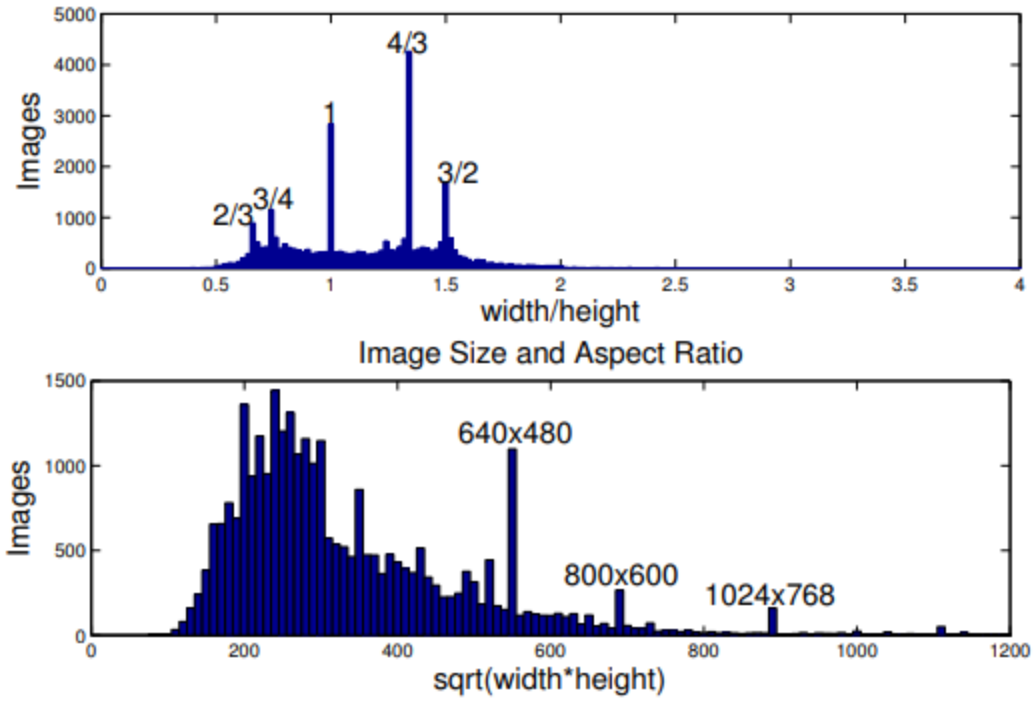
دسته بندی تصاویر مجموعه Caltech 101:

پس از بررسی دیتاست، اطلاعات کلی آن به شرح زیر بدست آمد:

- **تعداد کلاس ها : 101 کلاس**
- **تعداد کل تصاویر: 9144**
- **میانگین تعداد تصویر در هر کلاس :59**
- **حداقل تعداد تصویر در هر کلاس : 31**
- **حداکثر تعداد تصویر در هر کلاس : 800**

همین طور تصاویر هر کلاس در اندازه های مختلف هستند، لذا برای استفاده از تصاویر به عنوان ورودی CNN لازم است ابتدا اندازه همه تصاویر را یکسان کنیم.
اطلاعاتی در مورد ابعاد و نسبت اندازه های تصاویر هر کلاس:



3 دسته از تصاویر دارای تعداد خیلی بیشتری تصویر نسبت به بقیه دسته ها هستند. به عبارت دیگر دیتاست balance نیست که این موضوع می تواند بر روی کارایی مدل تاثیر گذار باشد. در ابتدا بدون توجه به این مشکل کار را پیش می بریم. در نهایت چند روش را برای حل این مشکل بررسی خواهیم کرد. از روش های موجود می توان به:

- **حذف داده از کلاسی که تعداد داده های بیشتری دارد**
- **دادن وزن به مقادیر احتمال هر کلاس در loss function مورد استفاده در فرآیند آموزش**
- **استفاده از روش های data augmentation برای رساندن تعداد تصاویر در کلاس های با تعداد تصویر کم به حالت عادی**

