بسمه تعالی

دانشکده مهندسی کامپیوتر

یادگیری عمیق

نام استاد: دکتر محمدی

تمرین ششم

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

آبان ۱۴۰۰

۱. پاسخ سوال اول

سوالات عمومی:

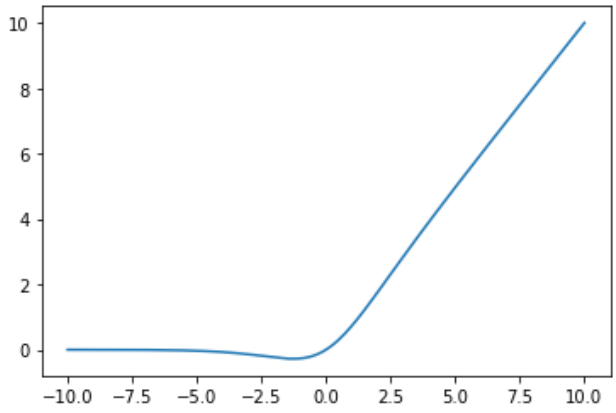
الف)

Swish(x) = x * sigmoid(beta*x)

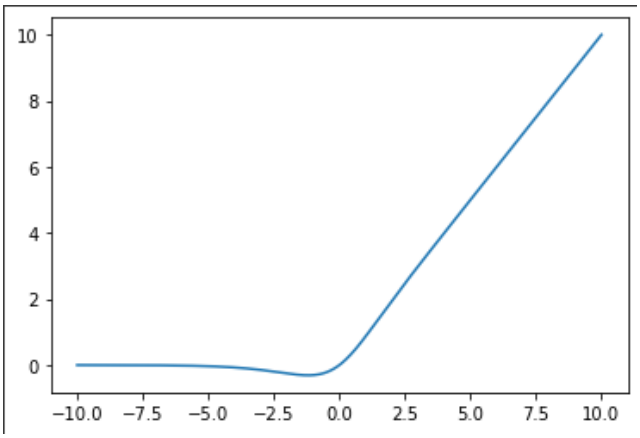
Mish(x) = x * tanh(ln(1+e^x))

نمودار ها در section=Question 1، در فایل HW6.ipynb رسم شده اند:

Swish diagram: (beta=1)



Mish diagram:

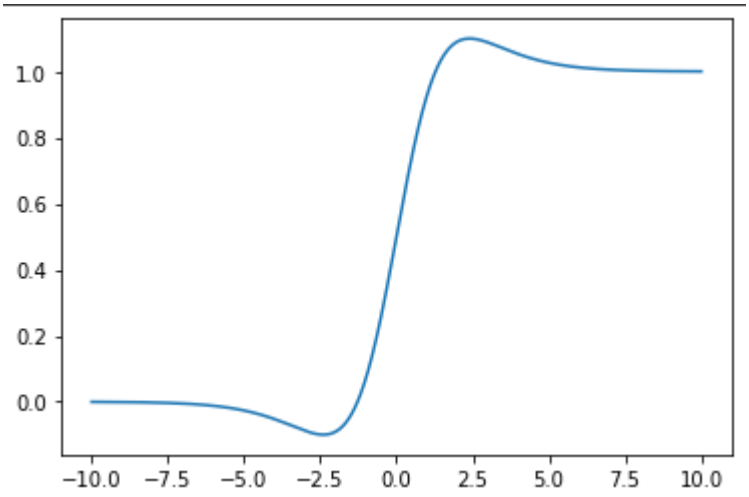


ب) طبق قاعده زنجیره ای مشتق به راحتی مشتق ها را محاسبه میکنیم:

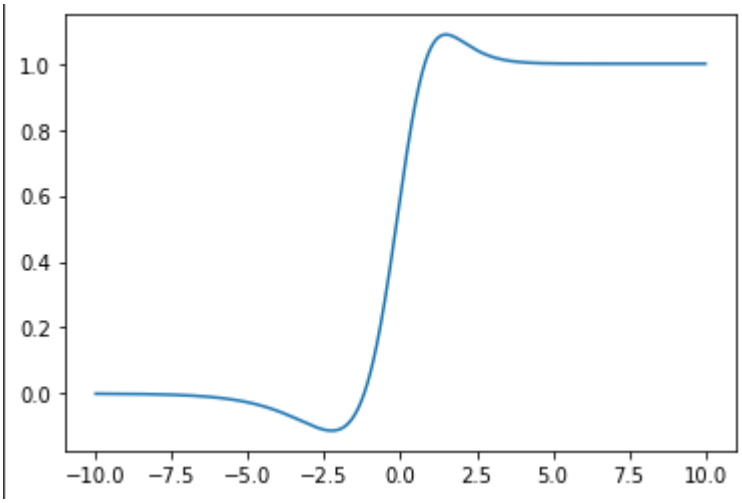
$d(swish(x))/dx = sigmoid(x) + x * sigmoid(x) * (1-sigmoid(x))$

$d(mish(x))/dx = \tanh(\ln(1+e^x)) + x * e^x/1+e^x * sech^2(\ln(1+e^x))$

swish derivative diagram:



mish derivative diagram:



ت) از مزایای relu میتوان به ساده بودن و سرعت بالای محاسبه خروجی آن، محدود نبودن این تابع از بالا و در نتیجه توانایی گرادیان برای بزرگ شدن در لایه های مختلف شبکه اشاره کرد. که این دو ویژگی را Sigmoid و tanh ندارند و به همین دلیل شبکه ها با Relu زودتر همگرا می شوند.

این دو تابع معرفی شده هم مانند relu از پایین محدود و از بالا نامحدود هستند. اما مزایای این دو تابع نسبت به Relu یک این است که بخاطر داشتن smooth ، وابستگی شبکه به نرخ یادگیری و وزن‌های ابتدایی کاهش میابد و این امر باعث میشود که سریعتر به نقطه بهینه برسیم. و دوم این که پدیده dying relu که به معنی محو شدن مقادیر زیر صفر در آن است، در این دو تابع وجود ندارد. چون هر دو به ازای مقادیر منفی خروجی کوچکی را خواهند داشت.

سوالات اختصاصی تابع فعالسازی Swish:

ث) با استفاده از این تابع می تواند با کنترل پارامتر بتا، نوع خروجی تابع را تغییر داد. که با میل دادن بتا به سمت بی نهایت تابع را به relu و با میل دادنش به سمت صفر، تابع را به تابع خطی نزدیک میکنیم. طبق صحبت نویسنده میتواند ۰.۵ درصد افزایش دقت به کمک آن داشته باشیم.

سوالات اختصاصی تابع فعالسازی Mish:

ج) وجود این پارامتر $\Delta(x)$ ، باعث می شود که گرادیان خروجی این تابع smooth تر باشد. که در تصویر ۴ هم سطح هموارتر آن مشخص است. و در نتیجه بهینه سازی سریع تر خواهد بود.

۲. پاسخ سوال دوم

الف) میانگین مربع خطاها (MSE) مقدار خیلی زیادی خطا را بر نمیگرداند. تفاوت اصلی با binary cross entropy هم همینجاست. چون این تابع با توجه به فرمول لگاریتمی که دارد، در صورتی که خطا خیلی زیاد باشد و شبکه کاملا اشتباه کند می تواند مقادیر زیادی را بازگرداند. در ابتدای روند آموزش هم طبیعتا وزن ها رندوم هستند و خطای شبکه بالاست. پس مقدار binary cross entropy به طور معنا داری بیشتر است. برای مثال اگر مقدار 0.5 پیشبینی شبکه برای یک خروجی که y=1 است را به این دو تابع بدهیم، BinaryCrossEntropy مقدار حدودا ۰.۷ و MSE مقدار حدودا ۰.۲۵ را بازمیگرداند. یعنی نقطه شروع (epoch=1) الگوریتم ما هم چنین حالتی بوده است.

ب) به مرور با زیاد شدن epoch، دقت روی داده های آموزشی و آزمایشی به ثبات می رسد. مقدار loss در داده های آزمایشی طبیعتا بیشتر است زیرا شبکه آن ها را ندیده است و وزن ها را مطابق آن ها تنظیم نکرده است. اما تفاوت اصلی بین MSE و BinaryCrossEntropy در این است که MSE اگر شبکه برای برخی مقادیر اشتباه باشد، مقدار خطای اندکی را برای آن ها بازمیگرداند که معمولا بین صفر تا ۱ است. در واقع تابع ضرر MSE تابعی محدود است. اما BinaryCrossEntropy اینطور نیست. شبکه روی داده آموزشی خطای کمی داشته اما روی آزمایشی که خطا شاید کمی بیشتر باشد، به علت مقادیر زیادی که این تابع برای خطاها بر میگرداند مقدار میانگین خطا هم بیشتر می شود.

