Deep learning

Homework 3

Parsa Eissazadeh 97412364

سوال 1

الف)

توضیح overfit : هنگامی که یادگیری بر روی مجموعه ای داده ها افزایش می یابد ، شبکه تبدیل به دیکشنری ای میشود که دوگانه های داده برچسب را حفظ می کند (به عبارتی accuracy به 1 می رسد) . اما به دلیل خطاهایی که در ثبت داده ها وجود دارد امکان دارد برچسبی که بر روی داده های train است لزوما بر روی داده های test یا بر روی داده های واقعی وجود نداشته باشد .

راهکار های پیشگیری : به طور کلی راهکار ها باید در جهتی باشند که امکاناتی را از شبکه بگیرند که باعث شده اند شبکه به یک map از داده به برچسبش برسد .
باید بررسی کنیم چه چیز هایی باعث می شوند که یک شبکه زیاد یاد بگیرد . عوامل :

1. قابل پیش بینی بودن داده ها
2. Iteration های زیاد
3. نورون های زیاد

با ضعیف تر کردن هر یک از این ها می توان از overfitجلو گیری کرد . راه جلوگیری :

پیش بینی پذیر بودن داده ها : افزایش داده ها . که به دو طریق می تواند انجام شود :

1. داده افزایی ، به طریق ساختن داده های جدید از روی داده های قبلی (با trasnlation ها ، rotation ها ، crop کردن ها و ...)
2. افزودن دیتاست های جدید

نورون های زیاد : حذف کردن بعضی از داده ها به صورت رندم .

ب)

ذهنیت کلی جلوگیری از overfit ضعیف کردن شبکه یادگیری است . یکی از راه ها این است که یادگیری را زودتر از زمانی که به accuracy صد درصد برسد متوقف کنیم .

یکی دیگر از راه ها دادن داده های بیشتر به شبکه است .

ب)

دلیل این اتفاق خطا در ثبت اطلاعات است که در دنیای واقعی وجود دارد . گاهی به دلیل همین خطا ، فیچر موقعیت یک داده در صفحه مختصات (که هر کدام از محور ها یکی از فیچر هاست) در میان داده های گروه مقابل می افتد برای همین هنگامی که با همچین داده هایی (که در اثر خطا در گروه اشتباهی قرار گرفته اند) به مانند داده پرت رفتار می شود و ازشان صرف نظر می شود ، به قضاوتی دقیق تر بر روی داده های جدید می رسیم .

سوال 2

مرحله forward propagation :

داده x=1 را وارد شبکه می کنیم .

$$1*1 + x * 2 + x^2 *3 + x^3 *-2 + x^4 *-1 =$$

$$1 + 2 + 3 - 2 - 1 =$$

$$3$$

چون activation function امان linear است ، خروجی شبکه 3 است و برچسب 20 . از loss مشتق میگیریم که برابر است با :

$$d(Loss) = 20 - (3) = 17$$

وزن ها را از بالا به پایین w_1 تا w_5 نام گذاری می کنیم .

$d(neuron) / d (w_1) = 1$

$d(neuron) / d (w_2) = 1$

$d(neuron) / d (w_3) = 1$

$d(neuron) / d (w_4) = 1$

$d(neuron) / d (w_5) = 1$

و در نهایت مشتق خروجی loss شبکه نسبت به وزن ها بدین صورت هستند :

$d(loss) / d (w_1) = 1 *17= 17$

$d(loss) / d (w_2) = 1 *17 = 17$

$d(loss) / d (w_3) = 1 *17 = 17$

$d(loss) / d (w_4) = 1 *17 = 17$

$d(loss) / d (w_5) = 1 *17 = 17$

برای لحاظ کردن regularizer ، باید تابع جریمه اش را در فرمول update کردن وزن ها دخیل کنیم و به کمکش w را کم کنیم . آپدیت کردن وزن ها (0.1 نرخ یادگیری است که و 0.9 نرخ l2 است که در وزن ضرب می شود) :

