Week 4.1
Using Keras and Tensorflow in Python to Generate and Evaluate a Basic Neural Network

Produced By: Abdullah Abdul Majid*(linkedin)*

Completion Date: August 3, 2024

Email: abdullahpisk@gmail.com

Jupyter Notebook Links:

GitHub

Table of Contents:

Problem Statement:

Use the Python Tools to import Zelando's Fashion MNIST Dataset, prepare the dataset for model training, and then train a Keras Neural Network on it. Also use callbacks and evaluate the model.

Purpose:

The purpose of this assignment is to prepare an image dataset, then perform and optimize a basic Convoluted Neural Network (CNN) model for it to familiarize ourselves with the Keras and Tensorflow Python ML libraries. These libraries are useful for Deep Learning.

Process:

Importing and Loading the Dataset

- Imported all the necessary libraries, i.e pandas, numpy, sklearn, seaborn, matplotlib and, most importantly tensorflow.
- Import the dataset from Keras/Tensorfow using the following command:
  from tensorflow.keras.datasets import fashion_mnist
- Load the dataset using the following command:
  from tensorflow.keras.datasets import fashion_mnist

Preprocessing the Dataset

- Normalize the dataset to the range [0,1] to make it easier for the Neural Network to process.
- View the dataset shapes:

```
Training images shape: (60000, 28, 28)
Training labels shape: (60000,)
Testing images shape: (10000, 28, 28)
Testing labels shape: (10000,)
```

  The dataset has 60k 28x28 pixel images.
- Since our CNN model requires a color channel parameter too, we reshape the dataset to include a color channel, i.e 1 for grayscale.

Training a Basic Keras CNN Model

- Define the Model Checkpoint callback to save the model (or it's weights) periodically to preserve the model and provide a backup of the training progress in case of a failure.
- Define the Early Stopping callback to finish the CNN training as soon as the model stops improving. This speeds up the training progress and saves up on computing resources.
- Compile the model using the *adam* optimizer and the *SparseCategoricalCrossentropy* loss function, along with the accuracy metric.
- Now train the model using the callbacks and 50 epochs.
- This may take some time, but our model will stop before 50 epochs if the Early Stopping callback takes effect.
- The training process:

```
Epoch 1/50
1858/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7663 - loss: 0.8228
Epoch 1: val_loss improved from inf to 0.38054, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7669 - loss: 0.8199 - val_accuracy: 0.8665 - val_loss: 0.3805
Epoch 2/50
1859/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8742 - loss: 0.3417
Epoch 2: val_loss improved from 0.38054 to 0.34099, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8742 - loss: 0.3417 - val_accuracy: 0.8772 - val_loss: 0.3410
Epoch 3/50
1872/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8924 - loss: 0.2934
Epoch 3: val_loss improved from 0.34099 to 0.33791, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8924 - loss: 0.2934 - val_accuracy: 0.8784 - val_loss: 0.3379
Epoch 4/50
1873/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8984 - loss: 0.2718
Epoch 4: val_loss improved from 0.33791 to 0.33714, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8984 - loss: 0.2718 - val_accuracy: 0.8789 - val_loss: 0.3371
Epoch 5/50
1866/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9047 - loss: 0.2575
Epoch 5: val_loss improved from 0.33714 to 0.32852, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9047 - loss: 0.2575 - val_accuracy: 0.8817 - val_loss: 0.3285
Epoch 6/50
1857/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9099 - loss: 0.2405
Epoch 6: val_loss improved from 0.32852 to 0.31951, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9099 - loss: 0.2405 - val_accuracy: 0.8911 - val_loss: 0.3195
Epoch 7/50
1865/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9188 - loss: 0.2164
Epoch 7: val_loss did not improve from 0.31951
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9187 - loss: 0.2164 - val_accuracy: 0.8878 - val_loss: 0.3242
Epoch 8/50
1858/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9226 - loss: 0.2063
Epoch 8: val_loss did not improve from 0.31951
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9226 - loss: 0.2063 - val_accuracy: 0.8884 - val_loss: 0.3208
Epoch 9/50
1872/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9257 - loss: 0.1952
Epoch 9: val_loss did not improve from 0.31951
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9257 - loss: 0.1952 - val_accuracy: 0.8911 - val_loss: 0.3214
Epoch 10/50
1868/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9288 - loss: 0.1865
Epoch 10: val_loss did not improve from 0.31951
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9288 - loss: 0.1865 - val_accuracy: 0.8945 - val_loss: 0.3309
Epoch 11/50
1871/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9327 - loss: 0.1751
Epoch 11: val_loss did not improve from 0.31951
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9327 - loss: 0.1752 - val_accuracy: 0.8878 - val_loss: 0.3493
Epoch 11: early stopping
Restoring model weights from the end of the best epoch: 6.
```

Model Evaluation

- Get the accuracy by comparing with the test set:

```
313/313 - 0s - 1ms/step - accuracy: 0.8911 - loss: 0.3195

Test accuracy: 0.8910999894142151
```

  The accuracy is good.

