به نام خدا

ملیکا احمدی رنجبر 97521036

❖

- با توجه به توضیحات سایت در رابطه با نمودار و همچنین مطالب کلاس, Optimizer مدل Adam که نسخه‌ای از Stochastic Gradient Descent هست, با استفاده از دو مزیت مدل‌های AdaGrad, که برای Parameter یک نرخ یادگیری مشخص بر اساس مقادیر گذشته وجود داشت و تغییرات را در جهت‌های مختلف کنترل می‌کرد. به مرور زمان نیز به سمت صفر می‌رد.

```
grad_squared = 0
while True:
  dx = compute_gradient(x)
  grad_squared += dx * dx
  x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```

همچنین RMSProp, که همان تکنیک AdaGrad را دارد با این تفاوت که کنترل بیشتری بر روی تغییر نرخ یادگیری دارد و تمرکز روی Gradientهای جدیدتر است نسبت به مربعات Gradientهای پیشین.

```
grad_squared = 0
while True:
  dx = compute_gradient(x)
  grad_squared = decay_rate * grad_squared + (1 - decay_rate) * dx * dx
  x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```

اما مدل Adam با به کارگیری دقیق متغییرها این تغییرات را بهتر کنترل کرده و در Datasetهای بزرگ که برخی موارد نیز داده‌های پراکنده دارند بسیار مورد استفاده قرار می‌گیرد و عملکرد بهتری نسبت به سایر Optimizerها دارد. هم سرعت را در نظر گرفته, هم همانند RMSProp. و پس از آن نسبت هایشان را تعیید کرده است, تا از تغییرات شدید ناگهانی جلوگیری شود.

```
first_moment = 0
second_moment = 0
for t in range(1, num_iterations):
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment  + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  first_unbias = first_moment / (1 - beta1 ** t)
  second_unbias = second_moment / (1 - beta2 ** t)
  x -= learning_rate * first_unbias / (np.sqrt(second_unbias) + 1e-7))
```

Momentum

RMSProp

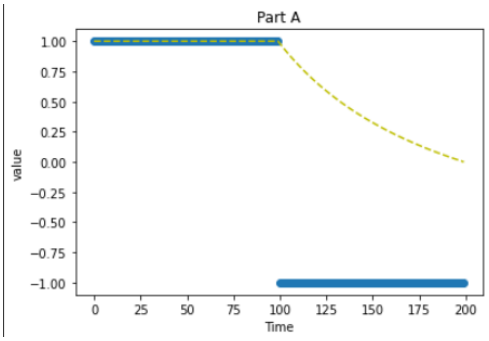
Bias correction

AdaGrad / RMSProp

- به میزان پراکندگی داده‌ها و همچنین تعدادشان, مثلا اگر Nesterov Momentum برای یک Dataset کوچک خوب باشد, ممکن از تعداد بیشتر داده به علت حجم محاسباتی بالا عملکرد ضعیف‌تری داشته باشد. مقدار Memory که هر Optimizer استفاده می‌کند نیز دارای اهمیت است که باز هم بستگی به تعداد داده دارد. Noise را نیز می‌توان در نظر گرفت برای عاملی موثر در عملکرد Optimizer. اما یکی از مهم ترین عوامل اندازه Dataset است. طراحی شبکه نیز ممکن است عاملی موثر در این عملکرد باشد برای مثال, تعداد لایه‌ها, Activation Function و موارد دیگر.

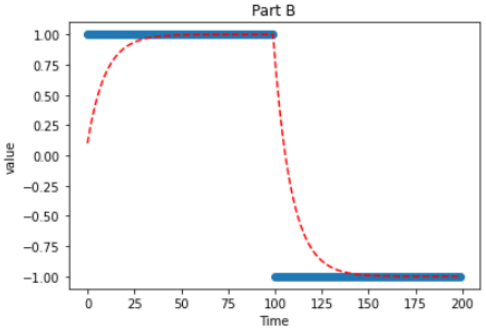
❖

- برای محاسبه میانگین در این حالت نیاز به Memory داریم و باید تمامی مقادیر قبلی را محاسبه شده داشته باشیم. نمودار حاصل نیز به شکل روبه رو است:



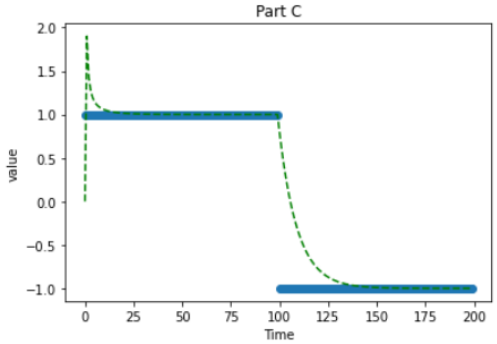
به طور کلی این روش میانگین تخمین دقیقی از پراکندگی داده دارد همانطور که در شکل مشخص است و تمامی روزها را حساب می‌کند.