$W_1^{new} = w_1 * (1 - 0.1 * 0.9) + 0.1 * 17 = 1 * 0.91 + 1.7 = 2.61$

$W_2^{new} = w_2 * (1 - 0.1 * 0.9)+ 0.1 * 17 = 2 * 0.91 + 1.7 = 3.52$

$W_3^{new} = w_3 * (1 - 0.1 * 0.9)+ 0.1 * 17 = 3 * 0.91 + 1.7 = 4.43$

$W_4^{new} = w_4 * (1 - 0.1 * 0.9)+ 0.1 * 17 = -2 * 0.91 + 1.7 = -0.12$

$W_5^{new} = w_5 * (1 - 0.1 * 0.9)+ 0.1 * 17 = -1 * 0.91 + 1.7 = -0.79$

ب)

هر چه مقدار ضریب کمتر باشد تاثیر خود وزن در آپدیت شدن کمتر می شود . هر چه کمتر شود جریمه تغییر وزن ها کمتر می شود و امکان تغییر بالای وزن ها و در نتیجه overfit شدن افزایش می یابد .

و هر چه این ضریب بیشتر شود ، تاثیر وزن بر روی آپدیت شدنش کمتر می شود که باز هم اتفاق خوبی لزوما نیست . زیرا هر چه تاثیر مقدار قبلی شبکه کمتر بشود ، شبکه کمتر امکان استفاده از دانشی که از قبل به دست آورده را دارد . که این باعث کاهش دقت می شود .

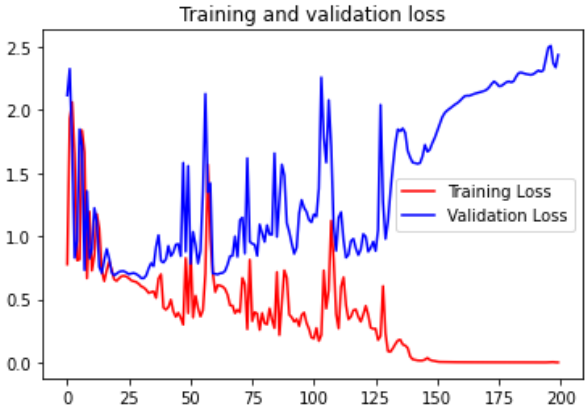
سوال 3

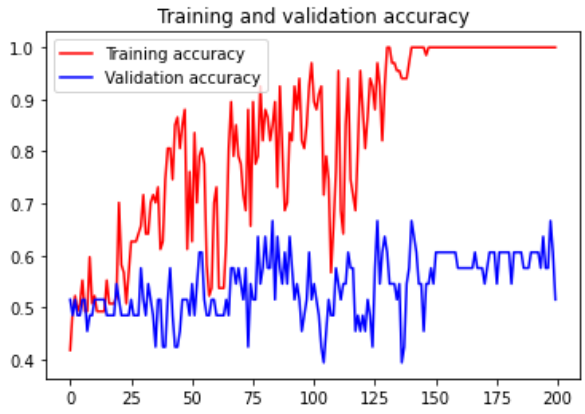
ابتدا فایل zip را باز کردم ، سپس بعد از resize کردن همه را درون یک numpy array ریختم . به ازای اضافه کردن هر عکس یک برچسب (که بر حسب نام فایل حساب میشد) را اضافه کردم .

سپس یک مدل نوشتم . مدلی با سه لایه . چون تنها دو کلاس داشتیم از1 نورون در لایه آخر به همراه sigmoid استفاده کردم .

