به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین ششم

نام و نام خانوادگی	بهراد موسایی شیرمحمد - محمد جواد رنجبر کلهرودی
شماره دانشجویی	۸۱۰۱۰۱۲۷۸ - ۸۱۰۱۰۱۱۷۳
تاریخ ارسال گزارش	۱۴۰۲.۱۰.۳۰

فهرست

فهرست تصاویر

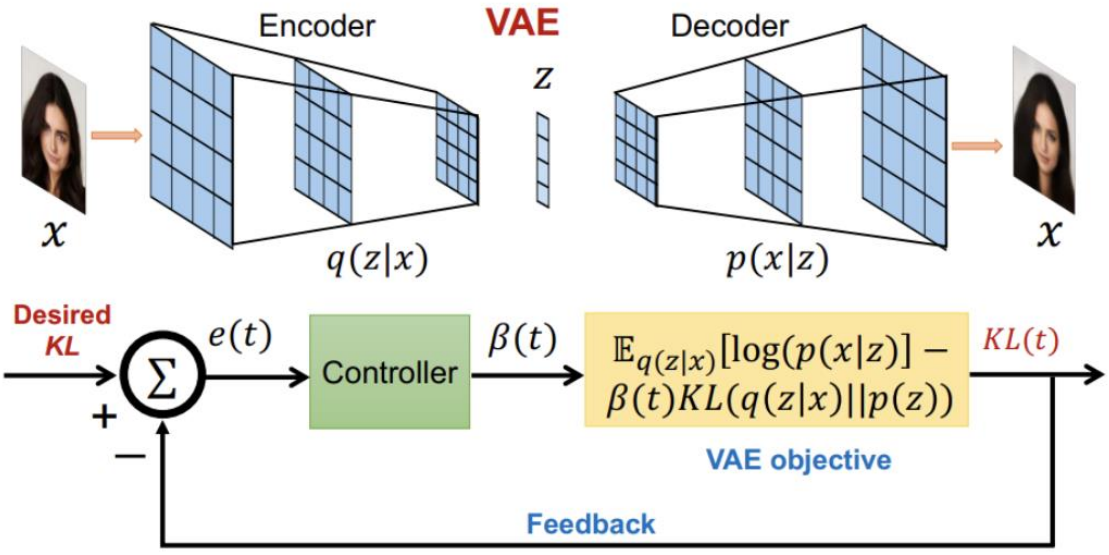
پاسخ ۱. Control VAE

۱-۱. مقدمه

مقاله "ControlVAE: رمزگذار خودکار متغیر قابل کنترل" یک نسخه پیشرفته از رمزگذار خودکار متغیر سنتی VAE را مورد بحث قرار می دهد. این مدل جدید، ControlVAE، یک کنترل کننده بر اساس تئوری کنترل خودکار را با چارچوب استاندارد VAE ادغام می کند. هدف این ادغام بهبود عملکرد مدل‌های تولیدی مشتق‌شده از VAE است.

ControlVAE با تنظیم خودکار یک فراپارامتر در هدف VAE با استفاده از واگرایی خروجی KL به عنوان بازخورد در طول آموزش، محدودیت‌های مدل‌های سنتی VAE را برطرف می‌کند. این رویکرد به متعادل کردن کیفیت بازسازی داده ها با الزامات کاربردی خاص، مانند تنوع خروجی یا نمایش جدا شده کمک می کند. این مقاله ControlVAE را در برنامه‌های مختلف، از جمله مدل‌سازی زبان، یادگیری بازنمایی جداشده، و تولید تصویر ارزیابی می‌کند و توانایی آن را برای دستیابی به کیفیت بازسازی بهتر در حین حفظ یا افزایش سایر معیارهای عملکرد نشان می‌دهد.

با کنترل واگرایی KL، ControlVAE به طور موثری مبادله بین دقت بازسازی و تنوع یا از هم گسیختگی خروجی های تولید شده را مدیریت می کند و ابزاری قابل انطباق برای کاربردهای مختلف VAE ارائه می دهد.



شکل ۱: معماری مدل **control VAE**

برای آموزش مدل از بخشی از مجموعه داده D shape prite۲ استفاده شده است؛ به این صورت که از ویژگی orientation در این مجموعه داده مقدار صفر آن انتخاب شده است که از کل داده ها است و در نهایت ۱۸۴۳۲ داده خواهیم داشت این مجموعه داده کوچک شده را میتوانید از این پیوند دریافت کنید. تصاویر این مجموعه داده با اندازه ۶۴ در ۶۴ و تک کاناله هستند و مقادیر پیکسلهای آن به صورت باینری ۰ و ۱ است.

