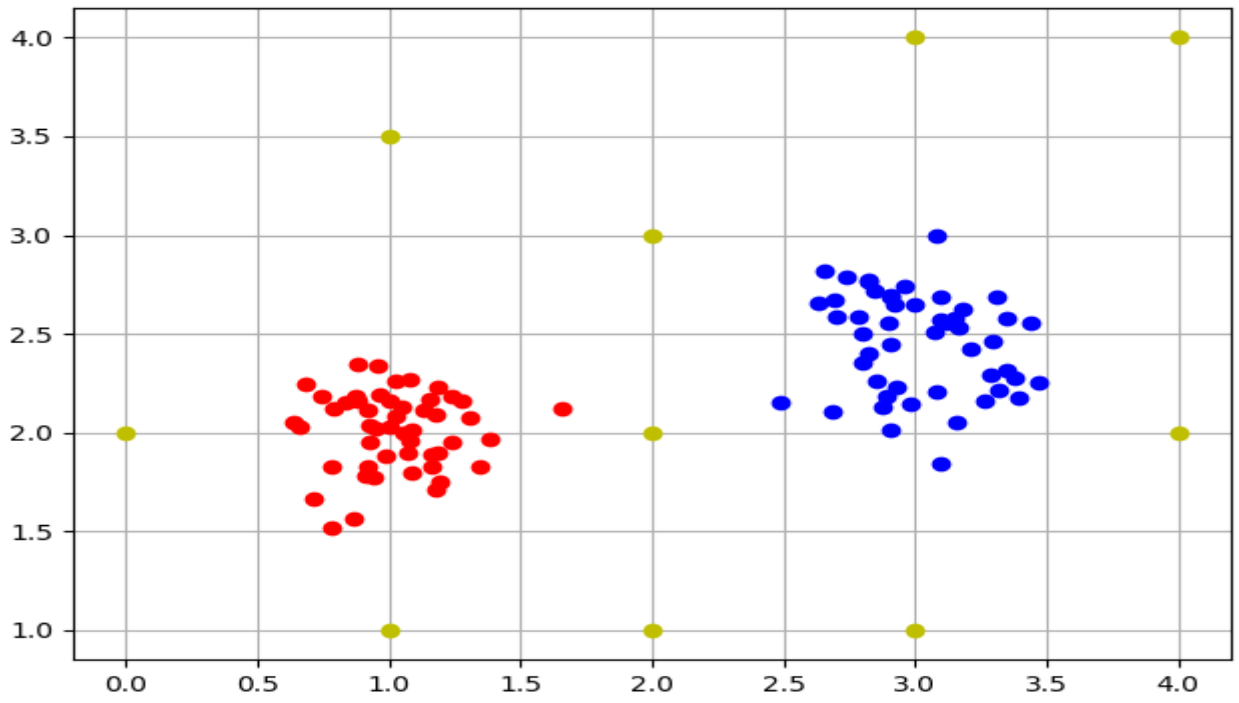
سؤال ۱:

در این سؤال می‌خواهیم یک شبکه مادلاین ای طراحی کنیم که ۲ دسته از نقاط را در ۲ کلاس مختلف از سایر نقاط آن صفحه جدا کنیم

برای این کار ابتدا باید نقاط را در صفحه بکشیم این کار را مانند زیر انجام می دهیم:

```
Orange_x1 = np.random.normal(3, 0.2, 50)
Orange_x2 = np.random.normal(2.5, 0.3, 50)

Green_x1 = np.random.normal(1, 0.2, 50)
Green_x2 = np.random.normal(2, 0.2, 50)

Blue_x1 = [0, 1, 1, 2, 2, 2, 3, 3, 4, 4]
Blue_x2 = [2, 1, 3.5, 1, 2, 3, 1, 4, 2, 4]

plt.plot(Orange_x1, Orange_x2, "bo")
plt.plot(Green_x1, Green_x2, "ro")
plt.plot(Blue_x1, Blue_x2, "yo")

plt.grid()
plt.show()
```

که بعد از اجرای آن شکل زیر بدست می اید:



همان‌طور که دیده می‌شود نقاط قرمز نورون خروجی اول و نقاط آبی نورون خروجی دوم را تشکیل می‌دهند و بقیه نقاط برای هیچ یک از این دو خروجی نیستند

سپس یک دیتا ست با استفاده از این نقاط بدست می آوریم:

```
for i in range(len(Orange_x1)):
    dataset_list.append([Orange_x1[i], Orange_x2[i], [-1, 1]])
for i in range(len(Green_x1)):
    dataset_list.append([Green_x1[i], Green_x2[i], [1, -1]])
for i in range(len(Blue_x1)):
    dataset_list.append([Blue_x1[i], Blue_x2[i], [-1, -1]])
```

که در آن نقاط و خروجی مربوط به آن‌ها را به آن اضافه کرده‌ایم

سپس این دیتا ست را به شبکه می‌دهیم تا خط ها را جدا کند:

```
weights, bias = train_weights_madaline_mr1(x_in=dataset_list, number_of_layer1_neuron=7, number_of_and_input=2, l_rate=0.2, n_epoch=epoch)
```

برای طراحی این شبکه از الگوریتم mr1 در شبکه مادلاین استفاده می‌کنیم
در این الگوریتم از ۱ لایه مخفی و یک لایه که متشکل از ۲ نورون اند است استفاده می‌کنیم
ابتدا وزن های اولیه را مقدار می‌دهیم برای هر دو لایه:

```
input_weights_vector = [[0.0 for i in range(len(x_in[0]) - 1)] for j in range(number_of_layer1_neuron)]
input_bias_vector = [0.0 for i in range(number_of_layer1_neuron)]
and1_weights_vector = [[1, 1, 1, 0, 0, 0, 0], [0, 0, 0, 1, 1, 1, 1]]
and1_bias_vector = [-2, -3]

predict_layer1_value = [0.0 for i in range(number_of_layer1_neuron)]
predict_and_value = [0.0 for i in range(number_of_and_input)]
```

سپس ابتدا به ازای تمام ایپاک ها به ازای هر دیتا ابتدا نت و اکتیویشن لایه یک را بدست می اوریم:

```
for neuron_layer1_number in range(number_of_layer1_neuron):
    
    activation_layer1[neuron_layer1_number] = find_net(data,
                                                       input_weights_vector[neuron_layer1_number],
                                                       input_bias_vector[neuron_layer1_number])
    predict_layer1_value[neuron_layer1_number] = predict_step(activation_layer1)
```

سپس به ازای هر اند در لایه آخر مقدار نت و اکتیویشن آن را حساب می‌کنیم

```
for and_number in range(number_of_and_input):
    activation_and = find_net(predict_layer1_value,
                              and1_weights_vector[and_number],
                              and1_bias_vector[and_number])
    predict_and_value[and_number] = predict_step(activation_and)
```

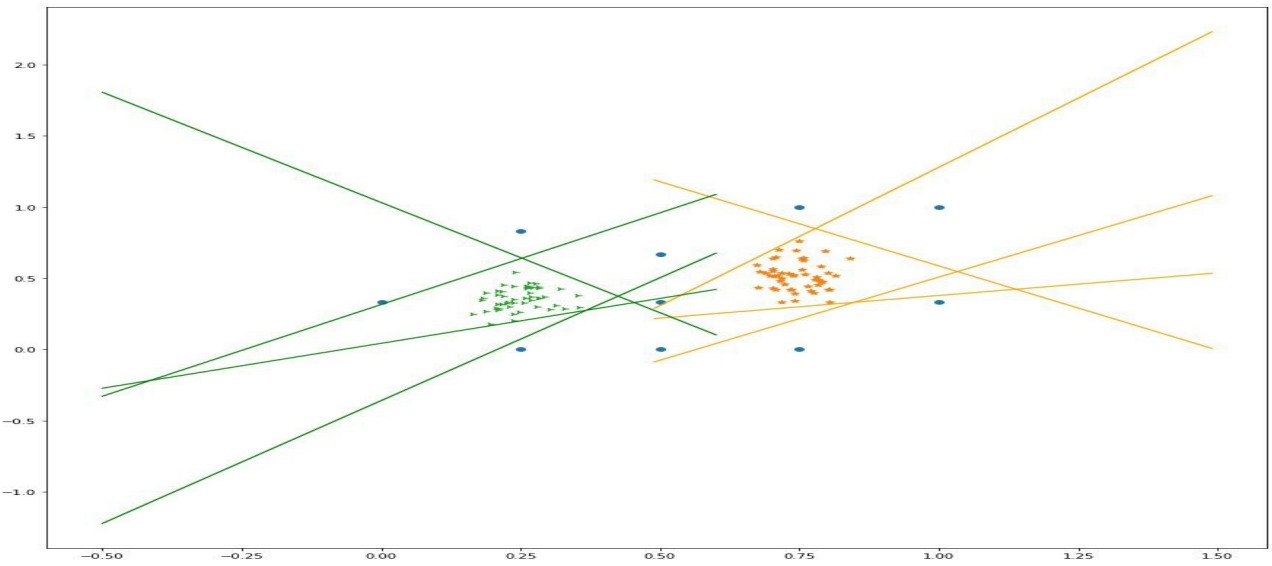
سپس ارور را بدست می اوریم:

```
error = x_in[2][and_number] - predict_and_value[and_number]
```

سپس طبق الگوریتم mr1 برای ارور زیر وقتی تارگت ۱ یا تارگت -۱ است مقدار وزن ها را متناسب با آن و الگوریتم اپدیت می کنیم:

```
if error != 0 and x_in[2][and_number] == 1:
    for neuron_layer1_number in range(number_of_layer1_neuron):
        if predict_layer1_value[neuron_layer1_number] < 0:
            input_bias_vector[neuron_layer1_number] = input_bias_vector[neuron_layer1_number] + l_rate * (activation_layer1[number_of_layer1_neuron] - x
            for input_number in range(data):
                input_weights_vector[neuron_layer1_number][input_number] = input_weights_vector[neuron_layer1_number] + l_rate * (activation_layer1[numb

if error != 0 and x_in[2][and_number] == -1:
    for neuron_layer1_number in range(number_of_layer1_neuron):
        if predict_layer1_value[neuron_layer1_number] > 0:
            input_bias_vector[neuron_layer1_number] = input_bias_vector[neuron_layer1_number] + l_rate * (activation_layer1[number_of_layer1_neuron] - x
            for input_number in range(data):
                input_weights_vector[neuron_layer1_number][input_number] = input_weights_vector[neuron_layer1_number] + l_rate * (activation_layer1[numb
```

این کار را انقدر تکرار می‌کنیم تا خطای شبکه کم شود و شبکه خط‌های جدا کننده را به درستی بکشد



سؤال ۲:

در این سؤال می‌خواهیم از روی دیتا های اکسل مربوط به خانه‌ها قیمت های آن را پیشبینی کنیم
از ۴۰۰۰ داده اول برای آموزش و ۱۰۰۰ داده بعدی برای تست استفاده می کنیم

برای این کار یک بار از شبکه با ۱ لایه مخفی و یک بار با ۲ لایه مخفی انجام می‌دهیم

برای شبکه با ۱ لایه داریم:

ابتدا دیتا ستی از اکسل مربوطه می‌سازیم و می‌دانیم ۲ ستون اول که شامل ایدی هر خانه و تاریخ آن است برای شبکه لازم نیست پس داریم:

```
dataset = pd.read_csv("/home/sspc/Desktop/Neural Networks/MLP Madaline DimensionalityReduction/House Sales.csv")
dataset = dataset.iloc[:5000]

x_train = dataset.values[:4000, 2:]
y_train = dataset.values[:4000, 2]

x_test = dataset.values[4000:, 2:]
y_test = dataset.values[4000:, 2]
print(x_test)
```

سپس یک مدل با یک لایه مخفی مانند شکل زیر می سازیم:

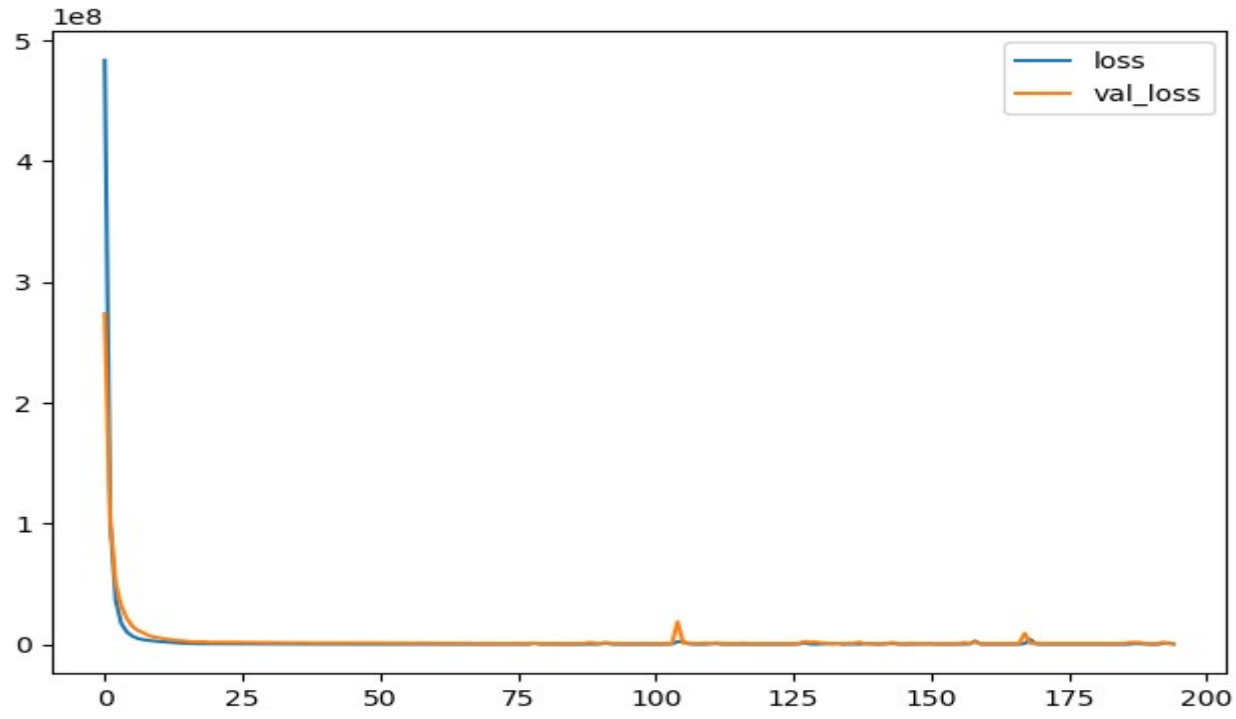
```
my_model = Sequential()

my_model.add(Dense(units=19, activation=relu, input_shape=(19,)))
my_model.add(Dense(units=1))

my_model.compile(optimizer="Adam", loss='MSE', metrics=['mean_squared_logarithmic_error'])

my_train = my_model.fit(x=x_train, y=y_train, batch_size=32, epochs=200, validation_split=0.2)
```

که در آن ۱۰ نورون در لایه اول و اکتیویشن relu دارد و مدل را با batch_size 10 و برای 10 ایپاک با متریس msle و لاس MSE و اپتیمایزر Adam آموزش می‌دهیم و از 0.2 داده‌ها برای ولیدیشن استفاده می‌کنیم

سپس مقادیر لاس و دقت را در نمودار های جدا گانه کشیده و انهارا گزارش می کنیم:

```
history = my_train.history
test_loss, test_acc = my_model.evaluate(x=x_test, y=y_test)

predicted_labels = my_model.predict(x_test)

print(test_loss)
print(test_acc)
print(predicted_labels)

print(history)
print(history["loss"])
print(history["val_loss"])

plt.plot(history["loss"][5:])
plt.plot(history["val_loss"][5:])
plt.legend(["loss", "val_loss"])
plt.show()
```

بعد از اجرای آن داریم:



که در آن مقدار loss و val_loss در نمودار کشیده شده‌اند
و برای آموزش داریم:

```
loss: 80095.9062 - mean_squared_logarithmic_error: 5.1391e-07 - val_loss: 525532.4309 - val_mean_squared_logarithmic_error: 4.4545e-07
```

```
start = datetime.datetime.now()
one_layer_network_prediction()
end = datetime.datetime.now()
print(end - start)
```

```
 32/3200 [..............................] - ETA: 0s - loss: 4867.1475 - mean_squared_logarithmic_error: 2.5836e-08
 992/3200 [========>.....................] - ETA: 0s - loss: 123368.7236 - mean_squared_logarithmic_error: 4.3618e-07
1952/3200 [=================>............] - ETA: 0s - loss: 108726.3691 - mean_squared_logarithmic_error: 3.8273e-07
2816/3200 [=========================>....] - ETA: 0s - loss: 87689.9675 - mean_squared_logarithmic_error: 3.5516e-07
3200/3200 [==============================] - 0s 63us/step - loss: 80095.9062 - mean_squared_logarithmic_error: 5.1391e-07 - val_loss: 525532.4309 - val_m

 32/1000 [..............................] - ETA: 0s
1000/1000 [==============================] - 0s 23us/step
48023.6625
4.860252715843671e-07
```

و برای آموزش و تست آن در آخرین مرحله داریم:

برای حالت با ۲ لایه مخفی داریم:

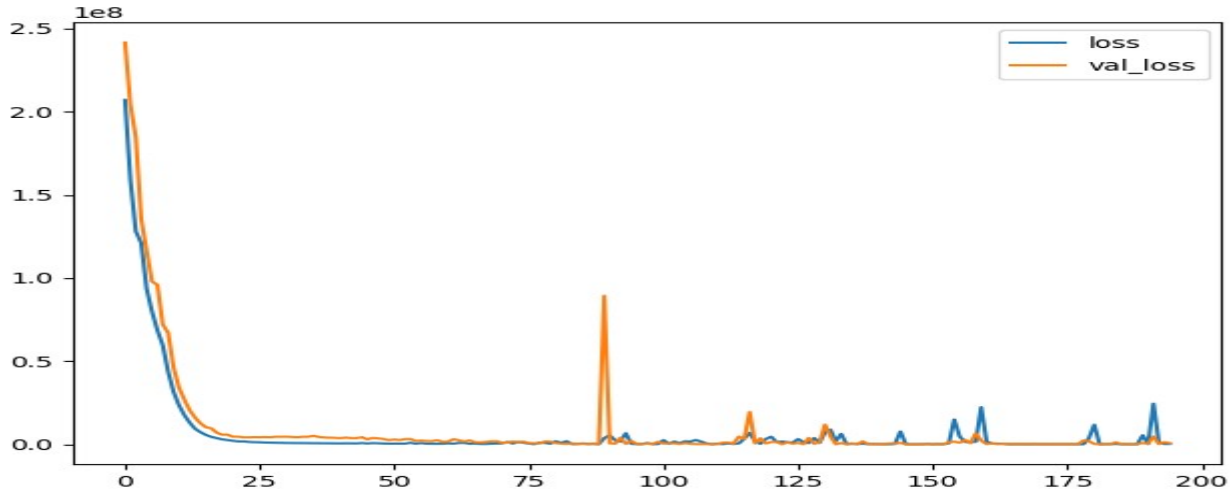
ابتدا مانند قبل دیتا ست را می سازیم:

```
dataset = pd.read_csv("/home/sspc/Desktop/Neural Networks/MLP_Madaline_DimensionalityReduction/House_Sales.csv")
dataset = dataset.iloc[:5000]

x_train = dataset.values[:4000, 2:]
y_train = dataset.values[:4000, 2]

x_test = dataset.values[4000:, 2:]
y_test = dataset.values[4000:, 2]
print(x_test)
```

سپس مانند زیر مدل شبکه را ساخته و آن را آموزش می دهیم:

```
my_model = Sequential()

my_model.add(Dense(units=19, activation='relu', input_shape=(19,)))
my_model.add(Dense(units=19, activation='relu'))
my_model.add(Dense(units=1,))

my_model.compile(optimizer="Adam", loss='MSE', metrics=['mean_squared_logarithmic_error'])

my_train = my_model.fit(x=x_train, y=y_train, epochs=200, validation_split=0.2, )
```

در اینجا از ۲ لایه با تعداد نورون ۱۹ و اکتیویشن های relu استفاده شده وبرای آموزش آن از اپتیمایزر adam و loss MSE و متریس MSLE استفاده شده
شبکه را برای ۲۰۰ ایپاچ با ۰.۲ داده برای validation آموزش می‌دهیم و داریم:



نمودار بالا loss و val_loss ر برای آموزش شبکه نشان می دهد

```
 32/3200 [..............................] - ETA: 0s - loss: 435751.2500 - mean_squared_logarithmic_error: 2.0108e-06
608/3200 [====>.........................] - ETA: 0s - loss: 473955.9597 - mean_squared_logarithmic_error: 1.8265e-06
1152/3200 [=========>....................] - ETA: 0s - loss: 417969.1314 - mean_squared_logarithmic_error: 2.2497e-06
1696/3200 [==============>...............] - ETA: 0s - loss: 333155.4116 - mean_squared_logarithmic_error: 1.6743e-06
2464/3200 [====================>.......] - ETA: 0s - loss: 244341.3152 - mean_squared_logarithmic_error: 1.2151e-06
```

```
loss: 241601.7918 - mean_squared_logarithmic_error: 1.1147e-06 - val_loss: 705358.9258 - val_mean_squared_logarithmic_error: 2.1858e-06
```

و برای تست loss و تست acc به ترتیب داریم:

```
902398.1665625
2.6181025987170 87e-06
```

و شبکه را با

```
start = datetime.datetime.now()
two_layer_network_prediction()
end = datetime.datetime.now()
print(end - start)
```

اجرا کرده‌ایم

سوال ۳:

در این سؤال می‌خواهیم داده‌های Fashion-MNIST را برای آموزش به شبکه بدیم

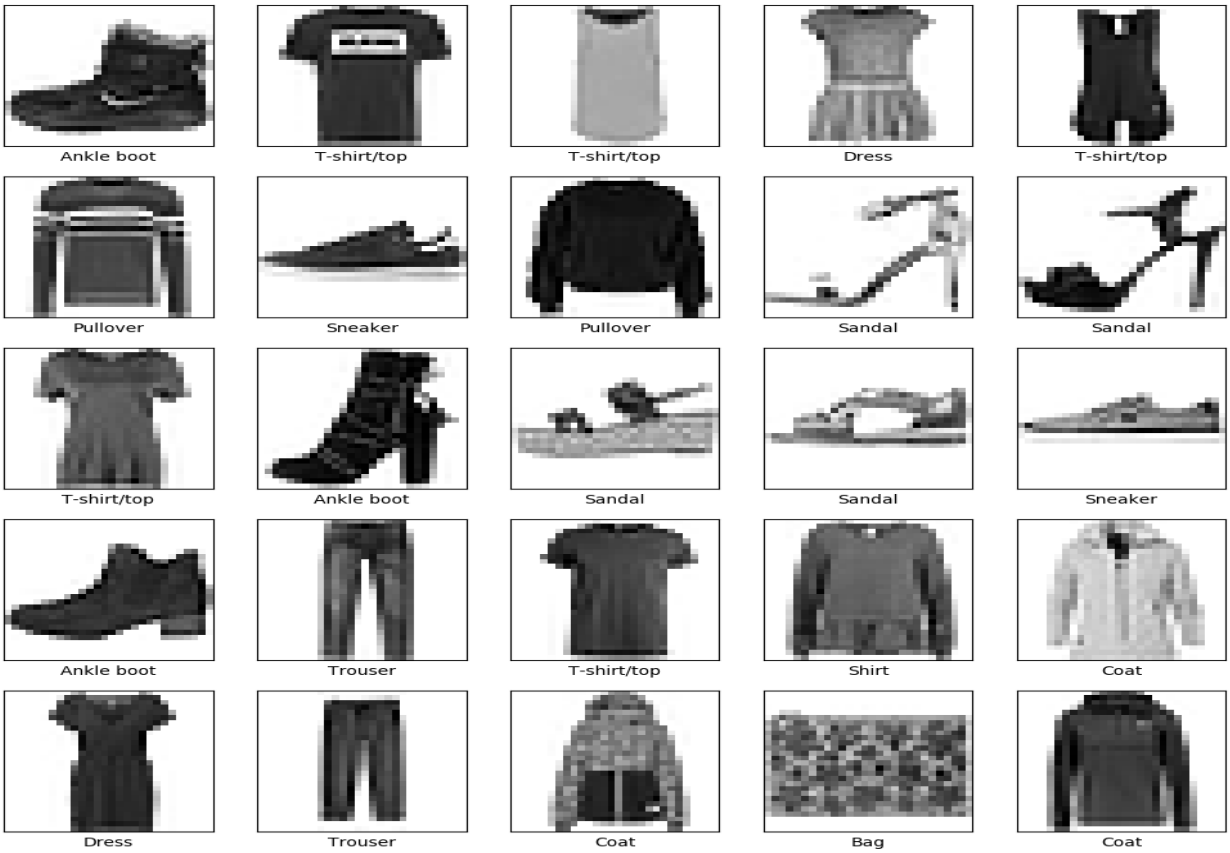
برای این کار ابتدا داده‌ها را از روی دیتابیس keras لود می کنیم:

