Neural Networks for Classifying Fashion MNIST

Htet Aung Myin

March 13, 2020

Abstract

The application of neural networks and convolutional neural network is applied to build a classifier for a data set called Fashion-MNIST, native from the TensorFlow Library.

1 Introduction and Overview

Similar to the the popular MNIST data set, which consisted of handwritten datasets, the Fashion-MNIST dataset contains images of 10 different classes of fashion items. The dataset, consisting of several thousand images are split into training and testing data sets where each set contains a list of images and their associated labels. The labels are as follows:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

2 Theoretical Background

2.1 Neural Network

A neural network, also called an artificial neural network, is based on a collection of nodes called artificial neurons which are loosely modeled after a biological brain. A neural network is a combination of many layers where a linear combination is performed between the previous layer's output and the current layer's weight before passing the data to the next layer through and activation function.

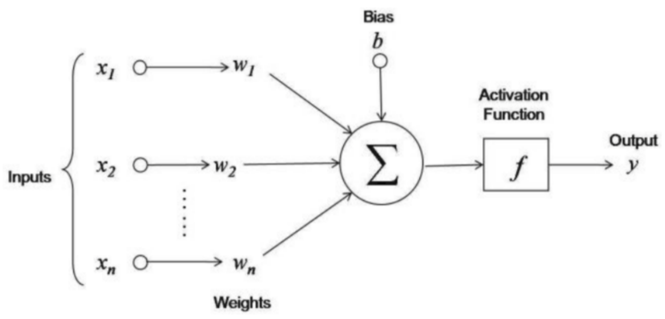


Figure 1: Test 1/3: Layer of a Neural Network

2.2 Convolutional Neural Networks

A Convolutional Neural Network(CNN) is a type of neural network that is mainly used to analyze visual imagery. One of the defining aspects of it is that it has one or more layers of convolution units. The layers of a CNN consists of an input layer, an output layer, and a hidden layer that includes multiple convolution layers, pooling layers, fully connected layers and normalization layers.

3 Algorithm Implementation and Development

The data set consisting of 10 different classes of fashion items are loaded from TensorFlow.

1. The dataset is then split into training and test data.
2. The training data is then split into training and validation data and also normalized and converted to float from uint8.
3. A model is constructed with varying number of layers and density. The best result is selected through trial and error.
4. The model is fitted to the training data and accuracy of the training and validation data is determined. If the training data is higher than the validation data, the model is adjusted.
5. The accuracy and loss is plotted and a confusion matrix is displayed.
6. This process is repeated with a convolutional neural network with the addition of a convolutional layer where the hyper-parameters are adjusted based on trial and error.

4 Computational Results

4.1 Fully Connected Neural Network

The Fully Connected Neural Network ran over 40 epochs, with the best validation accuracy at 86.36% and training accuracy at 86.31% at the 27th epoch. At a greater epoch, there are signs of over fitting as the training accuracy is larger than the validation accuracy. A training accuracy greater than 90% was possible but there would be severe over fitting issues.

The Fully Connected Neural Network had the most trouble identifying Shirt(0) and T-Shirts(6) in both the prediction and test data as seen in figure 4 and 5.

4.2 Convolutional Neural Network

The CNN ran for 10 epochs due to time and memory constraints as many convolution layers were defined. The model best performed at its 7th epoch where the validation accuracy was at 90.02 % and the training accuracy was at 90.47%. Any greater epochs showed signs of overfitting.

Similarly, the CNN had the most trouble identifying Shirt(0) and T-Shirts(6) in both the prediction and test data as seen in figure 6 and 7.

