Testing on 1/5 of data for finding best hyperparameters

Testing nodes numder :

1 Dense layer :

Activation : relu

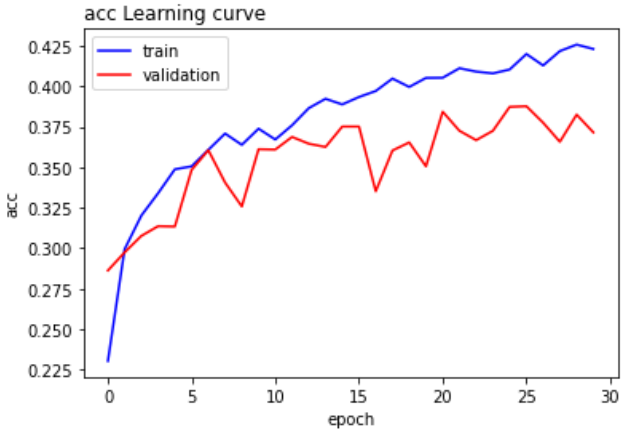
Output Activation : Softmax

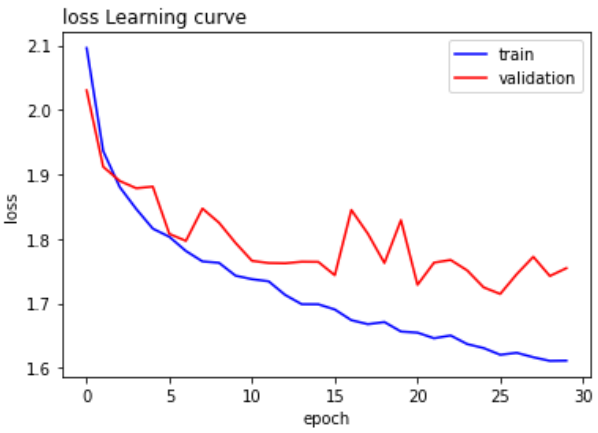
Bach size = 32

Epochs = 30

100 :

دقت train و validation هردو کمتر از 50% است(validation = 0.37 & train = 0.42) و نشان دهنده خوب نبودن بودن مدل است و همچنین با پیش رفتن و افزایش epoch ، فاصله بین خطوط train , validation هم در acc هم در loss افزایش میابد (مدل به سمت حفظ کردن داده های train میرود) پرش های زیاد و افزایش فاصله بین نمودار های train و validation نشان دهنده overfit بودن مدل است ولی از آنجایی که دقت خیلی پایین است، مدل بیشتر underfit حساب میشود.





300 :

دقت مدل با این تعداد نورون بهتر از قبل شده است ، **(train=0.52 , validation = 0.42)**اما هنوز هم دقت پایین است و با توجه به نمودار مدل overfitحساب میشود چون دقت هنوز هم پایین است میتوان بار گفت بیشتر underfit هست تا overfit . با افزایش epoch فاصله بین خطوط در نمودار acc ممکن است رو به کاهش برورد و مدل بهتری داشته باشیم.

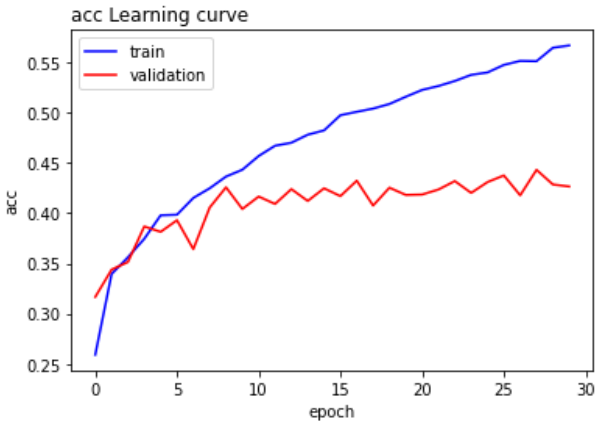
500 :

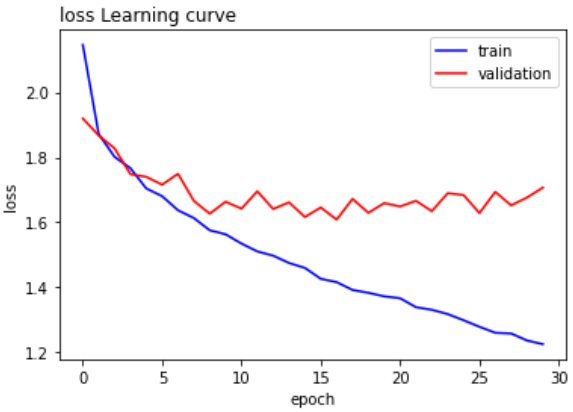
با افزایش تعداد نورون به 500 ، با اینکه دقت روی داده train افزایش یافته (train_acc =0.56)

اما دقت داده validation نسبت به نورون 300 تایی ، پیشرفت زیادی نداشته است(validation_acc = 0.42). مدل هنوز هم overfit محسوب میشود و همچنین

با افزایش epoch فاصله نمودار های trainو validation هم در نمودار acc و هم در lossدر حال افزایش است . درنتیجه مدل به سمت حفظ اطلاعات train میرود و با افزایش epochنتیجه بهتری نخواهد داشت .

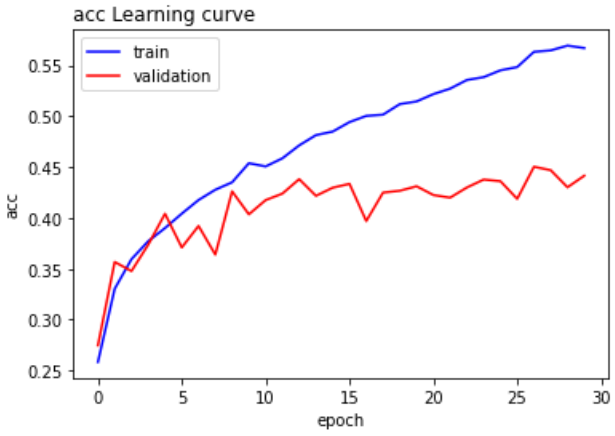
شکل نمودار بیانگر overfit بودن و دقت خیلی پایین بیانگر underfitبودن است .

