Exp 2 Add-on

```python
# Import libraries
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.utils import to_categorical

# 1. Load dataset
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# 2. Preprocess data
X_train = X_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
X_test = X_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# 3. Build CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

# 4. Compile model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 5. Train model
history = model.fit(X_train, y_train,
                    epochs=10,
                    batch_size=64,
                    validation_data=(X_test, y_test),
                    verbose=1)

# 6. Evaluate model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Test accuracy: {test_acc:.4f}")
```

Output:

```
Epoch 1/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 69s 70ms/step - accuracy: 0.6679 - loss: 0.9205 - val_accuracy: 0.8372 - val_loss: 0.4342
Epoch 2/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 74s 62ms/step - accuracy: 0.8253 - loss: 0.4797 - val_accuracy: 0.8637 - val_loss: 0.3745
Epoch 3/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 78s 58ms/step - accuracy: 0.8551 - loss: 0.4134 - val_accuracy: 0.8807 - val_loss: 0.3276
Epoch 4/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 81s 57ms/step - accuracy: 0.8672 - loss: 0.3702 - val_accuracy: 0.8864 - val_loss: 0.3055
Epoch 5/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 82s 57ms/step - accuracy: 0.8769 - loss: 0.3429 - val_accuracy: 0.8915 - val_loss: 0.2922
Epoch 6/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 54s 58ms/step - accuracy: 0.8828 - loss: 0.3230 - val_accuracy: 0.8968 - val_loss: 0.2814
Epoch 7/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 81s 57ms/step - accuracy: 0.8861 - loss: 0.3150 - val_accuracy: 0.8904 - val_loss: 0.2927
Epoch 8/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 53s 56ms/step - accuracy: 0.8876 - loss: 0.3016 - val_accuracy: 0.9044 - val_loss: 0.2630
Epoch 9/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 54s 58ms/step - accuracy: 0.8939 - loss: 0.2886 - val_accuracy: 0.9039 - val_loss: 0.2611
Epoch 10/10
938/938 ━━━━━━━━━━━━━━━━━━━━ 52s 56ms/step - accuracy: 0.8966 - loss: 0.2820 - val_accuracy: 0.9032 - val_loss: 0.2620
Test accuracy: 0.9032
```

Test case 2

```python
import numpy as np

# Class names for Fashion-MNIST
class_names = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# Predict on test set
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)  # predicted class indices
y_true = np.argmax(y_test, axis=1)          # true class indices

# Print first 10 results in table format
print(f"{'Input Image':<12}{'True Label':<12}{'Predicted Label':<16}{'Correct (Y/N)':<12}")
for i in range(10):
    true_label = class_names[y_true[i]]
    pred_label = class_names[y_pred_classes[i]]
    correct = 'Y' if true_label == pred_label else 'N'
    print(f"{i:<12}{true_label:<12}{pred_label:<16}{correct:<12}")
```

Output:

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
Input Image True Label  Predicted Label Correct (Y/N)
0           Ankle boot  Ankle boot      Y
1           Pullover    Pullover        Y
2           Trouser     Trouser         Y
3           Trouser     Trouser         Y
4           Shirt       Shirt           Y
5           Trouser     Trouser         Y
6           Coat        Coat            Y
7           Shirt       Shirt           Y
8           Sandal      Sandal          Y
9           Sneaker     Sneaker         Y
```