به نام خدا

محمدامین جمشیدی گوهرریزی

Project2:

توجه شود چون که تعداد عکس ها چه برای train ,validation,test مقدار قابل توجی است و اینکه در طول این پروژه با شافل کردن های داده های train و validation و انجام فراین train مشاهده شده مقادیرacc به ازای هر شافل واریانس زیادی ندارد و می توان فهمید که تعداد دیتا برای هر بخش مناسب است و نیازی به ارزیابی مدل از طریق روش های پویا نیست و در اینجا از روش ایستا برای ارزیابی مدل ها استفاده شده است

به عنوان preprocess در مراحل اولیه از نورمالیزه کردن مقادیر ورودی یعنی با تقسیم مقادیر بر 255 استفاده می کنیم و در ادامه کار با رسید به معماری نهایی از تکنیک های دیگری نیز استفاده خواهیم کرد.

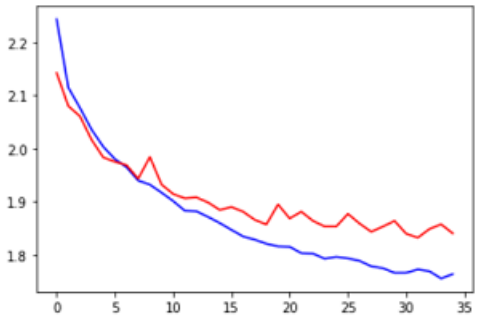
1:تعیین تعداد نورون ها با یک لایه : (در فایل
setting_number_of_norons_of_shalow_NN.ipynb
کد های این قسمت هست)

در این قسمت با تعیین مقادیر مختلف برای نورون های لایه پنهان و ثابت نگه داشتن سایر پارامتر ها به دنبال بهترین معماری تک لایه می رویم:

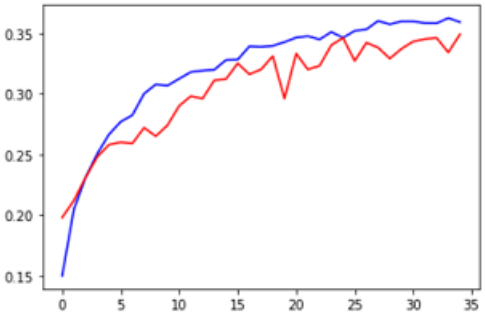
1:تعداد نورون=50:

÷

نمودار loss:

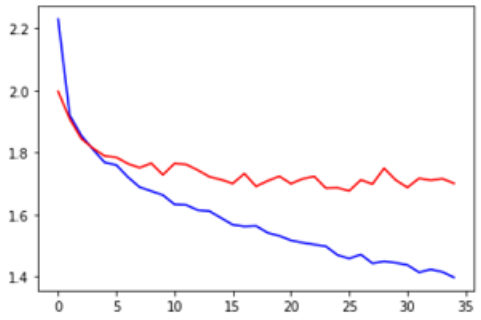


نمودار acc:

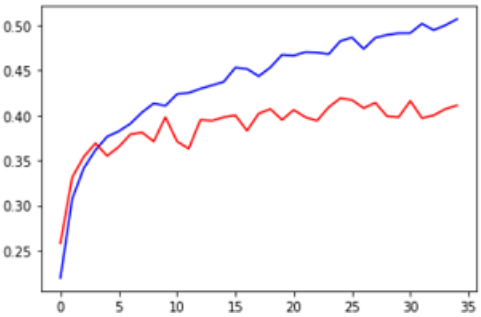


تعداد نورون =150:

نمودار loss:

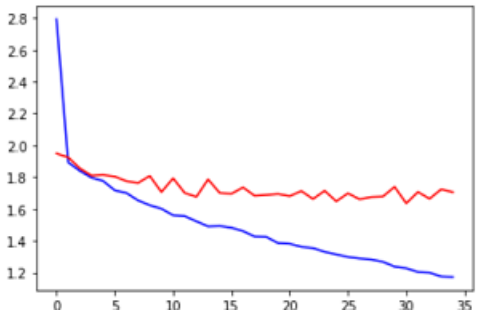


نمودار acc:

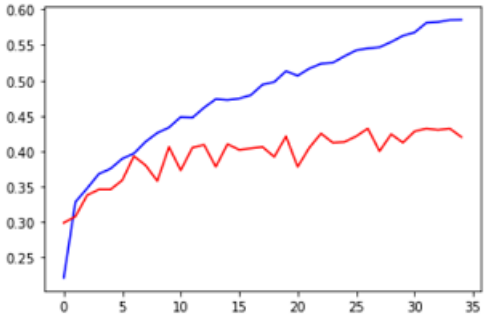


تعداد نورون=1000:

نمودار loss:



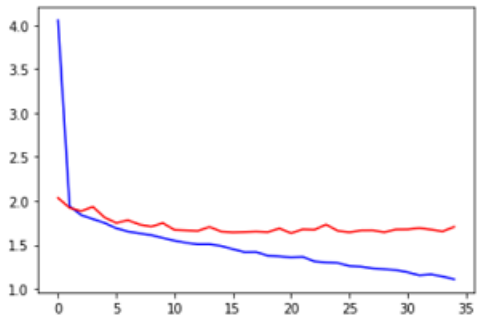
نمودار acc:



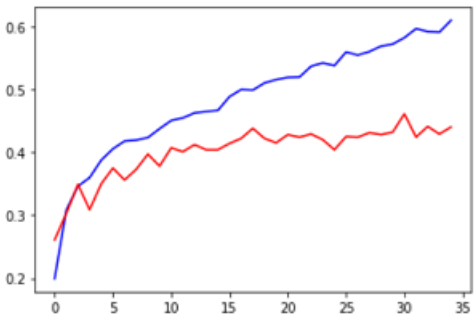
÷

تعداد نورون =3000:

نمودار loss:

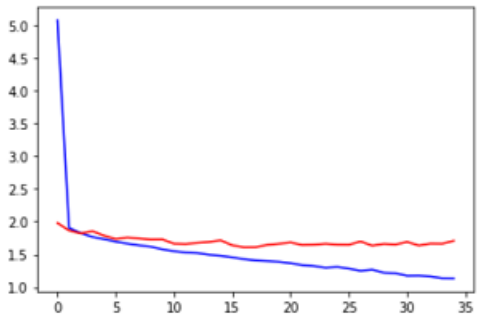


نمودار acc:

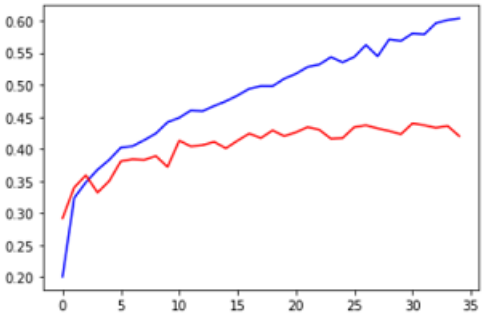


تعداد نورون =6000:

نمودار loss:



نمودار acc:

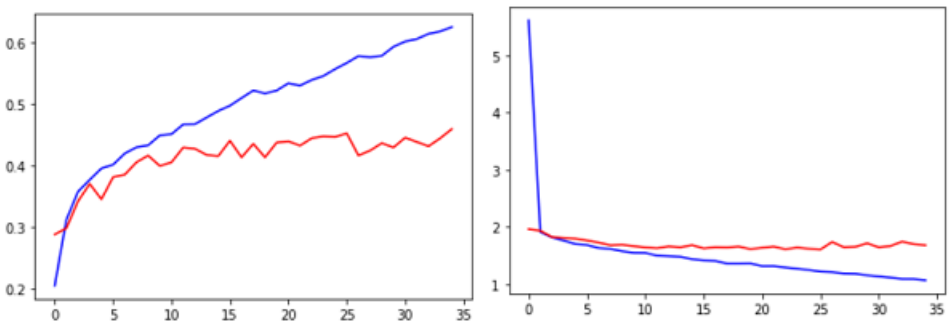


تعداد نورون =10000:

نمودار loss:

نمودار acc:

÷



همانطور که مشاهد می شود در نوع معماری یعنی shallow با افزایش تعداد نورون ها شاهد بهبود هایی در مدل هستیم و آخرین تست یعنی 10000 بهترین نتیجه را داد.

البته از طرفی دقت شود هر چه تعداد نورون ها بیشتر شده فاصله نمودار acc_trainو

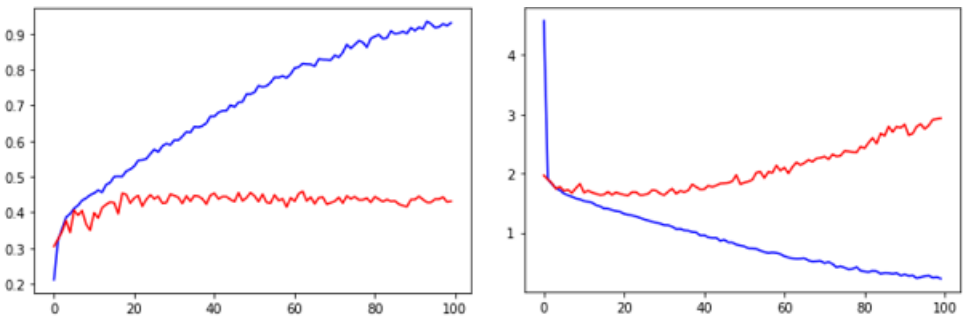
acc_val بیشتر شد ومدل به سمت overfitرفت ولی در حالت تعداد کم سرعت ترین مدل از سایر مدل ها بسیار کمتر بود و بعد طی epoch برابر با سایر حالات مدل به acc_train مطلوبی نمیرسید و همچنان در underfit باقی می ماند .

ولی همانطور که مشاهد شد با افزایش های زیاد در تعداد نورون ها تغییرات در acc_train خیلی محسوس نبود وبرای همین سنگین کردن شاید بهتر باشد بین با epochبیشتری تعداد نورون های 1000و50 و 10000 را مقایسه کنیم تا به انتخاب بهتری برسیم:

تعداد epoch=100:

تعداد نورون =10000

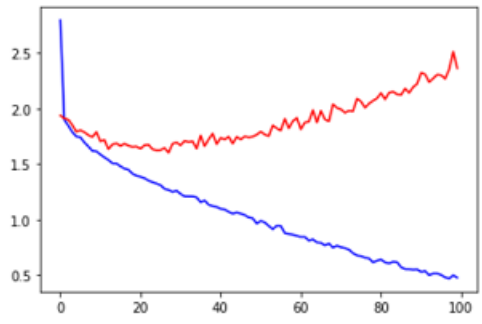
نمودار loss: نمودار acc:



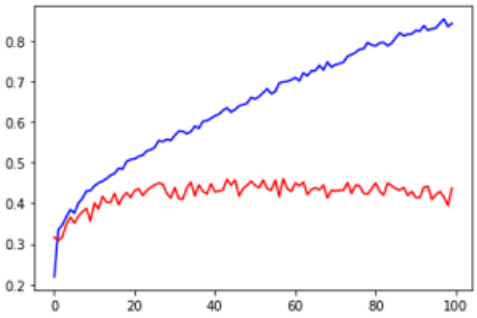
÷

تعداد نورون =1000:

نمودارloss:

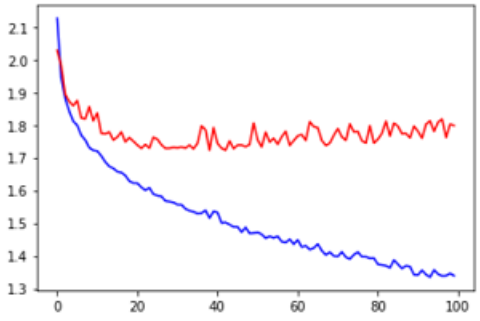


نمودار acc:

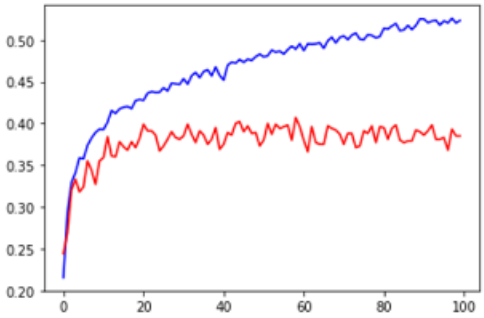


تعداد نورون =50:

نمودار loss:



نمودار acc:



همانطور که از مقایسه این 3 حالت و حالت های قبل نتیجه میشود که معماری shallow نمیتواند acc_valبیشتر از 46 در صد را تامین کند ولی در acc_train تفاوت های اساس ایجاد میشود برای همین نورون =1000 یا به نوعی در همین رنج را می توان بهترین انتخاب دانست چون که به acc_val ماکسیمم یعنی 46می رسند و هم می توان با تعداد acc_train تقریبا برابری با مدل هایی با تعداد نورون خیلی بشتر (مثلا 10000تا) دارند و این تفاوت جزی را هم در acc_valمیتوان با تعداد epochبیشتر تامین کرد

÷

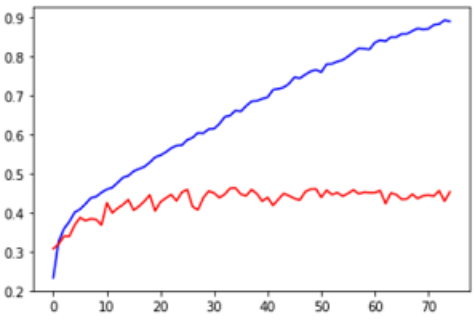
2:تعیین تابع فعالیت مناسب: (در فایل
setting_activation_function_for_shallow_NN

کد های مربوط به این بخش موجود است):

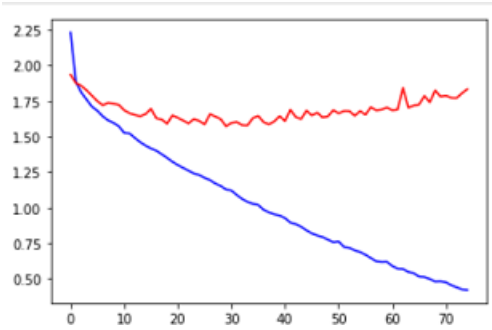
در این قسمت با تنظیم تعداد epoch=75 و تعداد نوروها بر 500 سه تابع relu,sigmoid,tanh را با هم قیاس کرده:

تابع sigmoid:

نمودار loss:



نمودار acc:

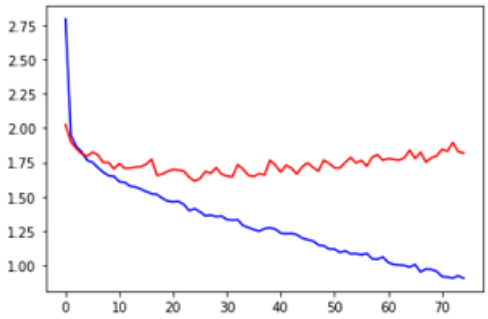


با این تابع acc_train تا 89 درصد میرسد

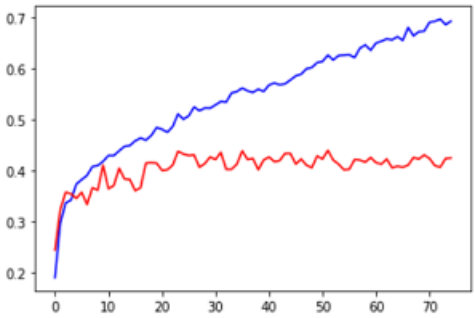
تابع tanh:

÷

نمودار loss:



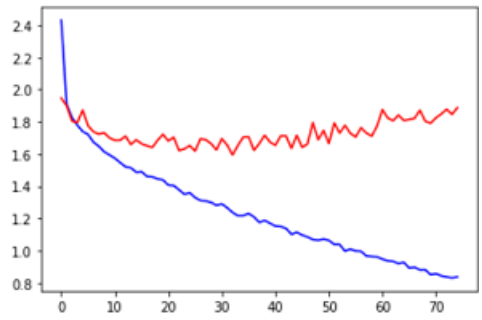
نمودار acc:



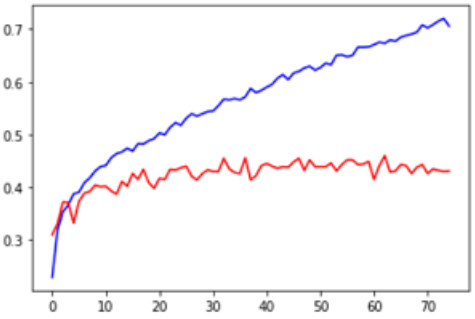
با این تابع acc_train تا 69 درصد میرسد

تابع relu:

نمودار loss:



نمودار acc:



با این تابع acc_train تا 72 درصد میرسد

به نظر میرسد که در این حالت تابع sigmoidبهترین عملکرد را در تسریع فرایند train انجام داده است.

÷

2: تعیین optimizer و learning rate (در فایل setting_optimizer_for_shallow_NN

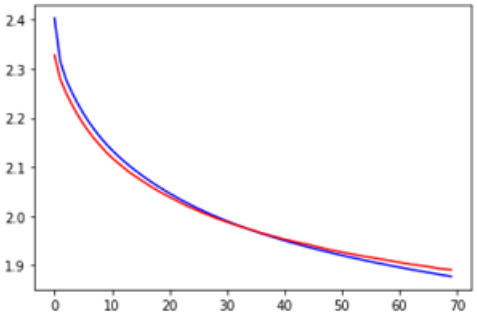
کد این بخش موجود هست)

:Adadelta

Learning rate=0.001

```
max acc of validation: 0.32899999618530273
max acc of trian: 0.3602222204208374
```

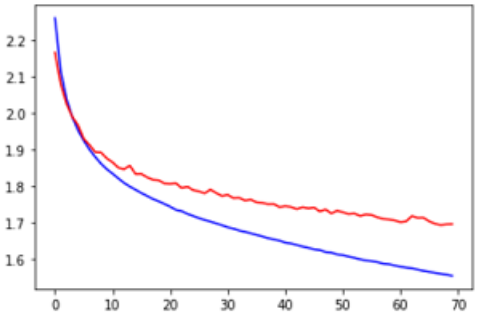
نمودار loss:



Learning rate=0.01

```
max acc of validation: 0.41200000047683716
max acc of trian: 0.4735555648803711
```

نمودار loss:

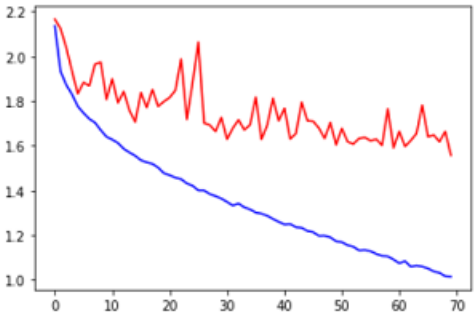


Learning rate=0.1

÷

```
max acc of validation: 0.46799999475479126
max acc of trian: 0.6667777895927429
```

نمودار loss:

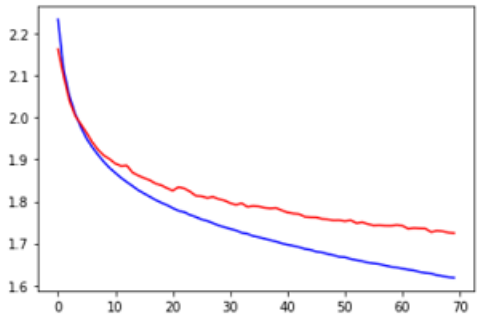


:Adagrad

Learning rate=0.001

```
max acc of validation: 0.4000000059604645
max acc of trian: 0.4502222239971161
```

نمودار loss:

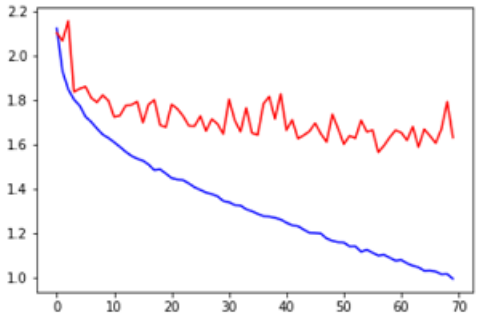


Learning rate=0.01

```
max acc of validation: 0.4690000116825104
max acc of trian: 0.6752222180366516
```

نمودار loss:

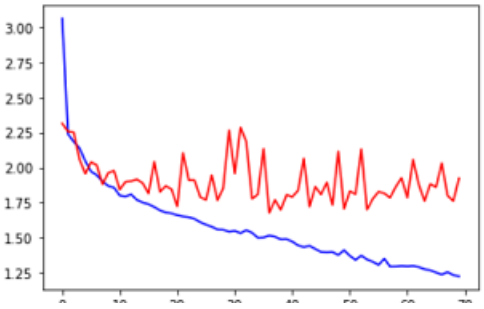
÷



Learning rate=0.1

```
max acc of validation: 0.4230000078678131
max acc of trian: 0.5619999766349792
```

نمودار loss:



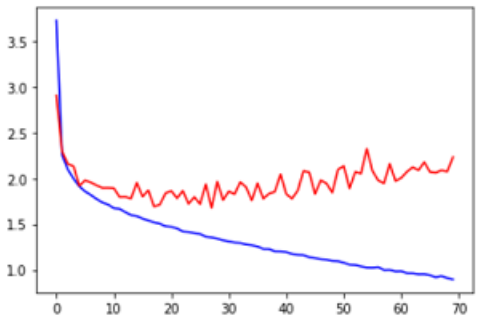
RMSAprop:

Learning rate=0.001

```
max acc of validation: 0.4309999942779541
max acc of trian: 0.6887778043746948
```

نمودار loss:

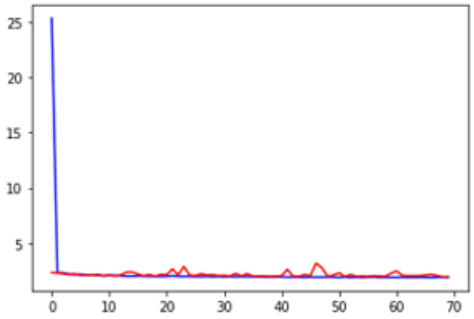
÷



Learning rate=0.01

```
max acc of validation: 0.2809999883174896
max acc of trian: 0.2835555672645569
```

نمودار loss:

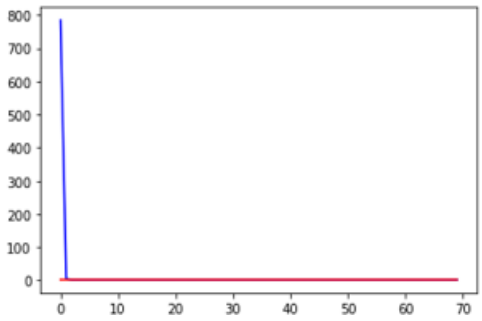


Learning rate=0.1

```
max acc of validation: 0.11500000208616257
max acc of trian: 0.10711111128330231
```

نمودار loss:

÷

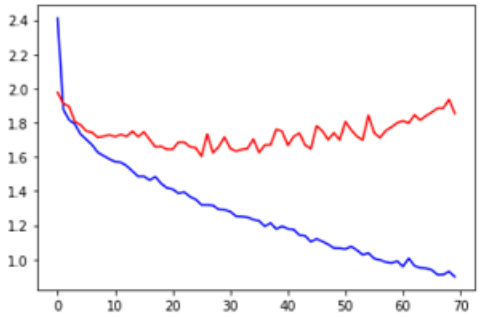


:Adam

Learning rate=0.001:

```
max acc of validation: 0.4560000002384186
max acc of trian: 0.6844444274902344
```

:loss نمودار

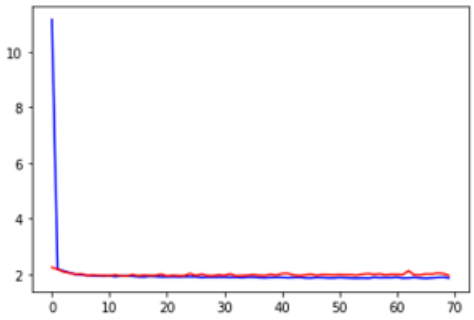


Learning rate=0.01:

```
max acc of validation: 0.2529999911785126
max acc of trian: 0.2777777910232544
```

:loss نمودار

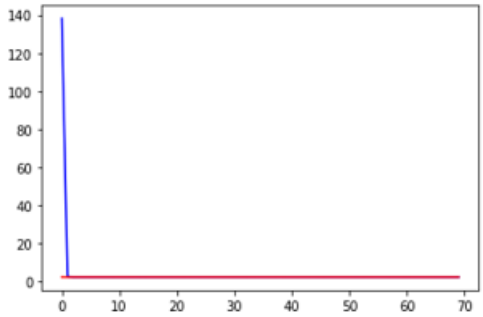
÷



Learning rate=0.1:

```
max acc of validation: 0.11500000208616257
max acc of trian: 0.10499999672174454
```

نمودار loss:



همانطور که از نتایج مشخص است الگوریتم adam با learning rate=0.001 وrmsprop هم با همین نرخ یادگیر بهترین انتخاب ها هستند البته با توجه به محتوا الگوریتم adamو پیچیده بودن loss function شبکه های چند لایه ه بهترین گزینه همین optimizerاست و از این استفاده خواهیم کرد.

4:بررسی معماری deep:(فایل مربوط به این قسمت:setting_number_of_layers)

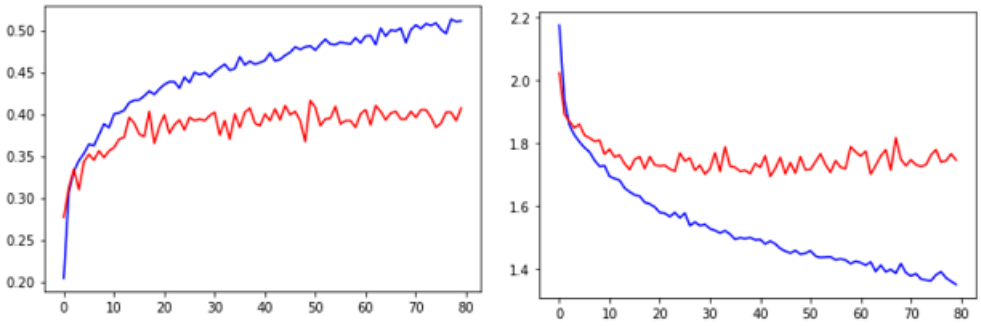
در این حالت با توجه به تجربه و تحقیقات acc_val در حدود 50 در صدر است و حتی با استفاده از یک bacth از داده ها در حدود 45 درصد است پس برای تعیین معماری مناسب اینکه کدام مدل نورون کمتری دارد (چون که با هزینه زمان و.... کمتری میتوان به مدل مطلوب رسید)و اینکه

