به نام خدا

پروژه ششم درس یادگیری عمیق
آشنایی با شبکه‌های مولد تقابلی

استاد درس: دکتر صفابخش

نگارش: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

زمستان ۱۴۰۲

فهرست مطالب

بخش‌اول: تئوری

الف)

معماری کلی شبکه مولد تقابلی (GAN) از دو شبکه عصبی تشکیل شده است:

- مولد (Generator): این شبکه وظیفه تولید داده‌های جدید را بر عهده دارد. مولد یک شبکه عصبی بازگشتی است که از ورودی‌های تصادفی برای تولید داده‌های جدید استفاده می‌کند. ورودی‌های تصادفی می‌توانند از توزیع نرمال یا توزیع گاوسی انتخاب شوند. خروجی مولد معمولاً یک تصویر، یک متن، یک صدای یا یک عدد است. نوع خروجی به کاربرد مورد نظر شبکه GAN بستگی دارد.
- متمایزگر (Discriminator): این شبکه وظیفه تشخیص داده‌های واقعی از داده‌های جعلی را بر عهده دارد. متمایزگر یک شبکه عصبی پیش‌رونده است که از ورودی‌های داده (اعم از واقعی یا جعلی) برای تشخیص داده‌های واقعی از داده‌های جعلی استفاده می‌کند. خروجی متمایزگر معمولاً یک عدد است که نشان می‌دهد داده ورودی واقعی است یا جعلی.

این دو شبکه به صورت رقابتی با یکدیگر آموزش می‌بینند. مولد سعی می‌کند داده‌های جعلی تولید کند که برای متمایزگر قابل تشخیص نباشد. متمایزگر نیز سعی می‌کند داده‌های واقعی را از داده‌های جعلی تشخیص دهد.

تابع هزینه مولد به گونه‌ای طراحی شده است که مولد سعی کند داده‌های جعلی تولید کند که برای متمایزگر قابل تشخیص نباشد و تابع هزینه متمایزگر به گونه‌ای طراحی شده است که متمایزگر سعی کند داده‌های واقعی را از داده‌های جعلی تشخیص دهد.

در هر مرحله از آموزش، تابع هزینه هر دو شبکه محاسبه می‌شود و سپس پارامترهای شبکه‌ها به گونه‌ای به‌روز می‌شوند که تابع هزینه کاهش یابد. این فرآیند به صورت تکراری انجام می‌شود تا زمانی که شبکه‌ها به حداکثر کارایی خود برسند.

ب)

سه مورد از مهم‌ترین مشکلات شبکه مولد تقابلی در زمان آموزش به شرح زیر است:

- **پایداری**

در برخی موارد، ممکن است شبکه‌ها در یک حالت ناپایدار گیر کنند و آموزش آن‌ها متوقف شود. این مشکل معمولاً زمانی رخ می‌دهد که یکی از دو شبکه، مولد یا متمایزگر، بیش از حد قوی شود.

- **نابودی**

در این حالت، یکی از دو شبکه، مولد یا متمایزگر، کاملاً شکست می‌خورد و دیگر قادر به انجام وظیفه خود نیست. این مشکل معمولاً زمانی رخ می‌دهد که یکی از دو شبکه، مولد یا متمایزگر، بیش از حد ضعیف شود.

- **فروافتادگی حالت**

در این حالت، مولد شروع به تولید داده‌هایی با ویژگی‌های تکراری می‌کند. این مشکل معمولاً زمانی رخ می‌دهد که مولد بیش از حد به داده‌های آموزشی محدود شود.

ج)

WGAN ها با استفاده از یک تابع هزینه جدید به نام تابع هزینه واسرستاین برای آموزش شبکه های مولد طراحی شده اند. تابع هزینه واسرستاین از یک شبکه عصبی متمایزگر استفاده می کند تا تفاوت بین توزیع داده های واقعی و توزیع داده های تولید شده توسط مولد را اندازه گیری کند.

شبکه متمایزگر در یک شبکه WGAN وظیفه تشخیص داده های واقعی از داده های جعلی را بر عهده دارد. تابع هزینه واسرستاین به گونه ای طراحی شده است که متمایزگر را تشویق کند تا توزیع داده های تولید شده توسط مولد را به توزیع داده های واقعی نزدیک کند.

مولد در یک شبکه WGAN وظیفه تولید داده هایی را دارد که برای متمایزگر قابل تشخیص نباشد. تابع هزینه واسرستاین به گونه ای طراحی شده است که مولد را تشویق کند تا توزیع داده های تولید شده خود را از توزیع داده های واقعی دور کند.

