|  |
| --- |
| BỘ GIAO THÔNG VẬN TẢI  TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI THÀNH PHỐ HỒ CHÍ MINH   REPORT ELEANING DEEP LEARNING    GVHD : TS. Nguyễn Thị Khánh Tiên  Tên lớp học phần : Deep learning  Mã lớp học phần : 7460108039316  *TPHCM, ngày 16, tháng 8, năm 2025* |

MỤC LỤC

|  |
| --- |
| Trang  Lời cam đoan/Bảng đánh giá công việc………………………………………………………. 3  Bản nhận xét của giảng viên hướng dẫn tiểu luận………………………………………...….. 4  ELEARNING  PRACTICE 1: PyTorch FashionMNIST Classification |

Lời cam đoan

Tôi: Nguyễn Văn Thương sinh viên lớp Học sâu, cam đoan rằng:

1. Bài báo cáo này là kết quả nghiên cứu và thực hiện của tôi. Tất cả các công trình nghiên cứu, dữ liệu, phương pháp phân tích và kết quả trong bài báo cáo đều là công sức riêng tôi.
2. Bài báo cáo này không sao chép từ bất kỳ bài báo cáo hay nghiên cứu nào của người khác mà không có sự trích dẫn đầy đủ.
3. Tôi cam kết rằng tất cả các thông tin trong bài báo cáo đều đúng sự thật và chúng tôi chịu trách nhiệm về tính chính xác của các dữ liệu và kết quả được trình bày.

Nếu có bất kỳ vấn đề nào phát sinh liên quan đến nội dung của báo cáo, tôi xin chịu hoàn toàn trách nhiệm.

Nguyễn Văn Thương.

Ngày 16 tháng 8 năm 2025.

BẢN NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TIỂU LUẬN

1. Họ và tên sinh viên: Nguyễn Văn Thương

2.Tên đề tài: ……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

3. Nhận xét:

a) *Về tinh thần, thái độ làm việc của sinh viên:*

……………………………………………………………………………………………………………………………………………………………………………………………………………………

b) *Những kết quả đạt được của tiểu luận đạt được:*

……………………………………………………………………………………………………………………………………………………………………………………………………

c) *Những hạn chế của bài tiểu luận:*

……………………………………………………………………………………………………………………………………………………………………………………

*TP. HCM, ngày ... tháng ... năm 2025*

Giảng viên hướng dẫn

*TS. Nguyễn Thị Khánh Tiên*

ELEARNING

PRACTICE 1: CLASSIFYING SPAM EMAILS

Giới thiệu

Bài thực hành này nhằm giúp sinh viên hiểu và vận dụng được quy trình cơ bản trong việc xây dựng, huấn luyện và đánh giá một mô hình học sâu (Deep Learning) bằng PyTorch.  
Cụ thể, bài tập hướng đến việc phân loại hình ảnh quần áo trong tập dữ liệu FashionMNIST – một phiên bản mở rộng của MNIST nhưng có độ khó cao hơn.

1.Learn

Study PyTorch Tensors, Datasets/Loaders, Transforms, Model Building, Autograd, Optimization, and Model Saving/Loading.

1.1. PyTorch Tensors

Tensor là cấu trúc dữ liệu cốt lõi trong PyTorch, tương tự như mảng NumPy, nhưng có thể chạy trên GPU giúp tăng tốc tính toán.

Học cách khởi tạo, thao tác, tính toán ma trận, và chuyển dữ liệu giữa CPU ↔ GPU.

Hiểu về thuộc tính .requires\_grad để bật/tắt việc tính đạo hàm tự động cho các biến.

1.2. Datasets và Dataloaders

Dataset là nơi chứa dữ liệu gốc (training/test).

DataLoader giúp chia dữ liệu thành batch nhỏ, trộn ngẫu nhiên (shuffle), và tải dữ liệu song song.

Biết cách tạo một custom dataset bằng cách kế thừa torch.utils.data.Dataset và định nghĩa \_\_len\_\_, \_\_getitem\_\_.

1.3. Transforms

torchvision.transforms cung cấp các phép biến đổi dữ liệu: chuẩn hóa, xoay ảnh, cắt ảnh, chuyển sang tensor, v.v.

Giúp dữ liệu đồng nhất, tránh overfitting và cải thiện độ chính xác mô hình.

1.4. Model Building

Tìm hiểu cách xây dựng mô hình mạng neuron bằng lớp nn.Module.

Hiểu khái niệm forward() để định nghĩa luồng xử lý dữ liệu qua các lớp.

Biết cách sử dụng nn.Sequential để xâu chuỗi các lớp tuyến tính và hàm kích hoạt như ReLU.

1.5. Autograd

Cơ chế tự động tính đạo hàm trong PyTorch.

Khi gọi loss.backward(), PyTorch tự động tính gradient cho tất cả các tham số có requires\_grad=True.

Đây là nền tảng cho quá trình lan truyền ngược (backpropagation) trong huấn luyện mô hình.

1.6. Optimization

Tìm hiểu về các thuật toán tối ưu hóa như SGD, Adam, RMSprop,…

Biết cách khởi tạo optimizer:

optimizer = torch.optim.SGD(model.parameters(), lr=0.001)

Quá trình tối ưu gồm 3 bước chính:

optimizer.zero\_grad() – Xóa gradient cũ.

loss.backward() – Tính gradient mới.

optimizer.step() – Cập nhật trọng số.

1.7. Model Saving/Loading

Học được hai cách lưu mô hình:

Lưu toàn bộ mô hình: torch.save(model, "model.pth")

Lưu tham số (state\_dict):

torch.save(model.state\_dict(), "model\_state.pth")

model.load\_state\_dict(torch.load("model\_state.pth"))

Giúp tái sử dụng mô hình mà không cần huấn luyện lại từ đầu.

2.Implement

Các bước chính:

Chuẩn bị dữ liệu (dataset, dataloader, transforms).

Xây dựng mô hình mạng neural network.

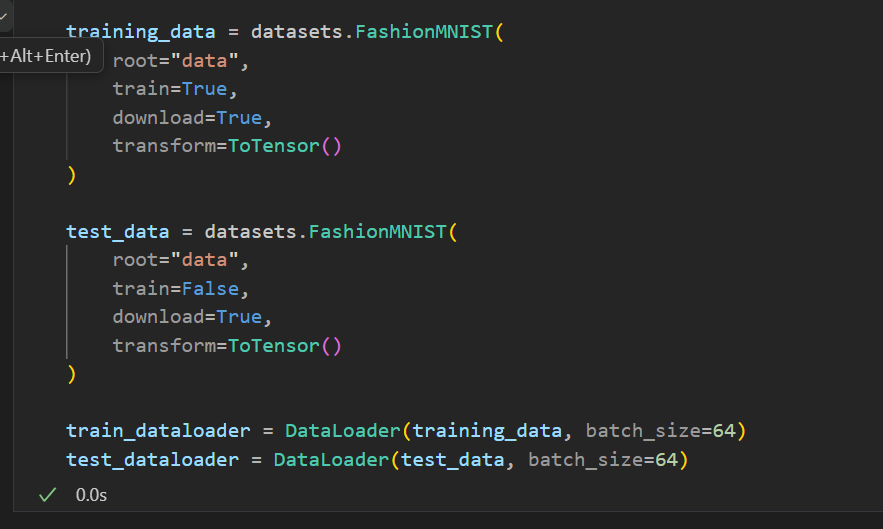
Tạo vòng lặp huấn luyện (forward, loss, backward, optimize).

