گزارش تمرین شماره 1

محمد زیاری

« بعضی از اصطلاحات انگلیسی به علت مشکلات فونت ، فارسی تایپ شده است»

توضیح کد

ابتدا خیلی کوتاه توضیحاتی نسبت به بعضی از بلاک های کد خواهم داد.

در بلاک اول که کتابخانه های مورد نیاز را امپیورت کرده ایم.

در بلاک دوم داده ها را از پوشه تمرین میخوانیم و در بلاک سوم بررسی میکنیم که آیا داده خالی دارد که جواب خیر است پس لازم به انجام کار اضافه ای نیست.

داده هایی که عددی نبودند را Lable encoding میکنیم.

و همچنین داده هایی که نیاز نداریم را از داده ها دراپ میکنیم و نتیجه را در y میریزیم.

در بلاک 5 هم میشه دید که همه داده ها در حال حاضر عددی هستند و مشکلی از این بابت نداریم. در بلاک 6 اسکیل را با مین مکس اسکیلر انجام داده ایم.

در بلاک 7ام هم داده های تست و تمرینی را از هم جدا کردیم .

در بلاک 8 ام ابزار لازم برای ساخت شبکه را ایمپورت میکنیم و در بلاک 9 ام شبکه را میسازیم که این بلاک به طور کل در بخش آزمون و خطا مورد بحث قرار میگیرد.

در بلاک 10 نمودار دقت و در بلاک 11 نمودار خطا را میکشیم تا بررسی ها روی آن انجام شود.

2 بلاک آخر هم برای بررسی داده های تست انجام شده است که درصد میزان درستی شبکه بر روی داده های تست مشخص شود.

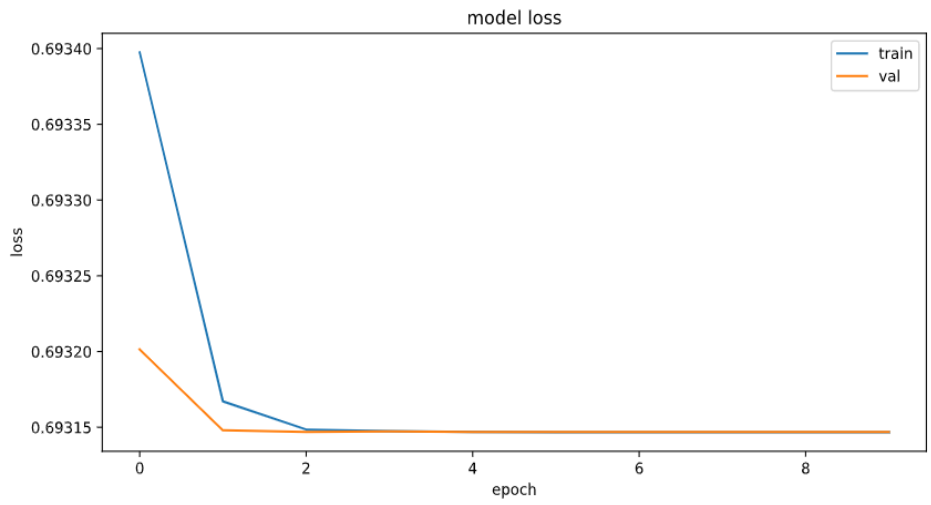
«در ادامه این گزارش تاثیر هایپر پارامتر ها و آزمون و خطای آن ها نوبت به نوبت بررسی میشود»

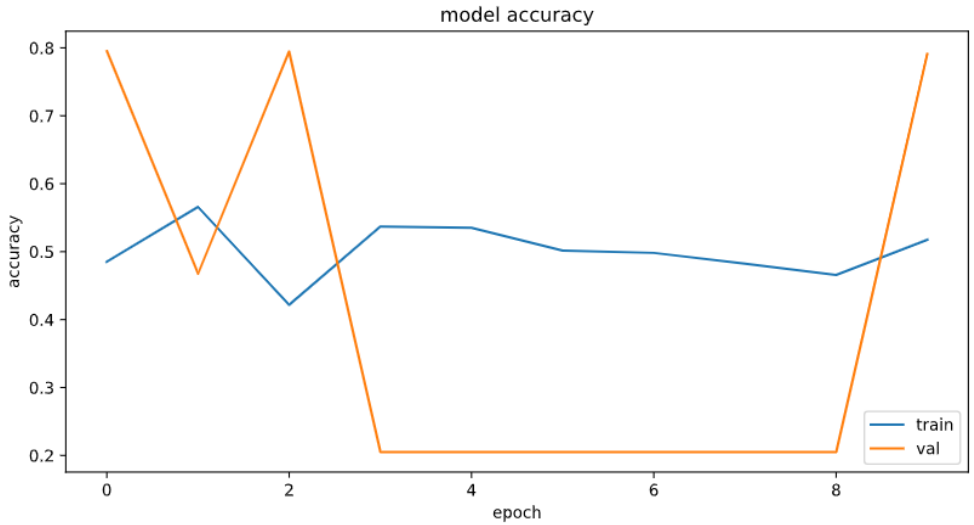
1 . تعداد لایه ها

سعی کردم با یک شبکه ساده تر کار را شروع کنم تا در هر مرحله پیشرفت مدل را بتوانیم شاهد باشیم.
اول از همه با 2 لایه شروع به کار میکنیم (تنها یک لایه hidden)
عکس پایین شبکه اولیه ماست که تغییرات هر مرتبه روی آن اعمال می شود :