- روش دوم دقیق‌تر است و برای Beta = 0.9 10 روز را در نظر می‌گیرد و با افزایش Beta تعداد روزهایی که برای محاسبه در نظر می‌گیرد بیشتر می‌شود و به همین دلیل نمودار Smoothتر خواهد بود, در اینجا داریم:

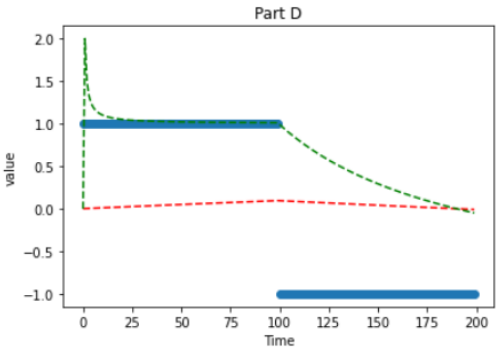


تخمین دقیق و نزدیک‌تر به داده است. از طرفی هزینه محاسباتی زیادی ندارد چراکه فقط مقدار پیشین لازم است.

- در این حالت سوم برای بهتر شدن تخمین در tهای ابتدایی الگوریتم را کمی تغییر می‌دهیم تا دقیق‌تر شود. اما برای tهای بزرگ همانند روش پیش عمل می‌کند:



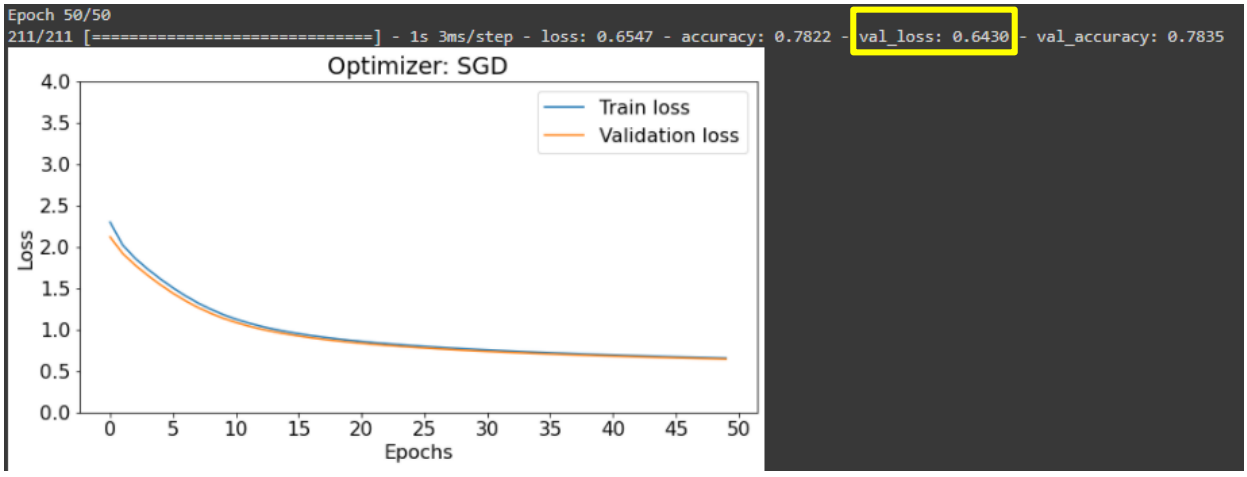
- در این قسمت مقایسه مقدار Beta است. همانطور که پیشتر توضیح دادم, با افزایش Beta میانگین با در نظر گرفتن تعداد t بیشتری انجام می‌شود. عکی روبه‌رو کاملا این مطلب را روشن می‌کند.



