گذارش تمرین دوم درس شبکه‌های عصبی و یادگیری عمیق

محمد هاشمی 810197423

چکیده – این گزارش شامل بررسی پنج مساله به ترتیب *Multi-Layer Perceptron* و پیاده‌سازی شبکه‌ای برای تشخیص داده‌های *MNIST* از نوع *MLP* و استفاده از *Autoencoder* در آموزش *MLP* و روش *RBM* و *MLP* با یک تک‌لایه مخفی می‌باشد.

کلید واژه- Autoencoder,MLP,MNIST,RBM

1- مقدمه

شبکه‌های عصبی مصنوعی یا شبکه‌های عصبی صناعی (Artificial Neural Networks - ANN) یا به زبان ساده‌تر شبکه‌های عصبی سیستم‌ها و روش‌های محاسباتی نوین برای یادگیری ماشینی، نمایش دانش و در انتها اعمال دانش به دست آمده در جهت بیش‌بینی پاسخ‌های خروجی از سامانه‌های پیچیده هستند. ایده اصلی این گونه شبکه‌ها تا حدودی الهام‌گرفته از شیوه کارکرد سیستم عصبی زیستی برای پردازش داده‌ها و اطلاعات به منظور یادگیری و ایجاد دانش قرار دارد. عنصر کلیدی این ایده، ایجاد ساختارهایی جدید برای سامانه پردازش اطلاعات است.

این سیستم از شمار زیادی عناصر پردازشی فوق‌العاده بهم‌پیوسته با نام نورون تشکیل شده که برای حل یک مسئله با هم هماهنگ عمل می‌کنند و توسط سیناپس‌ها (ارتباطات الکترومغناطیسی) اطلاعات را منتقل می‌کنند. در این شبکه‌ها اگر یک سلول آسیب ببیند بقیه سلول‌ها می‌توانند نبود آن را جبران کرده، و نیز در بازسازی آن سهیم باشند. این شبکه‌ها قادر به یادگیری‌اند. مثلاً با اعمال سوزش به سلول‌های عصبی لامسه، سلول‌ها یادمی‌گیرند که به طرف جسم داغ نروند و با این الگوریتم سیستم می‌آموزد که خطای خود را اصلاح کند. یادگیری در این سیستم‌ها به صورت تطبیقی صورت می‌گیرد، یعنی با استفاده از مثال‌ها وزن سیناپس‌ها به گونه‌ای تغییر می‌کند که در صورت دادن ورودی‌های جدید، سیستم پاسخ درستی تولید کند.

2- پاسخ به سوال‌های در محوریت MLP

در این‌جا به چهار سوال مورد نظر در تمرین پاسخ می‌دهیم:

سوال اول: همانطوری که میدانید شبکه‌های عصبی میتوانند تنها با 2 لایه مخفی خاصیت General Function approximator بودن خود را حفظ کنند. با این وجود، دلیل این که گاه بیش از 2 لایه برای این شبکه‌ها انتخاب میکنیم چیست؟

در پاسخ به این سوال می‌گوییم، تصمیم‌گیری و تشخیص کلاس‌ها می‌تواند بر اساس مشخصات مختلف صورت‌گیرد. در مسائلی که تعداد مشخصات محدود باشد این عمل حداکثر با دو لایه مخی قابل تشخیص است. اما در برخی مسائل مانند تشخیص دیابتی بودن یا نبودن یک شخص بر اساس 13 فاکتور دیگر نمی‌توان گفت که دو لایه مخفی کافست یا اصلا جواب درست و قابل اطمینانی می‌دهد. از طرفی برای مسائلی که سایز ورودی‌ها زیاد باشد می‌بایست در ابتدا از چند لایه مختلف عبور کنند. یکی از علت‌های مهمی که امروزیه از تعداد لایه‌های مخفی بیش از دو لایه (یادگیری‌های عمیق) استفاده می‌شود این است که می‌توانیم به سیستم آموزش دهیم که حدس یزند. برای مثال شبکه‌هایی که بیش از دو لایه عمیق دارند می‌توانند پس از یادگیری با داشتن تکه از یک تصویر یا اطلاعاتی ناقص از آن کلاس، تا حد خیلی خوبی کلاس را حدس بزنند. این عمل به شدت در ذهن انسان تکرار می‌شود مثال ماشینی که فقط صندوق عقب آن پیداست را می‌تواند به کلاس ماشین نسبت دهد. از طرفی بر اساس تعریف موجود در فایل روی سایت تعداد لایه‌های بیشتر سه برتری دارند:

الف) دقت بیشتر

ب) validation بهتر

ج) تعداد داده مورد نیاز برای یادگیری کمتر

نکته بسیار مهم در رابطه با این شبکه‌ها برتری آن‌ها در دقت تخمین نسبت به لایه‌های کمتر است.

سوال دوم: تابع هزینه cross entropy چگونه هزینه را محاسبه می‌کند؟

در جوا به این سوال می‌گوییم، فرم دقیق این تابع مانند رابطه 2.1 می‌باشد. پس از بررسی بهتر مشاهده می‌کنیم که ترم‌های آخر برای تعداد دفعات ضرب به دلیل کمتر از یک بودن می‌توانند چشم پوشی شوند و فرمول ساده تر در رابطه 2.2 موجود است.

$$H_{y'}(y) := -\sum_i (y'_i \log(y_i) + (1-y'_i)\log(1-y_i))$$

رابطه 2.1: فرم کامل تابع cross entropy

$$H_{y'}(y) := -\sum_i y'_i \log(y_i)$$

رابطه 2.2: فرم کوتاه‌شده و فاکتور گرفته شده تابع cross entropy

سوال سوم: تاثیر learning rate و batch size در آموزش شبکه به چه صورت است؟

در پاسخ به این سوال می‌گوییم، شبکه‌های عصبی به صورت ماتریسی محاسبه می‌شوند. از این رو به جای این‌که ورودی‌ها را تک‌تک به شبکه دهیم و وزن‌ها را به روز کنیم می‌توانیم یک دسته ورودی به شبکه داده و بر اساس نتایج آن‌ها در مورد وزن‌ها تصمیم‌گیری کنیم. به این روش، روش Batch based می‌گوییم و خوبی این روش در این است که سرعت آموزش شبکه مخصوصا برای داده‌هایی با حجم بسیار زیاد، به‌شدت افزایش می‌یابد. این عمل ممکن است تاثیر بسیار اندکی رو دقت خروجی شبکه داشته باشد. تاثیر Learning Rate در شبکه به این صورت است که پرش‌های وزن‌ها براساس این فاکتور است. اگر این فاکتور به درستی انتخاب نشود ممکن است شبکه یک نقطه آپتیموم[1] را نادیده بگیرد و از روی آن پرش کند. در برخی حالت‌ها ممکن است شبکه هیچ یک از این نقاط را نبیند و حتی نتواند آموزش داده شود. از طرفی با کاهش مقدار این فاکتور ممکن است سرعت یادگیری بسیار کند شود. درصورت کاهش مقدار آلفا[2] دقت و احتمال رسیدن به یک نقطه آپتیموم بسیار بالا می‌رود ولی این امر در ازای کاهش سرعت آموزش شبکه است.

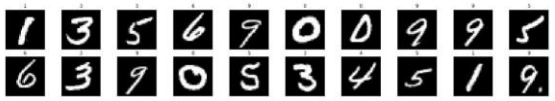
سوال چهارم: چرا از validation استفاده میکنیم؟

در پاسخ به این سوال می‌گوییم، شبکه‌ها درصورتی که تعداد نورون‌های زیادی در اختیار داشته باشند، شروع به حفظ کردن چند حالت ورودی می‌شوند و Generality خود را از دست می‌دهند. برای همین موضوع تعدادی از ورودی‌ها را (جدای از ورودی‌های شبکه) به مدار اعمال می‌کنیم که از صحت این شبکه اطمینان حاصل کنیم. به این ورودی‌ها validation set می‌گوییم.

3- طبقه‌بندی داده‌های مجموعه MNITS توسط یک شبکه Multi-Layer Perceptron (MLP)

پس از پاسخ به سوالات بخش قبل حال به سراغ طراحی شبکه‌ای جهت تشخیص داده‌های دسته MNITS می‌رویم.

الف) داده‌های MNITS را از وبسایت این گروه طبق روشی که TA اشاره کردند استخراج کرده و نمایش می‌دهیم. دسته‌داده‌ها در شکل ۳,۱ موجود است.



شکل ۳,۱: دسته داده‌های MNITS

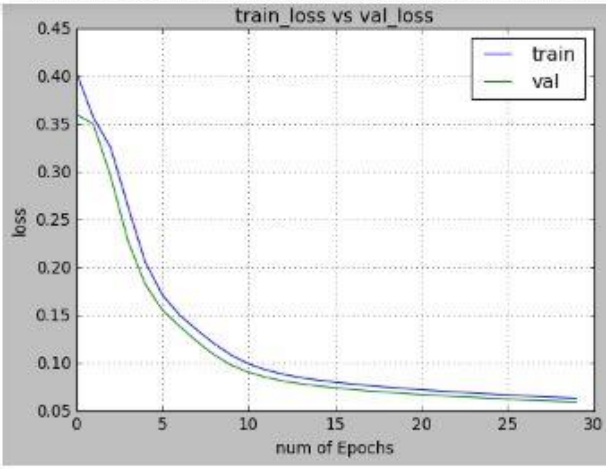
سپس لیبل‌های خروجی را مشخص می‌کنیم و با استفاده از کتابخانه Keras به طراحی شبکه می‌پردازیم. با توجه به دو لایه مخفی بودن شبکهٔ نیاز به سه لایه Dense که دوتای آن‌ها لایه‌های مخفی و یکی لایه خروجی می‌باشد شبکه را طراحی می‌کنیم. نتایج خواسته شده در قسمت الف و ب در جدول ۳,۱ موجود است و نتایج مربوط به تاثیر Learning_Rate , Batch Size در شکل‌های زیر به تزتیب آورده شده است.

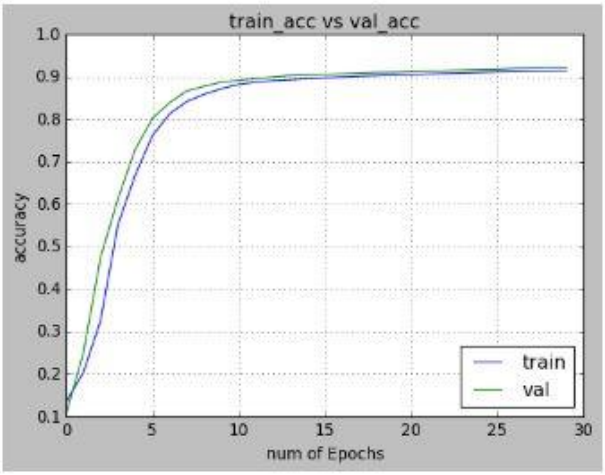
[1] Optimom

[2] Learning Rate

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 512)	401920
dense_9 (Dense)	(None, 512)	262656
dense_10 (Dense)	(None, 10)	5130

Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0



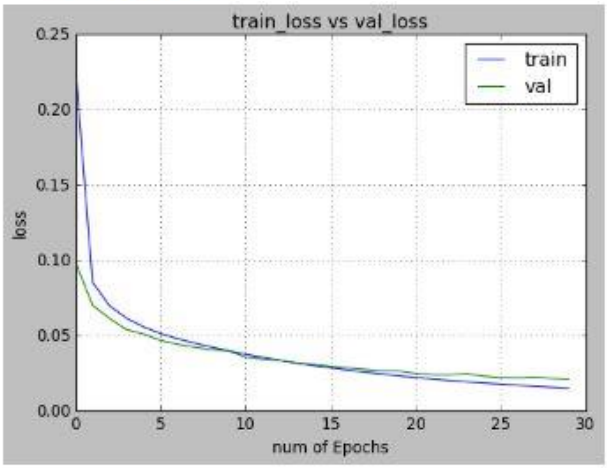


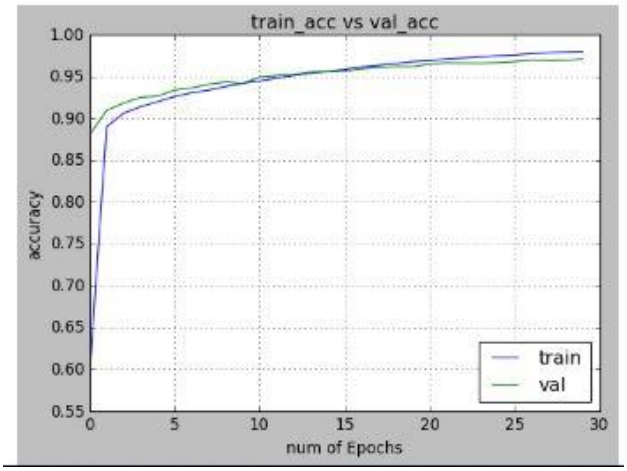
شکل ۳,۲: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.01 , Batch_Size: 32

Layer (type)	Output Shape	Param #
dense_11 (Dense)	(None, 512)	401920
dense_12 (Dense)	(None, 512)	262656
dense_13 (Dense)	(None, 10)	5130

Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0

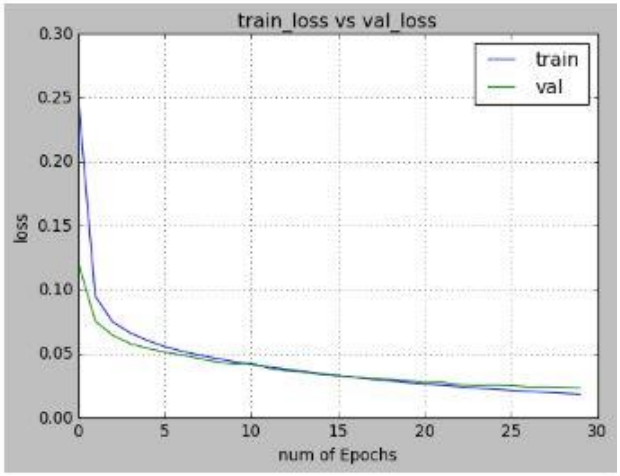
Test Loss: 0.08024307028576731
Test accuracy: 0.9232

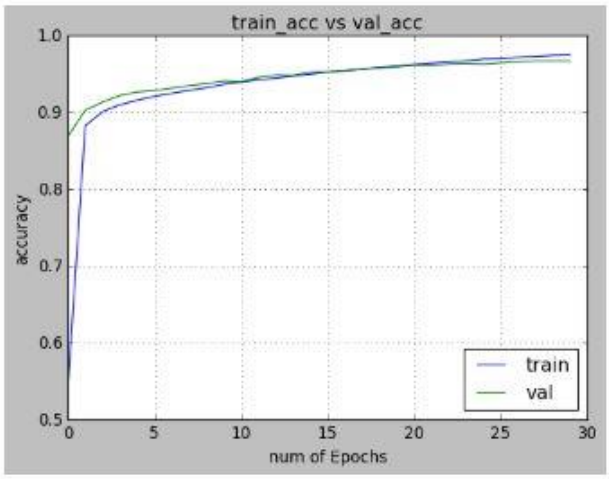




شکل ۳,۳: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.1 , Batch_Size: 32

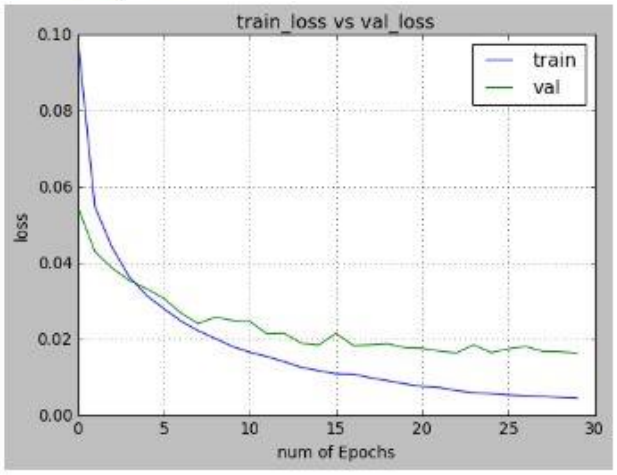
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 512)               401920
_________________________________________________________________
dense_18 (Dense)             (None, 512)               262656
_________________________________________________________________
dense_19 (Dense)             (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Test Loss: 0.06615394700677134
Test accuracy: 0.9273
```

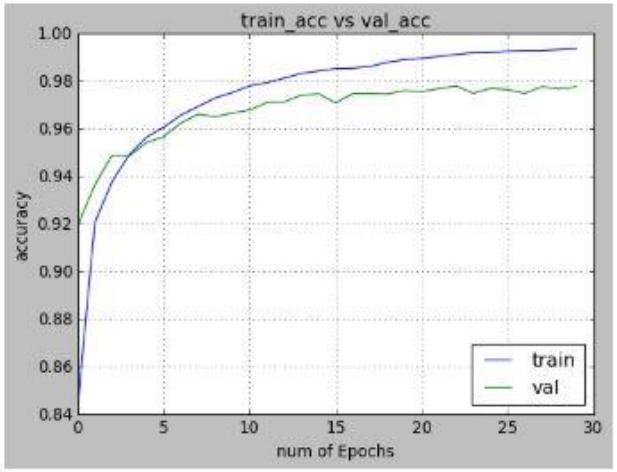




شکل ۳,۲: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.01 , Batch_Size:4

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 512)               401920
_________________________________________________________________
dense_15 (Dense)             (None, 512)               262656
_________________________________________________________________
dense_16 (Dense)             (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Test Loss: 0.019849788253017805
Test accuracy: 0.9731
```





شکل ۳,۴: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.1 , Batch_Size: 4

Batch_Size	Learnin_Rate	Validationscore	Test-Accuracy	Time (Min)
4	0.1	97.99%	97.32%	6’
4	0.01	98.32%	92.73%	9’
32	0.1	96.29%	92.32%	38’
32	0.01	91.32%	89.47%	67’

جدول ۳,۱: نتایج خواسته شده در سوال قسمت الف و ب

4- استفاده از Autoencoder برای کاهش فضای ویژگی-ها در داده‌های MNITS

پس از آموزش داده‌ها و مقایسه نتایج بر اساس Learning_Rate,Batch_Size حال از یک Autoencoder بهره می‌گیریم. کار Autoencoder کاهش فضای ویژگی‌ها می‌باشد. پس از اجرای عملیات نتایج را در جدول ۴,۱ با جدول ۳,۱ مقایسه می‌کنیم. نتایج مربوط به دقت تست به ترتیب در شکل-های ۴,۱ الی ۴,۴ موجود است. در این قسمت از برنامه از کتابخانه sklearn استفده می‌کنیم که بتوانیم شبکه را در این کتاب‌خانه راحت‌تر آموزش دهیم.

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 784)	0
dense_10 (Dense)	(None, 128)	100480
dense_11 (Dense)	(None, 64)	8256
dense_12 (Dense)	(None, 32)	2080
dense_16 (Dense)	(None, 512)	16896
dense_17 (Dense)	(None, 512)	262656
dense_18 (Dense)	(None, 10)	5130

Total params: 395,498
Trainable params: 395,498
Non-trainable params: 0

```
Test Loss: 0.029099167997564655
Test accuracy: 0.9609
```

شکل ۴,۱: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.1 , Batch_Size: 32

```
Test Loss: 0.043668862784281 37
Test accuracy: 0.9439
```

شکل ۴,۲ : خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.01 , Batch_Size: 32

```
Test Loss: 0.3999999507904053
Test accuracy: 0.1091
```

شکل ۴,۳: خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.1 , Batch_Size: 4

```
Test Loss: 0.030419730309955775
Test accuracy: 0.96
```

شکل.۴,۴ : خروجی‌های شبکه بر اساس پارامترهای Learnin_Rate: 0.01 , Batch_Size: 4

Batch_Size	Learning_Rate	With_Autoencoder	Without_Autoencoder
4	0.1	10.91%!!!!!!!!!!	97.32%
4	0.01	96%	92.73%
32	0.1	96.09%	92.32%
32	0.01	94.39%	89.47%

جدول ۴,۱ : مقایسه دو نتیجه با دو حالت مختلف ویژگی‌ها

5-پیاده‌سازی شبکه MLP پس از عبور ویژگی‌ها از واحد RBM بجای Autoencoder

پس از آموزش شبکه براساس ورودی‌های عبوری از یک Autoencoder حال بجای استفاده از یک Autoencoder از یک RBM استفاده می‌کنیم. برای بکارگیری RBM نیاز به استفاده از کتابخانهSklearn داریم. برای راحت‌تر شدن کار تابع MLP را نیز در همین کتابخانه تعریف می‌کنیم. نتایج در جدول ۵,۱ برای Batch_Size=32 , Learnin_rate_Initial=0.01 آمده است.

RBM_Method	Autoencoder_Method	Classic_MLP
81.38%	94.39%	89.47%

جدول ۵,۱: نتایج شبکه براساس استفاده از تکنیک‌های مختلف

با پیاده‌سازی تابع PCA بر داده‌های آموزش قبل و بعد از فشرده سازی مشاهده کردیم که واریانس داده‌ها از مقدار ۰,۰۹۵ به مقدار ۰,۰۷۳ مانند شکل ۵,۱ درآمد. این امر نشان می‌دهد که فشرده‌سازی ما باعث حذف داده‌های پرت مساله بوده است.

```
explained_variance
```

```
array([0.09704664, 0.07095924])
```

شکل ۵,۱ : خروجی تابع PCA

6-تخمین نرخ خانه در بوستون براساس شاخص‌های متفاوت توسط یک شبکه MLP

در این قسمت سعی داریم با آموزش یک شبکه MLP با یک و دو لایه مخفی تخمینی از قیمت خانه‌ها در شهر بوستون بزنیم و نتایج را مقایسه کنیم. ازین‌رو تابع خطای مربعات را برای‌هر ایپاک رسم می‌کنیم. پس از آموزش شبکه نتایج در شکل‌های ۶,۲ الی ۶,۶ و جدول ۶,۱ موجود است. در ابتدا با یک لایه شبکه با ۲۰ نورون و تعداد ایپاک کم شروع می‌کنیم. مشاهده می‌کنیم تعداد نورون و ایپاک‌ها هرکدام تاثیر خاصی بر دقت شبکه دارند. ولی این دقت تا حدی محدود قابل افزایش است و پس از آن تاثیری ندارد. از طرفی افزودن لایه دوم میتواند به افزایش دقت شبکه کمک کند. از طرفی فانکشن فعال‌ساز[3] در شبکه نقش

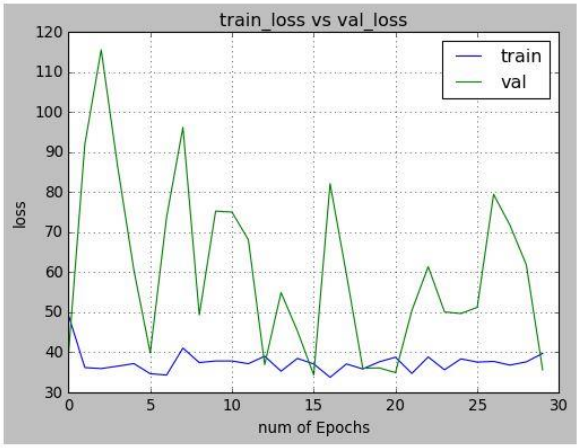
[3] Activation Function

بسیار مهمی در دقت شبکه ما دارد. بـر اسـاس دو مقایسـه بـین tanh و relu به این نتیجه رسیدیم که تابع relu نتایج بهتـری بـه ما می‌دهد.

Layer (type)	Output Shape	Param #
dense_73 (Dense)	(None, 20)	280
dense_74 (Dense)	(None, 1)	21

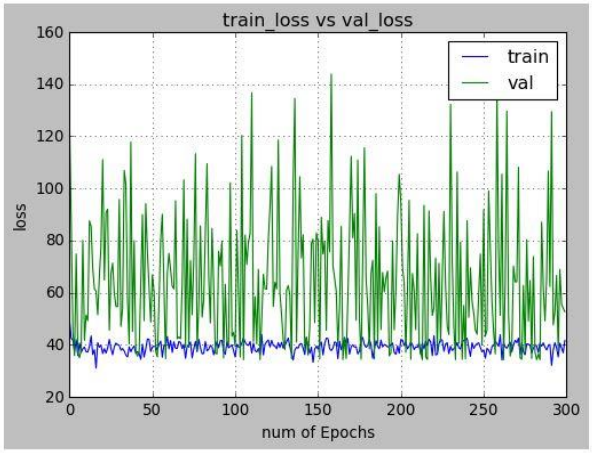
Total params: 301
Trainable params: 301
Non-trainable params: 0

Test Loss: 167.7854433224525
Test accuracy: 0.0024691358024691358



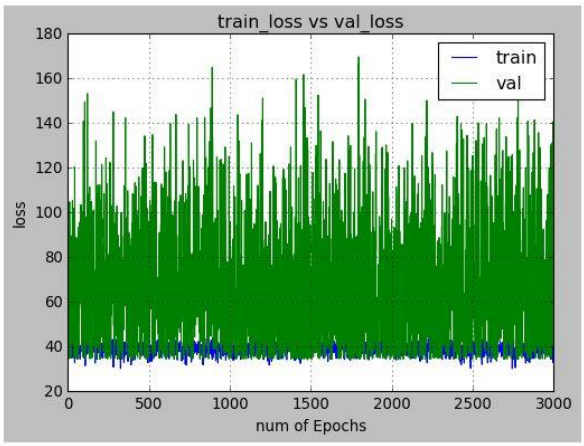
شکل ۶,۱: خروجی شبکه بر اساس تعداد ایپاک ۳۰ و تعـداد نورون ۲۰

Test Loss: 151.1745279194396
Test accuracy: 0.012345679058336917



شکل ۶,۲: خروجی شـبکه بـر اسـاس تعـداد ایپـاک ۳۰۰ و تعداد نورون ۲۰

Test Loss: 168.73666810871643
Test accuracy: 0.012345679058336917

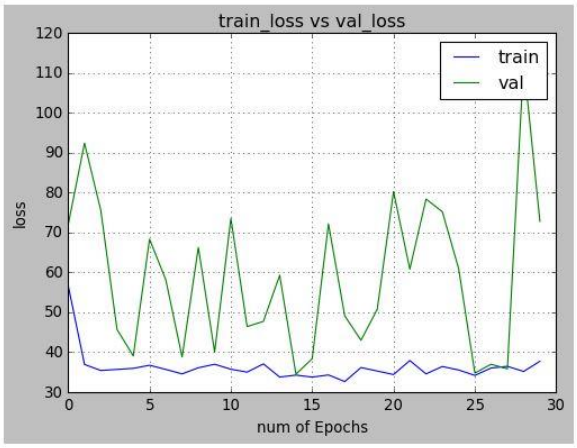


شکل ۶,۳: خروجی شبکه بـر اسـاس تعـداد ایپـاک ۳۰۰ و تعداد نورون ۲۰

Layer (type)	Output Shape	Param #
dense_13 (Dense)	(None, 20)	280
dense_14 (Dense)	(None, 18)	378
dense_15 (Dense)	(None, 1)	19

Total params: 677
Trainable params: 677
Non-trainable params: 0

Test Loss: 96.26028428254304
Test accuracy: 0.0049382716049382715

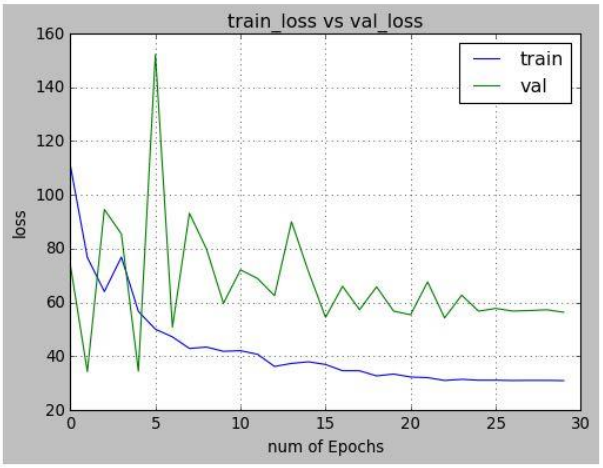


شکل ۶,۴: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌هـا و تـابع فعـال سـاز tanh

Layer (type)	Output Shape	Param #
dense_7 (Dense)	(None, 20)	280
dense_8 (Dense)	(None, 18)	378
dense_9 (Dense)	(None, 1)	19

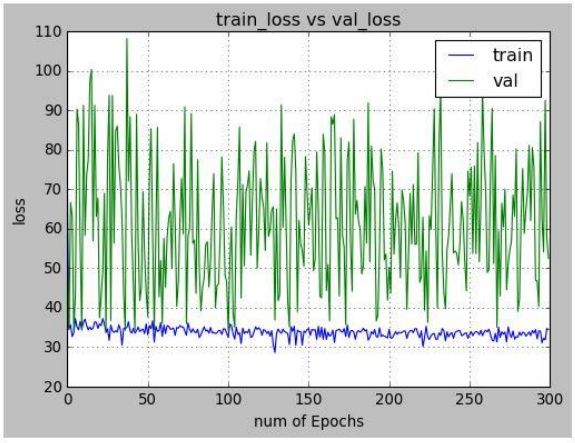
Total params: 677
Trainable params: 677
Non-trainable params: 0

Test Loss: 86.06229297967604
Test accuracy: 0.012345679058336917



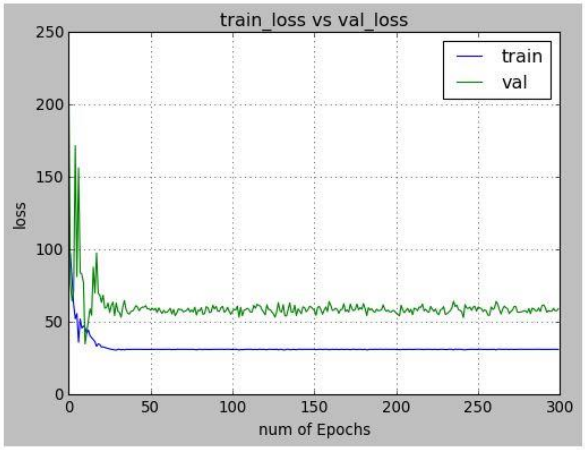
شکل ۶,۵: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌هـا و تـابع فعـال سـاز relu

```
Test Loss: 85.01287577122818
Test accuracy: 0.012345679058336917
```



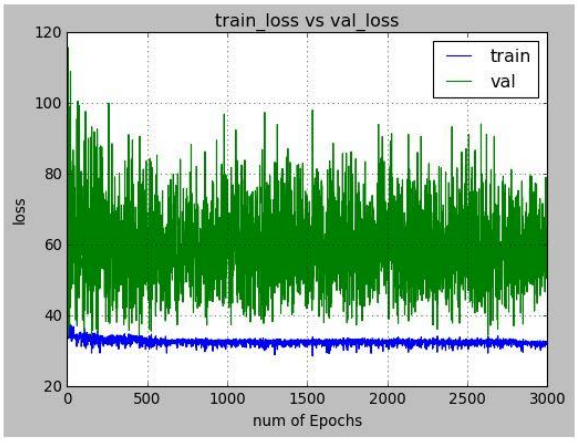
شکل ۶,۶: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌ها و تـابع فعـال سـاز tanh

```
Test Loss: 87.83857313085485
Test accuracy: 0.0024691358024691358
```