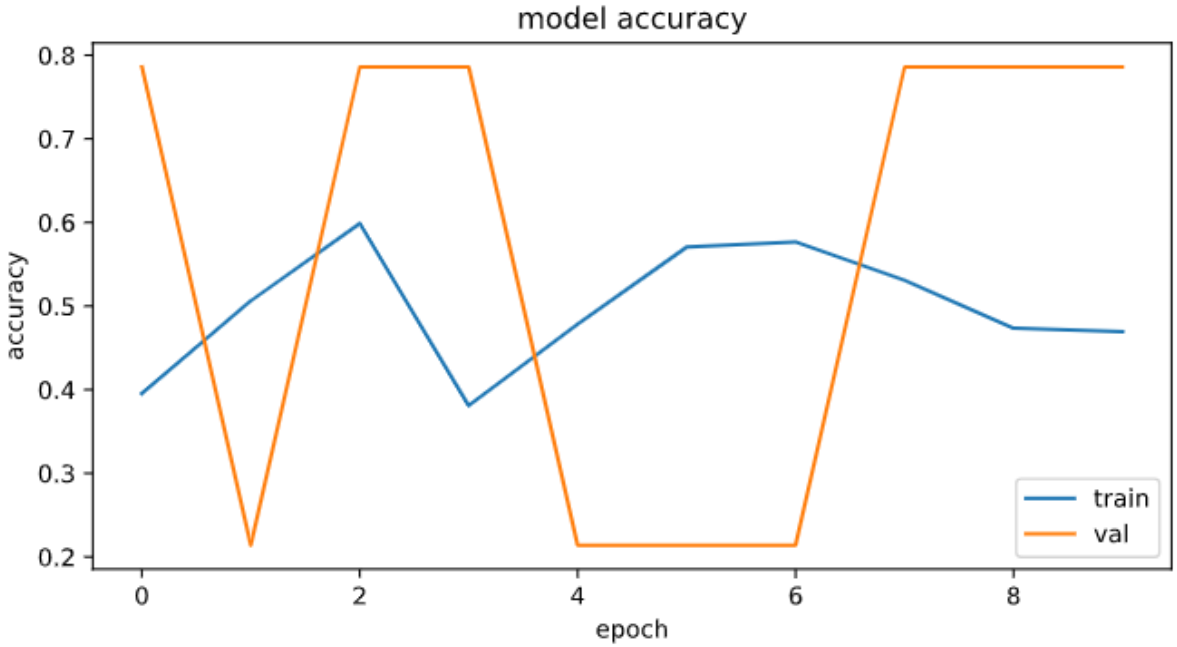
```
input_layer = Input(shape=X_train.shape[1], name ="input")
hidden_layer1 = Dense(18 ,activation='softmax' , name="layer1")(input_layer)
output_layer = Dense(2 ,activation='softmax' , name="layer3")(hidden_layer1)
network = Model(inputs=input_layer, outputs=output_layer , name = "network1")
adam= tf.keras.optimizers.Adam(learning_rate=0.01)
network.compile(optimizer=adam , loss='binary_crossentropy', metrics=['accuracy'])
history = network.fit(X_train,y_train , batch_size=250 , epochs=10 , validation_split=0.25 )
```

نتیجه به دست آمده به این شکل خواهد بود:





تعداد لایه های شبکه را افزایش میدهیم تا بررسی کنیم بر روی دقت چه تاثیری دارد



واضح است تغییر خاصی حاصل نشد و هنوز مدلمان متناوب است. ؛ پس در حال حاضر از روش های دیگر استفاده میکنیم.

پس از کلی آزمون و خطا به این نتیجه رسیدم با عوض کردن یکی یکی پارامتر ها نمیتوان به نتیجه رسید (که از کدام باید استفاده کرد)
از آن مثلا اگر در یک مدل سیگموید در لایه وسط کاربرد نداشت دلیل نمیشود به طور کل در لایه وسط استفاده نکنیم.
ولی بعد از چندین ساعت آزمون و خطا به یک سری نتایج کلی رسیدم که برای هر هایپر پارامتر توضیح میدهم.