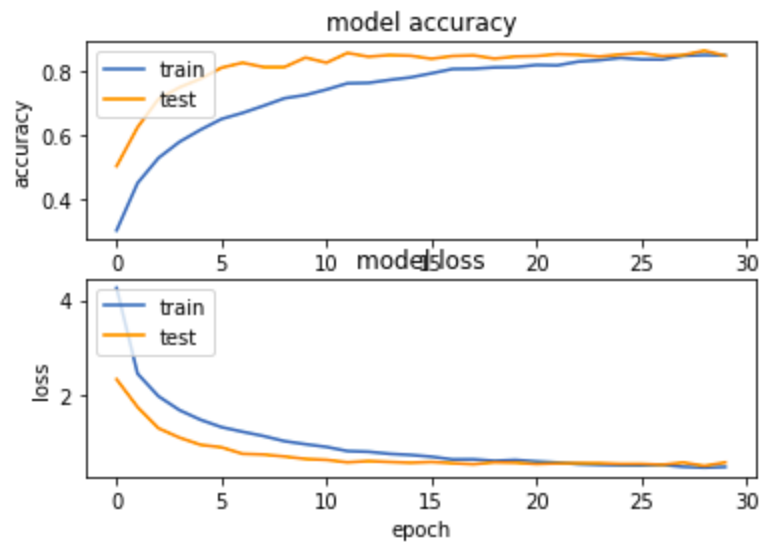
اشاره کرد.

با بررسی های صورت گرفته، با توجه به تعداد بالای دسته ها (101 دسته) و تعداد نسبی کم تصاویر آموزشی در هر کلاس، نمی توان انتظار داشت با طراحی و آموزش یک CNN از صفر، به نتایج قابل قبولی رسید. لذا تصمیم بر آن شد که از یکی از شبکه های آموزش دیده شده برای دسته بندی یا تشخیص تصاویر (مثل ***Vgg-16*** یا ***resnet***) استفاده شود.
برای این کار، لایه آخر این شبکه ها (لایه دسته بند) را حذف می کنیم. پس از این که شبکه را load کردیم، از لایه های convolution آن برای استخراج ویژگی های تصاویر استفاده می کنیم و از داده های بدست آمده، که برای هر تصویر یک بردار ویژگی است، برای آموزش یک دسته بند ساده (در این جا یک شبکه عصبی dense دولایه) استفاده می کنیم.

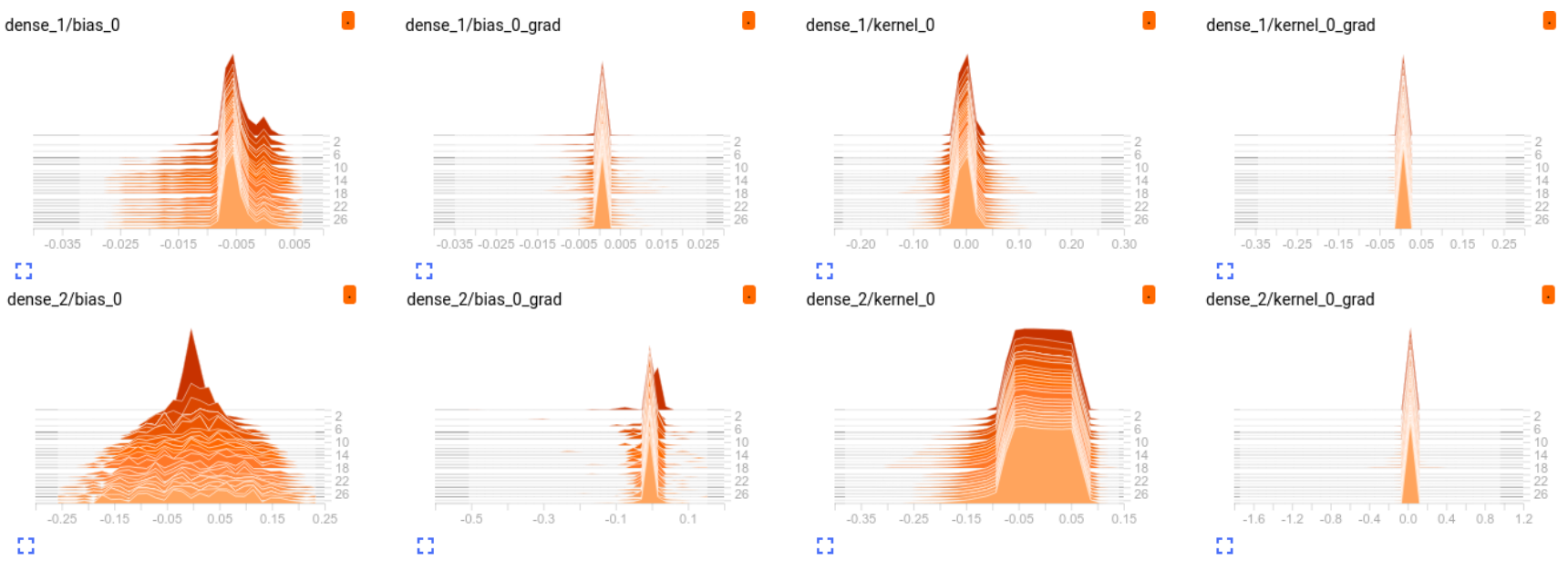
برای پیاده سازی به دلیل راحتی کار و ویژگی های فوق العاده موجود برای کتابخانه ***keras***، مثل **Tensorboard** که در این تمرین به وسیله آن فرآیند یادگیری شبکه و بقیه اطلاعات لازم را زیر نظر می گیریم، استفاده شده است.
همین طور شبکه ی ***vgg-16*** نیز به دلیل تجربه کار قبلی با هر دو شبکه یاد شده و ساده تر بودن ساختار ***vgg-16*** نسبت به ***resnet*** و سریع تر بودن آن با توجه به دقت نزدیکی که استفاده از این شبکه ها برای ما به ارمغان خواهد آورد، انتخاب شده است. از طرفی با توجه به نزدیکی تعداد کلاس ها با دیتاست ***Imagenet*** از وزن های بدست آمده از فرآیند train شبکه ***vgg-16*** با دیتاست ***Imagenet*** استفاده می کنیم.
از آن جایی که load کردن همه ی داده ها بر روی حافظه کار منطقی و شدنی ای نیست، برای load کردن داده ها از توابع ***generator***

