به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری دوم

نام و نام خانوادگی	علی ایزدی
شماره دانشجویی	۸۱۰۱۹۹۱۰۲
تاریخ ارسال گزارش	۱ آذر ۱۴۰۰

نکات پیاده سازی

۱- پیاده سازی با استفاده از pytorch انجام شده است.

۲- کدهای مربوط به سوالات ۱، ۲، ۳ در پوشه های Q1, Q2, Q3 قرار گرفته است.

۳- برای سوالات ۱ و ۲ مدل پایتورچ در فایل پایتون با نام models.py قرار گرفته و در فایل jupyter با نام results.ipynb از مدل استفاده شده و نتایج مربوط به هر یک از قسمت های خواسته شده در هر بخش سوال اجرا و تحلیل شده است.

۴- برای سوال ۳ نیز مدل پایتورچ Auto Encoder و PCA در فایل هایی پایتون با همین نام پیاده سازی شده اند. هم چنین در دو فایل jupyter برای دو حالت مسئله classification و regression نتایج کاهش ابعاد به دو روش Auto Encoder و PCA اجرا و تحلیل شده است. هم چنین در این سوال از همان پیاده سازی مدل های سوال ۱ و ۲ از ماژول Q1 و Q2 استفاده شده است.

۵- هم چنین export شده فایل های jupyter در کنار هر فایل jupyter قرار دارد.

در ادامه به تحلیل نتایج خواهیم پرداخت.

سوال ۱- شبکه پرسپترون چند لایه در کاربرد طبقه بندی

الف)

تقسیم بندی داده های آموزش و تست و ارزیابی

- برای تقسیم بندی داده های آموزش و ارزیابی و تست به ترتیب ۶۰ و ۲۰ و ۲۰ درصد داده ها به هر دسته تعلق گرفت. میتوان با درصدهای دیگری نیز این کار را انجام داد که درصدهای استفاده شده جز اعداد معمول است.
- قبل از تقسیم بندی نیز داده ها shuffle شده اند تا اگر ترتیب داده ها دارای biasای باشد از بین برود.

- با توجه به این که ممکن است تفاوت تعداد هر کلاس متفاوت باشد بنابراین به روش stratify داده ها تقسیم میشوند تا در هر دسته به نسبت همان داده های هر کلاس در هر دسته قرار گیرد تا این موضوع تفاوتی در داده های آموزش و تست و ارزیابی ایجاد نکند.

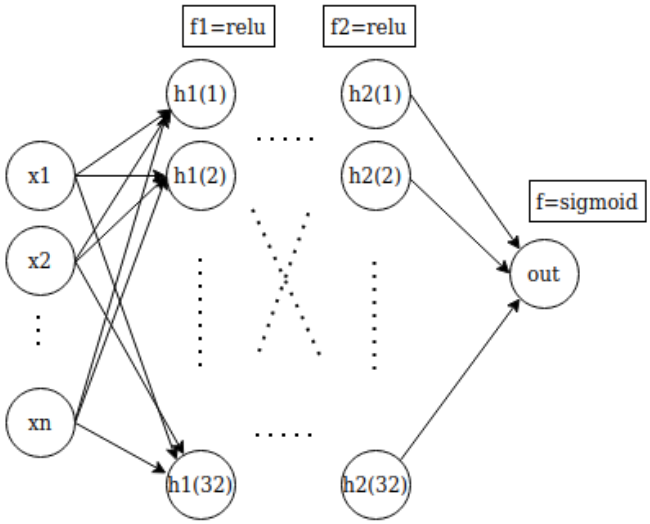
- **برای مقایسه مدل های مختلف نیز مقدار loss بر روی داده های validation در نظر گرفته شده است.**

پیش پردازش:

- با توجه به رنج داده ها و عدم تفاوت بین scale آن ها نیازی به استاندارد سازی آن ها نبود.
- دسته g و b نیز به اعداد 0 و 1 مپ شدند تا پایتورچ دسته ها را شناسایی کند.

مدل:

در قسمت اول مدل ساده با دو لایه مخفی مطابق با معماری زیر پیاده سازی شد. که در آن از تابع فعال ساز relu در دو لایه مخفی و sigmoid برای مسئله دسته بندی در نود خروجی استفاده شده است تا عددی بین صفر و یک تولید کند که اگر بزرگتر از 0.5 بود مربوط به دسته 1 و اگر کوچک تر از 0.5 بود مربوط به دسته 0 است.



خروجی مدل پایتورچ نیز در زیر آورده شده است.

