به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری دوم

نام و نام خانوادگی	میلاد محمدی
شماره دانشجویی	810100462
تاریخ ارسال گزارش	1401/1/18

فهرست گزارش سوالات

سوال 1 – MLP (Classification)

ابتدا با استفاده از قطعه کد ذکر شده در صورت سوال cifar10 را دانلود می‌کنیم. از 50000 تصویر اول برای آموزش و از 10000 تصویر آخر برای تست استفاده می‌شود. 10 تصویر این مجموعه در زیر نشان داده شده است :

شکل 1. 10 تصویر از دیتاست **cifar 10**

قسمت الف) از 60000 داده موجود، 50000 داده به عنوان داده آموزش (تقریبا 84 درصد) و 10000 داده به عنوان داده تست (16 درصد) و 10 درصد داده تست به عنوان داده ارزیابی در نظر گرفته شده است. تقسیم بندی داده‌ها با توجه به تعداد داده‌ها و ساختار و عملکرد شبکه انجام می‌شود. در تقسیم داده‌ها باید هر دسته با توجه به توزیع احتمال اصلی نمونه برداری شود تا بتواند نمونه خوبی از کل مجموعه باشد.

Validation Set Approach : تقسیم تصادفی مجموعه داده‌ها به یک نسبت خاص که معمولا 30/70 یا 20/80 است که درصد بیشتر برای داده آموزشی است. این مدل زمانی استفاده می‌شود که متغیر هدف یک متغیر categorical باشد.

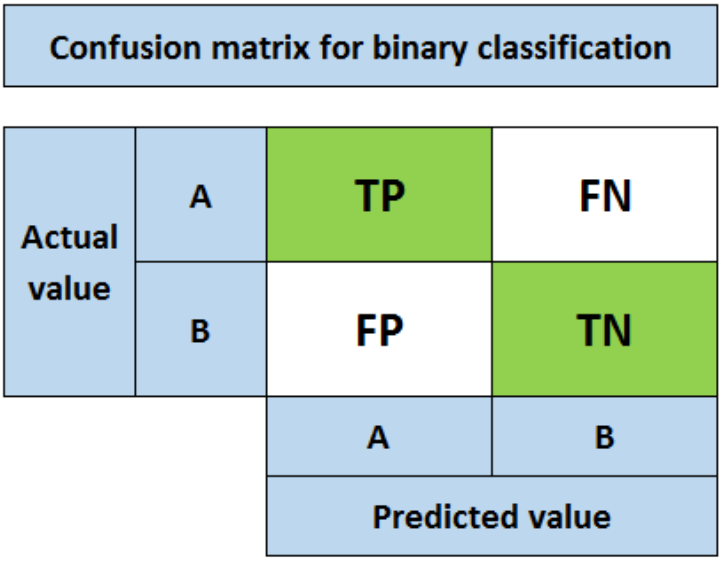
leave one out cross validation : هر نمونه یک بار به عنوان یک تست استفاده می شود، در حالی که نمونه‌های باقی مانده مجموعه آموزشی را تشکیل می دهند. این روش برای دیتاست‌های کوچک مناسب است.

k-folds : دیتاست را به k گروه تقسیم می‌کنیم. برای هر گروه این کار را انجام می‌دهیم : یک گروه را به عنوان داده تست و بقیه گروه‌ها را به عنوان داده آموزشی در نظر می‌گیریم. در این روش چندین مدل خواهیم داشت.

در این سوال ما از روش Validation Set Approach استفاده می‌کنیم. چون دیتاست ما بزرگ است و برای کم‌کردن هزینه محاسبات از این روش استفاده می‌کنیم.

قسمت ب)

ماتریس آشفتگی در مواقعی که دقت و صحت تشخیص صحیح یک دسته در در مقایسه با دقت و صحت تشخیص کلی، اهمیت بیشتری دارد مطرح می‌شود. اگر 4 حالت FP,FN,TP,TN را در نظر بگیریم ماتریس آشفتگی زیر را خواهیم داشت :



شکل 2. ماتریس آشفتگی

recall : درصد پاسخ‌هایی است که درست پیش بینی شده‌اند و درست واقعی است که با رابطه ریاضی روبرو محاسبه می‌شود : TP/ TP + FN

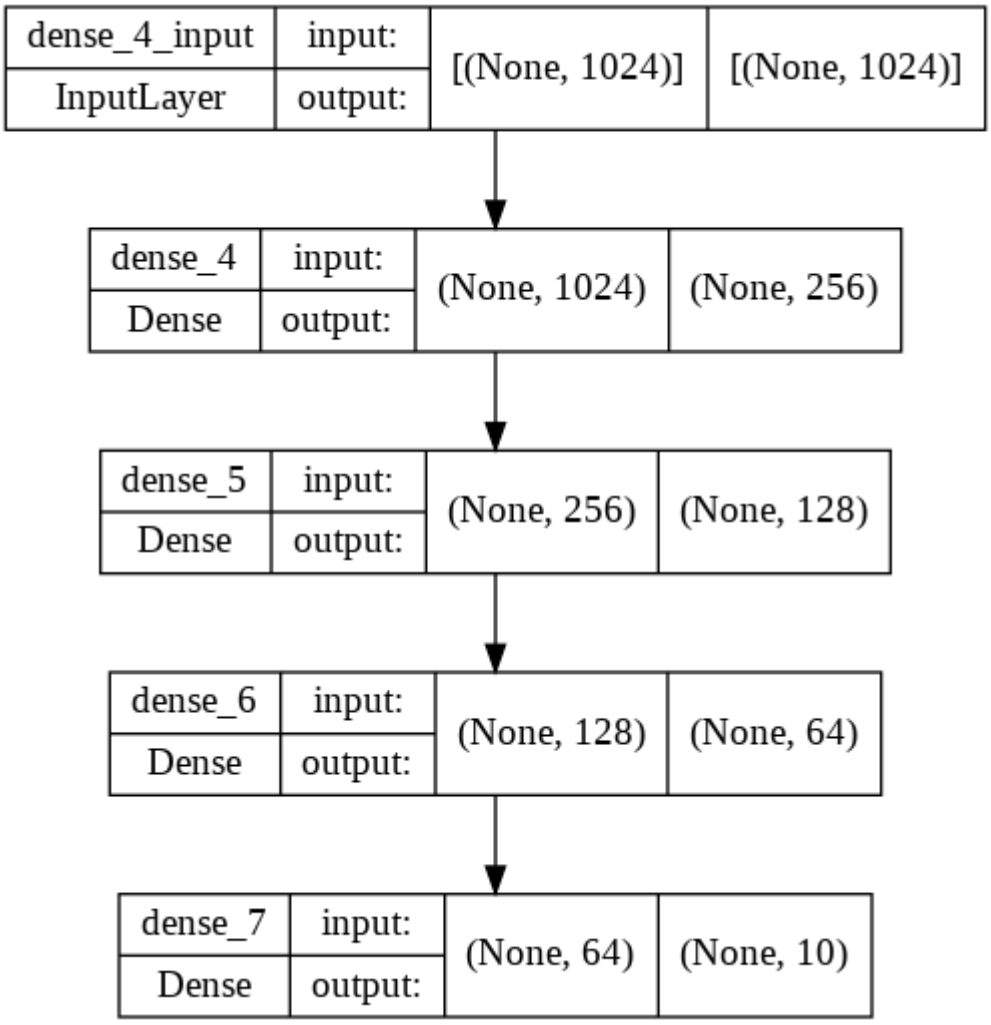
Precision : درصد پاسخ‌هایی است که درست پیش‌بینی شده‌اند و درست یا غلط بودن مشخص نیست و با رابطه ریاضی روبرو محاسبه می‌شود : TP / (TP+FP)

f1-score : برای ارزیابی عملکرد دسته‌بندی استفاده می‌شود و میانگین هارمونیک دو متغیر قبلی است:

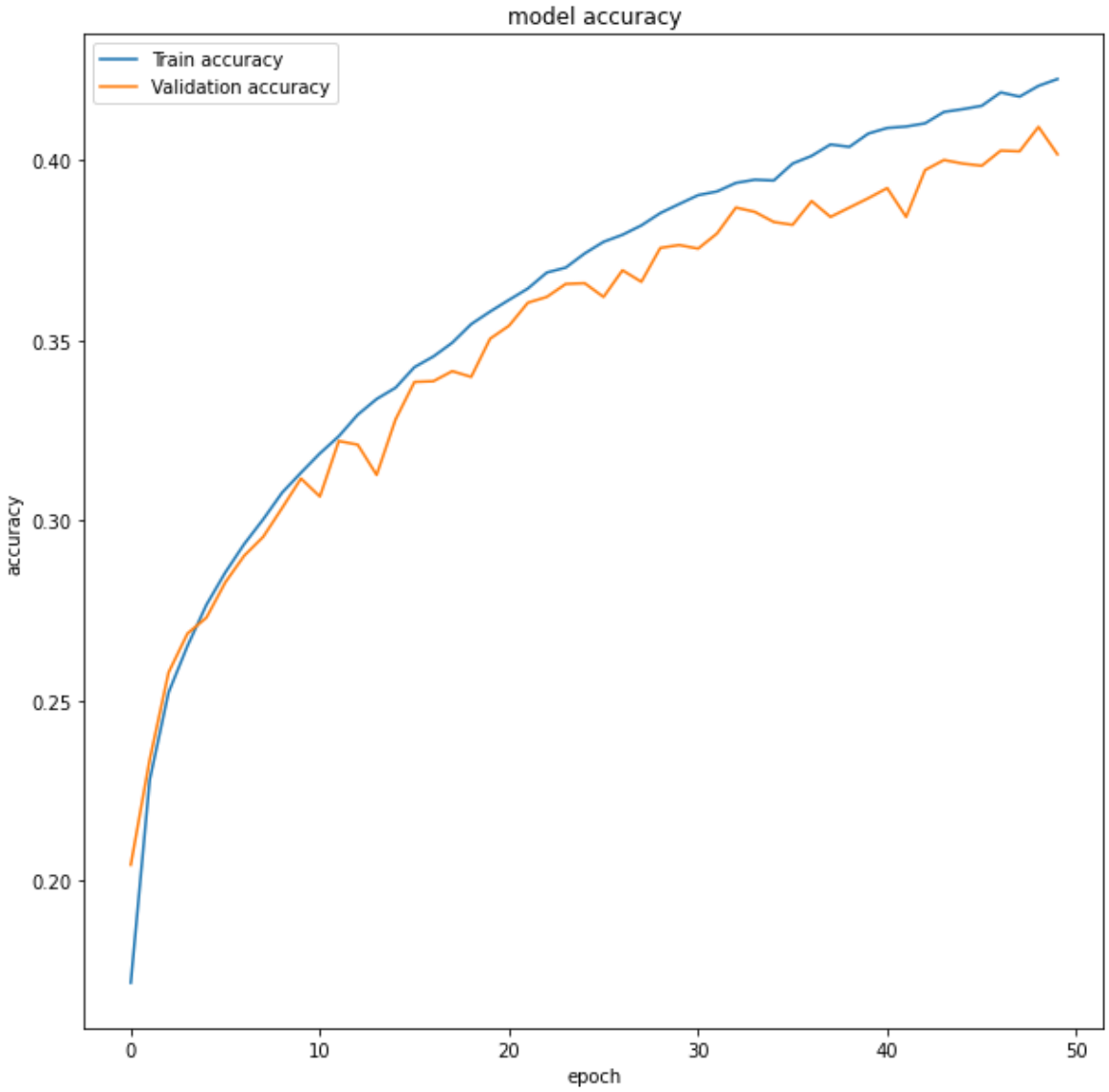
F-measure= 2 * (Recall * Precision) / (Recall + Precision)

قسمت ت) learning rate = 0.001 , batch size = 32, epochs = 50

معماری شبکه، نمودار دقت و تغییرات خطا، خطا، دقت، زمان، ماتریس آشفتگی، مقدار f1-score، recall و precision برای نورون‌های با تعداد 256، 128، 64 و 10 :



شکل 3. معماری شبکه با نورون‌های 256، 128، 64 و 10



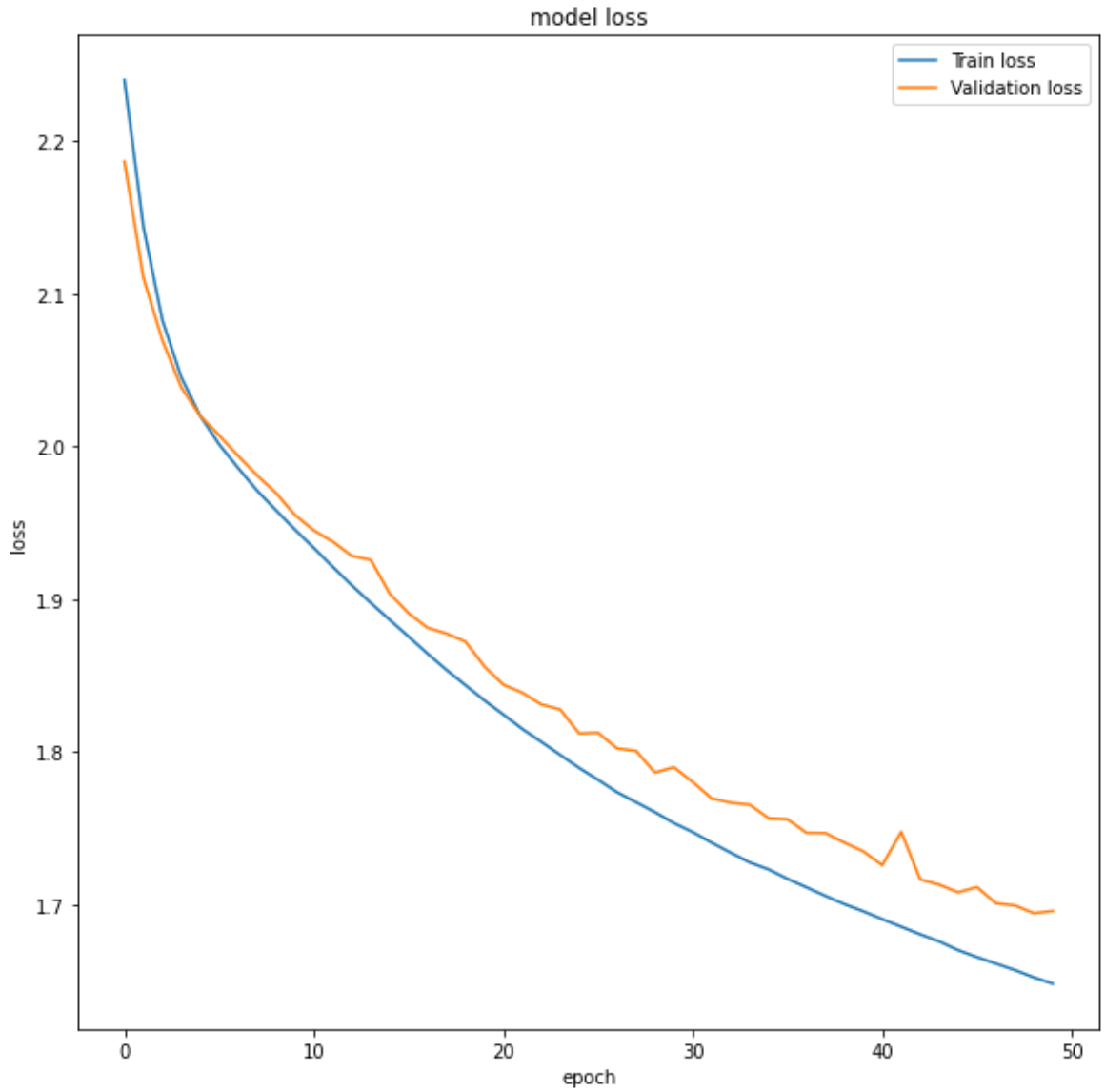
شکل 4. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision :

recall is : 0.3971

precision is : 0.400682519972351

f1 is : 0.3955143451115557



شکل 5. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

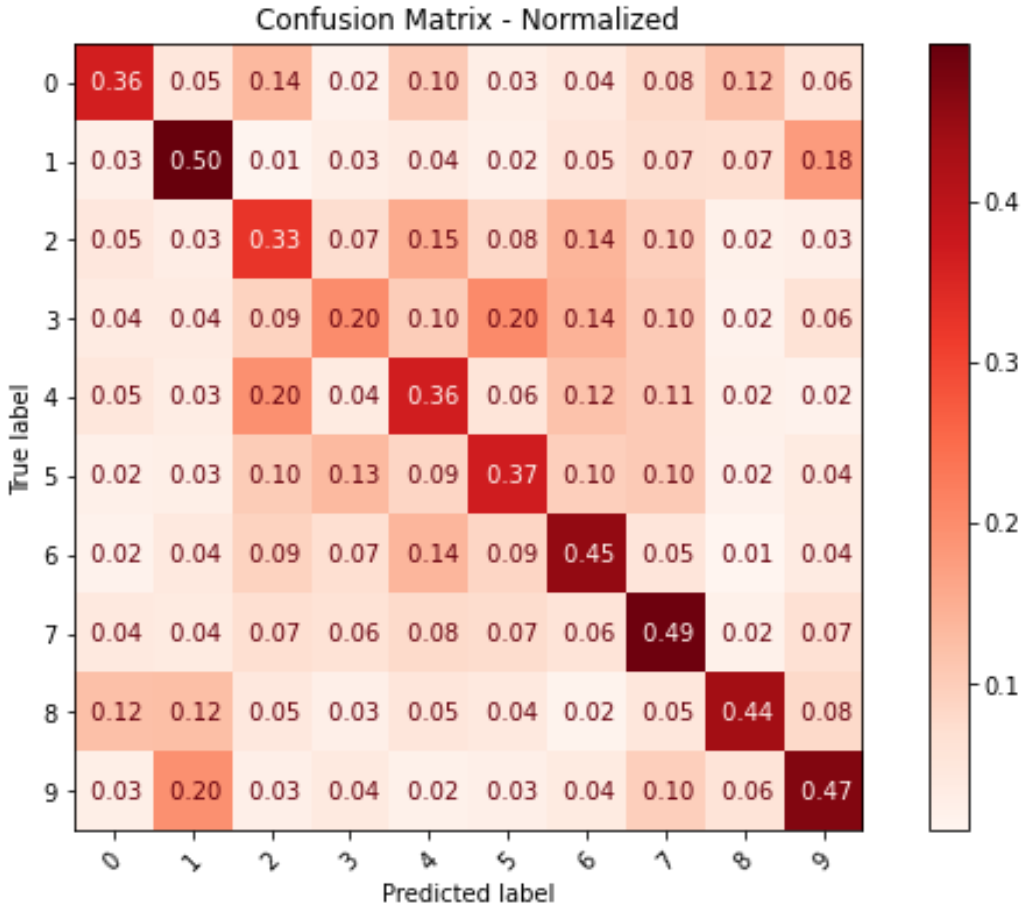
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 2ms/step - loss: 1.6965 - accuracy: 0.3971

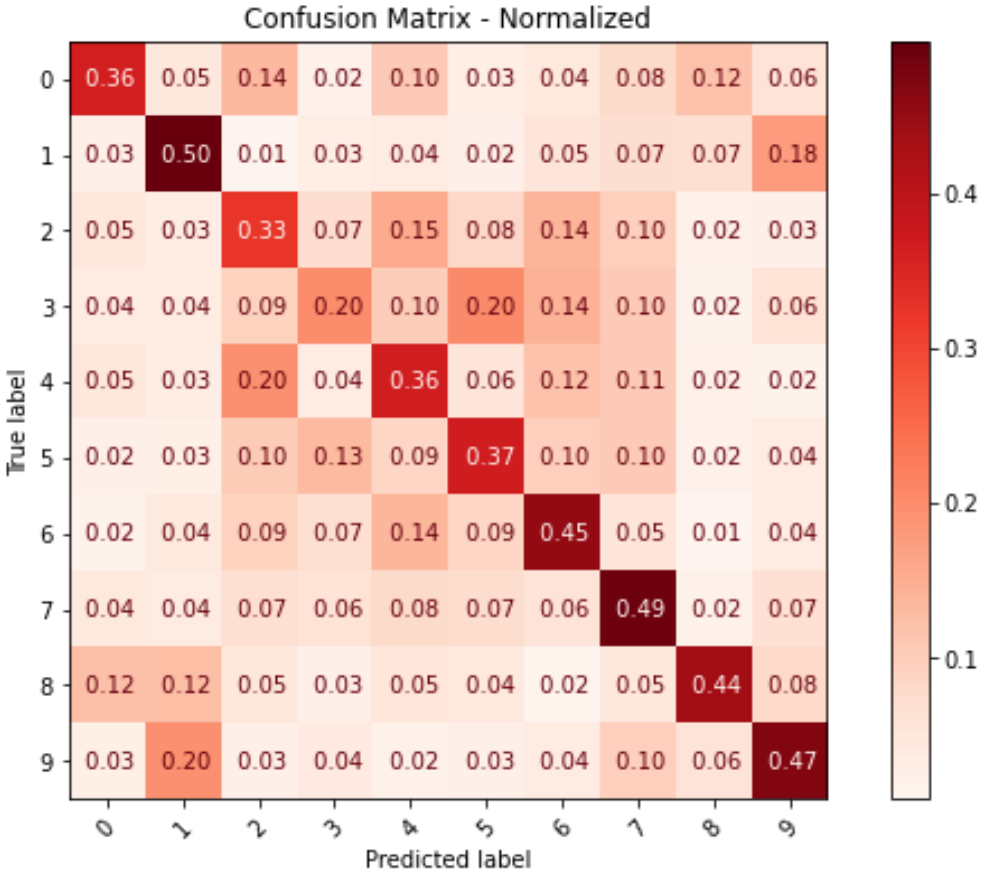
loss in test data is: 1.6964564323425293

accuracy in test data is : 0.3971000015735626

Training time is : 0.05189657211303711



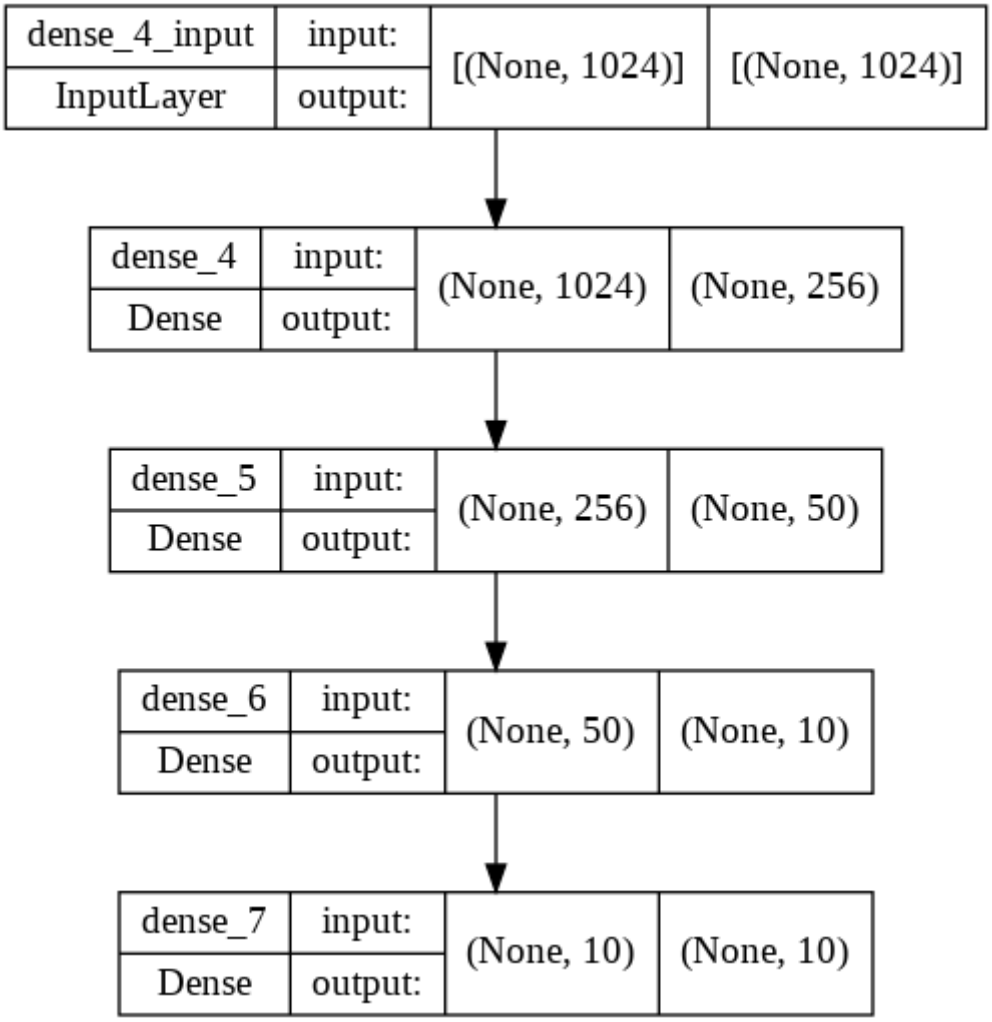
شکل 6. ماتریس آشفتگی نرمال نشده برای نورون‌های 256، 128، 64 و 10



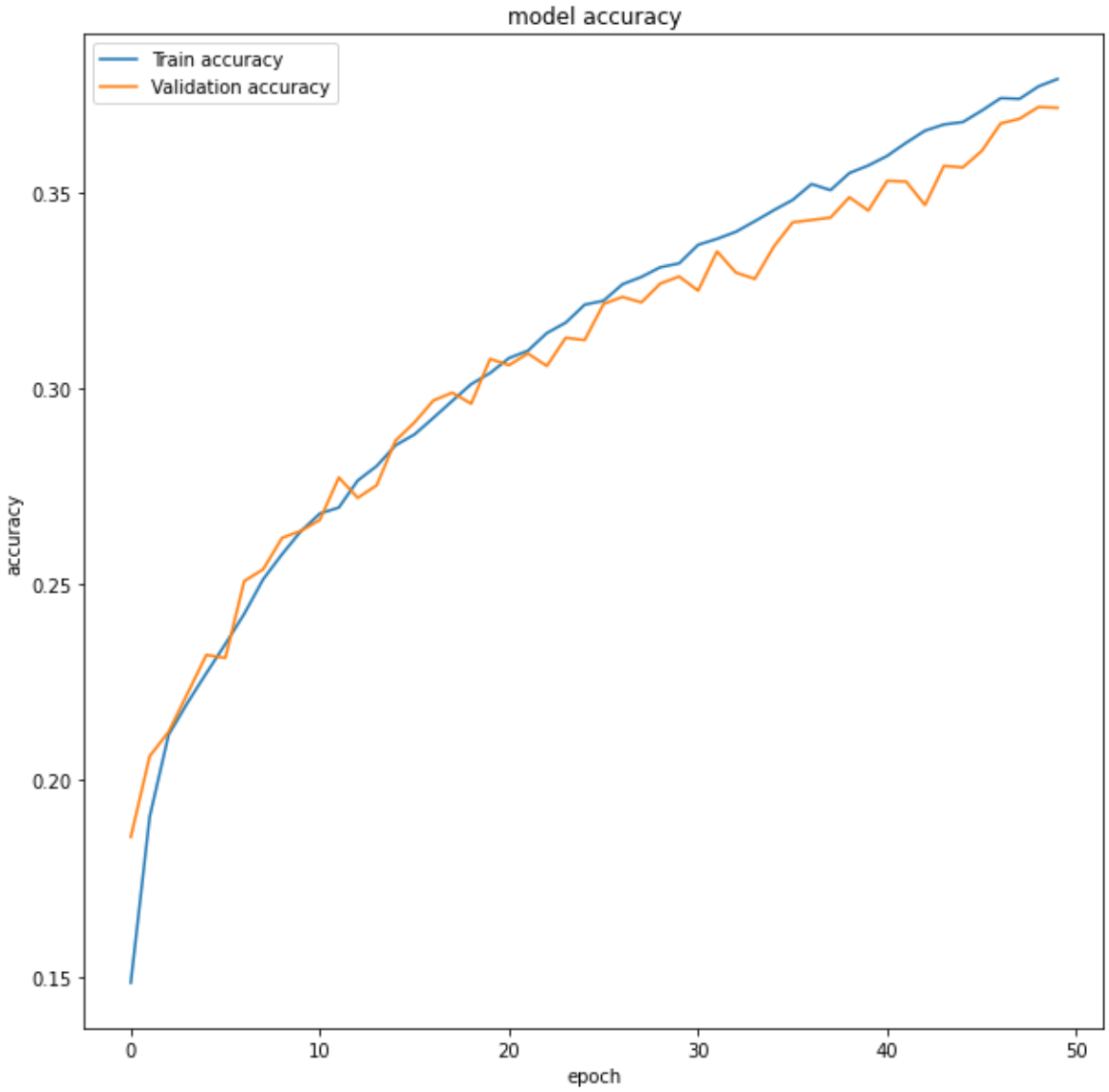
شکل 7. ماتریس آشفتگی نرمال شده برای نورون‌های 256، 128، 64 و 10

حالت اول : تعداد نورون‌های لایه‌ها را تغییر می‌دهیم و تعداد هر دو لایه را کم می‌کنیم :

معماری شبکه، نمودار دقت و تغییرات خطا، خطا، دقت، زمان، ماتریس آشفتگی، مقدار f1-score، recall و precision برای نورون‌های با تعداد 256، 50، 10 و 10 :



شکل8. معماری شبکه با نورون‌های 256، 50، 10 و 10



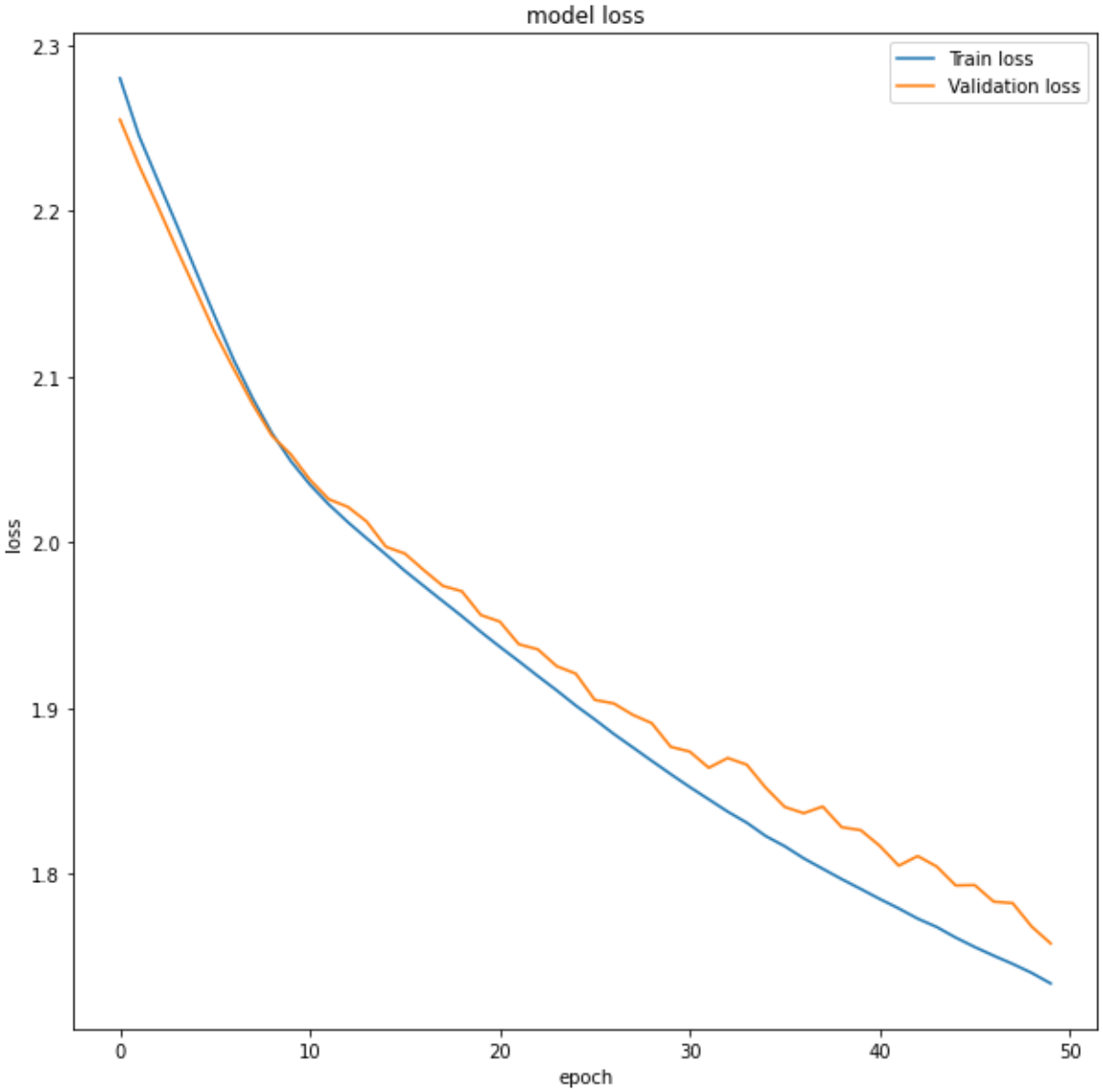
شکل 9. نمودار تغییرات دقت با نورون‌های 256، 50، 10 و 10

مقدار f1-score، recall و precision :

recall is : 0.3697

precision is : 0.36687691965053476

f1 is : 0.3640415031176309



شکل 10. نمودار تغییرات خطا با نورون‌های 256، 50، 10 و 10

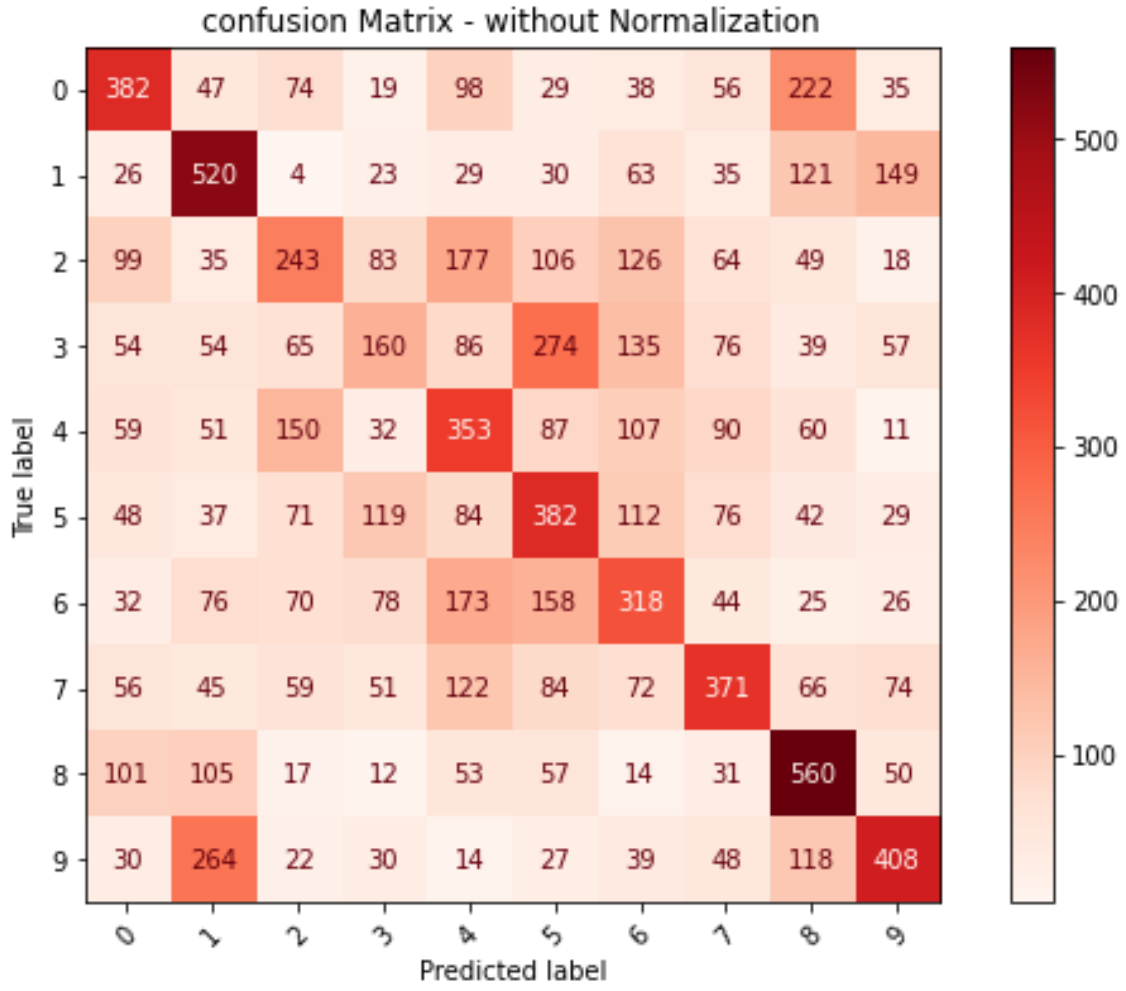
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 4ms/step - loss: 1.7585 - accuracy: 0.3697

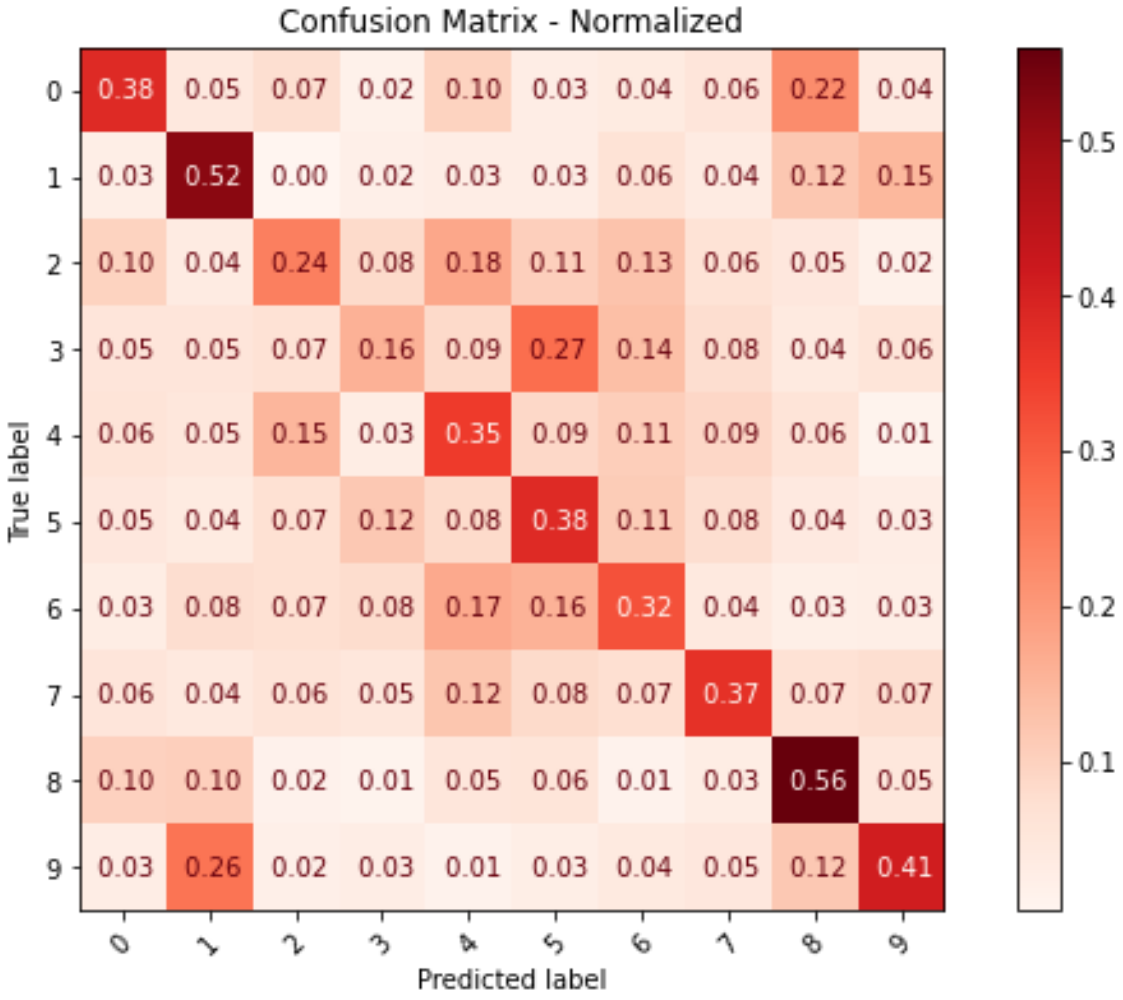
loss in test data is: 1.7585458755493164

accuracy in test data is : 0.36970001459121704

Training time is : 0.05846285820007324



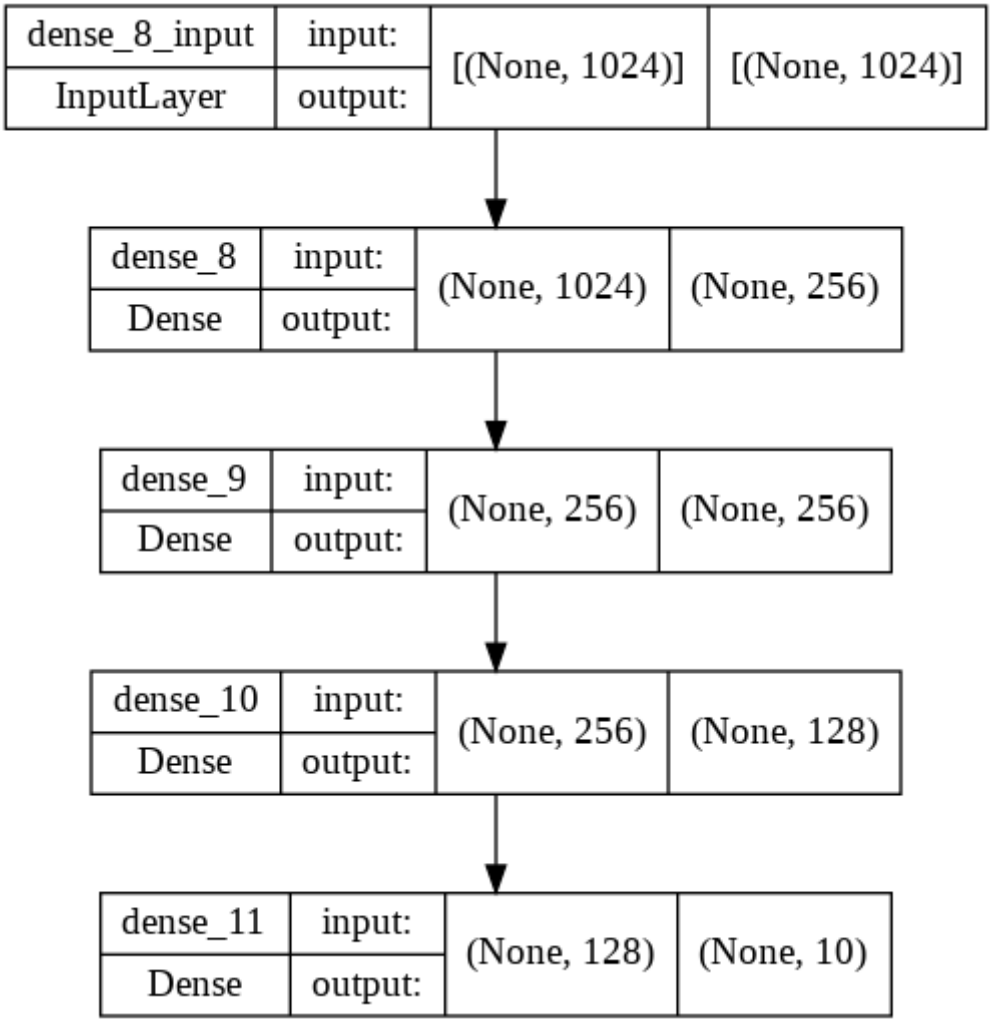
شکل 11. ماتریس آشفتگی نرمال نشده با نورون‌های 256، 50، 10 و 10