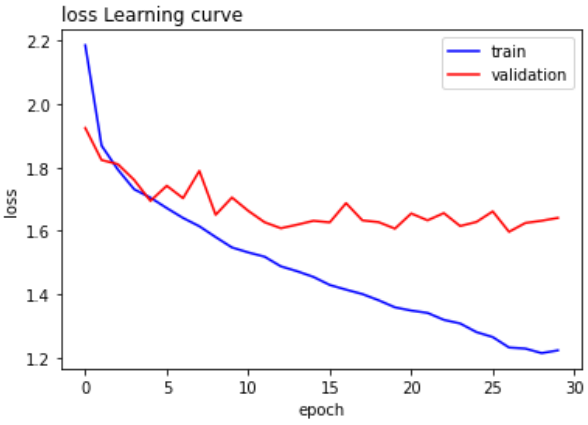




1000:

با این تعداد نورون ، برخلاف افزایش زیاد تعداد نورون ها ، مقدار دقت و loss تغییر زیادی ندارد و با توجه به نمودار ها ، درصورت افزایش epoch نموداری overfit خواهیم داشت ، و الان مدل underfit است .فاصله بین نمودار ها ، درمقایسه با نورون های قبلی بیشتر است .





Testing norourns numder Conclusion :

nodes	100	**300**	500	1000
Val_scor	0.37	**0.42**	0.42	0.46
Train_score	0.42	**0.52**	0.56	0.58

در قسمت تست نورون ها متوجه شدیم ، افزایش تعداد نورون نتیجه بهتری در پیش نخواهد داشت و با افزایش آن مدل overfit خواهد شد .تعداد نورون 300 به عنوان تعداد متوسط که نسبتا نتیجه بهتری داشت انتخاب میشود .

Testing epoch number :

Nodes = 300

Epoch = 32

Train_acc=0.52 , validation_acc = 0.42

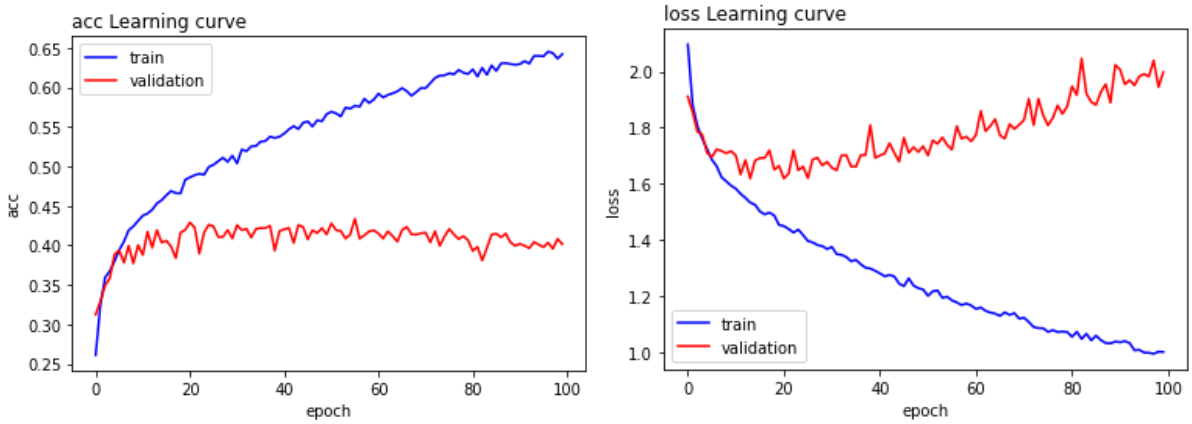
Epoch = 100:

Train_acc=0.64, validation_acc = 0.40

با افزایش به این تعداد epoch ، نتیجه طبق پیش بینی ، بهترنبود و مدل دچار overfit شد .

با توجه به این نمودار ، فاصله زیاد دو نمودار بیانگر مشکل overfit است و تقریبا از epoch ، 20 به بعد نتیجه بهتری حاصل نشده است .

*شکل نمودار شبیه نمودار های overfit است ولی دقت پایین بیانگر underfit بودن مدل است .

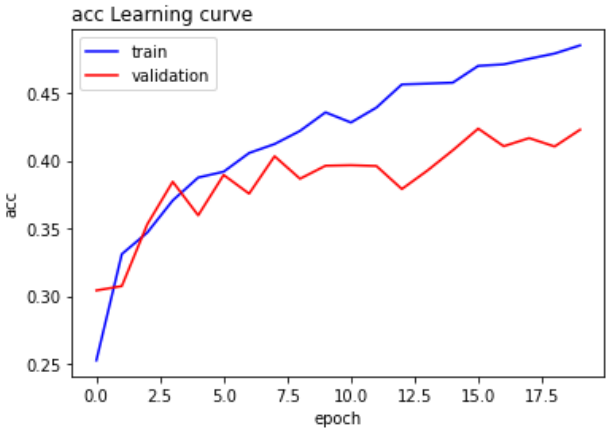


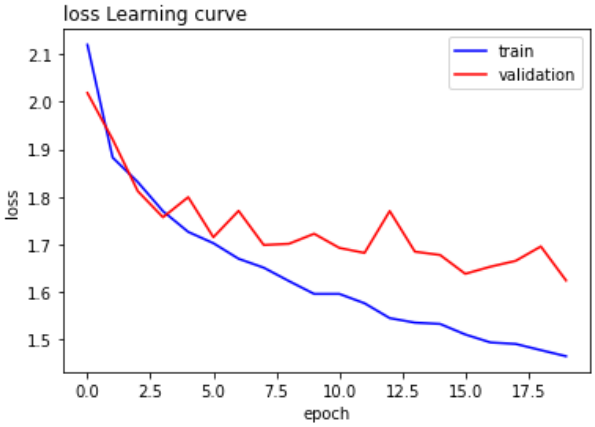
Testing epoch number conclusion :