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_6 (Flatten)         (None, 2352)              0

 dense_21 (Dense)            (None, 50)                117650

 activation_21 (Activation)  (None, 50)                0

 dense_22 (Dense)            (None, 50)                2550

 activation_22 (Activation)  (None, 50)                0

 dense_23 (Dense)            (None, 20)                1020

 activation_23 (Activation)  (None, 20)                0

 dense_24 (Dense)            (None, 1)                 21

 activation_24 (Activation)  (None, 1)                 0

=================================================================
Total params: 121,241
Trainable params: 121,241
Non-trainable params: 0
_________________________________________________________________
```

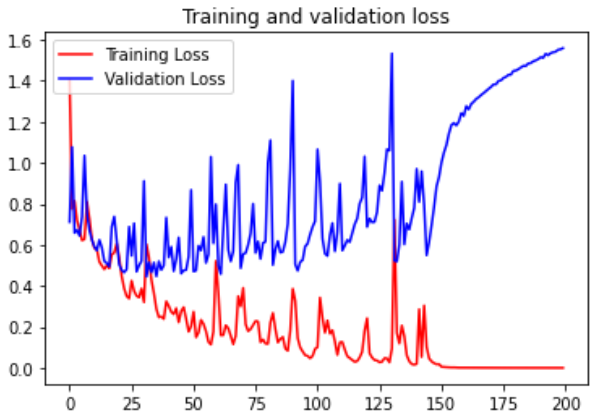
به کمک توابع آماده keras ، داده ها را به داده های train و validation تقسیم کردم . ولی در epoch 200 ، از epoch 140 به بعد به دقت 1 رسیدیم که به معنی overfit است . نمودار های دقت و loss در training و validation به صورت زیر است :

