امیرحسین احمدی ۹۷۵۲۲۲۹۲ – تمرین هشتم

۱- الف) Dropout یکی از روش‌های Regularization میباشد که برای جلوگیری از Overfitting استفاده میشود. در این روش در هر مرحله از آموزش تعدادی از نورون‌های لایه را به صورت تصادفی نادیده میگیریم و آموزش را با بقیه واحدها انجام میدهیم. با این کار وابستگی نورون‌های آن لایه به هم کمتر می‌شود و مدل بهتر ترین می‌شود.

از نظر فنی، در هر مرحله نورون‌ها را یا به احمال 1-p از شبکه حذف میکنیم و یا با احتمال p نگه میداریم. P را معمولن برابر با .۵ در نظر میگیریم. اگر این مقدار را خیلی پایین بیاوریم، شبکه هر دفعه با تعداد کمی نورون train می‌شود و که باعث Underfit شدن شبکه می‌شود. همچنین در صورت بالا بردن احتمال p تاثیر آن کم می‌شود و امکان دارد دوباره Overfit شویم. توصیه می‌شود که در شبکه‌های کانولوشنی در لایه‌های ابتدایی مقدار p کم نگذاریم و هر چه جلو میرویم مقدار آن را کاهش دهیم.

ب) اگر مقدار p را افزایش دهیم، تعداد نورون‌های مورد استفاده بیشتر شده و ظرفیت مدل افزایش پیدا میکند و مستعد Overfitting میشود و با کاهش p ظرفیت مدل کاهش یافته و خطر Undefitting دارد، به همین دلیل باید p را به دقت انتخاب کنیم.

۲- لایه Fully connected به لایه‌ای میگویند که هر نورون به همه ی نورون های لایه قبلی خود متصل است، به همین دلیل میتواند ویژگی‌های کلی تصاویر را یاد بگیرد و در شبکه های کانولوشنی بیشتر برای لایه های آخر استفاده می‌شود چون شبکه باید یک جواب کلی به مجموع ورودی ها بدهد و باید تمام ویژگی‌های استخراج شده در شبکه دخیل باشند.

لایه Convolutional مهم‌ترین لایه در شبکه‌های کانولوشنیست. عملکرد آن به این صورت است که تعدادی کرنل (بخش‌های کوک از ورودی، مثلا بخش‌های ۵*۵ روی ورودی با سایز ۱۲۸*۱۲۸) بر روی ورودی‌ها در نظر میگیرد و از آن‌ها ویژگی‌های محلی استخراج میکند. در این لایه وزن‌های کرنل ها با هم مشترک است، کرنل‌های یک لایه یک ویژگی مشترک را از قسمت‌های مختلف ورودی استخراج می‌کنند و یاد می‌گیرند.

لایه Locally connected مانند لایه Convolutional روی ورودی کرنل تعریف کرده با این تفاوت که وزن‌های هر کرنل با یکدیگر متفاوت است و درواقع وابستگی نورون‌های این لایه کم می‌شود و هر کدام می‌توانند ویژگی‌های جداگانه‌ای را یاد بگیرند. مشکل اصلی این لایه مقدار بسیار بالای

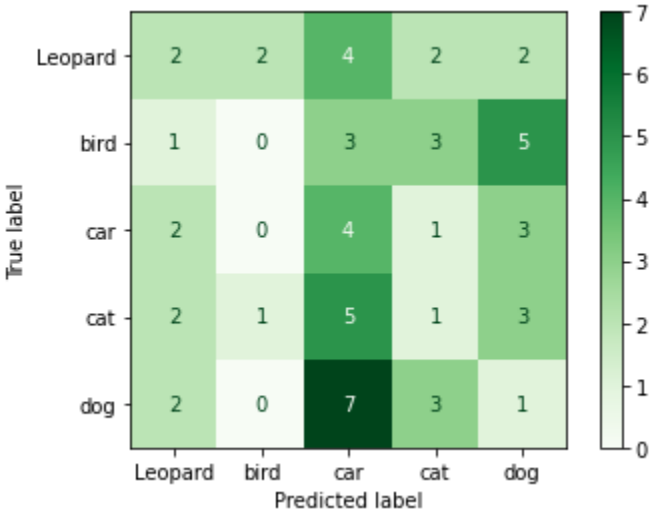
پارامترهاست که باید به ازای هر کرنل مجموعه وزن‌های جدا نگه داریم ولی با توجه به وابسته نبودن کرنل‌ها می‌توان با استفاده از GPU به صورت موازی آن‌ها را اجرا کرد.
در کل می‌توان نتیجه گرفت استفاده از لایه‌های Convolutional و درصورت داشتن ریسورس کافی، Locally connected در لایه‌های ابتدایی برای استخراج ویژگی‌های جزعی بخش‌های مختلف ورودی و استفاده از Fully connected در لایه های پایانی و برای جمع بندی ویژگی‌های استخراج شده و نتیجه گیری کلی مناسب است.

