به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین ششم

امیرعباس رضا سلطانی	610399205
نیما نیرومند	610399199

قوانین

قبل از پاسخ دادن به پرسش‌ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ‌های خود یک گزارش در قالبی که در صفحه‌ی درس در سامانه‌ی Elearn با نام ***REPORTS_TEMPLATE.docx*** قرار داده شده تهیه نمایید.
- پیشنهاد می‌شود تمرین‌ها را در قالب گروه‌های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره‌ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژه‌ای برخوردار است**؛ بنابراین، لطفا تمامی نکات و فرض‌هایی را که در پیاده‌سازی‌ها و محاسبات خود در نظر می‌گیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل‌ها زیرنویس و برای جدول‌ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- **تحلیل نتایج الزامی می‌باشد، حتی اگر در صورت پرسش اشاره‌ای به آن نشده باشد.**
- **دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند**؛ بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر می‌شود.
- **کدها حتما باید در قالب نوت‌بوک با پسوند .ipynb تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتما در این فایل ارسالی شما ذخیره شده باشد.** بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده‌اید، این نمودار باید هم در گزارش هم در نوت‌بوک کدها وجود داشته باشد.
- **در صورت مشاهده‌ی تقلب امتیاز تمامی افراد شرکت‌کننده در آن، 100- لحاظ می‌شود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- **استفاده از کدهای آماده برای تمرین‌ها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می‌شود.**
- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.

- سه روز اول: بدون جریمه
- روز چهارم: ۵ درصد
- روز پنجم: ۱۰ درصد
- روز ششم: ۱۵ درصد
- روز هفتم: ۲۰ درصد

- حداکثر نمره‌ای که برای هر سوال می‌توان اخذ کرد ۱۰۰ بوده و اگر مجموع بارم یک سوال بیشتر از ۱۰۰ باشد، در صورت اخذ نمره بیشتر از ۱۰۰، اعمال نخواهد شد.
  - برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.
- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه‌ی Elearn بارگذاری نمایید:

HW[Number]
[Lastname][StudentNumber]_[Lastname]_[StudentNumber].zip

(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

- برای گروه‌های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می‌شود هر دو نفر بارگذاری نمایند

پرسش ۱ – Variational Auto-Encoder

۱-۲. ساخت VAE روی دیتاست‌ها

در ابتدا در مورد مدل‌های VAE و نحوه عملکرد آن‌ها توضیح می‌دهیم.

می‌دانیم که منیمم کردن فاصله KL معادل است با maximum likelihood است حال به دنبال $Q(z|X)$ هستیم که تا حد ممکن به $P(z|X)$ نزدیک باشد.

پس می‌خواهیم

$$D_{KL}(Q(z|X)\,||P(z|X)) = \sum_z Q(z|X)\log\frac{Q(z|X)}{P(z|X)} = E[log\frac{Q(z|X)}{P(z|X)}]$$

را مینیمم کنیم حال با باز کردن لگاریتم و قانون بیز داریم:

$$D_{KL}(Q(z|X)\,||P(z|X)) = E[logQ(z|X) - \log P(X|z) - \log P(z) + \log P(x)\,]$$

حال از اونجایی که امید ریاضی ما روی متغییر z بود پس $P(X)$ را می‌توانیم بیرون بیاریم پس داریم:

$$\begin{aligned}&\log P(X) - D_{KL}(Q(z|X)\,||P(z|X))\\ &= E[\log P(X|z)] - E[\,logQ(z|X) - \log P(z)]\\ &= E[logP(X|z)] - D_{KL}(Q(z|X)||P(z))\end{aligned}$$

حال قسمت چپ عبارت بالا در واقع نشان دهنده همان انکودر مدل VAE می‌باشد که از متغییر پنهان به دیتا می‌رسیم و قسمت راست انکودر است که می‌خواهیم فاصله تولیدی شبکه در انکودر را به توزیع متغییرهای پنهان نزدیک کنیم پس اگر عبارت بالا را ماکزیمم کنیم به ماکسیمم لایکی‌هود نزدیک می‌شویم می‌شویم که البته یه اختلاف دیکودینگم داریم

حال گفتیم که می‌خواهیم توزیع متغییرهای تولیدی انکودر را به توزیه متغییر پنهان نزدیک کنیم برای راحتی محسابات فرض می‌کنیم توزیع متغییر تصادفی پنهان ما گوسی با میانگین ۰ و واریناس که ۱ باشد همچنین توزیع $Q(z|X)$ از گوسی $N\sim(\mu(X),\Sigma(X))$ باشد و باز برای راحتی فرض می‌کنیم که $\Sigma(X)$ یک ماتریس قطری باشد با توجه به فرضایی که کردیم $D_{KL}(Q(z|X)||P(z))$ برابر با $\frac{1}{2}\sum_k(\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X))$ می‌شود و همچنین برای $E[logP(X|z)]$

برای این مسئله خاص که تصویر است و چون تصاویر ما نرمال و بین ۰ تا ۱ هست هر پیکسل پس بنظرم باینری کراس انتروپی رو هر پیکسل برای تصویر تولیدی با اصلی مناسب بود.

و هچنین وقتی که توزیع $\mu(X), \Sigma(X)$ را میبایبم برای اینکه z را به دست بیاوریم از این قضیه در فرآیندای تصادفی استفاده می‌کنیم که اگر اپسیلون یک متغییر تصادفی گوسی با توزیع میانیگن ۰ و واریانس ۱ باشد آنگاه z با اون توزیع‌هایی که گفتیم به شکل زیر به دست می‌اید (توجه شود از آنجایی که سیگما را ماتریس قطری گرفتیم در اینجا نیز محاسبات ما ساده میشود)

