دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین سری یازدهم

علی صداقی

۹۷۵۲۱۳۷۸

۱ سوال اول

روابط استفاده شده در حل سوال (با فرض برابر بودن پارامترها در Height و Width)

$Input\ shape: (h, w, c)$

$Conv2D: f, (k, k), (s, s), (p, p)$

$Output\ shape: (\left\lfloor \frac{h - k + 2p}{s} + 1 \right\rfloor, \left\lfloor \frac{w - k + 2p}{s} + 1 \right\rfloor, f)$

$\#\ parameters: (k \times k \times c + 1) \times f$

$MaxPool2D: pool\ size: k, stride: s, padding: p$

$Output\ shape: (\left\lfloor \frac{h - k + 2p}{s} + 1 \right\rfloor, \left\lfloor \frac{w - k + 2p}{s} + 1 \right\rfloor, c)$

$\#\ parameters: 0$

$Dense: Input\ shape: (v)$

$Output\ shape: u = number\ of\ units$

$\#\ parameters: u \times v + u = u \times (v + 1)$

$RNN: v = number\ of\ input\ features, u = number\ of\ units, return\_sequence = False$

$Output\ shape: u = number\ of\ units$

$\#\ parameters: = u^2 + u(v + 1) = u^2 + uv + 1$

$Embedding: v = number\ of\ vocabs, u = number\ of\ outputs, i = input\ size$

$Output\ shape: (i, u)$

$\#\ parameters: = uv$

$Flatten: input\ shape(x_1, x_2, \dots, x_n)$

$Output\ shape: (x_1 \times x_2 \times \dots \times x_n)$

$\#\ parameters: = 0$

$Concatenate: input\ shape(x_1, x_2, \dots, x_n)$

$Output\ shape: (x_1 + x_2 + \cdots + x_n)$

$\#\ parameters: = 0$

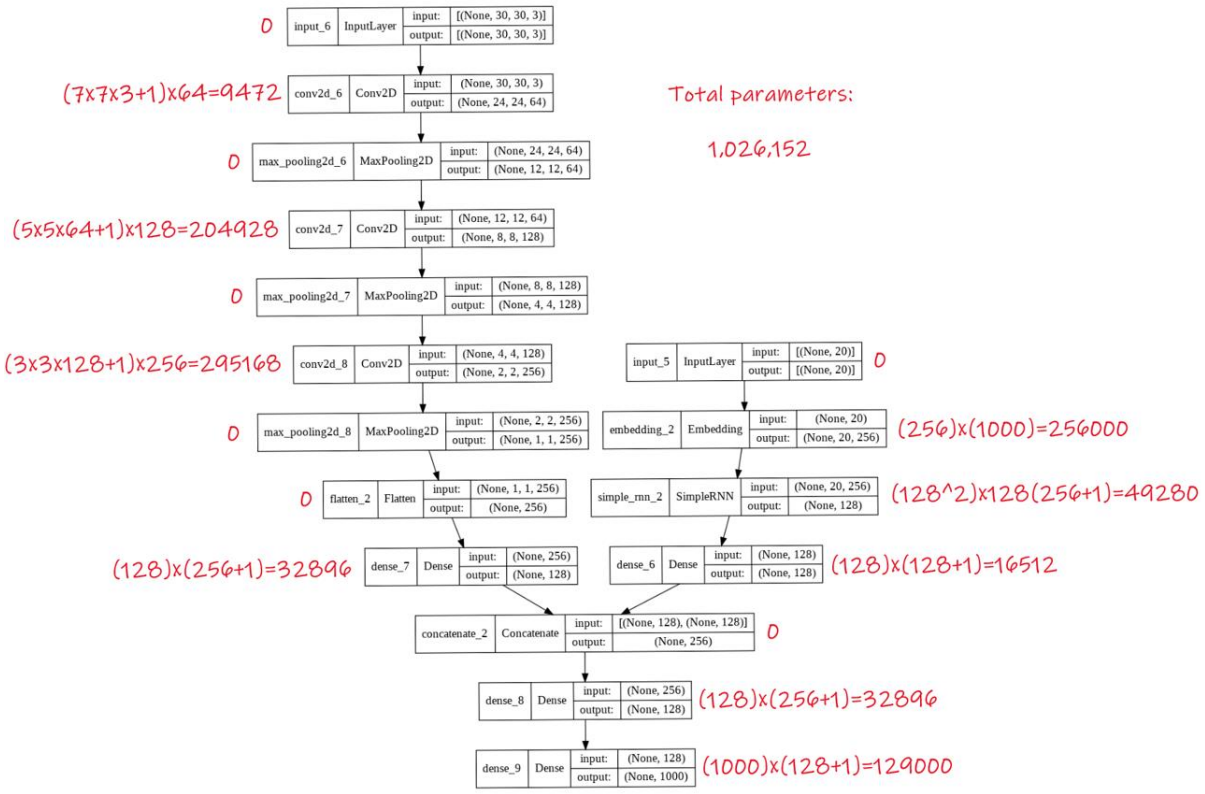
لایه Dense آخر که دارای 1000 نورون است یک کلمه از میان 1000 کلمه را انتخاب می‌کند (تابع فعالسازی Softmax). در حل این سوال فرض کردیم که برای هر تصویر دقیقا 20 کلمه تولید خواهد شد. برخی کلمات می‌توانند رشته خالی باشند. (بر اساس انتخاب مدل)