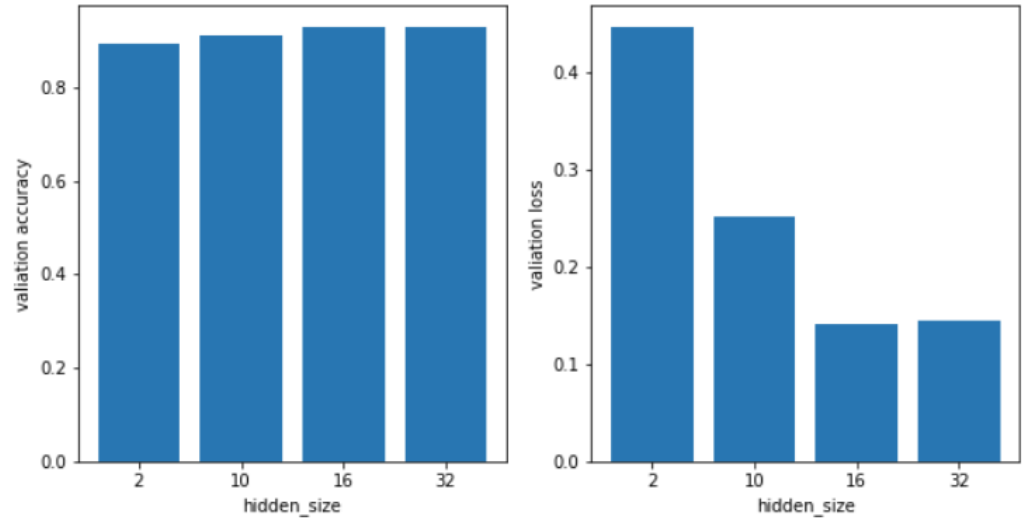
```
MLP(
  (fc1): Linear(in_features=34, out_features=32, bias=True)
  (hidden_layers): ModuleList(
    (0): Linear(in_features=32, out_features=32, bias=True)
  )
  (fc_last): Linear(in_features=32, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

ب)

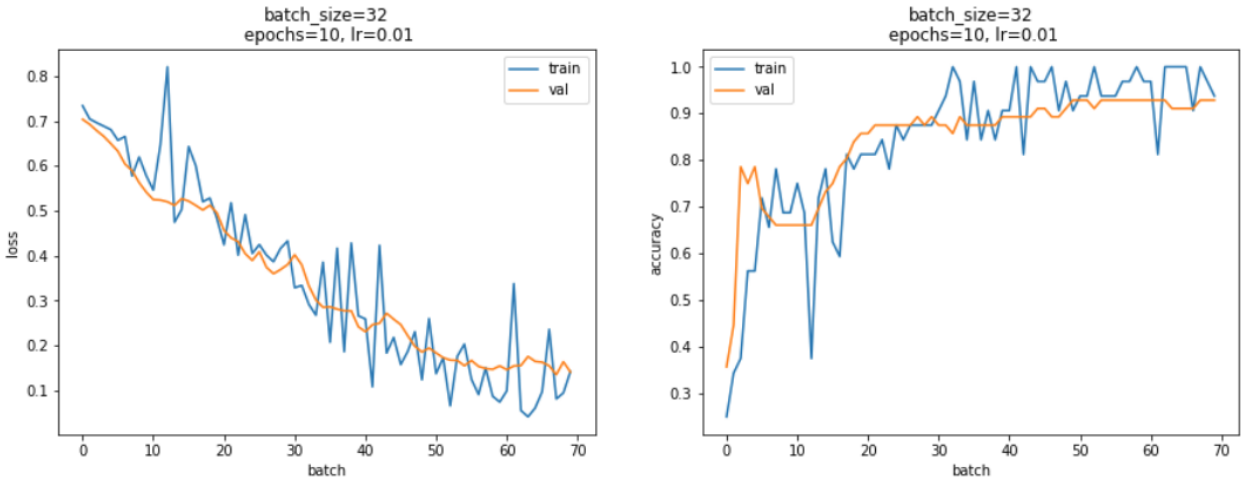
مقدار batch برابر ۳۲ و تعداد نورون های لایه مخفی ۴ حالت ۲ و ۱۰و ۱۶ و ۳۲ در نظر گرفته شد. Loss نیز Binary Cross Entropy و optimizer نیز Adam و epoch ها ۱۰ در نظر گرفته شده است.

نمودار خطا و دقت بر روی داده های ارزیابی برای حالت های مختلف نورون های لایه مخفی در زیر آورده شده است.

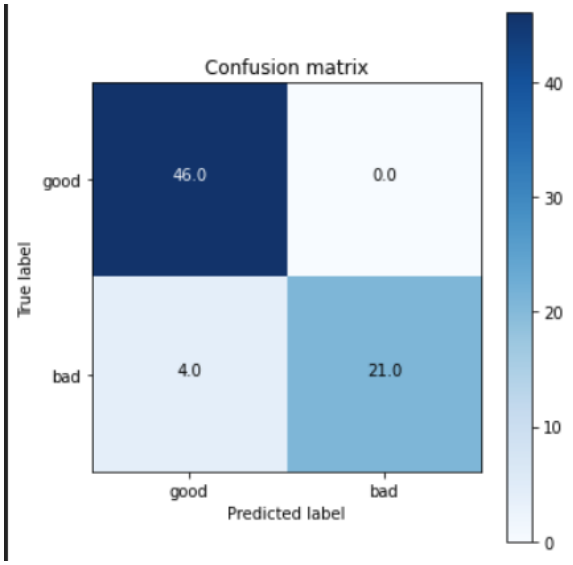
همان طور که مشاهده میشود تعداد نورون های ۳۲ بهترین دقت و کمترین خطا را داشته است. که به دلیل مدل سازی بهتر برای داده های موجود با ابعاد 34 است. بنابراین از این به بعد تعداد نورون ها را ۳۲ در نظر میگیریم.



هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که تعداد نورون های ۳۲ است در زیر آورده شده است. همان طور که مشاهده میشود تعداد epoch=10 منجر به overfit نشده است و بیشتر از با توجه به آزمایش انجام شده منجر به بیشتر شدن خطای validation و overfit میشد.



در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت ۹۴ درصدی الگوریتم بر روی داده های تست دارد.



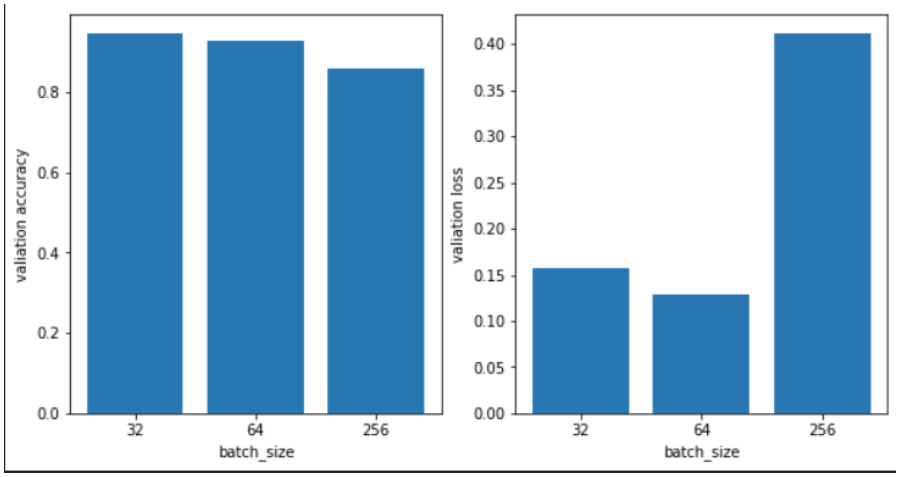
ج)

در ادامه حالات مختلف **batch** به ازای **۳۲ و ۶۴ و ۲۵۶** تست شده است.

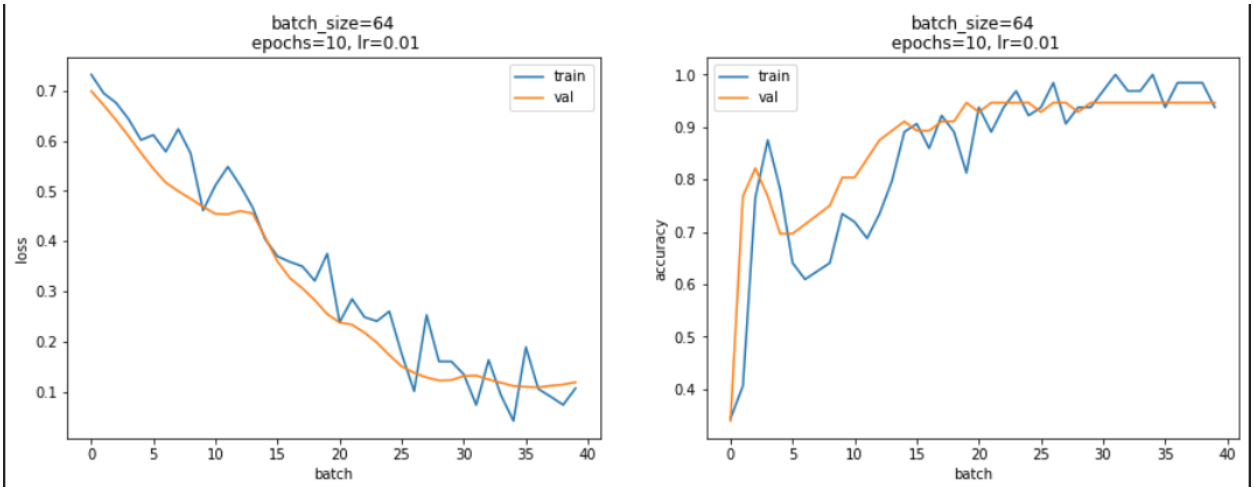
نمودار خطا و دقت بر روی داده های ارزیابی برای حالت های مختلف batch در زیر آورده شده است.

همان طور که مشاهده میشود تعداد batch=256 منجر به کم شدن دقت شده است اما بهترین تعداد batch=64 است که از این به بعد در نظر گرفته شده است.

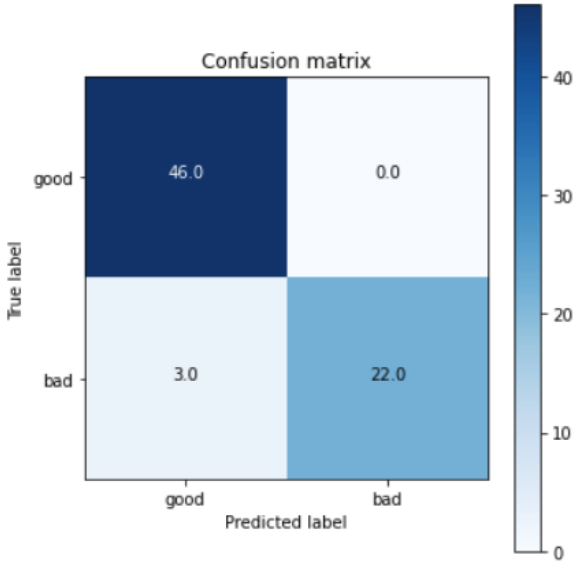
افزایش تعداد batch زمان آموزش را کاهش میدهد اما منجر به کاهش generalization میشود که نتایج نیز همین موضوع را تصدیق میکند.



هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که تعداد نورون های ۳۲ است در زیر آورده شده است.



در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت 95 درصدی الگوریتم بر روی داده های تست دارد.

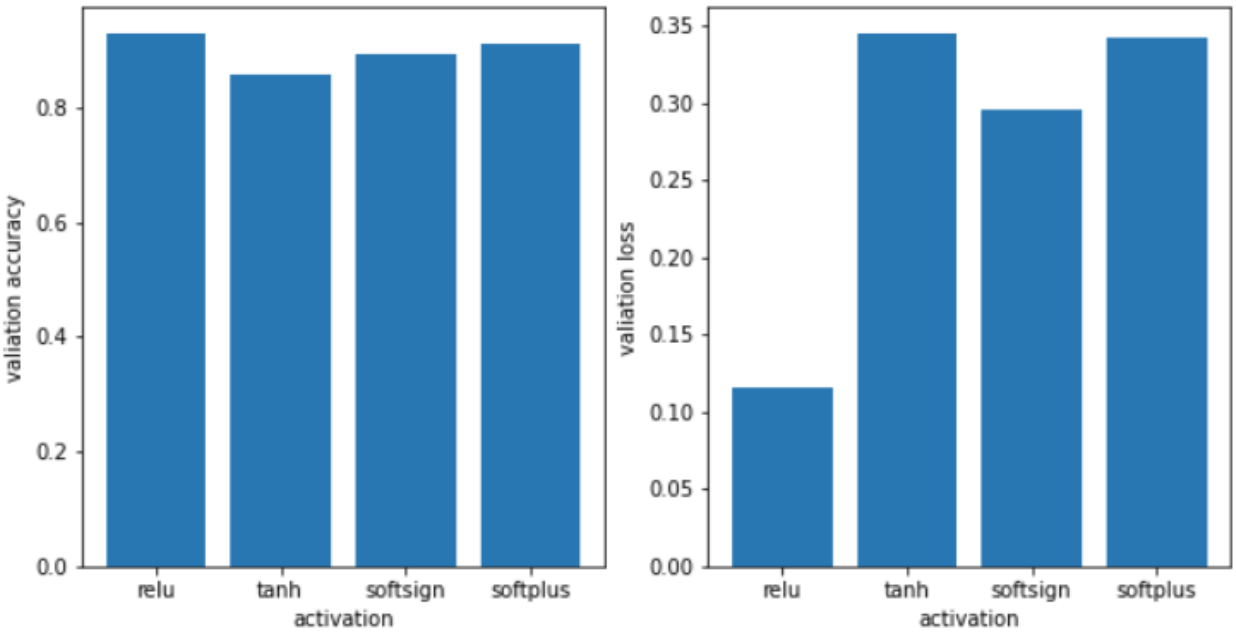


د)

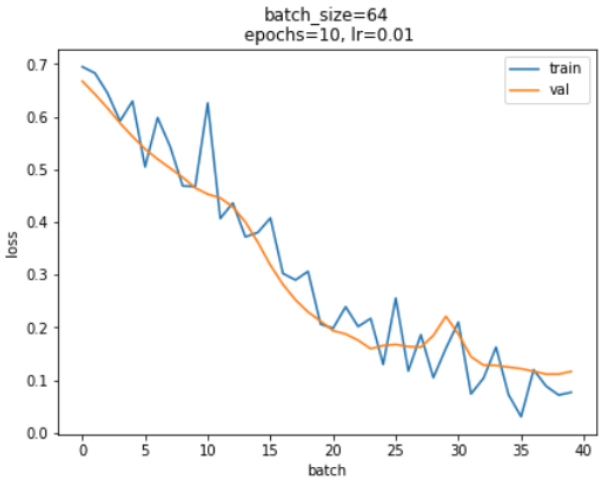
در ادامه حالات مختلف **تابع فعال ساز** به ازای **relu, tanh, softsign, softplus** تست شده است.
نمودار خطا و دقت بر روی داده های ارزیابی برای حالت های مختلف تابع فعال ساز در زیر آورده شده است.

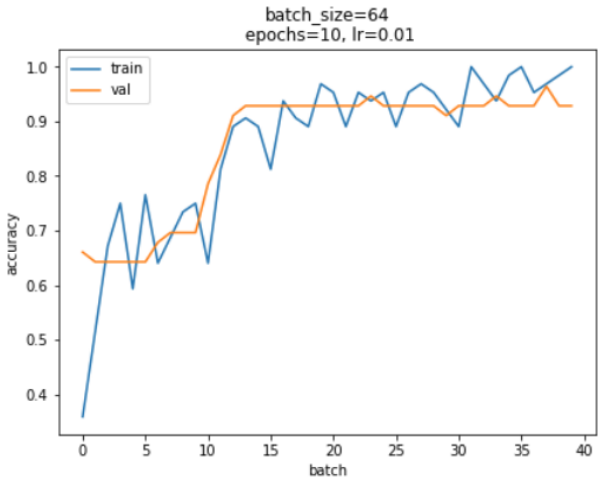
همان طور که مشاهده میشود تابع **فعال ساز relu** بهترین نتیجه را داشته است که از این به بعد در نظر گرفته شده است.

تابع فعال ساز relu نسبت به دیگر توابع فعال ساز مقادیر کمتر از صفر را برابر با صفر میکند که مزیت هایی از جمله جلوگیری از vanishing gradient دارند. هم چنین به خاطر ساده تر بودن آن نسبت به غیر خطی بودن دیگر توابع فعال ساز مدل سازی general تری برای این داده ایجاد کرده است.

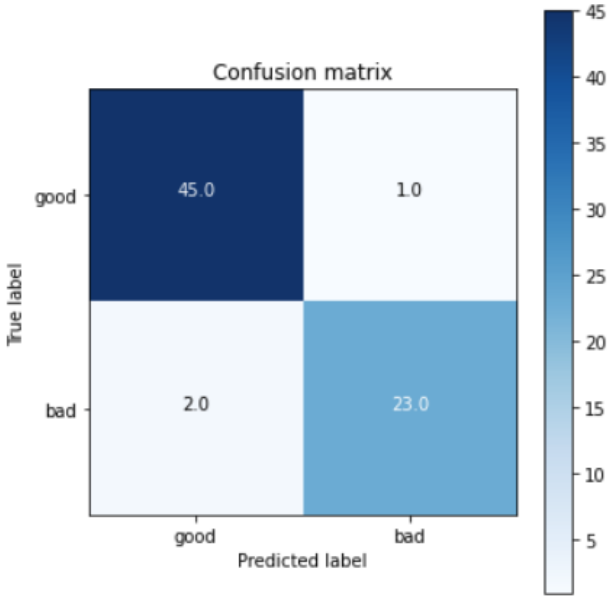


هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که **تابع فعال ساز relu** است در زیر آورده شده است.





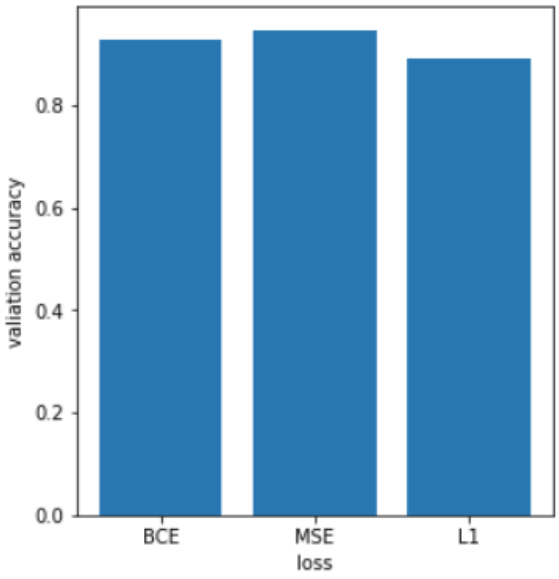
در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت 95 درصدی الگوریتم بر روی داده های تست دارد.



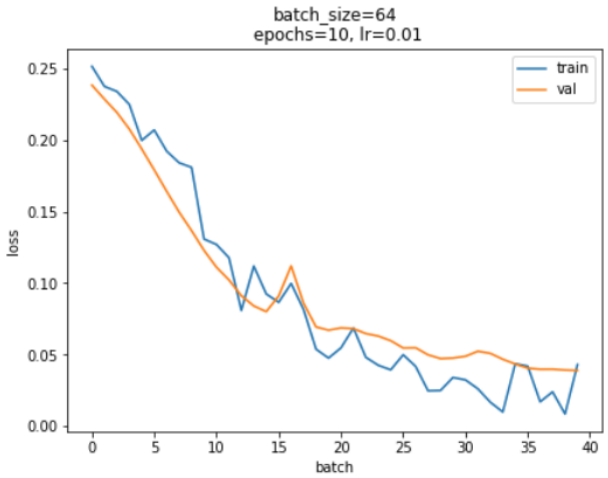
ه)

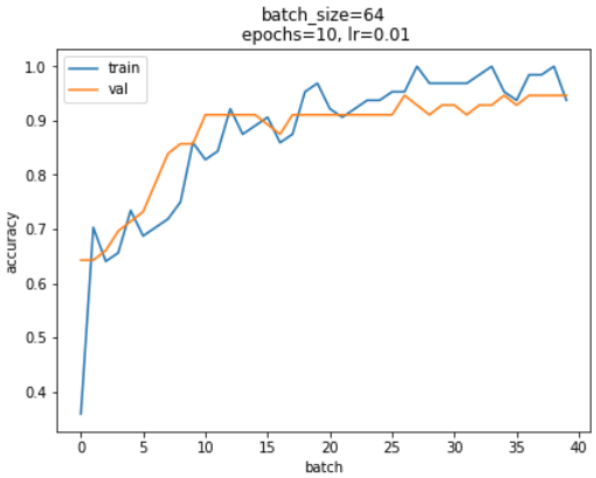
در ادامه حالات مختلف **تابع loss** به ازای **BCE, MSE, MAE** تست شده است.
نمودار دقت بر روی داده های ارزیابی برای حالت های مختلف تابع فعال ساز در زیر آورده شده است.
همان طور که مشاهده میشود تابع **loss=MSE** بهترین نتیجه را داشته است که از این به بعد در نظر گرفته شده است.

- MSE از منظر ریاضی تابع likelihood با فرض گوسی بودن متغیر target را بهینه میکند.
- MAE نیز همان فرض های MAE را دارد با این تفاوت که به وجود outlier در داده ها robust تر است.
- BCE نیزاز منظر ریاضی likelihood را به طور کلی بهینه میکند و هدف آن کم کردن میانیگن تفاوت کلاس پیش بینی شده و حقیقی برای پیش بینی کلاس ۱ است.

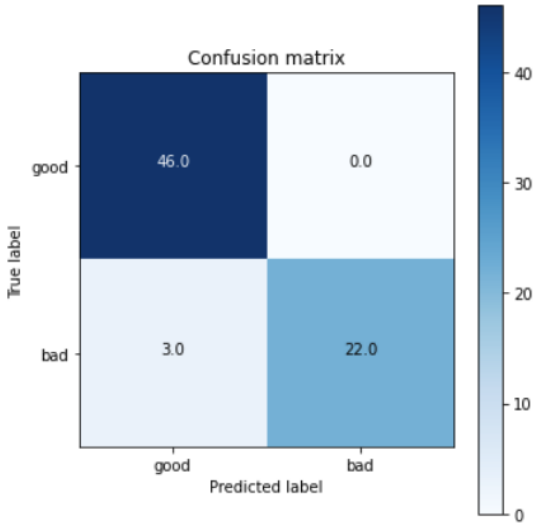


هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که **تابع loss=MSE** است در زیر آورده شده است.



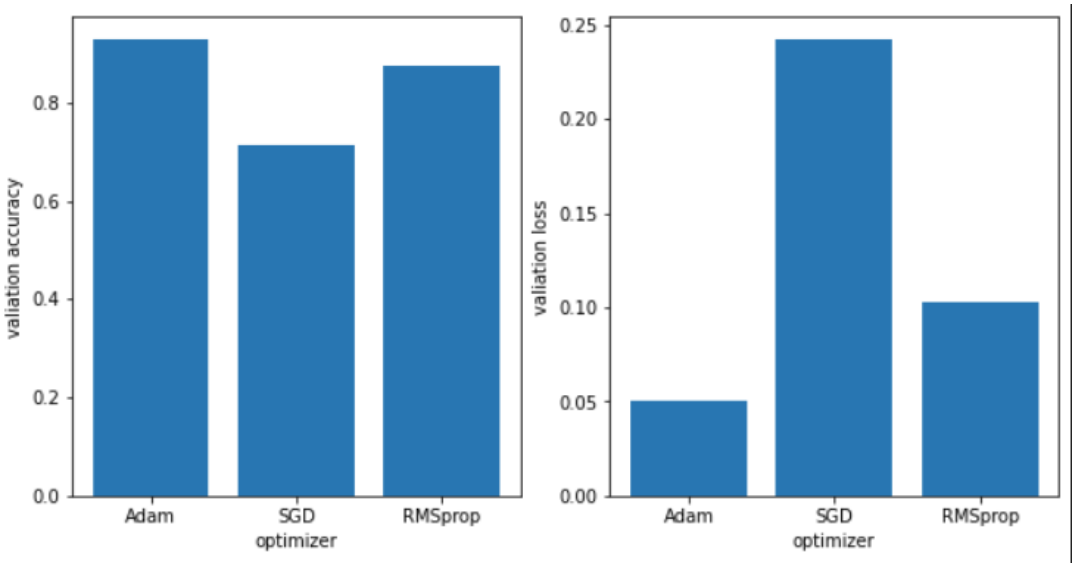


در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت 95 درصدی الگوریتم بر روی داده های تست دارد.

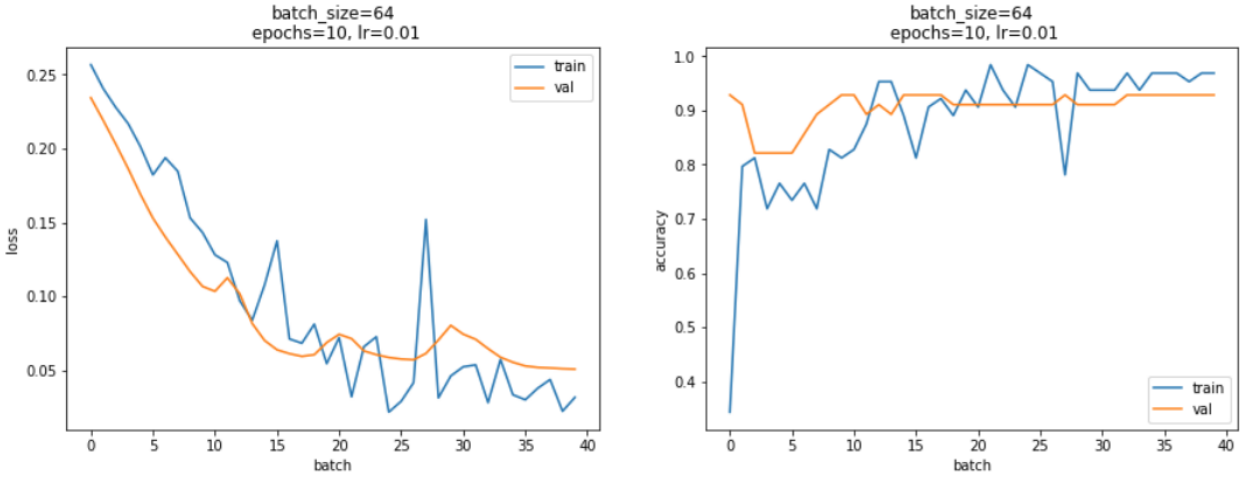


و)

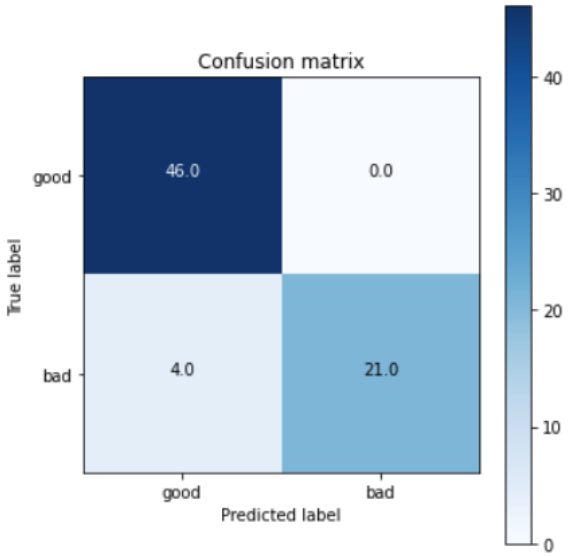
در ادامه حالات مختلف **روش بهینه سازی** به ازای **Adam, SGD, RMSprop** تست شده است.
نمودار دقت بر روی داده های ارزیابی برای حالت های مختلف روش بهینه سازی در زیر آورده شده است.
همان طور که مشاهده میشود روش **بهینه ساز Adam** بهترین نتیجه را داشته است که از این به بعد در نظر گرفته شده است.



هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که **روش بهینه سازی Adam** است در زیر آورده شده است.



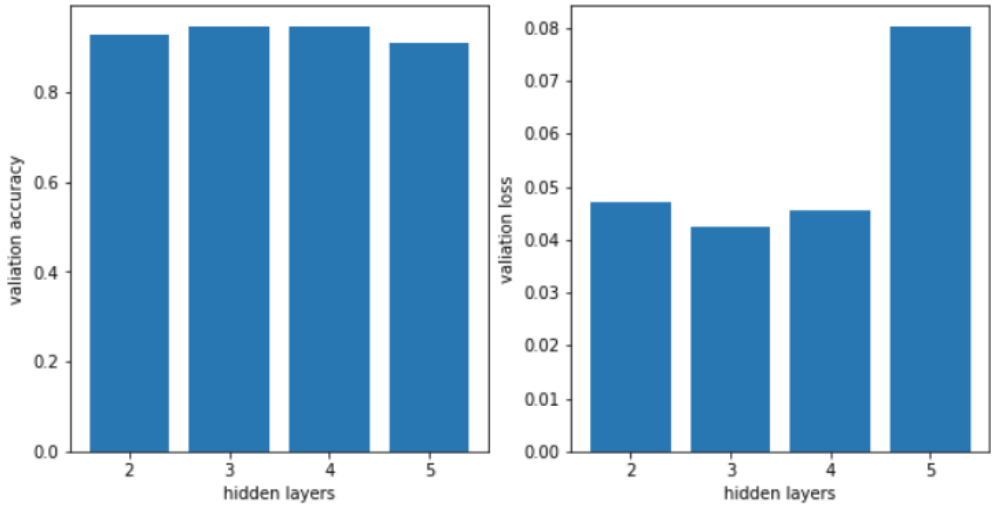
در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت 94 درصدی الگوریتم بر روی داده های تست دارد.



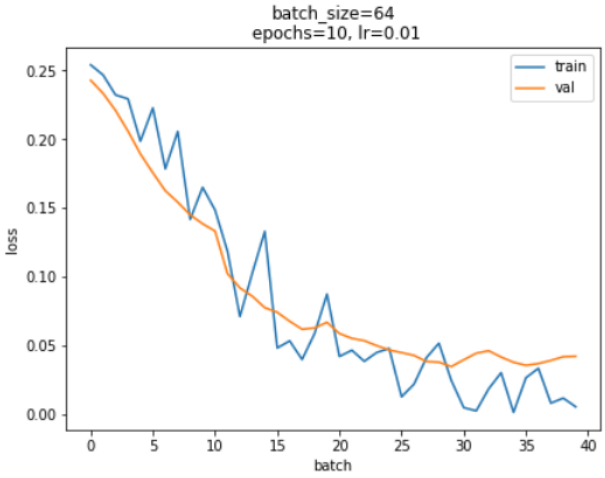
ح)

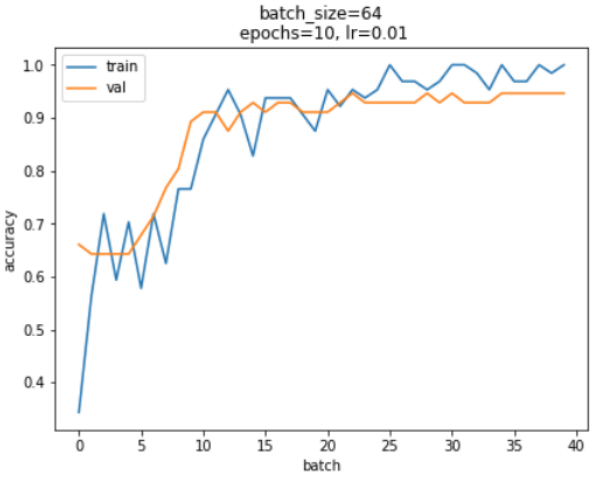
در ادامه حالات مختلف **لایه نهان** به ازای **2, 3, 4, 5** تست شده است.

نمودار دقت بر روی داده های ارزیابی برای حالت های مختلف تعداد لایه نهان در زیر آورده شده است. همان طور که مشاهده میشود **تعداد لایه نهان=۳** بهترین نتیجه را داشته است که از این به بعد در نظر گرفته شده است. همان طور که مشاهده میشود با اضافه کردن یک لایه از ۲ به ۳ دقت افزایش یافته است اما با افزایش بیشتر دقت کاهش یافته باشد که میتواند به علت زیاد شدن تعداد پارامترهای مدل و کاهش generalization آن یا نیاز به بهینه سازی بیشتر و متفاوت برای لایه های بیشتر باشد