در هر مرحله از آموزش، شبکه متمایزگر و مولد به طور همزمان به روز می شوند. شبکه متمایزگر به گونه ای به روز می شود که بتواند داده های واقعی را از داده های جعلی با دقت بیشتری تشخیص دهد. شبکه مولد به گونه ای به روز می شود که بتواند داده هایی را تولید کند که برای متمایزگر قابل تشخیص نباشد.

شبکه های WGAN در مقایسه با شبکه های GAN سنتی، پایداری بیشتری در آموزش دارند. این امر به دلیل استفاده از تابع هزینه واسرستاین است که از بیش از حد قوی شدن شبکه متمایزگر جلوگیری می کند. WGAN ها همچنین کمتر مستعد فروافتادگی حالت هستند، زیرا از یک شبکه متمایزگر غیرخطی استفاده می کنند.

WGAN ها مشکلات زیر را در شبکه های GAN سنتی حل می کنند:

- پایداری: WGAN ها پایداری بیشتری در آموزش نسبت به شبکه های GAN سنتی دارند. این امر به دلیل استفاده از تابع هزینه واسرستاین است که از بیش از حد قوی شدن شبکه متمایزگر جلوگیری می کند.
- نابودی: WGAN ها کمتر مستعد نابودی هستند، زیرا از یک شبکه متمایزگر غیرخطی استفاده می کنند.
- فروافتادگی حالت: WGAN ها کمتر مستعد فروافتادگی حالت هستند، زیرا از یک شبکه متمایزگر غیرخطی استفاده می کنند.

د)

PGGAN ها در مقایسه با شبکه های GAN سنتی، پایداری بیشتری در آموزش دارند. این امر به دلیل استفاده از تکنیک رشد تدریجی است (شبکه مولد به تدریج با افزودن لایه های جدید به آن، بزرگتر و پیچیده تر می شود. این تکنیک باعث

می شود که شبکه مولد بتواند به تدریج از داده های آموزشی یاد بگیرد و داده های تولید شده با کیفیت بالاتری تولید کند) که از بیش از حد قوی شدن شبکه متمایزگر جلوگیری می کند. PGGAN ها همچنین کمتر مستعد فروافتادگی حالت هستند، زیرا از یک شبکه متمایزگر غیرخطی استفاده می کنند.

PGGAN ها برای بهبود تولید تصاویر با کیفیت بالا، کارهای زیر را انجام می دهند:

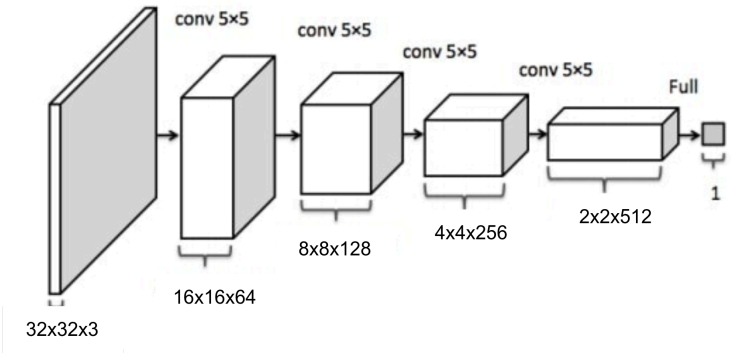
- توانایی تولید تصاویر با کیفیت بالاتر: PGGAN ها می توانند تصاویری تولید کنند که از نظر وضوح، جزئیات و واقع گرایی از تصاویر تولید شده توسط شبکه های GAN سنتی بهتر هستند. این به دلیل استفاده از شبکه مولد بزرگتر و پیچیده تر است که می تواند از داده های آموزشی دقیق تر یاد بگیرد.
- توانایی تولید تصاویر متنوع تر: PGGAN ها می توانند تصاویری تولید کنند که از نظر سبک، موضوع و محتوا متنوع تر از تصاویر تولید شده توسط شبکه های GAN سنتی هستند. این به دلیل استفاده از شبکه مولد بزرگتر و پیچیده تر است که می تواند از داده های آموزشی متنوع تری یاد بگیرد.
- توانایی تولید تصاویر از ویژگی های داده های آموزشی: PGGAN ها می توانند تصاویری تولید کنند که از ویژگی های داده های آموزشی استفاده می کنند. این به دلیل استفاده از شبکه مولد بزرگتر و پیچیده تر است که می تواند از داده های آموزشی دقیق تر یاد بگیرد.

