به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی پروژه سری اول

نام و نام خانوادگی	سینا شریفی	صدف صادقیان
شماره دانشجویی	۸۱۰۱۹۵۴۱۲	۸۱۰۱۹۵۴۱۹
تاریخ ارسال گزارش	۱۳۹۹/۵/۱۴	

فهرست گزارش سوالات

سوال ۱ – Variational Autoencoder

الف) تفاوت VAE و GAN را در تولید داده‌ها بیان کنید و مزایا و معایب آن‌ها را نسبت به هم بیان نمایید.

از نظر ساختاری، VAE به دو بخش encoder و decoder تقسیم میشود در حالیکه GAN از دو بخش discriminator و generator تشکیل شده است.
در VAE با داشتن یک loss function مشخص، هدف شبیه ساختن فضای ورودی و خروجی است.
و در GAN هدف بالا بردن کیفیت عکس‌های تولیدی با استفاده از minmax است بدین صورت که بخش discriminator تلاش میکند که به خوبی عکس واقعی از تولیدی رو تشخیص دهد در حالیکه generator سعی میکند با واقعی تر کردن عکس‌های خود discriminator را فریب دهد.

یکی از برتری‌های VAE نسبت به GAN این است که در زمان آموزش، یک cost function مشخص آموزش می‌بیند و در نتیجه می‌توان کیفیت عملکرد شبکه را توسط loss آنها سنجید ولی در مورد GANها این مورد به این مشخصی نیست. یعنی به این علت که در مراحل یادگیری GAN ها ما عملا مسئله‌ی unsupervised را به supervised تبدیل میکنیم، معیار loss به خوبی VAE loss تعریف نمی‌شود.

علاوه بر این یکی از مشکلات GAN عدم همگرایی به یک نقطه خاص و مشخص است چرا که با minimax ترین می‌شود اما VAE نقطه همگرایی دارد چرا که به شکل supervised و با یک loss function مشخص ترین می‌شود.
همچنین وجود یک latent space مشخص، باعث می‌شود که بتوانیم ویژگی ها را با هم ترکیب کنیم و این کار در GAN امکان پذیر نیست.

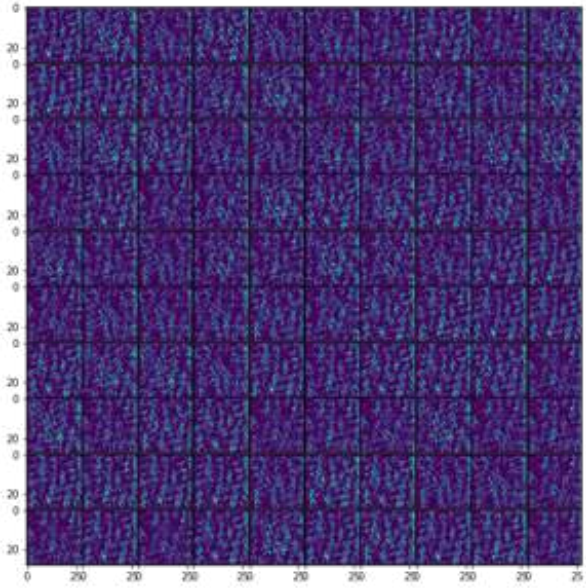
یکی دیگر از مشکلات GAN کند شدن فرآیند آموزش به علت کوچک شدن مقدار مشتقات در طول روند آموزش است.

یکی از برتری‌های GAN نسبت به VAE این است که در VAE ممکن است به علت وجود نویز در مرحله انتخاب دیتا برای دیکود، خروجی کمی مات به نظر آید و کیفیت عکس لزوما بالا نباشد ولی GAN در صورت یادگیری مناسب میتواند کیفیت بهتری تولید کند.

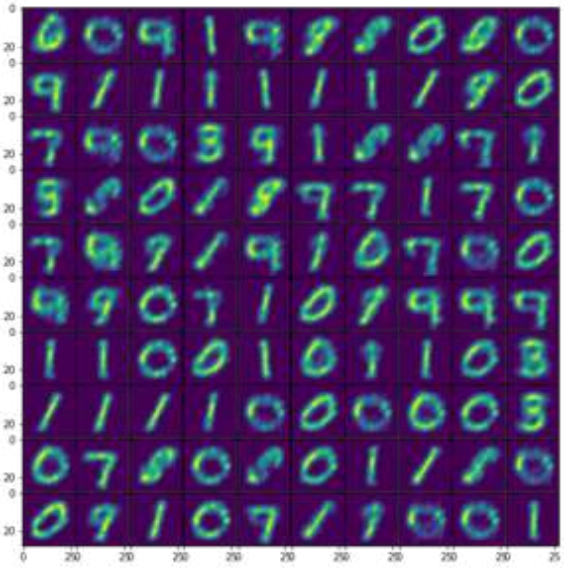
ب) توضیحات و نتایج پیاده‌سازی

در ادامه ابتدا تصاویر ۱۰ در ۱۰ خواسته شده صورت سوال به ازای اپیاک‌های صفر تا ۱۰۰ با گام‌های ۲۵ نمایش می‌دهیم.