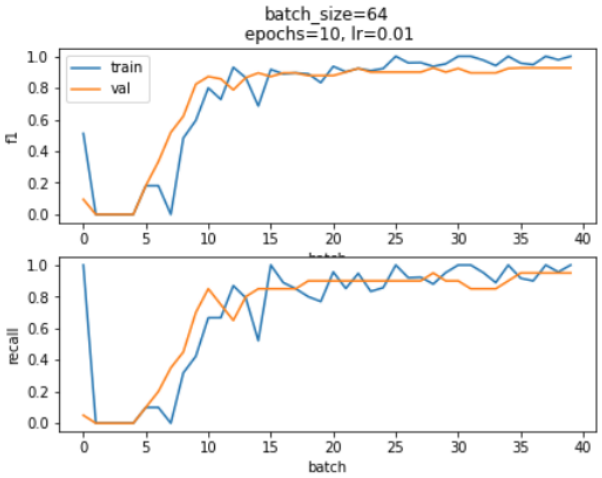


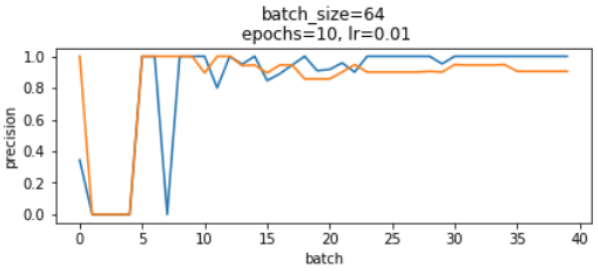
هم چنین نمودار خطا و دقت بر روی داده های ارزیابی و آموزش برای بهترین حالت که **تعداد لایه نهان ۳** است در زیر آورده شده است.



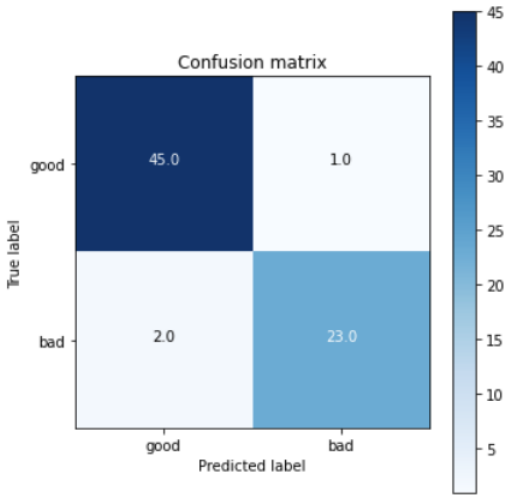


هم چنین تغییرات معیارهای precision, recall, f1 بر روی داده های ارزیابی و آموزش در زیر آورده شده است.





در زیر نیز ماتریس confusion برای داده های تست رسم شده است که نشان از دقت 95 درصدی الگوریتم بر روی داده های تست دارد.



نتایج معیارهای خواسته شده بر روی داده تست در زیر آورده شده است.

test accuracy: 0.95
test f1: 0.93
test precision: 0.95
test recall: 0.92

ط)

با توجه به قسمت های قبل پارامترهای زیر به عنوان بهترین پارامترها انتخاب میشود.

number of hidden neurons: **32**
batch_size: **64**
activation function: **relu**
loss: **MSE**
optimizer: **Adam**
number of hidden layers: **3**

epoch: 10
learning_rate: 0.01

تعداد نورون های ۳۲ نزدیک به ابعاد داده است پس مدل سازی نزدیکی برای جدا سازی بر اساس ویژگی های مختلف است.

تعداد batch_size نیز نه زیاد است که منجر به generalization کم شود و نه آن قدر کم است که بهینه سازی را دچار مشکل کند.

تابع فعال ساز relu به خاطر فوایدی که منجر به بهینه سازی بهتر به خاطر جلوگیری از vanishing gradient میشود استفاده شده است.

بهینه ساز Adam نیز به خاطر وجود یادگیرنده learning_rate در خود بر sgd مزبت داشته و هم چنین به خاطر ترکیب Adadelta و RMSprop ورژن پیشرفته تری از optimizer است که از خواص این دو استفاده میکند.

تعداد لایه های نهان نیز نه آنقدر زیاد است که دچار overfit شویم و بهینه سازی سخت تر شود و نه آنقدر کم است که دچار underfit شویم/

ی)
مشکل imbalanced dataset منجر به ایجاد bias روی کلاس خاصی میشود و هم چنین معیار مناسب برای آموزش شبکه باید در نظر گرفت.

۱- استفاده از معیارهای دیگری مثل f1 به جای accuracy که پیش بینی صحیح هر دو کلاس را با استفاده از precision و recall در نظر میگیرد.

۲- استفاده از تکنیک های under sampling و upsampling داده برای کم کردن تعداد نمونه های کلاس با تعداد بیشتر و یا بیشتر کردن نمونه های کلاس با تعداد کمتر

۳- استفاده از روش های bagging و boosting که چندین مدل بر روی حالت های مختلف داده های آموزش آموزش داده میشوند.

سوال ۲- شبکه پرسپترون چند لایه در کاربرد رگرسیون

- **نکات مربوط به پیاده سازی در ابتدای گزارش آورده شده است.**

تقسیم بندی داده های آموزش و تست و ارزیابی

- برای تقسیم بندی داده های آموزش و ارزیابی و تست به ترتیب ۶۰ و ۲۰ و ۲۰ درصد داده ها به هر دسته تعلق گرفت. میتوان با درصدهای دیگری نیز این کار را انجام داد که درصدهای استفاده شده جز اعداد معمول است.
- قبل از تقسیم بندی نیز داده ها shuffle شده اند تا اگر ترتیب داده ها دارای biasای باشد از بین برود.

- **برای مقایسه مدل های مختلف نیز مقدار loss بر روی داده های validation در نظر گرفته شده است.**

- **هم چنین تابع loss تابع mean squared error در نظر گرفته شده است**

پیش پردازش:

- با توجه به این که scale ویژگی ها متفاوت است برای نرمال سازی داده ها از روش standar scaling استفاده شده است که میانگین داده ها را بر صفر و covariance آن ها را برابر covariance داده ها میکند.

قسمت ۱)

با آموزش مدل رگرسیون خطی بر روی داده نتیجه زیر برای تابع MSE بر روی داده های تست به دست می آید.

mse test for linear regression 1151.56

هدف از مدل رگرسیون خطی پیدا کردن ضریب های وزن تتا برای مدل سازی تابع زیر است.

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x$$

برای به دست آوردن این وزن ها نیاز است تا تابع هزینه MSE مطابق زیر مینیمم شود.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$$

برای minimum کردن تابع بالا از آن مطابق زیر مشتق گرفته میشود.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y}) \\
&= \frac{1}{2} \nabla_\theta \left((X\theta)^T X\theta - (X\theta)^T \vec{y} - \vec{y}^T (X\theta) + \vec{y}^T \vec{y} \right) \\
a^T b = b^T a \quad &= \frac{1}{2} \nabla_\theta \left(\theta^T (X^T X)\theta - \vec{y}^T (X\theta) - \vec{y}^T (X\theta) \right) \\
&= \frac{1}{2} \nabla_\theta \left(\theta^T (X^T X)\theta - 2(X^T \vec{y})^T \theta \right) \\
\nabla_x b^T x = b \quad & \\
\nabla_x x^T A x = 2Ax \quad &= \frac{1}{2} \left(2X^T X\theta - 2X^T \vec{y} \right) \\
&= X^T X\theta - X^T \vec{y}
\end{aligned}$$

با صفر قرار دادن مشتق داریم:

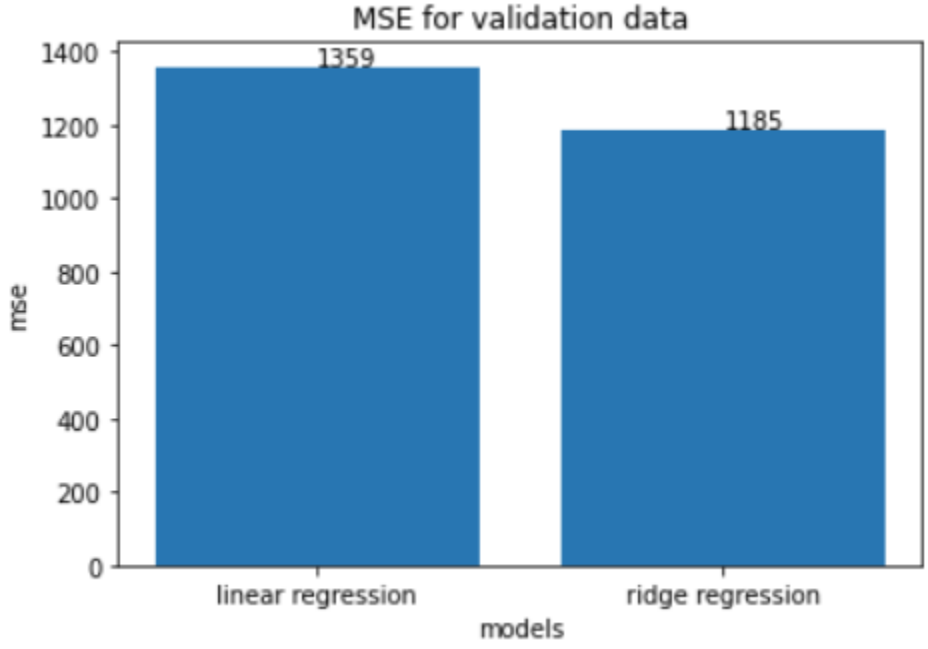
$$X^T X\theta = X^T \vec{y}$$

که در نتیجه تتا مطابق با زیر به دست می آید.

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

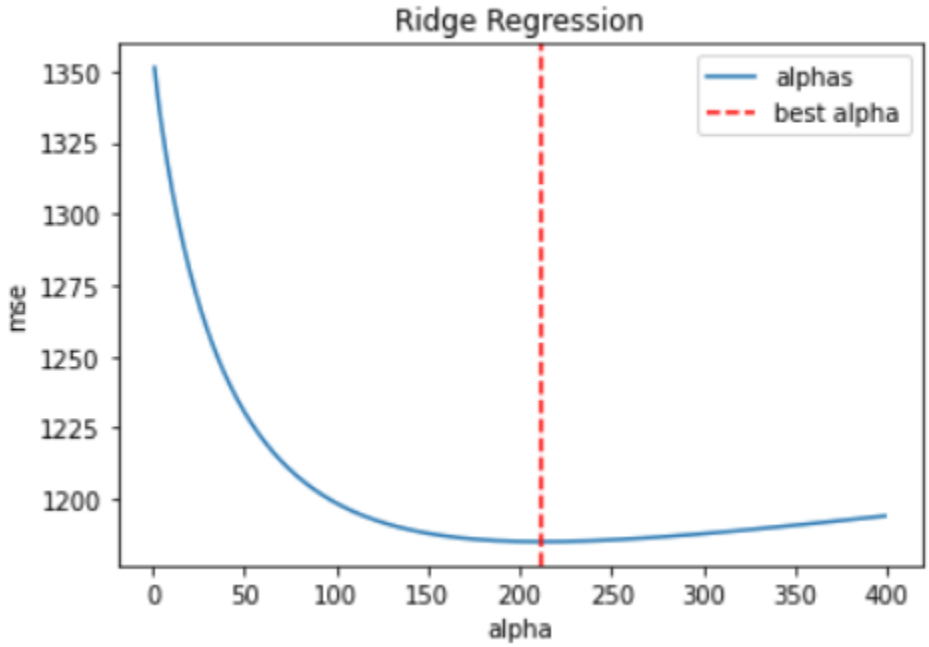
Ridge Regression به ترم MSE برای بهینه سازی ترم regularization برای جلوگیری از بزرگ شدن وزن های تتا اضافه میکند.

مدل Ridge نیز بر روی داده ها آموزش داده شده است و نتیجه MSE بر روی داده های validation در مقایسه با linear regression در زیر آورده شده است که نشان از بهبود نتایج دارد.



برای انتخاب بهترین مدل پارامتر سراسری ridge regression از ۱ تا ۴۰۰ تغییر داده شده است و mse آن بر روی داده های validation محاسبه شده است.

همان طور که در زیر مشاهده میشود alpha=212 بهترین mse را داشته است و از آن به بعد مدل دچار overfit و افزایش loss شده است.



قسمت ۲)

مطابق صورت سوال یک شبکه بدون لایه مخفی پیاده سازی شده است. خروجی مدل پایتورچ مطابق با زیر است.