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

برای دیدن داده‌ها و تست آن ابتدا می‌توانیم ۲۵ داده اول را مانند زیر ببینیم:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

که بعد از اجرا داریم:



الف)
برای داده‌های تست و آموزش از خود داده‌های دیتا بیس keras استفاده می‌کنیم و برای ارزیابی آن‌ها می‌توانیم از یک درصدی از داده‌های آموزش مثلاً ۰.۲ آن استفاده کنیم تا بتوانیم شبکه را برای داده‌های غیر آموزش نیز آماده کنیم

داده‌های اول در دیتابیس داده‌هایی برای یک سری عسک هستند که در ۱۰ دسته تقسیم می‌شوند
برای دادن داده‌ها به شبکه ابتده نیاز است که آن‌ها را نرمال کنیم زیرا در غیر این صورت مقادیر وزن ها به علت بزرگ بودن داده‌ها زیاد می‌شود و باعث می‌شود یک تغییر در داده‌ها خطای زیادی ایجاد کند

برای این کار ابتدا همه ی داده‌ها را بر ۲۵۵ تقسیم می کنیم:

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

در این صورت همه ی داده‌ها بیت ۰ تا ۱ می‌شوند
سپس چون داده‌ها عکس هایی با اندازه ۲۸ در ۲۸ هستند باید آن‌ها را یک راستا کنیم یعنی در یک ارایه ی ۲۸ * ۲۸ بریزیم تا بتوانیم آن‌ها را برای آموزش به شبکه بدهیم:

```
x_train = np.array(x_train).reshape(60000, 28*28)
x_test = np.array(x_test).reshape(60000, 28*28)
```

ب)
در این قسمت می‌خواهیم هر دفعه تعداد نورون های آن را تغییر دهیم و هر سری برای شبکه خود دو نمودار دقت و لاس رو می‌کشیم و هر سری ماتریس آشفتگی را نیز نشان می دهیم:
این تغییرات فقط در تعداد نورون است و بچ سایز برابر ۳۲ و تعداد ایپاچ برابر ۳۰ ثابت است

۱) تعداد نورون را 70 10 می گذاریم:

```
Network With layer1 of:  70 Nourons and layer2 of:  10 and Batch_size:  32 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  133.453458070755
```

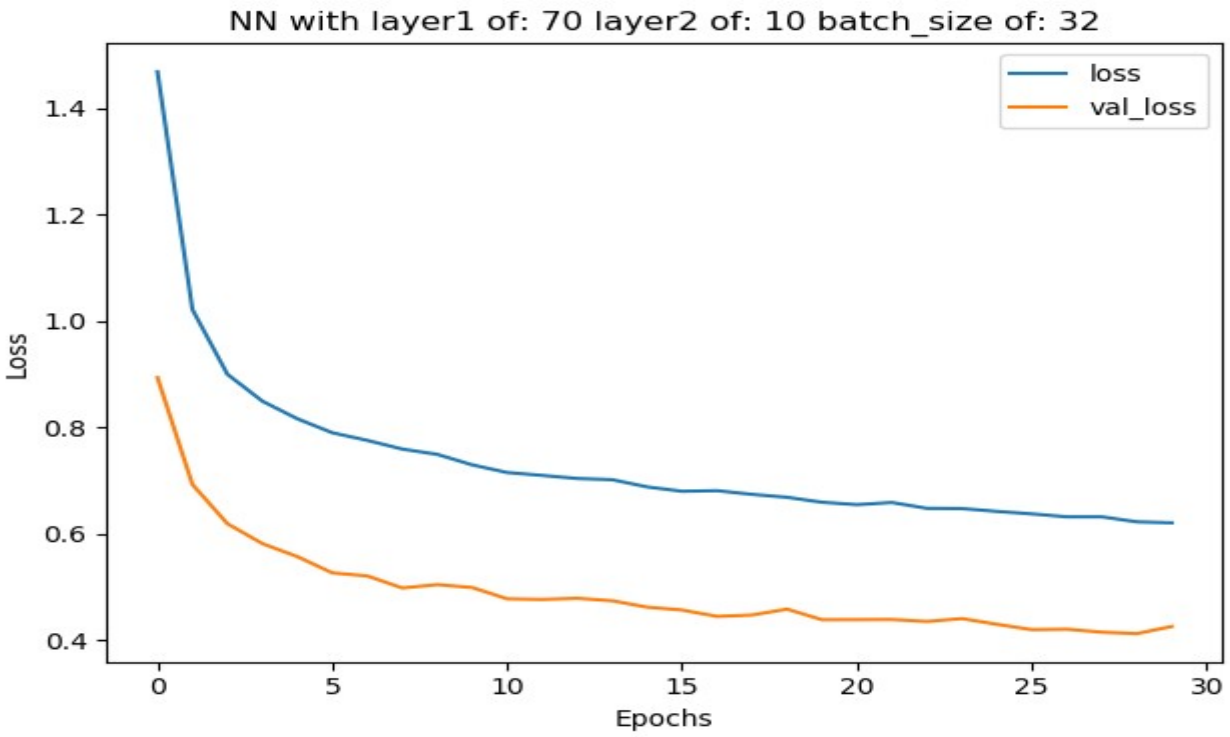
برای داده‌های ترین داریم:

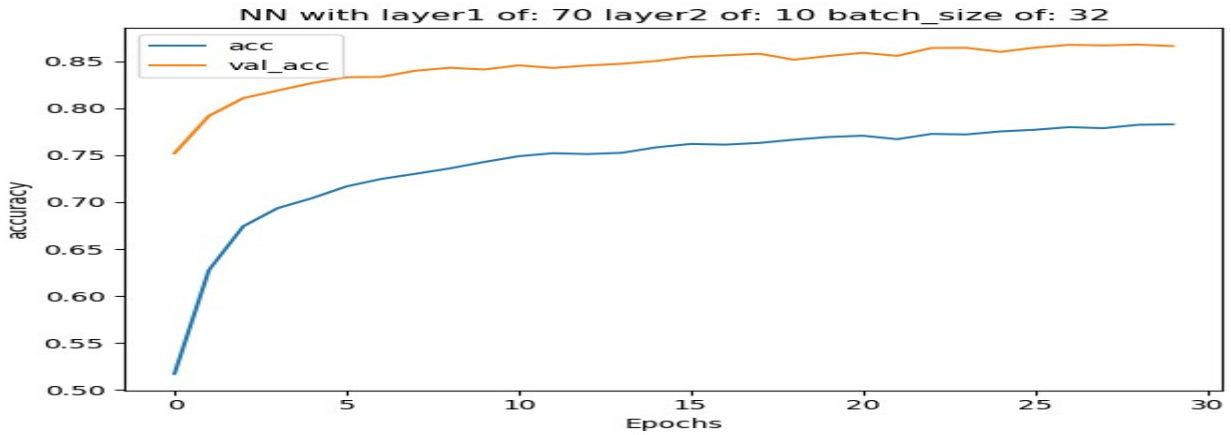
```
loss: 0.6209 - accuracy: 0.7828 - val_loss: 0.4260 - val_accuracy: 0.8661
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8587999939918518

Test loss: 0.4516950014591217
```

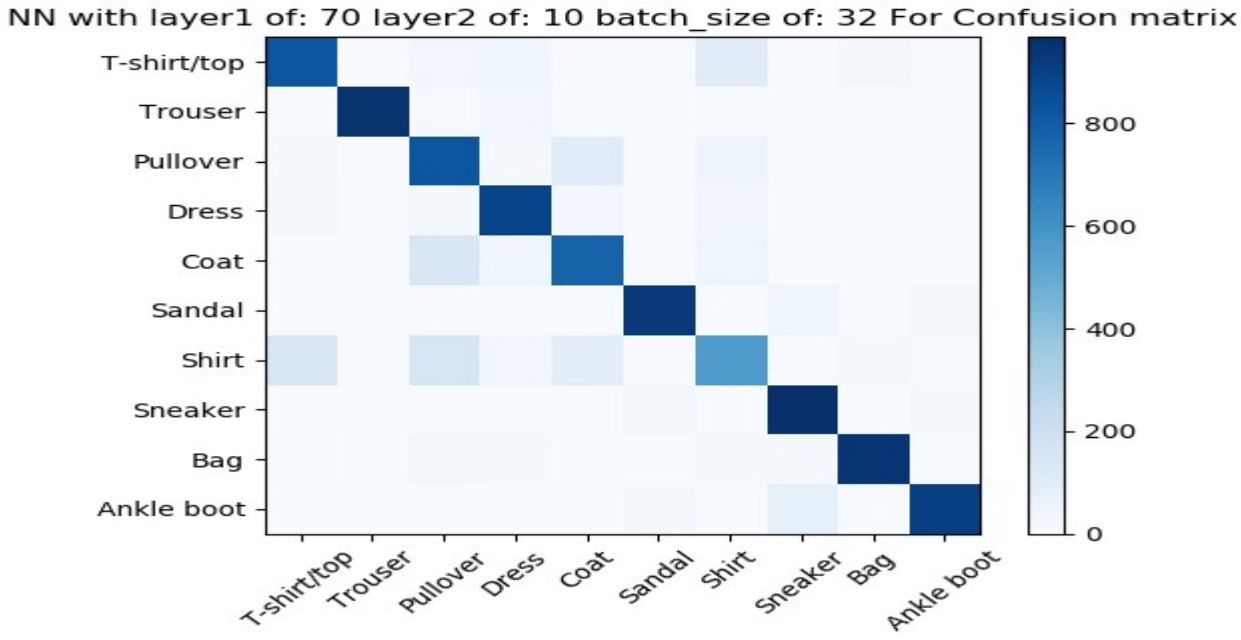
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[823   1  24  35   1   0 108   0   8   0]
 [  2 955   5  30   2   0   5   0   1   0]
 [ 10   1 827   9 103   0  47   1   2   0]
 [ 22   6  20 889  27   0  33   0   3   0]
 [  0   1 145  37 773   0  43   1   0   0]
 [  0   0   0   0   0 929   2  44   3  22]
 [144   1 158  28  89   0 570   0  10   0]
 [  0   0   0   0   0  22   0 968   0  10]
 [  5   1  17   9   0   1  14   8 945   0]
 [  0   0   0   0   0  10   0  81   0 909]]
```



۲) تعداد نورون را ۱۲۸ ۳۰ می گذاریم:

```
Network With layer1 of:  128 Nourons and layer2 of:  30 and Batch_size:  32 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  98.6938464641571
```

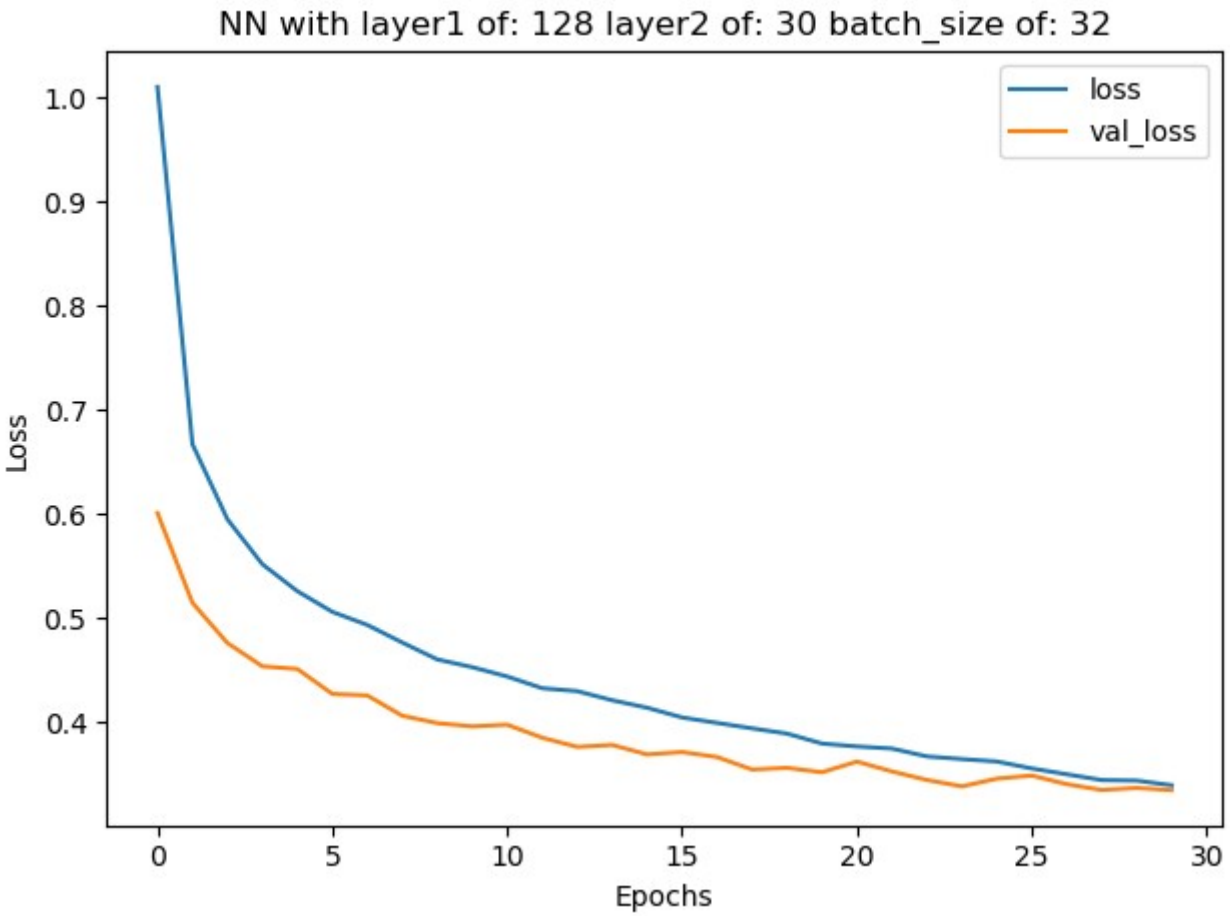
برای داده‌های ترین داریم:

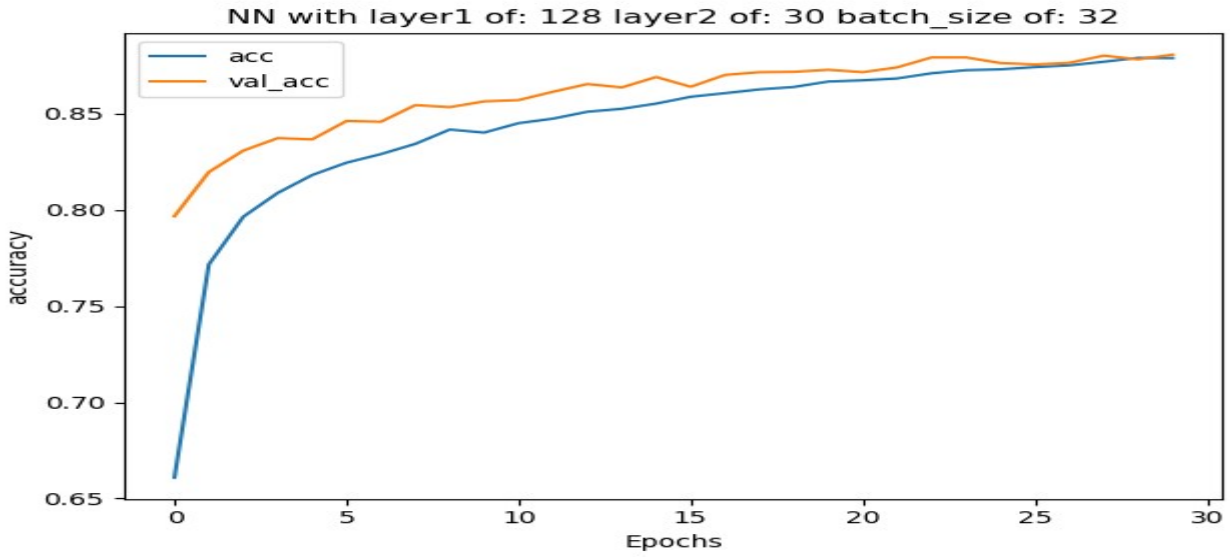
```
accuracy: 0.8973 - val_loss: 0.3110 - val_accuracy: 0.8878
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8794999718666077

Test loss: 0.33867004556655883
```

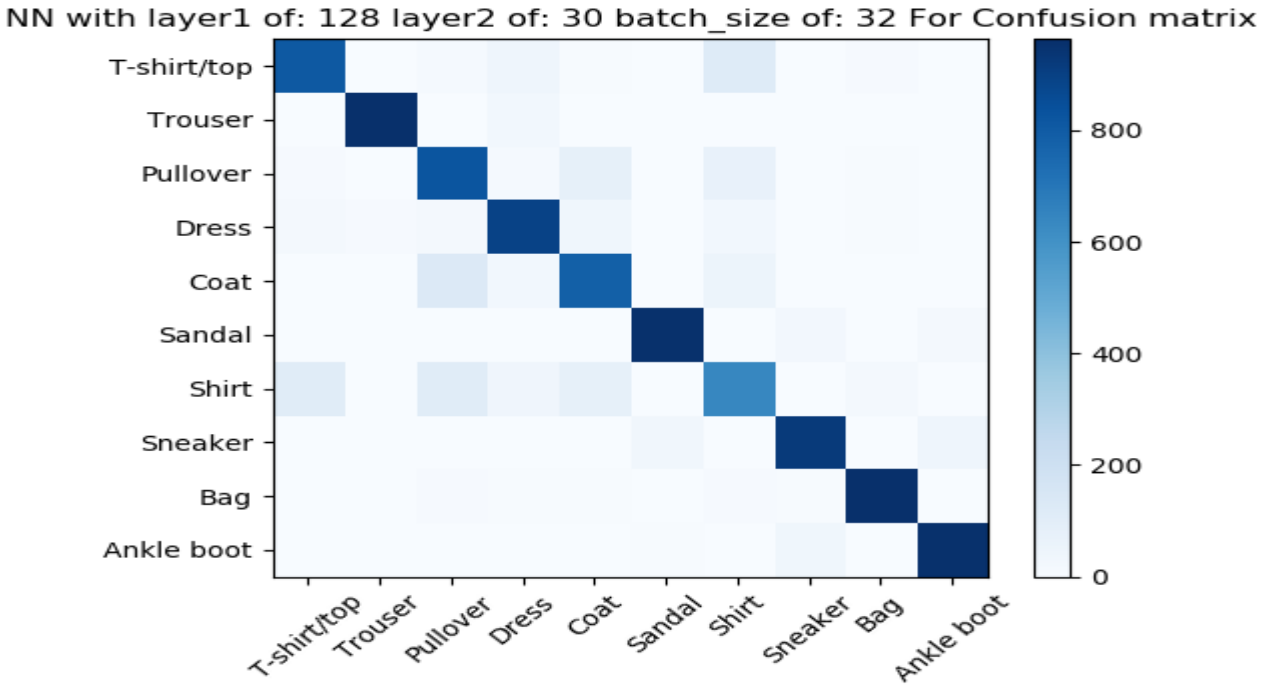
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[811   0  13  45   4   0 118   0   8   1]
 [  2 963   2  27   2   0   3   0   1   0]
 [ 10   0 825  14  79   0  68   0   4   0]
 [ 16   8  16 893  36   0  27   0   4   0]
 [  0   1 131  28 786   0  52   0   2   0]
 [  1   0   0   1   0 956   0  24   2  16]
 [112   1 110  41  80   0 640   0  16   0]
 [  0   0   0   0   0  33   0 922   0  45]
 [  3   1   8   5   6   2  10   4 961   0]
 [  0   0   0   0   0   6   1  34   0 959]]
```



۳) تعداد نورون را ۷۸۴ ۱۲۸ می گذاریم:

```
Network With layer1 of:  784 Nourons and layer2 of:  128 and Batch_size:  32 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  120.35519480705261
```

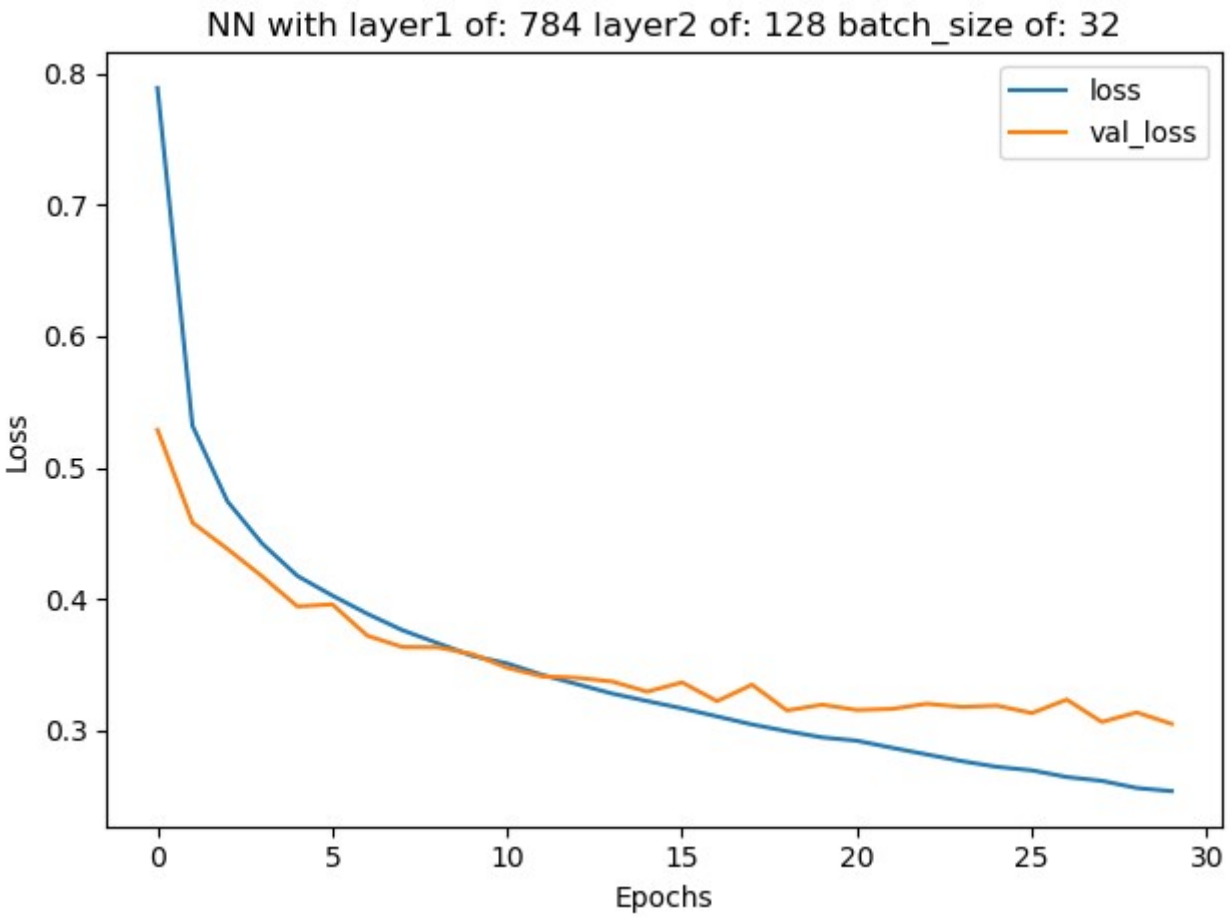
برای داده‌های ترین داریم:

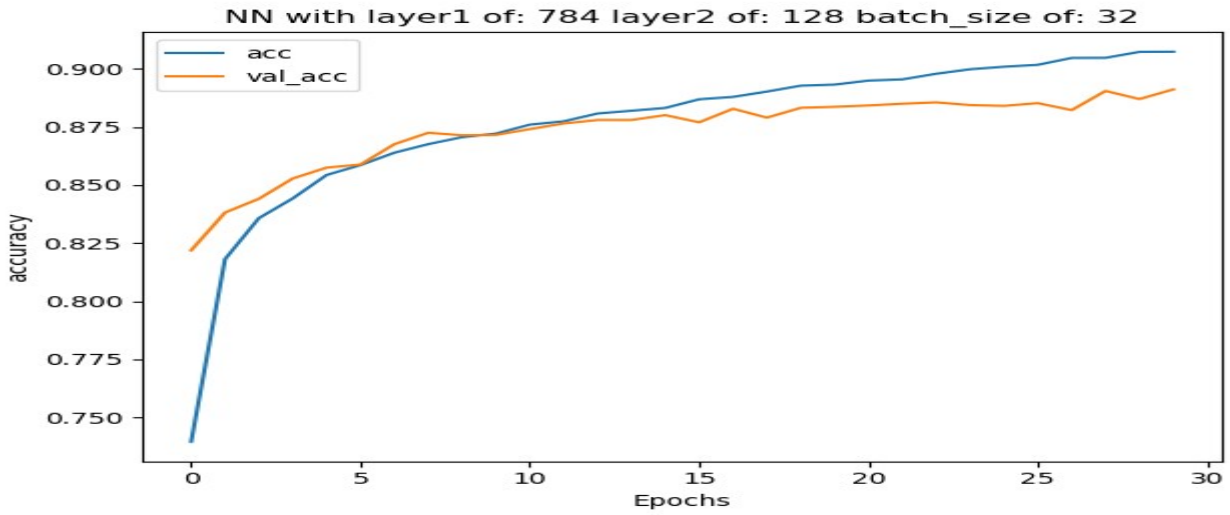
```
loss: 0.2540 - accuracy: 0.9074 - val_loss: 0.3052 - val_accuracy: 0.8912
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8841999769210815

Test loss: 0.33057641698122026
```

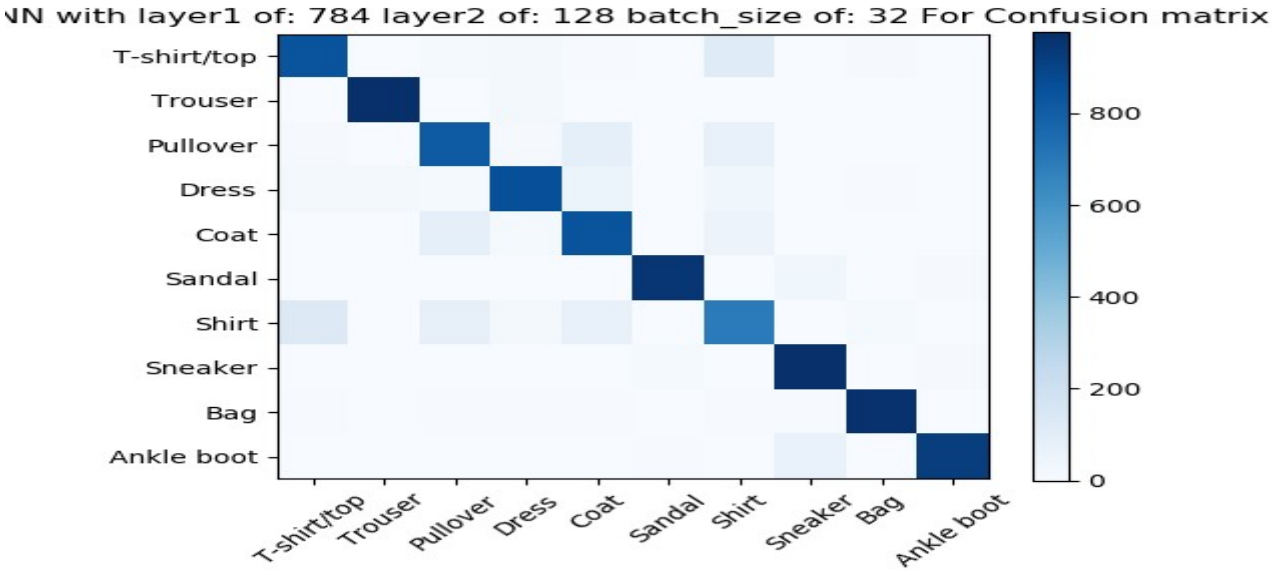
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[840   1  12  18   4   1 116   0   8   0]
 [  1 976   1  16   2   0   3   0   1   0]
 [ 11   0 815  10  86   0  76   0   2   0]
 [ 19  16  12 861  51   0  37   0   4   0]
 [  0   1  88  15 839   0  56   0   1   0]
 [  0   0   0   1   0 951   0  35   2  11]
 [123   1  80  22  69   0 693   0  12   0]
 [  0   0   0   0   0  14   0 974   1  11]
 [  4   1   4   5   5   1   7   6 967   0]
 [  0   0   0   0   0   6   1  67   0 926]]
```



ج) در این قسمت می‌خواهیم با تغییر بچ سایز تفاوت شبکه‌ها را بسنجیم
برای این کار از شبکه با ۱۲۸ نورون استفاده می‌کنیم و هر سری با تغییر بچ سایز شبکه را آموزش می‌دهیم
۱) بچ سایز را ۳۲ می گذاریم:

```
Network With layer1 of:  128 Nourons and layer2 of:  50 and Batch_size:  32 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  107.48452329635620
```

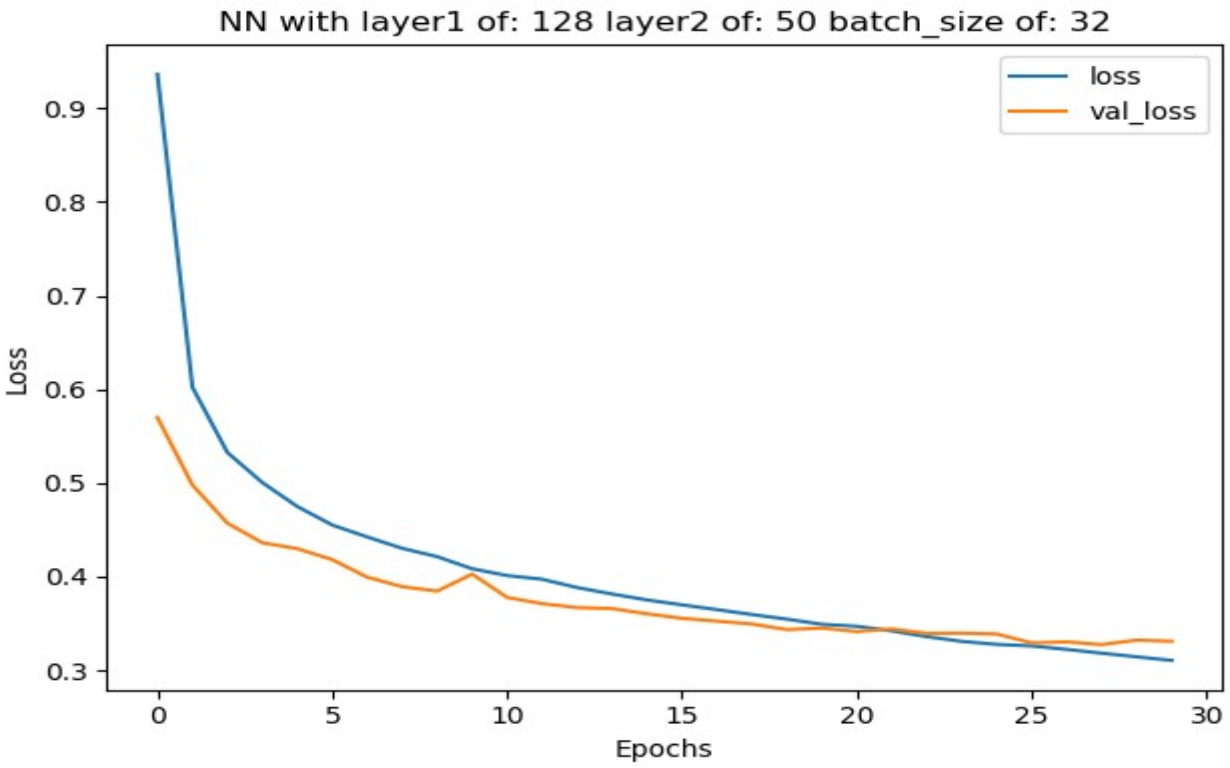
برای داده‌های ترین داریم:

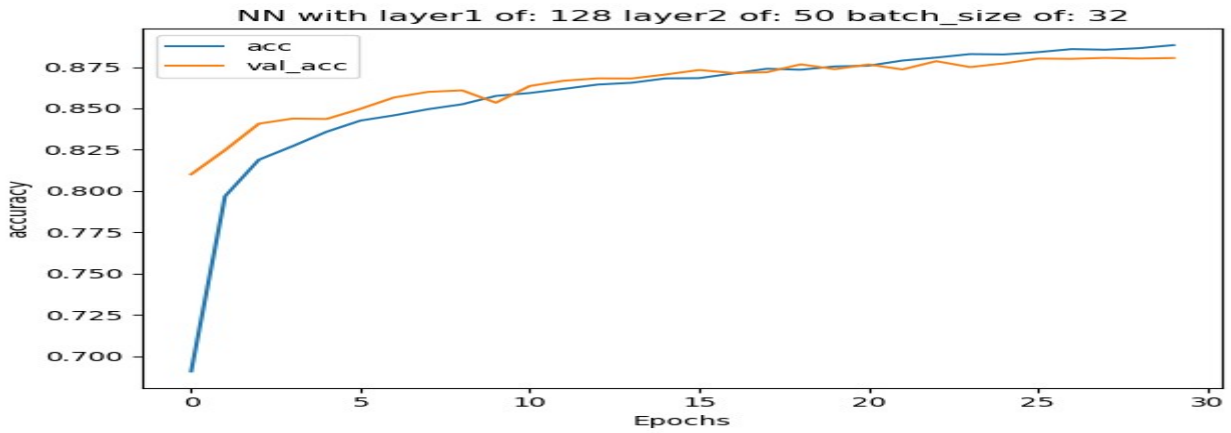
```
loss: 0.3105 - accuracy: 0.8883 - val_loss: 0.3310 - val_accuracy: 0.8805
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8737999796867371

Test loss: 0.3509850471019745
```

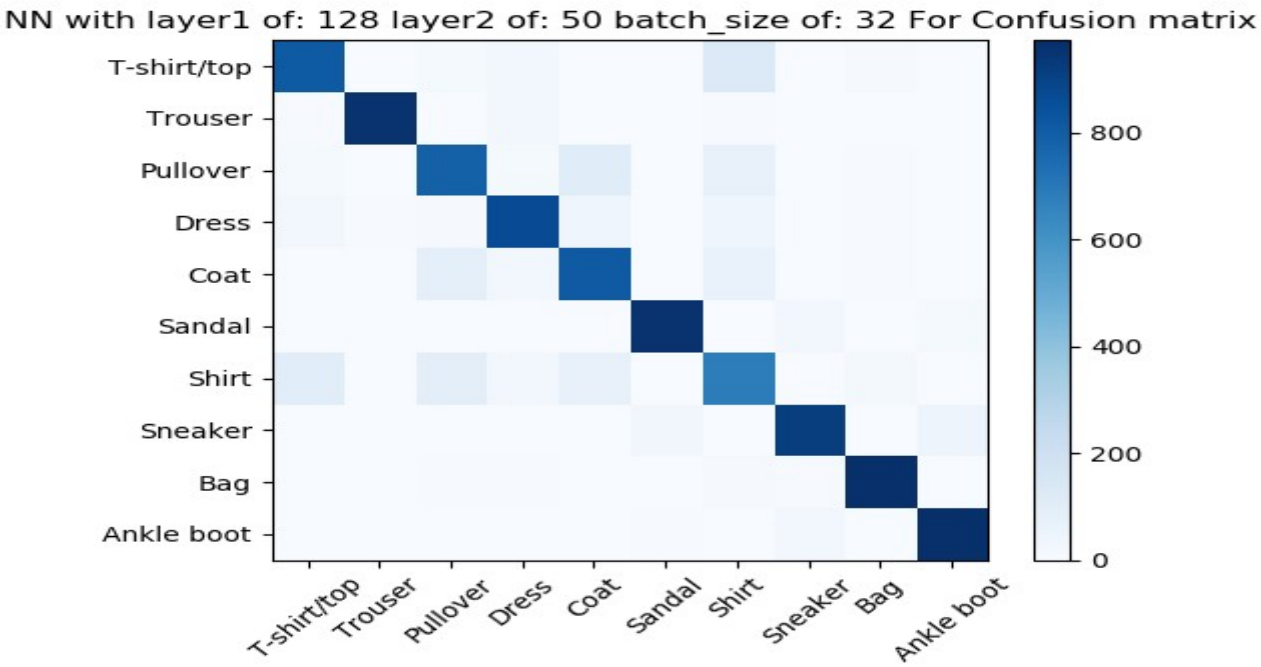
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[809   0  15  30   2   1 132   1   9   1]
 [  4 958   2  28   3   0   4   0   1   0]
 [ 12   0 786  12 112   1  73   0   4   0]
 [ 25   7  11 868  38   1  45   0   4   1]
 [  0   0  85  28 815   0  68   0   4   0]
 [  0   0   0   1   0 959   0  23   2  15]
 [103   1  96  25  75   0 683   0  17   0]
 [  0   0   0   0   0  31   0 922   0  47]
 [  1   1   4   4   2   3   9   4 972   0]
 [  0   0   0   0   0   7   1  26   0 966]]
```