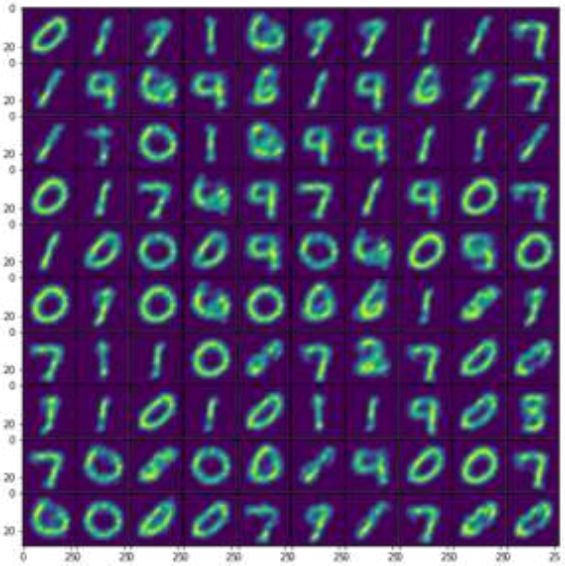
روند کلی یادگیری از روی این نتایج مشخص است.



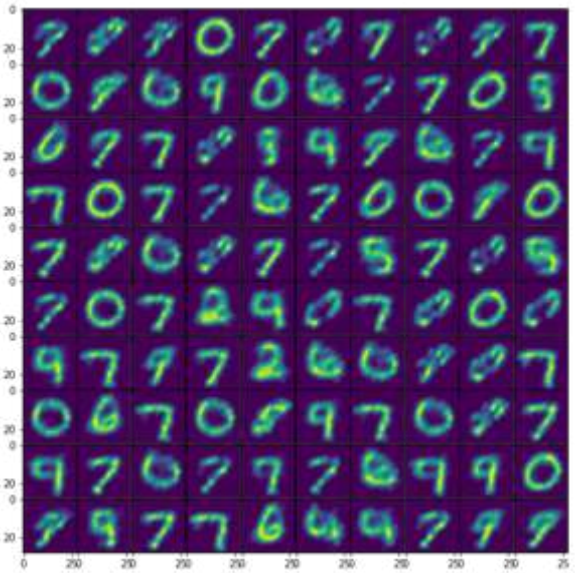
شکل ۱- نمونه خروجی در ایپاک صفر



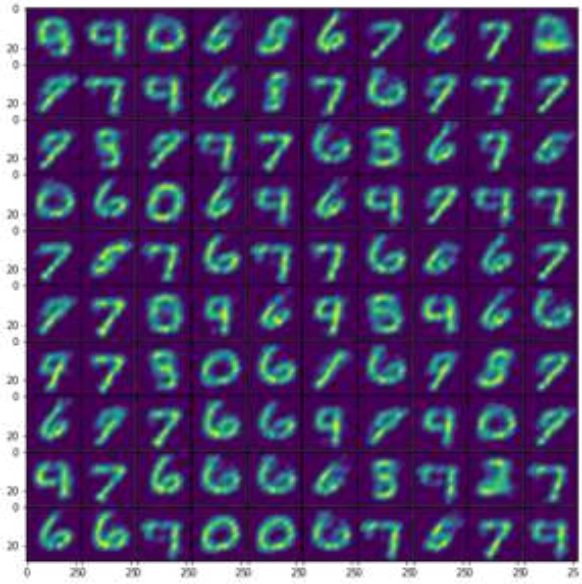
شکل ۲- نمونه خروجی در ایپاک ۲۵



شکل ۳- نمونه خروجی در ایپاک ۵۰



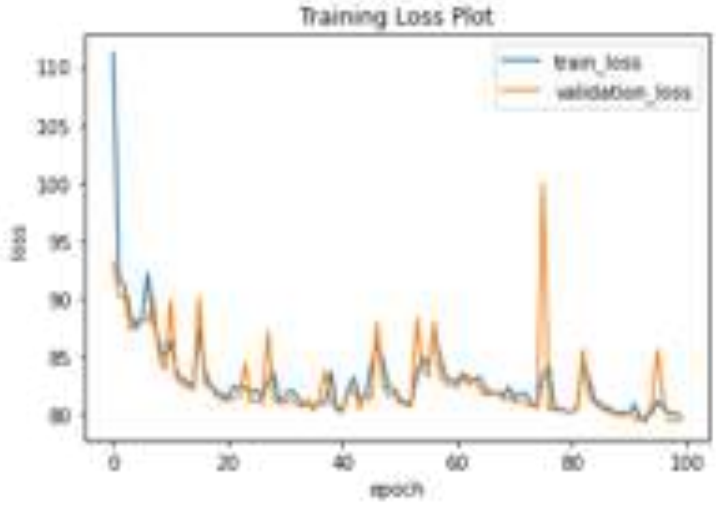
شکل ۴- نمونه خروجی در ایپاک ۷۵



شکل ۵- نمونه خروجی در ایپاک ۱۰۰

سپس نمودار loss وlatent space را نمایش میدهیم.

در این نمودار نیز روند کلی یادگیری کاملا مشخص است و احتمالا با افزایش تعداد ایپاک‌ها نیز نتیجه بهتر میشود.



شکل ۶- نمودار loss

ج) در قسمت ب از چه تابع هزینه‌ای استفاده کرده‌اید؟

همانطور که در کلاس و مقاله مربوط مطرح شد، از kl reconstruction loss استفاده شد.

