امیرحسین احمدی ۹۷۵۲۲۲۹۲ – تمرین چهاردهم

۱) سوال اول: در این بخش همانطور که مشاهده میکنید ۵ کلمه ی مختلف را به Embedding داده و برای هر کدا ۱۰ کلمه مشابه با آن به همراه درصد تشابه های آن برگردانده شده است که به ترتیب آن ها را بررسی میکنیم:

۱. John : خب از آنجایی که john یک اسم است اکثر کلمات مشابه آن اسم هستند. نکته جالب وجود .J است که مخفف john است و همچنین وجود sir که نشان میدهد جنسیت جان به خوبی در Embedding اعمال شده

۲. iran : برای ایران اکثر کلمات کشور هستند به خصوص کشورهای نزدیک ایران. همچنین کلمات مربوط به غنی سازی هسته ای نیز به دلیل این که بیشتر اخبار ایران در این زمینه است دیده میشود. حضور احمدی نژاد نیز به عنوان رییس جمهور سابق ایران احتمالن به دلیل قدیمی بودن دیتای مربوط به Embedding است. همچنین کلمات دیگری مانند تهران و ایرانی و ایرانی ها نیز به چشم میخورند.

۳. laptop : برای لپتاپ اکثر کلمات مربوط به کالا های دیجیتال (کامپیوتر، موبایل و ...) میشود. کلمه portable به دلیل قابل حمل بودن لپتاپ و اشاره به آن در متون آورده شده است.

۴. notebook : به غیر از کلمات diary، journal، column و ... که مستقیما به نوتبوک مربوط میشوند، کلمه atlanta را میبینیم که احتمالا یک برند دفتر یادداشت است. همچنین کلمات laptop و pc نیز به چشم میخورند با اینکه ربط مستقیمی به notebook ندارند که احتمالا میتواند به دلیل نوتبوک های جوپیتر یا مثلا نرم افزار های note در کامپیوتر ها باشد که به نظر نباید انقدر شباهت آن ها با نوتبوک زیاد میبود.

۵. Mother : اکثر کلمات به نسبت های فامیلی اشاره دارند که اکثرن نیز مونث هستند. کلمات her و she نیز احتمالا به دلیل مونث بودن mother در کلمات مشابه حضور دارند.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
john	(william, 0.67)	(george, 0.66)	(james, 0.65)	(thomas, 0.64)	(smith, 0.6)	(j., 0.6)	(edward, 0.6)	(paul, 0.6)	(richard, 0.6)	(sir, 0.59)
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)
laptop	(laptops, 0.83)	(computers, 0.7)	(phones, 0.62)	(computer, 0.62)	(portable, 0.61)	(notebooks, 0.6)	(cellphones, 0.59)	(pcs, 0.58)	(ipods, 0.57)	(desktop, 0.57)
notebook	(notebooks, 0.62)	(laptop, 0.5)	(laptops, 0.45)	(pc, 0.45)	(fbn, 0.44)	(atlanta, 0.44)	(notes, 0.44)	(column, 0.43)	(journal, 0.43)	(diary, 0.42)
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 دوم:

۱. کلمه john و alex هر دو اسم هستند و طبیعتن فاصله کمتری دارند و ربط خاصی به ایران ندارند که فاصله آن ها را بسیار زیاد نشان میدهد.

۲. مک به لپتاپ های اپل گفته میشود و فاصله کمتری دارند تا فاصله دفتر خاطرات و لپتاپ. البته فاصله بسیار نزدیک است که هم میتواند به دلیل مشکلی که در قسمت قبل مربوط به نزدیک بودن لپتاپ و نوتبوک باشد و هم به دلیل خوب تشخیص ندادن مک به عنوان نوعی لپتاپ.
۳. در اینجا نیز با این که فاصله لیوان تا آپ کمتر از فاصله آن تا پا است، ولی باز فاصله نزدیکی دارند که میتواند به دلیل معانی دیگر گلس و نزدیکی آن با مفاهیم دیگر باشد که آب را زیاد نزدیک تشخیص نداده است.