۲) بچ سایز را ۶۴ می گذاریم:

```
Network With layer1 of:  128 Nourons and layer2 of:  50 and Batch_size:  64 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  52.21734857559204
```

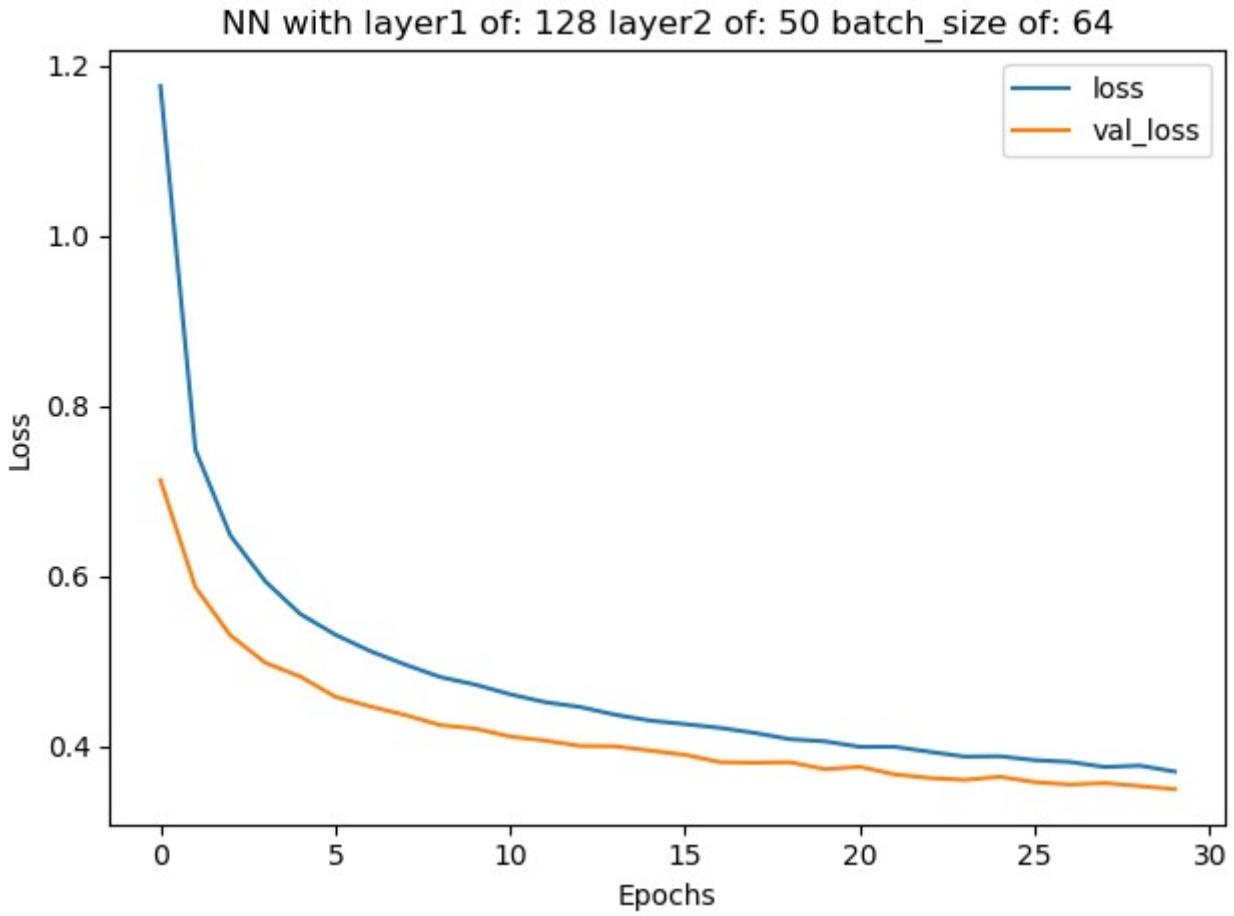
برای داده‌های ترین داریم:

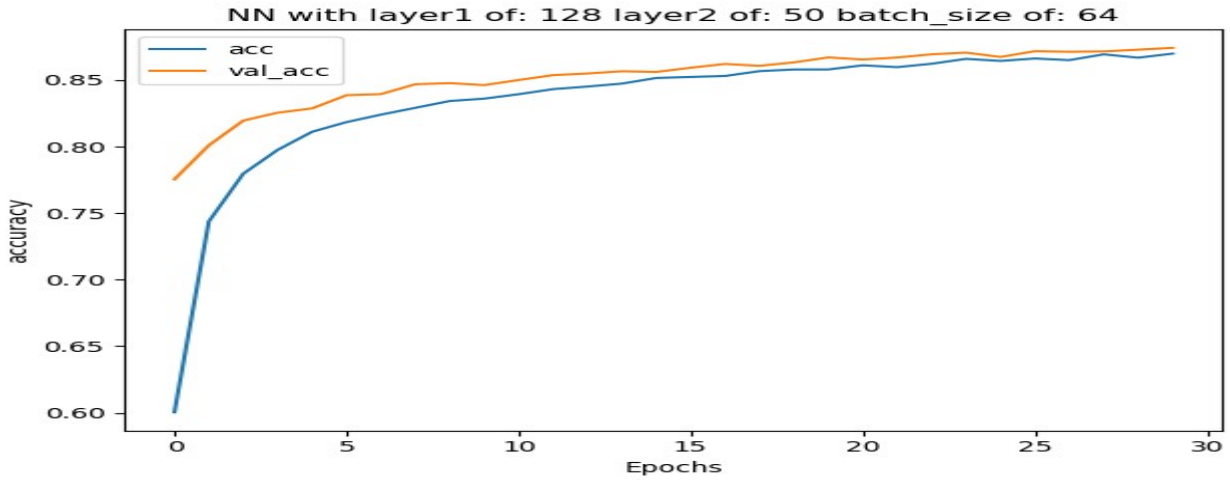
```
loss: 0.3706 - accuracy: 0.8699 - val_loss: 0.3500 - val_accuracy: 0.8742
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8661000132560730

Test loss: 0.37700191049575804
```

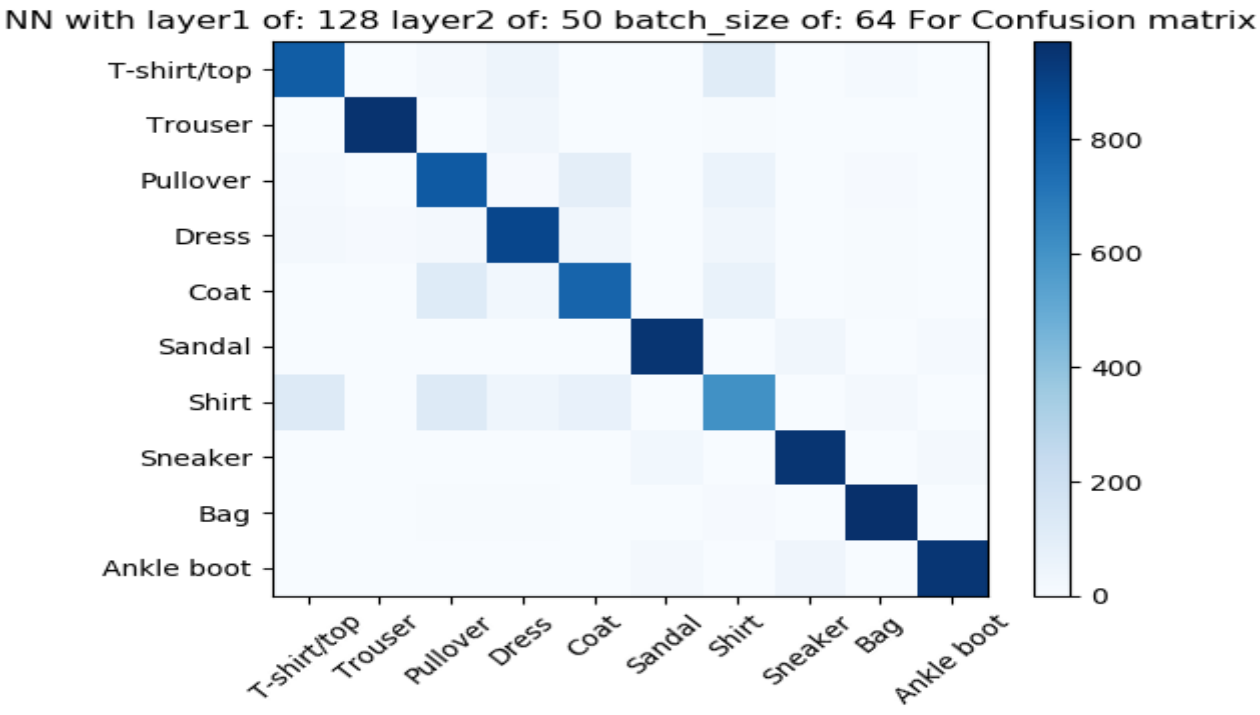
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[802   3  19  47   3   1 111   0  14   0]
 [  1 956   2  32   3   0   4   0   2   0]
 [ 14   1 815   9  94   0  58   0   9   0]
 [ 17  10  16 885  34   0  32   0   6   0]
 [  0   0 120  30 775   0  68   0   7   0]
 [  0   0   0   0   0 951   0  34   3  12]
 [126   1 125  44  75   0 610   0  19   0]
 [  0   0   0   0   0  27   0 951   0  22]
 [  2   1   5   4   3   2   9   3 971   0]
 [  0   0   0   0   0  16   0  38   1 945]]
```



۳) بچ سایز را ۲۵۶ می گذاریم:

```
Network With layer1 of:  128 Nourons and layer2 of:  50 and Batch_size:  256 And Epoch:  30
```

زمان train شبکه:

```
Trained finished in:  20.12170171737671
```

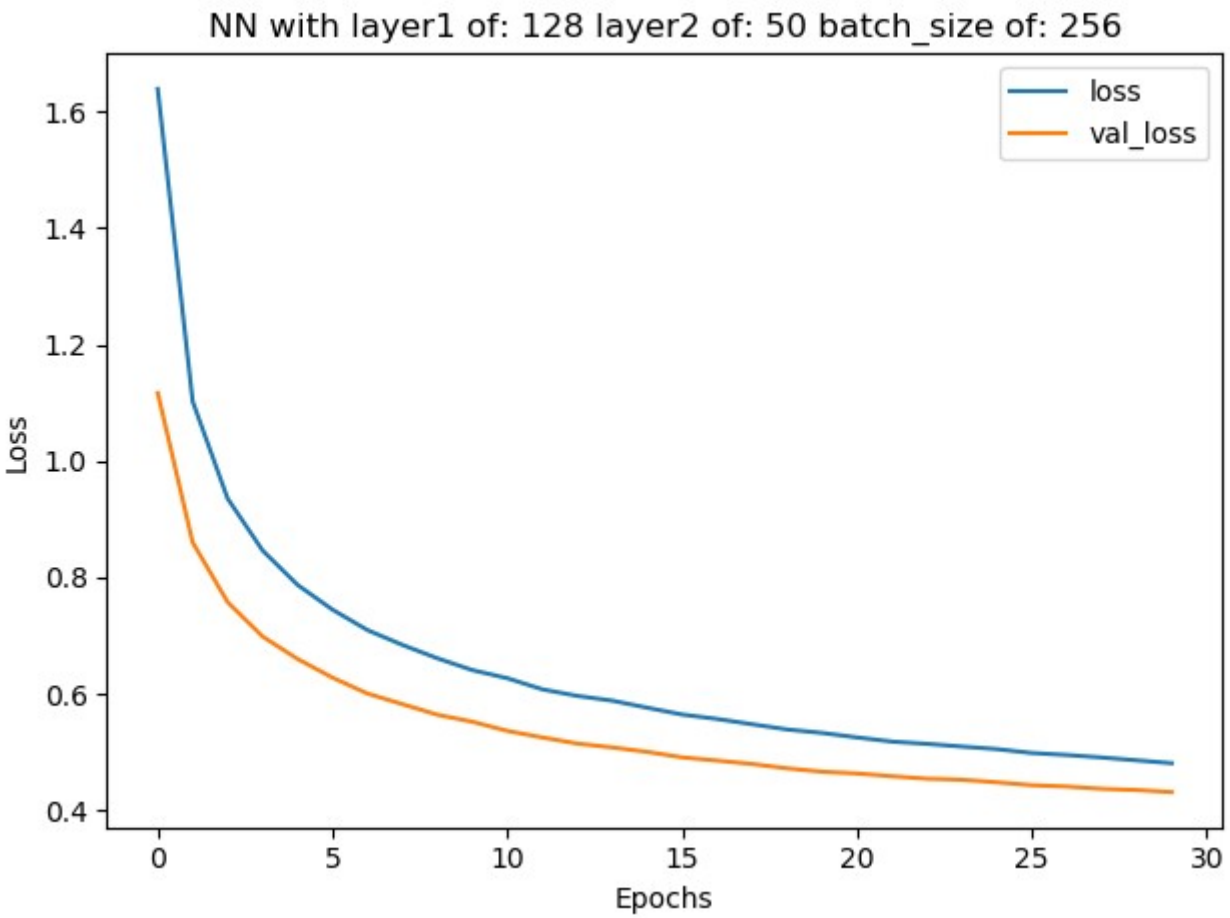
برای داده‌های ترین داریم:

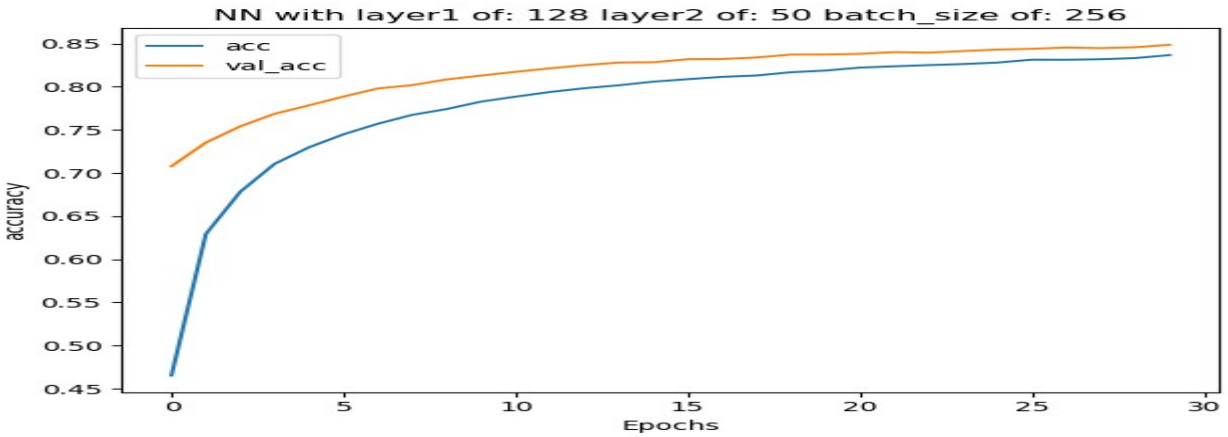
```
loss: 0.4808 - accuracy: 0.8366 - val_loss: 0.4311 - val_accuracy: 0.8486
```

و برای داده‌های تست داریم:

```
Test accuracy: 0.8374999761581421

Test loss: 0.4551974215507507
```

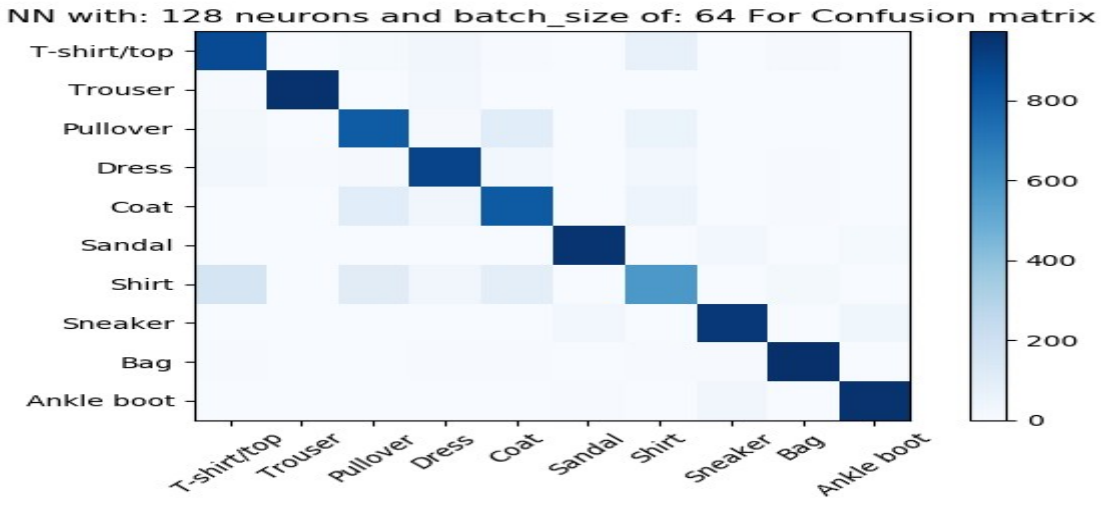
و نمودار ها به برتیب برای لاس و دقت:





و برای ماتریس آشفتگی داریم:

```
Matrix=
[[808   1  14  48   4   1 107   0  16   1]
 [  2 951   6  31   6   0   2   0   2   0]
 [ 20   2 727  10 147   1  81   0  12   0]
 [ 25  11  13 855  36   1  54   0   4   1]
 [  0   2  99  30 771   0  92   0   6   0]
 [  0   0   0   1   0 908   0  51   6  34]
 [157   1 128  31 111   0 542   0  30   0]
 [  0   0   0   0   0  32   0 917   0  51]
 [  0   1  10   7   3   2  20   6 951   0]
 [  0   0   0   0   0  13   0  41   1 945]]
```



تمامی شبکه‌های فوق با معماری به شکل زیر ساخته شده اند:

```
def NN_with(x_train, y_train, nouron_number_layer1, nouron_number_layer2, batch_size, epoch):
    my_model = Sequential()
    my_model.add(Dense(nouron_number_layer1, activation='relu', input_shape=(784,)))
    my_model.add(Dense(nouron_number_layer2, activation='relu'))
    my_model.add(Dropout(0.3))
    my_model.add(Dense(10, activation=softmax))
    my_model.compile(optimizer='sgd', loss=sparse_categorical_crossentropy, metrics=['accuracy'])
    trained_model = my_model.fit(x_train, y_train, batch_size=batch_size, epochs=epoch, validation_split=0.2)
    return trained_model, my_model
```

و برای کشیدن نمودار های دقت و لاس به شکل زیر عمل شده است:

```
plt.title('NN with: ' + str(nouron_number) + ' neurons and batch_size of: ' + str(batch_size))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.legend(['loss', 'val_loss'])

plt.figure()
plt.title('NN with: ' + str(nouron_number) + ' neurons and batch_size of: ' + str(batch_size))
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.legend(['acc', 'val_acc'])
```

و برای ماتریس آشفتگی داریم:

```
plt.title('NN with: ' + str(nouron_number) + ' neurons and batch_size of: ' + str(batch_size))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.legend(['loss', 'val_loss'])

plt.figure()
plt.title('NN with: ' + str(nouron_number) + ' neurons and batch_size of: ' + str(batch_size))
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.legend(['acc', 'val_acc'])
```

از نمودار ها و ماتریس ها نتایج زیر را بدست می اوریم:

در تغییر تعداد نورون ها دیدیم که بهترین حالت برای تعداد ۱۲۸ بود زیرا در تعداد ۷۸۴ شبکه overfit و در تعداد ۵۰ شبکه نتوانسته بود به دقت کافی برسد
پس بهترین حالت تعداد ۱۲۸ بود

در تغییر بچ سایز دیدم که هر چه بچ سایز را بیشتر می‌کنیم زمان آموزش داده‌های ما نیز کمتر می‌شود زیرا تعداد بیشتری داده به صورت همزمان به شبکه داده می‌شود و آموزش سریع‌تر انجام می‌شود
برای دقت آن می‌توان گفت که با زیاد کردن بچ سایز دقت کم شده است ولی زمان بهتر است
بهترین بچ سایز برای این شبکه ۶۴ است که با دقت تقریباً برابر زمان بهتری نسبت به ۳۲ داشته است

پس بهترین معماری در این سؤال تعداد ۱۲۸ نورون با بچ سایز ۶۴ بوده است

سؤال ۴:

در آن سؤال می‌خواهیم همان داده‌های سؤال قبل را ابتدا کاهش بعد دهیم سپس به همان شبکه ای که در سؤال قبل بدست آوردیم بدهیم تا دوباره شبکه آموزش ببیند و دقت و لاس را با هم مقایسه کنیم

ابتدا برای استفاده از autoencoder ابتدا با یک لایه تعداد بعد ورودی را مه ابتدا ۷۸۴ بود کاهش می‌دهیم سپس این داده‌های کاهش بعد داده شده را به شبکه می‌دهیم که آن دقیقاً مانند قبل می‌شود زیرا ما در اولین لایه ۱۲۸ نورون گذاشتیم که یعنی داده‌ها را به بعد ۱۲۸ کاهش دادیم

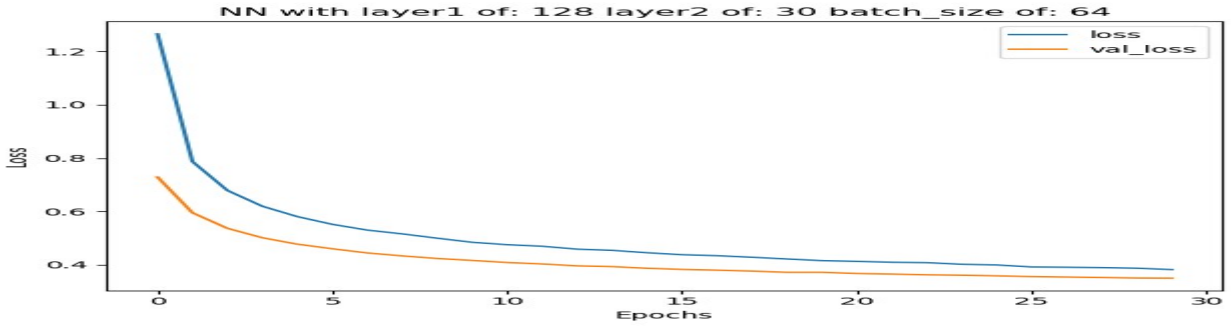
برای PCA از لایبرری استفاده می‌کنیم و بعد را با آن کاهش می‌دهیم برای اینکه پیدا کنیم بعد را تا چه حا می‌توان کم کرد از روش آزمون خطا رفته‌ام و انقدر اجرا کردم تا بهترین نتیجه کمترین لاس رو بگیرم برای این کار از کد زیر استفاده می کنیم:

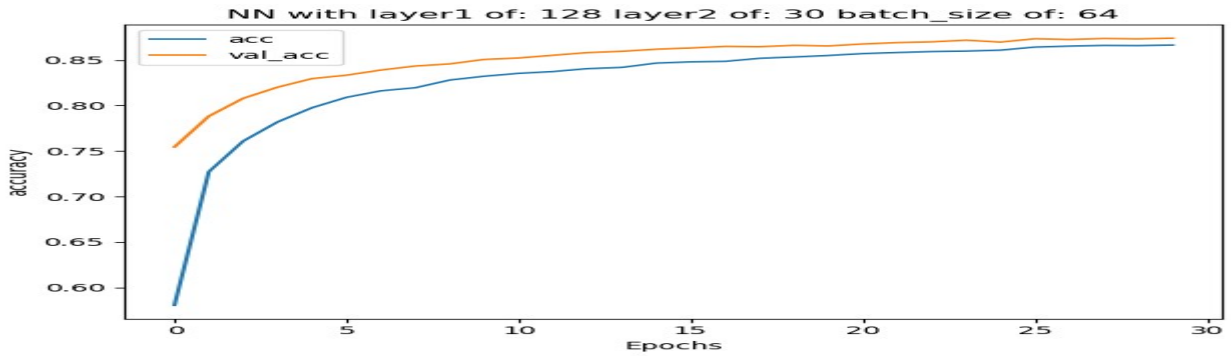
```
def NN_with_PCA(x_train, x_test):
    pca = PCA(n_components=128)
    pca_fit = pca.fit_transform(x_train)
    x_train = pca.transform(x_train)
    x_test = pca.transform(x_test)
    return x_train, x_test
```

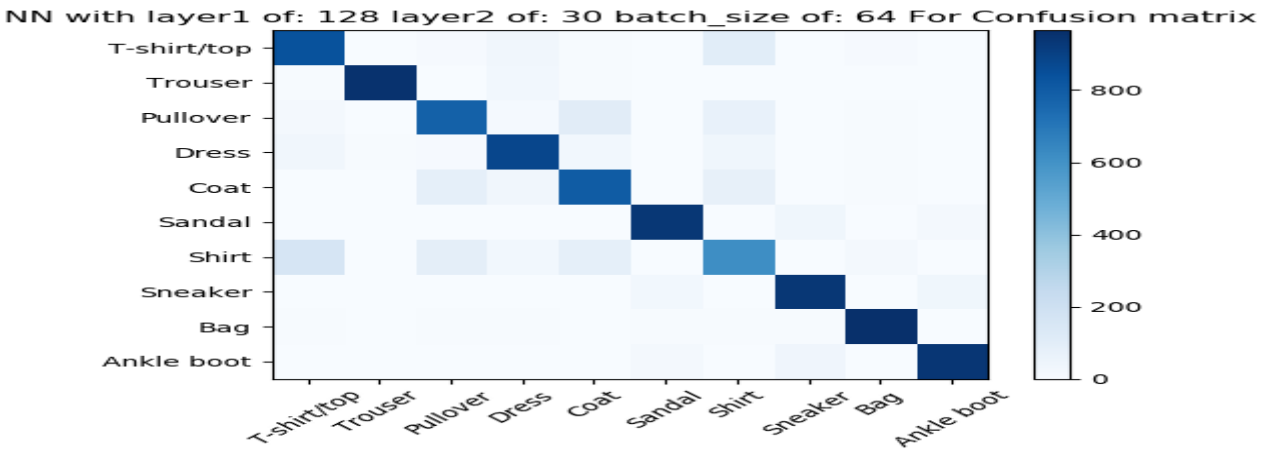
برای RBM نیز از لایبرری استفاده می‌کنیم و بعد را با آن کاهش می‌دهیم تا به یک بعد از داده‌ها برسیم تا کمترین خطا و بهترین دقت را داشته باشیم و آن را به شبکه می دهیم. این حالت را هم با امتحان پیدا می‌کنیم