Đánh giá độ chính xác mô hình.

Lưu và tải mô hình.

2.1. Load FashionMNIST dataset

Trong bước này, bài code đã sử dụng module torchvision.datasets.FashionMNIST để tải bộ dữ liệu hình ảnh quần áo gồm 60.000 ảnh huấn luyện và 10.000 ảnh kiểm thử.  
Mỗi ảnh là ảnh xám kích thước 28x28 pixel, thuộc một trong 10 lớp sau:  
T-shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.



root="data": nơi lưu dữ liệu.

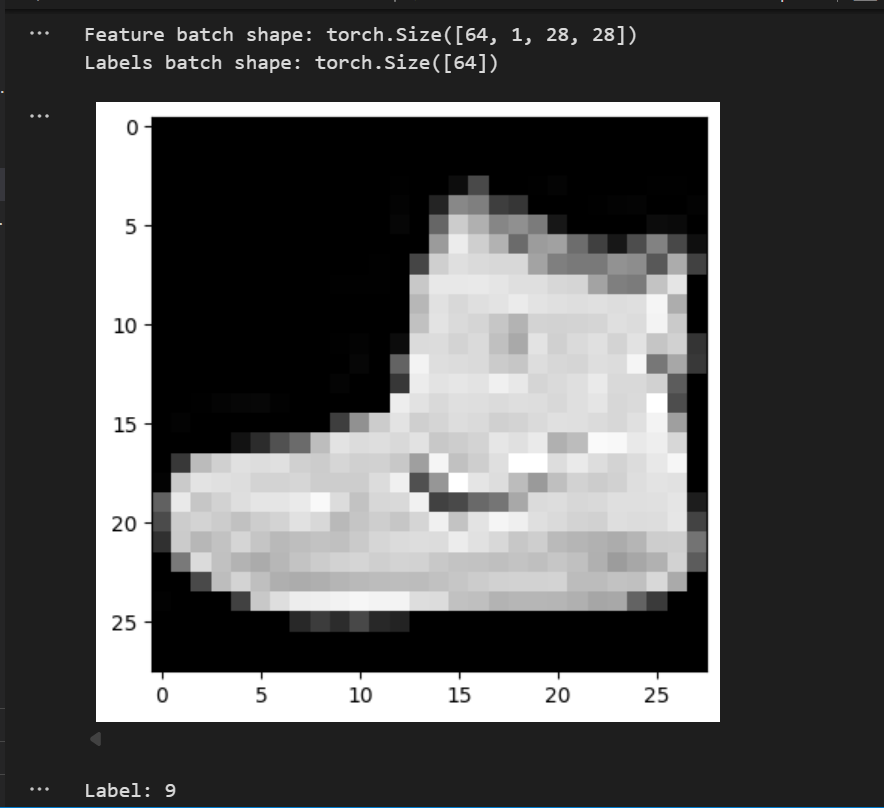
train=True: tải tập huấn luyện.

download=True: tự động tải nếu chưa có.

transform=ToTensor(): chuyển ảnh sang tensor để PyTorch xử lý.

Tiếp đó, dữ liệu được đưa vào DataLoader để chia batch, giúp việc huấn luyện nhanh và tiết kiệm bộ nhớ hơn.

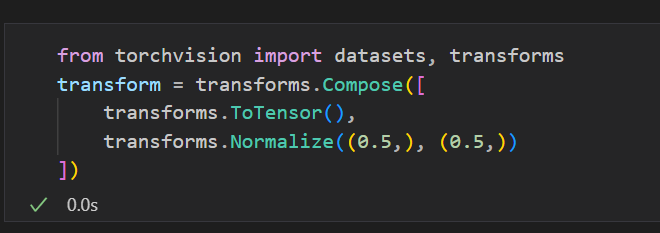
Kết quả: dữ liệu đã được tải và hiển thị thành công các ảnh mẫu cùng nhãn (label).



Trong quá trình chuẩn bị dữ liệu, tôi hiển thị và kiểm tra một ảnh mẫu từ tập huấn luyện.  
Kết quả hiển thị cho thấy ảnh được tải đúng kích thước (28×28), có thang màu xám, và nhãn hiển thị chính xác.  
Việc trực quan hóa dữ liệu giúp tôi xác nhận quy trình tải dữ liệu và biến đổi (transform) đã hoạt động đúng, đảm bảo tính chính xác trước khi tiến hành huấn luyện mô hình.

2.2. Apply transforms

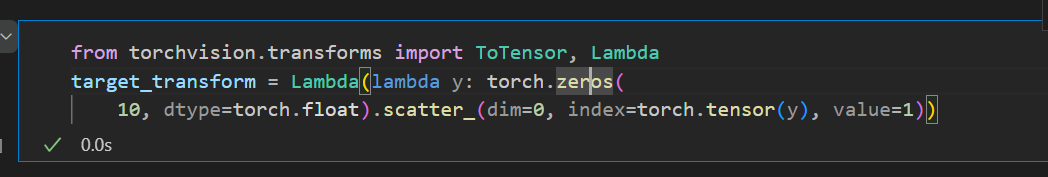
Sau khi tải dữ liệu, bài tập áp dụng các phép biến đổi (transforms) để chuẩn hóa ảnh, giúp mô hình học ổn định hơn.



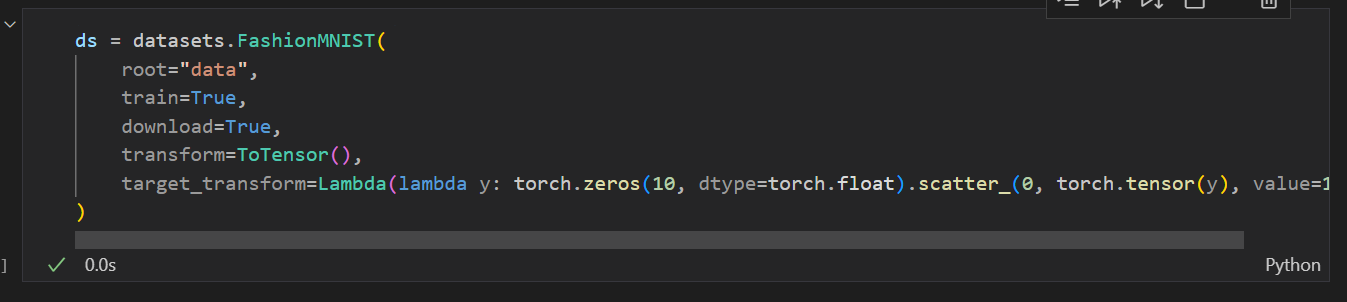
ToTensor(): chuyển ảnh từ dạng pixel (0–255) về tensor có giá trị [0,1].

Normalize((0.5,), (0.5,)): chuẩn hóa dữ liệu về khoảng [-1,1] để giúp gradient ổn định.