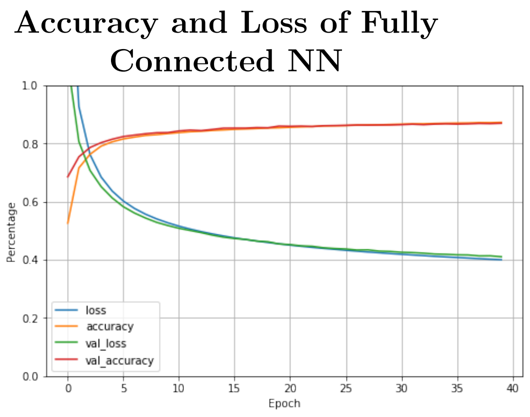


Figure 2: Fully Connected Neural Network

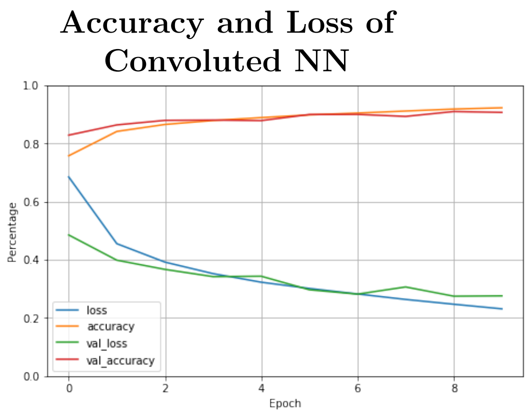
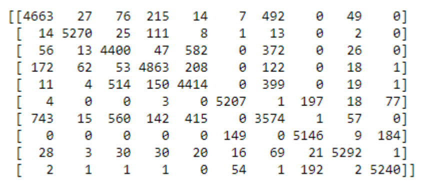
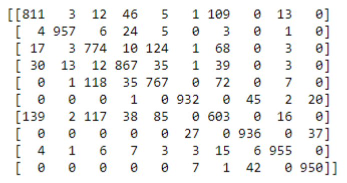


Figure 3: Convolutional Neural Network



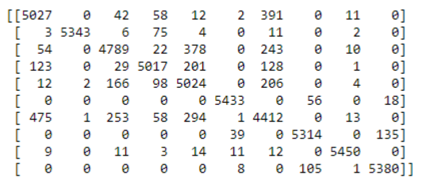
```
[[4663   27   76  215   14    7  492    0   49    0]
 [  14 5270   25  111    8    1   13    0    2    0]
 [  56   13 4400   47  582    0  372    0   26    0]
 [ 172   62   53 4863  208    0  122    0   18    1]
 [  11    4  514  150 4414    0  399    0   19    1]
 [   4    0    0    3    0 5207    1  197   18   77]
 [ 743   15  560  142  415    0 3574    1   57    0]
 [   0    0    0    0    0  149    0 5146    9  184]
 [  28    3   30   30   20   16   69   21 5292    1]
 [   2    1    1    1    0   54    1  192    2 5240]]
```

Figure 4: Confusion Matrix for Fully Connected NN (Training)



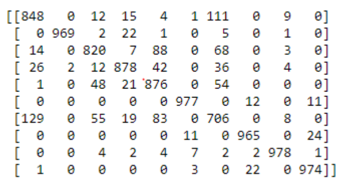
```
[[811   3  12  46   5   1 109   0  13   0]
 [  4 957   6  24   5   0   3   0   1   0]
 [ 17   3 774  10 124   1  68   0   3   0]
 [ 30  13  12 867  35   1  39   0   3   0]
 [  0   1 118  35 767   0  72   0   7   0]
 [  0   0   0   1   0 932   0  45   2  20]
 [139   2 117  38  85   0 603   0  16   0]
 [  0   0   0   0   0  27   0 936   0  37]
 [  4   1   6   7   3   3  15   6 955   0]
 [  0   0   0   0   0   7   1  42   0 950]]
```

Figure 5: Confusion Matrix for Fully Connected NN (Test)



```
[[5027    0   42   58   12    2  391    0   11    0]
 [   3 5343    6   75    4    0   11    0    2    0]
 [  54    0 4789   22  378    0  243    0   10    0]
 [ 123    0   29 5017  201    0  128    0    1    0]
 [  12    2  166   98 5024    0  206    0    4    0]
 [   0    0    0    0    0 5433    0   56    0   18]
 [ 475    1  253   58  294    1 4412    0   13    0]
 [   0    0    0    0    0   39    0 5314    0  135]
 [   9    0   11    3   14   11   12    0 5450    0]
 [   0    0    0    0    0    8    0  105    1 5380]]
```

Figure 6: Confusion Matrix for CNN (Training)



```
[[848   0  12  15   4   1 111   0   9   0]
 [  0 969   2  22   1   0   5   0   1   0]
 [ 14   0 820   7  88   0  68   0   3   0]
 [ 26   2  12 878  42   0  36   0   4   0]
 [  1   0  48  21 876   0  54   0   0   0]
 [  0   0   0   0   0 977   0  12   0  11]
 [129   0  55  19  83   0 706   0   8   0]
 [  0   0   0   0   0  11   0 965   0  24]
 [  0   0   4   2   4   7   2   2 978   1]
 [  1   0   0   0   0   3   0  22   0 974]]
```

Figure 7: Confusion Matrix for CNN (Test)

5 Summary & Conclusion

A fully connected neural network and a convolutional neural network was built to classify categories from the Fashion-MNIST dataset. The CNN performed slightly better compared to the fully connected neural network but in time performance, the fully connected neural network performed better. Both networks had issues in identifying a shirt or a t-shirt, possibly due to similarities.