این تابع هزینه از ترکیب وزن دار دو تابع هزینه Reconstruction و KL divergence استفاده شد که علت حضور هرکدام جداگانه توضیح داده خواهد شد.

علت وجود ترم reconstruction نزدیک کردن مقدار تولید شده به مقدار اولیه است، که این قسمت مشابه منطقی است که در اتوانکودر های عادی نیز داریم.

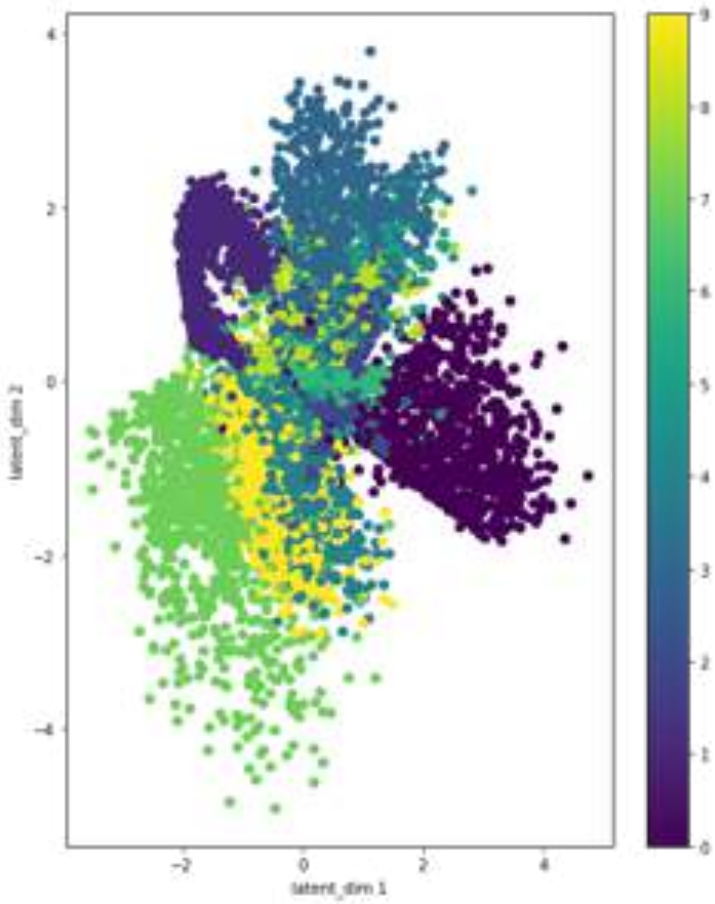
ترم Kullback-leibler هدفش این است که که توزیع داده‌ها در latent space به صورت دلخواه شود. این فرم دلخواه بدین صورت است که دسته‌ها از هم جدا شوند ولی به هم نزدیک باشند زیرا در صورتی که کاملا از هم فاصله بگیرند، روش نویزی کردن ورودی دیکودر توسط تابع رندوم، باعث ساخت تصاویر مناسبی نمیشود.

در انتها یک جمع وزن دار از این دو تابع هزینه مورد استفاده کلی برای شبکه قرار میگیرد.

د) ورودی را به فضای ۲ بعدی برده و نمایش دهید.

از آنجا که ما داده‌های validation را از train جدا نکرده بودیم، این بخش روی داده های تست انجام شد.

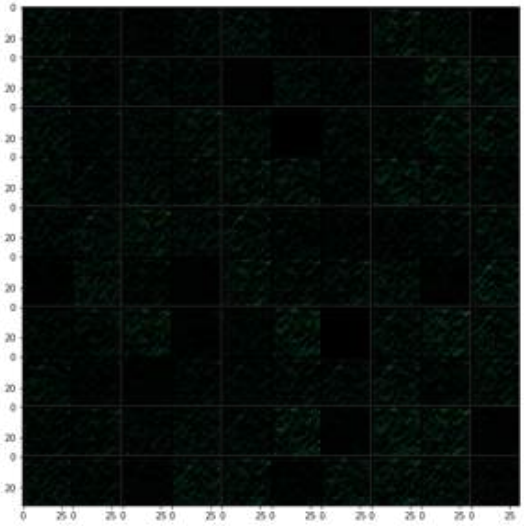
در تصویر زیر همانطور که مشخص است، بعضی از اعداد مثل ۷ و ۰ به خوبی دسته‌بندی شده اند و بعضی دیگر که شبیه هم هستند مثل ۶ و ۸ در دسته‌هایی نزدیک هم قرار گرفته اند.



