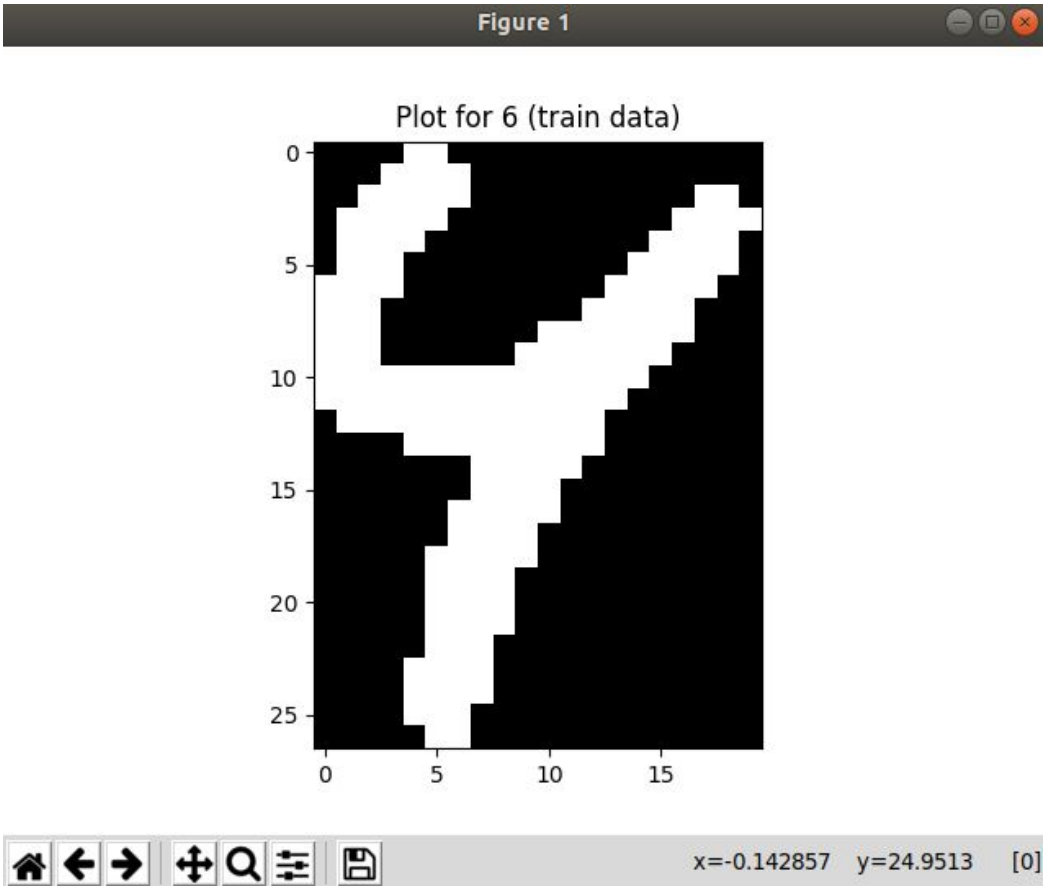
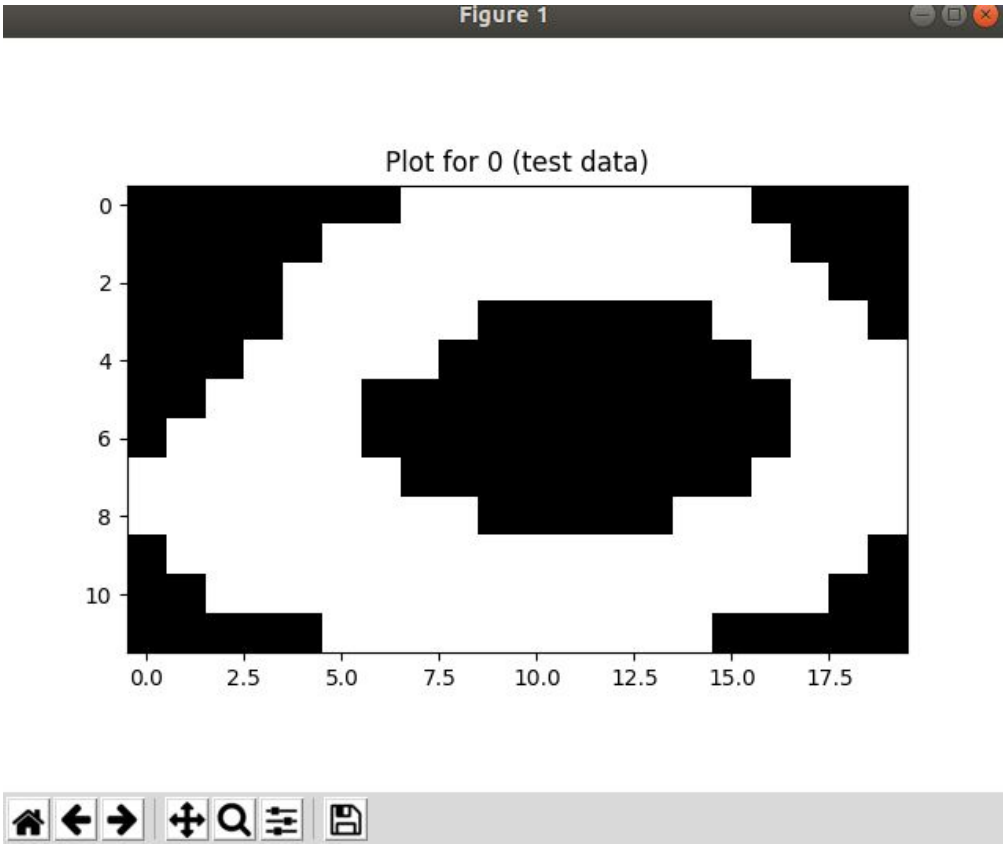
**زهرا بشیر 95521072**

**تمرین سری دوم**

الف) کد آن پیوست شده است.(A.py-1)






ب-1) روش محاسبه precision, recall , f1-score:

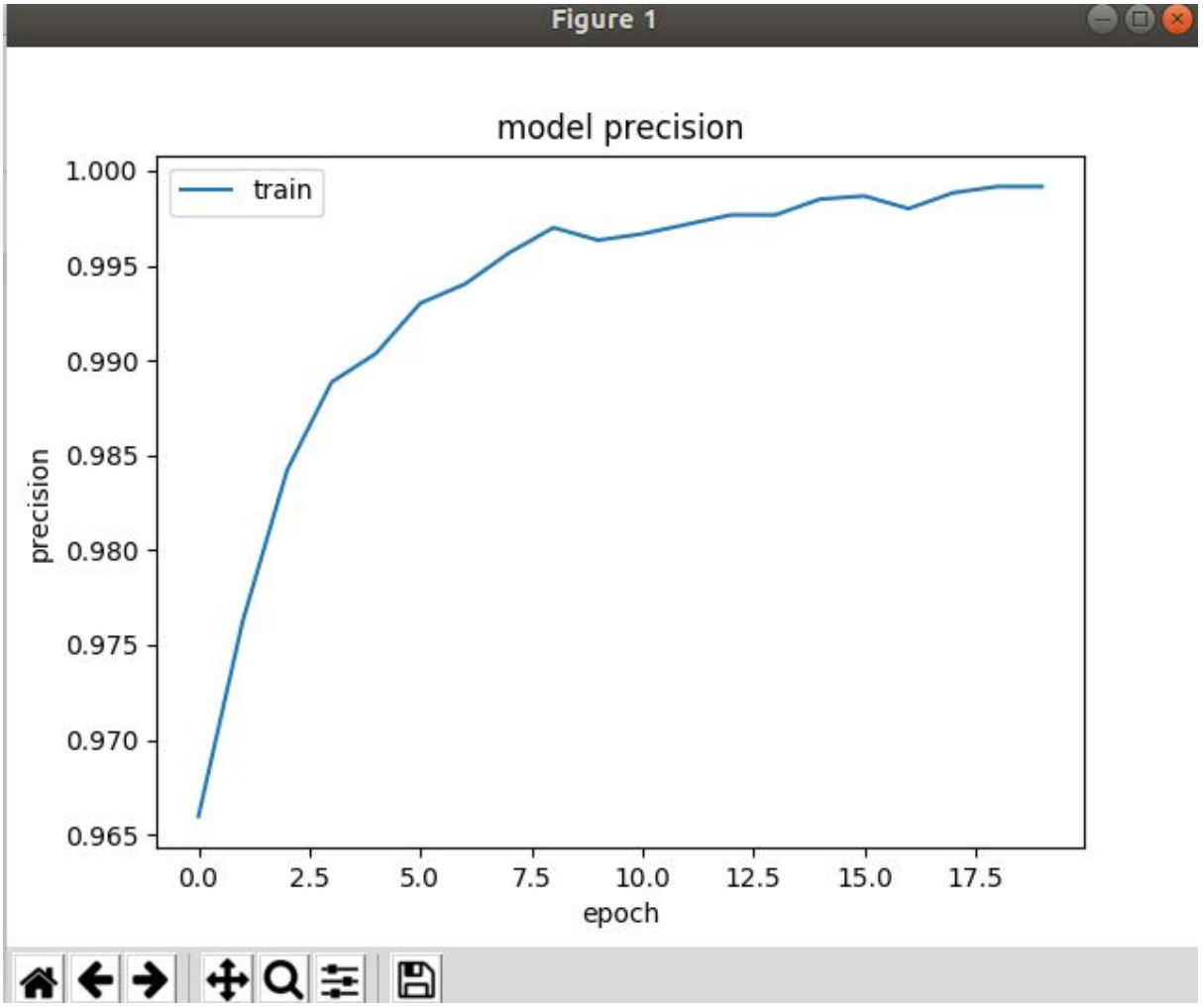
precision = TP/TP+FP

recall = TP/TP+FN

f1-score = 2*(precision*recall)/(precision+recall)

همچنین با import کردن keras_metrics و افرون این سه متریک به مدل میتوان آنرا اندازه گرفت

نتایج حاصل از محاسبه این موارد در epoch 20 :(بدون dropout)



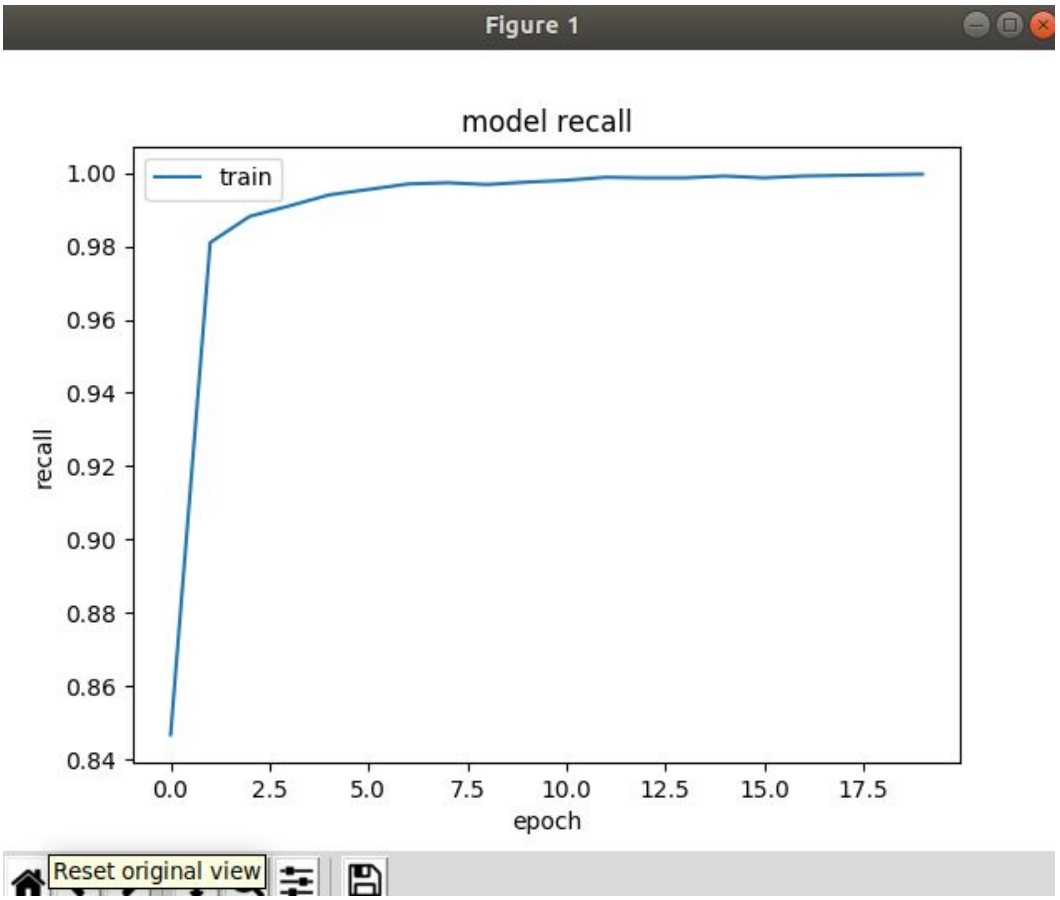


Figure 1
model recall
train
recall
epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Reset original view