	Near Word	Far Word
john	(alex, 0.68)	(iran, 0.95)
laptop	(mac, 0.69)	(diary, 0.74)
glass	(water, 0.6)	(foot, 0.72)

سوال سوم: در این سوال فاصله اول برای زمانی است که کلمه اول و سوم تاثیر مثبت و کلمه دوم تاثیر منفی داشت باشد و فاصله دوم برای زمانی است که کلمه اول دوم تاثیر مثبت و کلمه سوم تاثیر منفی داشته باشد. حال زمانی که بخواهیم کلمه اول متناسب با کلمه دوم باشد و کلمه سوم با چهارم باید تاثیر مثبت اول و سوم را بیشتر کنیم و وقتی برعکس باشد باید تاثیر اول و دوم مثبت باشد.

۱. همانطور که گفته شد در اینجا باید فاصله اول را بگیریم که تاثیر king و woman را بیتشر میکند که به خوبی به queen میرسیم. ولی اگر فاصله دوم را ببینیم با بالا رفتن تاثیر king و man مقامات سلطنتی مرد بیشتر شده اند که به درد ما نمیخورد.
۲. در این جا هم از آنجایی که کلمه اول با سوم متناسب است با مثبت کردن کلمه اول و دوم به نتیجه دلخواه actress رسیده ایم.
۳. با توجه به این که docter جنسیت ندارد احتمالا به جواب خوبی نمیرسیم. در فاصله اول که docter را با he متناسب گرفته، از آنجایی که معمولا پرستارها خانم هستند(یا در جامعه چنین تصور میشود) به کلمه پرستار رسیده ایم ولی در فاصله دوم کلمه he تاثیر مثبت خاصی نداشته و معادل های docter را میبینیم.
۴. در اینجا نیز homemaker جنسیت ندارد. زمانی که homemaker را با he معادل گرفتیم و تاثیر she را مثبت کردیم به کلمه housewife میرسیم که به معنای خانم خانه دار است ولی خب با این حال با homemaker تفاوت دارد اما احتمالا نزدیکترین کلمه میتوانسته باشد. در

فاصله دوم نیز کلمات تناسب خاصی را رعایت نکردند و بیشتر به سن homemaker ها اشاره دارند.

۵. در اینجا نیز از آنجایی که football جنسیت ندارد، در هر دو فاصله کلمات فقط معادل های فوتبال یا ورزش های دیگر هستند و تناسب خاصی را رعایت نمیکنند.

```
[33] show_DataFrame(first_distances, [f'Word {i + 1}' for i in range(10)])
```

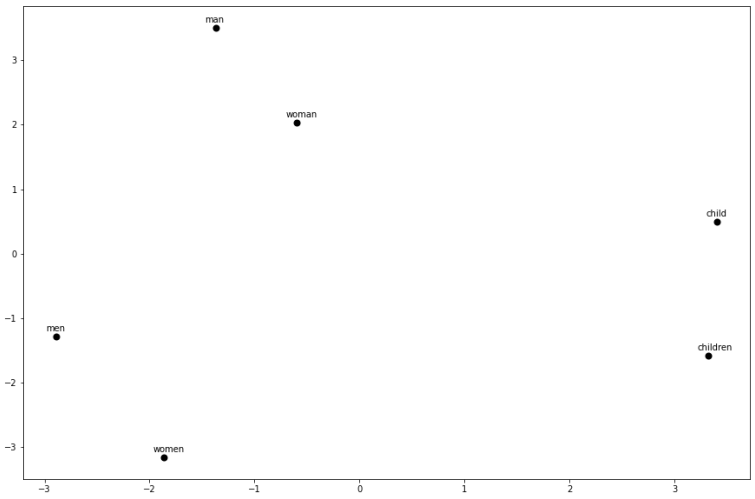
	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
('king', 'woman', 'man')	(queen, 0.7)	(princess, 0.61)	(monarch, 0.59)	(throne, 0.58)	(prince, 0.58)	(elizabeth, 0.55)	(daughter, 0.54)	(kingdom, 0.53)	(mother, 0.52)	(crown, 0.52)
('actor', 'girl', 'boy')	(actress, 0.87)	(starring, 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)

```
[34] show_DataFrame(second_distances, [f'Word {i + 1}' for i in range(10)])
```

	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)