۱-۲. پیاده سازی VAE

در این بخش با توجه به جداول داده شده به پیاده سازی مدل می پردازیم:

۱. آماده سازی مجموعه داده:

- یک مجموعه داده سفارشی "CustomDataset" از یک فایل NPZ حاوی داده های تصویر "dsprites_ndarray_co۱sh۳sc۶or۴۰x۳۲y۳۲_۶۴x۶۴.npz" ایجاد می شود.
- برای پردازش کارآمد، مجموعه داده با ضریب کاهش ۱۰ نمونه برداری می شود.

۲. تعریف مدل VAE:

- یک مدل رمزگذار خودکار متغیر VAE با معماری رمزگذار و رمزگشا تعریف شده است.
- رمزگذار از لایه های کانولوشن تشکیل شده است که به دنبال آن لایه های کاملاً متصل هستند که تصاویر ورودی را در فضای پنهانی با اندازه ۱۰ برای میانگین و ۱۰ برای log-variance رمزگذاری می کنند.
- رمزگشا نقاط را از فضای نهفته به فضای تصویر نگاشت می کند.

۳. آموزش:

- VAE برای ۱۰ دوره با استفاده از بهینه ساز Adam آموزش داده شده است.
- از دست دادن آموزش، از جمله از دست دادن واگرایی KL و از دست دادن بازسازی، نظارت می شود و در طول دوره ها ترسیم می شود.
- حافظه به صورت دوره ای برای مدیریت مصرف حافظه GPU آزاد می شود.

۴. تولید تصویر:

- پس از آموزش، VAE در حالت ارزیابی قرار می گیرد و ۵۰ بردار نهفته تصادفی تولید می شود.

- این بردارهای پنهان برای تولید ۵۰ تصویر مصنوعی رمزگشایی می شوند که سپس در یک شبکه۱۰ در ۵ نمایش داده می شوند.

۵. محاسبه FID:

- مسیری به دایرکتوری حاوی تصاویر واقعی برای محاسبه FID مشخص شده است 'path_to_real_images'.

- تصاویر واقعی از دایرکتوری با استفاده از "ImageFolder" بارگیری می شوند و به تانسور تبدیل می شوند.

- امتیاز FID با استفاده از کتابخانه "pytorch_fid" محاسبه می شود و مجموعه داده های تصاویر واقعی و تصاویر تولید شده از VAE را مقایسه می کند.

۶. امتیاز FID گزارش شده :

- امتیاز FID که نشان دهنده شباهت بین تصاویر واقعی و تولید شده است، در انتهای فیلمنامه محاسبه و چاپ می شود.

نتایج:

- کد با موفقیت یک VAE را روی مجموعه داده سفارشی آموزش می دهد.

- از دست دادن واگرایی و بازسازی KL در طول دوره ها ردیابی و نمایش داده می شود که به نظارت بر روند آموزش کمک می کند.

- تصاویر مصنوعی با استفاده از VAE آموزش دیده تولید و در یک شبکه نمایش داده می شوند.

- امتیاز FID برای ارزیابی کیفیت تصاویر تولید شده در مقایسه با تصاویر واقعی محاسبه می شود.

۱-۳. ارزیابی مدل VAE

FID Score

فاصله اولیه فریشت FID یک معیار محبوب است که برای ارزیابی کیفیت و تنوع تصاویر تولید شده توسط مدل‌های تولیدی، مانند شبکه‌های متخاصم GAN یا رمزگذارهای خودکار متغیر VAE استفاده می‌شود. این شباهت بین توزیع بردارهای ویژگی استخراج شده از تصاویر واقعی و تولید شده را با استفاده از یک مدل InceptionV۳ که روی یک مجموعه داده بزرگ از قبل آموزش داده شده است، اندازه گیری می کند. امتیاز FID اندازه گیری کمی از نزدیک بودن تصاویر تولید شده به تصاویر واقعی از نظر کیفیت و تنوع بصری را ارائه می دهد.

در اینجا توضیح گام به گام نحوه محاسبه امتیاز FID آورده شده است:

۱. مجموعه داده تصاویر واقعی را آماده کنید :

- شما باید مجموعه داده ای از تصاویر واقعی داشته باشید که نمایانگر نوع تصاویری باشد که مدل تولیدی شما قصد تولید آن را دارد. این تصاویر واقعی به عنوان مرجع مقایسه عمل خواهند کرد.