شکل ۷- داده‌ها در latent space

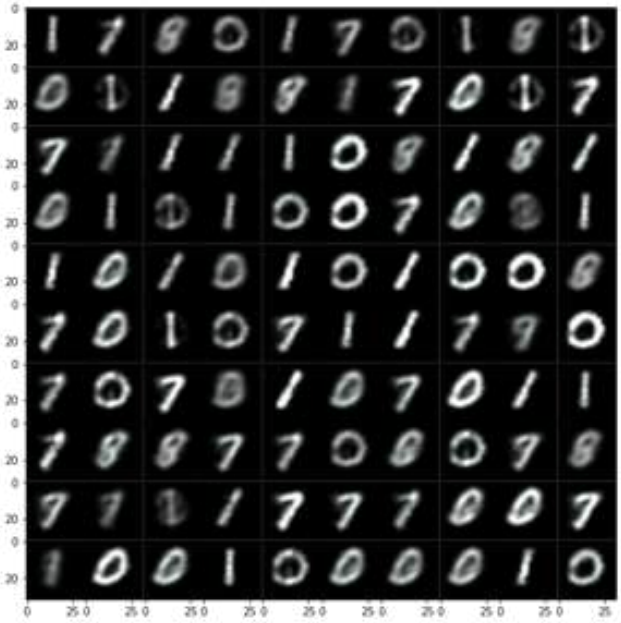
ه) پیاده‌سازی VAE با استفاده از EfficientNet

در این قسمت از سوال، هدف ما استفاده از efficient net برای انجام عمل feature extraction است.
برای اینکار از transfer learning استفاده کرده و وزن‌های شبکه‌ای که روی دیتاست imagenet بدست آمده را لود میکنیم و این شبکه‌ی آماده را (با freeze کردن وزن‌های آن) در قسمت اول encoder قرار می‌دهیم .
در ادامه مثل بخش‌های قبلی عمل میکنیم و از روی فیچرهای جدا شده، میانگین و واریانس را محاسبه میکنیم.
بخش decoder تغییر خاصی نمی‌کند و مثل بخش‌های قبل آموزش می‌بینند.
در انتها کل شبکه را آموزش می‌دهیم و نمودارها و سایر موارد خواسته شده را نشان می‌دهیم.



شکل ۸- نمونه تصویر خروجی در ایپاک صفر

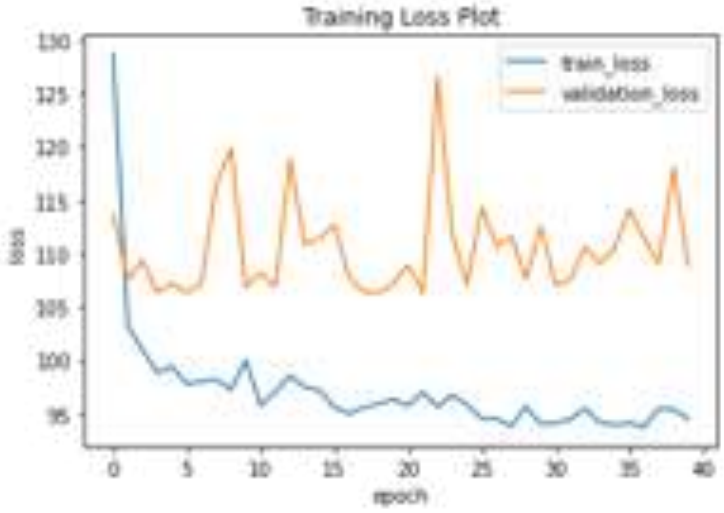
شکل ۸- نمونه تصویر خروجی در ایپاک ۱۰



شکل ۹- نمونه تصویر خروجی در ایپاک ۲۰

شکل ۱۰- نمونه تصویر خروجی در ایپاک ۳۰

شکل ۱۱- نمونه تصویر خروجی در ایپاک ۴۰



شکل ۱۲- نمودار loss

سوال ۲ – DCGAN

الف)سازوکار طراحی معماری DCGAN را توضیح دهید.

طراحی dcgan مشابه بقیه gan ها از یک generator و یک discriminator تشکیل میشود. با این تفاوت که در طراحی discriminator و generator به صورت کامل از لایه‌های کانولوشنال استفاده میشود. البته طبق معمول cnn ها، از لایه‌های dropout و batch normalization نیز استفاده میکنیم.

در ادامه، با همان منطق minmax همیشگی، یادگیری را انجام میدهیم به این صورت که در دو مرحله، یکبار discriminator را با عکسهای واقعی و تولیدی، trainمیکنیم و یکبار هم کل شبکه را با ثابت نگه داشتن وزن‌های discriminator آموزش میدهیم.

ب)نحوه‌ی ایجاد نویز را توضیح دهید.

برای ایجاد نویز، با استفاده از یک تابع نرمال با میانگین صفر و واریانس یک، با توجه سایز batch و اندازه ورودی جنراتور، یک ماتریس تشکیل داده و به شبکه میدهیم. این کار توسط تابع gennoise انجام میشود.

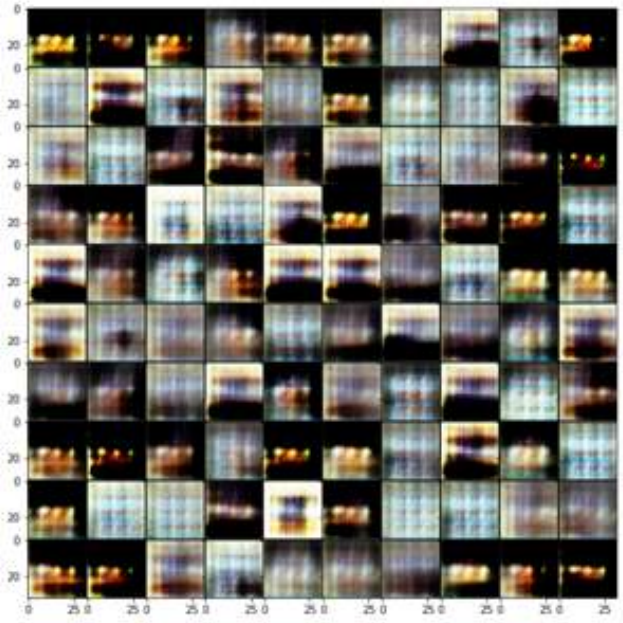
ج) دلیل استفاده از Strided Convolution و Fractional Strides Convolution را توضیح دهید .

