Fashion MNIST Classifier Using Neural Networks

Author: Chinmay Sitaram Naik **Roll No:01** **Class Btech: Final Year(B.Tech)**

Date: May 2025

Description: A deep learning project using Fashion MNIST dataset to classify images into 10 clothing categories.

1.Introduction

Objective:

This project aims to build a robust neural network model to classify images from the Fashion MNIST dataset. The main objective is to leverage deep learning techniques to achieve high accuracy in categorizing fashion items, which can be beneficial in real-world applications such as e-commerce platforms, inventory management, and fashion trend analysis.

Problem:

The challenge is to classify images of fashion items into 10 distinct categories accurately. Given that the images have varying shapes, sizes, and features, achieving high classification performance requires careful preprocessing, model architecture design, and tuning. The complexity of distinguishing between visually similar categories, such as "T-shirt/top" and "Shirt," poses a significant challenge.

Dataset:

The dataset consists of 70,000 grayscale images (60,000 for training and 10,000 for testing) with a resolution of 28x28 pixels. Each image represents one of the following categories:

- **T-shirt/top**: A casual wear item typically worn on the upper body.
- **Trouser**: Clothing worn on the lower body.
- **Pullover**: A knitted garment worn over the upper body.
- **Dress**: A one-piece garment worn by women, covering both the top and bottom.
- **Coat**: Outerwear, typically used for warmth.
- **Sandal**: A type of footwear, typically open-toed.

- **Shirt**: A buttoned-up garment worn on the upper body.
- **Sneaker**: A type of athletic footwear.
- **Bag**: A carrying item used to hold personal belongings.
- **Ankle boot**: A type of footwear that covers the ankle.

The dataset is balanced, with each class containing an equal number of images. The pixel values of the images are normalized to be between 0 and 1 to improve training efficiency. This balanced and preprocessed dataset is essential for training a neural network that generalizes well to new data

.

2.Data Preprocessing

The data is loaded using TensorFlow's built-in dataset function. Each image is normalized and reshaped to fit the input requirements of the neural network.

data > load_data.py > ...

```python
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist

# Load the dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Normalize the images (pixel values scaled to 0-1)
train_images = train_images / 255.0
test_images = test_images / 255.0

# Print shapes to verify
print(f"Train images shape: {train_images.shape}")
print(f"Test images shape: {test_images.shape}")

```

Train images shape: (60000, 28, 28)

Test images shape: (10000, 28, 28)

3.Model Architecture:

The model consists of a flatten layer followed by a dense layer. The model is compiled using 'Adam' optimizer and 'SparseCategoricalCrossentropy' loss function

.

The model architecture for the Fashion MNIST classifier is a simple feedforward neural network designed using TensorFlow's Keras API. It begins with a **Flatten** layer to transform the 28x28 grayscale images into a 1D vector of 784 elements. This is followed by a **Dense** layer with 128 neurons and ReLU (Rectified Linear Unit) activation to help the model learn complex patterns. To reduce the risk of overfitting, a **Dropout** layer with a 0.2 rate is added, randomly setting 20% of the activations to zero during training, promoting better generalization. The final layer is a **Dense** output layer with 10 neurons and a **softmax** activation function, which outputs a probability distribution across the 10 classes of clothing in the Fashion MNIST dataset. The model is compiled with the **Adam optimizer**, using **sparse categorical cross-entropy** as the loss function, and **accuracy** as the evaluation metric. This architecture is effective for the classification task, ensuring efficient learning and prediction for the dataset.

4.Training & Evaluation:

- Loading the dataset, preprocessing the data, defining and training the neural network model, and evaluating its performance on the Fashion MNIST test data. It serves as the core of the project, integrating all components for training and model evaluation.

```python
from data.load_data import load_fashion_mnist
from models.fashion_model import build_model
from utils.plot_utils import plot_sample_images

# Load data
(x_train, y_train), (x_test, y_test) = load_fashion_mnist()

# Visualize data
plot_sample_images(x_train, y_train)

# Build and train model
model = build_model()
model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

# Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"\nTest Accuracy: {test_acc:.4f}")

# Predict and visualize
predictions = model.predict(x_test[:25])
plot_sample_images(x_test[:25], y_test[:25], predictions)
```