```
MLP(
  (fc1): Linear(in_features=68, out_features=2, bias=True)
)
```

هم چنین در آزمایشات batch_size=32 و learning_rate=0.01 در نظر گرفته شده است.

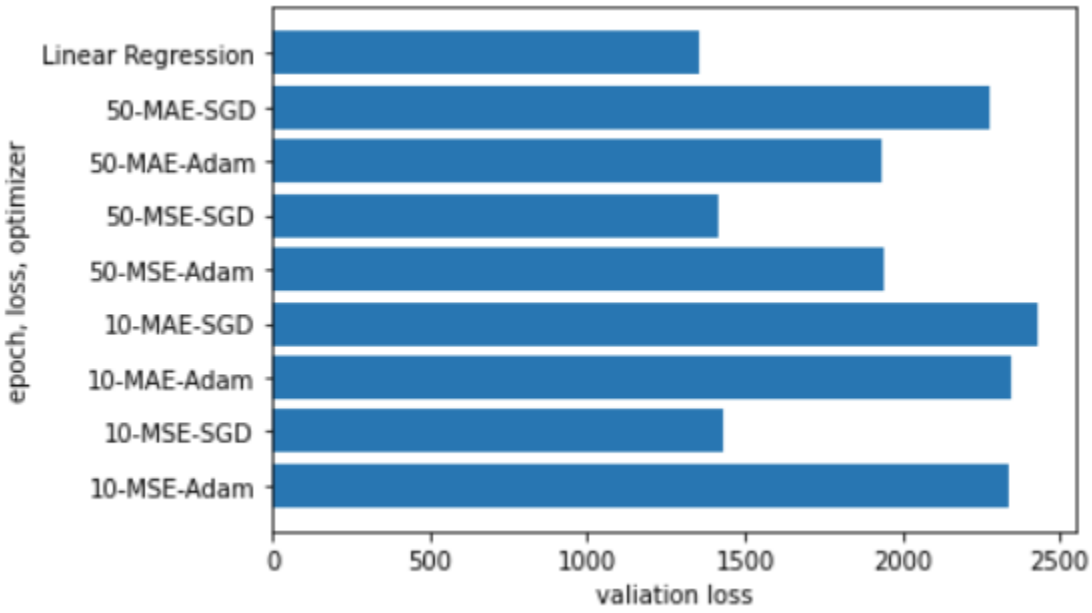
حالت های مختلف تابع هزینه MSE, MAE و بهینه ساز Adam, SGD با epochهای ۱۰ و ۵۰ در نظر گرفته شده است.

نمودار خطای MSE بر روی داده های ارزیابی برای حالت های مختلف در زیر آورده شده است و هم چنین با linear regression مرحله قبل مقایسه شده است.

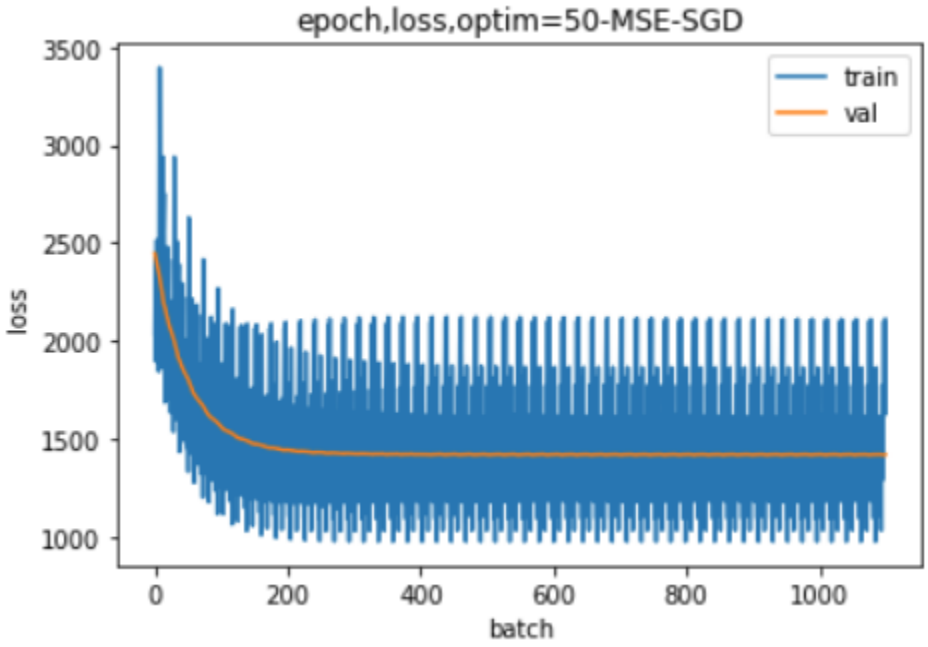
همان طور که مشاهده میشود بهترین مدل شامل پارامترهای زیر است. که برای سوالات بعدی در نظر گرفته شده است.

optimizer: **SGD**
loss: **MSE**
epoch: **50**

هم چنین این مدل توانسته است نسبت به بقیه متغیرها تا حد نزدیکی به performance مدل linear regression که فرم بسته دارد برسد. دلیل آن نیز میتواند مشکلات بهینه سازی و hyper parameterهای مختلف مدل شبکه عصبی باشد.



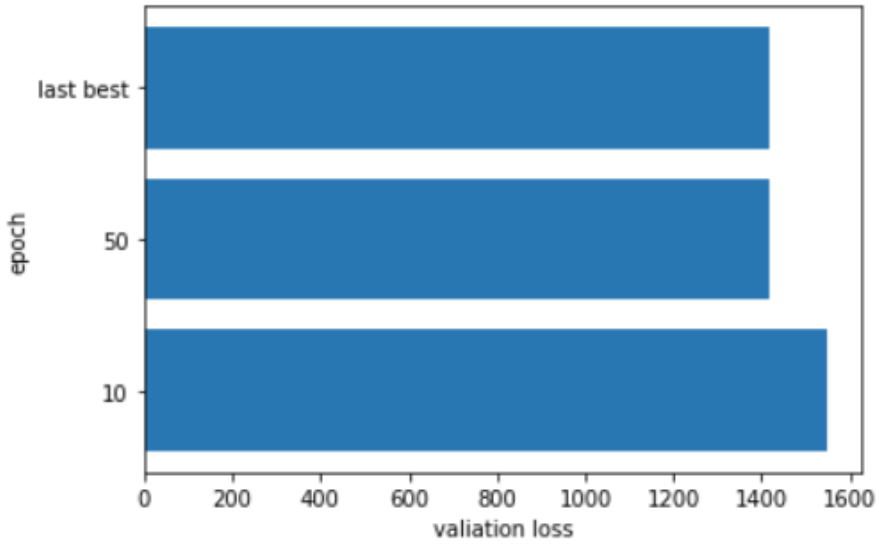
هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای بهترین حالت در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.



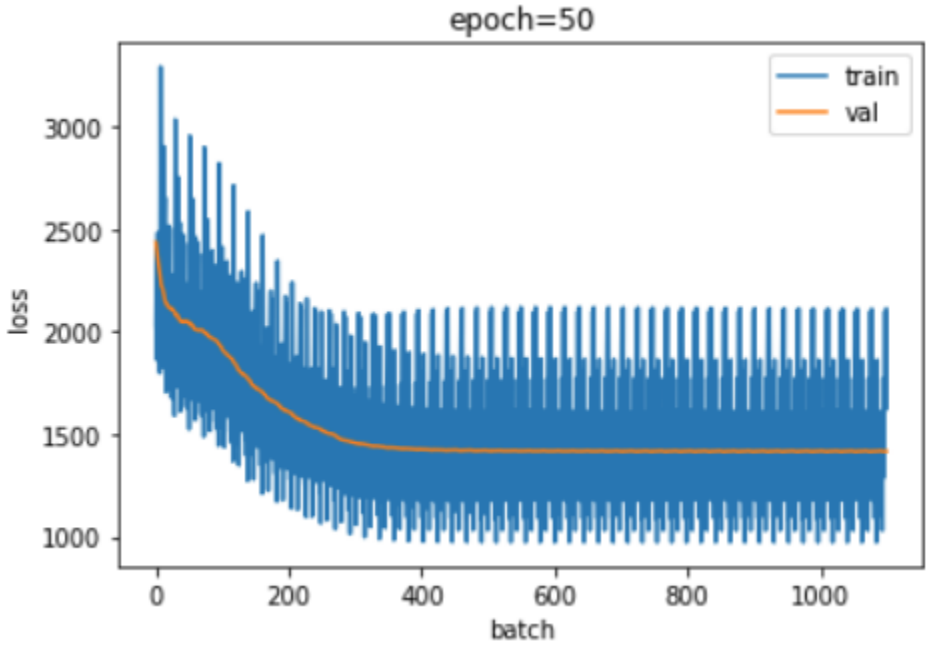
قسمت ۳)

با در نظر گرفتن بهترین پارامترهای مدل قبل یعنی Adam و MSE تابع فعال ساز غیر خطی relu را به مدل اضافه میکنیم.

همان طور که مشاهده میشود تغییر چندانی نسبت به حالت قبل نکرده است. هم چنین epoch=10 برای آموزش کم بوده و باید epoch=50 در نظر گرفته شود



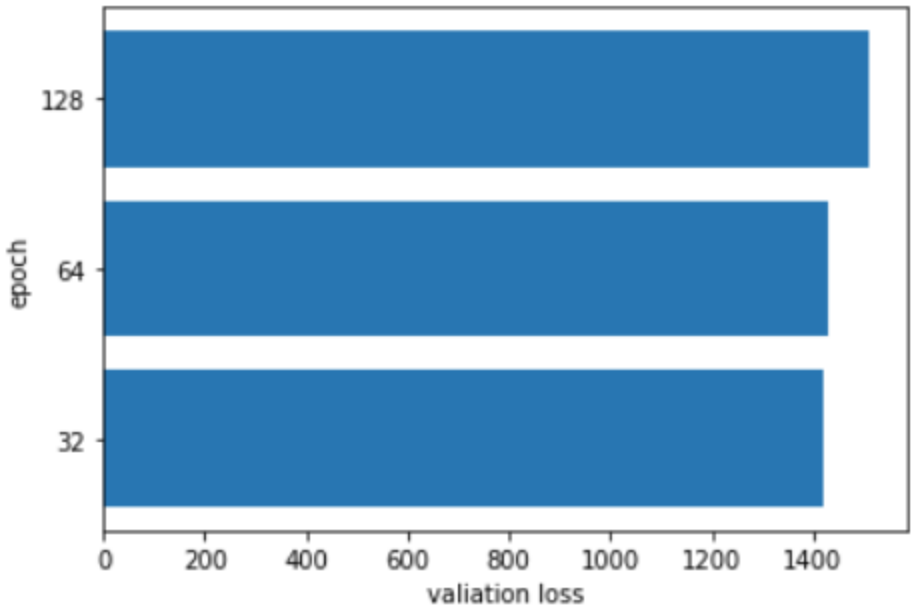
هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای بهترین حالت در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.



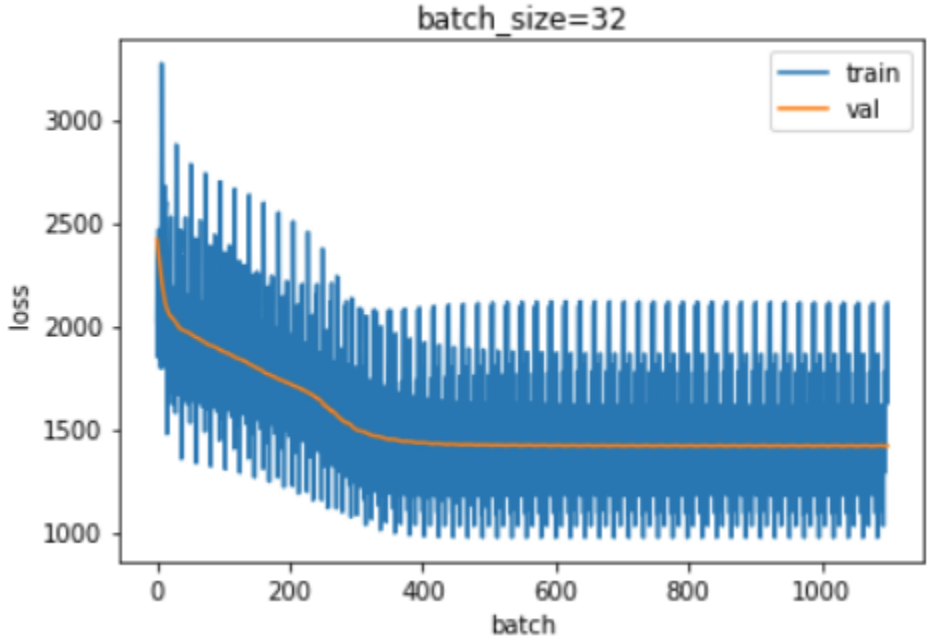
تابع فعال ساز relu نسبت به دیگر توابع فعال ساز مقادیر کمتر از صفر را برابر با صفر میکند که مزیت هایی از جمله جلوگیری از vanishing gradient دارند. هم چنین به خاطر ساده تر بودن آن نسبت به غیر خطی بودن دیگر توابع فعال ساز مدل سازی general تری برای این داده ایجاد کرده است.

قسمت ۴)

برای ارزیابی دسته داده حالت های مختلف ۳۲ و ۶۴ و ۱۲۸ در نظر گرفته شده است.
نمودار خطا بر روی داده های ارزیابی برای حالت های مختلف batch در زیر آورده شده است.
همان طور که مشاهده میشود تعداد batch=256 منجر به کم شدن دقت شده است اما بهترین تعداد batch=32 است که از این به بعد در نظر گرفته شده است.
افزایش تعداد batch زمان آموزش را کاهش میدهد اما منجر به کاهش generalization میشود که نتایج نیز همین موضوع را تصدیق میکند. البته batch بیشتر بهینه سازی را تغییر میدهد که ممکن است تعداد epoch بیشتری نیاز باشد.



هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای بهترین حالت در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.



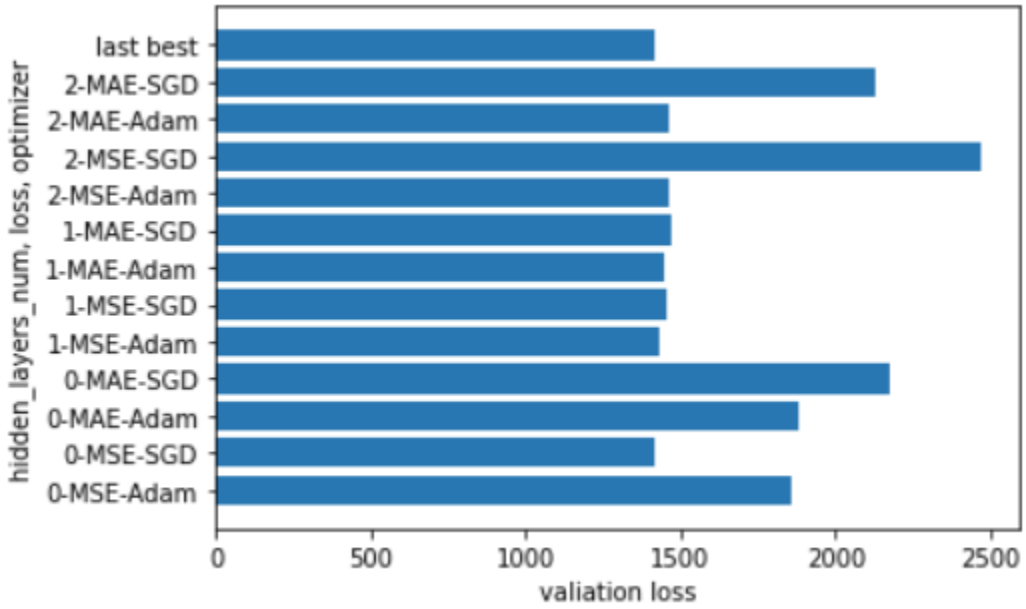
قسمت ۵)

تغییرات مدل با اضافه کردن ۱ و ۲ لایه مخفی و حالت های تابع هزینه MSE, MAE و بهینه ساز Adam, SGD در نظر گرفته شده است.

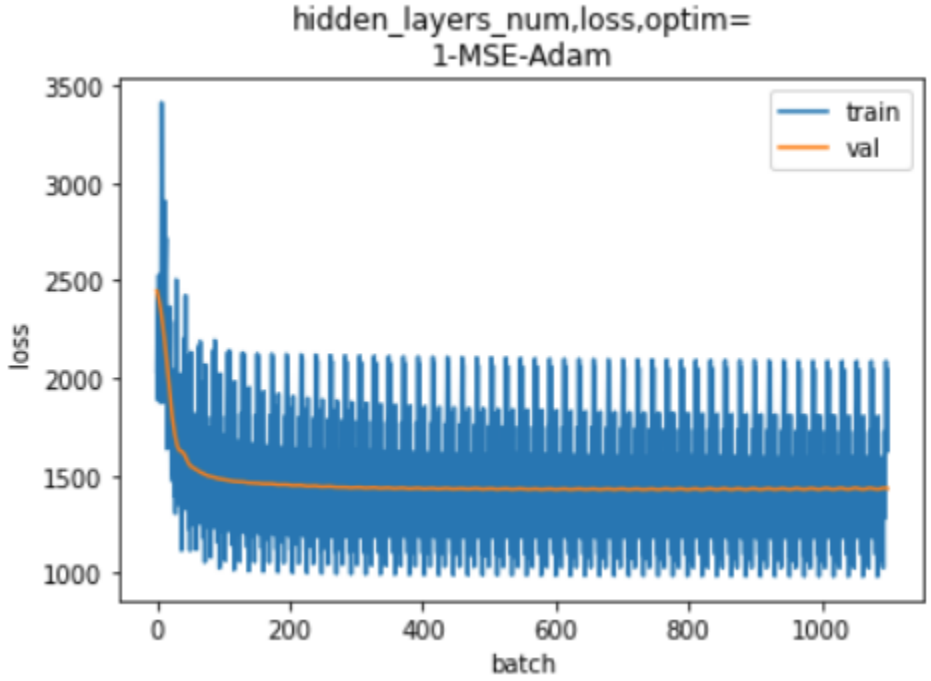
نمودار خطا بر روی داده های ارزیابی برای حالت های مختلف در زیر آورده شده است.
همان طور که مشاهده میشود افزایش یک لایه و دو لایه تفاوت چندانی ندارد.
اما نکته ای که وجود دارد این است که بهترین مدل بر اساس داده های validation بدون لایه مخفی است با در نظر گرفتن این که تابع loss آن MSE و بهینه ساز آن SGD باشد. اما اگر غیر از این باشد اضافه کردن لایه منجر به بهبود مدل میشود.

