đen trắng), không phản ánh đầy đủ độ phức tạp của hình ảnh sản phẩm thực tế trên các nền tảng thương mại điện tử.
  + Một số sản phẩm trong tập dữ liệu có sự tương đồng cao về hình dạng, dẫn đến nhầm lẫn giữa các lớp (ví dụ: áo thun và áo sơ mi).
* Tối ưu hóa mô hình
  + Việc lựa chọn số lượng tầng tích chập (Convolutional Layers), số lượng filter, kích thước kernel và các siêu tham số như learning rate, batch size đòi hỏi nhiều thử nghiệm và tinh chỉnh thủ công.
  + Hiện tượng overfitting xảy ra khi mô hình hoạt động tốt trên tập huấn luyện nhưng không đạt hiệu suất cao trên tập kiểm tra, đặc biệt khi số epoch quá lớn.
* Khó khăn trong triển khai thực tế
  + Khi áp dụng mô hình vào một hệ thống thực tế (ví dụ: ứng dụng gợi ý sản phẩm trong thương mại điện tử), cần thêm các bước tiền xử lý dữ liệu như cân bằng ánh sáng, loại bỏ nhiễu, nhận diện nền phức tạp, điều này không được đề cập trong nghiên cứu này.
  + Ngoài ra, nếu triển khai trên môi trường thực tế, cần cải thiện tốc độ nhận diện để đảm bảo trải nghiệm người dùng tốt hơn.
* Hạn chế về thời gian và kinh nghiệm
  + Do giới hạn thời gian thực hiện nghiên cứu, nhóm chưa có cơ hội thử nghiệm các phương pháp nâng cao như Transfer Learning hoặc tăng cường dữ liệu cao cấp hơn để cải thiện hiệu suất mô hình.
  + Việc tìm hiểu sâu hơn về các kỹ thuật tối ưu hóa CNN đòi hỏi nhiều thời gian và kinh nghiệm hơn để đạt hiệu suất tốt nhất.

## Hướng khắc phục và phát triển trong tương lai

Để cải thiện các hạn chế đã nêu, một số giải pháp sau có thể được áp dụng trong các nghiên cứu tiếp theo nhằm nâng cao hiệu suất và khả năng ứng dụng thực tế của mô hình CNN:

* Tối ưu hóa hiệu suất mô hình
  + Sử dụng GPU/TPU: Thay vì huấn luyện mô hình trên CPU, có thể tận dụng GPU hoặc TPU để rút ngắn thời gian huấn luyện và thử nghiệm.
  + Điều chỉnh siêu tham số: Dùng các kỹ thuật tối ưu hóa như Grid Search hoặc Bayesian Optimization để tìm ra bộ tham số tốt nhất.
  + Data Augmentation: Áp dụng các phương pháp tăng cường dữ liệu như xoay, lật, thay đổi độ sáng để giúp mô hình học được nhiều đặc trưng hơn và giảm overfitting.
* Cải tiến mô hình CNN
  + Sử dụng các kiến trúc CNN nâng cao như ResNet, MobileNet, EfficientNet để tăng độ chính xác mà không làm tăng quá nhiều độ phức tạp của mô hình.
  + Thử nghiệm với Transfer Learning: Tận dụng các mô hình đã được huấn luyện trước trên các tập dữ liệu lớn hơn để cải thiện hiệu suất khi áp dụng vào bài toán phân loại sản phẩm thời trang.
  + Thêm cơ chế Attention: Giúp mô hình tập trung vào các đặc trưng quan trọng của hình ảnh, nâng cao độ chính xác trong phân loại.
* Nâng cao chất lượng tập dữ liệu
  + Sử dụng hình ảnh có màu và độ phân giải cao thay vì ảnh grayscale 28x28 của Fashion-MNIST để mô hình có thể học được nhiều đặc trưng chi tiết hơn.
  + Thu thập dữ liệu thực tế: Xây dựng một tập dữ liệu sản phẩm thời trang thực tế với nhiều nhãn hơn để phản ánh tốt hơn sự đa dạng trong ngành công nghiệp thời trang.
* Ứng dụng vào thực tế
  + Tích hợp vào hệ thống thương mại điện tử: Ứng dụng mô hình vào các nền tảng bán hàng trực tuyến để tự động phân loại sản phẩm và gợi ý sản phẩm cho người dùng.
  + Phát triển ứng dụng di động: Xây dựng ứng dụng nhận diện thời trang qua camera, giúp người dùng tìm kiếm sản phẩm theo hình ảnh thực tế.
  + Cải thiện thời gian suy luận (Inference Time): Tối ưu mô hình để tăng tốc độ phân loại, giúp hệ thống phản hồi nhanh hơn khi áp dụng vào các ứng dụng thực tế.

**Kết luận**

Mô hình CNN cơ bản đã áp dụng cho bài toán phân loại ảnh với Fashion MNIST. Nhờ các bước tiền xử lý và tối ưu hóa đã được tích hợp (data augmentation, dropout, callbacks), mô hình đã đạt được hiệu suất đáng kể về độ chính xác và khả năng tổng quát. Việc triển khai các cải tiến bổ sung sẽ giúp nâng cao hơn nữa kết quả dự đoán của mô hình trong các tình huống phức tạp hơn.

# Tài liệu tham khảo

Dataset source: <https://github.com/zalandoresearch/fashion-mnist>

[Easy & Step-by-Step Guide to Fashion MNIST classification with Deep Learning Neural Networks](https://www.youtube.com/watch?v=HZZ_vsVHQ6g&list=LL&index=7)

TensorFlow. (n.d.). Build a Convolutional Neural Network (CNN) with TensorFlow. TensorFlow Documentation.

<https://www.tensorflow.org/tutorials/images/cnn>

→ Hướng dẫn chính thức của TensorFlow về cách xây dựng và huấn luyện mô hình CNN.