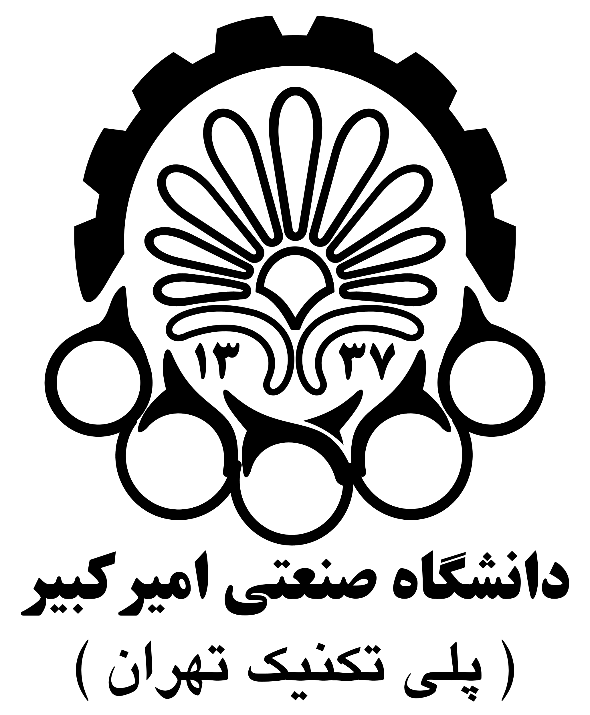
****

**دانشکده مهندسی کامپیوتر**

**تمرین چهارم درس شبکه عصبی**

**دکتر صفابخش**

**غلامرضا دار 400131018**

**بهار 1401**

فهرست مطالب

[سوال 1) 3](#_Toc103423430)

[سوال 2) 6](#_Toc103423431)

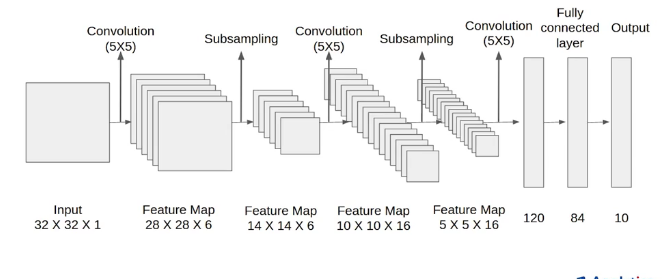
[سوال 3) 10](#_Toc103423432)

[سوال 4) 11](#_Toc103423433)

# سوال 1)

**معماری** **LeNet:** این معماری برای اولین بار در سال 1998 توسط Yann LeCun معرفی شد. کاربرد اصلی این معماری تشخیص ارقام دست نویس دیتاست mnist بود. سادگی و راحتی این معماری یکی از دلایل محبوبیت این مدل است. این معماری یک معماری بر پایه شبکه­های کانولوشنی است. در ادامه بیشتر با جزییات این معماری آشنا می­شویم.

معماری LeNet-5 همانطور که از اسمش مشخص است، شامل 5 لایه اصلی است. سه لایه کانولوشنی و 2 لایه کاملا متصل این معماری را می­سازند.

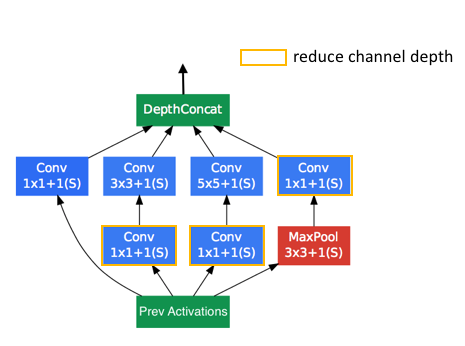


ورودی­های این معماری تصاویر تک کاناله 32 پیکسل در 32 پیکسل هستند. یک لایه کانولوشنی با اندازه 5 در 5، این تصاویر را به فیچرمپ­هایی با اندازه 28 در 28 تبدیل می­کند(تعداد فیلترهای 5 در 5 در این لایه 6 عدد در نظر گرفته شده اند). در ادامه یک لایه Average Pooling باعث می­شود که سایز فیچرمپ­ها نصف شود. پس از این مرحله یک لایه کانولوشنی با اندازه 5 در 5 دیگر و این­بار با تعداد 16 فیلتر به نتیجه Average Pooling اعمال می­شود. این روند مانند شکل یک ­بار دیگر نیز اتفاق می­افتد و پس از اعمال این سه مرحله کانولوشن، یک لایه کاملا متصل با تعداد 84 نورون، اطلاعات مرحله قبل را میگیرد و در نهایت یک لایه کاملا متصل دیگر که دارای 10 نورون است(هر نورون برای یک رقم) نتیجه دسته بندی نهایی را مشخص می­کند. این مدل تعداد 60 هزار پارامتر قابل تنظیم دارد که طبق استاندارد­های امروزی عدد کوچکی به حساب می­آید.

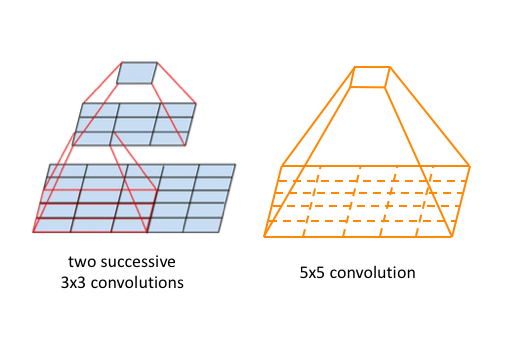
همانطور که دیدیم این مدل بسیار ساده و قابل فهم است و با اینکه بر روی تصاویر ارقام که اندازه کوچکی نیز دارند نتیجه قابل قبولی بدست می­آورد، در کاربردهای دیگر با اندازه تصاویر بزرگتر و تعداد کلاس بیشتر ممکن است نتیجه نامطلوبی بدست آورد. در کارهای اخیری که بر روی دسته بندی تصاویر رخ داده، نشان داده شده که معماری­هایی با عمق بیشتر و تعداد لایه های بیشتر عملکرد قابل قبول­تری دارند. در ادامه یکی از این مدل ها به نام Inception-v3 را مورد بررسی قرار می­دهیم.

