



به نام خدا



دانشگاه تهران
دانشکده مهندسی برق و کامپیووتر
شبکه های عصبی و یادگیری عمیق

مینی پروژه شماره ۲۵

امیرمحمد رنجبر پازکی، زهرا موسوی موحد	نام و نام خانوادگی
۸۱۰۱۹۶۶۲۹، ۸۱۰۱۹۹۳۴۰	شماره دانشجویی
۱۴۰۰ /۵ /۳	تاریخ ارسال گزارش

فهرست گزارش سوالات

3.....	Variational Autoencoder – ۱
18.....	CycleGAN – ۲
25.....	سوال ۳ – آشنایی با مقالات مرتبط

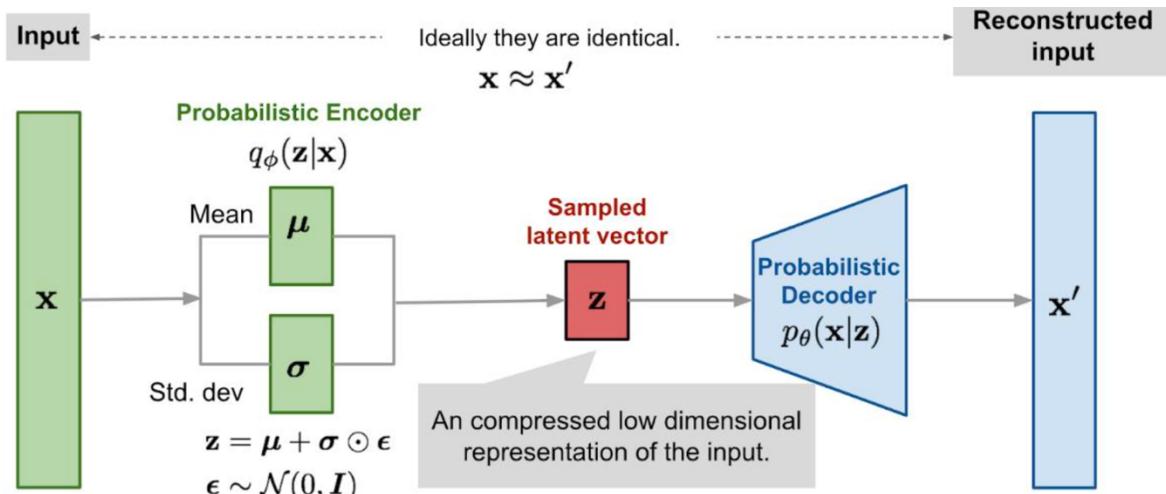
Variational Autoencoder – سوال ۱

الف) ساختار این شبکه برگرفته از Autoencoder است که دارای دو بخش encoding و decoding هستند. Autoencoder ها سعی بر ساخت یک نگاشت همانی داشتند. آنها در بخش decoding سعی داشتند با استفاده از کاهش لایه به لایه ابعاد به یک فضای کاسته شده بهینه (latent) و به دور از تکرار برسند. پس، در بخش decoding سعی می کردند تا با افزایش لایه به لایه ابعاد ورودی را بازسازی کنند. به این صورت مطمئن می شدند که در میانه راه ابعاد کاهش یافته تمام اطلاعات مورد نیاز را دارد.

حال می خواهیم از این شبکه ها برای تولید داده استفاده کنیم. مشکل اصلی آن است که داده ها در فضای latent به شکل دلخواه قرار گرفته اند. این پخش بودن دلخواه داده ها دو مشکل برای ما به وجود می آورد. اول آنکه با تغییرات اندک در فضای latent داده می تولید شده می تواند بسیار متفاوت باشد و به همین دلیل، این شبکه نسبت به نویز مقاوم نیست. دوم آنکه این دلخواه بودن توزیع اجازه اعمال کنترل بر روی خروجی را نمی دهد.

برای حل مشکل اول، نگاشت از یک فضا با ابعاد بالا به یک فضا با ابعاد پایین رخ نمی دهد بلکه نگاشت از یک فضا با ابعاد بالا به یک توزیع احتمالاتی (میانگین و واریانس) در فضای کاهش یافته انجام می شود. برای حل مشکل دوم نیز معماری conditional VAE معرفی شده است که در ادامه به آن پرداخته می شود.

معماری شبکه VAE در زیر (شکل ۱) قابل مشاهده است.



شکل ۱ – معماری Variational AutoEncoder

همانطور که در این ساختار از چپ به راست مشاهده می کنید، ورودی x وارد می شود. سپس، این ورودی به یک توزیع احتمالاتی در فضای ثانویه می رسد که با میانگین و انحراف معیار در دو جعبه سبز رنگ نشان داده شده است. حال از این دو جعبه نمونه برداری می شود و بردار latent میانی ساخته می شود که به دلیل نمونه برداری کمی متفاوت با x است ولی قالب کلی آن را دارد. حال فرآیند decoding انجام می شود و بردار x' به عنوان خروجی ساخته می شود. این بردار قالب کلی x را دارد با این تفاوت که بعضی ویژگی های عوض شده و داده می جدید تولید شده است.

دلیل این موضوع نمونه‌گیری پارامترهای مختلف است که باعث می‌شود که ویژگی از نمونه دیگری بر پیکر این نمونه بنشیند و نمونه‌ای جدید به وجود آورد. این کار با تغییر ریزی که در فضای latent رخ می‌دهد انجام می‌شود.

ساختار توضیح‌داده شده در کلاس VAE در کد پیاده‌سازی شده است که در notebook توضیحات کد داده شده است.

لازم به ذکر است که بعد فضای ثانویه ۲ در نظر گرفته شده است. یعنی ۲ مقدار برای میانگین و ۲ مقدار برای واریانس که هر زوج آن‌ها بیانگر بعدی از فضای latent می‌شوند.

ب) تابع هزینه‌ی استفاده شده در این سوال در تابع loss_function پیاده‌سازی شده است. در این سوال تابع هزینه مجموع Binary cross Entropy و KL divergence است. Binary cross Entropy نزدیکی پترن بازسازی شده با ورودی را می‌سنجد و مدل سعی دارد که این مقدار کمینه نگه دارد تا ذات ورودی حفظ شود و صرفاً ویژگی‌هایی به آن افزوده شود.

در قسمت KL divergence، سعی بر بیان این دارد که چقدر توزیع Q به خوبی توزیع P را تخمین زده است. این کار با استفاده کسر آنتروپی از Cross Entropy صورت می‌گیرد و بیانگر معیار آماری تفاوت دو توزیع در نظر گرفته می‌شود.

اگر X ورودی مدل باشد و Z متغیر latent باشد، $P(x)$ بیانگر توزیع داده و $p(z)$ بیانگر توزیع احتمالاتی متغیر latent است. $P(x|z)$ بیانگر توزیع تولید داده با داشتن متغیر latent است.

در variational autoencoder با استفاده $p(z|x)$ هستیم. $p(z|x)$ بیانگر توزیع احتمالاتی متغیر latent بر حسب ورودی است اما این توزیع را نداریم و به راحتی نیز نمی‌توانیم به دست بیاوریم. به همین دلیل، به سراغ تخمین $Q(z|x)$ می‌رویم. هدف ما آن است که این توزیع تا حد امکان به توزیع اصلی نزدیک باشد. این جایی است که لزوم استفاده از KL Divergence روش می‌شود. این معیار را باید کوچک کرد تا این توزیع به درستی به دست آید.

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

رابطه ۱ – رابطه KL Divergence در تابع هزینه

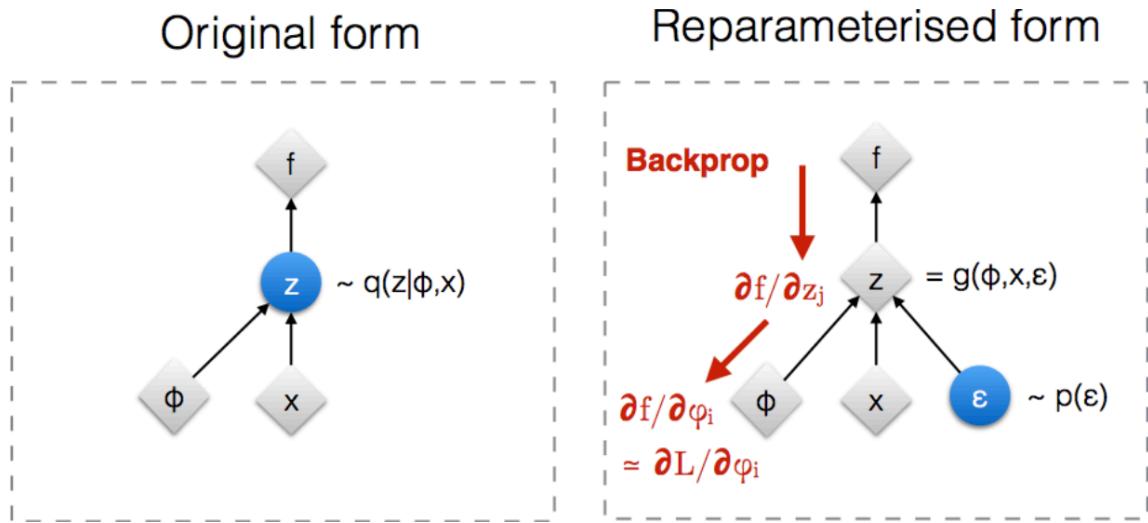
اگر این امید ریاضی را حساب کنیم، به عبارت زیر برای این بخش می‌رسیم که در کد مشاهده می‌شود.

$$\frac{1}{2} \left[-\sum_i (\log \sigma_i^2 + 1) + \sum_i \sigma_i^2 + \sum_i \mu_i^2 \right]$$

رابطه ۲ – رابطه نهایی KL Divergence در تابع هزینه

ج) این trick برای اعمال Back Propagation از یک گره تصادفی نیاز است. در VAE‌ها نمونه برداری از یک گره تصادفی رخ می‌دهد و به همین دلیل نمی‌توان از یک گره تصادفی عمل Back Propagate را انجام داد.

معرفی یک پارامتر ϵ اجازه می‌دهد تا z را یک reparametrize بکنیم و به این صورت عمل backpropagation از گره مشخصی انجام شود. در حقیقت، در اینجا منبع تصادفی بودن از z برداشته می‌شود و بر دوش ϵ گذاشته می‌شود. شکل زیر بیانگر حالت ساده و reparametrized شده را به طور کامل نشان می‌دهد. لازم به ذکر است که گره‌های آبی گره‌های تصادفی و گره‌های سفید گره‌های مشخص در گراف محاسباتی هستند.



شکل ۲ – حالت ساده و reparametrized شده گره محاسباتی

(۵) مجموعه داده‌ی MNIST به کمک کلاس VAE آموزش دیده شده تحت داده‌های این مجموعه به فضای ثانویه انتقال داده شده است که آن‌ها در شکل ۳ است.