۲. تصاویر مصنوعی تولید کنید:

- مدل تولیدی شما، مانند GAN یا VAE، باید مجموعه‌ای از تصاویر مصنوعی تولید کند که می‌خواهید ارزیابی کنید.

۳. پیش پردازش تصاویر:

- تصاویر واقعی و تولید شده باید از قبل پردازش شوند تا اطمینان حاصل شود که فرمت و اندازه یکسانی دارند که در مدل InceptionV۳ انتظار می رود. به طور معمول، این شامل تغییر اندازه تصاویر به اندازه ثابت مثلاً x۲۹۹۲۹۹ پیکسل و عادی سازی مقادیر پیکسل است.

۴. یک مدل InceptionV۳ از پیش آموزش دیده را بارگیری کنید:

- شما به یک مدل InceptionV۳ از قبل آموزش دیده نیاز دارید، یک شبکه عصبی کانولوشن عمیق که به طور گسترده برای کارهای طبقه بندی تصاویر استفاده می شود. این مدل برای استخراج بردارهای ویژگی از تصاویر استفاده می شود.

۵. ویژگی های استخراج :

- استخراج ویژگی‌ها جاسازی‌ها از تصاویر واقعی و تولید شده با استفاده از مدل InceptionV۳. این ویژگی ها نشان دهنده اطلاعات سطح بالا در مورد محتوای تصاویر است.

۶. محاسبه امتیاز FID:

- امتیاز FID بر اساس بردارهای ویژگی استخراج شده محاسبه می شود. این شامل محاسبه میانگین و کوواریانس ویژگی‌ها از هر دو مجموعه تصاویر و سپس محاسبه فاصله فرچه بین این توزیع‌های گاوسی چند متغیره است. فرمول FID:

$$FID = \|\mu_r - \mu_g\|^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

شکل ۲: فرمول **FID**

- در این فرمول:

- μ۱ و μ۲ به ترتیب میانگین بردارهای ویژگی از تصاویر واقعی و تولید شده هستند.

- Σ۱ و Σ۲ به ترتیب کوواریانس بردارهای ویژگی از تصاویر واقعی و تولید شده هستند.

- trΣ۱Σ۲ اثر حاصلضرب دو ماتریس کوواریانس است.

- ||μ۱ - μ۲||$_2^2$ مجذور فاصله اقلیدسی بین میانگین ها است.

- امتیاز FID پایین تر نشان می دهد که توزیع بردارهای ویژگی از تصاویر واقعی و تولید شده نزدیک تر است، که به معنای کیفیت و تنوع تصویر بهتر است.

جادوگران امتیاز FID بالا نشان می دهد که تفاوت های قابل توجهی بین تصاویر واقعی و تولید شده وجود دارد.

۷. تکرار برای اجرای چندگانه:

- این یک تمرین خوب است که امتیاز FID را برای چندین اجرا از مدل تولیدی خود محاسبه کنید و امتیازات را میانگین بگیرید تا ارزیابی قوی‌تری داشته باشید.