علت جایگزینی توابع pooling خاص (که به یک شکل خاص روی فضا عمل pooling را انجام می‌دهند) مثل max pooling با strided convolution ها این است که از یک روش یکتا و مشخص برای down sampling استفاده نشود بلکه لایه‌های شبکه‌ی discriminator خودشان روش down sampling مختص و مناسب برای فضای مسئله را بیابند. همچنین با استفاده از fractional strided convolution شبکه‌ی generator نحوه‌ی up sampling مناسب و مخصوص فضا را خودش یاد می‌گیرد.

د) توضیحات و نتایج پیاده‌سازی

ابتدا تصاویر ۱۰ در ۱۰ خواسته شده صورت سوال به ازای ایپاک‌های صفر تا ۱۰۰ را نمایش می‌دهیم.

روند کلی یادگیری از روی این نتایج مشخص است.



شکل ۱۳- نمونه خروجی در ایپاک صفر

شکل ۱۴- نمونه خروجی در ایپاک ۲۰

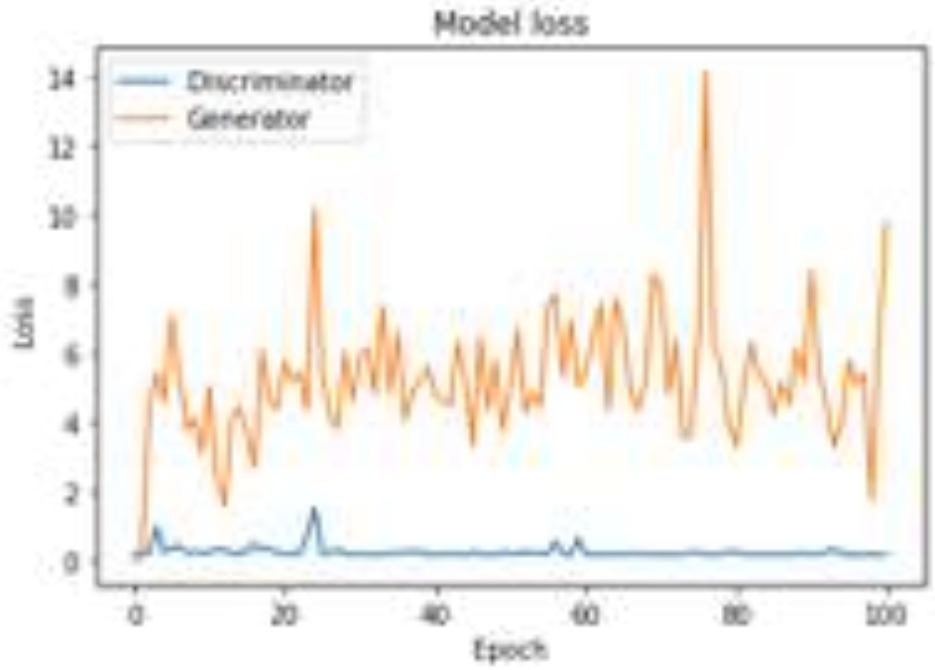
شکل ۱۵- نمونه خروجی در ایپاک ۴۰

شکل ۱۶- نمونه خروجی در ایپاک ۶۰

شکل ۱۷- نمونه خروجی در ایپاک ۸۰

شکل ۱۸- نمونه خروجی در ایپاک ۱۰۰

سپس نمودار loss را نمایش میدهیم. نمودار loss نزولی اکید نشد و علت آن بحث ایجاد رقابت بین دو بخش شبکه است.



شکل ۱۹- نمودار loss

ه) یکی از چالش‌ها در شبکه های GAN، چالش انتخاب تابع Loss مناسب است ، در مورد این مسئله تحقیق کنید و ۴ تابع هزینه معروف برای این کار را نام ببرید و مختصراً توضیح دهید .

Minimax Loss-۱

در مقاله ای که GAN ها را معرفی کرده است ، generator سعی می کند تابع زیر را به حداقل برساند در حالی که discriminator سعی در به حداکثر رساندن آن دارد:

$$E_x\left[\log\left(D(x)\right)\right] + E_x\left[\log\left(1 - D(G(z)\right)\right]$$

که در آن:

D(x) تخمین discriminator از احتمال واقعی بودن نمونه داده واقعی x است.

G(z) خروجی generator زمانی که نویز z به آن داده می‌شود.

D(G(z)) تخمین discriminator از احتمال واقعی بودن یک نمونه داده ساختگی است.

E_x مقدار expected value روی همه داده‌های واقعی است.

E_z مقدار expected value روی همه داده‌های ساختگی است.

generator نمی تواند بطور مستقیم $\log(D(z))$ را تحت تأثیر قرار دهد ، بنابراین ، آن به حداقل رساندن loss معادل با حداقل رساندن $\log(1 - G(D(z)))$ است.