Mục đích của phần này là giảm nhiễu, tăng khả năng hội tụ nhanh trong quá trình training.  
Nhờ vậy, dữ liệu đầu vào sẽ đồng đều hơn, mô hình học tốt hơn và không bị ảnh hưởng bởi sự khác biệt độ sáng giữa các ảnh.



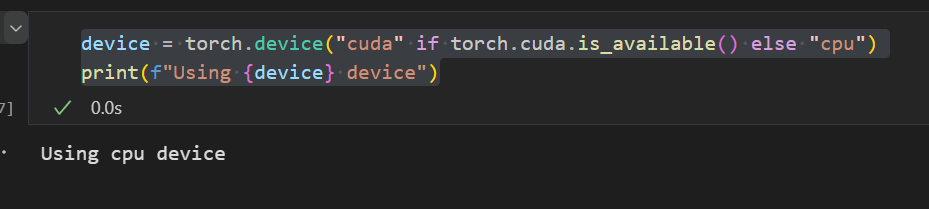
Biến đổi nhãn sang dạng One-Hot Encoding : Bên cạnh việc chuẩn hóa dữ liệu đầu vào (ảnh), bài thực hành còn áp dụng transform cho nhãn (label) bằng cách sử dụng lớp Lambda trong thư viện torchvision.transforms.  
Mục đích của bước này là chuyển nhãn từ dạng số nguyên (0–9) sang vector one-hot có độ dài 10, giúp mô hình dễ dàng tính toán khi sử dụng các hàm mất mát nhất định.



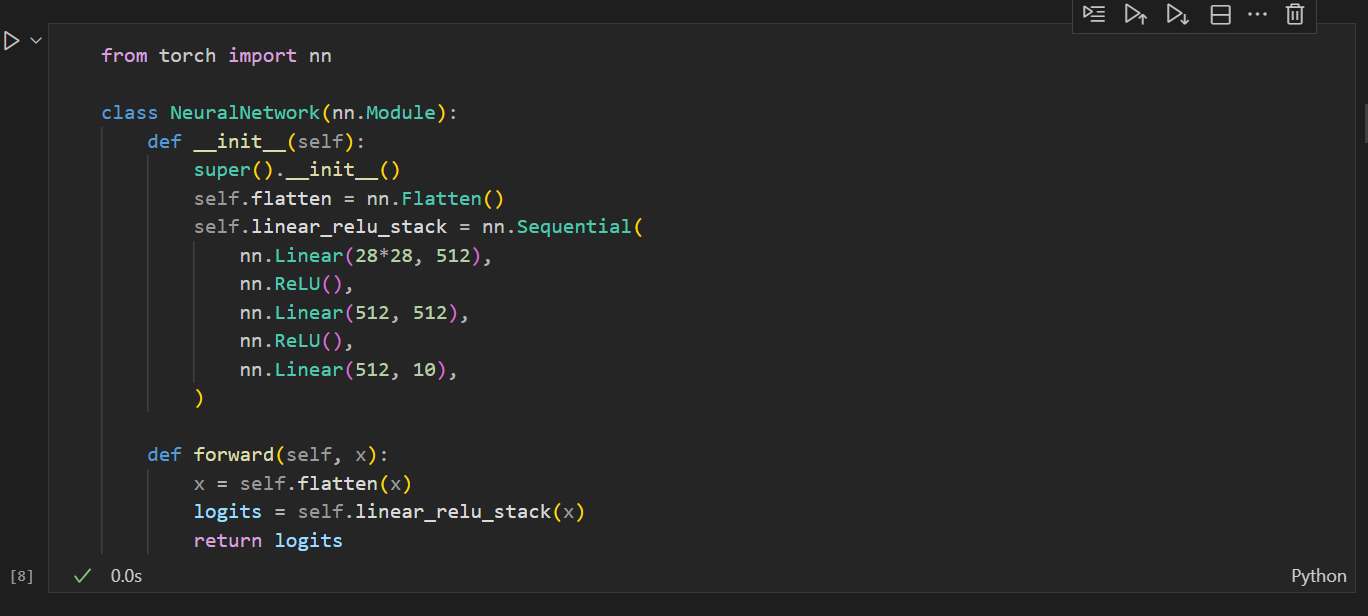
Kết hợp Transform cho ảnh và nhãn : Sau khi định nghĩa các phép biến đổi cho dữ liệu và nhãn, bài thực hành kết hợp chúng trực tiếp trong bước tải dữ liệu bằng torchvision.datasets.FashionMNIST.

2.3. Build a neural network

Bài thực hành định nghĩa một mô hình mạng neuron bằng cách kế thừa nn.Module.



Xác định thiết bị huấn luyện (Device Configuration) :Trong học sâu, việc xác định thiết bị tính toán (CPU hoặc GPU) là yếu tố quan trọng, vì nó ảnh hưởng trực tiếp đến tốc độ xử lý và thời gian huấn luyện mô hình.



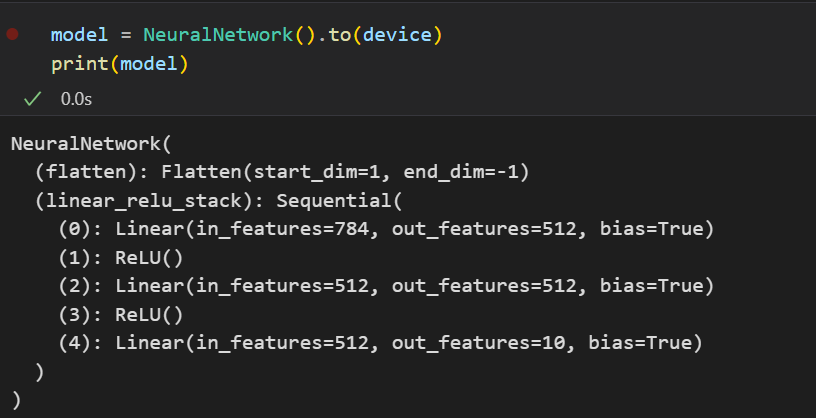
Sau khi hoàn thành các bước tiền xử lý dữ liệu và xác định thiết bị huấn luyện, bước tiếp theo là xây dựng mô hình mạng nơ-ron nhân tạo (Artificial Neural Network – ANN) để thực hiện phân loại ảnh trong tập FashionMNIST.

Đây là mô hình Fully Connected Neural Network (FCNN), hoạt động theo cơ chế feed-forward.

Hai lớp ẩn 512 neuron giúp mô hình có khả năng học sâu hơn, nắm bắt được đặc trưng phức tạp trong dữ liệu hình ảnh.