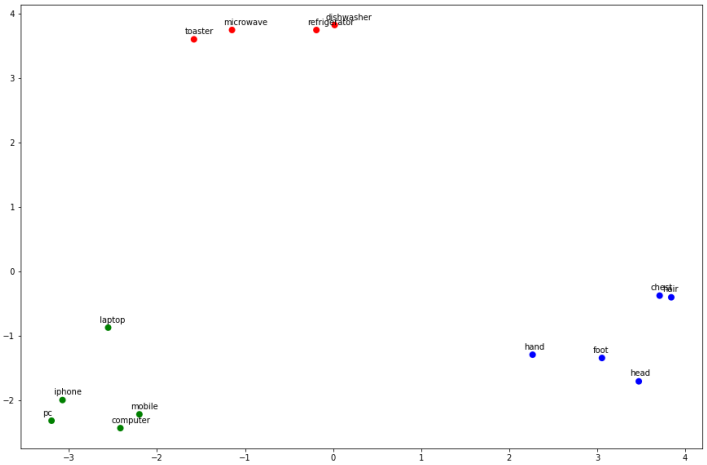
سوال چهارم: همانطور که میبینید کلمات شبیه تر به هم بسیار نزدیکتر به هم هستند. همچین میتوان دید که با تغییر جنسیت مذکر به مونث کلمات همه به سمت بالا چپ میروند. همچنین در بخش پایین سمت چپ تصویر مشاهده میشود که که با جوان تر شدن کلمه به سمت پایین چپ حرکت میکند.

سوال پنجم: در این سوال ۳ کلمه man، woman، child و جمع آن ها روی نمودار برده شده اند. همانطور که مشاهده میکنید تمامی کلمات جمع در قسمت پایین چپ مفرشان هستند(البته نه با زوایای یکسان) که نشان میدهد Embedding توانست تفاوت جمع و مفرد را تشخیص دهد.

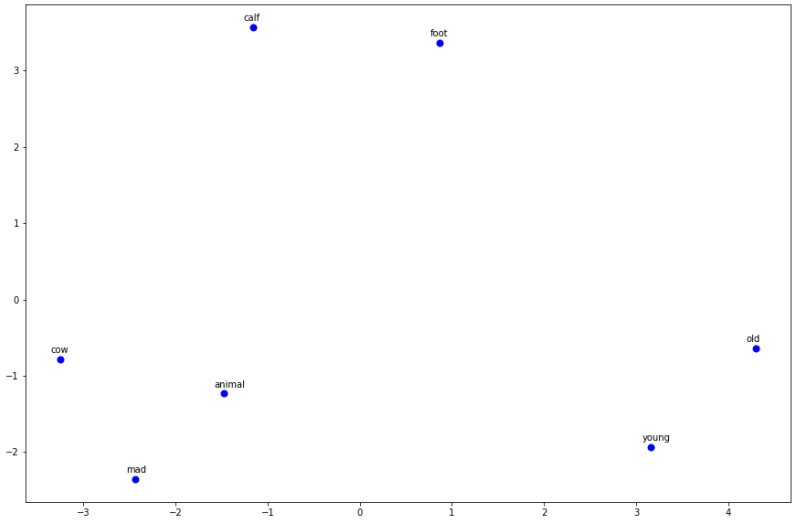


سوال ششم: در مورد سه دسته کلی رنگ، حیوان و کشور که به وضوح از یکدیگر جدا شده اند. در کشور ها میتوان اینطور بررسی کرد که ایران و عراق که شرایط کشوری مشابهی دارند بیشتر به هم نزدیکند و امارات و قطر که شبیه ترند، نزدیک ترند. در مورد بقیه دسته ها ولی الگوی خاصی به چشم نمیخورد و بیشتر فقط تفاوت بین دسته ها مشهود است.

سوال هفتم: در اینجا سه گروه لوازم الکترونیکی، لوازم آشپزخانه و اعضای بدن را داریم که همانطور که میبینید در Embedding به خوبی از هم جدا شده و در دسته های جدا قرار گرفته اند.



سوال هشتم: جواب درست برای این سوال گوساله یا calf است که در امبدینگ تشخیص داده نشده. با توجه به عکس زیر که فاصله تقریبی کلمات در دو بعد است، میبینیم که mad که به عنوان نزدیک ترین کلمه انتخاب شده، فاصله کمتری تا cow و young دارد که به نظر من میتواند به دلیل معنی دوم calf که ماهیچه ساق پا است باشد. همانطور که در تصویر هم میبینید calf به foot نزدیک تر است تا animal که میتواند نشان از کاربرد بیشتر معنی دوم باشد. ولی با این حال میبینیم که الگوی بین calf و cow همانند young و old است.