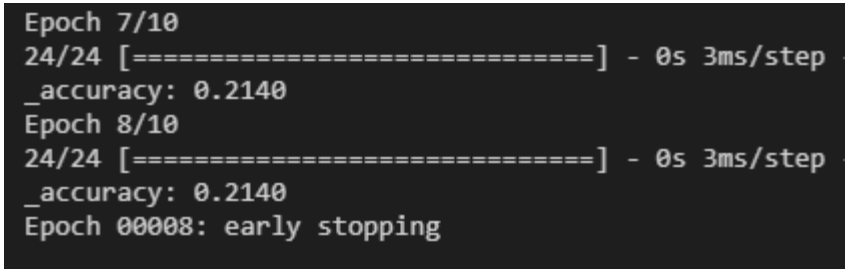
برای بررسی تعداد لایه ها فکر میکنم با توجه به داده هایی که داریم نیاز به لایه ی خیلی زیادی نداریم چون هرچه تعداد لایه ها بیشتر باشد ، پیدا کردن تابع فعالسازی برای آن سخت تر خواهد بود (که بتوانیم نتیجه گیری کنیم که کدام برای مدلمان درست کار میکند.)

من با توجه به یک سری از آزمایشات تصمیم گرفتم داده ها را با همان 3 لایه (2 لایه مخفی و 1 خروجی) ادامه بدهم.

2. تعداد ایپاک ها

به نظرم تغییر این متغیر وقتی کارساز خواهد بود که به یک مدل خوب رسیده باشیم تا با استفاده از ایپاک های بیشتر مدل بیشتریاد گرفته شود (learn)
و همینطور وقتی داده های بیشتری داشته باشیم تا بتوانیم باز هم یادگیری را افزایش ندهیم.
بهتر است در صورت زیاد شدن ایپاک ها یک تابعی برای overfitting البته جلوگیری از
داشته باشیم که در اینجا از Early Stopping استفاده میشود.

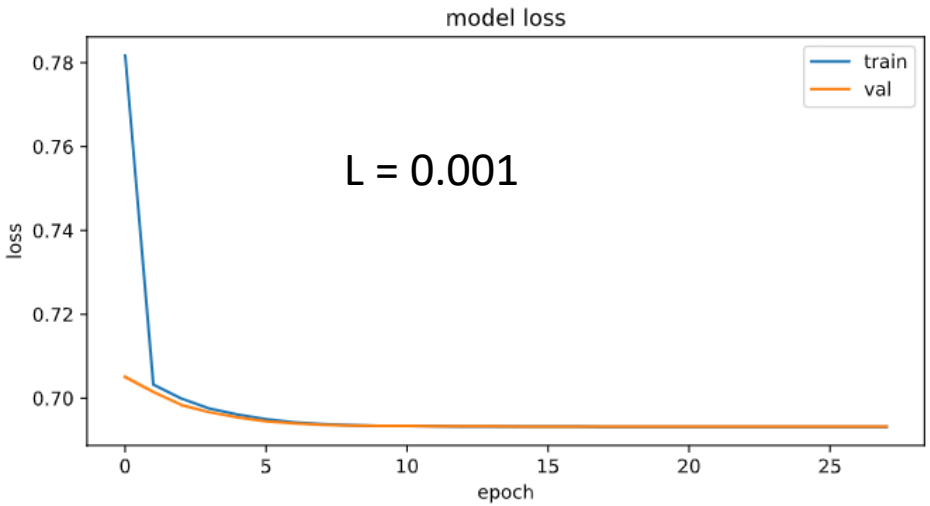
برای اینکه بند اول که گفته شد نیازی به ایپاک بالا در این شبکه نیست نمودار زیر را میبینیم که به Early stopping بر خورده ایم.

