Fashion Item Classification using Convolutional Neural Networks

Machine Learning Lab

CSE 432

Submitted To:
Md. Yousuf Ali, Lecturer, CSE, UITS

Submitted By:
Name: Maharaj Hossain Tanim
ID: 2215151040
Sec. : 7(A2)
Batch: 51
Dept. : CSE

Submission Date: **07/07/2025**

Report:

Title: Fashion Item Classification using Convolutional Neural Networks

1. Introduction:

This project explores image classification using Convolutional Neural Networks (CNNs), a powerful deep learning technique. The goal is to categorize fashion items from the Fashion MNIST dataset, a widely used benchmark. CNNs are chosen for their effectiveness in learning visual features directly from images. All development and training were performed within Google Colab, leveraging its GPU capabilities

2. Objective:

The key objectives of this project were to:

- Load and understand the Fashion MNIST dataset.
- Preprocess image data for CNN input.
- Design and implement a simple CNN model.
- Train the CNN model on the training data.
- Evaluate the model's performance on unseen test data.
- Visualize training progress and prediction examples.
- Utilize Google Colab for efficient deep learning execution.

3. Dataset Description:

The Fashion MNIST dataset is a collection of Zalando's article images.

- **Source:** https://www.tensorflow.org/datasets/catalog/fashion_mnist
- **Size:** Training Data: 60,000 grayscale images and Testing Data: 10,000 grayscale images.
- **Image Format:** 28x28 pixel
- **Classes:** Ten distinct clothing categories:
    1. T-shirt/top
    2. Trouser
    3. Pullover
    4. Dress
    5. Coat
    6. Sandal
    7. Shirt
    8. Sneaker
    9. Bag
    10. Ankle Boot

4. Code Screenshots:

```
#step-1. Load the Fashion MNIST Dataset
print("Loading Fashion MNIST dataset...")
(train_images, train_labels), (test_images, test_labels) =
keras.datasets.fashion_mnist.load_data()
print("Dataset loaded successfully!")

# Define class names for visualization
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
Loading Fashion MNIST dataset...
Dataset loaded successfully!
```

```
#step-2. Explore the Data (Optional, but good for reporting)
print(f"Train images shape: {train_images.shape}")
print(f"Train labels shape: {train_labels.shape}")
print(f"Test images shape: {test_images.shape}")
print(f"Test labels shape: {test_labels.shape}")
```

```
Train images shape: (60000, 28, 28)
Train labels shape: (60000,)
Test images shape: (10000, 28, 28)
Test labels shape: (10000,)
```

```
#step-3. Preprocess the Images
# Normalize pixel values to be between 0 and 1
train_images = train_images / 255.0
test_images = test_images / 255.0

# Reshape images to include a channel dimension (28, 28, 1) for CNN input
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

print(f"Reshaped train images shape: {train_images.shape}")
print(f"Reshaped test images shape: {test_images.shape}")
```

```
Reshaped train images shape: (60000, 28, 28, 1)
Reshaped test images shape: (10000, 28, 28, 1)
```

```
#step-4. Build the CNN Model
print("\nBuilding the CNN model...")
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax') # 10 classes for Fashion MNIST
])

# Display model summary
model.summary()
```

```
#step-6. Train the Model
print("\nTraining the model...")
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
print("Model training complete!")
```

```
#step-7. Evaluate the Model
print("\nEvaluating the model on the test set...")
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"\nTest accuracy: {test_acc}")
Evaluating the model on the test set...
313/313 - 1s - 2ms/step - accuracy: 0.9105 - loss: 0.2835

Test accuracy: 0.9104999899864197
```

5. Methodology

Model Architecture (CNN) - A simple Convolutional Neural Network (CNN) architecture was employed for image classification. The model consists of:

- ➔ Three Conv2D layers with ReLU activation, followed by MaxPooling2D layers for feature extraction and dimensionality reduction.
- ➔ A Flatten layer to convert the 2D feature maps into a 1D vector.
- ➔ Two Dense (fully connected) layers, with the final Dense layer using a softmax activation function to output probabilities for each of the 10 classes.

Approach -

1. Data Loading - The Fashion MNIST dataset was loaded directly using keras.datasets.fashion_mnist.load_data().
2. Preprocessing -
    - Image pixel values were normalized from the range [0, 255] to [0, 1] by dividing by 255.0.
    - Images were reshaped to (height, width, channels) i.e., (28, 28, 1) to be compatible with the CNN input layer.
3. Model Compilation -
    - Optimizer: adam
    - Loss Function: SparseCategoricalCrossentropy (suitable for integer labels)
    - Metrics: accuracy
4. Training - The model was trained for 10 epochs.
5. Evaluation - The trained model's performance was evaluated on the unseen test set to determine its generalization capability.

6. Conclusion

This project successfully implemented and evaluated a CNN for classifying fashion items from the Fashion MNIST dataset. The model achieved high accuracy, demonstrating the effectiveness of CNNs for image classification. The use of Google Colab provided an efficient environment for development and training.

Future work could include:

- **Hyperparameter Tuning**: Optimizing learning rate, batch size, and network architecture.
- **Data Augmentation:** Applying transformations to training images to improve robustness.
- **Regularization**: Incorporating techniques like Dropout to mitigate overfitting on more complex models.
- **Advanced Architectures**: Exploring deeper or more complex CNN models if needed for higher accuracy or different datasets.