Practical No: 03

Aim: Convolutional Neural Networks (Classification)
Implementing deep neural network for performing binary classification task.

Description:

Convolutional Neural Networks (Classification)

A **Convolutional Neural Network (CNN)** is a powerful deep learning architecture mainly used for analyzing visual data such as images and videos. Unlike traditional neural networks, CNNs can automatically learn important features (like edges, colors, patterns, shapes, and textures) directly from raw images without requiring manual feature extraction.

CNNs work through several key layers:

- **Convolutional Layers:**
  These layers slide small filters over the image to detect low-level to high-level features. Early layers detect simple patterns like edges, while deeper layers detect complex shapes.
- **Pooling Layers:**
  Pooling reduces the size of feature maps, which helps in lowering computation and preventing overfitting while keeping the most important information.
- **Fully Connected Layers:**
  After extracting features, these layers perform the final classification by learning relationships between those features.

The workflow includes:

1. **Data Preprocessing:**
   Images are resized, normalized, and split into training and testing sets.
2. **Model Architecture:**
   The CNN consists of multiple convolutional and pooling layers for feature extraction, followed by dense layers for classification. Activation functions like ReLU and sigmoid are used.
3. **Training the Model:**
   The network is trained on labeled images, adjusting weights using backpropagation and an optimizer like Adam. The loss function used is binary cross-entropy.
4. **Evaluation:**
   After training, the model's accuracy, precision, recall, and loss are evaluated on test data to measure performance.
5. **Prediction:**
   Finally, the trained model is able to predict the class of new, unseen images.

This CNN-based approach is efficient, scalable, and widely used in applications such as medical image diagnosis, object detection, face recognition, and defect detection.

Code:

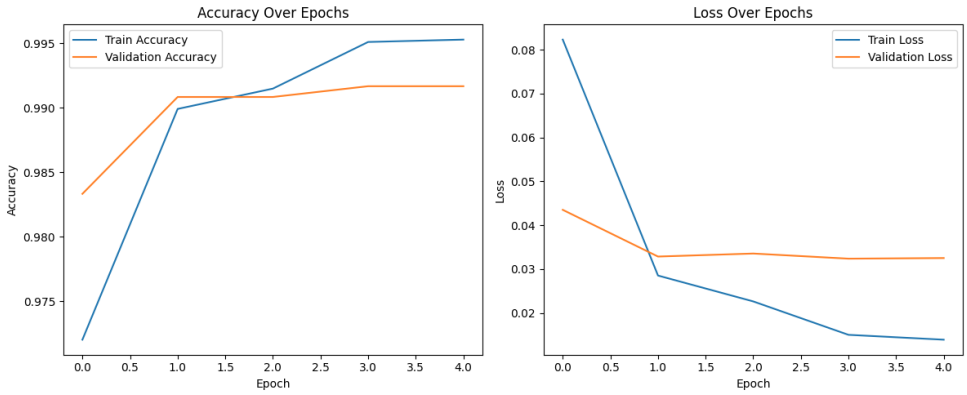
```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
```

```
# 1. Load Fashion-MNIST and keep only: 0=T-shirt, 1=Trouser
 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

train_mask = (y_train == 0) | (y_train == 1)
test_mask  = (y_test == 0) | (y_test == 1)

x_train, y_train = x_train[train_mask], y_train[train_mask]
x_test,  y_test  = x_test[test_mask],  y_test[test_mask]

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

# 2. CNN Model
model = models.Sequential([
   layers.Conv2D(32, 3, activation='relu', input_shape=(28,28,1)),
   layers.MaxPooling2D(),

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

   layers.Flatten(),
   layers.Dense(64, activation='relu'),
   layers.Dense(1, activation='sigmoid')   # Binary output
])

model.compile(optimizer='adam',
           loss='binary_crossentropy',
           metrics=['accuracy'])
# 3. Train the model
history = model.fit(
   x_train, y_train,
   epochs=5,
   batch_size=64,
   validation_split=0.1,
   verbose=1
)


# 4. Plot Accuracy and Loss
plt.figure(figsize=(12,5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.title("Accuracy Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
```

