دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین سری چهاردهم

علی صداقی

۹۷۵۲۱۳۷۸

۱ سوال اول

سوال ۱) کلمه علی با اسامی دیگر عربی بسیارشبیه است. این کلمه با هر دو فرم انگلیسی کلمات شبیه شده (Mohammed, Mohammad). کلمه Khan کمی عجیب است. احتمالا می‌توان گفت در این دیتاست این دو کلمه با هم زیاد آمده بودند. برای کلمه ایران کلمه احمدی‌نژاد خروجی داده شده پس احتمالا این دیتاست از متن‌های کمی قدیمی فراهم شده. وجود کلمه لیبی در کنار ایران نیز کمی عجیب است. نکته جالب این است که برای کلمه ایران کلمه هسته‌ای پیشنهاد شده اما در لیست کلمات شبیه هسته‌ای کلمه ایران وجود ندارد.

نزدیک‌ترین کلمه به لیورپول کلمه منچستر است که رقیب سنتی آنهاست و دارای تضاد زیادی است. این نکته که دو کلمه رقیب شبیه هم در نظر گرفته شده اند جالب است. آنفیلد که ورزشگاه لیورپول است انتظار می‌رفت شبیه‌تر باشد تا منچستر.

در برخی موارد جمع شده کلمه نزدیک‌ترین کلمه است که این نکته از نظر من خیلی جالب نیست (Book, Books).

در مجموع عملکرد مدل خوب است.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
ali	(hassan, 0.67)	(ahmed, 0.65)	(mohammad, 0.65)	(mohammed, 0.65)	(muhammad, 0.63)	(ahmad, 0.61)	(akbar, 0.61)	(khan, 0.61)	(saleh, 0.6)	(ibrahim, 0.58)
brother	(son, 0.89)	(father, 0.86)	(nephew, 0.86)	(cousin, 0.83)	(uncle, 0.83)	(grandson, 0.76)	(sons, 0.74)	(elder, 0.73)	(grandfather, 0.73)	(eldest, 0.73)
iran	(tehran, 0.8)	(iranian, 0.79)	(syria, 0.68)	(nuclear, 0.66)	(iranians, 0.65)	(iraq, 0.65)	(ahmadinejad, 0.62)	(enrichment, 0.61)	(libya, 0.61)	(arabia, 0.6)
nuclear	(atomic, 0.78)	(weapons, 0.73)	(reactor, 0.69)	(reactors, 0.68)	(iran, 0.66)	(enrichment, 0.65)	(pyongyang, 0.64)	(proliferation, 0.64)	(plutonium, 0.64)	(uranium, 0.63)
liverpool	(manchester, 0.8)	(anfield, 0.74)	(newcastle, 0.72)	(leeds, 0.72)	(chelsea, 0.71)	(everton, 0.69)	(blackburn, 0.67)	(portsmouth, 0.67)	(tottenham, 0.67)	(southampton, 0.66)
england	(wales, 0.71)	(scotland, 0.65)	(ireland, 0.64)	(australia, 0.62)	(manchester, 0.6)	(cricket, 0.6)	(indies, 0.6)	(yorkshire, 0.58)	(lancashire, 0.58)	(english, 0.58)
book	(books, 0.85)	(author, 0.77)	(novel, 0.75)	(published, 0.75)	(memoir, 0.7)	(wrote, 0.7)	(written, 0.7)	(essay, 0.68)	(biography, 0.68)	(autobiography, 0.68)
university	(professor, 0.79)	(graduate, 0.77)	(harvard, 0.76)	(college, 0.75)	(yale, 0.73)	(faculty, 0.73)	(stanford, 0.68)	(universities, 0.68)	(campus, 0.68)	(student, 0.67)
soldier	(soldiers, 0.75)	(wounded, 0.73)	(policeman, 0.7)	(army, 0.65)	(killed, 0.63)	(prisoner, 0.62)	(dead, 0.6)	(serviceman, 0.6)	(man, 0.59)	(sergeant, 0.58)
mother	(daughter, 0.86)	(wife, 0.86)	(grandmother, 0.84)	(husband, 0.81)	(sister, 0.8)	(father, 0.79)	(her, 0.78)	(daughters, 0.76)	(woman, 0.76)	(she, 0.75)

سوال ۲) نتایج به دست آمده کاملا منطقی است. اما فاصله دو کلمه woman و man نسبت به کلمه Ali بسیار کم است. علی بر اساس ملاک مذکر بودن، فوتبال بر اساس ملاک ورزش بودن، لپتاپ وسیله الکترونیکی بودن، دانشگاه و مدرسه محل تحصیل، خودکار و مداد وسیله نگارشی بودن نزدیک هم انتخاب شده اند.

	Near Word	Far Word
ali	(man, 0.69)	(woman, 0.76)
footbal	(basketball, 0.87)	(war, 1.16)
laptop	(phone, 0.49)	(gasoline, 0.79)
university	(school, 0.36)	(police, 0.72)
pen	(pencil, 0.5)	(charger, 0.99)

سوال ۳) فاصله و کلماتی که توسط روش اول خروجی داده شده اند بسیار بهتر هستند. زیرا فرمولی که روش اول دارد به این صورت است:

$$w_4 - w_1 \approx w_2 - w_3$$

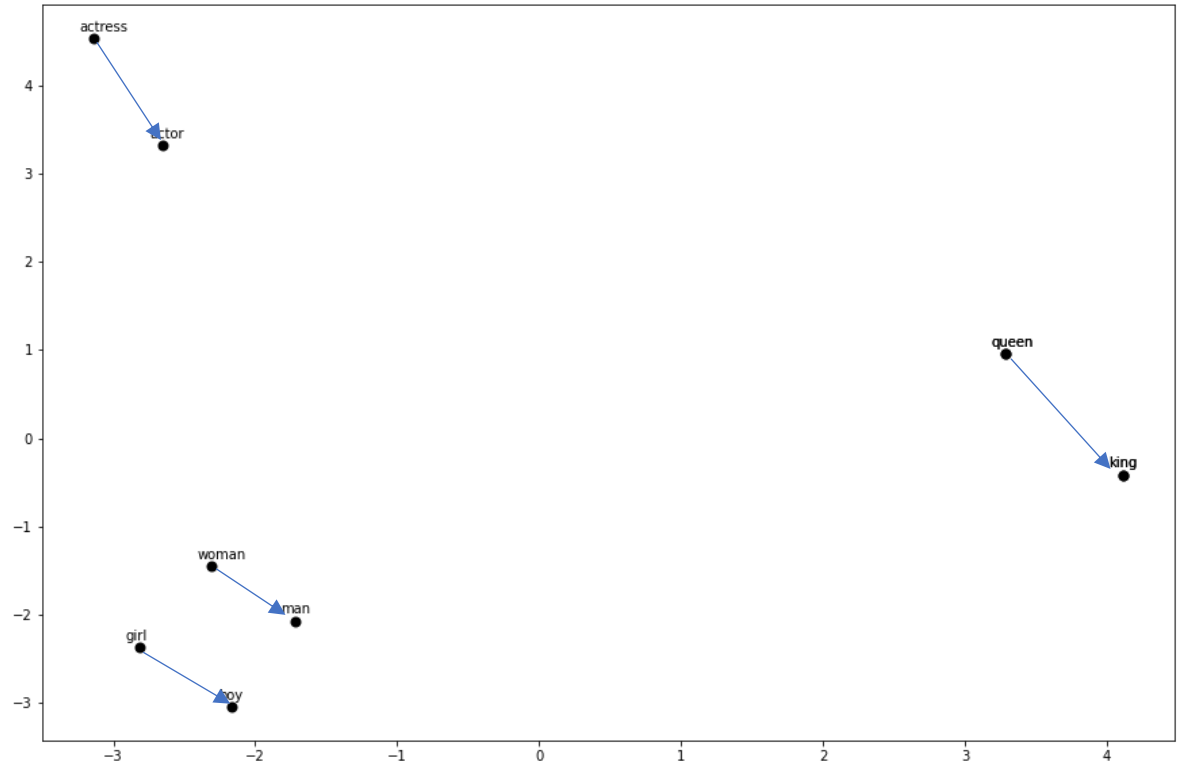
$$w_4 \approx w_1 + w_2 - w_3$$

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
('king', 'woman', 'man')	(queen, 0.7)	(princess, 0.61)	(monarch, 0.59)	(throne, 0.58)	(prince, 0.58)	(elizabeth, 0.55)	(daughter, 0.54)	(kingdom, 0.53)	(mother, 0.52)	(crown, 0.52)
('actor', 'girl', 'boy')	(actress, 0.87)	(starring, 0.71)	(actresses, 0.69)	(actors, 0.69)	(starred, 0.68)	(screenwriter, 0.63)	(comedian, 0.63)	(film, 0.61)	(movie, 0.6)	(filmmaker, 0.58)
('doctor', 'she', 'he')	(nurse, 0.7)	(mother, 0.6)	(woman, 0.6)	(her, 0.59)	(physician, 0.57)	(pregnant, 0.57)	(dr., 0.56)	(doctors, 0.56)	(patient, 0.55)	(hospital, 0.55)
('homemaker', 'she', 'he')	(housewife, 0.71)	(schoolteacher, 0.61)	(widowed, 0.55)	(businesswoman, 0.55)	(mom, 0.55)	(waitress, 0.53)	(hairdresser, 0.53)	(mother, 0.52)	(socialite, 0.52)	(grandmother, 0.51)
('football', 'woman', 'man')	(basketball, 0.67)	(soccer, 0.64)	(volleyball, 0.58)	(league, 0.55)	(softball, 0.55)	(hockey, 0.54)	(rugby, 0.53)	(ncaa, 0.52)	(club, 0.52)	(collegiate, 0.52)

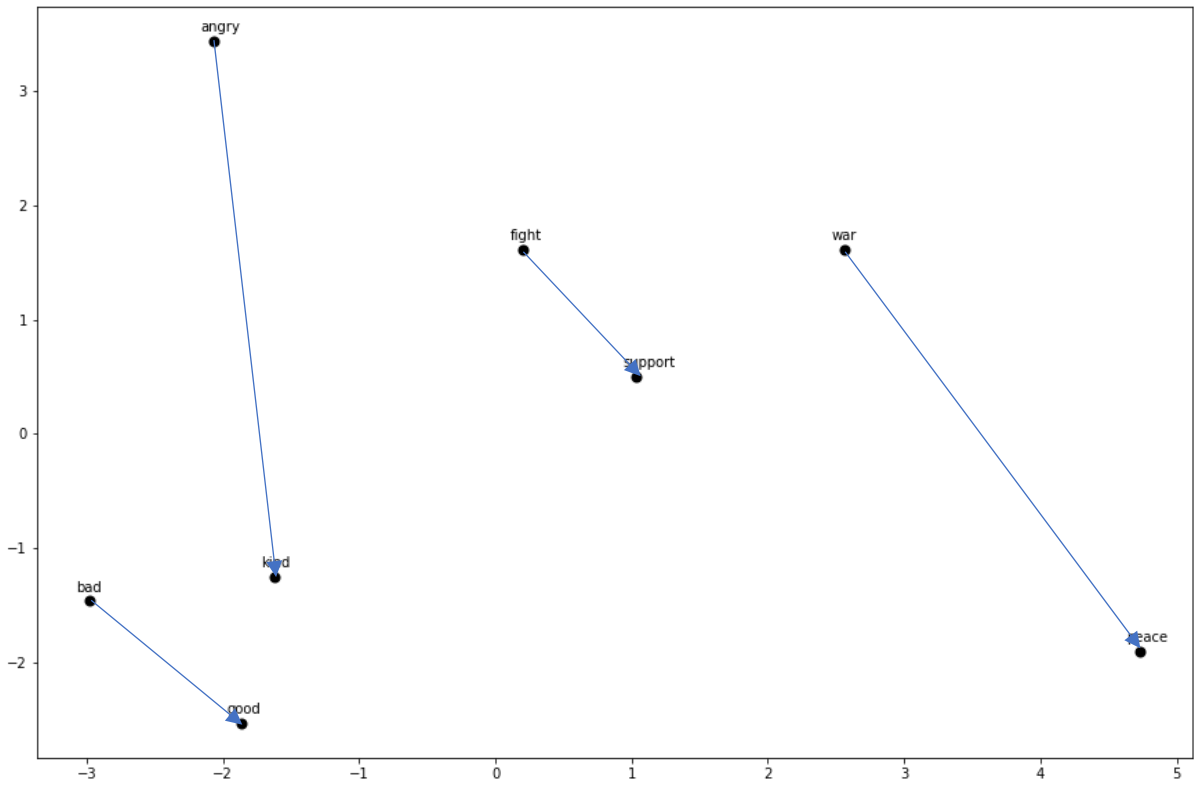
روش دوم به دلیل غلطی که در فرمولاسیون دارد نتایج غیر منطقی تولید می کند.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
('king', 'woman', 'man')	(prince, 0.55)	(ii, 0.54)	(brother, 0.54)	(iii, 0.53)	(reign, 0.53)	(uncle, 0.52)	(kingdom, 0.5)	(henry, 0.5)	(kings, 0.5)	(iv, 0.48)
('actor', 'girl', 'boy')	(comedian, 0.61)	(starring, 0.6)	(actors, 0.58)	(starred, 0.56)	(movie, 0.55)	(brother, 0.55)	(father, 0.55)	(film, 0.53)	(musician, 0.53)	(filmmaker, 0.52)
('doctor', 'she', 'he')	(physician, 0.66)	(surgeon, 0.57)	(doctors, 0.57)	(medical, 0.56)	(him, 0.54)	(dr., 0.54)	(himself, 0.53)	(his, 0.52)	(hospital, 0.52)	(man, 0.51)
('homemaker', 'she', 'he')	(43-year, 0.59)	(schoolteacher, 0.59)	(42-year, 0.55)	(housewife, 0.55)	(55-year, 0.54)	(48-year, 0.53)	(bricklayer, 0.53)	(47-year, 0.52)	(44-year, 0.52)	(39-year, 0.52)
('football', 'woman', 'man')	(soccer, 0.68)	(baseball, 0.64)	(team, 0.63)	(basketball, 0.62)	(league, 0.62)	(players, 0.61)	(rugby, 0.61)	(club, 0.6)	(game, 0.58)	(hockey, 0.58)

