Neural Network Analysis

In [ ]:

```
import numpy as np
import pandas as pd

from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
```

By using the Fashion_mnist data set we are going to display the neural network analysis

In [7]:

```
    (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz (https://storage.g
oogleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz)
29515/29515 [==============================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz (https://storage.g
oogleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz)
26421880/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz (https://storage.go
ogleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz)
5148/5148 [==============================] - 0s 0s/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz (https://storage.go
ogleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz)
4422102/4422102 [==============================] - 0s 0us/step
```

In [8]:

```
x_train = x_train.reshape(x_train.shape[0], -1) / 255.0
x_test = x_test.reshape(x_test.shape[0], -1) / 255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [15]:

```
print(f'x_train = {y_train.shape}')
```

```
x_train = (60000, 10)
```

In [16]:

```
print(f'y_train = {y_train.shape}')
```

```
y_train = (60000, 10)
```

first by using the 10% training data as the validation data,

In [17]:

```
model = Sequential()
model.add(Dense(10, input_dim=784, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [18]:

```
model.fit(x_train, y_train, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
1688/1688 [==============================] - 2s 935us/step - loss: 0.6669 - accuracy: 0.7694 - val_loss: 0.4951 - val_accuracy:
0.8280
Epoch 2/10
1688/1688 [==============================] - 1s 806us/step - loss: 0.4745 - accuracy: 0.8364 - val_loss: 0.4530 - val_accuracy:
0.8417
Epoch 3/10
1688/1688 [==============================] - 1s 794us/step - loss: 0.4455 - accuracy: 0.8448 - val_loss: 0.4413 - val_accuracy:
0.8403
Epoch 4/10
1688/1688 [==============================] - 1s 810us/step - loss: 0.4283 - accuracy: 0.8512 - val_loss: 0.4316 - val_accuracy:
0.8485
Epoch 5/10
1688/1688 [==============================] - 1s 807us/step - loss: 0.4177 - accuracy: 0.8544 - val_loss: 0.4218 - val_accuracy:
0.8565
Epoch 6/10
1688/1688 [==============================] - 1s 815us/step - loss: 0.4086 - accuracy: 0.8562 - val_loss: 0.4368 - val_accuracy:
0.8468
Epoch 7/10
1688/1688 [==============================] - 1s 803us/step - loss: 0.4048 - accuracy: 0.8598 - val_loss: 0.4201 - val_accuracy:
0.8525
Epoch 8/10
1688/1688 [==============================] - 1s 805us/step - loss: 0.3979 - accuracy: 0.8606 - val_loss: 0.4362 - val_accuracy:
0.8473
Epoch 9/10
1688/1688 [==============================] - 1s 800us/step - loss: 0.3947 - accuracy: 0.8626 - val_loss: 0.4208 - val_accuracy:
0.8532
Epoch 10/10
1688/1688 [==============================] - 1s 798us/step - loss: 0.3913 - accuracy: 0.8632 - val_loss: 0.4238 - val_accuracy:
0.8533
```

Out[18]:

```
<keras.callbacks.History at 0x251b4b23760>
```

In [19]:

```
_, test_acc = model.evaluate(x_test, y_test)
print(test_acc)
```

```
313/313 [==============================] - 0s 639us/step - loss: 0.4598 - accuracy: 0.8407
0.8406999707221985
```

Make the network wider we increased these from 10 to 50.

In [20]:

```
model2 = Sequential()
model2.add(Dense(50, input_dim=784, activation='relu'))
model2.add(Dense(10, activation='softmax'))
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model2.fit(x_train, y_train, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5363 - accuracy: 0.8118 - val_loss: 0.4215 - val_accuracy:
0.8537
Epoch 2/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.4049 - accuracy: 0.8571 - val_loss: 0.4145 - val_accuracy:
0.8578
Epoch 3/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3689 - accuracy: 0.8688 - val_loss: 0.3768 - val_accuracy:
0.8630
Epoch 4/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3409 - accuracy: 0.8765 - val_loss: 0.3568 - val_accuracy:
0.8727
Epoch 5/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3228 - accuracy: 0.8831 - val_loss: 0.3527 - val_accuracy:
0.8752
Epoch 6/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3107 - accuracy: 0.8869 - val_loss: 0.3383 - val_accuracy:
0.8790
Epoch 7/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2974 - accuracy: 0.8922 - val_loss: 0.3427 - val_accuracy:
0.8798
Epoch 8/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2857 - accuracy: 0.8946 - val_loss: 0.3306 - val_accuracy:
0.8750
Epoch 9/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2771 - accuracy: 0.8986 - val_loss: 0.3266 - val_accuracy:
0.8813
Epoch 10/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2666 - accuracy: 0.9024 - val_loss: 0.3226 - val_accuracy:
0.8812
```