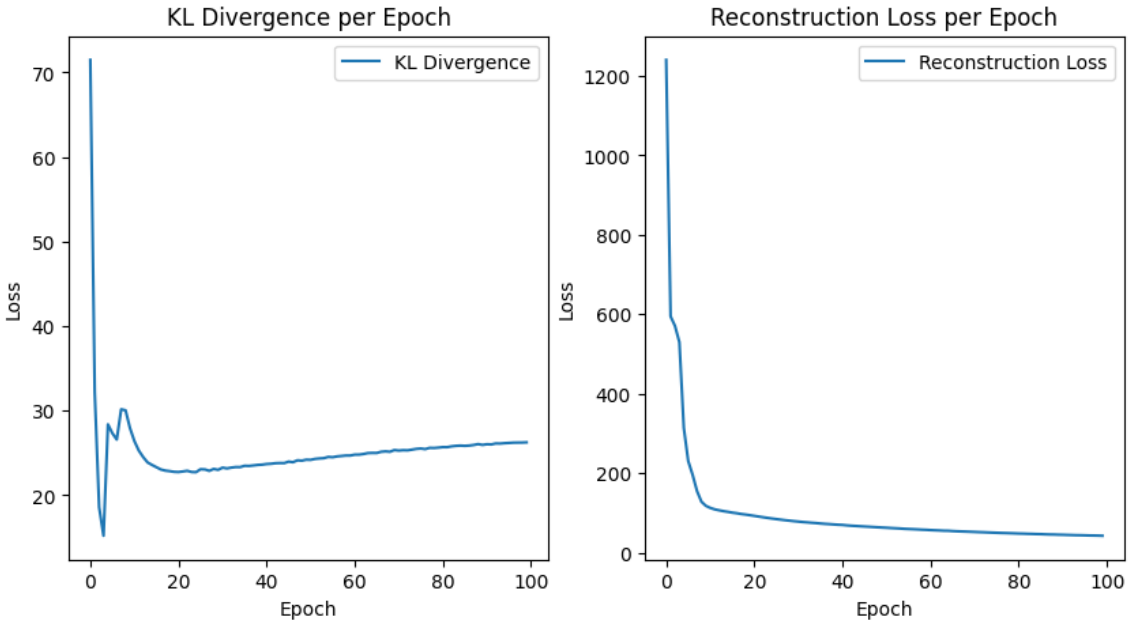
امتیاز FID یک معیار ارزشمند برای ارزیابی کمی عملکرد مدل‌های تولیدی است، زیرا هم کیفیت و هم تنوع تصاویر تولید شده را در نظر می‌گیرد. این یک مرجع مفید برای مقایسه مدل های مختلف و تنظیم دقیق پارامترهای آنها برای تولید نتایج واقعی تر و متنوع تر است.

مراتب پیاده سازی این کد به صورت زیر ارائه می شود:

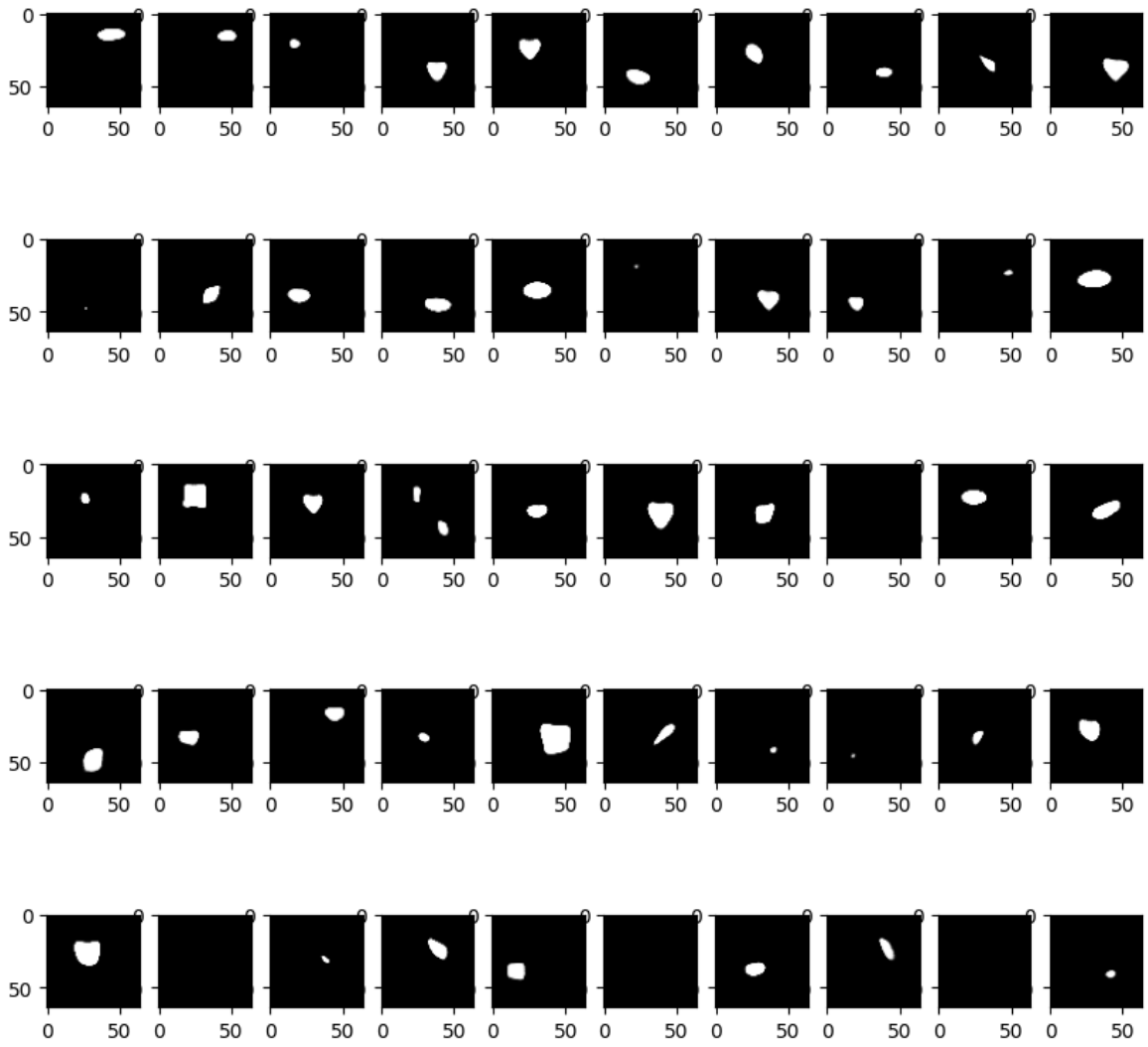
در این بخش یک رمزگذار خودکار متغیر VAE را در PyTorch برای تولید تصاویر جدید بر اساس مجموعه داده‌های موجود ارائه می‌کند. بخش اول کد، واردات و پیکربندی های لازم، از جمله راه اندازی GPU برای محاسبات در صورت وجود را ایجاد می کند. یک کلاس مجموعه داده سفارشی، «CustomDataset»، برای بارگیری و پیش پردازش تصاویر از یک فایل «.npz» مشخص شده است. اندازه مجموعه داده با یک عامل مثلاً ۱۰۰ برای کارایی کاهش می یابد. جزء اصلی کلاس `VAE` است که از یک رمزگذار و یک رمزگشا تشکیل شده است. رمزگذار از لایه های کانولوشن برای فشرده سازی تصاویر ورودی در یک نمایش فضای پنهان استفاده می کند، در حالی که رمزگشا تصاویر را از فضای پنهان بازسازی می کند. روش "reparameterize" در کلاس VAE برای جنبه تغییرات بسیار مهم است و تصادفی را به نمایش نهفته اضافه می کند.

