**EXPERIMENT2**

**1. Mục đích và cách hoạt động của MLP trên Fashion-MNIST**

**Mục đích**

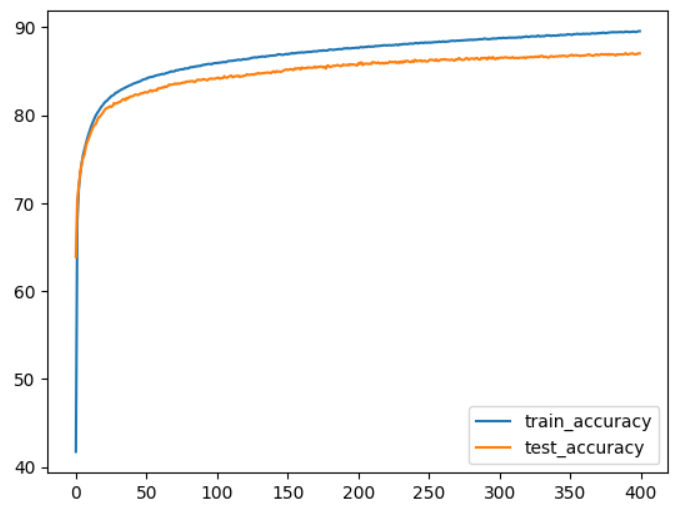
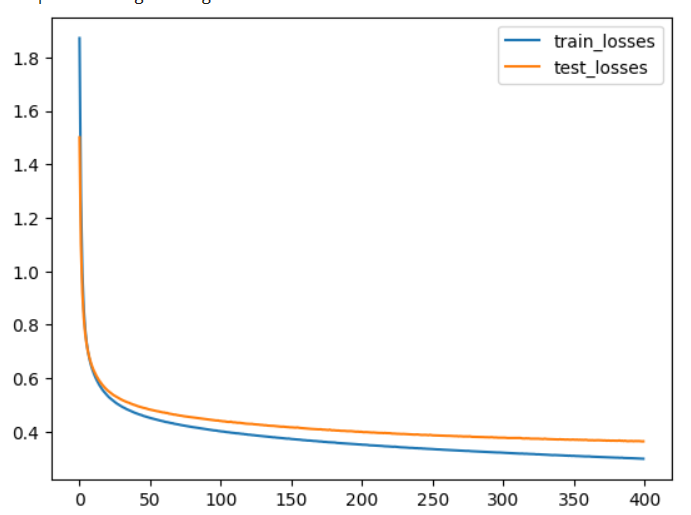
* Mô hình được huấn luyện để phân loại hình ảnh **quần áo, giày dép, túi xách** trong tập dữ liệu **Fashion-MNIST** gồm **10 loại khác nhau**.
* Dữ liệu **Fashion-MNIST** gồm **60.000 ảnh dùng để huấn luyện** và **10.000 ảnh để kiểm tra**, mỗi ảnh có kích thước **28x28 pixels** và được biểu diễn dưới dạng **một vector 784 chiều** (28x28 = 784).
* Mục tiêu của MLP là **học cách nhận dạng đặc trưng** của từng loại sản phẩm thời trang để dự đoán nhãn chính xác.