با توجه به نزدیک بودن حالت های مختلف و برای این که مدل به بهینه ساز و تابع هزینه وابستگی چندانی نداشته باشد **بهترین مدل با پارامترهای تعداد لایه های مخفی ۱ و بهینه ساز نیز Adam به علت رفتار ثابت و تابع هزینه MSE به علت بهترین نتیجه در نظر گرفته میشود.**

hidden_layers: 2
loss: MSE
optimizer: Adam

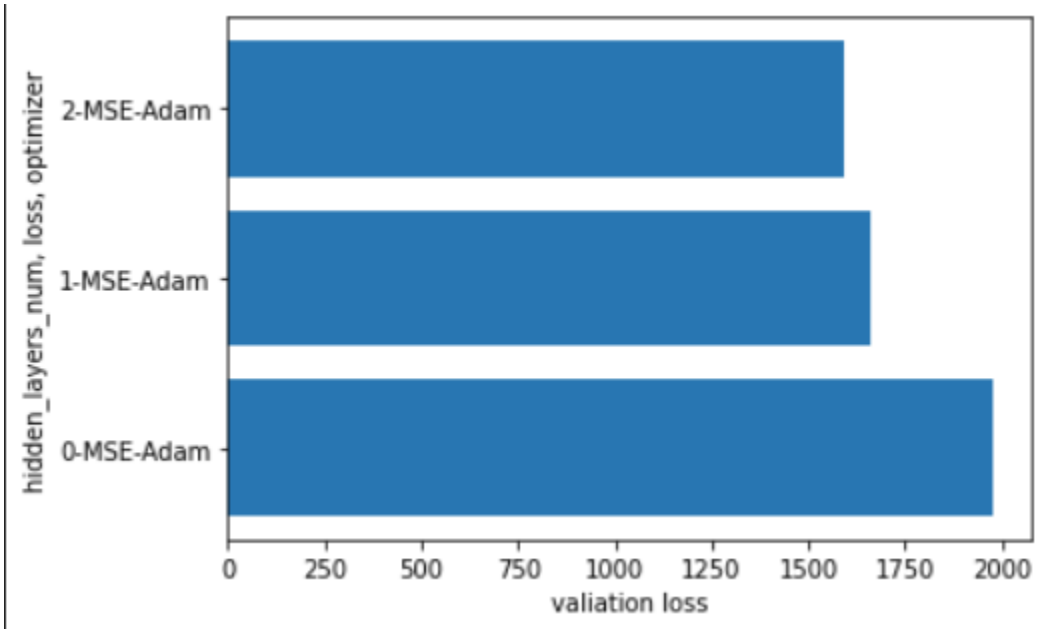


هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای حالت نتیجه گرفته شده در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.



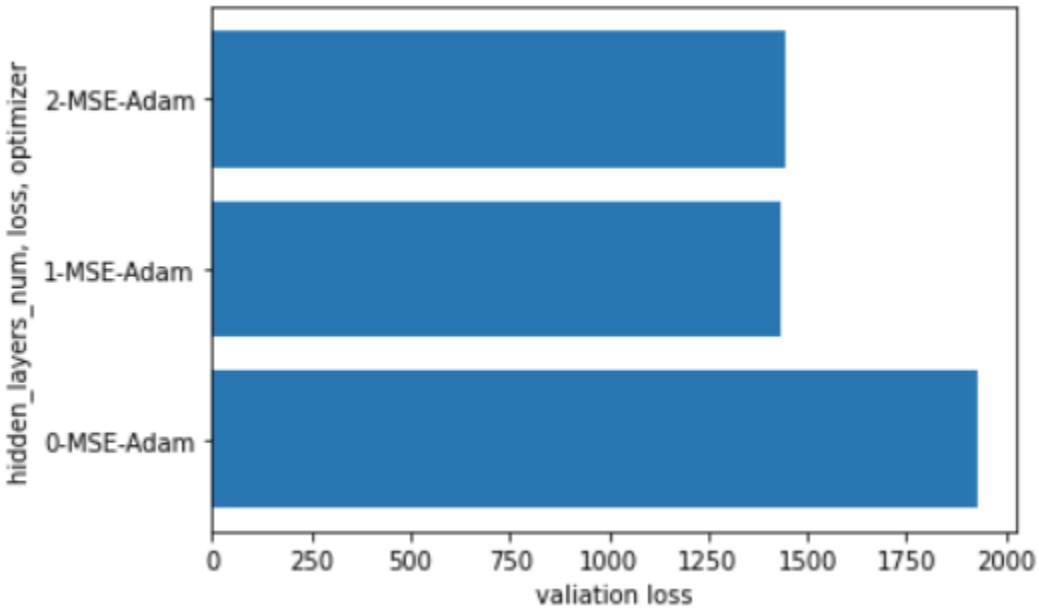
قسمت ۶)

به مدل برنده با تابع هزینه MSE و بهینه ساز Adam در حالت 0, 1, 2 لایه مخفی dropout=0.2 اضافه کرده ایم.

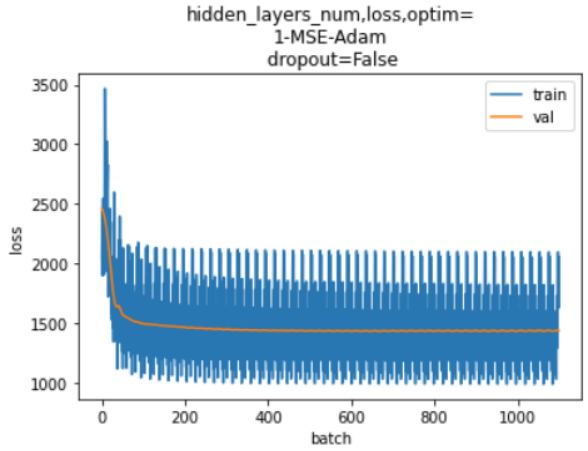


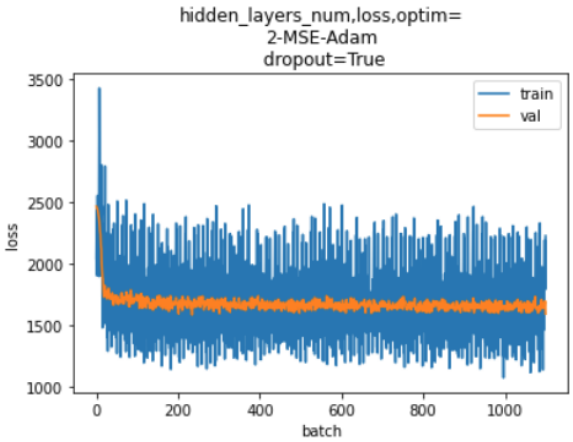
همان طور که مشاهده میشود میزان خطا بر روی داده validation با dropout که در شکل بالا آورده شده است و بدون dropout که در شکل زیر آورده شده است کاهش یافته است.بنابراین اضافه کردن dropout منجر به بدتر شدن عملکرد مدل میشود.

گرچه هم چنان مدل با یک لایه مخفی بهترین مدل است.



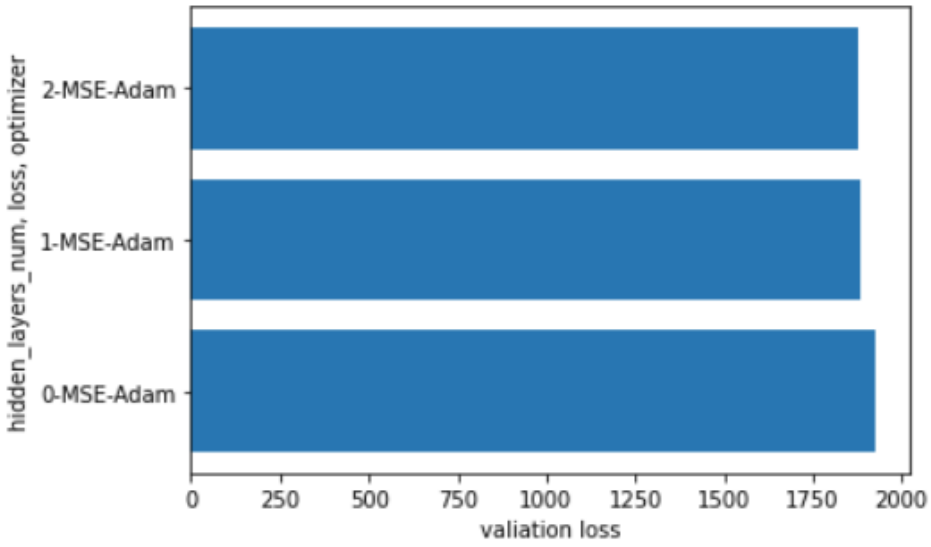
هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای دو حالت dropout=True, droput=False شده در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.





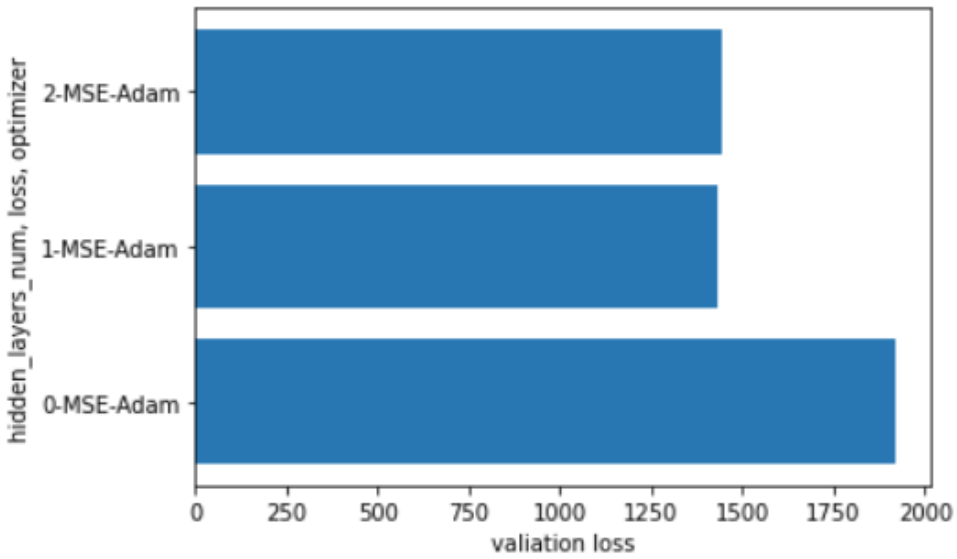
قسمت ۷)

به مدل برنده با تابع هزینه MSE و بهینه ساز Adam در حالت 0, 1, 2 لایه مخفی batch normalization اضافه کرده ایم.



همان طور که مشاهده میشود میزان خطا بر روی داده validation با batch normalization که در شکل بالا آورده شده است و بدون batch normalization که در شکل زیر آورده شده است کاهش یافته است.بنابراین اضافه کردن batch normalization منجر به بدتر شدن عملکرد مدل میشود.

گرچه هم چنان مدل با یک لایه مخفی بهترین مدل است.



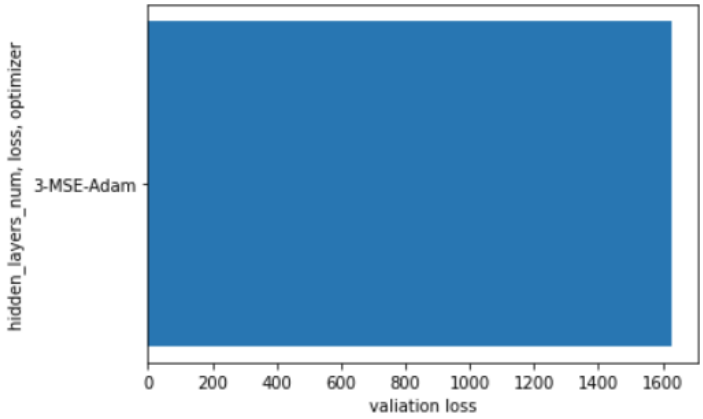
قسمت ۸)

با توجه به قسمت های قبل پارامترهای زیر به عنوان بهترین پارامترها انتخاب میشود.

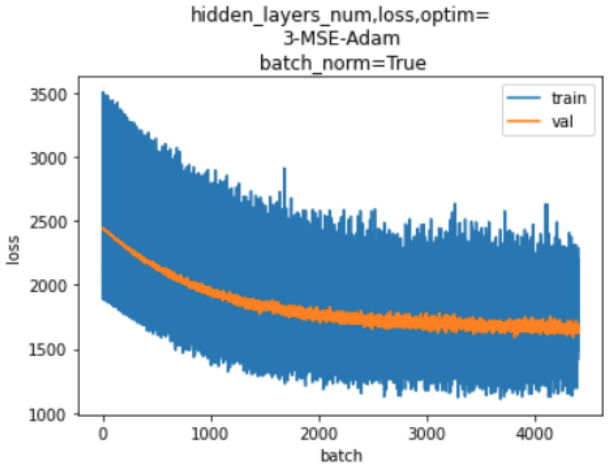
activation function: **relu**
loss: **MSE**
optimizer: **Adam**
number of hidden layers: **1**

epoch: 50
learning_rate: 0.01

نمودار خطا بر روی داده های ارزیابی برای حالت لایه نهان در زیر آورده شده است که نشان از افزایش خطا نسبت به بهترین حالت گفته شده در بالا است.



هم چنین نمودار خطا بر روی داده های ارزیابی و آموزش برای تعداد لایه مخفی ۳ در زیر آورده شده است که نشان از عدم overfit مدل بر روی داده ها دارد.
هم چنین برای converge کردن نیز نیاز به epochهای بیشتری است.



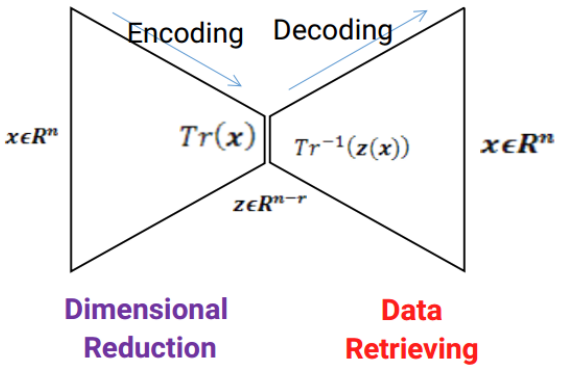
سوال ۳- آشنایی با کاهش بعد

- **نکات مربوط به پیاده سازی در ابتدای گزارش آورده شده است.**

۱) کاهش بعد برای مسئله اول: classification

قسمت اول: Auto Encoder

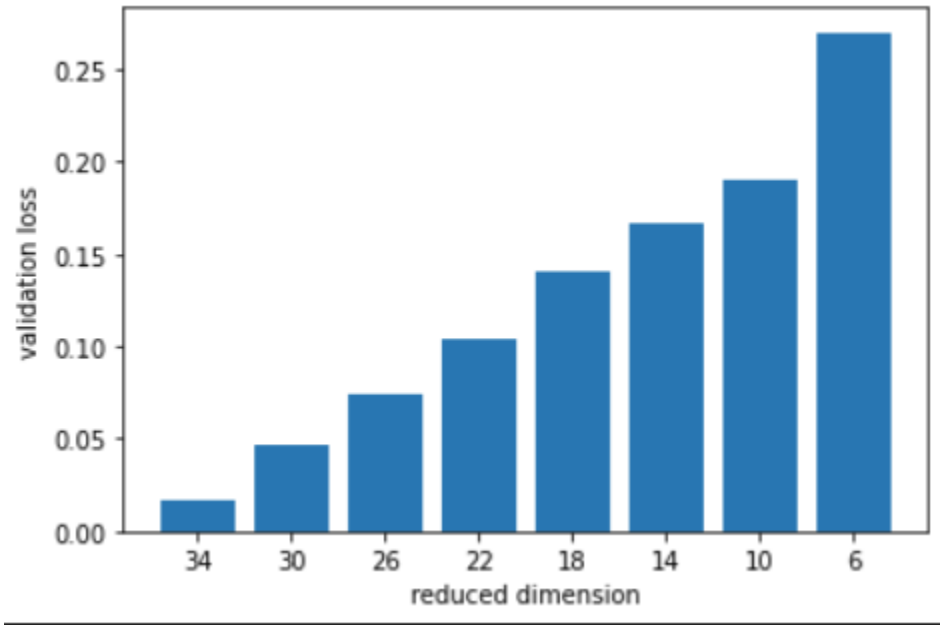
Autoencoderای مطابق شکل زیر پیاده سازی شده است.
که هر لایه ۴ بعد کمتر از لایه قبلی است. تابع فعال سازی نیز بین لایه ها استفاده نشده است.



برای تشخیص این که تا چه حد کاهش بعد را انجام دهیم لایه های نهان encoder به ازای حالت های زیر تست شده اند. بعد اولیه داده نیز ۳۴ است. لایه های decoder نیز عکس هر حالت هستند.