۲) برای این سوال من تغییراتی در ساختار مدل از پیش تعریف شده دادم که توضیح میدهم. در ابتدا مدل به این صورت بود که سه مدل embedding_pred_net در کنار هم قرار میداد (concat میکرد) و سپس تابع لاس را روی خروجی این سه اعمال میکرد. من با اضافه کردن یک لایه Lambda به انتهای سه مدل embedding_pred_net لاس را در همان لایه حساب میکنم و خروجی میدهم و سپس در آخر لاس را که indentity_loss نام دارد برابر با میانگین لاس حساب شده برای هر تست قرار میدهم. خوبی استفاده از Lambda این بود که به جای کانکت شده ی خروجی های سه مدل امبدینگ به من یک لیست میداد که راحت تر جدا میشدند.

```
anchor_in = Input((28,28,1, ), name='input-anchor')
positive_in = Input((28,28,1, ), name='input-positive')
negative_in = Input((28,28,1, ), name='input-negative')

# 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)

adam_optim = Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999)
model.compile(loss=identity_loss, optimizer=adam_optim)

model.summary()
```

در تابع triplet_loss برای لایه ی Lambda، ابتدا سه لیست مربوط به anchor، positive و negative را جدا کرده، سپس d(a, n) و d(a, p) را حساب میکند و در آخر triplet loss را از آن ها

بدست میاورم و خروجی میدهم. نکته قابل توجه این است که برای مشتق پذیر بودن تمام توابع موجود در triplet_loss و identity_loss، آن ها را با استفاده از توابع backend keras پیاده سازی کرده ام.

```
def identity_loss(y_true, y_pred):

    return K.mean(y_pred)

def triplet_loss(y_pred, alpha=0.4):

    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
```

برای شبکه پایه مان میخواهیم برای embedding عکس ها استفاده کنیم، از شبکه زیر استفاده کرده ایم، که با استفاده از چند لایه کانولوشنی تعدادی ویژگی از عکس تا بدست آورده و سپس آن را با استفاده از تعدادی لایه dense در نهایت به ۱۰ نورون خروجی میرسانیم که امدینگ تصویر ما به حساب میاید. در ابتدا مدل مقداری اورفیت میشد به همین دلیل تعدادی لایه dropout و MaxPooling اضافه کردیم تا کمی از آن پیشگیری کنیم.

```
def embedding_pred_net(dim):
    """
    embedding predictions: Base network to be shared
    """

    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28,28,1)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(10))

    model.summary()
    return model
```

برای ترین مدل X_train و X_test ساخته شده اصلا مناسب نبود. در ابتدا X_train یک numpy array با شیپ (180000, 3, 784) بود که در واقع ۱۸۰۰۰۰ تست که در آن ۳ تایی های anchor، positive و negative موجود بودند که عکس ها فلت شده بودند. ولی برای ورودی مدل ساخته شده ما

بایستی یک لیست پایتون که خانه های آن به ترتیب با تست های anchor، positive و negative پر شده باشد. همچنین ورودی های مدل تنسور های ۲۸ در ۲۸ میگیرد و نیازی به فلت بودن نیست، به همین دلیل X_train و X_test را به صورت زیر تغییر دادیم.

```
X_train = X_train.transpose(1, 0, 2).reshape((3, 180000, 28, 28))
X_train = [X_train[0], X_train[1], X_train[2]]

X_test = X_test.transpose(1, 0, 2).reshape((3, 45000, 28, 28))
X_test = [X_test[0], X_test[1], X_test[2]]

(len(X_train), X_train[0].shape), (len(X_test), X_test[0].shape)
```

```
((3, (180000, 28, 28)), (3, (45000, 28, 28)))
```

حال مدل را به راحتی fit میکنیم. فقط از آن جایی که در triplet_loss لیبل نداریم، به جای لیبل ها، تنسور های صفر میدهیم که فقط به مشکل نخوریم، وگرنه تاثیری در روند آموزش ندارند.

```
model.fit(X_train, np.zeros(180000), epochs=10, batch_size=256, validation_data=[X_test, np.zeros(45000)])
```