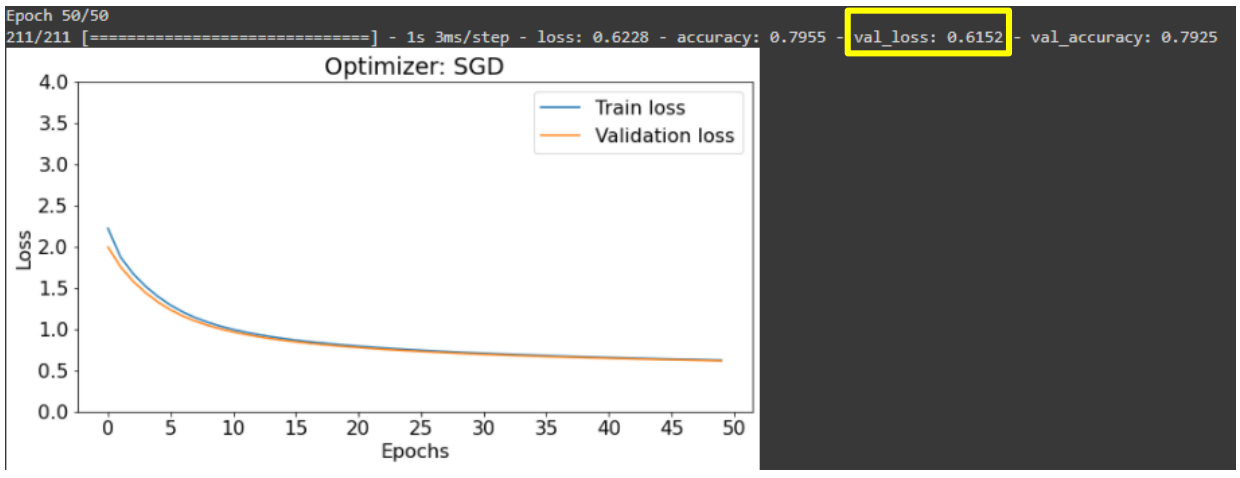
به طور کلی این سوال برای آشنا شدن با روش محاسبه مربوط به الگوریتم‌های Optimization بود و دلیل برتری و دقیق بودن برخی از روش‌ها مثل Adam.

❖ با استفاده از 16 نورون برای لایه مخفی نتیجه زیر بدست آمد:

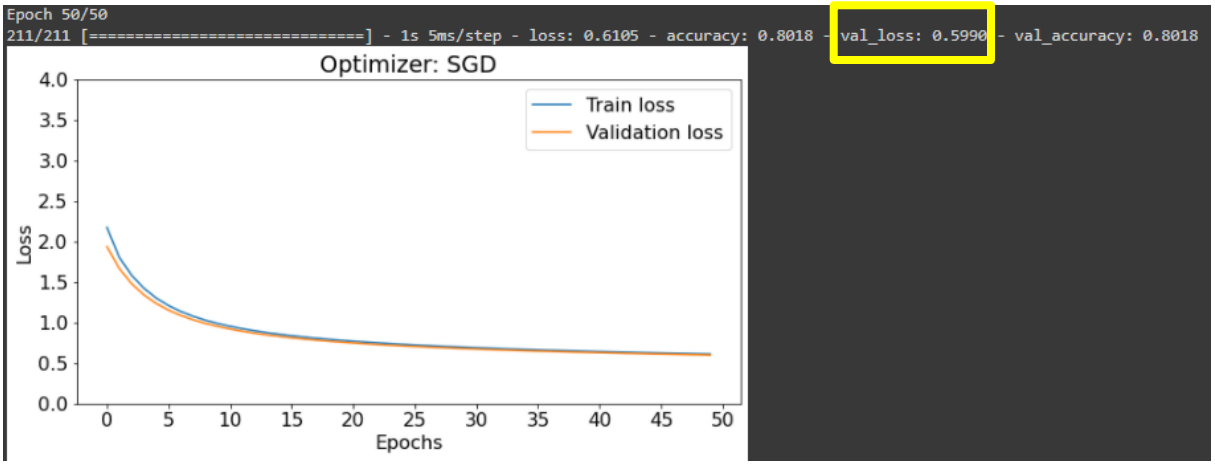
32 نورون:



64 نورون:



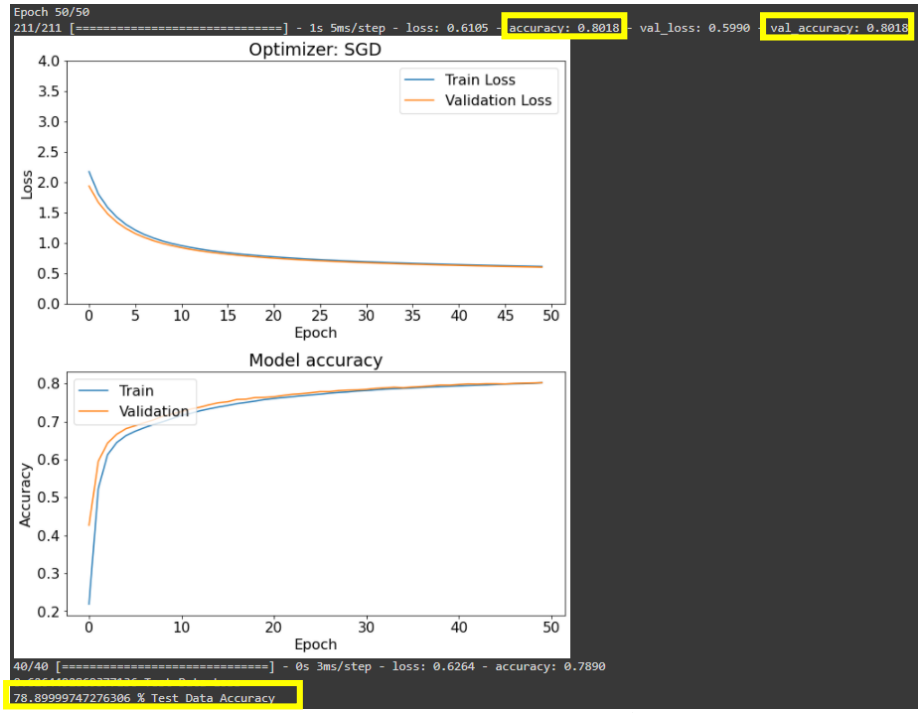
128 نورون:



با توجه به قایسه نتایج حاصل از تعداد نورون‌های مختلف, انتخاب لایه مخفی با 128 نورون بهتر است, به دلیل اینکه می‌تواند Featureها را دقیق‌تر تفکیک کند و جزئی‌تر مراحل انجام شود, برای مثال در 16 نورون تمام جزییات 16 دسته خواهد شد و در نهایت در جمع این 16 تا نتیجه بدست می‌آید, در حالی که در لایه با 128 نورون روی جزییات می‌توان دقیق‌تر شد و نتیجه بهتری دریافت کرد.

- Fashion MNIST: مجموعه‌ای از داده‌های انواع مختلف لباس, شامل Category 10 است که 60000 نمونه برای آموزش و 10000 نمونه برای تست دارد. Tensor برای این داده‌های آموزش به شکل (60000, 28, 28) است و برای Label به شکل (,60000) می‌باشد. برای تست نیز به همین ترتیب است.
- در ابتدا 10 درصد که شامل 6000 نمونه می‌شود. به این دلیل که مطمئن شویم داده‌ها Over fit نشوند و همچنین میزان دقت را برای داده‌های به جز آموزش در حین Learning سنجیم. تعداد زیاد آن, Training Data را کم می‌کند و مقدار کم آن نیز شاید بی‌فایده باشد بنابراین 0.1 را د نظر گرفتم برای شروع و در سوالات جلوتر نیز با تغییر آن به مقادیر مختلف، مناسب بود همان 0.1 ثابت می‌شود.

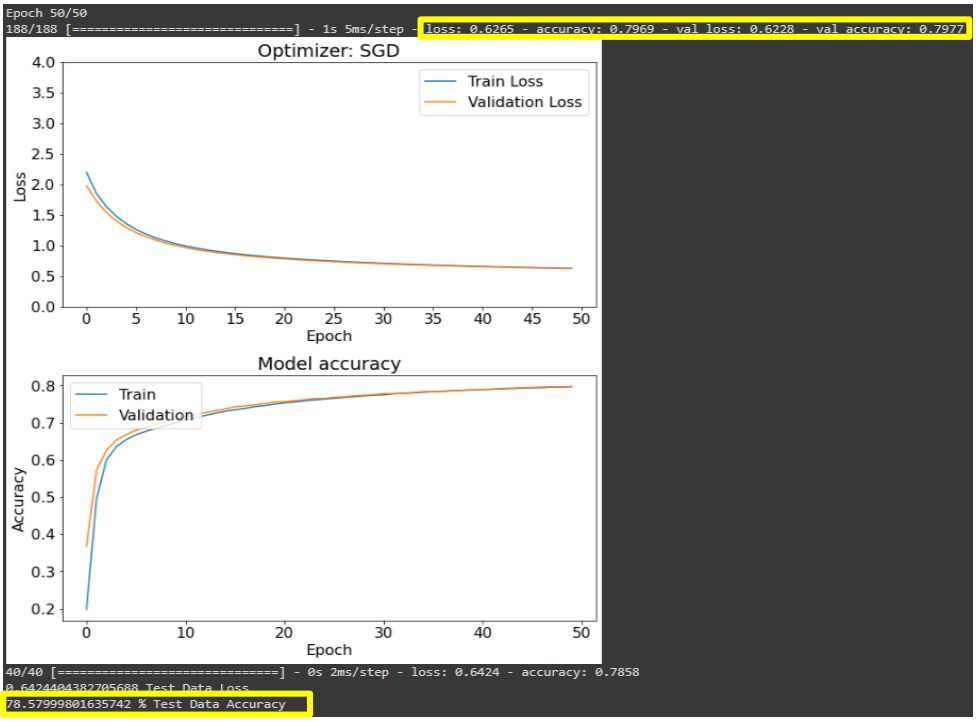
- دقت به شکل زیر است:



از این مقادیر متوجه می‌شویم که دقت برای Test Data پایین‌تر از دقت Training و Validation است. ممکن است در صورن تغییر اندازه Validation Data تاثیر بهتری بگیریم که در ادامه بررسی می کنیم. دقت Training و Validation هم لوزما برابر نیست.

