بسمه تعالی

دانشکده مهندسی کامپیوتر

آبان ۱۴۰۰

یادگیری عمیق

نام استاد: دکتر محمدی

تمرین پنجم

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

۱. پاسخ سوال اول

الف) ۵ نقطه میخواهیم که هر کدام x و y خود را دارند. پس لایه خروجی باید ۱۰ نورون داشته باشد تا هر نورون مسئول یکی از خروجی های شبکه باشد.

به عنوان تابع فعالسازی، اگر مقادیر عکس ها را scale کنیم و درواقع داده ها را نرمال کنیم تا بین صفر و ۱ باشند، میتوانیم از سیگموید استفاده کنیم. چون تایع فعالسازی مناسبی برای مقیاس کردن است. استفاده از tanh هم در صورتی که طول و عرض را بین −۱ و ۱ مقیاس کنیم مناسب است. حتی میتوانیم در لایه آخر از تابع فعالسازی خاصی استفاده نکنیم که در آن حالت باید حواسمان به حدود x و y باشد. استفاده از Relu اصلا مناسب نیست چون بخشی از landmark ها را از بین میبرد!

چون با نقاط و فاصله بین آن ها سر و کار داریم، تابع ضرری که مبتنی بر فاصله بین نقطه بهینه و به دست آمده باشد به نظر مناسب است. مثلا mean square error که مربع فاصله نقطه را بازمیگرداند یا تابع ضرری که بر اساس فاصله اقلیدسی کار کند.

ب) از تابع فعال ساز سیگموئید در آن استفاده شده است.

همچنین mse_with_dontcare به عنوان تابع ضرر به کار رفته است. که خود فرد آن را طراحی کرده است و از توابع keras نیست. این تابع فاصله اقلیدسی نقطه پیشبینی شده با نقطه اصلی را برمیگرداند.

```
def mse_with_dontcare(T, Y):
    ##Find the pairs in T having x-y coordinate = (0, 0)
    #Reshape to have x-y coordinate dimension
    T_reshaped = tf.reshape(T, shape=[-1, 2, NUM_LANDMARKS])
    Y_reshaped = tf.reshape(Y, shape=[-1, 2, NUM_LANDMARKS])
    #Calculate Euclidean distance between each pair of points
    distance = tf.norm(T_reshaped - Y_reshaped, ord='euclidean', axis=1)

    #If summing x and y yields zero, then (x, y) == (0, 0)
    T_summed = tf.reduce_sum(T_reshaped, axis=1)
    zero = tf.constant(0.0)
    mask = tf.not_equal(T_summed, zero)
    #Get the interested samples and calculate loss with Mean of Euclidean distance
    masked_distance = tf.boolean_mask(distance, mask)
    return tf.reduce_mean(masked_distance)
```

۲. پاسخ سوال دوم

با توجه به این که label های خروجی ۰ و ۱ و ۲ و ۳ هستند، مسئله یک مسئله classification با ۴ کلاس است. پس لایه آخر شبکه را با ۴ نورون طراحی میکنیم.

برای مسائل این چنینی که بهتر است خروجی شبکه را برای هر کدام از ۴ کلاس ببینیم و بیشترین را برگزینیم، softmax بهترین تابع فعالسازی لایه آخر است. که در واقع تعمیم یافته سیگموید برای چند کلاس است.

همانطور که قبلا بررسی کردیم و دیدیم که برای مسائل دو کلاسه crossentropy بهترین تابع ضرر است (به علت بزرگ شدن گرادیان با وجود مشتق های زنجیره ای)، در اینجا هم از تعمیم یافته آن یعنی categorical_crossentropy استفاده میکنیم که برای چند کلاس طراحی شده است.

استفاده از بهینه ساز adam به نظر منطقی میرسد چون قبلا در مورد مزایای آن صحبت کردیم و دلیلی برای استفاده نکردن آن با این دیتاست نیست.

در نهایت با اضافه کردن صرفا همین دو خط شبکه را میسازیم:

```
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy']
              )
```

خروجی روی داده آموزشی و داده ارزیابی به این صورت بود:

```
train_loss, train_accuracy = model.evaluate(x_train, y_train)
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("train accuracy:",train_accuracy*100,"%")
print("test accuracy:",test_accuracy*100,"%")
```

```
50/50 [==============================] - 0s 3ms/step - loss: 0.3992 - accuracy: 0.8381
13/13 [==============================] - 0s 3ms/step - loss: 0.5604 - accuracy: 0.7800
train accuracy: 83.81249904632568 %
test accuracy: 77.99999713897705 %
```

با برخی حالات دیگر تابع ضرر یا بهینه ساز یا تغییر ابعاد لایه آخر هم برای اطمینان بررسی کردم و بهترین نتایج در همین حالت گفته شده بود.