```
[34]
[34, 30]
[34, 30, 26]
[34, 30, 26, 22]
[34, 30, 26, 22, 18]
[34, 30, 26, 22, 18, 14]
[34, 30, 26, 22, 18, 14, 10]
[34, 30, 26, 22, 18, 14, 10, 6]
```

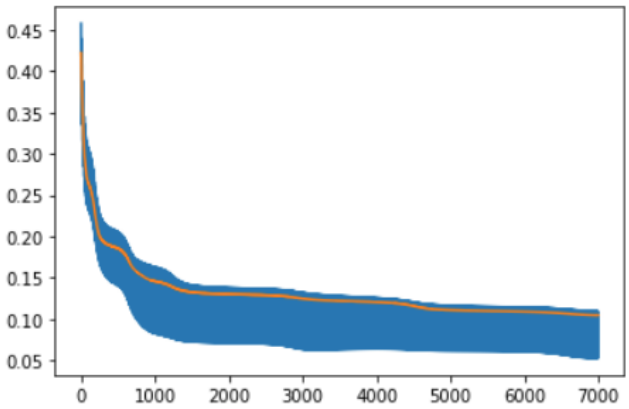
هم چنین نمودار خطای MSE بر روی داده validation که بیانگر قدرت شبکه در reconstruct کردن داده هاست به ازای حالت های مختلف encoder , decoder که در بالا آورده شده اند در زیر رسم شده است.



همان طور که مشاهده میشود با کاهش بعد بیشتر توانایی مدل در reconstruct کردن داده ها کم میشود و خطای مدل افزایش می یابد.

طبق نمودار بالا **کاهش بعد تا ۲۲** در نظر گرفته شد که خطا نزدیک 0.1 و کمتر از آن است.

هم چنین نمودار خطای داده اموزش و ارزیابی برای این حالت در زیر آورده شده است.

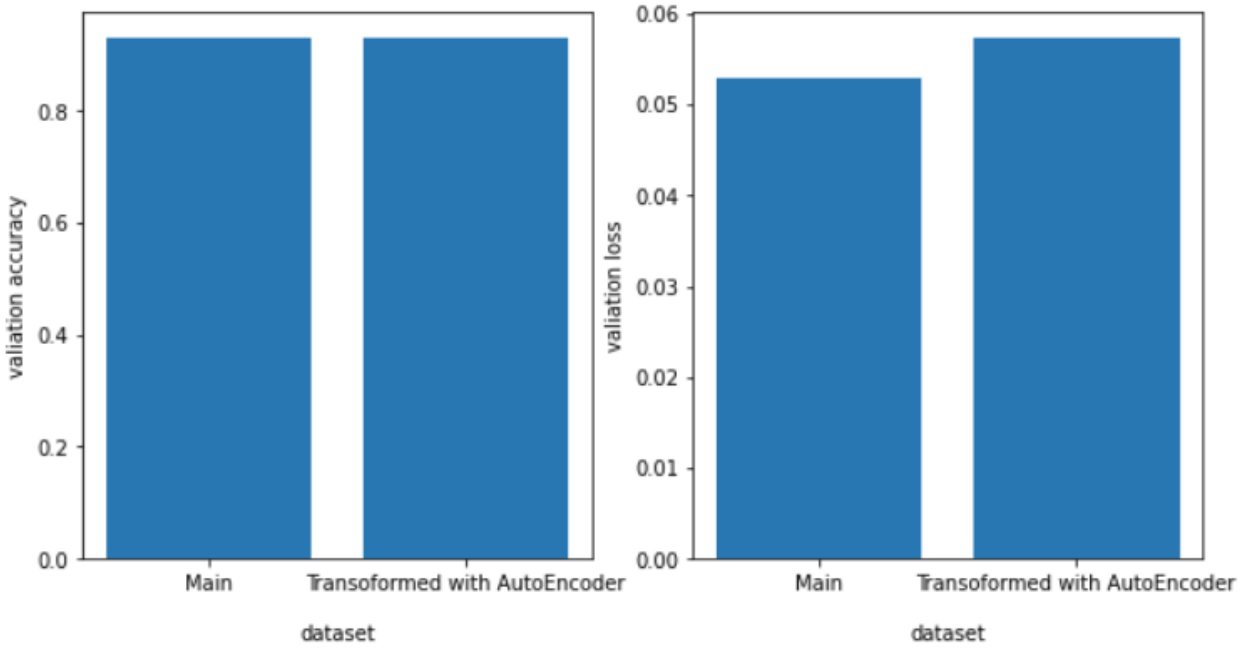


داده های با استفاده از encoder به بعد ۲۲ کاهش یافتند تا نتیجه آن در تسک classification سوال اول ارزیابی شود.

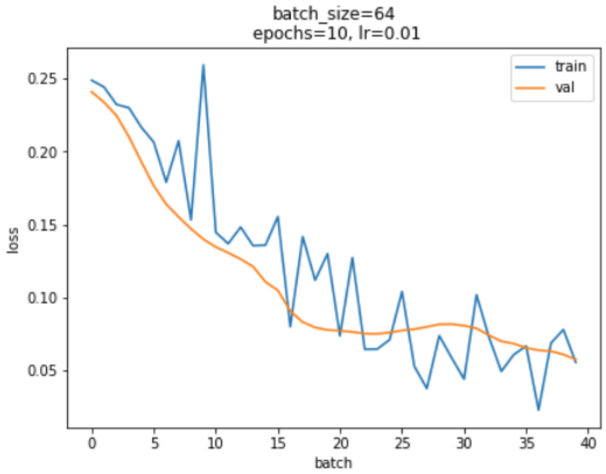
نمودار زیر خطا و دقت مدل را برای دو حالت کاهش بعد یافته با استفاده از autoencoder و بدون کاهش بعد نشان میدهد.

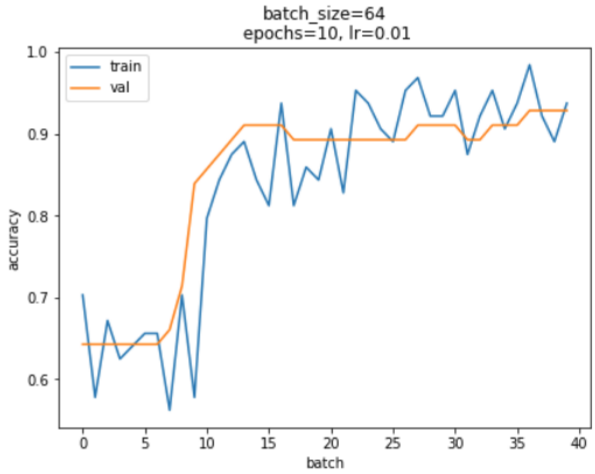
همان طور که مشاهده میشود کاهش بعد منجر به افزایش loss شده است.

علت عدم بهبود میتواند مدل نوع مدل autoencoder استفاده شده باشد. با توجه به نتایج مرحله قبل مدل برای حالت کاهش بعد یافته 22 نتوانسته است به خطای reconstruction کمی برسد و تنها در حالات با ابعاد بالا خطا کم بوده است. مشکل میتواند از نحوه بهینه سازی یا معماری شبکه باشد.
در نهایت افزایش بعد 22 و نزدیک کردن آن به حالت بعد اصلی مثلا ۳۴ که خطای autoencoder نزدیک به صفر میشود کمکی به ما نمیکند چون هدف بررسی کاهش بعد در این مسئله بود.
گر چه جلوتر بر روی داده های تست دقت مدل افزایش میابد(شاید به خاطر این باشد که اگر در نمودار زیر accuracy را به جای loss در نظر بگیریم مدل تغییر نکرده است و در کل بتوانیم بگوییم generalization مدل خوب بوده است و فقط روی داده های validation بعد عمل کرده است و اگر cross validation انجام دهیم بهبود مدل خود را نشان دهد.)

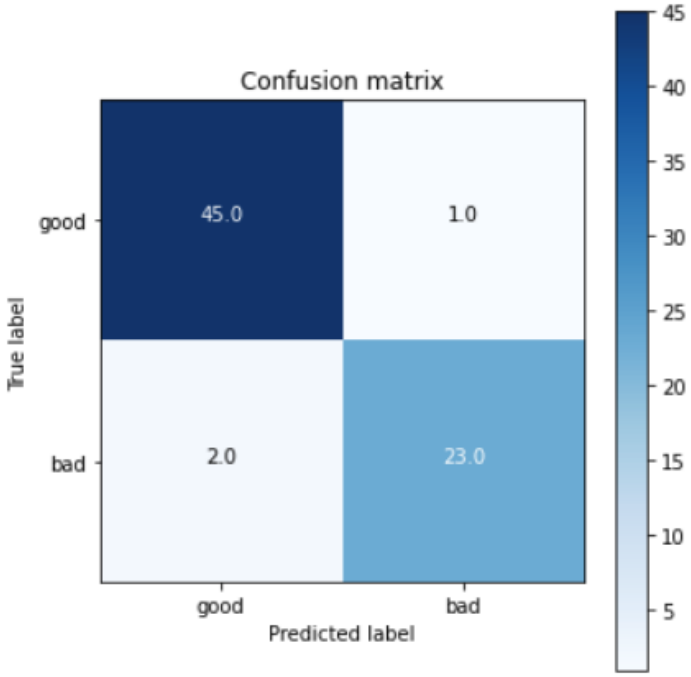


نمودار خطا و دقت بر روی داده های آموزش و ارزیابی در زیر آورده شده است که نشان از عدم overfit با داده های کاهش بعد یافته است.





هم چنین ماتریس confusion برای داده های تست کاهش بعد یافته در زیر آورده شده است که نشان از دقت بالای مدل دارد.



قسمت دوم: PCA

ماتریس کوریلشن چون یک ماتریس متقارن است پس میتوان eigenvalue decomposition بر روی آن مطابق با زیر انجام داد.

$$R = \sum_{i=1}^{m} \boldsymbol{x}^i \boldsymbol{x}^{i^T} = \mathrm{V}\Lambda\mathrm{V}^T$$

میتوان نشان داد که مقادیر ویژه به دست آمده از این تجزیه میتواند میزان variability داده ها را نشان دهد. در واقع اگر تعداد مقدار ویژه صفر مشابه به زیر وجود داشته باشند این مقادیر ویژه صفر پوچی های فضا را نشان میدهد.

$$\Lambda = \begin{bmatrix} \Lambda^1 & 0 \\ 0 & \Lambda^2 \end{bmatrix} \quad \Lambda^1 = diag([\lambda_1, .., \lambda_r]) = \mathbf{0} \quad \Lambda^2 = diag([\lambda_{r+1} \dots., \lambda_n])$$

بردار ویژه های غیر صفر متناظر با مقدار ویژه های غیر صفر مولفه های اصلی محسوب میشوند.

$$\mathrm{V} = \left[\underbrace{\boldsymbol{v}_1 .. \boldsymbol{v}_r}_{\mathrm{V}_1} \underbrace{\boldsymbol{v}_{r+1} \dots \boldsymbol{v}_n}_{\mathrm{V}_2} \right] = [\mathrm{V}_1\ \mathrm{V}_2]$$

و ضرب داخلی داده ها در آن میتواند بعد داده ها را کاهش دهد و پوچی های فضا را از بین ببرد.

$$\mathrm{V} = \left[\underbrace{\boldsymbol{v}_1 .. \boldsymbol{v}_r}_{\mathrm{V}_1} \underbrace{\boldsymbol{v}_{r+1} \dots \boldsymbol{v}_n}_{\mathrm{V}_2} \right] = [\mathrm{V}_1\ \mathrm{V}_2]$$

بنابراین transformation ای مطابق با زیر خواهیم داشت.

$$\mathrm{V} = \left[\underbrace{\boldsymbol{v}_1 .. \boldsymbol{v}_r}_{\mathrm{V}_1} \underbrace{\boldsymbol{v}_{r+1} \dots \boldsymbol{v}_n}_{\mathrm{V}_2} \right] = [\mathrm{V}_1\ \mathrm{V}_2]$$

برای پیاده سازی از eigenvalue decomposition استفاده شده است و چون ابعاد داده ها کم است میتوان از این روش به جای روش svd برای پیاده سازی pca استفاده کرد.

برای پیدا کردن بهترین کاهش بعد از مقادیر ویژه استفاده میکنیم. چون مقادیر ویژه نشان دهنده میزان variability داده ها هستند مقادیر ویژه به دست آمده را از بزرگ به کوچک مرتب میکنیم سپس بر جمع کل مقادیر ویژه تقسیم میکنیم.

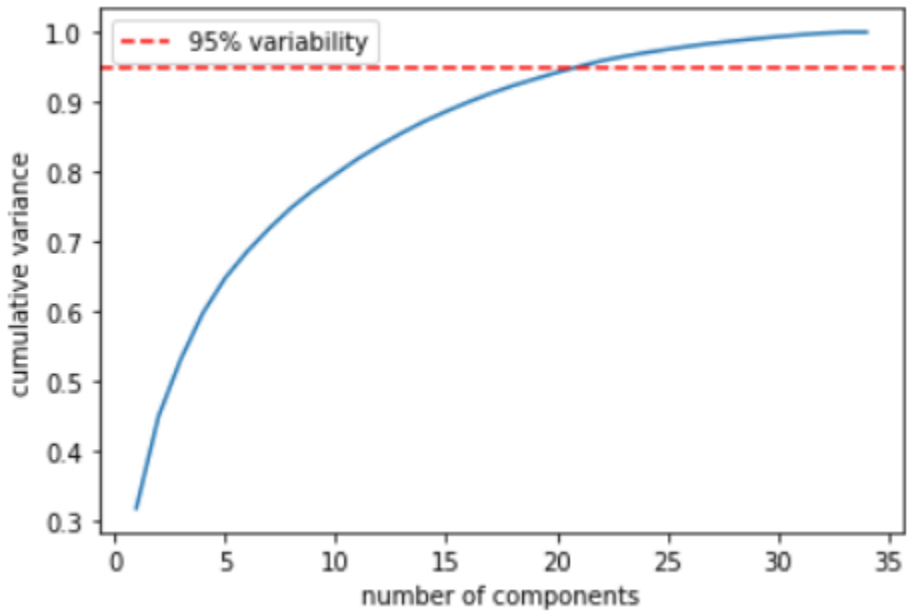
```
a = eigenvalues / self.eigenvalues.sum()
```

سپس جمع تجمعی آن ها را حساب میکنیم.