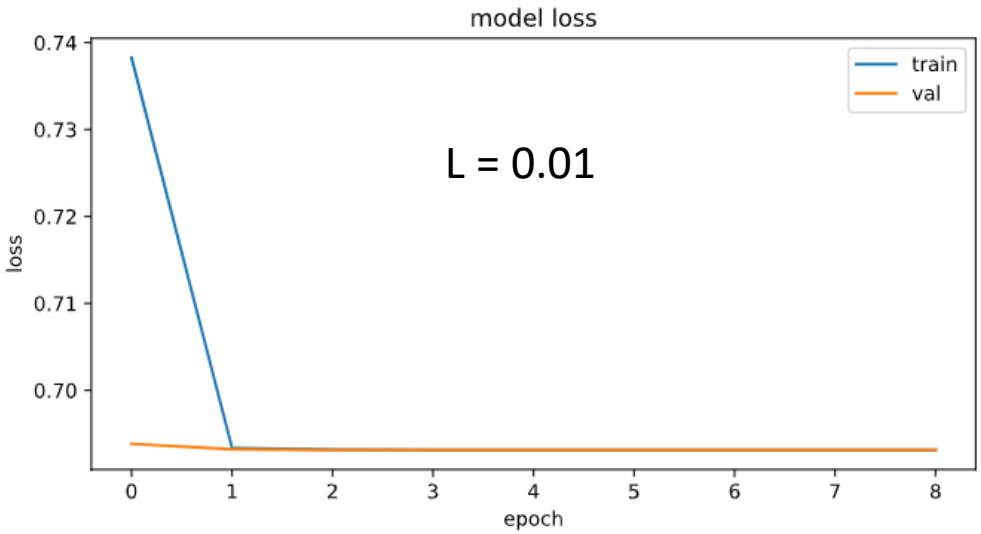


به علت درست نبودن مدل و همچنین کم بودن داده که تصمیم گرفتم در این مدل از 50 ایپاک استفاده کنم چون طبق چیزی که تست کردم بیش از این مقدار نیاز نیست و استپ میشه توسط Early stopping

Learning rate , Momentum .3

اول از همه من از 2 راه برای بهینه سازی استفاده کردم (که در ادامه توضیح میدهم)
اگر از آدام استفاده کنیم تنها نرخ یادگیری خواهیم داشت که با 2 نمودار بررسی میکنیم چه عددی به طور کلی برای آن مناسبتر است.





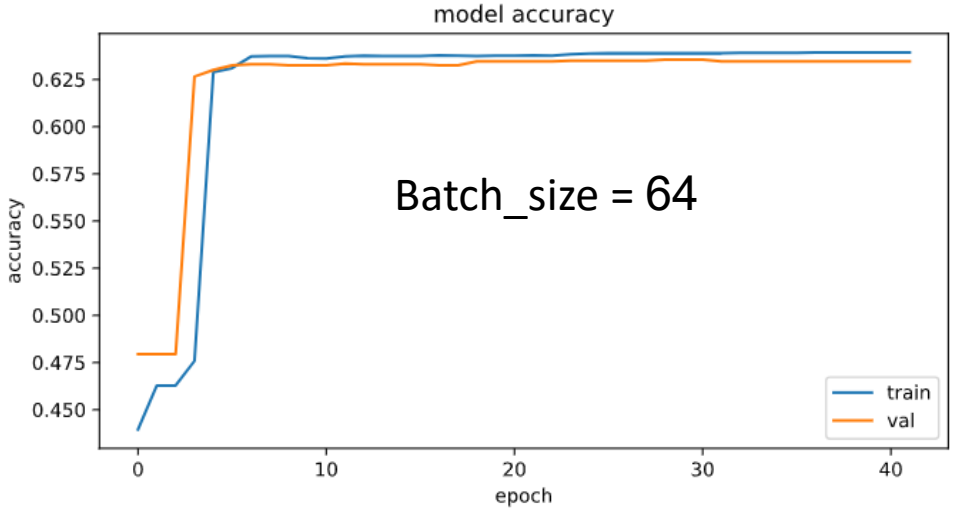
همانطور که در صفحه قبل دیدیم علاوه بر اینکه در 0.01 به اورفیتینگ رسیدیم یک خوبی دیگری که نرخ یادگیری 0.001 دارد آن است که تابعمان خطی نیست (در حالی که 0.01 کاملا خطی است)
حالا این یک مثال بود ولی به طور کل در آزمایشاتم نرخ یادگیری 0.001 موثر تر بود و از آن استفاده میکنم.

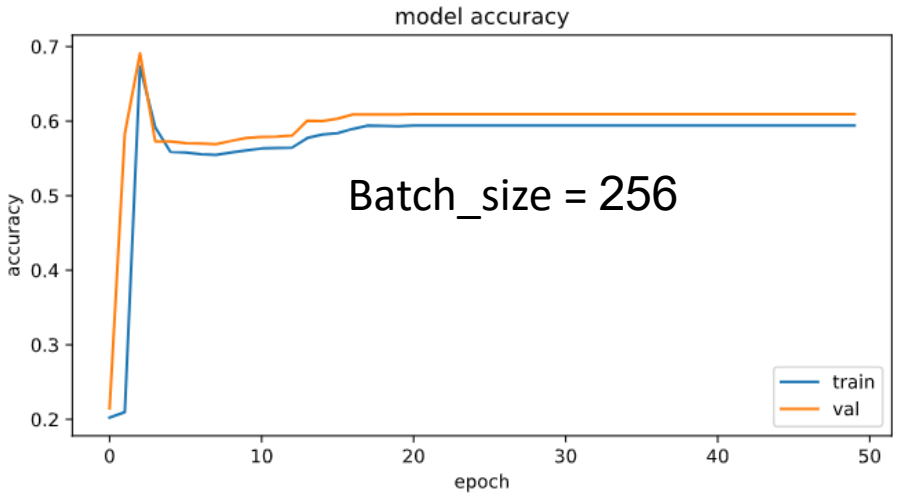
برای sgd هم مومنتوم که تاثر لایه های قبلی بر این لایه است را میشود با آزمون و خطا میزان درستی پیدا کرد که طبق نتایج من عدد بین 0.8 و 0.9 عدد به نسبت خوبی است.
نرخ یادگیری هم همان بین 0.01 و 0.001 خوب نتیجه میدهد. که البته 0.001 استفاده کردم.