۳. پاسخ سوال سوم

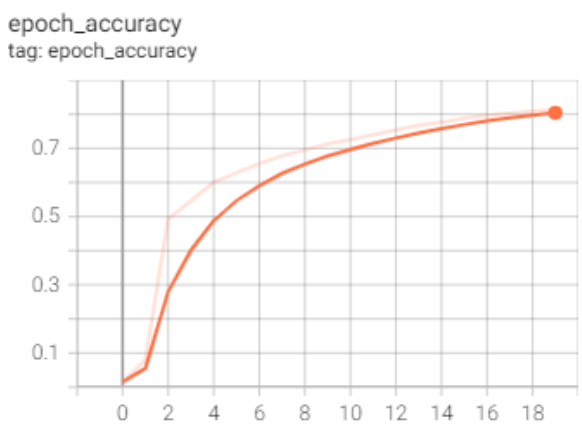
کدهای مربوط به این سوال در فایل HW5.ipynb موجود است. (لینک گوگل کولب)

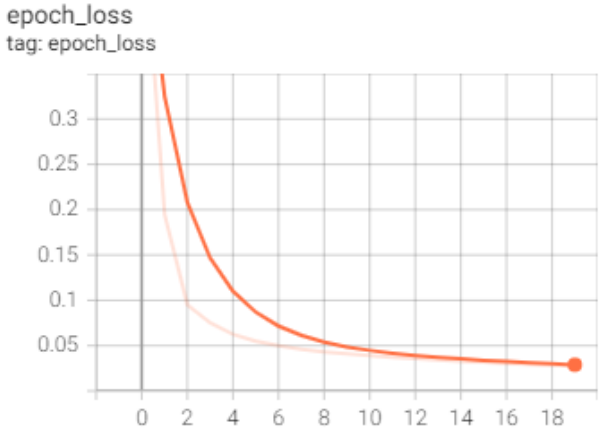
خروجی اجرای شبکه روی هر کدام از بخش ها در جدول زیر آمده است. گزارش کردن مقدار عددی loss فایده خاصی نداشت چون توابع ضرر متفاوت هستند.

ACTIVATION, LOSS	TRAIN ACCURACY	TEST ACCURACY
SIGMOID, BINARY CROSS	82.27%	75.55%
SOFTMAX, CATEGORICAL	93.05%	78.81%
SIGMOID, MSE	60.70%	59.39%
SOFTMAX, MSE	87.23%	76.27%
LINEAR, MSE	77.77%	70.30%

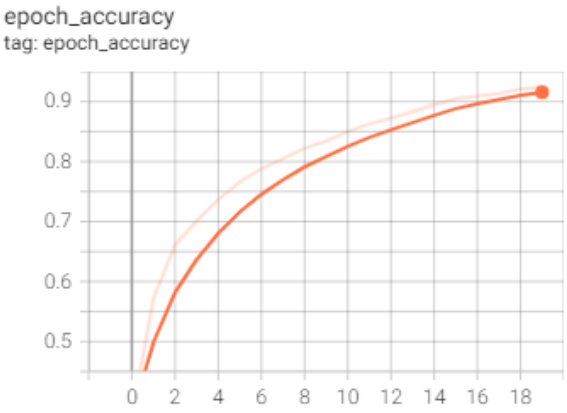
نتایج tensor board هم به صورت زیر بودند:

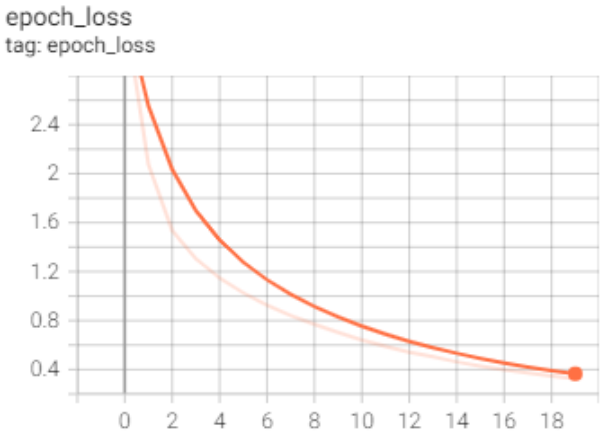
- تابع فعالسازی sigmoid و تابع ضرر binary crossentropy



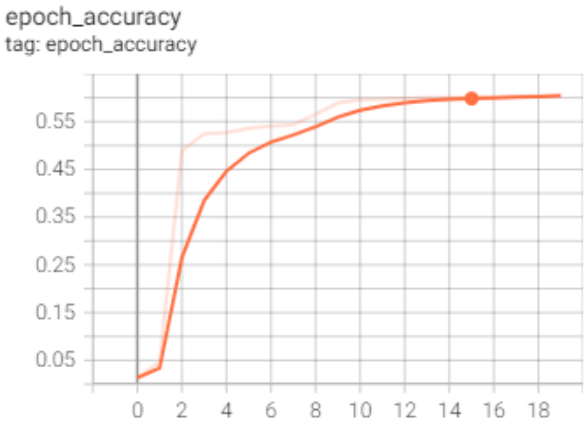


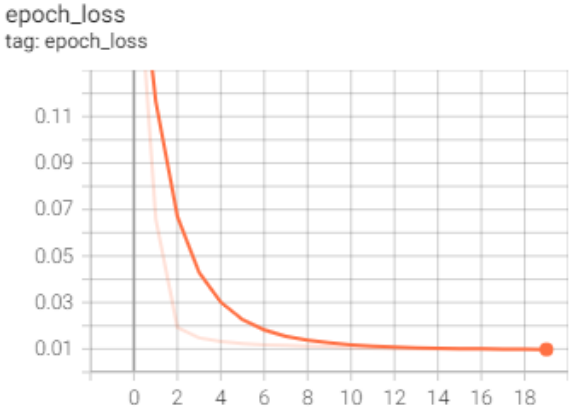
- تابع فعالسازی softmax و تابع ضرر categorical crossentropy