بخش‌دوم: پیاده سازی

الف)

کلاس DCDiscriminator از چهار لایه کانولوشنی تشکیل شده است، همه لایه‌ها اگر (spectral_norm=True) از نرمال‌سازی طیفی برای بهبود پایداری آموزش استفاده می‌کنند. ساختار لایه‌ها به صورت زیر است(این لایه‌ها به تدریج تعداد کانال‌ها را افزایش می‌دهند تا ویژگی‌های پیچیده‌تری از ورودی را استخراج کنند و از فیلترهای نسبتاً بزرگ (5 در 5) و گام‌های بزرگ (2) برای کاهش سریع اندازه خروجی استفاده می‌شود):

1. self.conv1:
   - ورودی را از 3 کانال (تصاویر RGB) به conv_dim کانال تبدیل می‌کند.
   - از یک فیلتر 5 در 5 با گام 2 برای کاهش اندازه خروجی استفاده می‌کند.
2. self.conv2:
   - ورودی را از conv_dim کانال به conv_dim*2 کانال افزایش می‌دهد.
   - از یک فیلتر 5 در 5 با گام 2 برای کاهش بیشتر اندازه خروجی استفاده می‌کند.
3. self.conv3:
   - ورودی را از conv_dim*2 کانال به conv_dim*4 کانال افزایش می‌دهد.
   - از یک فیلتر 5 در 5 با گام 2 برای کاهش مجدد اندازه خروجی استفاده می‌کند.
4. self.conv4:
   - ورودی را از conv_dim*4 کانال به 1 کانال کاهش می‌دهد (احتمالاً برای خروجی تک کاناله).
   - از یک فیلتر 5 در 5 با گام 2 و پدگذاری 1 برای حفظ ابعاد فضایی استفاده می‌کند.
   - برخلاف لایه‌های قبلی، از نرمال‌سازی دسته‌ای استفاده نمی‌کند (batch_norm=False).



ب)

کد تکمیل شده شبکه مولد به صورت زیر است:

```
class DCGenerator(nn.Module):
    def __init__(self, noise_size, conv_dim, spectral_norm=False):
        super(DCGenerator, self).__init__()

        self.conv_dim = conv_dim

        self.linear_bn = nn.Sequential(nn.Linear(in_features=noise_size, out_features=conv_dim*4*4*4), nn.Flatten())

        self.upconv1 =  upconv(in_channels=conv_dim*4, out_channels=conv_dim*2,
                               kernel_size=5, stride=2, padding=2, spectral_norm=spectral_norm)
        self.upconv2 = upconv(in_channels=conv_dim*2, out_channels=conv_dim,
                               kernel_size=5, stride=2, padding=2, spectral_norm=spectral_norm)
        self.upconv3 = upconv(in_channels=conv_dim, out_channels=3,
                               kernel_size=5, stride=2, padding=2, batch_norm=False, spectral_norm=spectral_norm)
```

self.linear_bn: یک ماژول ترتیبی است که نویز ورودی (احتمالاً یک بردار با noise_size ویژگی) را به یک تنسوری با conv_dim*4*4*4 عنصر تبدیل می‌کند.

شبکه مولد از سه لایه کانولوشنی تشکیل شده است که به صورت زیر است:

1. Self.upconv1:
   - ورودی را با ضریب 2 افزایش می‌دهد.
   - تعداد کانال‌ها را از conv_dim*4 به conv_dim*2 کاهش می‌دهد.
   - از یک فیلتر 5 در 5، گام 1، پدگذاری 2 و نرمال‌سازی طیفی استفاده می‌کند.
2. Self.upconv2:
   - ورودی را با ضریب 2 افزایش می‌دهد.
   - تعداد کانال‌ها را از conv_dim*2 به conv_dim کاهش می‌دهد.
   - تنظیمات مشابه upconv1 را استفاده می‌کند.
3. Self.upconv3:
   - ورودی را با ضریب 2 افزایش می‌دهد.
   - خروجی نهایی را با 3 کانال (تصاویر RGB) تولید می‌کند.
   - تنظیمات مشابه را استفاده می‌کند اما نرمال‌سازی دسته‌ای را حذف می‌کند.

ج)

قسمت های خواسته شده در فایل train.py مطابق رابطه داده شده به صورت زیر تکمیل شده است:

```
        # 1. Compute the discriminator loss on real images
        D_out_real = D(real_images).type(torch.FloatTensor)
        ones = torch.ones(size=real_labels.size()).type(torch.FloatTensor)
        ones.requires_grad = False
```

```
            zeros = torch.zeros(size=real_labels.size()).type(torch.FloatTensor)
            zeros.requires_grad = False

            D_real_loss = loss(D_out_real, zeros)
            D_real_loss = D_real_loss.type(torch.FloatTensor)*(1/D_out_real.shape[0])
            real_labels = real_labels.type(torch.FloatTensor)
```