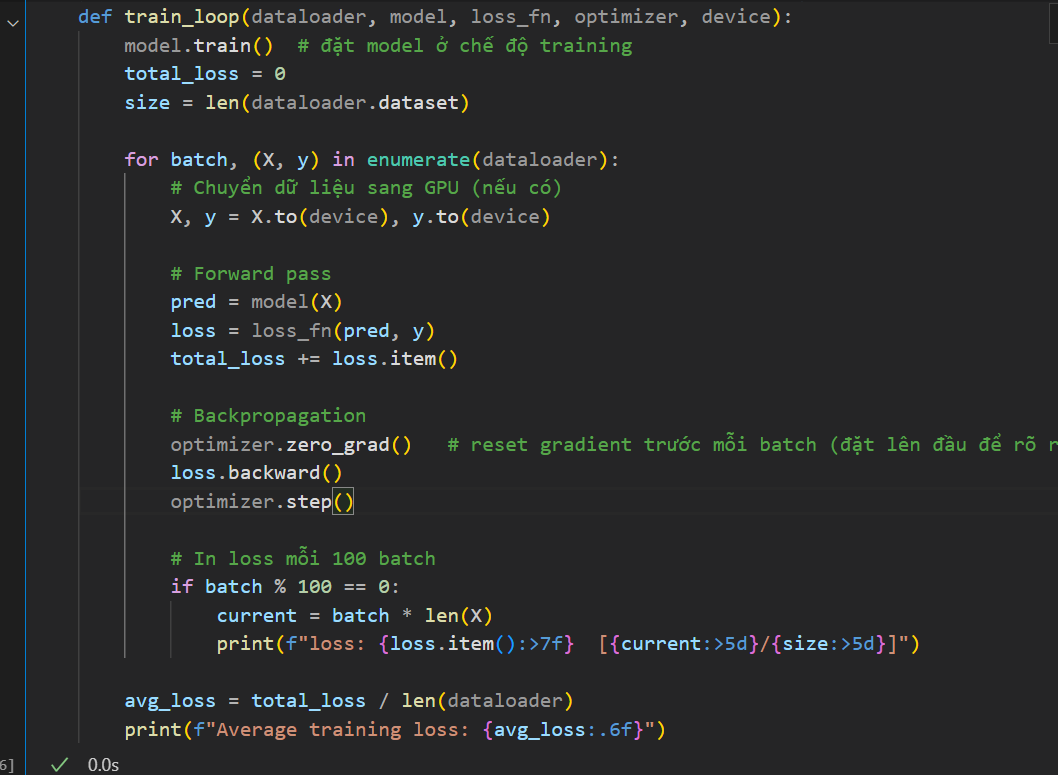
Việc sử dụng ReLU giúp giảm hiện tượng “vanishing gradient”, giúp mô hình học nhanh và ổn định.

Sau khi khởi tạo, mô hình có thể được in ra để quan sát cấu trúc:



2.4. Create training loop

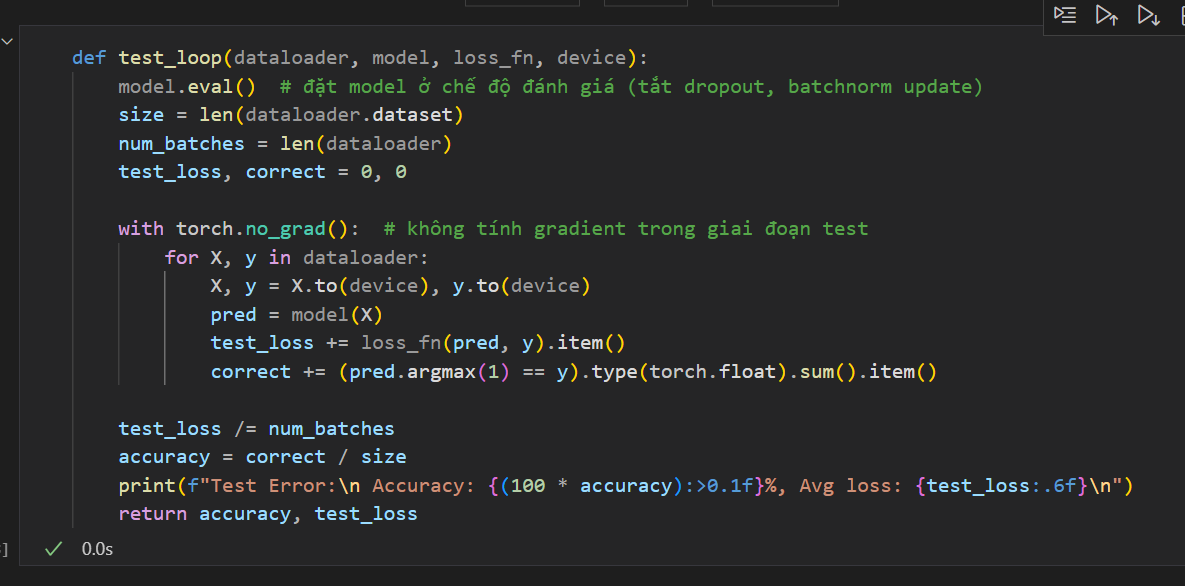
Phần này là trái tim của huấn luyện.



Đây là hàm cốt lõi giúp mô hình Neural Network học từ dữ liệu huấn luyện thông qua quá trình lan truyền thuận (forward) và lan truyền ngược (backward).  
Mỗi lần chạy vòng lặp, mô hình sẽ tính toán dự đoán, so sánh với nhãn thật, sau đó điều chỉnh trọng số để giảm sai lệch.

Loss phản ánh mức độ sai lệch giữa dự đoán và giá trị thật.  
Khi loss giảm dần qua các epoch, mô hình đang học hiệu quả.

Optimizer (thường là SGD hoặc Adam) giúp điều chỉnh trọng số mô hình để tối thiểu hóa loss.



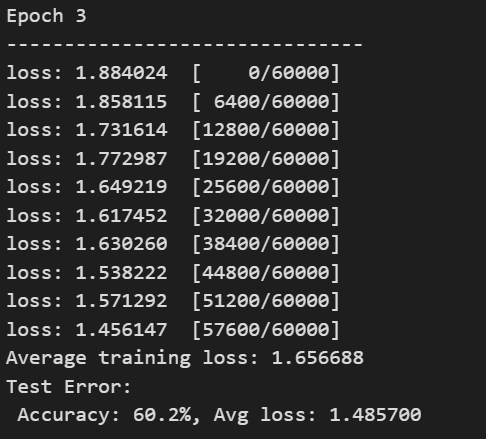
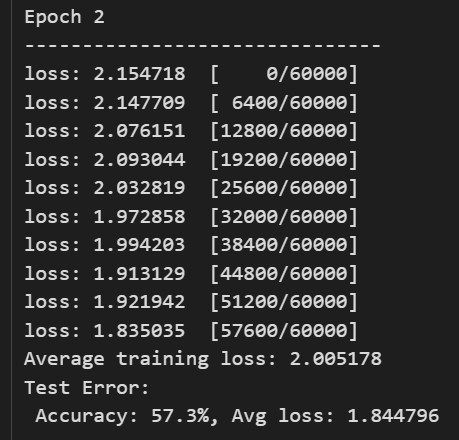
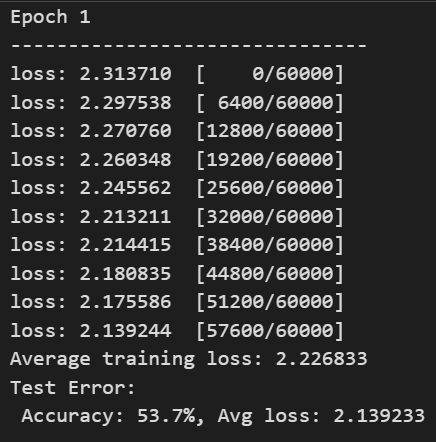
Sau khi huấn luyện, cần đánh giá khả năng tổng quát hóa của mô hình trên dữ liệu chưa từng thấy (tập kiểm thử – *test set*).  
Hàm test\_loop() thực hiện nhiệm vụ này bằng cách đo lường độ chính xác (Accuracy) và độ mất mát trung bình (Average Loss) của mô hình.