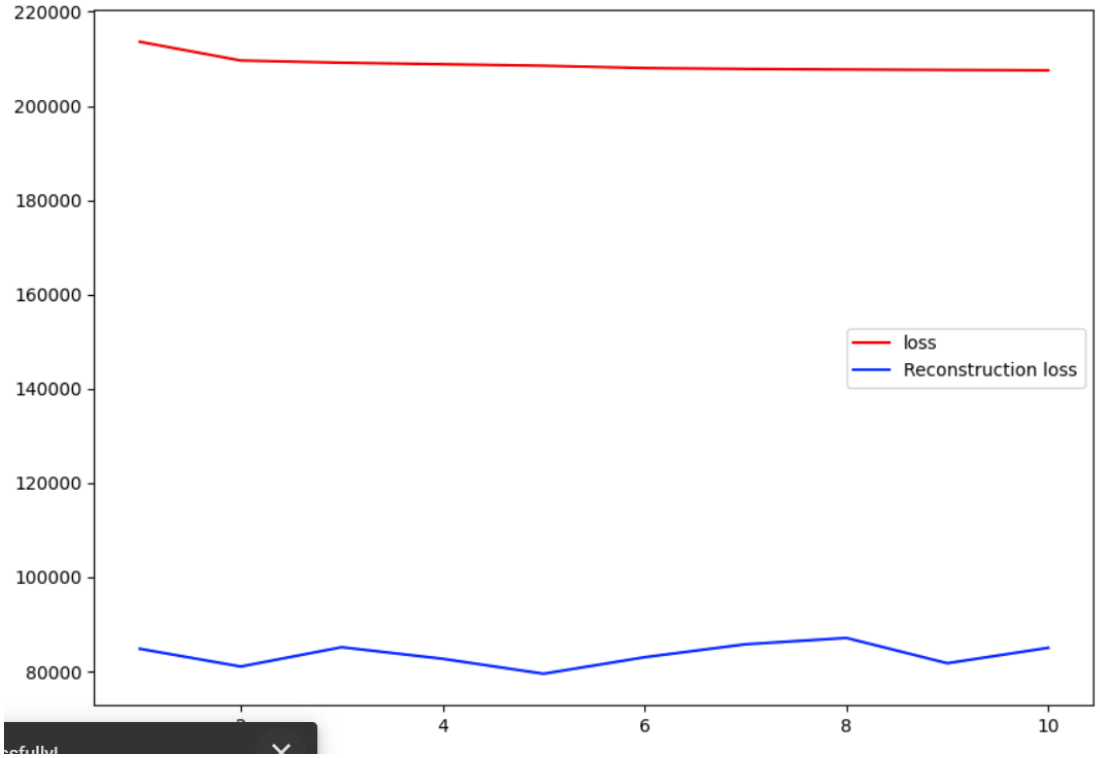
$$Z = \mu(X) + \Sigma^{0.5}(X)epsilon$$

خب پس مدل را به همین شکل که گفتیم می‌سازیم حال لاس ما مشخص است و فقط اینکه برای قسمت انکودر از کانولوشون استفاده می‌کنیم تا استخراج ویژگی داشتیم باشیم از سه لایه کانولوشون که به ترتیب ۱۶ ، ۳۲ ، ۶۴ تا فیلتر داشتن و فعال ساز رلو استفاده کردیم و در میان آن‌ها از مکس پولینگ و batchnormalization نیز استفاده کردیم و سپس خروجی را فلت می‌کنیم برای به دست آوردن میانگین و انحراف معیار به اندازه بعدی که برای متغییر پنهان گرفتیم از خروجی کانولوشون فولی کانکتت می‌زنیم و سپس همانطور که گفتیم از روی میانگین و واریانس متغییر پنهان را به دست می‌اریم و به دیکودر می‌دهیم که برعکس انکودر از از کانولوشون تنرسپوز استفاده می‌کنیم که به سایز عکس برگردیم و در نهایت برای خروجی چوم میخواهیم عددی بین ۰ تا ۱ باشد هر پیکسلمون پس از تابع فعال ساز سیگموید استفاده می‌کنیم.

حال ۵ تا از تصاویر انیمه‌ای را نشان می‌دهیم

شکل۱. ۵ تا نمونه از دیتاست چهره‌های انیمه‌ای

حال مدل را برای ۱۰ ایپاک آموزش می‌دهیم



شکل۲. نمودار لاس و recon_loss

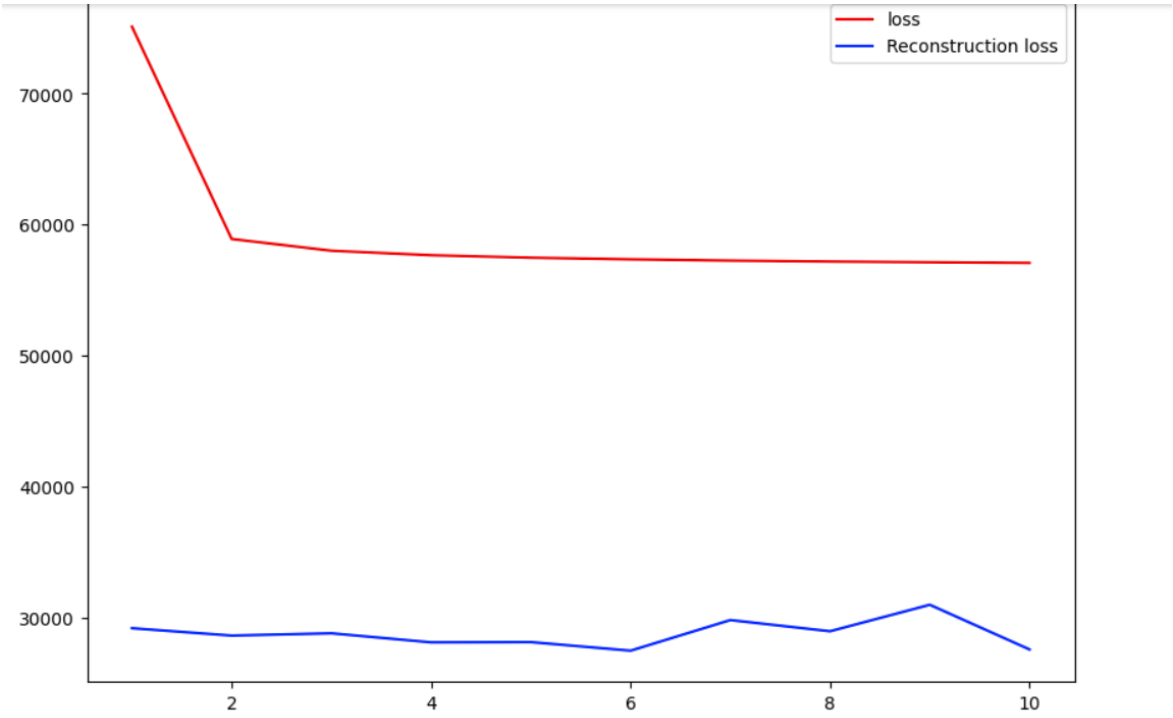
که لاس رو به کاهش بوده و بنظر می‌رسد مدل به خوبی آموزش دیده باشد

