## 1) Tổng quan ngắn

Bài lab này: **\*\*xây một bộ phân loại ảnh\*\* cho quần áo (Fashion-MNIST)** — từ ảnh xám 28×28, phân thành 10 nhãn (áo, quần, giày, v.v.). Mục tiêu thực tế: tự động phân loại ảnh sản phẩm, tiết kiệm công sức tiền nhân sự, áp dụng cho e-commerce. Ngoài việc xây model, lab yêu cầu \*\*tinh chỉnh siêu tham số (HPO)\*\* để tối ưu hiệu suất.

---

## 2) Dữ liệu (Fashion-MNIST) — giải thích dễ hiểu

\* \*\*Nguồn\*\*: bộ dữ liệu của Zalando, thay thế cho MNIST truyền thống nhưng hình ảnh là quần áo.

\* \*\*Kích thước\*\*:

**\* Training: \*\*60.000 ảnh\*\***

**\* Test: \*\*10.000 ảnh\*\***

**\* Mỗi ảnh: \*\*28×28 pixels\*\*, đơn kênh (grayscale) — tức là chỉ mức sáng, không màu.**

**\* \*\*Mục tiêu\*\*: cho mỗi ảnh dự đoán đúng \*\*1 trong 10 lớp\*\*.**

\*\*Ánh xạ nhãn (label → lớp)\*\*

0: T-shirt/top

1: Trouser

2: Pullover

3: Dress

4: Coat

5: Sandal

6: Shirt

7: Sneaker

8: Bag

9: Ankle boot

(Hiểu đơn giản: mỗi ảnh là một tấm hình nhỏ của một món đồ — model học để gán vào đúng “kệ hàng”.)

---

## 3) Tại sao bài toán này quan trọng (Business impact) — giải đơn giản

E-commerce có hàng ngàn ảnh sản phẩm. Nếu phân loại thủ công, tốn công, lỗi nhiều. Một hệ thống tự động:

\* Giúp \*\*gắn nhãn hàng loạt\*\* (ví dụ: tách áo vs quần vs giày).

\* \*\*Tự động hóa\*\* hiển thị/lọc sản phẩm, search, đề xuất.

\* Tiết kiệm thời gian và chi phí, cải thiện UX trên website/app.

---

## 4) Quy trình làm lab — từng bước dễ hiểu

### Bước A — Chuẩn bị dữ liệu

1. \*\*Load\*\* dataset: `tf.keras.datasets.fashion\_mnist.load\_data()`

2. \*\*Khám phá\*\*: xem vài ảnh, kiểm tra phân bố nhãn (cân bằng không?).

3. \*\*Tiền xử lý cơ bản\*\*:

\* Scale pixel về \[0,1]: `x = x.astype('float32') / 255.0`

\* Thêm chiều channel nếu cần: `x = x.reshape(-1,28,28,1)`

\* Chuyển nhãn nếu model yêu cầu one-hot: `to\_categorical(y)` (hoặc giữ integer + sparse loss)

4. (Tuỳ chọn) \*\*Data augmentation\*\*: xoay nhỏ, dịch, cắt — tăng khả năng tổng quát hoá.

### Bước B — Xây model (baseline)

\* \*\*Mô hình thường dùng\*\*: CNN đơn giản

\* Conv2D (filters nhỏ như 32) → ReLU → MaxPool

\* Conv2D (64) → ReLU → MaxPool

\* Flatten → Dense (64) → Dropout → Dense(10, softmax)

Giải thích ngắn: Conv layer “nhìn” các pattern (viền, kết cấu). Pooling giảm kích thước, Dense học quyết định cuối cùng.

### Bước C — Train & Evaluate

\* \*\*Compile\*\*: loss = `sparse\_categorical\_crossentropy`, optimizer = `Adam`, metric = `accuracy`.

\* \*\*Fit\*\*: train/validation split, dùng EarlyStopping để tránh overfit.

\* \*\*Đánh giá\*\*: accuracy trên test, confusion matrix để thấy lớp bị nhầm nhiều.

### Bước D — Hyperparameter Tuning (HPO)

\* Mục tiêu: tìm tổ hợp tốt nhất (learning rate, batch size, số filter, dropout, optimizer...).

\* Chiến lược thường dùng: Random/Hyperband/KerasTuner/Bayesian.

\* Sau tìm best config → \*\*retrain\*\* trên toàn train set (full budget) → đánh giá final.

---

## 5) Gợi ý kiến trúc & giải thích từng thành phần (dễ hiểu)

\* \*\*Input (28×28×1)\*\*: ảnh xám nhỏ.