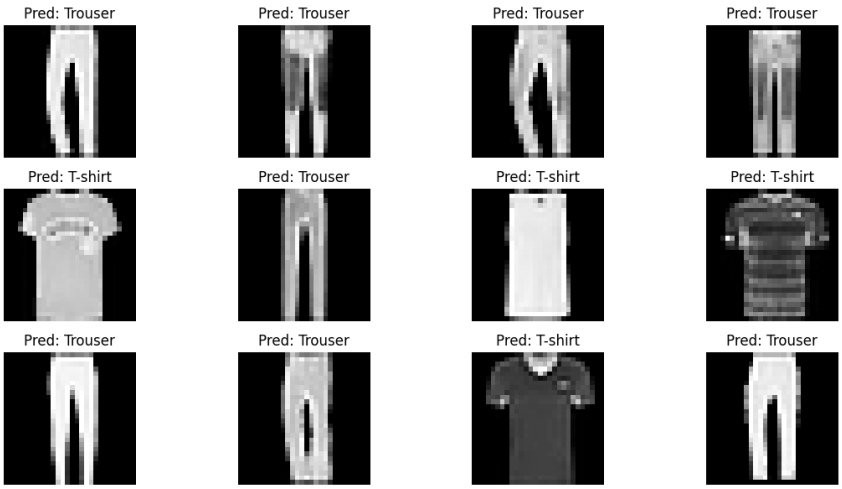
```
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.title("Loss Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.tight_layout()
plt.show()

# 5. Evaluate Model
loss, acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest Accuracy: {acc:.4f}")

# 6. Predict & Show Images (Simplified)
num_images = 12
images = x_test[:num_images]
labels = y_test[:num_images]
probs = model.predict(images).flatten()
preds = (probs >= 0.5).astype(int)

# Display images with predicted label
plt.figure(figsize=(12, 6))
for i in range(num_images):
    plt.subplot(3, 4, i+1)
    plt.imshow(images[i].reshape(28,28), cmap='gray')

    pred_label = "Trouser" if preds[i] == 1 else "T-shirt"
    true_label = "Trouser" if labels[i] == 1 else "T-shirt"
    plt.title(f"Pred: {pred_label}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

Output:

```
Epoch 1/5
169/169 ━━━━━━━━━━━━━━━━━━━━ 6s 31ms/step - accuracy: 0.9433 - loss: 0.1732 - val_accuracy: 0.9833 - val_loss: 0.0435
Epoch 2/5
169/169 ━━━━━━━━━━━━━━━━━━━━ 5s 31ms/step - accuracy: 0.9886 - loss: 0.0288 - val_accuracy: 0.9908 - val_loss: 0.0329
Epoch 3/5
169/169 ━━━━━━━━━━━━━━━━━━━━ 11s 33ms/step - accuracy: 0.9908 - loss: 0.0240 - val_accuracy: 0.9908 - val_loss: 0.0335
Epoch 4/5
169/169 ━━━━━━━━━━━━━━━━━━━━ 5s 31ms/step - accuracy: 0.9937 - loss: 0.0181 - val_accuracy: 0.9917 - val_loss: 0.0324
Epoch 5/5
169/169 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - accuracy: 0.9949 - loss: 0.0140 - val_accuracy: 0.9917 - val_loss: 0.0325
```




```
Test Accuracy: 0.9950
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step
```



Learnings:

In this project, a Convolutional Neural Network (CNN) was successfully implemented to perform a binary classification task using the Fashion-MNIST dataset (T-shirt vs. Trouser). The model was trained using convolution, pooling, and fully connected layers, which enabled it to automatically learn important visual features from the images. After training for 5 epochs, the model achieved strong performance on the test set, demonstrating its ability to accurately differentiate between the two classes.

The plotted accuracy and loss graphs clearly show the learning progress of the model, confirming that the network successfully generalized over the dataset. The prediction visualization further illustrates that the CNN can correctly identify unseen images with high confidence. Overall, this project demonstrates the effectiveness of CNNs for image-based binary classification tasks and highlights their importance in modern deep learning applications.