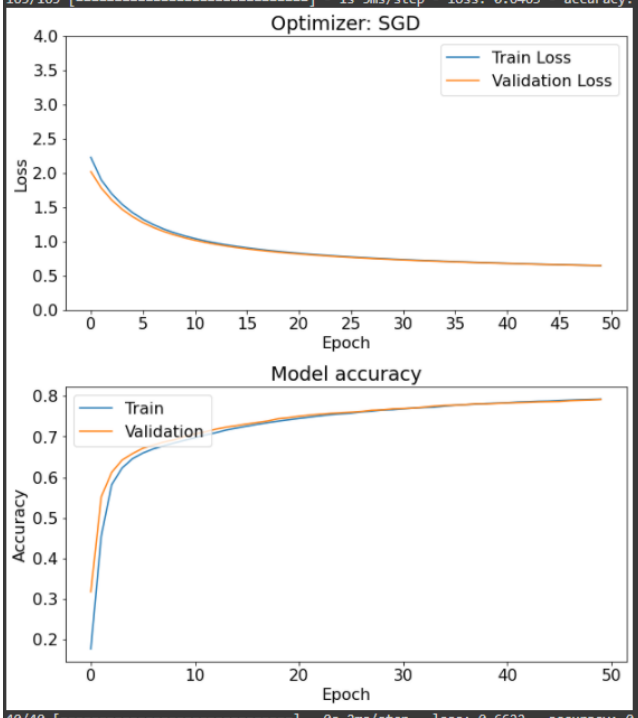
- در این قسمت 3 حالت برای Validation در نظر گرفتیم: 0.1و 0.2و 0.3 که 0.1 حالت ابتدایی بود. نتیجه به شکل زیر است:

0.2 و 128: در مقایشه با 0.1 دقت‌ها کمتر شد.



0.2 و 64: در اینجا دقت کمی از حالت پیش بهتر شده.

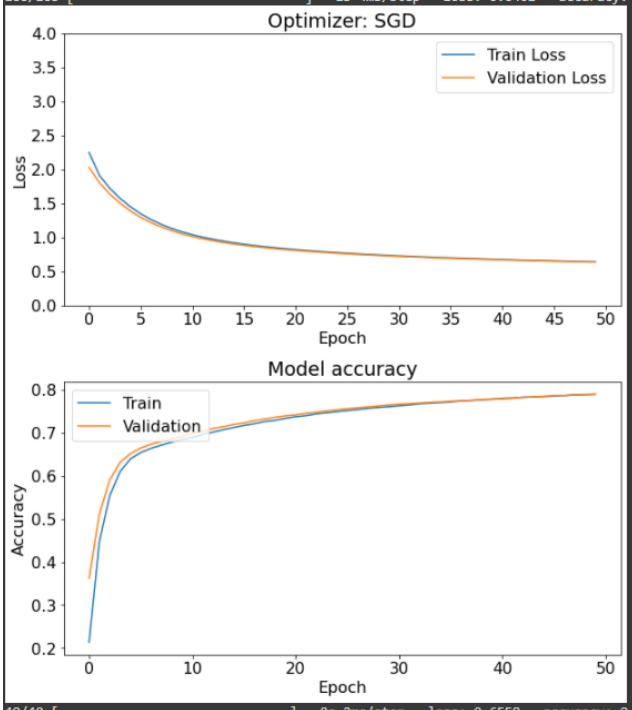
```
Epoch 50/50
165/165 [==============================] - 1s 5ms/step - loss: 0.6463 - accuracy: 0.7924 - val_loss: 0.6453 - val_accuracy: 0.7918
```




```
40/40 [==============================] - 0s 2ms/step - loss: 0.6622 - accuracy: 0.7794
0.6622012853622437 Test Data Loss
77.93999910354614 % Test Data Accuracy
```

