

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار ﺁﺯﻣﺎﻳﺶ ششم

ﺁﺯﻣﺎﻳﺸﮕﺎﻩ مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پیاده سازی CNN

نگارش

ارشیا اسمعیل طهرانی

علی بابالو

پویا ابراهیمی

استاد راهنما

سرکار خانم موسوی

آذر ماه 1401

# چکیده

در این آزمایش هدف پیاده سازی یک شبکه کانولوشنی روی(سی فار 10) است. شبكه عصبي كانولوشني از پركاربردترين شبكه هاي عصبي برای پردازش تصوير مي باشد. عملكرد بسياري از حملات نيز بر روي اين شبكه عصبي بررسي مي شوند .

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

**پیش گزارش....................................................................................................................................1**

ساختار یک شبکه عصبی CNN.................................................................................................................1

لایه کانوولوشنی...............................................................................................................................2

لایه ادغام...........................................................................................................................................2

لایه کاملا متصل.............................................................................................................................3

**شرح آزمایش..................................................................................................................................3**

معماری VGG.............................................................................................................................................4

Dropout Regularization..................................................................................................................4

تنوع Dropout Regularization.......................................................................................................5

Batch Normalization.........................................................................................................................5

Adam Learning Rate... ....................................................................................................................5

محیط Python..............................................................................................................................................6

دیتاست mnist...............................................................................................................................6

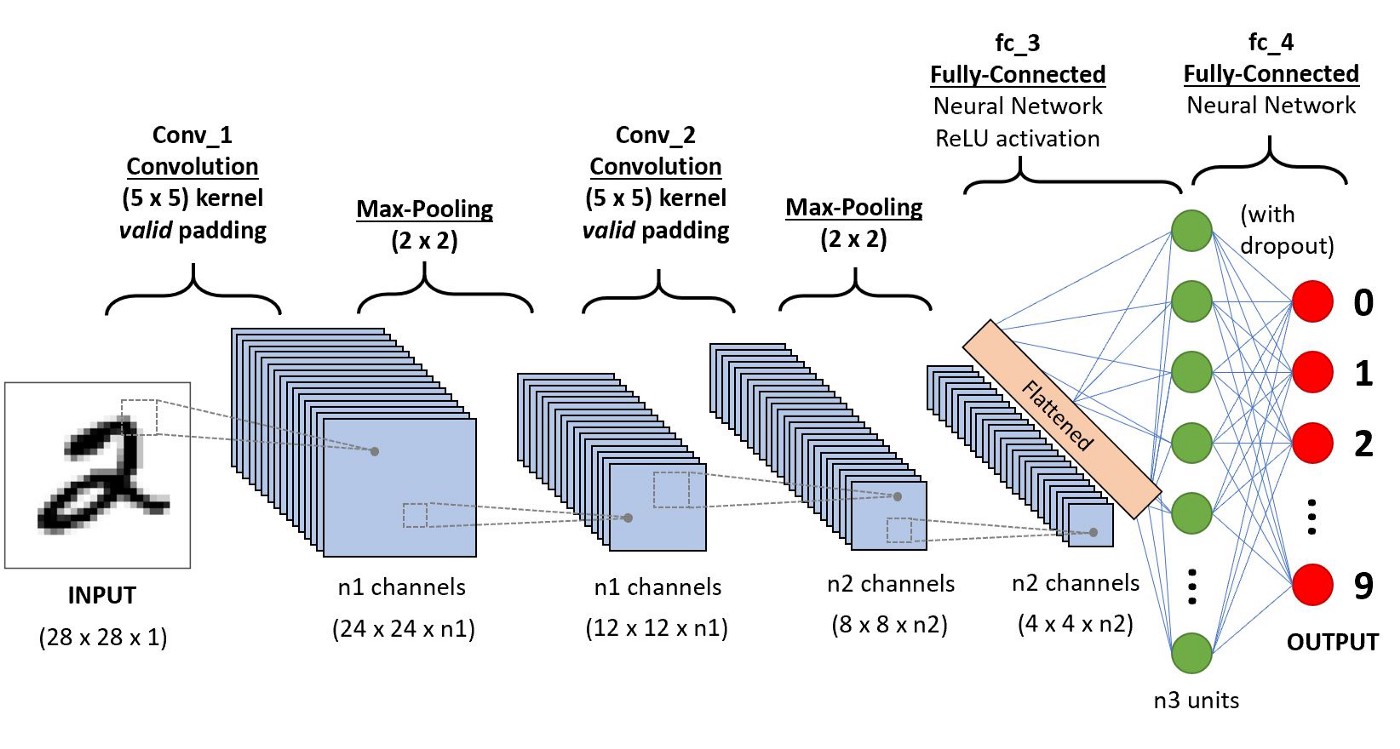
**تمارین........................................................................................................................................11**

دیتاست CIFAR-10 **.......................................................................................................**11

تمرین۲..........................................................................................................................................................21

# پیش گزارش

## ساختار یک شبکه عصبی CNN.

شبكه كانولوشني از لايه هاي اصلي زير تشكيل شده است:

شکل 1) نمونه ای شبکه عصبی کانوولوشنی

• لایه کانولوشنی

• لایه ادغام

• لایه کاملا متصل

### لايه كانولوشني:

شکل 2) عملگر کانوولوشن

در اين لايه عمل كانولوشن بر روي داده ورودي و با استفاده از تعدادي فيلتر انجام مي شود. براي محاسبه هر درايه خروجي، ماتريس فيلتر بر روي ماتريس ورودي لغزانده مي شود. عمل كانولوشن به اين صورت تعريف مي شود كه ابتدا اولين عنصر فيلتر بر روي اولين عنصر ماتريس ورودي قرار مي گيرد. سپس مجموع ضرب درايه هاي متناظر فيلتر با درايه هاي متناظز ماتريس ورودي محاسبه مي شود. در نهايت فيلتر بر روي تصوير ورودي به اندازه پارامتر از پيش تعيين شده s يه جلو برده مي شود. با تكرار اين مراحل ماتريس خروجي محاسبه مي شود.

### لايه ادغام:

يكي ديگر از لايه هاي شبكه كانولوشني لايه ادغام است. اين لايه هيچ پارامتر آموزشي ندارد. هدف اين لايه كاهش ابعاد ماتريس ورودي ورودي و همزمان حفظ اطلاعات ارزشمند ورودي است. در اين لايه ابتدا ماتريس با ابعاد از پيش تعيين شده k در نظر گرفته مي شود. اين پارامتر معمولا برابر با ۲ در نظر گرفته مي شود. سپس با لغزاندن فيلتر بر روي ورودي اندازه ماتريس ورودي كاهش مي يابد. يك نمونه از عملگرهايي كه براي نمونه برداري در اين لايه استفاده مي شود، عملگر بيشينه است. در اين حالت ماتريس از پيش تعيين شده بر روي داده ورودي لغزانده مي شود و تنها بيشينه عناصري كه در هر بخش قرار مي گيرند را به عنوان خروجي در نظر مي گيرد.

### لايه كاملا متصل:

در اين لايه يك شبكه عصبي كاملا متصل قرار گرفته است. در اين لايه هدف مرتبط كردن ماتريس نهايي با خروجي هاي نهايي شبكه است. وزن هاي شبكه كاملا متصل از طريق پس انتشار خطا بدست مي آيد.

# شرح آزمایش

CIFAR مخفف عبارت Canadian Institute for Advanced Research است و مجموعه داده CIFAR-10 همراه با مجموعه داده CIFAR-100 توسط محققان موسسه CIFAR توسعه یافته است.

مجموعه داده شامل 60000 عکس رنگی 32×32 پیکسل از اشیاء از 10 کلاس، مانند قورباغه، پرندگان، گربه‌ها، کشتی‌ها و غیره است. برچسب‌های کلاس و مقادیر عدد صحیح مرتبط با آنها در زیر فهرست شده‌اند.

1. هواپیما
2. خودرو
3. پرنده
4. گربه
5. آهو
6. سگ
7. قورباغه
8. اسب
9. کشتی
10. کامیون

این تصاویر بسیار کوچک هستند، بسیار کوچکتر از یک عکس معمولی، و مجموعه داده برای تحقیقات بینایی کامپیوتری در نظر گرفته شده است.

CIFAR-10 یک مجموعه داده کاملاً درک شده است که به طور گسترده برای محک زدن الگوریتم‌های بینایی رایانه در یادگیری ماشین استفاده می‌شود. مشکل حل شده است." دستیابی به دقت طبقه بندی 80 درصد نسبتاً ساده است. عملکرد برتر در مورد مشکل با یادگیری عمیق شبکه های عصبی کانولوشن با دقت طبقه بندی بالای 90٪ در مجموعه داده آزمایشی به دست می آید.

اکنون می‌توانیم یک مدل پایه برای مجموعه داده CIFAR-10 بررسی کنیم.

## معماری VGG

یک مدل پایه حداقل عملکرد مدل را ایجاد می کند که همه مدل های دیگر ما را می توان با آن مقایسه کرد و یک معماری مدلی که می توانیم به عنوان مبنای مطالعه و بهبود استفاده کنیم.

یک نقطه شروع خوب، اصول کلی معماری مدل های VGG است. اینها نقاط شروع خوبی هستند زیرا در رقابت ILSVRC 2014 به عملکرد برتر دست یافتند و به دلیل اینکه ساختار مدولار معماری قابل درک و پیاده سازی آسان است.

این معماری شامل انباشته شدن لایه‌های کانولوشن با فیلترهای کوچک 3×3 و به دنبال آن یک لایه ترکیبی حداکثر است. این لایه‌ها با هم یک بلوک را تشکیل می‌دهند و این بلوک‌ها می‌توانند در جایی که تعداد فیلترها در هر بلوک با عمق شبکه افزایش می‌یابد، تکرار شوند، مانند 32، 64، 128 و 256 برای چهار بلوک اول مدل. بالشتک روی لایه های کانولوشن برای اطمینان از مطابقت ارتفاع و عرض نقشه های ویژگی خروجی با ورودی ها استفاده می شود.

ما این معماری را در مسئله CIFAR-10 بررسی می کنیم و مدلی را با این معماری با بلوک های 1، 2 و 3 مقایسه می کنیم.

## Dropout regularization از جمله راه های بهبود مدل

Dropout یک تکنیک ساده است که به طور تصادفی گره ها را از شبکه خارج می کند. این یک اثر منظم کننده دارد زیرا گره های باقی مانده باید برای برداشتن سستی گره های حذف شده سازگار شوند.

Dropout را می توان با افزودن لایه های Dropout جدید به مدل اضافه کرد، جایی که مقدار گره های حذف شده به عنوان یک پارامتر مشخص می شود. الگوهای زیادی برای افزودن Dropout به یک مدل وجود دارد، از نظر اینکه در کجای مدل باید لایه‌ها را اضافه کرد و از چه تعداد حذفی استفاده کرد.

در این حالت، لایه‌های Dropout را بعد از هر لایه جمع‌آوری حداکثر و بعد از لایه کاملاً متصل اضافه می‌کنیم و از نرخ خروج ثابت 20% استفاده می‌کنیم (به عنوان مثال، 80٪ گره‌ها را حفظ می‌کنیم).

## تنوع dropout regularization

یک تغییر این است که میزان dropout از 20% به 25% یا 30% افزایش پیدا کند. یکی دیگر از تغییراتی که ممکن است جالب باشد، استفاده از الگوی افزایش dropout از 20% برای بلوک اول، 30% برای بلوک دوم و به همین ترتیب به 50% در لایه کاملا متصل در بخش طبقه‌بندی‌کننده مدل است.

این نوع افزایش dropout با عمق مدل یک الگوی معمولی است. موثر است زیرا لایه‌های عمیق مدل را مجبور می‌کند تا بیش از لایه‌های نزدیک به ورودی را منظم کنند.

## استفاده از Batch Normalization

نرمال سازی دسته ای در تلاش برای تثبیت یادگیری و شاید تسریع روند یادگیری اضافه شده است. برای جبران این شتاب، از الگوی dropout فزاینده استفاده می شود.

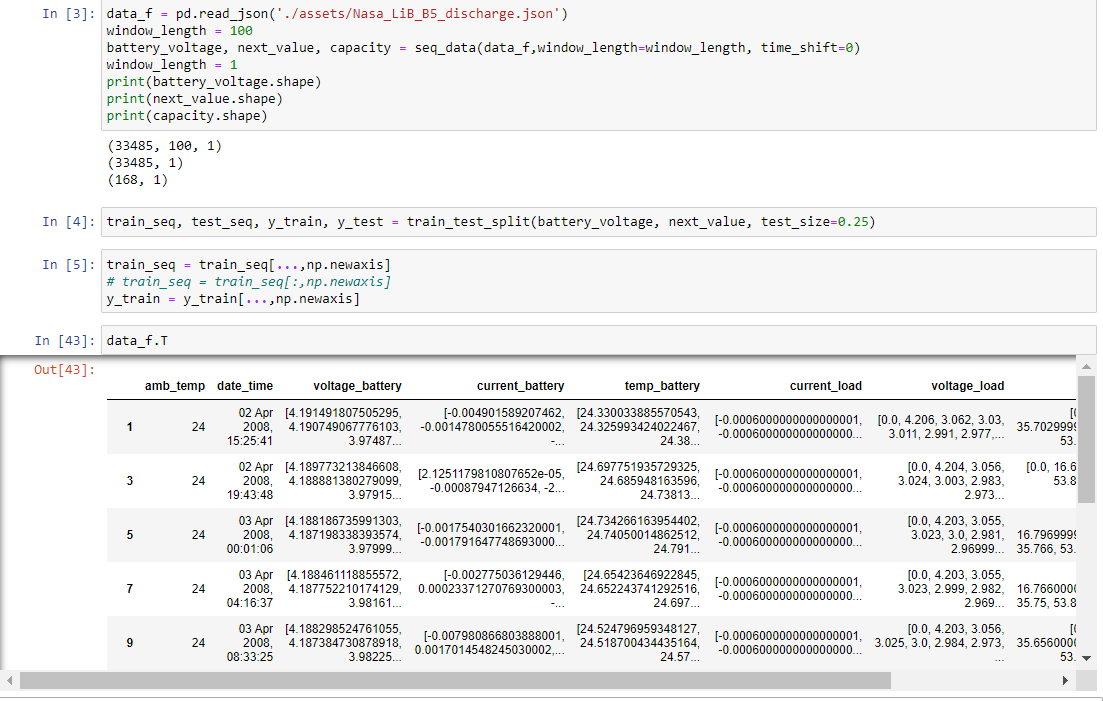
در ادامه به دو راهکار دیگر که برای تقویت این مدل استفاده شده است نیز پرداخته شده است؛

## استفاده از Adam learning rate

بهینه‌سازی آدام یک روش stochastic gradient descent است که مبتنی بر تخمین تطبیقی ممان‌های مرتبه اول و مرتبه دوم است. به خاطر ذات تطبیقی یا adaptive بسیار مورد استفاده قرار می‌گیرد به گونه ای که به گفته کینگما و همکاران، 2014، این روش "از نظر محاسباتی کارآمد است، نیاز به حافظه کمی دارد، نسبت به مقیاس مجدد مورب گرادیان ها ثابت است، و برای مسائلی که از نظر داده/پارامترها بزرگ هستند، مناسب است."

## محیط Python

به طور کلی مراحل زیر در پیاده سازی این پروژه طی شده است؛  
در ابتدا پس از import کردن کتابخانه های مورد نیاز به لود کردن دیتاست باتری می­پردازیم که در تصویر پایین آنرا مشاهده می­کنیم:

شکل؟؟) لود کردن دیتا و پنجره کردن دیتا

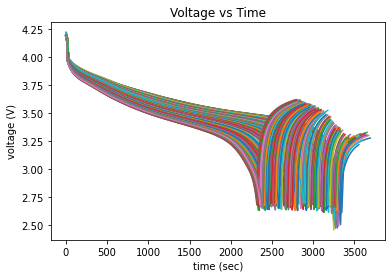
پس از لود کردن دیتاست با استفاده از تابع seq\_data() دیتاست را به پنجره های دلخواهمان جدا می­کنیم.

شکل؟) پنجره متحرک بر روی دیتاست

**اون دوتا حلقه اول رو توضیح بده**

پس از آنکه پنجره بر روی دیتا های هر سایکل حرکت کرد لیست lst\_bv, lst\_nv مان لیست ۳ بعدی که بعد اول ان برابر تعداد سایکل ها -۱۶۸- است که شامل لیستی از هر پنجره داخل سیکل است. یعنی در سیکل اول که شامل ولتاژ ثبت شده در ۱۹۷ تایم استپ هستش را اگر با پنجره های ۱۰۰ تایی جدا کنیم، آرایه اول لیست ۱۶۸تاییمان شامل ۹۷ لیست ۱۰۰تایی از ولتاژ ها می­باشد. برای آنکه بتوانیم این ولتاژ هارا به شبکه RNNمان بدهیم این لیست ۳ بعدی را به ۲ بعد تقلیل می­دهیم به این صورت که همه پنجره های تمام سیکل هایمان را پست سر هم قرار می­دهیم و بدین صورت ابعاد ماتریسمان بصورت (طول پنجره \* مجموع تمام پنجره ها در تمامی سیکل ها) می­شود. به عنوان مثال در پنجره با طول ۱۰۰ ابعاد ماتریس ورودی به صورت ۱۰۰\*۳۳۴۸۵ می­شود. سپس داده را به ۲ بخش اموزش و تست تقسیم می­کنیم.

در ابتدا ولتاژ دشارژ باتری را برای هر ۱۶۸ سیکل رسم می­کنیم که آنرا در شکل زیر مشاهده می­کنید.

شکل؟) ولتاژ تمامی سیکل ها بر اساس زمان

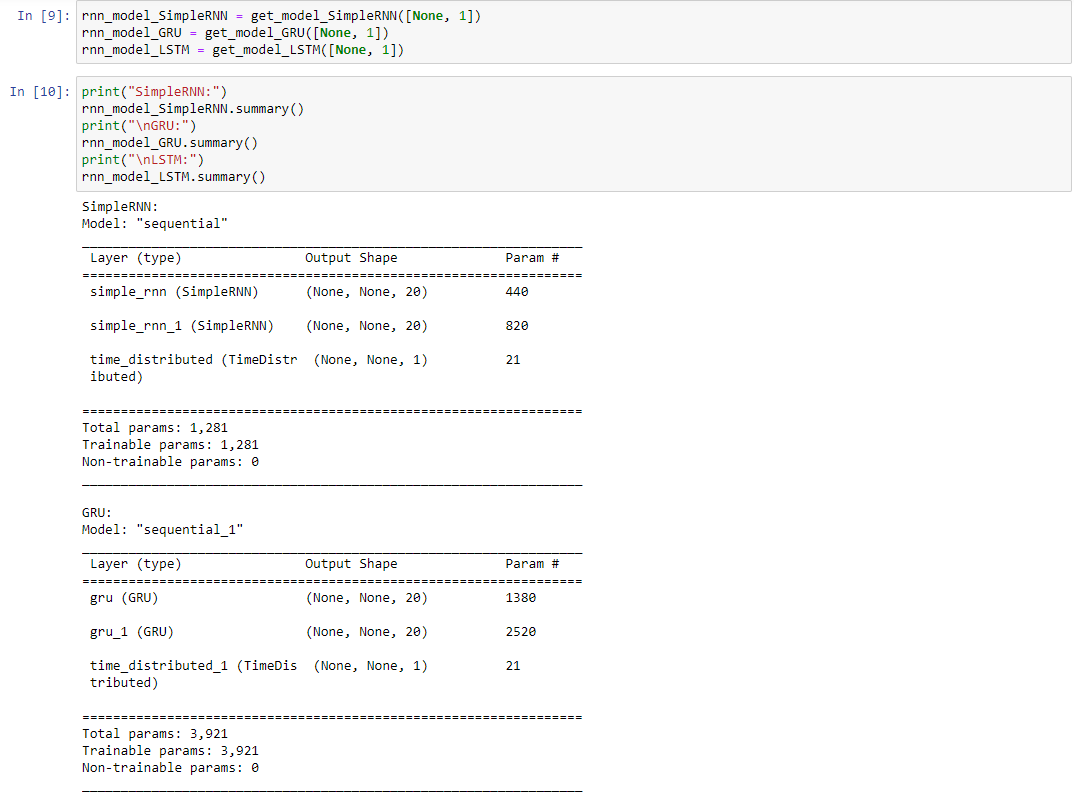
سپس به تعریف کردن مدلمان می­پردازیم. برای این دیتاست از ۳ مدل مختلف با لایه های SimpleRNN, LSTM, GRU بهره می­بریم که ۳ مدل تعریف شده را در شکل زیر مشاهده می­کنید.



شکل؟) مدل های تعریف شده

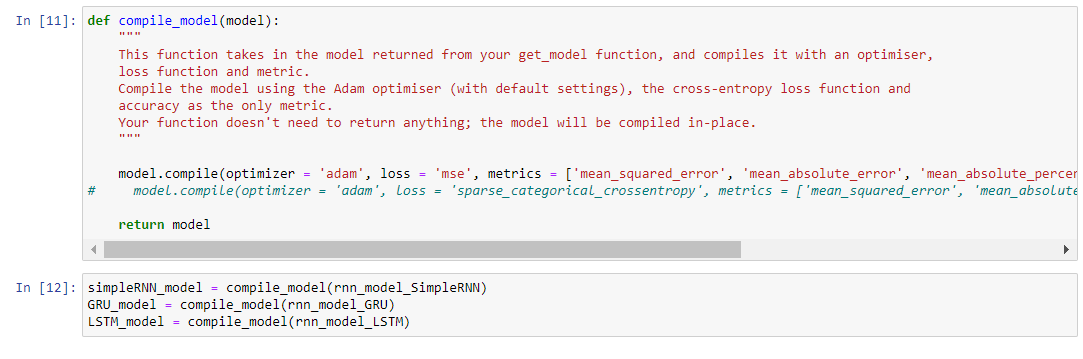
در هر ۳ مدل تعریف شده از ۲ لایه شبکه مورد نظر استفاده کردیم و در لایه آخر نیز یک لایه Dense قرار دادیم. نکته قابل تامل این است که با توجه به صورت سوال برای داشتن یک شبکه عصبی بازگشتی seq-to-seq باید در ارگیومنت های ورودی لایه مان ارگومان return\_sequences را True قرار دهیم برای اینکه seq-to-vec باشد باید False قرار دهیم(در تمرین مربوط به ظرفیت باتری باید False باشد).

سپس مدل هایمان را با استفاده از توابع بالا تعریف می­کنیم . سامری آنها را نیز می­توانید در تصویر ببینید.

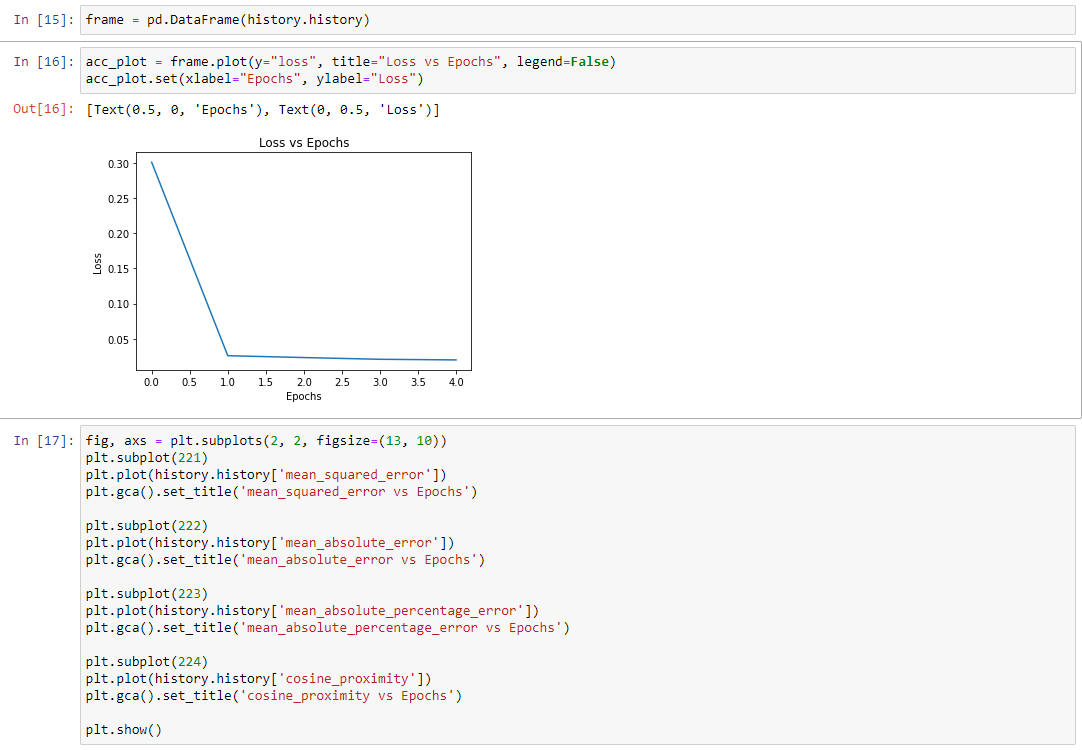


شکل؟) تعریف مدل های شبکه بازگشتی

سپس نوبت به کامپایل کردن مدل هایمان می­رسد:

شکل؟) کامپایل کردن مدل های شبکه بازگشتی

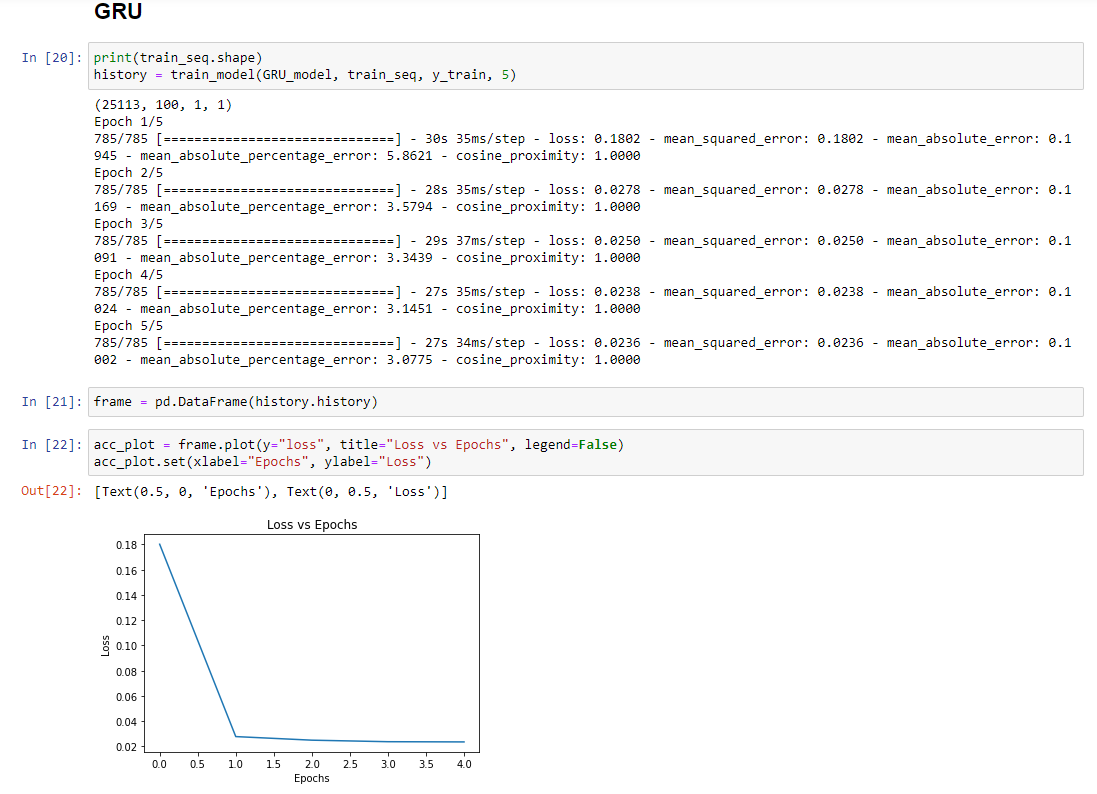
بعد از کامپایل کرد شبکه هایمان با استفاده از دستور fit() هر ۳ شبکه را اموزش می­دهیم که نتیجه در پایین مشخص است. در اینجا ما برای loss از mse استفاده کردیم اما مقدار خطا شبکه با متریکس های دیگر نظیرmean-absolute-error, mean-absolute-percentage-error, cosine-proximity بدست آوردیم.

شکل؟) مقدار خطا شبکه SimpleRNN بعد از اتمام آموزش

Shape

Description automatically generated

شکل؟) خطا شبکه SimpleRNN با متریکس ها مختلف

شکل؟) خطا مدل GRU پس از آموزش

Shape, rectangle

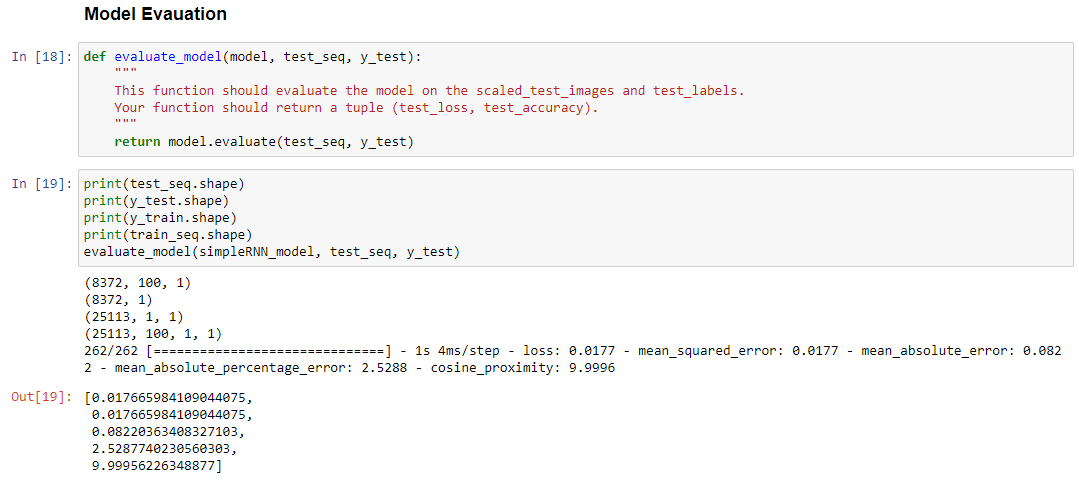
Description automatically generated شکل؟) خطا مدل GRU با متریکس های مختلف

شکل؟) خطا مدل LSTM پس از اتمام آموزش

Shape, rectangle

Description automatically generatedشکل؟) خطا شبکه LSTM با متریکس های مختلف

پس از اتمام آموزش با متد evaluate شبکه هارا با استفاده از داده های تست دقت سنجی می­کنیم:

شکل؟) تست شبکه SimpleRNN

Graphical user interface, text, application, email

Description automatically generatedشکل؟) تست شبکه GRU

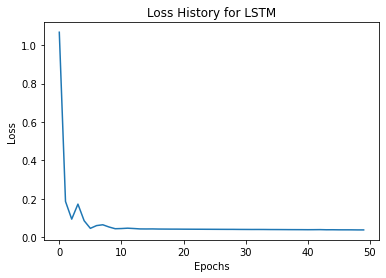
Graphical user interface, text, application, email

Description automatically generatedشکل؟) evaluate کردن شبکه LSTM

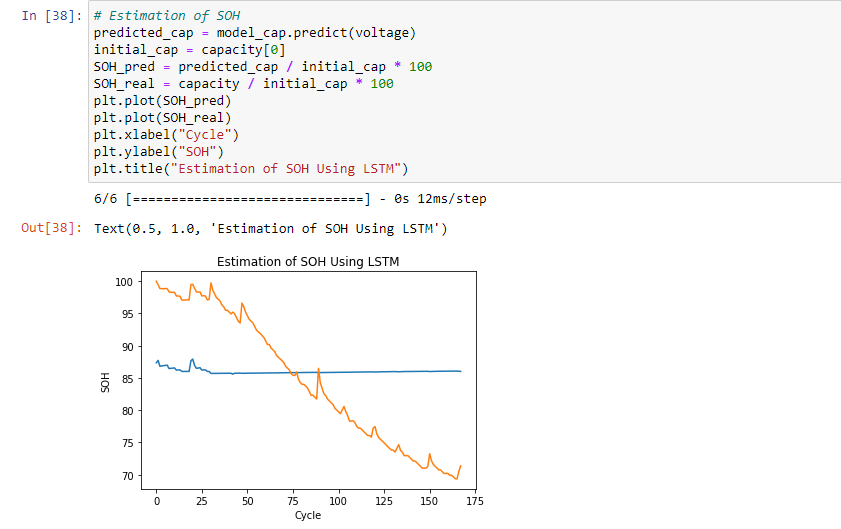
## ۳-۱ تمارین

در تمرین دوم از ما خواسته شده است که با طراحی شبکه بازگشتی میزان سلامت باتری را بسنجیم. ابتدا یک شبکه بازگشتی LSTM تعریف میکنیم با ۵۰ نورون در لایه مخفی سپس این شبکه را با داده های تستمان(ولتاژ) و لیبل خروجی که ظرفیت باتری است آموزش می­دهیم. سپس با استفاده از متد predict میزان ظرفیت باتری را پیشبینی می­کنیم.

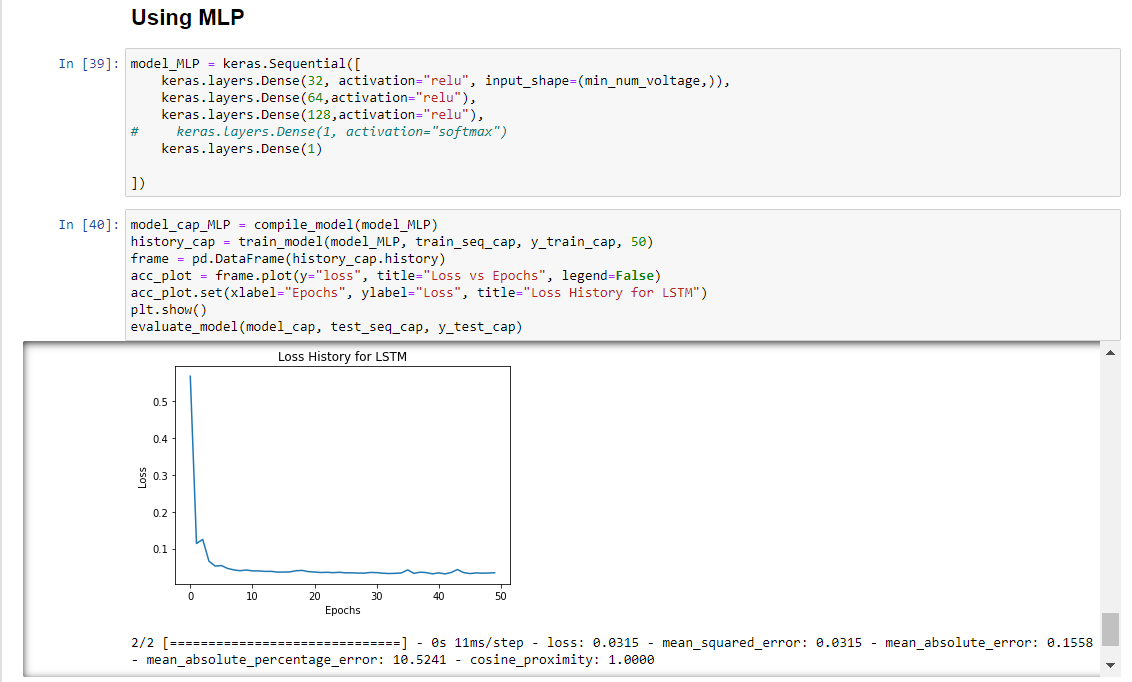
شکل؟) شبکه LSTM برای پیشبینی حالت سلامت باتری

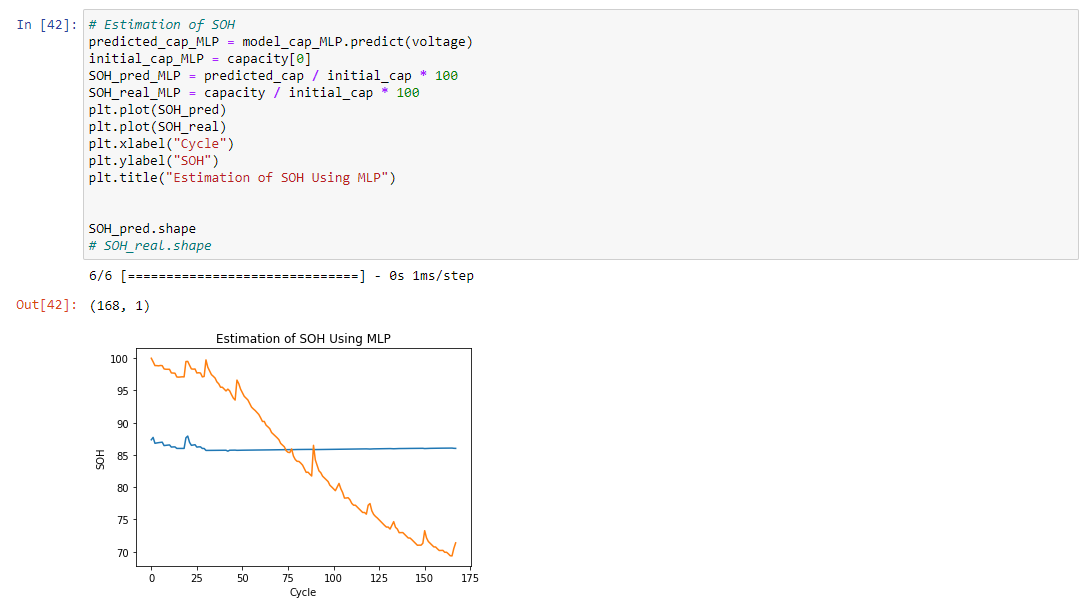
شکل؟) خطا شبکه LSTM آموزش داده شده برای پیدا کردن سلامت باتری

سپس با استفاده از فرمول داده شده سلامت باتری را می­سنجیم:

شکل؟) سلامت باتری پیش­بینی شده

در تمرین ۳ از ما خواسته شده سلامت باتری را در این مرحله با استفاده از یک شبکه MLP بدست بیاورید که شبکه طراحی شده ما شامل ۲ لایه مخفی می­باشد که آن را آموزش دادیم.

شکل؟) شبکه MLP طراحی شده

شکل؟) سلامت پیشبینی شده باتری با استفاده از شبکه MLP

**پایان**