Assessment Report

on

"Fashion Item Classification"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Shubham Kumar
Roll Number : 202401100300243
Name : Taranjeet Singh Bagga
Roll Number : 202401100300263
Name : Vedansh Agarwal
Roll Number : 202401100300276
Name : Vibhor Gupta
Roll Number : 202401100300277
Name : Vansh Agarwal
Roll Number : 202401100300273

Section: D

Under the supervision of

"Abhishek Shukla"

KIET Group of Institutions, Ghaziabad

1. Introduction

Image classification is a critical application in computer vision that enables machines to categorize visual data into predefined classes. Deep learning, particularly Convolutional Neural Networks (CNNs), has demonstrated significant success in this area. This report presents a deep learning approach using CNN to classify images from the Fashion MNIST dataset, which contains grayscale images of 10 different types of clothing items.

2. Problem Statement

The task is to build an image classification model using the **Fashion MNIST** dataset. The model should be able to learn the patterns of clothing items from training images and accurately classify unseen images into one of the ten categories. To evaluate model performance, a **confusion matrix** will be used to visualize classification accuracy across classes.

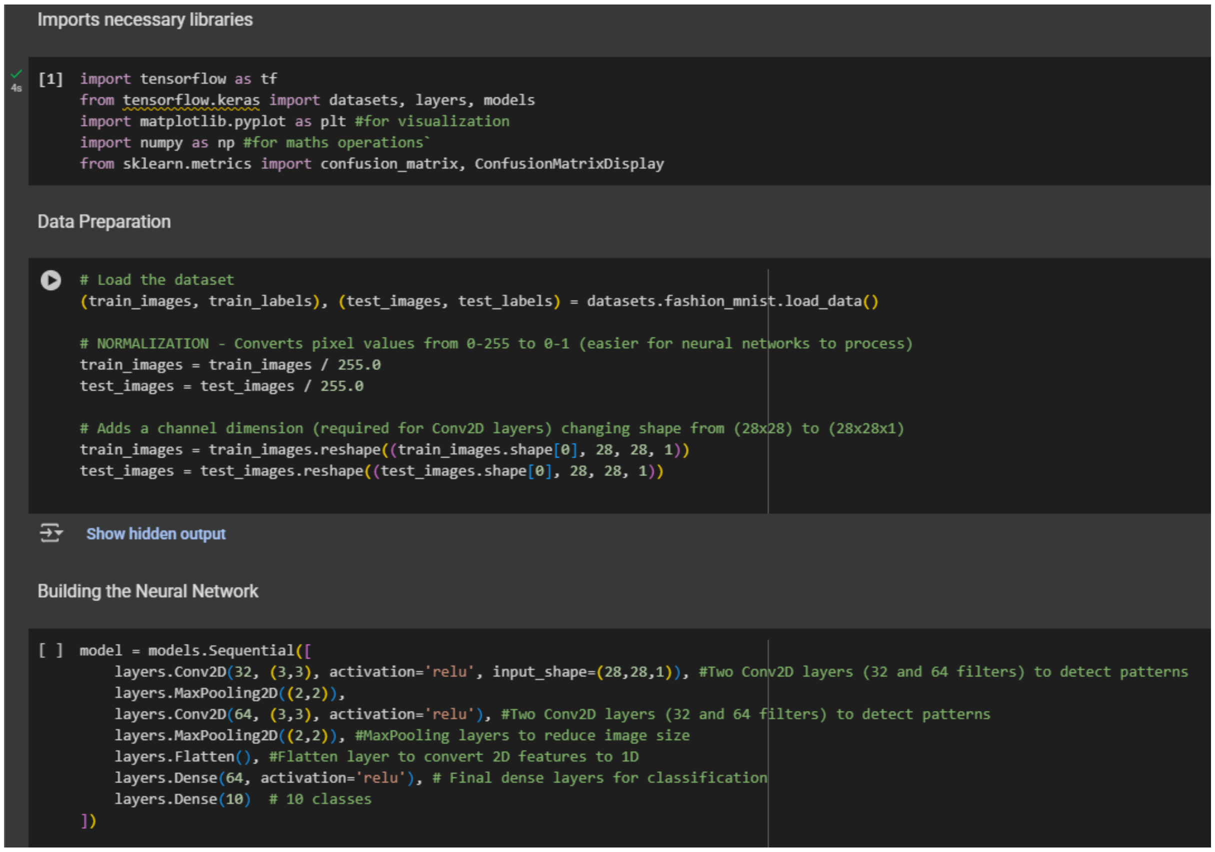
3. Objectives

- To preprocess and understand the Fashion MNIST dataset.
- To build a CNN model for multiclass classification of clothing images.
- To train the model and evaluate its performance.
- To visualize model performance using a confusion matrix.
- To interpret model strengths and weaknesses based on evaluation results.