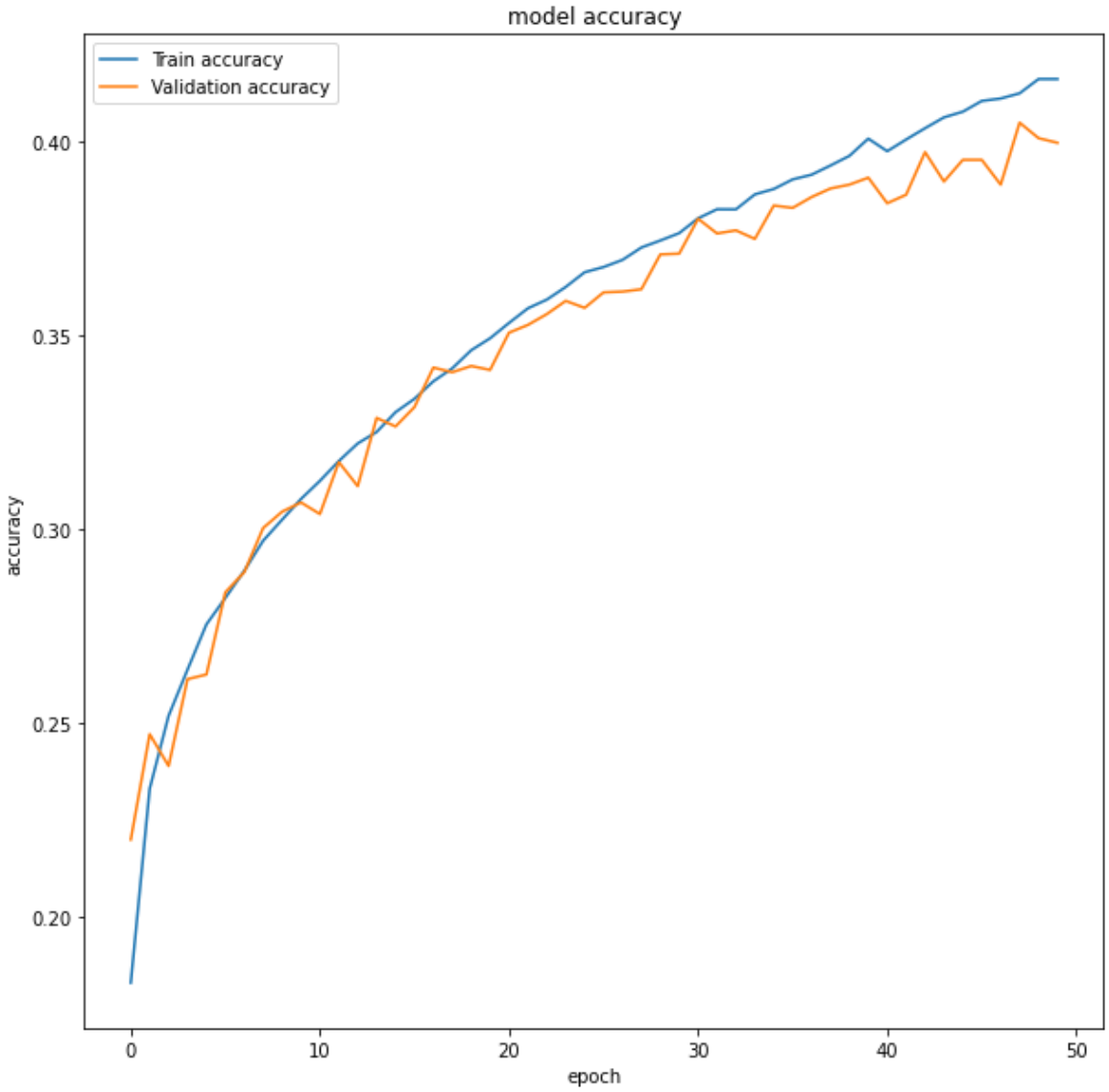
شکل 12. ماتریس آشفتگی نرمال شده با نورون‌های 256، 50، 10 و 10

حالت دوم : تعداد نورون‌های لایه‌ها را تغییر می‌دهیم و تعداد هر دو لایه را زیاد می‌کنیم :

معماری شبکه، نمودار دقت و تغییرات خطا، خطا، دقت، زمان، ماتریس آشفتگی، مقدار f1-score، recall و precision برای نورون‌های با تعداد 256، 50، 10 و 10 :



شکل 13. معماری شبکه با نورون‌های 256، 256، 128 و 10



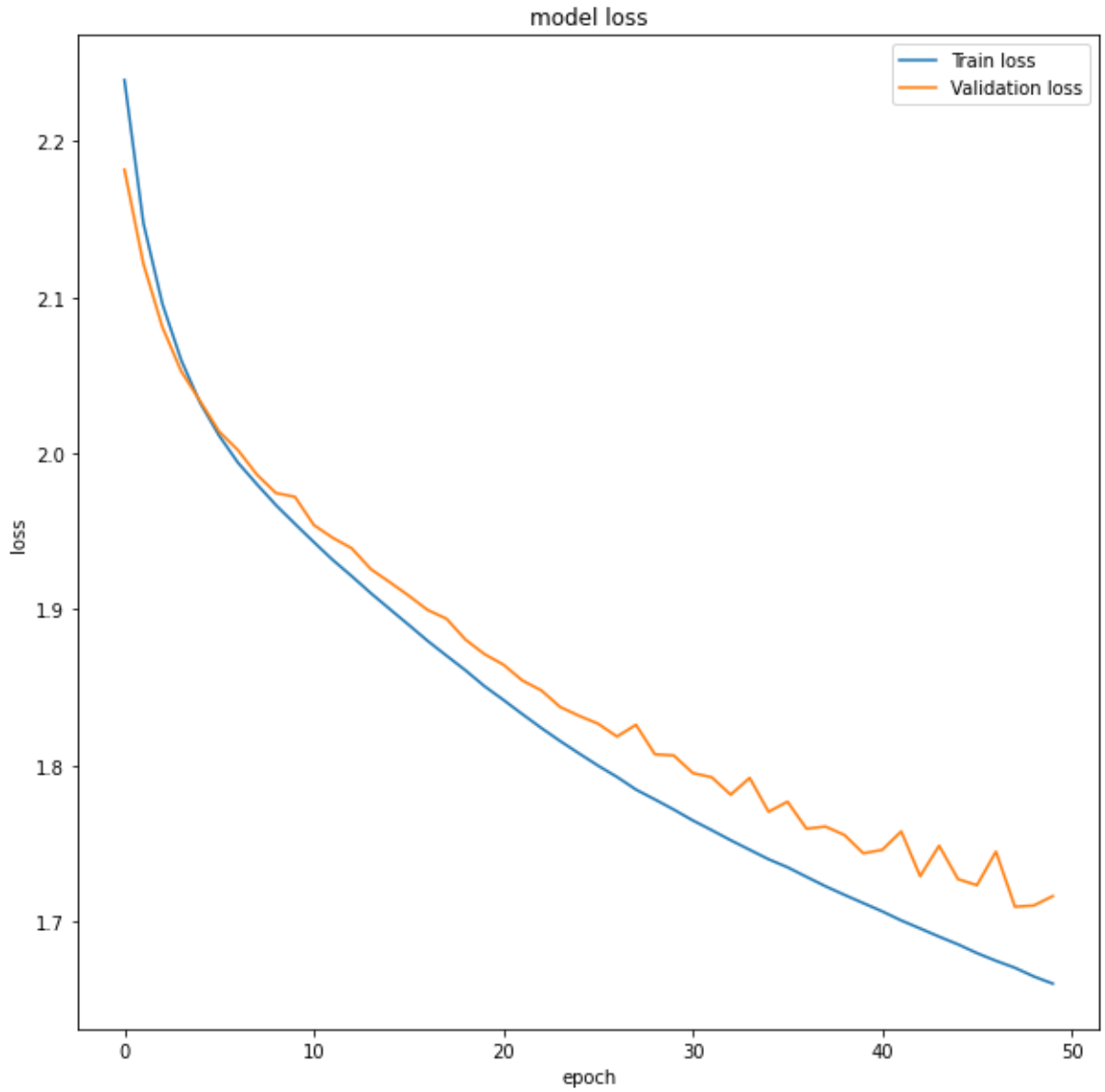
شکل 14. نمودار تغییرات دقت با نورون‌های 256، 256، 128 و 10

مقدار f1-score، recall و precision :

recall is : 0.396

precision is : 0.39755839188090164

f1 is : 0.3890604263589467



شکل 15. نمودار تغییرات خطا با نورون‌های 256، 256، 128 و 10

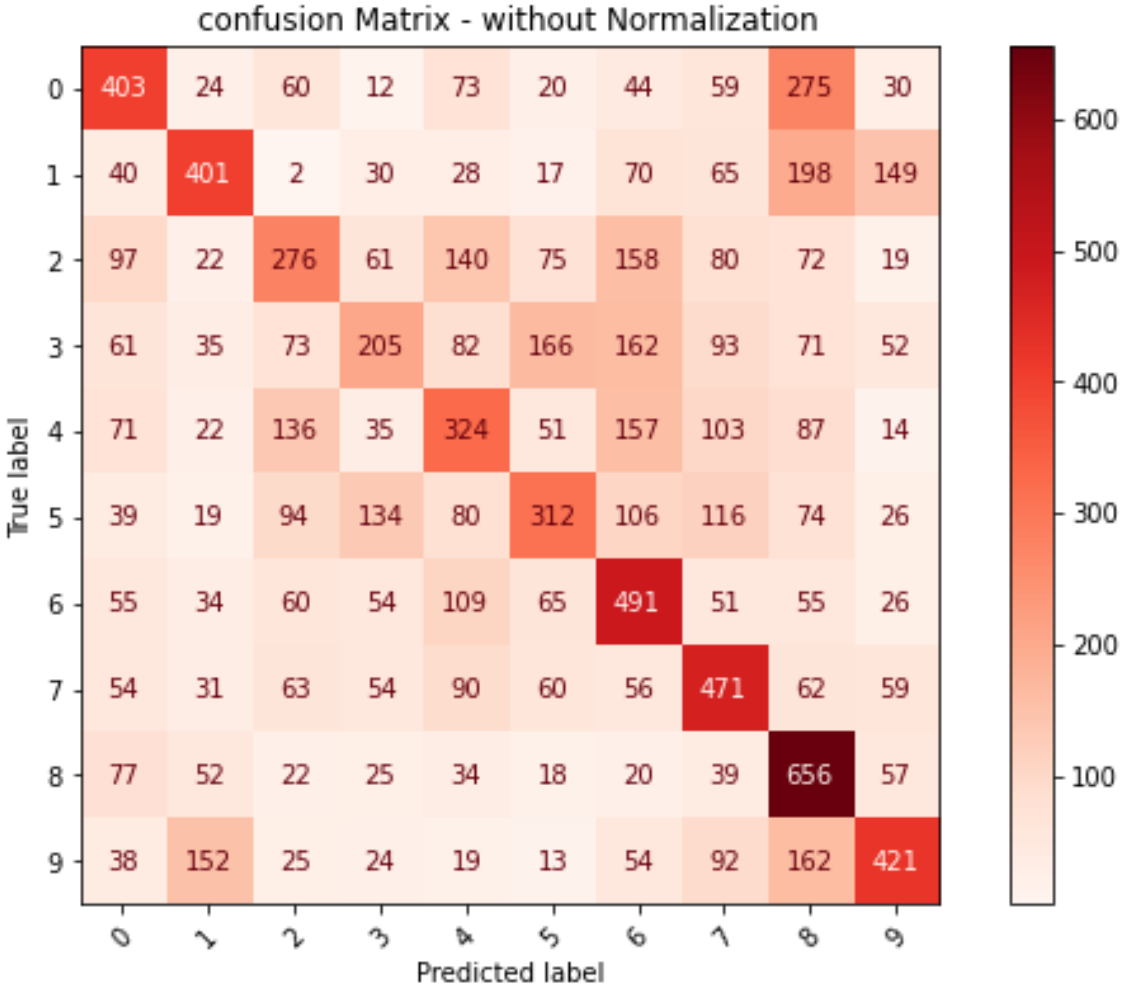
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 2ms/step - loss: 1.7137 - accuracy: 0.3960

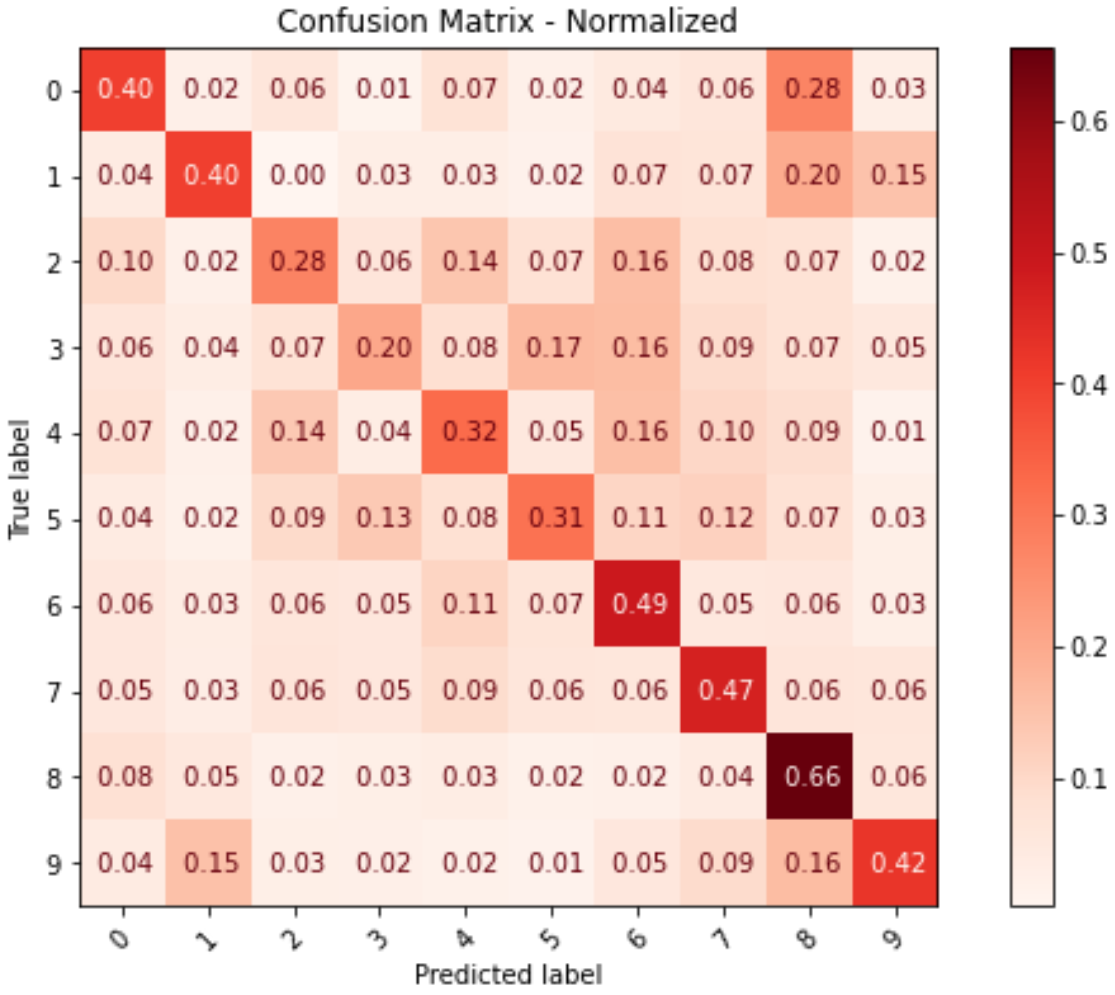
loss in test data is: 1.713722825050354

accuracy in test data is : 0.3959999978542328

Training time is : 0.057444810867309570



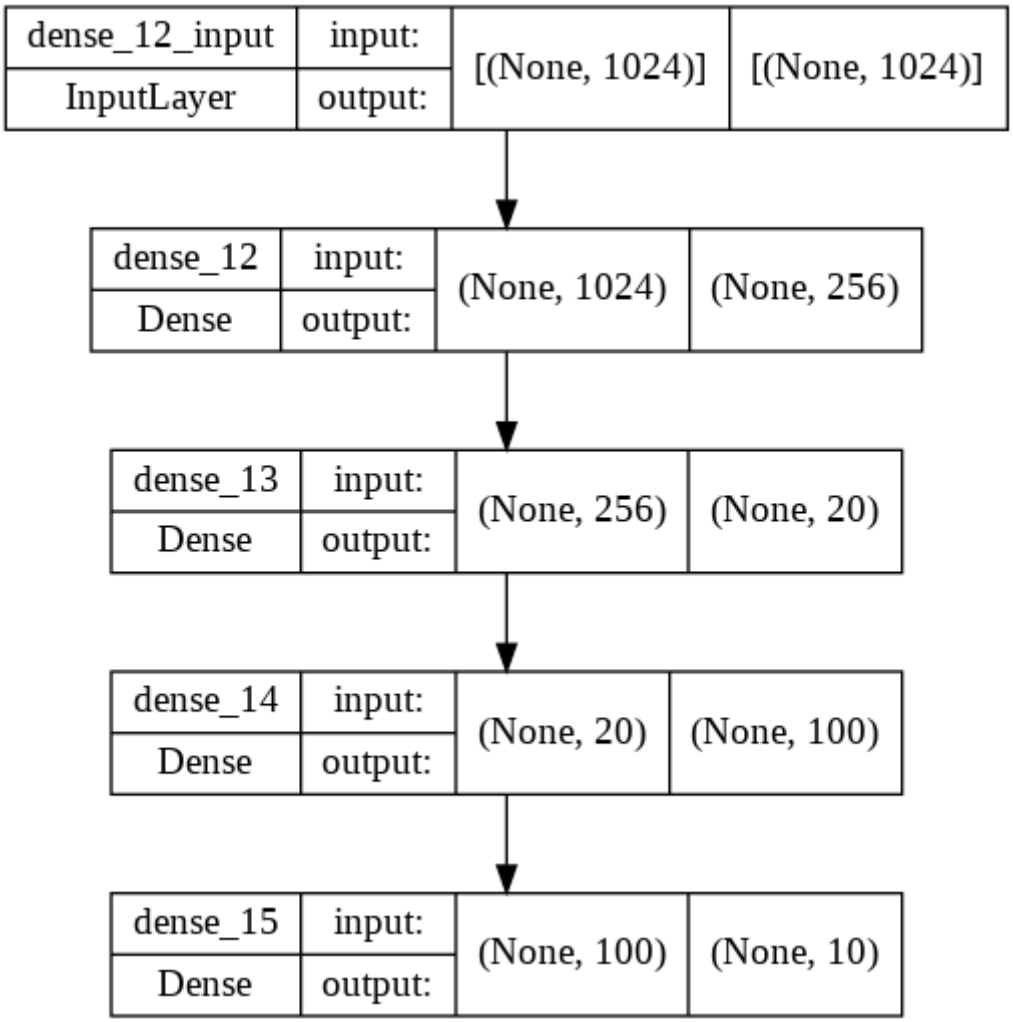
شکل 16. ماتریس آشفتگی نرمال نشده با نورون‌های 256، 256، 128 و 10



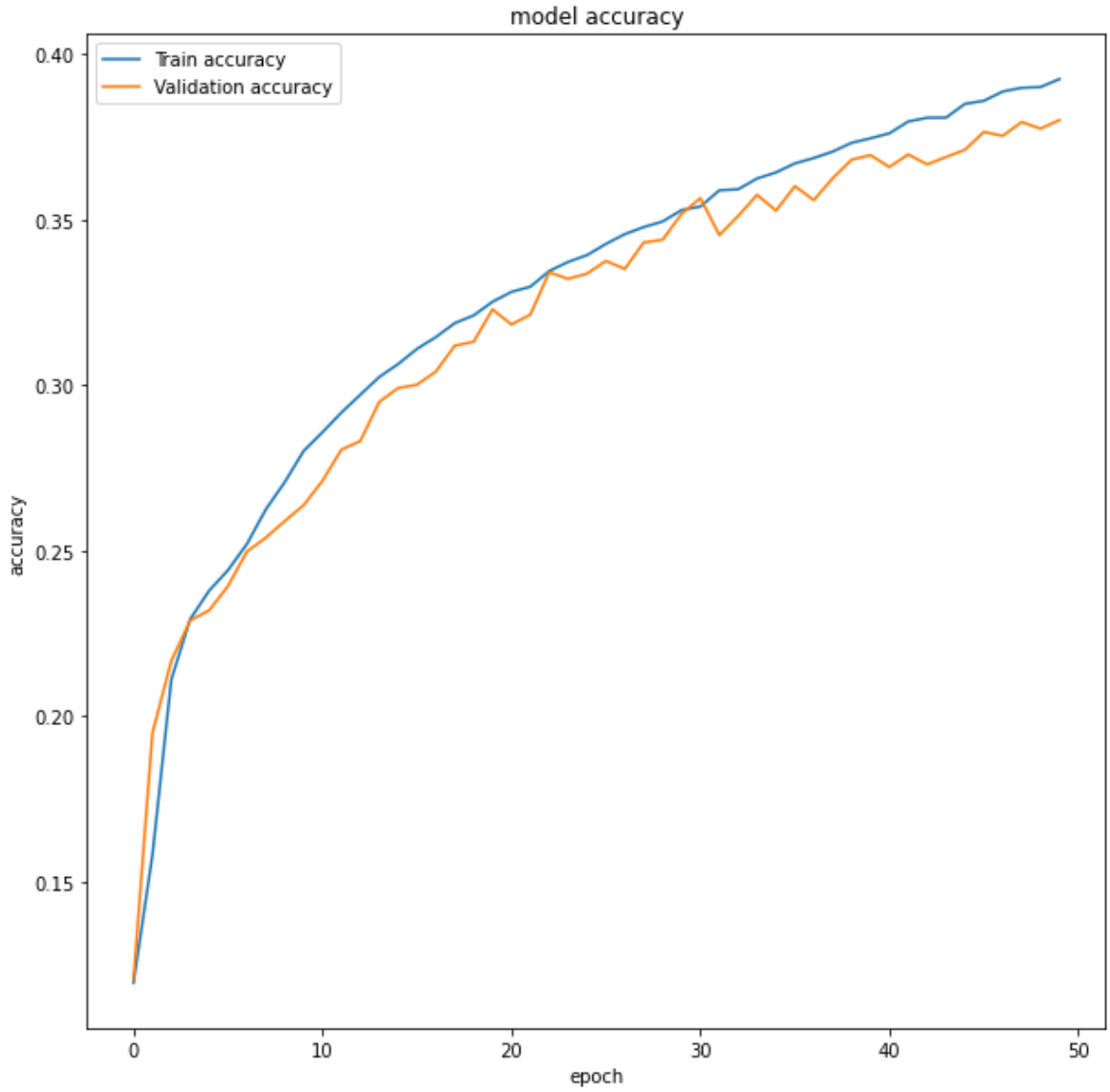
شکل 17. ماتریس آشفتگی نرمال شده با نورون‌های 256، 256، 128 و 10

حالت سوم : تعداد نورون‌های لایه‌ها را تغییر می‌دهیم و تعداد یکی را زیاد و دیگری را کم می‌کنیم :

معماری شبکه، نمودار دقت و تغییرات خطا، خطا، دقت، زمان، ماتریس آشفتگی، مقدار f1-score، recall و precision برای نورون‌های با تعداد 256، 20، 100 و 10 :



شکل 18. معماری شبکه با نورون‌های 256، 20، 100 و 10



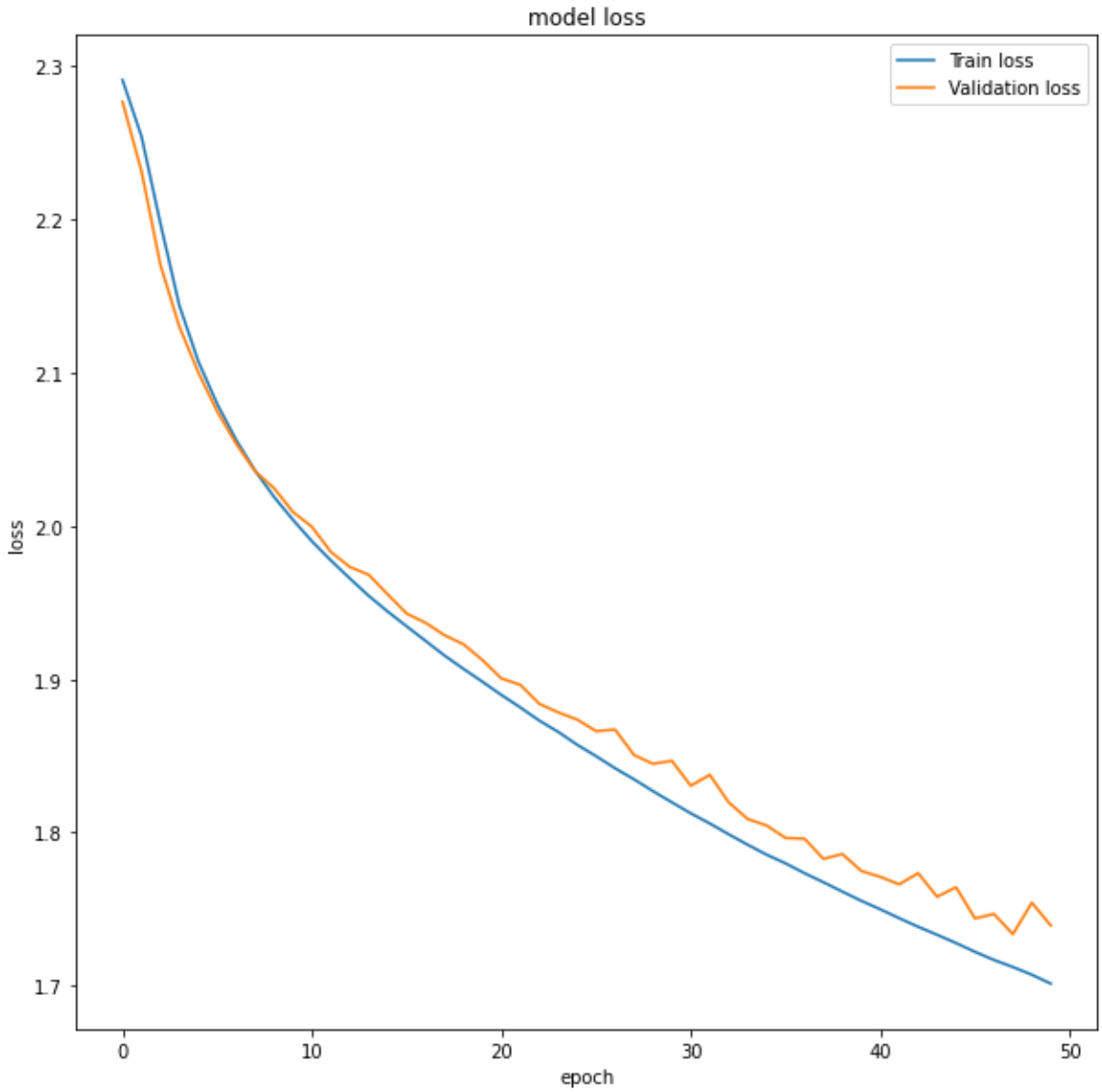
شکل 19. نمودار تغییرات دقت با نورون‌های 256، 20، 100 و 10

مقدار f1-score، recall و precision :

recall is : 0.3822

precision is : 0.3861409361131888

f1 is : 0.37659429226334606



شکل 20. نمودار تغییرات خطا با نورون‌های 256، 20، 100 و 10

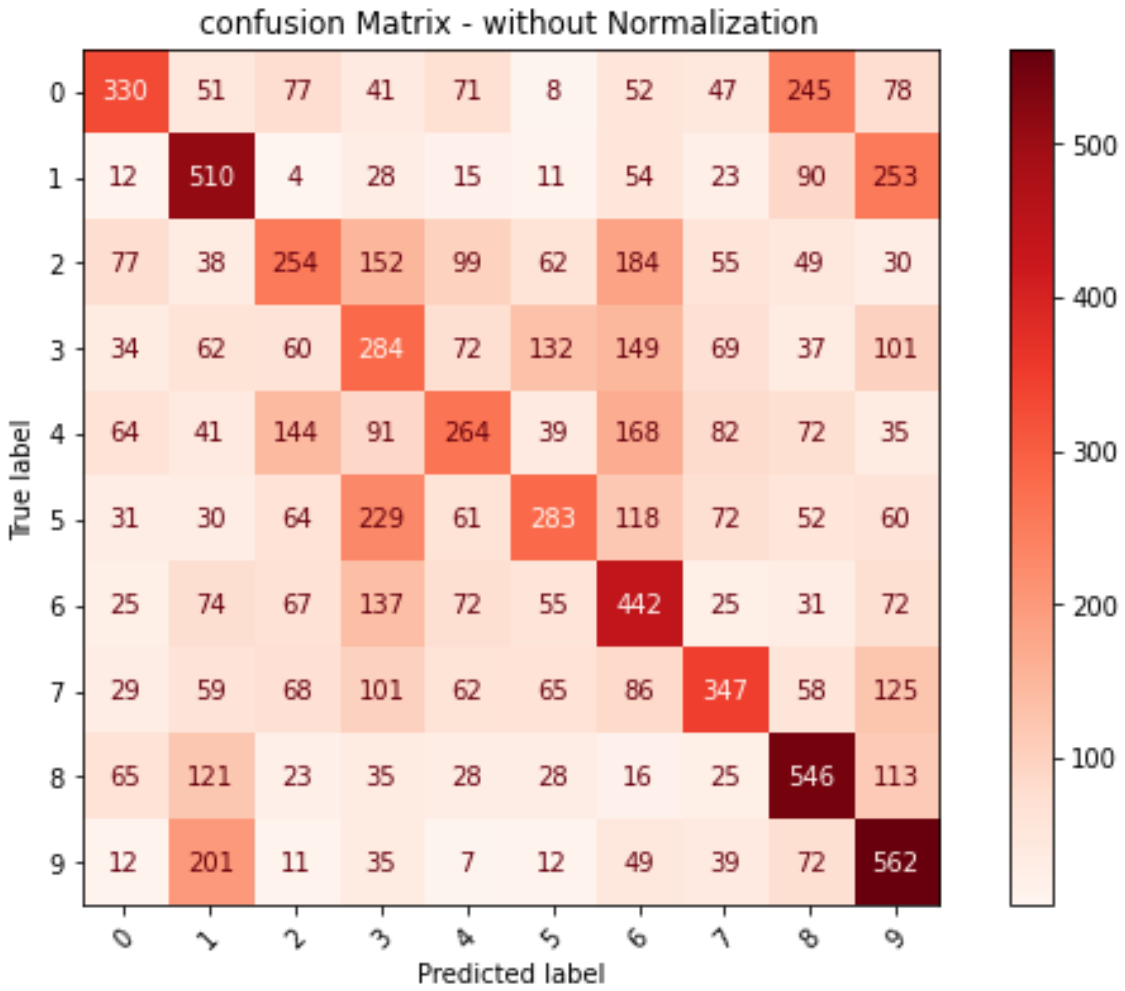
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 2ms/step - loss: 1.7420 - accuracy: 0.3822

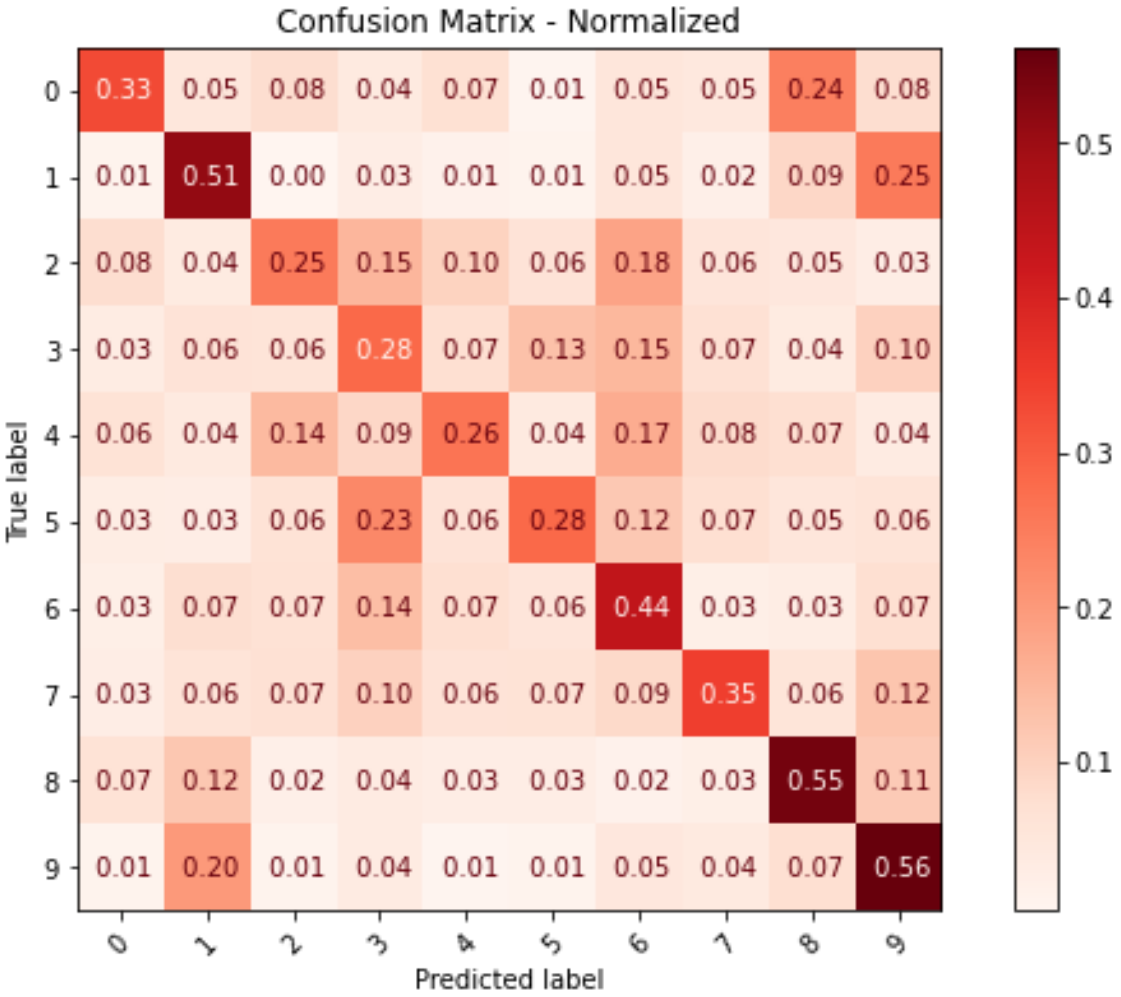
loss in test data is: 1.742018222808838

accuracy in test data is : 0.3822000026702881

Training time is : 0.0599234104156494 14



شکل 21. ماتریس آشفتگی نرمال نشده با نورون‌های 256، 20، 100 و 10

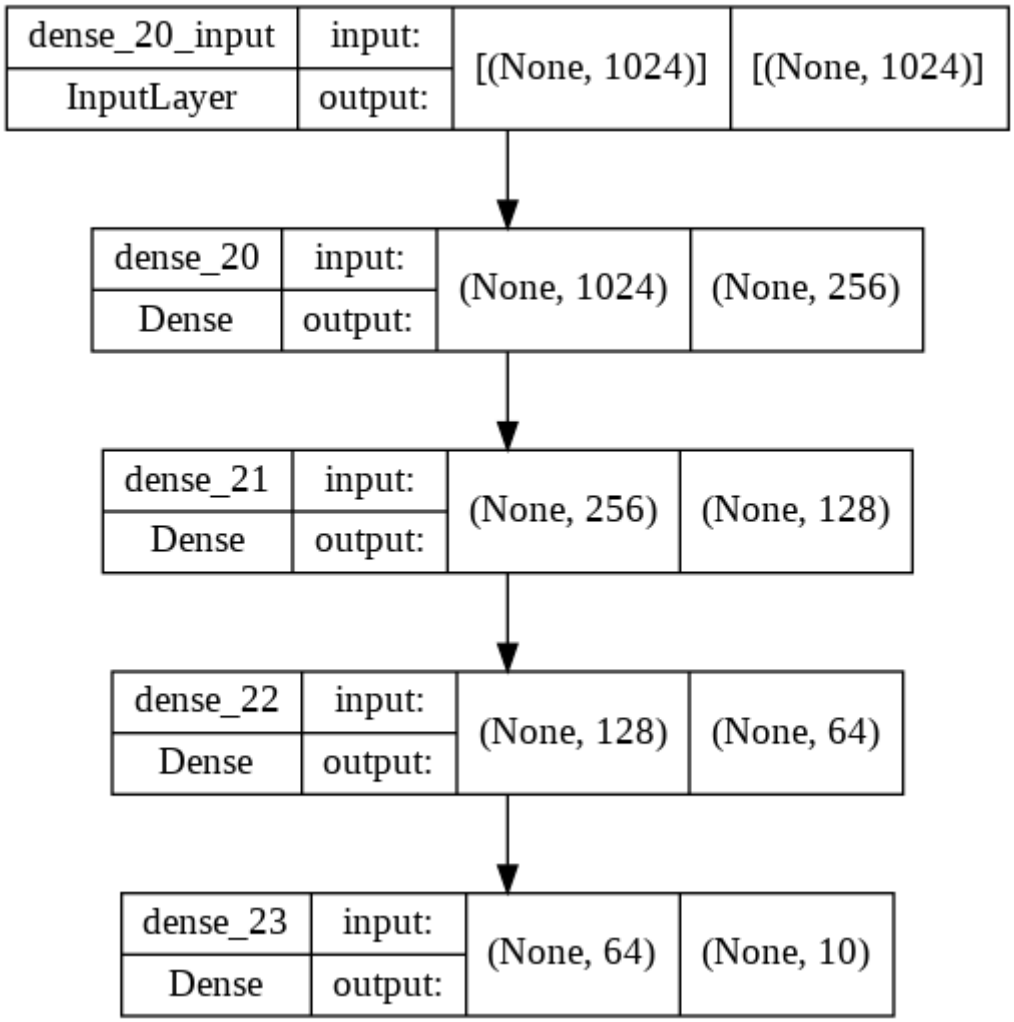


شکل 22. ماتریس آشفتگی نرمال نشده با نورون‌های 256، 20، 100 و 10

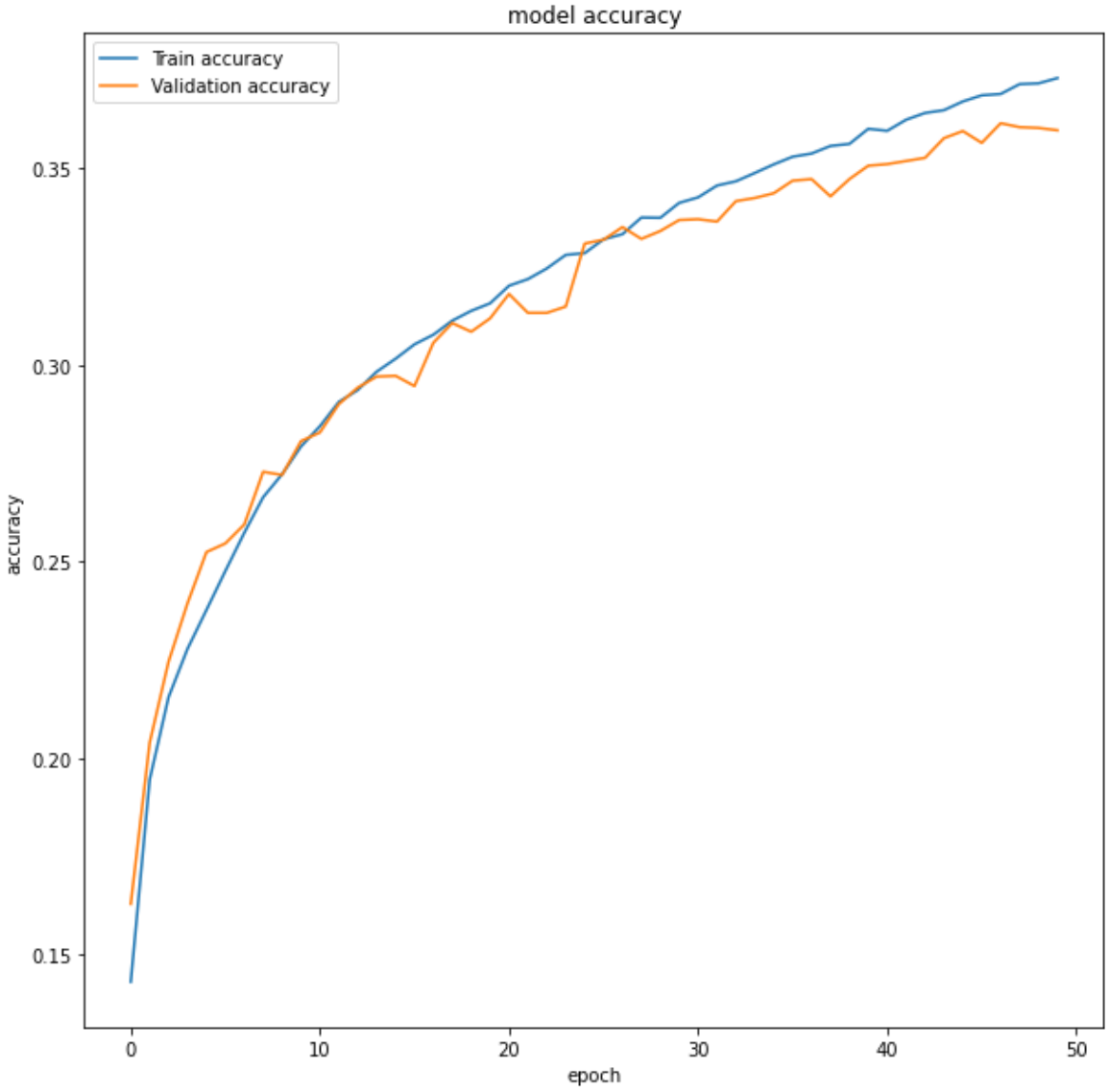
قسمت د)

بهترین مدل قسمت قبل با تعداد نورون‌های 256، 128، 64 و 10 با batch size = 32 بود.

Batch size = 64 :



شکل 23. معماری شبکه با نورون‌های 256، 128، 64 و 10



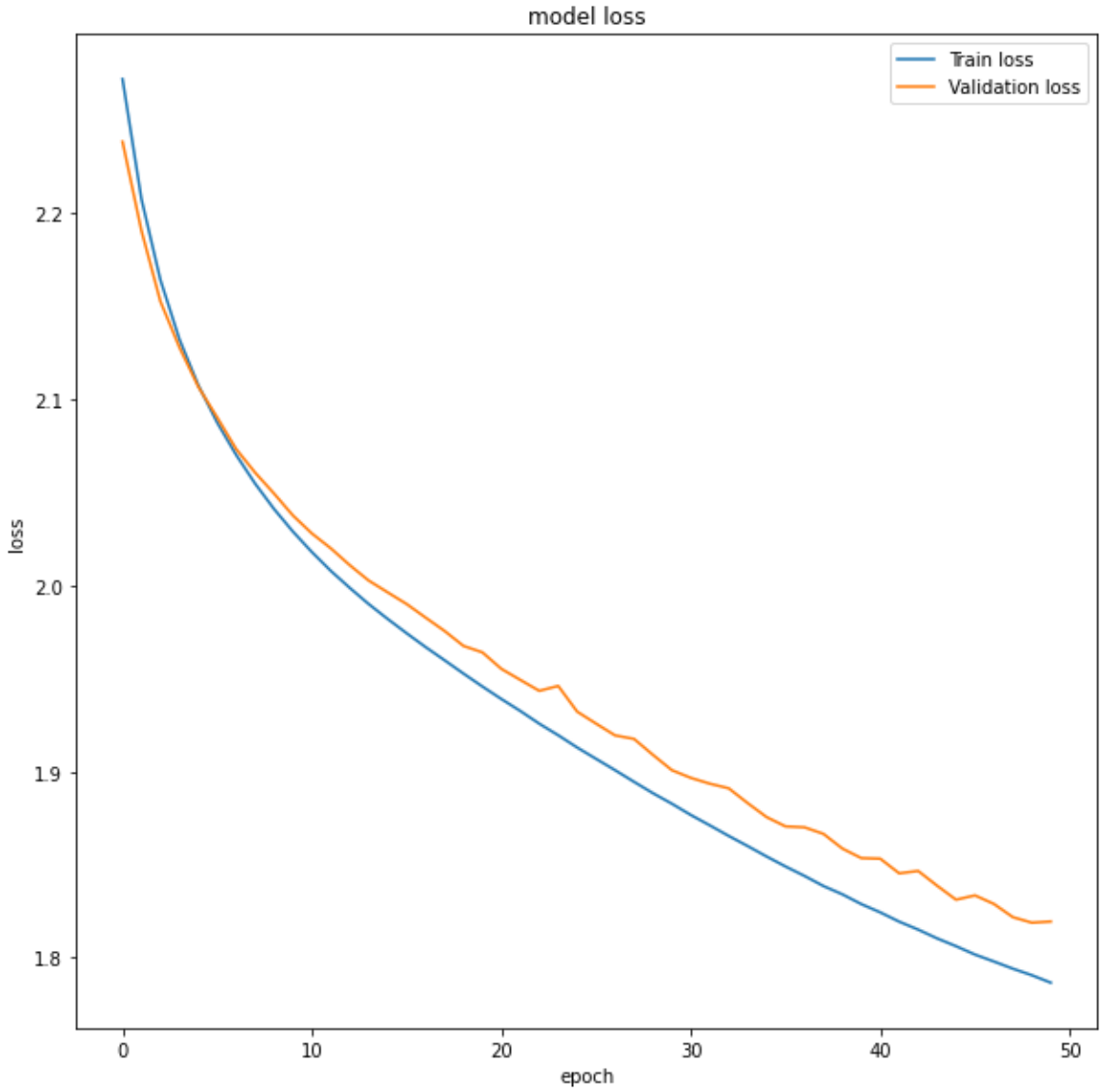
شکل 24. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision :

recall is : 0.3682

precision is : 0.3733344595847668

f1 is : 0.36380332217222905



شکل 25. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

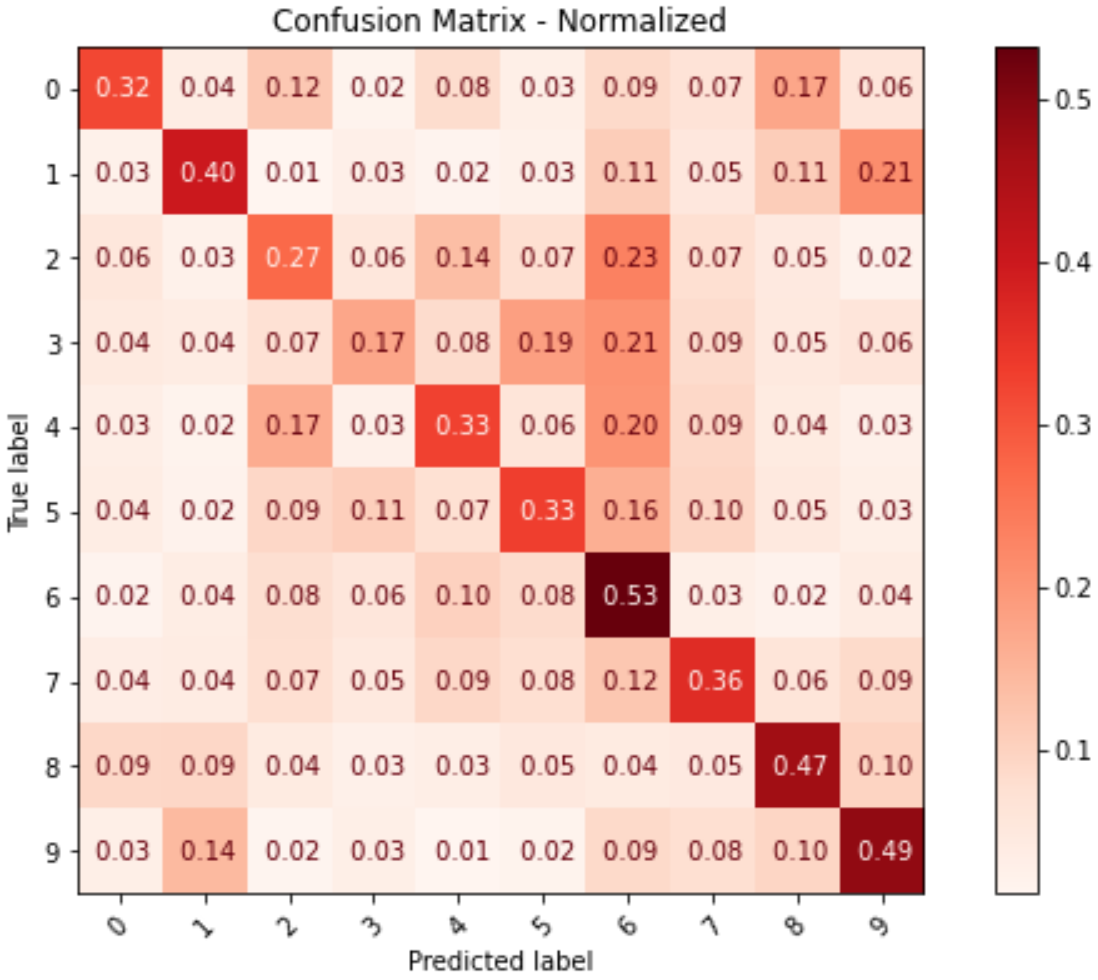
خطا، دقت و زمان آموزش برای داده تست :

```
313/313 [==============================] - 1s 2ms/step -
loss: 1.8024 - accuracy: 0.3682
```

loss in test data is: 1.8023937940597534

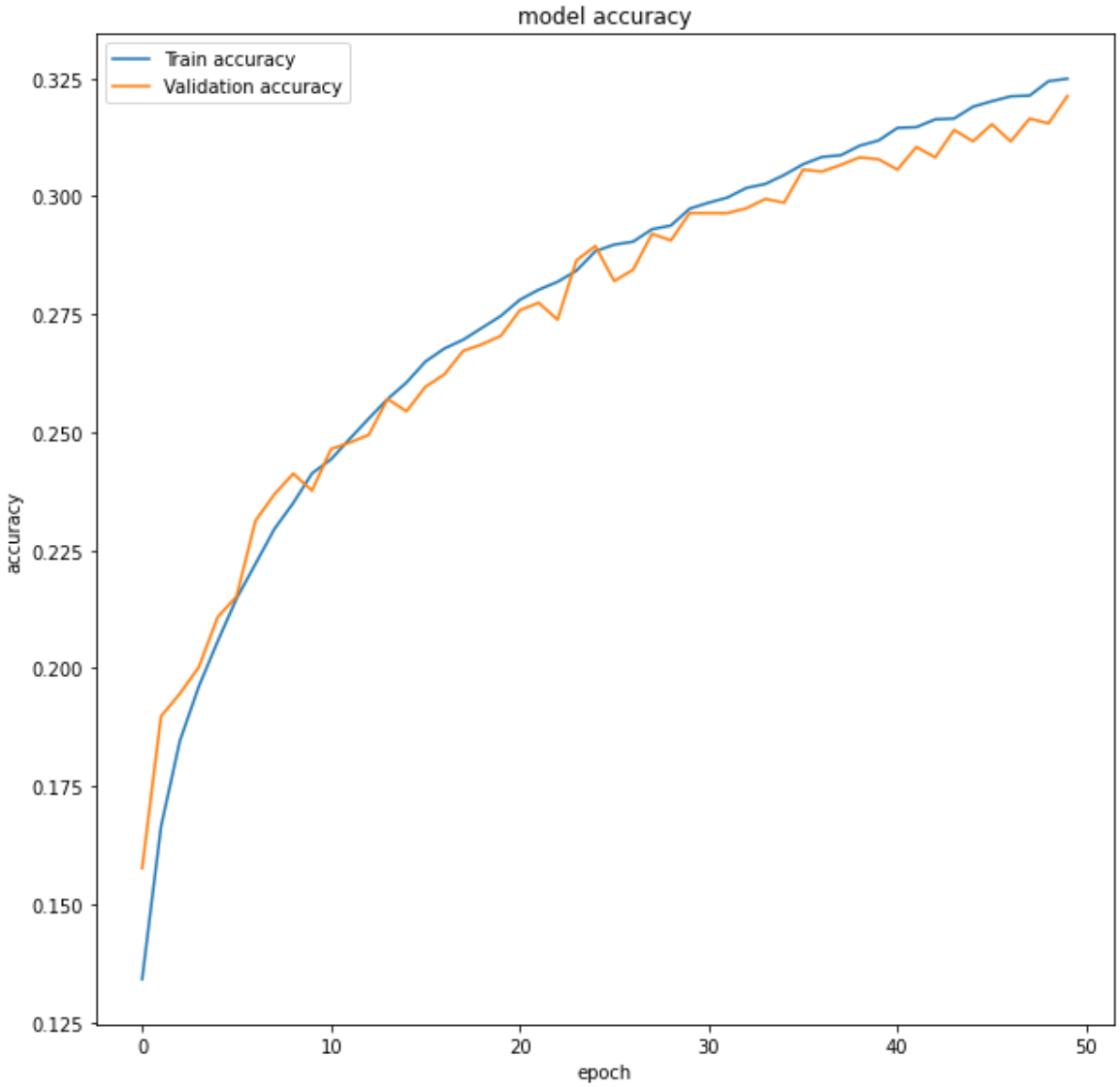
accuracy in test data is : 0.36820000410079956

Training time is : 0.04997587203979492



شکل 26. ماتریس آشفتگی نرمال شده با نورون‌های 256، 128، 64 و 10

: Batch size = 128



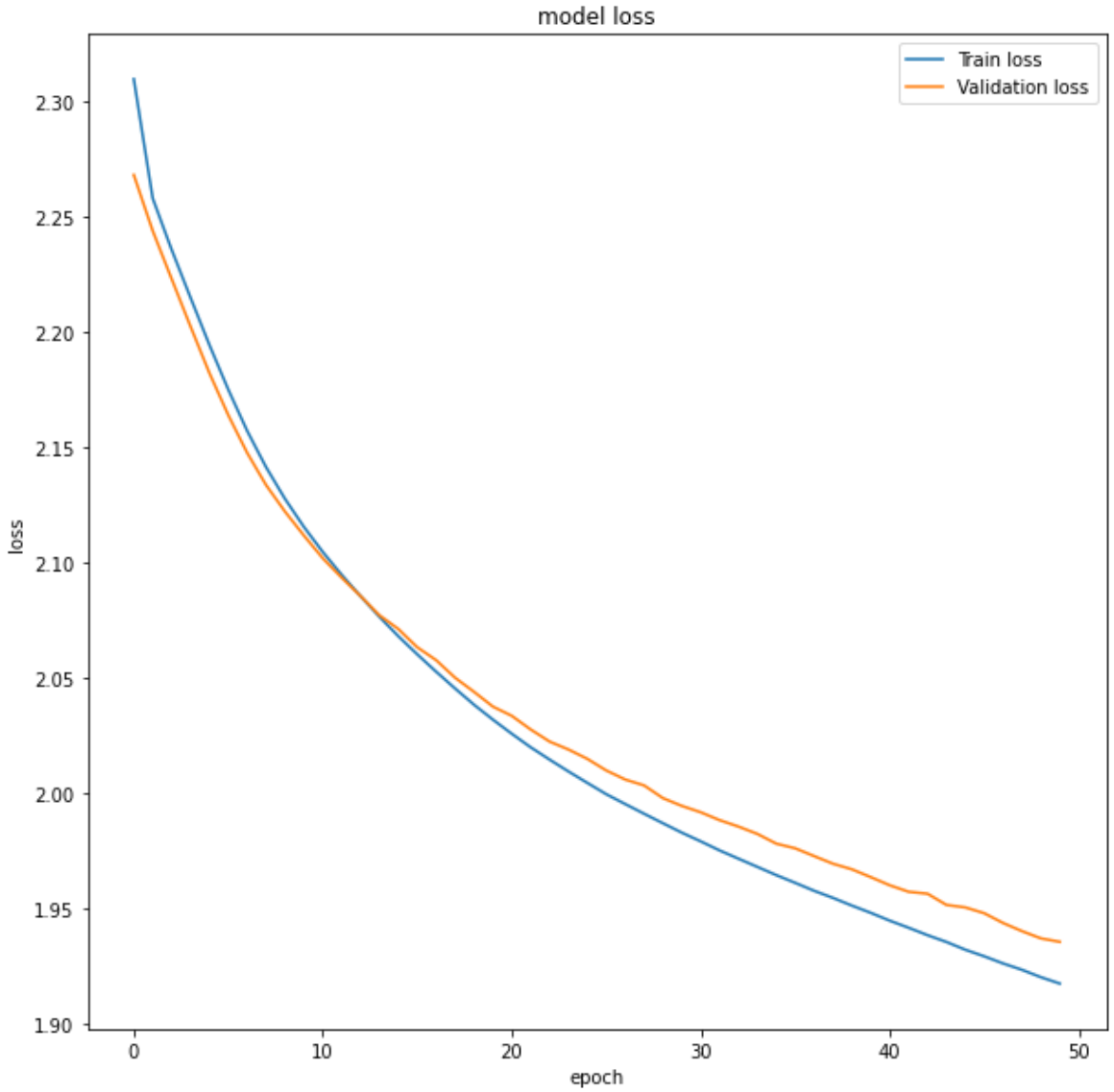
شکل 27. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

: مقدار f1-score، recall و precision

recall is : 0.3305

precision is : 0.3244507026938427

f1 is : 0.32440882295182316



شکل 28. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

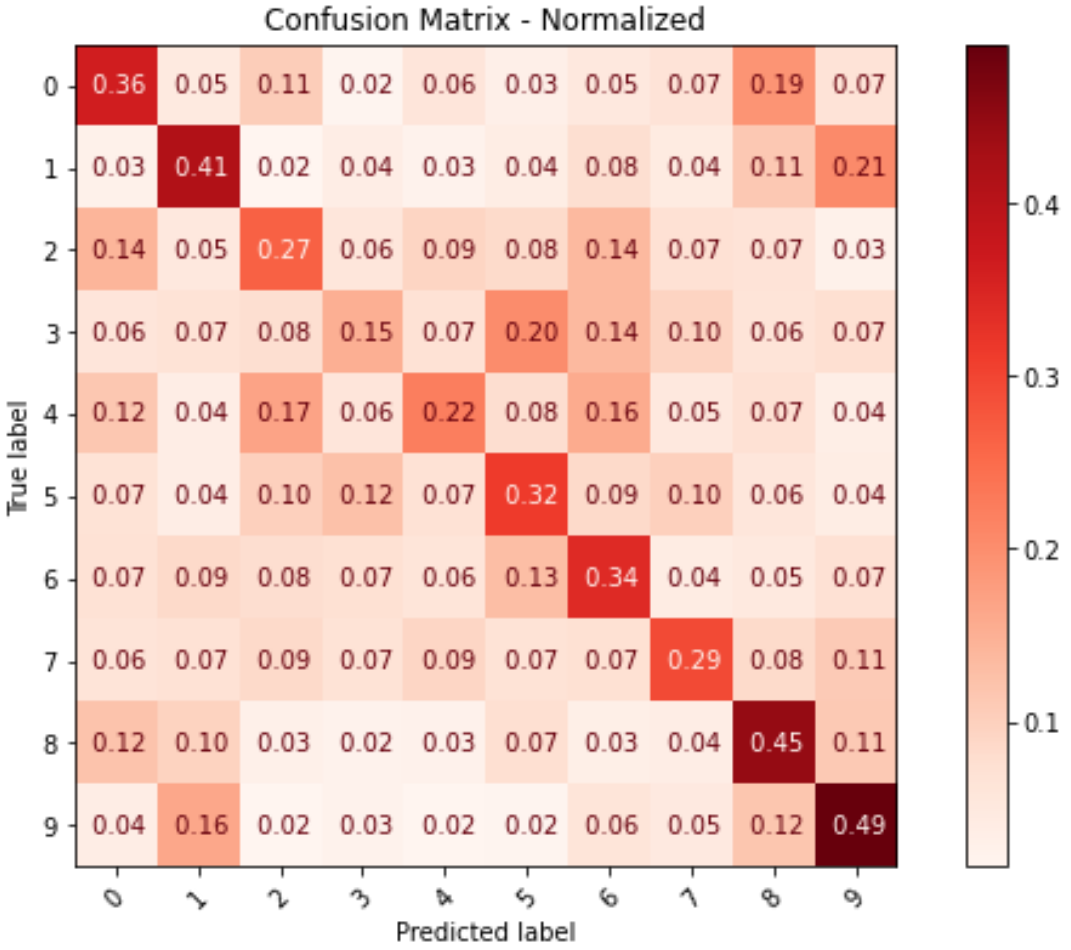
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 3ms/step - loss: 1.9242 - accuracy: 0.3305

loss in test data is: 1.9242254495620728

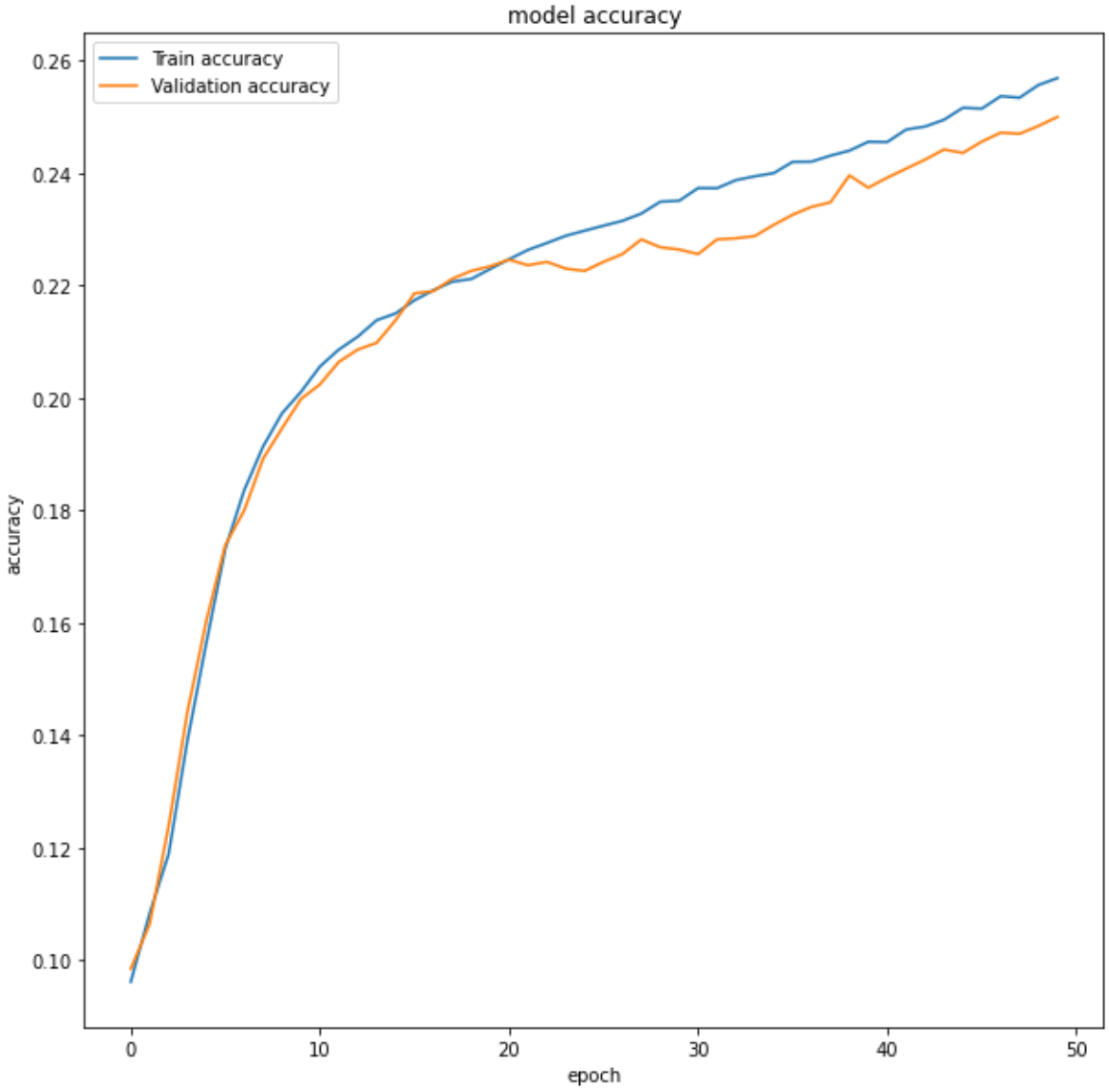
accuracy in test data is : 0.3305000066757202

Training time is : 0.052538394927978516



شکل 29. ماتریس آشفتگی نرمال شده با نورون‌های 256، 128، 64 و 10

Batch size = 512 :



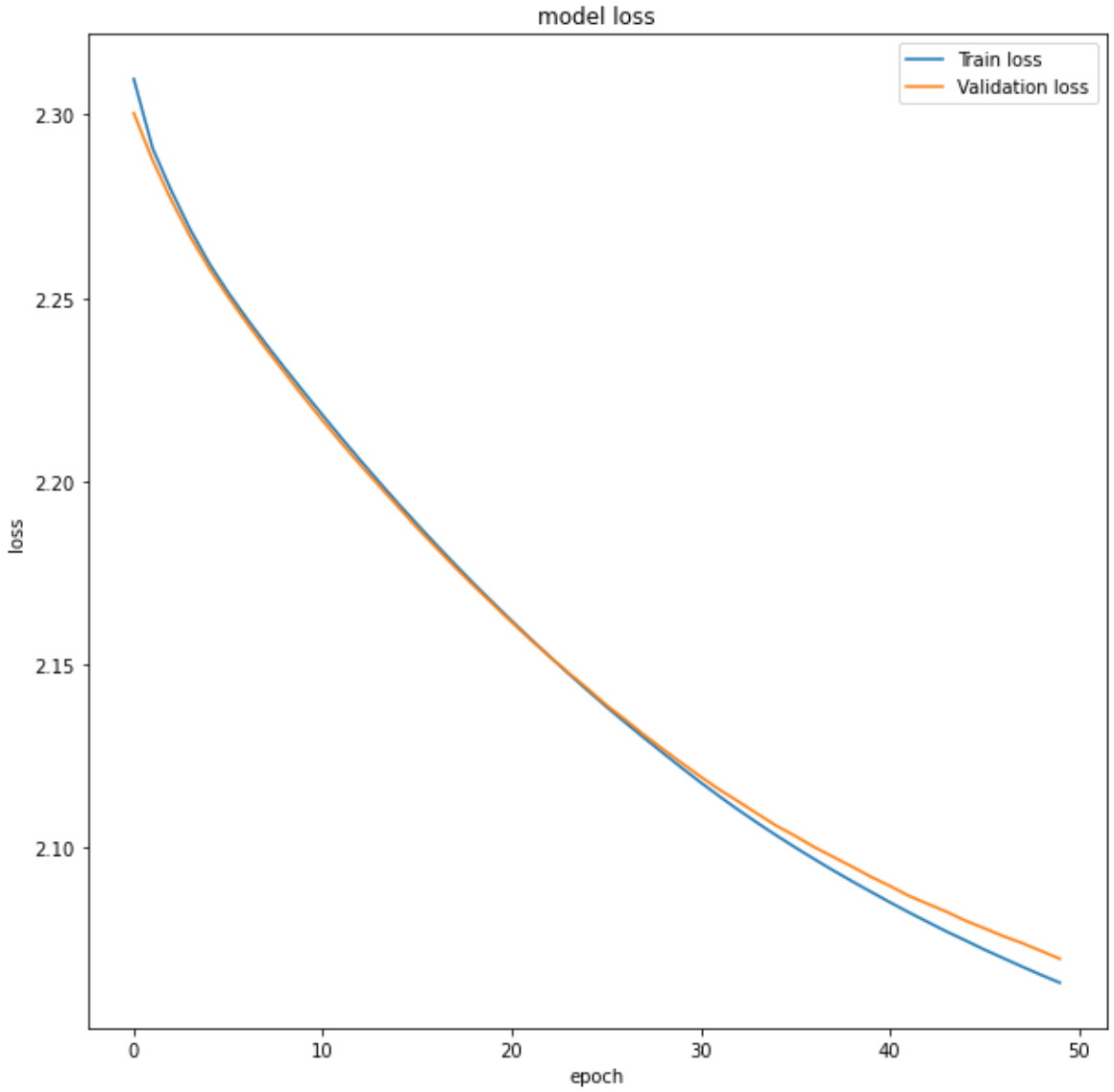
شکل 30. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision :

recall is : 0.2543

precision is : 0.24617862282932484

f1 is : 0.22857246569711873



شکل 31. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

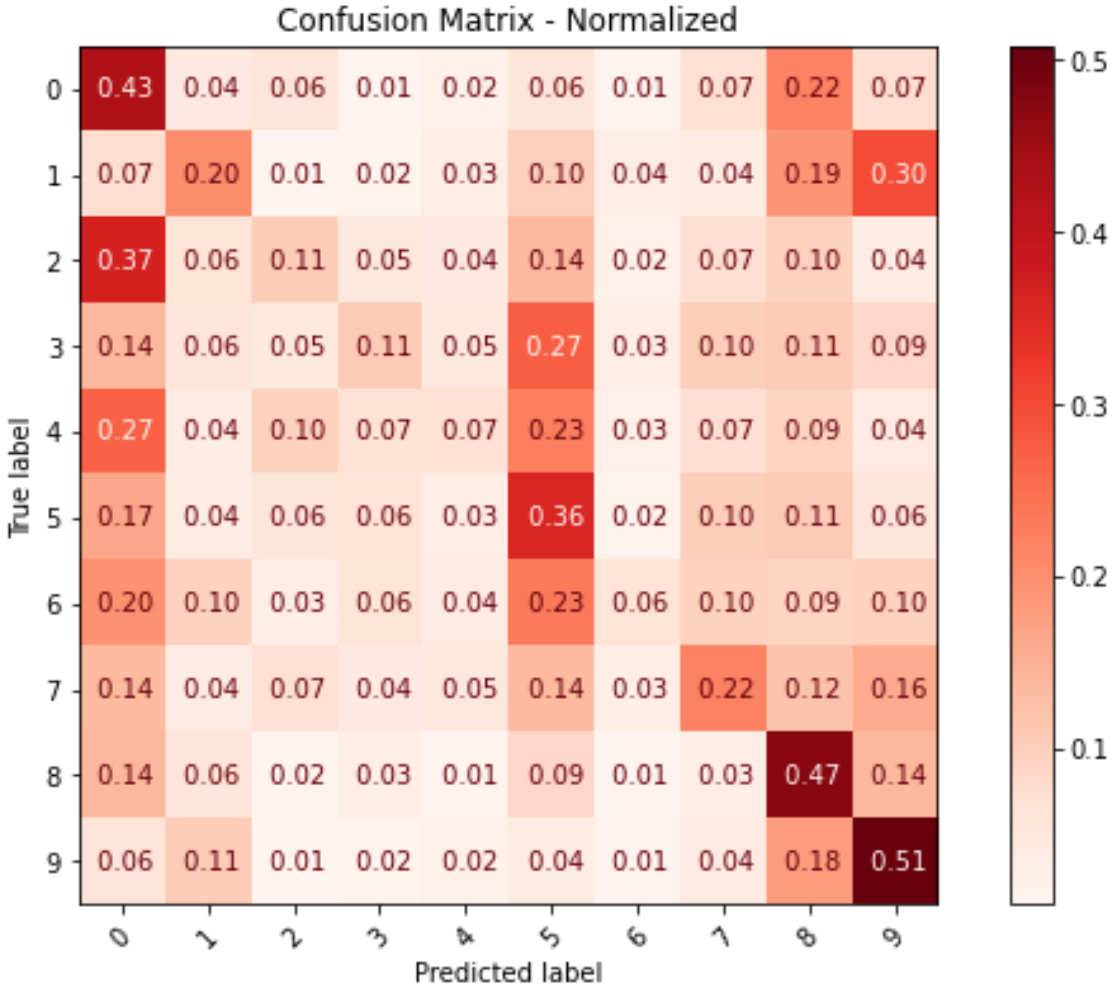
خطا، دقت و زمان آموزش برای داده تست :

```
313/313 [==============================] - 1s 2ms/step -
loss: 2.0667 - accuracy: 0.2543
```

loss in test data is: 2.066694498062134

accuracy in test data is : 0.25429999828338623

Training time is : 0.054247379302978516



شکل 32. ماتریس آشفتگی نرمال شده با نورون‌های 256، 128، 64 و 10

*** مناسب‌ترین مقدار batch size، 32 است چون مقدار دقت در این حالت بیشتر از بقیه و مقدار خطا نیز کمتر از بقیه موارد است. زمان batch size = 64 کمتر از زمان batch size = 32 است ولی بقیه موارد زمانشان از زمان batch size = 32 بیشتر است.

قسمت ه)

تابع sigmoid :

مزایا : این تابع تمایزپذیر (Differentiable) است؛ یعنی در هر قسمت از منحنی می‌توانیم شیب میان دو نقطه را حساب کنیم. از آنجا که این تابع مقادیر را میان صفر و یک قرار می‌دهد، نوعی عادی‌سازی را برای خروجی هر نورون انجام می‌دهد.

معایب : با محوشدگی گرادیان (Vanishing Gradient) مقادیر بسیار بزرگ یا بسیار کوچکx ، مشتق بسیار کوچک می‌شود و درواقع شبکه دیگر آموزش نمی‌بیند و پیش‌بینی‌هایش در خروجی ثابت می‌ماند. به‌دلیل مشکل محوشدگی‌گرادیان، تابع سیگموید هم‌گرایی کند دارد. خروجی تابع سیگموید صفرمحور (Zero-Centered)نیست؛ این امر کارایی به‌روزرسانی وزن‌ها را کم می‌کند. از آنجا که این تابع عملیات نمایی (Exponential Operations) دارد، می‌توان گفت هزینه‌ی محاسباتی بالایی دارد و کندتر پیش می‌رود.

تابع Tanh :

مزایا : این تابع صفرمحور است؛ بنابراین به مدل کمک می‌کند تا مقادیر ورودی منفی، خنثی و مثبت داشته باشد؛ به‌عبارت دیگر، مقادیر منفی، به‌شدت منفی و مقادیر صفر در گراف تانژانت هایپربولیک نزدیک به صفر نگاشت می‌شوند. تابع و مشتق آن هر دو یکنواخت (Monotonic) هستند.

معایب : محوشدگی گرادیان و همگرایی کند

تابع Relu :

مزایا: از نظر محاسباتی بسیار کارآمد است و به شبکه اجازه می‌دهد به سرعت همگرا شود؛ زیرا رابطه‌ی آن خطی است و به همین دلیل، در مقایسه با تابع‌های سیگموید وTanh ، سریع‌تر است.

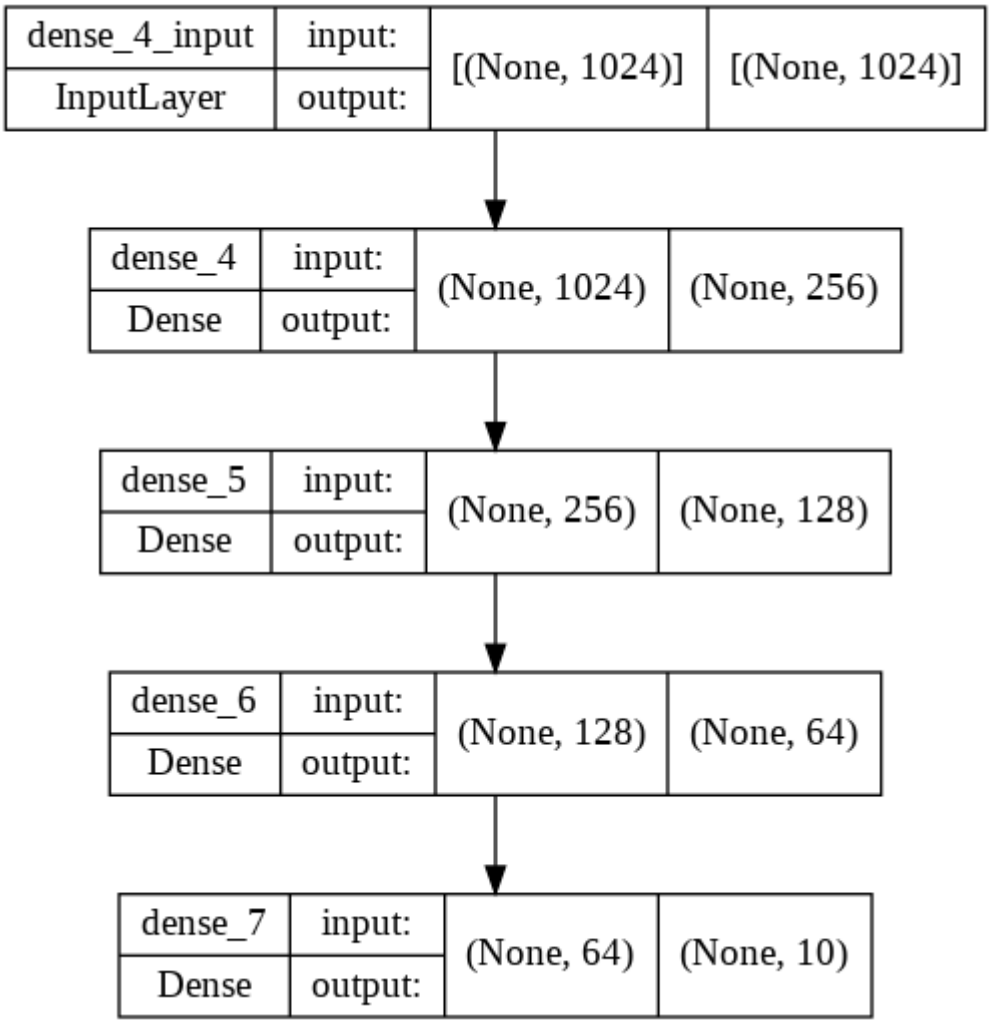
معایب: مشکل مرگ نورون یا مرگ Relu دارد؛ یعنی زمانی‌که ورودی صفر یا نزدیک به صفر باشد، تابع Relu دیگر عملکردی ندارد و به‌بیان دیگر، می‌میرد. در این صورت، مقدار گرادیان تابع صفر می‌شود و شبکه نمی‌تواند عملیات پس انتشار (Backpropagation) را انجام دهد و آموزش ببیند. خروجی این تابع صفر یا مثبت است و این یعنی صفرمحور نیست.

تابع softmax :

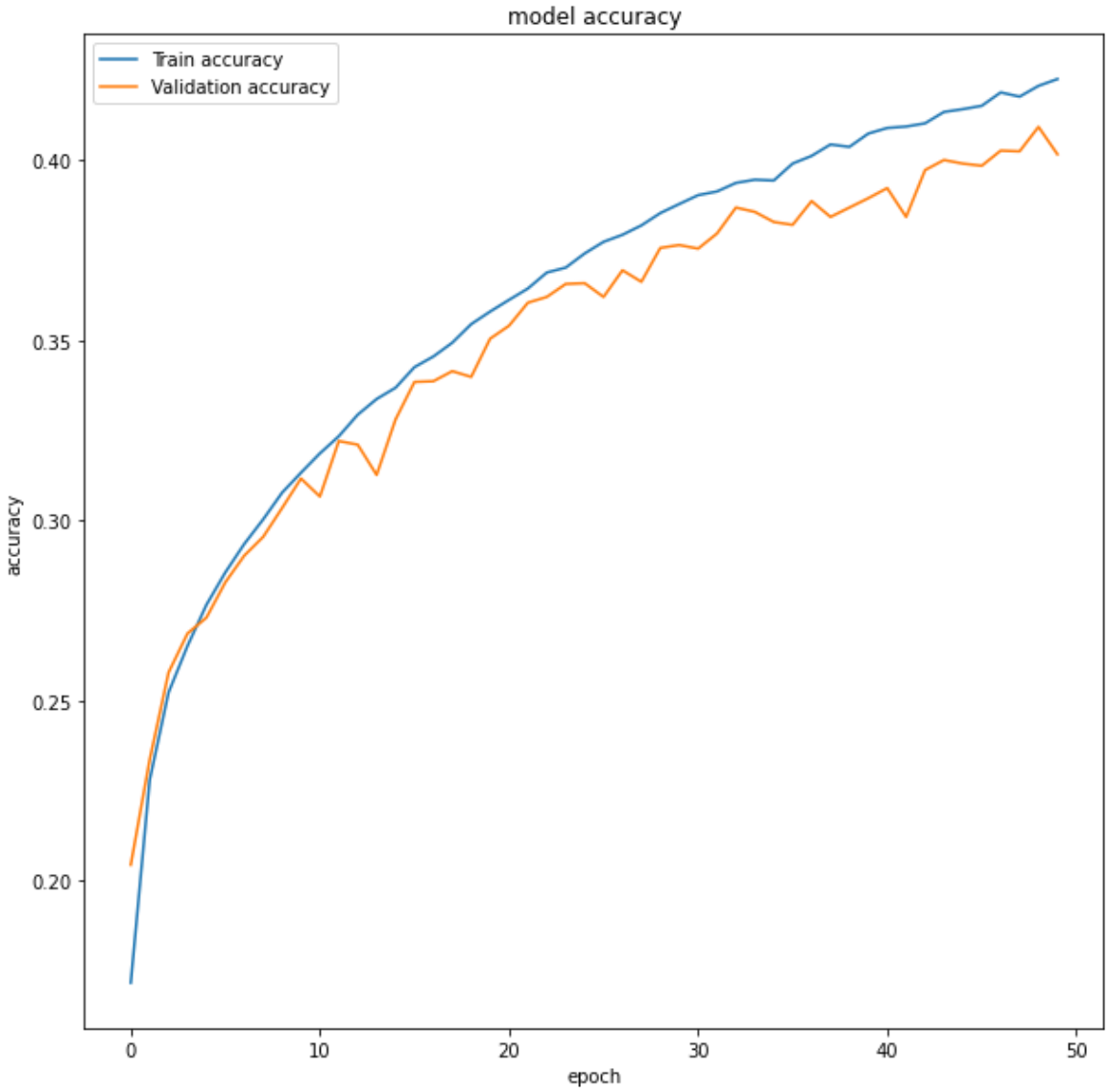
مزایا: این تابع قابلیت استفاده در تسک های چندکلاسه را دارد. خروجی هر کلاس را میان صفر تا ۱ عادی‌سازی می‌کند؛ سپس آن‌ها را بر مجموعه‌شان تقسیم و احتمال عضویت مقادیر ورودی را در هر کلاس به ما در خروجی ارائه می‌کند.

معایب : مقدار گرادیان برای مقادیر منفی صفر است؛ به‌این معنا که وزن‌ها در حین عملیات پس‌انتشار به‌روزرسانی نمی‌شوند و این می‌تواند مشکل مرگ نورون را ایجاد کند.

بهترین مدل قسمت قبل با تابع relu بود که نتایج زیر را داشت :



شکل 33. معماری شبکه با نورون‌های 256، 128، 64 و 10(تابع relu)



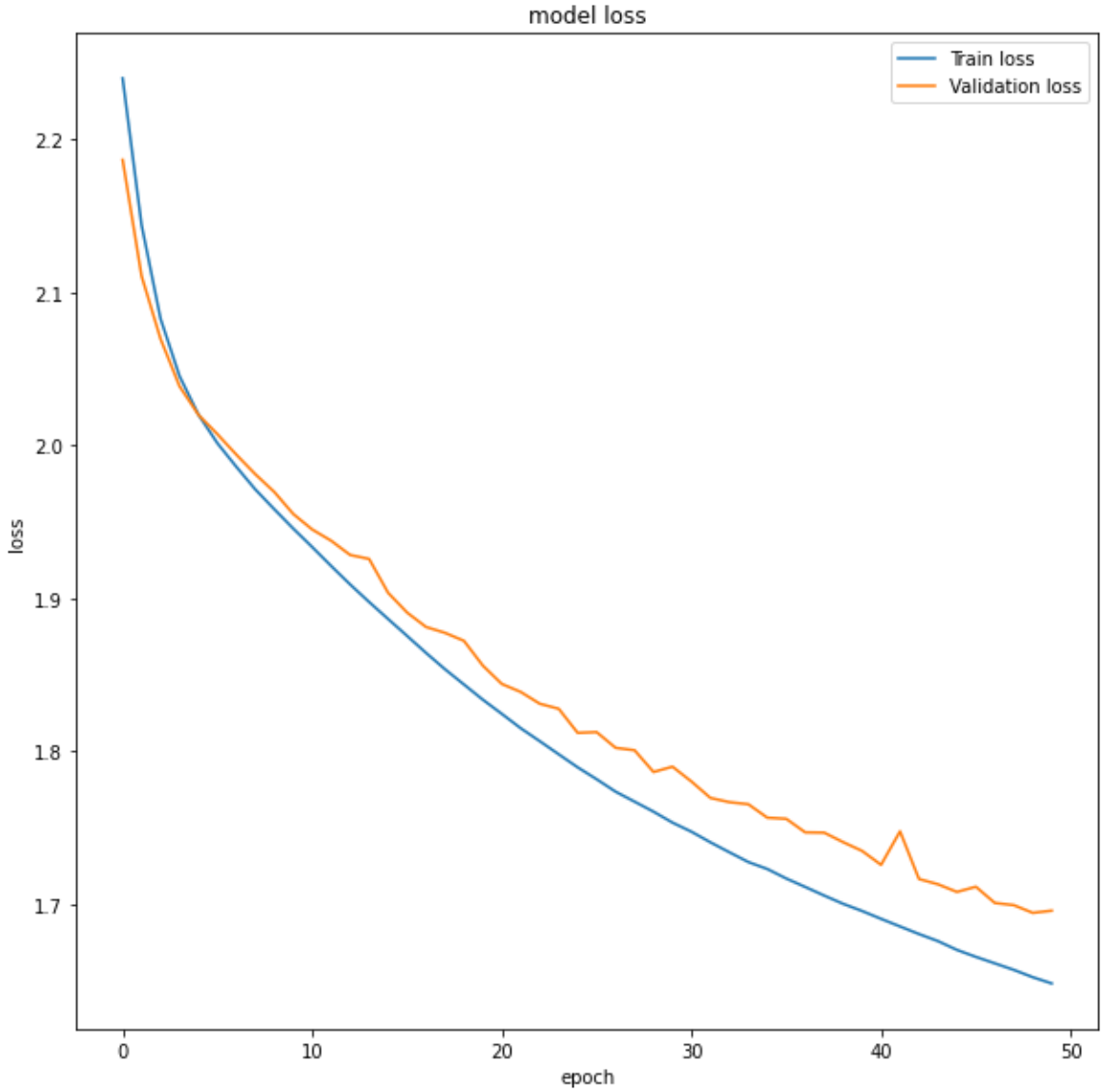
شکل 34. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10 (تابع relu)

مقدار f1-score، recall و precision :

recall is : 0.3971

precision is : 0.400682519972351

f1 is : 0.3955143451115557



شکل 35. **نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10 تابع**(relu)

خطا، دقت و زمان آموزش برای داده تست :

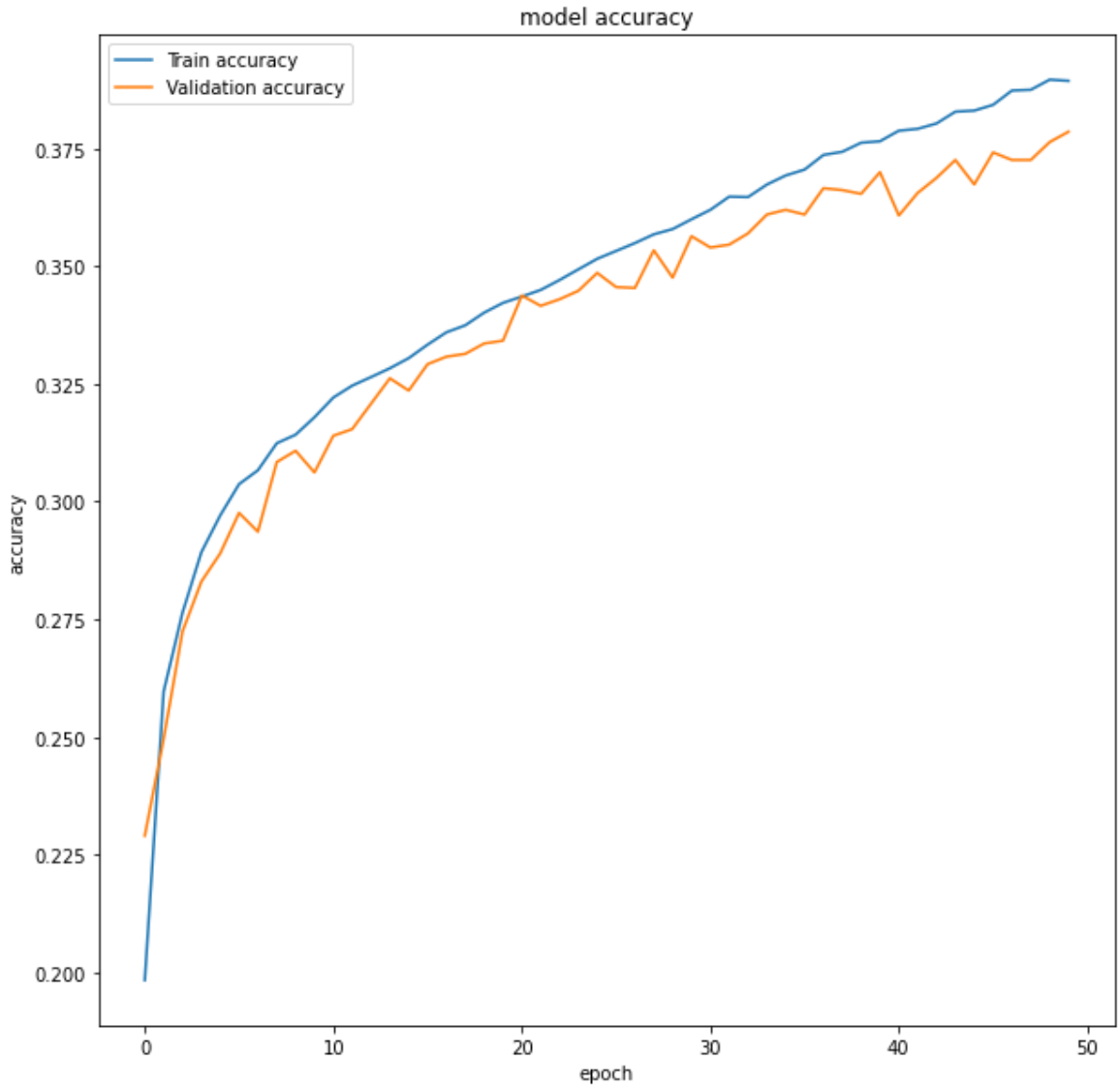
313/313 [==============================] - 1s 2ms/step - loss: 1.6965 - accuracy: 0.3971

loss in test data is: 1.6964564323425293

accuracy in test data is : 0.3971000015735626

Training time is : 0.05189657211303711

تابع tanh :



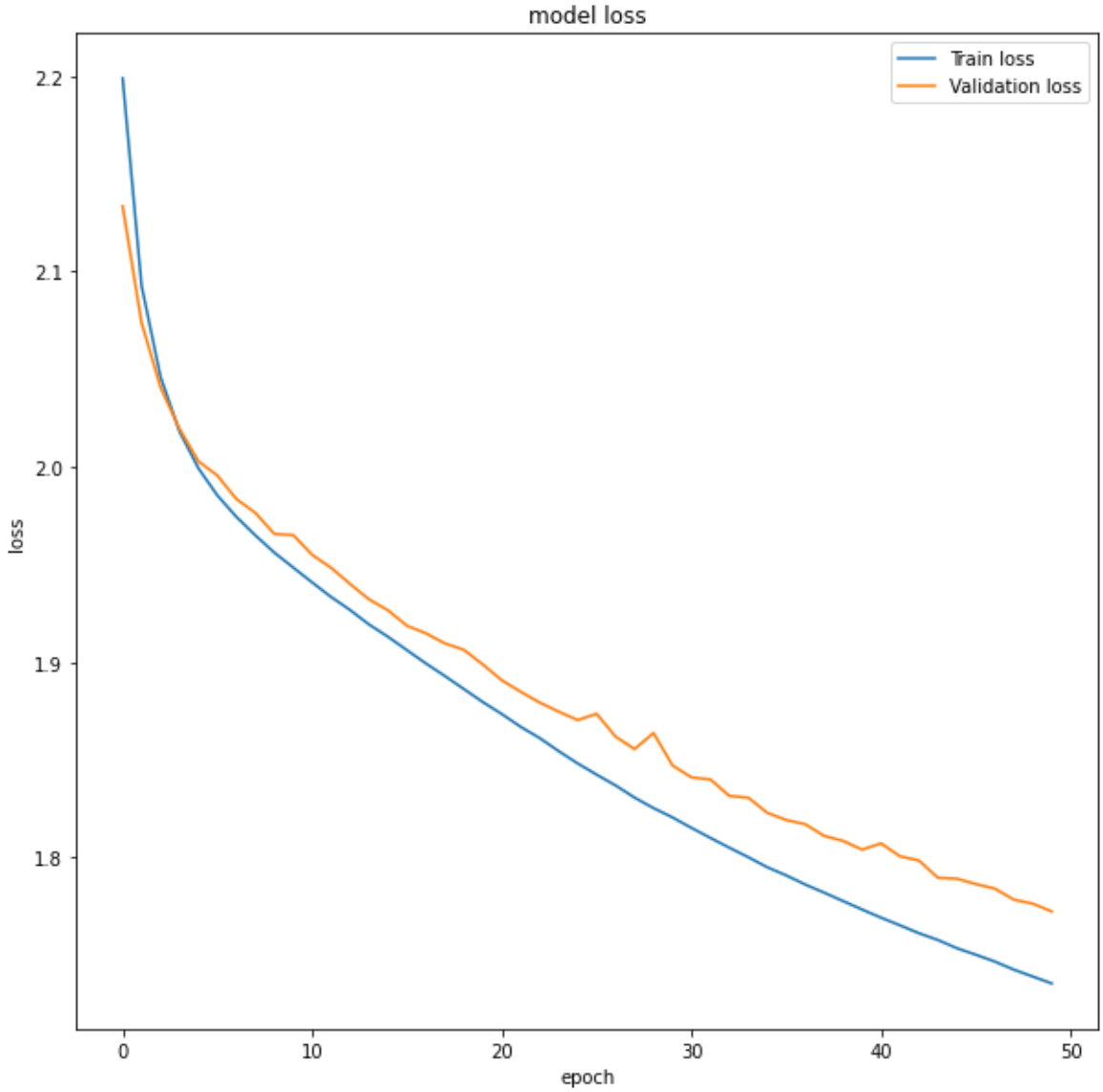
شکل 36. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision :

recall is : 0.3803

precision is : 0.3795621515061469

f1 is : 0.37558529481420566



شکل 37. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

خطا، دقت و زمان آموزش برای داده تست :

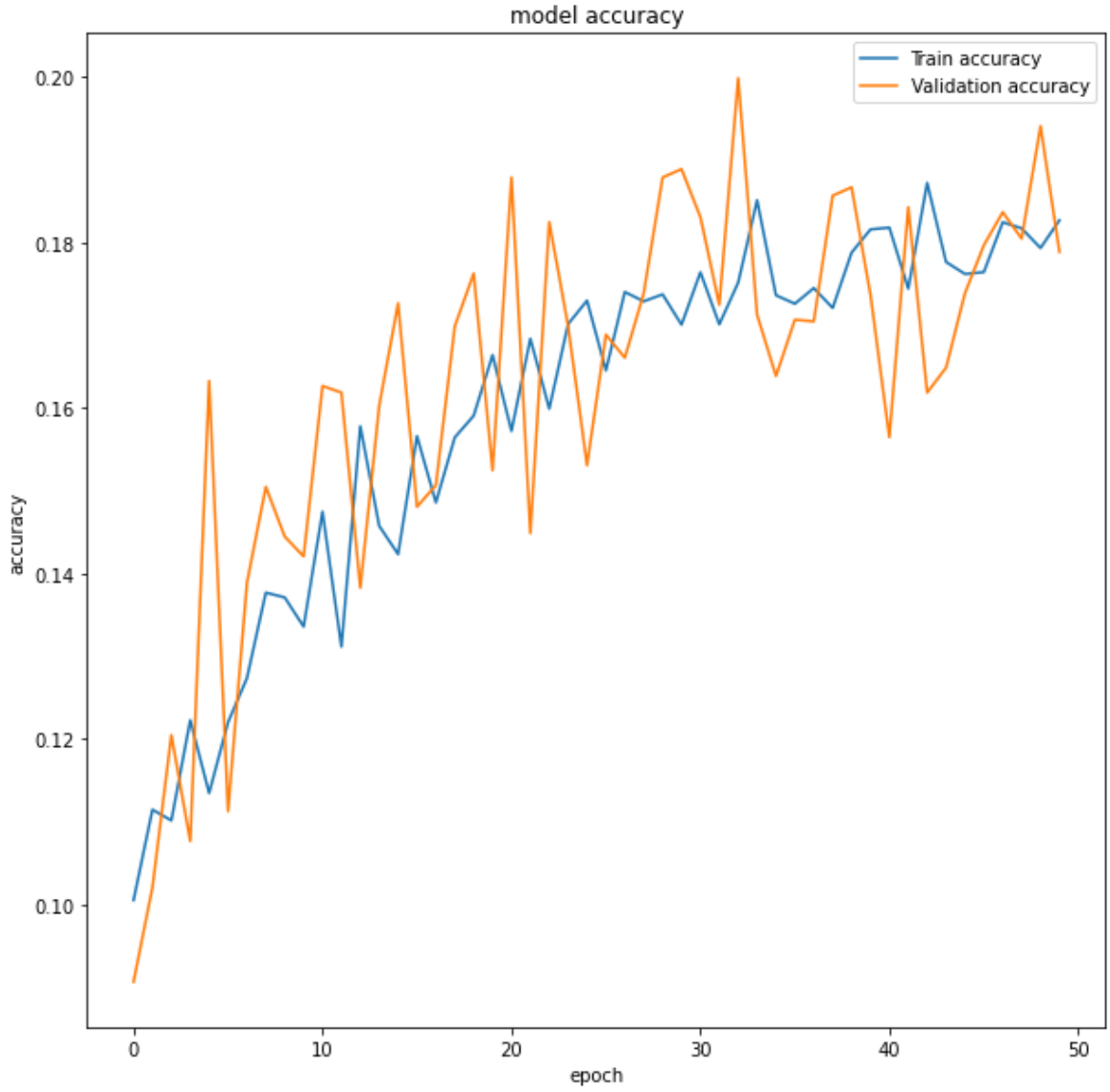
313/313 [==============================] - 2s 5ms/step - loss: 1.7631 - accuracy: 0.3803

loss in test data is: 1.763092041015625

accuracy in test data is : 0.38029998540878296

Training time is : 0.07146096229553223

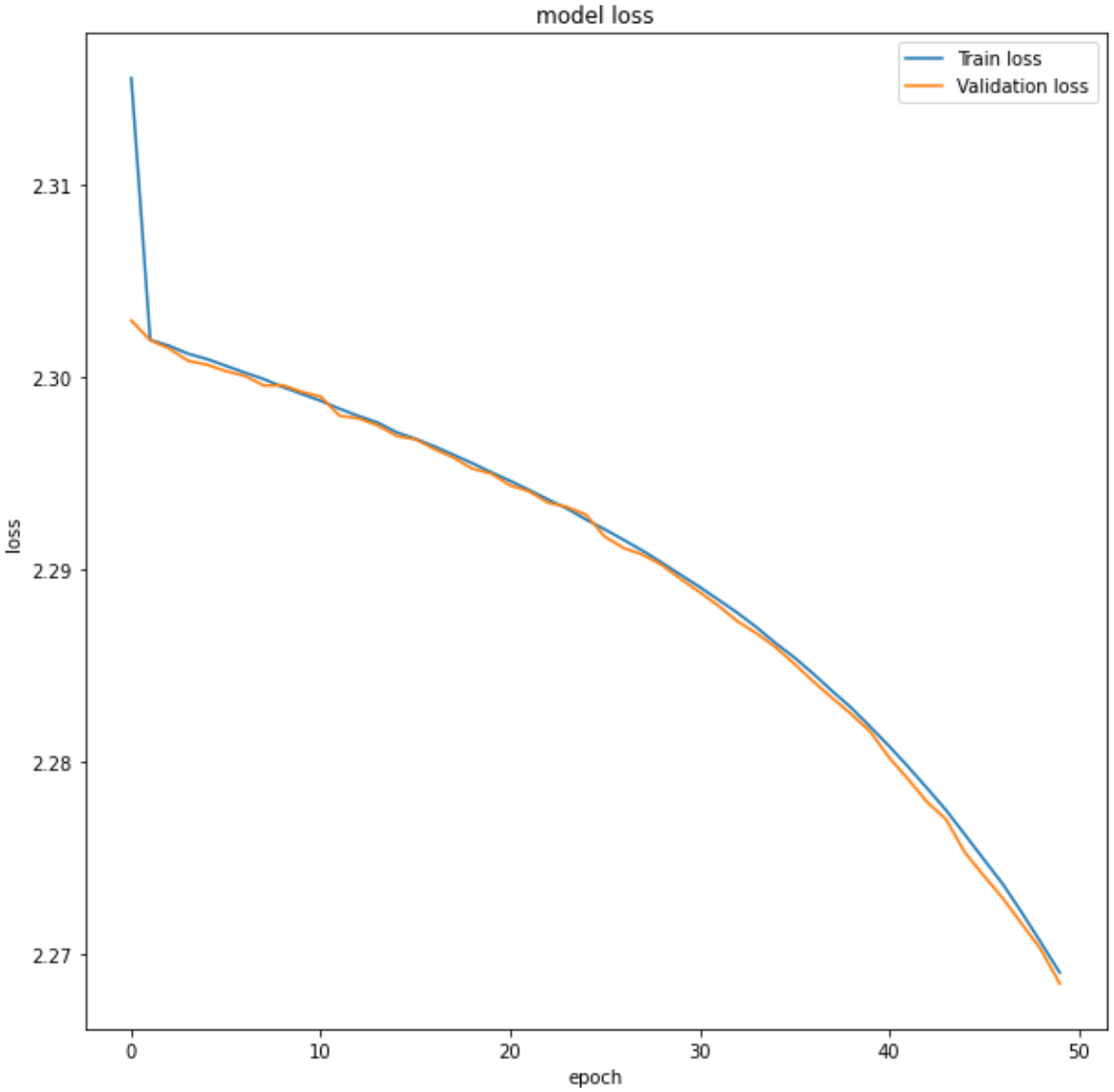
تابع sigmoid:



شکل 38. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision:

```
recall is : 0.1799

precision is : 0.20083028984851273

f1 is : 0.14494950732953957
```

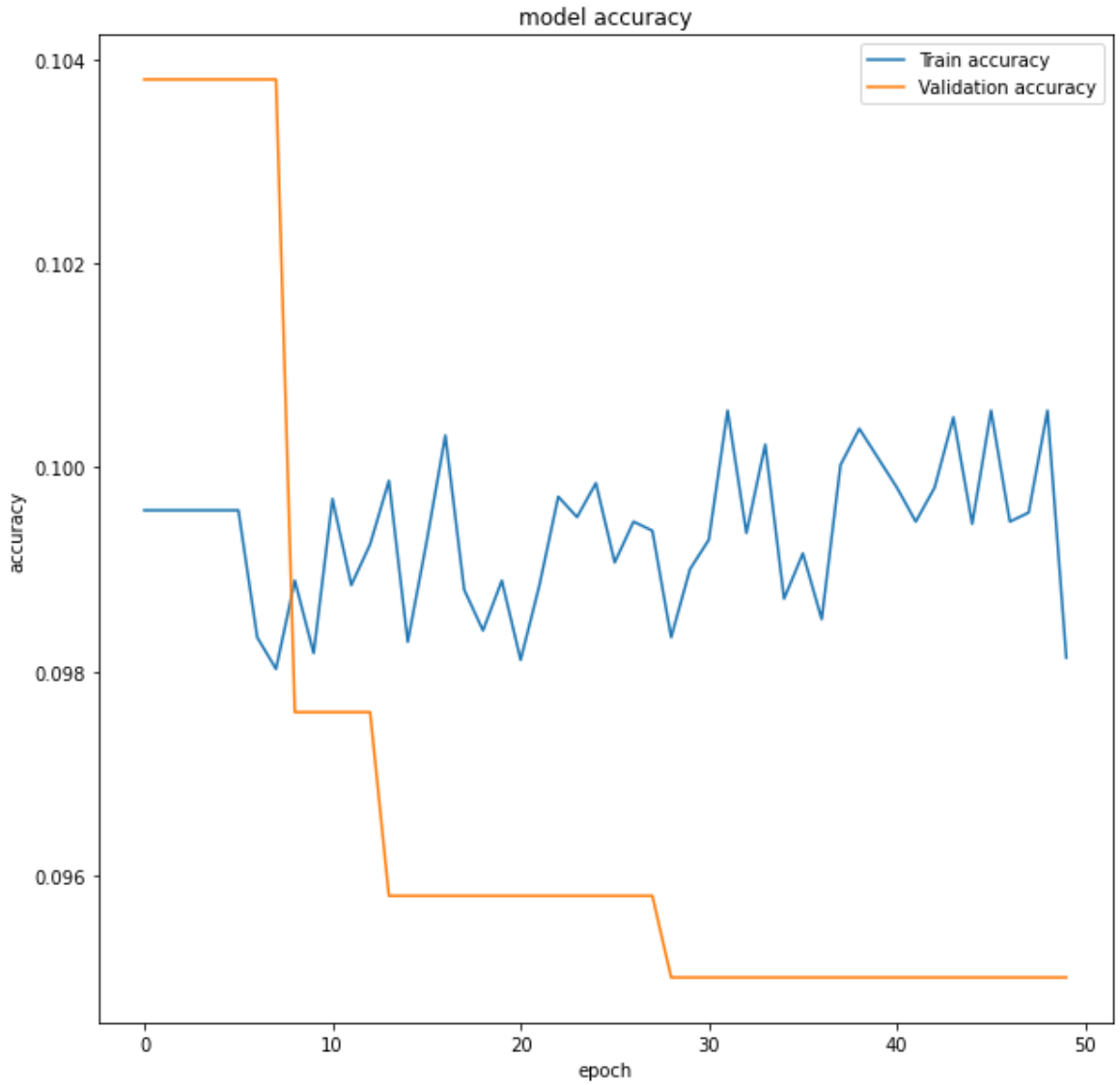


شکل 39. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 2ms/step - loss: 2.2681 - accuracy: 0.1799

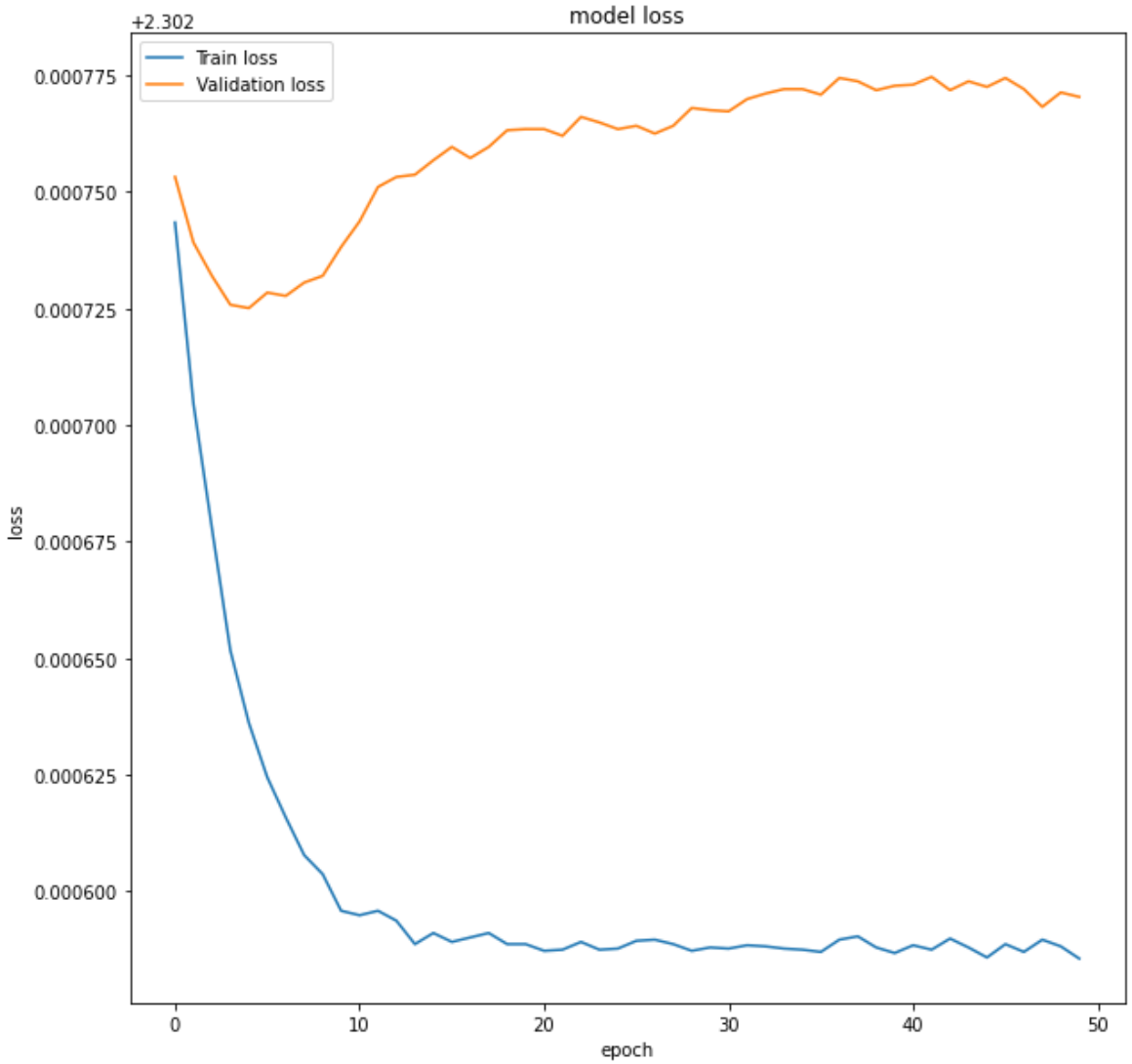
loss in test data is: 2.268113851547241
accuracy in test data is : 0.17990000545978546
Training time is : 0.14916658401489258



شکل 40. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10

مقدار f1-score، recall و precision :

```
recall is : 0.1

precision is : 0.01

f1 is : 0.018181818181818
```



شکل 41. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

خطا، دقت و زمان آموزش برای داده تست :

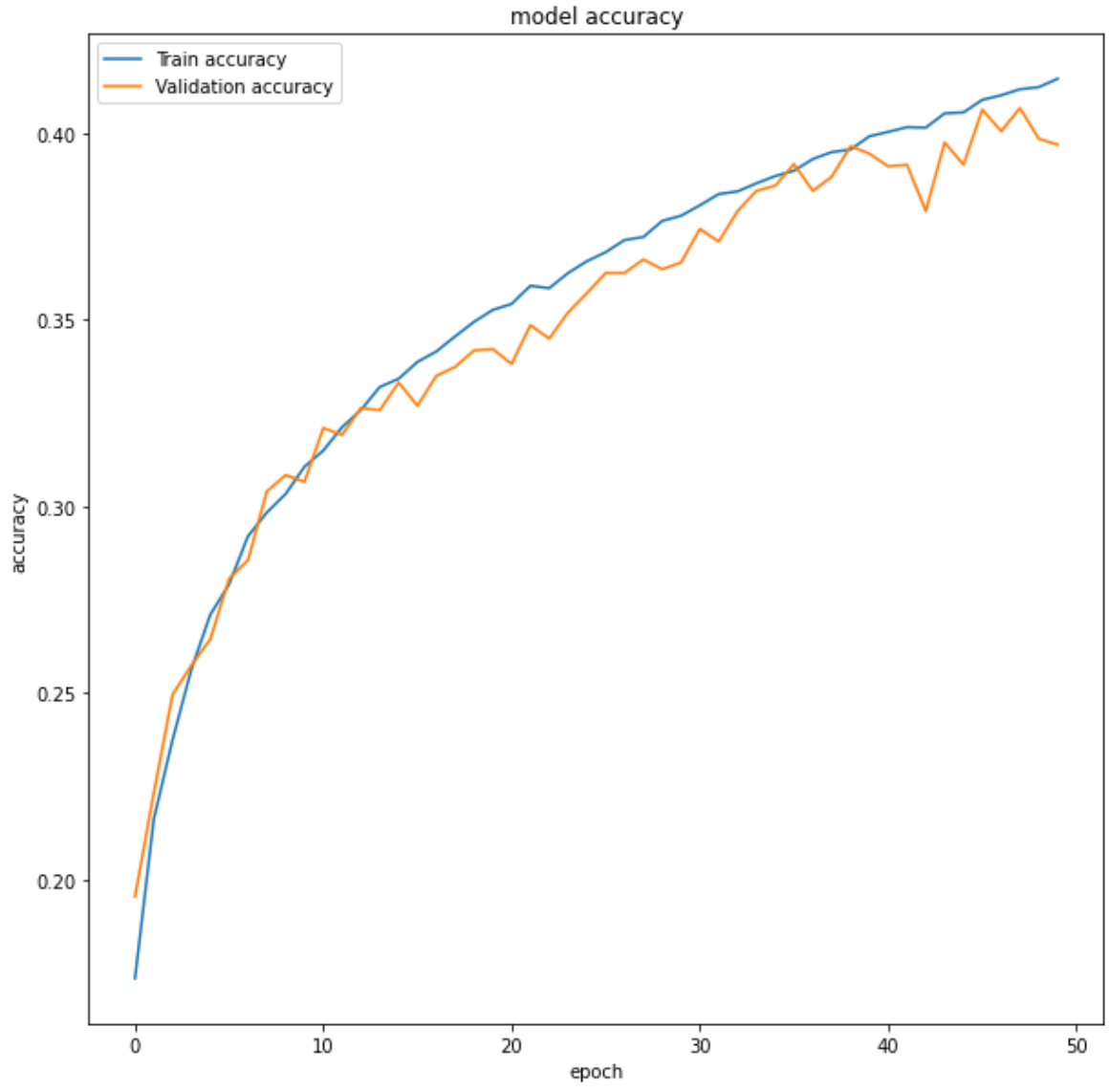
313/313 [==============================] - 1s 3ms/step - loss: 2.3026 - accuracy: 0.1000

loss in test data is: 2.302591323852539
accuracy in test data is : 0.10000000149011612
Training time is : 0.05780601501464844

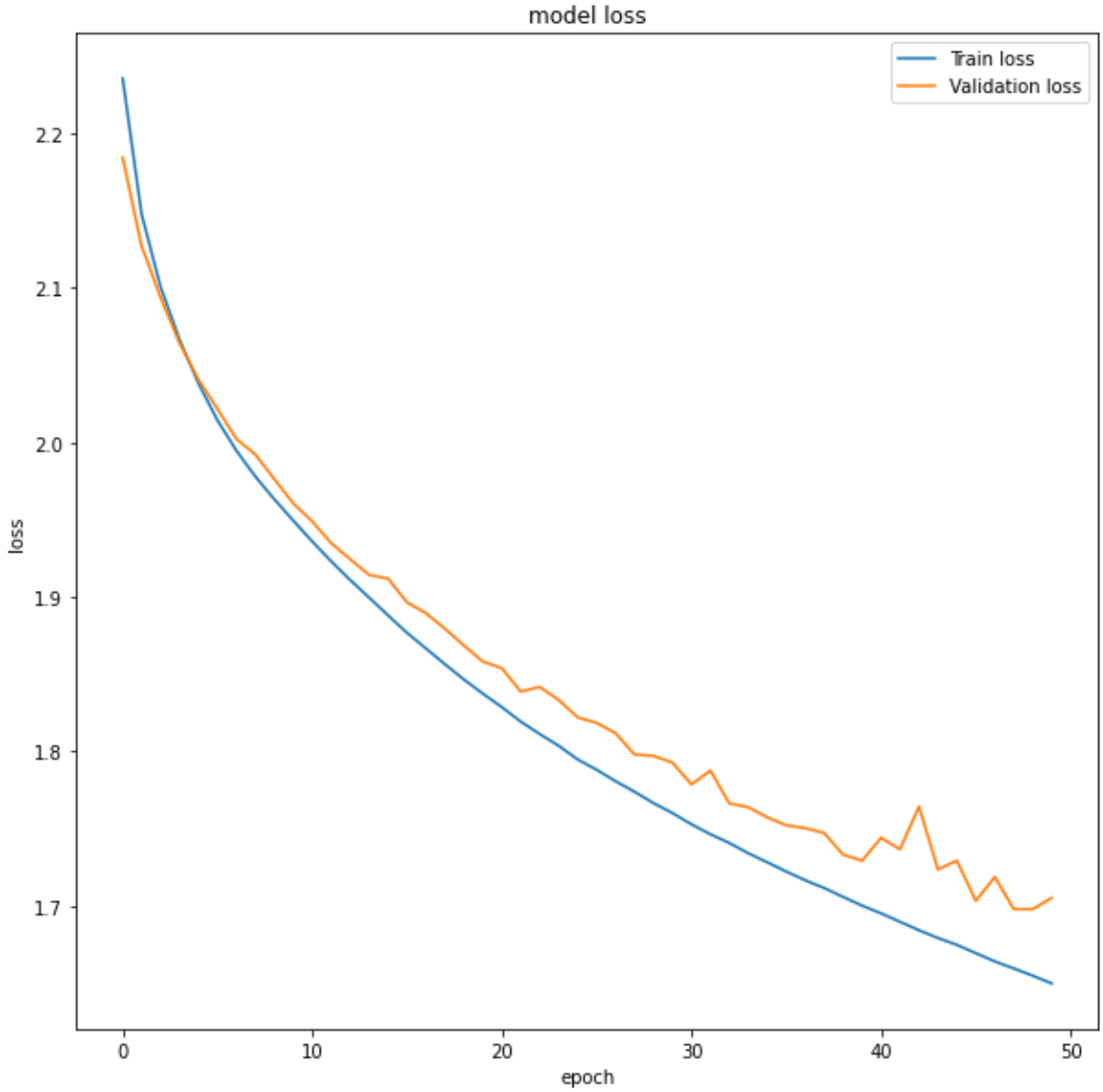
با توجه به نتایج حاصل شده زمان آموزش شبکه در تابع Relu از بقیه کمتر، خطا کمتر و دقت بیشتر است. بنابراین بهترین مدل به دست آمده براساس تابع Relu خواهد بود.

قسمت و)

Loss function = categorical_crossentropy



شکل 42. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



شکل 43. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

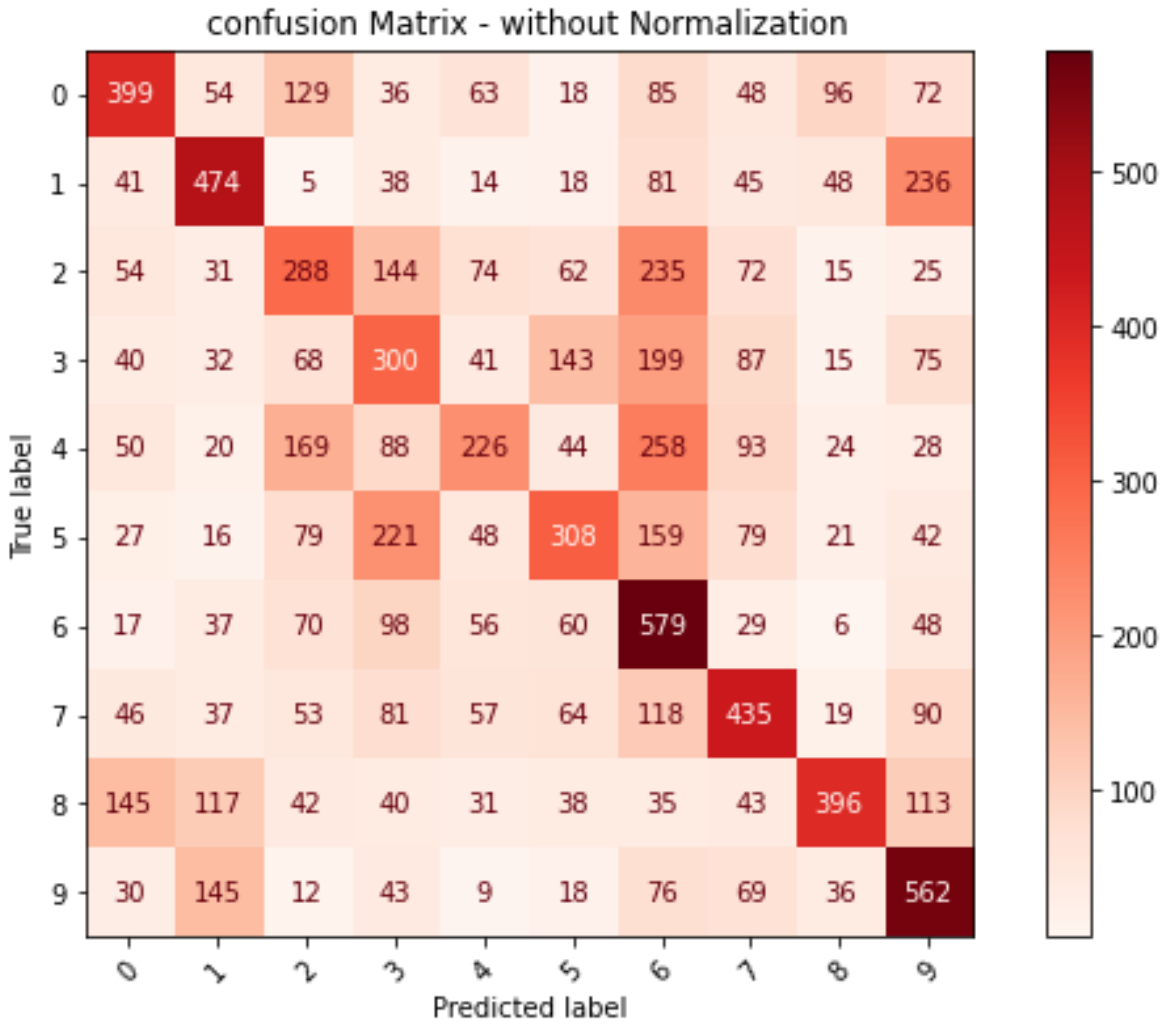
خطا، دقت و زمان آموزش برای داده تست :

313/313 [==============================] - 1s 2ms/step - loss: 1.6965 - accuracy: 0.3971

loss in test data is: 1.6964564323425293

accuracy in test data is : 0.3971000015735626

Training time is : 0.05189657211303711



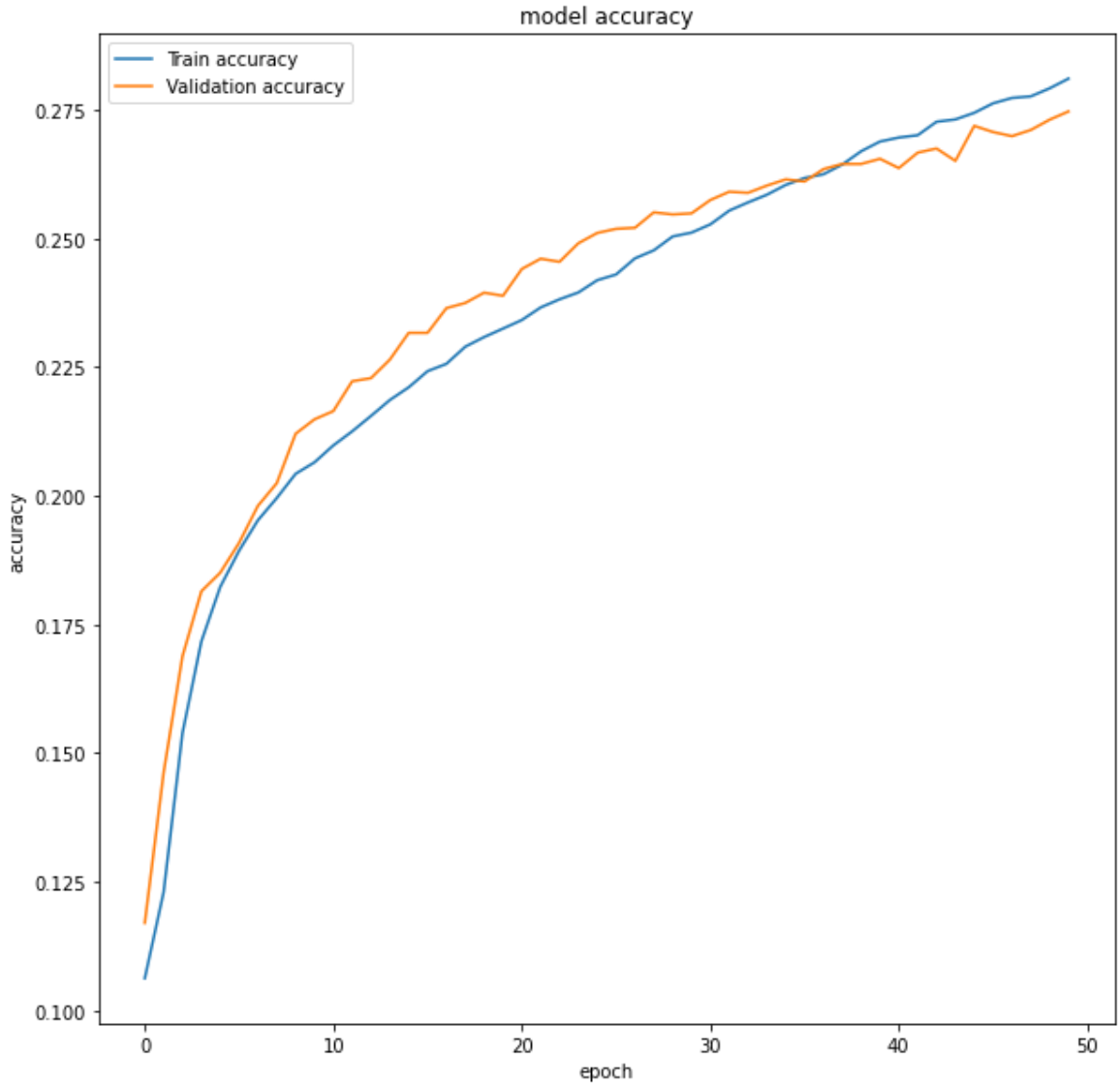
شکل 44. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

recall is : 0.3967

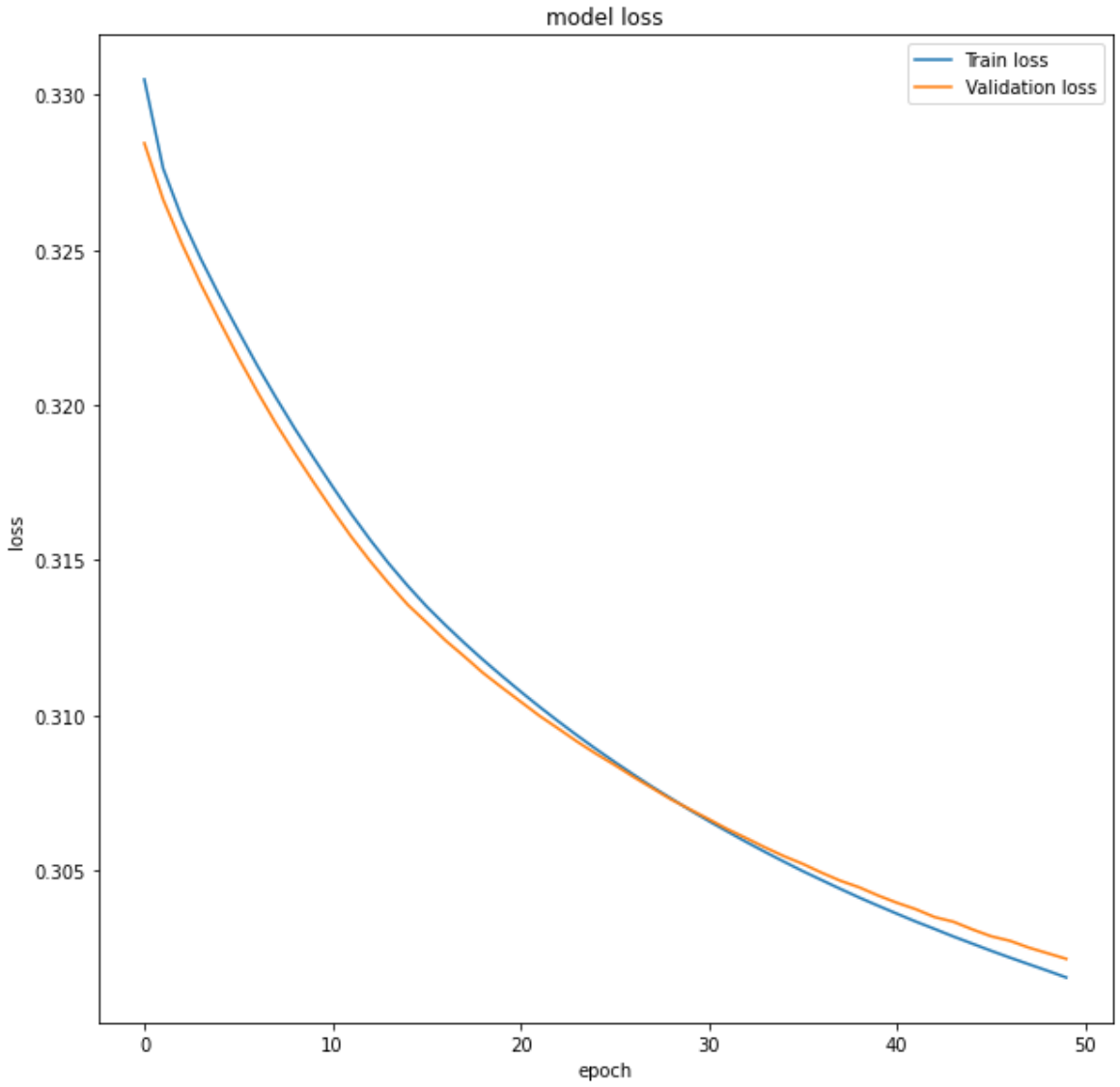
precision is : 0.40893458559224605

f1 is : 0.39372200449048356

loss function = poisson



شکل 45. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



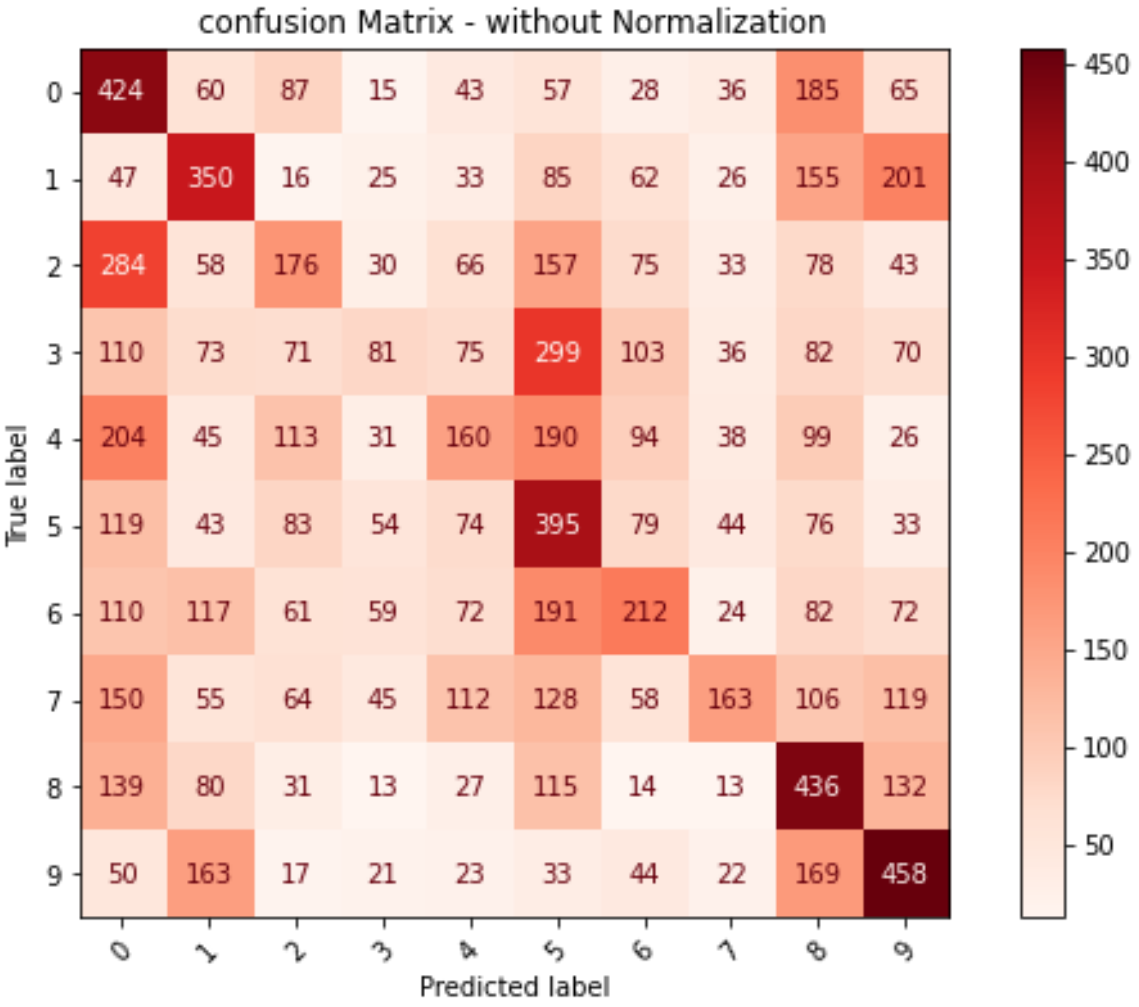
شکل 46. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

313/313 [==============================] - 2s 6ms/step - loss: 0.3017 - accuracy: 0.2855

loss in test data is: 0.30168625712394714

accuracy in test data is : 0.2854999899864197

Training time is : 0.05329442024230957



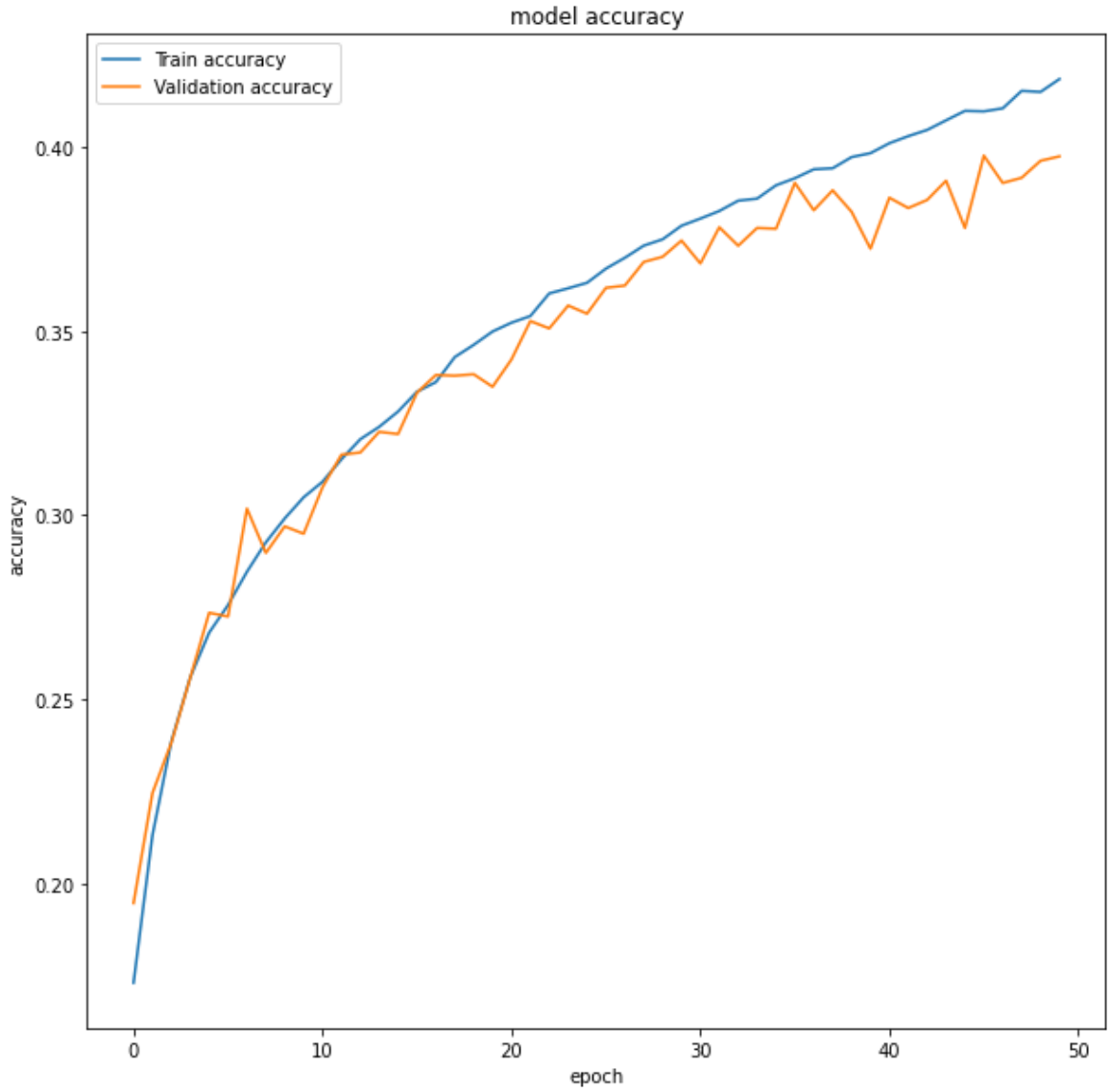
شکل 47. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

recall is : 0.2855

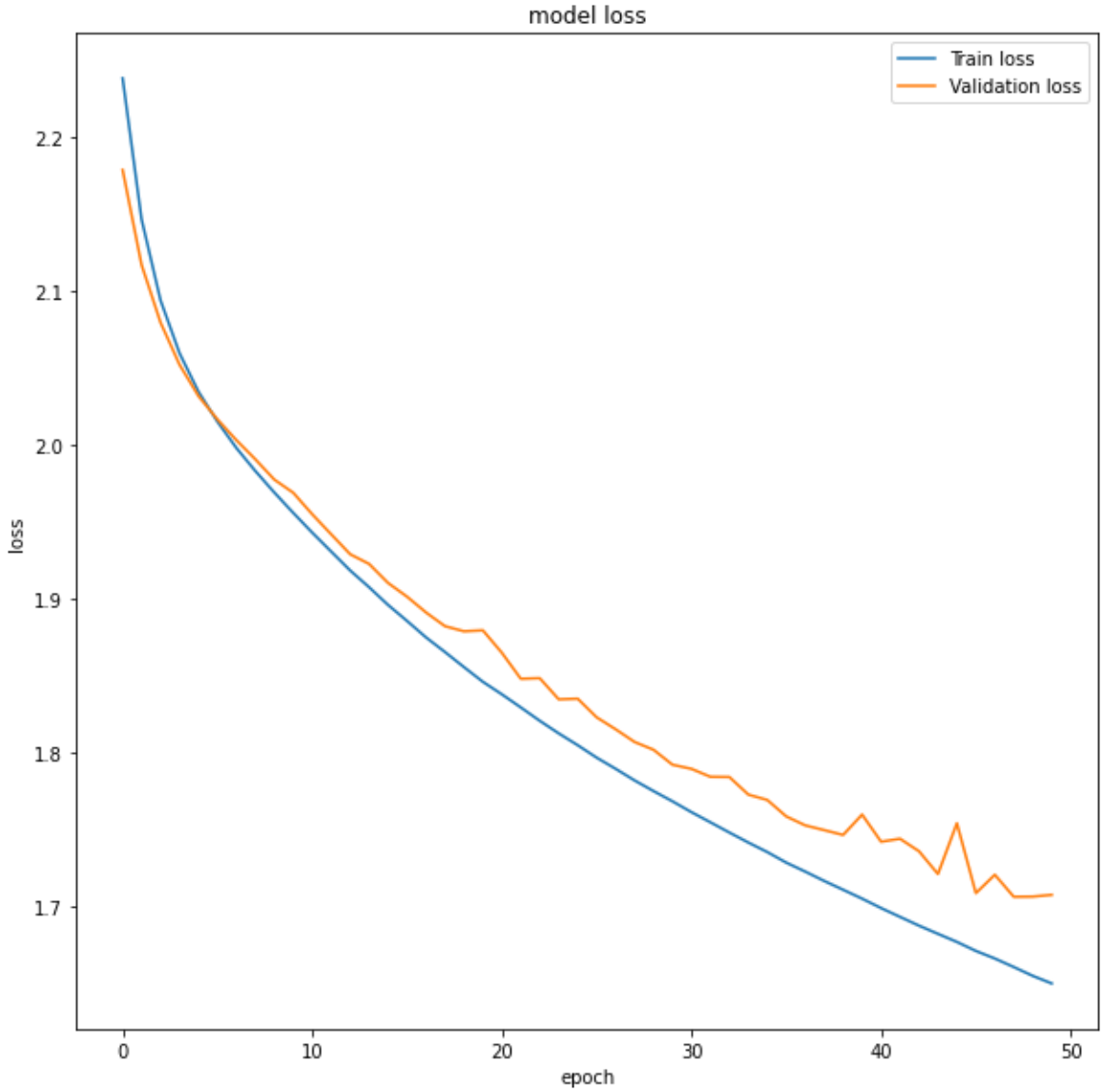
precision is : 0.285170784640391

f1 is : 0.27077238594097164

loss function = kl_divergence



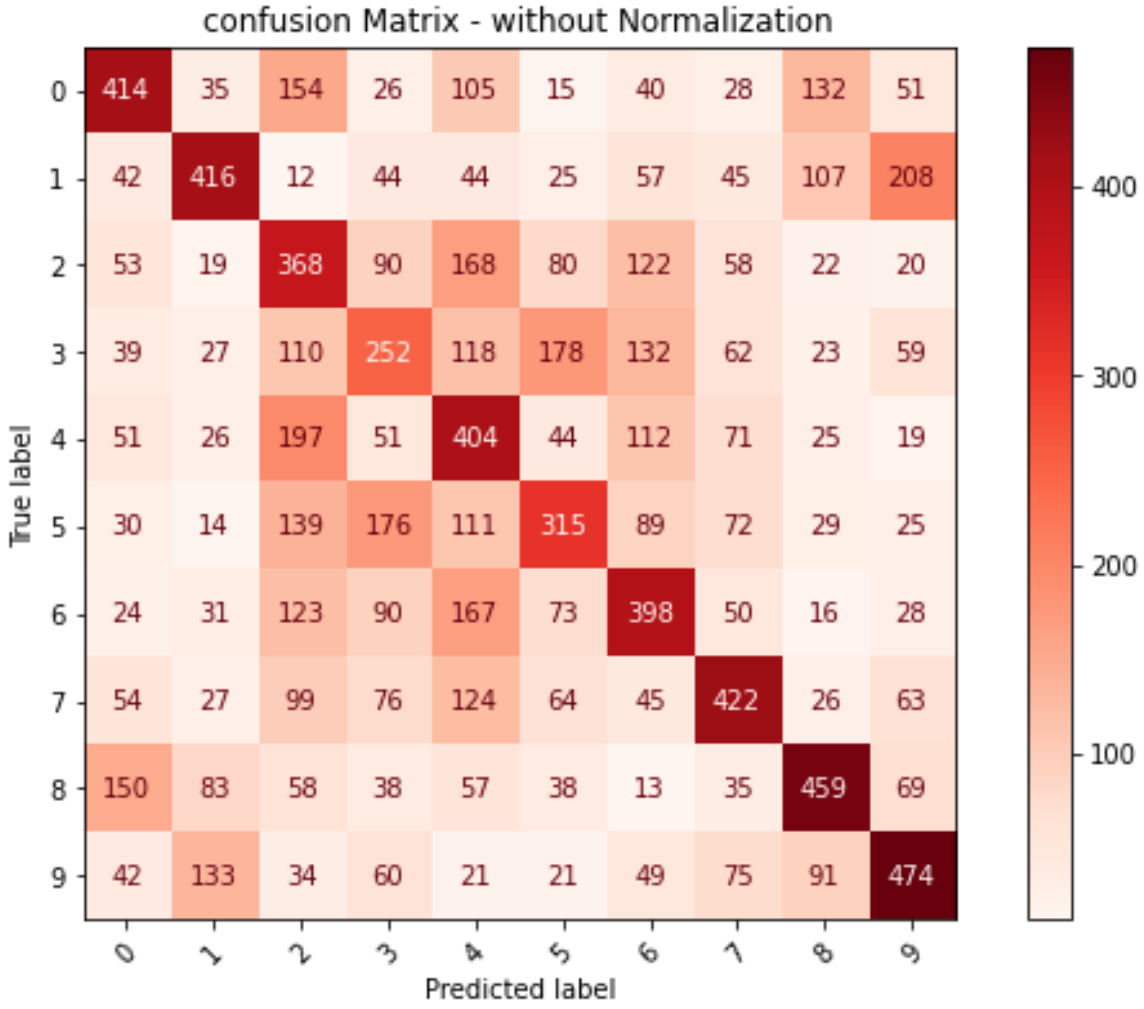
شکل 48. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



شکل 49. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

313/313 [==============================] - 1s 3ms/step - loss: 1.7014 - accuracy: 0.3922

loss in test data is: 1.7013591527938843
accuracy in test data is : 0.3921999931335449
Training time is : 0.05546736717224121



شکل 50. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

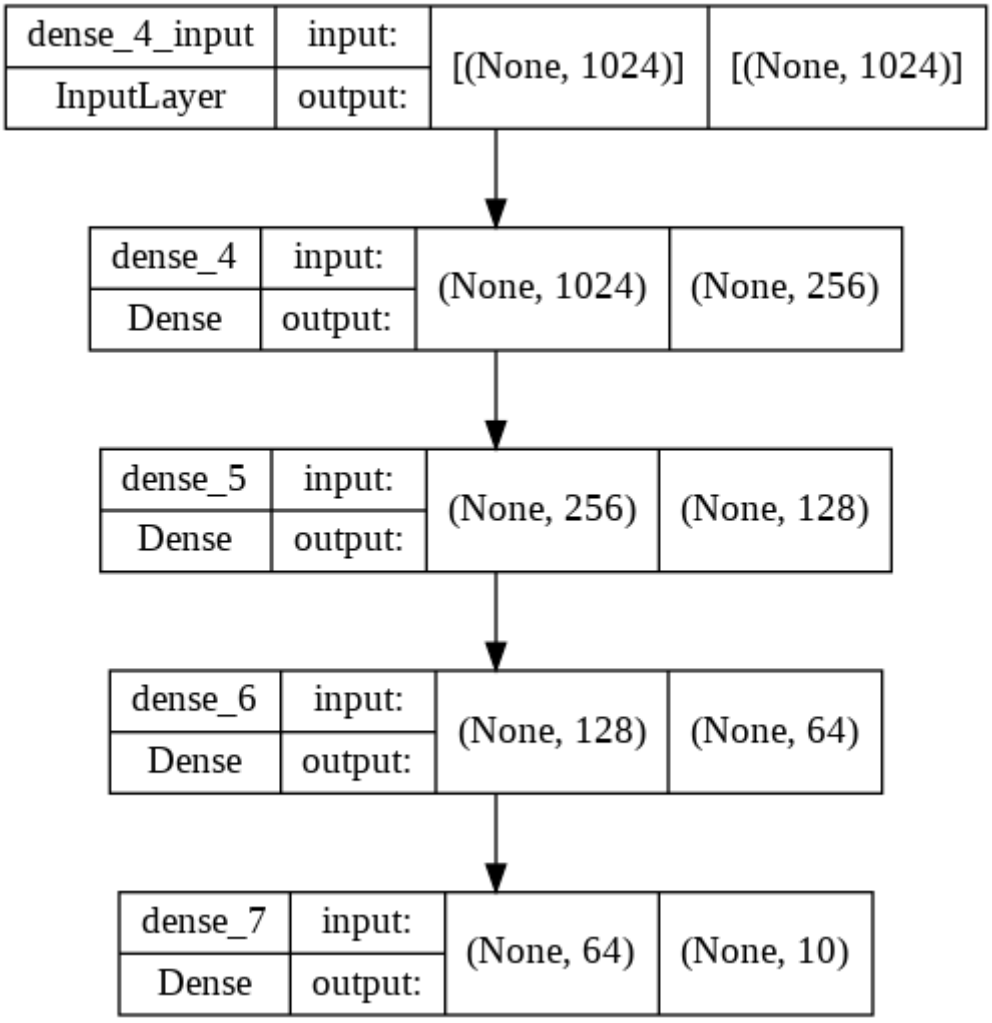
recall is : 0.3922

precision is : 0.4008811618115951

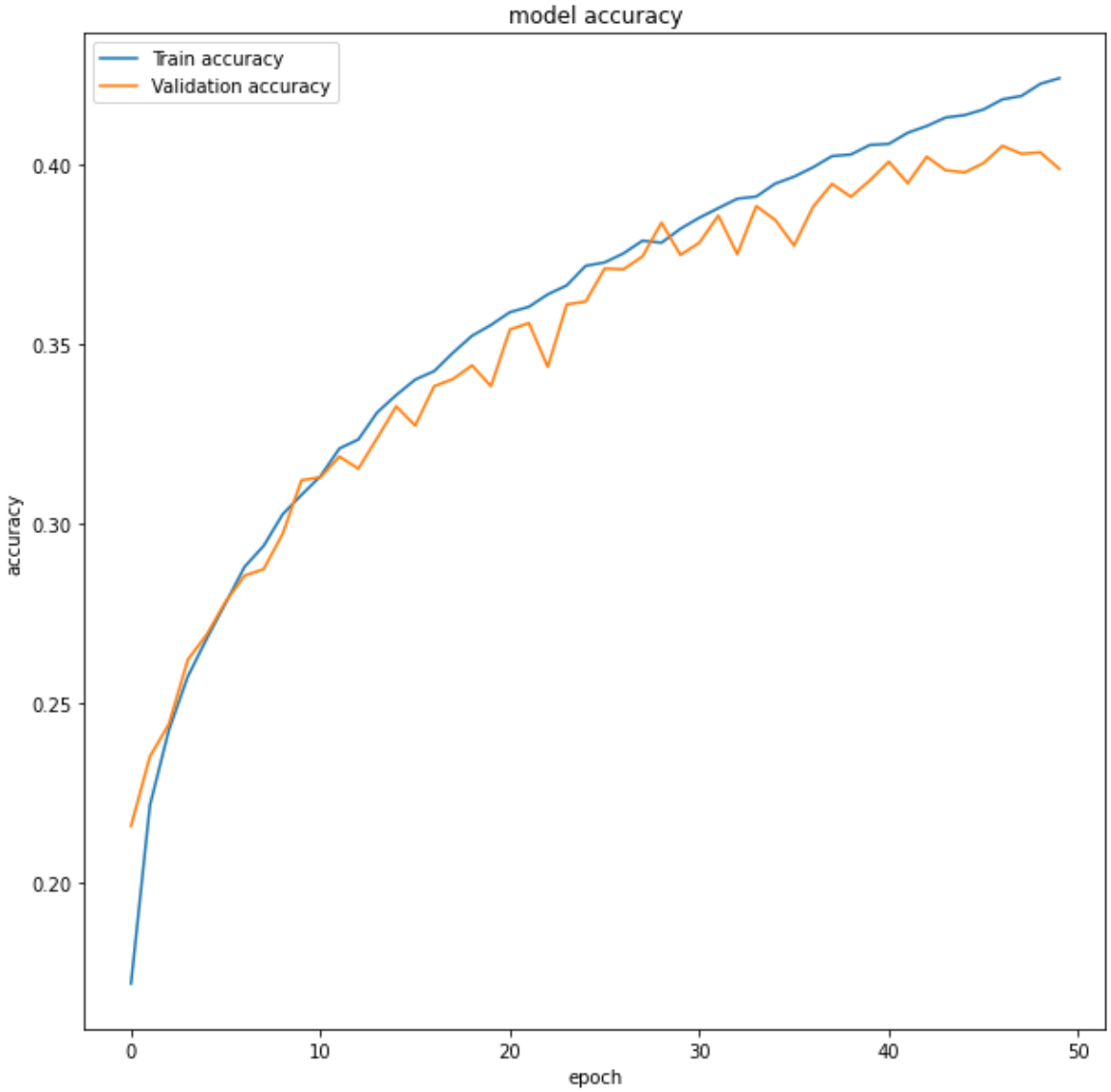
f1 is : 0.3942429636570275

با توجه به مقادیر خطا و دقت و زمان آموزش سه تابع خطا، میزان دقت در تابع poisson کمتر از بقیه است ولی میزان خطا نیز کمتر است. اگر بخواهیم یک trade off بین دقت و خطا در نظر بگیریم تابع cross_entropy بهتر از بقیه است.

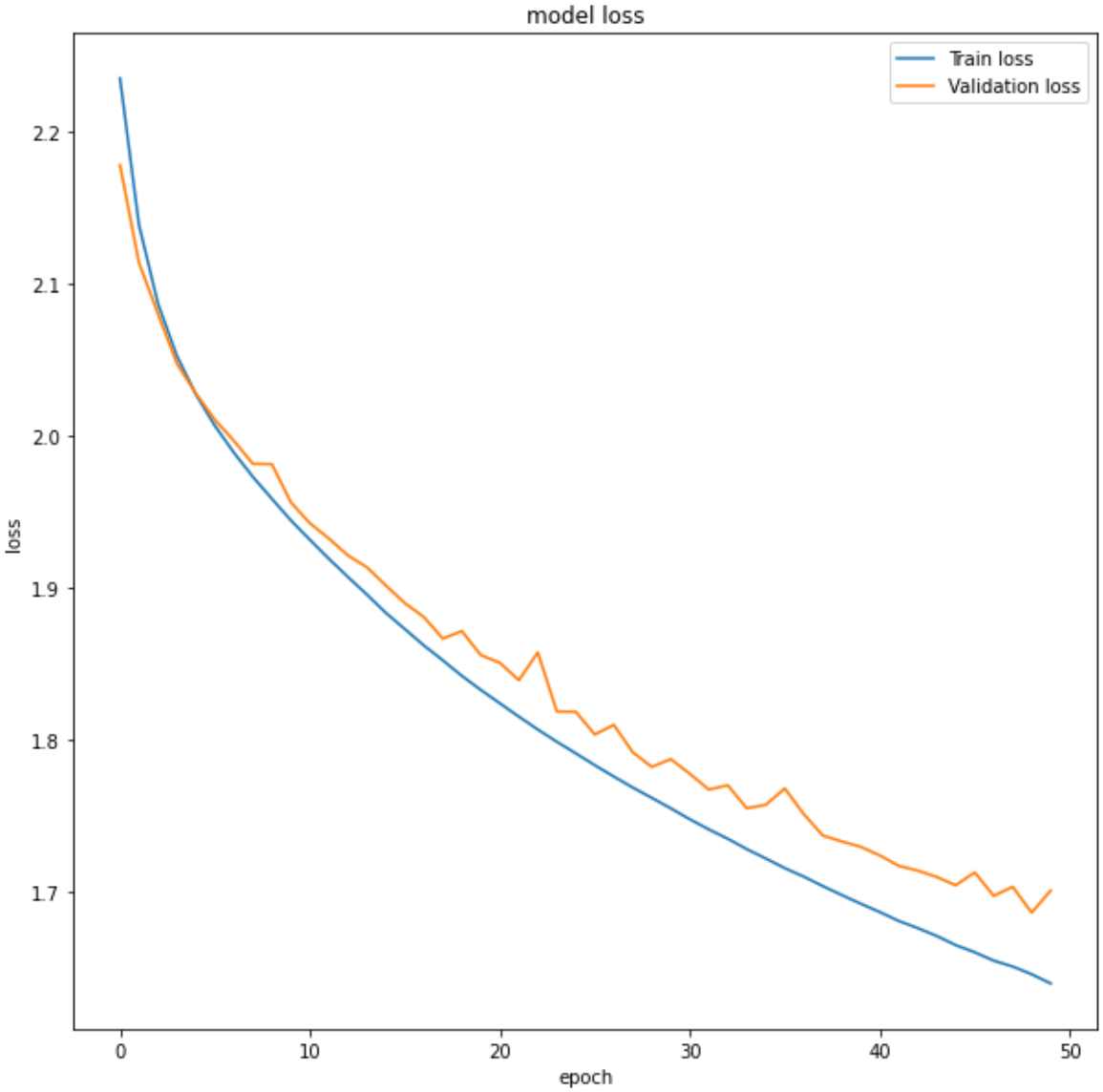
قسمت ج) *optimizer = SGD*



شکل 51. معماری شبکه با نورون‌های 256، 128، 64 و 10



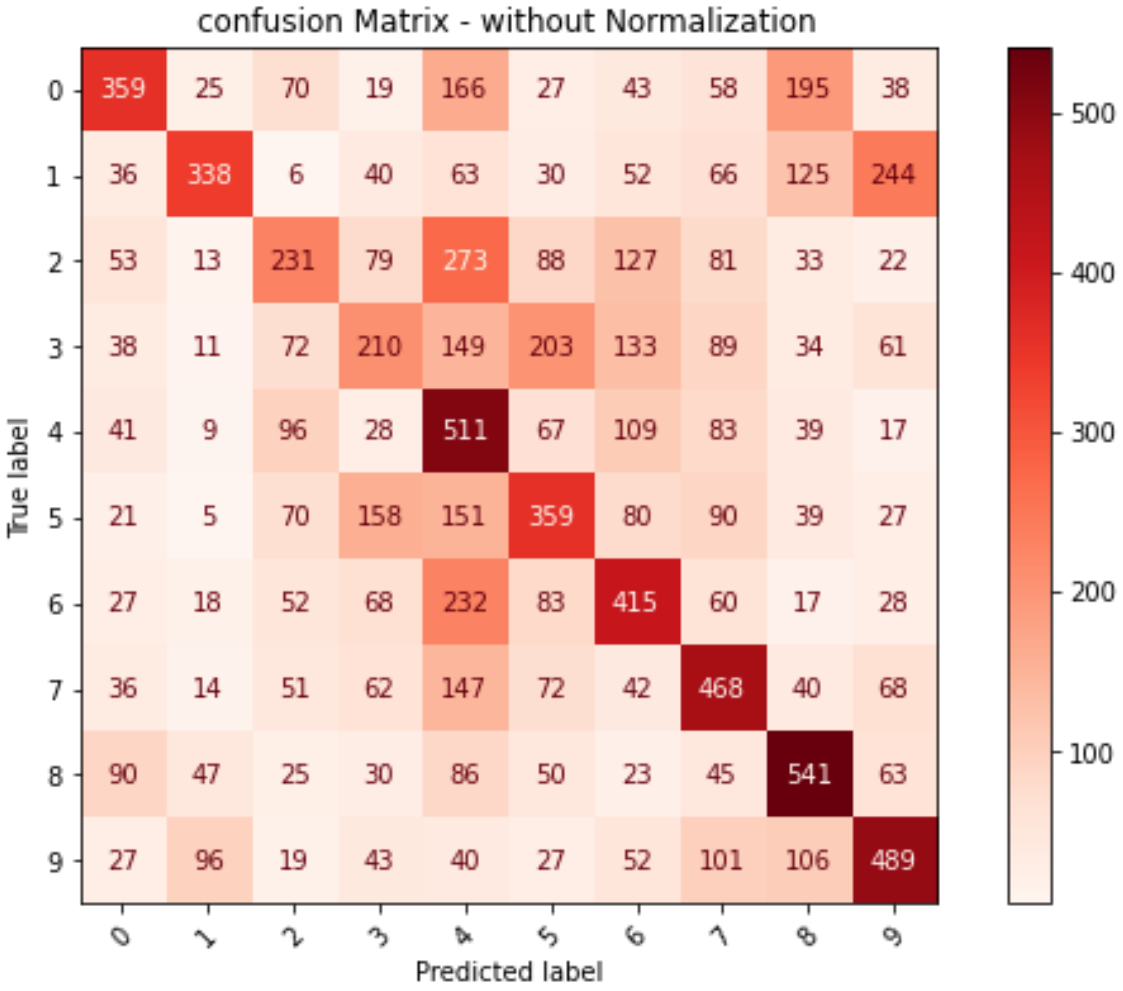
شکل 52. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



شکل 53. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

313/313 [==============================] - 1s 2ms/step - loss: 1.7041 - accuracy: 0.3921

loss in test data is: 1.7040538787841797
accuracy in test data is : 0.3921000063419342
Training time is : 0.14713621139526367



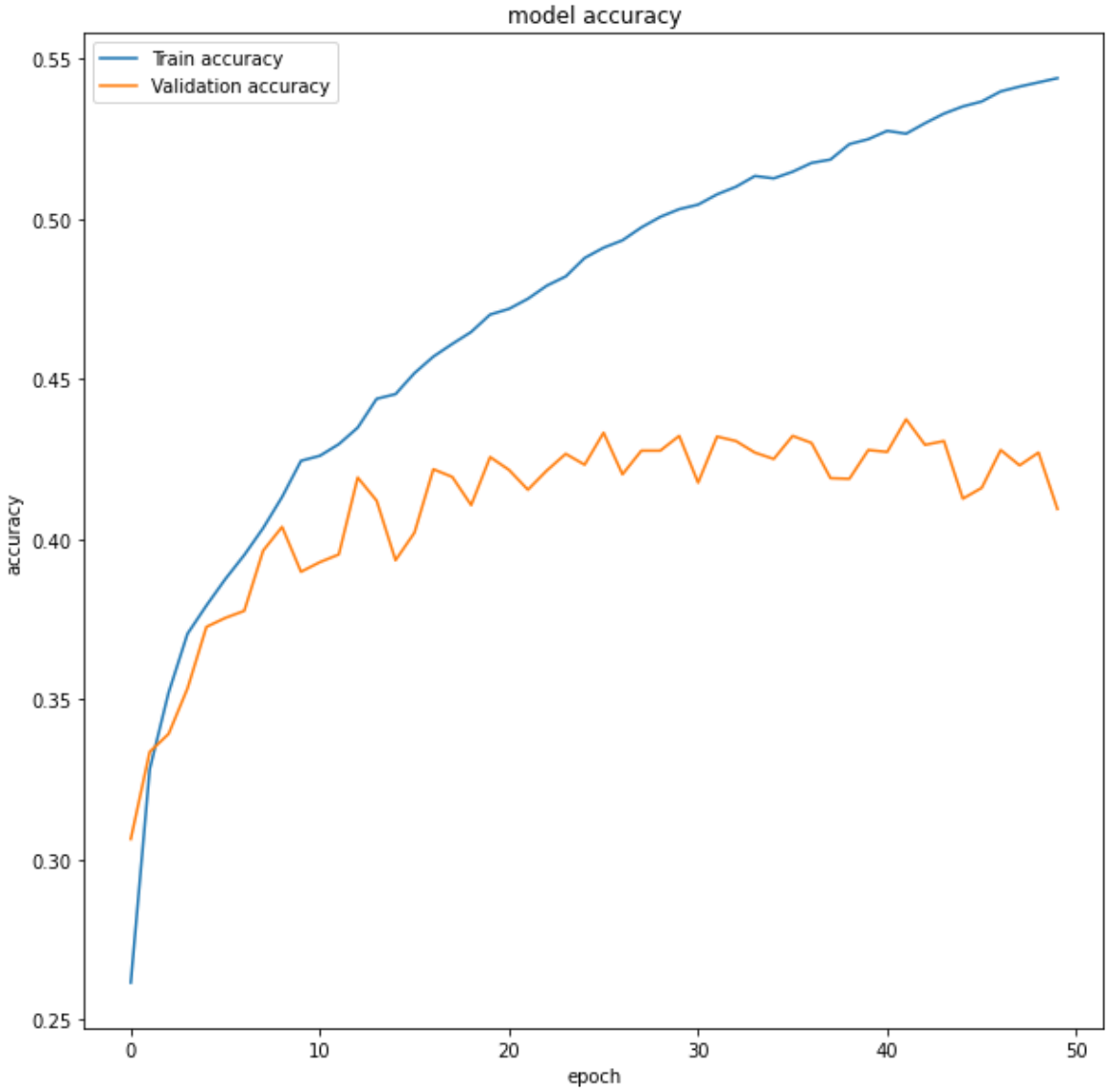
شکل 54. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

recall is : 0.3921

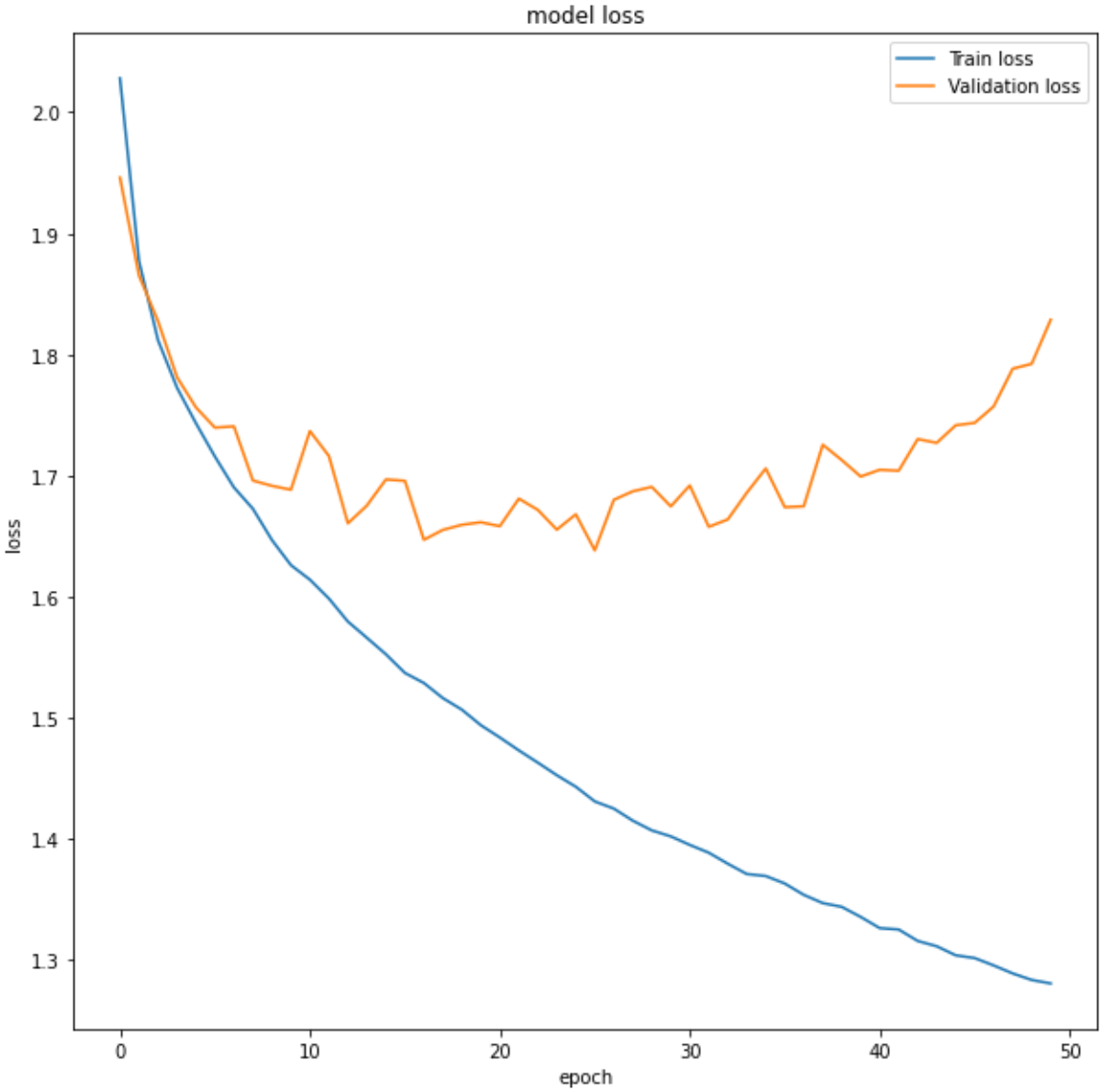
precision is : 0.40579013953560955

f1 is : 0.3891167185006267

optimizer = Adam



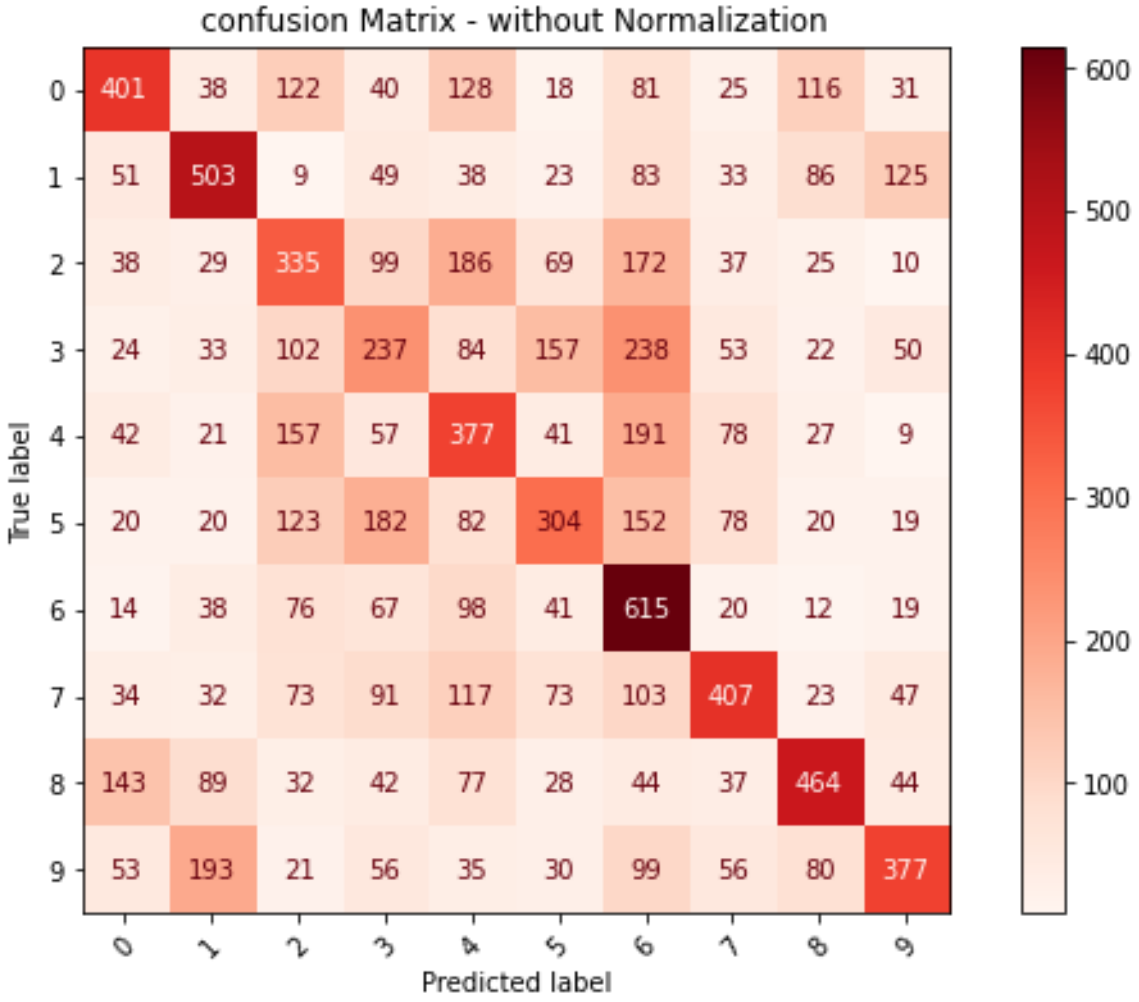
شکل 55. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



شکل 56. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

313/313 [==============================] - 1s 2ms/step - loss: 1.8397 - accuracy: 0.4020

loss in test data is: 1.8396515846252441
accuracy in test data is : 0.4020000100135803
Training time is : 0.06187009811401367



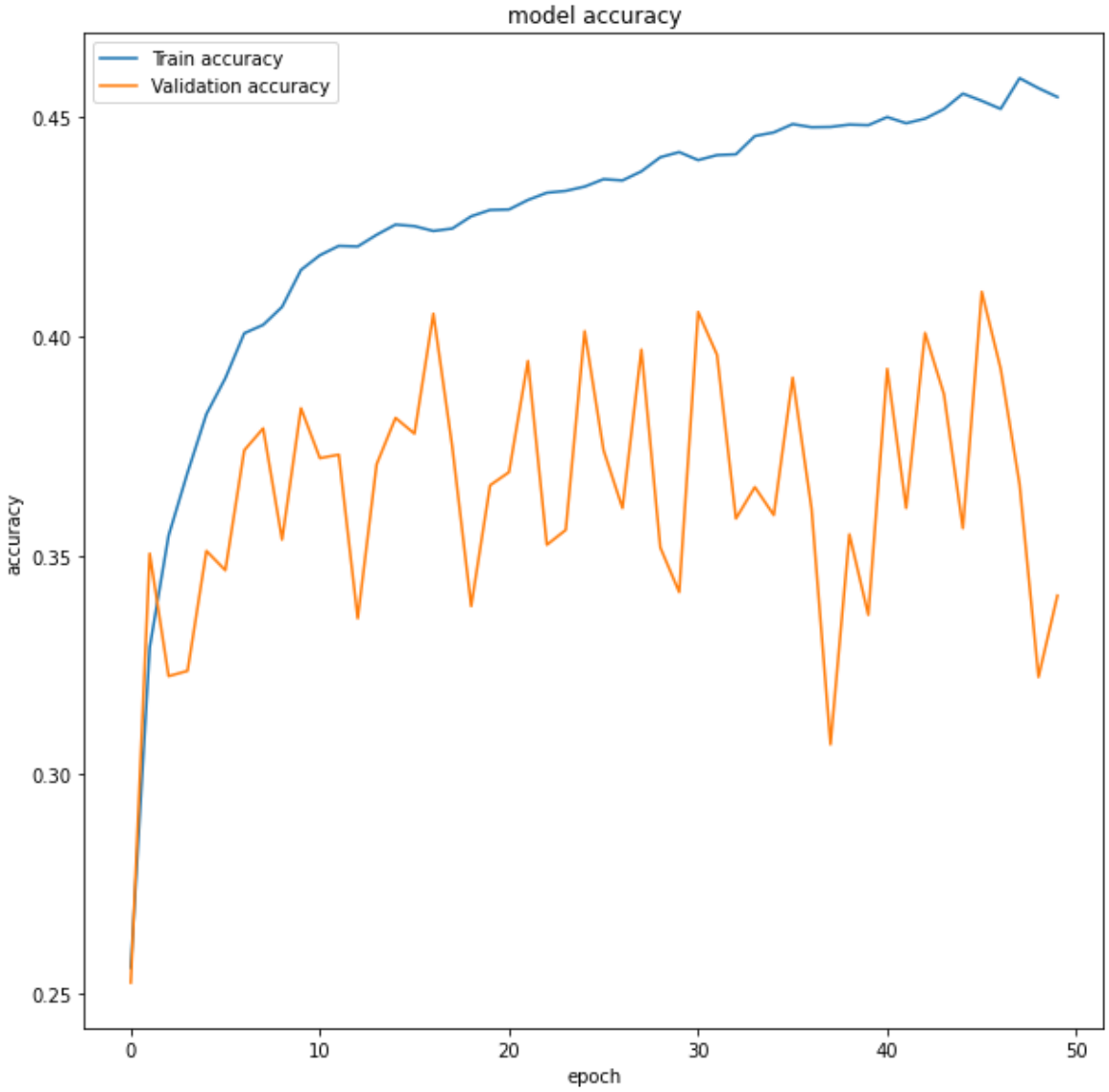
شکل 57. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

recall is : 0.402

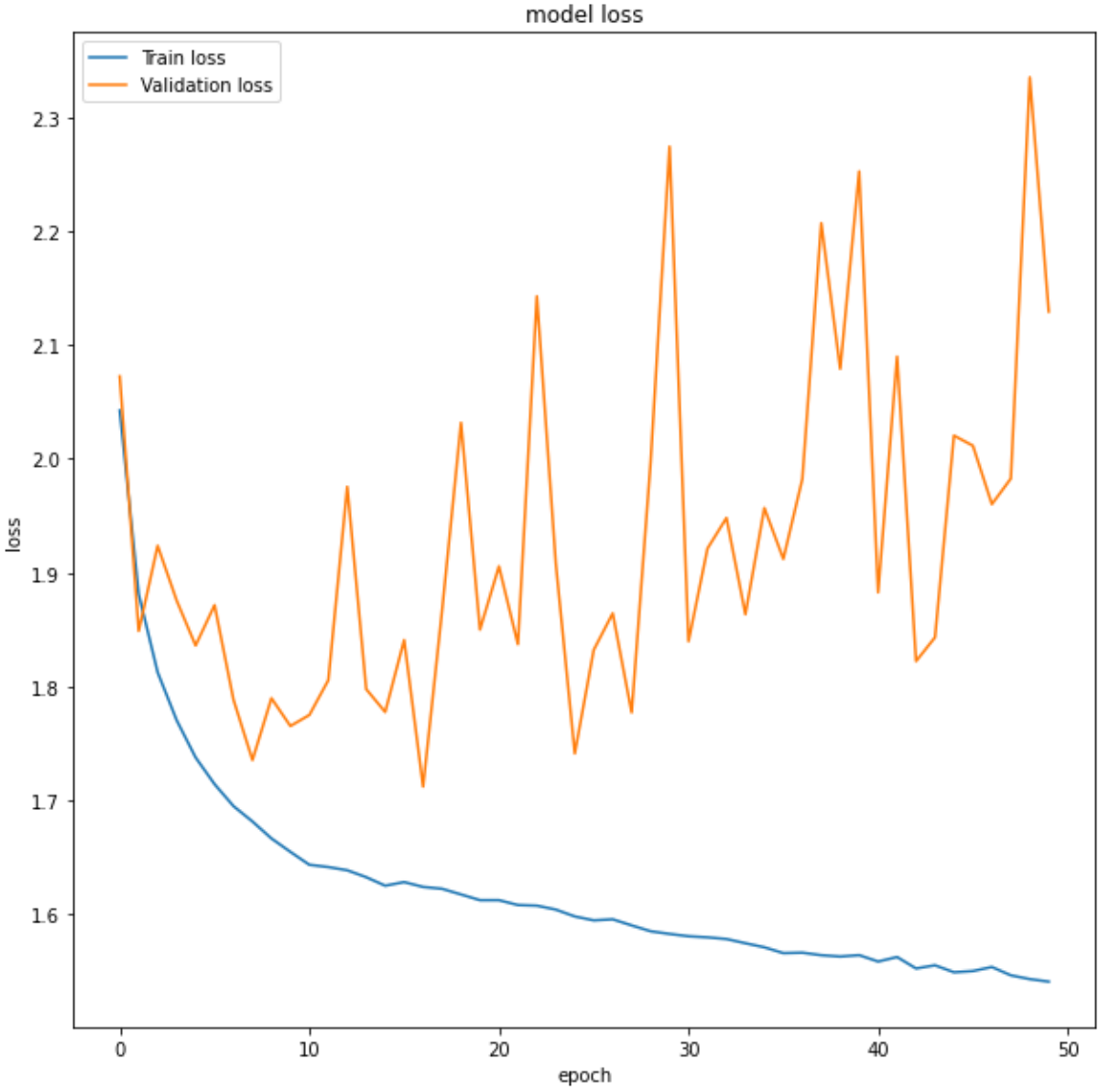
precision is : 0.415281041608644

f1 is : 0.4018068941519184

optimizer = Adamax



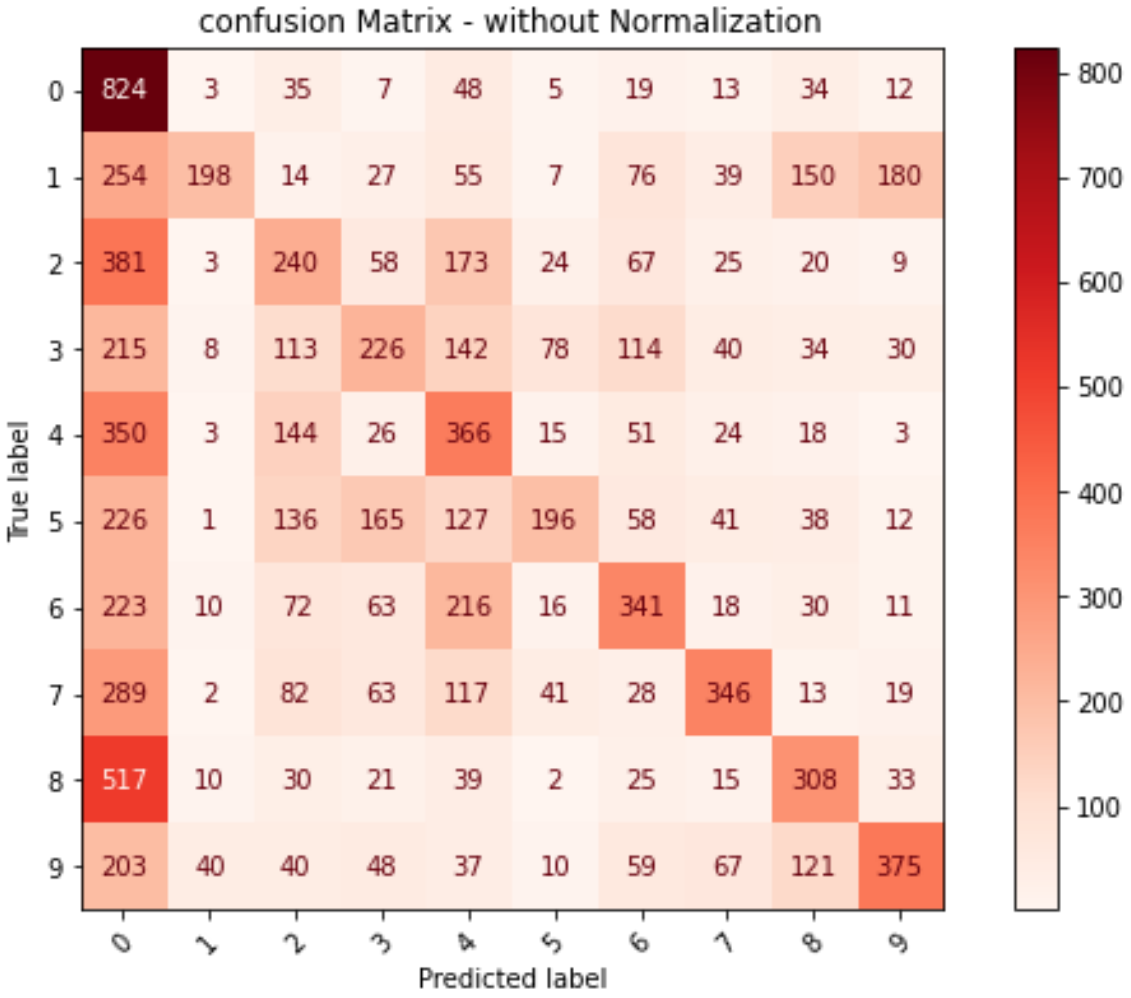
شکل 58. نمودار تغییرات دقت با نورون‌های 256، 128، 64 و 10



شکل 59. نمودار تغییرات خطا با نورون‌های 256، 128، 64 و 10

313/313 [==============================] - 1s 3ms/step - loss: 2.1147 - accuracy: 0.3420

loss in test data is: 2.114704132080078
accuracy in test data is : 0.34200000762939453
Training time is : 0.06223034858703613



شکل 60. ماتریس آشفتگی با نورون‌های 256، 128، 64 و 10

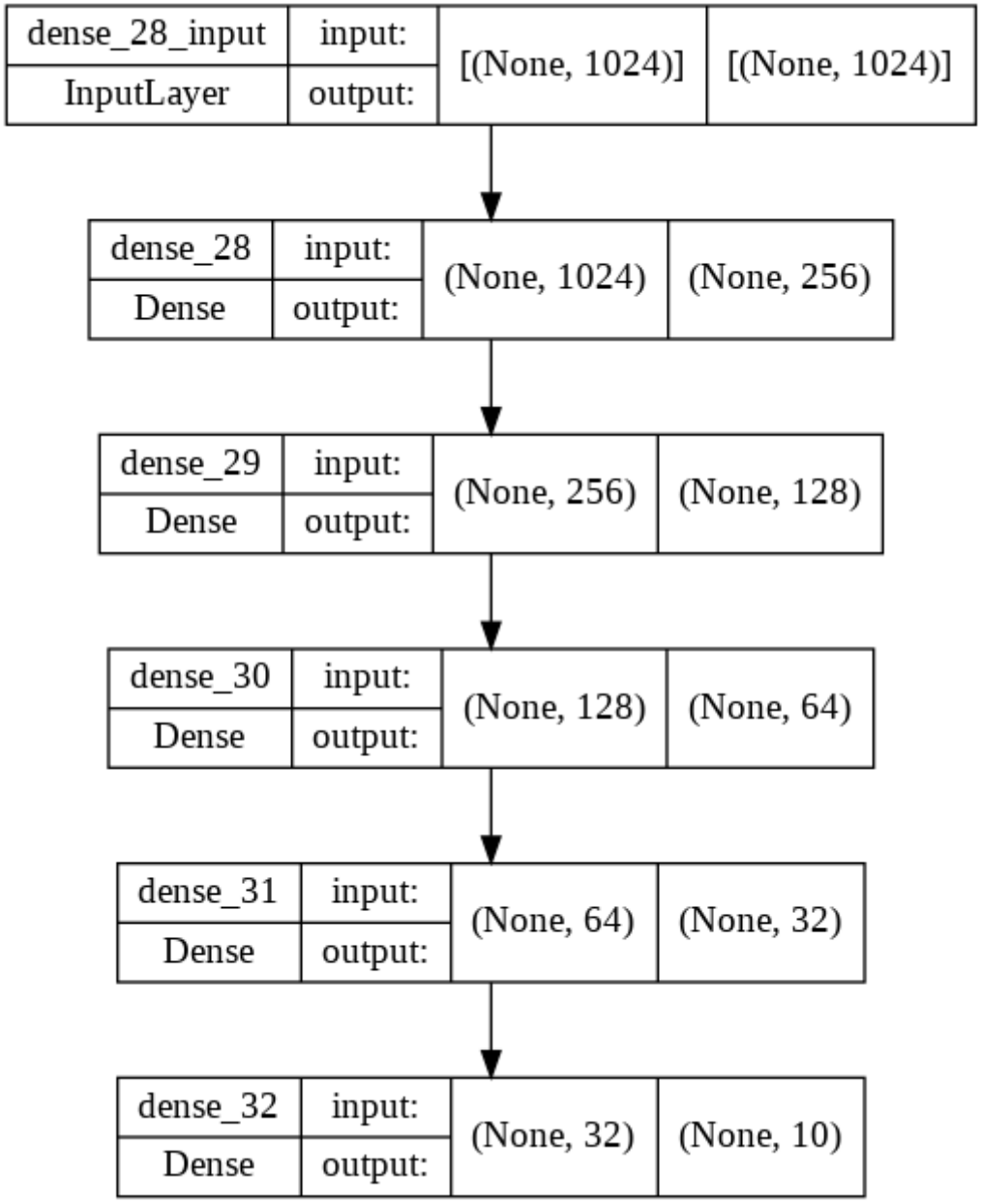
recall is : 0.342

precision is : 0.4217744909475936

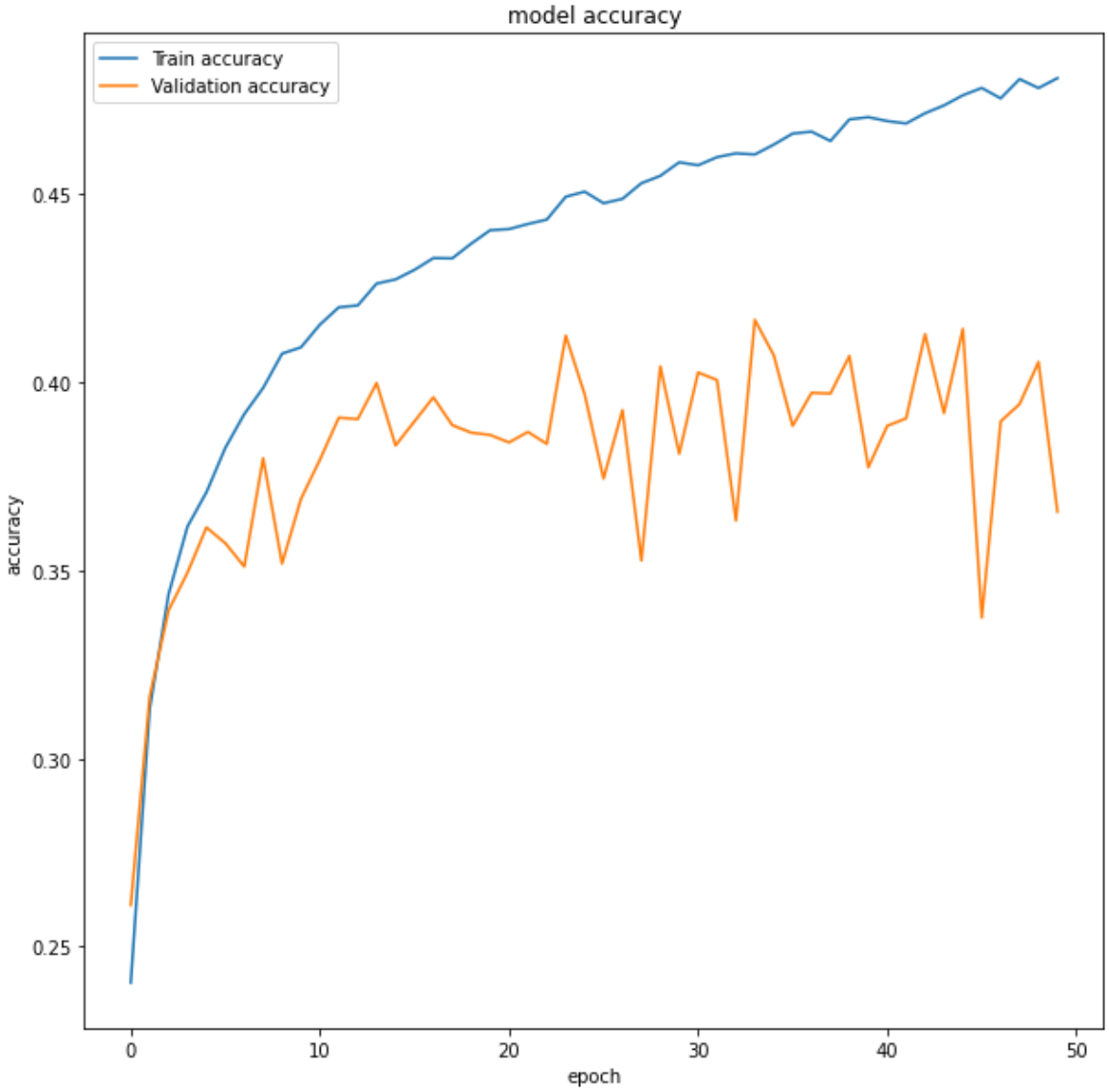
f1 is : 0.33816650410773696

با توجه به نمودارهای خطا و دقت و همچنین مقدار خطا و دقت در داده تست تابع SGD و Adam در میزان دقت تقریبا یکی هستند ولی میزان خطا در SGD کمتر از Adam است. تابع Adamax هم در خطا و هم در دقت بدتر از بقیه عمل می‌کند بنابراین تابع SGD را انتخاب می‌کنیم.

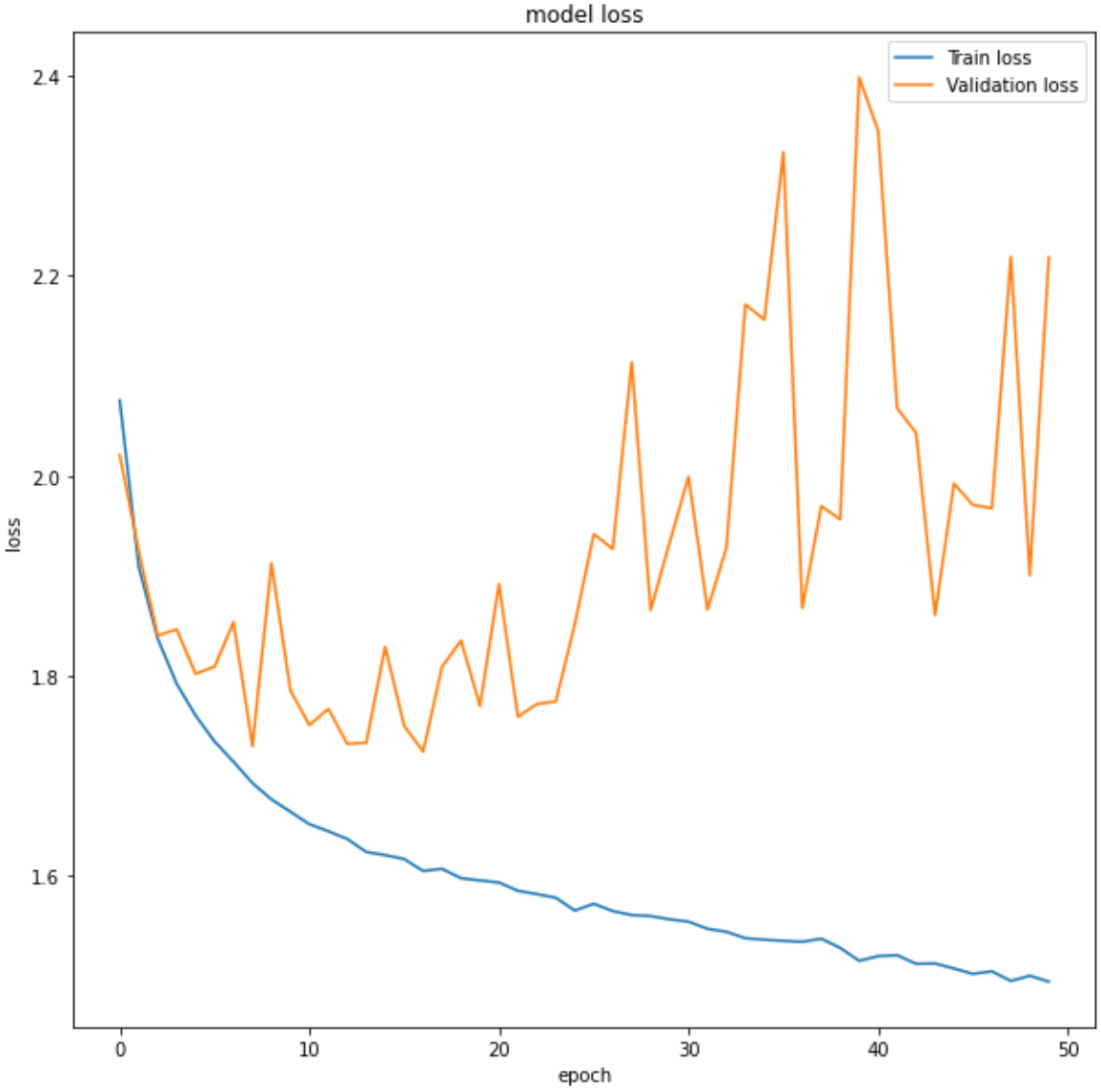
قسمت ح) تا به حال با دو لایه پنهان شبکه را طراحی کرده بودیم. با افزون لایه‌ها نتایج را بررسی می‌کنیم. *حالت اول : سه لایه*



شکل 61. معماری شبکه با نورون‌های 256، 128، 64، 32 و 10



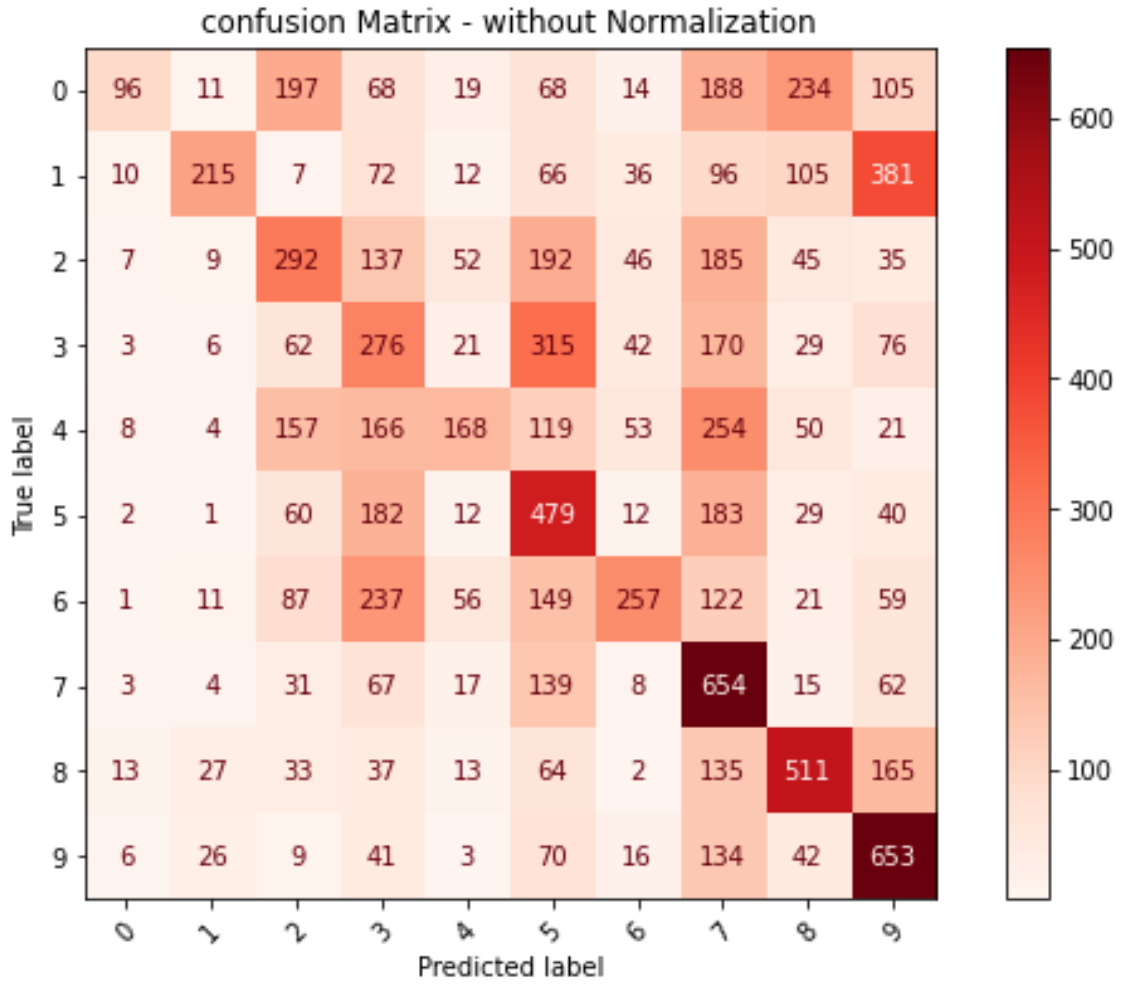
شکل 62. نمودار تغییرات دقت با نورون‌های 256، 128، 64، 32 و 10



شکل 63. نمودار تغییرات خطا با نورون‌های 256، 128، 64، 32 و 10

313/313 [==============================] - 1s 2ms/step - loss: 2.2047 - accuracy: 0.3601

loss in test data is: 2.204744577407837
accuracy in test data is : 0.36010000109672546
Training time is : 0.08505415916442871



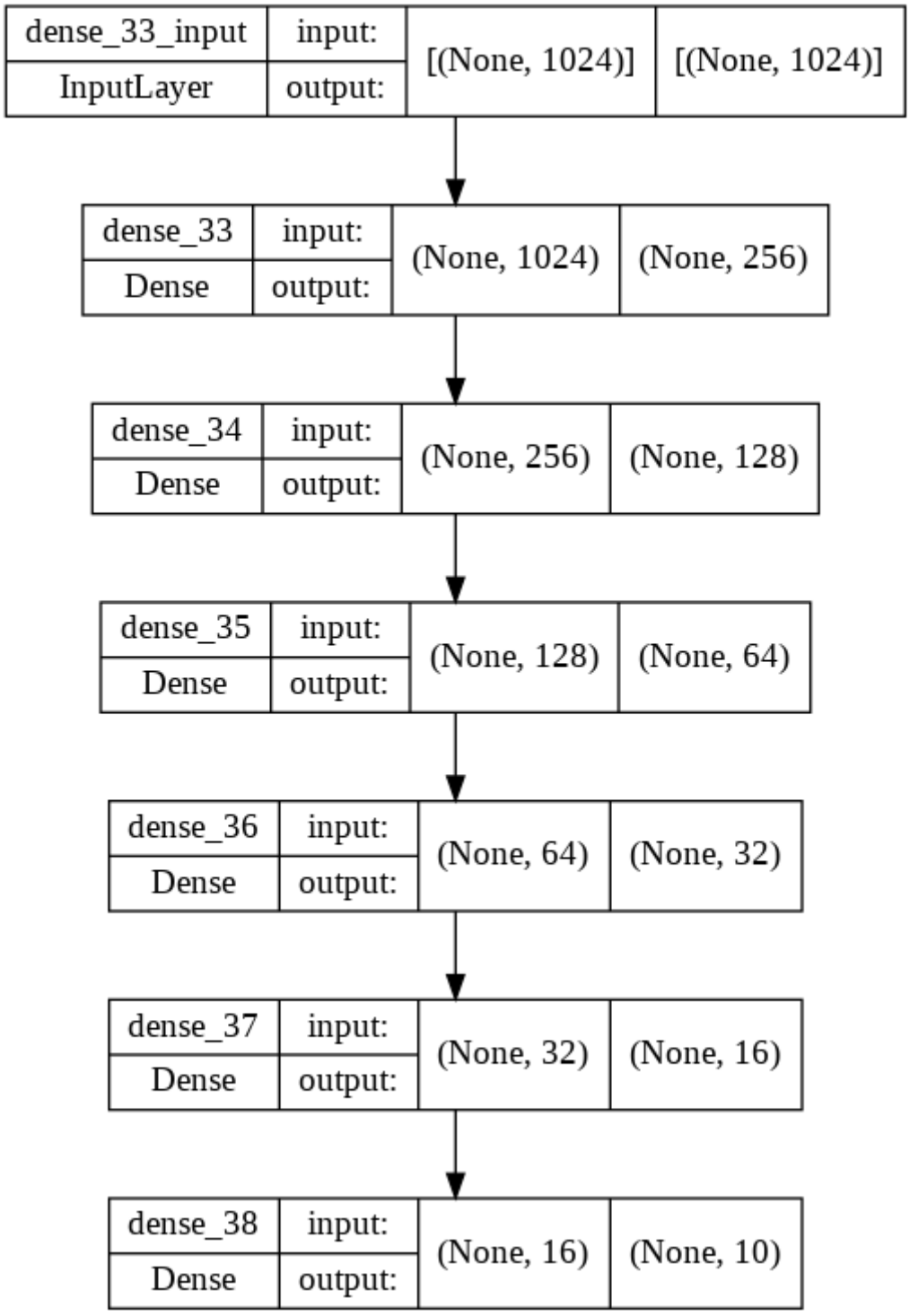
شکل 64. ماتریس آشفتگی با نورون‌های 256، 128، 64، 32 و 10

recall is : 0.3601

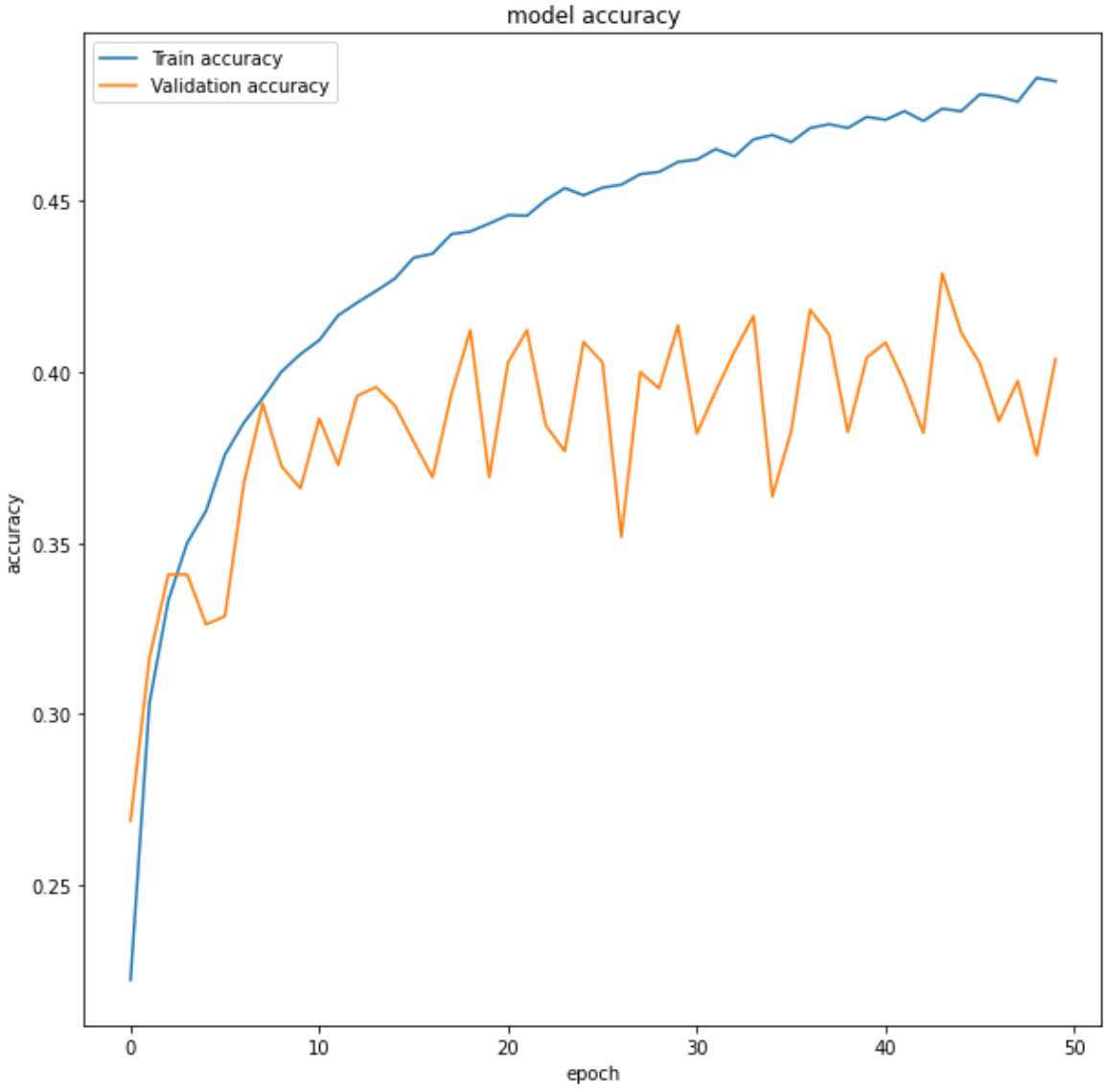
precision is : 0.4313965421858252

f1 is : 0.34016668672792794

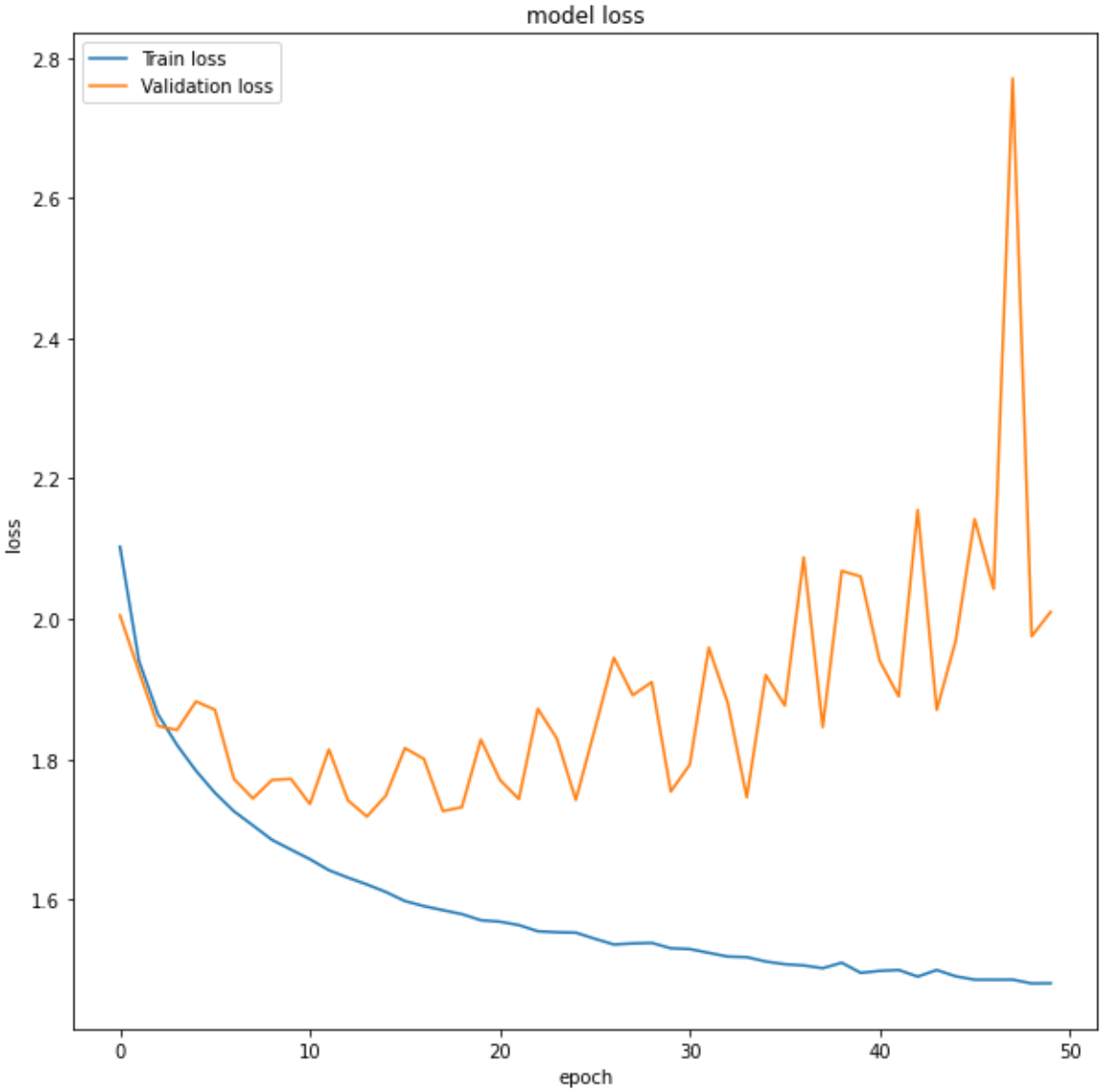
حالت دوم : چهار لایه پنهان



شکل 65. معماری شبکه با نورون‌های 256، 128، 64، 32، 16 و 10



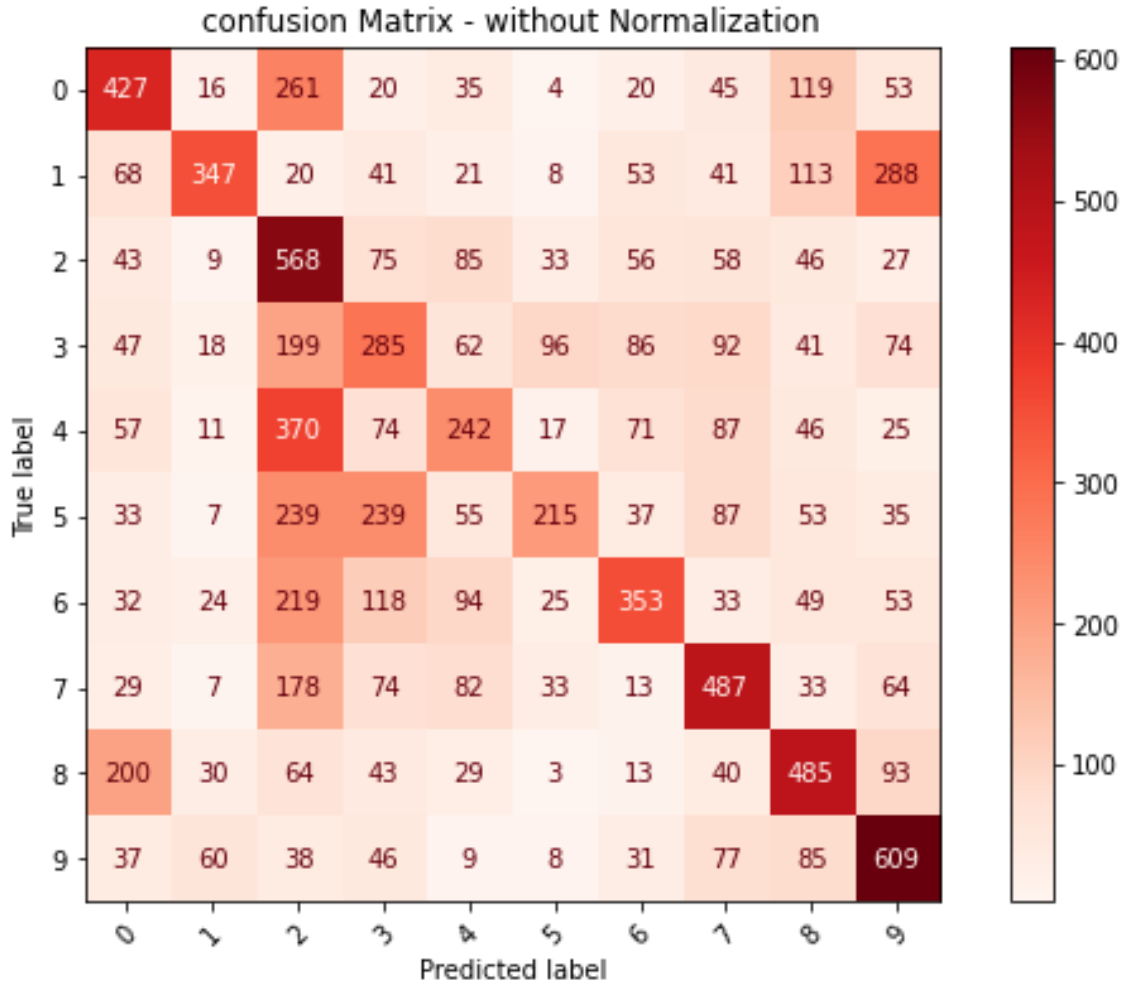
شکل 66. نمودار تغییرات دقت با نورون‌های 256، 128، 64، 32، 16 و 10



شکل 67. نمودار تغییرات خطا با نورون‌های 256، 128، 64، 32، 16 و 10

313/313 [==============================] - 1s 3ms/step - loss: 1.8890 - accuracy: 0.4018

loss in test data is: 1.8890472650527954
accuracy in test data is : 0.4018000066280365
Training time is : 0.07394099235534668



شکل 68. نمودار تغییرات خطا با نورون‌های 256، 128، 64، 32، 16 و 10

recall is : 0.4018

precision is : 0.43254099175828287

f1 is : 0.39867170228838245

با توجه به نمودار تغییرات خطا و دقت و همچنین دقت و خطا در داده تست افزودن لایه به شبکه باعث پیچیدگی بیشتر شبکه شده و میزان خطا افزایش می‌یابد و شبکه به خوبی آموزش نمی‌بیند. حالت بهینه همان شبکه با دو لایه پنهان و با نورون‌های 256، 128 و 64 و 10 می‌باشد.

قسمت ط) انتخاب پارامترهای زیر بهترین نتیجه را می‌دهد :

learning rate : 0.001

batch_size = 32

number of neurons in input layer : 256

number of neuron in first hidden layer : 128

number of neuron in second hidden layer : 64

number of neuron in output layer : 10

epochs: 50

loss function : categorical_crossentropy

پارامتر learning rate با آزمون و خطا به دست آمد. batch_size با مقادیر بیشتر از 32 را امتحان کردیم و میزان دقت و خطا در حالت 64 و 256 و ... به خوبی مقدار 32 نبودند. تعداد لایه‌ها نیز با بیشتر شدن باعث پیچیدگی بیشتر شبکه شدند و با همین مقادیر بهترین نتیجه به دست آمد. تعداد ایپاک با آزمون و خطا به دست آمد. تابع خطا نیز به دلیل اینکه توابع خطای دیگر باعث کاهش val_accuracy می‌شدند از این تابع استفاده کردیم.

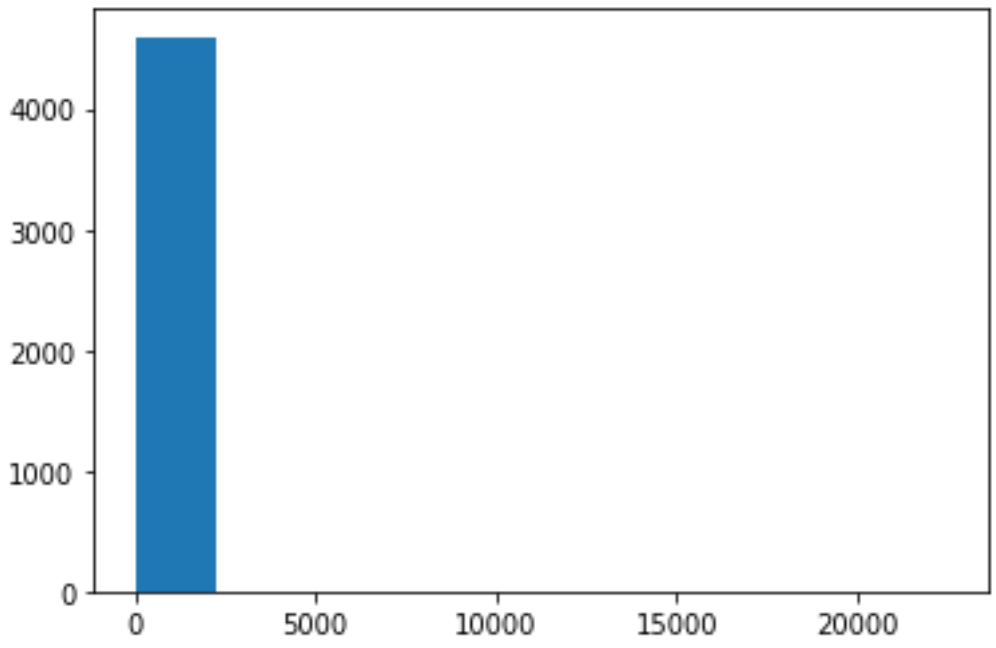
قسمت ن)

Learning rate	0.001
Batch size	32
Epochs	50
Loss function	categorical_crossentropy
number of neurons in input layer	256
number of neurons in first hidden layer	128
number of neurons in second hissen layer	64
number of neurons in output layer	10
Val_accuracy	40%
val_loss	1.69
Loss in test data	1.69
Accuracy in test data	40%
Training time	0.05

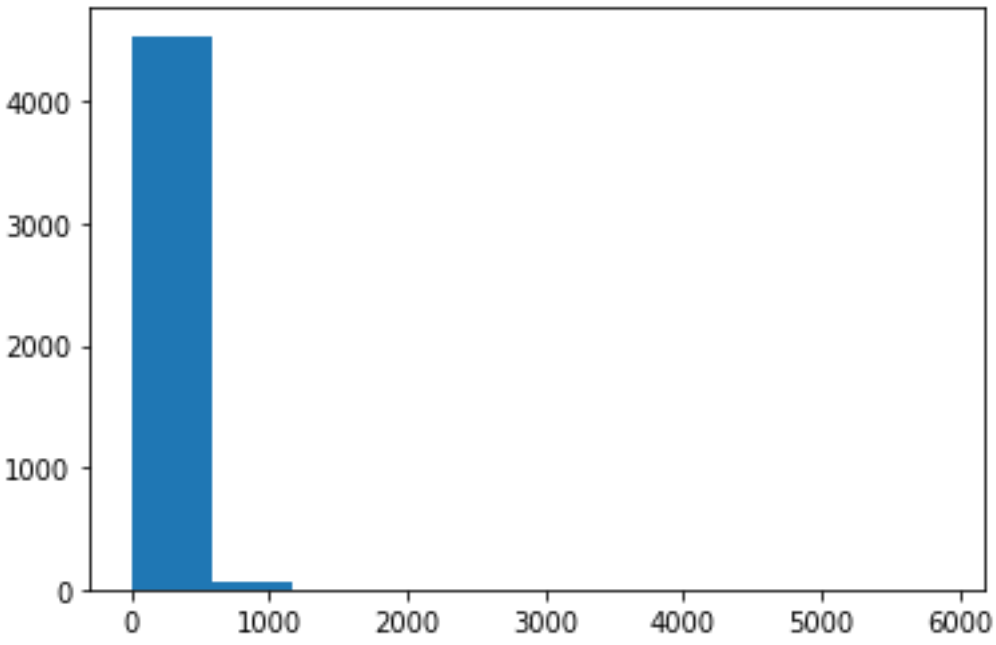
سوال ۲ – (Regression) MLP

قسمت الف) برای پیش‌پردازش داده ابتدا ستون‌هایی که در آنها مقادیر غیرعددی هستند، باید به عدد تبدیل کنیم. مقادیر X_train، X_test و X_val را با استفاده از StandardScaler نرمال می‌کنیم تا مقادیر عددی بین یک و صفر باشند. یک کار دیگر اینکه قیمت‌های صفر را که برای قیمت خانه بی‌معنی هستند از داده حذف کردم ولی میزان mse افزایش پیدا کرد. نمودار هیستوگرام زیر را که برای مقدار price/sqft_living رسم کردیم مشاهده می‌شود که بعضی اعداد پرت در داده وجود دارد که به شدت روی پیش بینی و mse تاثیر می‌گذارد که براساس کد زیر فیلتر می‌کند :

```
data=data[data["price"]/data["sqft_living"]<10000]
```



شکل 69. نمودار قیمت تقسیم بر متر خانه قبل فیلتر

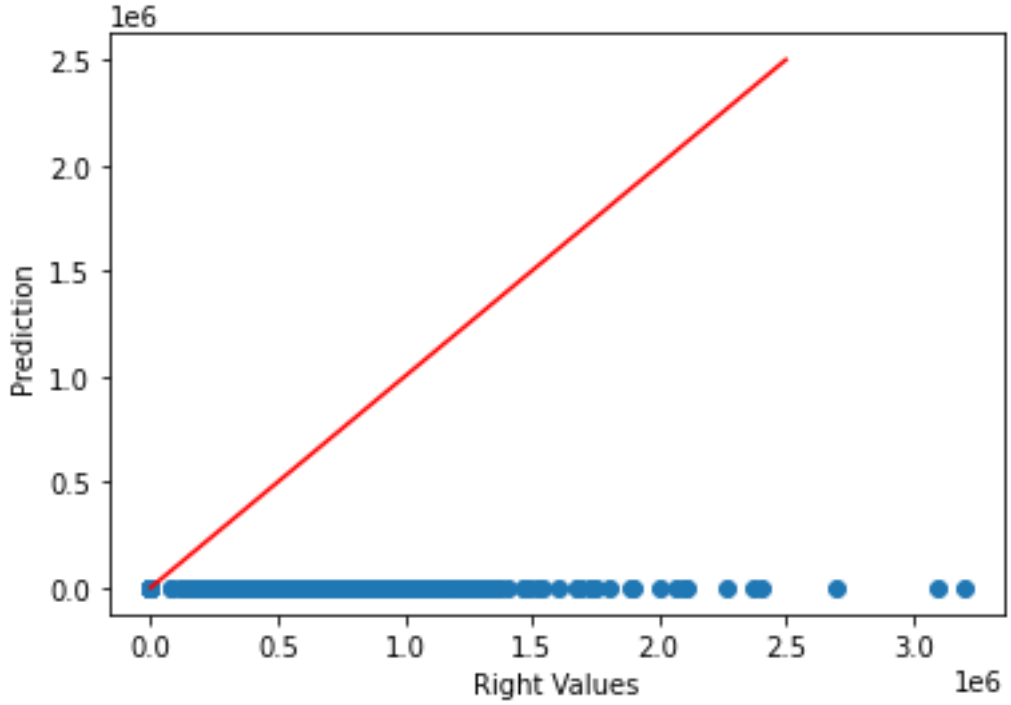


شکل 70. نمودار قیمت تقسیم بر متر خانه بعد فیلتر

قسمت ب) با استفاده از تکه کد زیر داده‌ها را به صورت تصادفی به داده‌های تست، آموزشی و ارزیابی تقسیم می‌کنیم:

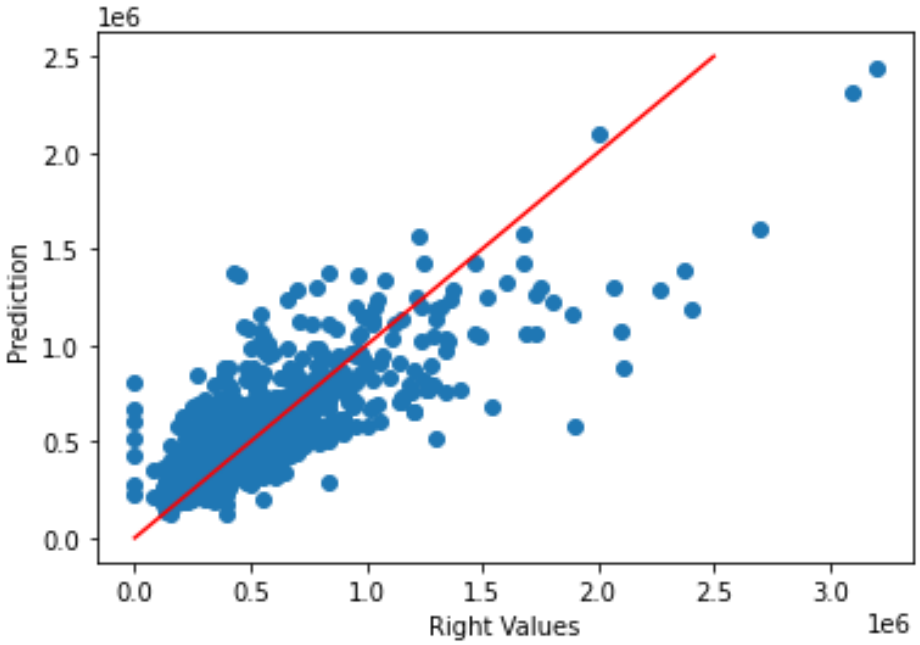
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
state = 50)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
1, random_state = 20)
```

وقتی از تابع فعال‌ساز sigmoid استفاده می‌کنیم چه در حالت دو لایه و چه در حالت تک لایه پنهان قیمت‌ها به صورت زیر پیش‌بینی می‌شود که نشان می‌دهد شبکه به درستی کار نمی‌کند :



شکل 71. شبکه با تابع **sigmoid**

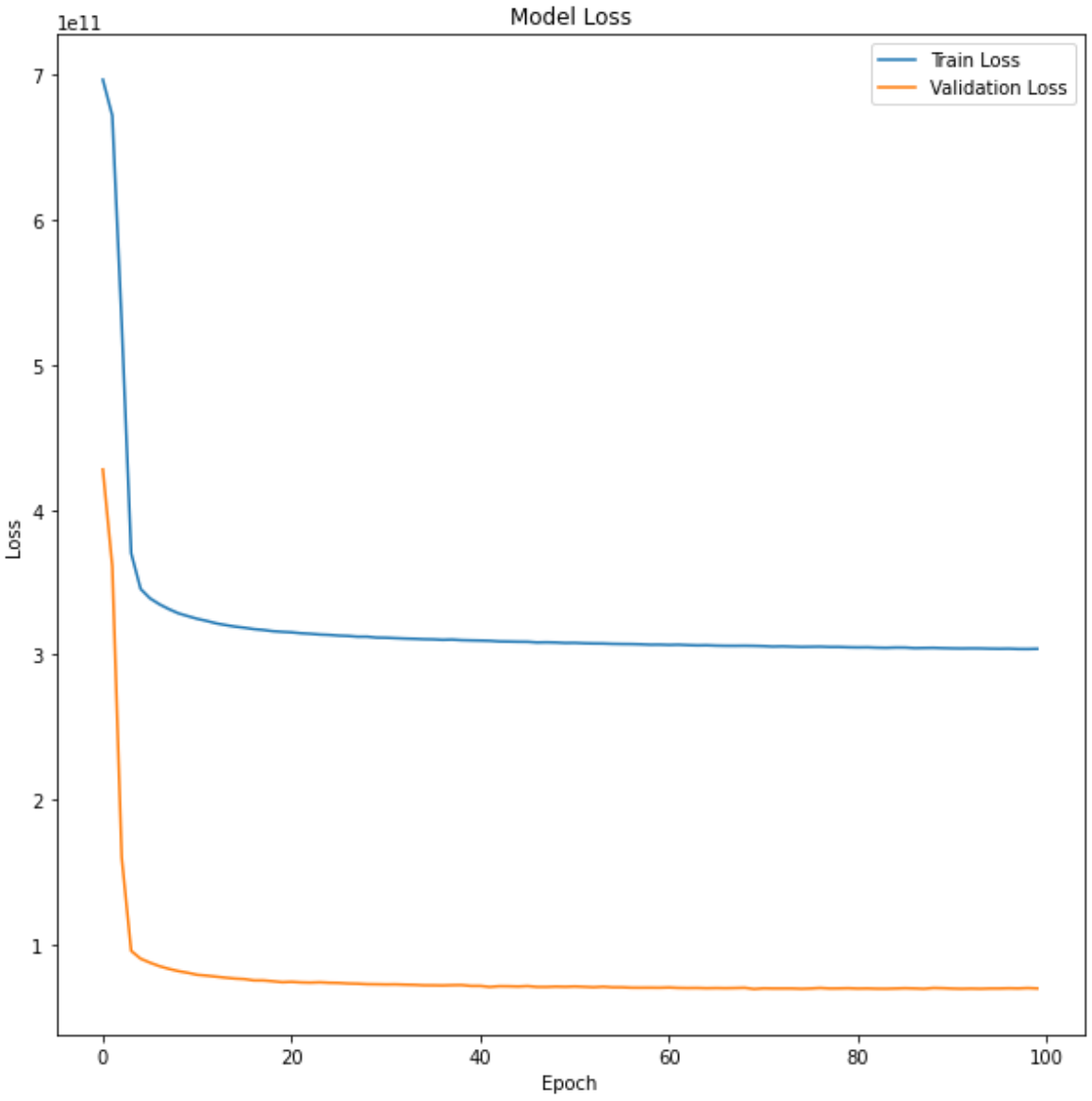
وقتی از تابع فعال‌ساز relu استفاده می‌کنیم در حالتی که از دو لایه پنهان استفاده می‌کنیم مقدار mse کمتری نسبت به حالت تک لایه دارد و بهتر از حالت تک لایه است.



شکل 72. شبکه با تابع relu

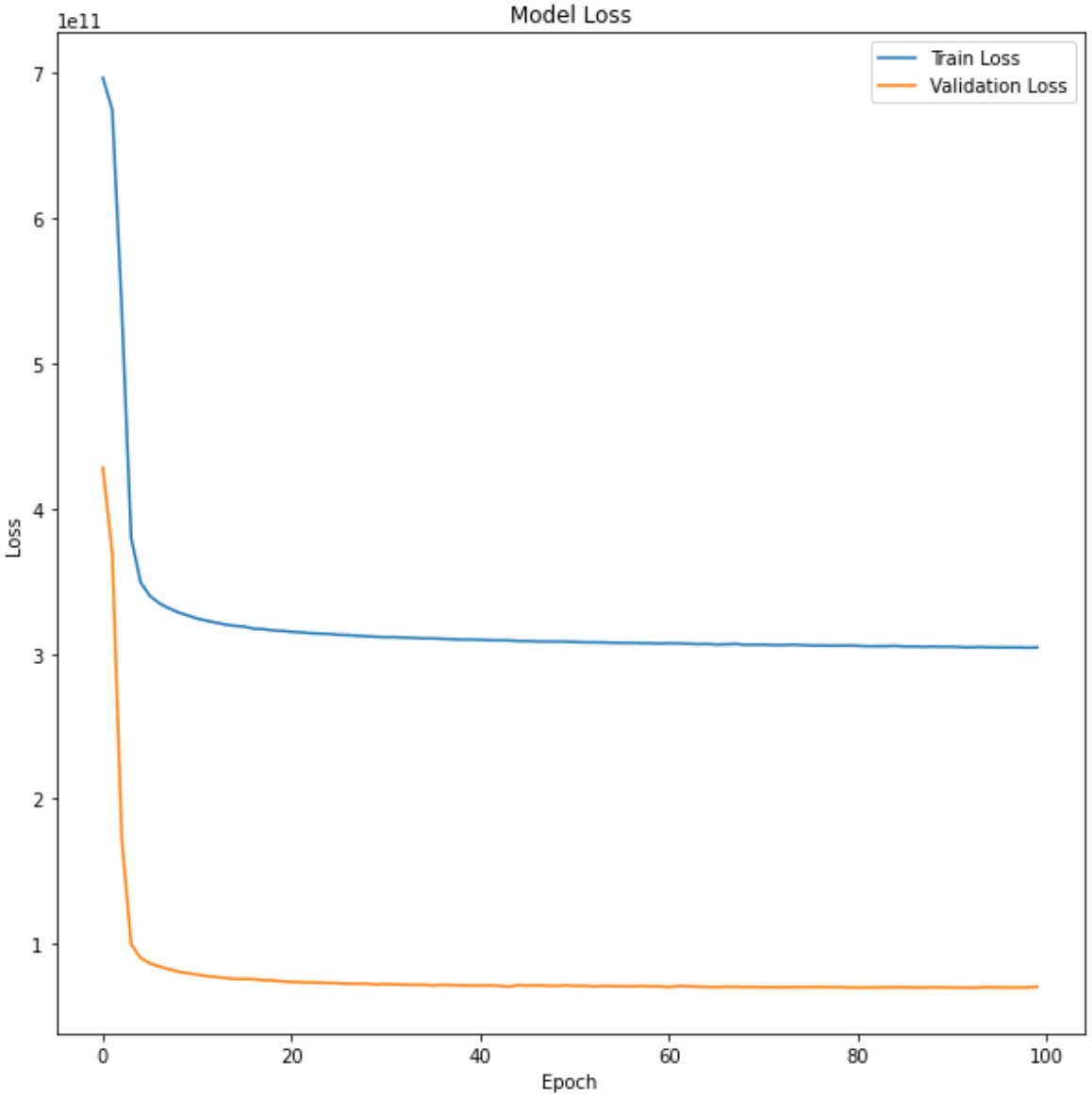
برای طراحی شبکه هنگامی که ستون country را حذف می‌کنیم چون فقط یک مقدار یکتا دارد و در این حالت مقدار mse کمتر می‌شود.

قسمت ج)



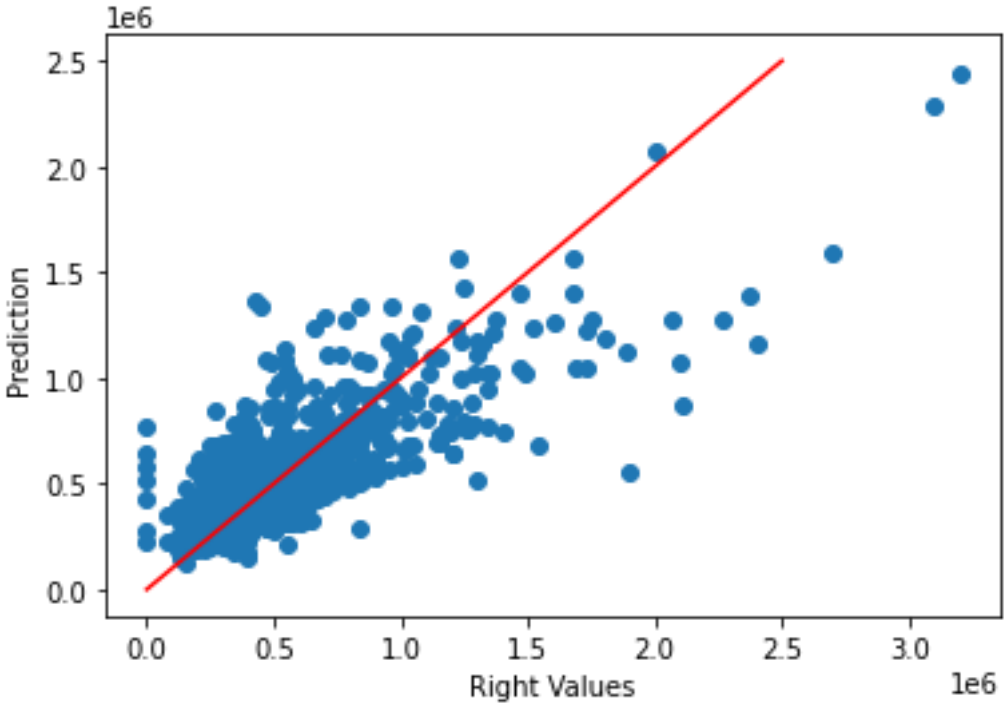
شکل 73. مقادیر mse در هر ایپاک با تابع mse

تعداد ایپاک بهینه : 121



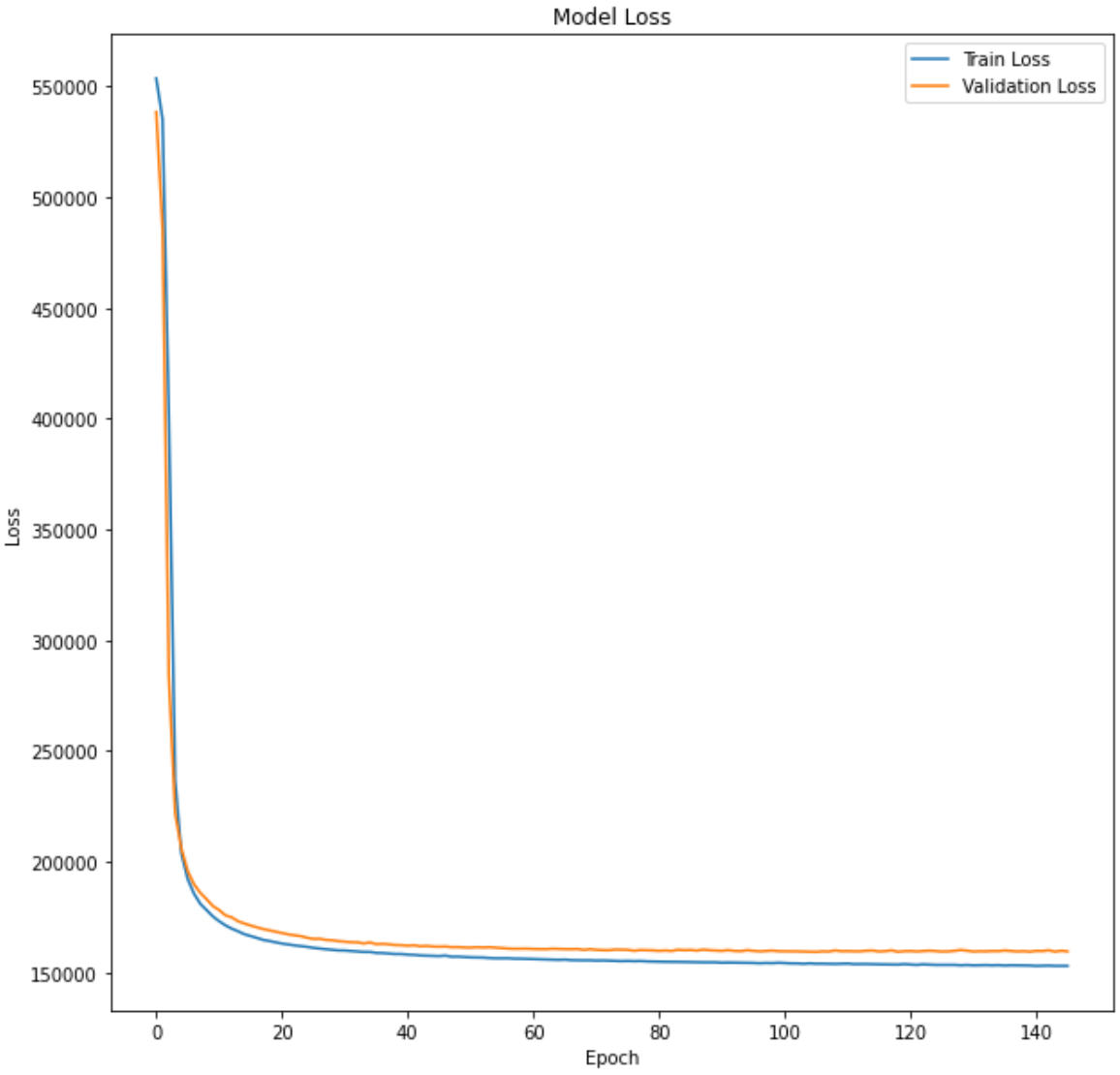
شکل **74**. مقادیر **mae** در هر ایپاک با تابع **mse**

تعداد ایپاک بهینه : 225



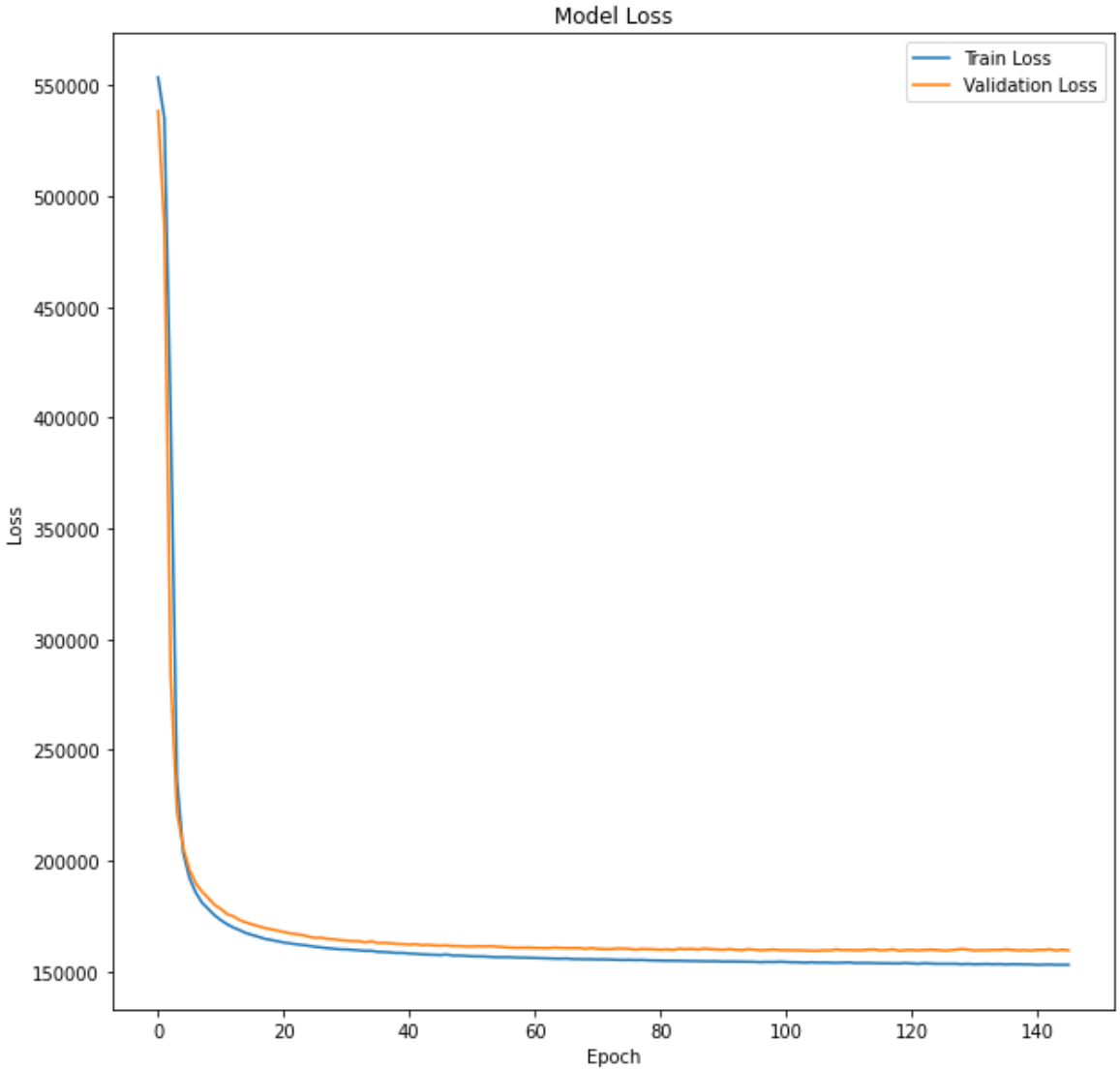
شکل 75. نمودار مقادیر پیش‌بینی شده بر حسب مقادیر واقعی

قسمت د)



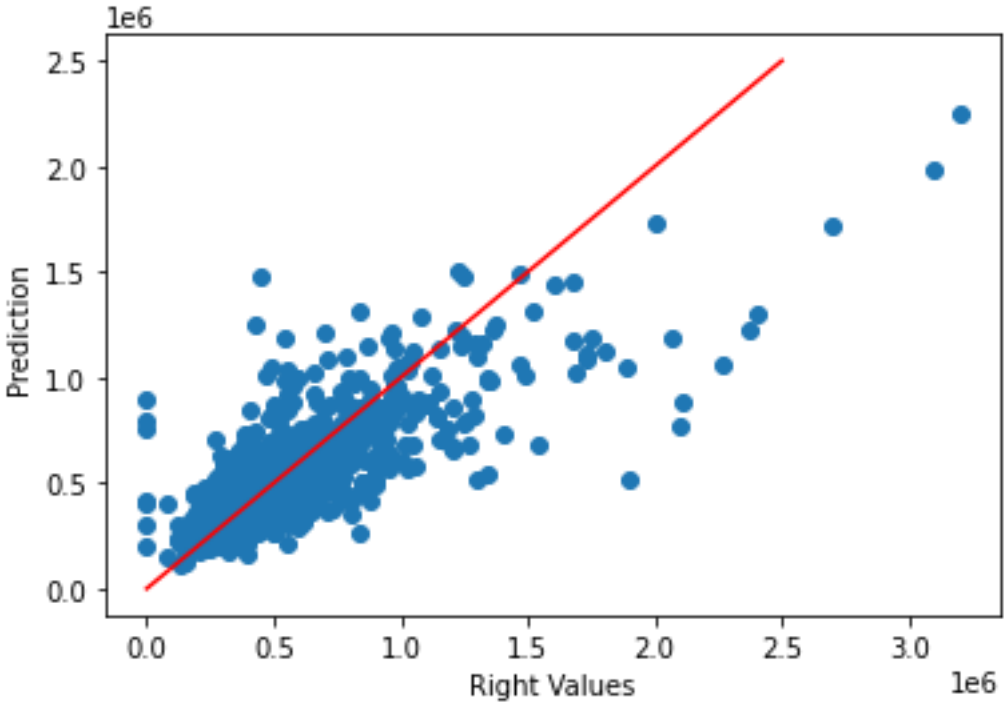
شکل 76. مقادیر **mse** در هر ایپاک با تابع **mae**

تعداد ایپاک بهینه : 146



شکل **77**. مقادیر **mae** در هر ایپاک با تابع **mae**

تعداد ایپاک بهینه : 167



شکل **78**. مقادیر پیش‌بینی شده بر اساس مقدار واقعی

قسمت ه)

$$\text{MSE} = \frac{\sum(y-\hat{y})\text{^}2}{n} \qquad \text{MAE} = \frac{\sum|y-\hat{y}|}{n}$$

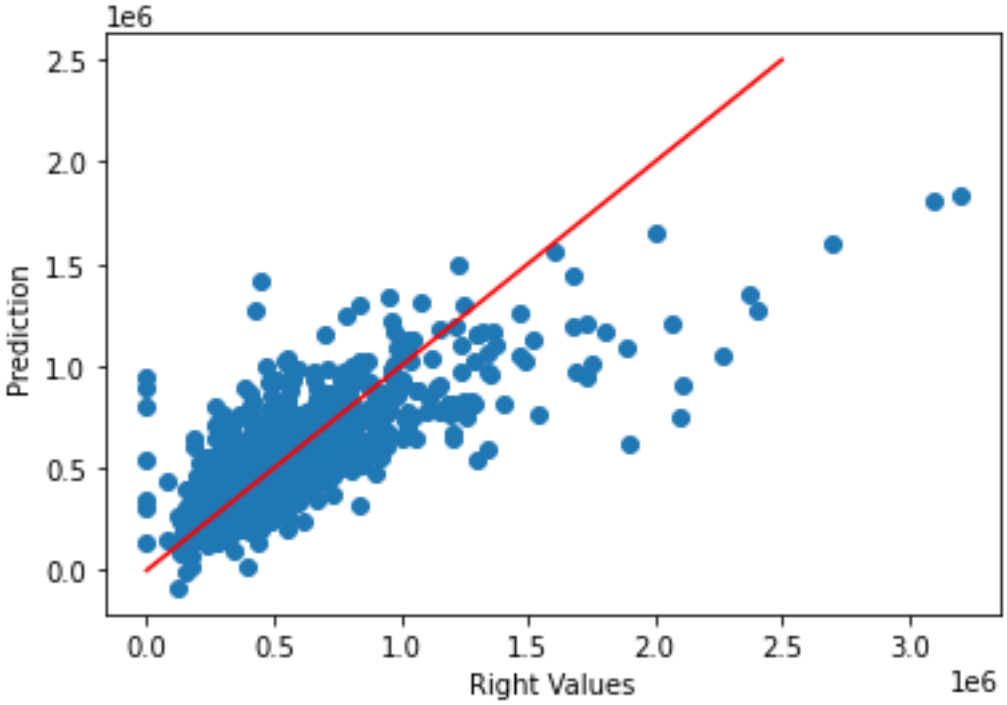
با توجه به اینکه در دیتاست مقادیر بزرگ و پرت وجود دارد مقدار خطا در mse بزرگ تر شده و خطا را از مقدار واقعی بزرگتر نشان می‌دهد. با توجه به نتایج قسمت ج و د در قسمت ج loss و val_loss فاصله زیادی با هم دارند ولی در قسمت د این دو مقدار به هم نزدیک هستند چون mae نسبت به مقادیر پرت مقاوم‌تر عمل می‌کند.

قسمت امتیازی)

مدل رگرسیون خطی یک روش برای مدل خطی بین یک متغیر پاسخ مانند قیمت در این سوال و یک یا چند متغیر توصیفی مانند خیابان، متر مربع و... در این سوال است. هدف از انجام رگرسیون خطی کشف مدل خطی بین متغیرهای توصیفی و متغیر پاسخ است.

رابطه ریاضی برای رگرسیون خطی به صورت $y = \beta_0 + \beta_1 * x$ است که β_0 نشان دهنده عرض از مبدا و β_1 نشان دهنده شیب است. شیب خط نشان دهنده میزان حساسیت متغیر پاسخ به متغیر توصیفی است و عرض از مبدا نشان دهنده مقداری از متغیر پاسخ است که به ازای مقدار متغیر توصیفی صفر محاسبه شده است.

در رگرسیون چندگانه با افزایش تعداد متغیرها Overfitting رخ داده و با کاهش آن‌ها نیز ممکن است با مسئله Underfitting مواجه شویم. در صورتی که مدل رگرسیونی دچار Overfitting شود، خطای آن برای برآورد مقدارهای جدید متغیر پاسخ زیاد خواهد بود. بنابراین با افزایش تعداد متغیرها مشکل Overfitting ظاهر شده و با کاهش آن‌ها، واریانس مدل افزایش خواهد یافت. یکی از روش‌های غلبه بر این مسائل در رگرسیون چندگانه، استفاده از مدل Ridge Regression است. در مسئله رگرسیون خطی، از یک تابع خطا استفاده شده و سعی بر آن است که «مجموع مربعات خطا را کمینه کنند. در Ridge Regression به کمک ترکیب تابع مجموع مربعات خطا و مقدار جریمه مرتبط با تعداد پارامترها، تابع جدیدی ساخته می‌شود که برای برآورد پارامترهای مدل رگرسیونی به کار می‌رود.



شکل 79. مقادیر پیش‌بینی شده بر اساس مقدار واقعی با Ridge

همان طور که مشاهده می‌شود پیش‌بینی این تابع با شبکه ما تقریبا یکی است و بهبودی حاصل نشده است. با توجه به نتایج به دست آمده در این سوال هم از شبکه و هم از مدل رگرسیون مشکل به علت حذف نشدن داده‌های پرت در دیتاست می‌تواند باشد.

سوال 3 – کاهش ابعاد

قسمت الف)

در روش کاهش بعد به کمک PCA ابتدا الگوریتم جهت‌هایی که مجموعه داده پراکندگی بیشتری دارد را به دست می‌آورد سپس به وسیله‌ی این جهت‌ها بردارهای ویژه ماتریس را می سازد. پس از آن یا استفاده از یک تبدیل خطی پایه‌ای فضا را بر این بردارها منطبق می‌کند. PCA همیشه روی ماتریس کوواریانس یا همبستگی اعمال می‌شود یعنی داده‌ها باید عددی و استاندارد شده باشند.

روابط ریاضی :

ابتدا با استفاده از رابطه زیر به استاندارد سازی مقادیر می‌پردازیم :