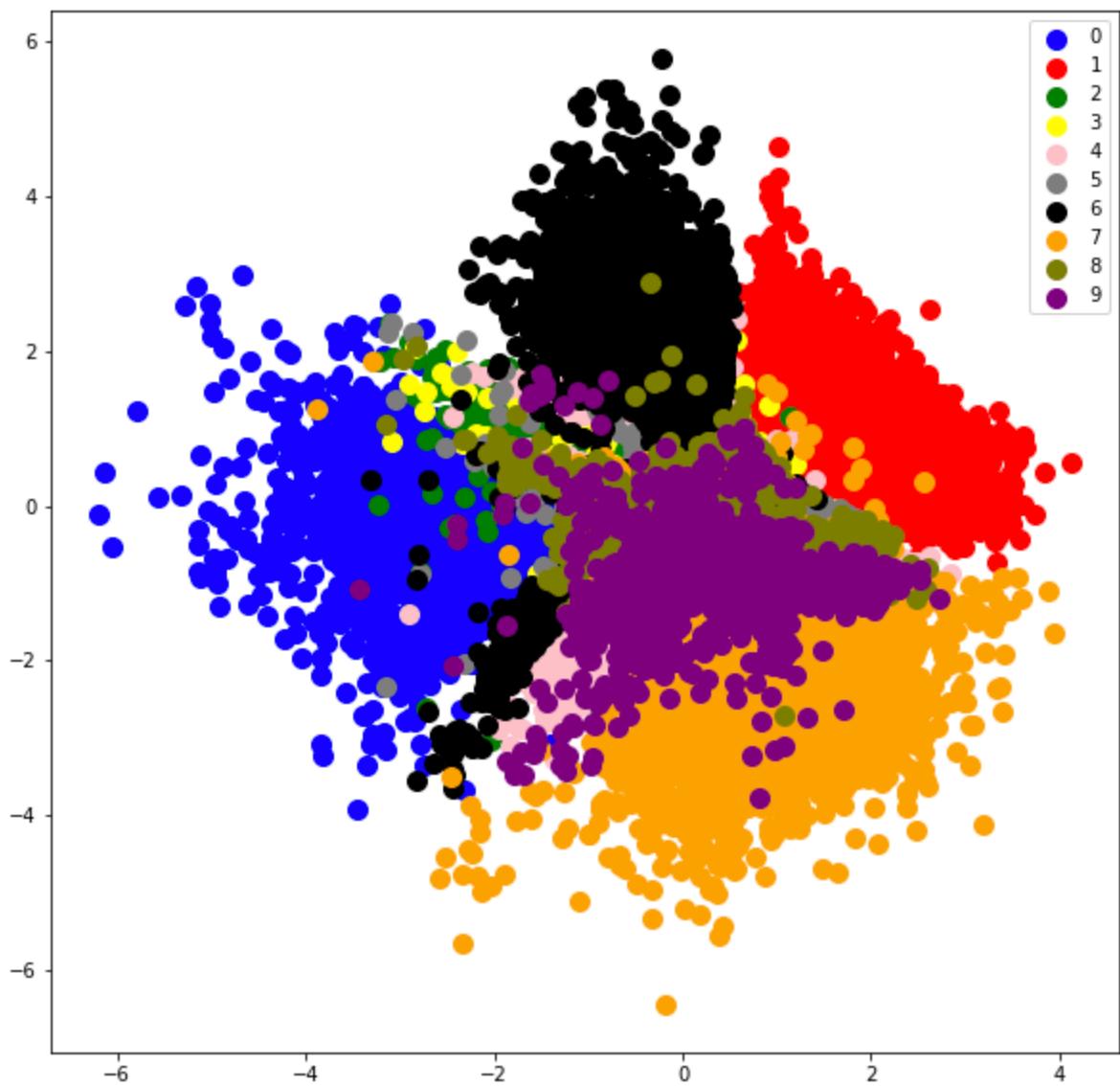
همانطور که در تصویر مشاهده می‌شود، داده‌های هر کلاس با رنگی مجزا نمایش داده شده‌اند. همچنین در گوشه تصویر رقم متعلق به هر رنگ نشان داده شده است.

همانطور که مشاهده می‌شود، داده‌های کلاس‌هایی از بقیه جدا هستند و دلیل این موضوع عدم شباهت یا شباهت کم آن‌ها در ویژگی‌ها به سایر کلاس‌های است. (مانند کلاس‌های صفر، یک و هفت)

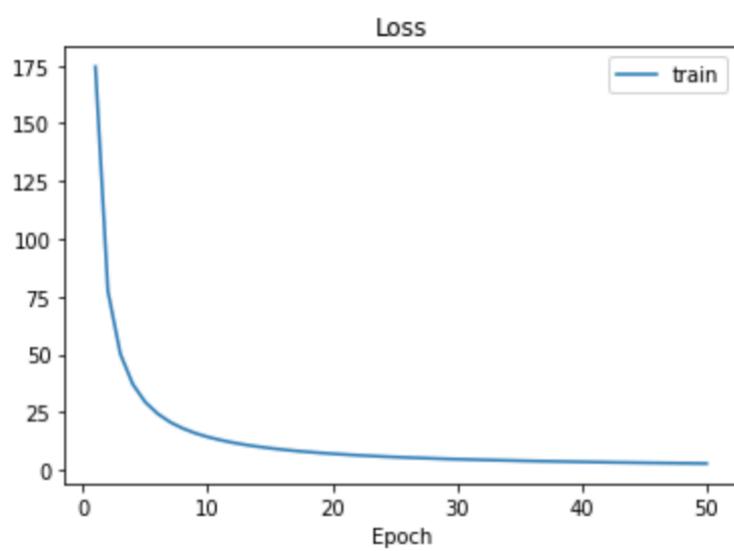
داده‌های چند کلاس نیز در میانه با یکدیگر تداخل دارند که دلیل این موضوع شباهت حدودی ویژگی‌های آن‌هاست.

لازم به ذکر است آموزش شبکه با موفقیت همراه بوده است چرا که خطای شبکه رفته کاهش یافته است که این نمودار در شکل ۴ قابل مشاهده است.

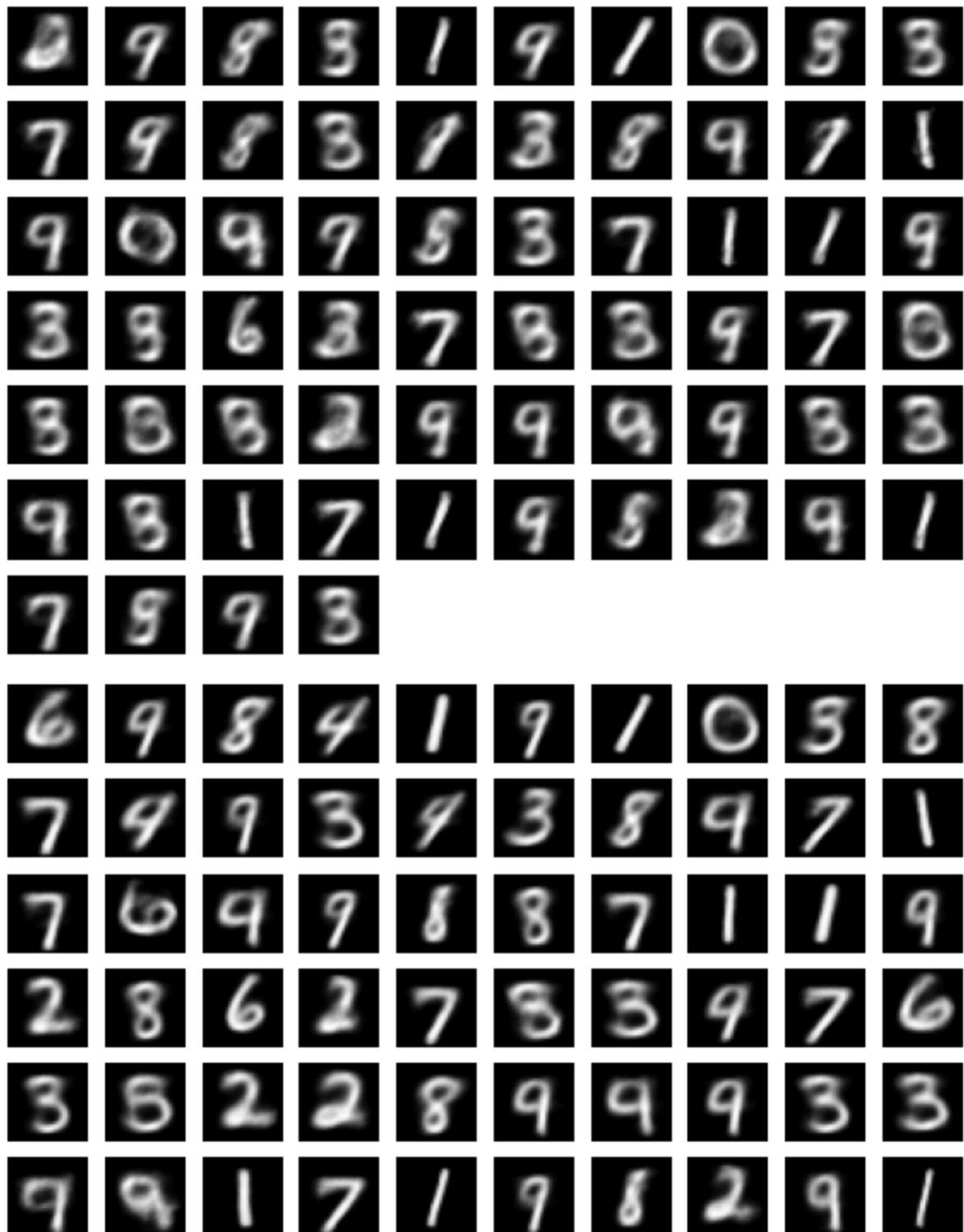
نمونه‌هایی نیز در پنج epoch مختلف نشان داده شده‌اند تا روند پیشرفت شبکه دیده شود. این روند نیز در مجموعه شکل ۵ قابل مشاهده است.

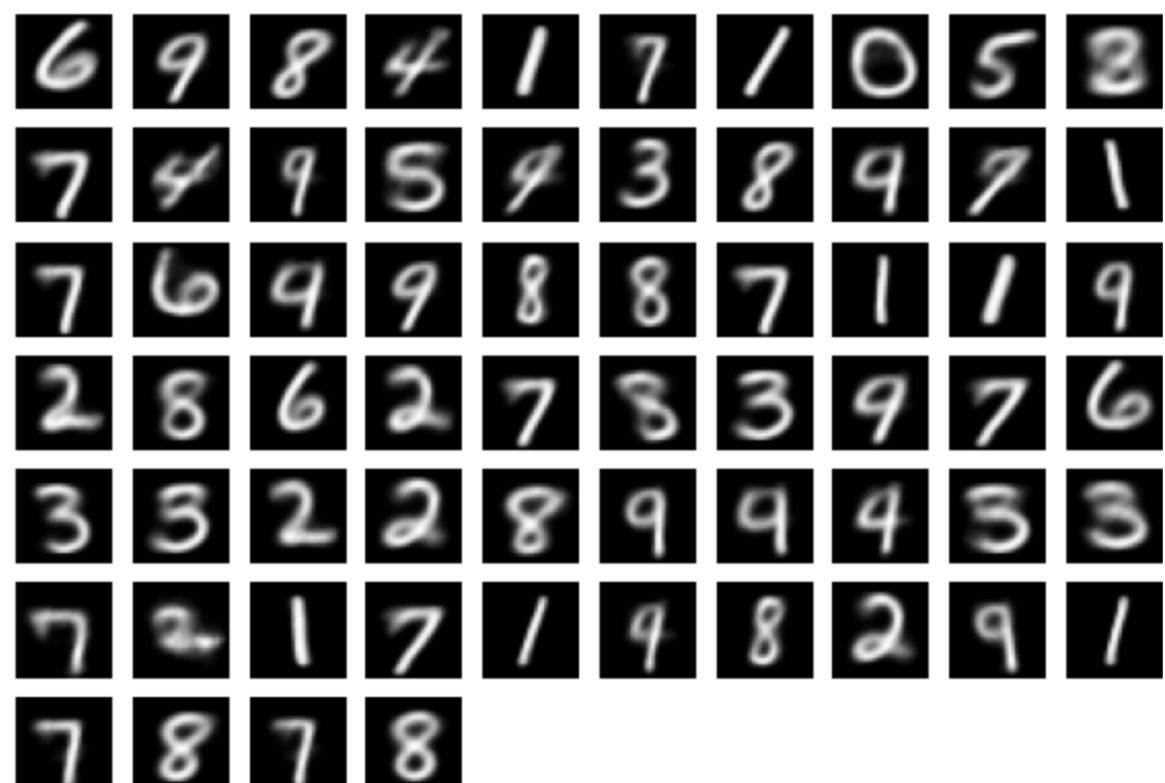
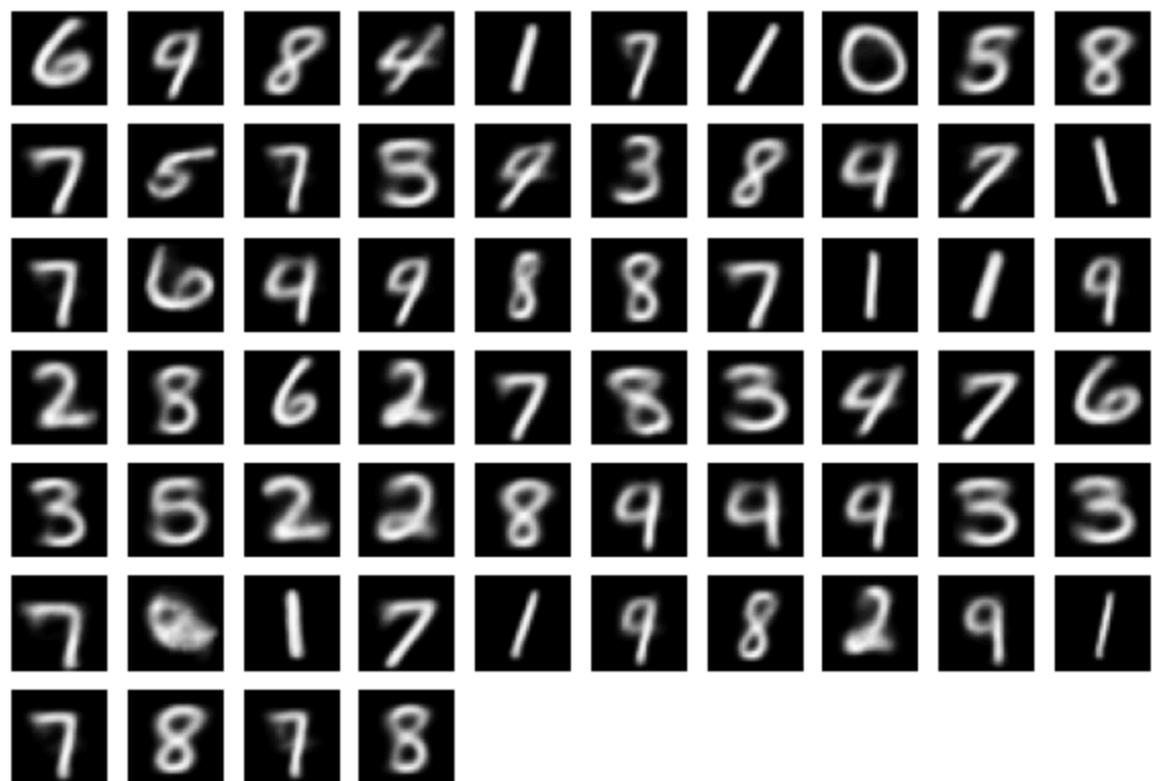