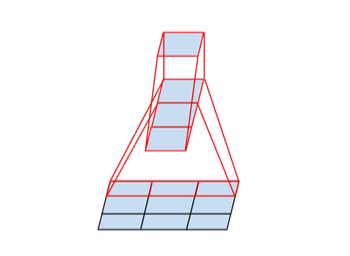
**معماری Inception-V3:** قبل از بحث در مورد معماری Inception-v3 بهتر است با نسخه ابتدایی تر آن یعنی GoogLeNet آشنا شویم. معماری Inception یا GoogLeNet در سال 2014 توانست در مسابقه تشخیص اشیا ImageNet مقام اول را کسب کند. این مدل از اولین مدل هایی بود که تصمیم گرفت با پهن کردن(نه عمیق کردن) مدل، قدرت مدل را افزایش دهد.

این معماری از تعدادی سلول Inception ساخته شده است. هر سلول Inception مانند تصویر زیر از تعدادی عمل کانولوشنی در سایز های مختلف ایجاد شده. نتیجه این عملیات کانولوشنی با هم ترکیب می­شوند و خروجی سلول Inception تولید می­شود.



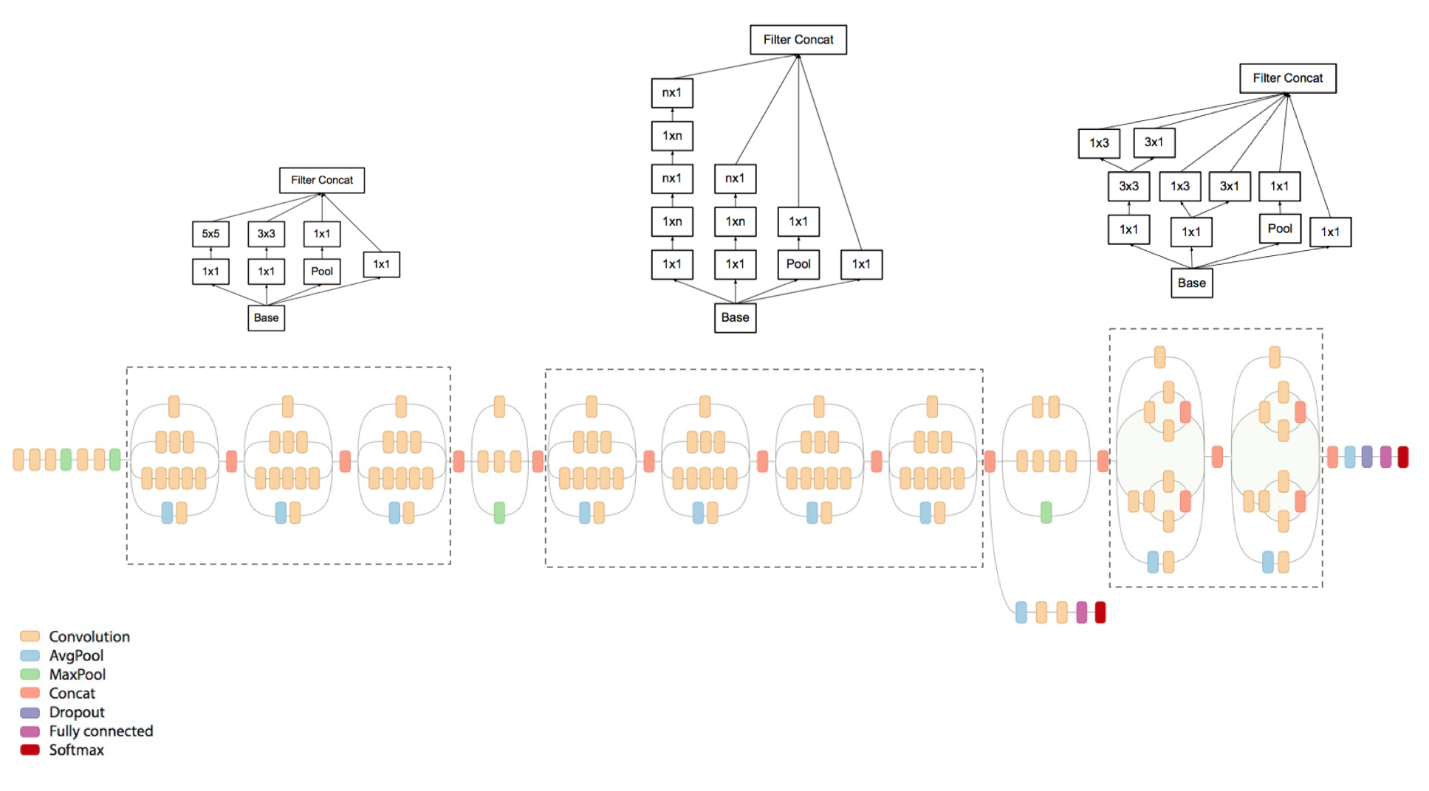
دو سال بعد نویسنده های مقاله Inception مقاله ای تحت عنوانRethinking the Inception Architecture for Computer Vision چاپ کردند که تعدادی تکنیک برای بهبود معماری قبلی را ارائه می­داد. یکی از این تکنیک­ها جایگزین کردن لایه های کانولوشنی 5 در 5 با دو لایه کانولوشنی 3 در 3 بود. همان­طور که در تصویر زیر می­بینید، این دو فیلتر نتیجه یکسانی دارند اما اعمال کردن دو فیلتر 3 در 3 بسیار بهینه تر است. تکنیک دیگری که مورد استفاده قرار گرفت، جایگزینی فیلتر 3 در 3 با دو فیلتر 3 در 1 و 1 در 3 بود. همه این بهینه سازی ها باعث شد معماری Inception-v3 بتواند با میزان زمان پردازشی مشابه، تعداد پارامترهای بسیار بیشتری داشته باشد و به نتایج قابل قبول تری دست یابد.





