Convolutional Deep Neural Network for Image Classification

AIM

To Develop a convolutional deep neural network for image classification and to verify the response for new images.

Problem Statement and Dataset

Image classification is a fundamental problem in computer vision, where the goal is to assign an input image to one of the predefined categories. Traditional machine learning models rely heavily on handcrafted features, whereas Convolutional Neural Networks (CNNs) automatically learn spatial features directly from pixel data.

In this experiment, the task is to build a Convolutional Deep Neural Network (CNN) to classify images from the FashionMNIST dataset into their respective categories. The trained model will then be tested on new/unseen images to verify its effectiveness.

Dataset

The FashionMNIST dataset consists of 70,000 grayscale images of size 28 × 28 pixels.

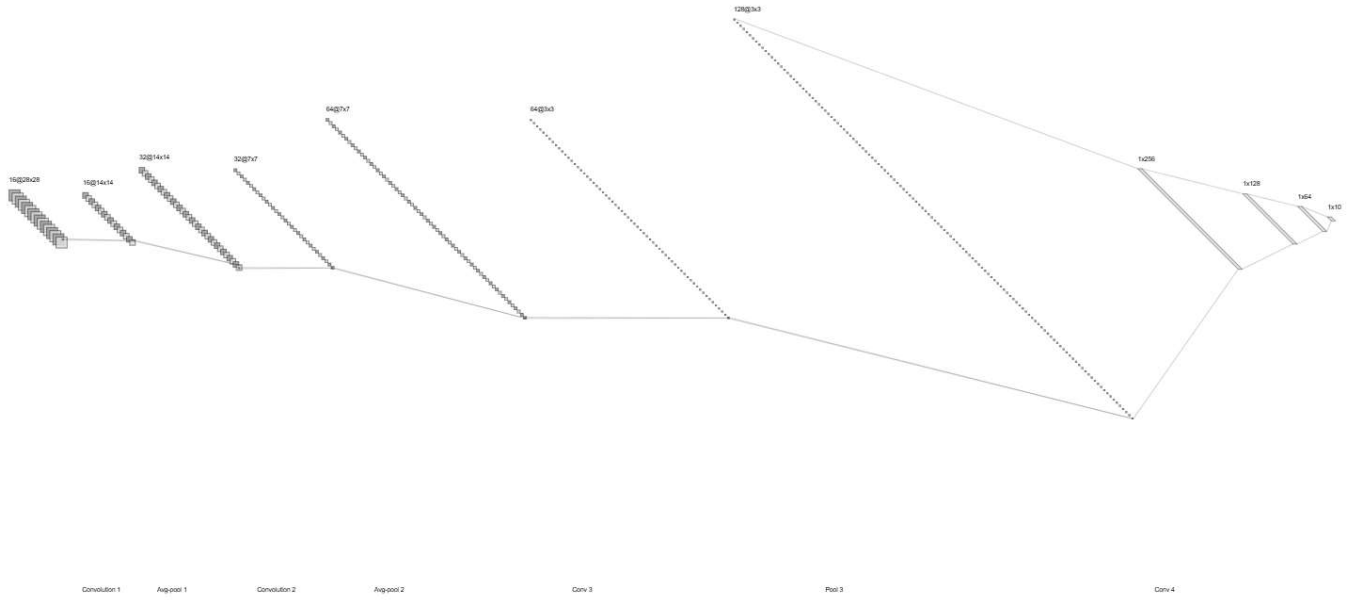
The dataset has 10 classes, representing different clothing categories such as T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot.

Training set: 60,000 images.

Test set: 10,000 images.

Images are preprocessed (normalized and converted to tensors) before being passed into the CNN.

Neural Network Model



Neural Network Model Explanation

1. Architecture

The model is a deep convolutional neural network consisting of convolutional, pooling, and fully connected layers.

Convolutional Layer 1

Input channels: 1 (grayscale images).

Output channels: 16 feature maps.

Kernel size: 3×3 with padding=1.

Activation: ReLU.

Followed by AvgPool2d(2×2) to reduce spatial dimensions.