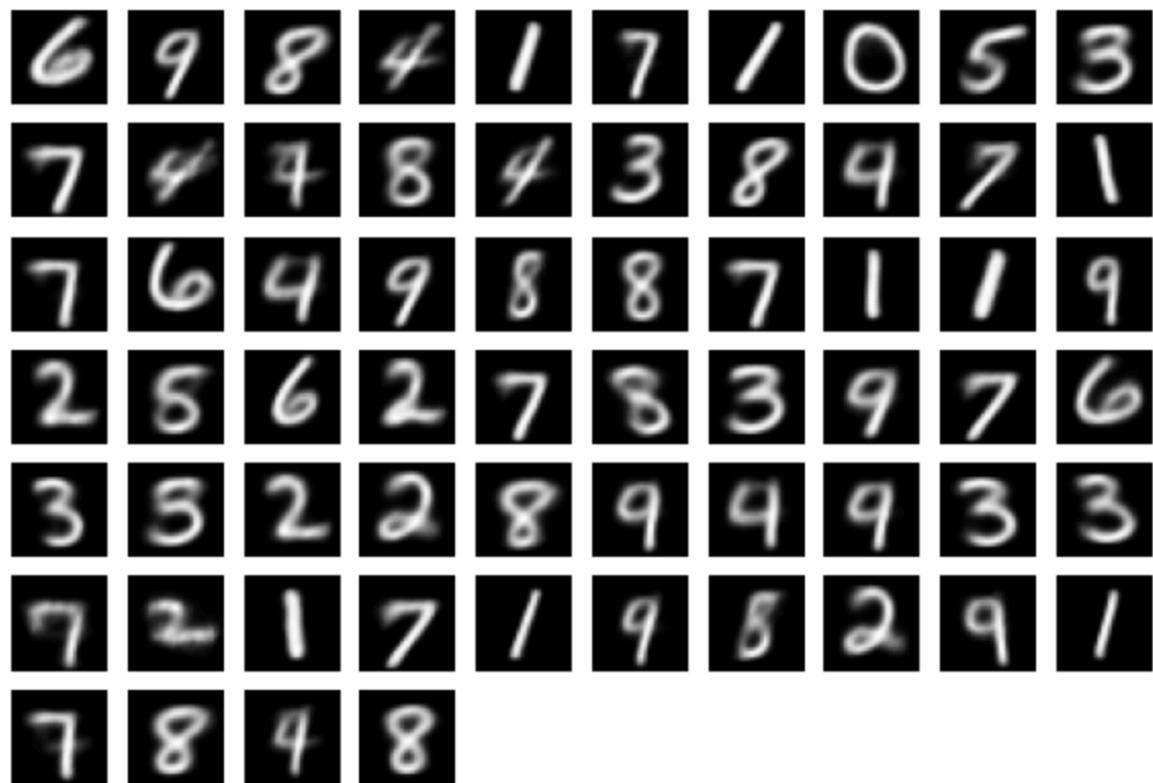
شکل ۳ – فضای latent به دست آمده از VAE برای داده‌ی MNIST



شکل ۴- نمودار خطای شبکه‌ی VAE بر روی داده‌ی آموزش در epoch ۵۰

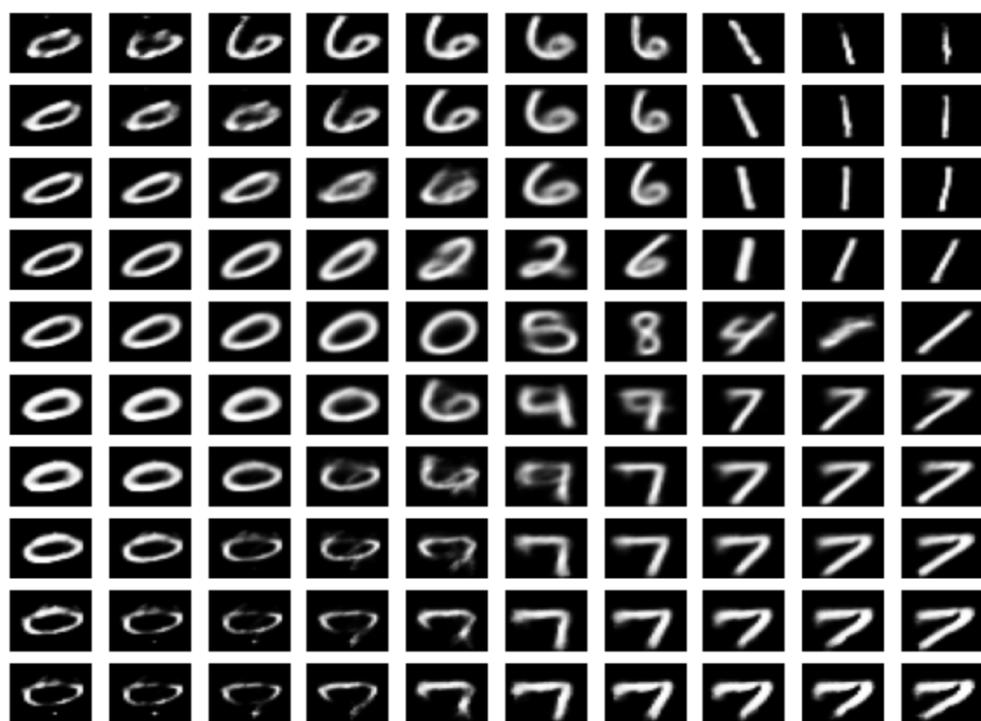






شکل ۵- خروجی شبکه در epoch ۵ مختلف (روندهای تولیدات شبکه)

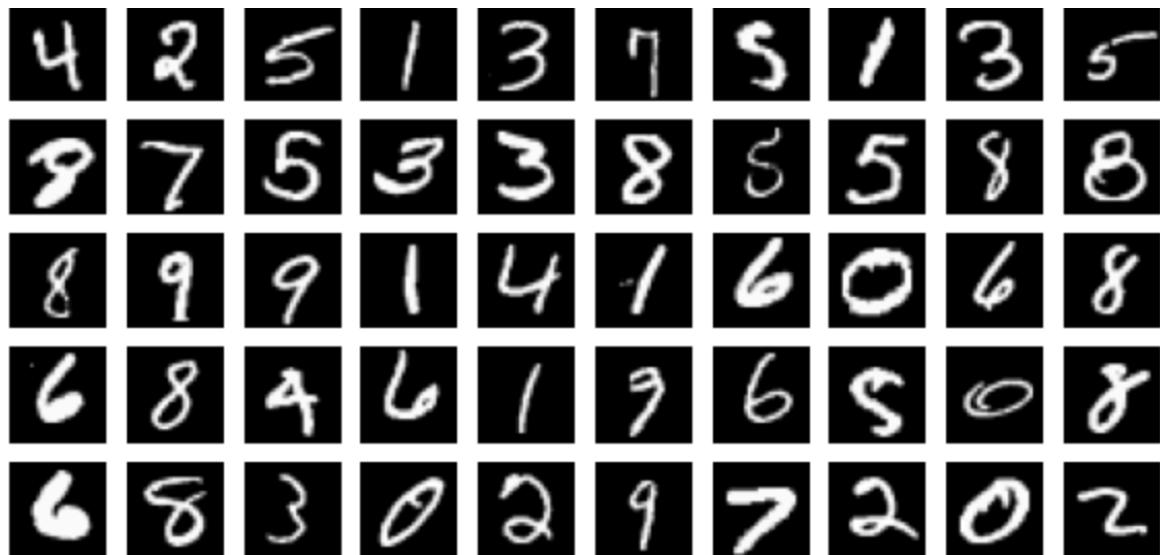
۵) از شکل سه مشخص است که پراکندگی داده‌ها در بعد اول در بازه‌ی حدودی ۶-تا ۴ و در بعد دوم در بازه‌ی ۶-تا ۶ است. در کد با استفاده از min و max این مقادیر به صورت دقیق محاسبه شده‌اند و با در نظر گرفتن ۸ نقطه میانی در هر بعد این داده‌ها از decoder رد شده‌اند و grid ای به شکل زیر به وجود آورده‌اند.



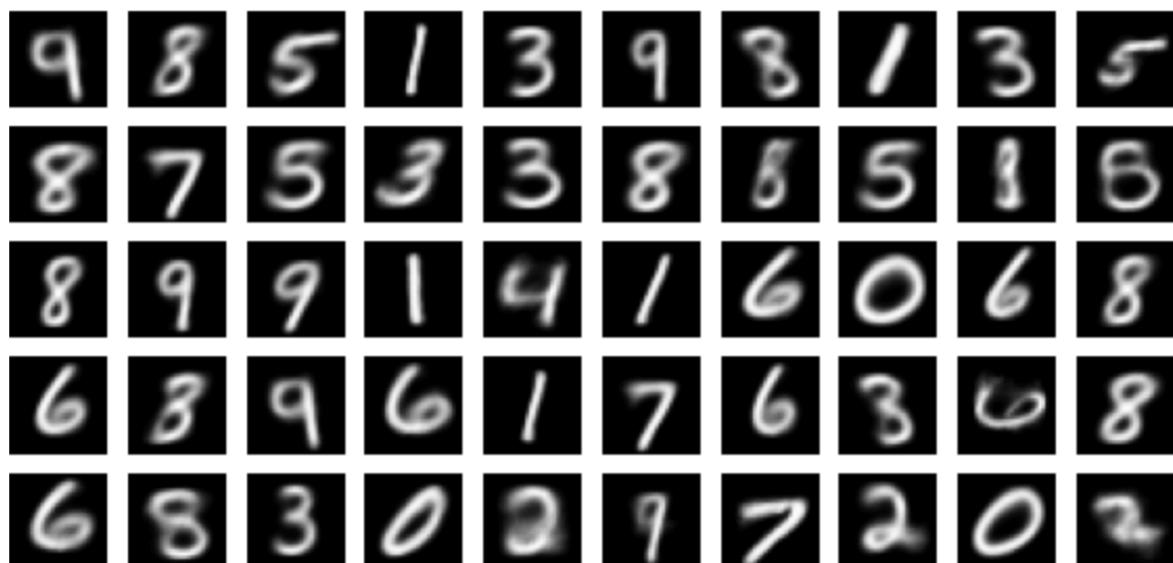
شکل ۶- تصاویر تولید شده از بازه های فضای latent با استفاده از decoder

همانطور که مشاهده می کنید، هر نقطه بر مبنای اینکه در کجای فضای latent افتاده است شبیه به عدد شده است و ویژگی های تصاویر آن حدود را به خود گرفته است. به طور مثال یک ها در بالا راست فضای latent قرار گرفته اند.

و) تعدادی از داده ها که به صورت تصادفی انتخاب شده اند در کنار تصاویر بازیابی شده آنها در دو شکل زیر نشان داده شده اند.



شکل ۷- تعدادی از تصاویر ورودی به شبکه



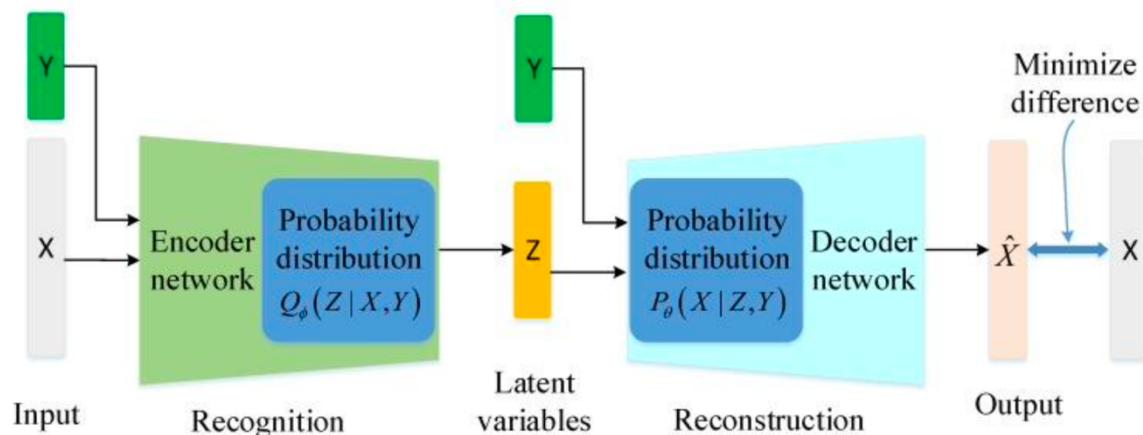
شکل ۸- تعدادی از تصاویر بازسازی شده متناظر ورودی های شکل ۷

همانطور که مشاهده می کنید تعدادی از داده ها مطابق ورودی بازسازی شده اند. (مانند تصاویر اول ردیف دوم و سوم) حتی تمیز از ورودی باز تولید شده اند.