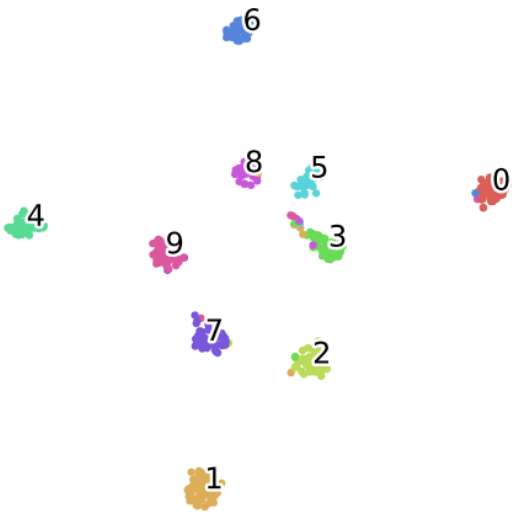
```
Epoch 1/10
704/704 [==============================] - 47s 53ms/step - loss: 0.0924 - val_loss: 0.0374
Epoch 2/10
704/704 [==============================] - 38s 54ms/step - loss: 0.0248 - val_loss: 0.0227
Epoch 3/10
704/704 [==============================] - 36s 51ms/step - loss: 0.0136 - val_loss: 0.0189
Epoch 4/10
704/704 [==============================] - 38s 54ms/step - loss: 0.0084 - val_loss: 0.0141
Epoch 5/10
704/704 [==============================] - 35s 50ms/step - loss: 0.0060 - val_loss: 0.0150
Epoch 6/10
704/704 [==============================] - 36s 50ms/step - loss: 0.0047 - val_loss: 0.0122
Epoch 7/10
704/704 [==============================] - 38s 53ms/step - loss: 0.0038 - val_loss: 0.0126
Epoch 8/10
704/704 [==============================] - 38s 53ms/step - loss: 0.0030 - val_loss: 0.0099
Epoch 9/10
704/704 [==============================] - 38s 53ms/step - loss: 0.0026 - val_loss: 0.0118
Epoch 10/10
704/704 [==============================] - 35s 50ms/step - loss: 0.0020 - val_loss: 0.0107
<keras.callbacks.History at 0x7f4a0a03add0>
```

همانطور که میبینید مدل به خوبی ترین شده و همچنین loss و val_loss نزدیک به هم هستند که نشان میدهد اور فیت نشده ایم.

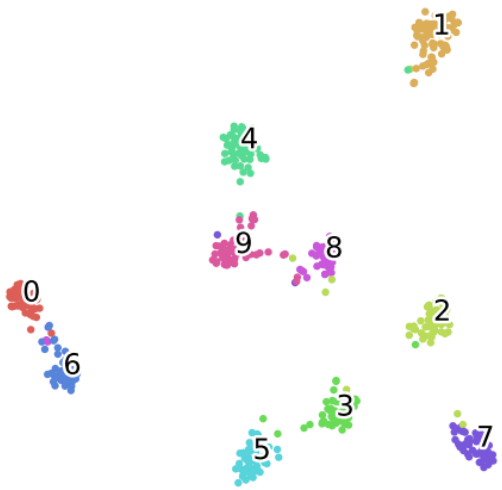
در ادامه با استفاده تعدادی از x_train و x_test اولیه که شامل عکس های ورودی هستند، امبدینگ های بدست آمده را تست میکنیم. نکته ای که هست، ما برای این بخش مدل جدیدی با استفاده از لایه های anchor_in و anchor_embedding میسازیم که وزن های آن در مدل قبلی ترین شده و دیگر نیاز به ترین ندارند، تا بتوانیم خروجی های امبدینگ ها را به طور مستقیم داشته باشیم. در نهایت با t-SNE آن را به دو بعد میبریم و نمایش میدهیم.

```
trained_model = Model(inputs=anchor_in, outputs=anchor_embedding)

X_train_trm = trained_model.predict(x_train[:512])
X_test_trm = trained_model.predict(x_test[:512])

tsne = TSNE()

train_tsne_embeds = tsne.fit_transform(X_train_trm)
scatter(train_tsne_embeds, y_train[:512], "Training Data After TNN")

eval_tsne_embeds = tsne.fit_transform(X_test_trm)
scatter(eval_tsne_embeds, y_test[:512], "Validation Data After TNN")
```