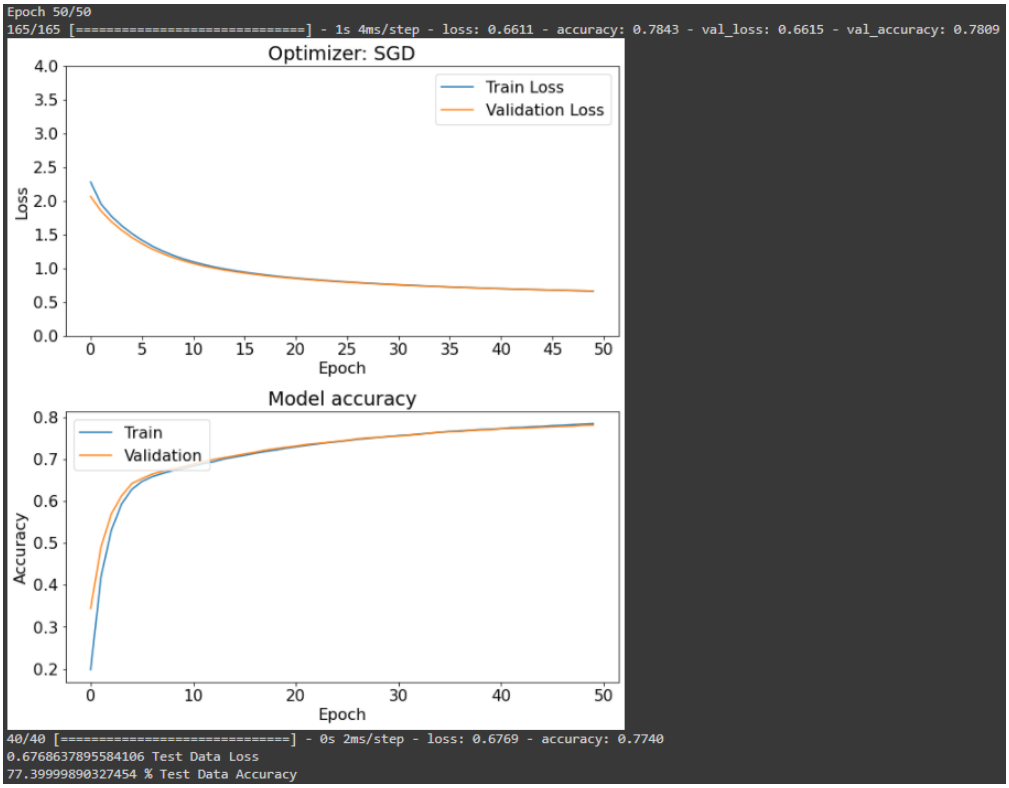
0.3 و 128:

```
Epoch 50/50
188/188 [==============================] - 1s 4ms/step - loss: 0.6401 - accuracy: 0.7901 - val_loss: 0.6374 - val_accuracy: 0.7897
```




```
40/40 [==============================] - 0s 2ms/step - loss: 0.6558 - accuracy: 0.7806
0.6557976007461548 Test Data Loss
78.06000113487244 % Test Data Accuracy
```

0.3 و 64:



بنابراین بهترین تعداد نورون لایه مخفی 128 است به همان دلیل مطرح شده در ابتدا و افزایش Validation Data تاثیر بسیار کمی بر روی تعداد نورون دارد. در نتایج Test Data قطعا تاثیرگذار خواهد بود, چراکه تعداد داده‌هایی که بر اساس آن آموزش می‌بینند تغییر می‌کند. باید حد تعادل را برای Validation Data یافت که در اینجا با توجه به نتایج همان 0.1 است.

- اکنون بهترین تعداد لایه 128 و Validation Data اندازه 0.1 داده‌های آموزش است. حال Optimizerهای مختلف را امتحان می‌کنیم. تمام مقادیر را برای SGD داریم.

Adam: با اختلاف زیادی از SGD بهتر است و دلایل آن تا حدی در سوال اول توضیح داده شده. (برتری Adam)

```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.1281 - accuracy: 0.9560 - val_loss: 0.3677 - val_accuracy: 0.8853
* * * * * * * * * * * * * * * * * * * * * *
 - - - - - - - - - - - - - - - - - - - - - -
40/40 [==============================] - 0s 2ms/step - loss: 0.3651 - accuracy: 0.8897
0.36508405208587646 Test Data Loss
88.96999955177307 % Test Data Accuracy
```

RMSProp: این Optimizer نیز بسیار بهتر از SGD تنها عمل میکند اما نتیجه از Adam کمی پایین‌تر است.

```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.1276 - accuracy: 0.9524 - val_loss: 0.3659 - val_accuracy: 0.8963
* * * * * * * * * * * * * * * * * * * * * *
 - - - - - - - - - - - - - - - - - - - - - -
40/40 [==============================] - 0s 2ms/step - loss: 0.4012 - accuracy: 0.8885
0.40117165446281433 Test Data Loss
88.84999752044678 % Test Data Accuracy
```

Adagrad: همان طور که با توجه به مطالب درس انتظار می‌رفت این نیز عملکرد بهتری از SGD دارد و نسخهimproved آن است.