Batchsize , validation split . 4

اگر بخواهیم از validation استفاده کنیم ، چون تعداد داده ها کم است بهتر است از 0.25 یا 0.2 استفاده کنیم چون در غیر اینصورت Val_accuracy ممکن است مقدار عجیب و غریبی بگیرد و هر دفعه بسیار متناوب حرکت کند.

راجع به سایز Batch هم طبق سرچ هایی که داشتم معمولا (نه الزاما) سایزهای بزرگ بهتر عمل میکردند اما در عمل من به طور کلی تست هایم با بچ 128 یا 64 مقدار قابل قبولی گرفتند



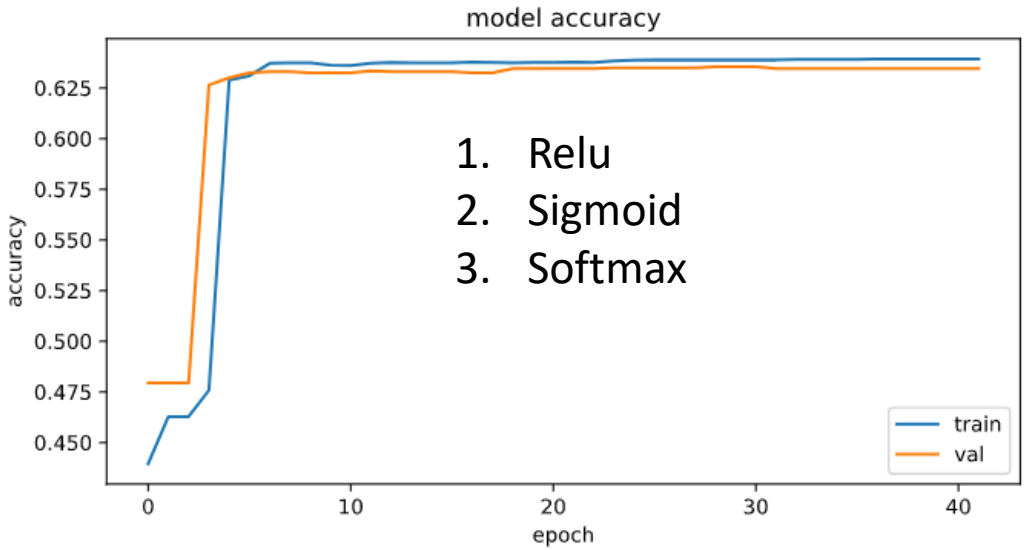


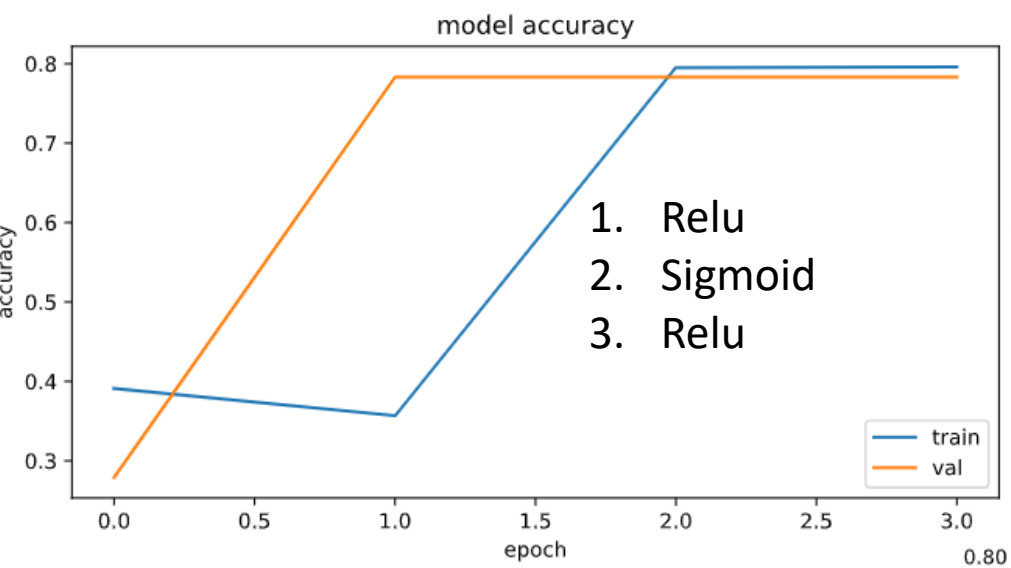
5 . توابع فعالسازی

در این مورد هم نمیشود به طور کلی پیشبینی کرد زیرا مثلا شاید اگر سایز بچ فلان قدر بود نرخ یادگیری یک مقدار دیگر آن گاه یک تابع فعالسازی دیگر میتوانستیم استفاده کنیم .