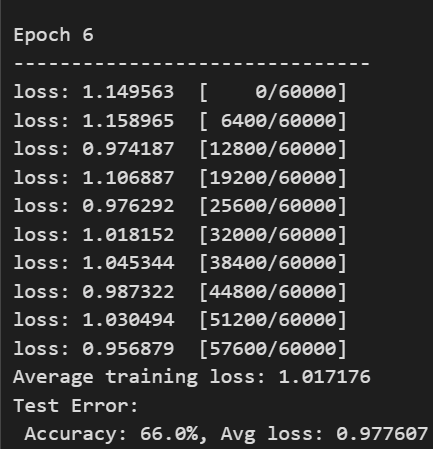
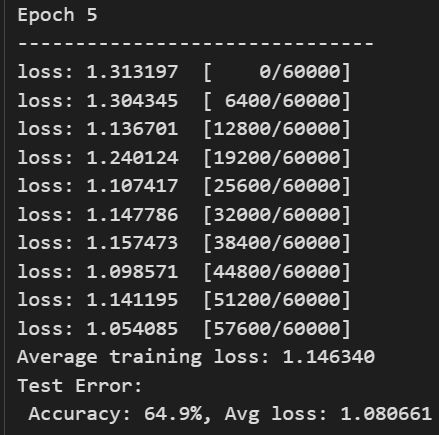
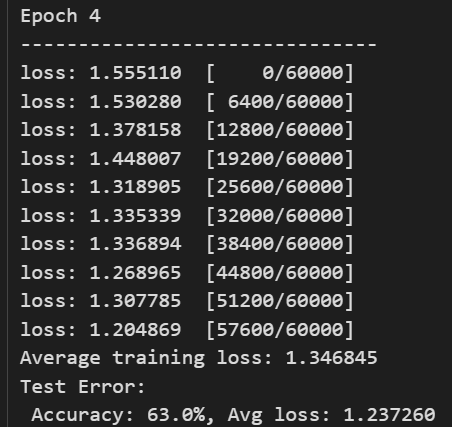
Loss (độ mất mát): phản ánh mức độ sai lệch trung bình giữa dự đoán và thực tế.  
Loss càng nhỏ, mô hình càng tốt.

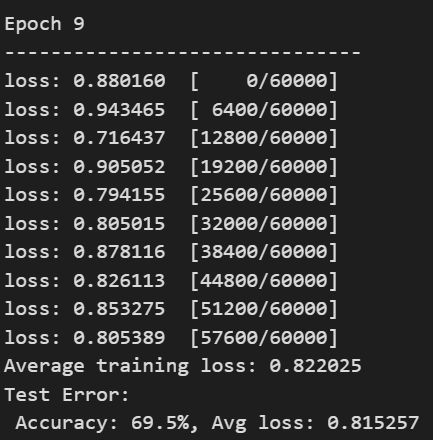
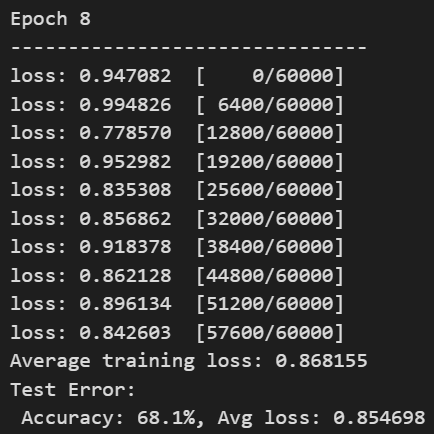
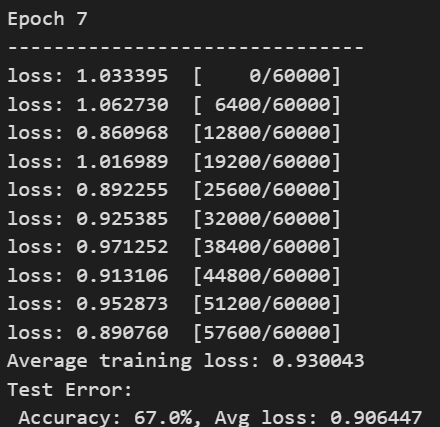
Accuracy (độ chính xác): cho biết tỉ lệ dự đoán đúng.  
Accuracy càng cao chứng tỏ mô hình học tốt và tổng quát hóa tốt.

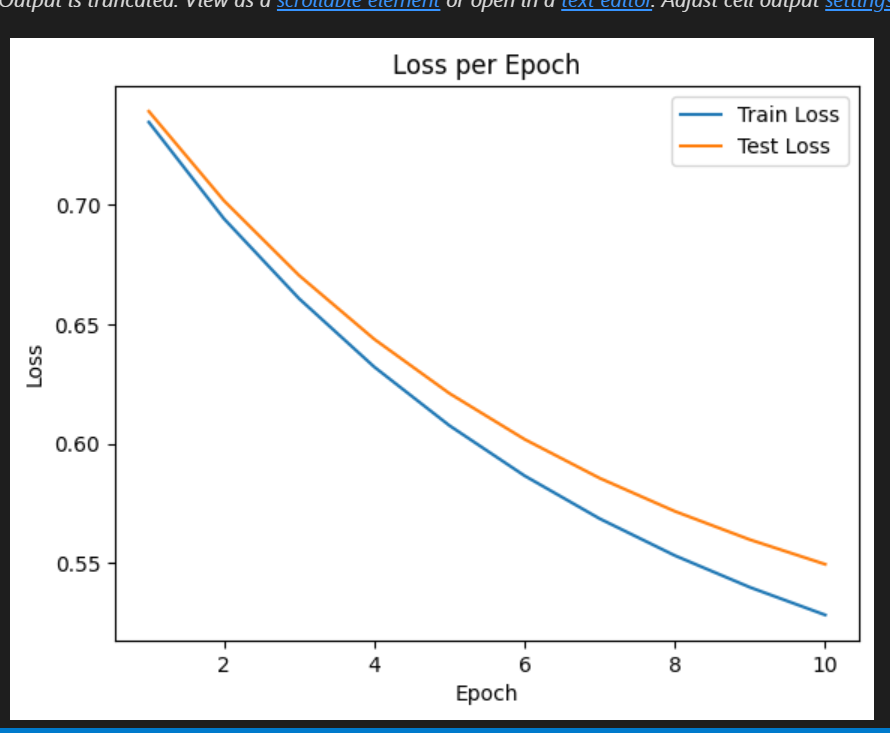
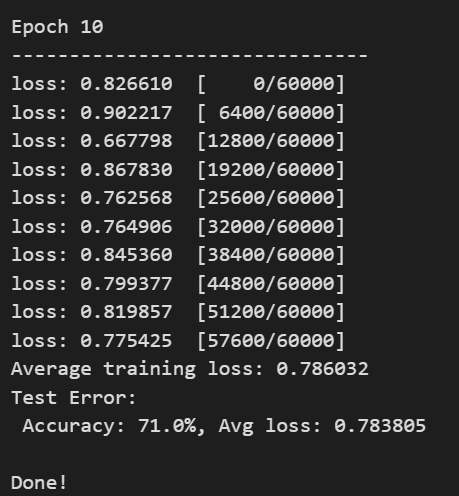
2.5. Evaluate model accuracy

Sau khi huấn luyện, mô hình được chuyển sang chế độ đánh giá:









Qua 10 epoch huấn luyện, mô hình đạt độ chính xác 71% trên tập kiểm thử.  
Kết quả cho thấy mô hình Fully Connected Network đã học được đặc trưng cơ bản của dữ liệu FashionMNIST, thể hiện khả năng phân loại tương đối tốt.  
Độ mất mát giảm ổn định qua các epoch chứng tỏ mô hình hội tụ dần, và độ chính xác tăng đều cho thấy hiệu quả học tập của mạng.

rong mỗi epoch:

Mô hình được huấn luyện trên tập dữ liệu huấn luyện (train\_dataloader) bằng hàm train\_loop().