جمع‌بندی کلی:

SGD < Adagrad < RMSProp < Adam

```
Epoch 50/50
211/211 [==============================] - 1s 4ms/step - loss: 0.5178 - accuracy: 0.8304 - val_loss: 0.5132 - val_accuracy: 0.8272
* * * * * * * * * * * * * * * * * * * * * * * *
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
40/40 [==============================] - 0s 2ms/step - loss: 0.5395 - accuracy: 0.8191
0.5394964814186096 Test Data Loss
81.9100022315979 % Test Data Accuracy
```

Adagrad برای هر Parameter یک Learning Rate خاص در نظر می‌گیرد, RMSProp آن را بهتر کنترل می‌کند و در نهایت Adam کامل‌ترین نسخه این Improvementها است.

- تعداد نورون 128, 0.1 برای Validation Data و Adam را به عنوان Optimizer انتخاب کردیم, حال نوبت Learning Rate است که تا الان 0.001 بوده.

با انتخاب 0.0001: دقت پایین آمد, کوچک است.

```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.2937 - accuracy: 0.8975 - val_loss: 0.3368 - val_accuracy: 0.8805
* * * * * * * * * * * * * * * * * * * * * * * *
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
40/40 [==============================] - 0s 2ms/step - loss: 0.3604 - accuracy: 0.8714
0.36041131615638733 Test Data Loss
87.1399998664856 % Test Data Accuracy
```

با انتخاب 0.01: کمی بهتر از قبل است اما از 0.001 دقت کمتری دارد.

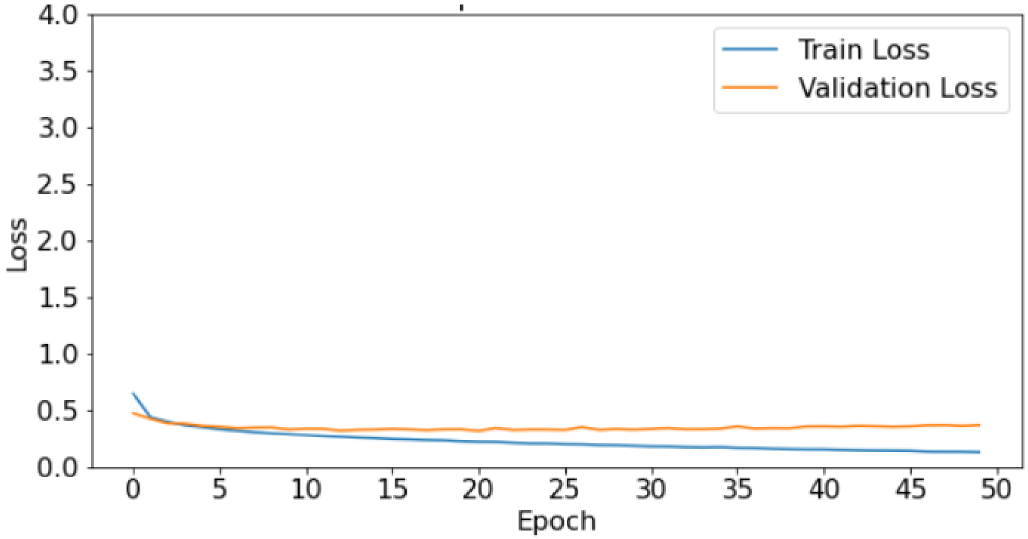
```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.2008 - accuracy: 0.9253 - val_loss: 0.4743 - val_accuracy: 0.8788
* * * * * * * * * * * * * * * * * * * * * * * *
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
40/40 [==============================] - 0s 2ms/step - loss: 0.4859 - accuracy: 0.8731
0.4858579933643341 Test Data Loss
87.30999827384949 % Test Data Accuracy
```