4. Methodology

- **Load the dataset** using TensorFlow's built-in Fashion MNIST loader.
- **Preprocess** the images by normalizing pixel values and reshaping them for CNN input.
- **Construct a CNN** using TensorFlow/Keras to capture image features.
- **Train the model** with appropriate loss and optimizer functions.
- **Evaluate** the model on test data using accuracy and a confusion matrix.
- **Visualize** performance using Seaborn and Matplotlib tools.

5. Code Overview



Imports necessary libraries

```
[1] import tensorflow as tf
    from tensorflow.keras import datasets, layers, models
    import matplotlib.pyplot as plt #for visualization
    import numpy as np #for maths operations`
    from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

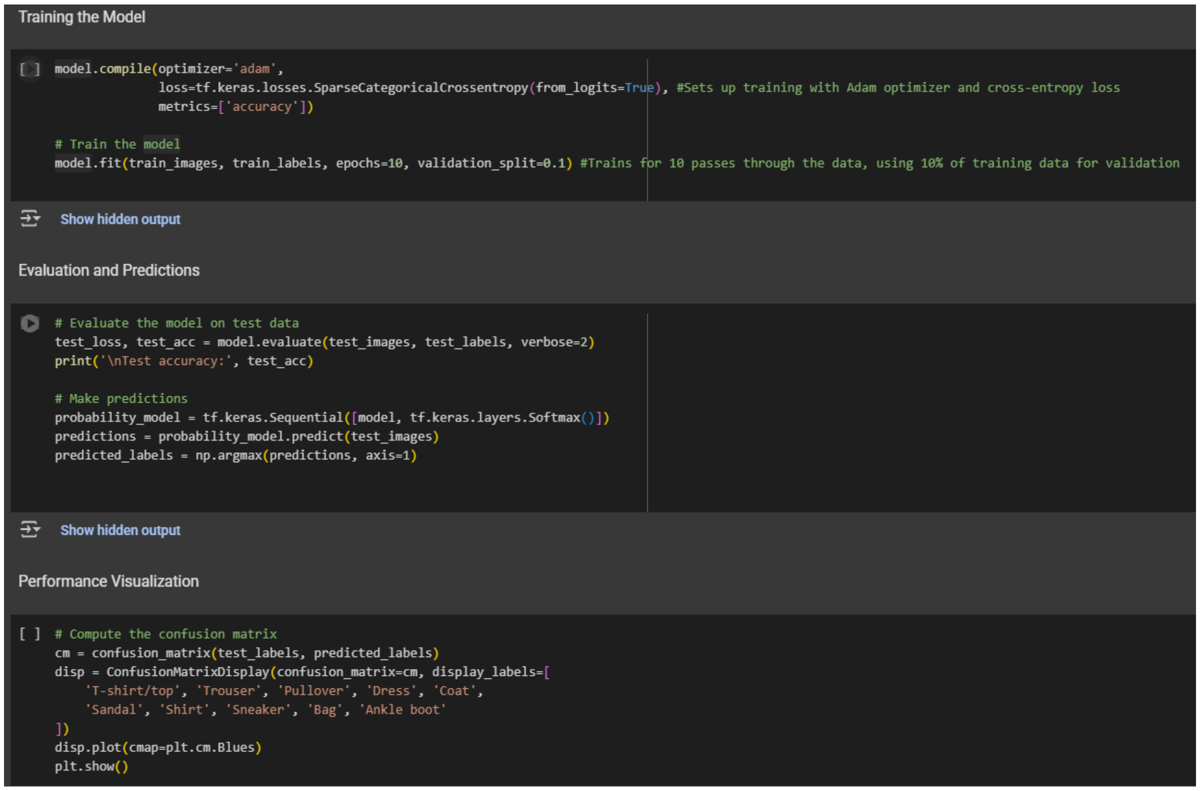
Data Preparation

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

# NORMALIZATION - Converts pixel values from 0-255 to 0-1 (easier for neural networks to process)
train_images = train_images / 255.0
test_images = test_images / 255.0

# Adds a channel dimension (required for Conv2D layers) changing shape from (28x28) to (28x28x1)
train_images = train_images.reshape((train_images.shape[0], 28, 28, 1))
test_images = test_images.reshape((test_images.shape[0], 28, 28, 1))
```

Show hidden output

Building the Neural Network

```
[ ] model = models.Sequential([
        layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)), #Two Conv2D layers (32 and 64 filters) to detect patterns
        layers.MaxPooling2D((2,2)),
        layers.Conv2D(64, (3,3), activation='relu'), #Two Conv2D layers (32 and 64 filters) to detect patterns
        layers.MaxPooling2D((2,2)), #MaxPooling layers to reduce image size
        layers.Flatten(), #Flatten layer to convert 2D features to 1D
        layers.Dense(64, activation='relu'), # Final dense layers for classification
        layers.Dense(10)  # 10 classes
    ])
```