در رابطه بالا مقدار loss برای تصاویر واقعی محاسبه می‌شود.

```
        # 2. Sample noise
            noise = sample_noise(batch_size=opts.batch_size, dim=opts.noise_size)
        # 3. Generate fake images from the noise
            fake_images = G(noise)

            # 4. Compute the discriminator loss on the fake images
            D_out_fake = D(fake_images).type(torch.FloatTensor)
            ones = torch.ones(size=D_out_fake.shape).type(torch.FloatTensor)
            ones.requires_grad = False
            D_fake_loss =loss(D_out_fake, ones)
            D_fake_loss = D_fake_loss.type(torch.FloatTensor)*(1/D_out_fake.shape[0])
```

در روابط بالا تصویر غیرواقعی تولید می‌شود و مقدار loss برای آنها محاسبه می شود.

```


            # 5. Compute the total discriminator loss
            D_total_loss = D_fake_loss + D_real_loss
            D_total_loss = D_total_loss.type(torch.FloatTensor)
            D_total_loss.backward()
            d_optimizer.step()
```

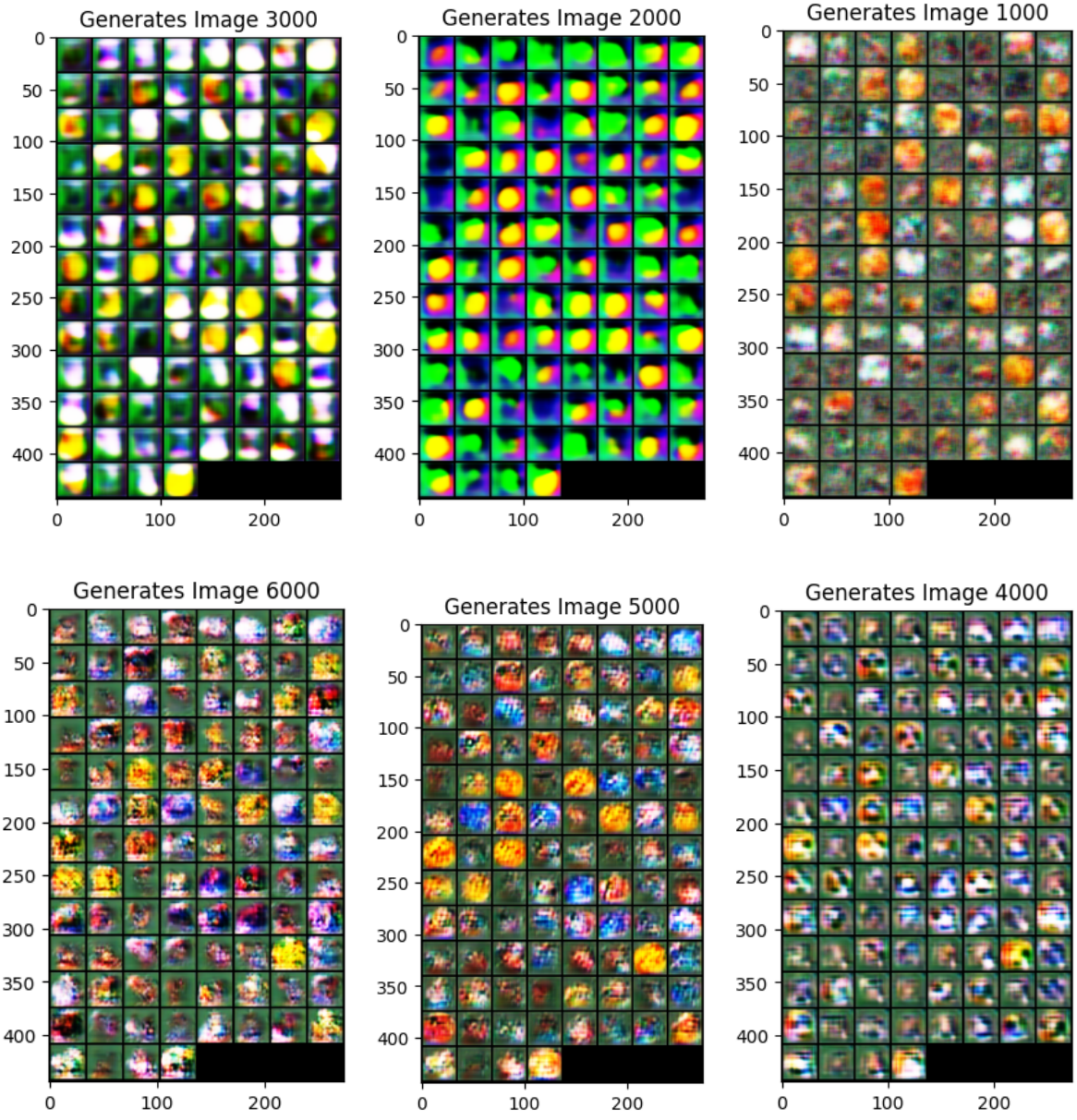
در رابطه بالا مقدار کلی تابع loss برای قسمت متمایزگر محاسبه می‌شود و شبکه طبق مقدار محاسبه شده، به روز می‌شود.