Modified Minimax Loss-۱'

مقاله بیان می‌کند minimax loss function بالا می‌تواند باعث شود آموزش GAN در مراحل ابتدایی گیر کند. به همین دلیل پیشنهاد می‌دهد برای generator از تابع loss را طوری تغییر دهیم که generator مقدار $\log(D(z))$ را به حداکثر برساند.

Wasserstein Loss-۲

از این loss برای Wasserstein GAN یا WGAN استفاده می‌شود. که در آن discriminator در واقع نمونه‌های داده را classify نمی‌کند بلکه برای هر نمونه داده یک عدد که باید بین صفر تا یک باشد خروجی می‌دهد. پس ما نمی‌توانیم از threshold برابر با ۰/۵ برای تشخیص واقعی یا ساختگی بودن استفاده کنیم بلکه discriminator تنها تلاش می‌کند برای نمونه‌های واقعی مقادیر بزرگتری از مقادیر نمونه‌های ساختگی تولید کند و به همین دلیل در این شبکه به جای discriminator به آن critic گفته می‌شود.

- تابع لاس critic (یا discriminator) : $E_x[D(x)] - E_z[D(G(z))]$
Discriminator سعی در به حداکثر رساندن این تابع دارد. به عبارتی تلاش می‌کند تفاوت خروجی‌اش برای نمونه‌های واقعی و خروجی‌اش برای نمونه‌های ساختگی را به حداکثر برساند.
- تابع لاس generator : $E_z[D(G(z))]$
generator سعی در به حداکثر رساندن این تابع دارد. به عبارت دیگر، تلاش می‌کند خروجی discriminator را برای نمونه های جعلی به حداکثر برساند.

که در آن:

$D(x)$ خروجی critic برای نمونه داده واقعی x است.

$G(z)$ خروجی generator زمانی که نویز z به آن داده می‌شود.

$D(G(z))$ خروجی critic برای یک نمونه داده ساختگی است.

E_x مقدار expected value روی همه داده‌های واقعی است.

E_z مقدار expected value روی همه داده‌های ساختگی است.

Least Squares Loss-۳

این نوع GAN شبیه به DCGAN است اما از loss functionهای متفاوتی برای Discriminator و Generator به نام least squares loss در آن استفاده می‌شود که این تغییر باعث می‌شود مشکلات vanishing gradient و loss saturation شبکه‌های DCGAN حل شود و یادگیری stable تر شود.

- تابع لاس discriminator : $E[(D(x)-1)^2] - E\left[D\left(G(z)\right)^2\right]$
- تابع لاس generator : $E\left[\left(D\left(G(z)\right)-1\right)^2\right]$

Info GAN loss function-۴

در این نوع GAN ، generator علاوه بر نویز یک شرط یا اطلاعات اضافه مانند لیبل کلاس را نیز به عنوان ورودی می‌گیرد (در واقع یک نوع conditional GAN است) و discriminator علاوه بر اینکه به عنوان خروجی احتمال واقعی یا ساختگی بودن تصویر را می‌دهد، توزیع احتمال Q(c|x) را نیز به عنوان خروجی اضافه می‌دهد(که نشان دهنده این است که تصویر با چه احتمالی هر کدام از کلاس‌هاست).

برای تابع لاس این شبکه از تابع لاس GAN معمولی جمله I(c;x) را کم می‌کنیم. I(c;x) اطلاعات متقابل نام دارد و نشان می‌دهد اگر x را داشته باشیم و بدانیم، با چه احتمالی c را می‌دانیم و خواهیم داشت. برای مثال I(c;x) برابر با صفر است اگر x و c کاملا نامرتبط باشند اما اگر discriminator بتواند c را به درستی پیش بینی کند، مقدار I(c;x) بزرگ خواهد بود و مقدار loss شبکه را کاهش می‌دهد.

- تابع لاس discriminator : $Loss_{disc}^{GAN} - \lambda I(c; G(z,c))$
- تابع لاس generator : $Loss_{gen}^{GAN} - \lambda I(c; G(z,c))$

سوال ۳ – Conditional GANs

الف) سازوکار طراحی معماری CGAN را شرح دهید.

به عنوان یک تعمیم از GAN ها می‌توان به هر دو generator و discriminator یک شرط بر روی اطلاعات اضافه (y) گذاشت. y می تواند هر نوع اطلاعات کمکی مانند برچسب کلاس یا داده های سایر اطلاعات باشد.

با استفاده از این اطلاعات اضافی می‌توان GAN را بهبود داد، این بهبود ممکن است به شکل آموزش پایدار ، آموزش سریعتر و یا تصاویر ایجاد شده با کیفیت بهتر باشد. همچنین برعکس GAN که کنترلی روی تصویر تولید شده نداریم، در CGAN می‌توان تصاویر را برای یک کلاس دلخواه تولید کرد.

معماری CGAN به این شکل است که از یک generator و یک discriminator تشکیل می‌شود که generator علاوه بر noise، یک شرط مثل برچسب کلاس را می‌گیرد و یک تصویر از آن کلاس را تولید می‌کند. همچنین discriminator هم علاوه بر تصویر، شرط اضافه، مثلا همان برچسب کلاس تصویر ورودی را می‌گیرد و تشخیص می‌دهد آیا تصویر داده شده از آن کلاس خاص واقعی و یا ساختگی است.