در تصویر زیر می­توان معماری کامل Inception-v3 را مشاهده کرد. این مدل شامل 23 میلیون پارامتر قابل تنظیم است.

این معماری علاوه بر تکنیک های ذکر شده، از یک سری دسته بند کمکی نیز بهره می­برد(Auxiliary Classifiers) که به نوعی به عنوان Regularizer عمل میکنند و تعمیم پذیری مدل را بالا می­برند.



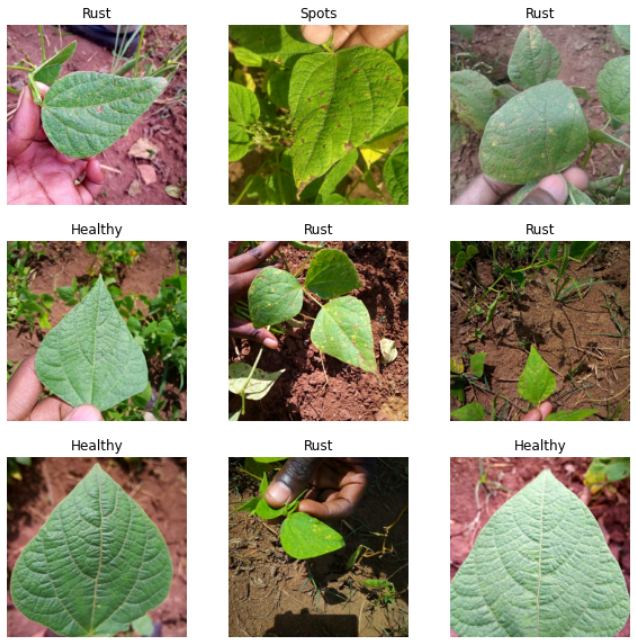
منابع:

<https://www.jeremyjordan.me/convnet-architectures/>

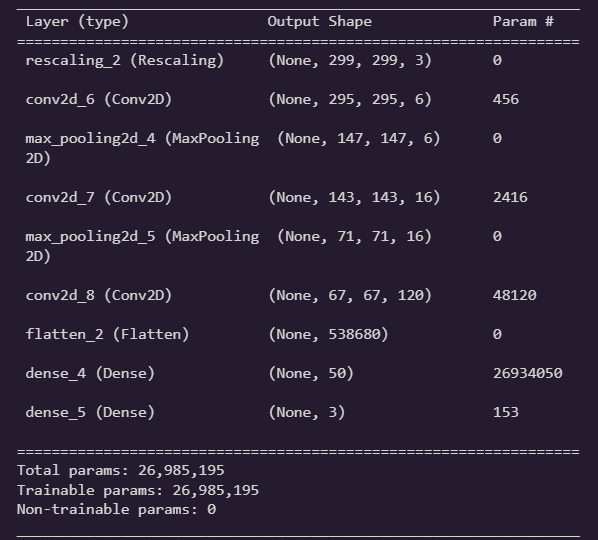
<https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/>

# سوال 2)

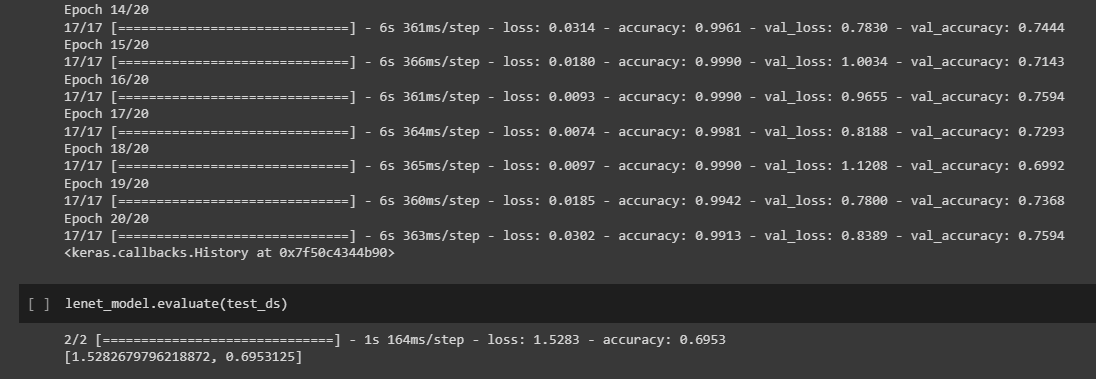
ابتدا با کمک ابزارtfds دیتاست beans را فراخوانی میکنیم. برای هماهنگی و سادگی انجام مقایسه ها، تصاویر را هم برای معماری LeNet و هم برای معماری Inception به یک اندازه در می­آوریم. اندازه انتخابی 299 در 299 است (وزن های مدلِ از قبل آموزش دیده Inception با این اندازه آموزش دیده اند).