کد مدل در نوتبوک سوال ۱ موجود است. پس از طراحی مدل با تابع summary صحت محاسبات خود را چک می‌کنیم.

Layer (type)	Output Shape	Param #	Connected to
input_8 (InputLayer)	[(None, 30, 30, 3)]	0	[]
conv2d_9 (Conv2D)	(None, 24, 24, 64)	9472	['input_8[0][0]']
max_pooling2d_9 (MaxPooling2D)	(None, 12, 12, 64)	0	['conv2d_9[0][0]']
conv2d_10 (Conv2D)	(None, 8, 8, 128)	204928	['max_pooling2d_9[0][0]']
max_pooling2d_10 (MaxPooling2D)	(None, 4, 4, 128)	0	['conv2d_10[0][0]']
input_7 (InputLayer)	[(None, 20)]	0	[]
conv2d_11 (Conv2D)	(None, 2, 2, 256)	295168	['max_pooling2d_10[0][0]']
embedding_3 (Embedding)	(None, 20, 256)	256000	['input_7[0][0]']
max_pooling2d_11 (MaxPooling2D)	(None, 1, 1, 256)	0	['conv2d_11[0][0]']
simple_rnn_3 (SimpleRNN)	(None, 128)	49280	['embedding_3[0][0]']
flatten_3 (Flatten)	(None, 256)	0	['max_pooling2d_11[0][0]']
dense_10 (Dense)	(None, 128)	16512	['simple_rnn_3[0][0]']
dense_11 (Dense)	(None, 128)	32896	['flatten_3[0][0]']
concatenate_3 (Concatenate)	(None, 256)	0	['dense_10[0][0]', 'dense_11[0][0]']
dense_12 (Dense)	(None, 128)	32896	['concatenate_3[0][0]']
dense_13 (Dense)	(None, 1000)	129000	['dense_12[0][0]']

Total params: 1,026,152
Trainable params: 1,026,152
Non-trainable params: 0

از تابع plot_model برای ترسیم شکل مدل و بیان ورودی و خروجی هر لایه استفاده کردیم.



تصویر با کیفیت درون فایل Zip موجود است.

۲ سوال دوم

ابتدا با استفاده از تابع get_corpus_text اخبار مربوط به دسته Crude را بارگذاری می‌کنیم. در دیتاست Reuters درون NLTK کلمات داخل یک متن با File ID تعریف شده اند و بایستی بررسی کنیم که هر ID مربوط به چه کلمه‌ای است. برای این کار از تابع reuters.words(file_id) استفاده می‌کنیم. دسته اخبار Crude شامل 578 خبر است که ما تنها 100 خبر ابتدایی را برای ادامه کار انتخاب می‌کنیم.

```
Crude category includes 578 news.
Crude category first news text is:
JAPAN TO REVISE LONG - TERM ENERGY DEMAND DOWNWARDS The Ministry of
International Trade and Industry ( MITI ) will ...
```

پیش‌پردازش: با استفاده از تابع text_preprocessing متن هر خبر را ورودی می‌گیریم و یک لیست از کلمات خبر را خروجی می‌دهیم. در این تابع پیش‌پردازش‌های زیر را انجام می‌دهیم:

- ابتدا تمامی punctuationsهای موجود در متن که شامل r'''!()-[]{};:'"\,<>./?@#$%^&*_“~''' می‌باشد را حذف می‌کنیم.
- تمامی کلماتی که در بین خود اعداد دارند را حذف می‌کنیم.
- تمامی رقم‌ها را حذف می‌کنیم.
- White spaceهای اضافه را حذف می‌کنیم.
- رشته‌های خالی را حذف می‌کنیم.
- Stop wordها را حذف می‌کنیم.

مفهوم Stop words: در هر زبانی کلماتی مانند کلمات معرفه، ضمیر، قید، حروف ربط و ... وجود دارد که معمولا بیشترین تکرار را در آن زبان دارند. این کلمات اطلاعات زیادی را به متن نمی‌افزایند و بار معنایی کمی دارند. در NLP این کلمات را از متن حذف می‌کنیم تا تمرکز بیشتری روی کلمات مهم‌تر داشته باشیم. همچنین با انجام این کار حجم دیتاست نیز کمتر می‌شود و زمان کمتری برای آموزش الگوریتم صرف می‌شود.چند Stop words که در این کد حذف شده اند شامل and, a, is, the, in, be, will است.

