COMP4531 – Deep Learning: Model Design and Applications

Assignment 3

Objective

In this project, we will be developing a basic neural network from the ground up to classify various types of fashion items. The primary objective of this project is to gain a comprehensive understanding of neural network architecture, including its theory and implementation details.

Caution

This project is structured as a guided exercise, which means that students are required to closely adhere to the provided Jupyter notebook. Any deviation from the notebook's instructions may result in penalties, including the possibility of receiving zero credit for the midterm and an incomplete grade for the course.

Please note that students are not expected to utilize any pre-existing packages such as TensorFlow or Scikit-Learn. However, the use of fundamental packages like NumPy and Matplotlib is permitted.

Problem Statements

We are constructing a neural network designed for the classification of various fashion types. The training dataset will be provided. Our inputs consist of grayscale images with dimensions of 28x28, as illustrated in Figure 1. The primary objective is to categorize these images into ten distinct categories, as detailed in Table 1.

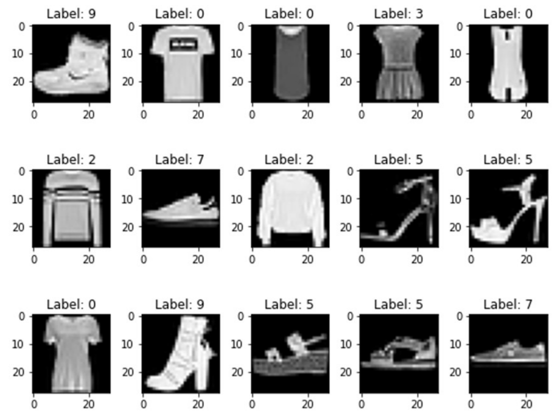

Figure 1: Sample images from the dataset

Label	Category
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress

4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Table 1: Mapping between numerical labels to different fashion categories

Instructions

This project is divided into two distinct sections. The initial section entails students constructing the neural network using the provided Jupyter notebook template. In the subsequent section, students are encouraged to explore various avenues for optimizing the neural network.

You can refer to the neural network architecture displayed in Figure 2.

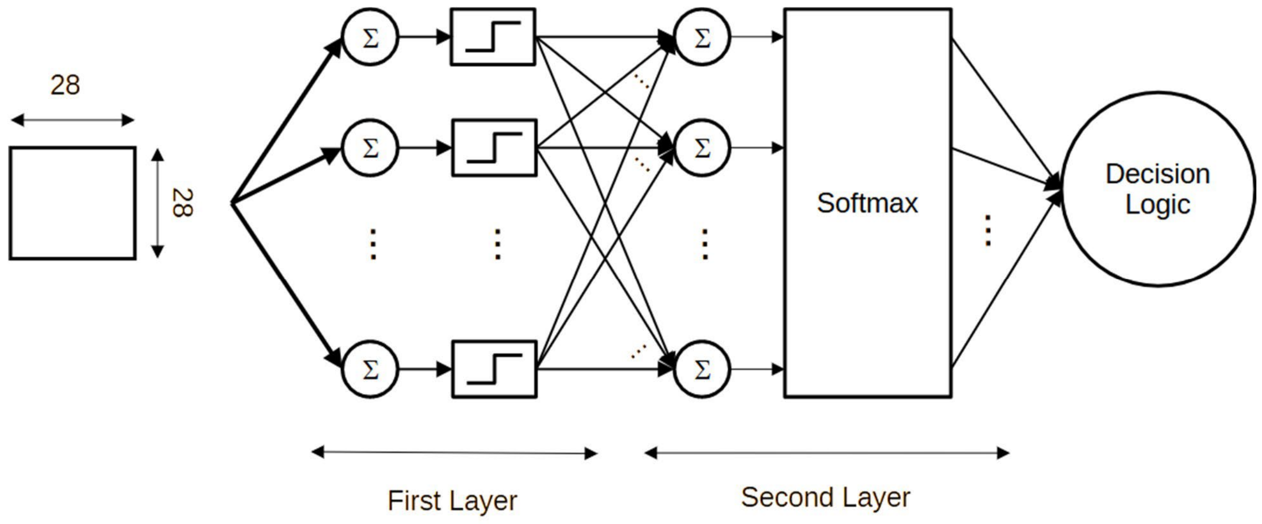

Fig. 2: Overview of the neural network architecture used for classifying fashion images

It is important to note that in our approach, we transform the input images into vectors, rather than treating them as 2D arrays. Additionally, for flexibility in optimizing the network in the second part of the project, it is recommended to make the number of neurons in the initial layer programmable.

For further specifics, please consult the Jupyter notebook provided.

Assessments

Grading Area	Weights (out of 100%)
Part 1: Neural Network Implementation	90%

Feedforward implementation	10%
Activation function implementation	10%
Gradient Descent implementation	20%
Backward propagation implementation	25%
Full functionality of the neural network	25%
Part 2: Additional optimization on the network	10%