در طراحی discriminator و generator به صورت کامل از لایه‌های کانولوشنال استفاده میشود. البته طبق معمول cnn ها، از لایه‌های dropout و batch normalization نیز استفاده میکنیم. در ادامه، با همان منطق minmax همیشگی، یادگیری را انجام میدهیم به این صورت که در دو مرحله، یکبار discriminator را با عکسهای واقعی و تولیدی، train می‌کنیم و یکبار هم کل شبکه را با ثابت نگه داشتن وزن‌های discriminator آموزش می‌دهیم.

ب) توضیحات و نتایج پیاده‌سازی

در ادامه ابتدا تصاویر ۱۰ در ۱۰ خواسته شده صورت سوال به ازای ایپاک‌های صفر تا ۷۰ را نمایش می‌دهیم. در تصاویر هر سطر متناظر با یک کلاس است که به ترتیب از سطر اول تصاویر هواپیما، ماشین، پرنده، گربه، گوزن، سگ، قورباغه، اسب، کشتی و کامیون در هر سطر وجود دارند.

شکل ۲۰ - نمونه خروجی در ایپاک صفر

شکل ۲۱- نمونه خروجی در ایپاک ۱۰

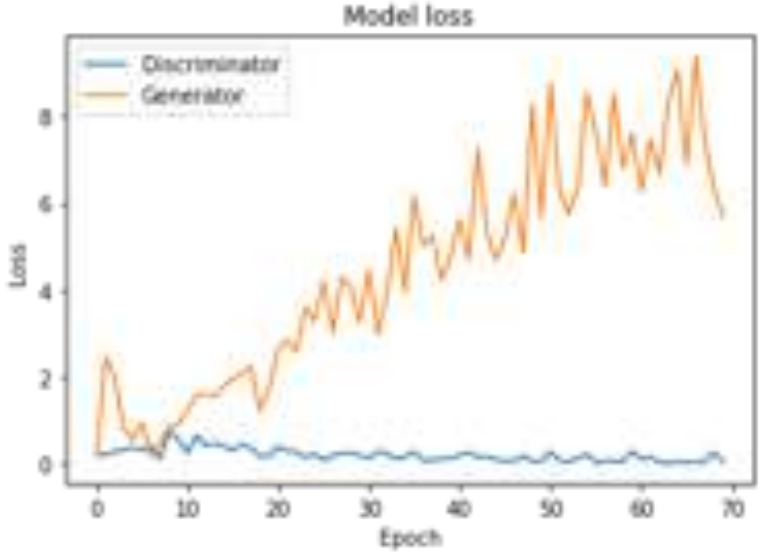
شکل ۲۲- نمونه خروجی در ایپاک ۳۰

شکل ۲۳- نمونه خروجی در ایپاک ۵۰

شکل ۲۴- نمونه خروجی در ایپاک ۷۰

سپس نمودار loss را نمایش میدهیم.

همان طور که قابل مشاهده است نمودار بصورت پیوسته برای هر دو کاهش نداشته است چرا که generator و discriminator در واقع به شکل agent های minimax ترین می‌شوند.



شکل ۲۵- نمودار loss

ج) سازوکار معماری SGAN را توضیح دهید .

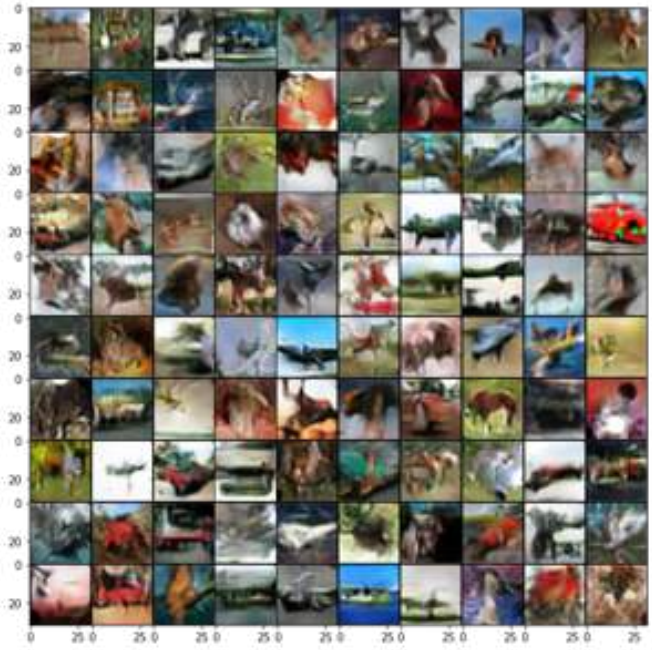
طراحی sgan مشابه بقیه gan ها از یک generator و یک discriminator تشکیل میشود. با این تفاوت که discriminator علاوه بر بیتvalidity ، به صورت one hot لیبل ورودی خود را در صورت واقعی بودن عکس برمیگرداند در طراحی discriminator و generator به صورت کامل از لایه‌های کانولوشنال استفاده میشود. البته طبق معمول cnn ها، از لایه‌های dropout و batch normalizationنیز استفاده میکنیم.

در ادامه، با همان منطق minmax همیشگی، یادگیری را انجام میدهیم با این تفاوت که در هنگام آموزش discriminator با عکس‌های واقعی، لیبل کلاس را نیز در یادگیری دخیل میکنیم.