تولید داده‌های آموزشی: برای تولید دیتاست پارامتری به نام window (پنجره) را تعریف می‌کنیم و مقدار آن را برابر 2 در نظر می‌گیریم. وظیفه این پارامتر تولید زوج‌های (Focus word, Context word) است. اگر مقدار پنجره را w

فرض کنیم برای ساختن این زوج به w کلمه قبل و w کلمه بعد از کلمه Focus توجه می‌کنیم. برای مثال برای جمله زیر تمامی زوج‌ها را به ازای کلمات Focus مختلف می‌سازیم.

```
The future king is the prince
```

```
(The, future), (The, king),
(future, the), (future, king), (future, is)
(king, the), (king, future), (king, is), (king, the)
(is, future), (is, king), (is, the), (is, prince),
(the, king), (the, is), (the, prince)
(prince, is), (prince, the)
```

همانطور که مشاهده کردید از 6 کلمه 18 زوج (x, y) ساختیم.

کاری که در بالا به ازای یک جمله انجام دادیم را بر روی تمامی خبرها انجام می‌دهیم و زوج‌های تولید شده را درون لیستی به نام word_list ذخیره می‌کنیم. همچنین تمامی کلمات موجود در تمامی اخبار را درون لیستی به نام all_text ذخیره می‌کنیم. بخشی از نتایج این بخش به صورت زیر است:

```
Some of Focus, Context pairs:
[['japan', 'to'], ['japan', 'revise'], ['to', 'revise'], ['to',
'japan'], ['to', 'long']]
All news together includes 14254 words
```

با داشتن لیستی از کلمات تمامی اخبار که شامل 14254 کلمه است، یک دیکشنری از کلمات منحصر به فرد تولید می‌کنیم که کلید آن برابر کلمه و ارزش آن برابر ایندکس کلمه بر اساس حروف الفبا است. ساختار این دیکشنری برای یک دیتاست دیگر به صورت زیر است:

```
unique_word_dict = {
'beautiful': 0,
'boy': 1,
'can': 2,
'children': 3,
'daughter': 4,
'family': 5,
...
}
```

اکنون ما 2728 کلمه منحصر به فرد داریم. که هر کلمه به عددی بین 0 تا 2727 نگاشت شده است. این نگاشت عددی برای آموزش در شبکه مناسب نیست زیر هر کلمه ارزشی برای شبکه دارد و مثلا کلمه 100م نسبت به کلمه 1م صد برابر ارزش دارد. برای حل این مشکل باید از نمایش One Hot استفاده کنیم.

برای انجام این کار، اگر N کلمه داشته باشیم به ازای هر کلمه یک بردار Nتایی از 0 می‌سازیم و فقط خانه مربوط به ایندکس کلمه را 1 می‌کنیم. برای مثال:

```
a = ['blue', 'sky', 'blue', 'car']
```

```
'blue' = [1, 0, 0]
'car' = [0, 1, 0]
'sky' = [0, 0, 1]
```

```
A =
[
1, 0, 0
0, 0, 1
1, 0, 0
0, 1, 0
]
```

اکنون با زوج‌های (x, y) که درون لیست word_list داریم و نمایش One Hot آن‌ها دو ماتریس X و Y را می‌سازیم. ماتریس X شامل Focus Wordها در نمایش One Hot و ماتریس Y شامل Context Wordها در همین نمایش. شکل هر دو ماتریس X و Y به صورت n*m خواهد بود که n برابر تعداد زوج‌های (Focus, Context) و m برابر تعداد کلمات منحصر به فرد است.

```
X shape: (56416, 2728)
Y shape: (56416, 2728)
X includes focus words
Y includes context words
So we have (x, y) == (focus, context) pairs
```

مزایا و معایب کاهش پارامتر Window: پارامتر Window size of the context تعداد کلمات قبل و بعد از کلمه Focus را مشخص می‌کند که با آن زوج‌های (Focus word, Context word) را می‌سازیم. کمینه مقداری که این پارامتر می‌تواند داشته باشد 1 است که به یک کلمه قبل و یک کلمه بعد نگاه می‌کند.

اگر این پارامتر را زیاد کنیم تعداد زوج‌ها زیاد می‌شود و حجم دیتاست بیشتر می‌شود و زمان آموزش نیز افزایش می‌یابد (عیب) اما ممکن است روابط معنایی بیشتری میان کلمات استخراج شود و دقت نیز افزایش یابد (مزیت)

در واقع هر چه Window بزرگ‌تر باشد اطلاعات موضوع و دامنه محور بیشتری استخراج می‌شود. هر چه این پارامتر کمتر باشد اطلاعات بیشتری درباره خود کلمه استخراج می‌شود.