Training Data After TNN



Validation Data After TNN



در کد فعلی، سه تایی تا به این صورت رندوم انتخاب میشوند، یعنی به ازای هر لیبل یک anchor انتخاب کرده و positive و negative را از بین تصاویر با لیبل یکسان و تصاویر با لیبل متفاوت به صورت رندوم انتخاب میکنیم. مشکل الگوریتم فعلی این است که ممکن است هر دفعه سه تایی های راحتی را انتخاب کنیم که anchor بسیار به positive شبیه و بسیار دور از negative باشد و در این صورت مدل خوب ترین نشده و اور فیت میشود. برای این کار بهتر است که سه تایی ها را به صورت آنلاین انتخاب کنیم، یعنی هر دفعه برا اساس فاصله فعلی عکس های موجود، یک positive و negative سخت انتخاب کنیم و مدل را به چالش بکشیم. البته اگر این را برای همه ی سه تایی های موجود بخواهیم امتحان کنیم بسیار طول میکشد، پس بهتر است هر دفعه یک batch انتخاب کرده و از بین آن ها سه تایی ها را انتخاب کنیم.
منبع: https://github.com/Ekeany/Siamese-Network-with-Triplet-Loss/blob/master/MachinePart1.ipynb

۳) برای این سوال یک مدل ResNet با عمق ۱۴ ایجاد کرده ایم که با استفاده از keras functional api ساخته شده است. گراف آن در نوت بوک موجود است که به دلیل بزرگ بودن در اینجا آورده نشده است.

الف) خیر مناسب نیست، از آن جایی که دیتا های این سوال نامتوازن هستند اگر مثلا هیچ وقت جواب را ۳ یا ۶ پیشبینی نکنیم، با توجه به کم بودن دیتای آن ها، دقت ما تغییر زیادی نمیکند، به همین دلیل accuracy معیار خوبی به حساب نمیاید.

ب) تمام معیار های گفته شده برای داده های Train، Test و Validation محاسبه شده که در زیر میبینیم.

CR train

	precision	recall	f1-score	support
0	0.80	0.90	0.85	228
1	0.98	0.39	0.56	359
2	0.65	0.97	0.78	769
3	1.00	0.21	0.35	80
4	0.47	0.99	0.64	779
5	1.00	0.79	0.88	4693
6	0.99	0.84	0.91	99
accuracy			0.81	7007
macro avg	0.84	0.73	0.71	7007
weighted avg	0.89	0.81	0.82	7007

CR validation

	precision	recall	f1-score	support
0	0.32	0.21	0.25	33
1	0.60	0.06	0.11	52
2	0.30	0.57	0.40	110
3	0.00	0.00	0.00	12
4	0.26	0.71	0.38	111
5	0.92	0.63	0.75	671
6	1.00	0.36	0.53	14
accuracy			0.58	1003
macro avg	0.49	0.36	0.35	1003
weighted avg	0.73	0.58	0.61	1003

CR test

	precision	recall	f1-score	support
0	0.35	0.27	0.31	66
1	0.86	0.06	0.11	103
2	0.31	0.51	0.38	220
3	0.00	0.00	0.00	23
4	0.27	0.72	0.39	223
5	0.92	0.66	0.77	1341
6	0.80	0.55	0.65	29
accuracy			0.60	2005
macro avg	0.50	0.40	0.37	2005
weighted avg	0.75	0.60	0.63	2005

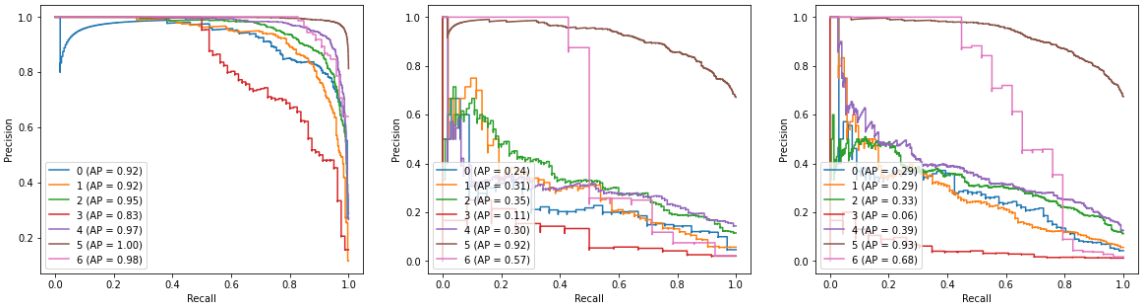
فرمول پرسیژن در مسائل دو کلاسه برابر با TP / (TP + FP) است. در واقع با کاهش false positive مقدار آن افزایش میابد و معیار خوبیست برای زمانی که میخواهیم false positive را کاهش دهیم. مثلا در مسئله تشخیص دادن ایمیل های اسپم، میتوان اندازه گرفت که تا چه اندازه زمانی که یک ایمیل را اسپم تشخیص داده ایم واقعا اسپم بوده است؟