اما طبق چیزی که من تست کردم به طور کلی اگر در لایه مخفی اولیه رلو ، در تابع خروجی سافتمکس و در تابع مخفی دوم بین 2 تابع سیگنوید و رلو انتخاب شود (بستگی به بهینه ساز و دیگر هایپرپارامترها یکی از دیگری بهتر عمل میکند) مدل به نسبت بهتری خواهیم داشت.

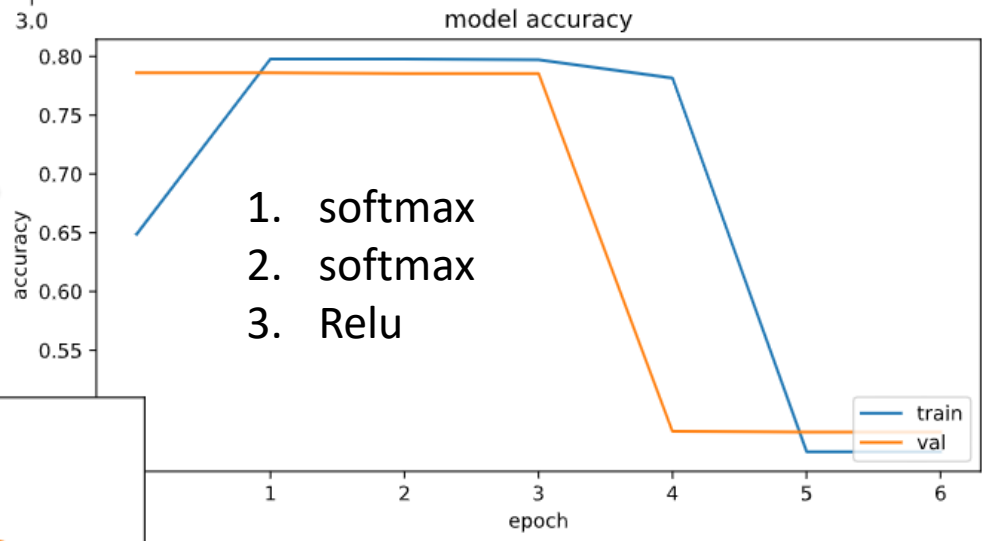
دراین صفحه نمودار بهتر را خواهیم دید ولی در صفحه بعد 3 نمودار برای 3 مدل خواهیم دید که کاملا واضح است بقیه پرت هستند.



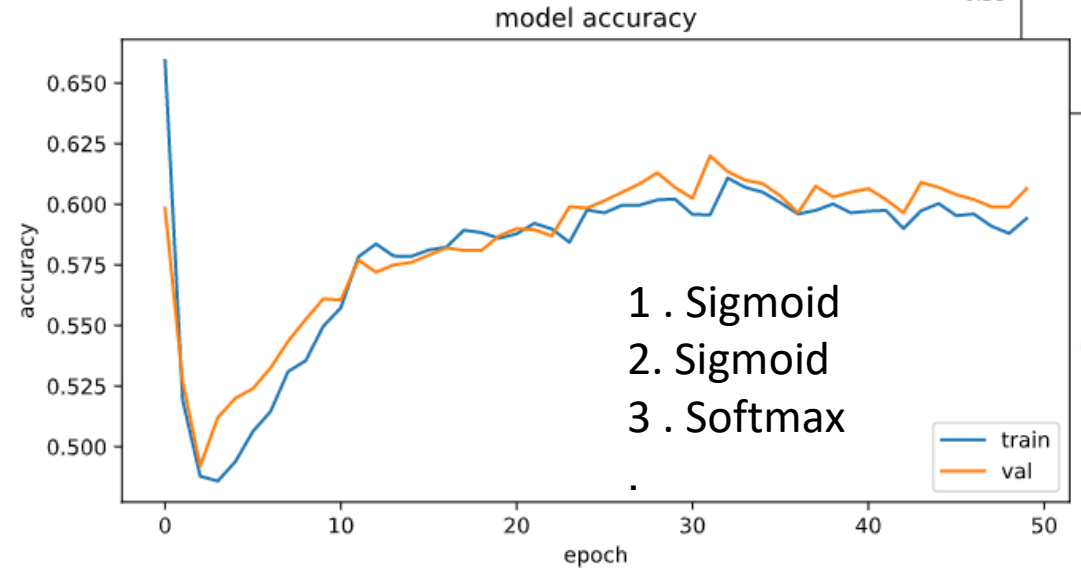


خیلی زود به استاپ میخوردیم

(در اینجا در اپیاک سوم)



دقت در حال افت!



تناوب بالایی دارد ولی از بالایی ها بهتر است.