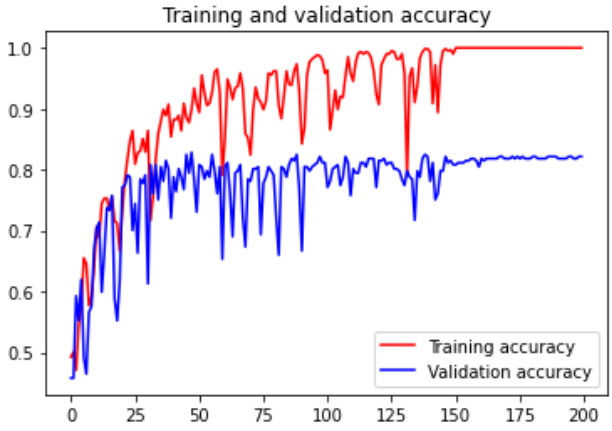




در مرحله بعدی ، همانند قبل عکس ها را از دایرکتوری خواندم و resize اشان کردم . سپس توابع ایجاد عکس های جدید را بر روی عکس ها اعمال کردم و همان برچسب های عکس منشاشان را اضافه کردم .

سپس دوباره شبکه را آموزش دادم . نتایج :





همانطور که مشاهده می شود ،همچنان مدل overfit شده ولی هم در epoch های دیرتری دقت به مقدار 1 رسید و هم مقدار loss در validation از حالت قبل کمتر شد ، (از ۲.۵ به حدود ۱.۶ رسید) و همچنین Accuracy نیز افزایش یافت (از 0.6 به 0.8 رسید) . شاید با افزایش داده ها این دقت افزایش بیابد.

دلیل :

یکی از راه های اصلی برای جلوگیری از overfit سخت تر کردن کار شبکه است . با ساختن عکس های جدید و اعمال تغییراتی روی آن ها کار شبکه را سخت تر می کنیم .

همچنین اگر تعداد داده ها زیاد باشند و قرار باشد شبکه تنها به نگاشتی از عکس به برچسب برسد (یعنی تنها حفظ کند و الگویی را یاد نگیرد) ، داده های بیشتری را حفظ کند .
