به نام خدا

شماره پروژه انتخابی: ۱

برای پیاده سازی این پروژه از لینک یوتوب زیر کمک گرفته شده است:

https://www.youtube.com/watch?v=AALBGpLbj6Q

این لینک شبکه gan را با استفاده از کتابخانه تنسورفلو پیاده سازی میکند بنابراین با کمک آن میتوان یک کد اولیه نوشت و سپس با استفاده از راهنمای متلب ساختار generator و discriminator و باقی تنظیمات را تطبیق داد.

تمامی کد ها در نوت بوک پیاده سازی و روی گوگل کولب ران شده اند.

اولین اجرا مربوط به پوشه GAN_Flower_500_Epochs است. در این پوشه ۴ فایل مشاهده میشود. فایل نوت بوک آن شامل تمام کد های اجرایی میباشد. تابع ضرر این کد مطابق با لینک یوتوبی است که بالاتر ارجاع داده شده اما معماری دو شبکه مطابق با دستور متلب است. به جز اینکه خروجی شبکه generator تصویری است که به جای بازه [-1, 1] مقادیر بازه [0, 1] را دارد. فایل gan_model.h5 فایل چک پوینت مدل FlowerGAN است. این فایل زمانی مفید است که ران تایم ما به هر دلیلی در کولب قطع شود. در این صورت میتوان وزن هایی داخل این فایل را لود کرد و ادامه آموزش دادن را از سر گرفت. البته در این اجرا ذخیره کردن وزن های generator برای اینکه اگر کسی خواست نتیجه عکس های تولید شده را مشاهده کند فراموش شد. فایل بعدی که ذخیره شده است فایل generated_img_490.png، تصویر تولید شده توسط مدل generator در ایپاک ۴۹۰ امی است. در کد هر ۱۰ ایپاک یک بار یک بردار که از توزیع نرمال نمونه برداری میشود به مدل generator داده و تصویر خروجی ذخیره میشود. آخرین فایل هم با پسوند v2. مربوط به تنسوربرد است. این فایل جهت اطمینان در صورتی که نتایج نمودار ها در تنسوربرد روی نوت بوک بالا نیامد باید استفاده شود.

از آنجایی که تصویر خروجی این کد مطلوب نبود در اینترنت سرچ کردم تا دلیل همگرا نشدن شبکه را متوجه بشوم که به این پست برخوردم:

https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/

در این پست یکی از دلایلی که اشاره کرده بود اعمال همزمان گرادیان شبکه discriminator برای تصاویر واقعی و فیک است. به عبارت دیگه ابتدا باید مقدار ضرر شبکه برای تصاویر واقعی محاسبه شود. سپس گرادیان آن گرفته

شود و این گرادیان در شبکه انتشار یابد. سپس بعد از آن مقدار ضرر شبکه برای تصاویر فیک محاسبه شود. سپس گرادیان آن گرفته شود و ... پس گرادیان ها جدا جدا اعمال میشوند.

من این روش را روی کد امتحان کردم اما به تنهایی خروجی مناسب را نگرفتم. نتایج آن در پوشه GAN_Flower_500_Epochs_Separated پوش شده است. لازم به ذکر است که ازین پوشه به بعد به جای وزن های کلی شبکه، وزن های شبکه generator را ذخیره کردم چون این وزن ها هستند که برای تولید تصویر گل نیاز هستند.

در ران بعدی کار دیگری که انجام دادم تغییر شبکه discriminator بود. در این شبکه از تجارب قبلی که داشتم و کد هایی که دیده بودم پارامتر stride و padding را حذف کردم و به جای آن لایه های max pooling اضافه کردم. همچنین لایه های batch normalization را حذف کردم و یک لایه dense هم جای لایه کانولوشنال آخر گذاشتم. تغییر دیگری که دادم لایه drop out که بعد از لایه ورودی بود را حذف کردم و قبل از لایه dense گذاشتم. با این تغییرات وقتی شبکه را ران کردم خروجی مطلوب در ایپاک ۲۶۰ ام و ۲۷۰ ظاهر شدند که به ترتیب به صورت زیر هستند:

تمامی تغییراتی که گفته شد در پوشه GAN_Flower_500_Epochs_Separated_Dis در گیت هاب پوش شدند.

از آنجایی که روی ایپاک ۲۷۰ام گل مناسبی مشاهده کردم تصمیم گرفتم با ۲۷۰ هم اجرا بگیرم تا اخرین وزن های generator روی این تعداد ایپاک سیو و ارزیابی شود. بنابراین یک پوشه هم به نام GAN_Flower_270_Epochs_Separated_Dis در گیت هاب وجود دارد.

اخرین کدی که امتحان کردم در پوشه GAN_Flower_500_Epochs_Matlab است. در این پوشه تمام دستور العملی که در متلب گفته شد با تنسورفلو پیاده سازی کرده ام. به جز اینکه دو تغییر ایجاد کردم. اولین تغییر مربوط به همان ساختار discriminator است که دقیقا همان تغییرات بالا را لحاظ کردم چون اینجا هم نتوانستم با استفاده از معماری گفته شده در متلب خروجی مطلوب بگیرم و دیگر اینکه learning rate های generator و discriminator را از 0.0002 به 0.0001 و 0.00001 تغییر دادم چون وقتی کد را اجرا کردن بعد از چند ایپاک مقادیر nan را در مقادیر ضرر دو شبکه مشاهده کردم پس نرخ یادگیری را کاهش دادم. متاسفانه با این تغییرات باز هم خروجی دلخواه را با تابع ضررهایی که متلب پیاده سازی کرده بود مشاهده نکردم.

در نهایت بهترین نتیجه از نظر اینجانب مربوط به پوشه GAN_Flower_500_Epochs_Separated_Dis است.