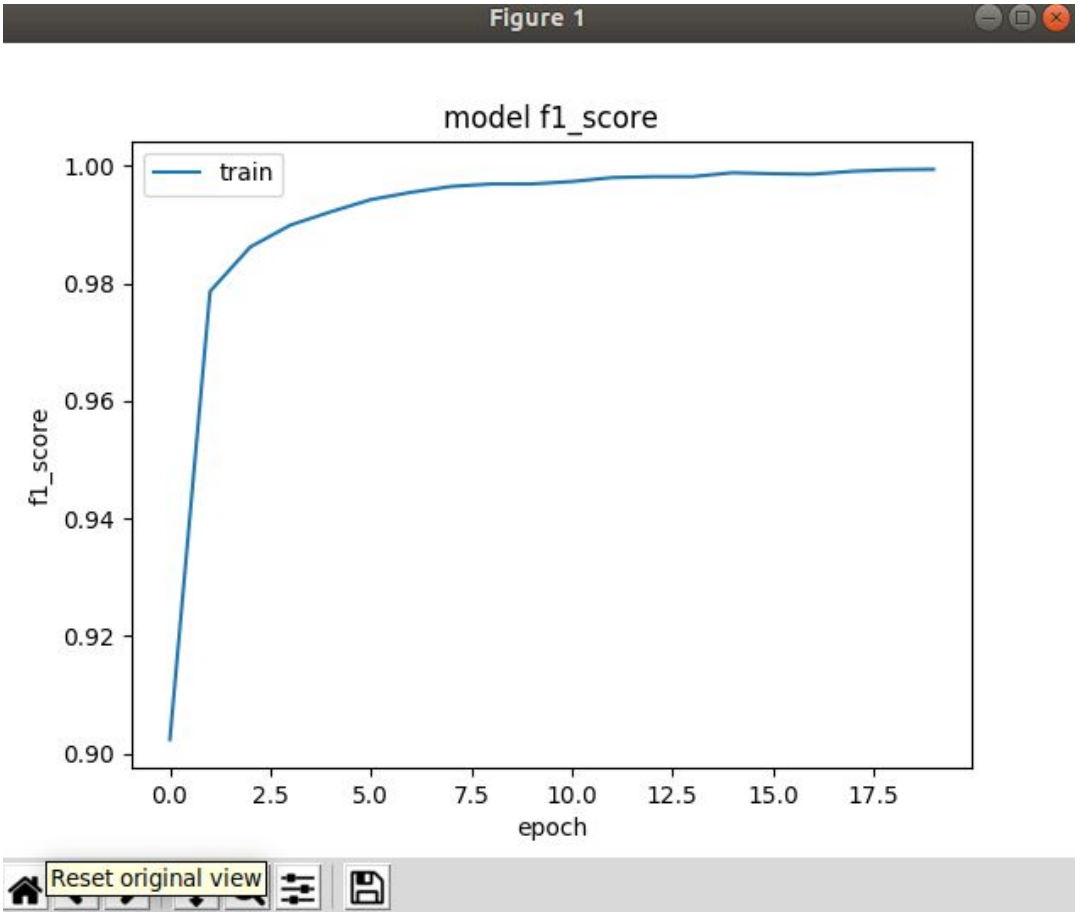
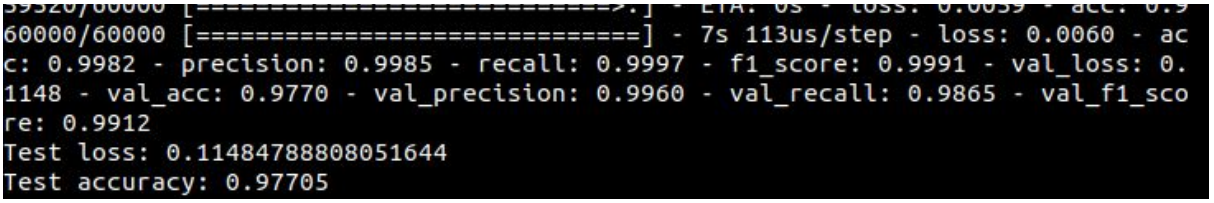


Figure 1
model f1_score
train
f1_score
epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Reset original view

نتایج نهایی در صفحه بعد




نتایج حاصل در کل (epoch آخر):