در شبکه دوم از هر عکس تنوع های مختلفی وجود دارد . این قابلیت تعمیم را برای مدل افزایش می دهد . به خاطر همین بیشتر بودن تعمیم دهی مدل ، loss در مدل دوم (اختلاف بین نتیجه پیش بینی شده و نتیجه واقعی) کاهش می یابد .

در این سوال ما تنها تعداد داده ها را افزایش دادیم ، همچنان تعداد epoch ها بالاست و هیچ regularization ای نداریم . دلیل overfit شدن مدل دوم می تواند همین باشد . (هر چند مدل دوم با شدت کمتری overfit شد ، چون در epoch های دیرتری به دقت 1 رسیدیم .)

امتیازی :

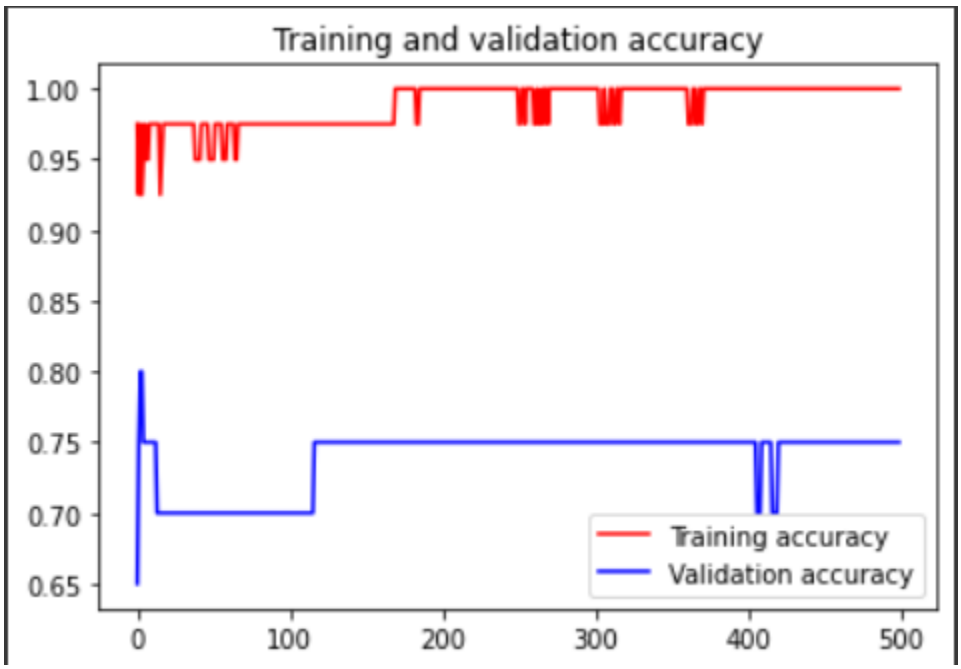
از تابع آماده

سوال 4

در ابتدا مدلی میسازیم که activation function آخرش sigmoid است . تعداد لایه ها 30 تا است . مدل را با epoch 100 به اجرا در می آوریم .

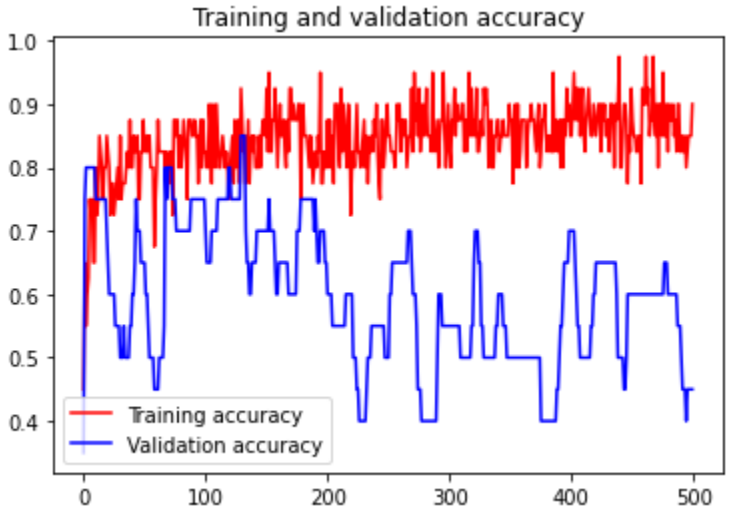
{عکس}

به accuracy صد درصد رسیدیم و نمودار را رسم کردیم :

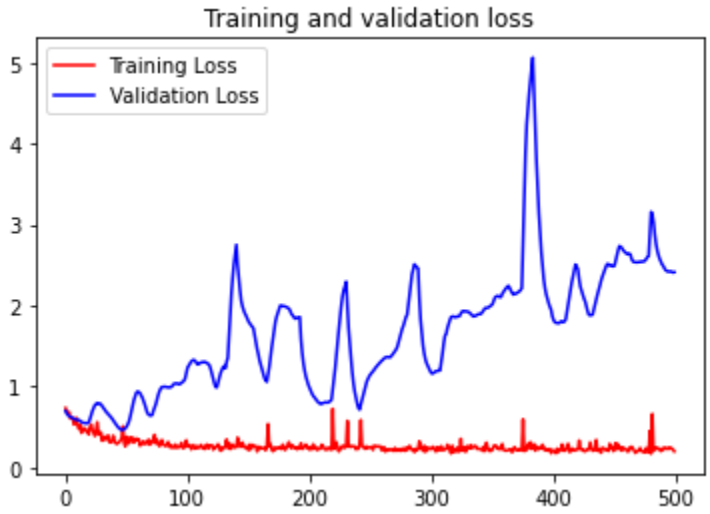


همانطور که معلوم است اختلاف بین training و validation بسیار زیاد است . حال از روش های رفع overfit استفاده میکنیم .

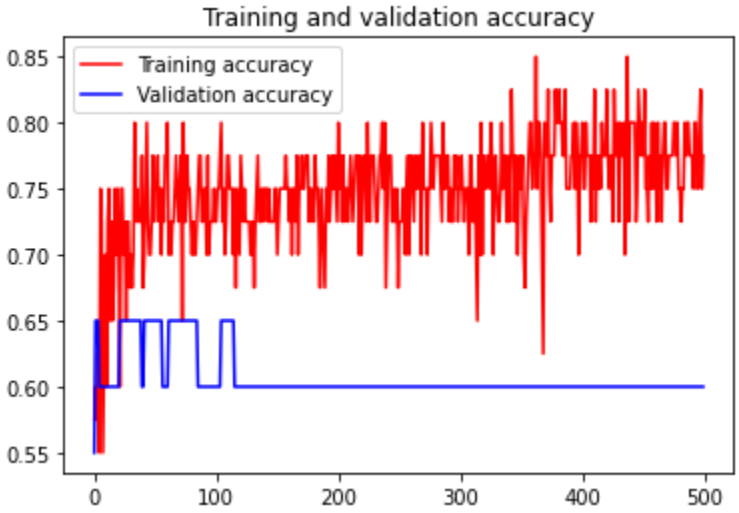
سه لایه Drop out با نرخ 0.5 اضافه کردیم ، مقدار accuracy کمتر شد و مقدارش در training و validation به هم بسیار نزدیک تر شد .نمودار را رسم کردم :