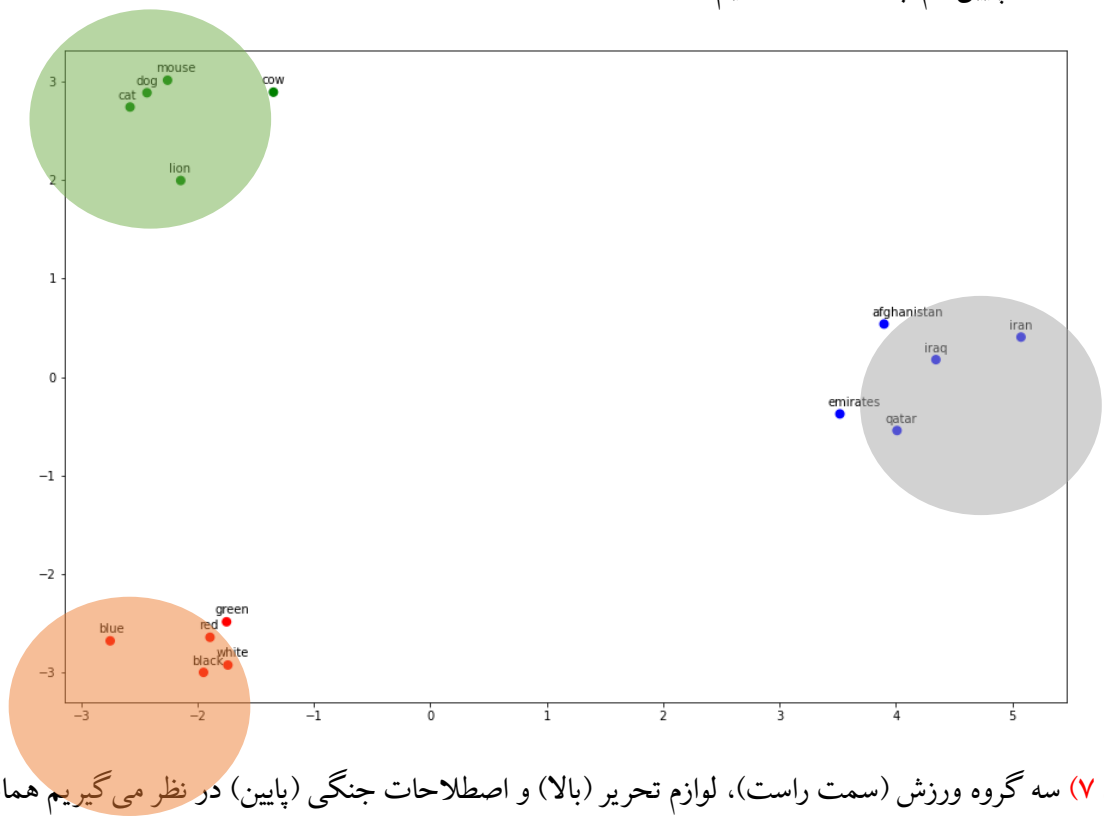
سوال ۴) بله. هر چه به سمت راست و پایین می‌رویم از حالت مونث به حالت مذکر شبیه‌تر می‌شویم. بردارهای ترسیم شده در شکل تقریبا معادل هم هستند و با همان برداری که از کلمه Actress به کلمه Actor می‌رویم می‌توانیم از کلمه Woman به Man برویم. این اکر نشان می‌دهیم که Embedding ویژگی‌های خوبی آموخته است.



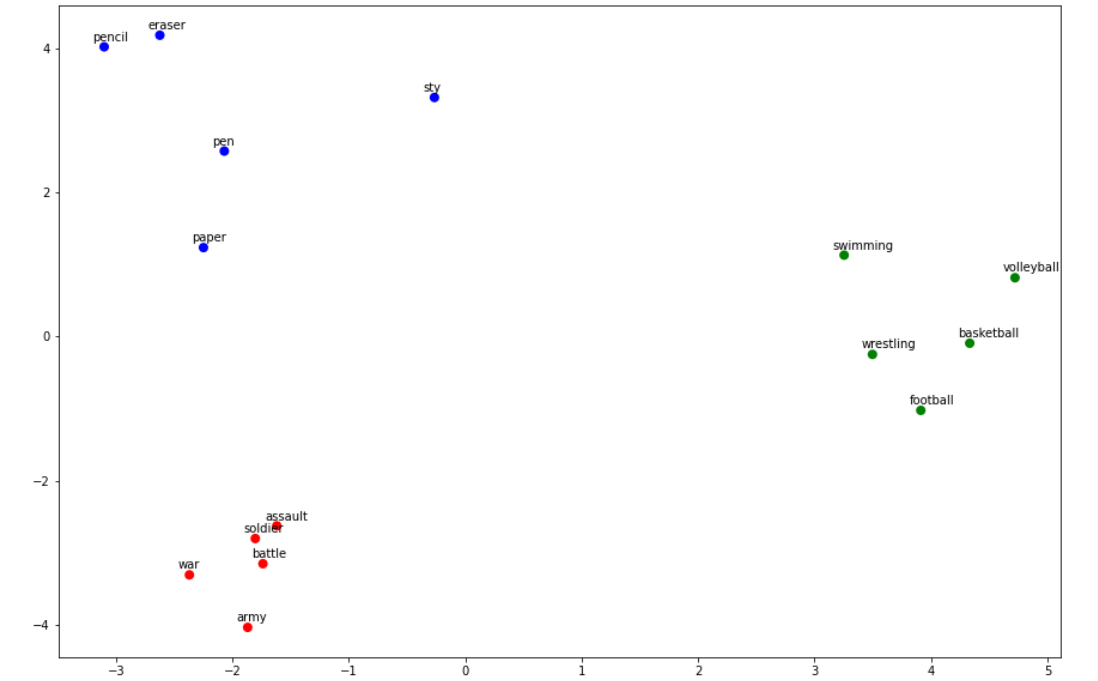
سوال ۵) کلمات در دو گروه (صلح و دوستی و مهربانی) و (جنگ و خشم و بدی) هستند. همانطور که در شکل و بردار ها نمایان هست، با بردارهایی تقریبا هم جهت (با طول متفاوت) می‌توانیم از یک گروه به گروه دیگر منتقل شویم. البته جهت این بردار از Angry به Kind کمی متفاوت است. دلیل آن این است که کلمه Kind معنی نوع نیز می‌دهد و احتمالا در این متن به معنی مهربان کم استفاده شده است.



سوال ۶) می‌توان یک خوشه‌بندی (Clustering) مشاهده کرد و گویی الگوریتم Embedding بدون داشتن داده برچسب‌دار توانسته گروه‌بندی کند (Unsupervised Learning). در سمت راست نمودار نام کشورها، در بالا نام حیوانات و در پایین نام چند رنگ را داریم.