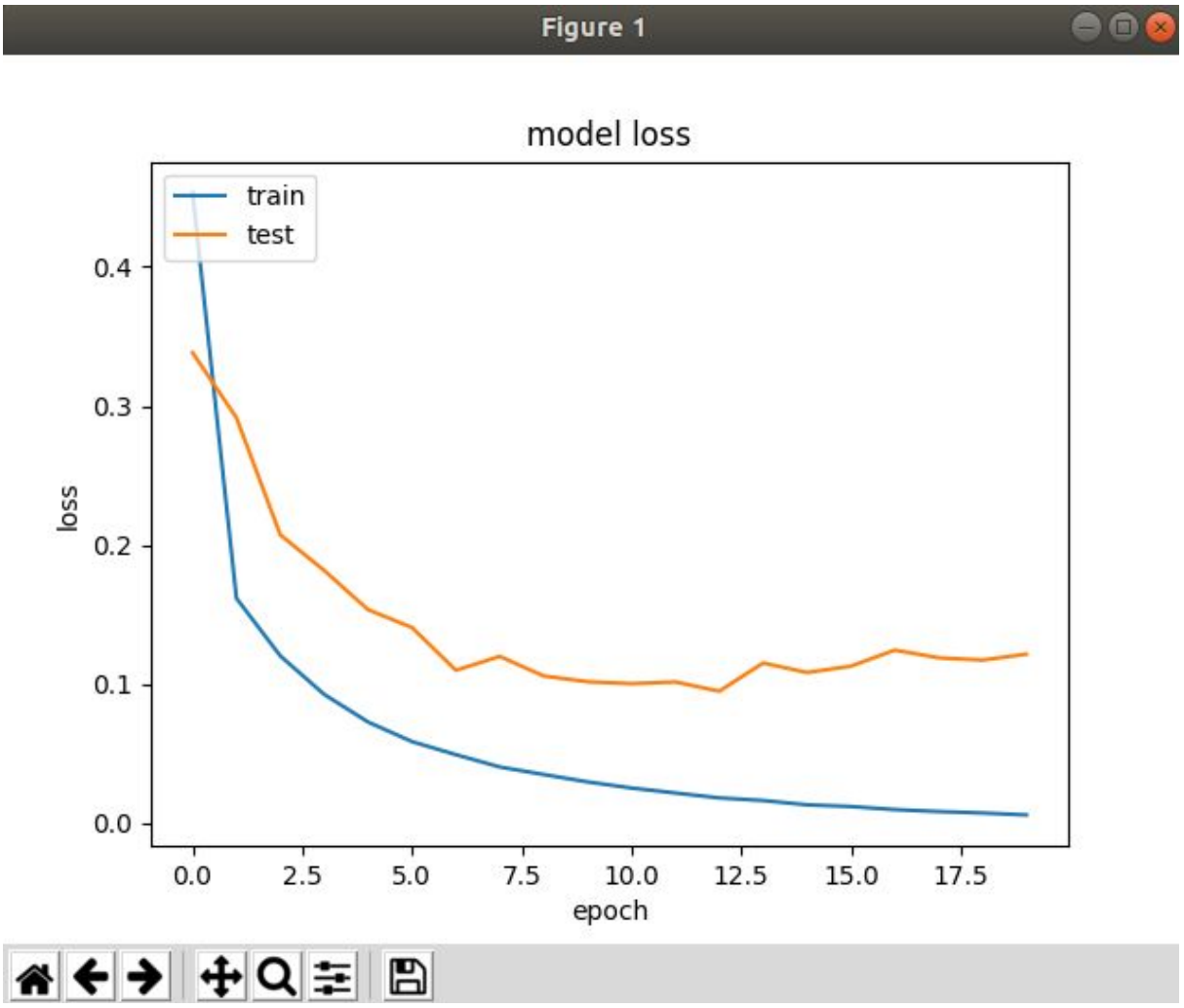
precision = .993

recall = .989

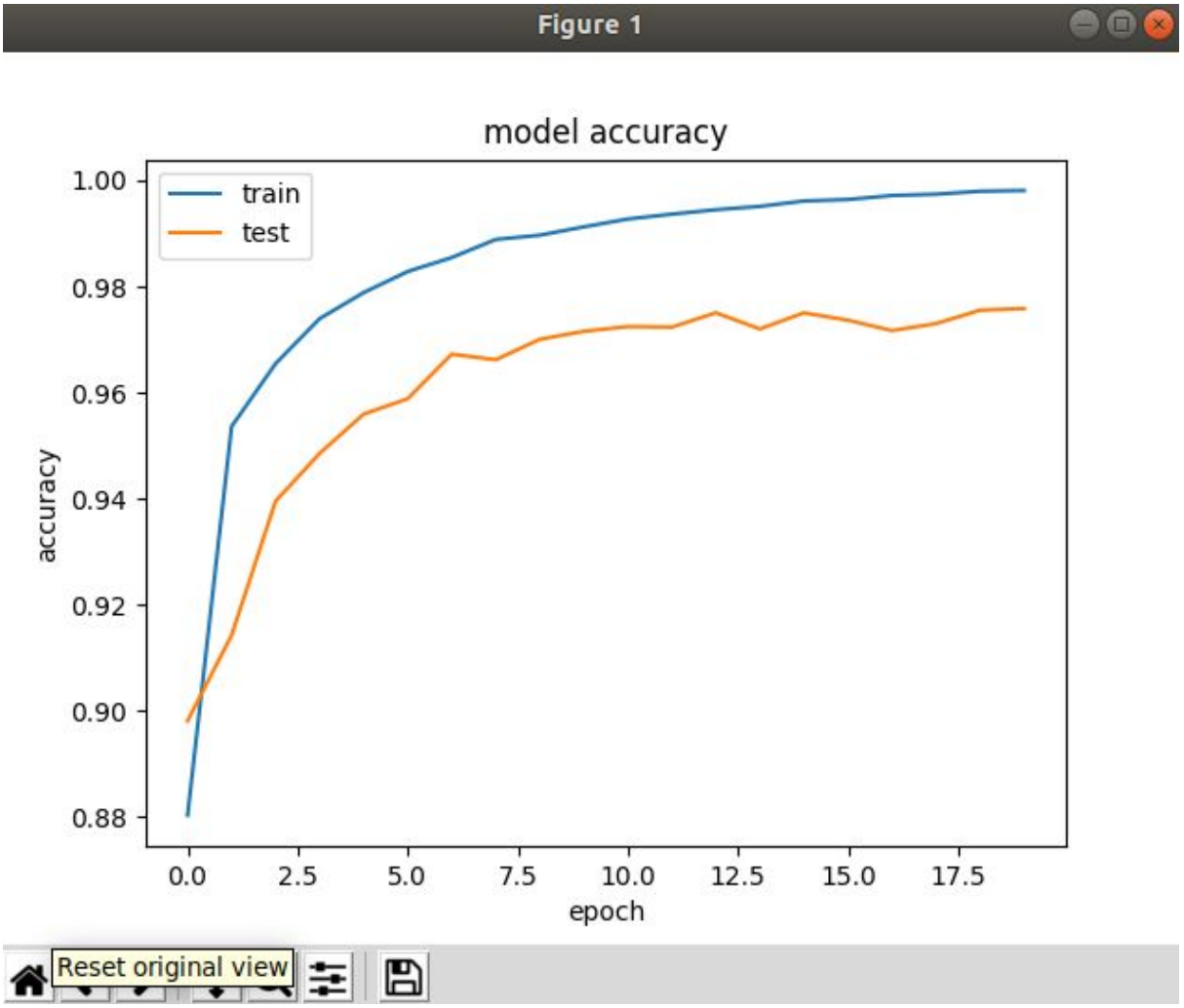
f1-score = .9912

---

ب-2) تابع ارور(loss):بدون drop out




تابع accuracy:بدون drop out




---

C) Drop out :

تعدادی از خروجی های یک لایه را صفر میکند و در نتیجه مدل ما در برابر نویز پاسخ بهتری میدهد و generalized تر است و همچنین برای جلوگیری از overfitting آنرا اضافه میکنیم

به این ترتیب در هربار اعمال drop out شبکه را به صورت رندم کوچک میکنیم و نرون هایی که حذف کرده ایم با وزن های قبلی به شبکه باز میگردند

نتایج قسمت ب با دراپ اوت در زیر اورده شده است

(5 نمودار بعدی نمودارهای acc,loss, precision, recall ,f1-score با در نظر گرفتن drop out هستند .)

