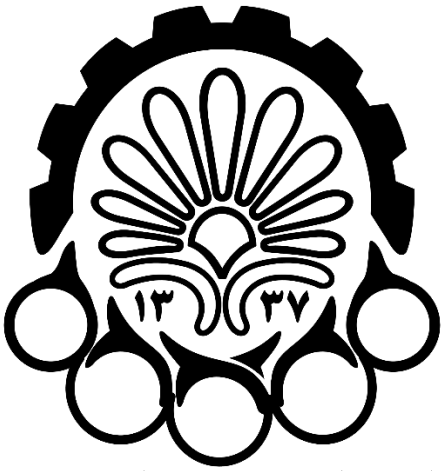


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمرین هفتم درس شبکه عصبی

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

بهار ۱۴۰۱

فهرست مطالب

پیاده سازی GAN و توضیحات اولیه

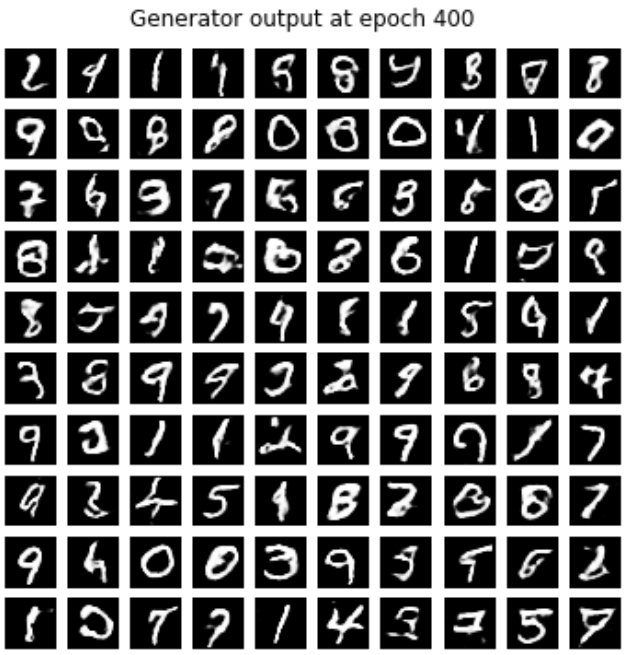
در این تمرین قصد داریم شبکه GAN را با کمک لایه های FC و Conv پیاده سازی کنیم. از آن برای تولید تصاویر استفاده کنیم و درنهایت این دو پیاده سازی را با هم مقایسه کنیم.

مجموعه داده مربوط به این سوال مجموعه ای از تصاویر سگ ها و گربه ها در حالات مختلف است. متاسفانه تصاویر این مجموعه داده Variation زیادی دارند و این باعث می‌شود شبکه GAN راه بسیار سختی برای تولید این تصویر داشته باشد.

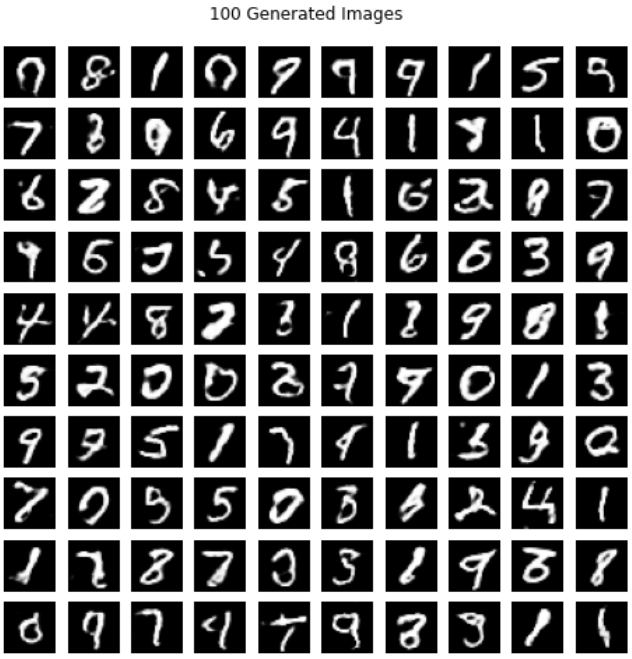
مشکل دیگری که وجود داشت مشکل زمان آموزش زیاد بود. آموزش این مدل ها چندین ساعت طول می‌کشید. همچنین عدم دسترسی دائم به GPU این روند را سختتر کرده بود.