**Cách hoạt động của MLP**

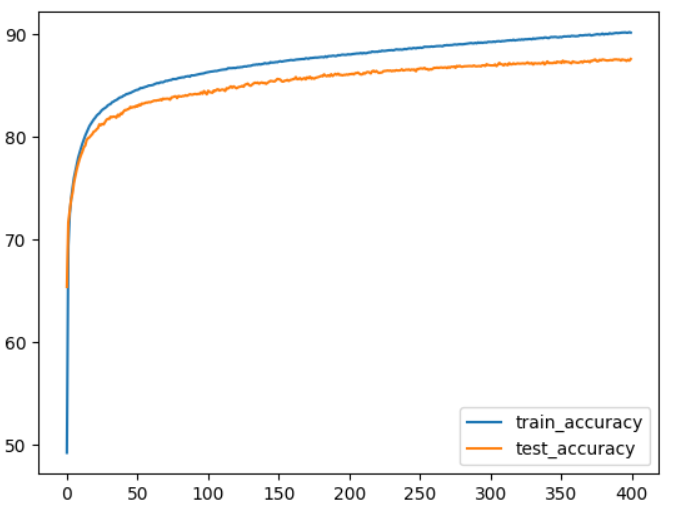
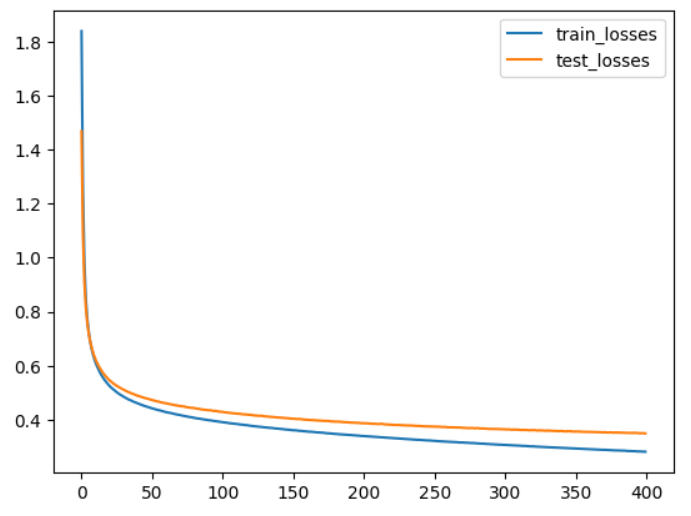
1. **Nhập dữ liệu:**
   * Ảnh 28x28 được chuyển thành một vector 784 phần tử.
2. **Truyền qua các tầng ẩn:**
   * Dữ liệu đi qua **các tầng fully connected (FC)** với số lượng **nodes ẩn khác nhau** (64, 256, 1024).
   * Các node này chứa trọng số (weights) được học trong quá trình huấn luyện.
   * Hàm kích hoạt ReLU (Rectified Linear Unit) được sử dụng để tăng khả năng học phi tuyến.
3. **Tầng đầu ra:**
   * Tầng cuối có **10 neurons** (tương ứng 10 nhãn của Fashion-MNIST).
   * Dùng hàm **Softmax** để chuyển đổi kết quả thành xác suất dự đoán.
4. **Tính hàm mất mát & tối ưu hóa:**
   * Sử dụng **Cross-Entropy Loss** để đo lường sai số.
   * Dùng **Adam hoặc SGD Optimizer** để cập nhật trọng số và giảm lỗi.
5. **Huấn luyện & đánh giá:**
   * Mô hình được huấn luyện trong nhiều epoch (ở bài này, ta sử dụng 400 epochs), sau đó kiểm tra trên tập test để tính độ chính xác.

**2. Phân tích từng mô hình MLP**

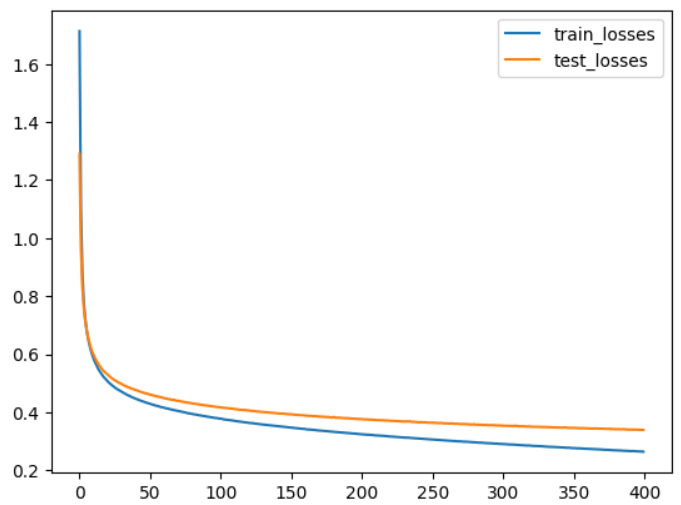
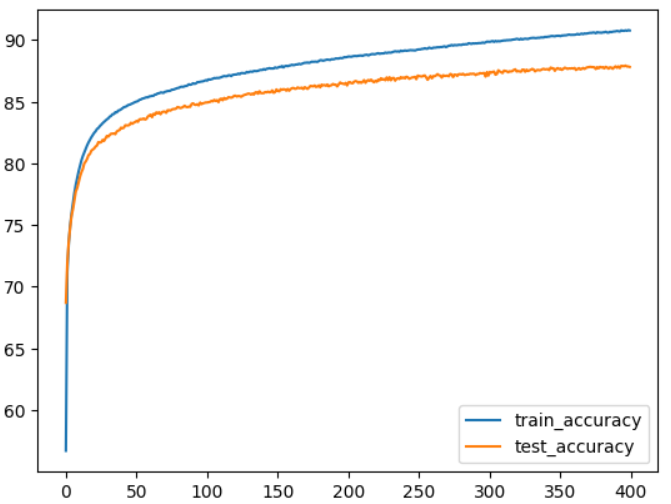
**Phân tích mô hình:**