برخی از تصاویر نیز به دلیل نمونه برداری تصادفی ویژگی اضافی پیدا کرده اند و حتی گاهی به عدد دیگری تبدیل شده اند. (مانند تصویر هفتم ردیف اول که ۵ با اضافه شدن یک منحنی به ۸ تبدیل شده است).

ز) این شبکه‌ها به این دلیل معرفی شد تا بتواند داده‌ها را به صورت کنترل شده‌تر تولید کند. به عنوان مثال، در مجموعه داده‌ی MNIST بتوان گفت که چه رقمی تولید شود.

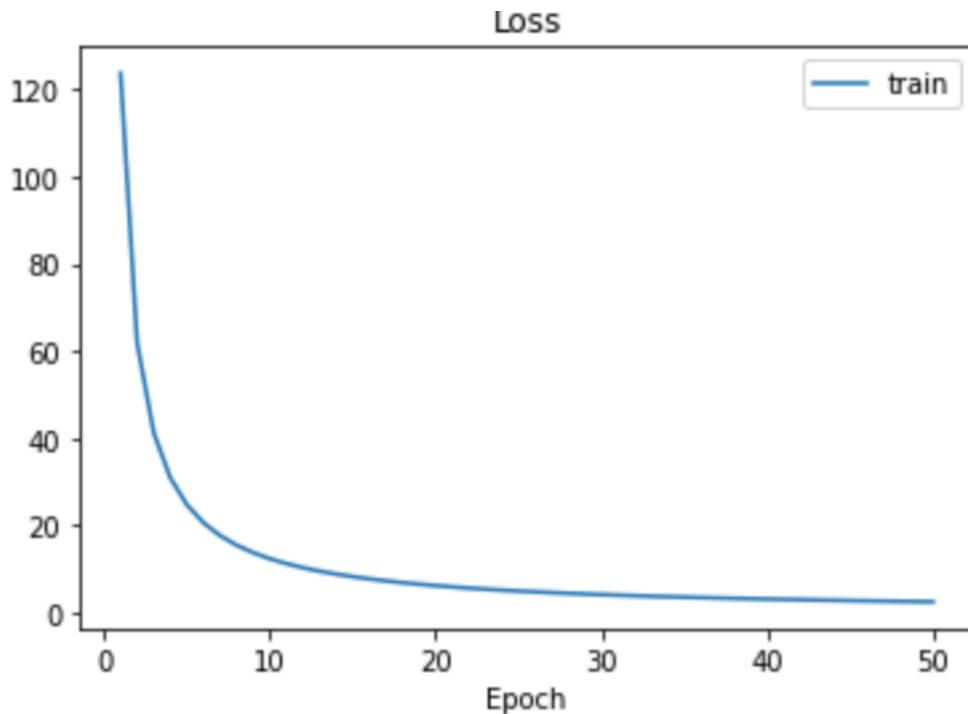
معماری این شبکه که در شکل ۹ قابل مشاهده است، مانند معماری بحث شده برای VAE است با این تفاوت که در ورودی یک condition یا شرط نیز به ورودی ضمیمه می‌شود و به encoder داده می‌شود. همچنین، این شرط با ورودی latent decoder که فضای decoder است نیز ضمیمه می‌شود و به decoder داده می‌شود. با این کار به نوعی تولیدات شبکه کنترل می‌شود و توسط شرط سمت و سو می‌یابد.



شکل ۹- معماری و ورودی‌های شبکه‌ی CVAE

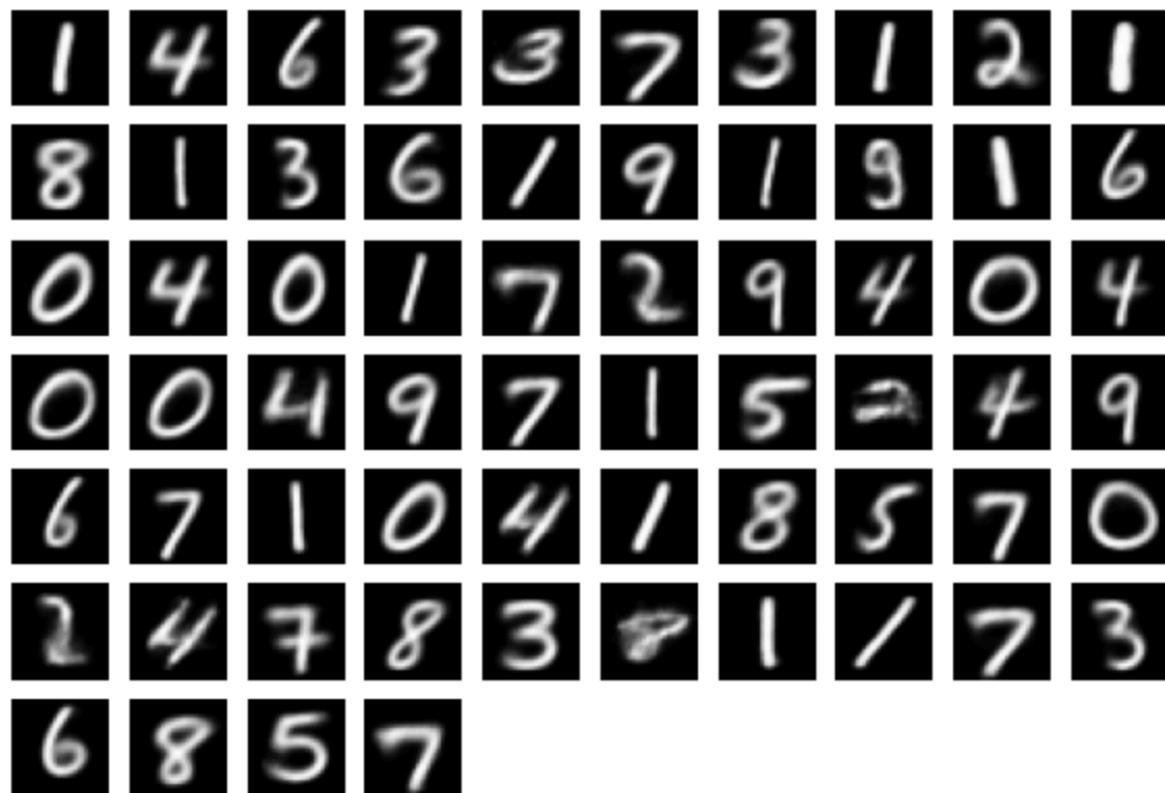
معماری گفته شده در کلاس CVAE پیاده‌سازی شده است. تفاوت آن با VAE الحاق شرط با ورودی‌هاست.

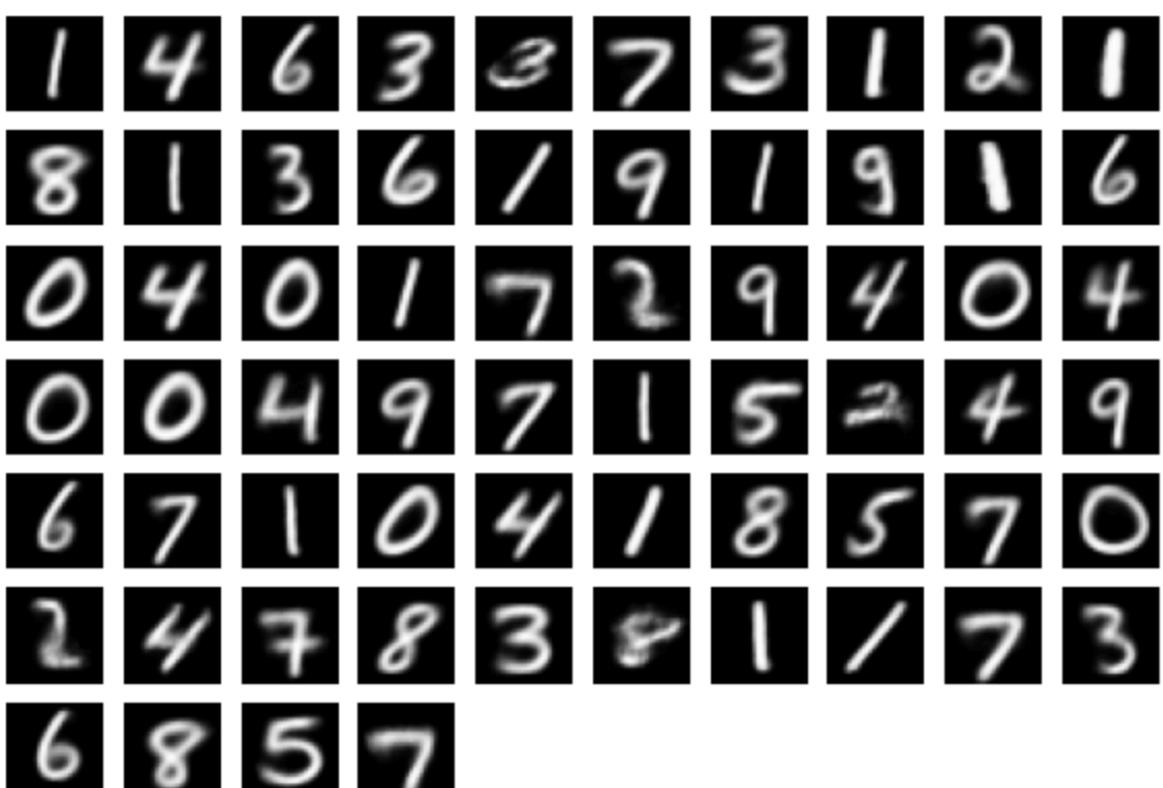
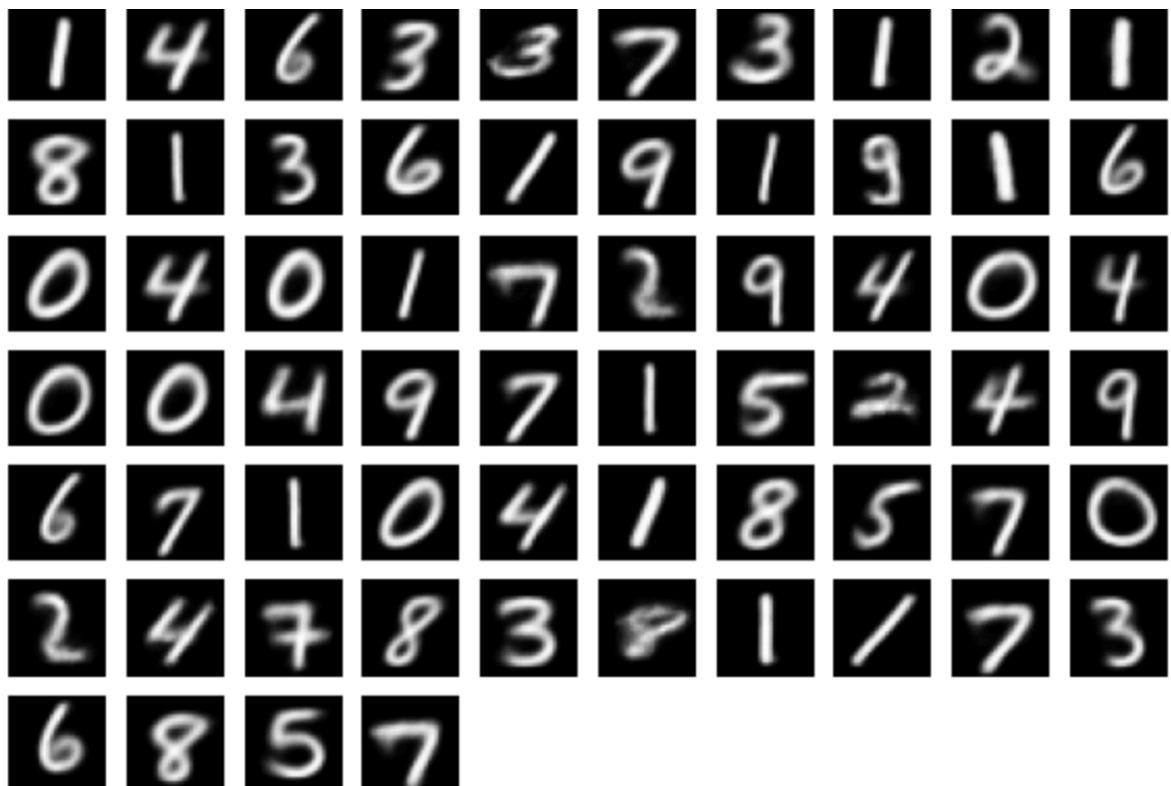
این شبکه بر روی داده‌ی MNIST آموزش داده شده است که این آموزش موفقیت‌آمیز بوده است. گواه این قضیه کاهش خطای epoch‌های متوالی است که در نمودار زیر قابل مشاهده است.