فرمول recall برابر با TP / (TP + FN) است و در مسئله اسپم، معادل این است که ما چند درصد اسپم ها را درست تشخیص داده ایم. به همین دلیل وقتی نیاز داریم که false negative را کم کنیم recall میتواند معیار خوبی باشد.

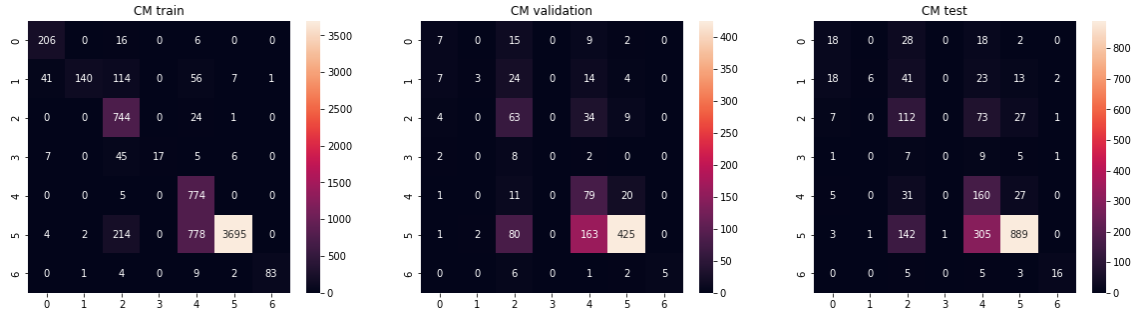
F1-score ترکیبی از دو معیار بالاست و زمانی به درد میخورد که false negative و false positive را در تعادلی از یکدیگر نگه داریم.

همانطور که میبینین مدل عملکرد خوبی به خصوص روی لیبل های با دیتای کم نداشته. البته روی لیبل های با داده بیشتر بهتر عمل کرده است.

پ) نمودار AUC در واقع مقدار Precision و Recall را به ازای Threshold های مختلف نسبت به هم رسم میکند که میتوانید شکل آن را برای داده های Train، Test و Validation در زیر ببینید.



ت) confusion_matrix نمودار بسیار خوبیست که در مسائل دسته بندی به ما کمک میکند تا بهتر بتوانیم نقاط ضعف و قوت مدل را در قالب یک جدول n*n برای مسئله های با n کلاس ببینیم. در واقع برای لیبل i خانه ی (i,i) نشان دهنده مقدار TP، جمع خانه های (i,j) برای j های مخالف با i نشان دهنده FN، جمع خانه های (j, i) برای j های مخالف i نشان دهنده FP و جمع بقیه خانه ها برابر با TN است.



همانطور که در بالا میبینیم، مدل عمل کرد خیلی خوبی نداشته ولی روی لیبل ۵ توانسته TP خوبی بدست بیاود ولی از بقیه مقادیر معلوم است مه مدل روی لیبل ۵ اورفیت شده است.

ث) از آنجایی که این دیتا برای کارهای پزشکی است و بسیار حساس است، اگر بگویم که یک بیمار، بیماریی را ندارد و آن را داشته باشد هیچ وقت بیماری را تشخیص نمیدهیم ولی خب اگر بگوییم بیماریی را دارد و آن را نداشته باشد با طی کردن روند درمان اتفاق خاصی برای بیمار نمیوفتد، پس بهتر است تا FN را کم کنیم پس به نظر recall برای ما مناسب باشد.

ج) مدل به نظر به دلیل دیتای کم و همچنین متوازن نبودن داده ها، نتوانسته عملکرد خوبی از خود نشان بدهد، بنابراین احتمالا بهتر است با استفاده از Data Augmentation داده را به طور کلی و همچنین بیشتر برای لیبل هایی که داده های کمی از آن ها داریم زیاد کنیم تا مدل عملکرد بهتری پیدا کند.