حلقه آموزشی VAE شامل از دست دادن بازسازی و واگرایی KL است که برایVAE ها معمول است تا هم بازسازی دقیق و هم ساختار فضای نهفته معنادار را تضمین کند. این کد همچنین دارای عملکردی برای تجسم فرآیند آموزش و نتایج خروجی است، مانند ذخیره شبکه‌های تصاویر و ترسیم منحنی‌های از دست دادن. علاوه بر این، کد دارای یک تابع «calculate_fid» برای محاسبه امتیاز فاصله اولیه فریشت FIDاست، که معیاری برای ارزیابی کیفیت تصاویر تولید شده در مقایسه با تصاویر واقعی است. این معیار از مدل InceptionV۳ برای استخراج ویژگی ها از تصاویر واقعی و تولید شده استفاده می کند و سپس فاصله بین توزیع های آنها را محاسبه می کند. وجود محاسبه امتیاز FID نشان دهنده تاکید بر ارزیابی کیفیت تصاویر تولید شده است. به طور کلی، این کد یک پیاده سازی جامع از یک VAE برای تولید و ارزیابی تصویر است که برای آموزش، تجسم و ارزیابی کیفیت به خوبی ساختار یافته است.

حال برای خروجی داریم این مدل داریم:



شکل:۳ نتایج مربوط به **VAE**



شکل ۴: خروجی اپیاک آخر **VAE**

همچنین برای امتیاز FID داریم :

FID SCORE = 89.598342

۴-۱. پیاده سازی Control VAE

برای افزودن کنترلر PI به مدل VAE Variational Autoencoder به منظور پیاده‌سازی Control VAE، می‌توانید از الگوریتم ۱ در مقاله‌ای که اشاره کردید پیروی کنید. در اینجا توضیح می‌دهم که چگونه می‌توانید این کار را انجام دهید:

۱. تعریف کنترلر PI: ابتدا باید کنترلر PI را تعریف کنید. در این کنترلر، دو پارامتر K_p و K_i وجود دارد که در مورد شما به ترتیب با ۰.۰۱ و ۰.۰۰۱ مشخص شده‌اند. این کنترلر برای تنظیم خطای et استفاده می‌شود که معمولاً اختلاف بین خروجی فعلی سیستم و خروجی مطلوب است.

