

به نام خدا



دانشگاه تهران



دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین ششم

فاطمه رشیدی شهری

نام عضو اول

۶۱۰۳۹۹۱۳۱

شماره دانشجویی

آزاد وزیر پناه

نام عضو دوم

۶۱۰۳۹۹۱۸۲

شماره دانشجویی

فهرست

1.....	قوانين
3.....	پرسش ۱. Variational Auto-Encoder .
3.....	۱-۱. پیش پردازش دیتاست
3.....	۲-۱. ساخت VAE روی دیتاستها
7.....	۱-۲-۱. Anime Face دیتاست
9.....	۲-۲-۱. Cartoon Faces دیتاست
11.....	۳-۱. استفاده از یک مدل برای دو دیتاست
17.....	VQ-VAE .۴-۱
24.....	VQ-VAE-2 .۵-۱
27.....	پرسش ۲. Image Translation .
27.....	۱-۲. آشنایی با pix2Pix و معماری Image translation
36.....	۲-۲. پیاده سازی معماری Pix2Pix

شکل‌ها

5.....	شکل 1.تابع هزینه برای آموزش VAE
5.....	شکل 2.بخش encoder مربوط به VAE
6.....	شکل 3.بخش Decoder مربوط به VAE
7.....	شکل 4.نمونه دیتاست Anime Face
7.....	شکل 5.رونده آموزش VAE بر روی Anime Face
8.....	شکل 6.تصاویر تولیدی Anime با VAE
9.....	شکل 7.نمونه دیتاست Cartoon Face
9.....	شکل 8.رونده آموزش VAE بر روی Cartoon Face
10.....	شکل 9.تصاویر تولیدی Cartoon با استفاده از VAE
15.....	شکل 10.آموزش شبکه CVAE
15.....	شکل 11.تصاویر تولید شده با برچسب یک توسط CVAE
16.....	شکل 12.تصاویر تولید شده با برچسب صفر توسط CVAE
19.....	شکل 13.انکودر مربوط به VQ-VAE
20.....	شکل 14.دیکور VQ-VAE
21.....	شکل 15.هزینه آموزش VQ-VAE
22.....	شکل 16.تصاویر اصلی داده شده به مدل
22.....	شکل 17.تصاویر بازسازی شده توسط مدل
23.....	شکل 18.تصاویر تولیدی از مدل VAE
23.....	شکل 19.خروجی تصاویر ساختگی
28.....	شکل 20.ساختار مدل CGAN
29.....	شکل 21.ساختار discriminator در یک GAN ساده
29.....	شکل 22.ساختار discriminator
31.....	شکل 23.ساختار مولد
36.....	شکل 24.سه نمونه از تصاویر مجموعه آموزشی
37.....	شکل 25.تابع هزینه بر حسب استپ در فرایند یادگیری
38.....	شکل 26.تابع هزینه آخرین استپ در ایپاک در فرایند یادگیری
38.....	شکل 27.میانگین مقدار تابع هزینه در هر استپ بر حسب ایپاک در فرایند یادگیری
40.....	شکل 28.تعدادی از تصاویر مجموعه اعتبارسنجی پس از اولین ایپاک به همراه برچسب و تصویر تولید شده
41.....	شکل 29.تعدادی از تصاویر مجموعه اعتبارسنجی پس از دومین ایپاک به همراه برچسب و تصویر تولید شده
42.....	شکل 30.تعدادی از تصاویر مجموعه اعتبارسنجی پس از سومین ایپاک به همراه برچسب و تصویر تولید شده
43.....	شکل 31.تعدادی از تصاویر مجموعه اعتبارسنجی پس از چهارمین ایپاک به همراه برچسب و تصویر تولید شده
44.....	شکل 32.تعدادی از تصاویر مجموعه اعتبارسنجی پس از پنجمین ایپاک به همراه برچسب و تصویر تولید شده

- شکل 33. تعدادی از تصاویر مجموعه اعتبارسنجی پس از ششمین ایپاک به همراه برچسب و تصویر تولید شده 45
- شکل 34. تعدادی از تصاویر مجموعه اعتبارسنجی پس از هفتمین ایپاک به همراه برچسب و تصویر تولید شده 46
- شکل 35. تعدادی از تصاویر مجموعه اعتبارسنجی پس از هشتمین ایپاک به همراه برچسب و تصویر تولید شده 47
- شکل 36. تعدادی از تصاویر مجموعه اعتبارسنجی پس از نهمین ایپاک به همراه برچسب و تصویر تولید شده 48
- شکل 37. تعدادی از تصاویر مجموعه اعتبارسنجی پس از دهمین ایپاک به همراه برچسب و تصویر تولید شده 49

جدول‌ها

13.....	جدول 1. بخش انکودر CVAE
14.....	جدول 2. بخش دیکودر CVAE
36.....	جدول 2. تعداد پارامترها در مولد
36.....	جدول 3. تعداد پارامترها در discriminator
37.....	جدول 4. هایپرپارامترهای مورد استفاده در آموزش مدل pix2pix

قوانين

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 تمرين‌ها را در قالب گروه‌های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحويل تک نفره نیز نمره‌ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرين اول را با شخص A و تمرين دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژه‌ای برخوردار است؛** بنابراین، لطفاً 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 تمرين کسر می‌شود.
- کدها حتماً باید در قالب نوت‌بوک با پسوند **.ipynb** تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتماً 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 - لحظ می‌شود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- استفاده از کدهای آماده برای تمرين‌ها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحويل 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.

لطفاً 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

۱-۱. پیش پردازش دیتاست

چون که تسک مورد استفاده از این دیتاست، آموزش یک VAE است و همچنین تعداد داده‌ها کافی بوده و نیازی به تقویت داده وجود ندارد، بنابراین تنها پیش‌پردازش منطقی که می‌توان روی این داده‌ها انجام داد، این است که:

- برای ساده‌تر شدن آموزش شبکه، این داده‌ها را به سایزهای کوچک‌تر resize کنیم. ما این کار را با سایز نهایی $64 \times 64 \times 3$ انجام دادیم. بنابراین به مدل تصاویری به این اندازه را ورودی خواهیم داد. (با اینکه احتمالاً تصاویر به دست آمده بهترین حالت خود نباشند، اما به ناچار مجبور هستیم که سایز تصاویر را کوچک کنیم تا قادر به اجرا و آموزش مدل باشیم)
- کار دیگری که انجام می‌دهیم، نرمالایز کردن تصاویر است. با تقسیم کردن هر پیکسل بر ۲۵۵ تصاویر در واقع یک ماتریس با مقادیر بین ۰ تا ۱ خواهند بود که آموزش را بهتر می‌کند.

۲-۱. ساخت VAE روی دیتاست‌ها

یک نوع معماری شبکه‌های عصبی عمیق است که برای یادگیری VAE یا Variational Autoencoder یک فضای نهان (فضای latent) از داده‌ها بدون نیاز به برچسب و لیبل استفاده می‌شود. این مدل‌های شبکه عصبی معمولاً برای تولید داده‌های جدید یک توزیع احتمالی یاد می‌گیرند که نماینده‌ی فضای ورودی است. این مدل از بخش‌های مختلفی تشکیل شده که به توضیح آن‌ها می‌پردازیم.

• Encoder یا بخش کدگذار مدل:

در VAE، داده‌های ورودی به یک فضای نهان با بعد کمتر نگاشت می‌شوند. این مرحله به عنوان Encoder شناخته می‌شود. Encoder در واقع یک شبکه عصبی است که با دریافت داده‌های ورودی، یک توزیع احتمالی نهان برای هر داده محاسبه می‌کند. این توزیع معمولاً یک توزیع نرمال با میانگین ۱ و واریانس ۰ است. در واقع این بخش سعی می‌کند تا با تنظیم وزن‌های خود، توزیع فضای نهان داده‌های ورودی را به توزیع نرمال با میانگین ۰ و واریانس ۱ نزدیک کند.

• Latent Space یا همان فضای نهان:

فضای نهان معمولاً یک فضای بعد پایین است که نمایانگر ویژگی‌های مهم و توزیع آماری از داده‌های ورودی است. هدف اصلی این است که داده‌های ورودی به یک نقطه در این فضای نهان نگاشت شوند که می‌تواند برای تولید داده‌های جدید استفاده شود. این نقطه در واقع بیانگر توزیع نرمال با میانگین ۰ و واریانس ۱ است.

Decoder یا همان بخش کد گشا:

پس از اینکه داده‌ها به فضای نهان نگاشت شوند، یک شبکه کد گشا (decoder) نیز وجود دارد که این نقاط را به داده‌های اولیه بازمی‌گرداند. این مرحله به عنوان فرایند تصادفی بازسازی معروف است، زیرا از یک نقطه در فضای نهان، می‌توان چندین توزیع احتمالی مختلفی را بازسازی کرد.

حال به بررسیتابع هزینه یا همان loss function می‌پردازیم. Loss function در VAE شامل دو بخش اصلی است که به عنوان یک ترند واریانس تعیین شده است:

• Reconstruction loss (هزینه بازسازی):

این بخش از loss function تلاش می‌کند تا اطمینان حاصل کند که داده‌های ورودی از فضای نهان به درستی بازسازی می‌شوند. این هزینه معمولاً بر اساس تفاوت بین داده‌های ورودی اصلی و داده‌های بازسازی شده محاسبه می‌شود. معمولاً از خطای میانگین مربعات خطأ (mean squared error) یا خطای میانگین مطلق (mean absolute error) استفاده می‌شود.

• KL Divergence Loss (Kullback-Leibler):

این بخش از تابع هزینه به کمک محاسبه انحراف KL بین توزیع احتمالی چگالی نهان Encoder و توزیع مرجع (ممولاً یک توزیع نرمال است که با میانگین صفر و واریانس یک) محاسبه می‌شود. این بخش از loss function مطمئن می‌شود که توزیع چگالی نهان نزدیک به توزیع نرمال استاندارد باشد، که این کمک می‌کند تا فضای نهان منظم و قابل تفسیر باشد.

در کل، ترکیب این دو هزینه در تابع هزینه VAE باعث می‌شود که شبکه بتواند به طور همزمان داده‌ها را از فضای نهان بازسازی کند و همچنین این فضا را به گونه‌ای مشخص و قابل تفسیر شکل دهد که برای تولید داده‌های جدید مفید باشد. همچنین می‌توانیم برای آموزش بهتر، یک وزن، بسته تسک مورد نظر ما (مثلًا در اینجا هزینه بازسازی) به هر یک از هزینه‌ها بدھیم تا مدل بیشتر به آن اهمیت بدهد. در اینجا ضریب هزینه بازسازی، ۱۰۰ است.

بنابراین هزینه به این صورت تعریف می‌شود:

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_{\theta}(z|x_i)} [\log p_{\phi}(x_i | z)] + \mathbb{KL}(q_{\theta}(z | x_i) || p(z))$$

شکل 1.تابع هزینه برای آموزش VAE

حال VAE خود را ساخته و تعریف می‌کنیم. این مدل از لایه‌های کانولوشنی، batch normalization و همچنین از لایه‌های fully connected برای نمایش توزیع‌ها و در واقع فضای نهان یا latent DropOut استفاده می‌کند. چگونگی تعریف مدل خود را بررسی می‌کنیم.

بخش آن به این صورت است:

Model: "encoder"				
Layer (type)	Output Shape	Param #	Connected to	
input_45 (InputLayer)	[None, 64, 64, 3]	0	[]	
conv2d_75 (Conv2D)	(None, 32, 32, 8)	224	['input_45[0][0]']	
conv2d_76 (Conv2D)	(None, 16, 16, 16)	1168	['conv2d_75[0][0]']	
conv2d_77 (Conv2D)	(None, 8, 8, 32)	4640	['conv2d_76[0][0]']	
conv2d_78 (Conv2D)	(None, 4, 4, 64)	18496	['conv2d_77[0][0]']	
flatten_20 (Flatten)	(None, 1024)	0	['conv2d_78[0][0]']	
dense_68 (Dense)	(None, 1024)	1049600	['flatten_20[0][0]']	
z_mean (Dense)	(None, 256)	262400	['dense_68[0][0]']	
z_log_var (Dense)	(None, 256)	262400	['dense_68[0][0]']	
z (Sampling)	(None, 256)	0	['z_mean[0][0]', 'z_log_var[0][0]']	
<hr/>				
Total params: 1598928 (6.10 MB)				
Trainable params: 1598928 (6.10 MB)				
Non-trainable params: 0 (0.00 Byte)				

شکل 2. بخش encoder مربوط به VAE

همچنین بخش decoder آن به این صورت است:

Layer (type)	Output Shape	Param #
input_46 (InputLayer)	[(None, 256)]	0
dense_69 (Dense)	(None, 1024)	263168
dense_70 (Dense)	(None, 1024)	1049600
reshape_24 (Reshape)	(None, 4, 4, 64)	0
conv2d_transpose_108 (Conv2DTranspose)	(None, 8, 8, 64)	36928
conv2d_transpose_109 (Conv2DTranspose)	(None, 16, 16, 32)	18464
conv2d_transpose_110 (Conv2DTranspose)	(None, 32, 32, 16)	4624
conv2d_transpose_111 (Conv2DTranspose)	(None, 64, 64, 8)	1160
conv2d_transpose_112 (Conv2DTranspose)	(None, 64, 64, 3)	219
<hr/>		
Total params: 1374163 (5.24 MB)		
Trainable params: 1374163 (5.24 MB)		
Non-trainable params: 0 (0.00 Byte)		

شکل 3. بخش Decoder مربوط به VAE

همانطور که مشخص است، فضای نهان، یک فضای ۲۵۶ بعدی در نظر گرفته شده تا بتوانیم دقیق‌تر تصاویر ورودی را بررسی کنیم و همچنین بتوانیم در بخش generation، تصاویر متنوع‌تری بسازیم.

تابع هزینه مدل نیز در بخش قبل توضیح داده شد که شامل هزینه بازسازی و هزینه KL بین توزیع فضای نهان و توزیع نرمال می‌باشد. از Adam optimizer برای آموزش شبکه‌های استفاده می‌شود تا بتواند در تعداد کم ایپاک (در اینجا مدل‌ها را بین ۱۰ تا ۱۵ ایپاک آموزش می‌دهیم)، نتایج نسبتاً قابل قبولی داشته باشد. Data Generator را نیز برابر ۳۲ در نظر می‌گیریم که این هایپرپارامتر را در قسمت Batch size پیاده‌سازی کردیم.

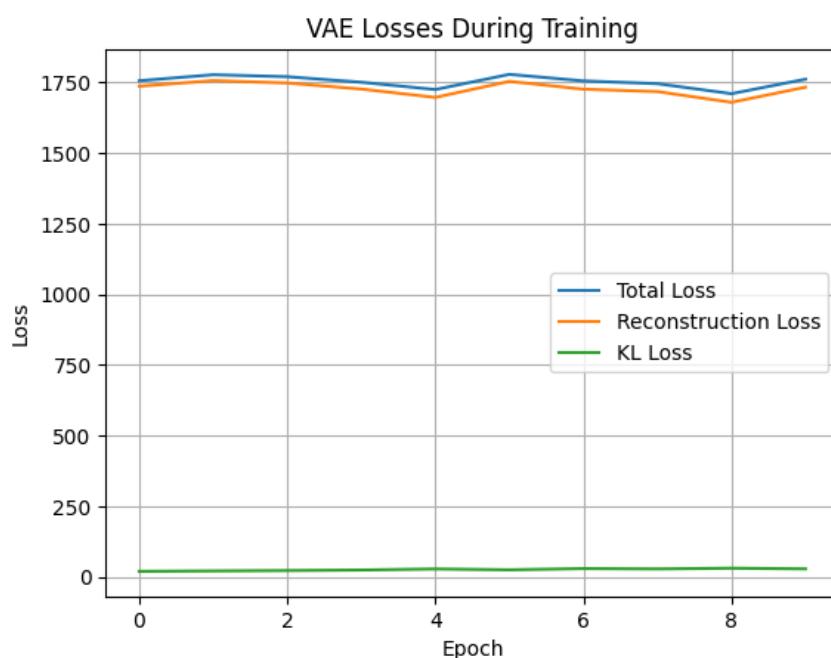
۱-۲-۱ دیتاست Anime Face

در این مرحله، سعی داریم VAE تعریف شده خود را با دیتاست آموزش بدهیم. در تصویر زیر می‌توانیم ۸ نمونه از این دیتاست را که به اندازه 64×64 ، resize شده‌اند را ببینیم.