با توجه به نکات ذکرشده، تعدادی تصمیم گرفته شد. ابتدا سایز تصاویر ورودی را بسیار پایین در نظر گرفتیم. اندازه مورد استفاده ۲۸ در ۲۸ بود که سایز بسیار پایینی است. همچنین برای ساده تر کردن محاسبات تصاویر را تک کاناله کردیم. تعدادی از تصاویر این مجموعه داده پس از اعمال تغییر اندازه و تبدیل به سطح خاکستری را در تصویر زیرمشاهده میکنید.

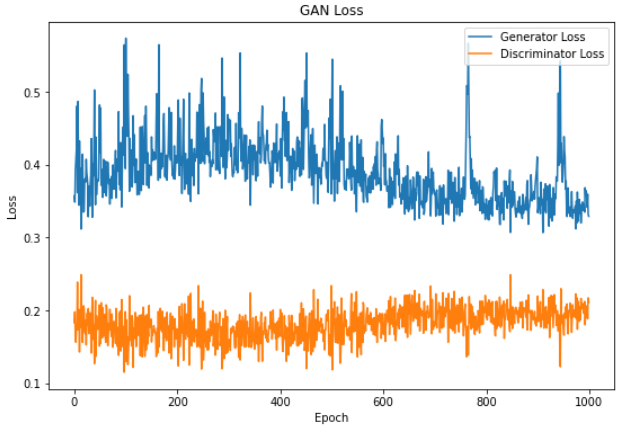
همانطور که قبل تر گفتیم، تصاویر این مجموعه داده بسیار متفاوت از هم هستند به طوری که مخصوصا در این اندازه، حتی برای انسان هم نیز به MNISTتشخیص برخی از این تصاویر سخت است. پس از انجام تعدادی آزمایش و به نتیجه نرسیدن، تصمیم گرفتیم از مجموعه داده استفاده کنیم تا متوجه شویم که آیا پیاده سازی دارای مشکلاتی است یا صرفا پارامترها باید برای مسئله Baselineطور موازی به عنوان یک به نتایج خوبی دست یافتیم.MNISTاصلی بهینه شوند. پس از انجام آزمایش ها بر روی مجموعه داده



میتوانید داده های واقعی MNIST را در تصویر زیر مشاهده کنید.



همانطور که مشاهده می‌شود، نتایج تولید شده توسط مولد بسیار به تصاویر MNIST شبیه هستند و این نشان میدهد که پیاده سازی دارای مشکل نیست و صرفا نیاز است با صرف زمان و انجام آزمایش های بیشتر بهترین ابرپارامترها را برای مسئله اصلی پیدا کنیم. در نهایت پس از چندین ساعت آموزش و تغییردادن پارامترها به نتیجه زیر برای تصاویر مجموعه داده Cats and Dogs رسیدیم. با توجه به نمودار خطا مشاهده میکنیم که خطای مولد و متمایزکننده در حال رسیدن به هم هستند که نشانه خوبی است. این آموزش در دو مرحله انجام شد. هر مرحله مدل را به میزان ۱۰۰۰epoch آموزش داد. بنابراین نتیجه زیر حاصل آموزش مدل برای ۲۰۰۰ مرحله است.



لازم به ذکر است که مقادیر تصاویر را نیز به بازه -۱ تا +۱ بردیم. در تصویر زیر میتوانید معماری مدل نهایی را مشاهده کنید.

مولد

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 12544)	1254400
batch_normalization (BatchNormalization)	(None, 12544)	50176
leaky_re_lu (LeakyReLU)	(None, 12544)	0
reshape (Reshape)	(None, 7, 7, 256)	0
conv2d_transpose (Conv2DTranspose)	(None, 7, 7, 128)	819200
batch_normalization_1 (BatchNormalization)	(None, 7, 7, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 7, 7, 128)	0
conv2d_transpose_1 (Conv2DTranspose)	(None, 14, 14, 64)	204800
batch_normalization_2 (BatchNormalization)	(None, 14, 14, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 14, 14, 64)	0
conv2d_transpose_2 (Conv2DTranspose)	(None, 28, 28, 1)	1600