6 . Optimizer , Dropout

اول از همه راجب اپتیمایزر این نکته رو باید بگم که از 2 تایی که گفته شد یک سری از شبکه ها بهترعمل میکردند ولی در مدل نهایی از SGDاستفاده شد.
ولی دلیلی برای ناکارآمدی هیچکدام نیست.

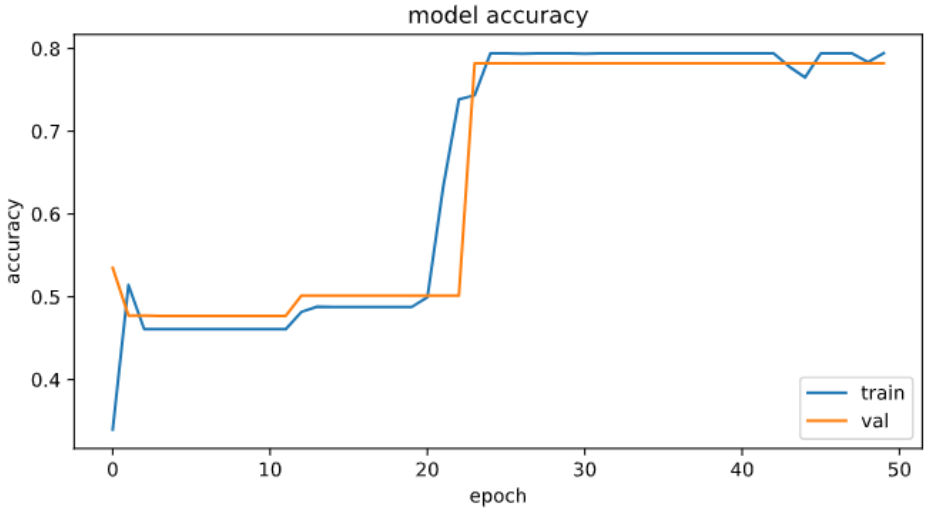
برای استفاده از دراپ اوت هم در بعضی مدل ها به کار ما آمد و در مدل نهایی هم از آن استفاده کردم و البته این نکته وجود دارد که فکر میکنم ممکنه چون داده های ما مقدار فراوانی ندارد به مشکل بخوریم که البته هایپرپارامتر های بهتر این مشکل را رفع میکند.

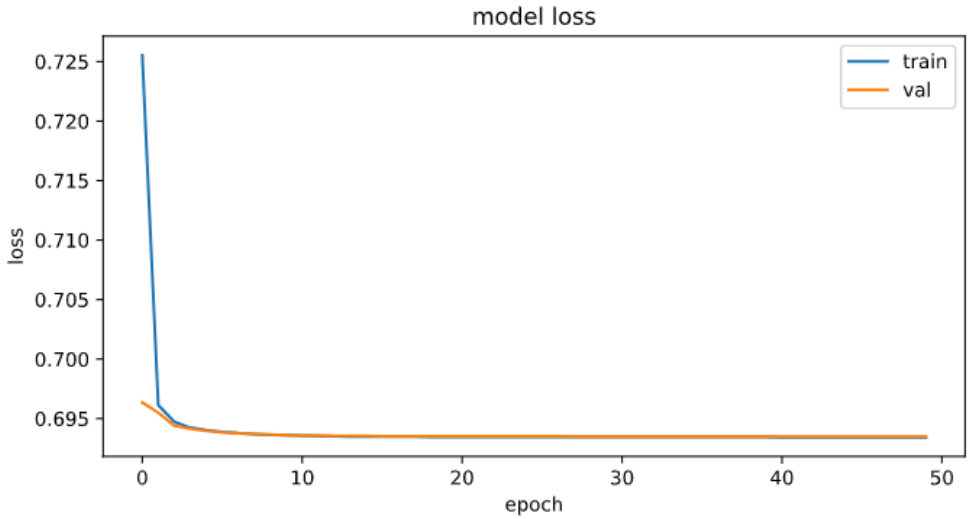
7 . تعداد نورون ها

صحبت از تعداد نورون های هر لایه که به بخش آخر تبدیل شده رو به صورت کوتاه بررسی هایم رو توضیح میدم .

بهتر است از عمق (نورون در لایه اول) بیشتری استفاده کنیم و در لایه دوم یک میزان به نسبت کمتر از اولی داشته باشیم . همینطور در لایه خروجی من از 2 نورون استفاده کردم.

یک نمونه خوب !