شکل ۴. نمونه دیتاست Anime Face

حال به آموزش مدل می‌پردازیم. روند آموزش مدل به صورت زیر است که می‌توانیم نمودار آن را نیز در شکل ۴ ببینیم.



شکل ۵. روند آموزش VAE بر روی Anime Face

همانطور که از روند آموزش مشخص است، گویی که آموزشی رخ نداده و loss ثابت مانده است. اما چون ما ضریب 100 برابر Reconstruction loss در نظر گرفتیم، و حتی با تغییر کوچک آن مدل بهتر می‌شود، بنابراین لزومی به تغییرات آنچنانی loss نبوده و مدل همچنان آموزش دیده. در تصویر زیر، می‌توانیم 8 نمونه از تصاویر تولیدی با استفاده از یک نویز که از توزیع نرمال استاندارد آمده و به وسیله این مدل، بدست آمده را ببینیم. همچنانی لازم به ذکر است که به دلیل خیلی کوچک بودن KL loss نسبت به Reconstruction loss، این هزینه تقریباً صفر به نظر می‌رسد.



شکل 6. تصاویر تولیدی **VAE** با **Anime**

همانطور که مشخص است، مدل به خوبی آموزش دیده و تصاویر بسیار نزدیکی به تصاویر دنیای واقعی تولید می‌کند. این مدل هم از نظر تولید تصاویر متفاوت، اما نزدیک به هم و در یک زمینه و هم از نظر کیفیت ساخت، نتیجه خوبی دارد.

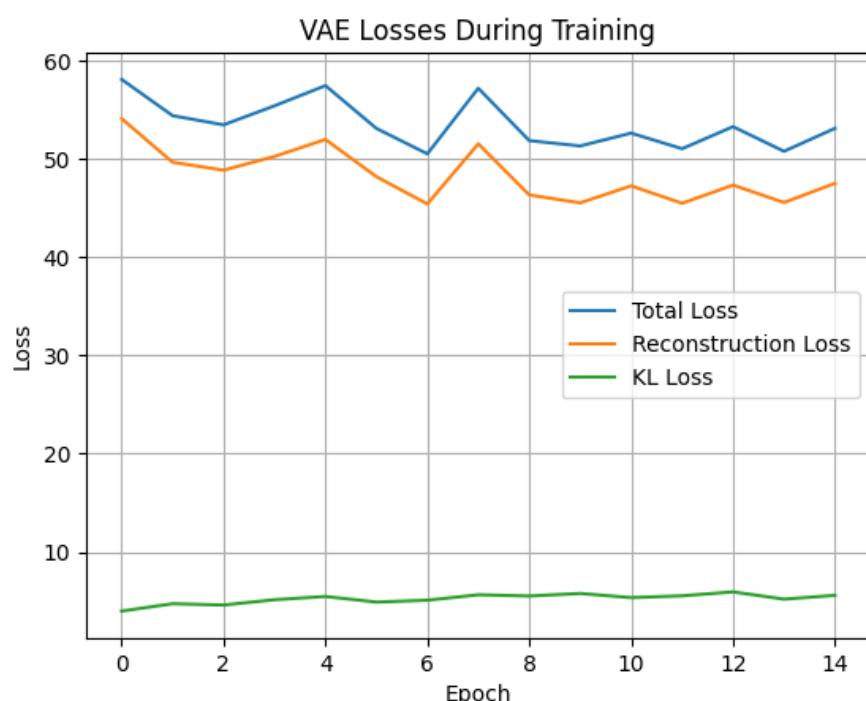
۲-۲-۱ دیتاست Cartoon Faces.

در این مرحله، سعی داریم VAE تعریف شده خود را با Cartoon Face دیتاست آموزش بدهیم. در تصویر زیر می‌توانیم ۸ نمونه از این دیتاست را که به اندازه 64×64 ، resize شده‌اند را ببینیم.



شکل ۷. نمونه دیتاست Cartoon Face

حال به آموزش مدل می‌پردازیم. روند آموزش مدل به صورت زیر است که می‌توانیم نمودار آن را نیز در شکل ۷ ببینیم.



شکل ۸. روند آموزش VAE بر روی Cartoon Face

همانطور که از نمودار شکل ۷ مشخص است، هزینه در طول آموزش تا حدی کم شده. KL به دلیل اینکه کوچک است، تفاوت آن نسبتاً نامحسوس بوده و کامل مشخص نیست. اما KL از شروع کمی بیشتر شده، افزایش این هزینه، کمی به تنوع تصاویر تولیدی مدل در آینده کمک خواهد کرد، چرا که اگر این توزیع کمی متفاوت‌تر از توزیع نرمال استاندار باشد (برای مثال واریانس بیشتری داشته باشد)، در این صورت تنوع تولید تصاویر بالاتر خواهد رفت.

در شکل شماره ۸، می‌توانیم تصاویر و نمونه‌های تولیدی توسط این VAE را ببینیم.



شکل ۹. تصاویر تولیدی **Cartoon** با استفاده از VAE

همانطور که مشخص است، مدل بر روی این دیتاست نیز عملکرد خوبی داشته و به خوبی آموزش دیده. علت کمی بد کیفیت بودن تصاویر، سایز کوچک آن‌ها است. اگر سخت افزار مناسب به اندازه کافی موجود باشد و تصاویر بزرگ‌تر در نظر بگیریم، نتایج بهتر خواهند شد.

طبق گفته قبلی در مورد KL loss، نمونه‌های تولیدی با یکدیگر تفاوت دارند و در واقع diversity نسبتاً خوبی را از نمونه‌های تولیدی شاهد هستیم. تفاوت‌هایی از قبیل مو، رنگ مو و رنگ پوست به خوبی در نمونه‌های تولیدی دیده می‌شوند که نشان دهنده آموزش خوب مدل و به رسیدن به هدف از تعریف VAE است.

۱-۳. استفاده از یک مدل برای دو دیتاست

یک نمونه و در واقع حالت extend شده از Conditional Variational Autoencoder (CVAE) است که امکان تولید داده‌های مشروط بر روی ویژگی‌های خاصی را فراهم می‌کند. در حالی که VAE یک مدل تولید کننده یا generator غیرشرطی است که بدون هیچ گونه شرط خاصی داده‌های جدید را تولید می‌کند، CVAE به شما اجازه می‌دهد که داده‌های تولید شده را با توجه به ویژگی‌های ورودی خاصی کنترل کنید در واقع می‌توانید از مدل بخواهید تا داده‌های مختلف در کلاس‌های مختلفی را تولید کند و تنها چیزی که لازم است، این است که به مدل، کلاسی که قصد تولید آن را دارید، را اطلاع دهید. حال به بررسی مدل بپردازیم.

Encoder •

مشابه با VAE، در CVAE هم داده‌های ورودی به یک فضای نهان با بعد کمتر نگاشت می‌شوند. تفاوت اصلی این است که در CVAE، علاوه بر داده‌های ورودی، یک متغیر شرطی هم وارد کدگذار می‌شود که در واقع بیانگر کلاس آن تصویر است. این متغیر شرطی می‌تواند هر ویژگی یا برچسبی باشد که بخواهیم داده‌های تولید شده بر اساس آن کنترل شوند. خروجی Encoder، توزیع احتمالی چگالی نهان با میانگین و واریانس مشروط بر لیبل ورودی است.

فضای نهان: •

فضای نهان مشابه VAE است، اما اکنون نگاشت داده‌ها به این فضا به طور مشروط بر متغیر کلاس ورودی انجام می‌شود.

Decoder •

کدگشا در CVAE داده‌های نهان را همراه با متغیر شرطی c به داده‌های اولیه بازمی‌گرداند. این به این معناست که بازسازی داده‌ها نه تنها بر اساس نقطه‌ای در فضای نهان بلکه با توجه به ویژگی‌های شرطی انجام می‌شود.

تابع هزینه در این مدل نیز مانند همان VAE ساده است. تنها تفاوتی این تابع هزینه این مدل دارد، این است که مقدار آن مشروط و بر اساس لیبل ورودی محاسبه شده و بر مدل تاثیرگذار است و یک هزینه کلی برای مدل تعریف نشده، بلکه هزینه مشروط است.

تفاوت دیگر این دو مدل در این است که CVAE، علاوه بر داده‌های خام، یک کلاس و لیبل نیز به عنوان ورودی دارد.

حال به بررسی مدل پیاده‌سازی شده می‌پردازیم. در جدول‌های ۱۰ و ۱۱ به ترتیب می‌توانیم لایه‌های استفاده شده در دیکودر و انکودر این مدل را مشاهده کنیم. سمپلینیگ استفاده شده در این مدل دقیقاً مانند سمپلینیگ است که در VAE ساده استفاده می‌کنیم. همچنین loss function تعريف شده برای این مدل نیز دقیقاً مانند همانتابع استفاده شده در VAE‌هایی است که آموزش دادیم.

این بار از لایه‌های MaxPooling و DropOut علاوه بر لایه‌های کانولوشنی و نیز استفاده کردیم fully connected.

Model: "encoder"

Layer (type)	Output Shape	Param #	Connected to
input_layer_7 (InputLayer)	(None, 2)	0	-
reshape_4 (Reshape)	(None, 1, 1, 2)	0	input_layer_7[0]...
input_layer_6 (InputLayer)	(None, 64, 64, 3)	0	-
up_sampling2d_2 (UpSampling2D)	(None, 64, 64, 2)	0	reshape_4[0][0]
concatenate_4 (Concatenate)	(None, 64, 64, 5)	0	input_layer_6[0]... up_sampling2d_2[...]
conv2d_8 (Conv2D)	(None, 64, 64, 16)	736	concatenate_4[0]...
batch_normalizatio... (BatchNormalizatio...)	(None, 64, 64, 16)	64	conv2d_8[0][0]
re_lu_16 (ReLU)	(None, 64, 64, 16)	0	batch_normalizat...
max_pooling2d_8 (MaxPooling2D)	(None, 32, 32, 16)	0	re_lu_16[0][0]
conv2d_9 (Conv2D)	(None, 32, 32, 32)	4,640	max_pooling2d_8[...]
batch_normalizatio... (BatchNormalizatio...)	(None, 32, 32, 32)	128	conv2d_9[0][0]
re_lu_17 (ReLU)	(None, 32, 32, 32)	0	batch_normalizat...
max_pooling2d_9 (MaxPooling2D)	(None, 16, 16, 32)	0	re_lu_17[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 64)	18,496	max_pooling2d_9[...]
batch_normalizatio... (BatchNormalizatio...)	(None, 16, 16, 64)	256	conv2d_10[0][0]
re_lu_18 (ReLU)	(None, 16, 16, 64)	0	batch_normalizat...

max_pooling2d_10 (MaxPooling2D)	(None, 8, 8, 64)	0	re_lu_18[0][0]
conv2d_11 (Conv2D)	(None, 8, 8, 128)	73,856	max_pooling2d_10...
batch_normalizatio... (BatchNormalizatio...)	(None, 8, 8, 128)	512	conv2d_11[0][0]
re_lu_19 (ReLU)	(None, 8, 8, 128)	0	batch_normalizat...
max_pooling2d_11 (MaxPooling2D)	(None, 4, 4, 128)	0	re_lu_19[0][0]
flatten_12 (Flatten)	(None, 2048)	0	max_pooling2d_11...
z_mean (Dense)	(None, 256)	524,544	flatten_12[0][0]
z_log_var (Dense)	(None, 256)	524,544	flatten_12[0][0]
z (Sampling)	(None, 256)	0	z_mean[0][0], z_log_var[0][0]

Total params: 1,147,776 (4.38 MB)
Trainable params: 1,147,296 (4.38 MB)
Non-trainable params: 480 (1.88 KB)

CVAE جدول 1. بخش انکودر

همان طور که قابل مشاهده است، لیبل‌ها نیز به این بخش داده شده و سپس این لیبل‌های به اندازه تصویر (64×64) در آمده و سپس با تصویر ورودی کانکت شده و به مدل وارد می‌شود و سپس مابقی محاسبات انجام می‌شود.

Model: "decoder"

Layer (type)	Output Shape	Param #	Connected to
input_layer_8 (InputLayer)	(None, 256)	0	-
input_layer_7 (InputLayer)	(None, 2)	0	-
concatenate_5 (Concatenate)	(None, 258)	0	input_layer_8[0]... input_layer_7[0]...
dense_2 (Dense)	(None, 2048)	530,432	concatenate_5[0]...
reshape_5 (Reshape)	(None, 4, 4, 128)	0	dense_2[0][0]
conv2d_transpose_10 (Conv2DTranspose)	(None, 8, 8, 128)	147,584	reshape_5[0][0]
batch_normalizatio... (BatchNormalizatio...)	(None, 8, 8, 128)	512	conv2d_transpose...
re_lu_20 (ReLU)	(None, 8, 8, 128)	0	batch_normalizat...
conv2d_transpose_11 (Conv2DTranspose)	(None, 16, 16, 64)	73,792	re_lu_20[0][0]
batch_normalizatio... (BatchNormalizatio...)	(None, 16, 16, 64)	256	conv2d_transpose...

re_lu_21 (ReLU)	(None, 16, 16, 64)	0	batch_normalizat...
conv2d_transpose_12 (Conv2DTranspose)	(None, 32, 32, 32)	18,464	re_lu_21[0][0]
batch_normalizatio... (BatchNormalizatio...	(None, 32, 32, 32)	128	conv2d_transpose...
re_lu_22 (ReLU)	(None, 32, 32, 32)	0	batch_normalizat...
conv2d_transpose_13 (Conv2DTranspose)	(None, 64, 64, 16)	4,624	re_lu_22[0][0]
batch_normalizatio... (BatchNormalizatio...	(None, 64, 64, 16)	64	conv2d_transpose...
re_lu_23 (ReLU)	(None, 64, 64, 16)	0	batch_normalizat...
conv2d_transpose_14 (Conv2DTranspose)	(None, 64, 64, 3)	435	re_lu_23[0][0]

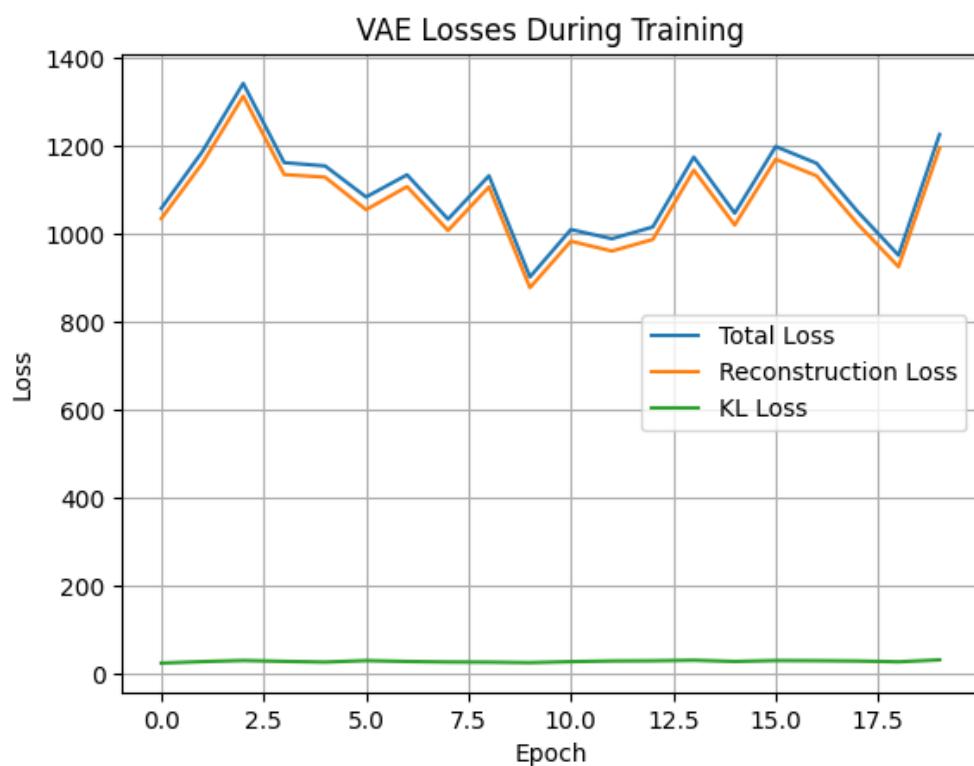
Total params: 776,291 (2.96 MB)
Trainable params: 775,811 (2.96 MB)
Non-trainable params: 480 (1.88 KB)

جدول 2. بخش دیکودر CVAE

در این قسمت از مدل نیز همانطور که مشخص است، لیبل‌ها تاثیر گذار هستند و با ورودی دادن آن‌ها به مدل و سپس کانکت کردن آن‌ها با sample ورودی، می‌توانیم مابقی محاسبات را برای بازسازی و یا در واقع reconstruction در پیش بگیریم.