۳- برای پارت اول سوال بدون data augmentation، فقط دیتای موجود را از درایو خواندم و برای ImageDataGenerator فقط پارامتر rescale برای نرمالایز کردن عکس ها ورودی دادم. همانطور که خواسته شده بود یک مدل با ۴ لایه کانولوشنی که فیلتر همه را ۳۲ و کرنل را (5,5) در نظر گرفتم، دو لایه dense در آخر، که یکی با ۱۲۸ نورون و بعد از لایه آخر، و دیگری با ۵ نورون به دلیل ۵ کلاسی بودن مسئله (به دلیل زیاد بودن پارامترها و کند بودن مدل، ۴ لایه maxpooling با pool_size (3, 3) بین لایه های کانولوشنی در نظر گرفتم). Activation تمام لایه ها را elu و لایه آخر به دلیل سینگل کلس بودن مسئله softmax در نظر گرفتم. در تمامی مدل ها لاس فانکشن را categorical_crossentropy به دلیل categorical بودن مسئله و optimizer را آدام در نظر گرفتم و تمام مدل ها را در ۴۰ ایپوک ترین کردم.
همانطور که در نوت بوک قابل مشاهده است دقت مدل اول از epoch ۱۷ به بعد برابر ۱۰۰ است ولی دقت ولیدیشن آن در آخر برابر با ۵۵ درصد شده است که به وضوح overfit شدن مدل را نشان میدهد.
برای پارت دوم بدون تغییر مدل، تعداد زیادی data augmentation انجام دادیم. (شیفت دادن تصاویر به چپ و راست، برش دادن، زوم کردن تصاویر، چرخاندن و فلیپ کردن افقی تصاویر) اینبار دقت داده های ترین و ولیدیشن بسیار نزدیک به هم و به ترتیب برابر با ۷۷ درصد و ۶۴ درصد بودند که نشان از تاثیر گذاری data augmentation در جلوگیری از overfitting و نتیجه بهتر دارد.
در پارت سوم با اضافه کردن dropout بعد از flatten و لایه اول dense نتایج را بررسی میکنیم. وقتی با مقدار dropout را برابر با 5. میگذاریم میبینیم که نتیجه نسبت به پارت دو خیلی کمتر overfit شده و دقت ترین و ولیدیشن تقریبا برابر است (۵۹ و ۵۸) ولی دقت نسبت به حالت دو کمتر است، زیرا Regularization های زیاد باعث شده مدل Underfit بشه. برای مقدار 2. حتی دقت

ولیدیشن بالاتر از ترین است ولی دقت خیلی پایین و برابر با ۵۲ و ۴۹ است. با زیاد کردن ورودی dropout به 7. میبینیم که دقت بالا میرود و به ۷۰ درصد میرسد.

امتیازی: د) برای مدل آخر مقدار percision، recall و f1 را با استفاده از تابع classification_report نمایش دادیم که به توضیح هر کدام میپردازیم.
فرمول پرسیژن در مسائل دو کلاسه برابر با TP / (TP + FP) است. در واقع با کاهش false positive مقدار آن افزایش میابد و معیار خوبیست برای زمانی که میخواهیم false positive را کاهش دهیم. مثلا در مسئله تشخیص دادن ایمیل های اسپم، میتوان اندازه گرفت که تا چه اندازه زمانی که یک ایمیل را اسپم تشخیص داده ایم واقعا اسپم بوده است؟
فرمول recall برابر با TP / (TP + FN) است و در مسئله اسپم، معادل این است که ما چند درصد اسپم ها را درست تشخیص داده ایم. به همین دلیل وقتی نیاز داریم که false negative را کم کنیم recall میتواند معیار خوبی باشد.
F1-score ترکیبی از دو معیار بالاست و زمانی به درد میخورد که false negative و false positive را در تعادلی از یکدیگر نگه داریم.

امتیازی: ه) confusion_matrix نمودار بسیار خوبیست که در مسائل دسته بندی به ما کمک میکند تا بهتر بتوانیم نقاط ضعف و قوت مدل را در قالب یک جدول n*n برای مسئله های با n کلاس ببینیم.



مثلا برای شکل بالا که برای مدل آخر این سوال کشیده شده میتوانیم ببینیم که کلاس سگ خیلی زیاد با کلاس ماشین اشتباه گرفته شده و کلاس ماشین در اکثر مواقع درست تشخیص داده شده و در کل مدل خیلی به سمت کلاس ماشین بایاس شده است.