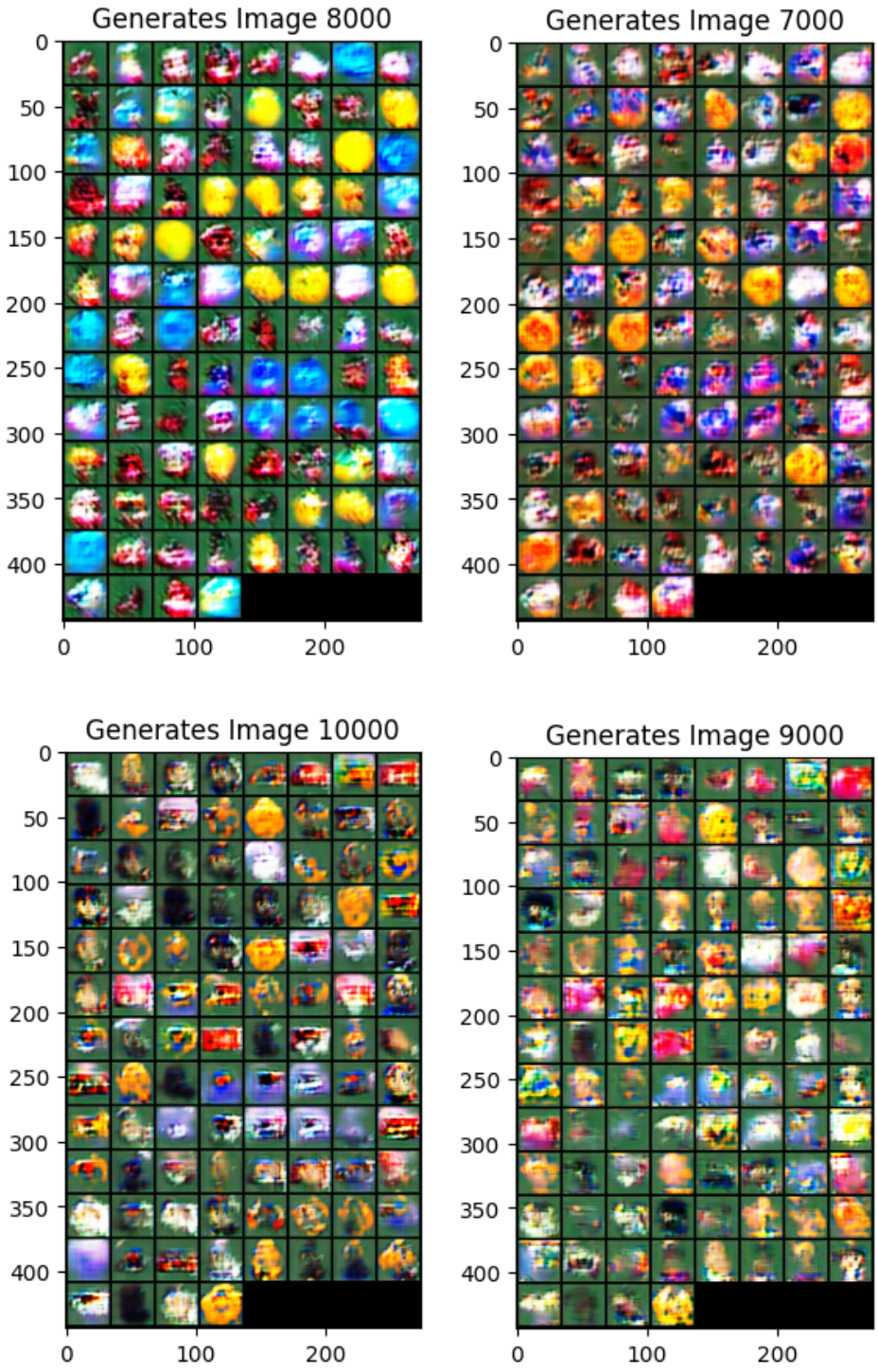
```
        # 1. Sample noise
        noise = sample_noise(batch_size=opts.batch_size, dim=opts.noise_size)

        # 2. Generate fake images from the noise
        fake_images = G(noise)

        # 3. Compute the generator loss
        D_out_fake1 = D(fake_images).type(torch.FloatTensor)
        zeros = torch.zeros(size=D_out_fake1.shape).type(torch.FloatTensor)
        zeros.requires_grad = False
        G_loss = loss(D_out_fake1, zeros)
        G_loss = G_loss.type(torch.FloatTensor)*(1/D_out_fake1.shape[0])

        G_loss.backward()
        g_optimizer.step()
```

در قسمت بالا مقدار کلی تابع لاس براس شبکه مولد محاسبه شده و شبکه بر اساس آن آموزش می‌بینند.