برای آموزش این مدل، از هر کدام از مجموعه دادگان ۱۵۰۰۰ نمونه برداشته (حدود ۵۰ درصد از هر کدام) و لیبل متناظر با آن‌ها را نیز به صورت one-hot می‌سازیم و همراه تصویر، به مدل می‌دهیم. علت انتخاب تعداد یکسان از هر کدام از مجموعه‌ها این است که مدل تقریباً تعداد یکسانی از هر مجموعه دیده و تقریباً هر کدام را به صورت یکسان و به یک اندازه خوب یاد خواهد گرفت.

می‌توانیم نمودار loss‌های مختلف برای این مدل را در زمان آموزش در شکل ۱۰ بررسی کنیم. مدل در طول زمان (ایپاک) loss خود را کم کرده و فرآیند آموزش خود را به خوبی طی می‌کند. KL loss به دلیل کوچک‌تر بودن (وجود ضریب برای reconstruction loss)، تغییرات نامحسوسی داشته و همچنین در مقابل هزینه کلی، نزدیک به صفر به نظر می‌رسد.



شکل 10. آموزش شبکه CVAE

حال برای ارزیابی مدل آموزش دیده، طبق خواسته سوال، ۸ عدد تصویر به وسیله این مدل تولید خواهیم کرد.



شکل 11. تصاویر تولید شده با برچسب یک نوسط CVAE



شکل 12. تصاویر تولید شده با برچسب صفر توسط CVAE

همانطور که مشخص است مدل به خوبی هر چه تمام آموزش دیده و تصویر جدید تولید می‌کند. چرا که اولاً کیفیت تصاویر تولیدی نسبتاً بالا بوده و از طرف دیگر، مدل قادر به تشخیص دقیق تصویر تولیدی با ورودی دادن لیبل مرتبط است. در واقع مدل به دقت تمام می‌داند که با چه ورودی‌ای (لیبل)، چه تصویری تولید کند. (لازم به ذکر است که این تصاویر داخل فایل نوت بوک ارسال چسبیده، یا در واقع در یک سطر قرار دارد. اما به علت نمایش بهتر در گزارش، این تصاویر ۴ تا ۴ تا از یکدیگر جدا شده و در دو سطر نمایش داده شده‌اند)

VQ-VAE .۴-۱

نوعی خاص از VAE است که از متغیرهای نهان گسسته (Discrete Latent Variables) استفاده می‌کند. در ابتدا به بررسی ساختار و تفاوت‌های اصلی آن با VAE معمولی می‌پردازیم.

- فضای نهان گسسته:

در VQ-VAE، فضای نهان به صورت گسسته تعریف می‌شود. این فضا شامل K بردار نهان (در اینجا ما از 512 بردار استفاده کردیم) است که هر بردار ابعاد D (در مدل ما طول هر کدام از این بردارها برابر 64 است) دارد. به جای تولید توزیع احتمالی پیوسته برای فضای نهان، مدل یک توزیع دسته‌ای (Categorical Distribution) تولید می‌کند که هر نمونه را به نزدیکترین بردار نهان نگاشت می‌کند.

- Encoder (رمزگذار):

رمزگذار وظیفه دارد که داده‌های ورودی (مثلًا تصاویر) را به یک نمایش نهان (latent representation) با ابعاد پایین‌تر فشرده کند. این بخش شامل چندین لایه کانولوشنی (convolutional layers) است که ویژگی‌های داده‌های ورودی را استخراج کرده و به یک فضای برداری فشرده منتقل می‌کند. خروجی رمزگذار یک نگاشت پیوسته به فضای نهان است.

- Vector Quantizer (کمیت‌ساز برداری):

این بخش یکی از ویژگی‌های متمایز مدل VQ-VAE است. کمیت‌ساز برداری وظیفه دارد که نمایش پیوسته نهان خروجی رمزگذار را به نزدیکترین بردار در یک فضای برداری گسسته نگاشت کند. این فضای برداری گسسته شامل مجموعه‌ای از کدبوک‌ها (codebooks) است که هر کدام یک بردار گسسته را نشان می‌دهند.

مراحل کمیت‌ساز برداری:

1. محاسبه فاصله هر بردار نهان از رمزگذار به تمام بردارهای کدبوک.
2. انتخاب نزدیکترین بردار از کدبوک به هر بردار نهان.
3. نگاشت بردار نهان به نزدیکترین بردار کدبوک.

این فرآیند منجر به نمایشی گسسته و فشرده از داده‌های ورودی می‌شود.

• Decoder (رمزگشا):

رمزگشا وظیفه دارد که نمایش گسسته تولید شده توسط کمیت‌ساز برداری را به شکل اصلی داده‌های ورودی بازسازی کند. این بخش نیز شامل چندین لایه کانولوشن معکوس است که بردارهای گسسته را به تصاویر یا داده‌های ورودی بازسازی می‌کند.

در فرآیند آموزش این مدل، از ۳ نوع تابع هزینه یا loss function استفاده می‌شود.

• :Reconstruction Loss

تفاوت بین داده‌های ورودی و داده‌های بازسازی شده توسط مدل را اندازه‌گیری می‌کند. معمولاً از خطای mean squared error یا خطای binary cross-entropy استفاده می‌شود.

• :Quantization Loss

تفاوت بین نمایش نهان پیوسته و نمایش نهان کمیت‌شده را اندازه‌گیری می‌کند. هدف این تابع هزینه، کم کردن فاصله بین این دو نمایش است.

• :Commitment Loss

این تابع هزینه مدل را تشویق می‌کند که به فضای گسسته نگاشت کند و به حفظ ویژگی‌های اصلی داده‌های ورودی متعهد باشد.

تفاوت اصلی این دو مدل به این صورت است که:

• نوع متغیرهای نهان:

در VAE معمولی، متغیرهای نهان پیوسته هستند و از توزیع نرمال (Normal Distribution) استفاده می‌شود. در VQ-VAE، متغیرهای نهان گسسته هستند و از بردارهای نهان از پیش تعریف شده استفاده می‌شود.

• یادگیری:

در VAE معمولی، تابع هزینه شامل دو بخش اصلی است: هزینه بازسازی و انحراف-Kullback-Leibler (KL). در VQ-VAE، علاوه بر هزینه بازسازی، از یک الگوریتم کمینه‌سازی خطای Vector Quantization (Vector Quantization) برای یادگیری فضای تعبیه‌یافته استفاده می‌شود. همچنین یک هزینه تعهد (Commitment Loss) برای اطمینان از اینکه خروجی کدگذار به بردارهای نهان نزدیک باشد، اضافه می‌شود.

مدل VQ-VAE با استفاده از تکنیک کمیتسازی برداری، نمایش نهان پیوسته داده‌های ورودی را به بردارهای گسسته نگاشت می‌کند. این فرآیند باعث می‌شود که مدل بتواند داده‌های پیچیده مانند تصاویر را به شکلی فشرده و با استفاده از فضای گسسته نمایش دهد. این مدل علاوه بر بهبود دقیق بازسازی، می‌تواند برای کاربردهایی مانند تولید داده‌های جدید و بهبود مدل‌های تولیدی نیز استفاده شود. این مدل به خصوص در کاربردهایی که نیاز به متغیرهای نهان گسسته دارند، مانند فشرده‌سازی داده و بازسازی دقیق، مفید است.

حال به بررسی مدل تعریف شده برای آموزش می‌پردازیم. انکودر آن به این صورت است که:

Model: "encoder"		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[None, 64, 64, 3]	0
conv2d_4 (Conv2D)	(None, 32, 32, 8)	224
conv2d_5 (Conv2D)	(None, 16, 16, 16)	1168
conv2d_6 (Conv2D)	(None, 8, 8, 32)	4640
conv2d_7 (Conv2D)	(None, 4, 4, 64)	18496
conv2d_8 (Conv2D)	(None, 4, 4, 64)	4160
<hr/>		
Total params: 28688 (112.06 KB)		
Trainable params: 28688 (112.06 KB)		
Non-trainable params: 0 (0.00 Byte)		

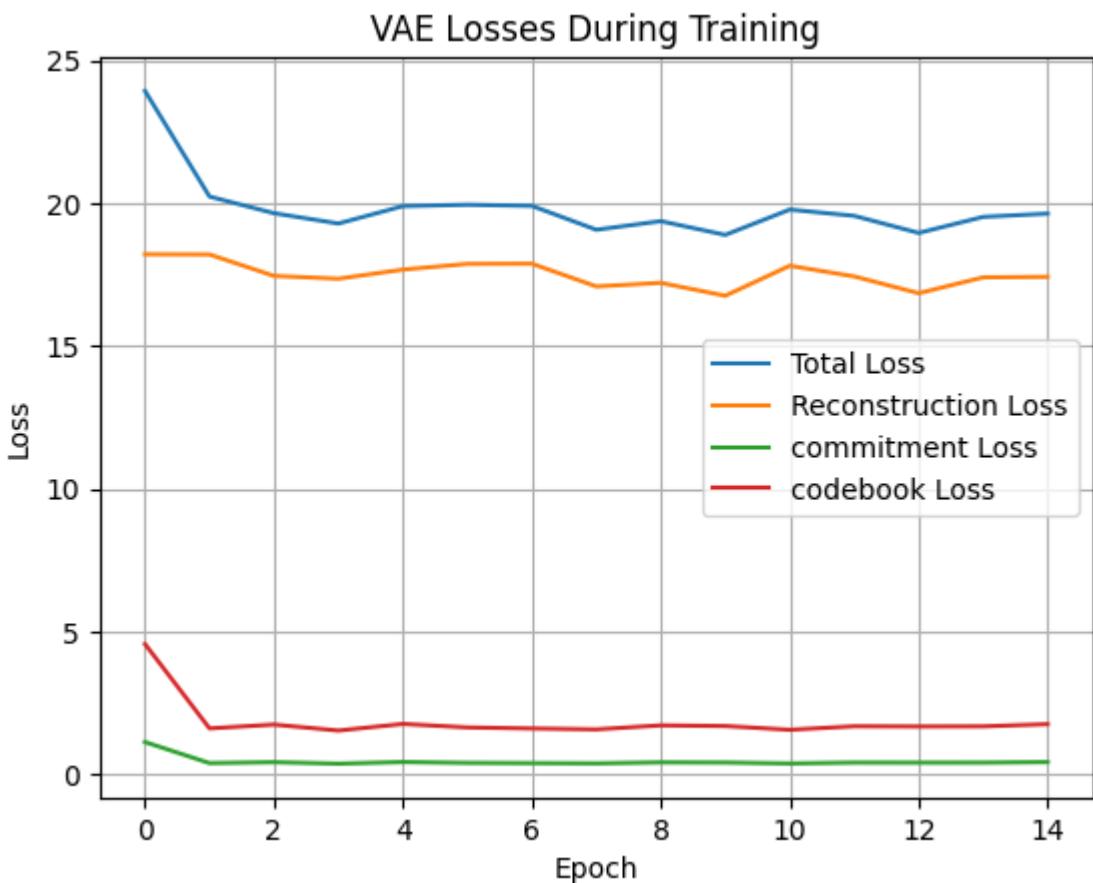
شکل 13. انکودر مربوط به VQ-VAE

و دیکودر آن به صورت زیر است.

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 4, 4, 64)]	0
conv2d_transpose (Conv2DTranspose)	(None, 8, 8, 64)	36928
conv2d_transpose_1 (Conv2DTranspose)	(None, 16, 16, 32)	18464
conv2d_transpose_2 (Conv2DTranspose)	(None, 32, 32, 16)	4624
conv2d_transpose_3 (Conv2DTranspose)	(None, 64, 64, 8)	1160
conv2d_transpose_4 (Conv2DTranspose)	(None, 64, 64, 3)	219
<hr/>		
Total params:	61395 (239.82 KB)	
Trainable params:	61395 (239.82 KB)	
Non-trainable params:	0 (0.00 Byte)	

شکل 14. دیکور VQ-VAE

حال به بررسی آموزش مدل می‌پردازیم. تابع هزینه مدل همانگونه و طبق تعریف آن پیاده‌سازی شده. از adam optimizer برای آموزش شبکه استفاده کردیم. مانند آموزش‌های قبلی از batch size = 32 که در قسمت Data Generator هندل شده، استفاده کردیم. مدل را ۱۵ ایپاک آموزش داده و نتایج آن را می‌توانیم در نمودار زیر ببینیم.



شکل ۱۵. هزینه آموزش VQ-VAE

همانطور که مشاهده می‌کنیم، هزینه در طول زمان (ایپاک) و فرآیند آموزش رفته کم شده و در نهایت به مقدار ۲۰ رسیده. بنابراین شبکه تا حد خوبی آموزش دیده است. بنابراین مدل احتمالاً نتایج خوبی خواهد داشت که در ادامه آن‌ها را بررسی می‌کنیم. همچنین لازم به ذکر است که در آموزش این مدل، از ۵۱۲ بردار امبدینگ که طول هر کدام ۶۴ است استفاده کرده و ایده VQ-VAE را پیاده سازی کردیم.

طبق خواسته سوال ابتدا تصاویر تصادفی از دیتاست انتخاب کرده و به مدل می‌دهیم. سپس خروجی مدل که بازسازی شده تصویر ورودی است را بررسی می‌کنیم.