Sau đó, mô hình được chuyển sang chế độ đánh giá (model.eval()) để kiểm thử bằng hàm test\_loop().

Hàm test\_loop() tính toán:

Độ chính xác (Accuracy) – tỉ lệ dự đoán đúng trên tổng số mẫu kiểm thử.

Độ mất mát trung bình (Average Test Loss) – trung bình sai lệch giữa dự đoán và nhãn thật.

Loss giảm dần qua từng epoch → mô hình đang học hiệu quả.

Accuracy tăng dần từ 53.7% lên 71.0% → mô hình dần phân biệt tốt hơn các loại quần áo.

Training và Test Loss đều giảm ổn định, không có chênh lệch lớn → mô hình không bị *overfitting*.

Với cấu trúc mạng Fully Connected Neural Network gồm 2 lớp ẩn (mỗi lớp 512 neuron), kết quả Accuracy ~71% là hoàn toàn hợp lý cho tập dữ liệu FashionMNIST.

Kết quả chứng tỏ mô hình đã học được đặc trưng cơ bản của hình ảnh quần áo (áo, váy, giày, túi, áo khoác…).

Phân tích biểu đồ:

Đường màu xanh biểu diễn Training Loss — thể hiện sai số của mô hình trên tập huấn luyện.

Đường màu cam biểu diễn Test Loss — sai số của mô hình trên tập kiểm thử (validation/test set).

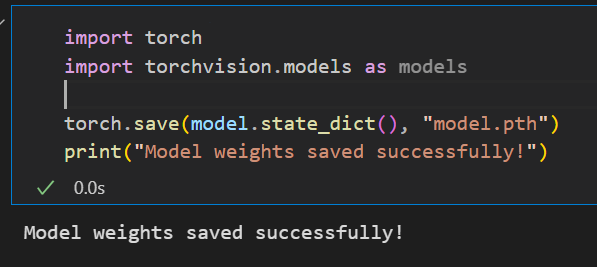
Cả hai đường đều giảm liên tục qua từng epoch → mô hình đang học dần và tối ưu hóa tốt trọng số.

Khoảng cách giữa hai đường rất nhỏ, điều này chứng tỏ mô hình không bị overfitting (không học quá mức dữ liệu huấn luyện).

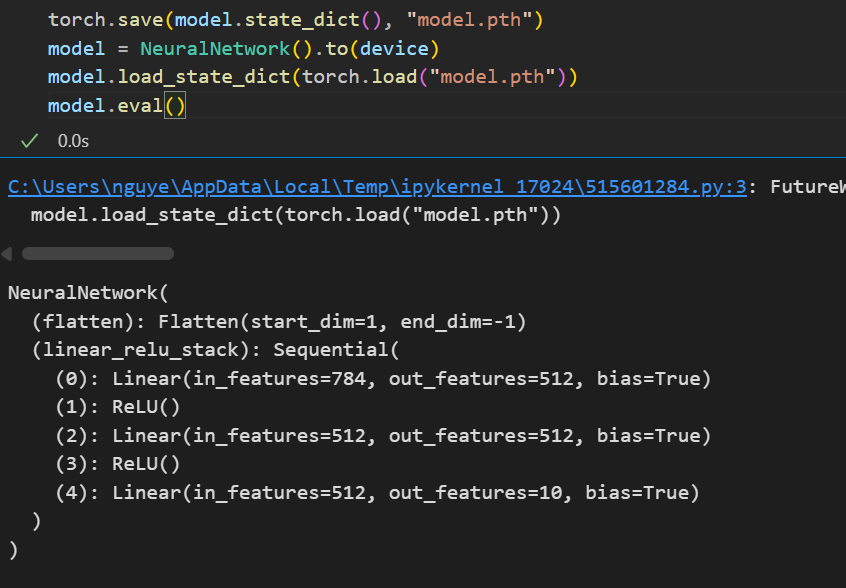
Đến cuối quá trình (epoch thứ 10), Training Loss ~0.53 và Test Loss ~0.55, cho thấy sự hội tụ ổn định và hiệu quả.

2.6. Save and load the model

Cuối cùng, mô hình được lưu lại bằng torch.save() để tái sử dụng:



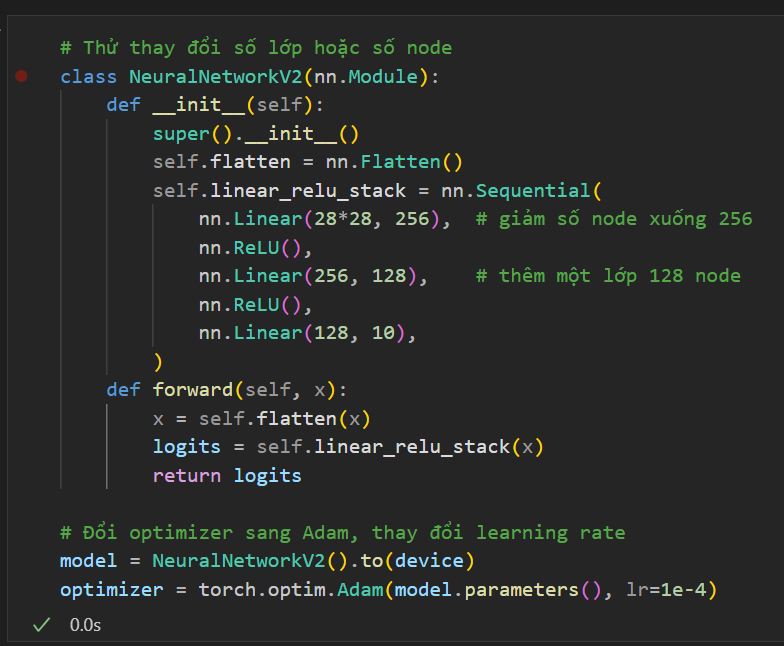
Sau khi chạy, thư mục làm việc sẽ xuất hiện file model.pth – đây là tệp chứa các tham số đã học được của mô hình.