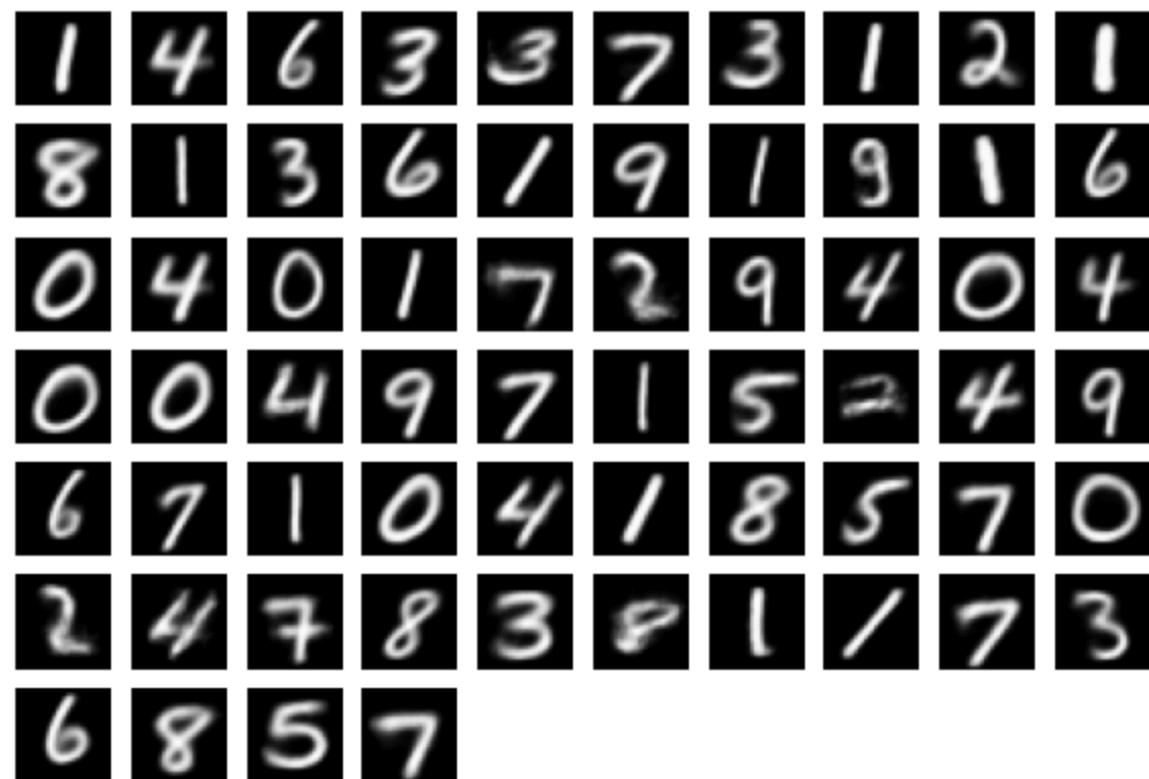
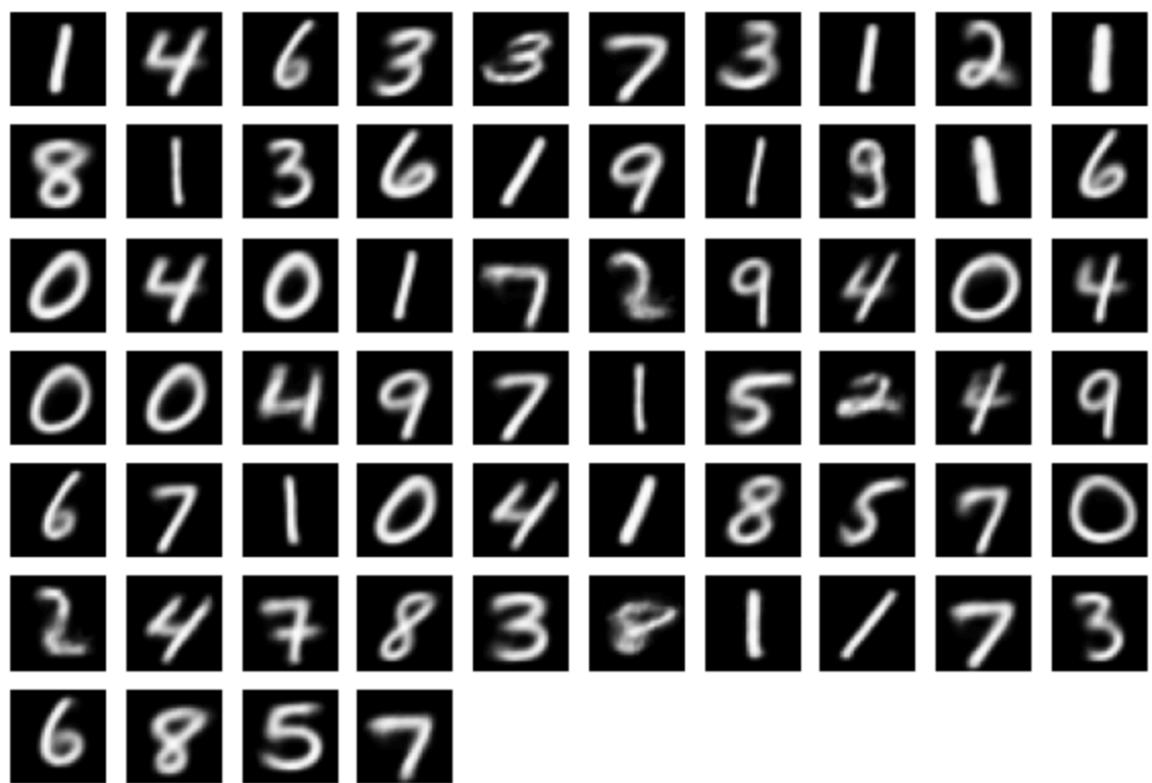


شکل ۱۰- نمودار خطای شبکه CVAE بر روی داده‌ی آموزش در epoch ۵۰

خروجی‌های تولیدشده از این شبکه با شرط برچسب واقعی داده‌ها در epoch ۵ به صورت زیر است.



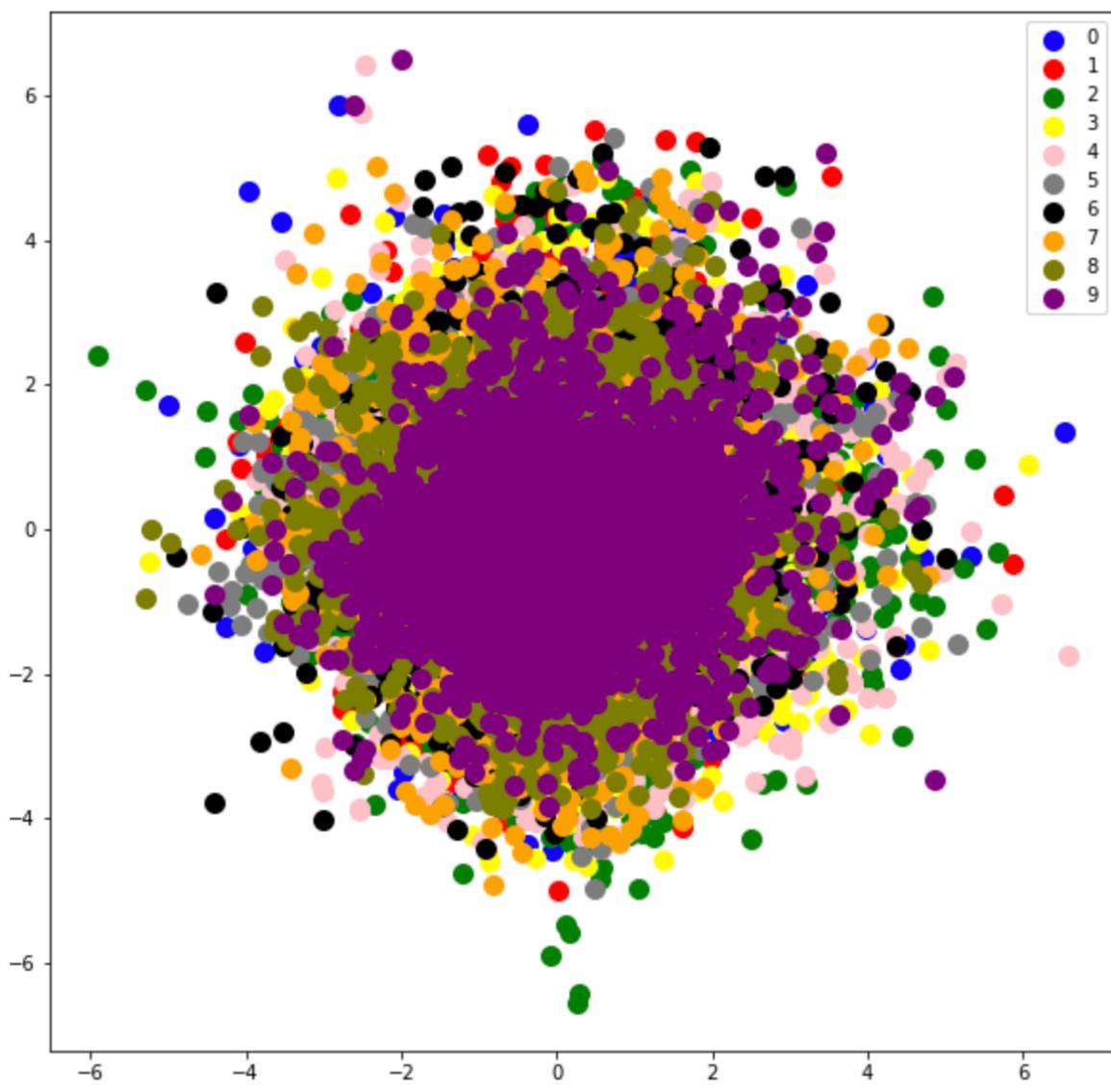




شکل ۱۱- خروجی شبکه در ۵ epoch (روندهای تولیدات شبکه)

همانطور که مشاهده می‌کنید، این تولیدات این شبکه به دلیل مشروط بودن به شرط از ابتدا قابل قبول هستند. اما در مواردی که کمی مشکل وجود دارد این مشکل رفته حل شده است و شبکه اصلاح شده است. (به عنوان مثال داده‌ی اول ردیف یکی مانده به آخر)