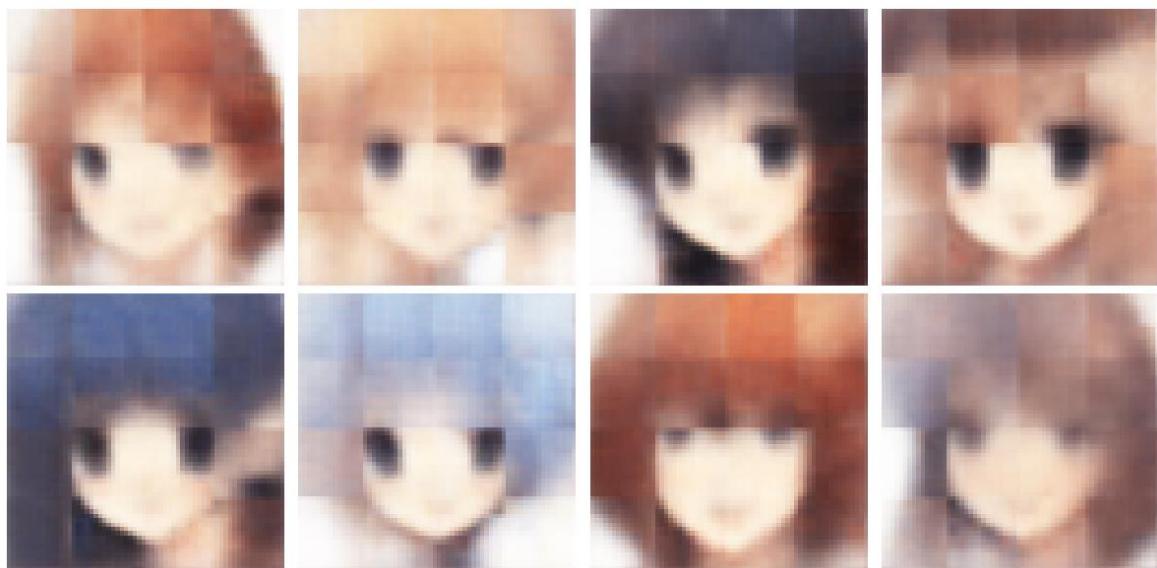
همچنین از ۸ تصویر تولید شده در قسمت ۱-۲ که در شکل ۶ نیز موجود هستند، استفاده کرده، آن‌ها را به مدل می‌دهیم و سپس کیفیت بازسازی آن‌ها در خروجی را بررسی می‌کنیم.

در ابتدا تصاویر تصادفی را ببینیم. در شکل زیر نسخه اصلی تصاویر مشخص است.



شکل 16. تصاویر اصلی داده شده به مدل

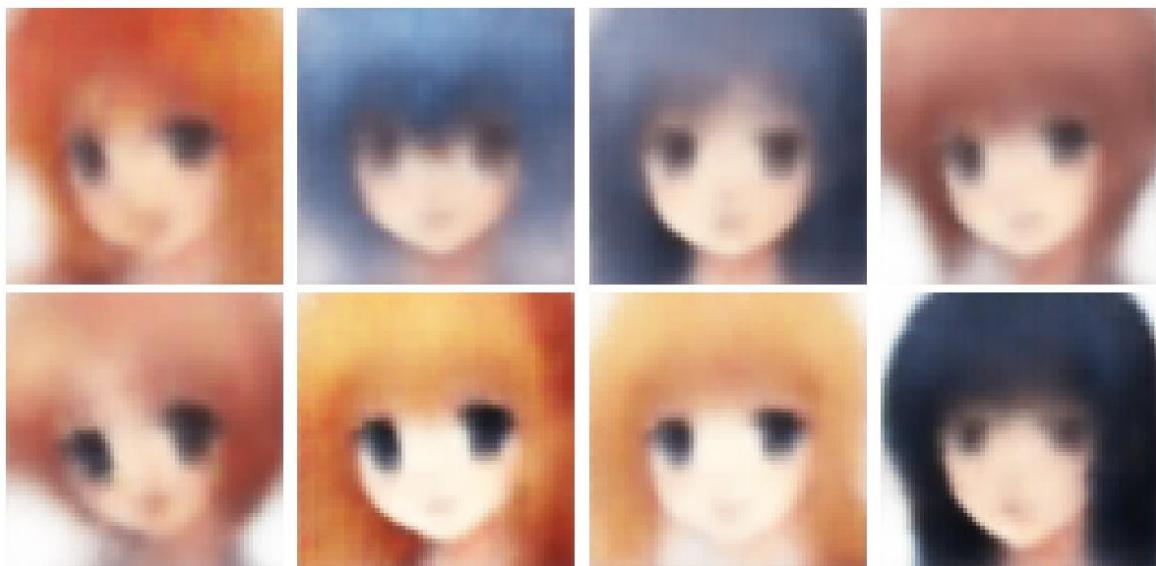
و در شکل زیر تصاویر بازسازی شده را می‌توانیم ببینیم. (تصاویر به ترتیب هستند)



شکل 17. تصاویر بازسازی شده توسط مدل

همانگونه که مشخص است مدل تا حد نسبتاً خوب و قابل قبولی توانسته تا شکل‌های اولیه را بازسازی کند. البته کیفیت بد تصویر خروجی، به خاطر کیفیت بد ورودی (64×64) است که ناچاراً به علت محدودیت‌های سخت افزاری مجبور به استفاده از این سایز شدیم. اما با وجود این، مدل توانسته ویژگی‌های مهم تصویر را دوباره در خروجی نمایش بده و آن را بازتولید کند.

حال به بررسی تصاویر تولید شده در بخش ۱-۲ می‌پردازیم. تصاویر اصلی برای مقایسه بهتر دوباره در شکل زیر قابل مشاهده هستند.



شکل ۱۸. تصاویر تولیدی از مدل **VAE**

خروجی مدل برای هر کدام از تصاویر به صورت زیر خواهد بود.



شکل ۱۹. خروجی تصاویر ساختگی

همانطور که قابل مشاهده است، برای این تصاویر، خروجی بازسازی شده بسیار نزدیک‌تر به تصویر اصلی و واقعی است. این اتفاق دلیل در ساده‌تر بودن تصویر ورودی دارد. مدل تقریباً توانسته بازسازی نسبتاً خوبی از همه تصاویر داشته باشد.

VQ-VAE-2 .۵-۱

یک مدل پیشرفته است که برای رفع برخی محدودیت‌های مدل اولیه Vector VQ-VAE-2 توسعه یافته و قابلیت‌های آن را، به ویژه در تولید تصاویر با کیفیت بالا، افزایش می‌دهد. در ادامه توضیحاتی در مورد این مدل می‌دهیم.

VQ-VAE-2 مدل اولیه VQ-VAE را با یک ساختار سلسله مراتبی گسترش می‌دهد تا بتواند سطوح مختلفی از جزئیات تصویر را مدل کند. این مدل برای جدا کردن ساختار کلی و جزئیات محلی تصویر به سطوح مختلفی از متغیرهای پنهان یا همان latent variables استفاده می‌کند که به آن امکان می‌دهد تصاویر با کیفیت و جزئیات بیشتری تولید کند.

مشابه VQ-VAE-2، VQ-VAE از یک رمزگذار برای نگاشت تصاویر ورودی به نمایش‌های پنهان و از یک رمزگشا برای بازسازی تصاویر از این نمایش‌ها استفاده می‌کند. با این حال، VQ-VAE-2 از چندین رمزگذار و رمزگشا برای سطوح مختلف سلسله مراتب استفاده می‌کند.

VQ-VAE-2 از یک رویکرد چندمقیاسی استفاده می‌کند که تصاویر را به کدهای پنهان در وضوح‌های مختلف تبدیل می‌کند. کد پنهان سطح بالا ساختار کلی را می‌گیرد، در حالی که کدهای پنهان سطوح پایین‌تر جزئیات بیشتری را می‌گیرند.

هر نمایش پنهان توسط کمیتساز به نزدیک‌ترین ورودی کدبوک نگاشت می‌شود. این فرآیند نمایش‌های پنهان پیوسته را به کدهای گستته کاهش می‌دهد که به حفظ کیفیت تصاویر بازسازی شده کمک می‌کند.

VQ-VAE-2 از مدل‌های خودرگرسیو قوی مانند PixelCNN برای مدل‌سازی توزیع کدهای پنهان استفاده می‌کند. این مرحله برای تولید تصاویر واقع‌گرایانه از طریق نمونه‌گیری از توزیع یادگرفته شده در فضای پنهان بسیار مهم است.

فرآیند آموزش این مدل، به دو بخش تقسیم می‌شود.

مرحله اول: یادگیری کدهای پنهان سلسله مراتبی:

- رمزگذاری: تصویر ابتدا به یک کد پنهان سطح بالا توسط رمزگذار بالا نگاشت می‌شود. این کد پنهان سطح بالا سپس کمیتسازی می‌شود.
- رمزگذاری مشروط: کد پنهان سطح بالا برای شرطی کردن رمزگذاری تصویر به یک کد پنهان سطح پایین‌تر استفاده می‌شود که آن هم کمیتسازی می‌شود.
- رمزگشایی: رمزگشا تصویر را از کدهای پنهان کمیتسازی شده سطح بالا و پایین بازسازی می‌کند. هزینه بازسازی محاسبه شده و پارامترهای مدل به روزرسانی می‌شوند.

مرحله دوم: یادگیری توزیع کدهای پنهان:

- آموزش توزیع قبلی: پس از یادگیری کدهای پنهان سلسله مراتبی، مدل‌های قدرتمند PixelCNN برای مدل‌سازی توزیع کدهای پنهان آموزش داده می‌شوند. توزیع قبلی سطح بالا، توزیع کدهای پنهان سطح بالا را مدل می‌کند، در حالی که توزیع قبلی سطح پایین، توزیع کدهای پنهان سطح پایین را مشروط بر کدهای سطح بالا مدل می‌کند.
- نمونه‌گیری: در طول تولید، کدهای پنهان از این توزیع‌های قبلی نمونه‌گیری می‌شوند و رمزگشا تصویر را از این کدهای پنهان نمونه‌گیری شده بازسازی می‌کند.

مزایای این مدل عبارت‌اند از:

- ساختار سلسله مراتبی: با جدا کردن جزئیات کلی و محلی، VQ-VAE-2 می‌تواند تصاویری با انسجام بالا و جزئیات دقیق تولید کند. ساختار سلسله مراتبی به مدل اجازه می‌دهد تا بر روی سطوح مختلف جزئیات به طور جداگانه تمرکز کند.
- نمونه‌گیری کارآمد: استفاده از مدل‌های خودرگرسیو در فضای پنهان به جای فضای پیکسلی، فرآیند نمونه‌گیری را بسیار سریع‌تر و کارآمدتر می‌کند.
- تولید تصاویر با کیفیت بالا: VQ-VAE-2 تصاویری تولید می‌کند که از نظر کیفیت با تصاویری که توسط مدل‌های پیشرفته GAN تولید می‌شوند قابل مقایسه است، بدون اینکه مشکلاتی مانند فروپاشی حالت (mode collapse) را داشته باشد.
- تنوع و کیفیت: آموزش مبتنی بر احتمال تضمین می‌کند که VQ-VAE-2 همه حالت‌های توزیع داده‌ها را پوشش دهد، که منجر به نمونه‌های تصویر متعدد می‌شود. علاوه بر این، از نمونه‌گیری رد برای تعادل بین کیفیت و تنوع نمونه‌ها استفاده می‌شود.

تفاوت بین VQ-VAE و VQ-VAE-2 از این قبیل است:

:VAE .1

- از متغیرهای پنهان پیوسته با توزیع گاوی استفاده می‌کند.
- تمرکز بر بیشینه‌سازی احتمال داده‌ها دارد که می‌تواند منجر به بازسازی‌های تار شود.
- آسان‌تر برای آموزش ولی ممکن است تصاویر با کیفیت بسیار بالا تولید نکند.

:VQ-VAE .2

- از یک فضای پنهان کمیت‌سازی شده تک‌سطحی استفاده می‌کند.
- کیفیت بازسازی خوبی دارد اما ممکن است در تصاویر با وضوح بالا دچار مشکل شود.
- تمرکز بر بازسازی تصاویر از یک فضای پنهان فشرده.

:VQ-VAE-2 .3

- از یک فضای پنهان سلسله مراتبی با چندین سطح کمیت‌سازی استفاده می‌کند.
- از مدل‌های خودرگرسیو برای مدل‌سازی توزیع کدهای پنهان استفاده می‌کند.
- قادر به تولید تصاویر با وضوح بالا و کیفیت بالا با تعادل مناسب بین ساختار کلی و جزئیات محلی.
- از نظر محاسباتی در نمونه‌گیری کارآمدتر نسبت به مدل‌های مبتنی بر فضای پیکسلی.

VQ-VAE-2 یک پیشرفته قابل توجه و مدل بسیار عالی در زمینه مدل‌های مولد عمیق است که با رفع محدودیت‌های مدل‌های قبلی و ارائه چارچوبی برای تولید تصاویر با کیفیت بالا به طور کارآمد، بهبود یافته است. ساختار سلسله مراتبی آن، همراه با مدل‌های خودرگرسیو قدرتمند، به آن اجازه می‌دهد تا تصاویری با جزئیات دقیق و انسجام بالا تولید کند، که آن را به یک انتخاب قوی برای کاربردهای مختلف در تولید تصویر و فراتر از آن تبدیل می‌کند.

پرسش ۲. Image Translation

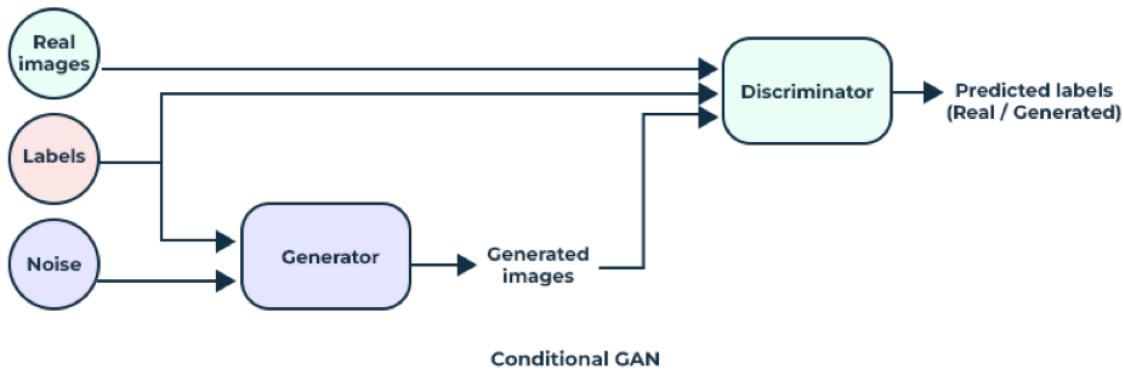
۱-۲. آشنایی با pix2Pix و معماری Image translation

۱. مقایسه مدل مقاله و GAN

در این مقاله از pix2pix استفاده شده که یک CGAN است. تفاوت این مدل، Conditional Generative Network با GAN معمولی در نوع ورودی‌هایی است که هر یک از generator و discriminator می‌گیرند. در GAN Generator آموزش می‌دید تا با استفاده از نویزهای تصادفی یک تصویر تصنی بسازد؛ فارغ از اینکه برای آن مشخص کنیم چه خروجی‌یی از آن انتظار داریم. اما در CGAN علاوه بر اینکه از Generator میخواهیم از روی نویزهای تصادفی برای ما تصویر تولید کند، به آن یک لیبل نیز ورودی می‌دهیم تا متناظر با آن برای ما تصویر را ایجاد کند. برای مثال، فرض کنید بخواهیم با داده‌های MNIST fasion کار کنیم. اگر یک GAN معمولی را برای این منظور آموزش دهیم، هنگامی که مثلا ۱۰۰ عدد تصویر را بخواهد برای ما تولید کند، خروجی‌ها توزیع مشخصی ندارند و ممکن از این میان تمام تصاویر تولیدی، هیچ یک، کفش نباشد و فقط تی‌شرت و شلوار را تولید کرده باشد. اما در CGAN generator ملزم می‌کنیم که برای ما تصویری براساس اطلاعات اضافی ورودی که به عنوان condition داده شده‌اند، تولید کند. مثلا edge map کفش را به آن می‌دهیم و طبق آموزش‌هایی که قبلا دیده، تضمین می‌شود برای ما یک تصویر کفش تولید خواهد کرد.