انتخاب این پارامتر بستگی به متن (Corpus) ما نیز دارد برای مثال گوگل در مقاله ای کل جمله را به عنوان Window در نظر گرفت. اما در English corpus of short messages این پارامتر برابر 2 در نظر گرفته شده است.

نکته مهمی که باید در نظر بگیریم این است که همانند سایر هایپرپارامترها اندازه پنجره نیز باید Tune شود و بهترین حالت روی Validation انتخاب شود.

ساختن بردار نهایی Embedding: ابتدا باید طول بردار Embedding را انتخاب کنیم که در صورت سوال گفته شده است آن را برابر 2 در نظر بگیریم. در واقع با این کار هر کلمه را به یک بردار با دو مقدار نگاشت می‌کنیم. هر چه طول این بردار بیشتر باشد می‌توانیم اطلاعات بیشتری درباره کلمه را داشته باشیم. برای مثال:

“dad” = $[0.1548, 0.4848]$

“mom” = $[0.8785, 0.8974]$

برای به دست آوردن این دو مقدار برای هر کلمه از یک شبکه عصبی کاملا متصل دو لایه استفاده می‌کنیم. تعداد نورون لایه میانی همان اندازه بردار Embedding ما است که مقدار آن برابر 2 است. تابع فعالسازی لایه میانی Linear و لایه خروجی Softmax است. ابعاد لایه ورودی و خروجی نیز برابر تعداد کلمات منحصر به فرد است. بنابراین هر واحد لایه ورودی (کلمه منحصر) با دو وزن به نورون‌های لایه میانی متصل است. این دو وزن همان Embedding به ازای آن کلمه است.

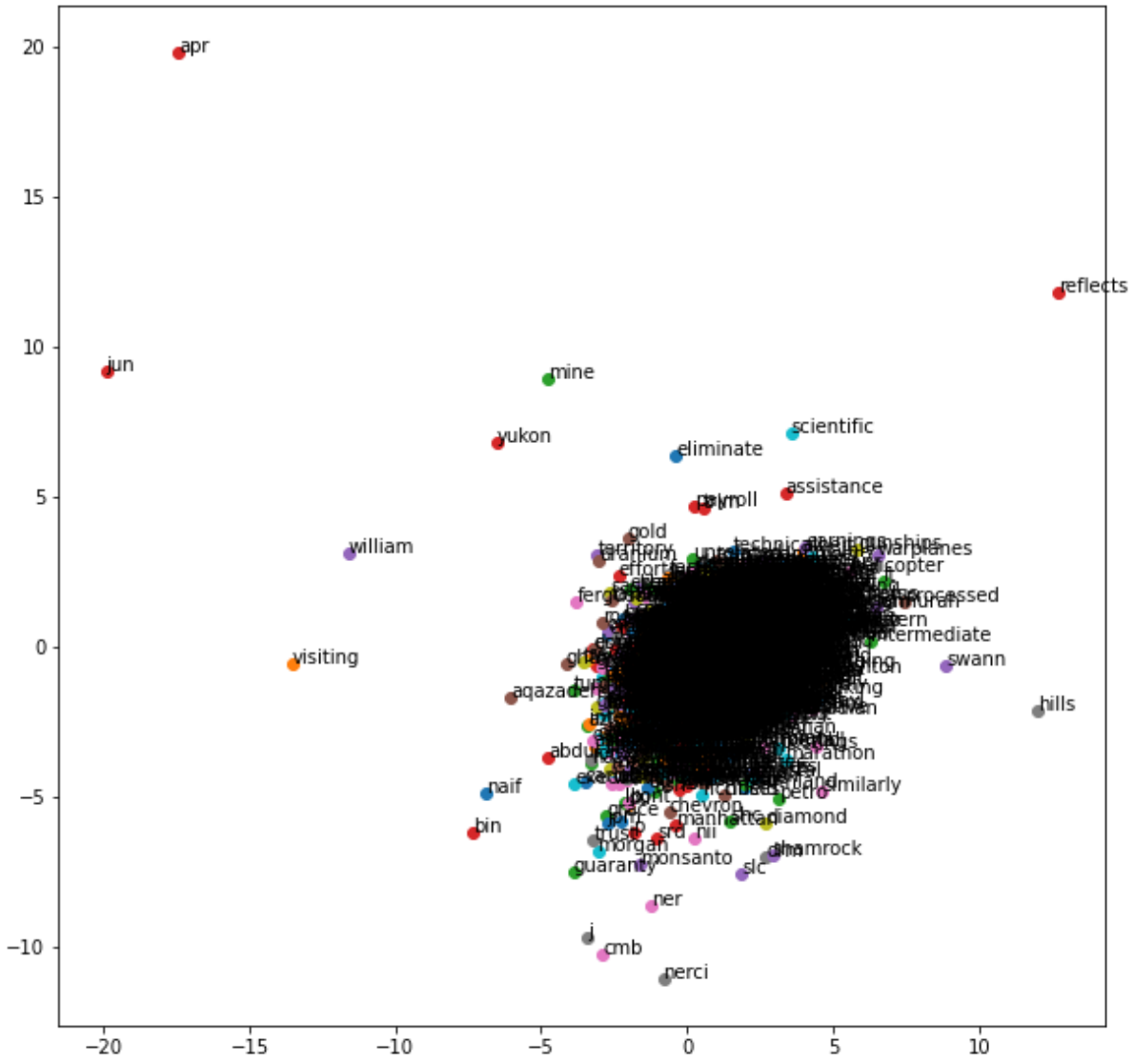
یک مدل Keras با توضیحات بالا ایجاد می‌کنیم. از بهینه‌ساز Adam و تابع ضرر Categorical Cross-entropy استفاده می‌کنیم.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 2728)]            0