÷

acc_train مطلوبی داشته معیار است و در مورد acc_val اینکه به حدود 45 درصد برسد کافی است

حالت1: 10-50-50:

Loss: acc:



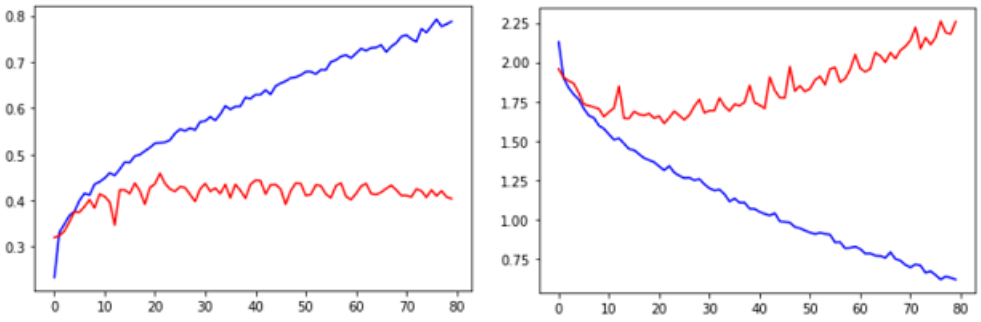
```
max acc of validation: 0.41600000858306885
max acc of trian: 0.5130000114440918
```

یک مدل بسیار سبک داریم و مشاهده میشه که acc_val به نوعی ثابت شده است

و تعداد epochبیشتر کمکی به generalization مدل نخواد کرد پس برای همین مدل را پیچیده تر میکنیم تا با همین تعدادepoch بتوان ظرفیت مدل را افزایش داد

حالت2: 10-100-200

Loss: acc:



```
max acc of validation: 0.45899999141693115
max acc of trian: 0.7926666736602783
```

در این حالت trainمدل خیل بهتر انجام شده است ولی تعداد نورون بسیار بیشتر از حالت قبل است

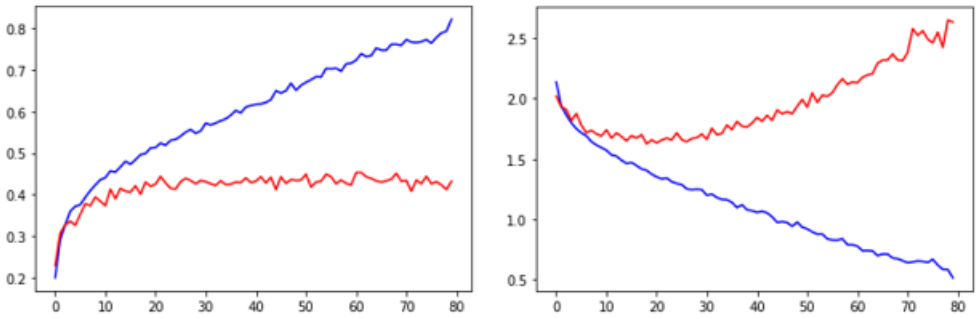
÷

برای همین سعی کرده به مدلی با تعداد لایه بیشتر ولی نورون کمتر برسیم که به خوبی train شود

بعد از تست حالت های مختلف به این نتیجه رسیده که با کم در نظر گرفتن تعداد نورون ها و افزایش تعداد لایه به نتیجه خیلی بهتری نسبت به حالت دو لایه نرسیده و حتی مدل سنگین تر شده و از طرفی هم کارایی مدل خیلی زیاد نمی شود

برای مثال از حالت 10-50-50-100-200 استفاده کرده و نتایج به این صورت است:

:Loss :acc

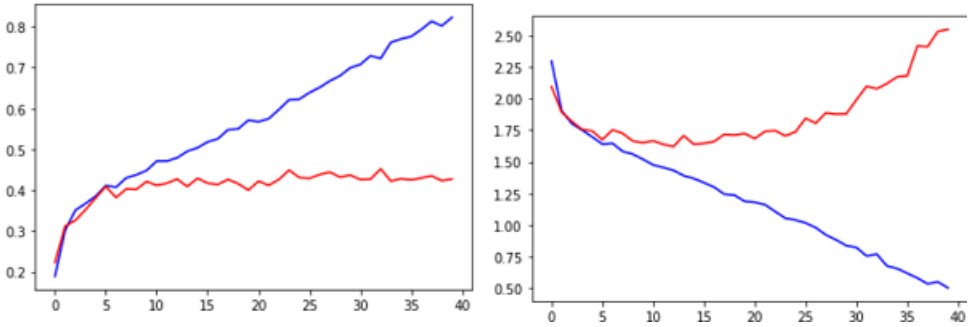


```
max acc of validation: 0.453000009059906
max acc of trian: 0.8226666450500488
```

که مشاهده شده بهبود خیلی خاصی در مدل حاصل نشده است که میتوان این مقدار را با تعداد بیشتری epochجبران کرد ولی مشکل که در تعداد epoch بیشتر وجود دارد این است که با افزایش تعداد epoch acc_val کاهش وloss_val افزایش می یابد پس باید پیچیدگی مدل را بیشتر کرد تا با تعداد کمتری epoch بتواند به مقدار acc_train بیشتری برسد

حالت:10-500-500-1500:

که این با 40 epochمدل به نتایج خوبی میرسد:

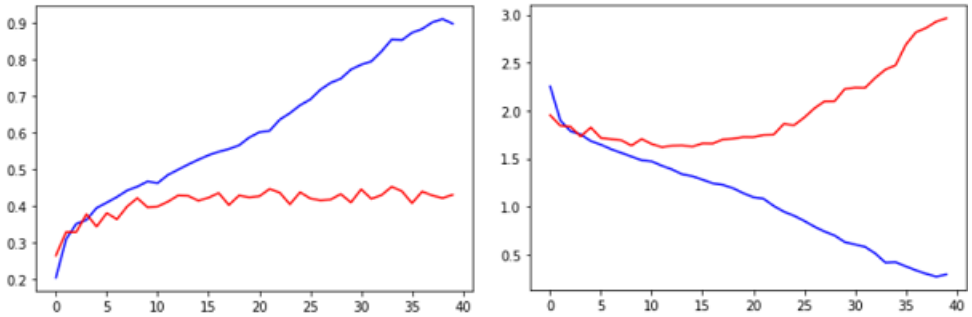


÷

```
max acc of validation: 0.4519999921321869
max acc of trian: 0.8223333358764648
```

بعد از امتحان حالات بسیار مدل 10-500-1000-1000 انتخاب میشود که نتایج آن به شرح زیر است:

:Loss :acc



```
max acc of validation: 0.4519999921321869
max acc of trian: 0.9108889102935791
```

که این مدل به نسبت سایر مدل با سرعت بهتری trainشده است

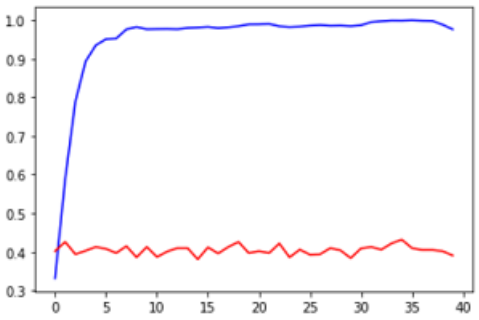
در این مدل پس از بررسی به این نتیجه رسیده که تعداد 3 لایه پنهان از همه مناسب تر است و بهتر است که لایه اول و دوم تعداد نورون زیادی در رنج 1000 تا داشته باشند و تعداد کمتری در لایه سوم نیاز است
درضمن افزایش تعداد لایه ها از این پس کمک خاصی به مدل نخواهد کردو کاهش آن سرعت آموزش را کند می کند و برای همین مدل برای آموزش بیشتر epoch بیشتر لازم دارد و در نتیجه مدل بیشتر به سمت overfitمی رود.

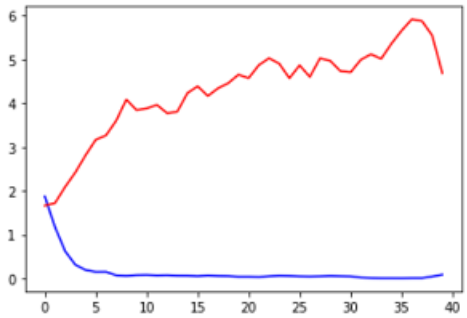
حالا که مدل تعین شده سعی کرده با تکنیک های preprocess های مدل را بهبود ببخشیم:

1:استفاده از PCA whitening:
(فایل مربوط به قسمت :
(preprocessing_data_for_ increase_acc_val

÷

:Loss :acc





```
max acc of validation: 0.4320000112056732
max acc of trian: 0.9994444251060486
```

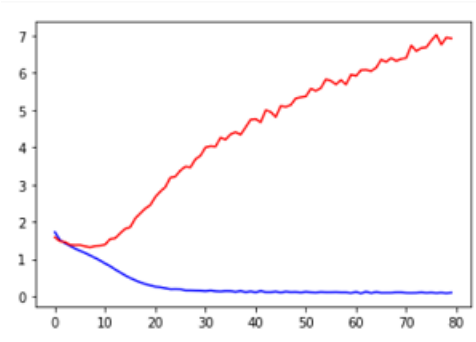
که مشاهده شده سرعت trainخیلی خیلی افزایش یافته است ولی تغییر خاصی در acc_valرخ نداده و با این تعداد epochمدل over fit شده که البته این مدل توانسته حداقل val _acc یعنی 43(نزدی ک 45)را تامین کند

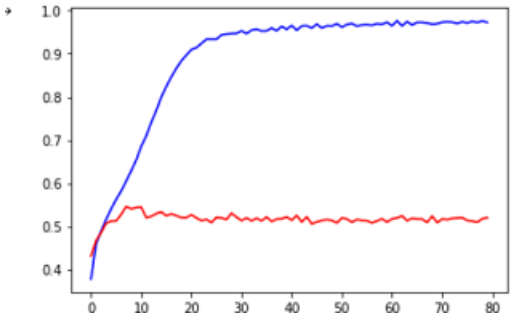
در یک مرحله دیگر برای از grayکرد و resize to 16x16 استفاده کرده که resizeکردن کمی در میزان acc_val تاثیر می گذارد (که این مراحل در فایل preprocessing_data_resizeing_dataTo_smaller_image موجود است)

(تمامی فرایند زیر در فایل final_model.ipynb موجود است):

در نهایت با استفاده از مدل 1000-1000-500- 10 استفاده میکنیم و برای preprocess از تقسیم بر 255 و resize to 16x16 استفاده می کنیم همینطور با flip کردن عکس های ترین تعداد مجموعه آموزشی را دو برابر می کنیم که نتایج از قرار زیر است:

:Loss :acc

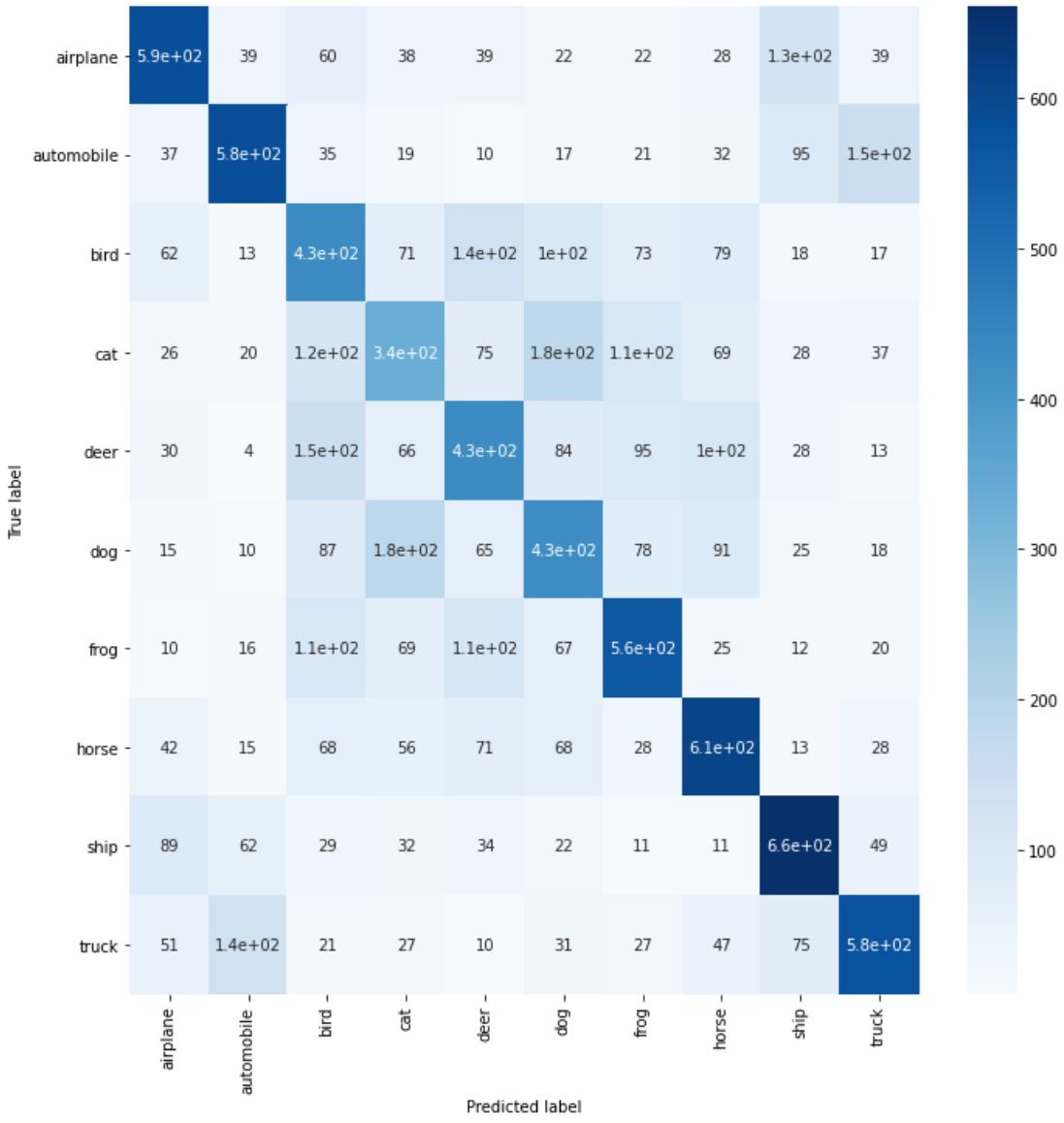




÷

همانطور که از نمودار ها معلوم است برای مدل over fit شده است ولی در نظر داشته با mlpمقدار acc_valحدود 53 درصد است پس برای اینکه مدل overfit نشود باید Acc_train حدود 60 تا 70 در صد باشد که در اینصورت با تعداد epoch کمتر میشود مدل را از overfit خارج کرد که طبق نمودار با تغییر epochبه حدود 15 تا می توان این حالت را برآورده کرد.
ولی از آنجایی که generalizeمدل قابل قبول نبود برای همین می گذاریم مدل overfit شود تا به روی داده های train نتیجه خوبی داشته باشد و از آنجایی هم که acc_val ثابت می شود چیزر را از دست نمی دهیم
پس طبق نمودار و مراحل قبل تعداد epoch مناسب برابر با 30 است
همنطور راجع bacth size عدد 128 طبق تجربه و تحقیقات عدد مناسبی است.
Max of acc_val=0.5457000136375427
Max of acc_train=0.9764400124549866
Confiusion matrix:

÷



Recalls:

airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0.61	0.65	0.38	0.37	0.43	0.41	0.54	0.55	0.61	0.60

Precisions:

airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0.58	0.58	0.42	0.33	0.43	0.42	0.55	0.61	0.66	0.57

f1:

airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0.60	0.61	0.40	0.35	0.43	0.42	0.55	0.58	0.63	0.59

÷

÷