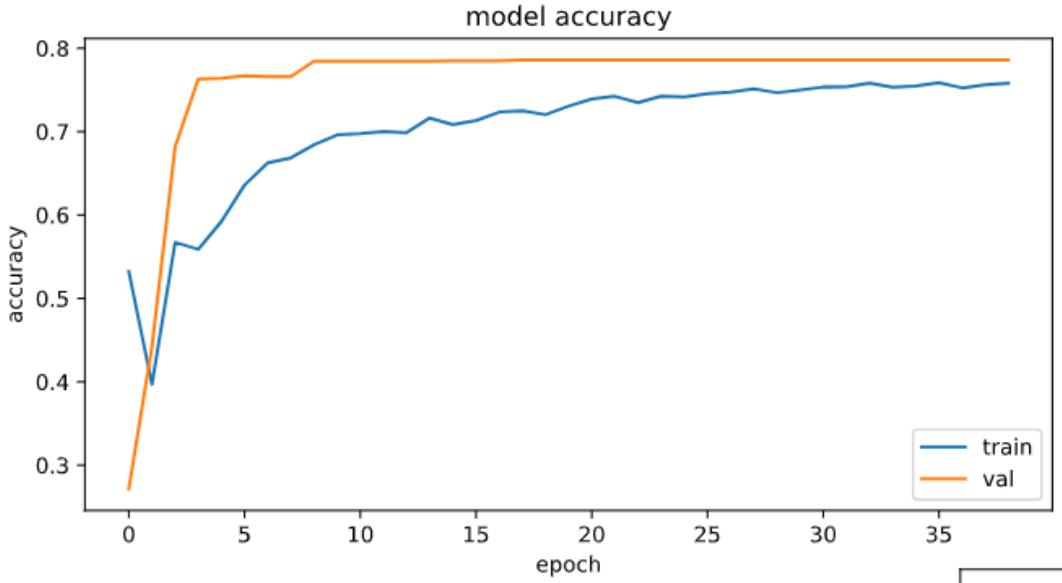


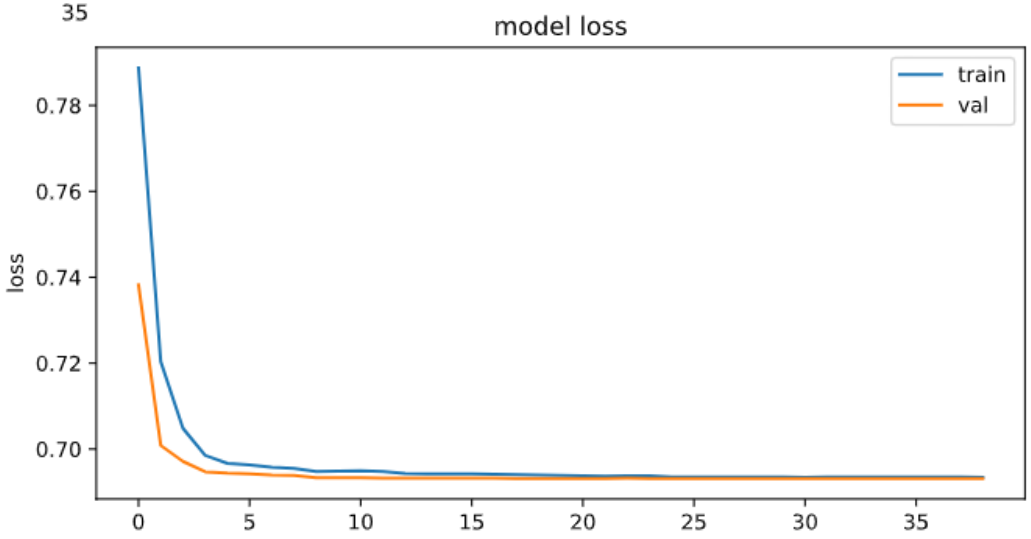
متاسفانه وزن این داده پاک شد.

نمونه پایانی

فکر میکنم توضیحات در مراحل پیشین به طور کامل داده شد و در مدل پایانیمان فقط کد و 2 نمودار را بررسی میکنیم.

```
sgd = SGD(lr=0.001 , momentum=0.87)
es = EarlyStopping(monitor='val_loss', mode='auto', verbose=1 , patience=3)
input_layer = Input(shape=X_train.shape[1], name ="input")
hidden_layer1 = Dense(128 ,activation='relu' , name="layer1")(input_layer)
dropout_layer = Dropout(0.2)(hidden_layer1)
hidden_layer2 = Dense(16 ,activation='sigmoid' , name="layer2")(dropout_layer)
output_layer = Dense(2 ,activation='softmax' , name="layer3")(hidden_layer2)
network = Model(inputs=input_layer, outputs=output_layer , name = "network1")
adam= tf.keras.optimizers.Adam(learning_rate=0.001)
network.compile(optimizer=sgd , loss='binary_crossentropy', metrics=['accuracy'])
history = network.fit(X_train,y_train , batch_size=64 , epochs=50 , validation_split=0.25 , callbacks = [es])
```





دقت نهایی روی داده های تست هم 80.1 درصد بود که قابل قبول است زیرا

دقت روی داده های تمرین و ولیویشن هم تقریبا نزدیک به این عدد بود

(همانطور که در نمودار مشاهده میکنید)