Best epoch=20 nodes =300

Overfit with bad score

Train_acc=0.48, validation_acc = 0.42





Testing batch size :

epoch = 20 , unit = 300

batch size =32 ,

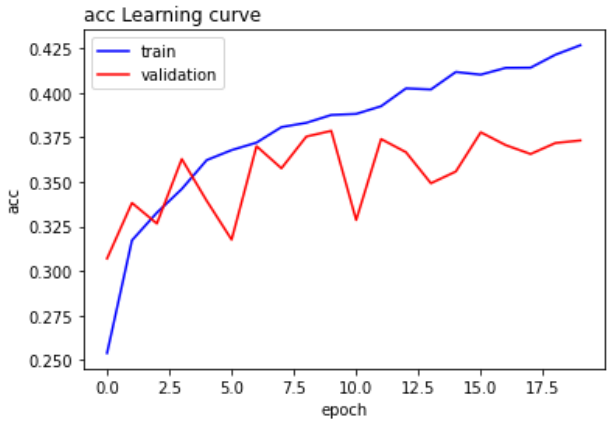
Train_acc=0.48, validation_acc = 0.42

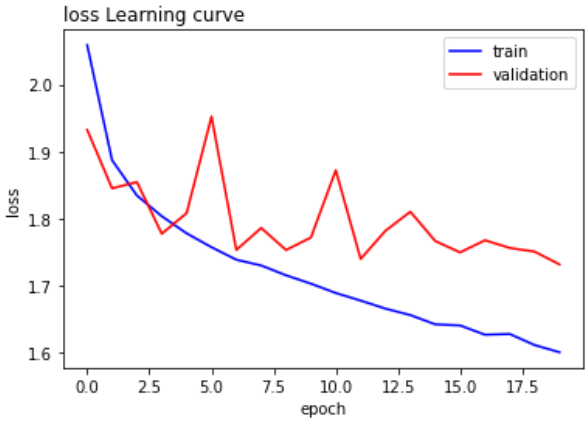
Batch size = 8 :

Train_acc=0.42, validation_acc = 0.37

دقت نسبت به قبل کاهش یافته و خطا افزایش و پرش در نمودار validationزیاد است .

*شکل نمودار شبیه نمودار های overfit است ولی دقت پایین بیانگر underfit بودن مدل است .

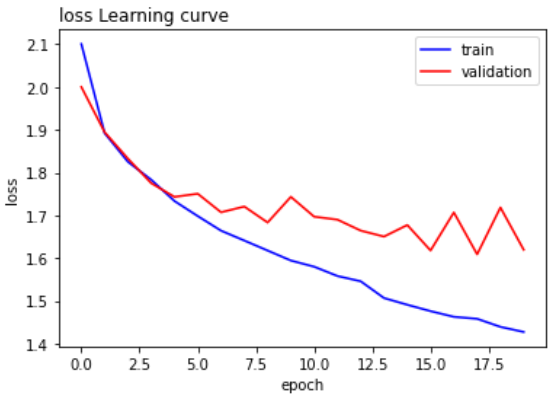


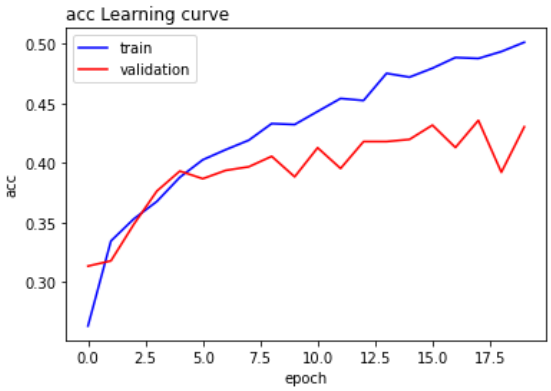


Batch size = 64:

Train_acc=0.50, validation_acc = 0.43

نسبت به بقیه موارد ، دقت با این Batch size بیشتر است و فاصله بین نمودار ها در حال کاهش است .

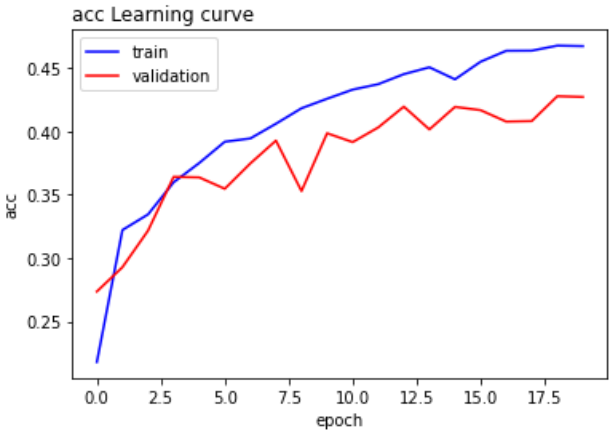


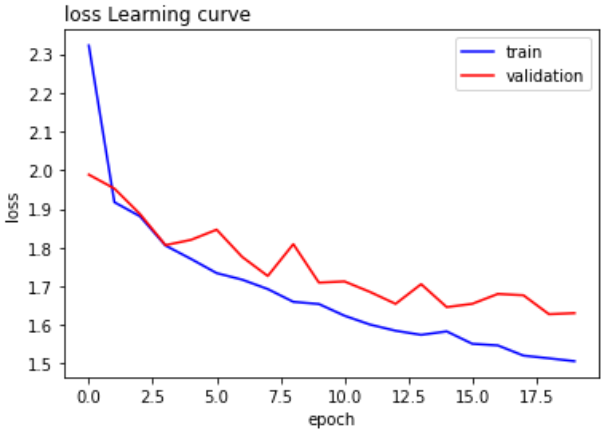


Batch size = 128:

Train_acc=0.46, validation_acc = 0.42

نمودار های خوبی داریم فاصله بین خطوط validation و train در هر دو پایین است ولی هنوز دقت پایین است .





Testing batch size conclusion :

هر سه عدد 128 , 32 , 64 نتایج نزیدکی داشتند ،

مدل در هر سه underfit حساب میشود ولی فاصله بین نمودار ها خوب است

با تست های سایر پارامتر ها بعدا میتوان ، مقدار مناسب تر را انتخاب کرد.

فعلال مقدار متوسط 64 را انتخاب میکنیم

Testing activation function :

– 20 epochs – 300 node - 64 batch size:

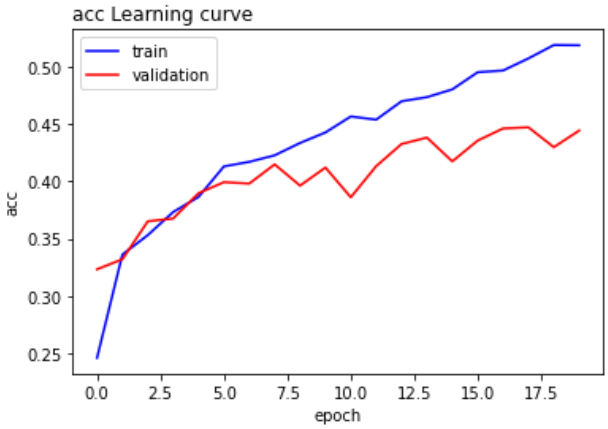
Activation function = sigmoid

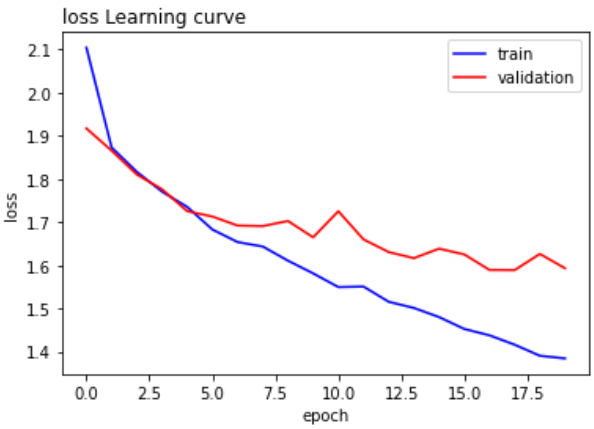
Train_acc=0.51, validation_acc = 0.4446

Train_loss =1.38, validation_loss =1.59

دقت بهتر و loss کمتر نسبت به بقیه موارد تا بکنون

پرش نسبت به نمودار های relu کمتر است .



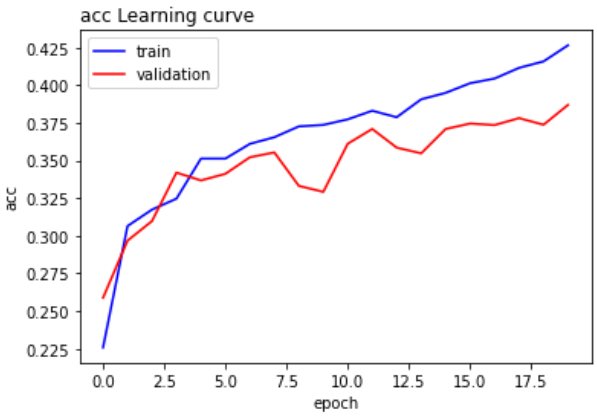


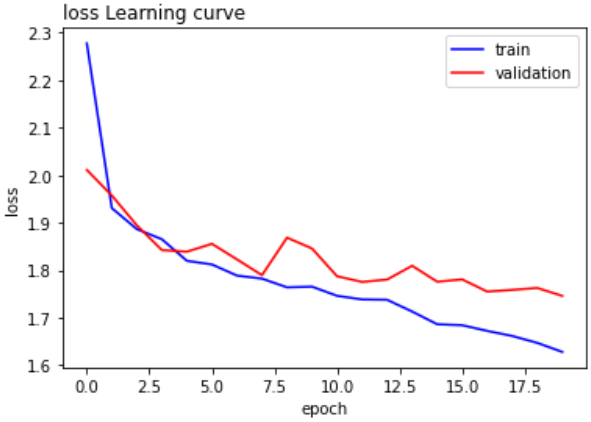
Activation function = tanh

Train_acc=042, validation_acc = 0.38

Train_loss =1.62, validation_loss =1.74

با اینگه نمودار خوبی داریم و فاصله بین خطوط کم است ، اما دقت نسبت به relu و tanh پایین تر است .





Testing activation function conclusion:

دو تابع relu وsigmoid نتایج نزدیکی دارند و بابررسی پارامتر های دیگر ، میشود از بین این دو انتخاب کرد .

Testing optimizer and learning rate :

300 node , epoch = 20 , batch size = 64 ,

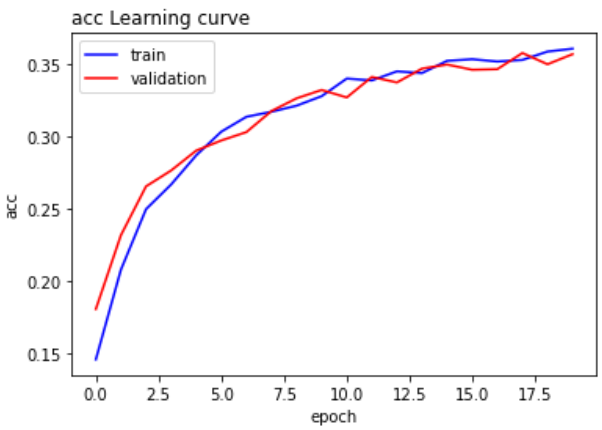
ADAM :

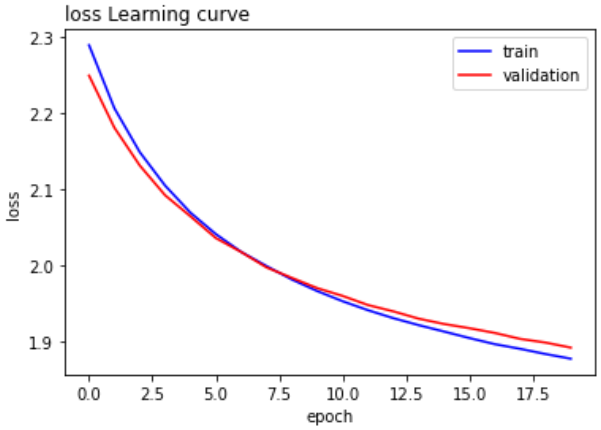
Learning_rate : 0.00001

Train_acc=0.36, validation_acc = 0.35

Train_loss =1.87, validation_loss =1.89

پرش های زیاد در نمودار از بین رفته ولی دقت بسیار پایین است .