متمایزکننده

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 64)        1664

 leaky_re_lu_3 (LeakyReLU)   (None, 14, 14, 64)        0

 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_1 (Conv2D)           (None, 7, 7, 128)         204928

 leaky_re_lu_4 (LeakyReLU)   (None, 7, 7, 128)         0

 dropout_1 (Dropout)         (None, 7, 7, 128)         0

 flatten (Flatten)           (None, 6272)              0

 dense_1 (Dense)             (None, 1)                 6273

=================================================================
Total params: 212,865
Trainable params: 212,865
Non-trainable params: 0
```

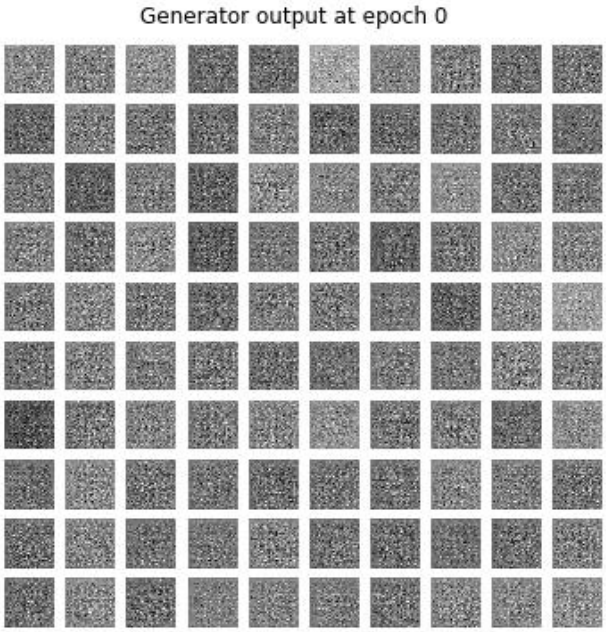
برای بهینه سازی وزن ها از بهینه ساز Adam استفاده کردیم. و همچنین به برچسب ها مقداری نویز اضافه شد.

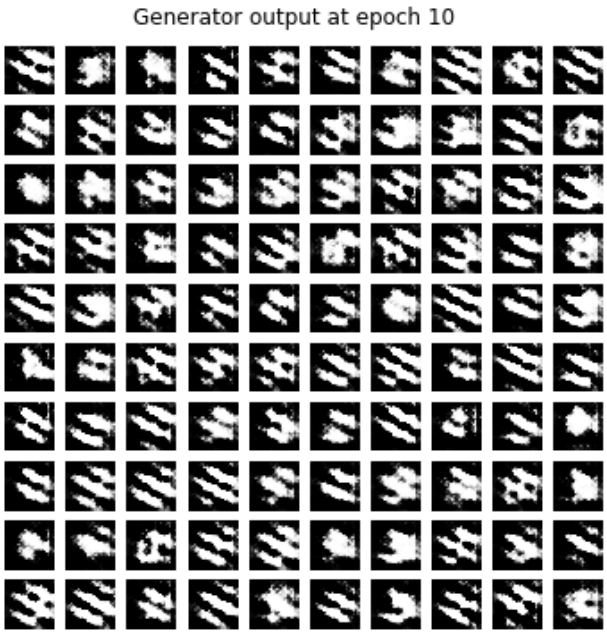
```python
def generator_loss(fake_output):
    # Add small amount of noise to labels to help with training
    noise_fake = 0.05 * tf.random.uniform(tf.shape(fake_output))