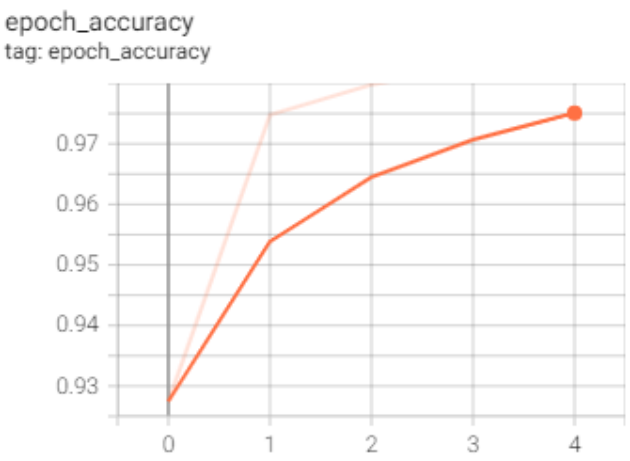
پ) زمانی که بیشترین دقت را روی داده ها آزمایشی داریم احتمالا بهترین زمان است. در BinaryCrossEntropy مطابق انتظار سرعد آموزش شبکه به خاطر گرادیان های بزرگتر، بیشتر بوده است. و epoch=60 به نظر بهینه میرسد. اما با 100 بار تکرار در MSE میبینیم که روند آموزش همچنان جلو می رود، پس ادامه دادن آموزش هم امری منطقی است و اگر منظور سوال حتما توقف در جای خاصی است، epoch=100 منطقی است.

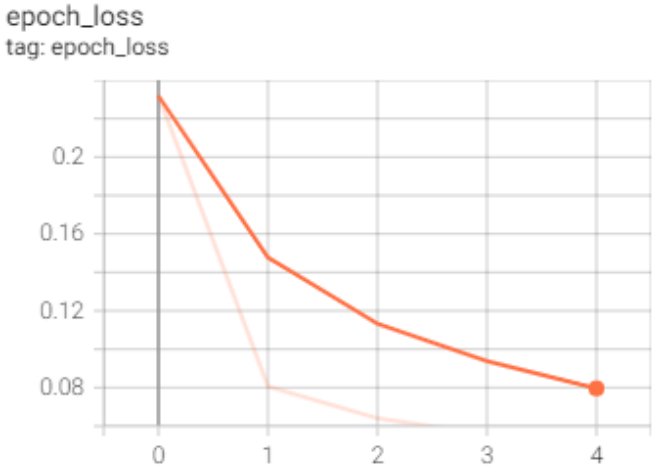
۳. پاسخ سوال سوم

پیاده سازی مربوط به این سوال در فایل HW6.ipynb انجام شده است. (لینک گوگل کولب)

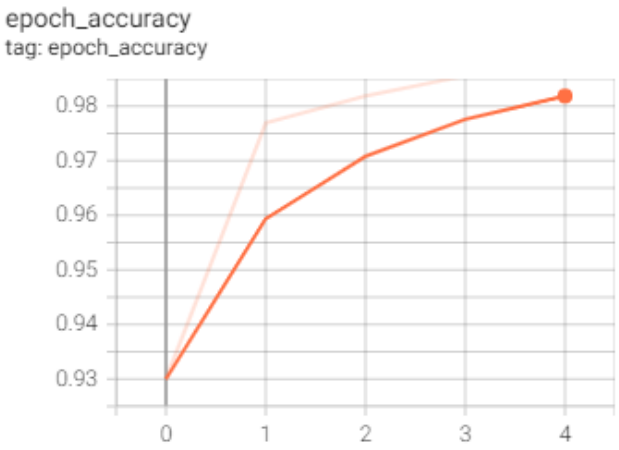
- نمودارها با مقادیر مختلف آلفا به عنوان ورودی تابع leaky relu در ادامه آمده است:

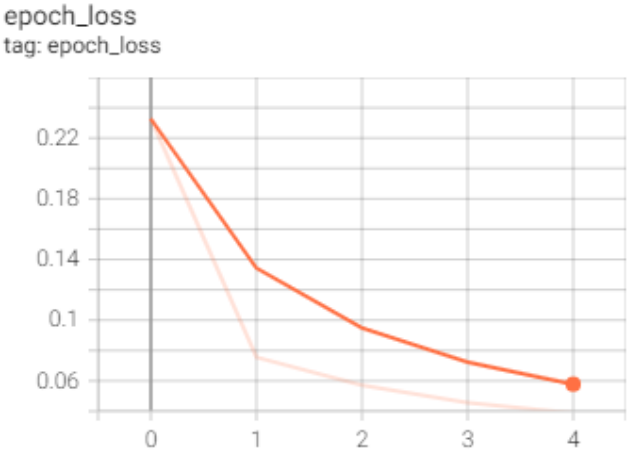
Alpha = 1 → test accuracy=98.22%



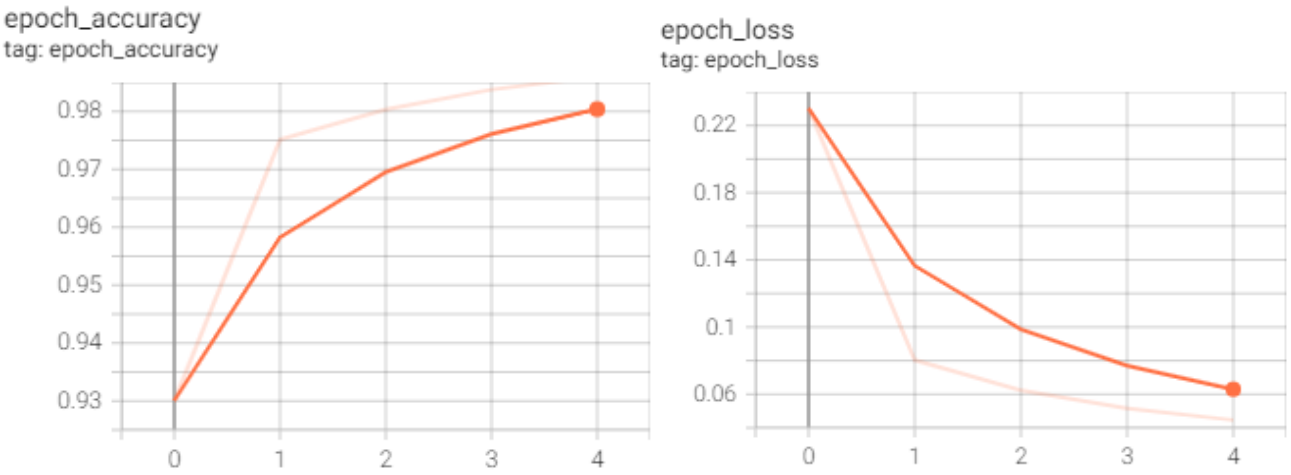


Alpha = 0.5 → test accuracy=97.46%

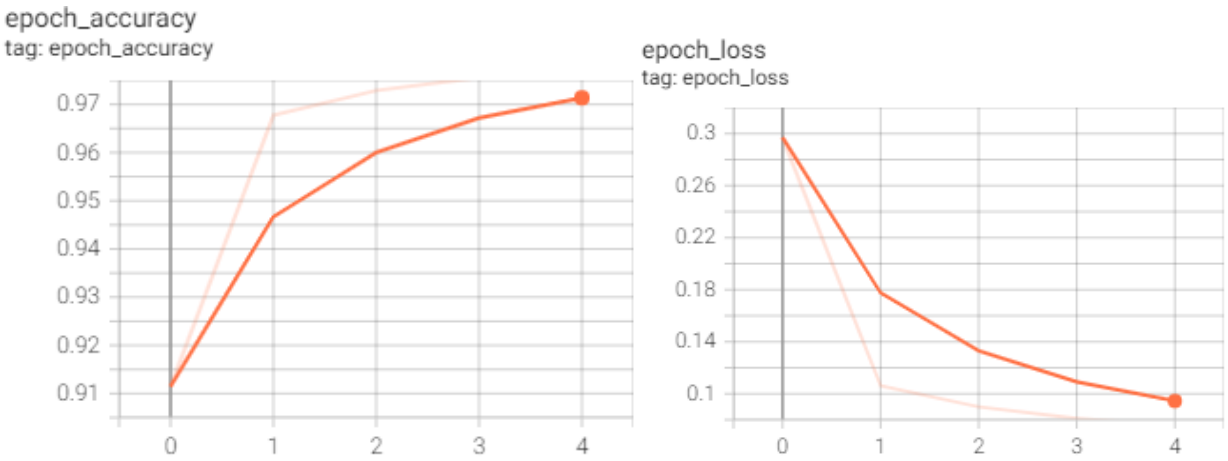




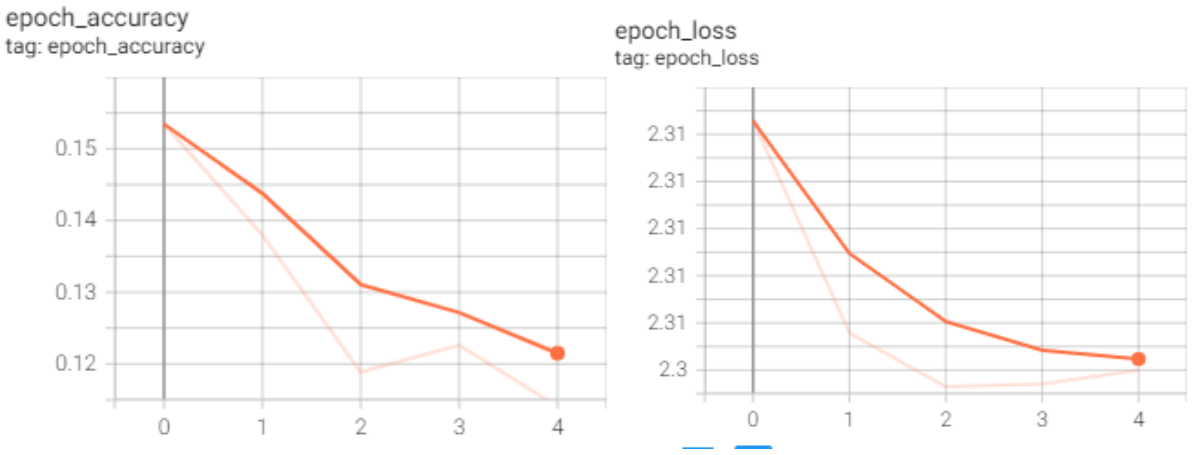
Alpha=0 → test accuracy=98.62%



Alpha=0.5 → test accuracy=97.82%

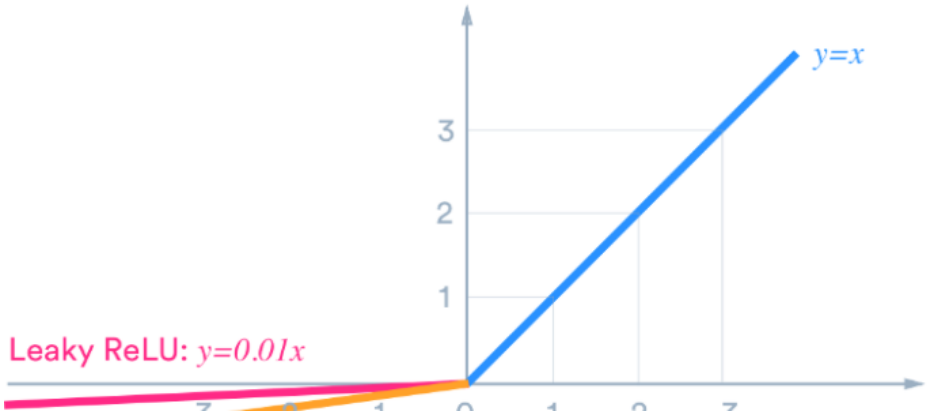


Alpha=1, test accuracy=10.74%



با alpha=1 نتایج عجیب شده است. هم دقت روی داده تست و هم داده آموزشی بسیار پایین است! همچنین loss هم در مقایسه با حالات قبلی تغییر بسیار کمی با جلو رفتن شبکه کرده است و مقدار بالایی دارد. چون وقتی آلفا ۱ باشد، انگار تابع ما f(x) = x است و در واقع تابع فعالسازی نداریم! در چنین حالتی لایه های مختلف شبکه بی معنی می شوند و همگی یک ویژگی را یاد میگیرند و انگار شبکه ساده تک لایه ای برای چنین مسئله پیچیده ای داریم و گرادیان ها از هر لایه به لایه بعد تفاوتی ندارند.