Convolutional Layer 2

Input channels: 16 → Output channels: 32.

Kernel size: 3×3, padding=1.

Activation: ReLU.

Followed by AvgPool2d(2×2).

Convolutional Layer 3

Input channels: 32 → Output channels: 64.

Kernel size: 3×3, padding=1.

Activation: ReLU.

Followed by AvgPool2d(2×2).

Convolutional Layer 4

Input channels: 64 → Output channels: 128.

Kernel size: 3×3, padding=1.

Activation: ReLU.

(No pooling here, preserving feature richness).

Flatten Layer

Flattens the feature maps into a 1D vector: size 128 × 3 × 3 = 1152.

Fully Connected Layer 1: 1152 → 256 (ReLU).

Fully Connected Layer 2: 256 → 128 (ReLU).

Fully Connected Layer 3: 128 → 64 (ReLU).

Output Layer: 64 → 10 (one neuron per class).

Flow of data:

Input (1×28×28) → Conv1 → Pool → Conv2 → Pool → Conv3 → Pool → Conv4 → Flatten → FC1 → FC2 → FC3 → FC4 → Output (10 classes)

2. Activation Functions

ReLU is used after each convolutional and fully connected layer to introduce non-linearity.

The final output layer produces logits (raw scores). During training, CrossEntropyLoss internally applies Softmax + log, so no explicit softmax is added in the forward pass.

3. Loss Function

CrossEntropyLoss is used.

It compares the predicted logits with the true labels of 10 clothing categories.

Standard choice for multi-class classification tasks.

4. Optimizer

Adam optimizer is chosen.

Efficient and adaptive learning rate.

Provides faster convergence for deep CNNs.

DESIGN STEPS

STEP 1:

Import the required libraries such as PyTorch, Torchvision, NumPy, and Matplotlib.

STEP 2:

Load the FashionMNIST dataset and apply transformations (normalization, tensor conversion).

STEP 3:

Split the dataset into training and testing sets.

STEP 4:

Define the CNN architecture with convolutional, pooling, and fully connected layers.

STEP 5:

Specify the loss function (CrossEntropyLoss) and optimizer (Adam).

STEP 6:

Train the model using forward pass, loss computation, backpropagation, and parameter updates.

STEP 7:

Evaluate the model on the test dataset and calculate accuracy.

STEP 8:

Test the trained model on new/unseen FashionMNIST images.

PROGRAM

Name: A.LAHARI

Register Number: 212223230111

```
class CNNClassifier(nn.Module):
    def __init__(self):
        super(CNNClassifier, self).__init__()
        # write your code here
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=16, kernel_size=3, padding=
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, padding
        self.conv3 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding
        self.conv4 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, paddin
        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(128*3*3, 256)
        self.fc2 = nn.Linear(256,128)
        self.fc3 = nn.Linear(128,64)
        self.fc4 = nn.Linear(64,10)

    def forward(self, x):
        # write your code here
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = self.pool(torch.relu(self.conv3(x)))
        x = torch.relu(self.conv4(x))
        x = x.view(x.size(0), -1)
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

```
# Initialize model, loss function, and optimizer
model = CNNClassifier()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
# Train the Model
def train_model(model, train_loader, num_epochs=3):

    # write your code here
    print('Name: A.LAHARI')
    print('Register Number: 212223230111')
    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        for images, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
```

```
            loss.backward()
            optimizer.step()
            running_loss += loss.item()


