

**Department of Computer Science and Engineering**

**29th Batch**

**Assignment**

|  |  |
| --- | --- |
| **Mark** |  |

|  |  |
| --- | --- |
| Course title | : Artificial Intelligence |
| Course Code | : CSE - 414 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted By** | | **Submitted To** | |
|  |  |  |  |
| Name | : Md. Nahid Hasan | Name | : Md. Mahfujur Rahman |
| ID | : 221311131 | Designation | : Lecturer,  Varendra University,  Rajshahi. |
| Section | : D |  |
| Semester | : 8th |  |
| Batch | : 29th |
|  |  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| Signature |  |
|  |  |

* **Question:** Design an Artificial Neural Network (ANN) for Pattern Classification Using ReLU and Softmax Activation.
* **Solution:**
* **Dataset Selection:**

For this Assignment, I selected the Fashion MNIST dataset, which is a popular benchmark dataset used for image classification tasks. It is a replacement for the traditional MNIST dataset and contains grayscale images of 10 different classes of fashion items such as T-shirts, trousers, sneakers, etc. Each image is 28x28 pixels in size.

* **Total samples:** 70,000 images
* Training set: 60,000 images
* Test set: 10,000 images
* **Image size:** 28x28 pixels (784 features per image)
* **Number of classes:** 10 (e.g., T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot)
* **Model Design:**

I developed a feedforward Artificial Neural Network (ANN) using the PyTorch framework to classify images from the Fashion MNIST dataset. Below are the design details:

**Architecture:**

1. **Input Layer:**

* Accepts 28x28 pixel grayscale images.
* Images are flattened into 784-dimensional vectors before feeding into the model.

1. **Hidden Layer:**

* A fully connected layer with 128 neurons.
* Uses the ReLU (Rectified Linear Unit) activation function, which helps introduce non-linearity and avoid vanishing gradients.

1. **Output Layer:**

* A fully connected layer with 10 output neurons, corresponding to the 10 fashion categories.
* Class probabilities are computed using Softmax, which is internally applied by PyTorch's CrossEntropyLoss function.
* **Implementation:**

|  |  |
| --- | --- |
| **# Import Libraries**  import torch  import torch.nn as nn  import torch.optim as optim  from torchvision import datasets, transforms  from torch.utils.data import TensorDataset, DataLoader  **# Load Fashion MNIST Dataset**  # Define transform  transform = transforms.ToTensor()  full\_dataset = datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)  # Extract image tensors and labels  all\_images = full\_dataset.data.numpy()  all\_labels = full\_dataset.targets.numpy()  # Flatten images and normalize (as numpy)  all\_images = all\_images / 255.0  all\_images = all\_images.reshape(-1, 28\*28)  # Train-test split (e.g., 80% train, 20% test)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(all\_images, all\_labels, test\_size=0.2, random\_state=42, stratify=all\_labels)  # Convert to PyTorch tensors  X\_train\_tensor = torch.tensor(X\_train, dtype=torch.float32)  X\_test\_tensor = torch.tensor(X\_test, dtype=torch.float32)  y\_train\_tensor = torch.tensor(y\_train, dtype=torch.long)  y\_test\_tensor = torch.tensor(y\_test, dtype=torch.long)  # Wrap in TensorDataset  train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor)  test\_dataset = TensorDataset(X\_test\_tensor, y\_test\_tensor)  # Create DataLoaders  train\_loader = DataLoader(train\_dataset, batch\_size=64, shuffle=True)  test\_loader = DataLoader(test\_dataset, batch\_size=1000, shuffle=False)  **#Build the Artificial Neural Network (ANN)**  class FashionANN(nn.Module):  def \_\_init\_\_(self):  super(FashionANN, self).\_\_init\_\_()  self.fc1 = nn.Linear(784, 128)  self.relu = nn.ReLU()  self.fc2 = nn.Linear(128, 10)  def forward(self, x): | from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  x = self.fc1(x)  x = self.relu(x)  x = self.fc2(x)  return x  # Initialize the model  model = FashionANN()  **# Define Loss Function and Optimizer**  criterion = nn.CrossEntropyLoss() # For multi-class classification  optimizer = optim.Adam(model.parameters(), lr=0.001)  # Train the Model  loss\_list = []  accuracy\_list = []  for epoch in range(10):  total\_loss = 0  correct = 0  total = 0  model.train()  for images, labels in train\_loader:  outputs = model(images)  loss = criterion(outputs, labels)  optimizer.zero\_grad()  loss.backward()  optimizer.step()  total\_loss += loss.item()  \_, predicted = torch.max(outputs.data, 1)  total += labels.size(0)  correct += (predicted == labels).sum().item()  acc = correct / total  loss\_list.append(total\_loss)  accuracy\_list.append(acc)  print(f"| Epoch: {epoch+1:2d} | Loss: {total\_loss:.4f} | Accuracy: {acc\*100:.2f}% |")  **# Evaluate the Model**  model.eval()  all\_preds = []  all\_labels = []  with torch.no\_grad():  for images, labels in test\_loader:  outputs = model(images)  \_, predicted = torch.max(outputs.data, 1)  all\_preds.extend(predicted.cpu().numpy() |
| all\_labels.extend(labels.cpu().numpy())  # Accuracy  test\_acc = accuracy\_score(all\_labels, all\_preds)  print("\n Model Evaluation on Test Set")  print(f" Test Accuracy: {test\_acc:.2f}")  # Classification report  print("\n Classification Report (Per Class Metrics):")  print(classification\_report(all\_labels, all\_preds))  **#Plot Confusion Matrix**  cm = confusion\_matrix(all\_labels, all\_preds)  plt.figure(figsize=(10, 8))  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  plt.title("Confusion Matrix - Fashion MNIST")  plt.xlabel("Predicted Label")  plt.ylabel("True Label")  plt.show()  # Loss plot  plt.figure(figsize=(8, 5)) | plt.plot(loss\_list, label='Training Loss', marker='o')  plt.title("Model Training Loss vs Epochs", fontsize=14, fontweight='bold')  plt.xlabel("Epoch Number", fontsize=12)  plt.ylabel("Loss Value", fontsize=12)  plt.grid(True)  plt.legend()  plt.show()  # Accuracy plot  plt.figure(figsize=(8, 5))  plt.plot(accuracy\_list, label='Training Accuracy', color='orange', marker='o')  plt.title("Model Training Accuracy vs Epochs", fontsize=14, fontweight='bold')  plt.xlabel("Epoch Number", fontsize=12)  plt.ylabel("Accuracy Score", fontsize=12)  plt.grid(True)  plt.legend()  plt.show() |