حال ۸ تا تصویر جدید را تولید می‌کنیم

شکل۳. چهره انیمه‌ای تولید شده توسطVAE

حال برای چهره‌های کارتونی نیز این کارها را انجام می‌دهیم

شکل۴. ۵ تا نمونه از دیتاست چهره‌های کارتونی



شکل۵. نمودار لاس و recon_loss آموزش داده‌های کارتونی

شکل۶. چهره کارتونی تولید شده توسطVAE

۱-۳. استفاده از یک مدل برای دوتا دیتاست

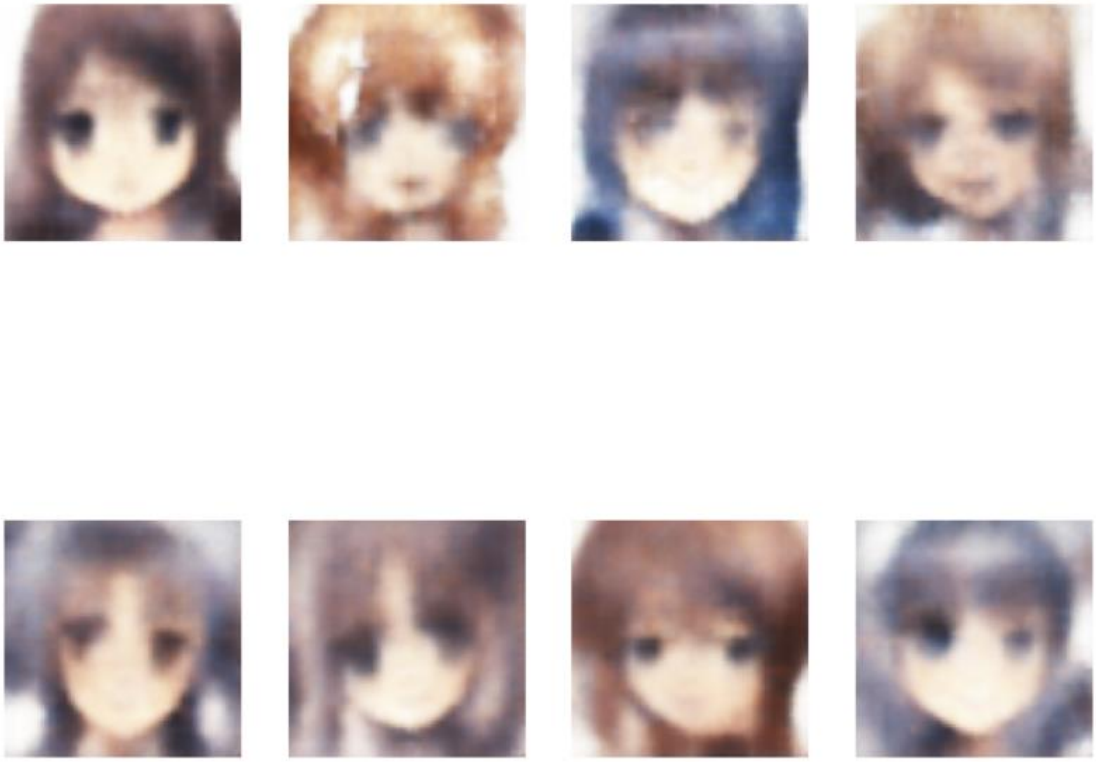
همانطور که در بخش قبل گفتیم درVAE کنترلی برروی تولیدی‌ها نداشتیم و نمیدونستیم اگر یک نمونه تصادفی بدیم کجای فضای نشون میدهد برای کنترل فرآیند تولد داده مفهوم CVAE به وجود آمد پس انکودر و دیکودر کاندیشانال باید آموزش ببینند

$$\log P(X|c) - D_{KL}(Q(z|X,c) \,||\, P(z|X,c)) =$$
$$= E[logP(X|z,c)] - D_{KL}(Q(z|X,c)||P(z|c))$$

پس می‌تونیم مثلا c یک توزیع کتگوریکال در نظر بگیریم پس وقتی تصویری از یک کلاس به مدل می‌دهیم یک کدنیگ از کلاس که ما در اینجا وان هات گرفتیم می‌گیریم و با دیتا کانکت می‌کنیم و که برای ورودی و همچنین برای z را با کدنیگ کلاس کانکت می کنیم و به شبکه می‌دهیم و در باقی قسمتا نسبت به بخش قبل تفاوتی نخواهیم داشت.

حال دیتاستی که تشکیل می‌دهیم نصف چهره‌های انیمه‌ای و نصف چهره‌های کارتونی و برای انیمه لیبل صفر و برای کارتون لیبل ۱(البته همانطور که گفتیم در مدل وان هاتش می‌کنیم.)