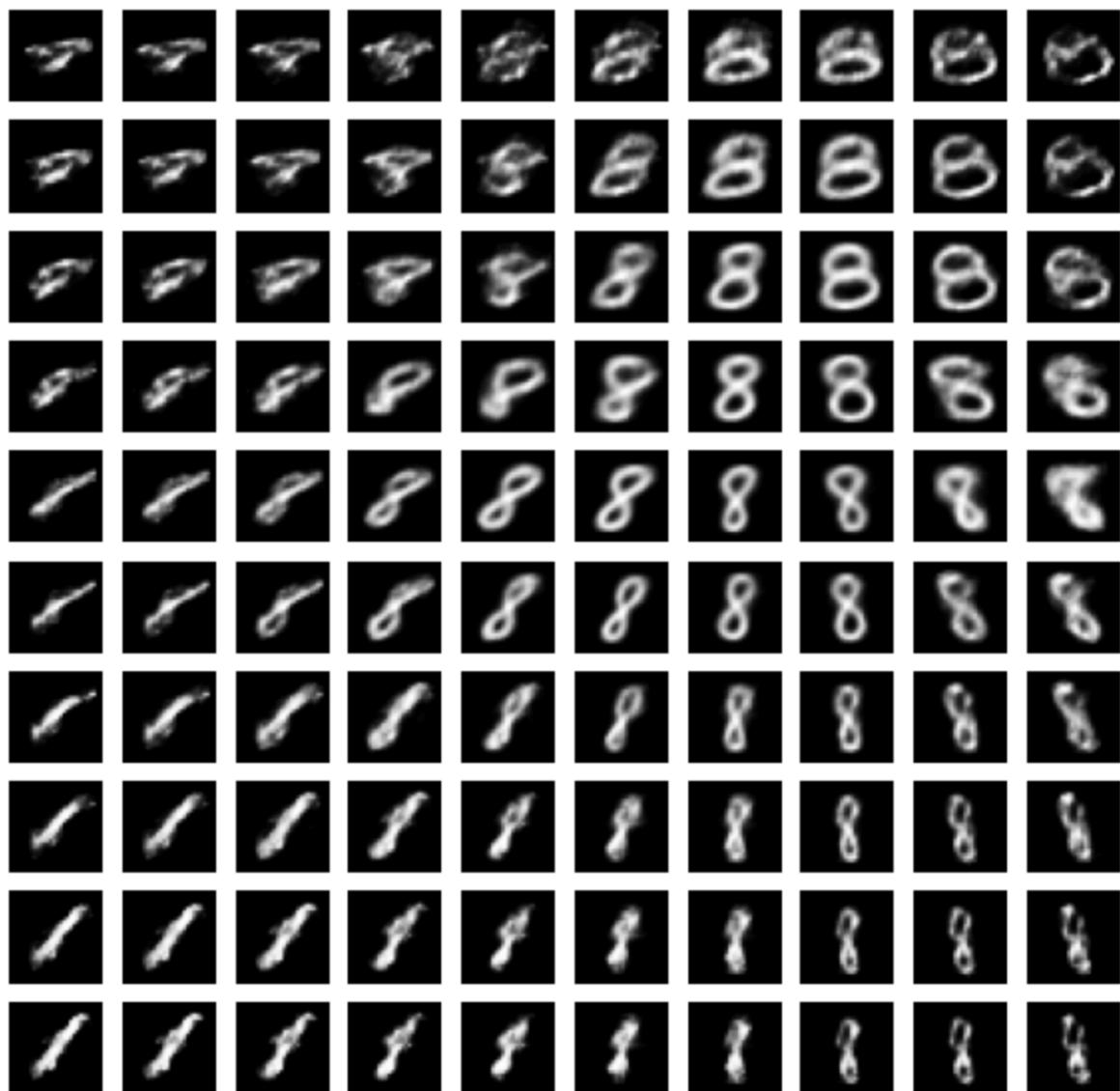
ح) مجموعه داده‌ی MNIST به کمک کلاس CVAE آموزش دیده شده تحت داده‌های این مجموعه به فضای ثانویه انتقال داده شده است که scatter plot آن‌ها در شکل زیر است. همانطور که در تصویر مشاهده می‌شود، داده‌های هر کلاس با رنگی مجزا نمایش داده شده‌اند. همچنین در گوشه تصویر رقم متعلق به هر رنگ نشان داده شده است.



شکل ۱۲ – فضای latent به دست آمده از CVAE برای داده‌ی MNIST

همانطور که در این تصویر می‌بینید، گویی تمام کلاس‌ها با یکدیگر درآمیخته‌اند. علت این موضوع آن است که در صورتی که label از کلاس y بخواهد با ویژگی‌های داده‌ی x تولید شود، داده‌ی نزدیک وجود داشته باشد. به عبارتی، با گرفتن نقطه در وسط آن‌ها داده‌ی کلاس y تولید شود که ویژگی‌های x را داشته باشد. در قسمت د، این در هم رفتگی وجود نداشت و به همین دلیل، نمی‌توانستیم از کلاس دور از هم داده‌های یکدیگر را تولید کنیم. این مشکل با این فضای latent به وجود آمده که با اضافه کردن شرط برچسب واقعی این گونه شده است، حل شده است.

ط) از شکل ۱۲ مشخص است که پراکندگی داده‌ها در بعد اول در بازه‌ی حدودی ۶-۸ و در بعد دوم در بازه‌ی ۶-۸ است. در کد با استفاده از min و max این مقادیر به صورت دقیق محاسبه شده‌اند و با در نظر گرفتن ۸ نقطه میانی در هر بعد این داده‌ها از decoder رد شده‌اند و grid ‌ای به شکل زیر به وجود آورده‌اند. لازم به ذکر است که شرط مورد نظر برچسب ۸ درنظر گرفته شده‌است.



شکل ۱۲- تصاویر تولید شده از بازه‌های فضای latent با استفاده از decoder با برچسب ۸

decoder به این برچسب شرطی شده‌است. به همین دلیل، تمام خروجی‌های ۸ است. اما فضای latent بیانگر ویژگی‌های دیگر مانند میزان چرخش و بسته بودن و باز بودن حلقه‌ها بوده‌است که در شکل این طیف را مشاهده می‌کنید.

ی) تعدادی از داده‌ها که به صورت تصادفی انتخاب شده‌اند در کنار تصاویر بازیابی شده آن‌ها در دو شکل زیر نشان داده شده‌اند. لازم به ذکر است که برچسب واقعی هر تصویری ورودی به عنوان شرط داده شده‌است.

4	2	5	1	3	7	5	1	3	5
8	7	5	3	3	8	5	5	8	8
8	9	9	1	4	1	6	0	6	8
6	8	4	6	1	9	6	5	0	8
6	8	3	0	2	9	7	2	0	2

شکل ۱۴- تعدادی از تصاویر ورودی به شبکه

4	2	5	1	3	7	5	1	3	5
8	7	5	3	3	8	5	5	8	8
8	9	9	1	4	1	6	0	6	8
6	8	4	6	1	9	6	5	0	8
6	8	3	0	2	9	7	2	0	2

شکل ۱۵- تعدادی از تصاویر بازسازی شده متناظر ورودی‌های شکل ۱۴

همانطور که مشاهده می‌کنید تمام داده‌ها مطابق ورودی بازسازی شده‌اند. دلیل این موضوع شرطی کردن به برچسب واقعی بوده است.

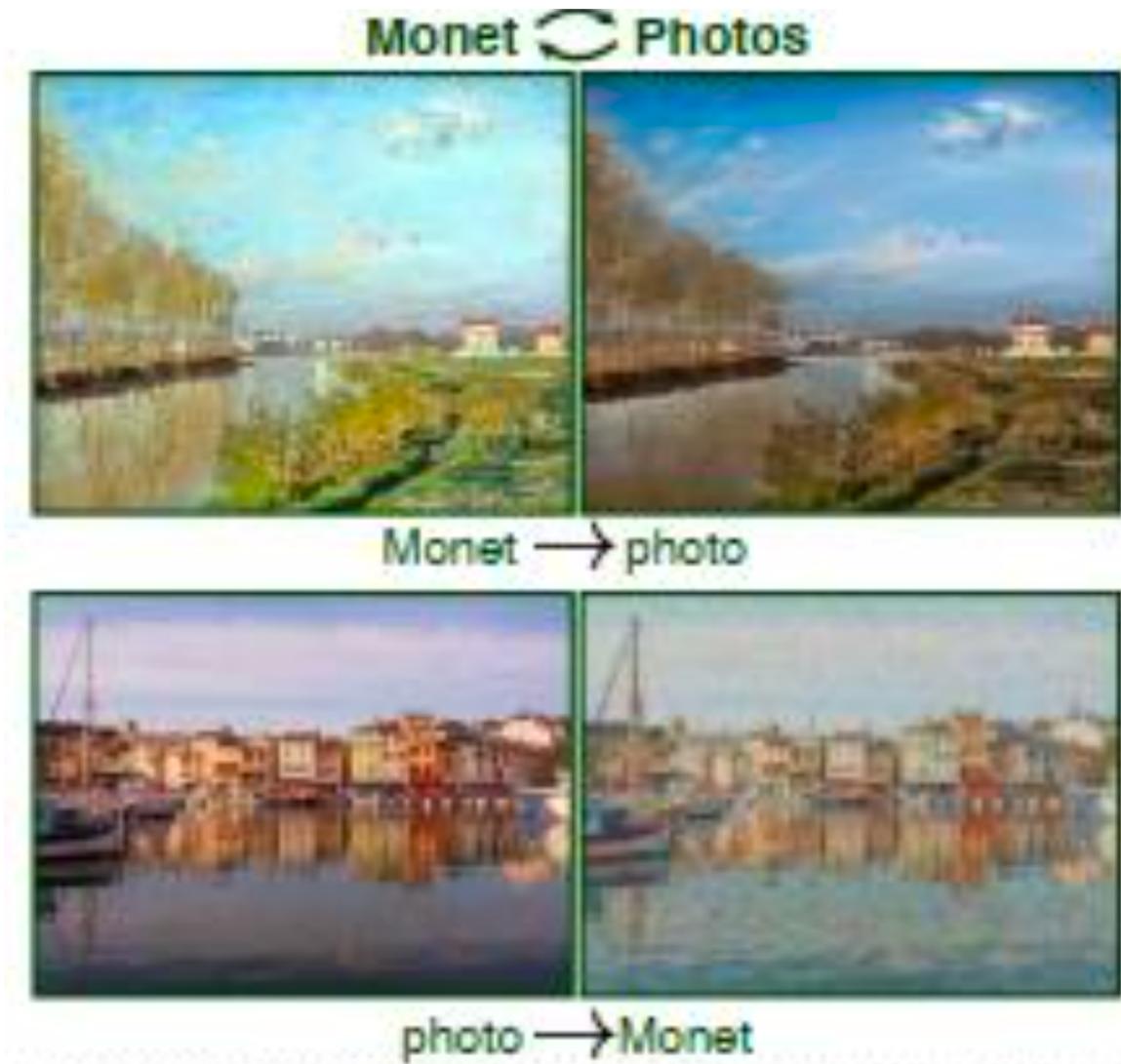
کیفیت بازیابی در این قسمت از دقت بیشتری برخوردار بوده است. زیرا به برچسب واقعی شرطی شده است. گویی به شبکه تقلب رسانده شده است. به عنوان مثال، داده اول در قسمت اول ۹ بازسازی شده بود ولی در اینجا به درستی ۴ بازسازی شده است. همچنین، داده‌ی دوم سطر اول به درستی ۲ بازسازی شده است در صورتی که در قسمت و، به اشتباه ۸ بازسازی شده بود. مثال‌های از این دست باز هم موجود است.

همچنین این شبکه خروجی مبهم نداده است زیرا تمرکزش بر روی تولید داده‌ی مورد نظر بوده است. اما شبکه‌ی قسمت و، چند مورد مبهم تولید کرده است.

لازم به ذکر است برای معتبر بودن این مقایسه ورودی‌ها در دو حالت یکسان هستند. همچنین، ابعاد فضای ثانویه نیز در دو قسمت یکسان (دو بعد) بوده است.

سوال ۲ CycleGAN

۱- شبکه‌ی CycleGAN برای تبدیل تصاویر غیرجفت مورد استفاده قرار می‌گیرد. تصاویر غیرجفت به این معنی است که اگر بخواهیم تصویر در فضای A را به تصویر در فضای B ببریم، در فضای B نمونه‌ی متناظر را در داده‌ی آموزش نداشته باشیم.
برای این که مثالی از این شبکه ببینم دو تصویر زیر می‌تواند راهگشا باشد.



شکل ۱۶ – مجموعه داده‌ی monet2photo و مصدق CycleGAN

در این مجموعه داده تعدادی نقاشی وجود دارد که می‌خواهیم عکسی شبیه به آن‌ها تولید کنیم اما این عکس را نداریم. اینجاست که CycleGAN می‌تواند مثمر فاید واقع شود و این تبدیل را برای ما انجام دهد. این کار با تبدیل photo به monet و سپس تبدیل بر عکس آن انجام می‌شود که در ادامه توضیح داده می‌شود.



