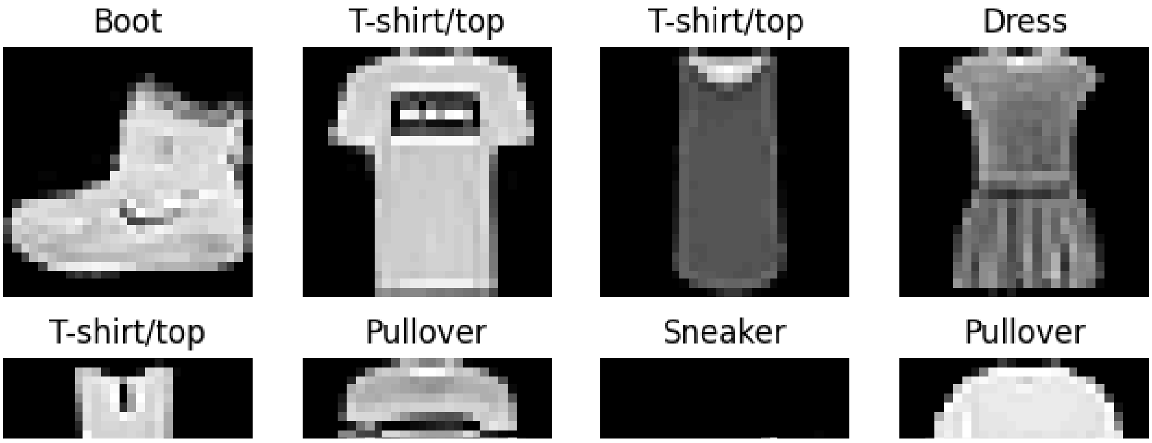
```
# =========================================================
# 1. IMPORTS
# =========================================================
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np

print("TensorFlow:", tf.__version__)

# =========================================================
# 2. LOAD DATASET
# =========================================================
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fa

# Normalize and reshape for CNN
x_train = x_train.reshape(-1, 28, 28, 1) / 255.0
x_test  = x_test.reshape(-1, 28, 28, 1) / 255.0

class_names = [
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Boot"
]

print("Train shape:", x_train.shape)
print("Test shape:", x_test.shape)

# =========================================================
# 3. VISUALIZE SAMPLE IMAGES
# =========================================================
plt.figure(figsize=(8,4))
for i in range(8):
    plt.subplot(2,4,i+1)
    plt.imshow(x_train[i].reshape(28,28), cmap="gray")
    plt.title(class_names[y_train[i]])
    plt.axis("off")
plt.show()

# =========================================================
# 4. BUILD CNN MODEL
# =========================================================
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.4),
    layers.Dense(10, activation="softmax")
])
```

```
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model.summary()

# =========================================================
# 5. TRAIN CNN MODEL
# =========================================================
history = model.fit(
    x_train, y_train,
    epochs=12,
    batch_size=64,
    validation_split=0.1,
    verbose=2
)

# =========================================================
# 6. PLOT ACCURACY & LOSS CURVES
# =========================================================
plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label="Train Acc")
plt.plot(history.history['val_accuracy'], label="Val Acc")
plt.title("Accuracy")
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label="Train Loss")
plt.plot(history.history['val_loss'], label="Val Loss")
plt.title("Loss")
plt.legend()

plt.show()

# =========================================================
# 7. FINAL EVALUATION ON TEST DATA
# =========================================================
test_loss, test_acc = model.evaluate(x_test, y_test, verbos
print("🔥 Final Test Accuracy:", round(test_acc * 100, 2),

# =========================================================
# 8. PREDICTION ON TEST IMAGES
# =========================================================
pred = model.predict(x_test[:16])
pred_classes = np.argmax(pred, axis=1)

plt.figure(figsize=(10,10))
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(x_test[i].reshape(28,28), cmap="gray")
```

```
        plt.title(f"P: {class_names[pred_classes[i]]}\nT: {clas
        plt.axis("off")
    plt.tight_layout()
    plt.show()
```

```
TensorFlow: 2.19.0
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Train shape: (60000, 28, 28, 1)
Test shape: (10000, 28, 28, 1)
```


Boot
T-shirt/top
T-shirt/top
Dress
T-shirt/top
Pullover
Sneaker
Pullover