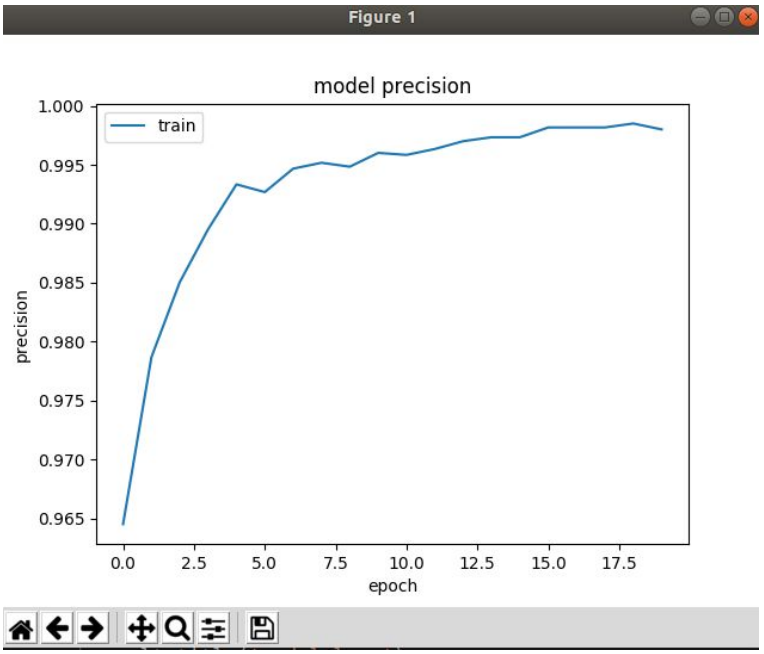


Figure 1
model precision
train
precision
epoch
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

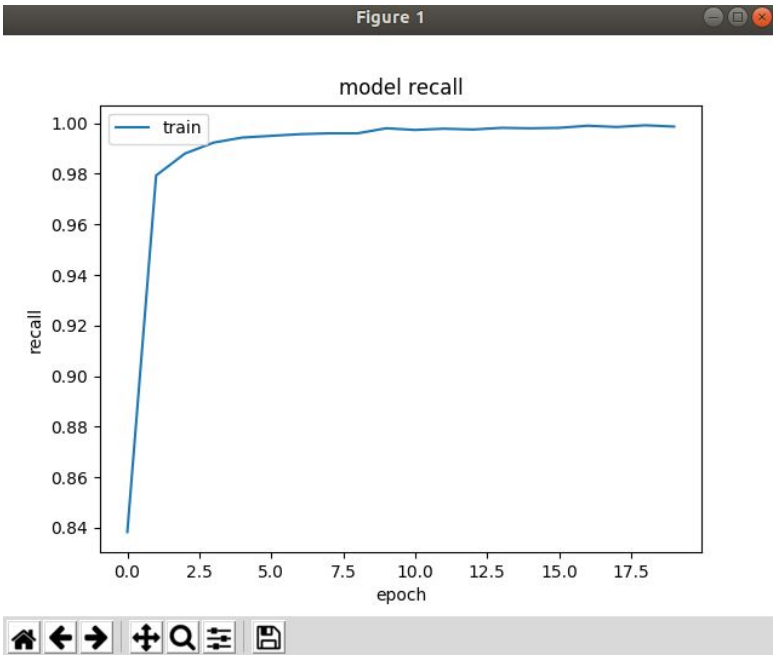


Figure 1
model recall
train
recall
epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

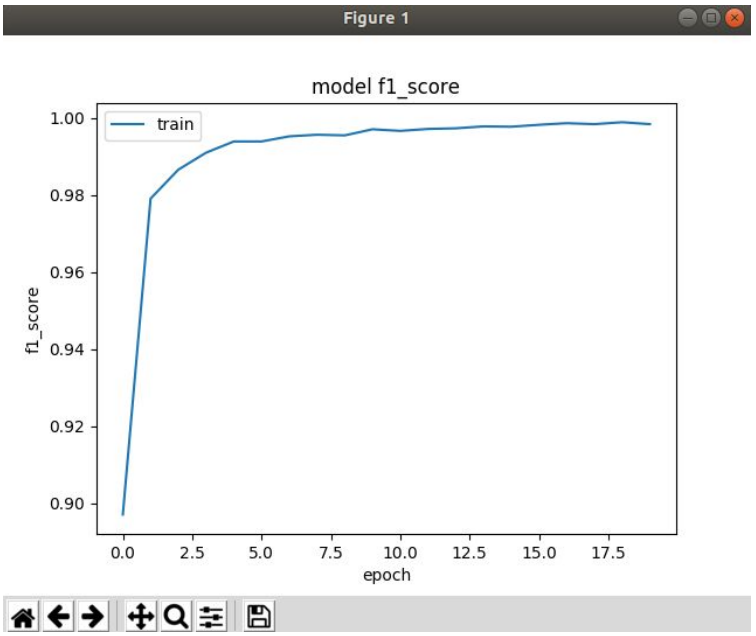


Figure 1
model f1_score
train
f1_score
epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