    return cross_entropy(tf.ones_like(fake_output)+noise_fake, fake_output)

```

در این قسمت میتوانید بخشی از روند آموزش این مدل را مشاهده کنید.





Generator output at epoch 15

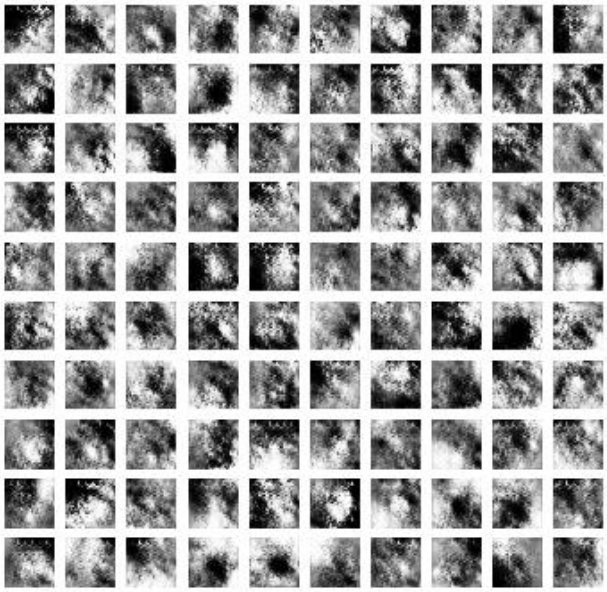
Generator output at epoch 45

Generator output at epoch 150

Generator output at epoch 405

دور دوم آموزش از ادامه ۱۰۰۰ مرحله قبل

Generator output at epoch 180



Generator output at epoch 900

سوال ۱)

فرآیند آموزش شبکه مولد متقابلی : برای آموزش این شبکه، ابتدا تعدادی تصویر تولید شده توسط مولد(که در مرحله اول دارای وزن های تصادفی و تنظیم نشده است) با برچسب **یک** و تعدادی تصویر واقعی از مجموعه داده مسئله با برچسب **صفر** را به متمایز کننده می‌دهیم. متمایز کننده که یک شبکه دسته بند است، با استفاده از تابع Loss مربوطه و مقایسه کردن برچسب های داده شده و برچسب های پیش بینی شده، یک Loss برای خود محاسبه میکند. در ادامه این Loss برای به‌روزرسانی وزن‌های متمایز کننده مورد استفاده قرار میگیرد.

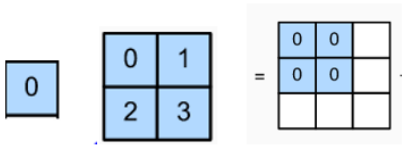
از آن طرف، مولد، با دریافت یک بردار نویز، سعی میکند تصویری تولید کند که اگر به عنوان ورودی به متمایزگر داده شد، متمایزگر به اشتباه به آن تصویر برچسب واقعی بزند. برای این‌کار تعدادی تصویر را توسط مولد تولید میکنیم و به آنها را به همراه برچسب نادرستِ "واقعی" به متمایزکننده می‌دهیم. متمایزکننده مانند بخش قبل سعی میکند یک Loss تولید کند. این Loss زمانی کم خواهد شد که مولد بتواند تصاویری تولید کند که متمایزکننده به اکثر آنها برچسب واقعی نسبت دهد. در نتیجه از این Loss برای به‌روزرسانی وزن مولد استفاده می-کنیم.

در حلقه اصلی آموزش، این دو مدل به طور همزمان آموزش داده خواهند شد. به مرور، مولد یاد میگیرد تصاویری تولید کند که متمایزکننده را به خطا بیندازد و متمایزکننده نیز یاد یادمی‌گیرد در مقابل تصاویر غیرواقعی تولید شده توسط مولد مقاوم تر عمل کند.

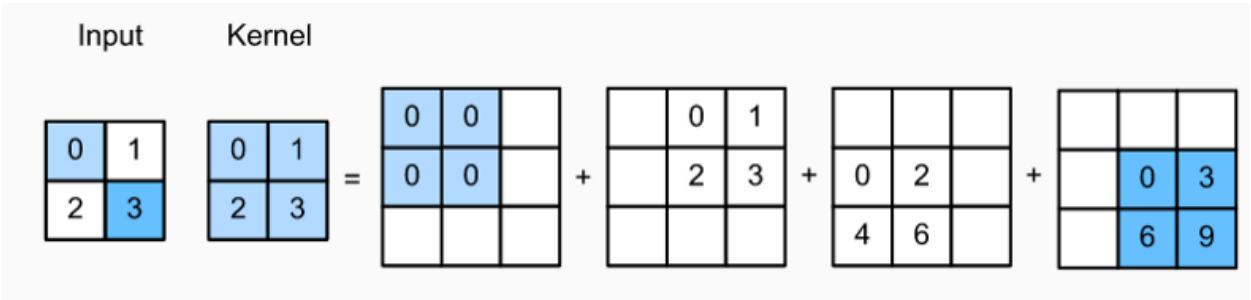