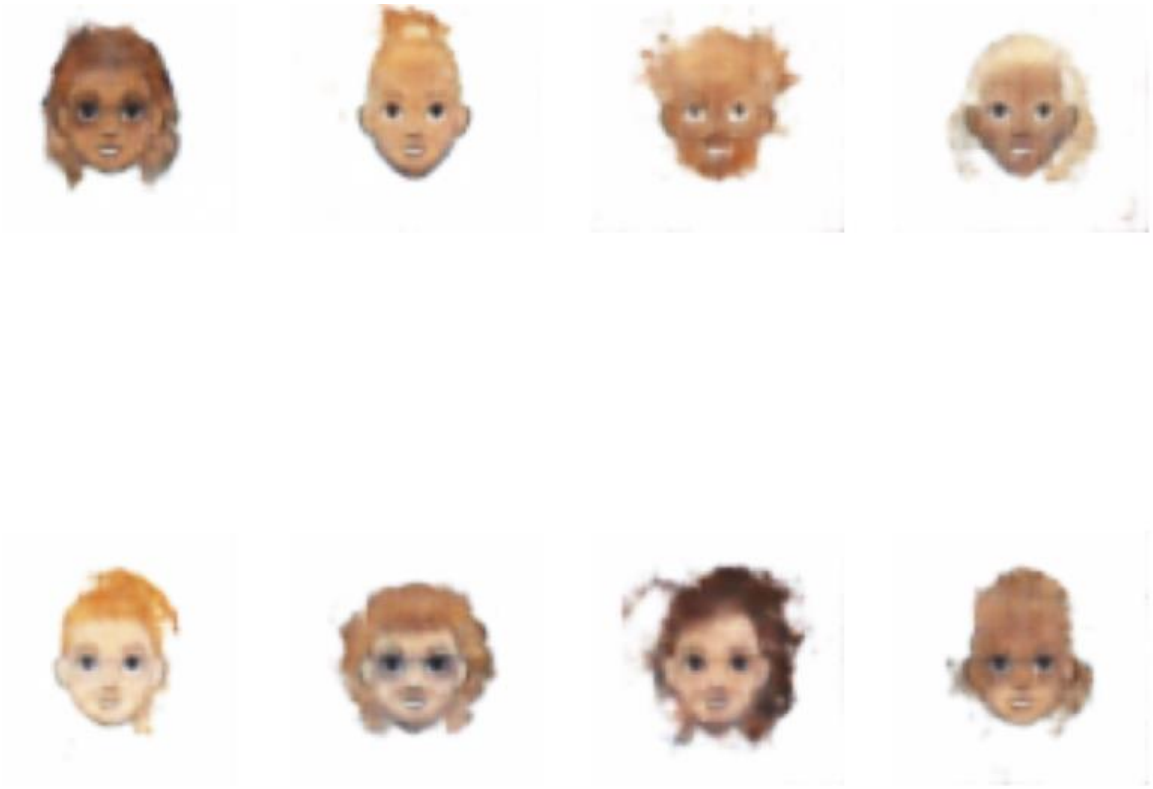
حال مدل را برای ۵ ایپاک آموزش می‌دهیم حال به مانند قسمت قبل متغیر تصادفی z را می‌سازیم و برای تولید یک لیبل را هم ورودی می‌دهیم حال ابتدا کلاس صفر را ورودی می‌دهیم که کلاس انیمه‌ها هست و حال خروجی را بررسی می‌کنیم.



تصویر چهره انیمه‌ای تولید شده توسط CVAE

همانطور که مشاهده می‌شود به خوبی توانسته بین دو تا کلاس تمیز قائل شه و چهره‌های انیمه‌ای خروجی دهد پس کنترل مدل با خودمان است.

حال برای کارتونی هم خروجی می‌دهیم که اونم به خوبی تمیز قائل شده مدل



تصویر چهره کارتونی تولید شده توسط CVAE

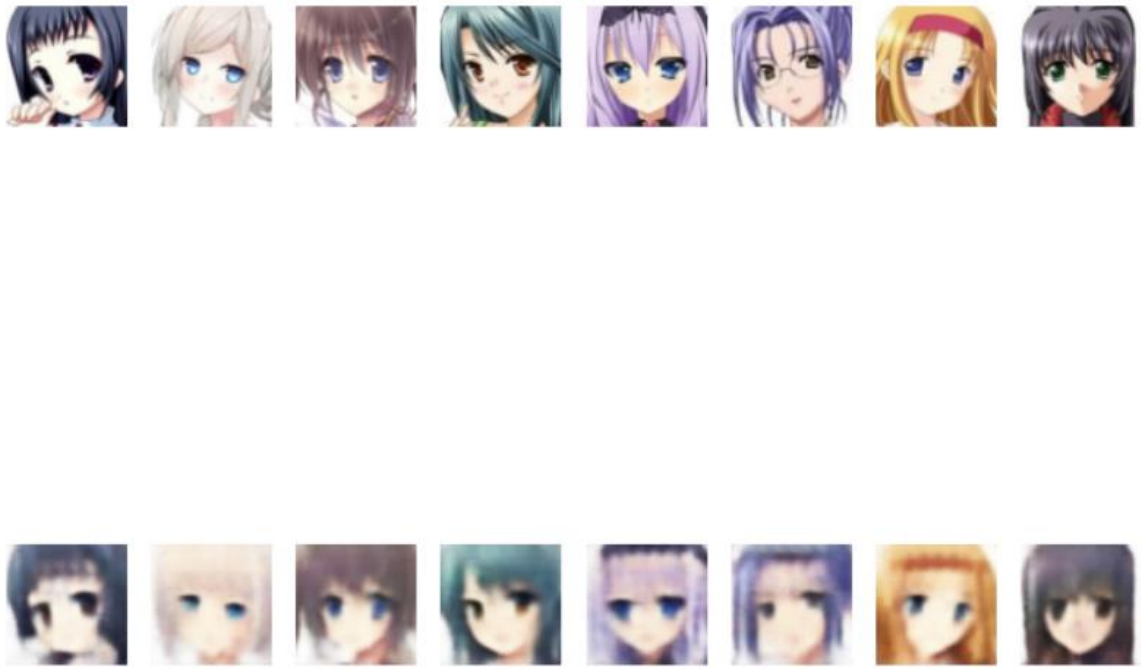
۱-۴. VQ_VAE

همانطور که گفتیم در vae ما برای ساده سازی فرض کردیم که توزیع اصلی نرمال با نیانگین صفر و یک باشد و توزیع هم نرمال به دست می‌اوردیم و گسسته و این از ایرداتش بود حال در اینجا ما برای جلوگیری از این اتفاق می‌اییم و به اندازه خروجی مدل کانولشون برای هرکدام یک بردار امبدینگ در نظر می‌گیریم و در ابتدا توزیع یونیفروم میگذاریم و حال خروجی کانولوشن را را مقایسه می‌کنیم با همه بردارای امبد شده و اونی که نرم فاصلش کمترین باشد را به عنوان متغییر تصادفی در نظر می‌گیریم و به انکودر می‌دهیم و حال برای لاس هم