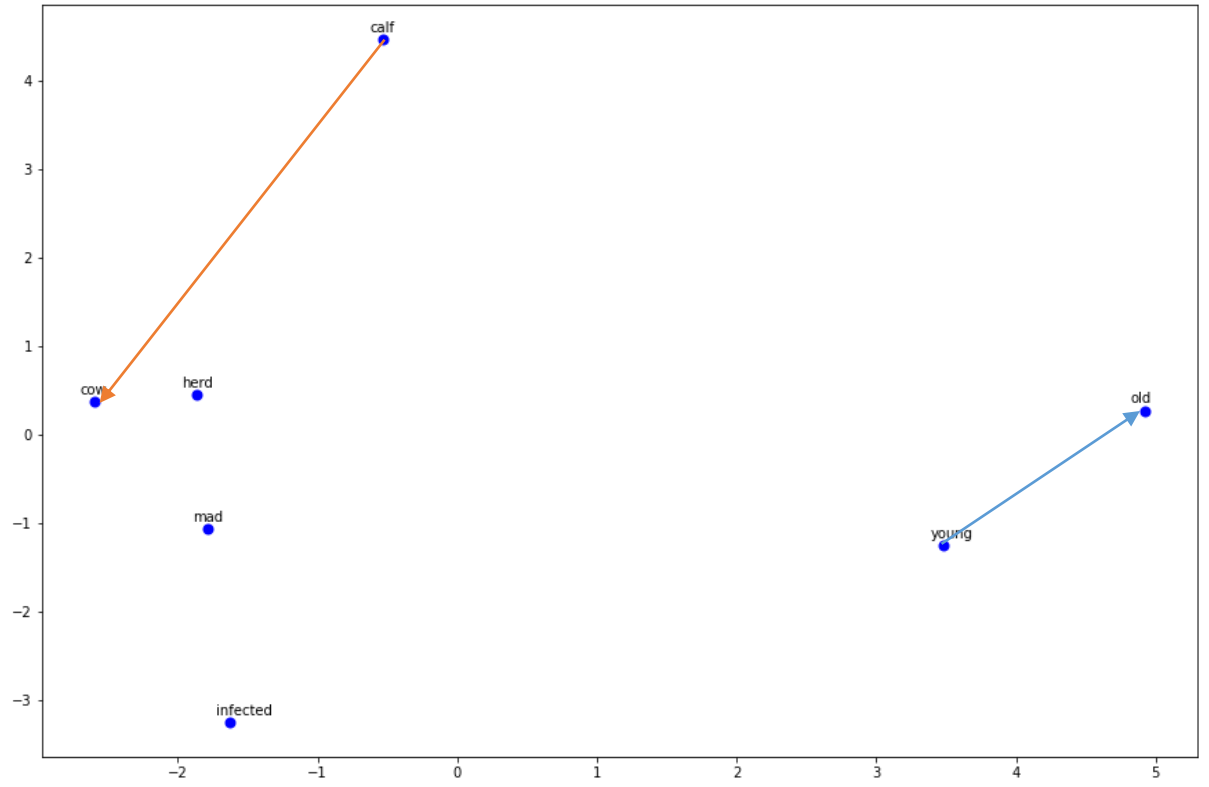


سوال ۷) سه گروه ورزش (سمت راست)، لوازم تحریر (بالا) و اصطلاحات جنگی (پایین) در نظر می گیریم همانطور که مشاهده می‌شود الگوریتم به خوبی توانسته این سه گروه را خوشه‌بندی (Cluster) کند. البته در گروه لوازم تحریر کمی فاصله کلمات از هم زیاد شده است!



سوال ۸) با توجه به این که رابطه بین Old و Young از جوان به پیر است پس از گاو (Cow) باید به گوساله (Calf) برسیم اما الگوریتم نتوانسته این کلمه را پیدا کند.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
('young', 'cow', 'old')	(mad, 0.6)	(cows, 0.58)	(sheep, 0.52)	(pigs, 0.52)	(herd, 0.51)	(bovine, 0.5)	(cattle, 0.49)	(infected, 0.48)	(spongiform, 0.48)	(animals, 0.48)



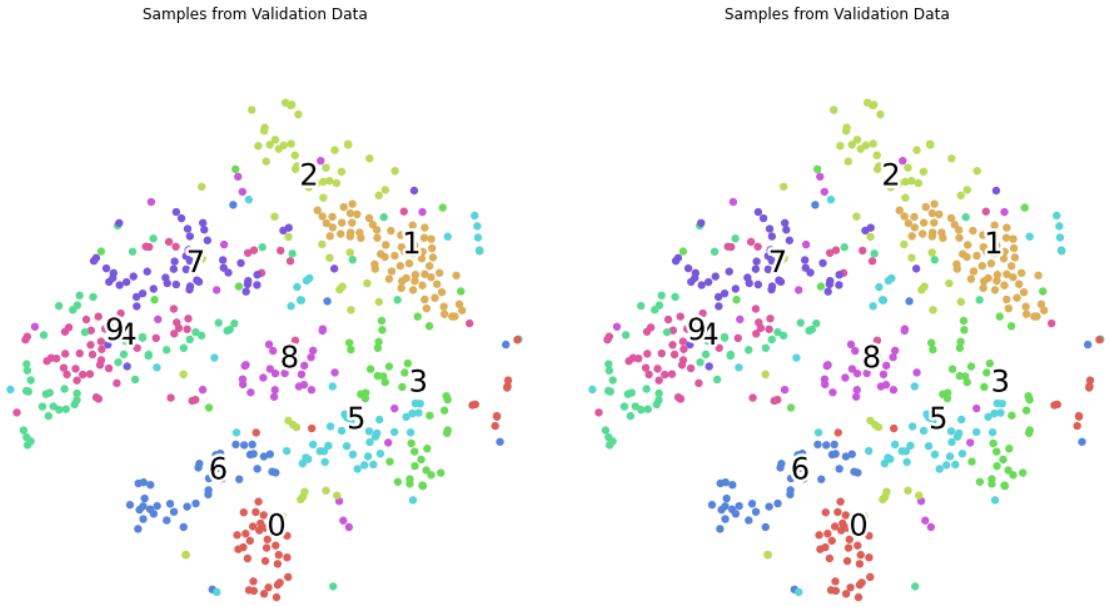
اگر نمودار را بررسی کنیم جهت این بردارها نیز کاملا برعکس است. دلیل این اتفاق می‌تواند کم بودن کلمات Calf و Cow در کنار هم باشد. دلیل دیگر این اتفاق این است که کلمه Calf در معنی گوشت (گوشت ساق پای گوساله) نیز استفاده می‌شود و ممکن است در این دیتاست این کلمه به معنی گاو جوان (گوساله) کم بکار برده شده باشد. همین امور باعث شده الگوریتم کلمه نامرتبط Mad را به عنوان بهترین کلمه خروجی دهد.

۲ سوال دوم

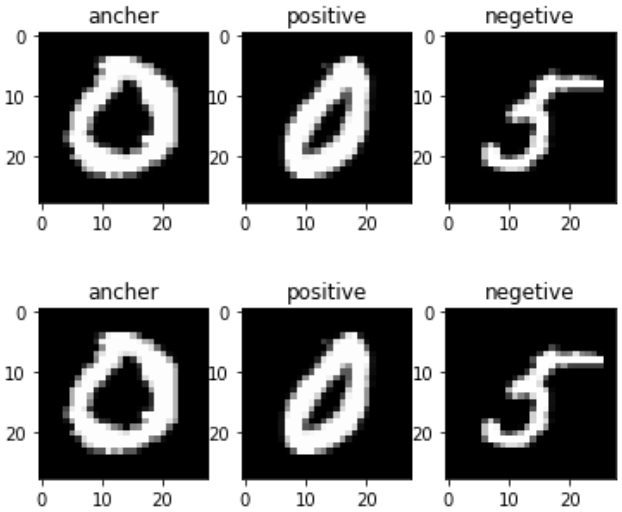
ابتدا داده ورودی را نرمالیزه می‌کنیم تا فرایند آموزش راحت‌تر شود.

```
x_train = x_train / 255.
x_test = x_test / 255.
```

قبل از آموزش توزیع داده‌ها در ناحیه دو بعدی که توسط T-SNE کاهش بعد داده شده است به صورت زیر است.