حال، در Pix2Pix دو تصویر، یکی به عنوان ورودی و یکی به عنوان خروجی به مدل می‌دهیم. generator در این مدل طوری طراحی شده که تصویر ورودی را به تصویر خروجی مپ (ترجمه) کند.

شکل زیر ساختار کلی CGAN را نمایش می‌دهد:



شکل 20. ساختار مدل CGAN

ساختار کلی Pix2Pix نیز به همین صورت است منتها بجای لیبل، یک تصویر را ورودی می‌دهیم.

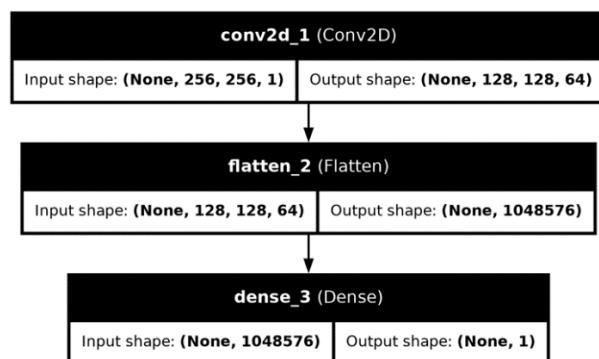
از جمله کاربردهای این مدل در تصویربرداری‌های پزشکی میتوان به موارد زیر اشاره کرد:

- تبدیل تصاویر CT به MRI : با این اطلاعات تکمیلی دربارهٔ شرایط بیمار در دسترس قرار می‌گیرد و به فرایند تشخیص و همینطور درمان کمک می‌کند.
- تبدیل تصاویر PET به CT: این کار منجر می‌شود مکان‌یابی آناتومیکی برای فعالیت‌های متابولیکی بهتر صورت بگیرد و به این صورت، در تشخیص‌های پزشکی موثر است.
- افزایش رزولوشن تصاویر پزشکی: تصاویر ممکن است دارای نویز باشند. مثلاً در شرایطی که بیمار حین تصویربرداری تکان خورده و یا تنظیمات زوم و رزولوشن دستگاه به درستی صورت نگرفته، تصویر دارای کیفیت پایین است و با استفاده از این شبکه میتوان به بهبود کیفیت آن کمک کرد تا در تشخیص‌ها عملکرد بهتری داشته باشیم. همچنین، می‌توان در شرایطی که دقیق خیلی بالایی مورد نیاز است، کیفیت تصاویری که در حالت کلی رزولوشن مطلوبی دارند را تا چند برابر بیشتر کرد.
- تشخیص آنومالی با عمل segmentation: از این معماری می‌توان برای segment کرد ارگان‌ها استفاده کرد و با استفاده از آن تومورها و آنومالی‌ها را در بیمار تشخیص داد.

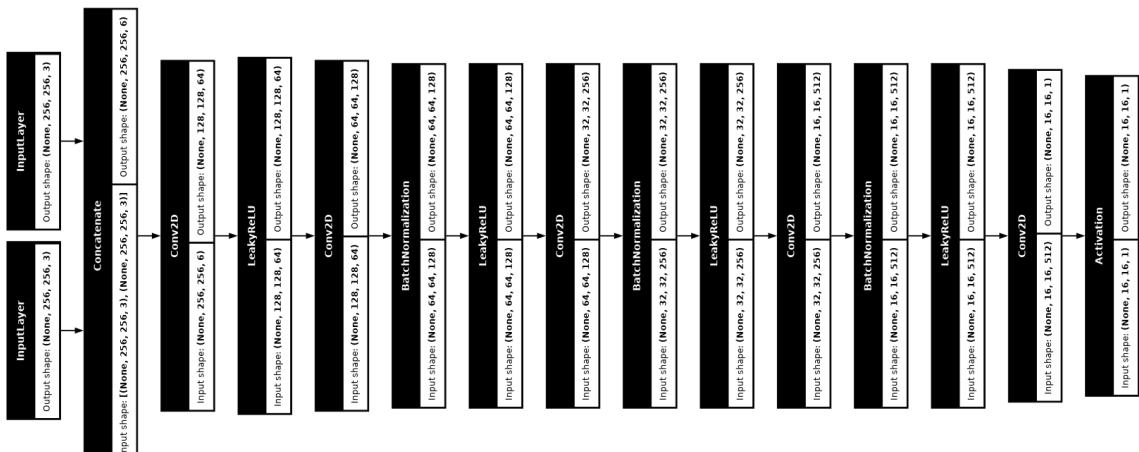
باز تولید تصاویر: این عمل به دو صورت امکان پذیر است، حالت اول زمانیست که به علت تکان خورد بیمار و یا وجود نویزهای خارجی، تصویربرداری به شکل کامل صورت نمی‌گیرد. در این صورت، از این معماری کمک می‌گیریم تا بخش‌های miss شده در تصویر را بازسازی کنیم. حالت دوم نیز زمانی است که مثلا برای تصویربرداری از بافت مورد نیاز، بیمار باید موارد رادیواکتیو زیادی مصرف کند اما برای جلوگیری از آسیب این موارد به بیمار، تصاویر سطحی تر و با دوز مصرفی رادیواکتیو کمتر گرفته می‌شوند. سپس با استفاده از این مدل، تصاویر کامل را از اسپارس بازسازی می‌کنیم.

۲. تفاوت cGAN و GAN در discriminator

در GAN و cGAN وظیفه‌ی تشخیص واقعی یا فیک بودن یک تصویر بر عهده‌ی discriminator است. ورودی discriminator در GAN، تنها یک تصویر است که در فرایند یادگیری، مدل آموزش می‌بیند که چطور با دقت بیشتری واقعی یا جعلی بودن آن را تشخیص دهد. اما در cGAN یک برچسب نیز به عنوان ورودی به discriminator داده می‌شود که طبق آن، روند آموزش را پیگیری می‌کند. در شکل‌های زیر، دو ساختار ساده از discriminator برای مدل‌های GAN و cGAN را مشاهده می‌کنید:



شکل 21. ساختار GAN ساده در یک discriminator



شکل 22. ساختار discriminator

همانطور که واضح است، در تصویر دوم دو ورودی برای مدل داریم.

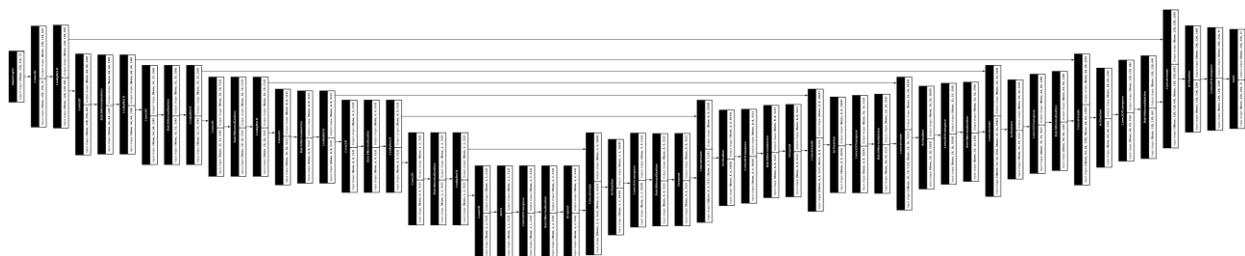
همچنین، در مدل ارائه شده در مقاله، ساختار patchGAN را به صورت cGAN کانولوشنی در نظر گرفته است که ساختار را در مقیاس patch های تصویر جریمه می کند. درواقع، این ساختار هر patch را بررسی می کند و مشخص می کند واقعی است یا فیک. این مدل، ساختار گفته شده را به صورت کانولوشنی در تمام تصویر پیاده می کند و میانگین تمام پاسخها را برای ارائه در خروجی نهایی آماده می کند. این در حالی است که در GAN تمام تصویر به صورت یکجا در نظر گرفته می شود نه اینکه روی patch های تصویر ارزیابی صورت گیرد.

موارد زیر از جمله دلایل برتری cGAN به GAN است:

- در cGAN کمتر روی خروجی های مشروط است. این در حالی است که در GAN داده ها میتوانند متنوع باشند و کنترلی روی ویژگی خاصی نداریم.
- برچسبی که به ازای هر تصویر در cGAN ورودی داده می شود، باعث کاهش ابهام شده و تصمیم گیری درباره میزان واقعی بودن تصویر را برای شبکه آسانتر می کند.

۳. ساختار *cGAN generator* در

شکل زیر ساختار مولد در مدل ارائه شده در مقاله را نشان می‌دهد. همانطور که میبینید، اندازه‌ی تصاویر ورودی را طبق مقاله 256×256 در نظر گرفته‌ایم.



شکل 23. ساختار مولد

شکل بالا صرفا نمایانگر شمای کلی مدل است و وضوح چندانی ندارد. تصویر واضح تر در نوت بوک ارائه شده است.

ساختار بخش downsampling به صورت زیر است:

- بلاک اول: شکل ورودی $(3, 256, 256)$ و شکل خروجی $(64, 64, 64)$
- بلاک دوم: شکل ورودی $(128, 128, 128)$ و شکل خروجی $(32, 32, 32)$
- بلاک سوم: شکل ورودی $(64, 64, 64)$ و شکل خروجی $(16, 16, 16)$
- بلاک چهارم: شکل ورودی $(32, 32, 32)$ و شکل خروجی $(8, 8, 8)$
- بلاک پنجم: شکل ورودی $(16, 16, 16)$ و شکل خروجی $(4, 4, 4)$
- بلاک ششم: شکل ورودی $(8, 8, 8)$ و شکل خروجی $(2, 2, 2)$
- بلاک هفتم: شکل ورودی $(4, 4, 4)$ و شکل خروجی $(1, 1, 1)$
- بلاک هشتم: شکل ورودی $(2, 2, 2)$ و شکل خروجی $(1, 1, 1)$

ساختار بخش upsampling به صورت زیر است:

- بلاک اول: شکل ورودی $(1, 1, 1, 512)$ و شکل خروجی $(2, 2, 512)$
- بلاک دوم(concatenate): شکل ورودی $[2, 2, 512], [2, 2, 512]$ و شکل خروجی $(4, 4, 512)$
- بلاک سوم: شکل ورودی $(2, 2, 2, 2, 1024)$ و شکل خروجی $(4, 4, 4, 512)$
- بلاک چهارم(concatenate): شکل ورودی $[4, 4, 4, 512], [4, 4, 4, 512]$ و شکل خروجی $(8, 8, 8, 512)$
- بلاک پنجم: شکل ورودی $(4, 4, 4, 4, 1024)$ و شکل خروجی $(1, 1, 1, 1024)$

- بلاک ششم(concatenate): شکل ورودی $[8, 8, 512]$, $(8, 8, 512)$ و شکل خروجی $(8, 8, 1024)$
- بلاک هفتم: شکل ورودی $(16, 16, 512)$, $(16, 16, 512)$ و شکل خروجی $(16, 16, 1024)$
- بلاک هشتم(concatenate): شکل ورودی $[(16, 16, 512), (16, 16, 512)]$ و شکل خروجی $(16, 16, 1024)$
- بلاک هفتم: شکل ورودی $(16, 16, 1024)$, $(8, 8, 512)$ و شکل خروجی $(16, 16, 512)$
- بلاک هشتم(concatenate): شکل ورودی $[(16, 16, 512), (16, 16, 512)]$ و شکل خروجی $(16, 16, 1024)$
- بلاک نهم: شکل ورودی $(16, 16, 1024)$, $(16, 16, 256)$ و شکل خروجی $(32, 32, 256)$
- بلاک دهم(concatenate): شکل ورودی $[(32, 32, 256), (32, 32, 256)]$ و شکل خروجی $(32, 32, 512)$
- بلاک یازدهم: شکل ورودی $(32, 32, 512)$, $(32, 32, 128)$ و شکل خروجی $(64, 64, 128)$
- بلاک دوازدهم(concatenate): شکل ورودی $[(64, 64, 128), (64, 64, 128)]$ و شکل خروجی $(64, 64, 256)$
- بلاک سیزدهم: شکل ورودی $(64, 64, 256)$, $(64, 64, 64)$ و شکل خروجی $(128, 128, 64)$
- بلاک چهاردهم(concatenate): شکل ورودی $[(128, 128, 64), (128, 128, 64)]$ و شکل خروجی $(128, 128, 128)$
- بلاک پانزدهم(conv2Dtranspose): شکل ورودی $(128, 128, 128)$, $(128, 128, 256)$ و شکل خروجی $(256, 256, 3)$

همانطور که میدانیم، این شبکه‌ی U-Net دارای دو بخش انکدر و دیکدر است که به ترتیب متناظر با *upsampling* و *downsampling* هستند.

در مورد نحوه‌ی کارکرد این مدل نیز میتوان گفت که عیناً مثل مدل U-Net است به این صورت که در گام‌های *downsampling* سایز 256×256 به 1×1 می‌رساند. سپس، در *upsampling* با استفاده از *deconvolution* سعی در بازیابی سایر اولیه‌ی تصاویر دارد. نکته‌ی قابل توجه در این بخش این است که *skip connection* های متناظر با *feature maps* در هر گام از *upsampling* به آنها *concatenate* می‌شود و این کار منجر به دو برابر شدن تعداد فیلترها در هر گام می‌شود.

استفاده از dropout در بخش upsampling این مدل منجر به افزایش قدرت تعمیم پذیری و robustness مدل می‌شود. همچنین، از overfitting جلوگیری می‌کند. از این رو، منجر می‌شود حتی اگر ورودی‌های یکسانی به مدل بدهیم، با توجه به اینکه به طور تصاویری نورون‌ها غیرفعال می‌شوند، خروجی‌های متنوعی خواهیم داشت. درواقع استفاده از Dropout به عنوان یک منبع نویز به مولد کمک می‌کند تا نتایج متنوع‌تری تولید کند.

۴. تابع هزینه

تابع هزینه‌ای که در این مقاله در نظر گرفته شده، ترکیبی از تابع هزینه‌ی اصلی در معماری cGAN و نرم L1 است که به صورت زیر تعریف می‌شود :

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

معادله ۱. تابع هزینه مورد استفاده در معماری مقاله

که در آن:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

معادله ۲ فاصله‌ی L1 مورد استفاده در تابع هزینه معماری مدل

۹

$$\begin{aligned} \mathcal{L}_{cGAN}(G, D) = & \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] , \end{aligned}$$

معادله ۳. تابع هزینه مورد استفاده در معماری اصلی شبکه cGAN

در تابع هزینه‌ی معادله‌ی ۳ G سعی در مینیمم کردن هزینه در مقابل discriminator با نام D که می‌خواهد آن را ماقزیم کند دارد. این هدف به این علت است که در روند آموزش باید برای بهبود عملکرد شبکه، مولد طوری آموزش ببیند که در هر مرحله تصاویری نزدیک‌تر به تصاویر واقعی تولید کند و discriminator نیز طوری بهبود بباید که تصاویر ساختگی نزدیک به واقعیت را از تصاویر واقعی تمیز دهد.