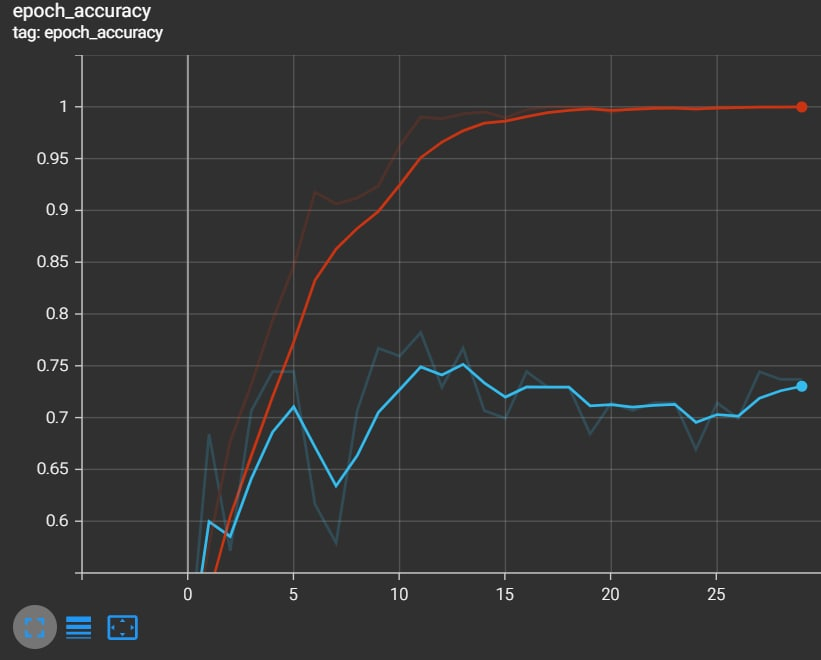
سپس معماری LeNet را پیاده سازی میکنیم.



پس از آموزش دادن مدل برای 20 epoch به صحت 69 درصد تست میرسیم.



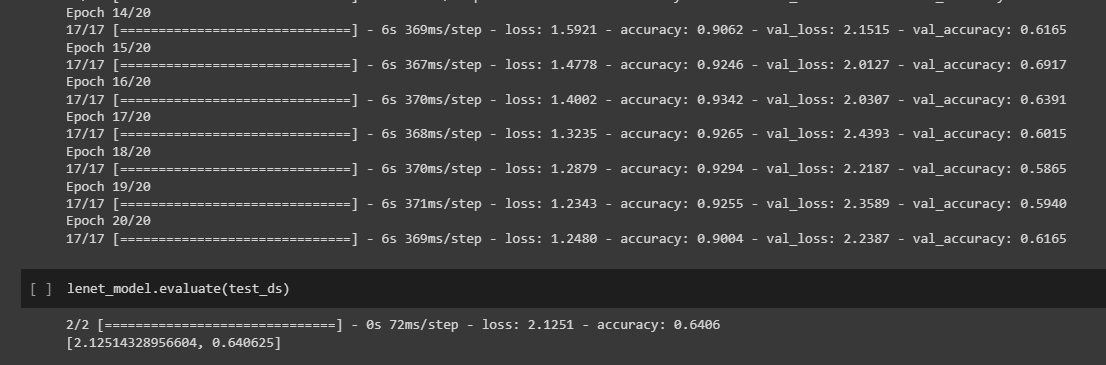
با بررسی نمودارهای accuracy, loss متوجه می­شویم که این مدل به شدت Overfit می­شود. پیچیده بودن مدل LeNet برای این حجم از داده (حدود 24 میلیون پارامتر)یکی از دلایل این بیش برازش است. (نمودار قرمز train\_acc و نمودار آبی valid\_acc است)



1. **آزمایش های مربوط به ریگولاریزیشن:**

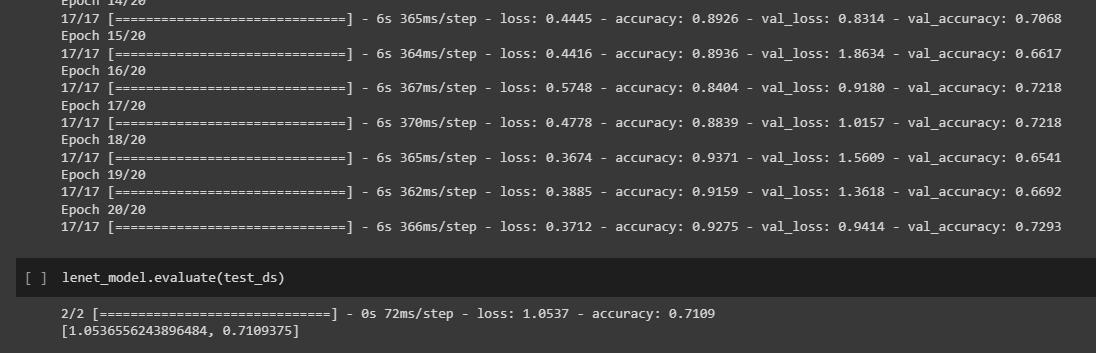
**اضافه کردن L1:**

اضافه کردن ریگولارایزر L1 به لایه­های کانولوشن باعث می­شود مدل از حالت بیش­برازشی که داشت خارج شود. (مقدار 0.001انتخاب شد)



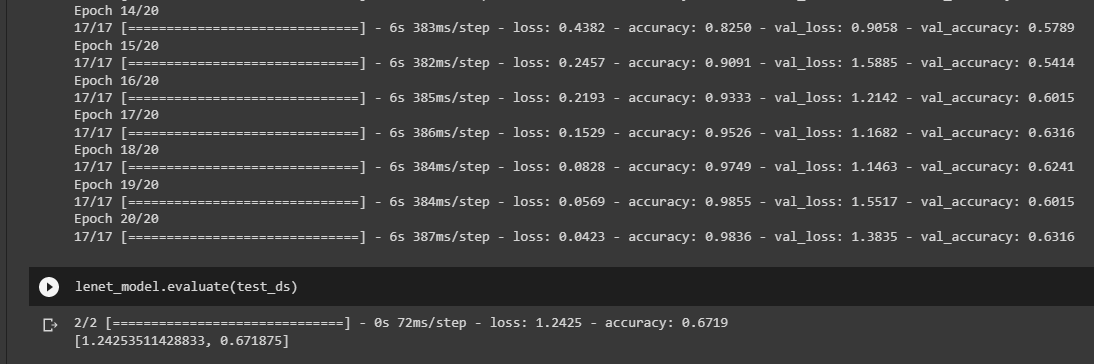
**اضافه کردن L2:**

اضافه کردن ریگولارایزر L2 به لایه­های کانولوشن باعث می­شود مدل از حالت بیش­برازشی که داشت خارج شود و همچنین نسبت به نتیجه L1 صحت بالاتری بدست می­آید.