تابعی که برای ایجاد Tripletها بود را اجرا می‌کنیم. نتایج به صورت زیر است.



دو تنسور X_train و X_test را به یک لیست با سه درایه تبدیل می‌کنیم تا بتوانیم وارد یک شبکه با سه ورودی (Input) کنیم. هر درایه از این لیست یک تنسور با شکل زیر است.

درایه 0 لیست مربوط به Anchor، درایه 1 مربوط به Positive و درایه 2 مربوط به Negative است.

```
[15]  1 print(X_train[0].shape)
      2 print(X_test[0].shape)

      (180000, 28, 28)
      (45000, 28, 28)
```

تابع محاسبه Loss را مطابق با فرمول Triplet Loss کامل می‌کنیم.

```
anchor, positive, negative = y_pred
dap = K.sum(K.square(anchor - positive), axis=1)
dan = K.sum(K.square(anchor - negative), axis=1)
loss = K.maximum(dap - dan + alpha, 0)
return loss
```

در پیاده‌سازی این تابع از Backend کتابخانه Keras یعنی K استفاده کردیم تا بتوانیم مشتق را نیز به صورت اتوماتیک محاسبه کنیم.

یک تابع نیز برای میانگین گرفتن از Triplet Loss تعریف می‌کنیم.

```
1 def mean_triplet_loss(y_true, y_pred):
2     return K.mean(y_pred)
```

یک مدل پایه Backbone به صورت زیر ایجاد می‌کنیم.

```
Model: "Backbone"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 26, 26, 32)        320

 conv2d_9 (Conv2D)           (None, 24, 24, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 12, 12, 32)       0
 2D)

 dropout_6 (Dropout)         (None, 12, 12, 32)        0

 conv2d_10 (Conv2D)          (None, 10, 10, 64)        18496

 conv2d_11 (Conv2D)          (None, 8, 8, 64)          36928

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 dropout_7 (Dropout)         (None, 4, 4, 64)          0

 flatten_2 (Flatten)         (None, 1024)              0

 dense_4 (Dense)             (None, 512)               524800

 dropout_8 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 10)                5130

=================================================================
Total params: 594,922
Trainable params: 594,922
Non-trainable params: 0
_________________________________________________________________
```

این مدل را با Concatenate کردن (ما یک Lambda تعریف کردیم که مقدار Triplet Loss را اعمال می‌کرد) روی سه Embedding (Anchor, Positive, Negative) تغییر شکل می‌دهیم.

```
# Shared embedding layer for positive and negative items
embedding_net = embedding_pred_net([28,28,1,])

anchor_embedding = embedding_net(anchor_in)
positive_embedding = embedding_net(positive_in)
negative_embedding = embedding_net(negative_in)

merged_vector = Lambda(triplet_loss)([anchor_embedding, positive_embedding, negative_embedding])

model = Model(inputs=[anchor_in, positive_in, negative_in], outputs=merged_vector)
```

خلاصه (Summary) مدل به صورت زیر است:

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================================
 input-anchor (InputLayer)   [(None, 28, 28, 1)]  0           []

 input-positive (InputLayer) [(None, 28, 28, 1)]  0           []

 input-negative (InputLayer) [(None, 28, 28, 1)]  0           []

 Backbone (Sequential)       (None, 10)            594922      ['input-anchor[0][0]',
                                                                 'input-positive[0][0]',
                                                                 'input-negative[0][0]']

 lambda_1 (Lambda)           (None,)               0           ['Backbone[0][0]',
                                                                 'Backbone[1][0]',
                                                                 'Backbone[2][0]']