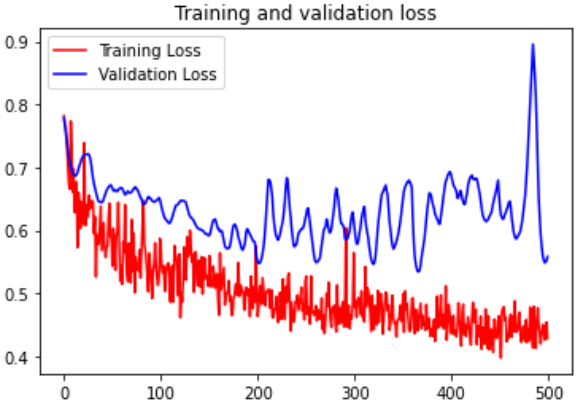
اما همچنان بین loss در training و validation اختلاف است :



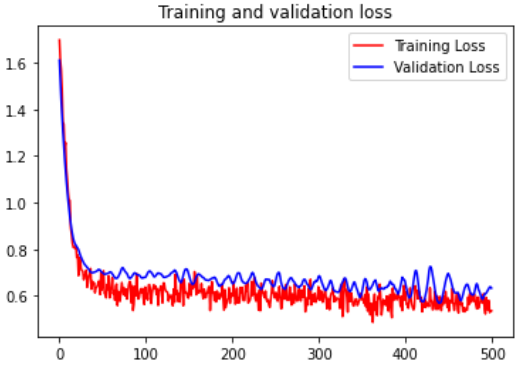
یکی دیگر از راه ها اضافه کردن regularization است . به کمک keras.reguralizers ، این rergularization ها را با شبکه اعمال کردم و ضریب را 10^{-3} گذاشتم و نتیجه بهتر شد :

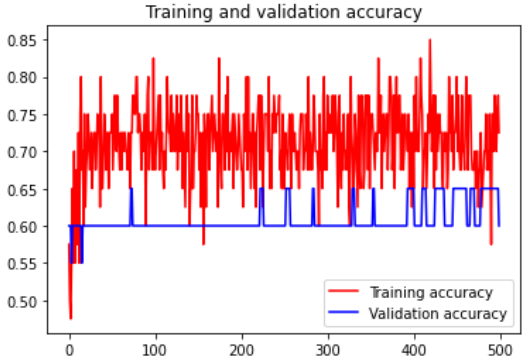


طبق نمودار accuracy ها همچنان به هم نزدیک ماندند ولی وضعیت نزدیکی loss ها بهتر شد :



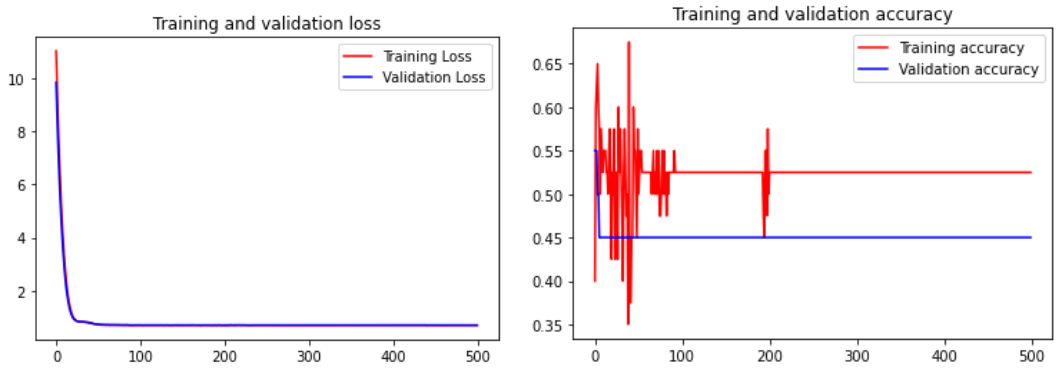
هنگامی که ضریب را به 10^{-2} افزایش دادم (تا مقدار وزن ها را کوچکتر کند) وضعیت همچنان بهتر شد :





دقت ها به هم نزدیک شدند و loss ها به هم خیلی نزدیک تر شدند .

اما هنگامی که مقدار را به ۰.۱ افزایش دادم نتیجه جالب نبود :



با اینکه دقت training از validation کمتر شد ولی هر دو دقت زیادی ندارند . همچنین loss افزایش پیدا کرد .پس بهترین ضریب 0.01 بود که نمودار ها و کدش در نوت بوک هستند .

دلیل کاهش دقت بعد از بیشتر کردن ضریب شاید این باشد که به خاطر کم تر کردن تاثیر وزن ها ، اجازه استفاده از آموخته های قبلی را به شبکه نمی دهد .