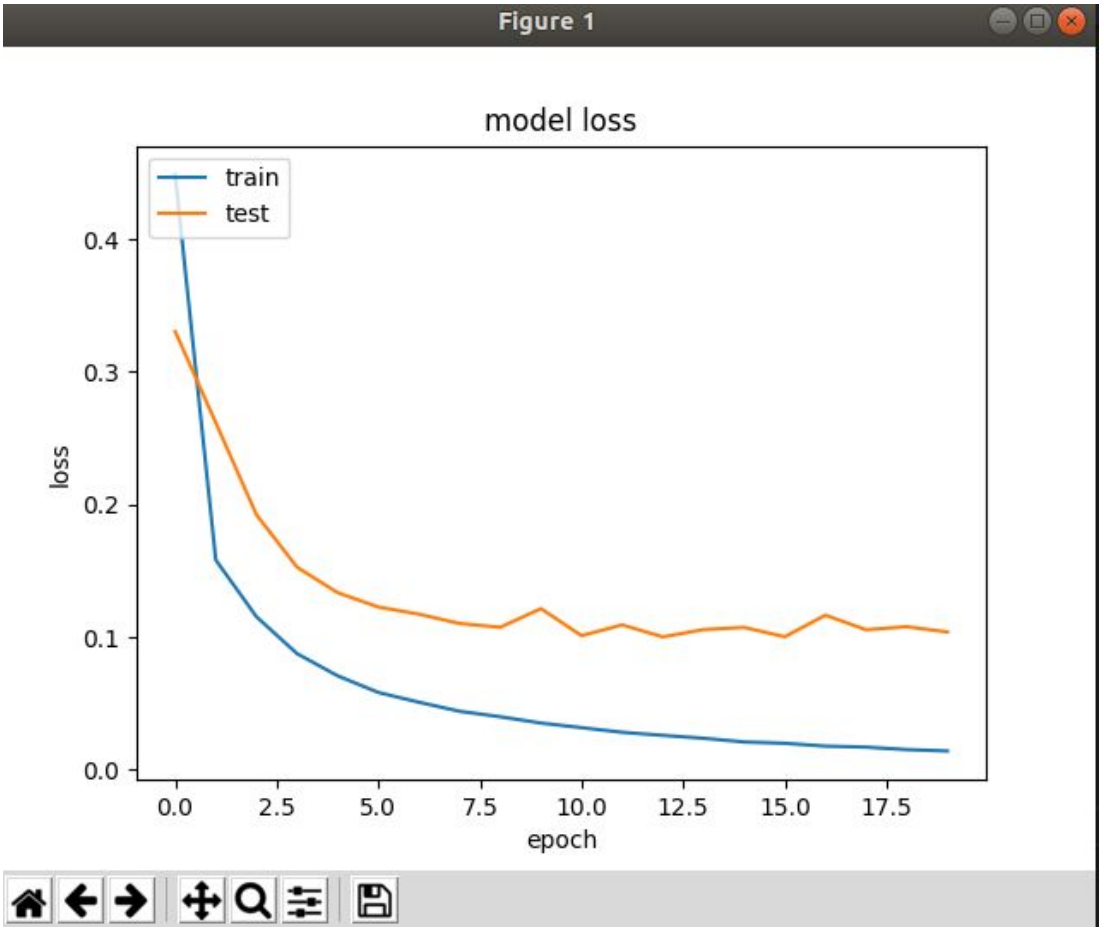
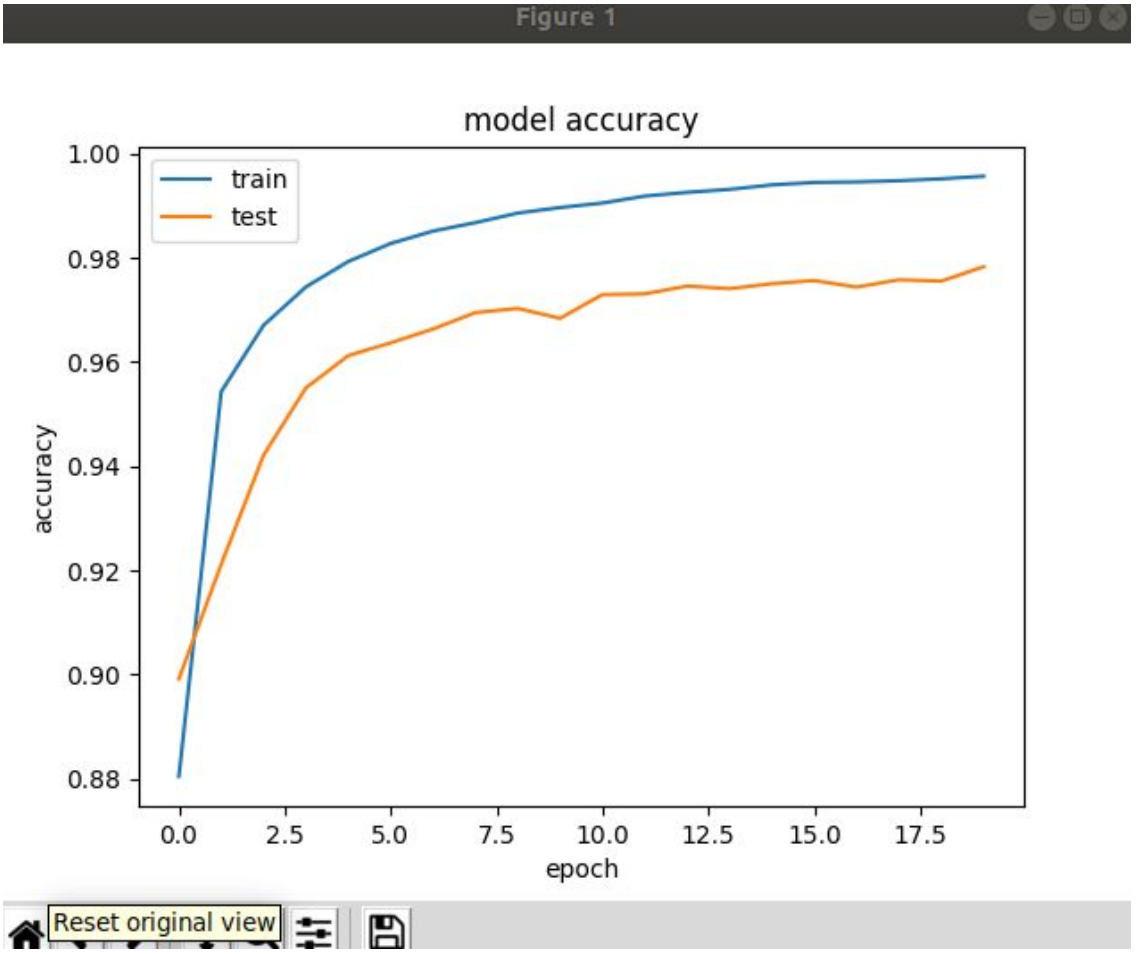