در دو شبکه بالا برای تابع لاس از MseLoss استفاده شده است و مقدار به دست آمده بر تعداد تقسیم شده است ا در نهایت به رابطه داده شده در سوال برسیم.

د)


Generates Image 3000
Generates Image 2000
Generates Image 1000
Generates Image 6000
Generates Image 5000
Generates Image 4000

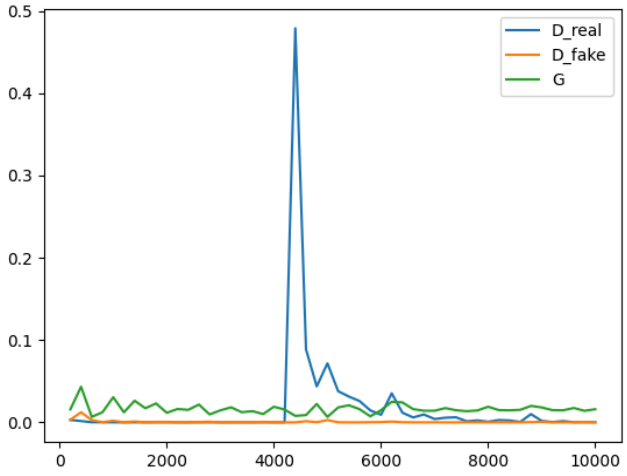


تصاویر بالا نشان دهنده خروجی مدل به ازای هر 1000 دوره می باشد، در روند آموزش شبکه مولد تقابلی، کیفیت تصاویر تولید شده به تدریج بهبود می یابد که به دلیل یادگیری شبکه مولد از داده های آموزشی است. در ابتدا، شبکه مولد تصاویری تولید می کند که دارای وضوح پایین، جزئیات کم و واقع گرایی ضعیف هستند. با گذشت زمان، شبکه مولد شروع به یادگیری

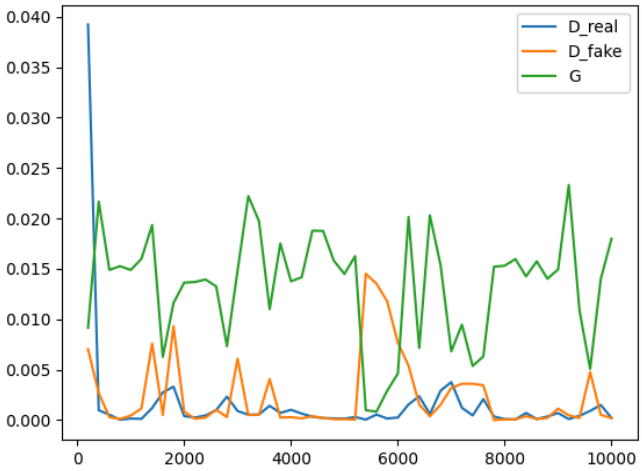
الگوهای پیچیده تر موجود در داده های آموزشی می کند. این باعث می شود که تصاویر تولید شده توسط شبکه مولد واضح تر، دقیق تر و واقع گرایانه تر شوند.

۵)

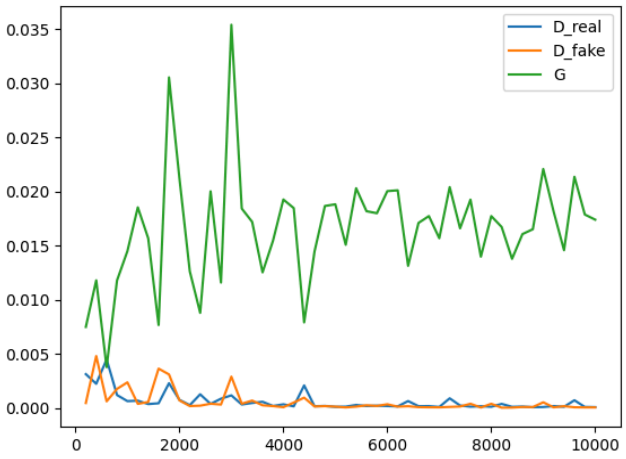
تاثیر نرخ یادگیری:



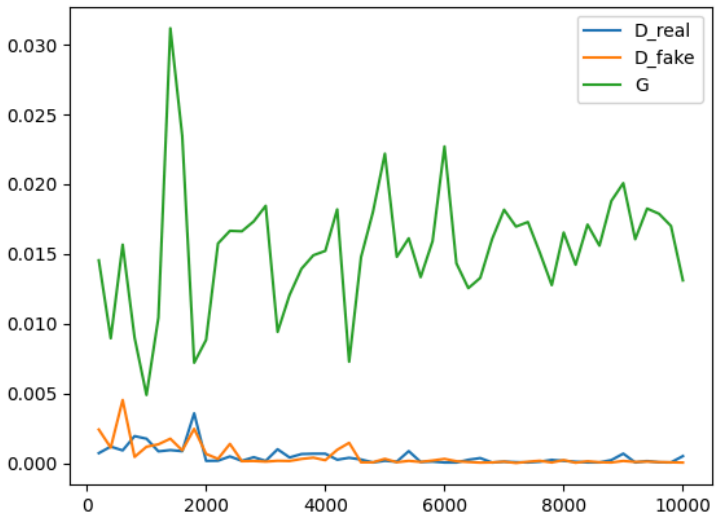
lr=0.1



lr=0.01



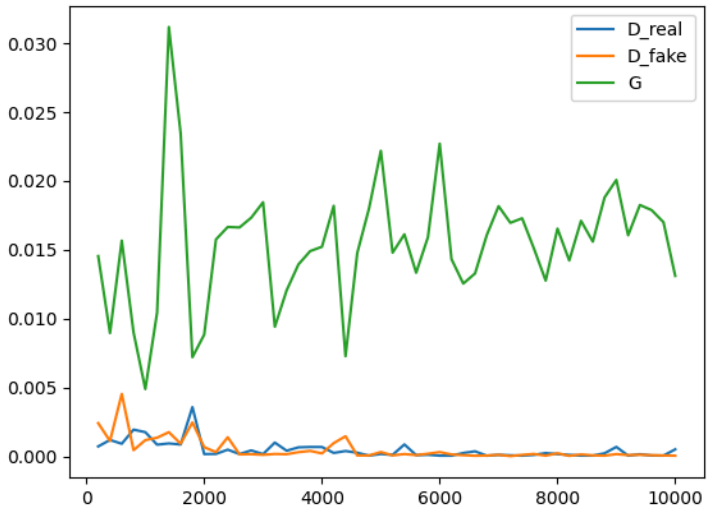
lr=0.001



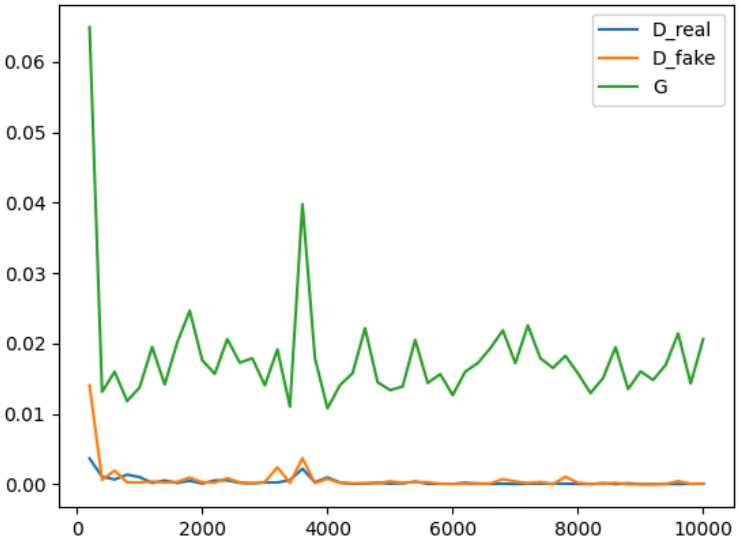
lr=0.0005

نرخ یادگیری می‌تواند بر پایداری آموزش و جلوگیری از بیش‌برازش تاثیر بگذارد. نرخ یادگیری بالاتر منجر به سرعت یادگیری سریع‌تر، اما نوسانات بیش از حد در فرآیند یادگیری و افزایش احتمال بیش‌برازش می‌شود. نرخ یادگیری پایین‌تر می‌تواند پایدارتر باشد، اما ممکن است منجر به یادگیری کندتر شود.

تاثیر spectral_norm:



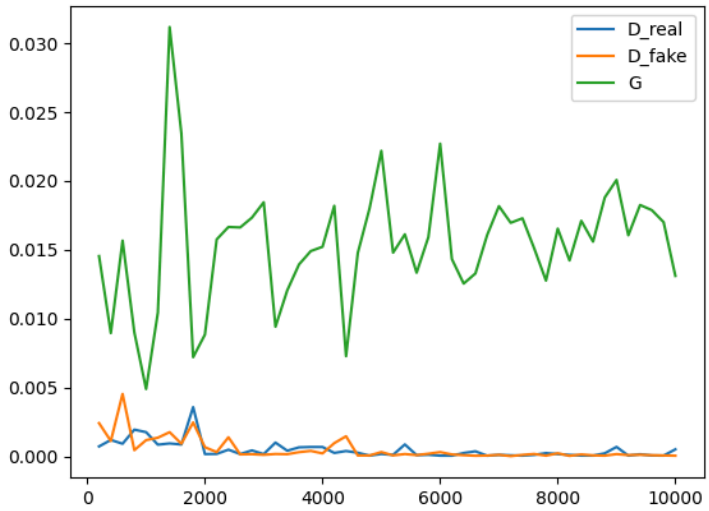
spectral_norm=false



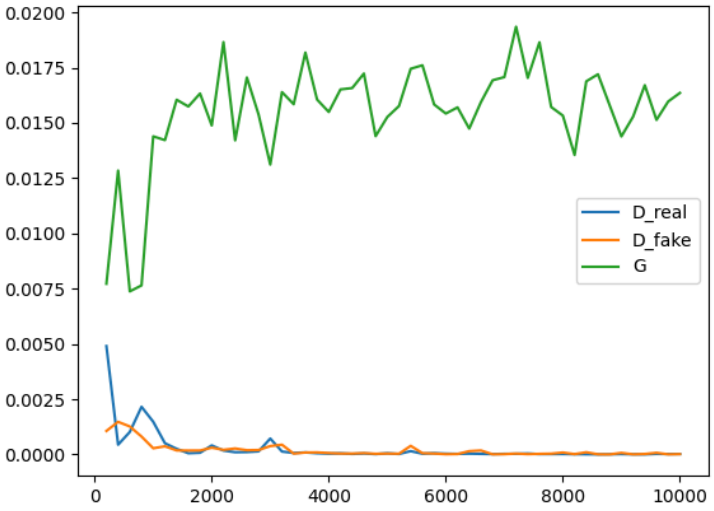
spectral_norm=true

نرمال‌سازی طیفی منجر به پایداری آموزش و جلوگیری از بیش‌برازش می‌شود، زیرا می‌تواند به جلوگیری از نوسانات بیش از حد در فرآیند یادگیری کمک کند. نرمال‌سازی طیفی با محدود کردن دامنه مقادیر وزن‌های شبکه، از بیش‌برازش جلوگیری می‌کند. این کار را با محاسبه مقدار طیف ویژه بزرگترین مقدار ویژه وزن‌های شبکه انجام می‌دهد. سپس، این مقدار طیف ویژه به عنوان یک محدودیت برای وزن‌ها استفاده می‌شود.

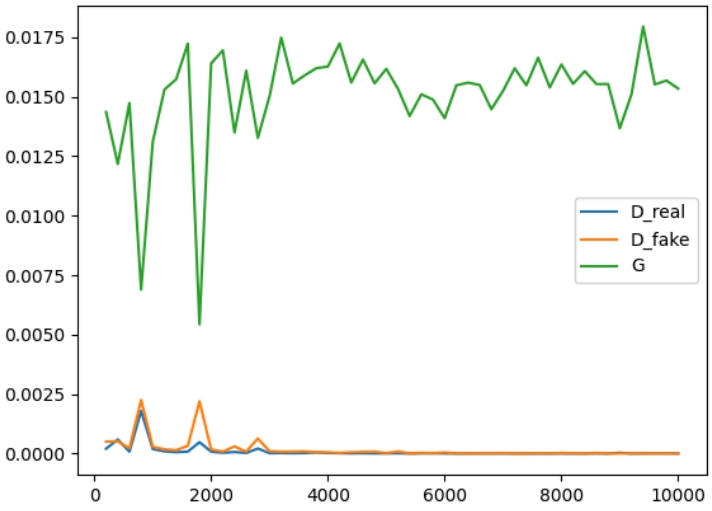
تاثیر d_train_iters:



d_train_iters =1



d_train_iters =2



d_train_iters =4

این پارامتر تعداد بروزرسانی وزن‌های شبکه تمایزگر به ازای هر بار به‌روز رسانی شبکه مولد را نشان می‌دهد، با افزایش مقدارd_train_iters در شبکه تمایزگز loss کاهش می‌یابد و شبکه تمایزگر بهتر آموزش می‌بیند.

تعداد تکرارهای بیشتر برای شبکه تمایزگر منجر به بهبود عملکرد شبکه تمایزگر در تشخیص تصاویر جعلی شود که منجر به بهبود کیفیت تصاویر تولید شده توسط شبکه مولد شود. با این حال، تعداد تکرارهای زیاد برای شبکه تمایزگر می‌تواند منجر به بیش‌برازش شود. این امر به این دلیل است که شبکه تمایزگر می‌تواند به طور خاص برای تشخیص تصاویر تولید شده توسط شبکه مولد برازش شود.