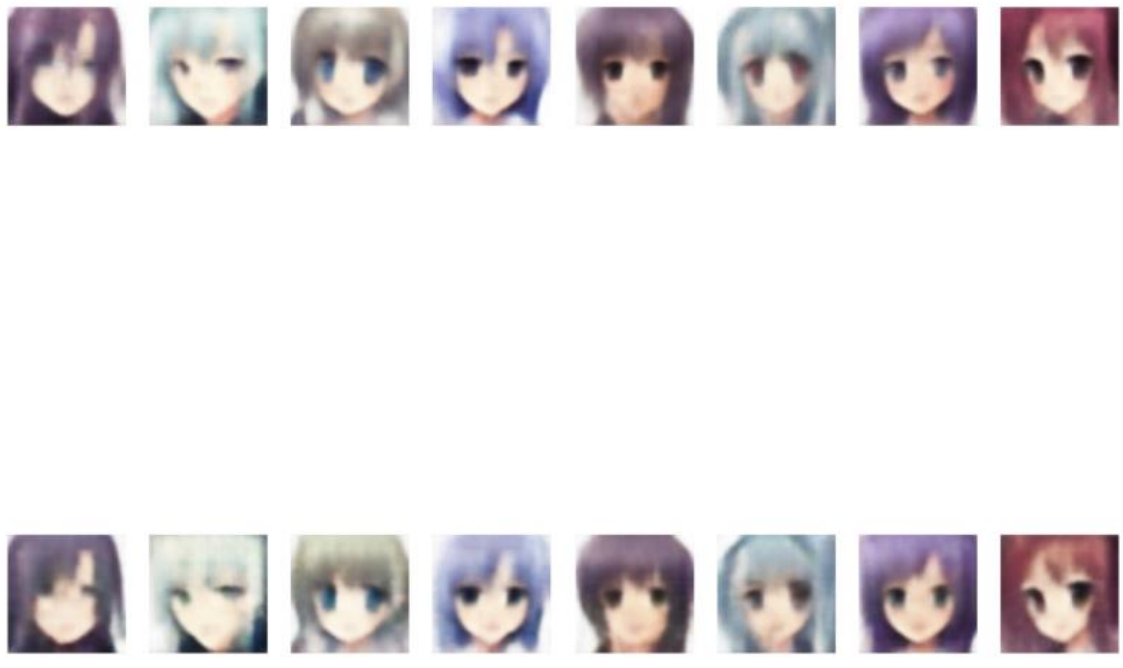
$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

همانطور که مشاهده می شود علاوه بر لاس عادی که ما مین اسکوار ارور گرفتیم در اینجا بر خروجی لایه گسسته سازم لاس داریم

حال بر همین اساس کد را مشابه حالتای قبل میزنیم



۸ تا ورودی و خروجی متانظر از vq_vae



۸ تا ورودی تولید شده توسط VAE و خروجی متانظر از vq_vae

که همانطور که مشاهده می‌شود خروجی‌هایی که برای ورودی های تویلد شده در بخش اول داشتیم را خوب تر یاد گرفته است

و خب در این حالت توزیع های دیگری هم میتونیم یاد بگیریم و از مزیت‌های آن است

پرسش ۲. Image Translation

1-2. آشنایی با image translatetion و معماری pixtopix

در مدل GAN ساده به دنبال پیدا کردن مپینگ از نویز رندوم z به تصویرخروجی y هستیم اما در pix2pix که یک معماری Conditional GAN است، هدف، یافتن تصویر خروجی y از ورودی تصویر x به همراه نویز رندوم z است.

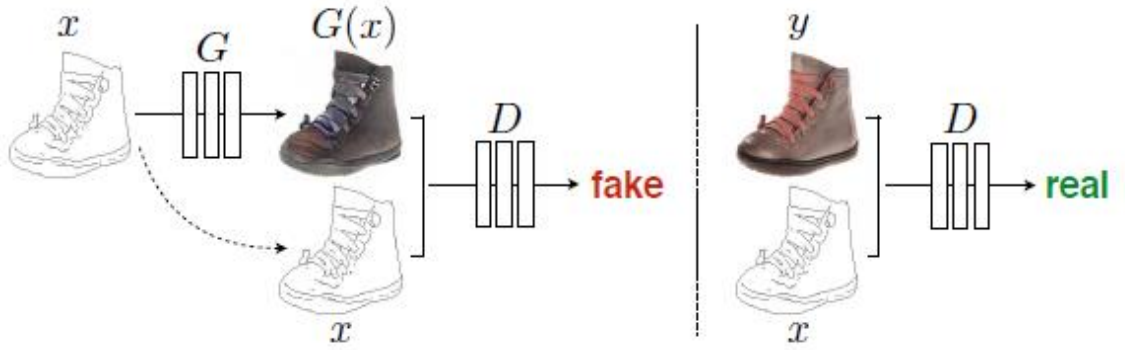
$$GAN: z \rightarrow y$$

$$CGAN(pix2pix):\{x, z\} \rightarrow y$$

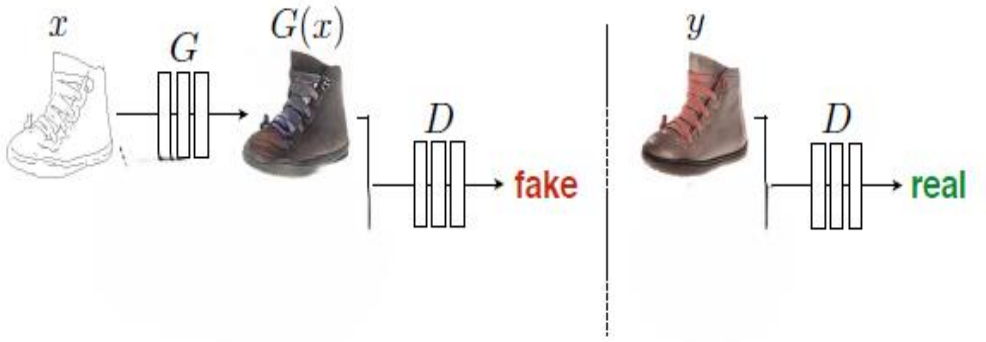
از کاربردهای این شبکه در زمینه پزشکی، می توان به بالابردن کیفیت تصاویر پزشکیMRI و کاهش نویز آن یا تبدیل تصویر CT scan به MRI scan اشاره کرد.

در مدل pix2pix سعی می شود علاوه بر فریب دادن بخش descriminator، خروجی تصویر، توسط تابع هزینه L1 (L2) به خروجی Grand Truth نزدیک شود. با این دو تابع هزینه، مپ کردن تصویر با فرکانس پایین (کلیات تصویر) موفقیت آمیز است اما در مپ کردن با فرکانس بالا (جزییات ویژگی های تصویر اعم از لبه ها و تغییر رنگ های شدید) خیر. بنابراین descriminator را به گونه ای طراحی می کنیم که در patch های N در N تصویر (N از ابعاد تصویر کوچک تر است) تصمیم گیری کند. بنابراین descriminator به صورت کانولوشن در مورد patch های تصویر، ساختار real یا fake بودن را بررسی می کند و وظیفه تشخیص صحت مپینگ در فرکانس بالا با descriminator و اعمال مپینگ در فرکانس پایین با تابع هزینه

(L2) L1 است. علاوه بر این، descriminator علاوه بر تصویر خروجی (grand truth یا خروجی generator)، تصویر ورودی به عنوان condition را دریافت می کند و با این دو تصویر، تشخیص fake یا real بودن را انجام می دهد.