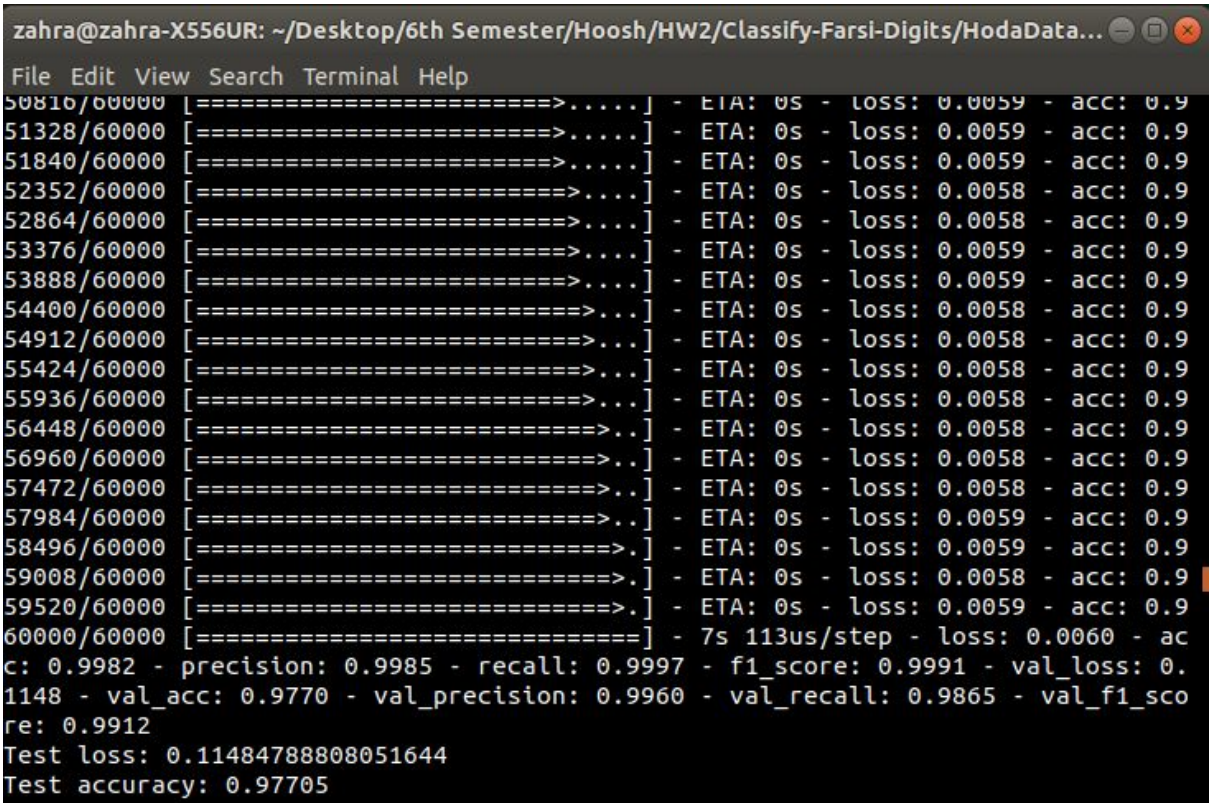
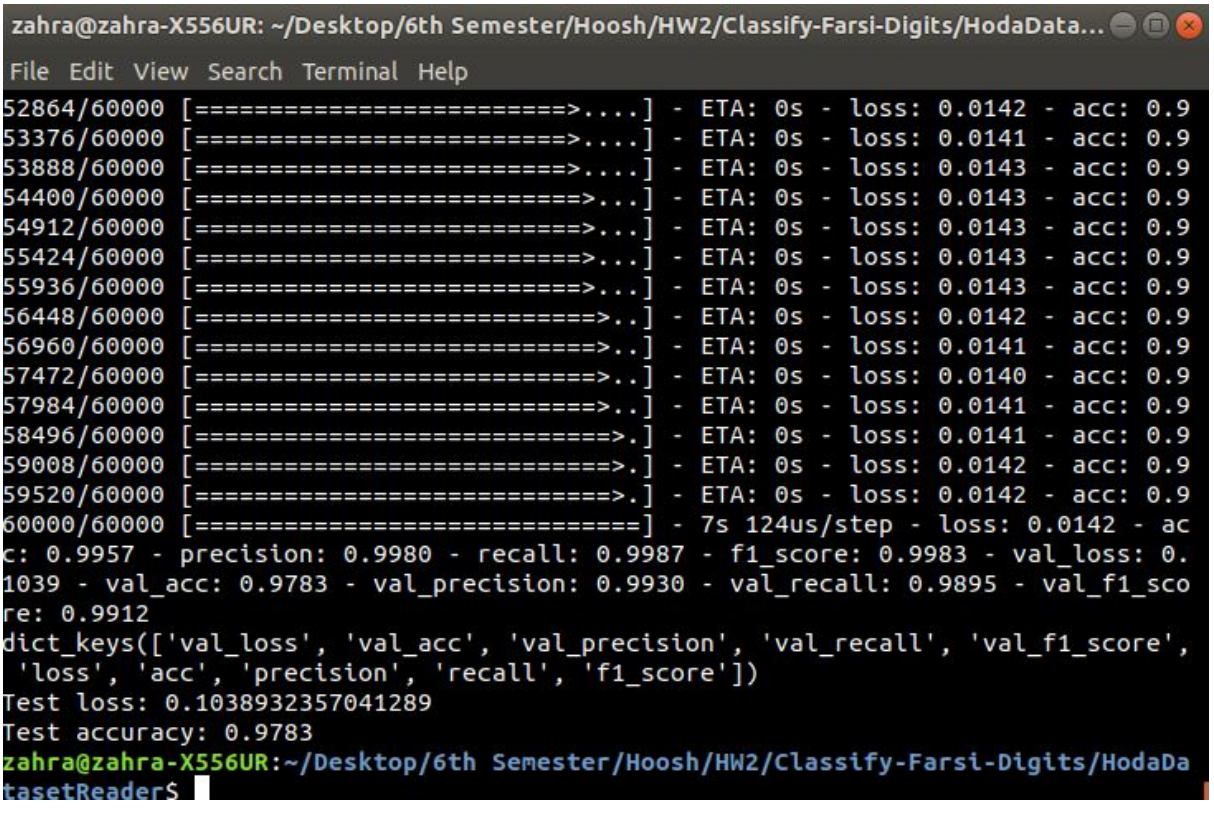