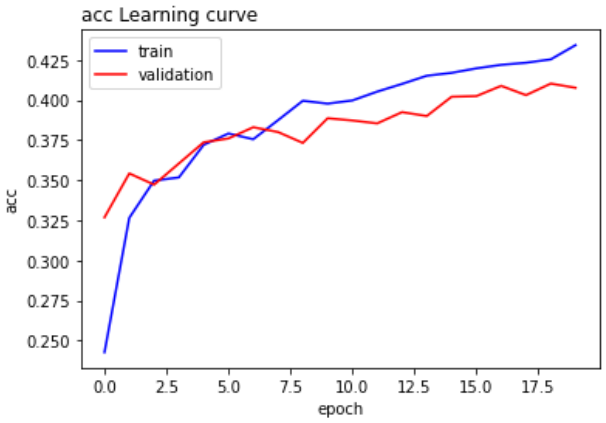
Learning_rate : 0.0001

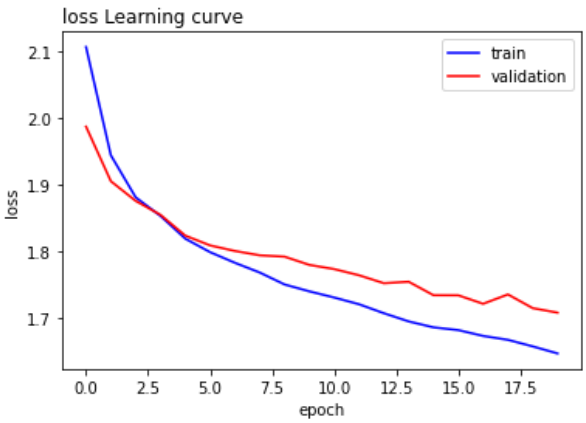
Train_acc=0.43, validation_acc = 0.40

Train_loss =1.64, validation_loss =1.70

دقت بهتر نسبت به learning rate = 0.0001 ، . پرش کمتر نسبت به مقدار دیفالت LR .

فاصله نمودار ها خوب است و همچنین رووبه کاهش. ولی مدل underfit حساب میشود .





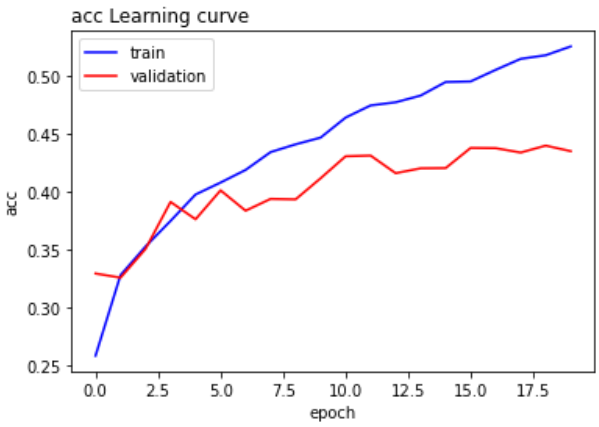
Learning_rate : 0.001 (default)

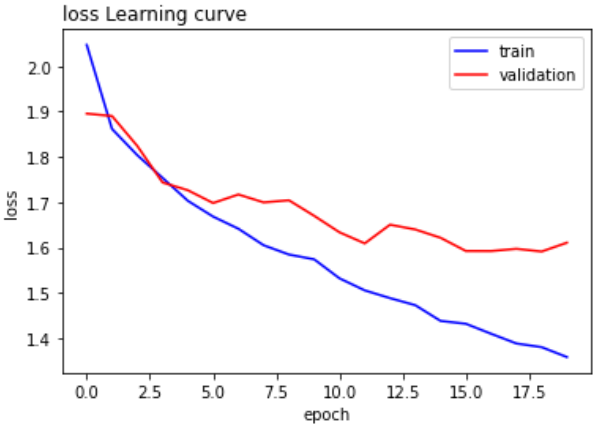
Train_acc=0.52, validation_acc = 0.43

Train_loss =1.35, validation_loss =1.61

با اینکه دقت بهتر و loss کمتر است ، اما با توجه به نمودار مدل به سمت overfit است .

افزایش learning rate ، نتیجه خوبی نخواهد داشت .





optimizer	ADAM	**ADAM**	ADAM	ADAM	**ADAM**	**ADAM**
LR	0.00001	**0.0001**	0.001	0.00001	**0.0001**	**0.001**
Activation function	SIGMOID	**SIGMOID**	SIGMOID	RELU	**RELU**	**RELU**
Val_acc	0.35	**0.40**	0.43	0.38	**0.44**	**0.43**

Train_acc	0.36	**0.43**	0.52	0.39	**0.51**	**0.48**
Val_loss	1.89	**1.70**	1.61	1.78	**1.6**	**1.61**
Train_loss	1.87	**1.64**	1.35	1.74	**1.4**	**1.43**

تابع RELU نتیجه بهتری داشته است ، با آن ادامه میدهیم .

SGD :

LR=0.01(default)

Train_acc=0.47, validation_acc = 0.41

Train_loss =1.51, validation_loss =1.66

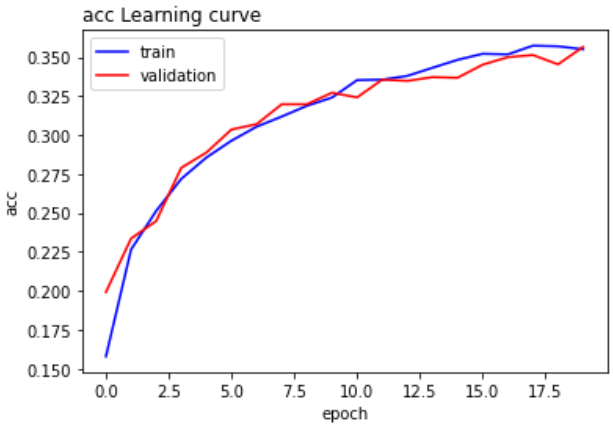
پرش نمودار بسیار زیاد است و دقت کافی نیست .

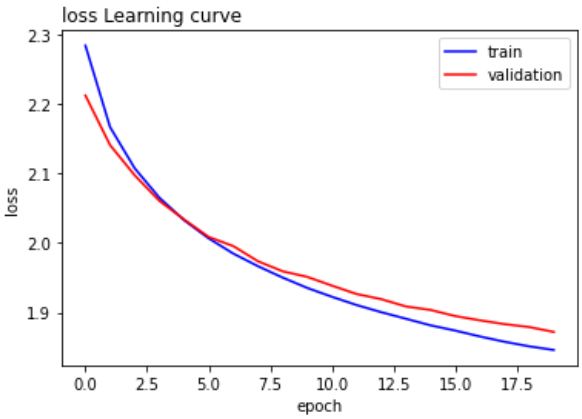
LR:0.001:

Train_acc=0.3551, validation_acc = 0.3566

Train_loss =1.84, validation_loss =1.87

نمودار های خوب ، بدون پرش با فاصله کم داریم اما دقت پایین است .