استفاده از نرم ۱ موجود در معادله‌ی ۱ در تابع هزینه نیز به این علت است که مولد باید علاوه بر اینکه discriminator را گول می‌زند، باید فاصله‌ی تصویر تولیدی‌اش تا برچسب ورودی را نیز به حداقل برساند و از آنجا که استفاده از نرم ۱ بجای نرم ۲ منجر به تاری کمتر می‌شود، از L1 استفاده می‌کنیم.

بنابراین، تابع هزینه نهایی در معماری pix2pix به صورت ترکیبی از تابع هزینه cGAN و تابع هزینه L1 است که هر دو برای بهبود کیفیت و تنوع نتایج تولید شده به کار می‌روند. discriminator به منظور تشخیص داده‌های واقعی از تولیدی آموزش می‌بیند و مولد تلاش می‌کند تا داده‌های واقعی تری تولید کند که discriminator نتواند آنها را تشخیص دهد.

برای بهروزرسانی وزن‌ها در هر یک از دو بخش، دو فرایند زیر تا زمان رسیدن به هدف شبکه، به طور متناوب انجام می‌شوند:

1. بروزرسانی وزن‌ها در بخش discriminator

- نمونه‌برداری از داده‌های واقعی: نمونه‌برداری یک دسته از داده‌های واقعی (x,y)
- تولید داده‌های جعلی: نمونه‌برداری از نویز z و استفاده از مولد برای تولید داده‌های جعلی $.G(x,z)$
- محاسبه تابع هزینه discriminator
- محاسبه گرادیان‌های تابع هزینه نسبت به وزن‌های discriminator
- بهروزرسانی وزن‌های discriminator: بهروزرسانی وزن‌های discriminator با استفاده از الگوریتم‌های بهینه‌سازی (adam)

2. بروزرسانی وزن‌ها در بخش مولد:

- تولید داده‌های جعلی: نمونه‌برداری از نویز z و استفاده از مولد برای تولید داده‌های جعلی $.G(x,z)$
- محاسبه تابع هزینه مولد
- محاسبه گرادیان‌های تابع هزینه نسبت به وزن‌های مولد
- بهروزرسانی وزن‌های مولد

۵. پژوهش‌های مرتبط با Pix2Pix

• Pix2PixHD: این معماری یک سری از محدودیت‌های pix2pix اصلی را برطرف می‌کند ضمن اینکه مهم‌ترین کاری که می‌کند، تولید خروجی با رزولوشن بیشتر است که برای بعضی موارد کاربرد بهتر و بیشتری دارد. در این مدل از طراحی چندمقیاسه هم برای مولد و هم برای discriminator استفاده می‌شود. این کار منجر می‌شود که تصاویری نزدیک‌تر به تصاویر واقعی تولید شود. بنابراین، بافت و جزئیات در تصاویر تولیدی توسط این مدل، بهتر از مدل اصلی pix2pix خواهد بود.

در این معماری، از feature mapping loss استفاده می‌شود که منجر به پایداری بیشتر فرایند آموزش می‌شود. نمایش‌های ویژگی میانی در discriminator را برای تصاویر واقعی و تولید شده مقایسه می‌کند، و مولد را تشویق می‌کند تا تصاویری تولید کند که از نظر ادراکی مشابه تصاویر واقعی هستند.

بنابراین، استفاده از feature mapping loss در این معماری منجر شده تا آموزش به شکل پایدارتر و با اطمینان بیشتری صورت بگیرد.

نکته‌ی دیگری که درباره‌ی این شبکه وجود دارد این است که از شرطی‌سازی instance-level بهره می‌گیرد که به مدل اجازه می‌دهد تا ترکیب تصویر در سطح شی را بهتر مدیریت کند. این به ویژه در کارهایی مانند تبدیل image translation به semantic segmentation مفید است؛ جایی که مدل نیاز به درک و تولید اشیاء متمایز در یک تصویر دارد.

درمورد معماری مولد در این شبکه نیز میتوان چنین گفت که مولد ساختاری پیچیده‌تر و زنجیروار دارد که تصاویر را از رزولوشن‌های کم به زیاد تولید می‌کند. Discriminator نیز عمیق‌تر از معماری اصلی است و این منجر می‌شود که robustness مدل افزایش پیدا کند و تصاویر با کیفیت بیشتر را از لحاظ واقعی یا جعلی بودن بهتر تشخیص دهد.

• CycleGAN: یکی از پیش‌نیازهای استفاده از مدل pix2pix دسترسی به دادگان جفت شده است طوری‌که به ازای هر ورودی باید یک برچسب نیز داشته باشیم. این مدل، این محدودیت‌ها را از بین می‌برد و نیازی به دادگان جفت شده ندارد. پس، باعث می‌شود مدل انعطاف‌پذیری بیشتری داشته باشد و به طور گسترده‌تری بتوان از آن استفاده کرد. این ویژگی cycleGAN منجر می‌شود در مواردی که دسترسی به دادگان جفت شده امکان پذیر نیست، بتوان از این معماری بهره برد.

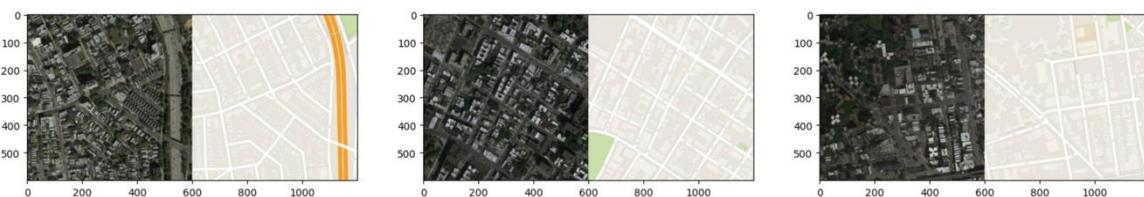
نکته‌ی دیگر درباره‌ی این معماریین است که از دو مولد و دو discriminator استفاده می‌کند. همچنین، برای اعمال سازگاری دو طرفه در ترجمه‌ها، cycle consistency loss را معرفی

می‌کند و تابع هزینه‌ی نهایی، ترکیبی از cycle consistency loss و adversarial loss می‌باشد. این در حالیست که در pix2pix از یک مولد و یک تشخیص دهنده استفاده می‌شود و تابع هزینه‌ی آن نیز ترکیبی از pixel-wise loss و adversarial loss بود.

۲-۲. پیاده سازی معما ری Pix2Pix

۱. خواندن مجموعه دادگان و نمایش چند تصویر تصادفی

شکل زیر، سه نمونه‌ی تصادفی از تصاویر در مجموعه دادگان آموزشی را نشان می‌دهد:



شکل 24. سه نمونه از تصاویر مجموعه آموزشی

همانطور که واضح است، تصاویر و لیبل‌ها به هم چسبیده هستند و نیاز به گام‌های پیش‌پردازش است تا بتوانیم تصاویر را به مدل بدهیم. بنابراین، این تصاویر را نصف کرده (از طول) و مجموعه دادگان آموزشی و اعتبارسنجی را به همین ترتیب می‌سازیم.

ابعاد تصاویر 1200×600 است. ابتدا هر تصویر را به 512×256 تغییر سایز می‌دهیم. سپس، از طول هر تصویر را تصف می‌کنیم تا تصاویر و برچسب‌ها از هم جدا شوند. نهایتاً، تصاویر تولید شده را نرمالیزه نیز می‌کنیم تا فرایند یادگیری بهتر انجام شود. از روی این تصاویر پردازش شده، مجموعه دادگان آموزشی و اعتبارسنجی را می‌سازیم تا در آینده از آنها استفاده کنیم.

۲. پیاده سازی generator و discriminator

تعداد پارامترها در generator و discriminator به شرح زیر است:

Total params	Trainable params	Non-trainable params
54429315	54419459	9856

جدول 3. تعداد پارامترها در مولد

Total params	Trainable params	Non-trainable params
2771393	2769601	1792

جدول 4. تعداد پارامترها در discriminator

پیاده سازی موارد خواسته شده در نوت بوک آورده شده است.

۲.آموزش مدل

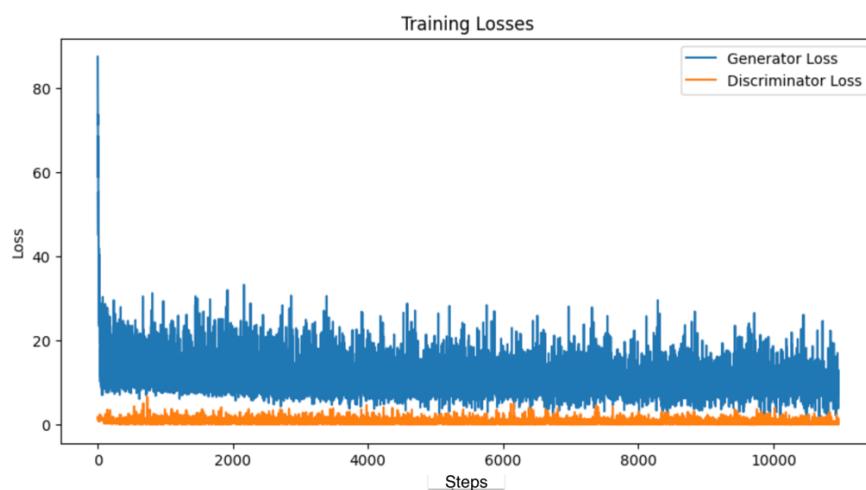
پیاده سازی تابع هزینه برای هر یک از مولد و تشخیص دهنده در نوت بوک آورده شده است.

مدل ساخته شده طی ۱۰ ایپاک و با هایپر پارامترهای زیر آموزش دیده است. در ادامه به بررسی نتایج عملکرد این مدل می پردازیم.

Epochs	Optimizer	Learning rate	Betha_1	Betha_2	Input shape	Batch size	Buffer size
10	Adam	0.0002	0.5	0.999	256*256	1	1096

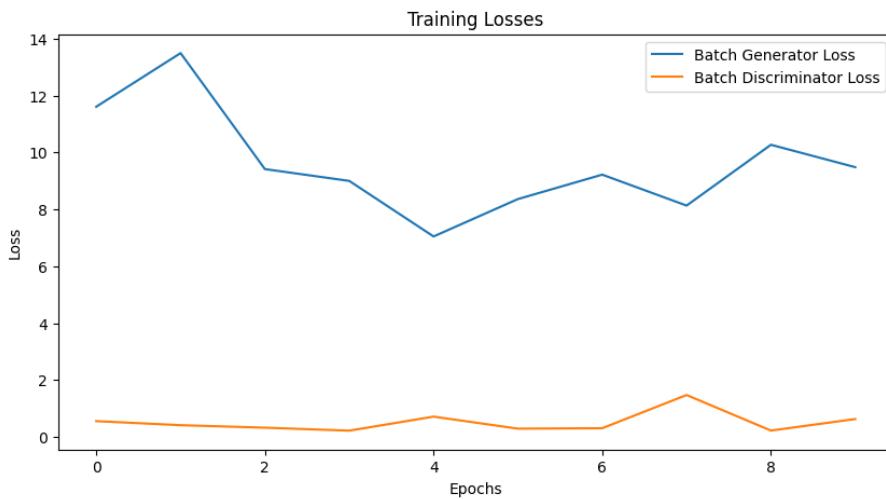
جدول ۵. هایپر پارامترهای مورد استفاده در آموزش مدل pix2pix

شکل زیر، هزینه را برای تمام استپ ها در طی فرایند یادگیری نمایش می دهد. از آنجا که در مجموع ۱۰۹۶ تصویر در دادگان آموزشی داشتیم و ۱۰ ایپاک را در نظر گرفته بودیم، تعداد کل استپ ها در این فرایند برابر ۱۰۹۶۰ عدد می باشد. همانطور که در شکل مشخص است، با شبکه کمی، در خالت کلی مقدار هزینه برای مولد کاهش یافته اما برای تشخیص دهنده ثابت است.



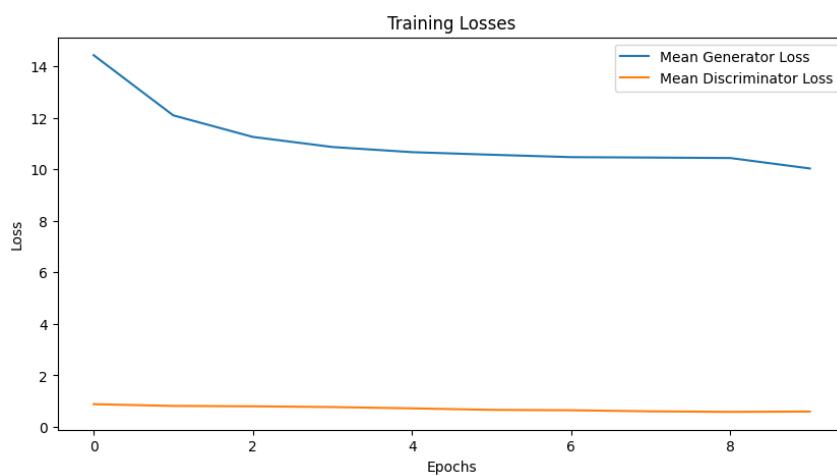
شکل 25. تابع هزینه بر حسب استپ در فرایند یادگیری

شکل زیر نیز، مقدار آخرین لاس محاسبه شده در هر ایپاک را نمایش میدهد. همانطور که در تحلیل نمودار بالا گفته شده، لاس مولد با وجود نوسانات در حالت کلی پس از ۱۰ ایپاک کاهش یافته در حالیکه لاس تشخیص دهنده تقریباً ثابت باقی مانده است.