 dense (Dense)               (None, 2)                 5458

 dense_1 (Dense)             (None, 2728)              8184

=================================================================
Total params: 13,642
Trainable params: 13,642
Non-trainable params: 0
_________________________________________________________________
```

مدل را در 1000 ایپک و اندازه Batch برابر 256 آموزش می‌دهیم. که نتایج هر ایپک در نوتبوک آمده است. Loss به مرور کاهش می‌یابد و مقدار آن از 7.7903 به 6.1848 می‌رسد.

همانطور که گفته شد وزن‌هایی که لایه ورودی را به لایه میانی متصل می‌کنند همان Embedding هر کلمه هستند. این وزن‌ها را استخراج می‌کنیم و Embedding هر کلمه را در یک دیکشنری تولید می‌کنیم.

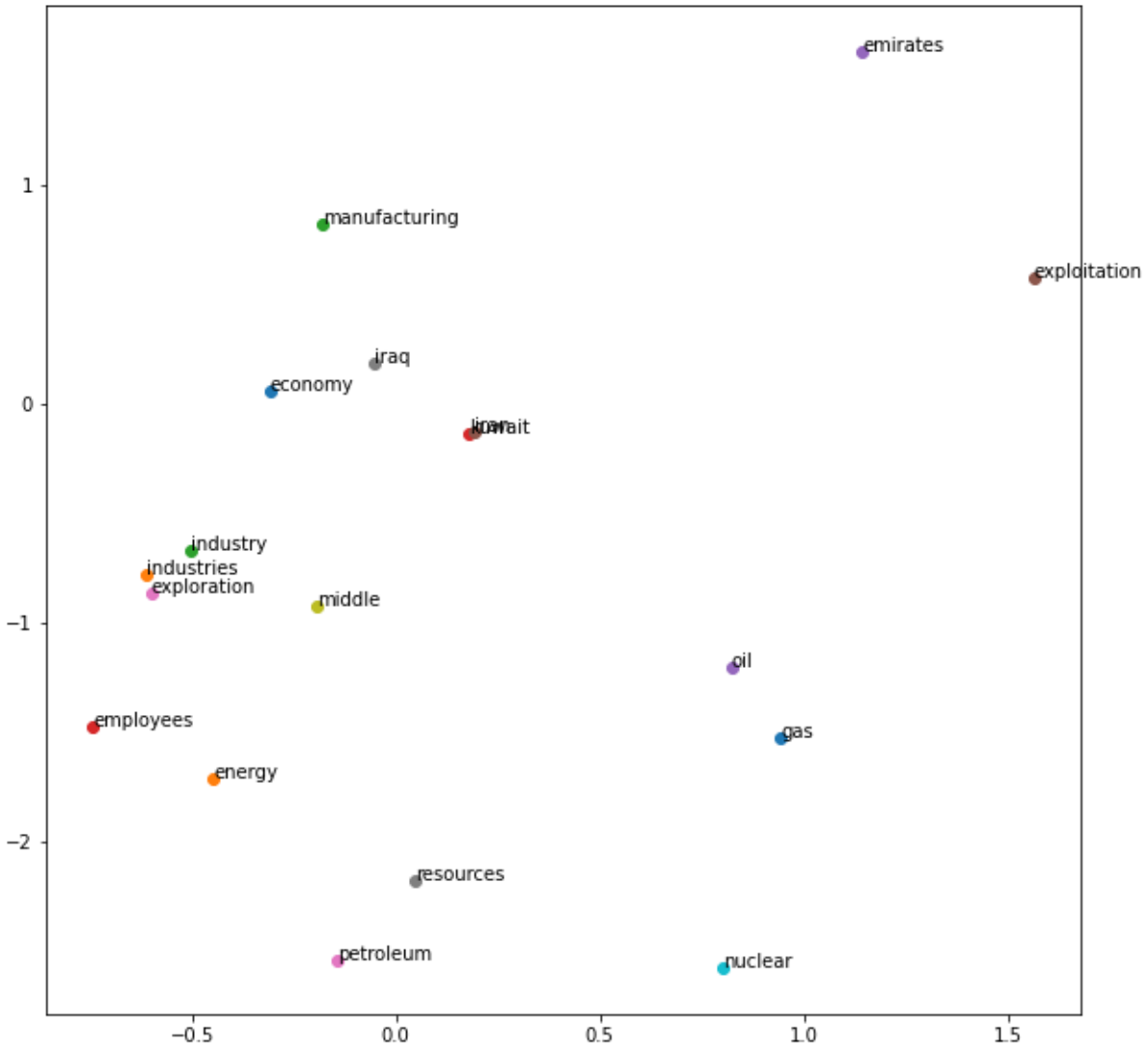
توزیع تمامی کلمات منحصر (2728 کلمه) را در یک صفحه مختصات دو بعدی رسم می‌کنیم تا Clusterهای شکل گرفته را مشاهده کنیم.



با توجه به این که کلمات مربوط به یک دسته خبری خاص (Crude) بود، اکثر کلمات در یک Cluster دور هم جمع شده اند.

همین نمودار را به ازای کلماتی که در صورت تمرین گفته شده است رسم می‌کنیم که نتیجه آن در صفحه بعد آورده شده است. همانطور که مشاهده خواهید کرد کلمات مرتبط با هم تشکیل کلاستر داده اند. مثلا کلمات ایران، کویت،

عراق، اقتصاد | یا کلمات نفت و گاز | یا کلمات صنعت، صنایع و اکتشاف. اما کلمه‌ی هسته‌ای یا امارات یا بهره‌برداری در داخل کلاستری وجود ندارند و جزو کلمات پرت هستند.



با استفاده از تابع find_similar که بر مبنای فاصله اقلیدسی بردار Embedding کلمات کار می‌کند، برای برخی کلمات ۵ تا از نزدیک‌ترین کلمات به همراه فاصله آن‌ها را چاپ می‌کنیم. که نتایج به صورت زیر است:

```
Similar words to iran:
[('meetings', 0.01711179), ('kuwait', 0.017799867), ('communication',
0.027351283), ('inch', 0.028207982), ('panel', 0.045745846)]

Similar words to gas:
[('shares', 0.013574208), ('marketed', 0.19188003), ('obligations',
0.19762509), ('bombings', 0.19985059), ('own', 0.21956536)]

Similar words to oil:
[('eia', 0.0619546), ('remaining', 0.0642409), ('stake', 0.069517784),
('old', 0.07789461), ('everyone', 0.110637926)]
```

```
Similar words to industry:
[('ec', 0.026133515), ('investigating', 0.041361853), ('latest',
0.055735853), ('agreeing', 0.07408273), ('sources', 0.078879856)]

Similar words to manufacturing:
[('lot', 0.028212644), ('aggressively', 0.051717106), ('building',
0.056598727), ('historic', 0.06897059), ('mergers', 0.07399485)]

Similar words to nuclear:
[('independent', 0.1448585), ('angered', 0.17614391), ('agip',
0.19224179), ('attorney', 0.2565492), ('congressman', 0.28112817)]

Similar words to emirates:
[('paid', 0.055510256), ('burning', 0.06943781), ('moscow', 0.07304275),
('growing', 0.08761337), ('wounded', 0.120995514)]
```

لیست کامل‌تر این کلمات در داخل نوتبوک موجود است. هر چه مقدار فاصله اقلیدسی کمتر باشد دو کلمه به همدیگر نزدیک‌تر هستند. مثلا در اینجا کلمه ایران بسیار شبیه به ملاقات، کویت، ارتباط است. یا کلمه هسته‌ای شبیه کلمه مستقل و خشم است که می‌توان ارتباط معنایی آن‌ها را متوجه شد. (نیروی هسته‌ای کشورها را مستقل می‌کند و ...)

- ✓ در پیاده سازی که به عنوان مرجع داده شده بود ماتریس‌های X و Y توسط کتابخانه Sklearn به ماتریس Sparse تبدیل می‌شدند زیرا تعداد صفرها در آن‌ها زیاد بود. در نسخه جدید تنسورفلو شبکه را نمی‌توان با ورودی Sklearn آموزش داد و بایستی از توابع toSparseTensor خود تنسورفلو استفاده کرد که من با بررسی حالت Sparse و Dense متوجه شدم سرعت در Dense بهتر است پس ماتریس‌ها را Sparse نکردم.

منابع:

https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8