- Generate predictions and print the classification report:

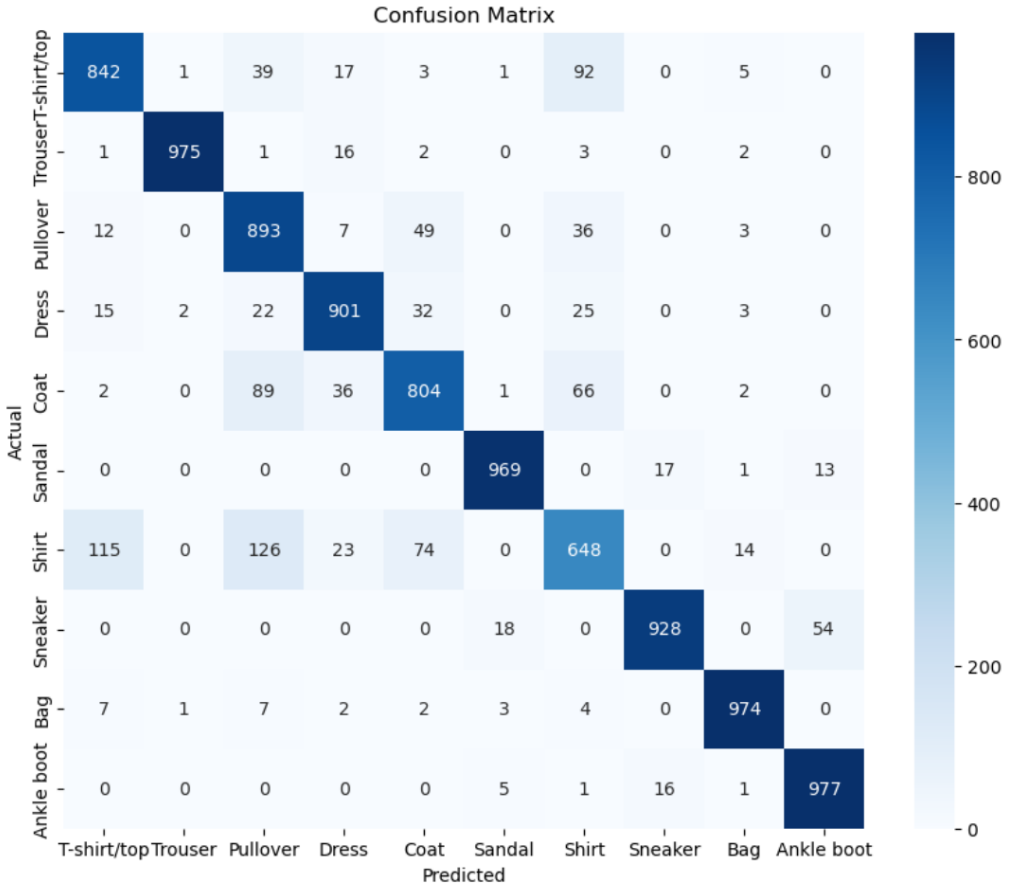
```
              precision    recall  f1-score   support

 T-shirt/top       0.85      0.84      0.84      1000
     Trouser       1.00      0.97      0.99      1000
    Pullover       0.76      0.89      0.82      1000
       Dress       0.90      0.90      0.90      1000
        Coat       0.83      0.80      0.82      1000
      Sandal       0.97      0.97      0.97      1000
       Shirt       0.74      0.65      0.69      1000
     Sneaker       0.97      0.93      0.95      1000
         Bag       0.97      0.97      0.97      1000
  Ankle boot       0.94      0.98      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

The Classification results are pretty good, i.e close to unity.

- Compute and plot the Confusion Matrix:



Our CNN model pretty good as mainly the diagonal elements are hot.

Conclusion

In this assignment, I was able to experiment with Deap Learning using the Tensorflow and Keras tools in Python. I imported the specified dataset, preprocessed it and used callbacks to improve the Neural Network's performance. In the end, I got a very accurate model as evident by the Confusion Matrix and Classification Report evaluations.