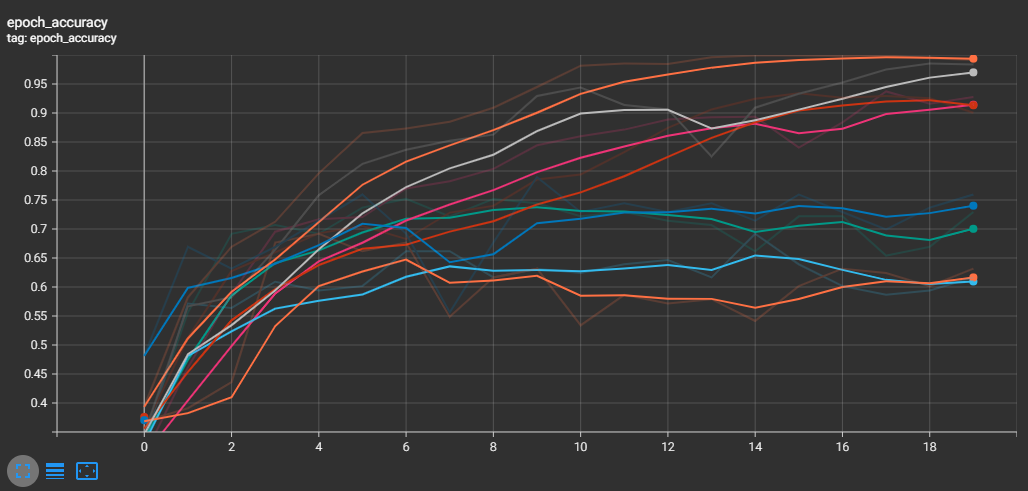


**اضافه کردن Dropout:**

افزودن لایه Dropout نیز یکی دیگر از راه­های کاهش بیش­برازش و افزایش تعمیم­پذیری مدل است که با توجه به آزمایش­های انجام شده نسبت به مدل اصلی نتیجه بهتری می­دهد. کار لایه Dropout غیرفعال کردن تعدادی از نورون­ها به صورت رندوم است. این اتفاق باعث می­شود یک سری ارتباطات بین نورونی که نباید وجود داشته باشند از بین برود و نورون ها به نورون های دیگر وابستگی کمتری داشته باشند. برای این سوال استفاده از Dropout 0.2 توانست از بیش برازش جلوگیری کند اما صحت دسته بندی را کاهش داد.



بخشی از آزمایش های انجام شده برای این سوال را مشاهده میکنید.



1. **آزمایش های مربوط به اندازه کرنل:**

برای آزمایش این بخش، از سه مقدار 3،5،7 برای اندازه کرنل استفاده کردیم که دقت تست را برای هر حالت در جدول زیر مشاهده میکنید.

|  |  |
| --- | --- |
| اندازه کرنل | صحت تست |
| 3 | 72.66% |
| 5 | 69.5% |
| 7 | 71.88% |

نتایج این آزمایش به دلیل وجود مقداری نوسان بین آزمایش ها و نزدیک بودن نتایج به یکدیگر قابل نتیجه گیری نیست اما اگر بخواهیم بهترین مدل را انتخاب کنیم مدل با اندازه کرنل 3 نتیجه بهتری داد.

1. **آزمایش های مربوط به تعداد کرنل:**

برای آزمایش این بخش، از سه حالت [3،8،60] و [6،16،120] و [12،32،240] استفاده کردیم که دقت تست را برای هر حالت در جدول زیر مشاهده میکنید.

|  |  |
| --- | --- |
| تعداد کرنل | صحت تست |
| [3،8،60] | 58.59% |
| [6،16،120] | 69.5% |
| [12،32،240] | آزمایش انجام نشد(مشکل رم) |

در مورد تعداد کرنل، با افزایش تعداد کرنل ها نتیجه بهتر شد اما از یک جایی به بعد به دلیل زیاد شدن پارامتر ها به مشکل رم برخورد کردیم و آموزش به اتمام نرسید.

# سوال 3)

تکنیک یادگیری انتقالی یک روش برای کوتاه کردن زمان آموزش و بهره گیری از مدل های از پیش آموزش داده شده و جنرال برای حل مسئله های خاص تر(که ممکن است دیتای کمتری داشته باشند) است.