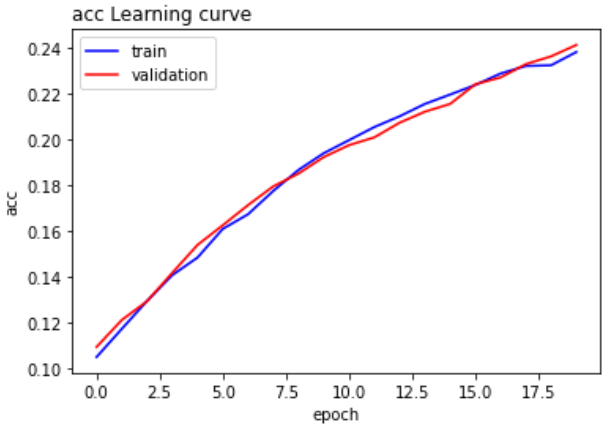


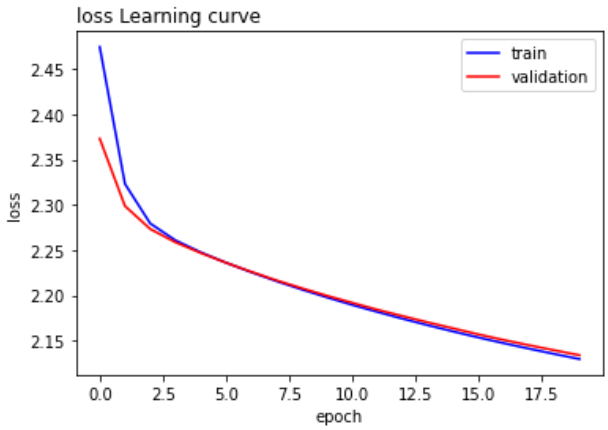
LR=0.0001

Train_acc=0.23, validation_acc = 0.24

Train_loss =2.12, validation_loss =2.13

مدل به شدت underfitاست . درنتیجه کاهش LR دیگر نتیجه مثبتی نخواهد داشت.





optimizer	SGD	SGD	SGD
LR	0.001	**0.0001**	**0.01**
Activation function	RELU	**RELU**	**RELU**
Val_acc	0.35	**0.24**	**0.41**
Train_acc	0.35	**0.23**	**0.47**
Val_loss	1.87	**2.13**	**1.66**

Train_loss	1.84	**2.12**	**1.51**

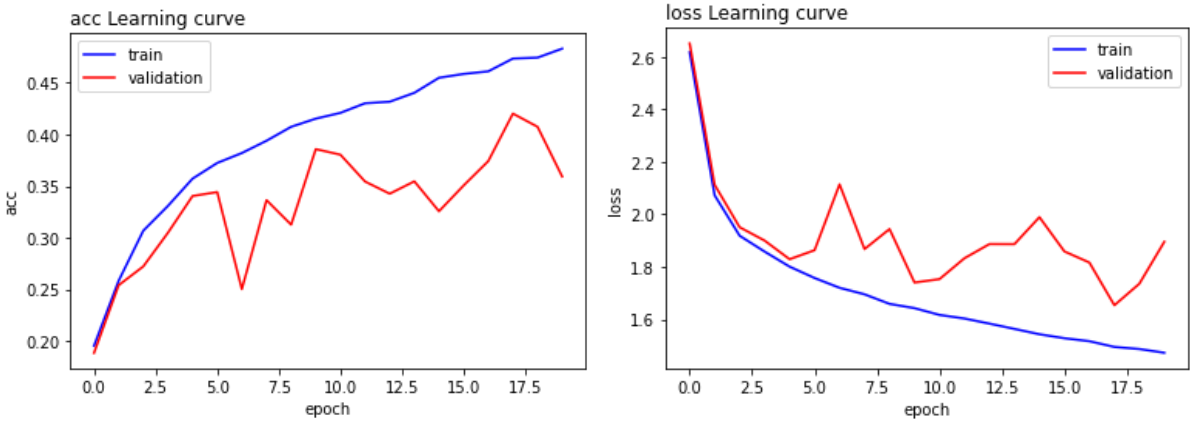
RMSprop:

LR:0.001

Train_acc=0.48, validation_acc = 0.35

Train_loss =1.47, validation_loss =.189

نه دقت خوبی داریم ، نه نمودار های مطلوبی ، پرش و فاصله زیاد است .



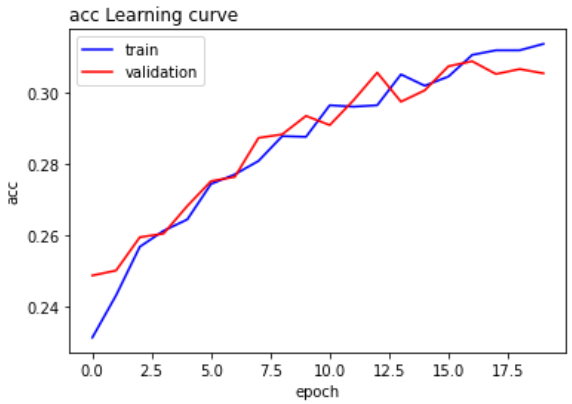
LR:0.0001

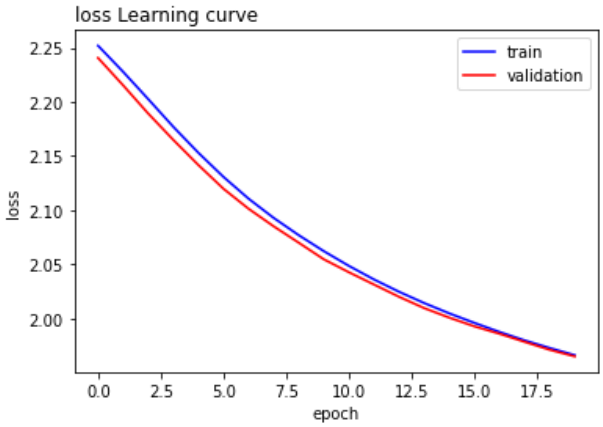
Train_acc=0.31, validation_acc = 0.30

Train_loss =1.96, validation_loss =1.94

نمودار های خوبی در نتیجه داریم ، اما دقت بسیار پایین است .

(با این آپتیمایزر ، 200 ، ایپاک تست شد و همچنان نمودار خوبی داشتیم ، و دقت validation به 0.40 رسید ، هنوز مدل underfit هست ولی ممکنه با افزایش داده ، دقت بهتری بگیریم)





.