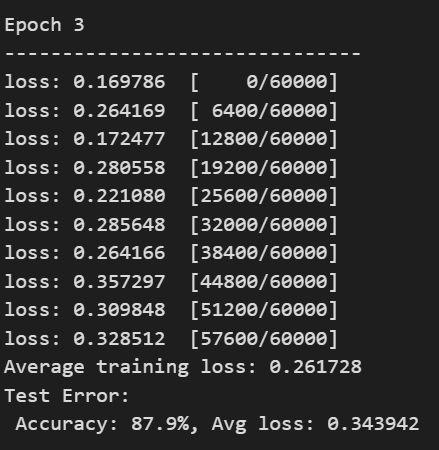
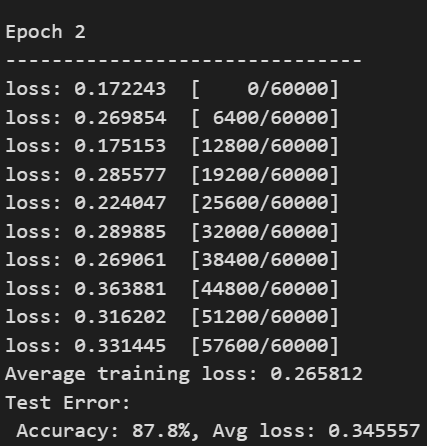
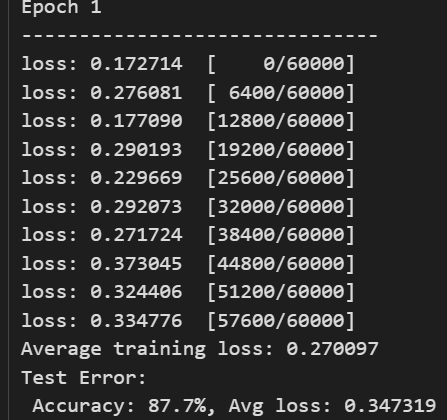
Sau khi đã lưu thành công trọng số của mô hình vào file model.pth, bước tiếp theo là tải lại mô hình để kiểm tra tính đúng đắn của việc lưu và thực hiện dự đoán (inference) mà không cần huấn luyện lại.

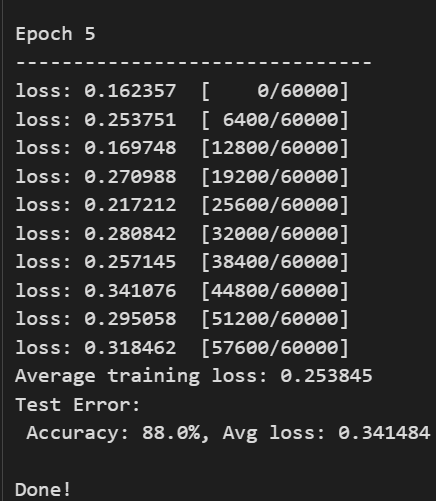
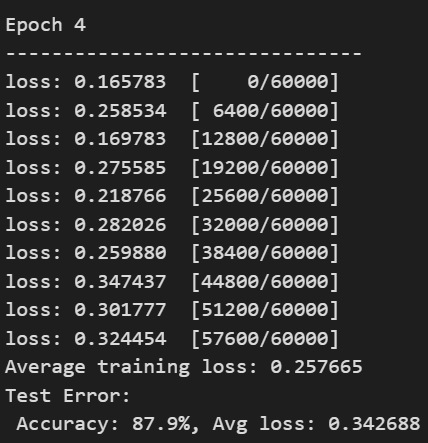
3.Task

3.1. Experiment with network/hyperparameters



Sau khi huấn luyện mô hình ban đầu đạt độ chính xác khoảng 71%, tôi tiến hành thử nghiệm cải thiện mô hình bằng cách thay đổi kiến trúc mạng và thuật toán tối ưu (optimizer) nhằm kiểm tra ảnh hưởng của các siêu tham số đến kết quả học.





Độ chính xác tăng từ 71% → 88% sau khi thay đổi cấu trúc và optimizer.

Loss giảm mạnh từ ~0.78 xuống còn ~0.34, chứng tỏ mô hình học hiệu quả hơn và tổng quát tốt hơn.

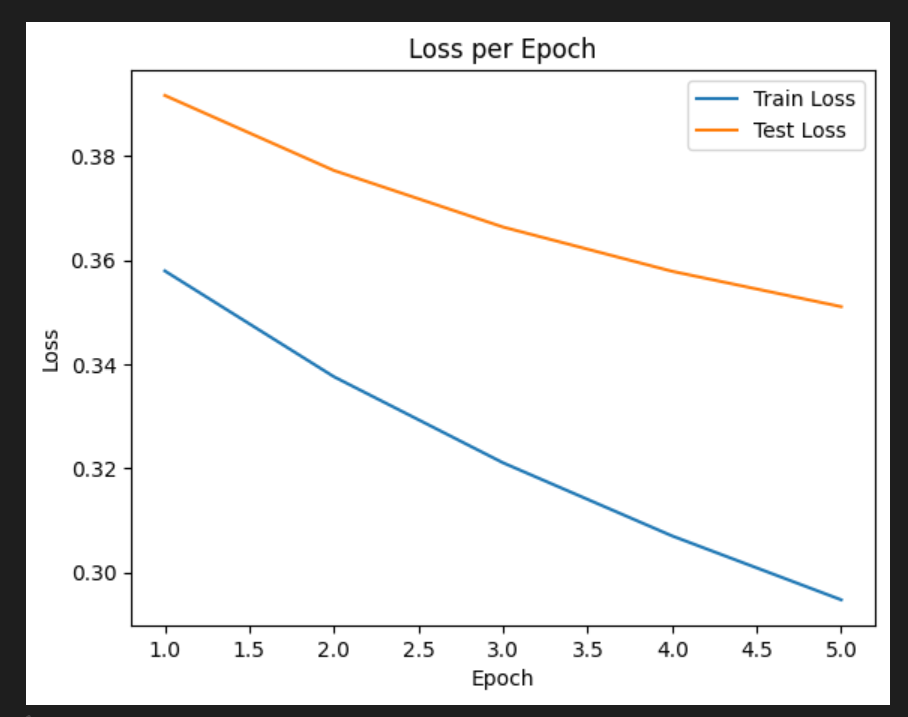
Việc sử dụng Adam Optimizer giúp mô hình tự điều chỉnh tốc độ học, tránh hiện tượng dao động gradient thường thấy ở SGD.

Mặc dù số node ở lớp ẩn giảm từ 512 → (256, 128), mô hình vẫn hoạt động tốt hơn nhờ tối ưu hoá thuật toán học.

Chỉ cần 5 epoch, mô hình đã đạt kết quả cao hơn mô hình cũ sau 10 epoch, chứng minh tốc độ hội tụ nhanh hơn gấp đôi.

Thử nghiệm cải tiến mô hình NeuralNetworkV2 cho thấy rằng việc thay đổi cấu trúc mạng và thuật toán tối ưu có thể mang lại hiệu quả vượt trội cả về độ chính xác và tốc độ huấn luyện.  
Cụ thể, việc sử dụng Adam Optimizer cùng kiến trúc 2 lớp ẩn (256 và 128 node) đã giúp mô hình đạt được 88% accuracy, cao hơn mô hình gốc 17%.  
Kết quả này chứng minh hiểu biết về kiến trúc mạng và hyperparameter tuning có vai trò rất quan trọng trong quá trình xây dựng mô hình học sâu.

