■ Fashion MNIST: Key Concepts & Data Understanding Guide

This guide walks through the essential machine learning and computer vision concepts demonstrated in the Fashion MNIST project. It's designed to help students and practitioners understand the data pipeline, modeling logic, and preprocessing challenges for real-world image classification tasks.

■ Image Classification Context

Fashion MNIST is a classic image classification dataset with grayscale images of clothes. The goal is to predict which category (e.g., T-shirt, Bag, Sneaker) a given image belongs to. This mirrors real-world use cases like smart checkout systems, visual search, and trend detection.

■ Dataset Overview

- Each image is 28x28 pixels in grayscale (1 channel) - There are 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot - The dataset contains 60,000 training and 10,000 test images - Labels are integers from 0 to 9 representing clothing types

■ Image Preprocessing Pipeline

1. **Grayscale Conversion**: Reduces complexity by working with one channel instead of three (RGB). 2. **Resizing**: All input images are resized to 28x28 pixels to match the trained model's input. 3. **Normalization**: Pixel values (0–255) are scaled to 0–1 to stabilize neural network training. 4. **Reshaping**: Depending on the model architecture, images are reshaped to either (1, 28, 28) for feedforward networks or (1, 28, 28, 1) for CNNs.

■ Core Concepts to Understand

- **Model Input Shape**: The shape of the input image must match the model's expected shape. Flattened arrays ≠ image tensors. - **Softmax Output**: Neural networks use softmax to predict class probabilities. Highest value = predicted class. - **Class Mapping**: The model returns an index (0–9); you must map that to human-readable class labels. - **Inference Pipeline**: Preprocessing → Predict → Convert output → Display prediction + confidence. - **Deployment Checkpoints**: Always verify that the preprocessing in the app matches how the model was trained.

■■ Common Pitfalls

- Mismatched shapes: Models expecting 3D (28, 28) input will break on flattened arrays - Forgetting to normalize: Unnormalized pixel values can slow or break training - Not checking prediction output shape: Most models output a softmax vector, not a single label - Incorrect label mapping: Output `3` might mean "Dress" but needs to be mapped explicitly

■ Final Tip

Whenever you're building an ML-powered image classifier, always align your **training data preprocessing**, **model input shape**, and **inference pipeline**. Debug shape mismatches early, and keep a visual reference of class labels when interpreting results.