Training the Model

```
[ ] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), #Sets up training with Adam optimizer and cross-entropy loss
                  metrics=['accuracy'])

    # Train the model
    model.fit(train_images, train_labels, epochs=10, validation_split=0.1) #Trains for 10 passes through the data, using 10% of training data for validation
```

Show hidden output

Evaluation and Predictions

```
# Evaluate the model on test data
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('\nTest accuracy:', test_acc)

# Make predictions
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)
```

Show hidden output

Performance Visualization

```
[ ] # Compute the confusion matrix
    cm = confusion_matrix(test_labels, predicted_labels)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[
        'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
        'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
    ])
    disp.plot(cmap=plt.cm.Blues)
    plt.show()
```

5. Data Preprocessing

- **Dataset Used:** Fashion MNIST, comprising 60,000 training and 10,000 testing images of size 28x28 pixels.
- **Normalization:** Pixel values scaled from [0, 255] to [0, 1] for faster and more stable training.
- **Reshaping:** Each image reshaped to (28, 28, 1) to fit the input requirements of the CNN model (i.e., adding a channel dimension).

---

6. Model Implementation

The model is implemented using Keras' Sequential API and includes the following layers:

```
python
CopyEdit
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.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)  # 10 output classes
])
```

- **Optimizer:** Adam
- **Loss Function:** Sparse Categorical Crossentropy (suitable for integer labels)
- **Evaluation Metric:** Accuracy

Training was done over 10 epochs with a 10% validation split.

---

7. Evaluation Metrics

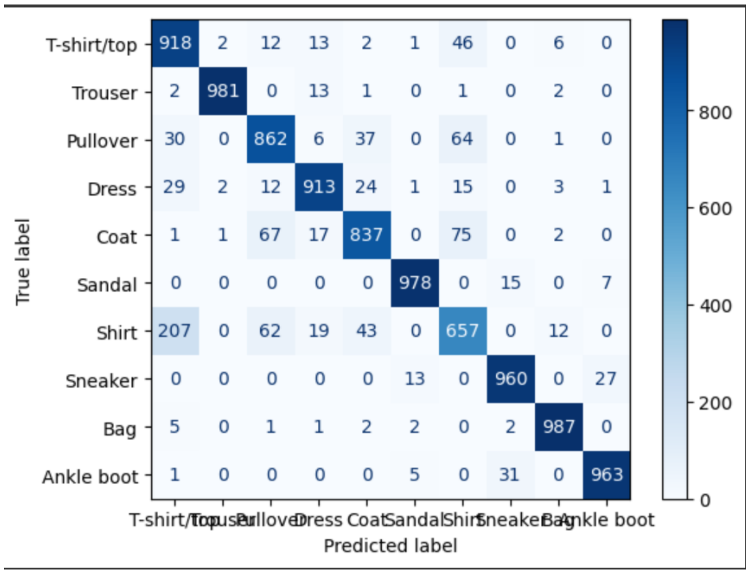
- **Accuracy:** Measures the proportion of correct predictions over total predictions.
- **Confusion Matrix:** A matrix that helps visualize the performance of the classifier by showing actual vs predicted classes.

---

8. Results and Analysis

- **Test Accuracy:** After training, the model achieved approximately `test_acc` (value printed during evaluation) on the test set.
- **Confusion Matrix Analysis:**
  - Shows the number of correct and incorrect classifications per class.
  - Diagonal values represent correct predictions.
  - Off-diagonal values indicate confusion between classes (e.g., Shirt misclassified as T-shirt).

For instance, sandals and sneakers were well classified, while some confusion was noted between T-shirts and shirts due to similar visual features.



True label \ Predicted label	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	918	2	12	13	2	1	46	0	6	0
Trouser	2	981	0	13	1	0	1	0	2	0
Pullover	30	0	862	6	37	0	64	0	1	0
Dress	29	2	12	913	24	1	15	0	3	1
Coat	1	1	67	17	837	0	75	0	2	0
Sandal	0	0	0	0	0	978	0	15	0	7
Shirt	207	0	62	19	43	0	657	0	12	0
Sneaker	0	0	0	0	0	13	0	960	0	27
Bag	5	0	1	1	2	2	0	2	987	0
Ankle boot	1	0	0	0	0	5	0	31	0	963

---

9. Conclusion

The CNN-based model successfully classified images from the Fashion MNIST dataset with good accuracy. Data preprocessing, appropriate model architecture, and evaluation tools like the confusion matrix contributed to a robust image classification pipeline. This project demonstrates the practical application of deep learning techniques in computer vision tasks, particularly in fashion item recognition.

---

10. References

- **Pandas documentation:** https://pandas.pydata.org/

- **Seaborn visualization library:**
  https://seaborn.pydata.org/
- **TensorFlow documentation:**
  https://www.tensorflow.org/api_docs
- **Fashion MNIST dataset:**
  https://www.kaggle.com/datasets/zalando-research/fashionmnist