سوال ۲)

لایه های Conv2DTranspose یا لایه های معکوس کانولوشنی همانطور که از اسم آنها نیز پیداست برعکس لایه های کانولوشن عمل میکنند. از لایه های Conv2DTranspose برای افزایش اندازه تصویر ورودی یا Upsampling استفاده می‌شود. تفاوت این لایه با لایه Upsampling2D ساده این است که این لایه دارای وزن است و در حین آموزش شبکه، می‌آموزد به چه نحوی عمل Upsampling را انجام دهد.

در مورد نحوه عملکرد این لایه ها میتوان گفت که بر خلاف لایه کانولوشن که بر روی تصویر ورودی اعمال میشد و تصویر خروجی را تولید میکرد، این لایه بر روی تصویر خروجی اعمال می‌شوند و فیلتر را با ضریب هر کدام از درایه های ورودی بر روی خروجی کپی میکند.



عملیات نشان داده شده را برای تک تک درایه های ورودی انجام میدهیم.



هممچنین برای حل مشکل Overlap شدن در این حالت، درایه هایی که Overlap دارند را با هم جمع میکنیم.

سوال ۳)

همانطور که در بخش اول توضیح دادیم، قسمت متمایزکننده شبکه GAN یک دسته بند دو کلاسه است. این دسته بند تصاویر واقعی و غیرواقعی ورودی را میگیرد و به آنها برچسب واقعی یا غیرواقعی می‌زند.تابع هزینه مناسب برای این گونه مسائل، **Binary Cross Entropy** است. همچنین با توجه به نحوه عملکرد شبکه GAN باید یکبار Loss تصاویر واقعی را با برچسب واقعی و بار دیگر تصاویر تولید شده توسط مولد با برچسب غیرواقعی را با استفاده از Binary Cross Entropy محاسبه کنیم. این دو هزینه بدست آمده با هم جمع شده و به عنوان یک هزینه مناسب برای کاهش توسط Optimizer مورد استفاده قرار میگیرند. کاهش این خطا یعنی متمایز کننده بهتر توانسته نمونه های واقعی را از غیر واقعی تمییز کند.

بخش مولد تنها نیاز دارد بخش متمایزکننده را فریب دهد بنابراین باید خروجی متمایزکننده در اثر دیدن ورودی ساخته شده توسط مولد، برچسب واقعی باشد. در این شرایط یعنی مولد توانسته متمایزکننده را فریب دهد. برای اینکار از **Binary Cross Entropy** با برچسب های تمام واقعی استفاده میکنیم.

سوال ۴)