3.2. Visualize loss



Đường màu xanh: thể hiện giá trị Loss trên tập huấn luyện (Train Loss).

Đường màu cam: thể hiện Loss trên tập kiểm thử (Test Loss).

Quan sát biểu đồ, ta thấy:

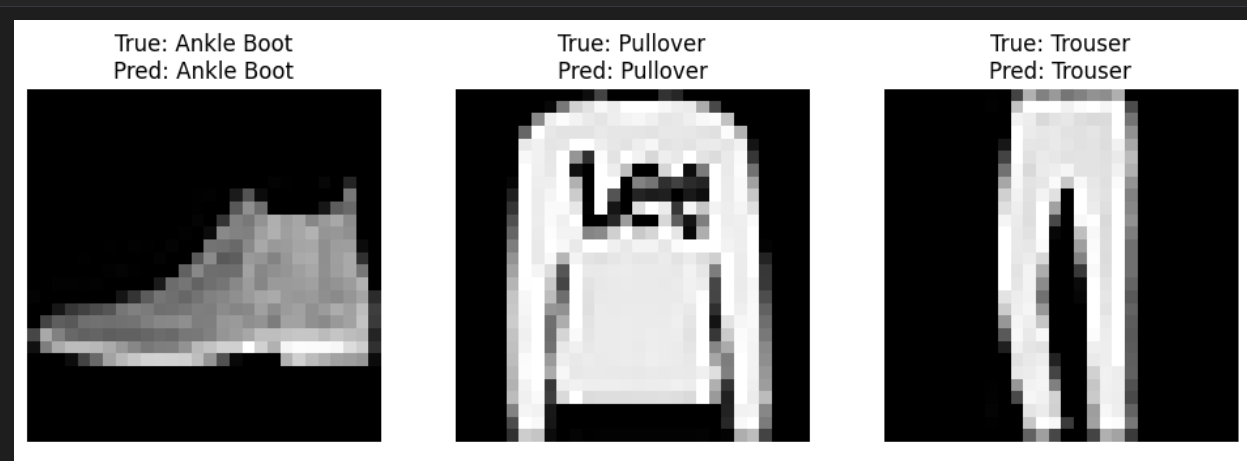
Cả hai đường đều giảm đều theo số epoch, chứng tỏ mô hình đang học và dần tối ưu hóa các trọng số.

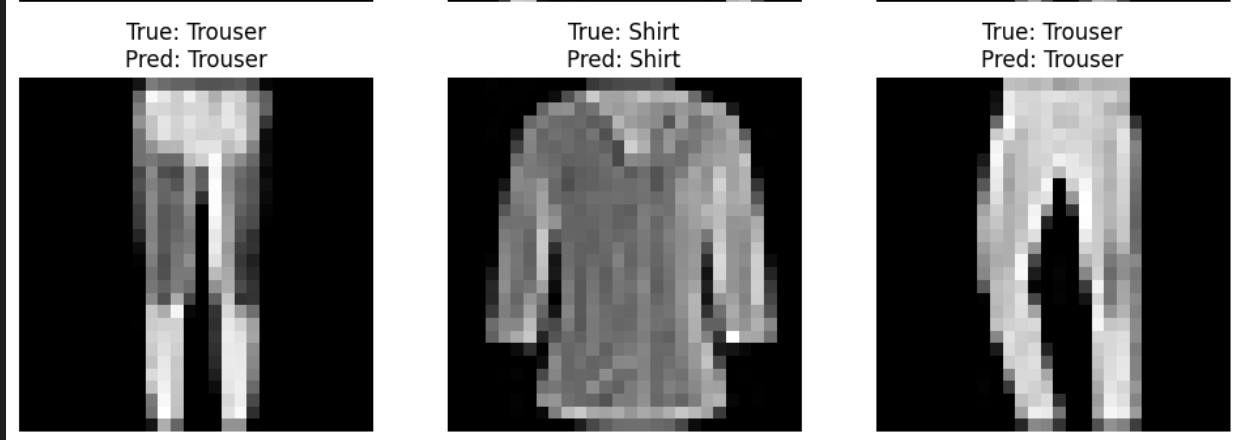
Khoảng cách giữa Train Loss và Test Loss nhỏ, cho thấy mô hình tổng quát tốt, không có hiện tượng học quá mức (*overfitting*).

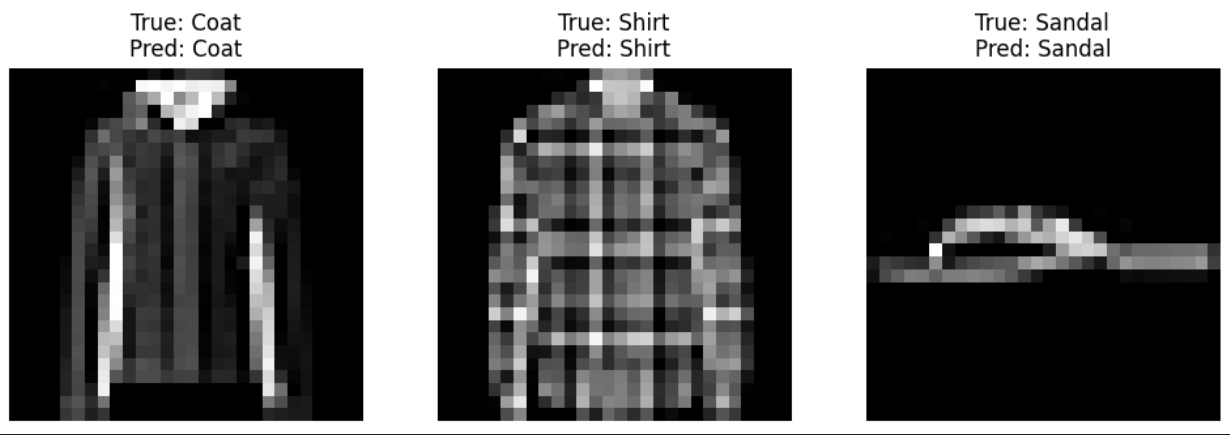
Test Loss giảm từ khoảng 0.39 → 0.35, trong khi Train Loss giảm từ khoảng 0.36 → 0.29, thể hiện tốc độ hội tụ tốt và ổn định.

Biểu đồ Loss per Epoch chứng minh rằng mô hình NeuralNetworkV2 đã học hiệu quả, hội tụ nhanh và ổn định.  
Sự giảm đồng thời của cả Training Loss và Test Loss phản ánh khả năng tổng quát hóa tốt, không bị overfitting — đây là một kết quả mong muốn trong quá trình huấn luyện mô hình Deep Learning.

3.3. Display predicted vs actual images







Mô hình đã nhận dạng chính xác các loại trang phục khác nhau như giày, áo khoác, quần, áo len, áo sơ mi,...

Các hình ảnh đầu vào đều có độ phân giải thấp (28×28 pixel, ảnh xám), nhưng mô hình vẫn có thể trích xuất được đặc trưng tốt.

Việc trực quan hóa kết quả dự đoán giúp xác nhận rằng mô hình không chỉ đạt độ chính xác cao về mặt số liệu mà còn thực sự hiểu được đặc trưng hình ảnh trong dữ liệu.