**مرحله اول یادگیری انتقالی**

نحوه کلی اجرای یادگیری انتقالی به این شکل است که یک مدل قبلا توسط افراد دیگر و صرف هزینه پردازشی زیاد آموزش داده شده است. معمولا این مدل ها مدل های جنرالی هستند که مسائل خیلی کلی مانند تشخیص اشیا(از بین 1000 نوع لیبل مختلف) را حل میکنند. این مدل ها، علاوه بر اینکه یاد گرفته اند مسئله مورد نظر خود را حل کنند، یاد گرفته اند فیچرهای بسیار قوی و مناسبی را تولید کنند. در یادگیری انتقالی عموما هدف این است که از ان مدل های از پیش آموزش دیده بهره بگیریم تا این فیچرهای ارزشمند را برای مسئله خود بدون صرف هزینه زیاد تولید کنیم. معمولا در انتهای بسیاری از مدل های دسته بند، یک لایه تماما متصل برای دسته بندی داده به کلاس های مدنظر مسئله اصلی وجود دارد. ما این لایه را حذف میکنیم و یک لایه مخصوص مسئله خودمان قرار می­دهیم. همچنین لایه های مربوط به مدل اصلی را از حالت قابل آموزش در میاوریم تا وزن های از قبل آموخته شده تغییر نکنند و در واقع عملکرد لایه های ابتدایی(لایه های مربوط به استخراج به ویژگی) عوض نشود. سپس مدل جدید را بر روی داده آموزش خود(که ممکن است خیلی کوچک باشد) آموزش میدهیم و با اینکار وزن های مربوط به لایه آخر(دسته بند) را متناسب با مسئله خود آپدیت میکنیم. در انتهای این مراحل، مدلی داریم که قدرت استخراج ویژگی مدل اصلی را دارد و همچنین یاد گرفته است که ازین ویژگی ها به چه شکل استفاده کند تا ورودی مسئله ما را به خروجی مورد نظرمان نسبت بدهد.

**مرحله دوم یادگیری انتقالی(Finetune کردن مدل)**

پس از انجام مرحله قبل، می­توانیم برای بهبود نتایج یک کار دیگر نیز انجام دهیم. میتوان با آزادسازی وزن چند لایه آخر مدل اصلی(مدل ترین شده) وزن این لایه ها را نیز برای مسئله خود بهینه کنیم. چون معمولا در اکثر مدل ها، فیچر های استخراج شده در لایه های آخر، مخصوص تر هستند و ویژگی های استخراج شده در لایه های اول، کلی تر و عمومی تر هستند، این تصمیم را گرفتیم که فقط لایه های آخر را آزاد کنیم و برای مسئله خود بهینه کنیم. دلیل اینکه تمام لایه ها را نیز آزاد نمیکنیم این است که علاوه بر بار محاسباتی زیادی که برایمان دارد، ممکن است مدلمان، به دلیل داشتن پیچیدگی زیاد نسبت به داده ای که داریم، اورفیت شود.

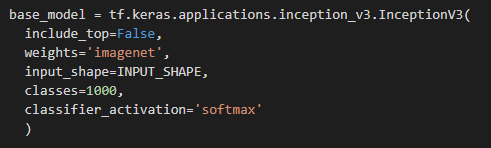
یک نکته مهم این است که حتما باید قبل از فاین تون کردن مدل، یکبار لایه آخری که اضافه کرده ایم را آموزش دهیم. اگر این کار را نکنیم باعث میشود خطاهای انتشار یافته به سمت لایه های مدل اصلی بسیار زیاد باشند و وزن های آموخته شده را خراب کند.

# سوال 4)

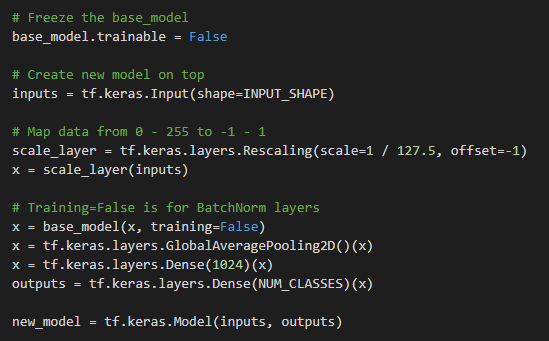
همانطور که در سوال 3 توضیح داده شد، در این بخش میخواهیم با استفاده از مدل آموزش دیده­ی Inception-v3 عملیات دسته بندی انواع برگ لوبیا را انجام دهیم. مدل Inception-v3 برای تشخیص انواع برگ های لوبیا آموزش دیده نشده است. این مدل برای تشخیص اشیا مختلف(از 1000 کلاس) در دیتاست ImageNet طراحی شده.

ما می­توانیم با استفاده از تکنیک یادگیری انتقالی، از فیچرهای آموخته شده در Inception-v3 و در حین یادگیری پترن­های بین تصاویر متنوع مجموعه ImageNet، بهره بگیریم و ادامه کار را از این نقطه شروع کنیم. در واقع با این کار، ما فرض میکنیم نیازی به استخراج فیچرهای خام از تصاویر نیست(عموما این مرحله بسیار ضروری است زیرا تصاویر خام برای مدل قابل درک نیستند) اما در این مثال، ما به Inception-v3 آموزش دیده دسترسی داریم که به طور خود به خود ویژگی ها را از تصاویر برای ما استخراج میکند.

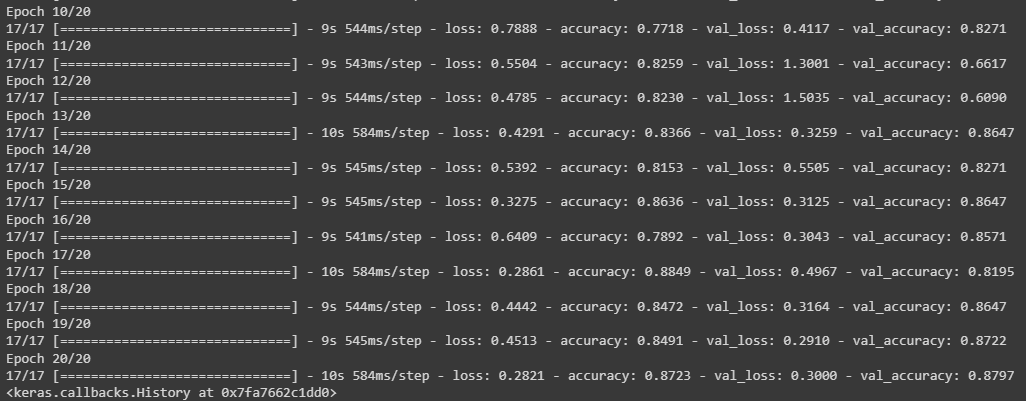
حال به پیاده سازی نکات گفته شده در بخش های قبل میپردازیم. ابتدا لازم است مدل Inception-v3 را از کتابخانه Keras فراخوانی کنیم. به هنگام فراخوانی، اعلام میکنیم که لایه آخر(دسته بند) این مدل را احتیاج نداریم و همچنین به وزن های از قبل تعیین شده این مدل نیازمندیم.