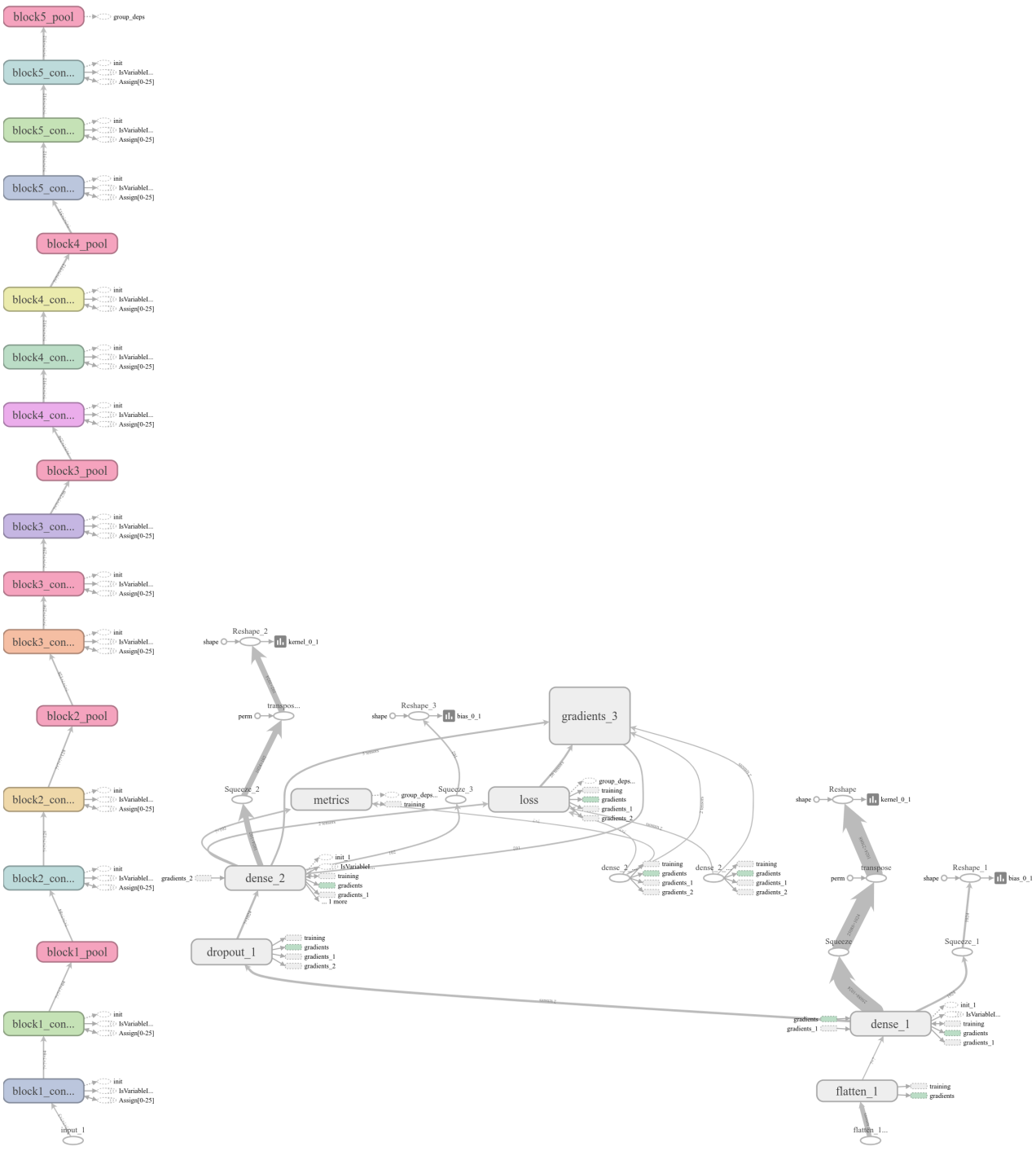
استفاده شده است. برای این کار لازم است که فولدر بندی داده ها حالت خاصی داشته باشد.
در ابتدا اسکریپت هایی برای دانلود داده ها بر روی colab، خروج داده ها از حالت فشرده و فولدر بندی درست آن ها نوشته شده است.
همین طور تنظیمات اولیه ای برای استفاده از **Tensorboard** در colab انجام شده است. برای مشاهده پنل **Tensorboard** ، پس از اجرای فرآیند، باید بر روی لینک موجود در 5 امین cell کلیک کنید.
عملیات تغییر اندازه عکس ها نیز به صورت خودکار توسط **generator** ها انجام می شود.
پس از انجام آماده سازی های لازم و load کردن وزن ها، داده ها را از لایه های **convolutional** عبور داده و بردار ویژگی های حاصله را ذخیره می کنیم. این کار با تابع **save*bottlebeck*features** انجام می گیرد.
در مرحله ی بعدی باید دسته بندی که مد نظر داشتیم را آموزش دهیم. برای این کار بردار های بدست آمده را دوباره load می کنیم و یک شبکه عصبی ۲ لایه ساده که تعداد خروجی های آن برابر با تعداد کلاس ها و اندازه ورودی آن برابر با بعد بردار های ویژگی بدست آمده در مرحله قبل است، ساخته و فرآیند آموزش را بر روی آن اعمال می کنیم.
با توجه به کم بودن داده ها امکان **overfit** شدن مدل بر روی داده های آموزشی بالاست؛ همین طور برای شبیه سازی فرآیندی مشابه با ***ensambling***، تعداد پارامتر های این ۲ لایه را نسبتا زیاد در نظر گرفته و از یک لایه ***dropout*** با مدار *0.75* بین دو لایه نهایی استفاده می کنیم.