6 Appendix A.

- tf.keras.datasets = a public API for datasets
- functool.partial = returns a new partial object when called and behaves like a func with arguments
- tf.keras.layers.Sequential = a linear stack of layers
- tf.keras.layers.Dense = a regular densly connected NN layer
- tf.keras.layers.Conv2D = a 2D convolution layer

7 Appendix B.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import confusion_matrix


# In[2]:


fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_da


# In[3]:


plt.figure()
for k in range(10):
    plt.subplot(3,4,k+1)
    plt.imshow(X_train_full[k], cmap="gray")
    plt.axis('off')
plt.show()


# In[4]:


X_valid = X_train_full[:5000] / 255.0
X_train = X_train_full[5000:] / 255.0
X_test = X_test / 255.0
```

```
y_valid = y_train_full[:5000]
y_train = y_train_full[5000:]


# In[5]:


from functools import partial

my_dense_layer = partial(tf.keras.layers.Dense, activation="relu", ker

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28, 28]),

    my_dense_layer(128),
    #my_dense_layer(64),
    my_dense_layer(32),
    #my_dense_layer(128),

    #my_dense_layer(400),
   # my_dense_layer(100),
   # my_dense_layer(100),


    #my_dense_layer(600),
    #my_dense_layer(400),
    #my_dense_layer(200),
    my_dense_layer(10, activation="softmax")
])


# In[6]:


model.compile(loss="sparse_categorical_crossentropy",
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001)
              metrics=["accuracy"])


# validation below training -> not overfitting. validation go down wh
# In[7]:


history = model.fit(X_train, y_train, epochs=40, validation_data=(X_va


# In[8]:


pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.ylabel("Percentage")
plt.xlabel("Epoch")
plt.show()


# In[9]:


model.evaluate(X_test,y_test)


# In[10]:


y_pred = model.predict_classes(X_train)
conf_train = confusion_matrix(y_train, y_pred)
print(conf_train)


# In[11]:


y_pred = model.predict_classes(X_test)
conf_test = confusion_matrix(y_test, y_pred)
print(conf_test)


# CNN
# In[12]:


fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_da
X_valid = X_train_full[:5000] / 255.0
X_train = X_train_full[5000:] / 255.0
X_test = X_test / 255.0

y_valid = y_train_full[:5000]
y_train = y_train_full[5000:]


# In[29]:


plt.figure()
for k in range(10):
    plt.subplot(3,4,k+1)
    plt.imshow(X_train_full[k], cmap="gray")
    plt.axis('off')
    plt.title(y_train_full[k])
plt.show()


# In[13]:


X_train = X_train[..., np.newaxis]
X_valid = X_valid[..., np.newaxis]
X_test = X_test[..., np.newaxis]


# In[14]:


my_dense_layer = partial(tf.keras.layers.Dense, activation="relu", ker
my_conv_layer = partial(tf.keras.layers.Conv2D, activation="relu", pad

model = tf.keras.models.Sequential([
    my_conv_layer(6,5,padding="same",input_shape=[28,28,1]),
    tf.keras.layers.AveragePooling2D(2),
   # my_conv_layer(8,5),
    #tf.keras.layers.AveragePooling2D(2),
    my_conv_layer(32,3),
    my_conv_layer(64,3),
    my_conv_layer(128,3),
    my_conv_layer(256,3),
    my_conv_layer(512,3),
    tf.keras.layers.Flatten(),
    my_dense_layer(256),
    #my_dense_layer(32),
    #my_dense_layer(8),
    my_dense_layer(10, activation="softmax")
])


# In[15]:


model.compile(loss="sparse_categorical_crossentropy",
             optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
             metrics=["accuracy"])


# In[16]:


history = model.fit(X_train, y_train, epochs=10, validation_data=(X_va


# In[17]:


pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.ylabel("Percentage")
plt.xlabel("Epoch")
plt.show()
```

```
# In[18]:


y_pred = model.predict_classes(X_train)
conf_train = confusion_matrix(y_train, y_pred)
print(conf_train)


# In[19]:


model.evaluate(X_test,y_test)


# In[20]:


y_pred = model.predict_classes(X_test)
conf_test = confusion_matrix(y_test, y_pred)
print(conf_test)


# In[ ]:
```