https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

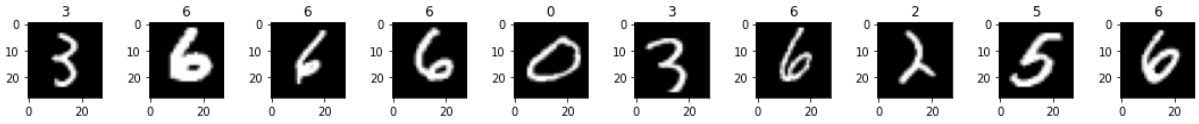
https://newbedev.com/word2vec-effect-of-window-size-used

۳ سوال سوم

مشابه کارهایی که در تمرین ۶ نیز انجام دادیم، ابتدا دیتاست را با تابع mnist.load_data() بارگذاری می‌کنیم. سپس با استفاده از تابع shuffle ترتیب داده‌ها را بهم می‌زنیم. شکل دیتاست تا الان به صورت زیر است که خطوط به ترتیب مربوط به x_train، y_train، x_test و y_test هستند:

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

حال ۱۰ تصویر اول در دیتاست آموزشی را به همراه برچسب نمایش می‌دهیم:



با استفاده از تابع np.expand_dims یک بعد دیگر به تنسورها اضافه می‌کنیم تا در لایه Conv2D قابل استفاده باشند. شکل x_train و x_test اکنون به صورت زیر است:

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

تنسورهای x_train و x_test را بر عدد 255 تقسیم می‌کنیم تا مقدار پیکسل‌ها بین دو عدد 0 و 1 بماند. با استفاده از تابع to_categorical تنسورهای y_train و y_test را به حالت One-hot می‌بریم. اکنون شکل این دو تنسور به صورت زیر است:

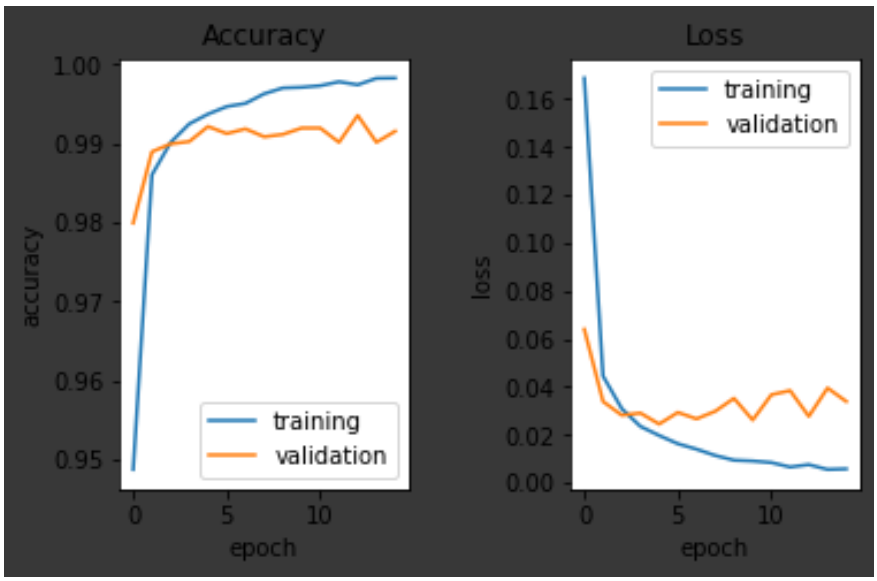
```
(60000, 10)
(10000, 10)
```

مطابق صورت تمرین یک مدل می‌سازیم که Summary آن به صورت زیر است:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d_3 (Conv2D)           (None, 28, 28, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 7, 7, 64)          36928

 max_pooling2d_5 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 576)               0

 dense_2 (Dense)             (None, 128)               73856

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 130,890
Trainable params: 130,890
Non-trainable params: 0
_________________________________________________________________
```

مدل ساخته شده را مطابق هایپرپارامترهای داده شده ابتدا کامپایل می‌کنیم و سپس آموزش می‌دهیم:

```
loss: 0.0058 - accuracy: 0.9982 - val_loss: 0.0339 - val_accuracy: 0.9915
```