پایه‌ی معماری دو بخش این gan مشابه بقیه‌ی gan هاست ولی از لایه‌های convoloutional و batch normalization و ... در آنها استفاده شده است.

در ادامه نتایج را مشاهده می‌کنید.

شکل ۲۶- نمونه خروجی در ایپاک یک



شکل ۲۷- نمونه خروجی در ایپاک ۲۰

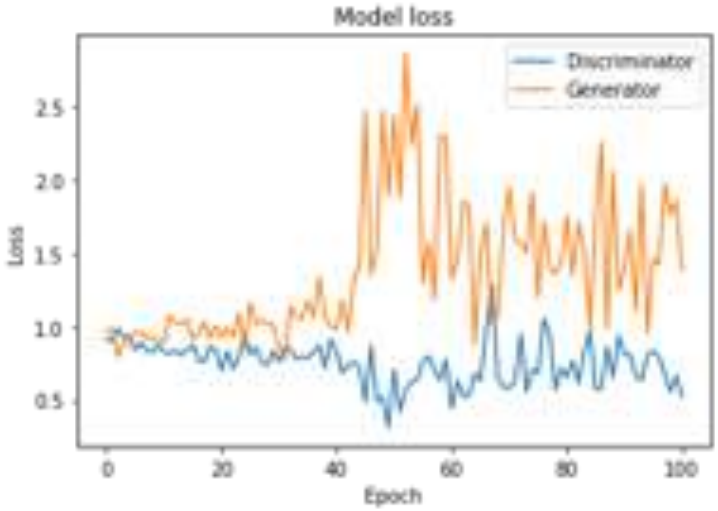
شکل ۲۸- نمونه خروجی در ایپاک ۴۰

شکل ۲۹- نمونه خروجی در ایپاک ۶۰

شکل ۳۰- نمونه خروجی در ایپاک ۸۰

شکل ۳۱- نمونه خروجی در ایپاک ۱۰۰

همان‌طور که انتظار داشتیم، نمودار loss نزولی اکید نشد و علت آن بحث ایجاد رقابت بین دو بخش شبکه است.

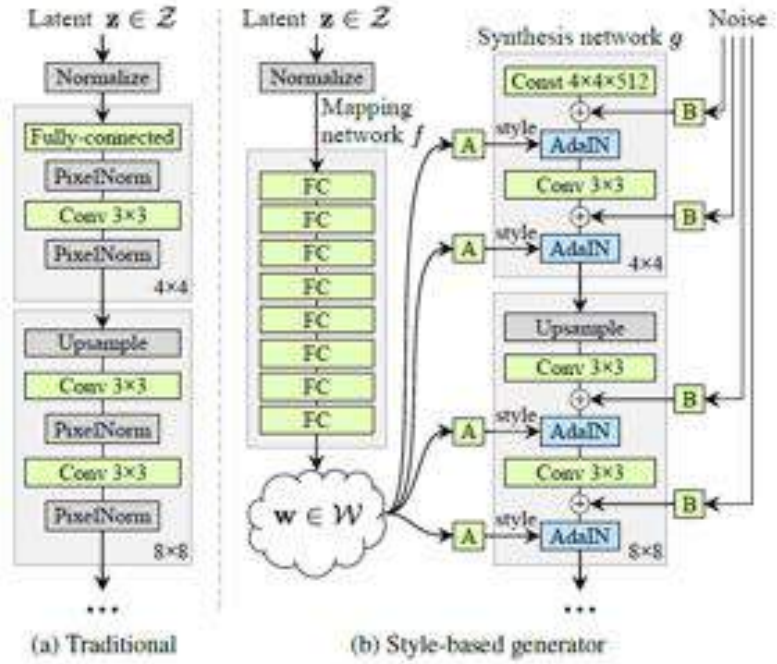


شکل ۳۲- نمودار loss

سوال ۴ – کاربردهای GAN (امتیازی)

الف) فقط یکی از ۶ کاربرد معرفی شده را انتخاب نمایید و یکی از مقالات معرفی شده مربوط به آن کاربرد را مطالعه نمایید و سازوکار آن شبکه و مدل را توضیح دهید .

یکی از مشکلاتی که در gan های عادی به وجود می‌آید این است که مشخص نیست که کدام ویژگی ها در کدام قسمت latent space انکود میشوند.
میتوان این ویژگی ها را به دو دسته‌ی ویژگی کلی (در مورد صورت اشخاص) مثل فرم صورت و ویژگی جزئی مثل چین و چروک و ... تقسیم کرد.
تفاوت اصلی این gan با gan های عادی این است که در بخش دادن نویز به generator کمی تغییر ایجاد میکنیم، و این کار باعث میشود کنترل بیشتری روی محل هر ویژگی داخل latent space داشته باشیم.
این تفاوت به این صورت است که بر خلاف gan های عادی بجای اینکه نویز مستقیم وارد generatorشود، مانند شکل زیر در مراحل مختلف به آن وارد میشود و علاوه بر این، وزن‌های این شبکه یادگیری میشوند.



شکل ۳۳- ساختار Style GAN

ب) توضیحات و نتایج پیاده‌سازی

شکل ۳۴- عکس‌های تولید شده توسط Style GAN