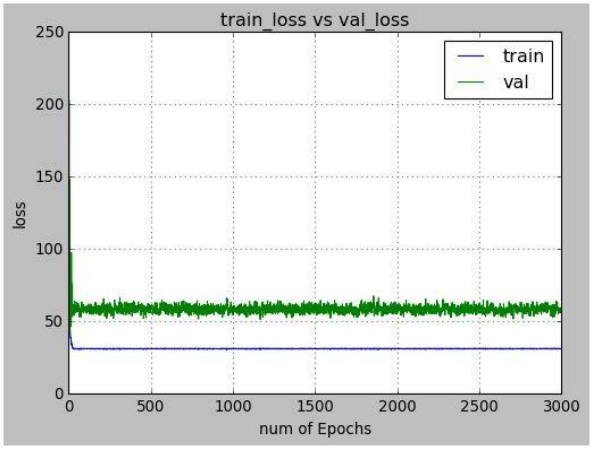
شکل ۶,۵: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌ها و تـابع فعـال سـاز relu

```
Test Loss: 89.79916490154501
Test accuracy: 0.0049382716049382715
```



شکل ۶,۵: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰۰۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌ها و تابع فعال سـاز tanh

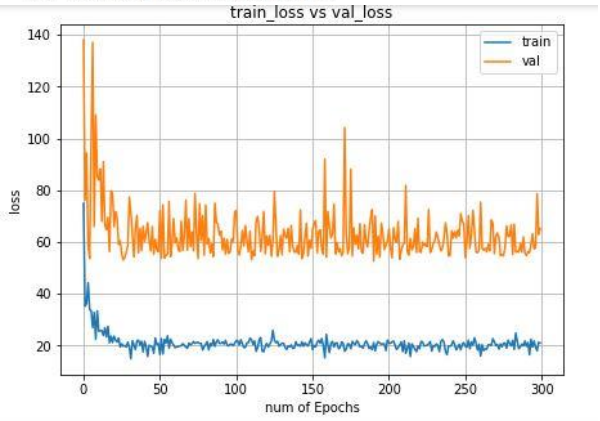
```
Test Loss: 85.41897917382511
Test accuracy: 0.012345679058336917
```



شکل ۶,۵: خروجی شبکه با دولایه بـر اسـاس تعـداد ایپـاک ۳۰۰۰ و تعداد نورون ۲۰ و ۱۸ به ترتیب لایه‌ها و تابع فعال سـاز relu

حال با استفاده از الگریتم PCA ابتدا فضای نمونـه‌را کـاهش می‌دهیم و سپس شبکه را آموزش می‌دهیم. مشاده می‌کنـیم کـه تابع هزینه بسیار کاهش و دقت تست ما افزایش می‌یابد.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 20)                60
_________________________________________________________________
dense_5 (Dense)              (None, 17)                357
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 18
=================================================================
Total params: 435
Trainable params: 435
Non-trainable params: 0
_________________________________________________________________
Test Loss: 97.82154260800209
Test accuracy: 0.0049382716049382715
```



شکل ۶,۶: خروجی شبکه با دولایه پس از کاهش فضای داده‌ها توسط الگوریتم PCA

برای دید بهتر نتایج را در کنار یکدیگر در جدول ۶,۱ آورده‌ایم.

ازین‌رو نتیجه می‌گیریم حداکثر خطا مربوط به روش شبکه تک‌لایه و حداقل خطا مربوط به روش PCA است.

نتیجه‌گیری

به صورت کلی کاهش Batch_Size باعث کاهش سرعت آموزش شبکه می‌شود ولی از طرفی دقت شبکه را بالا می‌برد. از طرفی کاهش Learnin_rate از ۰,۱ به ۰,۰۱ باعث کاهش سرعت می‌شود. استفاده از Autoencoder باعث افزایش دقت شد.

سپاسگزاری

با تشکر فراوان از جناب اشتری بابت راهنمایی در این تمرین.

مراجع

[1] ufldl.stanford.edu
[2] docs.scipy.org
[3] www.afternerd.com
[4] www.guru99.com
[5] www.geeksforgeeks.org

Epochs Neural	30	300	3000
One_*layer*_Neural_Network	167.75	151.174	168.73
Two_*layers*_Neural_Network_with_Tanh_Activation	96.26	85.01	89.79
Two_*layers*_Neural_Network_with_Relu_Activation	86.06	87.83	85.41
Two_*layers*_Neural_Network_with_PCA	73.74	82.92	78.34

جدول ۶,۱: شاخص خطای مربعات بر اساس دادگان تست در روش‌های مختلف