# تمرین اول شبکه عصبی

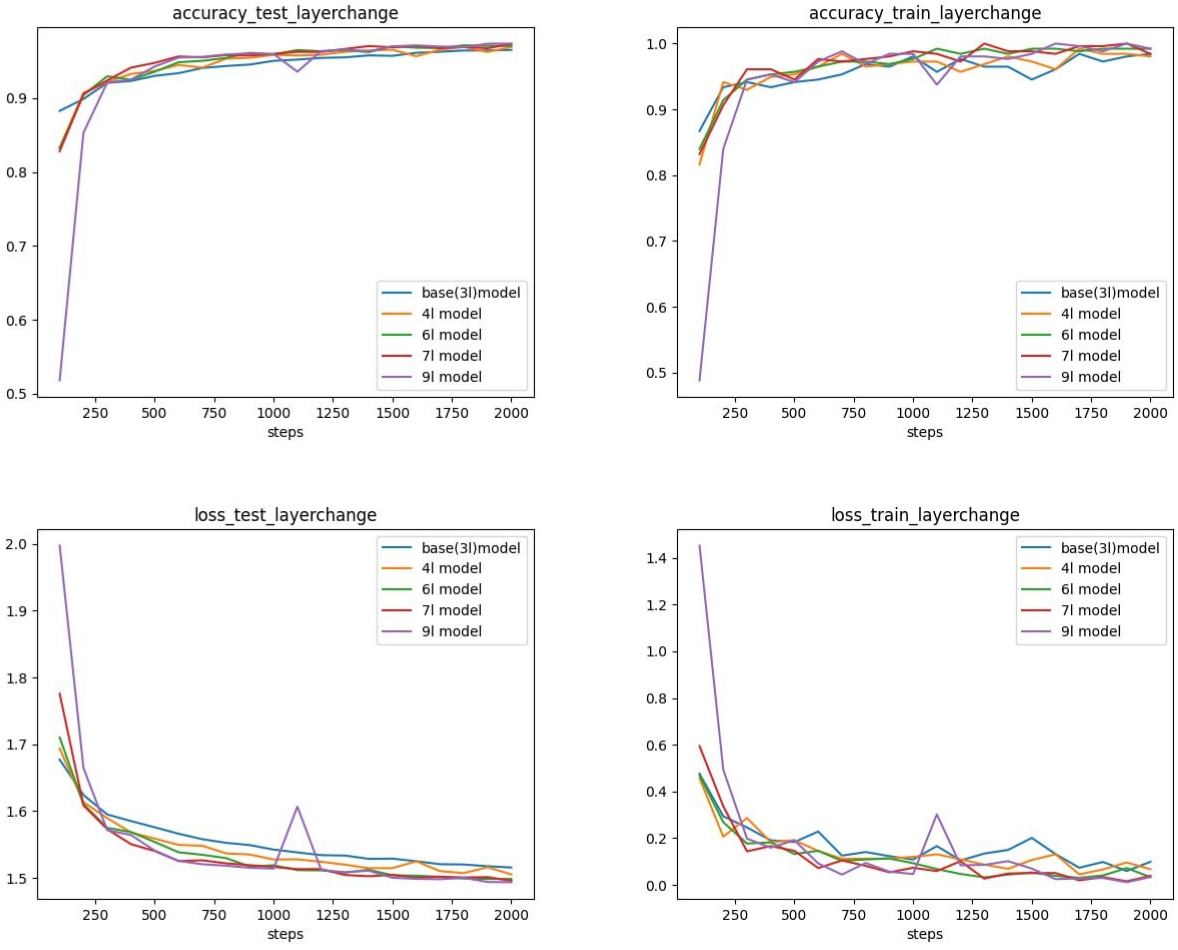
سارا قوام‌پور ۹۸۱۲۷۶۲۷۸۱

هدف این تمرین بررسی اثرات پارامتر های مختلف شبکه بر روی عملکرد آن میباشد.

## بخش اول)

### ۱- تاثیر تغییر تعداد لایه ها

با تغییر لایه‌ها در یک شبکه عصبی، توانایی یادگیری شبکه افزایش پیدا می‌کند و می‌تواند تابع های پیچیده تری را یادگیری کند ولی از طرفی ممکن است باعث این شود که تابعی خیلی پیچیده برای داده های ساده یادگیری شود و شبکه داده‌های train یا حفظ کند و یادگیری به معنای واقعی رخ ندهد. برای همین در این قسمت، تعداد لایه های مختلف برای ساخت شبکه در نظر گرفته شده است و از دیتاست MNIST و SGD به عنوان optimizer استفاده شده است.

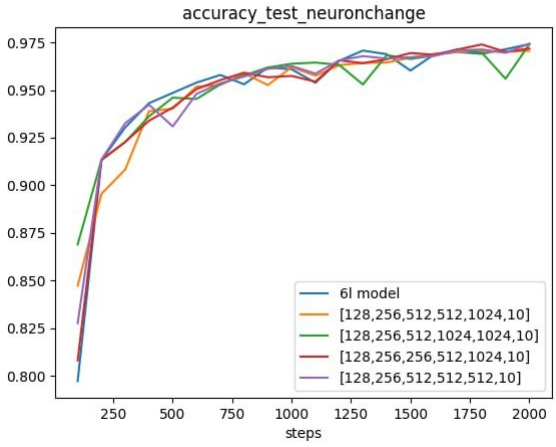


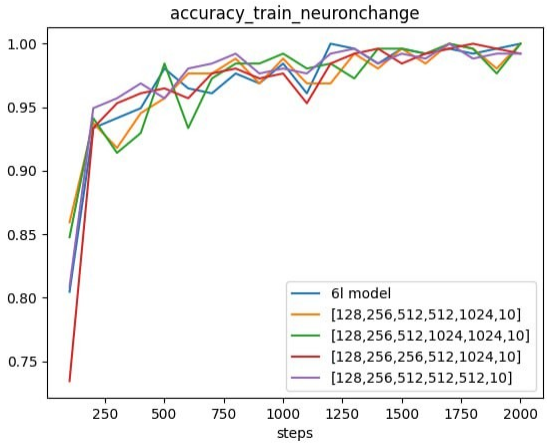
همانطور که مشاهده می‌شود, افزایش لایه ها باعث شده باعث افزایش مقدار جزئی در accuracy هم در دیتاست آموزش و هم تست شده است. اما شیب نمودار های به عبارتی سرعت روند یادگیری در شبکه هایی که لایه های بیشتر داشته اند, کند تر است و شبکه های عمیق دیرتر به همگرایی رسیده اند.
افزایش لایه ها امکان یادگیری توابع پیچیده تر را فراهم میکند اماروند یادگیری هم بیشتر طول میکشد وهمچنین به راحتی میتوانند به overfit هم منجر شود بنابراین همیشه زیاد کردن لایه ها بهترین جواب نیست اما همانطور که در ۴ شکل بالا دیده میشود, نتایج بهتری را در بردارد اما اگر شبکه پیش از حد لایه های زیادی داشته باشد overfit میشود.

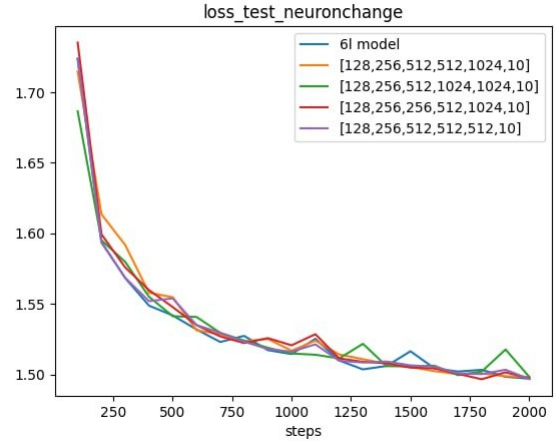
در این مرحله مدل با ۶ لایه به عنوان بهترین مدل انتخاب شد تا در قسمت های ۲ و ۳ بررسی ها بر روی همین مدل ۶ لایه انتخاب شود. مدل های ۶ لایه و ۱۰ لایه در این قسمت بهترین نتایج را ارائه داده اند اما به علت راحت تر بودن آزمایش های مرحله های ۲ و۳ بر روی مدل ۶ لایه در قسمت های بعد بررسی انجام میشود.

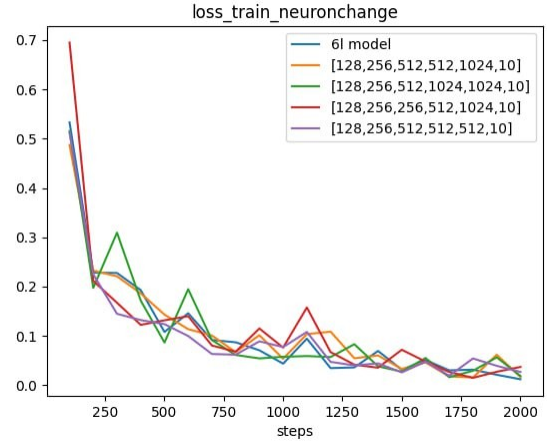
## ۲- تاثیر تغییر تعداد نورون‌ها

در این مرحله، تعداد نورون‌های موجود در لایه های شبکه انتخابی مرحله قبل را تغییر می‌دهیم تا هم تاثیر تغییر تعداد نورون‌ها را بررسی کرده و هم بهترین تعداد را برای شبکه انتخاب کنیم.



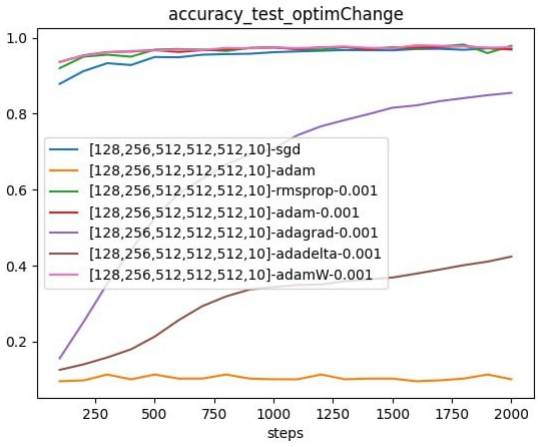



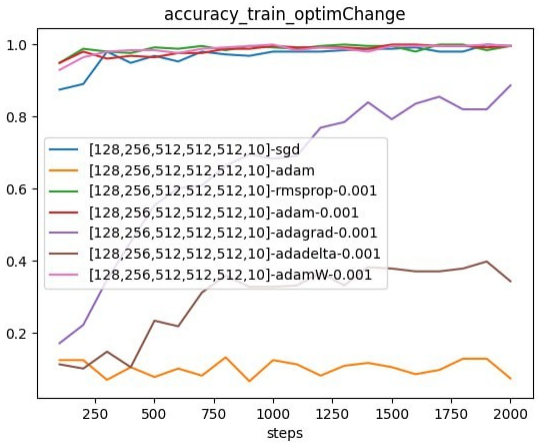



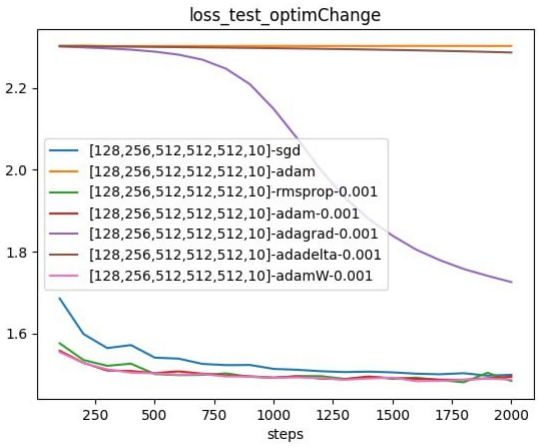
همانطور که مشاهده می‌شود، نمودار بنفش که مربوط به شبکه با لایه های مخفی با تعداد نورون های کمتر نسبت به بقیه شبکه ها، بهتر عمل کرده است و یادگیری . با افزایش تعداد نورون‌ها، قابلیت شبکه برای تشخیص الگوهای پیچیده بیشتر می‌شود ولی احتمال overfit شدن هم افزایش پیدا می‌کند و باید تعداد نورون های مناسبی برای شبکه انتخاب کنیم. تعداد نورون ها در هرلایه بستگی به سایز مسئله و تعداد لایه ها دارد و در صورت وجوپد داشتن تعداد نورن های زیاد در لایه های متعدد میتواند منجر به overfit شود.مدل بنفش به عنوان بهترین مدل در قسمت ۳ بررسی میشود.

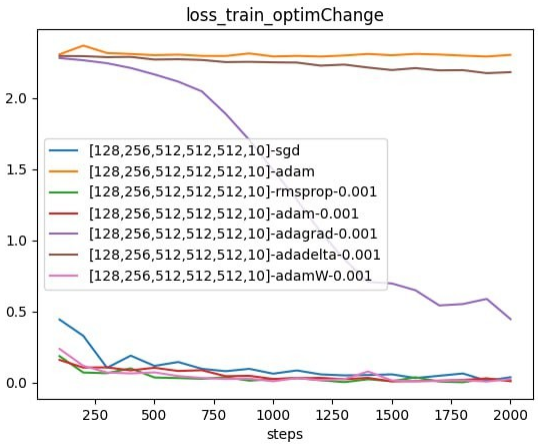
## ۳- تاثیر تغییر optimizer

در این قسمت، ۷ آپتیمایزر برای یادگیری شبکه منتخب دو بخش قبلی امتحان شده‌اند و بهترین آن‌ها انتخاب شده است.



الگوریتم Adagrad بر اساس میزان انتخاب یک پارامتر، نرخ یادگیری را تنظیم می‌کند به این صورت که اگر یک پارامتر میزان زیادی انتخاب شود، نرخ یادگیری برای آن کاهش می‌یابد و آپدیت های کوچک تری روی آن انجام می‌شود.

الگوریتم Adadelta یک الگوریتم مبتنی بر SGD می‌باشد که از adaptive learning استفاده می‌کند. این الگوریتم ورژن ارتقا یافته Adagrad می‌باشد که به جای استفاده از تمامی گرادیان های قبلی، از یک پنجره متحرک برای آپدیت های گرادیان استفاده می‌کند و به این صورت Adadelta می‌تواند به یادگیری ادامه دهد.

الگوریتم AdamW هم یک الگوریتم مبتنی بر SGD می‌باشد که از adaptive estimation برای مومنت های مرتبه اول و دوم در کنار متدی برای کاهش وزن استفاده می‌کند.

همانطور که مشاهده می‌شود الگوریتم AdamW از بقیه optimizer ها به accuracy بیشتری رسیده است.

SGD با نرخ یادگیری ۰.۱ عملکرد مناسبی دارد اما adam با نرخ یادگیری ۰.۱ کمترین عملکرد را دارد. SGD با نرخ یادگیری بزگتری مانند ۰.۱ به همگرایی میرسد درحالی که بقیه الگوریتم ها که در بالا تست شده اند همگی با نرخ یادگیری کوچی مانند ۰.۰۰۱ به همگرایی میرسند.

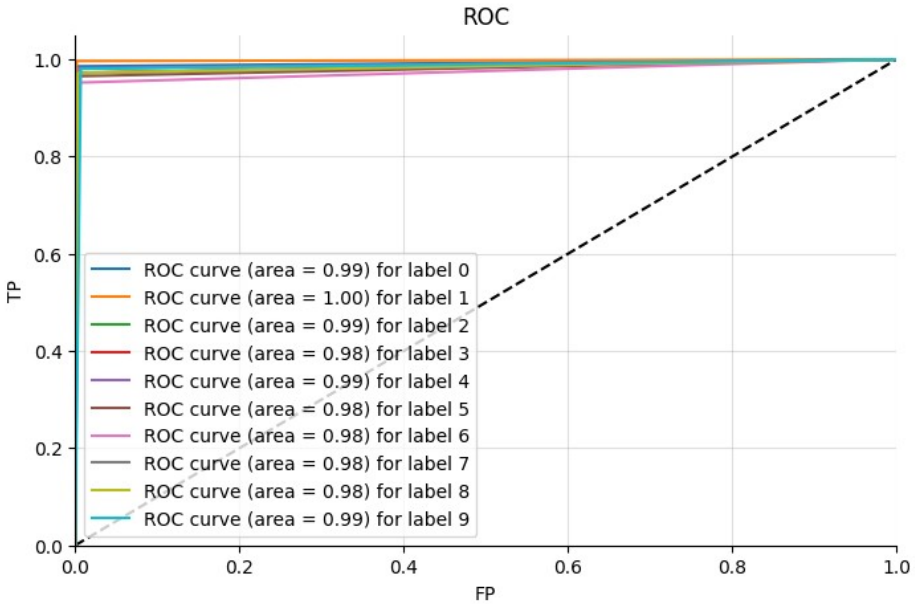
الگوریتم های AdamW و rmsprop نتایج نزدیکی نسبت به یکدیگر دریافت کرده اند.

در نتیجه الگوریتم های AdamW و rmsprop بر روی مدل ۶ لایه که به عنوان بهترین مدل از قسمت ۲ انتخاب شد و مدل عمیقی است نسبت به بقیه مدل ها عملکرد خوبی داشتند و AdamW به بهترین نتیجه رسید.

در نتیجه بهترین کانفیگی که به آن رسیدیم ۶ لایه و با تعداد نورون های [128,256,512,512,512,10] برای هرلایه و optimizer آن AdamW میباشد.

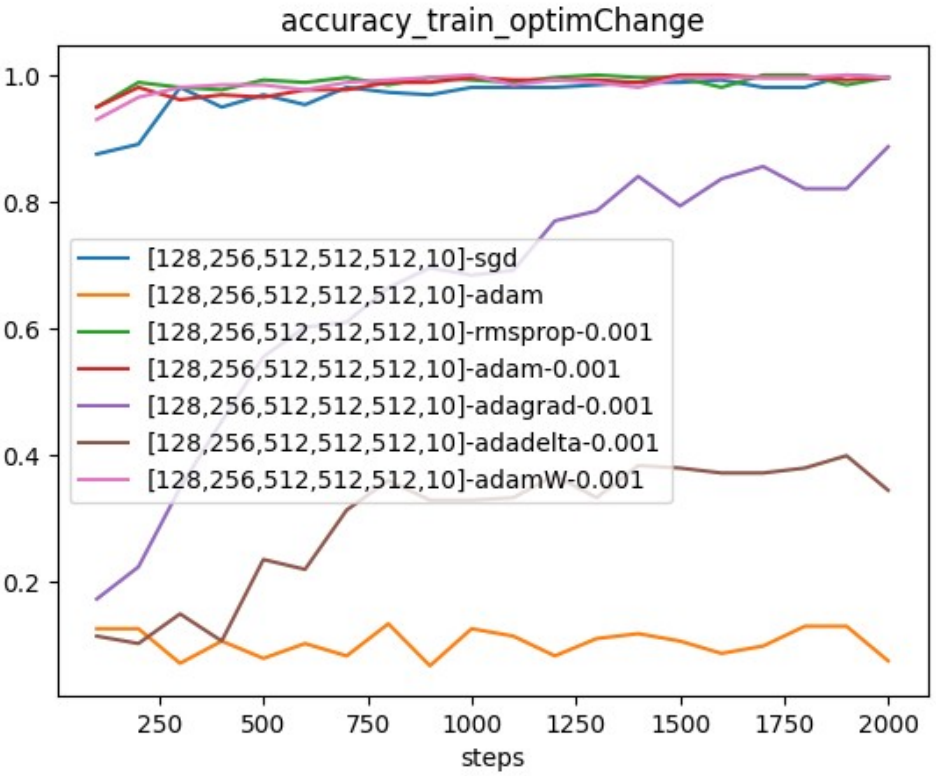
۰.۹۷۵۴ و ۰.۹۷۵۶ و۰.۹۷۵۴ به ترتیب accuracy و precision و recall برای این مدل هستند.

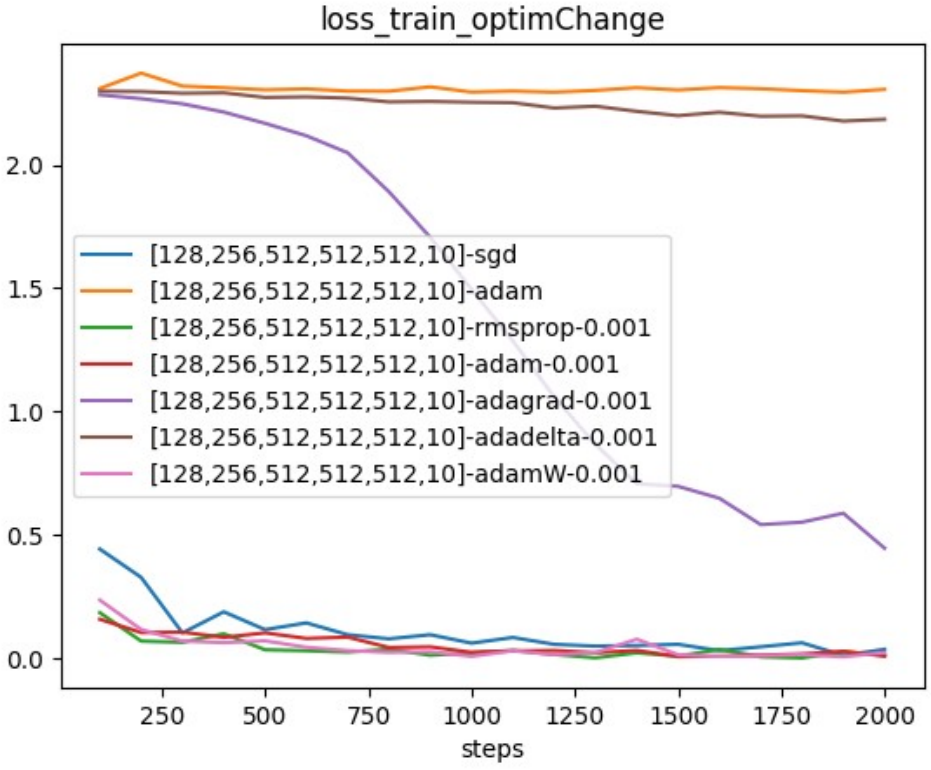
نمودار ROC برای این مدل:



نمودار یادگیری برای این مدل:

میتوان هم از loss و هم از accuracy استفاده کرد.



## ۴- بخش دوم

وقتی شبکه با داده های کلاس های ۰-۴ اموزش داده شد بر روی داده های دیتاست تست که لیبل ان ها ۰-۴ است اکیورسی بالا به اندازه ۰.۹۹۰۲۷۰ داشت اما برای داده های با لیبل کلاس های ۵-۹ که در فرایند اموزش موجود نبودند عملکرد خوبی ندارد. شبکه چون داده های با کلاس ۵-۹ را ندیده است و بر روی آن ها اموزش ندیده است و درمورد ان ها اطلاعاتی ندارد در هنگام مواجعه با ان ها سعی میکند تا یکی از کلاس هایی که میشناسد(۰-۴) را به ان ها نسبت دهد که منجر به عملکرد ضعیف میشود.

```
[71]  1 print('y_test_5_2')
      2 np.array(y_test_5_2[0:10])
```

```
y_test_5_2
array([7, 9, 5, 9, 6, 9, 5, 9, 7, 9], dtype=uint8)
```

```
[70]  1 print('y_pred_5_2')
      2 np.array(y_pred_5_2[0:10])
```

```
y_pred_5_2
array([3, 2, 2, 4, 0, 4, 3, 4, 0, 1])
```

خروجی اول مربوط به لیبل درست برای این داده است و خروجی اول مربوط به جواب شبکه برای این ورودی.
مشاهده میشود که جواب های شبکه بسیار متفاوت با جواب ها درست هستند.
راه حل های پیشنهادی در مواجهه با چنین شرایطی به صورت زیر میباشد:

۱- شناسایی outlier ها با با استفاده از تکنیک های unsupervised مانند کلاسترینگ و حذف ان ها.

۲- data augmentation : ایجاد کردن داده های جدید با اعمال تبدیل های متفاوت بر روی تصاویر مانند چرخاندن, برگرداندن و scaling که باعث ایجاد داده های متنوع میشود که میتواند تا حدی پاسخگو برای مشکل مطرح شده باشد.