تصویر . descriminator برای pix2pix



تصویر . descriminator برای GAN عادی

در ساختار Generator از U-net استفاده شده است. مدل encoder آن به صورت

CD64-C128-C256-C512-C512-C512-C512-C512

استفاده شده است (C64 یعنی کانولوشن با 64 فیلتر).

در decoder از ساختار

CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128

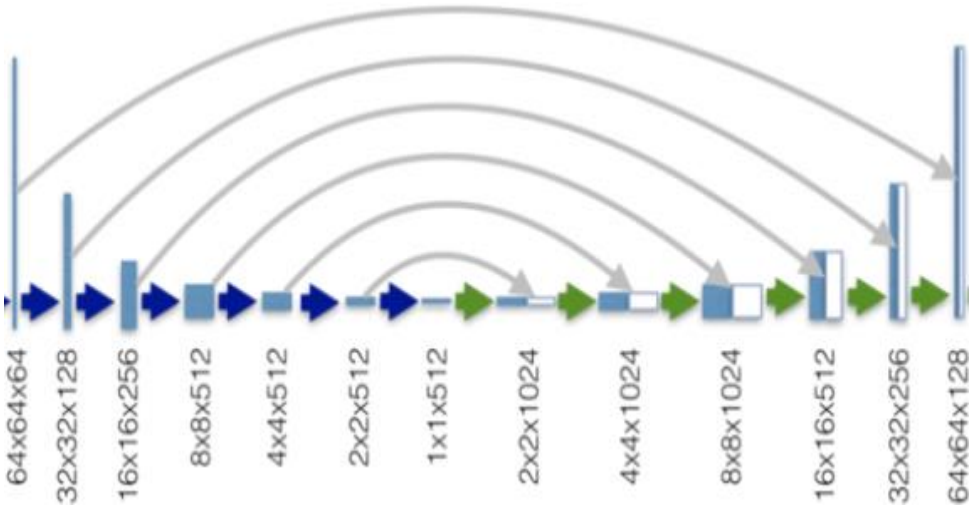
استفاده شده است (C1024 ، یعنی کانولوشن معکوس با 1024 فیلتر، CD512 یعنی کانولوشن معکوس با 512 فیلتر و سپس دراپ اوت).

پس از هر لایه در decoder از تابع فعالساز relu و در encoder از leakyrelu استفاده شده است. به جز در لایه اول encoder، در بقیه لایه ها در encoder، از batchnormalizatoin استفاده شده است. در این ساختار، از لایه i در encoder به لایه n-i از decoder اتصال برقرار است که n تعداد کل لایه های شبکه است و عمل concatenation بین لایه های این دو برقرار می شود. پس از آخرین لایه در decoder، با اعمال کانولوشن، خروجی شبکه را به صورت تصویری با سه کانال تبدیل می کنیم و سپس تابع فعالساز tanh را اعمال می کنیم. تمامی فیلترها 4 در 4 با 2 stride هستند.

جدول . ساختار شبکه U-net برای ورودی (3، 256، 256)

شماره لایه	اندازه طول خروجی	اندازه عرض خروجی	تعداد لایه خروجی
1	128	128	64
2	64	64	128
3	32	32	256
4	16	16	512
5	8	8	512
6	4	4	512
7	2	2	512
8	1	1	512
9	2	2	1024
10	4	4	1024
11	8	8	1024
12	16	16	1024
13	32	32	512
14	64	64	256
15	128	128	128

شبکه پیاده سازی شده، تنها از dropout هم در training و هم در test، به عنوان نویز در بخش generator برای تولید داده های تصادفی استفاده می کند. با وجود نویز البته، تولید تصادفی کمی در شبکه برقرار است اما از استفاده مقادیر تصادفی در ورودی شبکه موثرتر است.



تصویر . مدل U-net برای ورودی (3، 128، 128)

تابع هزینه نهایی شبکه به صورت زیر است. $L_{cGAN(G,D)} = E_{x,y}[logD(x,y)] + E_{x,z}[log(1 - D(G(z,x),x))]$

در این تابع هزینه، descriminator علاوه بر ورودی عادی (خروجی شبکه مولد یا عکس grandtruth) ورودی شبکه مولد را نیز دریافت می کند. فرایند یادگیری به صورت گام به گام انجام می شود.

- یادگیری Discriminator

1. تولید داده های فیک در generator
2. محاسبه هزینه در Discriminator (از رابطه بالا، مربوط به داده های تولیدی generator و داده های grandtruth)
3. آپدیت کردن وزن ها در Discriminator با ماکزیمایز کردن تابع هزینه (میتوان قرینه این تابع را، هزینه تلقی کرد و با مینیمایز کردن قرینه این تابع یادگیری را انجام داد)

- یادگیری Generator

4. تولید داده های فیک در generator
5. محاسبه هزینه در generator ($E_{x,z}[log(1 - D(G(z,x),x))]$)
6. آپدیت کردن وزن های generator با مینیمایز کردن تابع هزینه محاسبه شده در بخش قبل
7. رفتن به بخش 1

علاوه بر تابع هزینه مذکور، از آنجا که generator علاوه بر فریب discriminatorوظیفه تولید داده های شبیه grandtruth را دارد، از تابع هزینه L1 یا L2 استفاده می شود. در زیر تابع هزینه L1 افزوده شده به generator ذکر شده است.

$$L_{L_1} = E_{x,z}$$

از تحقیقات در زمینه مدل pix2pix می توان به موارد زیر اشاره کرد:

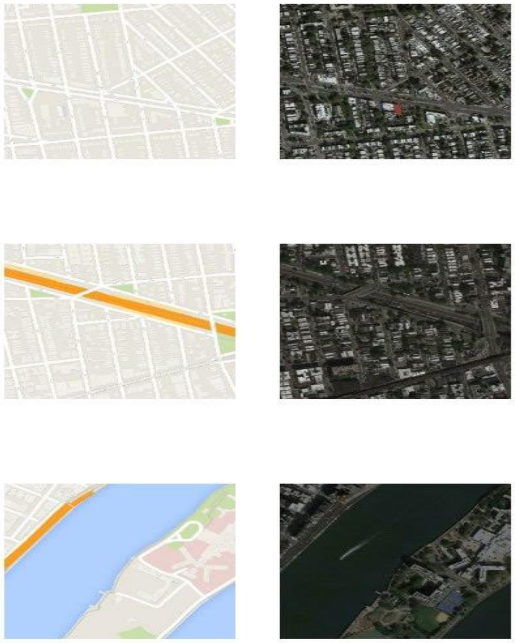
مقاله "Semantic Image Synthesis with Spatially-Adaptive Normalization" از روش نرمالسازی Spatially-Adaptive برای کنترل وزن ها و در نتیجه افزایش کیفیت عکس های تولید شده، استفاده می کند. این روش به صورت پویا پارامترهای نرمال سازی را تغیر می دهد و پارامترهای نرمالسازی برای هر فضای مکانی (پیکسل های نزدیک به همدیگر) از یک تصویر، متفاوت است (برخلاف روش های دیگر نرمال سازی که یک عمل یکسان را برای تمام پیکسل های یک کانال اعمال می کنند). این عمل باعث می شود که در حین فرایند تولید عکس ها، به ظاهر و روابط مفهومی، کنترل دقیق تری اعمال شود.

در مقاله " High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs" ، روش Pix2PixHD بیان شده است. ویژگی اصلی این روش استفاده از generator چند مقیاسه است که به صورت تدریجی تصویر با رزولوشن پایین را به تصویر به رزولوشن بالا تبدیل می کند. این بخش از چند بخش سری ازencoder-decoder (مثلا U-net) که با skip connection به یکدیگر متصل هستند تشکیل شده است. این روش همچنین از discriminator چند مقیاسه استفاده می کند تا بتواند تشخیص دقیق تری بر کیفیت داده های تولیدی داشته باشد.

2-2. پیاده سازی معماری Pix2Pix

لازم به ذکر است در پیاده‌سازی این شبکه، از پیاده‌سازی پیشنهادی مقاله در این لینک کمک گرفته شده است.

داده های map را لود می کنیم. این داده ها شامل 1096 داده آموزش و 1096 داده ولیدیشن است.



تصویر . نمونه تصاویر دیتاست (تصویر ستون راست، تصویر ورودی generator و تصویر ستون چپ، تصویر grandtruth خروجی است)

کلاس های Discriminator و Generator را مطابق توضیحات بخش قبل پیاده سازی می کنیم. در این دو کلاس از توابع Downsampling و Upsampling استفاده می کنیم. هر تابع Downsampling شامل یک لایه Convolution با استراید 2 و اندازه فیلتر 4 است (در صورت نیاز از batchnormalization استفاده می شود). همچنین در تابع Upsampling از یک لایه ConvolutionTranspose با استراید 2 و اندازه فیلتر 4 استفاده می شود. پس از لایه از Batchnormalization و در صورت نیاز از Dropout استفاده می شود.

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 256, 256, 3)]	0	[]
sequential_24 (Sequential)	(None, 128, 128, 64)	3072	['input_3[0][0]']
sequential_25 (Sequential)	(None, 64, 64, 128)	131584	['sequential_24[0][0]']
sequential_26 (Sequential)	(None, 32, 32, 256)	525312	['sequential_25[0][0]']
sequential_27 (Sequential)	(None, 16, 16, 512)	2099200	['sequential_26[0][0]']
sequential_28 (Sequential)	(None, 8, 8, 512)	4196352	['sequential_27[0][0]']
sequential_29 (Sequential)	(None, 4, 4, 512)	4196352	['sequential_28[0][0]']
sequential_30 (Sequential)	(None, 2, 2, 512)	4196352	['sequential_29[0][0]']
sequential_31 (Sequential)	(None, 1, 1, 512)	4196352	['sequential_30[0][0]']
sequential_32 (Sequential)	(None, 2, 2, 512)	4196352	['sequential_31[0][0]']
concatenate (Concatenate)	(None, 2, 2, 1024)	0	['sequential_32[0][0]', 'sequential_30[0][0]']
sequential_33 (Sequential)	(None, 4, 4, 512)	8390656	['concatenate[0][0]']
concatenate_1 (Concatenate)	(None, 4, 4, 1024)	0	['sequential_33[0][0]', 'sequential_29[0][0]']
sequential_34 (Sequential)	(None, 8, 8, 512)	8390656	['concatenate_1[0][0]']
concatenate_2 (Concatenate)	(None, 8, 8, 1024)	0	['sequential_34[0][0]', 'sequential_28[0][0]']
sequential_35 (Sequential)	(None, 16, 16, 512)	8390656	['concatenate_2[0][0]']
concatenate_3 (Concatenate)	(None, 16, 16, 1024)	0	['sequential_35[0][0]', 'sequential_27[0][0]']
sequential_36 (Sequential)	(None, 32, 32, 256)	4195328	['concatenate_3[0][0]']
concatenate_4 (Concatenate)	(None, 32, 32, 512)	0	['sequential_36[0][0]', 'sequential_26[0][0]']
sequential_37 (Sequential)	(None, 64, 64, 128)	1049088	['concatenate_4[0][0]']
concatenate_5 (Concatenate)	(None, 64, 64, 256)	0	['sequential_37[0][0]', 'sequential_25[0][0]']
sequential_38 (Sequential)	(None, 128, 128, 64)	262400	['concatenate_5[0][0]']
concatenate_6 (Concatenate)	(None, 128, 128, 128)	0	['sequential_38[0][0]', 'sequential_24[0][0]']
conv2d_transpose_14 (Conv2DTranspose)	(None, 256, 256, 3)	6147	['concatenate_6[0][0]']