مدل هایی که alpha<=0 است تفاوت چندانی ندارند اما از نظر کم بودن loss و دقت روی داده تست، alpha=0 کمی بهتر عمل کرده است. چون آموختیم که شبکه با تابع relu بسیار زود همگرا می شود و همچنین تابع ساده ای است و محاسبه مشتق و خروجی آن از نظر پیچیدگی زمانی بهینه است. وقتی alpha=0 باشد انگار تابع فعالسازی ما relu است. در حالاتی که آلفا عددی نزدیک صفر باشد، مقادیر منفی را صفر در نظر نمیگیرم و از جهت برابر در نظر نگرفتن مقادیر زیادی باهمدیگر، leaky relu بهتر از relu است. اما باید توجه کرد که این ضریب یا منفی باشد و یا مثبت نزدیک صفر که تابع فعالسازی ما بی معنی نشود.

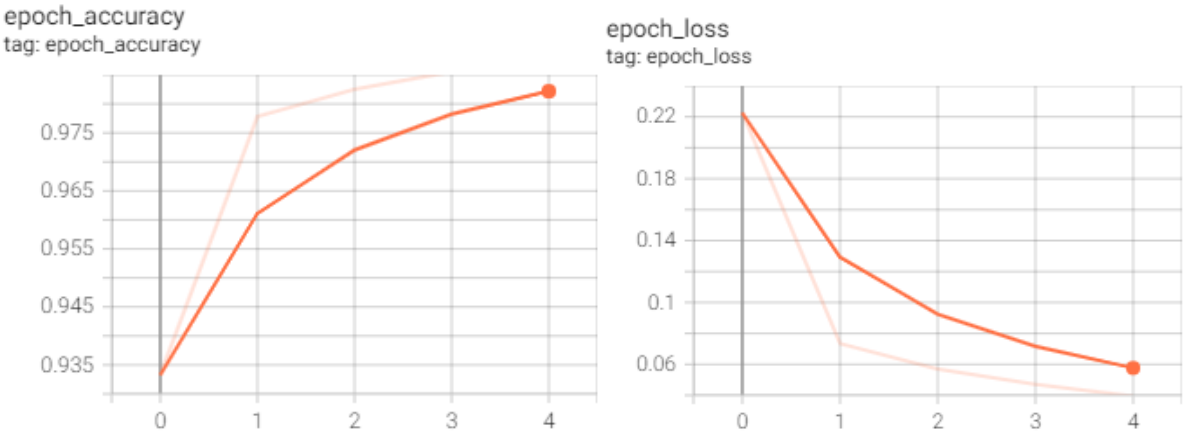


- نتایج ضرایب به دست آمده برای آلفا به شکل زیر بود:

```
[[[-0.36469597 -0.17768374 -0.121634   -0.22572663 -0.1658279
   -0.30103108 -0.17841119 -0.2352233 ]]]
[[[-0.14048888 -0.02653333  0.16946314  0.15941367 -0.22471881
    0.3128484   0.19629408 -0.2503699 ]]]
```

که آرایه اول نشاندهنده مقادیر آلفا برای لایه اول کانولوشن با تابع فعالسازی PRelu است و آرایه دوم برای دومین لایه است. این نتایج توضیحات و نتایج مربوط به قسمت قبل را تایید میکند. چون آلفاهای به دست آمده

همگی اعدادی مثبت یا منفی اما نزدیک به صفر هستند. و مقادیر مثبت هم از قدرمطلق مقادیر منفی کوچتر است تا مشکلی که در قسمت قبل در مورد alpha=1 داشتیم دیگر تکرار نشود. مقادیر loss و شکل نمودارها هم به شکل زیر به دست آمد:



که میبینیم مشابه مقادیر بهینه leaky relu است، چون در parametric relu، مدل مقادیر بهینه آلفا را یاد میگیرید.

منابع: medium