در این سوال با مشکل زمان آموزش زیاد مواجه شدیم. آموزش هر کدام از مدل ها زمان زیادی نیاز داشت و سرویس Colab نیز هر چند ساعت یک بار دسترسی به GPU را میگرفت. این ها باعث شد که پروسه آزمون و خطا و تغییر پارامترها تقریبا غیرممکن شود. به همین دلیل این بخش از گزارش در حاضر انجام نشده است. ممکن است زمان ارائه حضوری، این بخش و سایر آزمایش های مربوط به این تمرین کامل شوند. در حال حاضر سعی شد نتیجه نسبتا قابل قبولی ارائه شود. که در بخش های بعدی خواهیم دید.

اما با توجه به دانشی که از افزایش لایه ها در طول کسب کرده ایم میتوان گفت که افزایش تعداد لایه ها، تا حدی عملکرد مدل مولد را بهبود میدهند و اجازه میدهند مدل مولد بتواند تصاویر پیچیده تری را تولید کند اما از حدی به بعد، تعداد پارامترها بسیار زیاد میشوند و روند یادگیری را کند میکنند و همچنین احتمال Overfitting را افزایش میدهند.

سوال ۵)

تعدادی از تکنیک‌هایی که برای آموزش بهتر شبکه های GAN استفاده می‌شوند را در ادامه بیان میکنیم.

۱. **اعمال نویز به داده های ورودی و برچسب ها:** اعمال نویز به تصاویر ورودی باعث می‌شوند متمایزکننده کار سختتری داشته باشد چون باید تصاویر دارای نویز را دسته بندی کند. با استفاده از این تکنیک، به طور مصنوعی سرعت یادگیری متمایزگر را کم کرده این گونه، مولد زمان بیشتری برای پیشرفت خواهد داشت.

۲. میتوانیم به جای اینکه در هر epoch هم مولد و هم متمایزگر را آموزش دهیم و وزن های آنها را به روز رسانی کنیم، این کار را با **فرکانس کمتری** برای متمایزکننده انجام دهیم. به عنوان مثال به ازای هر ده epoch که وزن های مولد به روز رسانی میشوند، وزن های متمایزکننده یکبار به روز رسانی شوند.

۳. یکی از ساده ترین راهها برای متعادل سازی روند آموزش این است که **نرخ یادگیری** کمتری برای متمایزکننده در نظر بگیریم. با این کار، نتیجه ای مشابه روش قبل خواهیم داشت به این شکل که هر چند epoch متمایزگر به اندازه یک epoch آموزش مولد اثر خواهد داشت.

سوال ۶)

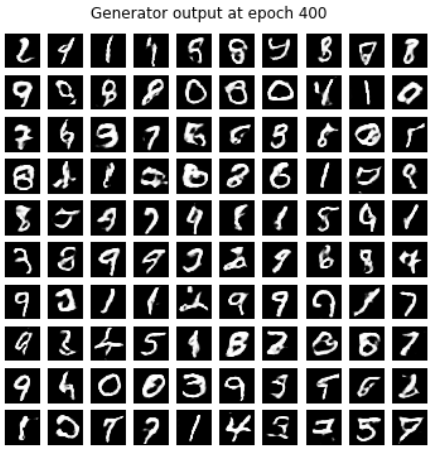