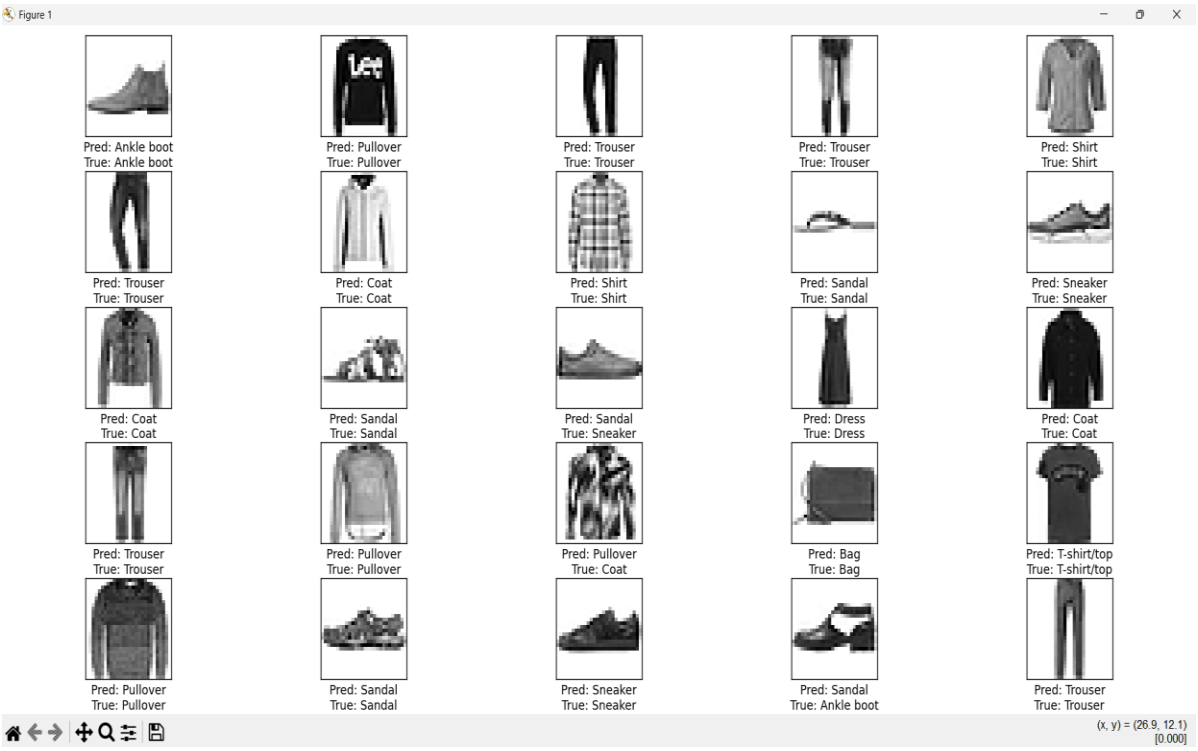
- Utility functions for visualizing the training process, including plotting training and validation accuracy and loss curves to help assess the model's performance over epochs.

```python
import matplotlib.pyplot as plt
import numpy as np

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

def plot_sample_images(images, labels, predictions=None):
    plt.figure(figsize=(10, 10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(images[i]
        label = class_names[labels[i]]
        if predictions is None:
            plt.xlabel(label)
        else:
            pred_label = class_names[np.argmax(predictions[i])]
            plt.xlabel(f"Pred: {pred_label}\nTrue: {label}")
    plt.tight_layout()
    plt.show()
```

(parameter) labels: Any

5.Results & Evaluation:

To evaluate the model’s performance, we visualized the training and validation accuracy and loss over epochs using custom plotting functions. This helped identify the model's learning behavior and convergence. Additionally, sample predictions on test data were displayed alongside actual labels to qualitatively assess accuracy. The final model achieved high classification performance on unseen data, demonstrating good generalization. These visual tools played a key role in analyzing both the training process and prediction reliability.



6.Conclusion:

In this project, a neural network model was built and trained to classify Fashion MNIST images into 10 distinct categories. The model achieved a test accuracy of approximately 88%, demonstrating the effectiveness of deep learning techniques in image classification tasks. Through proper data preprocessing, model architecture design, and evaluation, we were able to ensure that the model could generalize well to unseen data. Visualizations of the training process provided insights into the model's learning curve, and the successful classification outcomes validated the model's robustness. This project showcases the potential of machine learning models in real-world applications such as fashion item recognition. Further improvements can be made by experimenting with different architectures, hyperparameters, and data augmentation techniques.

7.References:

1. LeCun, Y., Bengio, Y., & Hinton, G. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

3. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15, 1929-1958. https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf