به نام خدا

تمرین سری یازدهم درس یادگیری عمیق

دکتر محمدی

محمد یارمقدم

۹۶۴۶۲۱۰۴

سوال اول)

برای حل این سوال به ترتیب از اولین لایه آغاز و پیش می رویم.

- اولین لایه کانولوشن است. این لایه ۶۴ فیلتر با سایز ۷*۷ دارد. تعداد پارامتر های این لایه طبق رابطه زیر بدست می‌آید:

# of parameters = ((7*7*1) + 1) * 64 = 3200

زیرا همه فیلتر ها برای لایه ها یکسان هستند پس ۴۹ پارامتر برای هر فیلتر موجود است که با بایاس ۵۰ عدد می شود.

خروجی نیز (24, 24, 1) می شود. زیرا باید ابعاد فیلتر را از ابعاد عکس ورودی کم کنیم. پس:

Output_size = in_size – filter_size = (30, 30) – ((7, 7) – (1, 1)) = (24, 24)

حال چون ۶۴ فیلتر داریم پس خروجی نهایی (24, 24, 60) می شود.

- لایه دوم maxpooling ۲ بعدی است. این لایه پارامتر قابل یادگیری ندارد. چون گام این لایه ۲*۲ است پس سایز ورودی را نصف می کند. پس سایز خروجی (12, 12, 64) می شود.
- لایه سوم کانولوشن است. این لایه ۱۲۸ فیلتر با سایز ۵*۵ دارد. به علاوه ۶۴ عمق دارد. پس ۲۵*۶۴ پارامتر دارد. یک بایاس نیز دارد. پس:

# of parameters = ((64*25) + 1) * 128 = 204928

مانند مورد قبل نیز سایز خروجی را به صورت زیر بدست می آوریم:

Output_size = in_size – filter_size = (12, 12) – ((5, 5) – (1, 1)) = (8, 8) → final = (8, 8, 128)

- لایه چهارم نیز maxpooling ۲ بعدی است. مانند قبل داریم:

Output_size = (8/2, 8/2, 128) = (4, 4, 128)

- لایه پنجم کانولوشن است. این لایه ۲۵۶ فیلتر با سایز ۳*۳ دارد. به علاوه ۱۲۸ عمق دارد. پس ۱۲۸*۹ پارامتر دارد. یک بایاس نیز دارد. پس:

# of parameters = ((128*9) + 1) * 256 = 295168

مانند مورد قبل نیز سایز خروجی را به صورت زیر بدست می آوریم:

Output_size = in_size – filter_size = (4, 4) – ((3, 3) – (1, 1)) = (2, 2) → final = (2, 2, 256)

- لایه ششم نیز maxpooling ۲ بعدی است. مانند قبل داریم:

Output_size = (2/2, 2/2, 256) = (1, 1, 256)

- لایه هفتم flatten است. این لایه ورودی را تبدیل به یک بعد می کند پس خروجی (256) است و پارامتر ندارد.
- لایه هشتم Dense است. این لایه به تمام لایه قبل متصل است. پس:

# of parameters = (256 + 1) * 128 = 32896

خروجی نیز تعداد unit است پس (128) می شود.

- لایه های نهم Embedding ها هستند.

در هر لایه داریم که این لایه قابلیت پشتیبانی از ۱۰۰۰ کلمه منحصر به فرد را دارد و به ازای هر کلمه یک بردار ۲۵۶ تایی در نظر خواهد می گیرد. به علاوه ۲۵۶ ورودی و ۱۰۰۰ نورون لایه مخفی و ۲۵۶ خروجی دارد. پس:

# of parameters = (1000+1) * 256 + (256+1) * 1000 = 256256 + 257000 = 513256

Output_size = (256)

- لایه های دهم RNN ها هستند. در هر کدام از آنها داریم:

# of parameters = # of FFNN * (# of hidden neurons + input_size) + # of hidden neurons)
= 1 * (128 * (128+256)+128) = 49280
Output_size = (128)

- لایه یازدهم لایه کاملا متصل است که به RNN ها متصل هستند.

# of parameters = (128 + 1) * 128 = 16512
Output_size = (128)

- لایه دوازدهم concatenate است که خروجی دو لایه Dense قبلی را با هم ترکیب می کند. در نتیجه (256) سایز خروجی میشود. پارامتر هم ندارد.
- لایه سیزدهم مجدد کاملا متصل است.

# of parameters = (256 + 1) * 128 = 32896
Output_size = (128)

ورودی لایه قبلی که (256) است را به تعداد unit که (128) تبدیل می کند.

- لایه چهاردهم که آخرین لایه نیز هست هم کاملا متصل است. پس داریم:

# of parameters = (128 + 1) * 1000 = 129000
Output_size = (1000)

خروجی با توجه به embedding کلمات این سایز می شود.

لینک های کمکی)

https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce

https://towardsdatascience.com/a-guide-to-word-embeddings-8a23817ab60f

https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889#192e

session_15 and session_13 and session_20 slides of course

سوال دوم)

شرح کد:

در این کد ابتدا مجموعه داده reuters را ازسایت ntlk دانلود و بر روی drive خود آپلود می کنیم. سپس با استفاده از تابع fields داده های با category خام یا crudes را جدا می کنیم و ۱۰۰ تا عضو اول آن را در حلقه جدا می کنیم. با استفاده از کد موجود در داک تمرین کلمات را در هر نمونه به هم چسباندیم تا جملات ساخته شود و آن ها را در لیست sentences ذخیره کردم.

حال برای ساخت شبکه برای آموزش word_embedding با طول ۲ طبق لینک داده شده داریم:

ابتدا لایه ورودی قرار دارد که ۲۸۲۳ نورون دارد. سپس لایه کاملا متصل قرار دارد که فعالساز آن linear است. در آخر یک لایه متصل دیگر با فعالساز softmax قرار دارد.

الف)

در این سوال پیش پردازش های مختلفی توسط تابع text_preprocessing انجام شده است. این موارد به ترتیب در زیر آمده است:

ابتدا در حلقه اولیه punctuation ها با رشته خالی جایگزین شده اند. سپس کلماتی که در آنها رقم وجود داشته باشد، حذف شده‌اند. سپس ارقام حذف شدند. در مرحله بعد تعداد اسپیس های خالی اضافی بهینه شدند. بعد تمامی کلمات به صورت lower در آمدند. در نهایت همه کلمات text در یک لیست ذخیره شده و stop_word ها از میان آنها حذف شدند.

Stop_word ها در واقع کلماتی هستند که به خودی خود معنا ندارند و در جمله معنی خواهند داد. کار آنها جداسازی و اتصال جملات است. به همین دلیل این نام را دارند.

سپس به اندازه پنجره، فاصله ای به همان میزان از کلمات جدا و آن ها را در لیست های جدا ذخیره می کند. در آخر با استفاده از تابع create_unique_word_dict موارد تکراری را حذف می کند.

ب)

کلمات هر خبر را با توجه به اندازه‌ی پنجره، دوتا دوتا در یک آرایه از کلمات دوتایی که word_lists نام دارد، قرار می‌دهیم. سپس مجموعه هایی که دو بار تکرار دارند را از این لیست به کمک تابع creat_unique_word_dict حذف می‌کند تا کلمات یکتا باشند و یک دیکشنری از کلمات می‌سازد.

سپس یک ماتریس با ابعاد num_words*num_words در نظر می‌گیریم و برای هر لیست از کلمات، در مختصات x کلمه‌ی اول و y کلمه‌ی دوم عدد ۱ را می‌گذارد تا نشان دهنده‌ی آن باشد که این دو کلمه با هم در ارتباط هستند. سپس بردار x ها را به عنوان ورودی شبکه و بردار y ها را به عنوان خروجی مورد انتظار در نظر می گیریم و دو xها و y ها را از هم جدا می‌کنیم. چون با اینکار، یک ماتریس بسیار بزرگ به وجود می‌آید که بسیاری از عناصر آن ۰ هستند، به کمک تابع sparse.csr_matrix ماتریس را متراکم می‌کند. در واقع ۲۸۲۳ نورون را با ۵۴۵۵۶ داده ورودی می دهیم.

ج)

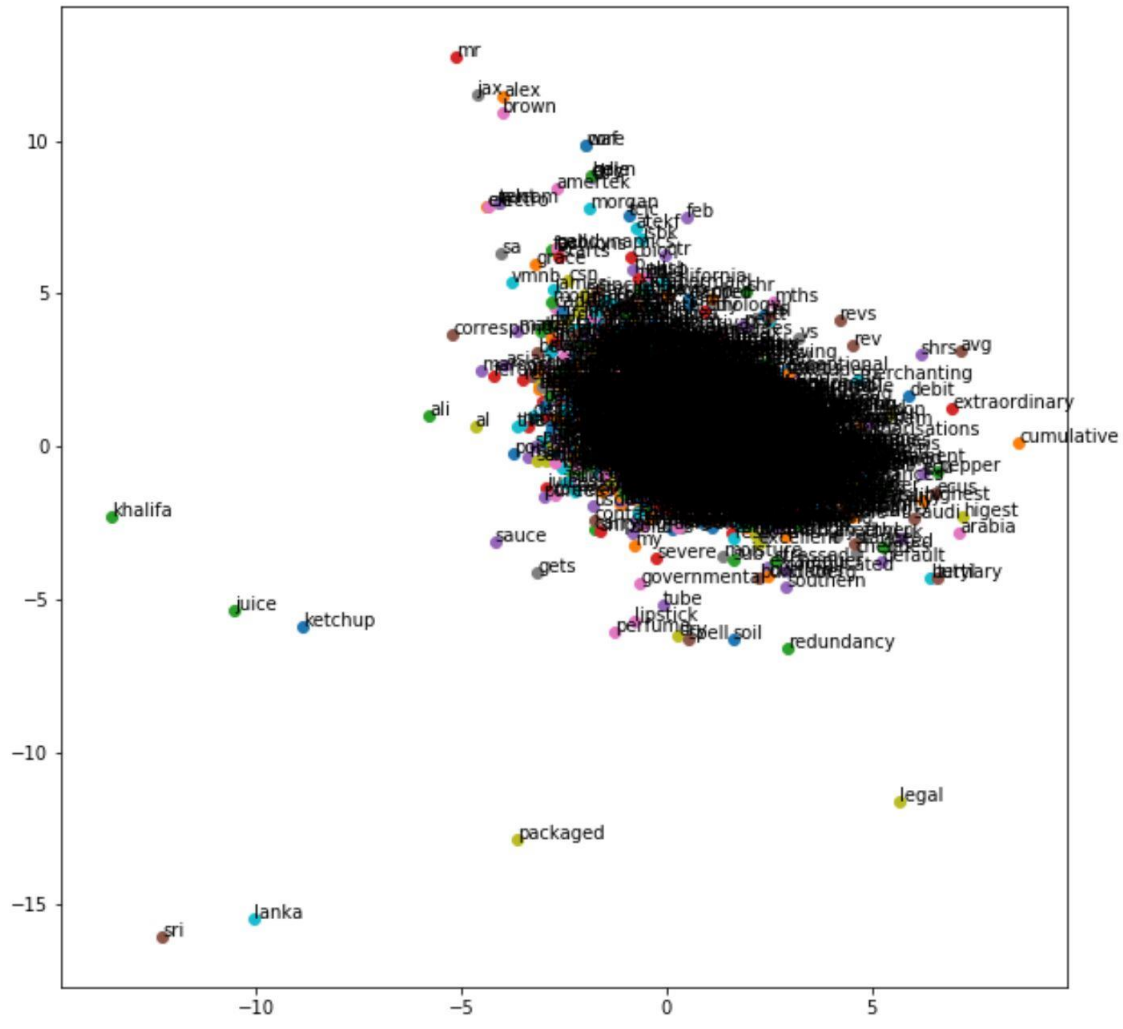
با افزایش سایز پنجره، تعداد امکان مقایسه برای هر کلمه بیشتر خواهد شد و تعداد زوج داده ما نیز بیشتر خواهند شد. در نتیجه ارتباط کلمات دقیق تر و با جزییات بهتر آموزش دیده می شود. اما زمان ران شدن شبکه نیز طولانی تر خواهد شد. سایز کمتر نیز دقیقا بالعکس حالت قبل است. به این معنی که دقت و آموزش ضعیف تر خواهد بود.

د)

به هریک از 2823 نورون ورودی دو نورون میانی متصل است. پس برای هر ورودی ما دو وزن خواهیم داشت. ورن نخست مربوط به x و وزن دوم مربوط به y خواهد بود.

تحلیل شبکه:

پس از آموزش شبکه، شکب روابط کلمات در دیتاست به صورت زیر در می آید:



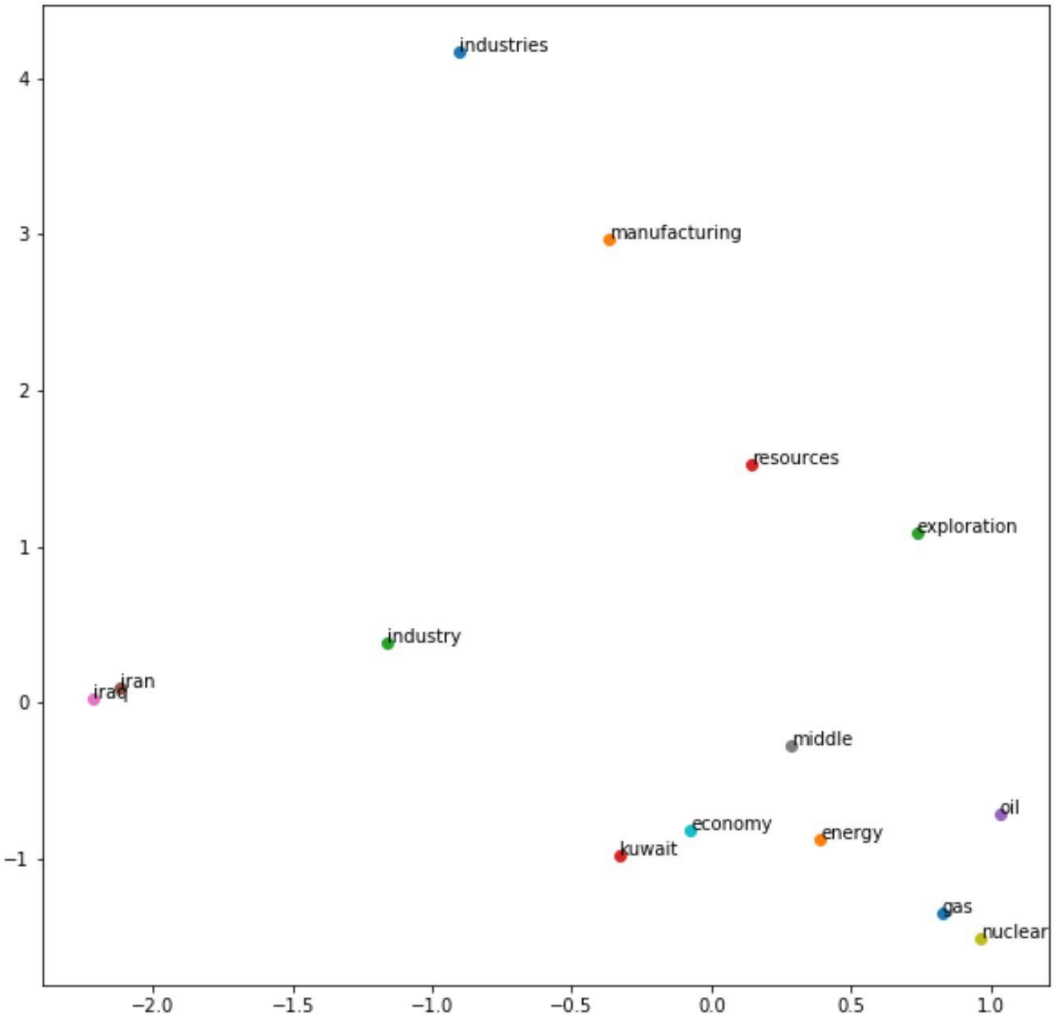
نتایج نهایی شبکه پس از ۱۰۰۰ epoch نیز به صورت مقابل است:

```
Epoch 1000/1000
      52920/52920 [==============================] - 6s 111us/step - loss:
                                                                  6.2440
```

در اینجا همچنان میزان خطا بالاست و تا بهینگی فاصله هست اما یک سری دسته بندی ها نیز شکل گرفته است.

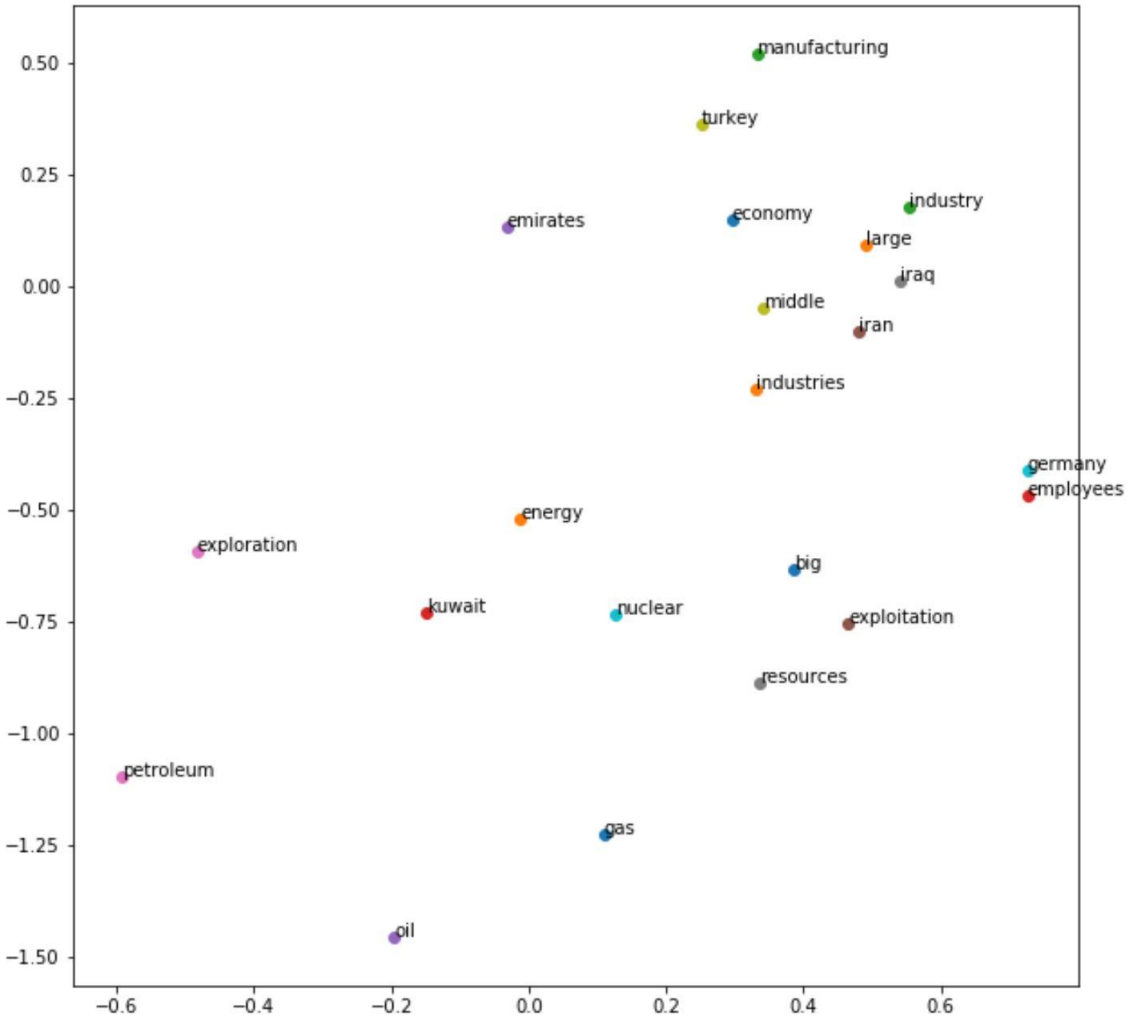
سپس کلمات خواسته شده را در یک لیست ذخیره و به شبکه می دهیم. Iran و Iraq به طور مثال به عنوان کشور تشخیص داده شده اند و در کنار هم هستند ولی مثلا Kuwait هنوز خطا دارد.

کلمات economy و energy و oil و gas و nuclear نیز در دسته انرژی و سوخت تشخیص داده شده و کنار هم آمده اند.



برای قسمت آخر مجبور شدم شبکه را دوباره ران کنم چون که ران تایم ریست شده بود. و اینبار با تعداد کمر ران کردم تا خروجی این قسمت هم نمایش دهم.

کلمات petrol، turkey، massive، big، large و Germany را اضافه کردم و نتیجه به صورت زیر در آمد:



کلمات big و large و massive تقریبا در نزدیکی هم قرار گرفتند و Germany و turkey نیز به سایر کشور ها نزدیک هستند.

لینک های کمکی:

Python Examples of nltk.corpus.reuters.fileids (programcreek.com)

https://machinelearningmastery.com/what-are-word-embeddings/

Session_21 slide of course

سوال سوم)

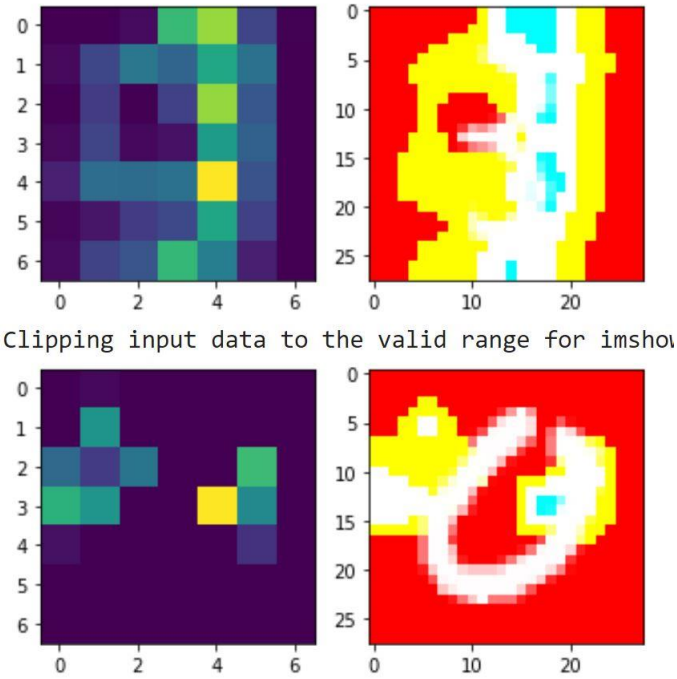
در این تمرین مشابه تمرین سری ششم، ابتدا مجموع داده MNIST که شامل 60K داده آموزشی و 10K داده تست است، را اضافه می کنیم. سپس دیتا را به هم میریزیم و با تابع reshape یک بعد به آن اضافه می کنیم تا بتوان عکس های این دیتاست را به عنوان ورودی لایه conv استفاده کنیم. سپس داده های را نرمالایز کرده و آن ها را به شکل one hot در می آوریم.

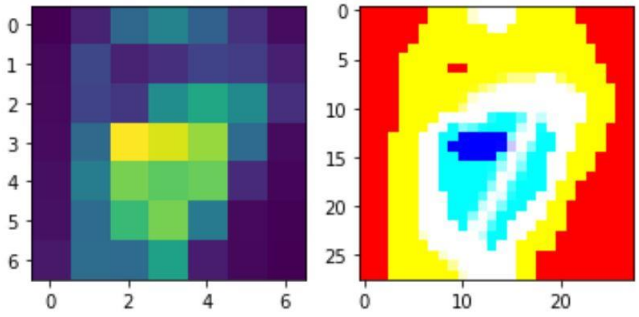
سپس شبکه را با مقادیر خواسته شده تشکیل می دهیم و در پانزده epoch و با batch_size 64 تایی آموزش می دهیم.

در آخر در قسمت grad_cam نیز نام آخرین لایه کانولوشن را از قسمت summary بدست می آوریم و از آن لایه را توسط تابع get_layer بدست می آوریم. سپس تعدادی از index های داده های آموزشی را جدا کرده و سپس عملیان نمایش heat map را آغاز می کنیم. در این بخش ابتدا برای بدست آوردن کلاس خروجی هر تصویر مورد نظر ابتدا تابع predict را روی هر کدام فراخوانی می کنیم.

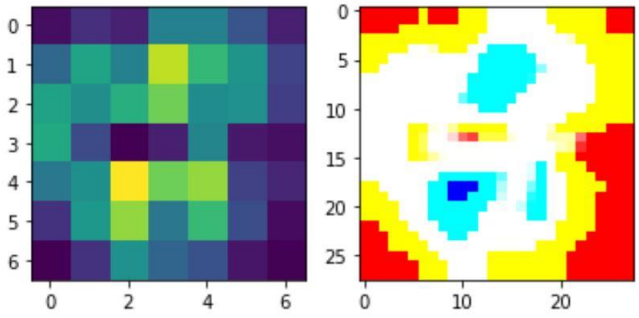
سپس یک آرایه از ایندکس کلاس های خروجی هر عکس تشکیل می دهیم. در نهایت طبق مراحل الگوریتم اسلاید پیش می رویم و مراحل محاسبه گرادیان و محاسبه خروجی لایه گرادیان را انجام می دهیم. در نهایت از خروجی لایه conv هیت مپ را تشکیل می دهیم و سپس عکس اصلی را با تاثیر هیپ مپ روی هم انداخته و نمایش می دهیم.

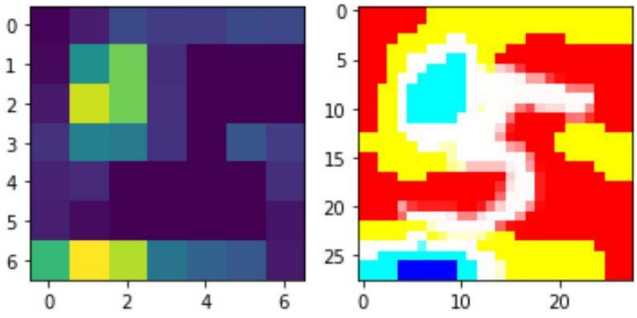
خروجی های این سوال را در قسمت زیر مشاهده می کنید:



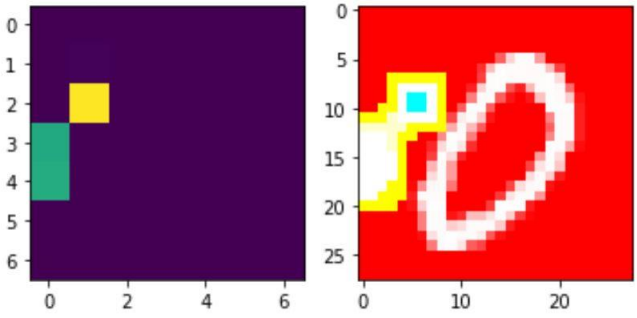


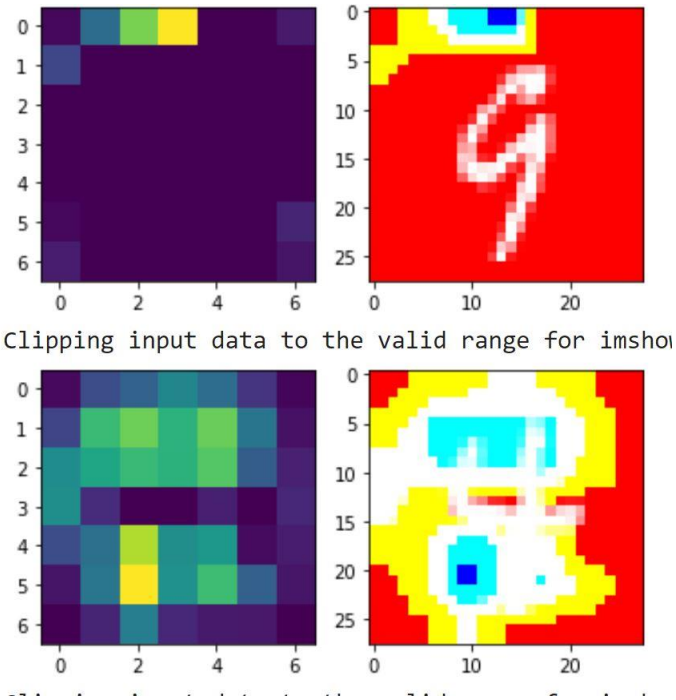
Clipping input data to the valid range for imshow





Clipping input data to the valid range for imsho





تحلیل خروجی ها: همانطور که مشاهده می کنید، در خروجی های سمت چپ هیپ مپ مربوط به هر عکس ورودی مشخص شده است که در واقع خروجی لایه conv است. سپس در خروجی های قسمت راست تاثیر این هیت مپ را روی عکس اصلی مشاهده می کنید.

برای تعدادی از این عکس های شبکه نتوانسته تشخیص خوبی را داشته باشد. به طور مثال عکس ۹ و یا ۰ را خوب تشخیص نداده اما ۳ و یا ۵ رو نماد های خوبی برای تشخیص و جداسازی آن ها پیدا کرده است.

لینک های کمکی:

https://www.machinecurve.com/index.php/2019/11/28/visualizing-keras-cnn-attention-grad-cam-class-activation-maps/

https://keras.io/examples/vision/grad_cam/

https://github.com/jacobgil/keras-cam/blob/master/cam.py

https://stackoverflow.com/questions/66221788/tf-gradients-is-not-supported-when-eager-execution-is-enabled-use-tf-gradientta

session_20 slide of course