```
np.cumsum(a)
```

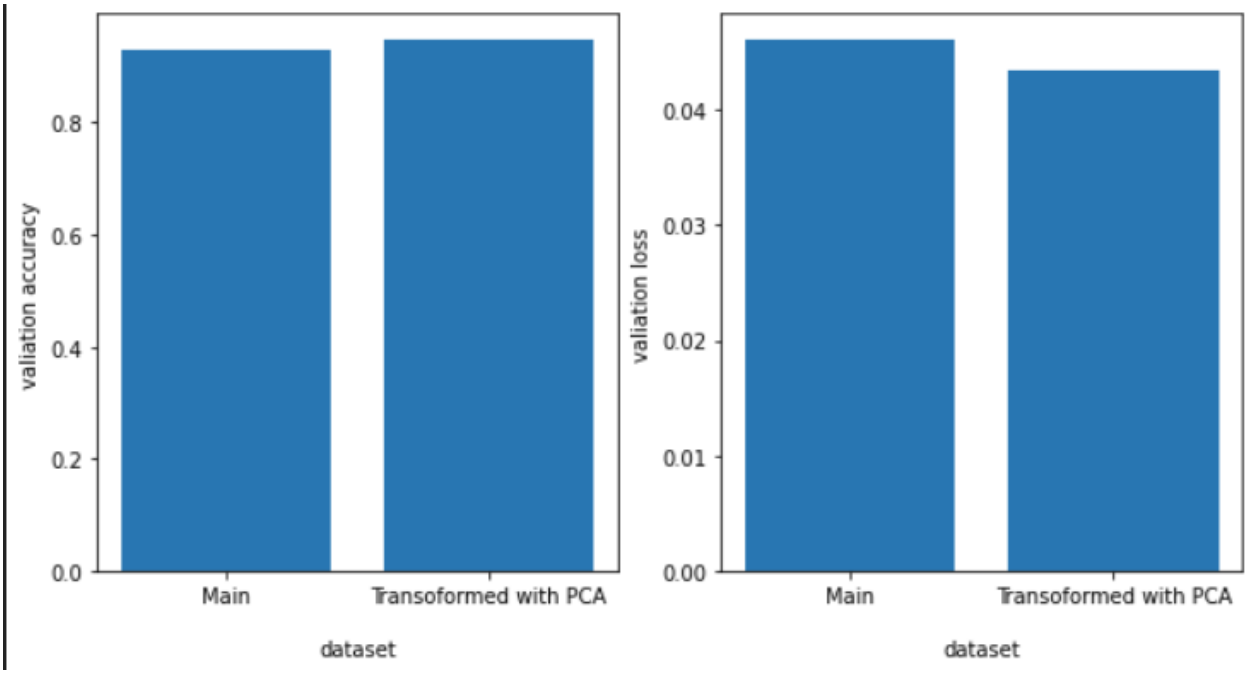
تا جایی که این مقدار به 0.95 برسد یعنی این مقادیر ویژه توانسته اند ۹۵ درصد variability در داده را توصیف کنند.

برای داده ها اگر این مقدار را محاسبه کنیم مشخص میشود که تا بعد ۲۰ بدون از دست رفتن اطلاعات مفید کاهش بعد داد.



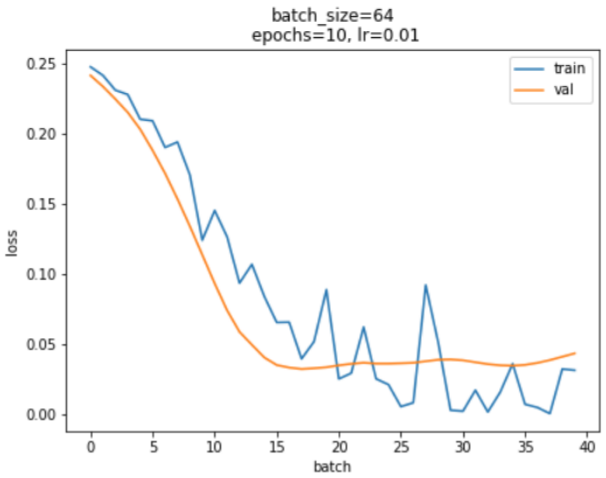
داده های با استفاده از encoder به بعد ۲۲ کاهش یافتند تا نتیجه آن در تسک classification سوال اول ارزیابی شود.

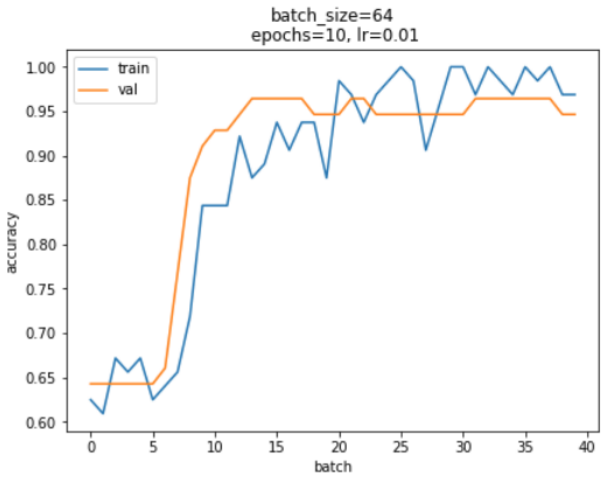
نمودار زیر خطا و دقت مدل را برای دو حالت کاهش بعد یافته با استفاده از pca و بدون کاهش بعد نشان میدهد.



همان طور که مشاهده میشود pca منجر به کاهش خطا و افزایش دقت شده است که میتواند به علت حذف نویز از داده ها و کمک به generalization مدل باشد.

نمودار خطا و دقت بر روی داده های آموزش و ارزیابی در زیر آورده شده است که نشان از عدم overfit با داده های کاهش بعد یافته است.





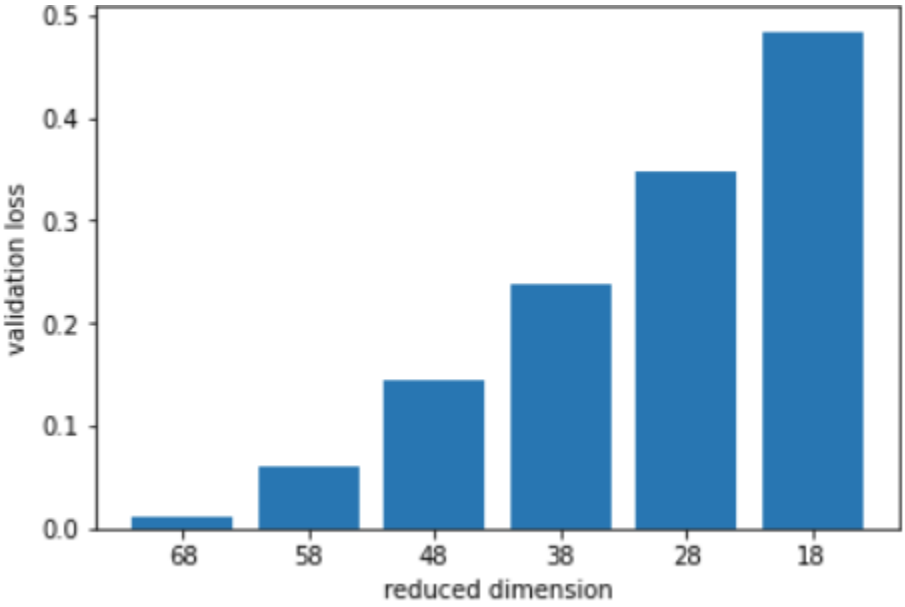
) کاهش بعد برای مسئله دوم: regression

قسمت اول: Auto Encoder

برای تشخیص این که تا چه حد کاهش بعد را انجام دهیم لایه های نهان encoder به ازای حالت های زیر تست شده اند. بعد اولیه داده نیز ۶۸ است. لایه های decoder نیز عکس هر حالت هستند.