تنظیمات و پارامتر های مورد نیاز برای ساخت شبکه ها و آموزش آن ها (مثل تعداد پارامتر های لایه های **dense** یا **learning rate** و ضریب **regularization**) با توجه به تجربیات مشابه قبلی تعیین شده اند.
در تمامی مراحل مقدار **۱۵ درصد** از داده ها را برای **validation** در نظر گرفته و عملیات آموزش را بر روی بقیه داده ها انجام می دهیم.
با توجه به میزان **dropout بالا** نیاز به حداقل ***20 تا 30 epoch*** در فرآیند آموزش است. در نهایت دقت بدست آمده بر روی مجموعه **validation** ، حدود ***84.7%*** است که با توجه به مجموعه تصاویر عدد قابل قبولی است. نمودار های دقت بر حسب epoch علاوه بر این که در Tensorboard قابل مشاهده اند، به صورت پیوست نیز ارسال می شوند. اطلاعات کاملی در مورد فرآیند آموزش و ساختار شبکه در پنل Tensorboard وجود دارد که بررسی آن ها خالی از لطف نیست.

```
[INFO] accuracy: 84.66%
[INFO] Loss: 0.5707652563620919
```



نمونه دیتای موجود در پنل Tensorboard




block5_pool
block5_con...
block4_pool
block4_con...
block3_pool
block3_con...
block2_pool
block2_con...
block1_pool
block1_con...
input_1
Reshape_2
transpos...
Reshape_3
bias_0_1
gradients_3
Squeeze_2
metrics
Squeeze_3
loss
dense_2
dropout_1
Reshape
transpose
Reshape_1
Squeeze
Squeeze_1
dense_1
flatten_1
flatten_1...
block1_con...[0-46]
Assign[0-25]
training
gradients
gradients_1
gradients_2
Adam
Placeholder[0-25]