Figure 1
model loss
train
test
loss
epoch
0.0
0.1
0.2
0.3
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5




**مقایسه اعداد نهایی:**


zahra@zahra-X556UR: ~/Desktop/6th Semester/Hoosh/HW2/Classify-Farsi-Digits/HodaData...
File Edit View Search Terminal Help
50816/60000 [=========================>.....] - ETA: 0s - loss: 0.0059 - acc: 0.9
51328/60000 [=========================>.....] - ETA: 0s - loss: 0.0059 - acc: 0.9
51840/60000 [=========================>.....] - ETA: 0s - loss: 0.0059 - acc: 0.9
52352/60000 [==========================>....] - ETA: 0s - loss: 0.0058 - acc: 0.9
52864/60000 [==========================>....] - ETA: 0s - loss: 0.0058 - acc: 0.9
53376/60000 [==========================>....] - ETA: 0s - loss: 0.0059 - acc: 0.9
53888/60000 [==========================>....] - ETA: 0s - loss: 0.0059 - acc: 0.9
54400/60000 [===========================>...] - ETA: 0s - loss: 0.0058 - acc: 0.9
54912/60000 [===========================>...] - ETA: 0s - loss: 0.0058 - acc: 0.9
55424/60000 [===========================>...] - ETA: 0s - loss: 0.0058 - acc: 0.9
55936/60000 [===========================>...] - ETA: 0s - loss: 0.0058 - acc: 0.9
56448/60000 [============================>..] - ETA: 0s - loss: 0.0058 - acc: 0.9
56960/60000 [============================>..] - ETA: 0s - loss: 0.0058 - acc: 0.9
57472/60000 [============================>..] - ETA: 0s - loss: 0.0058 - acc: 0.9
57984/60000 [============================>..] - ETA: 0s - loss: 0.0059 - acc: 0.9
58496/60000 [============================>.] - ETA: 0s - loss: 0.0059 - acc: 0.9
59008/60000 [============================>.] - ETA: 0s - loss: 0.0058 - acc: 0.9
59520/60000 [============================>.] - ETA: 0s - loss: 0.0059 - acc: 0.9
60000/60000 [==============================] - 7s 113us/step - loss: 0.0060 - ac
c: 0.9982 - precision: 0.9985 - recall: 0.9997 - f1_score: 0.9991 - val_loss: 0.
1148 - val_acc: 0.9770 - val_precision: 0.9960 - val_recall: 0.9865 - val_f1_sco
re: 0.9912
Test loss: 0.11484788808051644
Test accuracy: 0.97705


zahra@zahra-X556UR: ~/Desktop/6th Semester/Hoosh/HW2/Classify-Farsi-Digits/HodaData...
File Edit View Search Terminal Help
52864/60000 [==========================>....] - ETA: 0s - loss: 0.0142 - acc: 0.9
53376/60000 [==========================>....] - ETA: 0s - loss: 0.0141 - acc: 0.9
53888/60000 [==========================>....] - ETA: 0s - loss: 0.0143 - acc: 0.9
54400/60000 [===========================>...] - ETA: 0s - loss: 0.0143 - acc: 0.9
54912/60000 [===========================>...] - ETA: 0s - loss: 0.0143 - acc: 0.9
55424/60000 [===========================>...] - ETA: 0s - loss: 0.0143 - acc: 0.9
55936/60000 [===========================>...] - ETA: 0s - loss: 0.0143 - acc: 0.9
56448/60000 [============================>..] - ETA: 0s - loss: 0.0142 - acc: 0.9
56960/60000 [============================>..] - ETA: 0s - loss: 0.0141 - acc: 0.9
57472/60000 [============================>..] - ETA: 0s - loss: 0.0140 - acc: 0.9
57984/60000 [============================>..] - ETA: 0s - loss: 0.0141 - acc: 0.9
58496/60000 [============================>.] - ETA: 0s - loss: 0.0141 - acc: 0.9
59008/60000 [============================>.] - ETA: 0s - loss: 0.0142 - acc: 0.9
59520/60000 [============================>.] - ETA: 0s - loss: 0.0142 - acc: 0.9
60000/60000 [==============================] - 7s 124us/step - loss: 0.0142 - ac
c: 0.9957 - precision: 0.9980 - recall: 0.9987 - f1_score: 0.9983 - val_loss: 0.
1039 - val_acc: 0.9783 - val_precision: 0.9930 - val_recall: 0.9895 - val_f1_sco
re: 0.9912
dict_keys(['val_loss', 'val_acc', 'val_precision', 'val_recall', 'val_f1_score',
 'loss', 'acc', 'precision', 'recall', 'f1_score'])
Test loss: 0.10389323570412 89
Test accuracy: 0.9783
zahra@zahra-X556UR:~/Desktop/6th Semester/Hoosh/HW2/Classify-Farsi-Digits/HodaDa
tasetReader$

اعداد حاصل به ترتیب نتایج بدون drop out و به همراه drop out هستند

| | با drop out | بدون drop out |
|---|---|---|
| test loss | 0.10 | 0.11 |
| test accuracy | 0.978 | 0.977 |

پس نهایتا دراپ اوت به بهبود شبکه ما کمک کرد :))

---

## D) validation set:

به طور کلی سه دسته دیتا مورد بحث داریم

train , test . validation(dev)

این دسته دیتایی است که مدل ما آنرا آموزش ندیده است و در واقع تاثیری روی شبکه و اوزان آن ندارد و فقط برای چک کردن به درد ما میخورد و احتمال overfitting را کاهش میدهد.

این خط به کد ما اضافه میشود.

```
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test))
```

داده های بالای من همگی دارای ولیدیشن ست هستند و میبنید که دارای عملکرد بسیار خوبی هستند ولی هنگامی که آنرا بردارم کمی از دقت شبکه کاسته میشود .

**بدون validation :**

دارای drop out:(نمودار ها پیوست شده اند)

| | no validation set | validation set |
|---|---|---|
| test loss | 0.107 | 0.10 |
| test accuracy | 0.97 | 0.98 |

بدون drop out:(نمودار ها پیوست شده اند)

| | no validation set | validation set |
|---|---|---|
| test loss | 0.12 | 0.11 |
| test accuracy | 0.96 | 0.977 |

نتیجه گیری کلی : بهترین حالت با validation set و به همراه drop out می باشد.

---

E)batch size:

با دادن بچ سایز به مجموعه داده هایمان را تکه تکه کرده با هم دسته ای در نظر میگیریم و این کار باعث سرعت بخشیدن به آموزش میشود و مقدار کمتری از حافظه میگیرد

سه نوع داریم :

1)stochastic: batch size = 1

2)batch mode در این حالت تعداد کل داده ها مساوی سایز بچ است و در یک مرحله انجام میشود

3) mini batch :128 به طور مثال عدد

در حالت اول مدل را روی یک داده آپدیت میکند که خوب نیست و همیشه باعث کم شدن خطا نمیشود.
در حالت دوم مدل همه ی داده را میبنید و سپس وزن هارا آپدیت میکند که همیشه باعث کم شدن خطا میشود اما مشکلش این است که هزینه محاسباتی دارد.

بنابراین مدل سوم امروزه بیشتر استفاده میشود که چیزی بینابین این دوحالت است:))