شکل 26. تابع هزینه آخرین استپ در ایپاک در فرایند یادگیری

برای نمایش بهتر، میانگین هزینه‌ها در هر ایپاک را نیز محاسبه می‌کنیم که نمودار حاصل آن به صورت زیر خواهد بود:

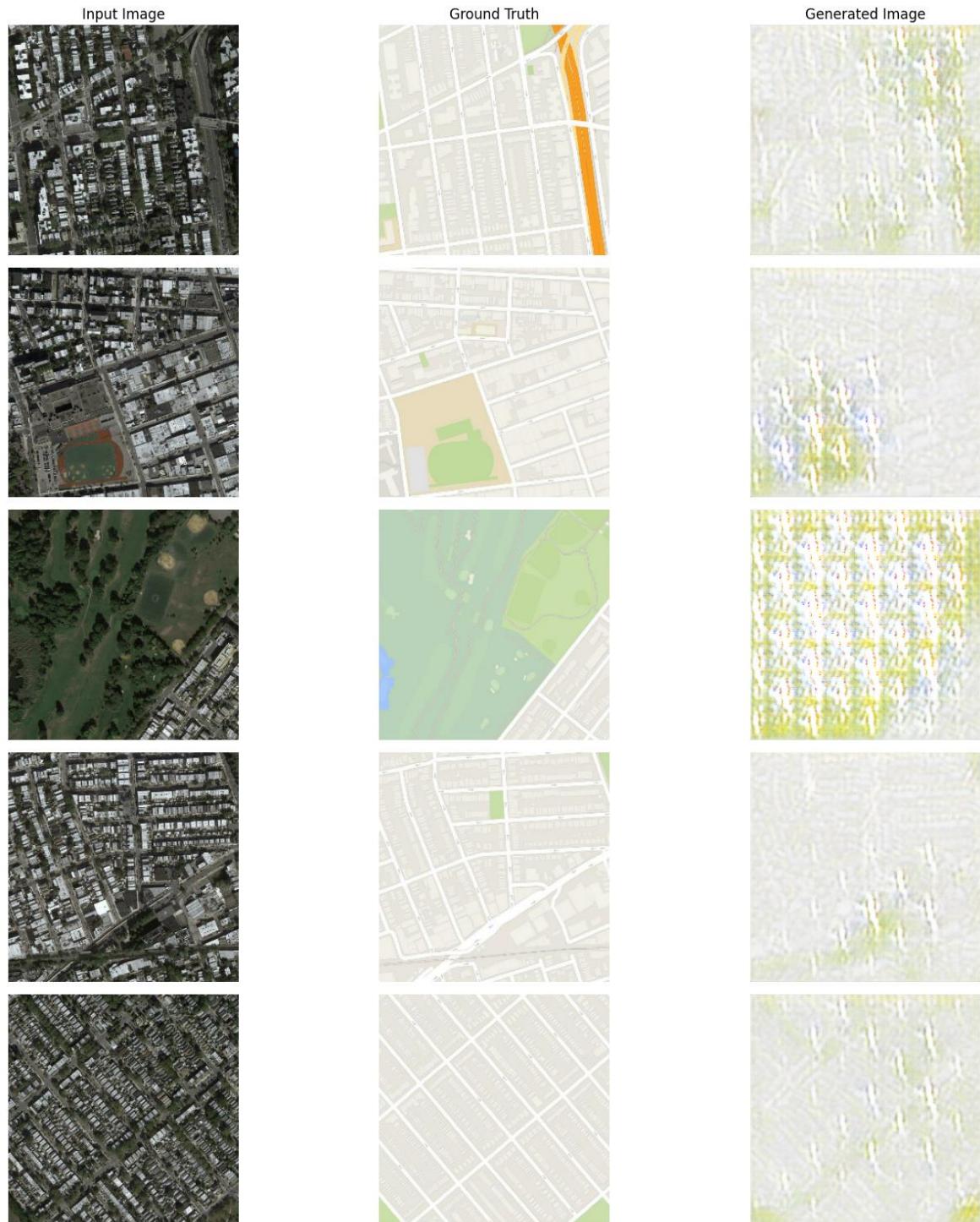


شکل 27. میانگین مقدار تابع هزینه در هر استپ بر حسب ایپاک در فرایند یادگیری

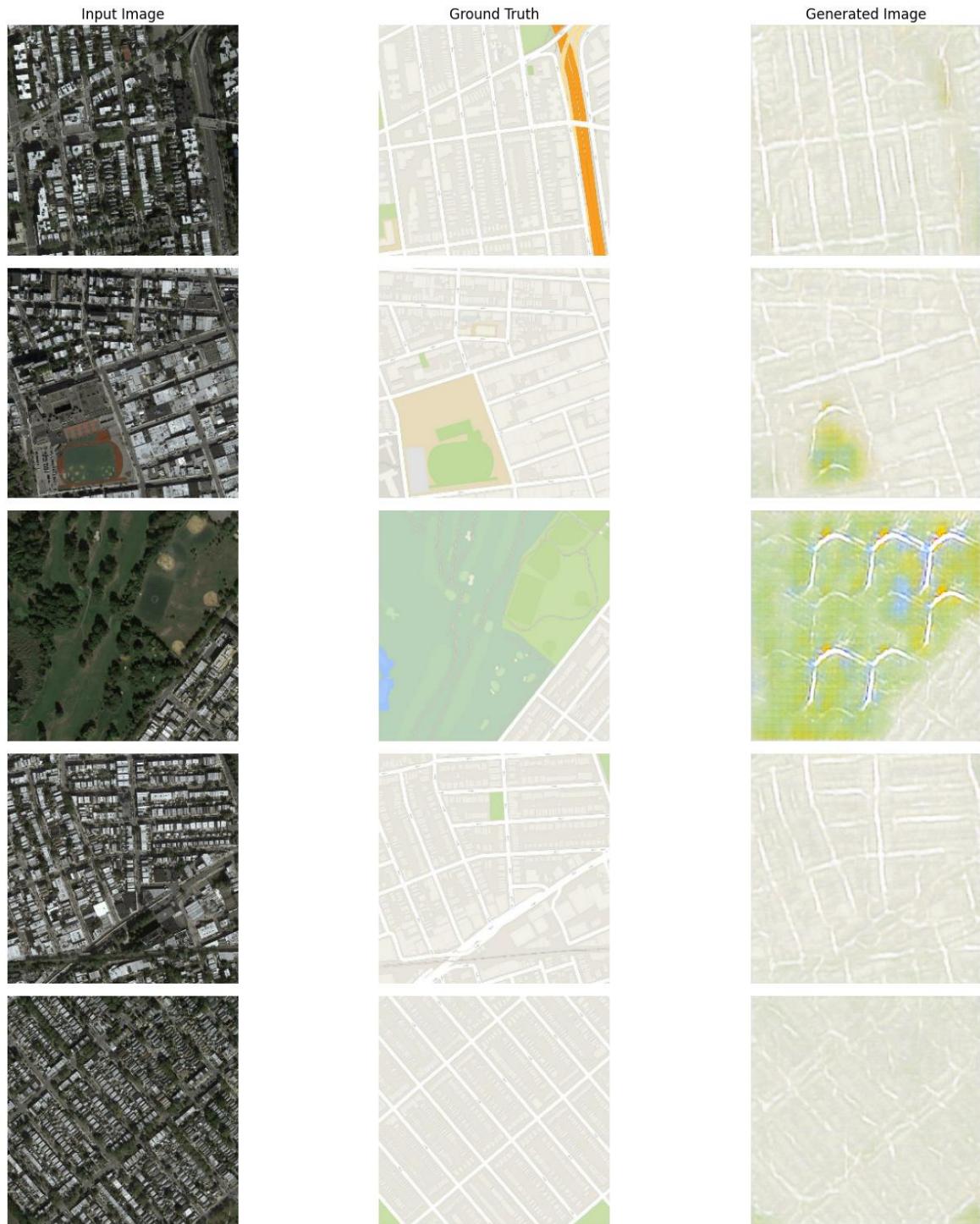
نتایج گفته شده در بالا در این نمودار مشخصا نمایان است؛ یعنی لاس مولد رو به کاهش و لاس تشخیص دهنده تقریبا ثابت و نزدیک صفر است که یعنی پیش‌بینی های خود را به درستی انجام داده است.

در فرایند یادگیری، میانگین لاس محاسبه شده در هر ایپاک، از حدود ۱۴ در ایپاک اول به حدود ۱۰ در ایپاک دهم رسیده که نشان می‌دهد شبکه مسیر خوبی را طی می‌کند و با افزایش تعداد ایپاک‌ها با توجه به شب نمودار، قطعاً شاهد عملکرد بهتری از مدل خواهیم بود و لاس مولد بیشتر کاهش خواهد یافت.

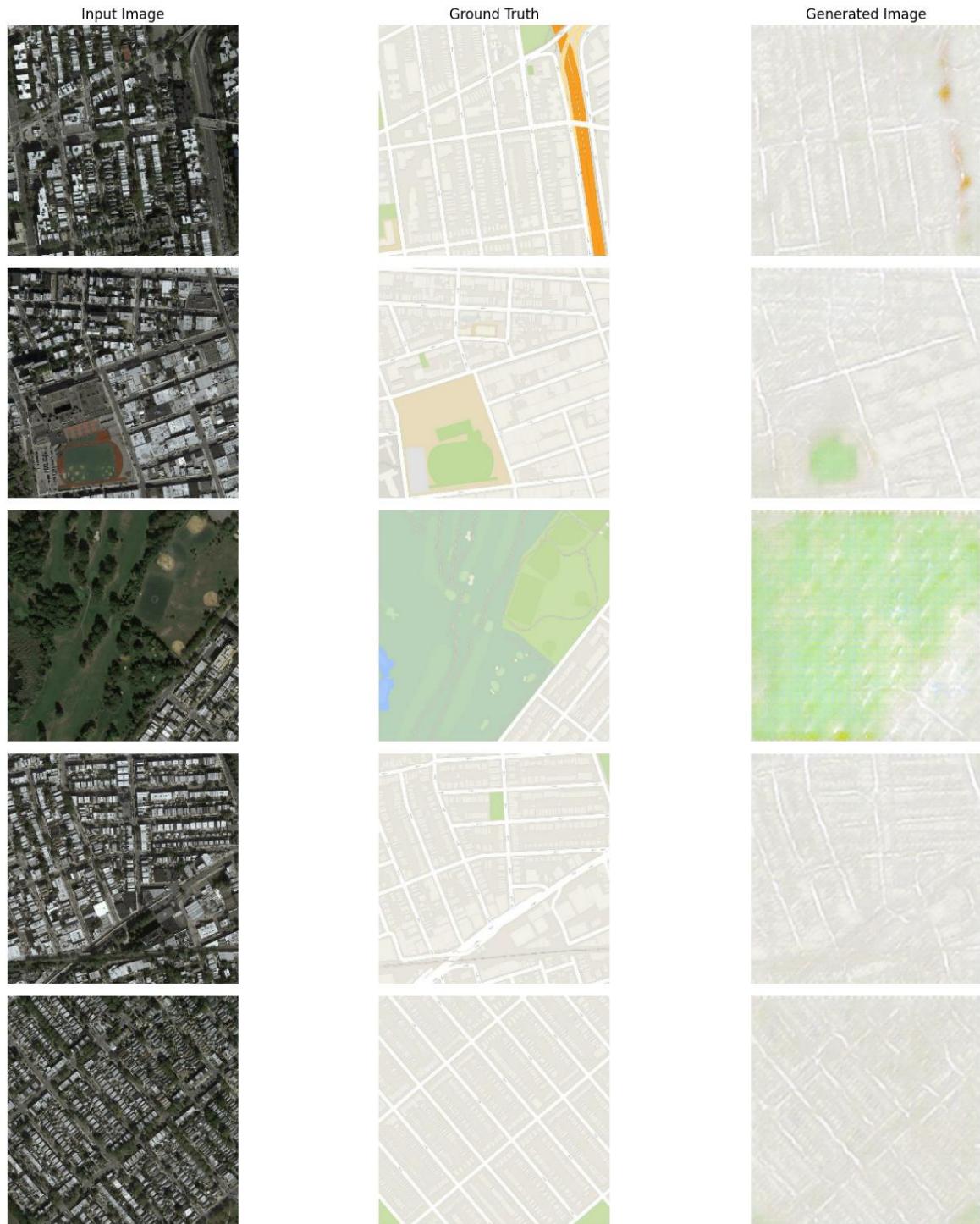
با بررسی ۵ تصویر تصادفی از مجموعه‌ی اعتبارسنجی در هر ایپاک نیز میتوان روند رو به بهبود عملکرد مدل را شاهد بود.



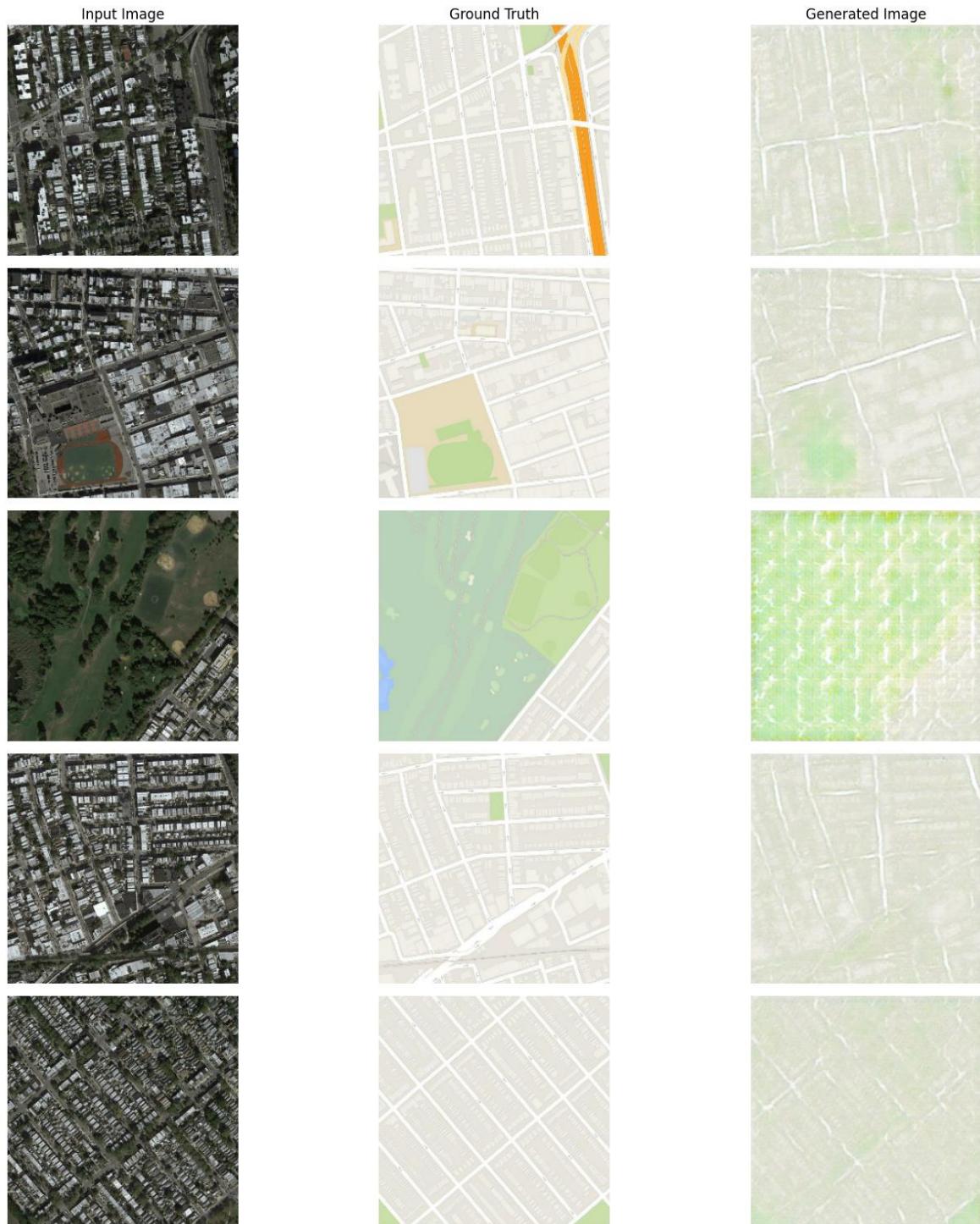
شکل 28. تعدادی از تصاویر مجموعه اعتبارسنجی پس از اولین ایپاک به همراه برچسب و تصویر تولید شده



شکل 29. تعدادی از تصاویر مجموعه اعتبارسنجی پس از دومین ایپاک به همراه برچسب و تصویر تولید شده



شکل 30. تعدادی از تصاویر مجموعه اعتبارسنجی پس از سومین ایپاک به همراه برچسب و تصویر تولید شده



