Technical Implementation Report

A Neural Network Model for Fashion-MNIST Image Classification

Student Name: Akaash Chellapandiyan - P2910240
Module: CSIP5103 - Artificial Neural Networks & NLP
Date: November 5, 2025

1. Introduction

The primary objective of this project was to develop a classifier that would be able to distinguish between ten types of clothing based on images within the Fashion-MNIST data set. Although the MNIST handwritten digit dataset is a well-known benchmark for computer vision, the Fashion-MNIST dataset presents a greater challenge due to its higher complexity. This report describes both the design and comparison of two different models, a baseline Multi-Layer Perceptron (MLP) or Artificial Neural Network (ANN) model and a more advanced Convolutional Neural Network (CNN) model. The major goal of the project was to compare the performance of each model and identify the architecture advantages of CNN's for image classification. All of the code was developed in Python and the data were loaded directly from a local repository using TensorFlow and Keras.

2. Data Handling and Preprocessing

The dataset was first acquired by cloning the official zalandoresearch/fashion-mnist GitHub repository. A custom Python function, load_apparel_dataset_from_files, was implemented to parse the raw, gzipped UBTYE files for training and testing.

2.1. Loading and Splitting the Data

The custom function loaded 60,000 training images (raw_training_images) and 10,000 testing images (raw_testing_images) along with their corresponding labels. During the model training phase, a validation set was partitioned from the training data by setting validation_split=0.1 in the fit method, reserving 10% of the training data for validation.

2.2. Data Transformation and Normalization

Two critical preprocessing steps were applied to the raw data:

1. **Normalization:** The pixel values, originally in an integer range of 0-255, were converted to a floating-point type and scaled to a 0-1 range. This was achieved by dividing each pixel value by 255.0, creating the normalized_training_images and normalized_testing_images arrays. This step is crucial for stable and efficient model training.
2. **Reshaping for CNN:** The CNN architecture requires an input shape that includes a channel dimension. Therefore, the normalized image arrays were reshaped from (28, 28) to (28, 28, 1), resulting in the cnn_training_input and cnn_testing_input arrays.

3. Model Architecture and Design

Two distinct models, the mlp_classifier and the convolutional_classifier, were constructed.

3.1. MLP (ANN) Topology

The mlp_classifier served as the baseline model. Its simple, fully connected architecture was defined as follows:

1. An **Input Layer:** A Flatten layer to transform each 28x28 image into a 1D vector of 784 pixels.
2. A **Hidden Layer:** A Dense layer containing 128 neurons with the ReLU (Rectified Linear Unit) activation function.

3. An **Output Layer:** A Dense layer with 10 neurons, corresponding to the 10 apparel categories, using the **Softmax** activation function to produce a probability distribution over the classes.

3.2. CNN Topology

The convolutional_classifier was designed to leverage the spatial features within the images. Its architecture was more complex:

1. **Convolutional Layer 1:** A Conv2D layer with 32 filters of size (3, 3) and ReLU activation to detect low-level features like edges.
2. **Max Pooling Layer 1:** A MaxPooling2D layer with a (2, 2) pool size to downsample the feature maps.
3. **Convolutional Layer 2:** A second Conv2D layer with 64 filters of size (3, 3) and ReLU activation to learn more intricate patterns.
4. **Max Pooling Layer 2:** Another MaxPooling2D layer.
5. **Flatten Layer:** To convert the 2D feature maps into a 1D vector for the dense layers.
6. **Dense Layer:** A fully connected layer with 128 neurons and ReLU activation.
7. **Output Layer:** A final Dense layer with 10 neurons and a Softmax activation function.

4. Training and Evaluation

4.1. Training Process

Both classifiers were compiled using the following configuration:

1. **Optimizer:** Adam, an efficient and widely used optimization algorithm.
2. **Loss Function:** sparse_categorical_crossentropy, the standard choice for multi-class classification with integer labels.
3. **Metrics:** accuracy was tracked to monitor performance during training.
4. **Epochs:** Each model was trained for 10 epochs.

4.2. Performance Evaluation

The trained models were evaluated on the unseen test set of 10,000 images. Your specific results were:

1. **Test Accuracy:** The convolutional classifier demonstrated superior performance.
    a. **MLP (ANN) Test Accuracy: 88.39%**
    b. **CNN Test Accuracy: 90.98%**
2. **Confusion Matrix:** A confusion matrix was generated for the convolutional_classifier to analyze its predictive behavior in detail.

5. Discussion and Analysis

Clearly the convolutional classifier (90.98%) performed better than the mlp classifier (88.39%). The large difference in percent, in excess of two-and-one-half, can be explained by the different architecture of the CNN versus the MLP. The CNN has several layers of convolutions and max-pooling that are designed to preserve spatial hierarchies of pixel values; therefore, it will naturally learn translation invariant features such as textures and shapes. The initial Flatten layer of the MLP removes all spatial data, thus preventing the MLP from being able to recognize meaningful visual patterns.

Most misclassifications of the CNN were due to misclassification between visually similar clothing types. Examples include the CNN confusing "Shirt", "T-shirt/top", "Pullover" and "Coat." It would have been reasonable to expect this level of error, given that the images were very small, grayscale and represented the same type of product.

6. Suggested Improvements

While the convolutional_classifier performed well, several techniques could further boost its accuracy:

1. **Data Augmentation:** Artificially increasing the diversity of the training set by applying random transformations like rotations, width/height shifts, and horizontal flips can help the model generalize better.
2. **Dropout Regularization:** Adding Dropout layers to the network is a powerful technique to combat overfitting by randomly setting a fraction of neuron activations to zero during training.
3. **Architectural Enhancements:** Experimenting with a deeper network (more convolutional layers), adjusting the number of filters, or using batch normalization could lead to performance gains.

7. Conclusion

The project was successful in demonstrating both the implementation and comparison of the MLP and CNN models for the purpose of image classification. The results showed that CNNs were particularly well suited to classify images since they were able to achieve a final accuracy rate of **90.98%.** The study also illustrated how it is essential to select the correct model architecture to suit the structure of data.