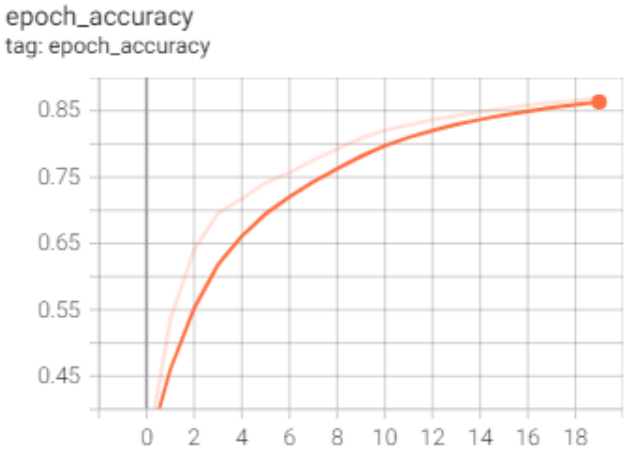


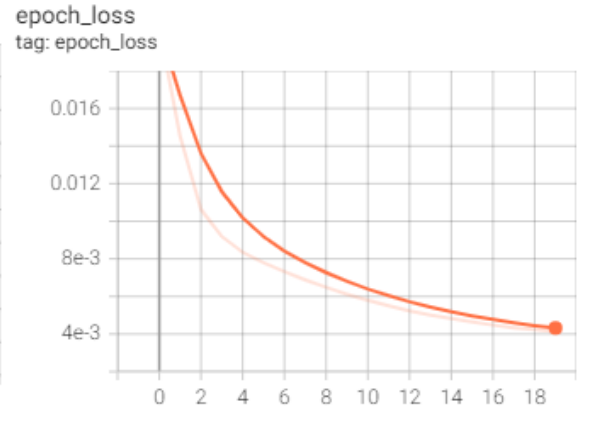
- تابع فعالسازی sigmoid و تابع ضرر mse



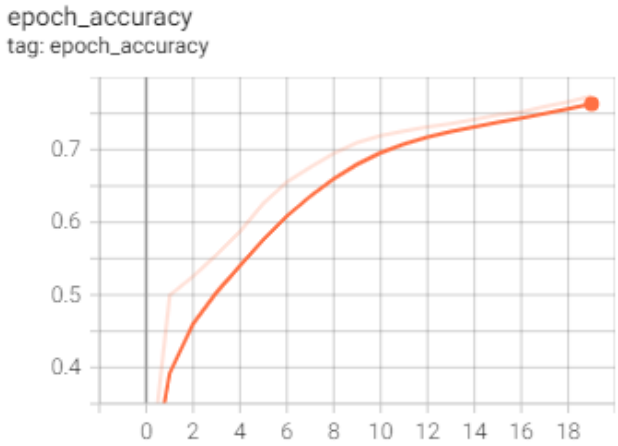


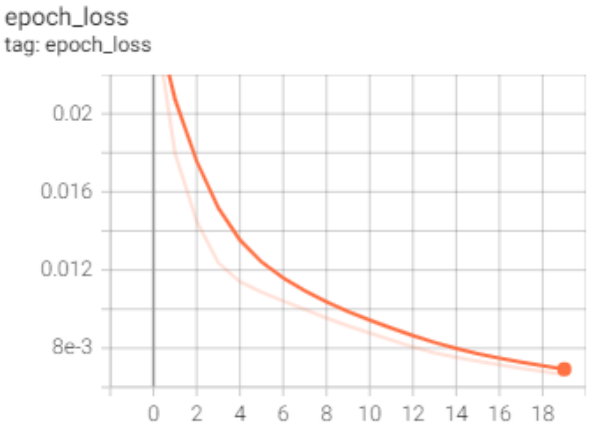
- تابع فعالسازی softmax و تابع ضرر mse





- تابع فعالسازی linear و تابع ضرر mse





مطابق انتظار چون مسئله چند کلاسه است، بهترین تابع فعالسازی softmax است که تعمیم یافته سیگموید برای چندین کلاس می باشد. همچنین بهترین تابع ضرر هم categorical cross entropy است که همان تابع کراس آنتروپی با تغییر رابطه آن برای چندین کلاس است.

بدترین عملکرد هم با اختلاف مربوط به MSE و sigmoid است که هم تابع ضرر و هم تابع فعالسازی مناسب انتخاب نشده است.