Testing optimizer and learning rate :

بهترین نتایج :

Adam : Learning_rate : 0.0001

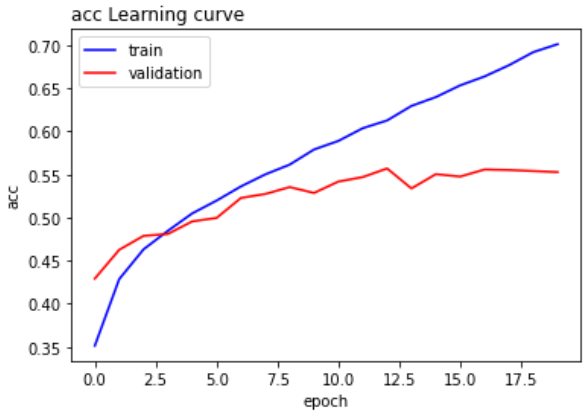
SGD: LR =0.001:

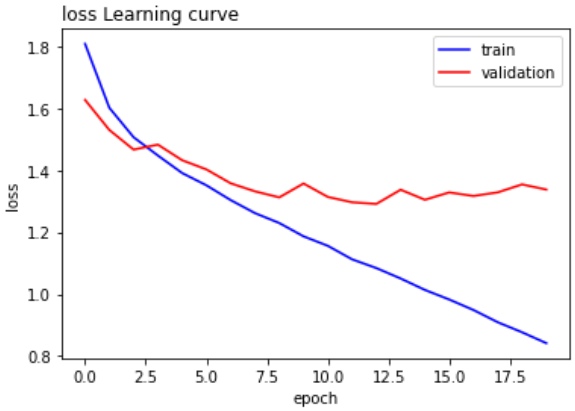
RMSprop: LR:0.0001

Working with actual data size :

1)4 Dense layer : 200 - 300 – 500 – 300 – 200

activation = relu – optimizer =adam(LR=0.001) – 20 EPOCH – 64 batch size





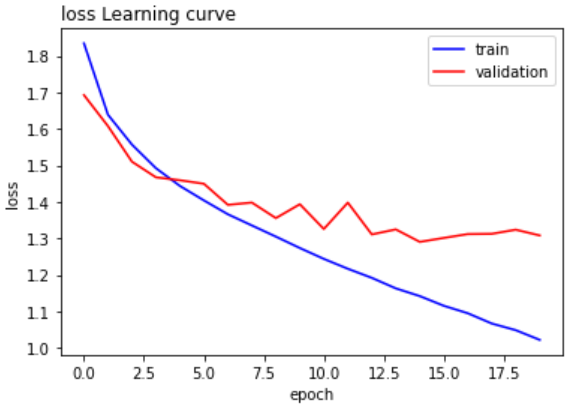
Test conclusion : loss =1.35 , acc =0.54

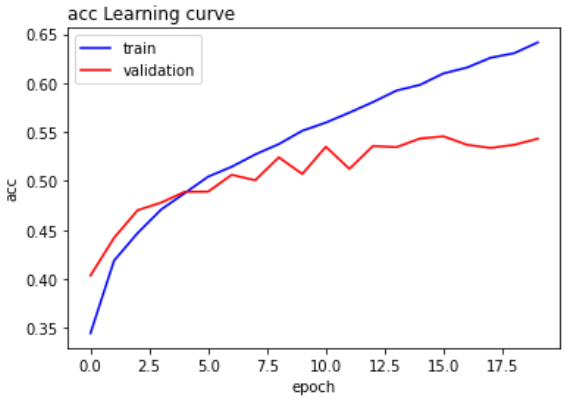
مدل نهایی : از این مدل به عنوان مدل نهایی استفاده میکنیم طبق نمودار مدل overfit هست و همچنین acc بالا نیست .

برای کاهش overfit ، تعداد لایه هارا کم میکنیم . :

Layers = 300 – 500 – 300

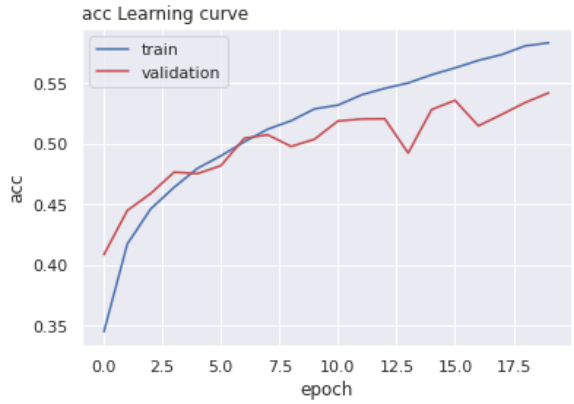
هنوز هم overfit حساب میشود

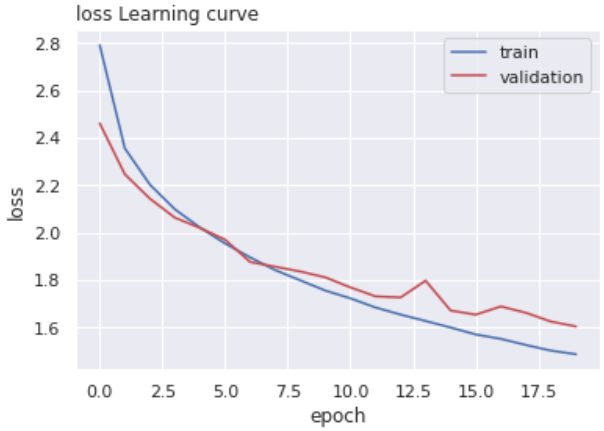




برای از بین رفتن این پرش ها و فاصله بین دو نمدار که بیانگر overfit است ، یک regularization ، اضافه میکنیم :

```
kernel_regularizer=l2(0.001)
```





با کاهش ، بیشتر LR این مقدار پرش هم از بین میرود ، ولی دقت بسیار کاهش میابد ، درنتیجه همین مدل را خواهیم داشت .