\* \*\*Conv2D(filters=32, kernel=3×3)\*\*: quét ảnh, tìm cạnh/họa tiết nhỏ.

\* \*\*ReLU\*\*: activation — biến đầu ra phi tuyến.

\* \*\*MaxPooling(2×2)\*\*: co gọn thông tin, giữ đặc trưng quan trọng.

\* \*\*Conv2D(filters=64)\*\*: nắm pattern phức tạp hơn.

\* \*\*Flatten → Dense(64)\*\*: chuyển đặc trưng sang vector để phân loại.

\* \*\*Dropout(0.3)\*\*: tắt ngẫu nhiên neuron để giảm overfitting.

\* \*\*Dense(10, softmax)\*\*: đầu ra xác suất cho 10 lớp.

Analogy: Conv = đầu dò chi tiết (mắt tinh), Pooling = tóm gọn thông tin, Dense = bộ não quyết định nhãn.

---

## 6) Hyperparameters đáng tune (ưu tiên)

\* \*\*Learning rate (lr)\*\*: quyết định tốc độ học — tune trên log-scale (1e-4 → 1e-2).

\* \*\*Batch size\*\*: {32, 64, 128} ảnh một lần.

\* \*\*Số filters\*\*: tầng 1: {32,64}, tầng 2: {64,128}.

\* \*\*Dense units\*\*: {64,128,256}.

\* \*\*Dropout\*\*: 0.0 → 0.5.

\* \*\*Optimizer\*\*: Adam / SGD + momentum.

\* \*\*Augmentation params\*\* (nếu dùng).

---

## 7) Các lỗi hay gặp & cách tránh

\* \*\*Không chuẩn hóa pixel\*\* → training chậm/không ổn. → \*Luôn chia 255.\*

\* \*\*Overfitting\*\* (train acc cao, val acc thấp) → tăng dropout, giảm model capacity, thêm augmentation.

\* \*\*Search space quá lớn\*\* → HPO tốn thời gian → giới hạn phạm vi, dùng Random + HyperBand trước.

\* \*\*Quên retrain best model\*\* → báo cáo sai lệch → retrain final model trên full data.

---

## 8) Ví dụ code nhỏ (copy vào .ipynb nhanh)

```python

# 1) Load & prep

import tensorflow as tf

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.fashion\_mnist.load\_data()

x\_train = x\_train.astype('float32')/255.0

x\_test = x\_test.astype('float32')/255.0

x\_train = x\_train.reshape(-1,28,28,1)

x\_test = x\_test.reshape(-1,28,28,1)

# 2) Simple CNN

from tensorflow.keras import layers, models

model = models.Sequential([

layers.Conv2D(32, (3,3), activation='relu', input\_shape=(28,28,1)),

layers.MaxPooling2D((2,2)),

layers.Conv2D(64, (3,3), activation='relu'),

layers.MaxPooling2D((2,2)),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dropout(0.3),

layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.summary()

# 3) Train (baseline)

from tensorflow.keras.callbacks import EarlyStopping

es = EarlyStopping(monitor='val\_accuracy', patience=3, restore\_best\_weights=True)

model.fit(x\_train, y\_train, validation\_split=0.1, epochs=20, batch\_size=64, callbacks=[es])

# 4) Evaluate

model.evaluate(x\_test, y\_test)

```

---

## 9) Checklist ngắn trước khi nộp lab

\* [ ] Đã show sample ảnh + phân bố nhãn.

\* [ ] Đã preprocess & giải thích lý do.

\* [ ] Đã xây baseline model + hiển thị `model.summary()`.

\* [ ] Đã train + show learning curves (loss/acc).

\* [ ] Đã thực hiện HPO (ít nhất Random/Hyperband) và nêu best params.

\* [ ] Đã retrain best config và báo cáo test accuracy + confusion matrix.

\* [ ] Đã viết ngắn phần Business Impact + limitations.

---

Nếu bạn muốn — mình sẽ:

\* \*\*(A)\*\* chèn đoạn code HPO (Keras Tuner) hoàn chỉnh vào notebook để bạn paste & chạy, hoặc

\* \*\*(B)\*\* chèn luôn một cell chạy training + vẽ learning curve (nhưng cần máy bạn chạy), hoặc

\* \*\*(C)\*\* tạo phần báo cáo (1 trang) giải thích lab để dán vào notebook.

Bạn chọn A/B/C. Mình làm ngay phần code Keras Tuner nếu bạn chọn A.







