به نام خدا

تمرین سری سیزدهم درس یادگیری عمیق

دکتر محمدی

محمد یارمقدم

۹۶۴۶۲۱۰۴

سوال اول)

روش های یادگیری مورد استفاده در شبکه های عصبی عمیق شامل موارد زیر می‌شود:

- Suprevised learning
- Unsupervised learning
- Self-Supervised learning
- Representation learning

در ابتدا به بررسی supervised learning می‌پردازیم. این نوع یادگیری در واقعا یادگیری با وجود ناظر است. در این روش برای آموزش و یادگیری شبکه از داده هایی که از قبل لیبل خورده‌اند استفاده می‌کنیم. این روش بیشترین کاربرد را در بین روش های یادگیری دارد و از نمونه های معروفی که از این روش استفاده می‌کنند می‌توان به linear and logistic regression و SVM و ... اشاره کرد. این روش داده های موجود در شبکه را دسته‌بندی می‌کند و می‌توان به کمک آن خروجی های شبکه را با دقت خوبی پیش‌بینی کرد.

در روش بعد به بررسی unsupervised learning می‌پردازیم. در این روش، از متدها و الگوریتم‌های machine learning استفاده می‌شود. هدف در این روش یافتن روابط و الگوهای موجود در میان داده های شبکه بدون استفاده از داده های آموزشی و دخالت انسان است. درواقع در این روش در تمام طول مجموعه داده سعی در یافتن برخی ساختار ها را داریم. درحالیکه در شکل داده تغییری ایجاد نمی‌کند و از شکل اصلی داده‌ها استفاده می‌کند.

در اینجا به بررسی SELF-SUPERVISED LEARNING می‌پردازیم. این روش به داده لیبل شده و دخالت انسانی نیاز ندارد. این روش توسط الگوریتم هایی که داده ها را لیبل دهی می‌کند آغاز به کار می‌کند و بدون نیاز به اطلاعات و پردازش انسان داده ها را لیبل گذاری می‌کند. در روش SUPERVISED برخی از شبکه هایی که از آن برای آموزش شبکه استفاده می‌کنند بیش از حد وابستگی میان داده و آموزش شبکه به وجود می‌آید. به این معنی که تعداد زیادی داده برای آموزش شبکه لازم است. در روش SELF-SUPERVISED هدف کاهش وابستگی به تعداد زیادی داده برای آموزش است. در واقع در روش SUPERVISED فقط تکیه بر داده برای آموزش است ولی در روش SELF-SUPERVISED روابط و الگو های داده ها را پیش‌بینی می‌کند. در واقع بخشی از داده را پنهان می‌کنیم تا شبکه بتواند رفتار آن را پیش‌بینی کند. در این نوع یادگیری تفاوتی که داریم نسبت به حالت SELF-SUPERVISED این است که در این حالت از DYNAMIC داده ها برای یادگیری استفاده می‌شود. DYNAMIC هم در سطح خام تعریف می‌شود.

در آخر به بررسی REPRESENTATION LEARNING می‌پردازیم. در این روش نیاز به استخراج دستی فیچر و ویژگی‌ها کاهش می‌یابد. این کار توسط استخراج بازنمایی های مختلف از داده های ورودی برای دسته‌بندی و طبقه‌بندی داده ها صورت می‌گیرد. کاهش نیاز به اطلاعات دست ساز توسط یادگیری ماشین از ویژگی‌ها اعمال می‌شود. در این روش بر روی داده‌های با ابعاد بالا، کاهش بعد صورت گرفته و استخراج پترن و الگو از آنها ساده‌تر می‌شود. شبکه در این روش به تنهایی بازنمایی داده را می‌آموزد. این روشی برای تعیین نمایش داده‌ای از ویژگی‌ها، تابع فاصله و تابع شباهت است که نحوه عملکرد مدل پیش بینی را تعیین می‌کند.

منابع:

https://towardsdatascience.com/supervised-semi-supervised-unsupervised-and-self-supervised-learning-7fa79aa9247c

https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

https://www.quora.com/What-is-the-difference-between-supervised-unsupervised-and-self-supervised-machine-learning

سوال دوم)

الف) این معیار در واقع Accuracy شبکه خودناظر است. در واقع میزان صرفه‌جویی و سودمندی شبکه را بیان می‌کند. به این معنی اگر ما بخواهیم در این روش به دقت حالتی که شبکه خودناظر نیست دست پیدا کنیم چه تعداد لیبل دیگر نیاز داریم.

ب) در مقاله ذکر شده از چهار زیر به عنوان downstream tasks یاد شده‌است:

* Object classification

* Object pose estimation

* Semantic segmentation

* Depth estimation

یکی از فاکتورهای انتخاب این تسک‌ها و وظایف سادگی و سهولت آنان می‌باشد. این تسک‌ها نیازی به پردازش های پیچیده و پایپلاین های دو یا چند مرحله‌ای ندارند. تصمیم برای انتخاب این تسک‌ها برای محدودکردن هایپرپارامترهای فاز آموزش است و تمرکز را روی فاز پیش‌آموزش بیشتر می‌نماید. اما نکته‌ای که این وظایف دارند ایجاد تضاد بین ویژگی های semantic و geometric است. از سوی دیگر این تضاد بین ویژگی‌های global و dense نیز ایجاد می‌شود. Semantic information شامل classification و segmentation می‌شود و geometric information شامل pose و depth میشود.

تسک اول ظبقه‌بندی اشیا است. این مورد یکی از معیار های اصلی برای self-supervised در فاز پیش‌آموزش است. در این روش شبکه را بین ده کلاس shapenet آموزش می‌دهیم تا تصاویر به گونه ای تولید شوند که هر یک فقط یک شی واحد را شامل شوند و توزیعی یکنواخت بین آنها برقرار باشد.

تسک دوم تخمین ژست است. برای این تسک ما دوباره عکس‌هایی که فقط شامل یک شی هستند را انتخاب می‌کنیم. به جای پیش‌بینی ماتریس چرخش کامل ما فقط ژست را به صورت گسسته بدست می‌آوریم. به این صورت که زست را به ۵ دسته تقسیم می‌کنیم و یک classifier را آموزش می‌دهیم. این ۵ حالت به گونه ای انتخاب می‌شوند که جهت‌گیری در امتداد محور نادیده گرفته شود. در نهایت مدل پیش‌بینی می‌کند که صورت جسم به سمت بالا، پایین، چپ، راست و یا به سمت روبرو است.

تسک سوم تقسیم بندی معنایی است. در این تسک تصاویر با چند شی رندر یا تولید می‌شوند. در اینجا نگرانی برای طراحی مدل با خروجی با رزولوشن بالا نیست. پس درعوض وضوح ناظر در این تسک نبست به رزولوشن عکس های ورودی بالاتر است. درنتیجه یک تابع خطای انتروپی برای هر پیکسل اعمال و میانگین دقت classification را بدست می‌آوریم.

تسک آخر تخمین عمق است. تخمین عمق بر روی تصاویری معنا دارد که دارای چندین شی باشند و نسبت به عکس های ورودی وضوح بیشتری داشته باشند. تخمین عمق در این تسک با نظارت L1 Loss انجام می‌شود و دقت با یک متریک استاندارد اندازه‌گیری می‌شود که درصد پیش‌بینی‌های که با نرخ ground truth depth خطا اندازه‌گیری می‌شوند را اندازه‌گیری می‌کند.

ج) روش های pretraining (پیش‌آموزش) معرفی شده شامل موارد زیر است:

* Viriational autocoder

* Rotation

* CMC(Contrastive Multiview Coding)

* AMDIM(Augmented Multiscale Deep InfoMax)

در الگوریتم اول، یک استانداردی تعیین شده است که تضاویر را به فضایی با ابعاد کمتر نگاشت کند.

در الگوریتم دوم الگوریتمی ساده برای پیش‌آموزش داده‌ها تعریف شده که پیش‌بینی می‌کند تصاویر بین ۰، ۹۰ ، ۱۸۰ و یا ۲۷۰ درجه چرخیده است.

در الگوریتم سوم، تصویر به کانال های مختلف تقسیم می‌شود. کانال‌هایی مانند l و ab . کانال‌های ایجاد شده از دو شبکه نیمه رد می‌شوند و embedding های خروجی به embedding های دیگر از مقایسه تصاویر دیگر نگاشت می‌شوند.

در الگوریتم آخر نیز به جای مقایسه بین کانال‌های ایجاد شده از تصاویر، representation های تقویت‌شده از تصویر و خروجی لایه‌های میانی شبکه مقایسه می‌شوند.

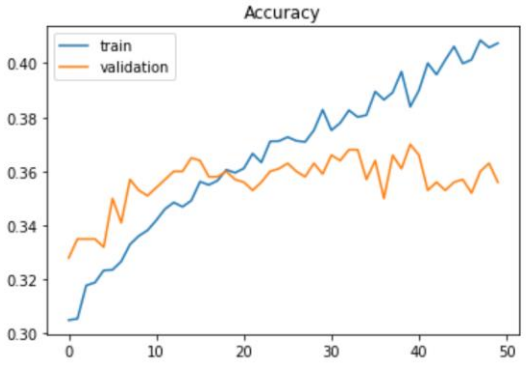
سوال سوم)

در این سوال هدف طراحی سیستم پرسش و پاسخ است. برای این کار ابتدا دیتاست مربوطه را دانلود و پردازش های لازم برای خواندن آن را انجام دادم. توسط تابع get_stories ابتدا خط به خط داده ورودی از فایل ذخیره‌شده در درایو خوانده شد و سپس در تابع parse_stories جدسازی جملات در هر خط صورت گرفت و داده ورودی شکل گرفت. سپس در قسمت vectorize_stories باید ورودی را برای دادن به شبکه RNN آماده سازی کنیم. در این متد سه تایی Story و Question و Answer به صورت بردار های عددی در می‌آید. سپس برای خروجی هر story و answer یک بردار از تعدادی صفر برای پیش‌بینی شبکه ایجاد می‌کنیم. سایز این بردار را به اندازه ماکسیمم سایز sequence ها که در بالای کد محاسبه شد قرار داده شده است.

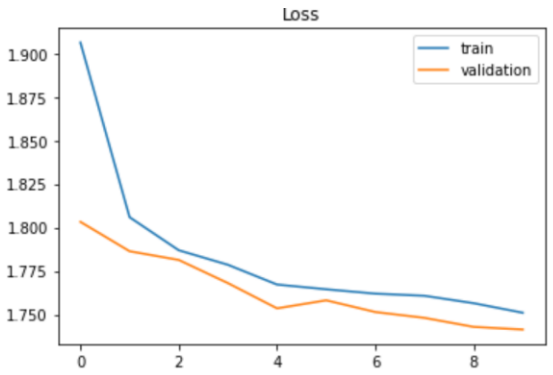
سپس در قسمت تشکیل مدل، طبق مدل ذکر شده در شکل مدل را مرحله به مرحله تشکیل می‌دهیم. در ابتدا question و story را به لایه embedding می‌دهیم و سپس در بعد از این لایه یک لایه dropout قرار می‌دهیم تا احتمال overfit را کاهش دهیم. سپس خروجی هر کدام را با هم match کرده و سپس خروجی لایه match را با embedding ورودی story جمع می‌کنیم. در نهایت خروجی این لایه را با embedding ورودی question جمع ادغام می‌کنیم و از لایه dropout و dense عبور می‌دهیم.

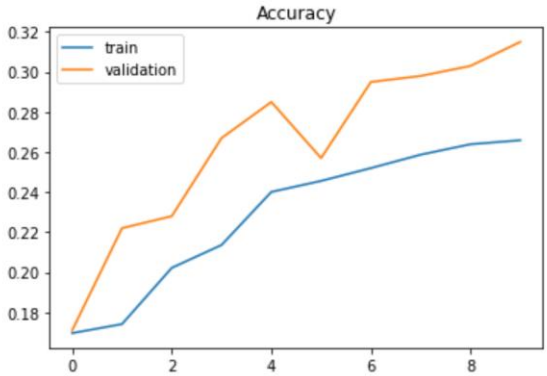
سپس مدل شکل داده را با پنجاه epoch و بچ سایز ۳۲ تایی آموزش دادم. سپس نمودار های دقت و خطای داده های آموزش و اعتبارسنجی را توسط متدهایی که بالا نوشتم رسم کردم که به شکل زیر درآمد:

```
Epoch 49/50
313/313 [==============================] - 19s 61ms/step - loss: 1.5235 - accuracy: 0.4058 - val_loss: 1.7026 - val_accuracy: 0.3630
Epoch 50/50
313/313 [==============================] - 20s 63ms/step - loss: 1.5262 - accuracy: 0.4074 - val_loss: 1.6954 - val_accuracy: 0.3560
```

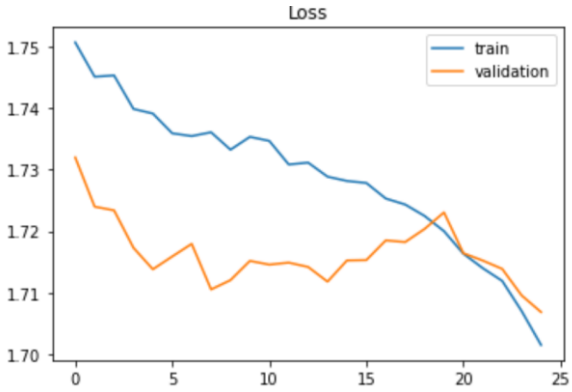


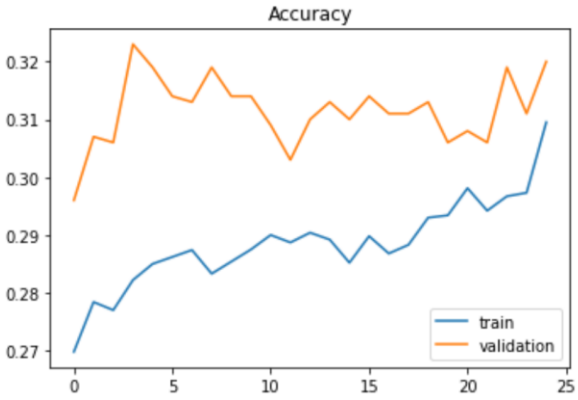
همانطور که مشخص است دقت فاز آموزش با پنجاه epoch به ۴۰رسیده است که دقت مناسبی نیست و نمی‌توان روی یادگیری خوب شبکه حساب باز کرد. به علاوه روند دقت و خطا واگرا شده است پس اورفیت شده‌است بنابراین بار دیگر آموزش شبکه را با ده epoch انجام دادم که نتایج زیر بدست آمد:





در این آزمایش روند صعودی در دقت و نزولی برای خطا در داده های آموزش و اعتبارسنجی حفظ شد پس اورفیت نداشتیم ولی دقت پایین است و در حدود ۳۲ درصد است. پس با بیست و پنج epoch به عنوان حد میانی برای بار آخر آموزش را انجام دادم.





در قسمت انتهایی نیز شبکه آموزش داده شده توسط چند جمله مورد تست قرار داده شد که نتایج آن در ذیل آمده است:

```
Sandra travelled to the kitchen . Sandra travelled to the hallway . Where is Sandra ? | Prediction: hallway
----------------------------------------------------------------------------------------------------
Please input a story
Sandra travelled to the kitchen . Sandra travelled to the hallway . Mary went to the bathroom . Sandra moved to the garden .
Please input a query
Where is Sandra ?
Result
Sandra travelled to the kitchen . Sandra travelled to the hallway . Mary went to the bathroom . Sandra moved to the garden . Where is Sandra ? | Prediction: garden
----------------------------------------------------------------------------------------------------
```

همانطور که در تصویر میبینید، شبکه سناریو هایی که کوتاه یا بلند هستند و فقط شامل یک نام هستند که به ترتیب به لوکیشن های مختلف رفته است را خوب یادگیری کرده است و پیش‌بینی صحیح انجام می‌دهد.

```
Result
John travelled to the hallway . Mary journeyed to the bathroom . Daniel went back to the bathroom . John moved to the bedroom . Where is Mary ? | Prediction: office
----------------------------------------------------------------------------------------------------
```

اما در این موارد که در ترتیب موارد گفته شده چند نام وجود دارد شبکه دچار اشتباه شده و پیش‌بینی فرد دیگر را به جای فرد موردنظر می‌دهد.

از مزایای مدل می‌توان به یادگیری خوب و تدریجی آن در فاز آموزش اشاره کرد. جملات تک فاعلی را هرچند طولانی هم باشند یادگیری می‌کند. سیستم می‌تواند به سوالات در این حوزه با دقت نسبتا خوبی پاسخ دهد اما در سایر قسمت ها نیاز به اصلاح دارد.

از معایب مدل نیز می‌توان به کاهش کم loss در آن اشاره کرد که احتمالا به علت طول زیاد داده های آموزش مشکل vanishing داشته باشیم که باعث می‌شود آپدیت شبکه در هر epoch با مشکل مواجه شود. به علاوه پیش‌بینی در جملات با اسم های مختلف مشکل دارد و

نشان می‌دهد در یادگیری این موارد روابط بین فعل و فاعل را در جمله نتوانسته کامل یادگیری کند. در این باره dependency parsing لازم است تا روابط موجود در جمله را به خوبی بتواند یادگیری کند. به علاوه در مدل اولیه از dropout در خروجی لایه های embedding استفاده نشده بود که من خودم اضافه کردم. در صورت عدم استفاده از آن overfit مدل در epoch های خیلی پایین تر رخ می‌دادکه یادگیری را با ضعف زیادی روبرو می‌کرد.

برای استفاده این شبکه در واقعیت می‌توان از لایه های LSTM استفاده کرد باعث می‌شود مشکل vanishing تا حدود خوبی مرتفع گردد. همچنین استفاده از الگوریتم های گراف برای dependency parsing می تواند جملات سنگین تر را تحلیل و یادگیری کند و پیش‌بینی درستی به ما بدهد. در دنیای واقعی نیز از این مدل در مدل های پرسش و پاسخ می‌توان استفاده کرد. پرسش و پاسخ نیز در حوزه های مختلفی کاربرد دارد. در دستیار های هوشمند تلفن‌های همراه به صورت گستره استفاده میشود یا همچنین برای حوزه های برگزاری برخی امتحان های آنلاین مانند Toefl این مدل سیستم ها مورداستفاده قرار می‌گیرد.

منبع)

https://www.kaggle.com/roblexnana/memory-network-for-qa-on-babi-tasks-97-accuracy

https://towardsdatascience.com/question-answering-for-enterprise-use-cases-70ed39b74296

https://cs224d.stanford.edu/reports/StrohMathur.pdf

سوال چهارم)

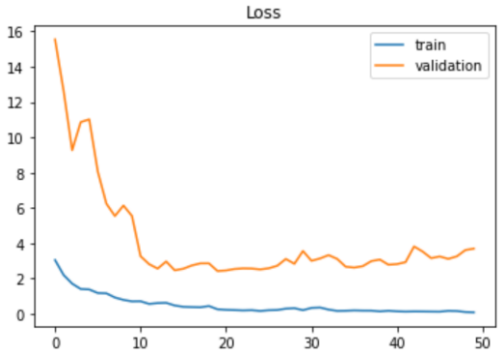
الف) ابتدا داده‌ها را از دیتاست CIFAR دریافت می‌کنیم و بخش های آموزش و تست آنرا تشکیل می‌دهیم.

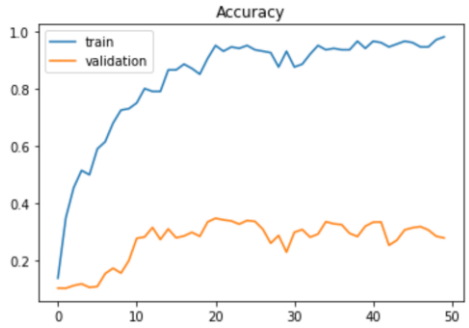
سپس مدل Sequential را مطابق کد زیر از لایه های مختلف تشکیل میدهیم:

```
layers.Input(shape=input_shape),
layers.Conv2D(32, (3, 3), padding="same"),
layers.Activation("relu"),
layers.BatchNormalization(axis=-1),
layers.MaxPooling2D(pool_size=(3, 3)),
layers.Dropout(0.25),

layers.Conv2D(64, (3, 3), padding="same"),
layers.Activation("relu"),
layers.BatchNormalization(axis=-1),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Dropout(0.25),

layers.Conv2D(128, (3, 3), padding="same"),
layers.Activation("relu"),
layers.BatchNormalization(axis=-1),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Dropout(0.25),

layers.Flatten(),
layers.Dense(1024),
layers.Activation("relu"),
layers.BatchNormalization(),
layers.Dropout(0.2),
layers.Dense(NUM_CLASSES),
layers.Activation("softmax"),
```

سپس مدل را با اپتیمایز Adam و تابع خطای cross_entropy آموزش دادم و نتایج زیر بدست آمد:

```
Epoch 44/50
7/7 [==============================] - 3s 441ms/step - loss: 0.1371 - accuracy: 0.9550 - val_loss: 3.5333 - val_accuracy: 0.2721
Epoch 45/50
7/7 [==============================] - 3s 442ms/step - loss: 0.1316 - accuracy: 0.9650 - val_loss: 3.1583 - val_accuracy: 0.3081
Epoch 46/50
7/7 [==============================] - 2s 299ms/step - loss: 0.1266 - accuracy: 0.9600 - val_loss: 3.2547 - val_accuracy: 0.3155
Epoch 47/50
7/7 [==============================] - 2s 295ms/step - loss: 0.1747 - accuracy: 0.9450 - val_loss: 3.1183 - val_accuracy: 0.3194
Epoch 48/50
7/7 [==============================] - 2s 301ms/step - loss: 0.1636 - accuracy: 0.9450 - val_loss: 3.2600 - val_accuracy: 0.3077
Epoch 49/50
7/7 [==============================] - 3s 441ms/step - loss: 0.1069 - accuracy: 0.9700 - val_loss: 3.6174 - val_accuracy: 0.2858
Epoch 50/50
7/7 [==============================] - 2s 297ms/step - loss: 0.0881 - accuracy: 0.9800 - val_loss: 3.7009 - val_accuracy: 0.2799
```





در انتهای شبکه دو لایه Dense قرارا دادم تا دسته بندی را بتوان انجام داد.

همانطور که مشاهده میشود در فاز آموزش دقت خوبی را شاهد بودیم اما در فاز اعتبارسنجی و تست دقت خوبی را شاهد نبودیم پس دچار اورفیت شدیم و علت آن کمبود داده ها و حفظ الگو های موجود در داده ها توسط شبکه است.

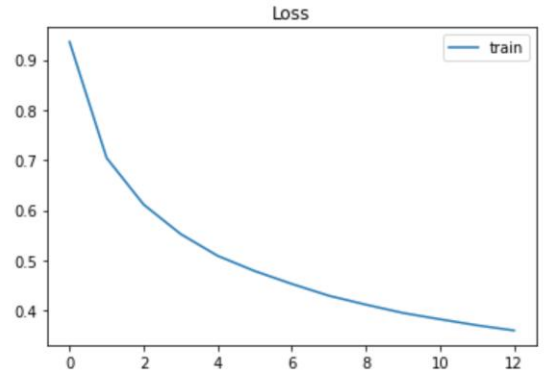
ب) در مرحله برای انجام تسک ابتدا لازم است داده موردنیاز را فراهم کنیم. برای اینکار تصویر را با زوایای 90، 180، 270 و 360 دوران داده و به مجموعه داده اضافه می‌کنیم.

مانند مدل قبلی شبکه را با تعدادی از لایه های کانولوشن و سایر لایه‌ها تشکیل داده و دولایه Dense در آخر برای دسته‌بندی قرار میدهیم.

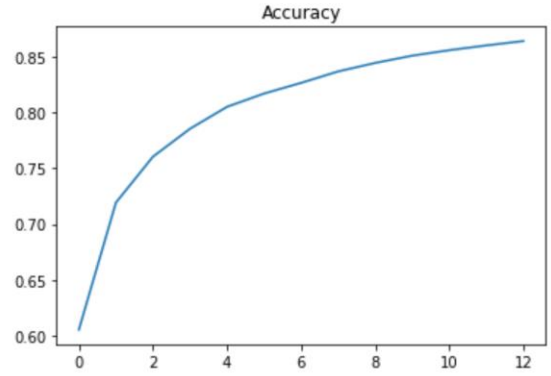
از آنجا که دسته‌بندی زوایا همانطور که بالاتر اشاره شد شامل ۴ کلاس ۹۰ و ۱۸۰ و ۲۷۰ و ۳۶۰ است پس در لایه آخر باید ۴ نورون برای دسته‌بندی قرار دهیم.

سپس مدل را در سیزده epoch آموزش داده و نتایج زیر را کسب می‌کنیم:

```
Epoch 11/13
1557/1557 [==============================] - 52s 34ms/step - loss: 0.3825 - accuracy: 0.8562
Epoch 12/13
1557/1557 [==============================] - 52s 33ms/step - loss: 0.3705 - accuracy: 0.8605
Epoch 13/13
1557/1557 [==============================] - 52s 33ms/step - loss: 0.3602 - accuracy: 0.8644
```

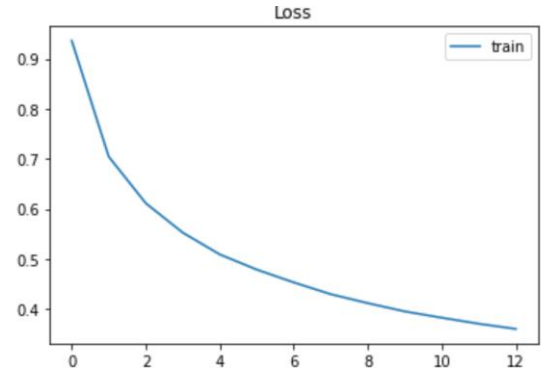


<Figure size 432x288 with 0 Axes>

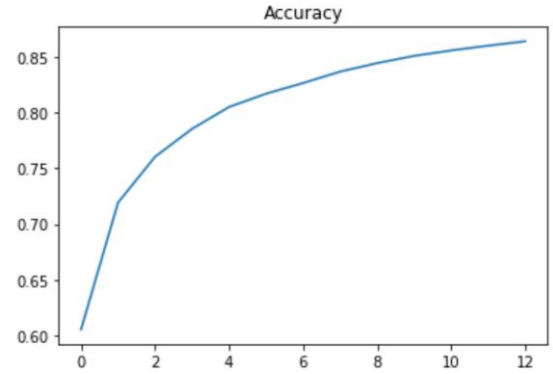


اما این دقت معیار ما نخواهد بود. زیرا باید از نتایج و الگو های بدست آمده در این قسمت برای دسته‌بندی قسمت دیگری استفاده کنیم. لایه آخر را به ۱۰ نورون تبدیل می‌کنیم تا توانایی دسته‌بندی ۱۰ کلاس را داشته باشیم. حال مجدد شبکه را در بیست epoch آموزش دادیم و نتایج زیر را کسب کردیم:

```
782/782 [==============================] - 18s 22ms/step - loss: 0.0396 - accuracy: 0.9269
Epoch 27/30
782/782 [==============================] - 18s 22ms/step - loss: 0.0374 - accuracy: 0.9325
Epoch 28/30
782/782 [==============================] - 18s 22ms/step - loss: 0.0381 - accuracy: 0.9307
Epoch 29/30
782/782 [==============================] - 17s 22ms/step - loss: 0.0362 - accuracy: 0.9348
Epoch 30/30
782/782 [==============================] - 18s 22ms/step - loss: 0.0357 - accuracy: 0.9367
<keras.callbacks.History at 0x7f66f649a7d0>
```



<Figure size 432x288 with 0 Axes>



دقت خوبی در این قسمت نیز کسب شد که نشان از موفقیت این قسمت دارد.

پ) در قسمت آخر نیز در ابتدا داده های یاد شده را طبق خواسته سوال و مورد قبل تشکیل می‌دهیم. سپس دولایه خروجی مطابق لینک داده شده برای شبکه می‌سازیم. یکی از خروجی ها شامل ۱۰ نورون(۱۰ کلاسه) و یکی شامل ۴ نورون (۴ کلاسه) است.

سپس وزن های مختلفی را روی تابع خطای دو تابع امتحان کردم و عملکرد مدل را بهینه کردم.

در این قسمت خروجی مدل من پاک شد و پس از آن دائما ارور پر شدن رم را میداد اما خروجی را در قسمت زیر می‌آورم و شما میتوانید با ران کردن کد در دیوایس خود آنرا تست نمایید.

Your session crashed after using all available RAM. View runtime logs ×

دقت برای خروجی اول ۹۸ درصد و برای خروجی دوم ۳۰ درصد بود.