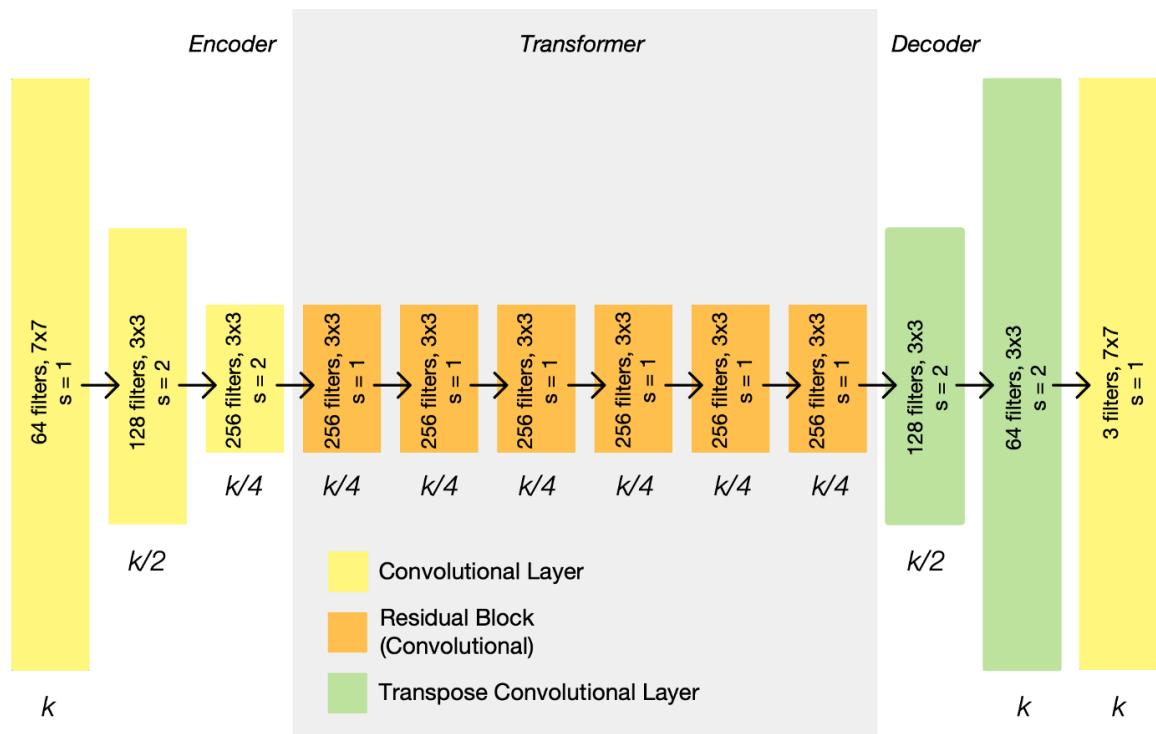
شکل ۱۷ – مجموعه داده‌ی ZebraHorses و مصادق CycleGAN

در این مثال نیز تعدادی عکس اسب/گورخر داریم که می‌خواهیم آن‌ها تبدیل به گورخر/اسپ کنیم اما تصاویر گورخری/اسپی متناظر را نداریم. به همین دلیل CycleGAN مورد استفاده قرار گرفته است. اسب/گورخر به گورخر/اسپ تبدیل شده است و سپس، دوباره به اسب/گورخر برگشته است. همانطور که مشاهده می‌کنید این کار موفق بوده است.

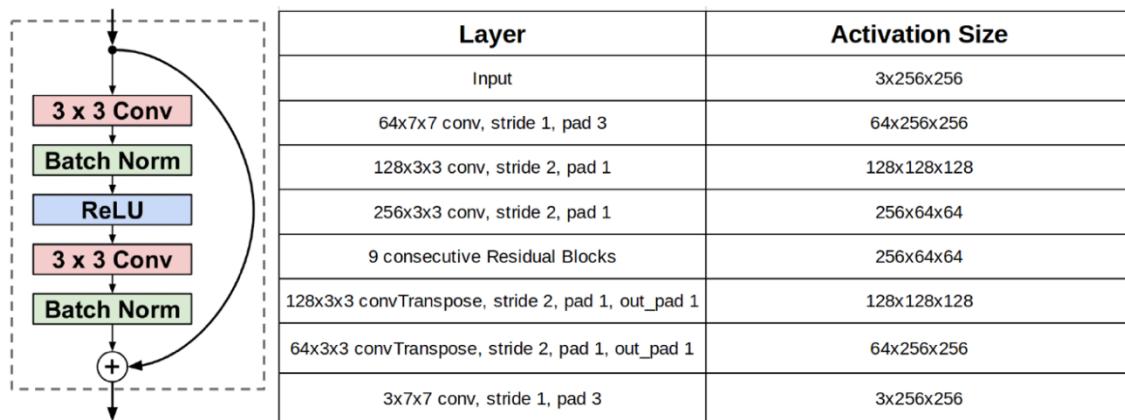
ساختر Generator‌های این شبکه در زیر قابل مشاهده است.

همانطور که در معماری زیر مشاهده می‌کنید این شبکه از سه بخش تشکیل شده است. Encoder، Decoder و بخش Transformer.

در بخش Encoder سه لایه کانولوشنی قرار گرفته‌اند که سعی بر استخراج ویژگی‌های عکس دارند. در بخش Transformer با استفاده از تعدادی Residual Block سعی بر عوض کردن ویژگی‌های استخراج شده برای رفتن به فضای دیگر دارد. در قسمت Decoder، سعی می‌شود تا از ویژگی‌ها در فضای جدید عکس تولید شود که این قسمت نیز دارای ساختار کانولوشنی است.

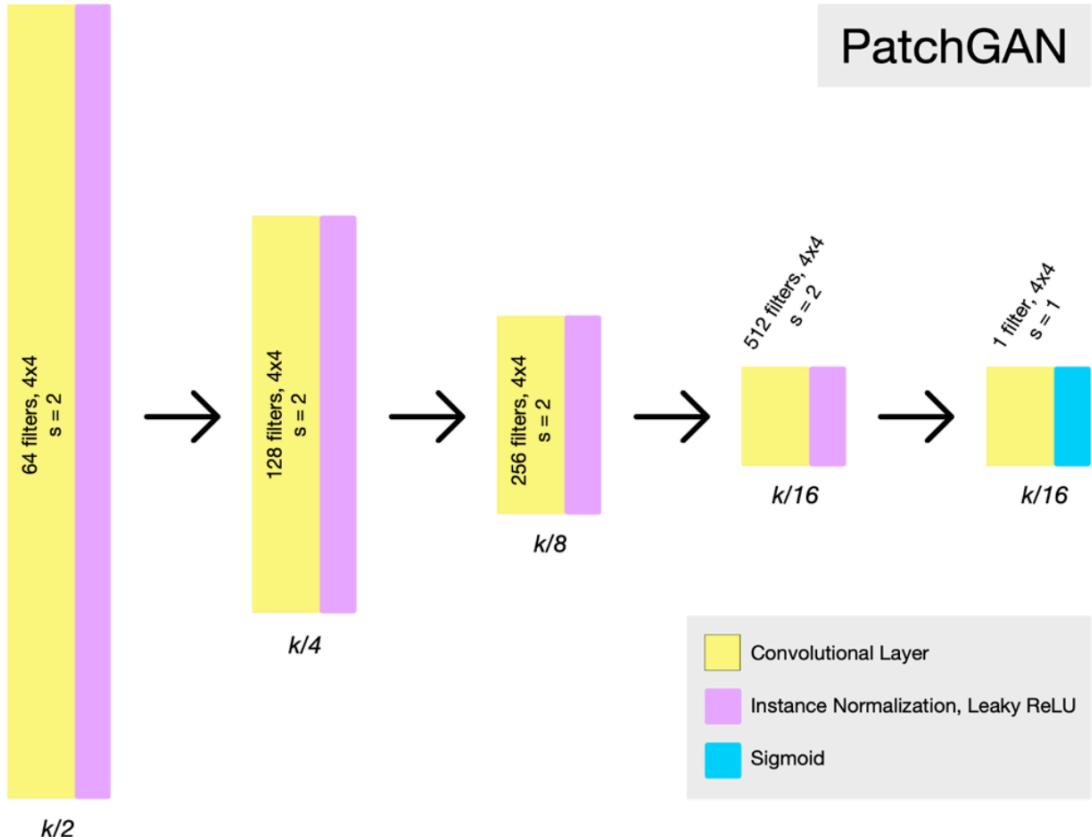


شکل ۱۸- ساختار شبکه‌ی Generator CycleGAN



شکل ۱۹- ساختار شبکه‌ی Generator CycleGAN به صورت دقیق‌تر

قسمت‌های Discriminator این شبکه نیز مشابه شبکه‌های GAN سعی بر تشخیص واقعی یا غیرواقعی بودن نمونه ورودی دارد. ساختار این قسمت در شکل زیر قابل مشاهده است.



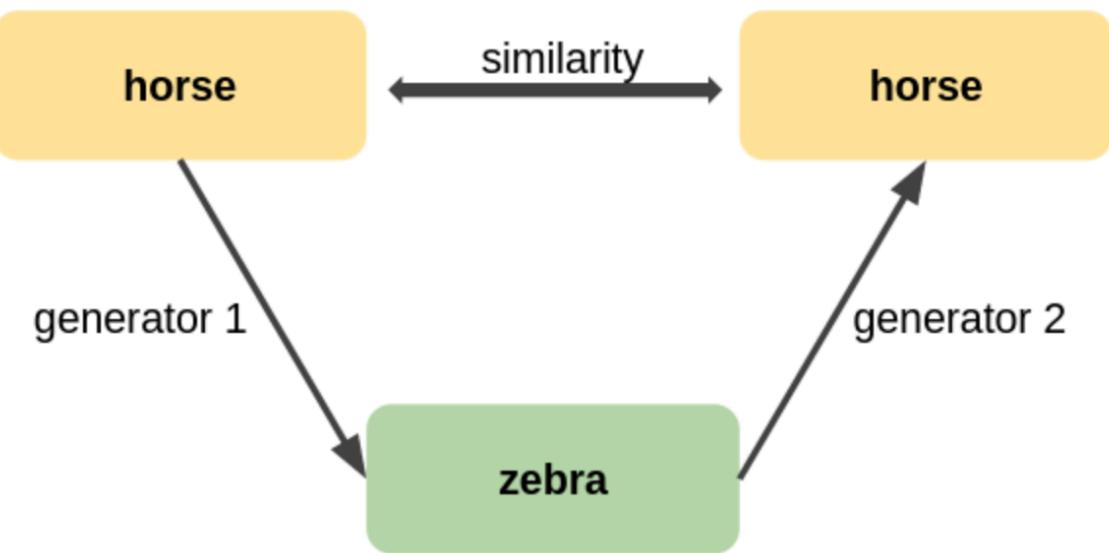
Discriminator

شکل ۱۹—ساختار شبکهی Discriminator

این قسمت ابتدا سعی می‌کند با تعدادی لایه‌ی کانولوشنی و نرمال‌سازی و توابع فعالسازی ویژگی‌های موثر برای تشخیص را در چند گام بیرون بکشد و هم‌زمان آن‌ها به ورودی‌های موثر Sigmoid برای گزارش نهایی تبدیل کند. Sigmoid در انتها عمل تشخیص را انجام می‌دهد.

ساز و کار کلی این شبکه به دلیل عدم وجود داده در فضای دوم به این صورت است که دارای دو Generator و دو Discriminator است. یک Generator و Discriminator برای تبدیل عکس از فضای A به فضای B است. Generator عکس مورد نظر را تولید می‌کند و Discriminator عکس‌هایی در فضای B و عکس‌های دیگری به صورت تولیدی دریافت می‌کند و سعی می‌کند واقعی یا تولیدی بودن آن‌ها را تشخیص دهد و با تکرار این کار، هم‌زمان با بهتر شدن تشخیص Generator، Discriminator در تولید نمونه‌ها متبحر می‌شود.

حال به همین شکل Generator و Discriminator دیگری برای تبدیل نمونه‌های B به فضای A تلاش می‌کنند و به این صورت یک حلقه به وجود می‌آید که نام شبکه نیز برگرفته از این موضوع CycleGAN گذاشته شده‌است. این کار نیاز به داده‌ی آموزش در فضای دیگر را حل می‌کند. این مفهوم به صورت شماتیک در تصویر زیر نشان داده شده‌است.



شکل ۲۰- سازوکار شبکه‌ی CycleGAN در مثال اسب و گورخر

در آموزش این شبکه سه نوع خطا مورد استفاده قرار گرفته است که به ترتیب هر کدام به همراه توضیح‌شان ارائه می‌شود.

Adversarial:

در اینجا سعی بر این است تصاویری یا monet‌های تولیدی توسط discriminator متناظر شان واقعی تشخیص داده شوند. به عبارتی آنقدر کار Generator‌ها خوب باشد که discriminator به اشتباه انداده شود و آن‌ها را واقعی تشخیص دهد. این خطا به نوعی بیانگر میزان کیفیت کار Generator است. این خطا برای هر دو طرف تعریف می‌شود.

Cycle Consistency:

این خطا تلاش بر این است که اگر تصاویری monet تولیدی از monet/monet را به Generator تبدیل عکس به monet/monet به عکس بدھیم، تصویر تولیدی مشابه monet/monet اولیه باشد. به عبارت دیگر، اگر دو بخش شبکه را پشت هم قرار دهیم، به نوعی چرخه کامل شود و تصاویری monet/monet اولیه بازسازی شود و این یعنی چرخه شبکه کامل شده است.

Identity:

این خطا کمی عجیب است. این خطا بیان می‌کند که اگر تصویر اصلی را به generator monet تبدیل به تصویر بدھیم سعی کند خودش را برگرداند. این خطا نیز دو طرفه تعریف می‌شود و برای monet نیز صادق است.

به عبارت دیگر، این خطا بیان می‌کند که generator باید تفاوت تصویر و monet را بداند و به نوعی حالت ایده‌آل خود را تصویر monet/monet ورودی بداند و خودشان را به خودشان تبدیل کند.