منبع: https://github.com/PacktPublishing/Advanced-Deep-Learning-with-Keras/blob/master/chapter2-deep-networks/resnet-cifar10-2.2.1.py

۴) ابتدا مدل اولیه را برای مقایسه با مدلی که در ادامه به آن میرسیم train میکنیم.

```
Epoch 50/50
1500/1500 [==============================] - 5s 4ms/step - loss: 0.2991 - accuracy: 0.8895 - val_loss: 0.6854 - val_accuracy: 0.8475
```

سپس به تابع model_builder پارامتر ورودی hp را اضافه میکنیم که بتوانیم در keras tuner از آن استفاده کنیم. برای لایه دنس با استفاده از hp یک int تعریف کرده و ماکسیموم و مینیموم خواسته شده را به آن میدهیم و همچنین برای learning ریت از بین مقادیر گفته شده یکی را انتخاب میکنیم.

```
def model_builder(hp):
  model = keras.Sequential()
  model.add(keras.layers.Flatten(input_shape=(28, 28)))
  model.add(keras.layers.Dense(units=hp.Int('dense_layer_units', min_value=16, max_value=512, step=16), activation='relu'))
  model.add(keras.layers.Dense(10))
  lr = hp.Choice('learning_rate', values=[.0001, .0005, .001, .005, .01])
  model.compile(optimizer=keras.optimizers.Adam(learning_rate=lr),
                loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                metrics=['accuracy'])

  return model
```

حال تابع model_builder را به tuner مورد نظرمان که در اینجا از BayesianOptimization استفاده کرده ایم میدهیم تا tune شود. آبجکتیومان هم val_accuracy قرار داده تا دقت مدل را بتوانیم بالا ببریم. سپس در ۵۰ ایپوک و سه بار مدل را tune کرده تا هایپر پارامتر های بهینه را بدست آوریم.

```
tuner = kt.BayesianOptimization(hypermodel=model_builder, objective="val_accuracy", max_trials=3)

tuner.search_space_summary()
```

```
Search space summary
Default search space size: 2
dense_layer_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}
```

```
tuner.search(img_train, label_train, epochs=50, validation_split=0.2)
```

```
Trial 3 Complete [00h 04m 26s]
val_accuracy: 0.8702499866485596

Best val_accuracy So Far: 0.8860833048820496
Total elapsed time: 00h 14m 20s
INFO:tensorflow:Oracle triggered exit
```

در زیر نیز میتوانید ۵ نتایج بدست آمده را به ترتیب دقت مشاهده کنید.

```
best_hps = tuner.get_best_hyperparameters(5)
model = model_builder(best_hps[0])

model.summary()
tuner.results_summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 784)               0

 dense_2 (Dense)             (None, 32)                25120

 dense_3 (Dense)             (None, 10)                330

=================================================================
Total params: 25,450
Trainable params: 25,450
Non-trainable params: 0
_________________________________________________________________
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name='val_accuracy', direction='max')
Trial summary
Hyperparameters:
dense_layer_units: 32
learning_rate: 0.001
Score: 0.8860833048820496
Trial summary
Hyperparameters:
dense_layer_units: 288
learning_rate: 0.005
Score: 0.8800833225250244
Trial summary
Hyperparameters:
dense_layer_units: 16
learning_rate: 0.0001
Score: 0.8702499866485596
```

در نهایت مدل را با هایپرپارامتر های جدید آموزش میدهیم و میبینیم که دقت آن نزدیک به ۴ درصد افزایش یافته است.

```
1500/1500 [==============================] - 5s 4ms/step - loss: 0.1954 - accuracy: 0.9275 - val_loss: 0.4143 - val_accuracy: 0.8763
Epoch 50/50
1500/1500 [==============================] - 5s 3ms/step - loss: 0.1939 - accuracy: 0.9272 - val_loss: 0.4021 - val_accuracy: 0.8780
```

در ابتدا مدل با نرخ آموزش ۰.۰۱ به دلیل بالا بودن به سختی میتوانست همگرا شود که در مدل جدید به کاهش آن به ۰.۰۰۱ توانستیم عملکرد آن را بهتر کنیم و همچنین با کم کردن تعداد یونیت ها ظرفیت مدل را پایین آوردیم تا بشود از اورفیت احتمالی جلو گیری کرد.