```
X_meaned = X - np.mean(X , axis = 0)
```

سپس ماتریس کوواریانس مقدار بالا را به دست می‌آوریم تا بدانیم کدام متغیرها با هم، همبستگی دارند.

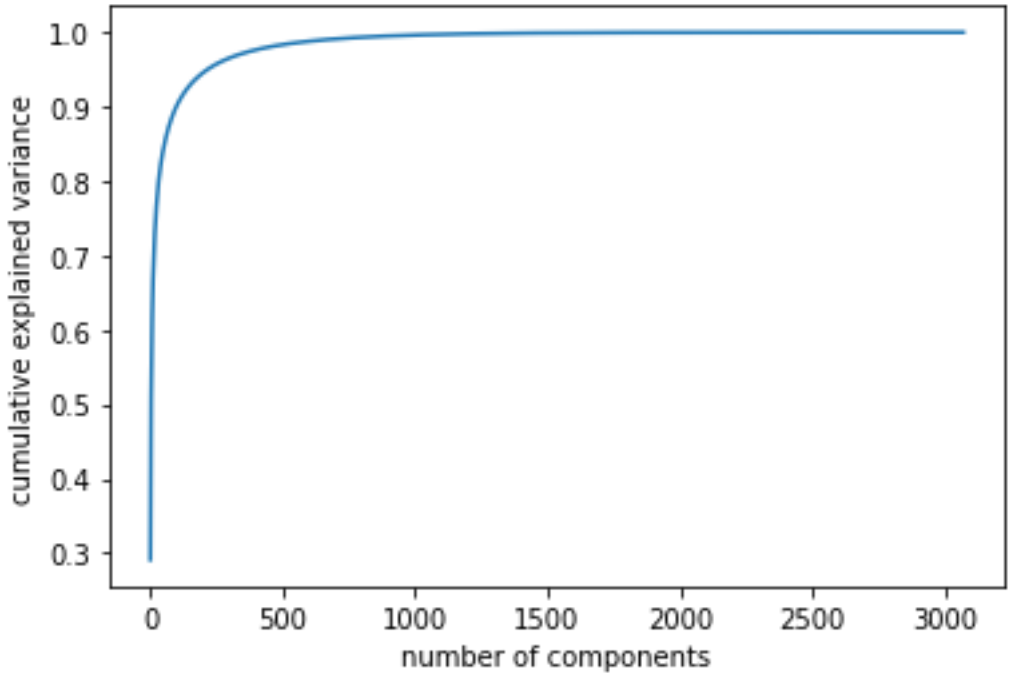
```
cov_mat = np.cov(X_meaned , rowvar = False)
```

پس از آن با استفاده از Eigenvectors و eigenvalues باید اجزای اصلی را حساب کنیم تا آنها را به ترتیب در pc_1، pc_2 و ... قرار دهیم و این مقادیر را به صورت نزولی مرتب می‌کنیم :

```
eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)

sorted_index = np.argsort(eigen_values)[::-1]
sorted_eigenvalue = eigen_values[sorted_index]
sorted_eigenvectors = eigen_vectors[:,sorted_index]
```

پس از آن دیتای فشرده شده را از طریق رابطه زیر بع دست می‌آید :

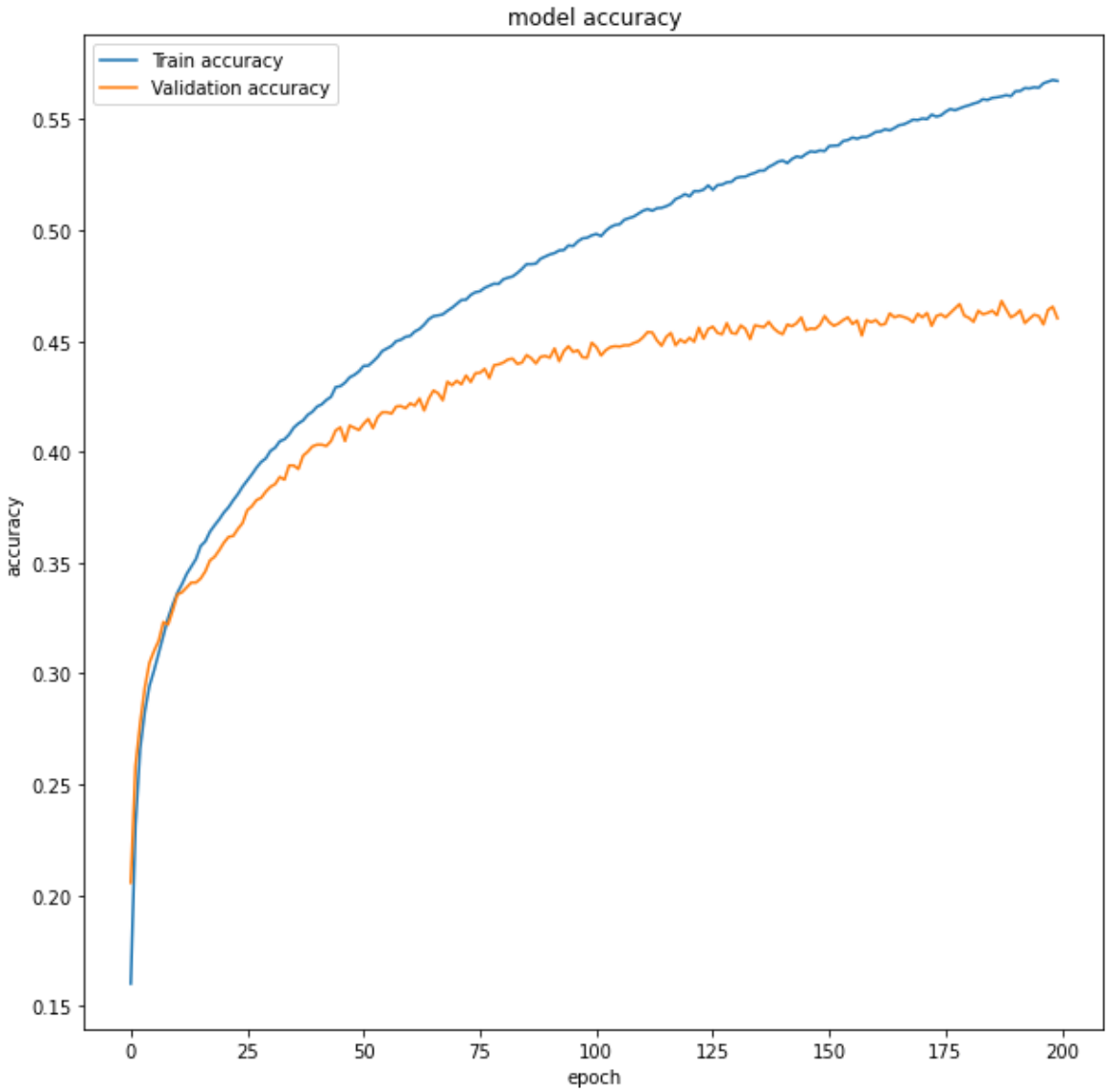
```
X_reduced = np.dot(eigenvector_subset.transpose() ,
                                    X_meaned.transpose() ).transpose()
```



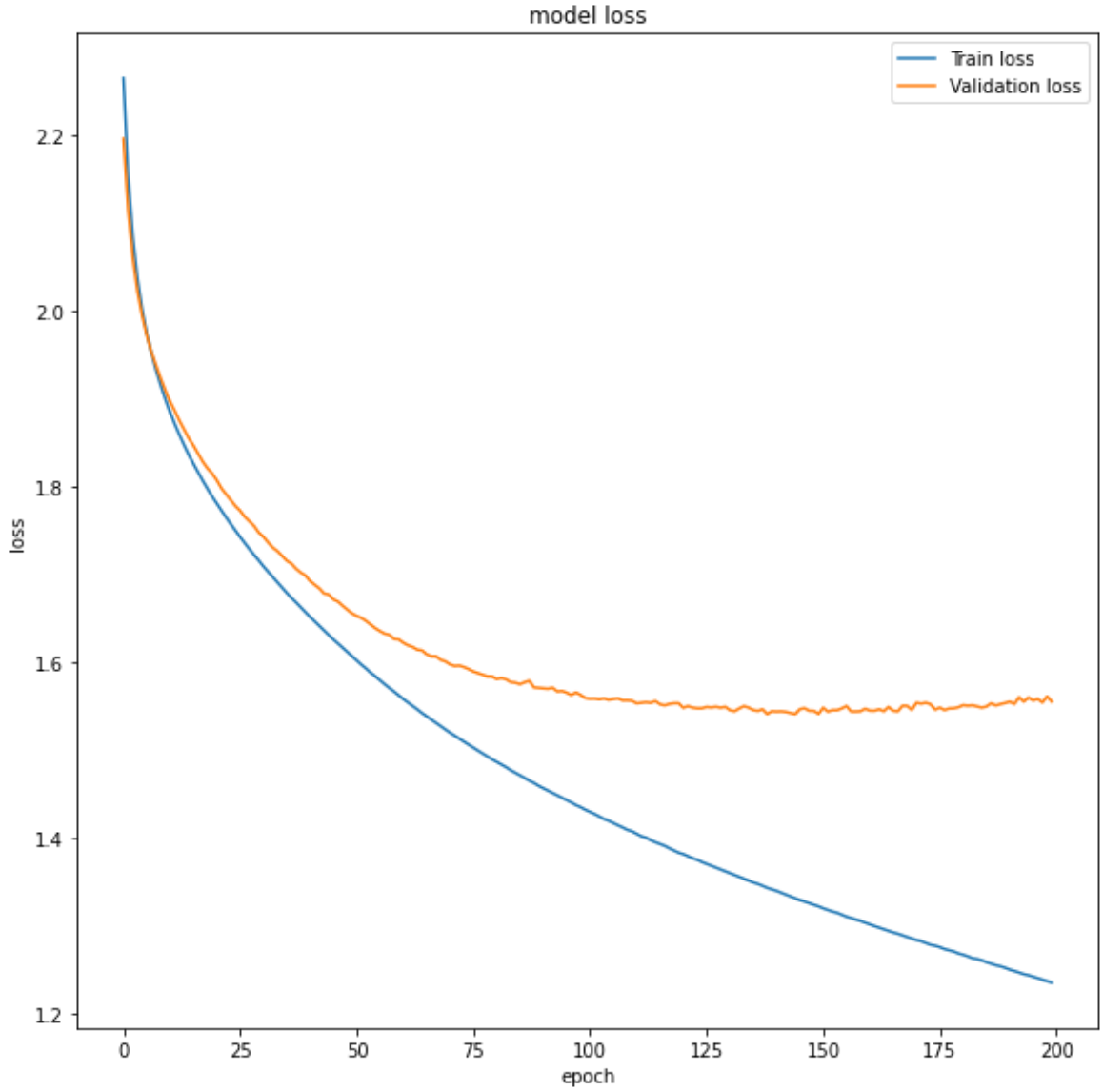
شکل 80. منحنی تعداد component **بر حسب واریانس تجمعی**

با توجه به منحنی کاهش ابعاد تا 100 قابل قبول است.

پس از اجرای PCA نتایج به دست آمد که کاهش ابعاد به 100 بهترین نتیجه را داشت:



شکل 81. نمودار تغییرات دقت با PCA

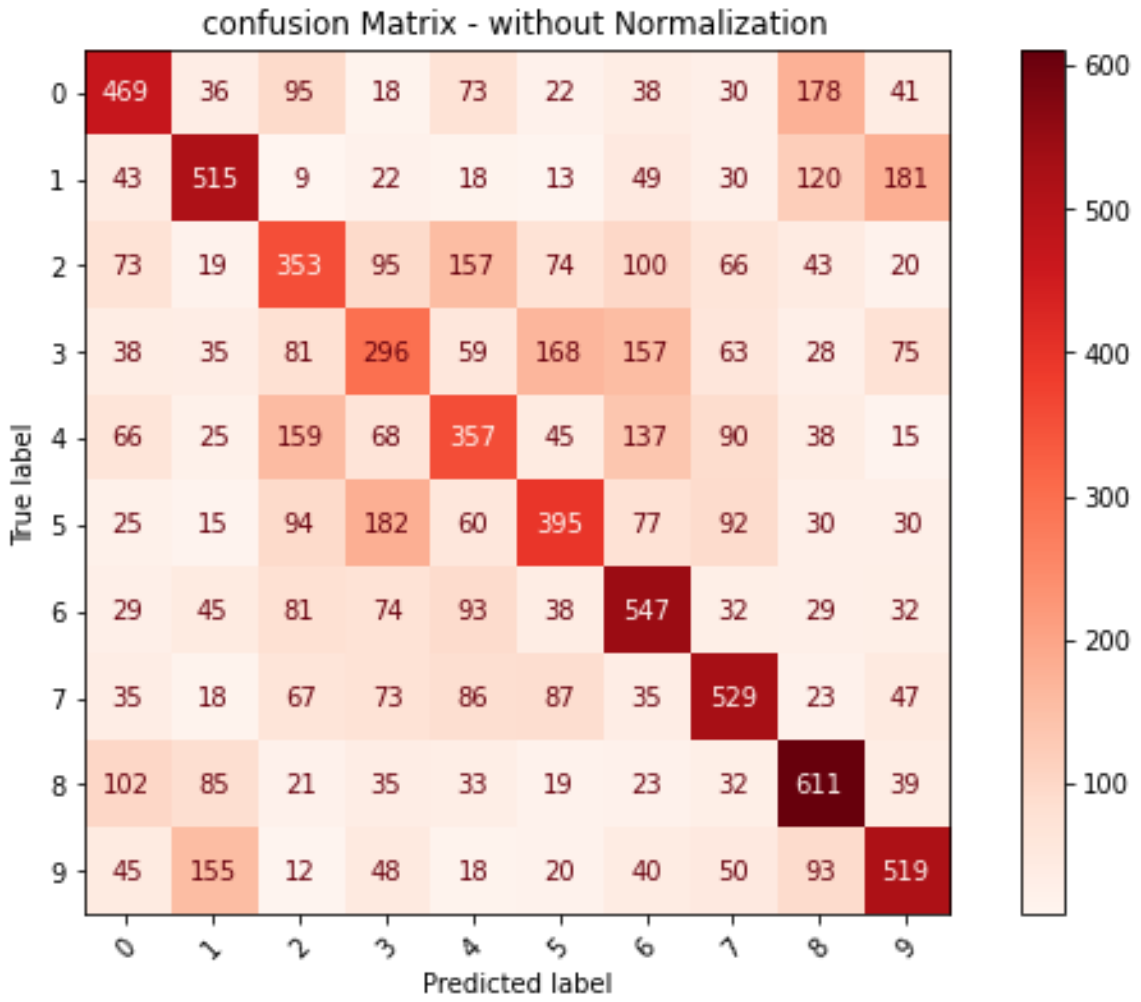


شکل 82. نمودار تغییرات خطا با **PCA**

زمان، دقت و خطای داده تست :

313/313 [==============================] - 1s 3ms/step - loss: 1.5507 - accuracy: 0.4591

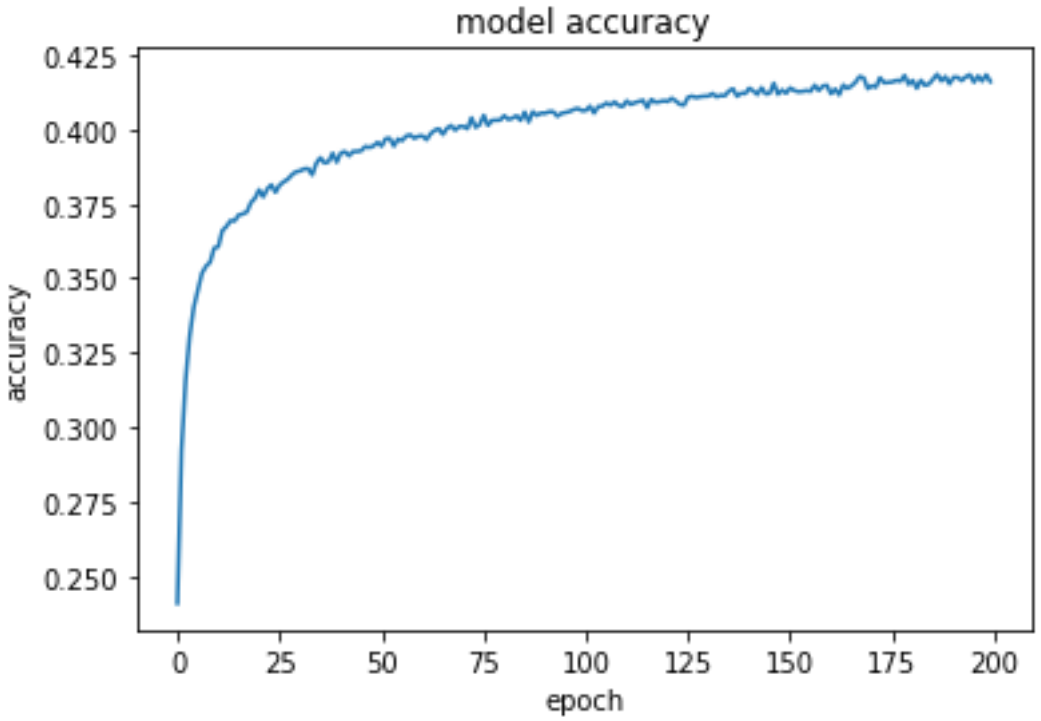
loss in test data is: 1.550654649734497
accuracy in test data is : 0.45910000801086426



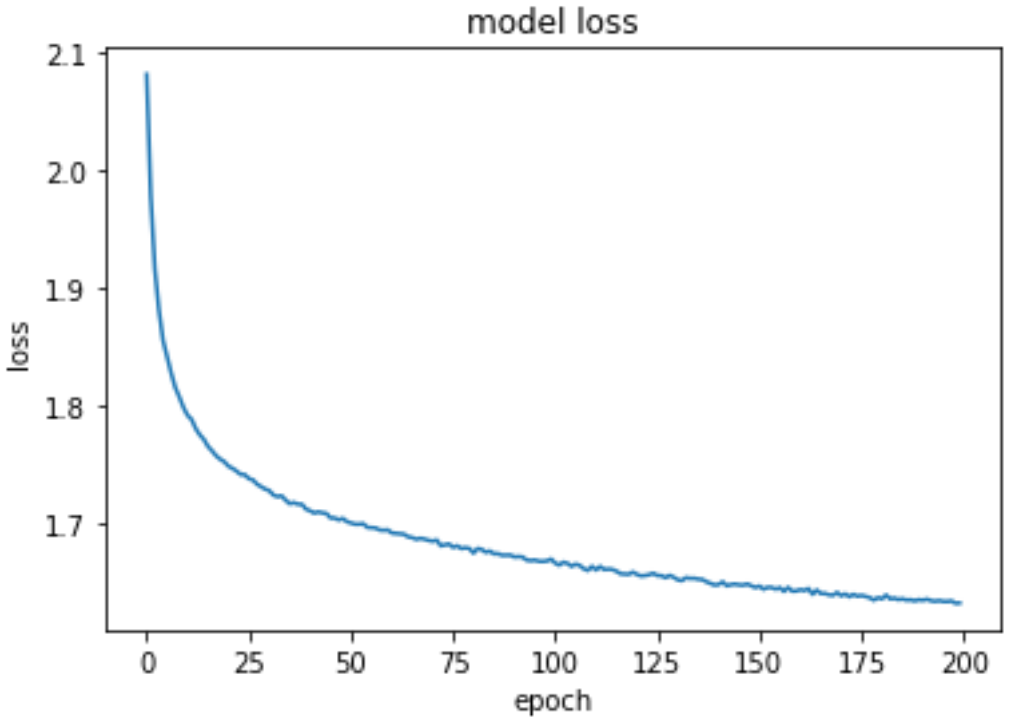
شکل 83. ماتریس آشفتگی با **PCA**

```
recall is : 0.4591

precision is : 0.456899694570345

f1 is : 0.456761691412702 23
```

قسمت ب) بعد از آموزش شبکه به وسیله Autoencoder نتایج زیر به دست آمد :



شکل 84. نمودار تغییرات دقت با Autoencoder



شکل **85**. نمودار تغییرات خطا با **Autoencoder**

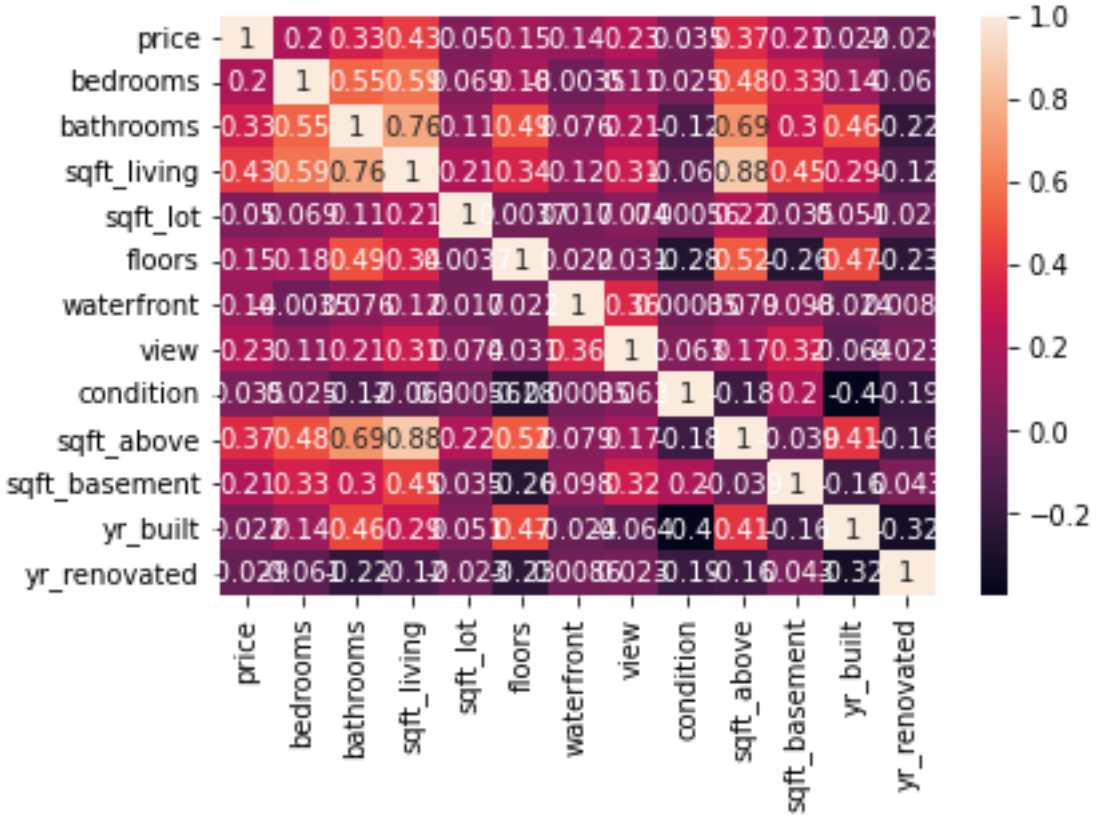
قسمت ج)

	دقت داده‌های تست	خطای داده‌های تست
بهترین شبکه سوال یک	0.39	1.69
PCA	0.45	1.55
Autoencoder	0.41	1.63

به طور کلی از کاهش بعد برای از بین بردن فضای پوچی استفاده می کنیم بدون آنکه اطلاعات مفید و مورد نیاز ما از بین برود. به طوری با عکس این عمل بتوانیم به بعد اولیه برسیم. استفاده از PCA نسبت به روش Autoencoder کارایی بهتری دارد ولی باید توجه داشت که از PCA برای کاهش بعد خطی می‌توان استفاده کرد.

قسمت د)

در ماتریس همبستگی متغیرهای ما همان ویژگی‌های مجموعه داده هستند. منظور از همبستگی بین دو متغیر، اندازه‌گیری میزان پیش‌بینی مقدارهای یکی براساس دیگری است. به این معنی که هر چه ضریب همبستگی بیشتر باشد، امکان پیش‌بینی مقدار یکی از متغیرها برحسب دیگری بیشتر است.

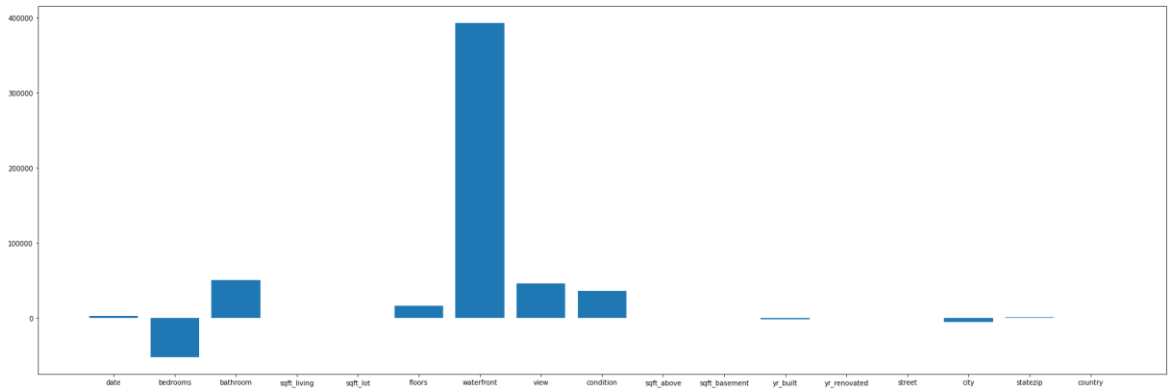


شکل 86. ماتریس همبستگی دیتاست قیمت خانه

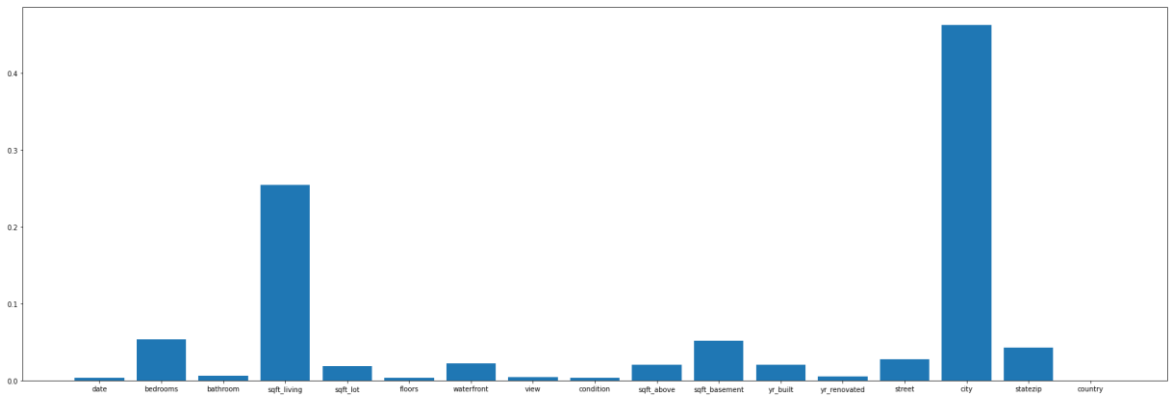
قسمت و) بارپلات مربوط به اهمیت ویژگی‌ها :

نام ستون‌ها به ترتیب در بارپلات از چپ به راست :

```
('date','bedrooms','bathroom','sqft_living','sqft_lot','floors','wate
rfront','view','condition','sqft_above','sqft_basement','yr_built',
'yr_renovated','street','city','statezip','country')
```



شکل 87. نمودار اهمیت ویژگی‌ها با Linear Regression



شکل 88. نمودار اهمیت ویژگی‌ها با Decision Tree