Layers = 300 – 500 – 300

Activation : relu

Output layer activation : softmax

Optimizer: adam(LR=0.0001)

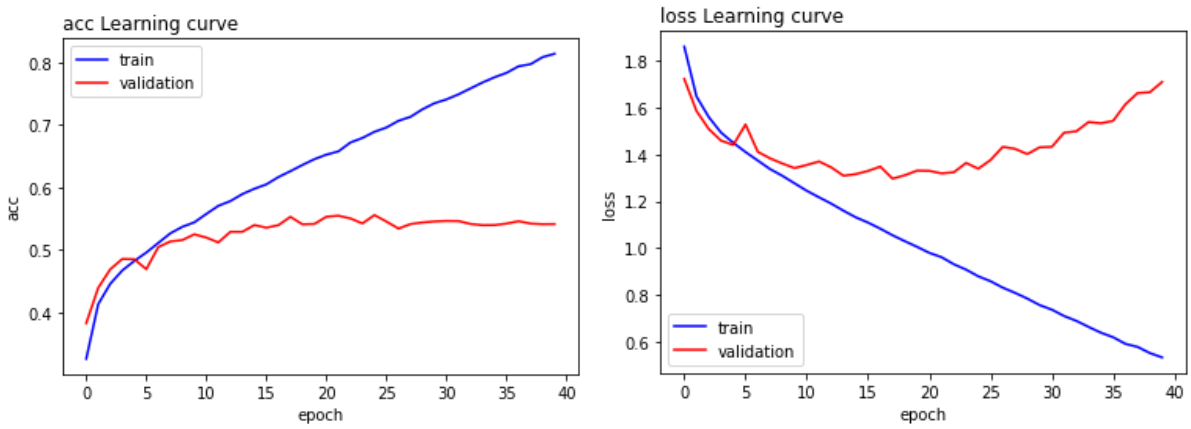
Test conclusion : loss =1.62 , acc =0.52

Testing some other models

2)5 Dense layer :200-300-500-300-200

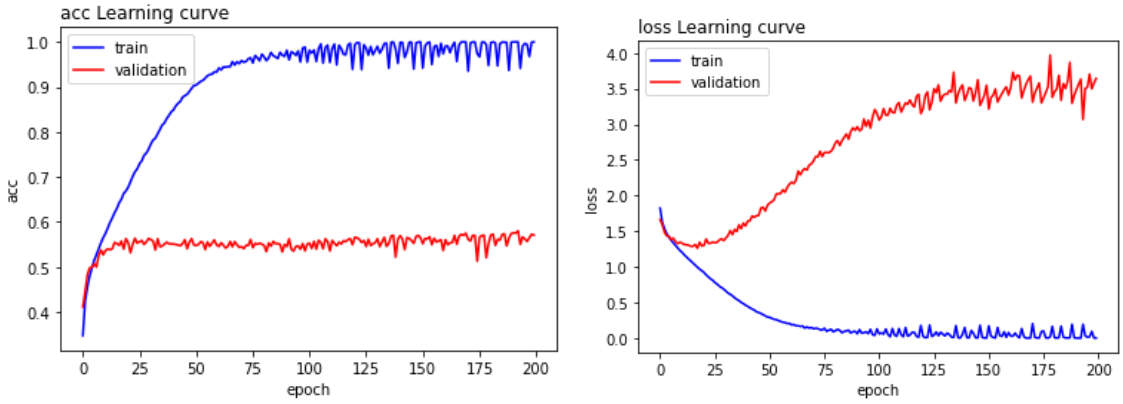
– activation = relu – optimizer =adam(LR=0.001) – 20 EPOCH – 64 batch size

افزایش epochو لایه تاثیر مثبتی نداشته



Test conclusion : loss =1.73, acc =0.51

3)3 Dense layer : 1024 , 512 , 512 – activation = relu optimaizer = adam(LR=0.0001) – 200 epoch – 128 bacth size

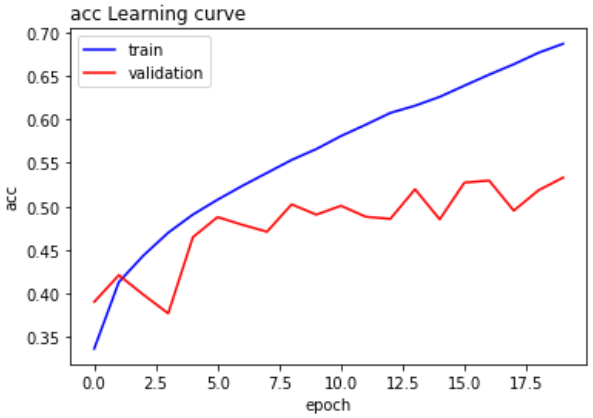


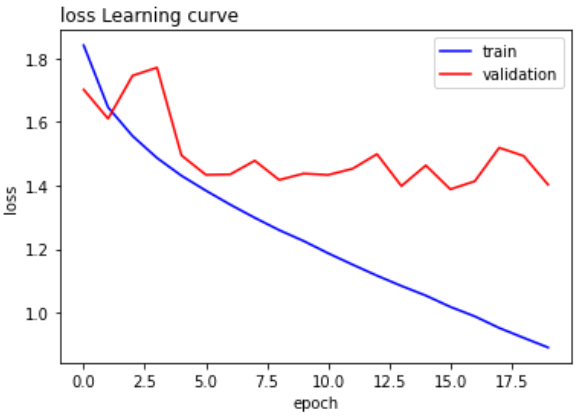
Test conclusion : loss =3.75 , acc =0.55

این یک مدل موجود در اینترنت هست ، مدل overfit هست و مشخص هست از epoch ، 20 به بعد تغییر مثبتی روی داده validation اتفاق نمی افتد .

4) 4 Dense layer : 300 700 700 300 – activation = relu – optimizer =RMSprop(LR=0.0001) – 20 EPOCH – 64 batch size

Test conclusion : loss =1.40 , acc =0.53





در این مدل هم از 4 لایه استفاده میشود ، و از optimizer RMSprop که احتمال میرفت خوب باشد ، ولی مدل خوبی نبود .

Testing KFOLD :

در مدل از یک KFOLD ، با K = 4 استفاده شد ، دقت نهایی ACC = 0.52 و LOSS = 0.62

هست که دقیقا نزدیک مقداری است که ما با TRAIN TEST SPLIT ساده به دست آوردیم ، با این تفاوت که تایم TRAIN شدن KFOLD خیلیییی بیشتر بود و واقعا به صرفه نیست از آن در این دیتاست با این مقدار استفاده کنیم.

مقایسه مدل با مدل های موجود :

مدل هایی که همه تنها از mlp ساده استفاده کرده اند ، همه دقت زیر 60 درصد داشتند و مدل ها با دقت بالا از لایه های convolutionalاستفاده کرده اند(cnn)

و همچنین حتی با به کاربردن epoch بسیار زیاد ، در mlp دقت خیلی پایینتر از 60 گرفتند (حدود 37 درصد)

نتیجه این است ، برای به دست آوردن دقت بالا در mlp باید از تعداد node و epoch کمتر استفاده کنیم تا تازه دقت به 50 برصد .
و برای بهتر شدن باید از لایه های drop out و convolutional استفاده کنیم .