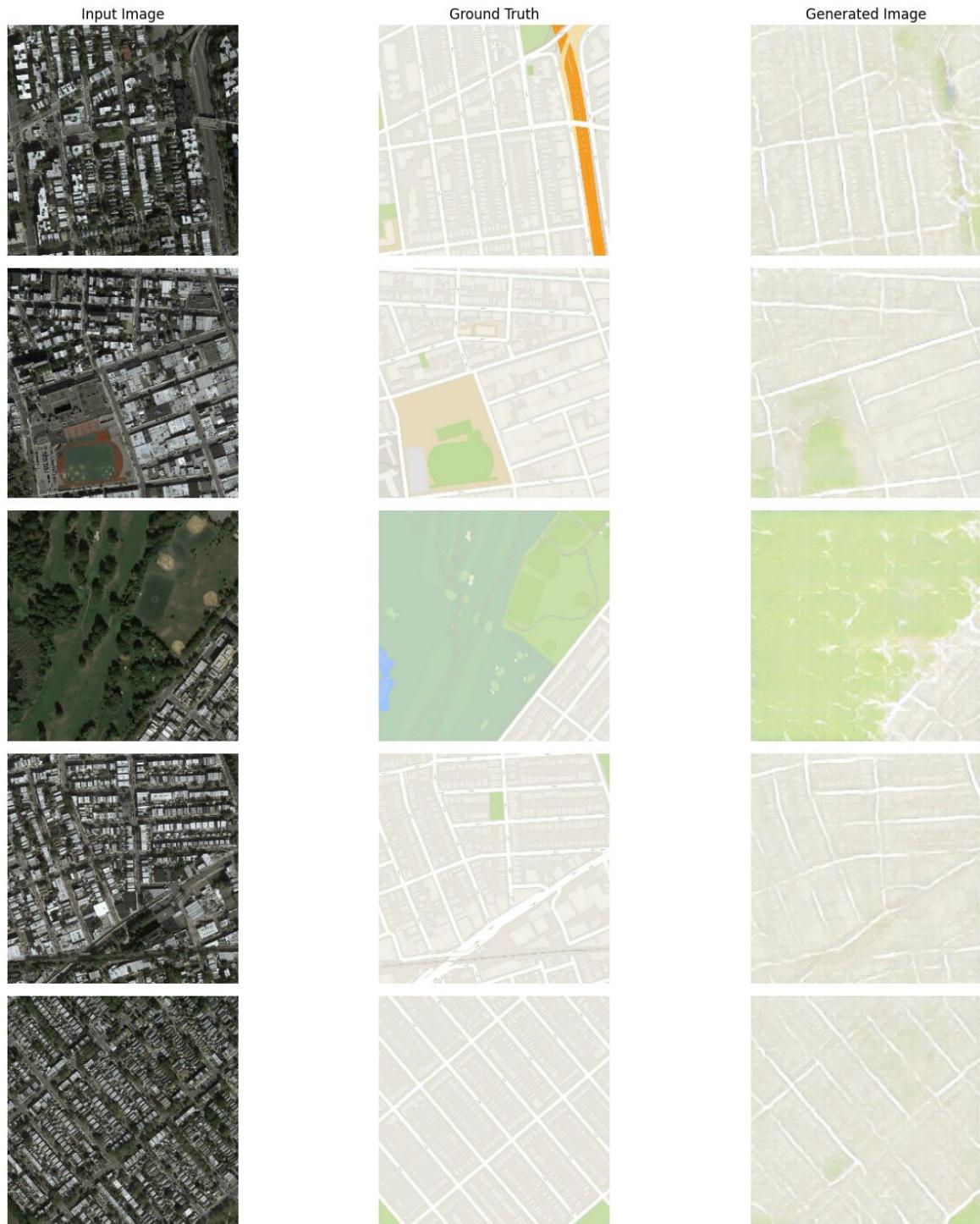
شکل 31. تعدادی از تصاویر مجموعه اعتبارسنجی پس از چهارمین ایپاک به همراه برچسب و تصویر تولید شده



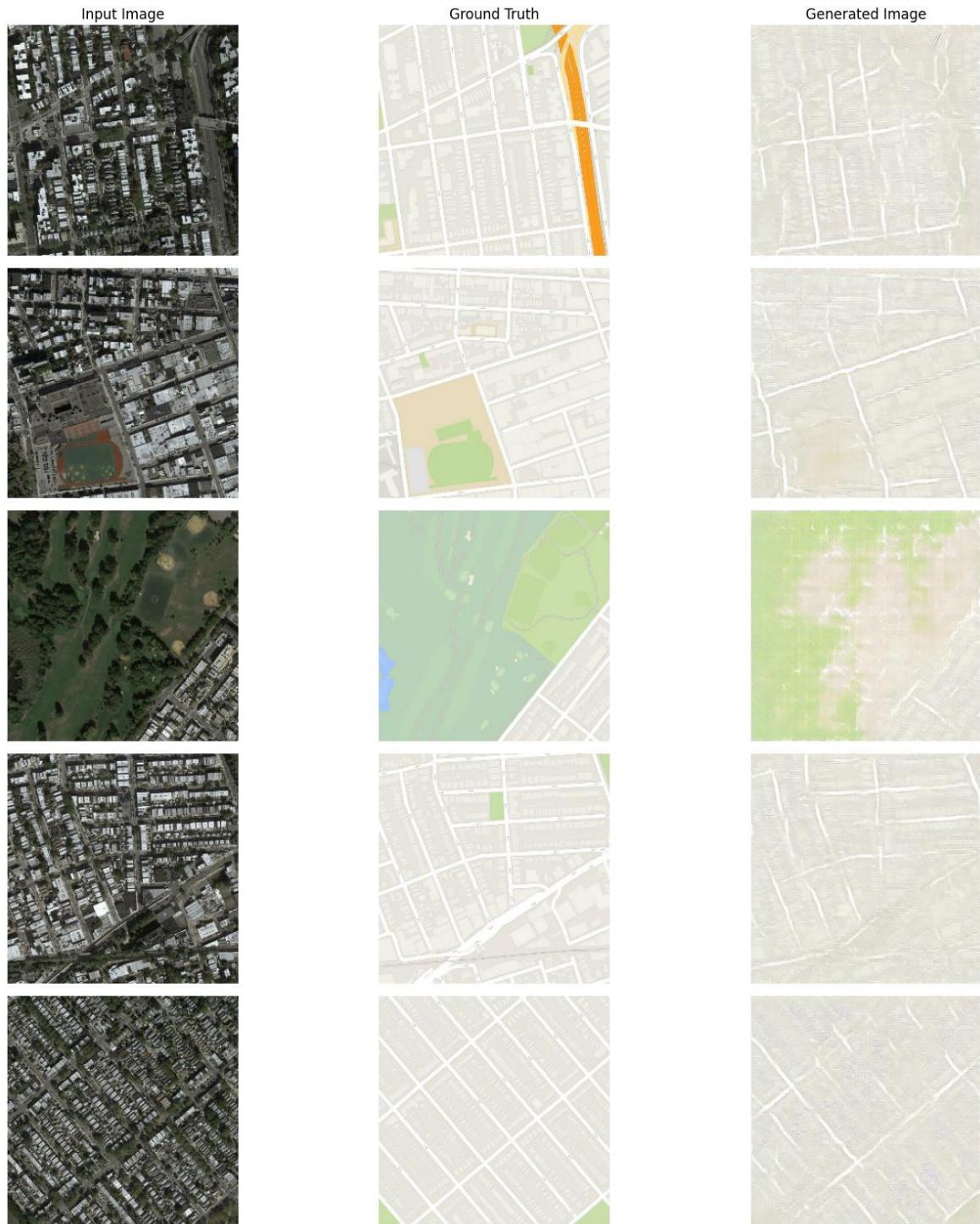
شکل 32. تعدادی از تصاویر مجموعه اعتبارسنجی پس از پنجمین ایپاک به همراه برچسب و تصویر تولید شده



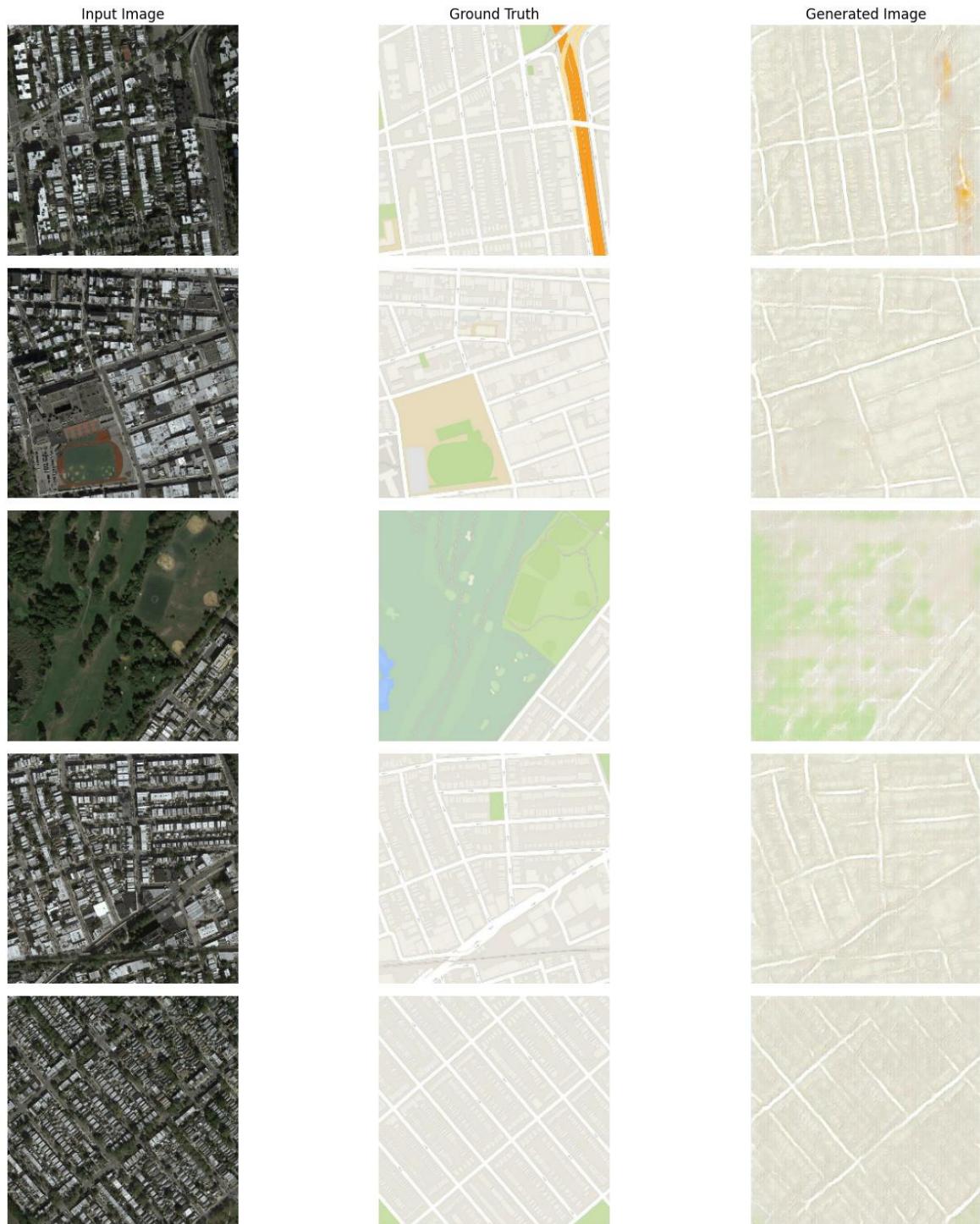
شکل 33. تعدادی از تصاویر مجموعه اعتبارسنجی پس از ششمین ایپاک به همراه برچسب و تصویر تولید شده



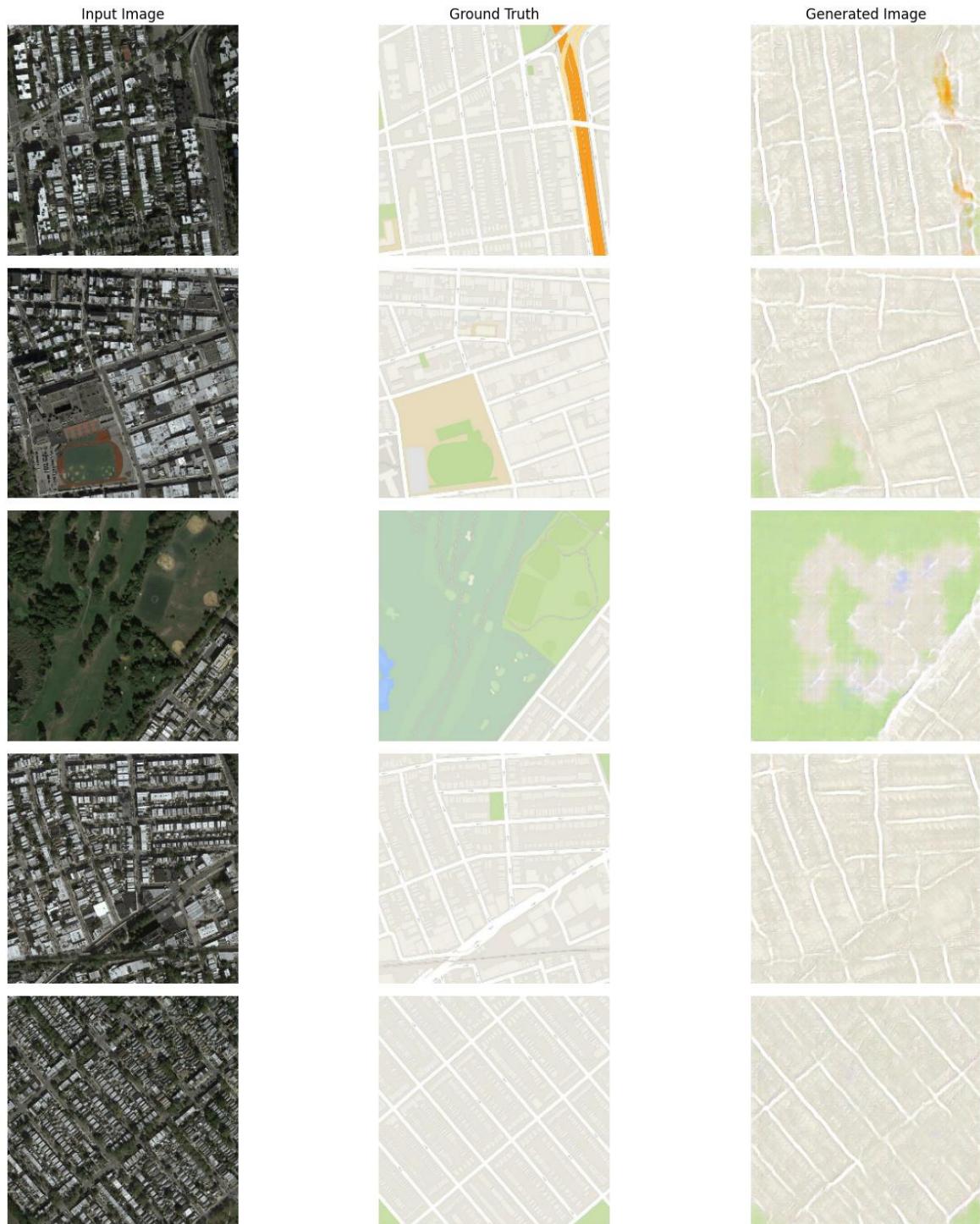
شکل 34. تعدادی از تصاویر مجموعه اعتبارسنجی پس از هفتمین ایپاک به همراه برچسب و تصویر تولید شده



شکل 35. تعدادی از تصاویر مجموعه اعتبارسنجی پس از هشتمین ایپاک به همراه برچسب و تصویر تولید شده



شکل 36. تعدادی از تصاویر مجموعه اعتبارسنجی پس از نهمین ایپاک به همراه برچسب و تصویر تولید شده



شکل 37. تعدادی از تصاویر مجموعه اعتبارسنجی پس از دهمین ایپاک به همراه برچسب و تصویر تولید شده

با مقایسه تصاویر متناظر در هر ایپاک در می‌یابیم که تصویر تولیدی رفته به برچسب ورودی نزدیک می‌شود و مدل به خوبی عمل می‌کند. صرفا با افزایش تعداد ایپاک‌ها می‌توان تصمین کرد که به نتایج بهتر و دقیق‌تری دست خواهیم یافت.