تصویر . خلاصه مدل Generator

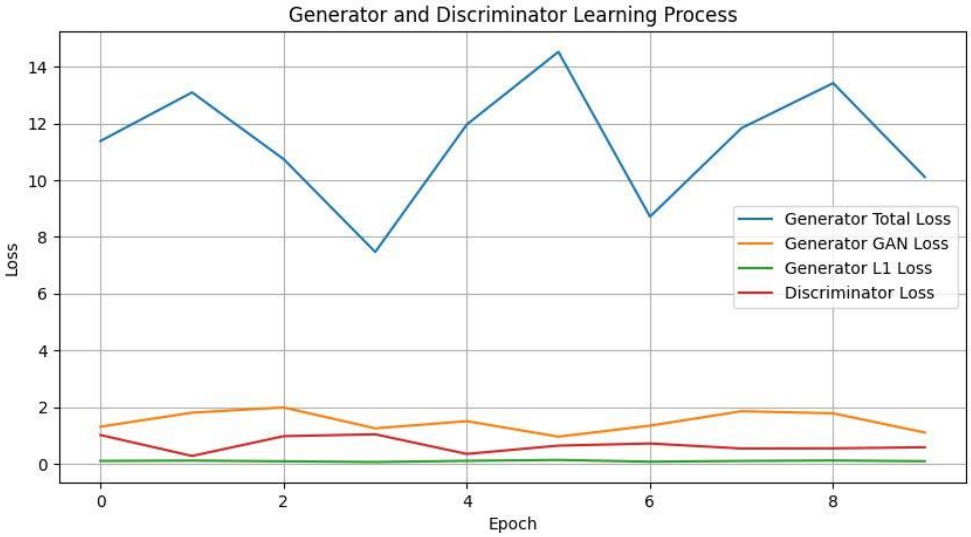
Layer (type)	Output Shape	Param #	Connected to
input_image (InputLayer)	[(None, 256, 256, 3)]	0	[]
target_image (InputLayer)	[(None, 256, 256, 3)]	0	[]
concatenate (Concatenate)	(None, 256, 256, 6)	0	['input_image[0][0]', 'target_image[0][0]']
sequential (Sequential)	(None, 128, 128, 64)	6144	['concatenate[0][0]']
sequential_1 (Sequential)	(None, 64, 64, 128)	131584	['sequential[0][0]']
sequential_2 (Sequential)	(None, 32, 32, 256)	525312	['sequential_1[0][0]']
zero_padding2d (ZeroPadding2D)	(None, 34, 34, 256)	0	['sequential_2[0][0]']
conv2d_3 (Conv2D)	(None, 31, 31, 512)	2097152	['zero_padding2d[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 31, 31, 512)	2048	['conv2d_3[0][0]']
leaky_re_lu_3 (LeakyReLU)	(None, 31, 31, 512)	0	['batch_normalization_2[0][0]']
zero_padding2d_1 (ZeroPadding2D)	(None, 33, 33, 512)	0	['leaky_re_lu_3[0][0]']
conv2d_4 (Conv2D)	(None, 30, 30, 1)	8193	['zero_padding2d_1[0][0]']

تصویر . خلاصه مدل Discriminator

تابع هزینه را برای Discriminator و Generator طبق بخش قبل به صورت جداگانه پیاده سازی می کنیم. از بهینه ساز Adam برای یادگیری استفاده می کنیم.

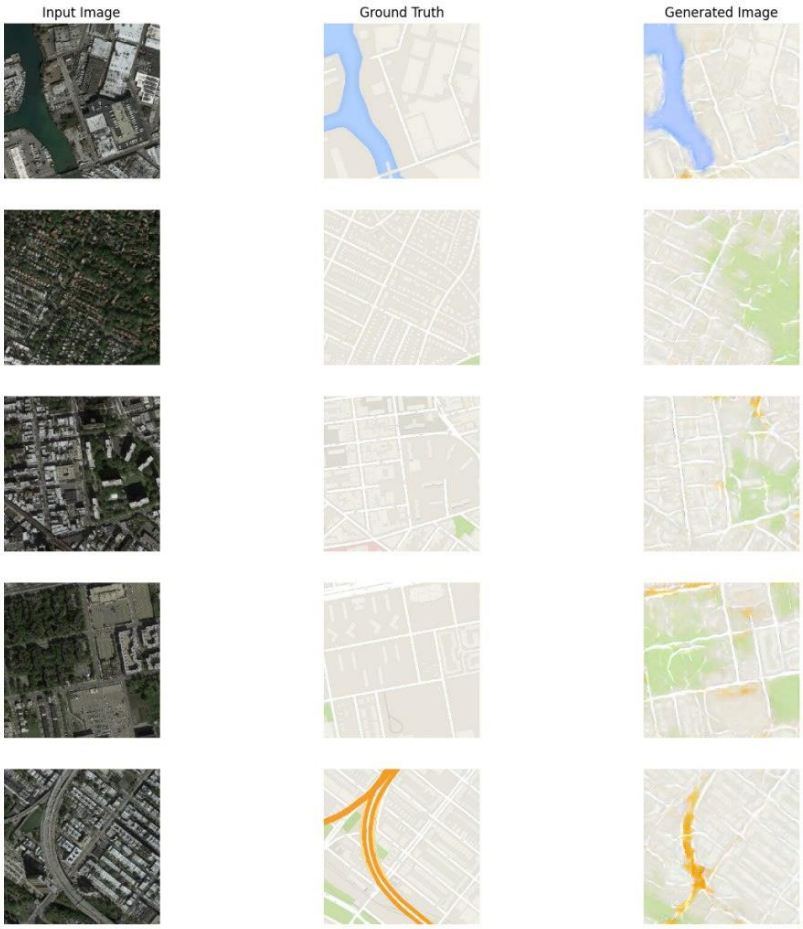
جدول .ابرپارامترهای مدل

Beta1_momentom	Learning_rate	Epochs	Batch_size	Lambda (coefficient of L1 in loss of generator)
0.5	0.0002	10	2	100



تصویر . نمودار خطا در Discriminator و Generator در فرایند آموزش

در هر ایپاک، 5 عکس از داده های validation را به عنوان ورودی به Generator داده و خروجی آن را نمایش می دهیم. تصاویر زیر، این نمایش برای epoch آخر یادگیری مدل است.



تصویر . نمونه تصاویر تولیدی شبکه Generator آموزش دیده به همراه تصویر ورودی و خروجی Grandtruth متناظر

همانطور که در تصویر مشاهده می شود، مدل Pix2Pix در تشخیص عوارض مشهود مثل رود و خیابان های مشهود و بزرگ و کوچه ها عملکرد مناسبی با وجود تعداد epoch آموزش کم داشته است. همچنین همانطور که در تصاویر 3 و 4 مشاهده می شود، مدل توانایی در تشخیص فضای سبز در تصویر ورودی داشته است، در صورتی که فضای سبز در تصویر Grandtruth دیده نمی شود. با این وجود با افزایش تعداد epoch می توان تصاویر با وجوح و روابط معنایی بهتر نیز تولید کرد.