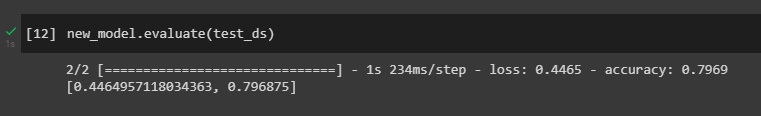


پس از فراخوانی مدل، وزن های آن را فریز میکنیم و با افزودن لایه های تماما متصل ذکر شده، یکبار مدل را آموزش میدهیم. همچنین اگر به doc اینسپشن در کراس مراجعه کنید میبینید که اعداد ورودی به این مدل باید بین منفی 1 تا 1 باشند بنابراین با افزودن یک لایه Rescaling این امر را محقق می­سازیم. تعداد نورون های لایه تماما متصل را 1024 در نظر میگیریم و یک لایه تماما متصل به اندازه 3 نورون نیز برای مرحله آخر دسته بندی اضافه میکنیم.



پس از آموزش این مدل برای 20 epoch به صحت ولیدیشن 87% و صحت تست 77% میرسیم.





این نشان می­دهد که مدل ما توانسته از فیچر­های استخراج شده به خوبی استفاده کند و روی مجموعه داده تست به 77 درصد تصاویر برچسب درست بزند. اما ما میتوانیم با استفاده از Finetuning عملکرد مدل را از این هم بالاتر ببریم.

برای اینکار ابتدا باید تصمیم بگیریم که چند لایه بالای مدل اینسپشن را آزاد کنیم. با بررسی های انجام شده و توضیحاتی که در اینترنت خواندیم، بهتر است لایه ها را به طور مستقل از هم آزاد و فریز نکنیم. بلکه لایه های مربوط به هم (یک بلاک) را همزمان یا فریز کنیم یا آزاد.

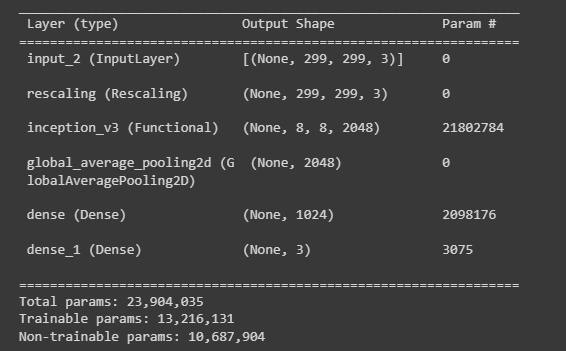
در این سوال سه حالت را بررسی میکنیم:

* آزادسازی 2 بلاک کانولوشنی بالا (لایه 249 به بعد)
* آزادسازی 3 بلاک کانولوشنی بالا (لایه 229 به بعد)
* آزادسازی 6 بلاک کانولوشنی بالا (لایه 133 به بعد)

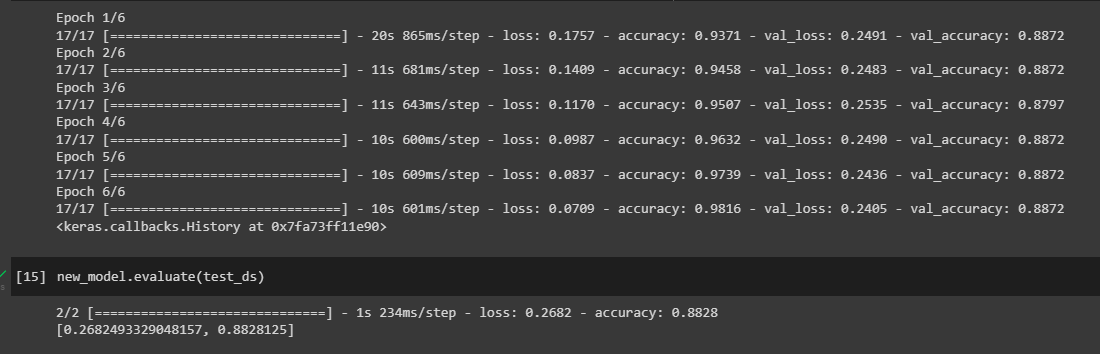
نکته دیگری که باید به خاطر بسپاریم این است که در این مرحله بهتر است از نرخ یادگیری بسیار کمتر استفاده کنیم تا وزن­های آموخته شده خراب نشوند و آرام آرام به سمت بهینه شدن حرکت کنند.

نکته دیگر تعداد epoch های لازم است که در این مرحله عموما بهتر است کمتر از مرحله قبل باشد زیرا فقط می­خواهیم کمی وزن­ها را به سمت بهینگی حرکت دهیم.