```
[68]
[68, 58]
[68, 58, 48]
[68, 58, 48, 38]
[68, 58, 48, 38, 28]
[68, 58, 48, 38, 28, 18]
```

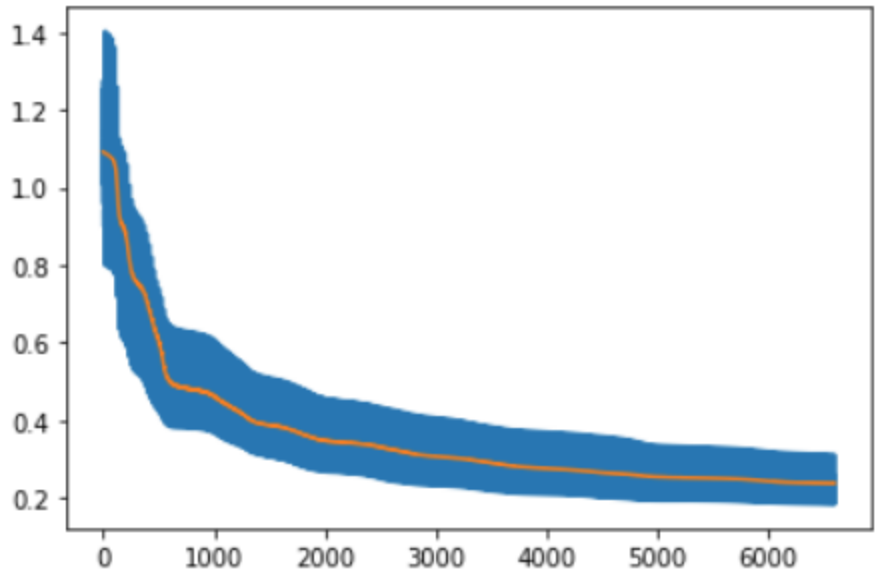
هم چنین نمودار خطای MSE بر روی داده validation که بیانگر قدرت شبکه در reconstruct کردن داده هاست به ازای حالت های مختلف encoder , decoder که در بالا آورده شده اند در زیر رسم شده است.



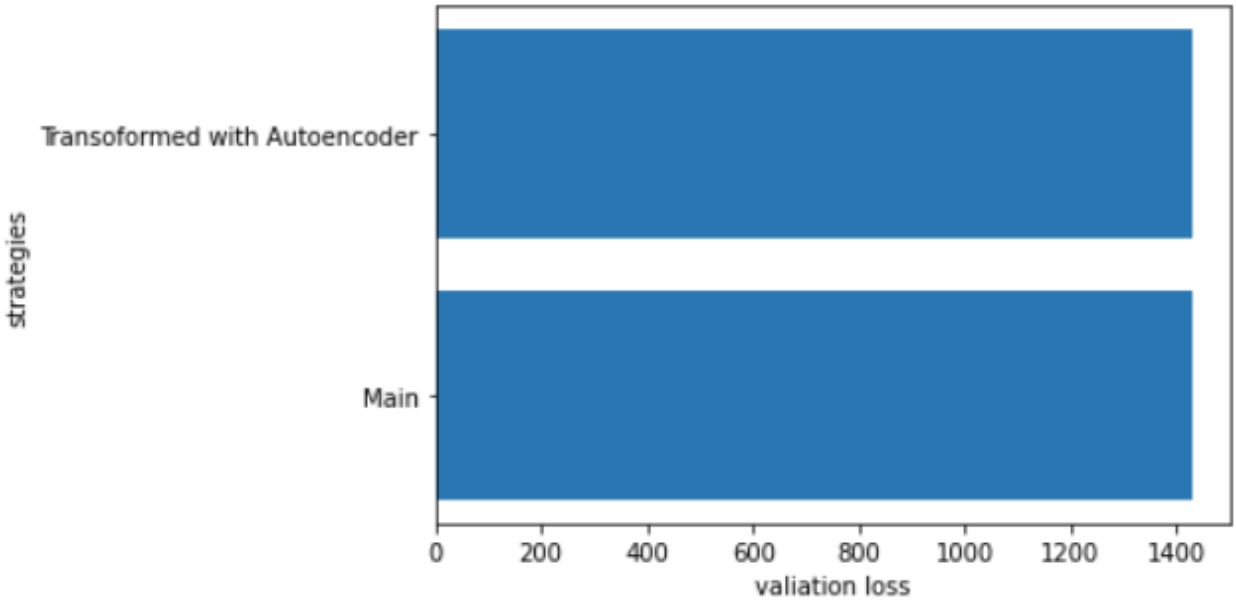
همان طور که مشاهده میشود با کاهش بعد بیشتر توانایی مدل در reconstruct کردن داده ها کم میشود و خطای مدل افزایش می یابد.

طبق نمودار بالا **کاهش بعد تا ۳۸** در نظر گرفته شد که خطا نزدیک 0.2 و کمتر از آن است.

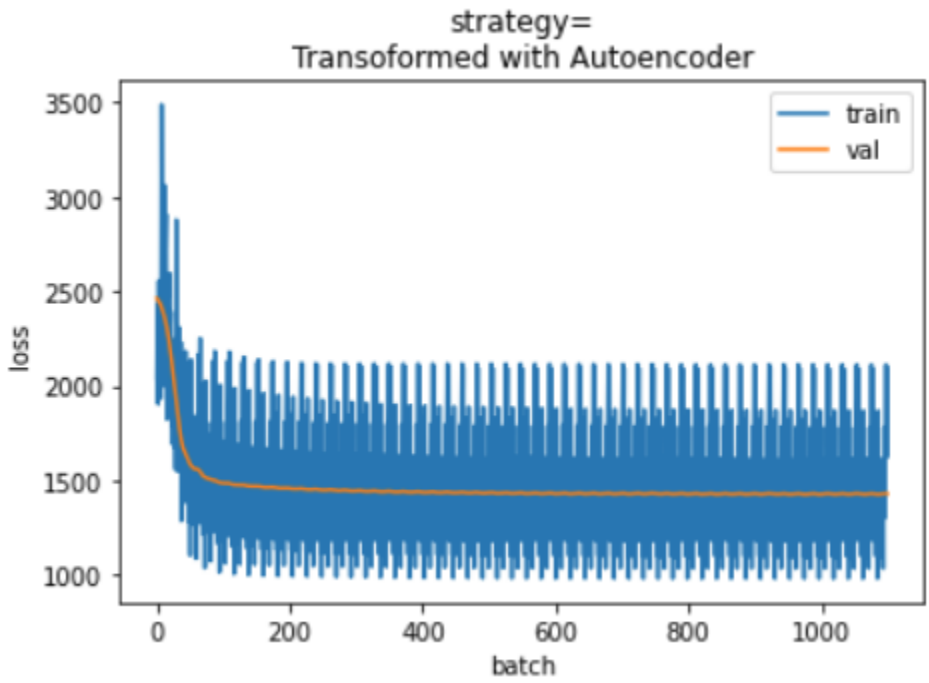
هم چنین نمودار خطای داده آموزش و ارزیابی برای این حالت در زیر آورده شده است.



داده های با استفاده از encoder به بعد 38 کاهش یافتند تا نتیجه آن در تسک regression سوال دوم ارزیابی شود.
نمودار زیر خطای مدل را برای دو حالت کاهش بعد یافته با استفاده از autoencoder و بدون کاهش بعد نشان میدهد.
همان طور که مشاهده میشود کاهش بعد منجر به تغییر loss نشده است.

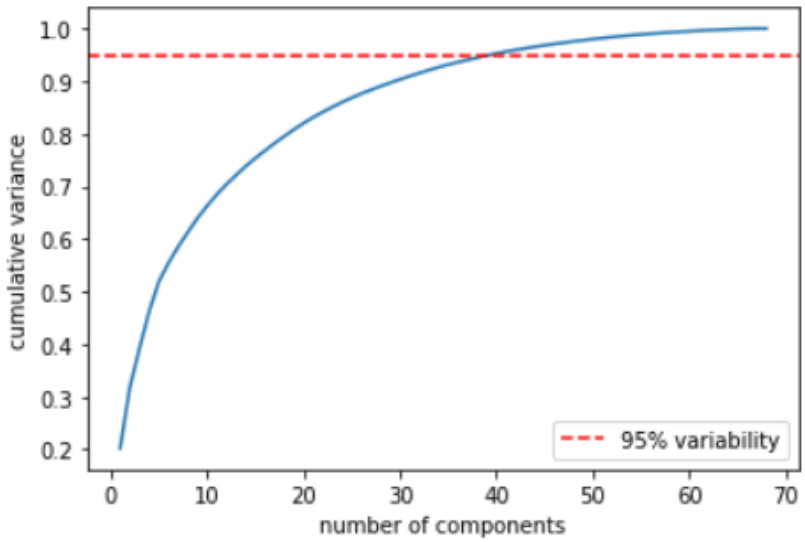


نمودار خطا بر روی داده های آموزش و ارزیابی در زیر آورده شده است که نشان از عدم overfit با داده های کاهش بعد یافته است.



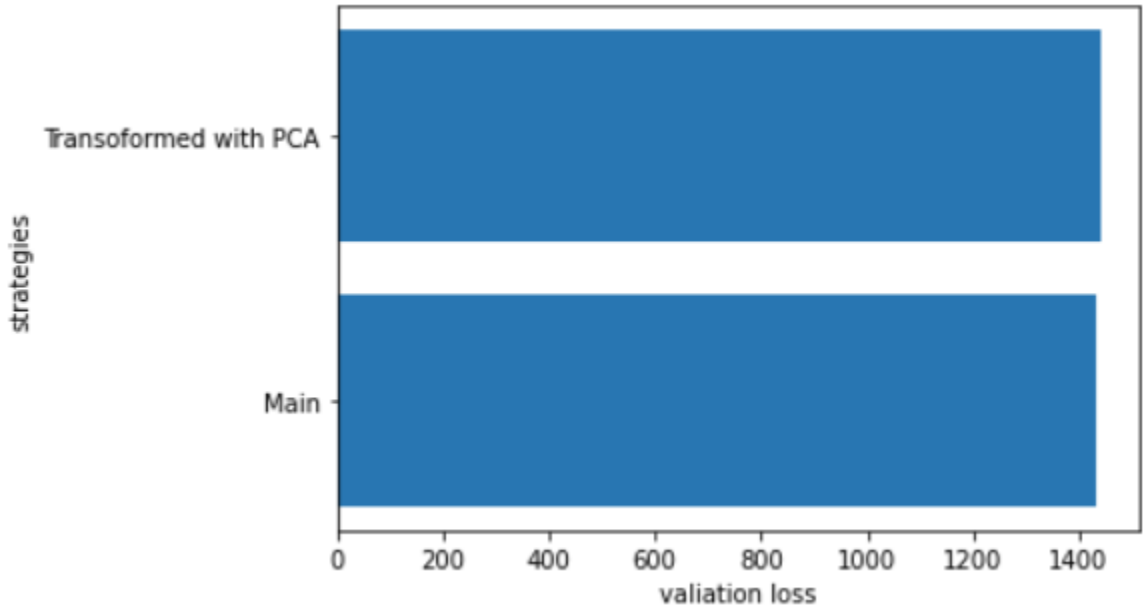
قسمت دوم: PCA

برای داده ها اگر این مقدار variability را با استفاده از مقادیر ویژه مطابق قبل محاسبه کنیم مشخص میشود که تا بعد ۳۹ بدون از دست رفتن اطلاعات مفید کاهش بعد داد.



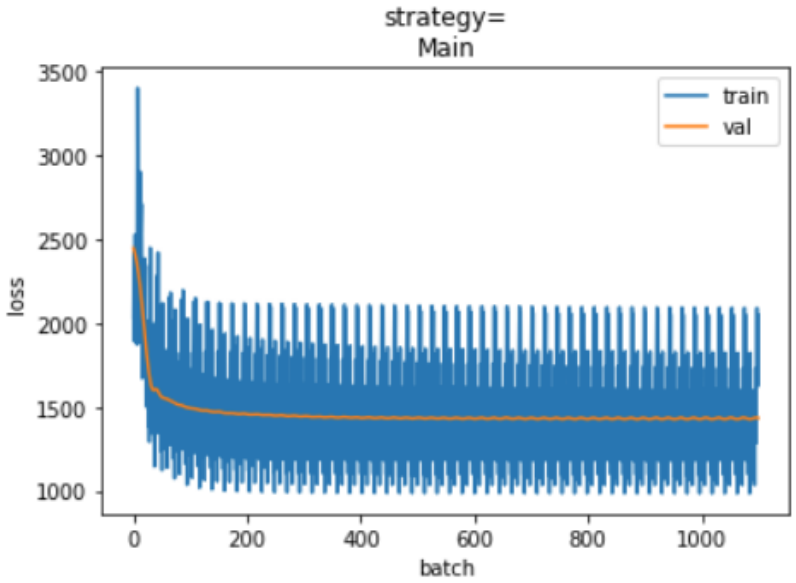
داده های با استفاده از encoder به بعد 38 کاهش یافتند تا نتیجه آن در تسک regression سوال دوم ارزیابی شود.

نمودار زیر خطا و دقت مدل را برای دو حالت کاهش بعد یافته با استفاده از pca و بدون کاهش بعد نشان میدهد.



همان طور که مشاهده میشود کاهش بعد منجر به تغییر loss نشده است.

نمودار خطا و دقت بر روی داده های آموزش و ارزیابی در زیر آورده شده است که نشان از عدم overfit با داده های کاهش بعد یافته است.



در نتیجه کاهش بعد در تسک regression برای داده های موجود تاثیری در بهبود و generalization مدل ندارد.

برای مسئله classification در این سوال پارامترهای مدل مطابق بهترین پارامترهای به دست آمده استفاده شد.

number of hidden neurons: **32**
batch_size: **64**
activation function: **relu**
loss: **MSE**
optimizer: **Adam**
number of hidden layers: **3**
epoch: 10
learning_rate: 0.01

برای مسئله regression نیز پارامترهای مدل مطابق بهترین پارامترهای به دست آمده استفاده شد.

activation function: **relu**
loss: **MSE**
optimizer: **Adam**
number of hidden layers: **1**

epoch: 50
learning_rate: 0.01