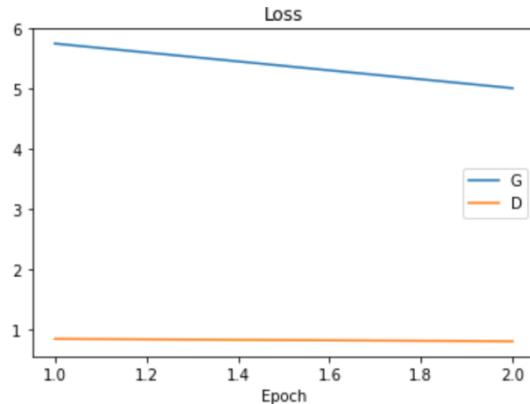
۲- شبکه‌ی CycleGAN برای بخش PatchGAN خود از ساختار discriminator استفاده می‌کند. PatchGAN یک discriminator است از این جهت که عددی به عنوان واقعی یا fake بودن می‌دهد اما تفاوتی که دارد آن است که یک عدد مربوط به کل تصویر تولیدی نمی‌دهد که واقعی یا

تقلیبی بودن آن را بگوید. این شبکه به صورت دقیق تر عمل می‌کند و واقعی یا تقلیبی بودن را به ازای هر پنجره به سایز مشخص می‌دهد که این ایده از CNN و پنجره متحرک به آن راه یافته است.

با این کار بررسی PatchGAN دقیق تر می‌شود و به همین دلیل discriminator دقیق‌تری خواهد بود. دلیل استفاده از لایه‌های کانولوشنی در ساختار آن نیز همین ساختار پنجره پنجره است.

۳- شبکه‌ی CycleGAN را در دو کلاس Generator و Discriminator سوال دو پیاده‌سازی شده است. این شبکه با استفاده از مجموعه‌داده‌ی monet2photo آموزش داده شده است. ساختارها در بخش‌های قبلی به طور کامل توضیح داده شدند. برای آموزش این شبکه از هر سه خطای Adversarial استفاده شده است. Cycle Consistency و Identity.

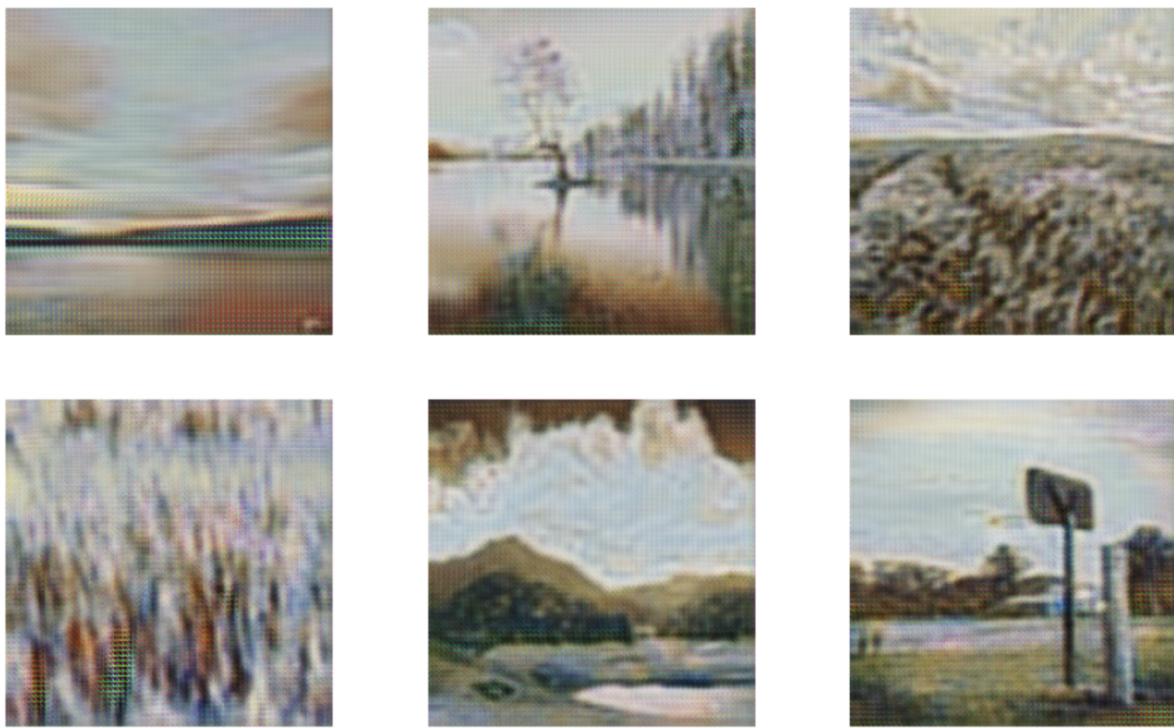
این آموزش به دلیل زمان بر بودن و طبق گفته‌ی دستیاران آموزشی در دو epoch انجام شده است. نمودار loss برای Generator و Discriminator و نتایج تصاویر تولید شده به صورت زیر بوده است. لازم به ذکر است این آموزش روی ۴۰۰۰ عکس انجام شده است.



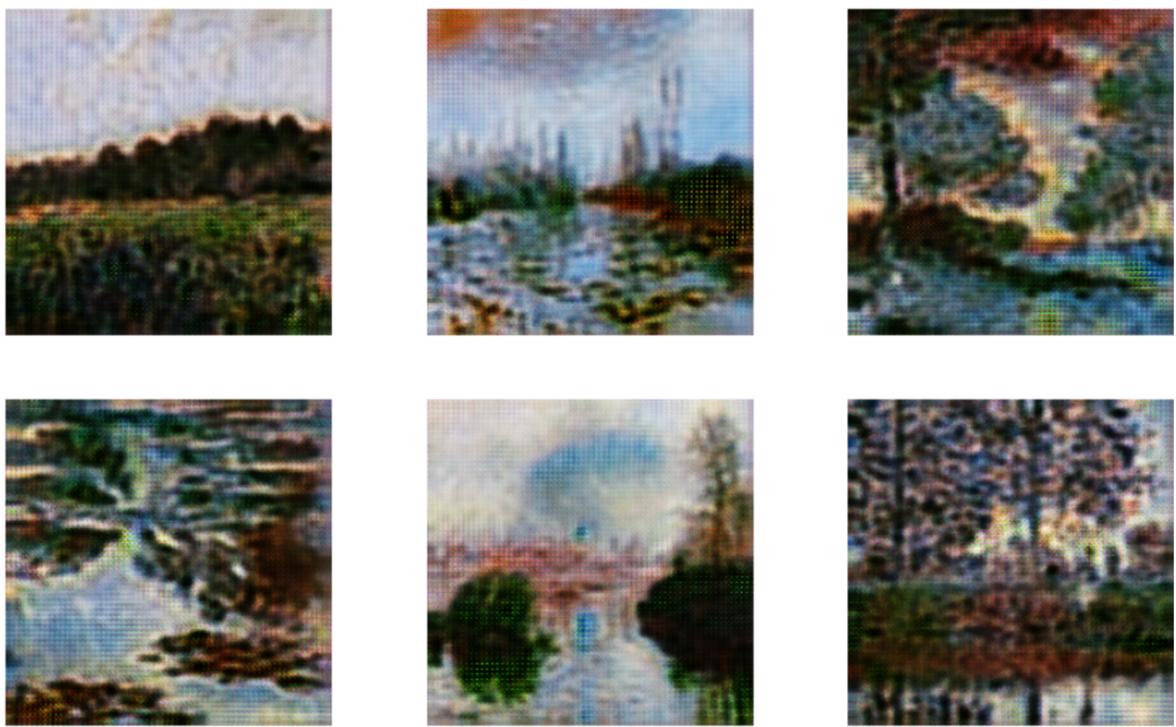
```
{'G': [5.753929816961288, 5.012006928682327], 'D': [0.8415226840376854, 0.8006938759982586]}
```

شکل ۲۱- نمودار خطای Generator و discriminator در دو epoch به همراه مقدار آن‌ها

این خطاهای مربوط به قسمت monet2photo است. این موضوع به دلیل نوع مجموعه داده است.



شکل ۲۲- برخی تصویرهای تولیدی شبکه monet



شکل ۲۳- برخی تصویرهای تولیدی شبکه

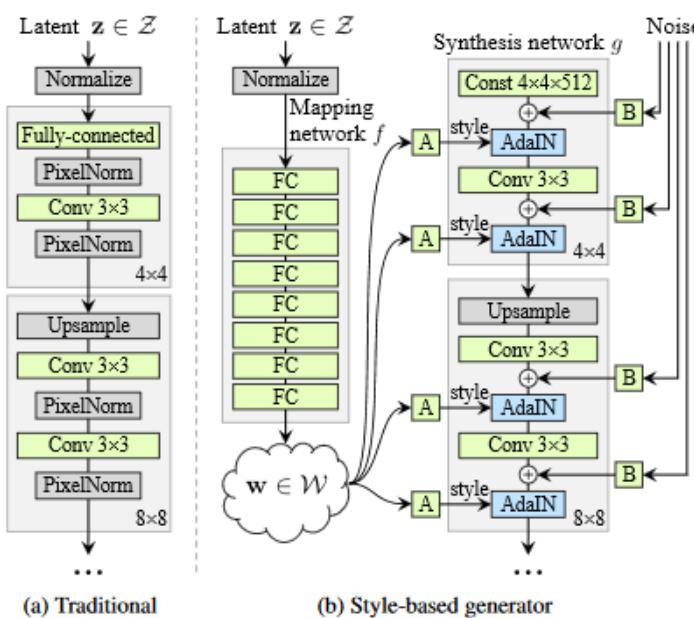
همانطور که مشاهده می‌کنید، شبکه در دو epoch پیشرفت خوبی داشته است و از حالت سیاه خارج شده و تصویر/تصویرهای جالبی تولید کرده است که در صورت ادامه آموزش روی سخت افزار بهتر و داشتن زمان بیشتر می‌توانست خروجی بهتری بدهد.

سوال ۳ – آشنایی با مقالات مربوط

مقاله انتخابی در این بخش:

StyleGAN

این شبکه بر روی ساخت تصویر جدید از چهره انسان تمرکز دارد. این شبکه یک نوع GAN است که از ادبیات style transfer بهره می‌برد. این شبکه به صورت unsupervised ویژگی‌ها سطح بالا را می‌آموزد و با تغییرات تصادفی ویژگی‌ها را عوض می‌کند و تصاویر جدید به وجود می‌آورد.
معماری این شبکه به صورت زیر است.



شکل ۲۴ – معما ری شبکه های StyleGAN در مقابل معما ری سنتی

سمت چپ بیانگر معما ری شبکه های سنتی تولید کننده عکس است که از تعدادی نرمالیزیشن، کانولوشن و upsample تشکیل شده است.

سمت راست تولید کننده های style-based نشان داده شده اند. این شبکه ها سه تفاوت اساس دارند.

۱. شبکه های mapping: این شبکه از یک MLP λ لایه تشکیل شده است. نقش آن encode کردن بردار latent ورودی به یک بردار میانی است که با w نشان داده شده است. در این شبکه لایه ورودی مستقیم حذف شده است. این بردار میانی به یک Affine transformation داده می شود و سپس، وارد شبکه می شود که در ادامه توضیح داده می شود.

The Affine Transformation (A) and Adaptive Instance Normalization .۲

:AdaIN)

Affine Transformation در حقیقت وظیفه استخراج style را بر عهده دارد و AdaIN یک روش نرمالیزیشن است که از batch normalization آمده است. این قسمت ورودی را از کانولوشن بخش قبل می گیرد و style گرفته شده از تبدیل را بر روی آن اعمال می کند.

۳. یک بردار نویز B نیز با ورودی هر بخش جمع زده می شود. دلیل این موضوع ایجاد کردن ویژگی های کوچک جدید است.

تابع هزینه مورد استفاده در این شبکه دقیقاً مانند GAN است و تغییر در آن رخ نداده است. تنها تفاوت در این شبکه مربوط به Generator است. البته این شبکه پارامترهای بسیار زیادی دارد و به همین دلیل، نیاز به زمان آموزش بالایی دارد.

مجموعه داده‌ی این مقاله مجموعه‌ی Flickr-Faces-HQ بوده است که توسط گردآورندگان این مقاله جمع آوری شده است. این مجموعه شامل ۷۰۰۰۰ عکس است.

پیشینه این مقاله به GAN style transfer و GAN برمی‌گردد که باعث این تحول در معماری شده است. شبکه‌های GAN سنتی برای این کاربرد مناسب نبودند و نویسنده‌گان این مقاله با ارائه این شبکه تحول عظیمی در تولید چهره انجام دادند. این مقاله مرز واقعیت و تولید ماشین را همانطور که در تصویر زیر می‌بینید جابجا کرده است.



شکل ۲۵ – تعدادی چهره‌ی غیرواقعی تولیدی توسط این مدل

پیاده‌سازی با استفاده از transfer learning در کد قابل مشاهده است.