مانند بخش ۴ این بخش نیز به دلیل کمبود وقت انجام نشد زیرا حتی آزمایش های مربوط به مدل اصلی و گرفتن نتیجه اصلی نیز به خوبی به نتیجه ننشستند.

سوال ۷)

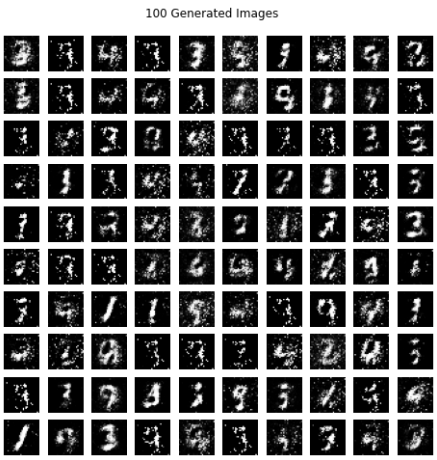
در مورد کار با تصاویر، همواره شبکه های کانولوشنی به دلیل در نظر گرفتن موقعیت مکانی محلی و ایده اعمال فیلترها از طریق کانولوشن، نتیجه بهتری نسبت به شبکه های تماما متصل بدست می‌آورند.

مسئله GAN نیز از این قائده مستثنی نیست. شبکه کانولوشنی در بخش متمایزکننده، میتواند با تعداد پارامتر کمتری و در تعداد Epoch کمتری به نتیجه بهتری برسد زیرا ویژگی هایی که این شبکه تولید میکندبرای تشخیص اشیا، بسیار پرکاربرد هستند. در بخش مولد نیز تبدیل کردن بردار نویز ورودی با استفاده از لایه های معکوس کانولوشنی هوشمند تر از لایه های تماما متصل عمل میکند و میتواند تصاویر بهتری تولید کند. در ادامه دو تصویر خواهیم دید. یکی با استفاده از شبکه FCGAN تولید شده و دیگری با استفاده از شبکه DCGAN میبینیم که تصویر تولید شده توسط شبکه FCGAN دارای نویز بیشتری است.

خروجی DCGAN



خروجی FCGAN

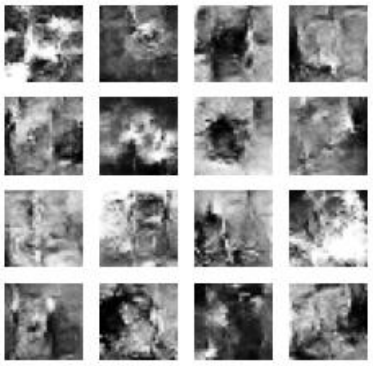


سوال ۸)

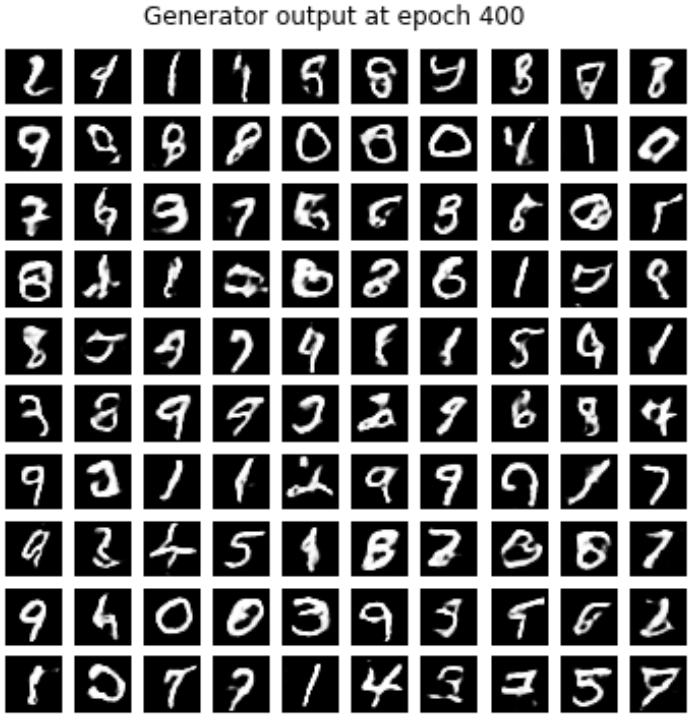
تعدادی از تصاویر را در بخش های قبل دیدید. در این قسمت تعدادی تصویر که بهترین(!) نتایج را تولید کردند را مشاهده میکنید.

تصاویر تولید شده توسط DCGAN پس از epoch۲۰۰۰

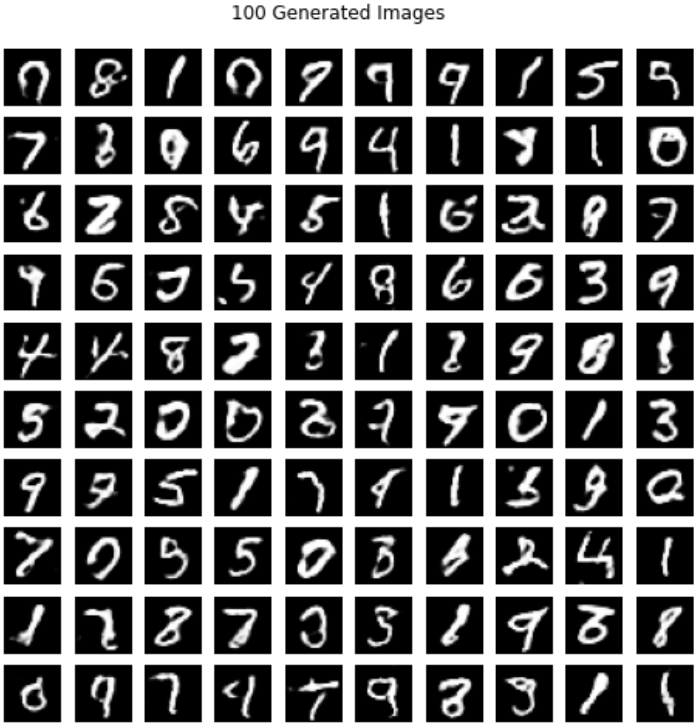
تعدادی از تصاویر تولید شده توسط DCGAN پس از ۱۰۰۰ epoch

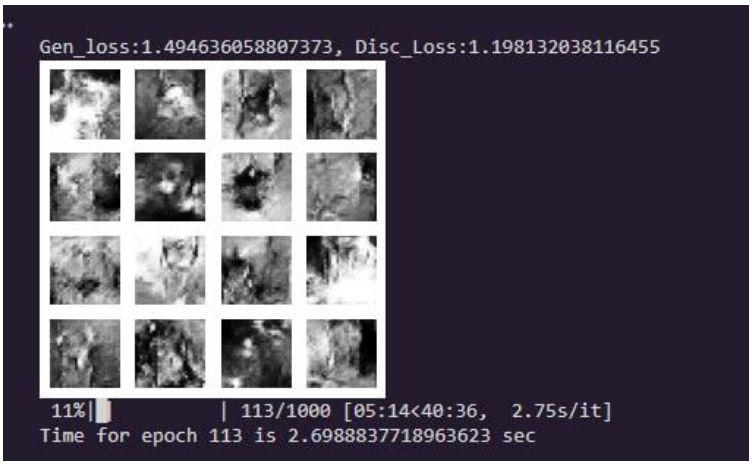


تصاویرتولید شده توسط DCGAN از دیتاست MNIST



تصاویر نمونه از دیتاست MNIST




Gen_loss:1.494636058807373, Disc_Loss:1.198132038116455
11%| | 113/1000 [05:14<40:36, 2.75s/it]
Time for epoch 113 is 2.6988837718963623 sec


Gen_loss:1.3102681636810303, Disc_Loss:1.0188560485839844
97%| | 971/1000 [45:07<01:19, 2.75s/it]
Time for epoch 971 is 2.706310749053955 sec