1. **آزادسازی 2 بلاک کانولوشنی**

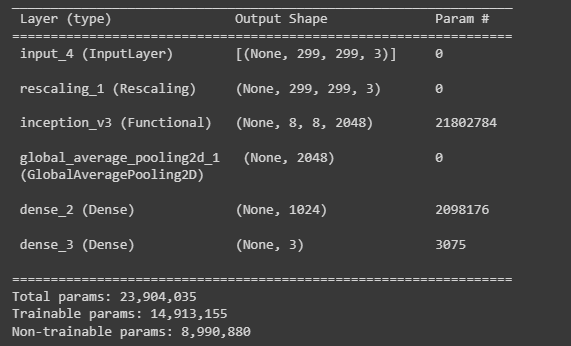


با آزادسازی 2 بلاک کانولوشنی، تعداد پارامترهای قابل آموزش به عدد **13** میلیون رسید. پس از آموزش این مدل برای تعداد epoch کم و نرخ یادگیری کم به صحت ولیدیشن و تست **88**% رسیدیم. که نسبت به مدل فاین تون نشده بهبود بزرگی است.

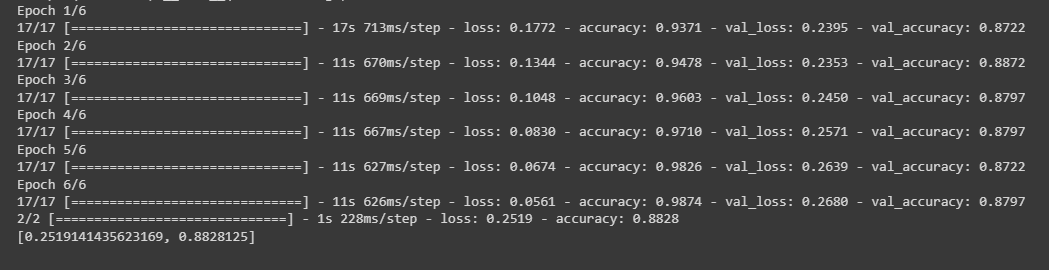


1. **آزادسازی 3 بلاک کانولوشنی**

با آزادسازی 3 بلاک کانولوشنی تعداد پارامترهای قابل آموزش به عدد **15** میلیون میرسد.

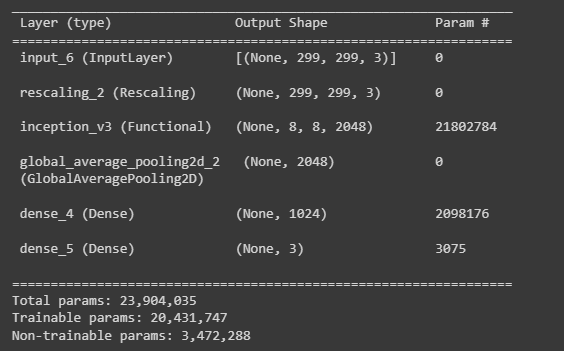


از نظر نتایج این مدل با مدل قبلی فرق چندانی نداشت. صحت آموزش و تست حدود **88%.**

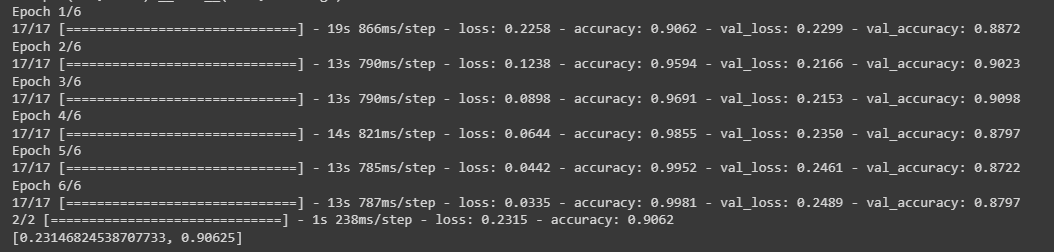


1. **آزادسازی 6 بلاک کانولوشنی**

با آزادسازی 6 بلاک کانولوشنی تعداد پارامترهای قابل آموزش به عدد **20** میلیون میرسد.



همانطور که در نتایج می­بینید این مدل بهترین نتیجه را داد و توانست به صحت تست **90%** برسد. اما باید دقت کنیم که مدل در حال اورفیت شدن نیز هست. در بخش جمع بندی به این قسمت اشاره ای میکنیم.



**نتیجه گیری در مورد تعداد لایه های فریز شده:**

افزایش تعداد لایه های قابل آموزش، می­تواند مدل را برای مسئله ما شخصی­سازی کند و عملکرد مدلرا در مورد مسئله ما را افزایش دهد. اما افزایش لایه ها به معنی افزایش تعداد پارامترهای قابل یادگیری و در نتیجه پیچیده شدن مدل است. میدانیم مدل پیچیده و تعداد داده آموزش کم معمولا به Overfitting می­انجامد بنابراین باید مراقب باشیم و با آزمون و خطا بهترین میزان لایه قابل آموزش را پیدا کنیم.

**مقایسه LeNet و Inception-V3 :**

همانطور که ذکر شد، معماری LeNet برای یک مسئله بسیار ساده تر از مسئله ما طراحی شده بود. با این وجود توانست دقت نه چندان بدی بدست بیاورد و البته بخاطر اینکه داده های ما هم از نظر تعداد و هم از نظر ابعاد برای این مدل بسیار زیاد بود دچار Overfitting شد. بنابراین با توجه به آزمایش هایی که دیدیم از نظر دقت دسته بندی، معماری Inception قطعا برنده است.

ازنظر سرعت همگرایی، مدل اینسپشن بهتر عمل کرد. البته باید توجه داشت که بخش عظیمی از آموزش Inception در مرحله قبل و توسط نویسنده های اصلی این مدل انجام شده و به این علت ما با تعداد epoch کم نیز به صحت بالایی رسیدیم.

از نظر تعمیم پذیری نیز معماری LeNet بیشتر مستعد بیش برازش بود و نیاز بود با استفاده از ریگولاریزیشن این بیش برازش را کاهش دهیم.

منبع مفید برای یادگیری Transfer Learning : <https://keras.io/guides/transfer_learning/>

پایان