==================================================================================================
Total params: 594,922
Trainable params: 594,922
Non-trainable params: 0
__________________________________________________________________________________________________
```

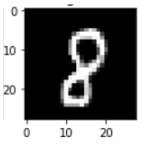
مدل را در 5 اپیک (چون دیتا زیاد و بچ کوچک است نیاز به اپیک بالا نیست) و اندازه بچ 32 آموزش می‌دهیم. تنسورهای Zero را به عنوان Y (برچسب) به تابع Fit می‌دهیم زیرا نیازی به برچسب نداریم. طبیعتا متریک دقت نیز بی معنی است.

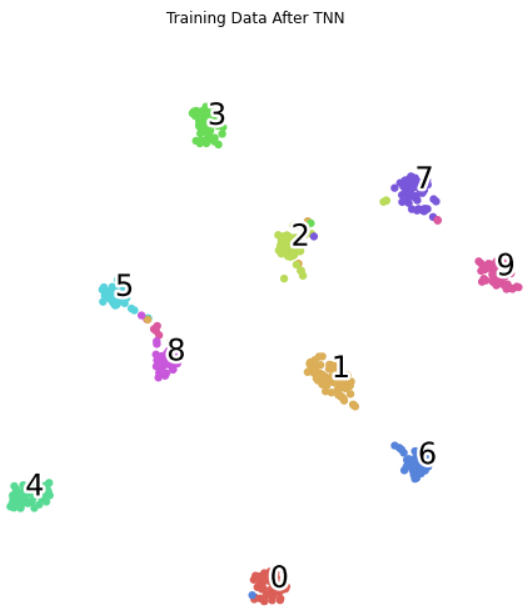
همانطور که در نتایج زیر مشاهده می‌شود میزان Loss هم روی آموزش هم روی ارزیابی به سرعت کاهش می‌یابد و الگوریتم به خوبی این تفکیک را انجام می‌دهد. بهترین عملکرد مدل در اپاک چهارم بوده است و می‌توانستیم نتایج آن بخش را برگردانیم.

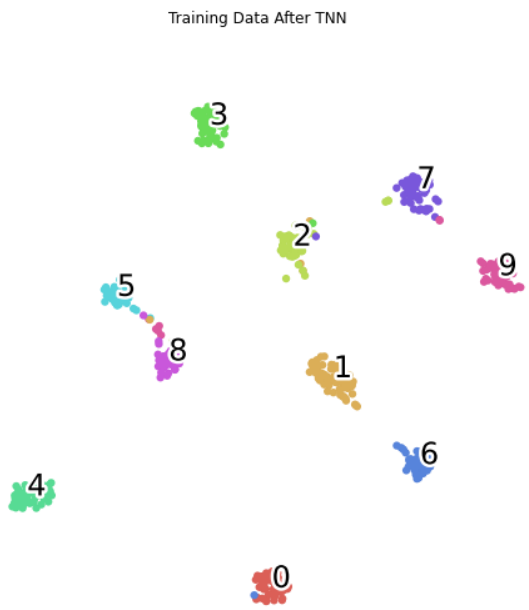
```
Epoch 1/5
5625/5625 [==============================] - 124s 20ms/step - loss: 0.0279 - val_loss: 0.0094
Epoch 2/5
5625/5625 [==============================] - 112s 20ms/step - loss: 0.0022 - val_loss: 0.0106
Epoch 3/5
5625/5625 [==============================] - 114s 20ms/step - loss: 0.0011 - val_loss: 0.0146
Epoch 4/5
5625/5625 [==============================] - 113s 20ms/step - loss: 8.1445e-04 - val_loss: 0.0082
Epoch 5/5
5625/5625 [==============================] - 112s 20ms/step - loss: 5.9545e-04 - val_loss: 0.0134
```

- Best train loss: 8.1445e-04
- Best val loss: 0.0082

دقت الگوریتم در تصاویر نیز بسیار بالاست و فقط در چند نمونه اشتباه پیش‌بینی کرده. نکته جالب نزدیک بودن دو کلاس 5 و 8 می‌باشد که در عمل نیز شبیه به هم هستند. الگوریتم به خوبی توانسته عدد زیر را در کلاس 8 قرار دهد.



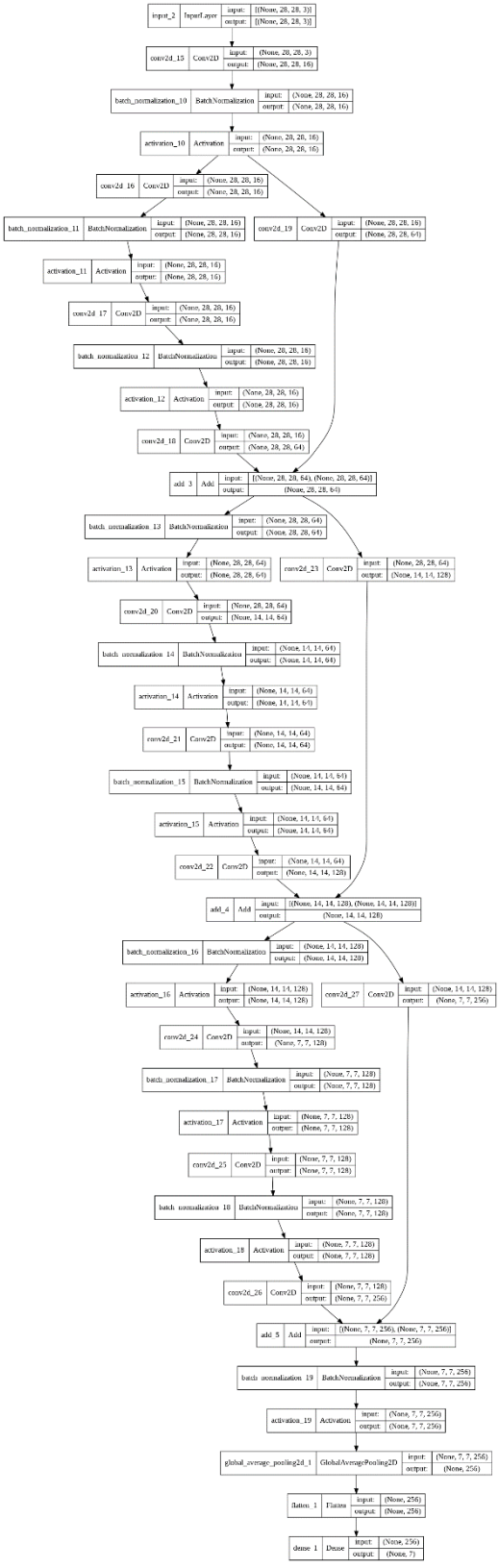




انتخاب Tripletها: خیر انتخاب‌ها اصلا خوب نبود زیرا کاملا به صورت Random اینکار صورت گرفت و سعی نکردیم Tripletهای چالشی برای مسئله ایجاد کنیم. آسان بودن این ۳تایی ها باعث می‌شود مقدار Loss نزدیک 0 بماند و الگوریتم وزن‌ها را آپدیت نکند در نتیجه چیزی یاد نگیرد. دلیل اینکه عملکرد خوب بود زیاد بودن تعداد این ۳تایی ها بود. اگر میخواستیم در زمان کمتری به نتیجه مطلوب برسیم حتما باید ۳تایی‌های با چالش بیشتری ایجاد می‌کردیم. یکی از روش‌های ایجاد ۳تایی چالشی محاسبه distanceها و انتخاب کمترین d(A,N) و بیشترین d(A,P) است. با این کار مدل ۳تایی هایی را می‌بیند که احتمال اشتباه در آنها بسیار زیاد است و با اشتباه کردن وزن‌های خود را به سمتی می‌برد که این اشتباه دیگر تکرار نشود.

۳ سوال سوم

یک مدل غیر خطی به نام ResNet11 Bottleneck ایجاد می‌کنیم که دارای عمق (Depth) 11 می‌باشد.



این مدل دارای 298هزار پارامتر است.

الف) خیر مناسب نیست زیرا کلاس‌های این دیتاست Balance نیستند و حتی اگر تمامی ورودی‌های کلاس 3 را اشتباه پیشینی کنیم دقت همچنان بالا خواهد بود. اما اشتباه پیشبینی کردن یک کلاس مخصوصا در مسئله پزشکی می‌توان فاجعه آفرین باشد. زیرا بیمار ممکن است درمان آن بیماری پوستی را شروع نکند.

ب) در هر قسمت از سوال مقادیر f1-score، recall، precision و support محاسبه شده است و در گزارش نیز آورده شده است. در این قسمت مفهوم هر یک را بررسی می‌کنیم:

Precision: نسبت True Positive به مجموع True Positive و False Positive است. در واقع نسبت True Positive به تمامی پیشبینی‌های Positive. مفهوم آن این است که مدل در پیشبینی‌هایی که گفته Positive است چقدر دقیق بوده است. این معیار برای زمانی مناسب است که هزینه False Positive زیاد است. مثلا نباید ایمیلی که اسپم نیست را اسپم پیشبینی کنیم.

Recall: نسبت True Positive به مجموع حالت‌هایی که واقعا True هستند می‌گویند. در واقع بیانگر این است که مدل ما چه تعداد از داده‌هایی را که واقعا Positive هستند را درست Positive پیشبینی کرده است. این معیار برای زمانی مناسب است که که هزینه False Negative زیاد است. مثلا اگر یک فرد بیمار را سالم تشخیص بدهیم بسیار بد است.

F1 Score: دو معیار Precision و Recall را با هم ترکیب می‌کند:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

این معیار زمانی مناسب است که می‌خواهیم یک Balance میان دو معیار بالا برقرار کنیم. همچنین این معیار برای زمانی که که توزیع داده‌ها در کلاس‌ها نا برابر است مناسب است.

بر اساس نتایج مدل عملکرد خوبی نداشته و نتوانسته دقت خوبی روی کلاس‌هایی که دیتای کمی داشتند داشته باشد. دقت تنها روی کلاس 5 که تعداد دیتای زیادی دارد خوب است.

Classification Report Train

	precision	recall	f1-score	support
c0	0.85	0.15	0.25	228
c1	0.83	0.29	0.43	359
c2	0.54	0.85	0.66	769
c3	0.29	0.19	0.23	80
c4	0.96	0.14	0.24	779
c5	0.84	0.96	0.90	4693
c6	0.87	0.84	0.86	99
accuracy			0.79	7007
macro avg	0.74	0.49	0.51	7007
weighted avg	0.81	0.79	0.74	7007

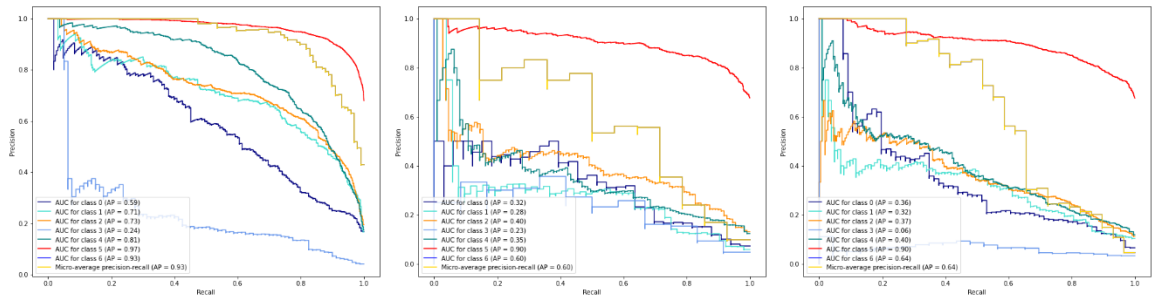
Classification Report Validation

	precision	recall	f1-score	support
c0	0.50	0.03	0.06	33
c1	0.27	0.08	0.12	52
c2	0.38	0.55	0.45	110
c3	0.30	0.25	0.27	12
c4	0.70	0.06	0.12	111
c5	0.78	0.93	0.85	671
c6	0.78	0.50	0.61	14
accuracy			0.71	1003
macro avg	0.53	0.34	0.35	1003
weighted avg	0.69	0.71	0.65	1003

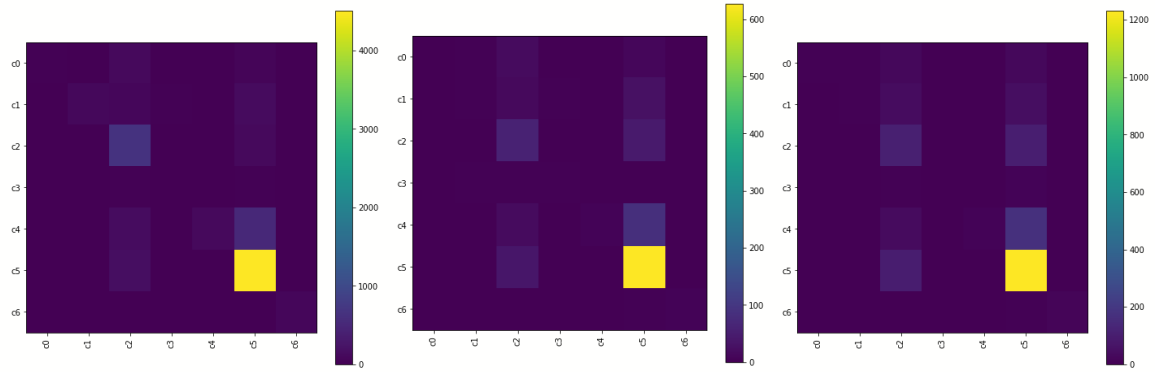
Classification Report Test

	precision	recall	f1-score	support
c0	0.54	0.11	0.18	66
c1	0.38	0.09	0.14	103
c2	0.34	0.50	0.40	220
c3	0.00	0.00	0.00	23
c4	0.70	0.06	0.12	223
c5	0.77	0.92	0.84	1341
c6	0.76	0.55	0.64	29
accuracy			0.69	2005
macro avg	0.50	0.32	0.33	2005
weighted avg	0.68	0.69	0.64	2005

پ) این نمودار مقادیر مختلف Precision و Recall را به ازای Threshold های متفاوت در یک فضای دو بعدی رسم می‌کند. چپ برای Train وسط Validation راست Test



ت) این معیار هم برای تمامی قسمت‌های این سوال محاسبه و رسم شده است. در این قسمت مفهوم آن در این مسئله را بررسی خواهیم کرد: در واقع یک ماتریس N*N است (N تعداد کلاس) و هر درایه مانند i,j از این ماتریس بیان می‌کند که چه تعداد از داده‌های دسته i به اشتباه به عنوان دسته j تشخیص داده شده است. هر چه طیف رنگی بالاتر می‌رود نشان این است که این دو کلاس با هم بیشتر به اشتباه گرفته شده اند.



طبق این نمودار مدل عملکرد خوبی نداشته و روی دو تا کلاس بسیار اشتباه داشته و آن‌ها را کامل قاطی کرده و دچار Confuse بسیار زیادی شده است.

ث) معیار Recall برای این مسئله مناسب است زیرا نمی‌خواهیم کسی که یک بیماری دارد را یک بیماری دیگر معرفی کنیم و این که اشتباه بیماری بدی برای او پیشبینی کنیم مشکل حادی ایجاد نمی‌کند. معیار F1 که ترکیب Recall و Precision هست بهترین معیار است زیرا ویژگی هر دو متریک را دارد و به خوبی می‌توان عملکرد مدل را بررسی کرد.

ج) از Data Augmentation و Resampling استفاده می‌کنیم تا هم داده بیشتری تولید کنیم هم توزیع داده‌ها را یکنواخت تر کنیم. در Resampling با Duplicate کردن داده‌ها تعداد آنها را یکسان می‌کنیم. آموزش و نتایج این قسمت به همراه نمودارهای Confusion و AUC و تمامی متریک‌ها داخل نوتبوک موجود است. همانطور که در تصاویر درون نوتبوک مشاهده می‌شود مدل توانسته عملکرد بهتری داشته باشد و متریک Accuracy نیز منطقی تر شده زیرا تعداد داده‌ها Balance است.

۴ سوال چهارم

ابتدا مدل پایه را می‌سازیم. نرخ یادگیری در این مدل برابر 0.01 است.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 512)               401920

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

این مدل را در 50 ایپاک آموزش می‌دهیم. و در آخر روی داده تست ارزیابی (Evaluate) می‌کنیم.