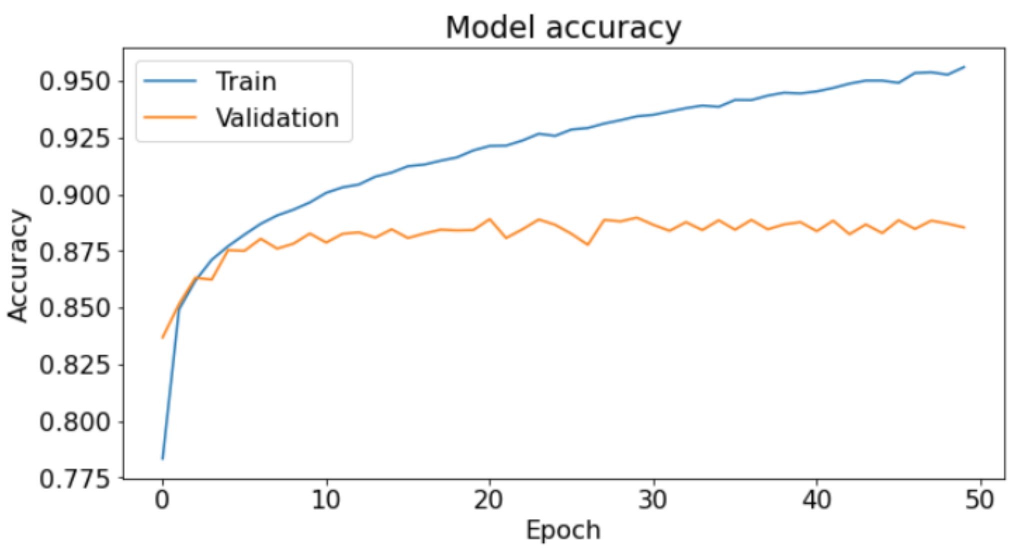
با انتخاب 0.1: مناسب نیز و کمی بزرگ است.

```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.6848 - accuracy: 0.7571 - val_loss: 0.7543 - val_accuracy: 0.7473
* * * * * * * * * * * * * * * * * * * * * * * *
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
40/40 [==============================] - 0s 2ms/step - loss: 0.7511 - accuracy: 0.7456
0.7511362433433533 Test Data Loss
74.55999851226807 % Test Data Accuracy
```

○ در این قسمت آموزش را 10 بار اجرا کردیم:

```
Epoch 50/50
211/211 [==============================] - 1s 5ms/step - loss: 0.1281 - accuracy: 0.9560 - val_loss: 0.3677 - val_accuracy: 0.8853
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
40/40 [==============================] - 0s 3ms/step - loss: 0.3651 - accuracy: 0.8897
0.36508405208587646 Test Data Loss
88.96999955177307 % Test Data Accuracy
```

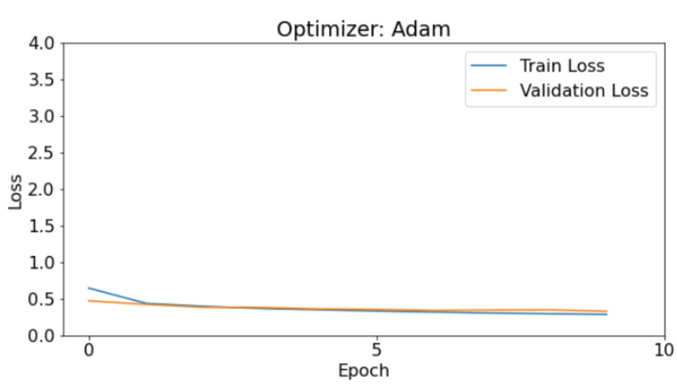


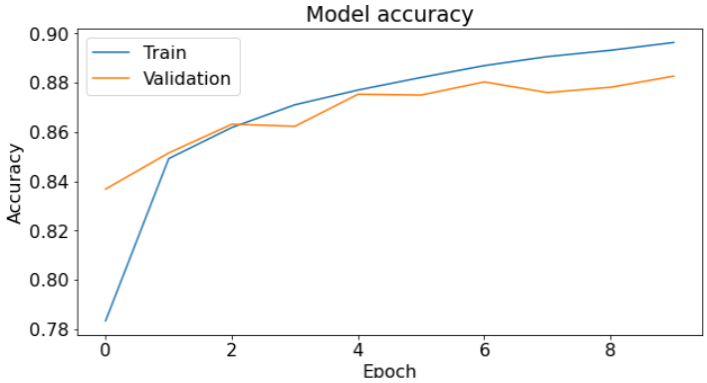


برای 50 اعداد بالا و برای 10 به شکل زیر:

```
Epoch 10/10
211/211 [==============================] - 1s 5ms/step - loss: 0.2861 - accuracy: 0.8964 - val_loss: 0.3295 - val_accuracy: 0.8827
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
40/40 [==============================] - 0s 2ms/step - loss: 0.3511 - accuracy: 0.8760
0.3511280119419098 Test Data Loss
87.59999871253967 % Test Data Accuracy
```

قطعا در تعداد Iteration بیشتر یادگیری و در نتیجه نتیجه تست بهتر است.





در رابطه با همگرا شدن نیز خیر نیست، چراکه در صورت ادامه دادن به Train مقدار Loss کمتر خواهد شد.

قسمت آخر، در ابندا مطرح شده، برای انتخاب تعداد نورون لایه میانی.

❖منابع:

- https://machinelearningjourney.com/index.php/2021/01/06/rmsprop/
- https://github.com/MariaStamouli/fashion-mnist/blob/master/Keras_MLP.ipynb
- Coursera Deep Learning Course