```
def NN_with_rbm(x_train, x_test):
    my_model = BernoulliRBM(n_components=128)
    rbm_fit = my_model.fit(x_train)
    x_train = my_model.transform(x_train)
    x_test = my_model.transform(x_test)
    return x_train, x_test
```

برای هر ۳ حالت ابعاد را به یک اندازه کاهش می دهیم:
در اینجا هر ۳ را برای کاهش بعد ۷۸۴ به ۱۲۸ انجام داده‌ایم و
سپس داده‌ها را به شبکه سؤال ۳ می دهیم
برای PCA داریم:

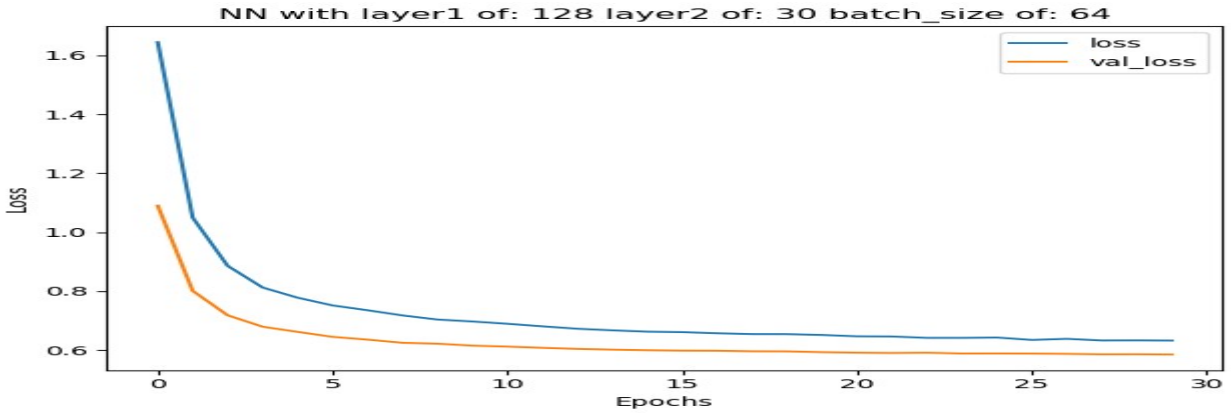


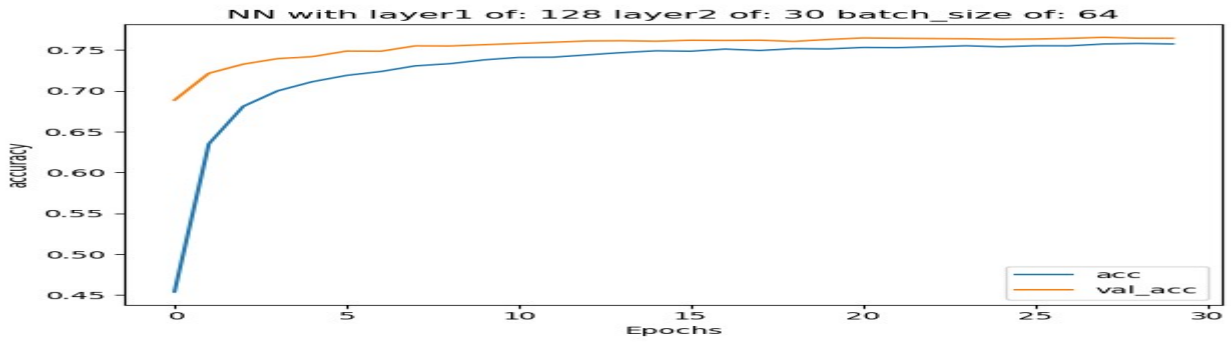


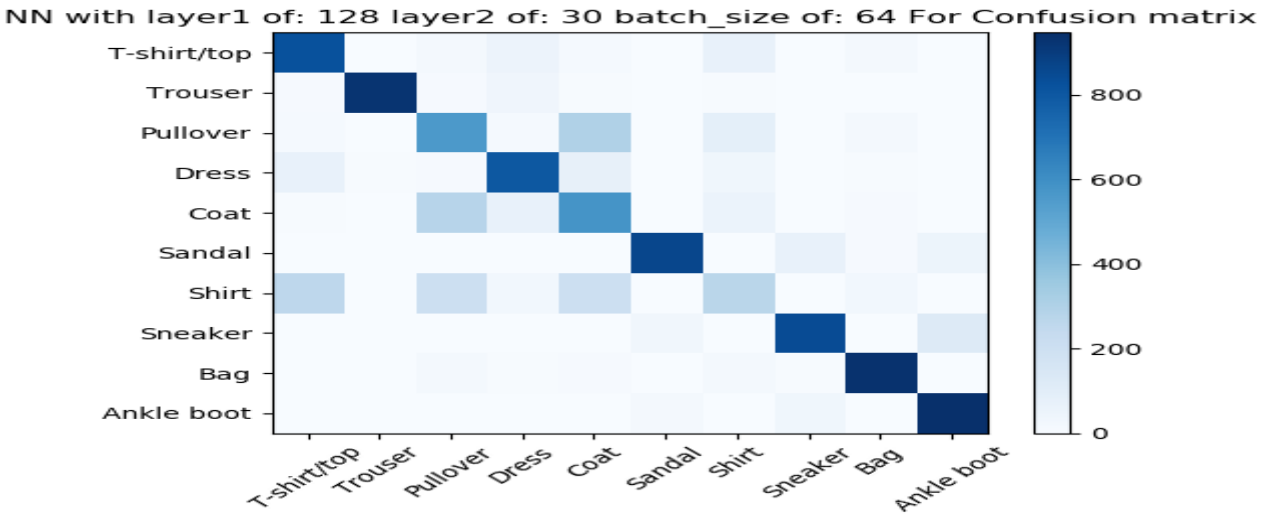



```
Test accuracy: 0.8651000261306763

Test loss: 0.3754546363353729
```

برای RBM داریم:








```
Test accuracy: 0.7566999793052673

Test loss: 0.6023172196865082
```

سپس می‌توانیم جدول را پر کنیم:

مورد	دقت داده تست	خطا داده تست
بهترین شبکه سؤال ۳	۰.۸۶	۰.۳۸
Autoencoder	۰.۸۳	۰.۴۵
PCA	۰.۸۵	۰.۳۸
CascadedRBM	۰.۷۵	۰.۶۰

همان‌طور که میبینیم کاهش بعد با PCA جواب بهتری نسبت به RBM و Autoencode می‌دهد

کد های مربوطه به پیوست آمده است

سؤال ۵:

الف) اگر داده مربوط به یک کلاس کمتر باشد شبکه نمی‌تواند آن داده را به خوبی آموزش ببیند و در هنگام پیش‌بینی نمی‌تواند ام کلاس را با دقت مناسب از بقیه جدا کند. یکی از راه‌های جلوگیری از این کار این است که همان داده‌ها را چند بار به شبکه بدهیم تا تعداد آن‌ها با بقیه برابر شود. راه دیگر استفاده از الگوریتم های ماشین لرنینگ و دیتا اگمنتیشن است تا در صدرت برابر نبود داده‌ها شبکه بتواند به درستی آموزش ببیند

ب) خیر
زیرا اگر دقت داده‌های تست بیشتر باشد ممکن است دقت داده‌های آموزش آن کمتر باشد و شبکه underfit شده باشد که در این صورت شبکه نتوانسته به خوبی آموزش ببیند و دقت ای که برای داده‌های تست داده شده دقیق نیست
در صورت برابر بودن دقت دادههای آموزش دقت بیشتر برای تست بهتر است

ج) باید برای هر ستون کوواریانس آن را حساب کنیم. کوواریانس نشان از وابستگی داده جواب به آن ستون است برای هر ستون هر چه این عدد بزرگ‌تر باشد آن ستون وابستگی بیشتر و در نتیجه ستون مهم‌تری نسبت به بقیه است. می‌توان ستون‌های با کمترین عدد و یا ستون‌های با عدد ۰ را حذف کرد و به یک شبکه ساده‌تری رسید

د) ماتریس آشفتگی برای نشان دادن عمل‌کرد یک شبکه است. در آن تعداد پیش‌بینی های درست و غلط را برای هر کلاس در رابطه با کلاس‌های دیگر نشان می‌دهد و نشان می‌دهد در کدام کلاس‌ها مدل ما بیشتر اشتباه داشته و نتوانسته به درستی پیش‌بینی کند.
این ماتریس هم ایراد های موجود در شبکه را به ما نشان می‌دهد و هم مدل و جایگاه آن ایراد را می‌تواند نشان دهد

۵)نرمال کردن به معنی این است که بازه ی داده‌ها را به ۰ تا ۱ برسانیم. این کار را معمولاً برای داده‌های بزرگ انجام می‌دهند زیرا در غیر این صورت به دلیل بزرگ بودن داده‌ها وزن ها نیز بزرگ می‌شوند و نمی‌توانند به دقت مناسب برسند ولی استاندارد کردن یعنی داده‌ها را طوری تغییر بدیم که میانگین آن‌ها ۰ و انحراف معیار ۱ داشته باشن که این کار باعث می‌شود داده‌ها زیاد پخش نباشند و شبکه بتواندبه درستی آموزش ببیند