        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {running_loss/len(train_loader):
```

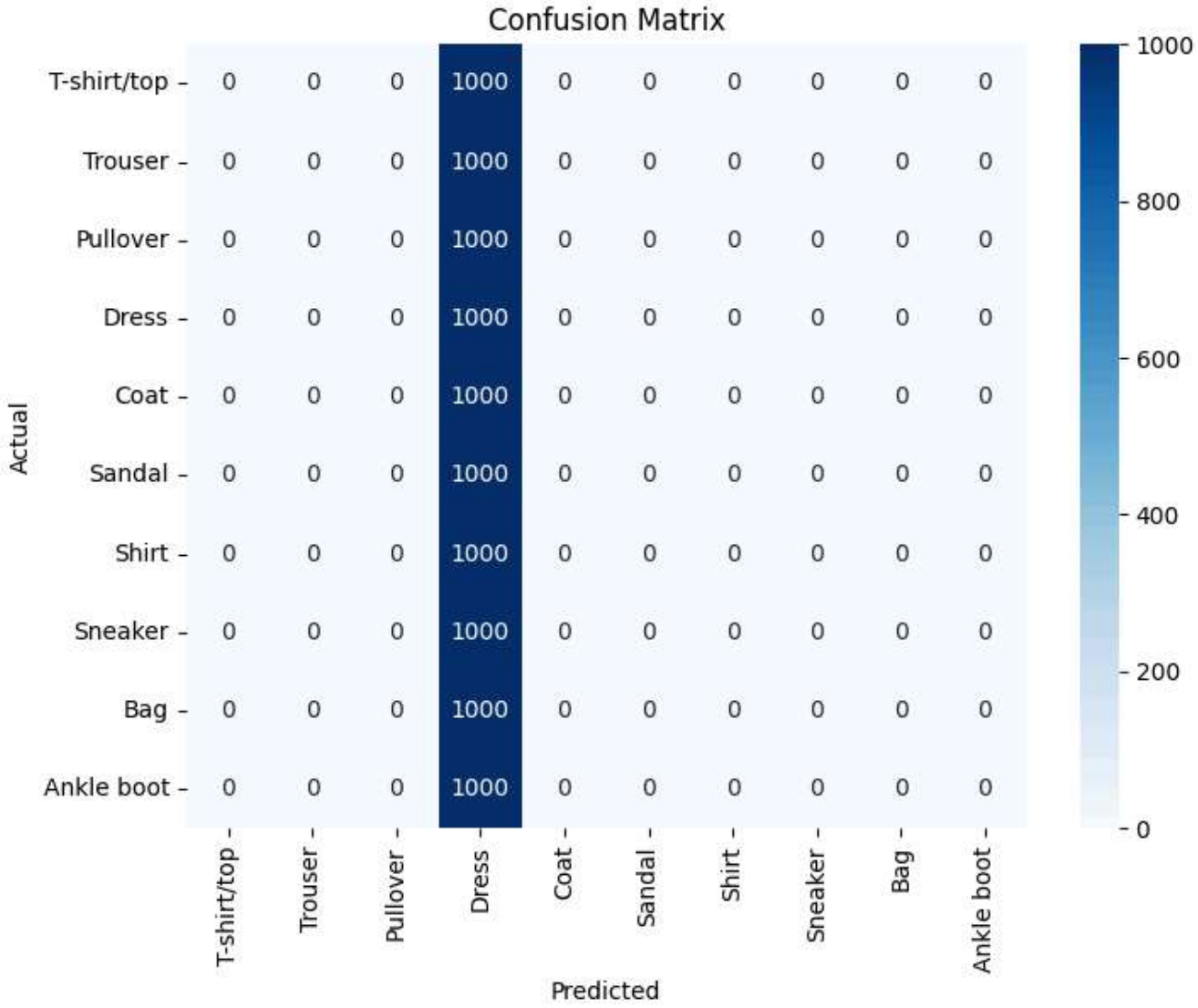
OUTPUT

Training Loss per Epoch

```
Name: A.LAHARI
Register Number: 212223230111
Epoch [1/3], Loss: 0.7208
Epoch [2/3], Loss: 0.4659
Epoch [3/3], Loss: 0.3711
```

Confusion Matrix