* **Đồ thị mô hình 64 nodes**

|  |
| --- |
| Epoch [400/400], Loss: 0.2986, Accuracy: 89.57%, Test Loss: 0.3638, Test Accuracy: 87.02% |

* **Đồ thị mô hình 256 nodes**

|  |
| --- |
| Epoch [400/400], Loss: 0.2820, Accuracy: 90.14%, Test Loss: 0.3499, Test Accuracy: 87.57% |

* **Đồ thị mô hình 1024 nodes**

|  |
| --- |
| Epoch [400/400], Loss: 0.2638, Accuracy: 90.78%, Test Loss: 0.3391, Test Accuracy: 87.82% |

**(a) MLP với 64 Nodes**

* **Cấu trúc**:
  + 1 tầng ẩn với **64 nodes**
  + Dùng hàm kích hoạt **ReLU**
* **Ưu điểm**:
  + Kích thước mô hình nhỏ, tốc độ huấn luyện nhanh.
  + Ít bị overfitting do số lượng tham số ít hơn.
* **Nhược điểm**:
  + Dễ bị underfitting do không đủ khả năng học đặc trưng phức tạp của dữ liệu.

**(b) MLP với 256 Nodes**

* **Cấu trúc**:
  + 1 tầng ẩn với **256 nodes**
  + Dùng hàm kích hoạt **ReLU**
* **Ưu điểm**:
  + Cân bằng giữa độ phức tạp và tốc độ huấn luyện.
  + Học đặc trưng tốt hơn so với mô hình 64 nodes.
* **Nhược điểm**:
  + Mất nhiều tài nguyên tính toán hơn mô hình 64 nodes.

**(c) MLP với 1024 Nodes**

* **Cấu trúc**:
  + 1 tầng ẩn với **1024 nodes**
  + Dùng hàm kích hoạt **ReLU**
* **Ưu điểm**:
  + Mô hình mạnh mẽ hơn, có khả năng học nhiều đặc trưng hơn.
  + Độ chính xác thường cao hơn.
* **Nhược điểm**:
  + Dễ bị **overfitting** nếu dữ liệu không đủ lớn.
  + Tốn nhiều bộ nhớ và thời gian huấn luyện.

**3. Hiệu suất của từng mô hình**

**Dự đoán hiệu suất**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Số nodes ẩn** | **Tốc độ huấn luyện** | **Độ chính xác (dự đoán)** | **Rủi ro overfitting** |
| **64 Nodes** | 64 | 🟢 Nhanh nhất | 🔴 Thấp nhất (~80-85%) | 🟢 Ít overfitting |
| **256 Nodes** | 256 | 🟡 Trung bình | 🟡 Trung bình (~85-88%) | 🟡 Cân bằng |
| **1024 Nodes** | 1024 | 🔴 Chậm nhất | 🟢 Cao nhất (~88-92%) | 🔴 Dễ overfitting |

* **MLP 64 nodes**: Huấn luyện nhanh nhưng có thể không đủ mạnh để học hết đặc trưng của Fashion-MNIST.
* **MLP 256 nodes**: Cân bằng giữa tốc độ và độ chính xác.
* **MLP 1024 nodes**: Mạnh nhất nhưng có thể bị overfitting nếu không có kỹ thuật regularization.

**4. Đánh giá và kết luận**

Khi nào nên chọn mỗi mô hình?

🔹 MLP 64 nodes

✅ Khi cần tốc độ nhanh, tài nguyên hạn chế, tránh overfitting.

🚫 Không phù hợp nếu cần độ chính xác cao.

🔹 MLP 256 nodes

✅ Cân bằng giữa hiệu suất và tốc độ, phù hợp với hầu hết ứng dụng.

🚫 Không phải là mô hình mạnh nhất nhưng có độ chính xác tốt.

🔹 MLP 1024 nodes

✅ Khi cần độ chính xác cao nhất.

🚫 Cần nhiều tài nguyên và có thể bị overfitting.