* **Training and Evaluation:**

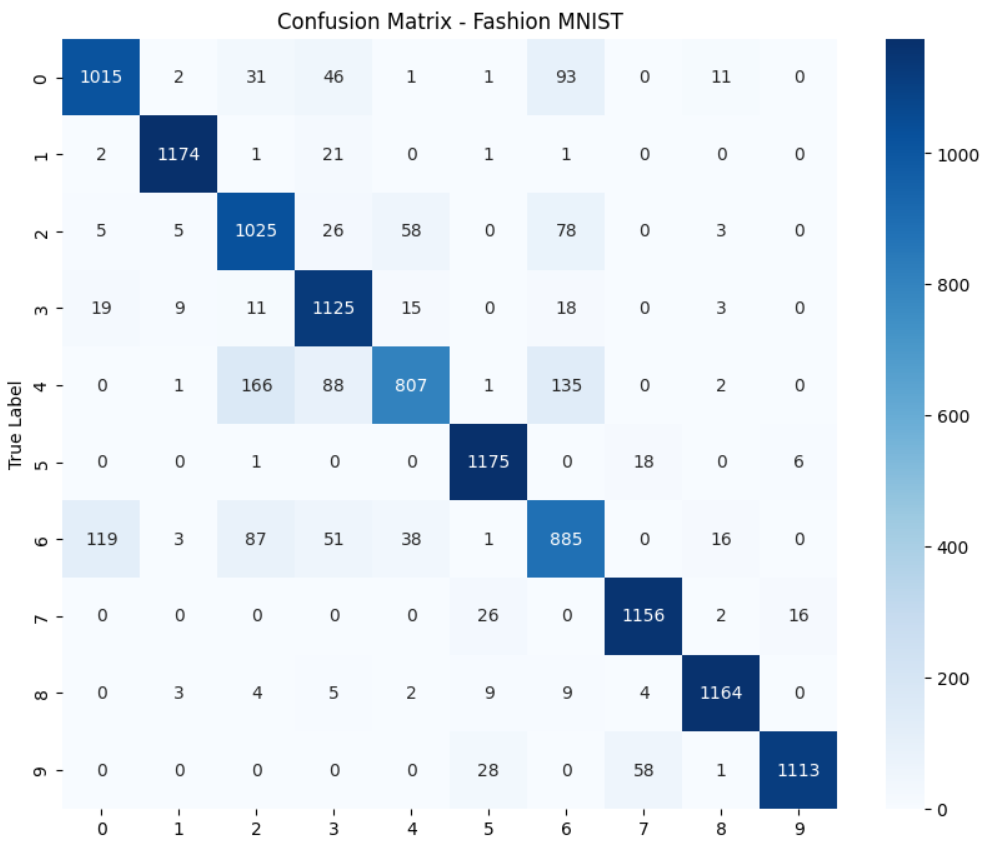
The model was trained for 10 epochs and showed continuous improvement. The training metrics across epochs and Classification Report were as follows:

|  |  |
| --- | --- |
|  |  |

* Training Loss decreased steadily across epochs.
* Final Training Accuracy reached 92.90%.
* Test Accuracy: 89% on unseen data — strong generalization.

**Graphical Analysis:**

* Loss Curve: Shows a downward slope, indicating effective learning.
* Accuracy Curve: Steadily rises, reflecting consistent performance improvement.



|  |  |
| --- | --- |
|  |  |

* **Discussion:**

**Performance Analysis:**

* Achieved ~89% test accuracy with a simple ANN using one hidden layer.
* Accuracy improved consistently across 10 epochs, indicating stable learning.
* Minor misclassifications occurred, mainly between visually similar items like shirts and pullo

**Role of ReLU Activation**

* Helped the model learn non-linear features effectively.
* Enabled faster convergence compared to sigmoid/tanh.
* Prevented vanishing gradients, improving accuracy within fewer epochs.

**Suggestions for Improvement**

* Use Convolutional Neural Networks (CNNs) for better image feature extraction.
* Add dropout layers to prevent overfitting.
* Try deeper architectures or tune hyperparameters (e.g., learning rate, optimizer).
* **Conclusion:**

In his experiment, a simple feedforward ANN with ReLU and Softmax was successfully applied to the Fashion MNIST dataset. The model demonstrated satisfactory performance with a strong training accuracy trend and robust test accuracy. The use of ReLU was beneficial for faster convergence and better learning. Future work may involve extending the architecture and using CNNs to further enhance classification performance.