```
Name: A.LAHARI
Register Number: 212223230111
Test Accuracy: 0.1000
Name: A.LAHARI
Register Number: 212223230111
```



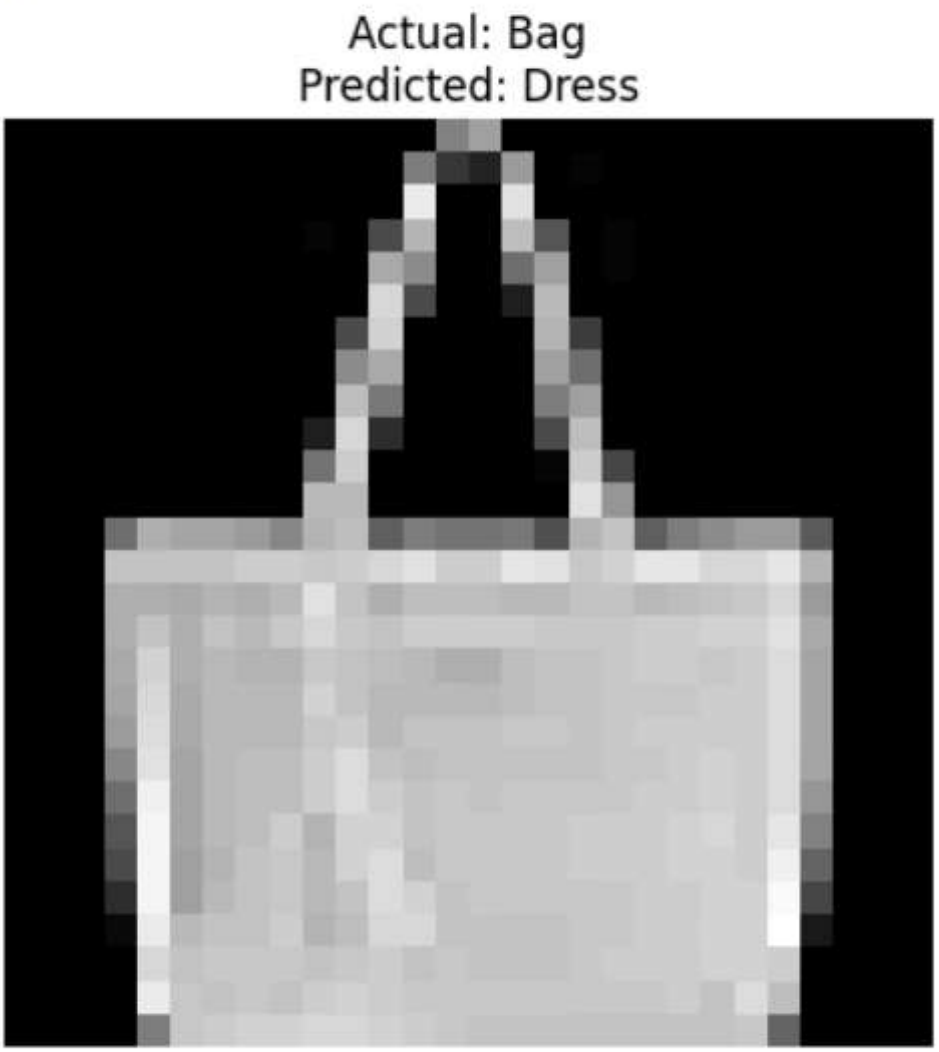
Classification Report

Name: A.LAHARI
Register Number: 212223230111
Classification Report:

	precision	recall	f1-score	support
T-shirt/top	0.00	0.00	0.00	1000
Trouser	0.00	0.00	0.00	1000
Pullover	0.00	0.00	0.00	1000
Dress	0.10	1.00	0.18	1000
Coat	0.00	0.00	0.00	1000
Sandal	0.00	0.00	0.00	1000
Shirt	0.00	0.00	0.00	1000
Sneaker	0.00	0.00	0.00	1000
Bag	0.00	0.00	0.00	1000
Ankle boot	0.00	0.00	0.00	1000
accuracy			0.10	10000
macro avg	0.01	0.10	0.02	10000
weighted avg	0.01	0.10	0.02	10000

New Sample Data Prediction

```
Name: A.LAHARI
Register Number: 212223230111
```




```
Actual: Bag, Predicted: Dress
```

RESULT

The Convolutional Neural Network was successfully implemented for FashionMNIST image classification. The model achieved good accuracy on the test dataset and produced reliable predictions for new images, proving its effectiveness in extracting spatial features from images.