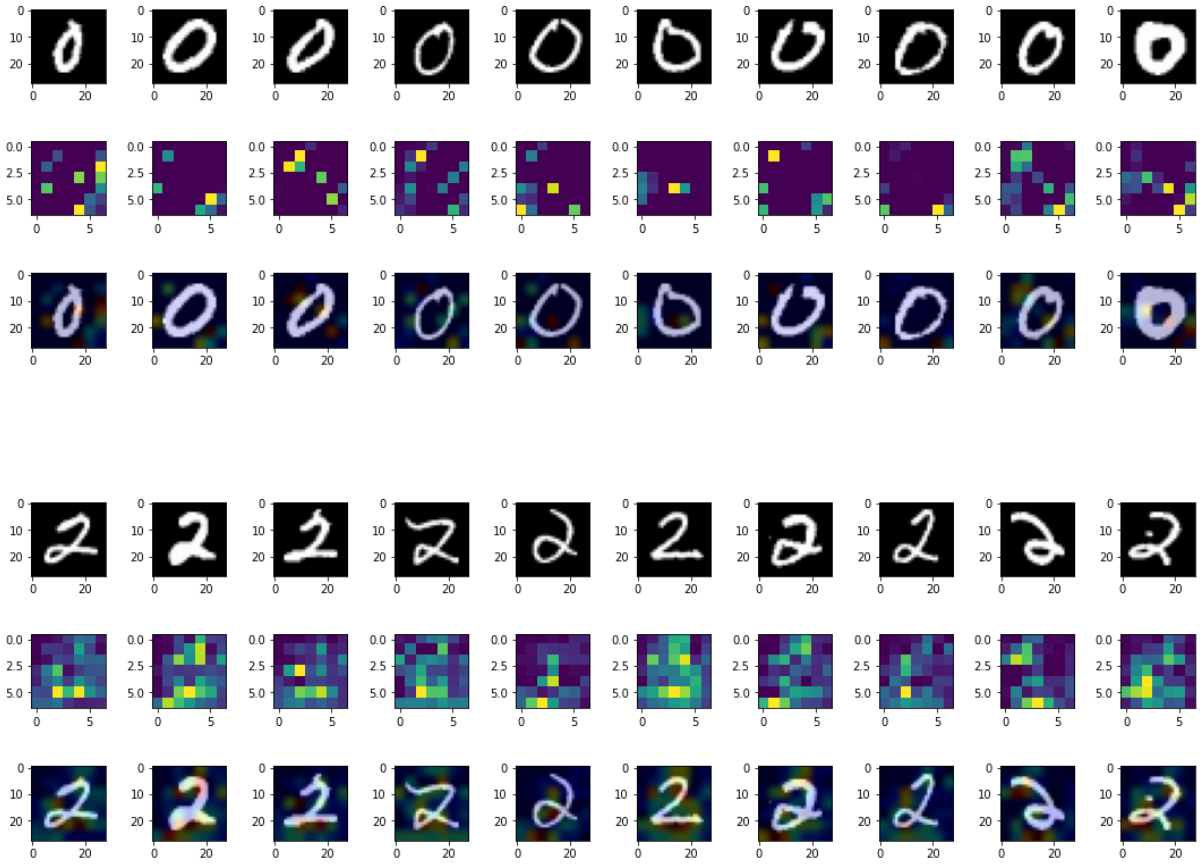


اکنون به پیاده‌سازی الگوریتم Grad-CAM می‌پردازیم. مبنای پیاده‌سازی اسلاید جلسه ۲۰ کلاس درسی و لینک زیر که توسط خود Keras ارائه شده‌است می‌باشد. لینک زیر به صورت کامل مطالعه شده است و توابعی از آن در پیاده‌سازی من استفاده شده است.

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

تابع make_gradcam_heatmap با ورودی گرفتن یک تصویر و همچنین مدل، لایه آخر کانولوشنی را در نظر می‌گیرد. تابع فعالسازی لایه آخر در مدل را باید None کنیم زیرا دیگر مسئله دسته‌بندی نداریم که بخواهیم Softmax بزنیم. این روش گرادیان نورون مربوط به کلاس تصویر در لایه آخر مدل را نسبت به آخرین لایه کانولوشنی محاسبه می‌کند. نتیجه اینکار تولید یک ماتریس 7x7 به نام Heatmap است که نشان دهنده نواحی مهم در تصویر برای پیشبینی کلاس مربوطه است. یعنی اگر بخواهیم کلاس 0 انتخاب شود به چه نقاطی از تصویر باید توجه کنیم.

تابع make_superimposed با ورودی گرفتن تصویر اصلی و Heatmap یک تصویر ترکیب شده از این دو را می‌سازد تا بتوانیم نقاط مهم روی تصویر اصلی را بهتر درک کنیم. در واقع این تابع تصویر اصلی را روی تصویر Heatmap می‌اندازد.



همانطور که از تصاویر پیداست لایه آخر کانولوشنی به ازای هر کلاس به بخش خاصی دقت بیشتری کرده و موقعیت این بخش در تمامی ۱۰ مثال برای آن رقم تقریبا یکسان است. در واقع هر رقم یک ناحیه منحصر به فرد متمایز کننده دارد که شبکه سعی کرده با پیدا کردن این نواحی کلاس مربوط به اعداد را تشخیص دهد. اگر دقت کنیم می‌بینم نقاط پر رنگ روی Heatmap به ازای هر کلاس در جای متفاوتی قرار گرفته که این به شبکه کمک می‌کند دسته بندی راحت تر صورت بگیرد.