```
Epoch 45/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.3014 - accuracy: 0.8913 - val_loss: 0.5670 - val_accuracy: 0.8476
Epoch 46/50
1500/1500 [==============================] - 5s 3ms/step - loss: 0.3002 - accuracy: 0.8919 - val_loss: 0.5460 - val_accuracy: 0.8612
Epoch 47/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2983 - accuracy: 0.8922 - val_loss: 0.5732 - val_accuracy: 0.8565
Epoch 48/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.3002 - accuracy: 0.8911 - val_loss: 0.5453 - val_accuracy: 0.8602
Epoch 49/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2943 - accuracy: 0.8934 - val_loss: 0.5507 - val_accuracy: 0.8592
Epoch 50/50
1500/1500 [==============================] - 5s 3ms/step - loss: 0.3005 - accuracy: 0.8930 - val_loss: 0.6382 - val_accuracy: 0.8599
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.7085 - accuracy: 0.8490
[0.7084887623786926, 0.8489999771118164]
```

- Train loss: 0.3005 Train acc: 0.8930
- Val loss: 0.6382 Val acc: 0.8599
- Test loss: 0.7085 Test acc: 0.8490

از ابزار Keras Tuner برای تعریف هایپرپارامترها به صورت زیر استفاده می‌کنیم:

```
hidden_units = hp.Int(
    name='hidden_units',
    min_value=16,
    max_value=512,
    step=16
)
learning_rate = hp.Choice(
    'learning_rate',
    values=[0.0001, 0.0005, 0.001, 0.005, 0.01]
)
```

در این ابزار Tunerهای مختلفی وجود دارد که هر یک با الگوریتم مخصوص خود فضای جست‌وجو را پیمایش می‌کنند و سعی در پیدا کردن بهترین هایپرپارامترها دارند.

- Base Tuner: این یک کلاس Parent برای سایر تیونرها است و باقی تیونرها از آن ارث می‌برند. وظیفه آن Build، Train، Evaluation و ذخیره سازی مدل است.
- Random Search: یک راه برای پیدا کردن بهترین هایپرپارامترها این است که تمامی ترکیب های ممکن را امتحان کنیم. به این کار Grid Search نیز می‌گویند. اما مشکل این روش این است که تعداد ترکیب‌ها با افزایش هایپرپارامترها به صورت نمایی افزایش می‌یابد. امتحان کردن هر یک از این ترکیب‌ها بسیار طول می‌کشد. جست‌وجوی تصادفی به ما کمک می‌کند تا زمانی کمتری نسبت به Grid Search صرف کنیم. مشکل جست‌وجوی تصادفی این است که تضمینی برای پیدا کردن بهترین هایپرپارامتر ندارد. اما نتیجه نزدیک به بهترین است.
- Hyperband: این روش سعی در رفع کردن یکی از مشکلات Random Search دارد. مشکل موجود در Random Search به این صورت است که ما هایپرپارامترهایی را امتحان می‌کنیم که واضح است نتیجه بدی خواهند داشت و ما زمان زیادی را برای آموزش ور ارزیابی این شبکه‌ی بد سپری می‌کنیم. روش Hyperband پس از انتخاب تصادفی یک مجموعه از هایپرپارامترها به جای اینکه شبکه را به صورت کامل آموزش دهد و در ادامه ارزیابی کند، مدل را تنها برای چند ایپاک آموزش می‌دهد و بهترین نتایج را در این چند ایپاک به مرحله‌بعدی آموزش می‌فرستد. این کار مکررا انجام می‌شود تا در نهایت آموزش کامل بر روی نمایندگان نهایی صورت گیرد.
- Bayesian Optimization: این روش سعی می‌کند مشکل مشترک در Random Search و Hyperband را حل کند. مشکل این است: تمامی هایپرپارامترها به صورت Random با هم ترکیب می‌شوند. این کار کمک می‌کند که فضای حالت را جست‌وجو کنیم اما تضمینی برای پیدا کردن بهترین و بهینه‌ترین هایپرپارامترها ندارد. در این روش به‌جای اینکه تمامی حالات به صورت رندوم باشند، چند حالت ابتدایی را رندوم در نظر می‌گیرد سپس یا توجه به نتایج این حالات ابتدایی، بهترین حالت ممکن بعدی را می‌سازد. در واقع در این روش از تاریخچه‌ی حالت‌های قبلی برای ساختن حالت جدید استفاده می‌شود. این کار تا زمانی ادامه می‌یابد که یا به بهترین جواب برسیم یا به ماکس تعداد آزمایش‌ها برسیم. در این روش با آرگومان beta مشخص می‌کنیم که باید به کاوش بپردازیم (Explore) یا از دانسته‌های قبلی استفاده کنیم (Exploit).
- Sklearn Tuner: این تیونر مخصوص مدل‌هایی است که با ابزار Sci-Kit Learn ایجاد شده اند. ما در این سوال مدل را با Keras ساخته ایم.

ما در این سوال از روش دوم یعنی Hyperband استفاده کردیم زیرا منابع سخت‌افزاری و زمانی محدودی داشتیم و نمی‌خواستیم زمان زیادی را صرف آموزش مدل‌های معیوب کنیم. اما اگر در منابع و زمان محدودیتی نداشتیم روش Grid Search گزینه‌ی بهتری بود زیرا تمامی حالات به صورت کامل بررسی می‌شد و مطمئن بودیم بهترین حالت انتخاب شده است. دلیل آنکه از روش بیزین استفاده نکردیم این است هایپرپارامترهای موجود Orthogonal نیستند و همچنین فضای حالت بزرگی نداریم.

فضای جست‌وجوی مسئله به صورت زیر است:

```
Search space summary
Default search space size: 2
hidden_units (Int)
{'default': None, 'conditions': [], 'min_value': 16, 'max_value': 512, 'step': 16, 'sampling': None}
learning_rate (Choice)
{'default': 0.0001, 'conditions': [], 'values': [0.0001, 0.0005, 0.001, 0.005, 0.01], 'ordered': True}
```

جست‌وجو را آغاز می‌کنیم. نتایج به صورت زیر است:

```
Trial 90 Complete [00h 04m 10s]
val_accuracy: 0.8945833444595337

Best val_accuracy So Far: 0.9020000100135803
Total elapsed time: 01h 33m 28s
INFO:tensorflow:Oracle triggered exit
```

```
Results summary
Results in ./tuner_results/untitled_project
Showing 10 best trials
Objective(name='val_accuracy', direction='max')
Trial summary
Hyperparameters:
hidden_units: 432
learning_rate: 0.0005
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 2
tuner/round: 2
tuner/trial_id: bb38582319ee5054c42e3dfa5603bdf5
Score: 0.9020000100135803
```

بهترین تعداد نورون برای لایه میانی عدد 432 شده و بهترین نرخ آموزش 0.0005 شده است. مدل حاصل شده را آموزش می‌دهیم و سپس ارزیابی می‌کنیم. نتایج به صورت زیر است.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 784)               0

 dense_4 (Dense)             (None, 432)               339120

 dense_5 (Dense)             (None, 10)                4330

=================================================================
Total params: 343,450
Trainable params: 343,450
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 45/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0680 - accuracy: 0.9750 - val_loss: 0.4429 - val_accuracy: 0.8942
Epoch 46/50
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0658 - accuracy: 0.9756 - val_loss: 0.4593 - val_accuracy: 0.8904
Epoch 47/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0660 - accuracy: 0.9757 - val_loss: 0.4428 - val_accuracy: 0.8982
Epoch 48/50
1500/1500 [==============================] - 5s 3ms/step - loss: 0.0617 - accuracy: 0.9767 - val_loss: 0.4585 - val_accuracy: 0.8960
Epoch 49/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0604 - accuracy: 0.9781 - val_loss: 0.4735 - val_accuracy: 0.8957
Epoch 50/50
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0628 - accuracy: 0.9773 - val_loss: 0.4566 - val_accuracy: 0.8947
```

Train evaluation:

```
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1588 - accuracy: 0.9456
[0.15877357125282288, 0.9456166625022888]
```

Test evaluation:

```
313/313 [==============================] - 1s 3ms/step - loss: 0.3823 - accuracy: 0.8907
[0.38228559494018555, 0.8906999826431274]
```

- Train loss: 0.1588 Train acc: 0.9456
- Test loss: 0.3823 Test acc: 0.8907

همانطور که مشاهده می‌شود دقت روی داده تست حدود ۵ درصد نسبت به مدل پایه افزایش یافته است. دلیل این بهبود این است که مدل در حالت پایه دارای 512 نورون در لایه میانی و نرخ آموزش 0.01 بود که این دو مقدار ظرفیت شبکه را بالا برده بودند و باعث می‌شدند شبکه در همان Epochهای ابتدایی Converge شود و از اواسط آموزش دیگر بهبودی نداشته باشیم (Underfit)

با کاهش نرخ آموزش به 0.0005 مشکل Underfit را حل کردیم و همچنین به ازای این نرخ آموزش تعداد 512 نورون می‌توانست منجر به Overfit شود (۵۰ اپک) بنابراین Keras Tuner تعداد 432 نورون را انتخاب کرد که هم مشکل Underfit و هم مشکل Overfit حل شود. معیار Keras Tuner دقت روی داده Validation بود.