Out[20]:

```
<keras.callbacks.History at 0x2518ea47820>
```

In [21]:
```
_, test_acc = model2.evaluate(x_test, y_test)
print(test_acc)
```

```
313/313 [==============================] - 0s 688us/step - loss: 0.3471 - accuracy: 0.8743
0.8743000030517578
```

To get more deeper network we will add another 50%

In [22]:
```
model3 = Sequential()
model3.add(Dense(50, input_dim=784, activation='relu'))
model3.add(Dense(50, activation='relu'))
model3.add(Dense(10, activation='softmax'))
model3.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model3.fit(x_train, y_train, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.5368 - accuracy: 0.8121 - val_loss: 0.4108 - val_accuracy:
0.8553
Epoch 2/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3935 - accuracy: 0.8589 - val_loss: 0.3878 - val_accuracy:
0.8603
Epoch 3/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3557 - accuracy: 0.8696 - val_loss: 0.3803 - val_accuracy:
0.8595
Epoch 4/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3318 - accuracy: 0.8792 - val_loss: 0.3454 - val_accuracy:
0.8718
Epoch 5/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.3134 - accuracy: 0.8843 - val_loss: 0.3590 - val_accuracy:
0.8733
Epoch 6/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2993 - accuracy: 0.8891 - val_loss: 0.3522 - val_accuracy:
0.8722
Epoch 7/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2884 - accuracy: 0.8932 - val_loss: 0.3333 - val_accuracy:
0.8783
Epoch 8/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2795 - accuracy: 0.8969 - val_loss: 0.3535 - val_accuracy:
0.8692
Epoch 9/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2687 - accuracy: 0.9006 - val_loss: 0.3312 - val_accuracy:
0.8785
Epoch 10/10
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2609 - accuracy: 0.9030 - val_loss: 0.3383 - val_accuracy:
0.8802
```

Out[22]:
```
<keras.callbacks.History at 0x2518ed72500>
```

In [23]:
```
_, test_acc = model3.evaluate(x_test, y_test)
print(test_acc)
```

```
313/313 [==============================] - 0s 752us/step - loss: 0.3589 - accuracy: 0.8781
0.8780999779701233
```

Convolutional neural network

In [24]:
```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten
import numpy as np
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
x_train = x_train[:,:,:,np.newaxis] / 255.0
x_test = x_test[:,:,:,np.newaxis] / 255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [26]:
```
model4 = Sequential()
model4.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(28,28, 1)))
model4.add(MaxPooling2D(pool_size=2))
model4.add(Flatten())
model4.add(Dense(10, activation='softmax'))
model4.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
In [27]: model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0         
 )                                                               
                                                                 
 flatten (Flatten)           (None, 12544)             0         
                                                                 
 dense_7 (Dense)             (None, 10)                125450    
                                                                 
=================================================================
Total params: 125,770
Trainable params: 125,770
Non-trainable params: 0
_________________________________________________________________
```

```
In [28]: model4.fit(x_train, y_train, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.4330 - accuracy: 0.8485 - val_loss: 0.3295 - val_accuracy:
0.8823
Epoch 2/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.3135 - accuracy: 0.8891 - val_loss: 0.3090 - val_accuracy:
0.8915
Epoch 3/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2801 - accuracy: 0.9015 - val_loss: 0.3003 - val_accuracy:
0.8928
Epoch 4/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2580 - accuracy: 0.9082 - val_loss: 0.2918 - val_accuracy:
0.8948
Epoch 5/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2419 - accuracy: 0.9135 - val_loss: 0.2761 - val_accuracy:
0.9025
Epoch 6/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2272 - accuracy: 0.9189 - val_loss: 0.2758 - val_accuracy:
0.9018
Epoch 7/10
1688/1688 [==============================] - 14s 8ms/step - loss: 0.2167 - accuracy: 0.9227 - val_loss: 0.2677 - val_accuracy:
0.9080
Epoch 8/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.2046 - accuracy: 0.9257 - val_loss: 0.2707 - val_accuracy:
0.9035
Epoch 9/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.1959 - accuracy: 0.9295 - val_loss: 0.2782 - val_accuracy:
0.9028
Epoch 10/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.1857 - accuracy: 0.9336 - val_loss: 0.2832 - val_accuracy:
0.9062
```

```
Out[28]: <keras.callbacks.History at 0x251901b2770>
```

The accuracy level is 93%

```
In [ ]: 
```