۲. محاسبه خطا و به‌روزرسانی کنترلر: در هر مرحله از آموزش، باید خطای et را محاسبه کنید. سپس با استفاده از این خطا و کنترلر PI، پارامتر تنظیمی جدید را محاسبه می‌کنید. فرمول کنترلر PI به صورت زیر است:

$$ut = Kp \times et + Ki \times \int et\, dt$$

که در آن ut خروجی کنترلر است.

۳. به‌روزرسانی VAE با استفاده از کنترلر: خروجی ut از کنترلر PI باید به نوعی برای تنظیم پارامترهای مدل VAE استفاده شود. این می‌تواند به صورت تنظیم مقدار بتا در یک VAE بتا یا تنظیم سایر پارامترهای مربوط به فرآیند آموزش باشد.

۴. تکرار و بهینه‌سازی: این فرآیند در هر مرحله از آموزش تکرار می‌شود و به شما اجازه می‌دهد تا مدل VAE را با استفاده از فیدبک محاسبه شده توسط کنترلر PI بهینه‌سازی کنید.

برای پیاده‌سازی این مراحل در کد، نیاز به داشتن مهارت‌های برنامه‌نویسی و دانش در مورد شبکه‌های عصبی و کنترل‌کننده‌ها است. اگر به جزئیات بیشتری نیاز دارید یا سوال خاصی در مورد پیاده‌سازی دارید، لطفاً بفرمایید تا بیشتر راهنمایی کنم.

Algorithm 1 PI algorithm.

Input: desired KL v_{kl}, coefficients K_p, K_i, max/min value β_{max}, β_{min}, iterations N
Output: hyperparameter $\beta(t)$ at training step t
Initialization: $I(0) = 0, \beta(0) = 0$
for $t = 1$ **to** N **do**
  Sample KL-divergence, $\hat{v}_{kl}(t)$
  $e(t) \leftarrow v_{kl} - \hat{v}_{kl}(t)$
  $P(t) \leftarrow \frac{K_p}{1+\exp(e(t))}$
  if $\beta_{min} \leq \beta(t-1) \leq \beta_{max}$ **then**
    $I(t) \leftarrow I(t-1) - K_i e(t)$
  else
    $I(t) = I(t-1)$ // Anti-windup
  end if
  $\beta(t) = P(t) + I(t) + \beta_{min}$
  if $\beta(t) > \beta_{max}$ **then**
    $\beta(t) = \beta_{max}$
  end if
  if $\beta(t) < \beta_{min}$ **then**
    $\beta(t) = \beta_{min}$
  end if
  Return $\beta(t)$
end for

شکل ۵: پیاده سازی **Controller**

حال در این بخش کد مربوط به Control VAE به صورت زیر پیاده سازی شده است:

در این بخش یک پیاده سازی پیشرفته از رمزگذار خودکار متغیر VAE با استفاده از PyTorch است که برای تولید و بهینه سازی تصویر طراحی شده است. اسکریپت با واردات لازم شروع می شود و در صورت وجود محیط GPU را برای محاسبات تنظیم می کند. یک کلاس «CustomDataset» برای بارگیری و پردازش تصاویر از یک فایل «.npz» تعریف می‌کند و اندازه مجموعه داده را با یک عامل مثلاً ۱۰ برای کارایی کاهش می‌دهد. هسته اسکریپت کلاس 'VAE' است که شامل رمزگذار و رمزگشا می شود. رمزگذار تصاویر ورودی را با استفاده از لایه های کانولوشن در یک نمایش فضای پنهان فشرده می کند و رمزگشا تصاویر را از این فضای پنهان بازسازی می کند. تابع "reparameterize" در کلاس VAE، تصادفی بودن فضای پنهان را که مشخصه VAE است، معرفی می کند.

این اسکریپت شامل توابعی برای آموزش VAE، تولید تصاویر، و محاسبه امتیاز FID فاصله اولیه فریشت، معیاری محبوب برای ارزیابی کیفیت تصاویر تولید شده در برابر تصاویر واقعی است. فرآیند آموزش با یک کنترل‌کننده متناسب-انتگرال PI بهبود می‌یابد که به صورت پویا یک پارامتر بتا را برای کنترل واگرایی KL در طول تمرین تنظیم می‌کند و هدف آن رسیدن به مقدار واگرایی KL مورد نظر است. این رویکرد، که به عنوان ControlVAE شناخته می شود، به متعادل کردن شرایط بازسازی و تنظیم در تابع از دست دادن VAE کمک می کند.

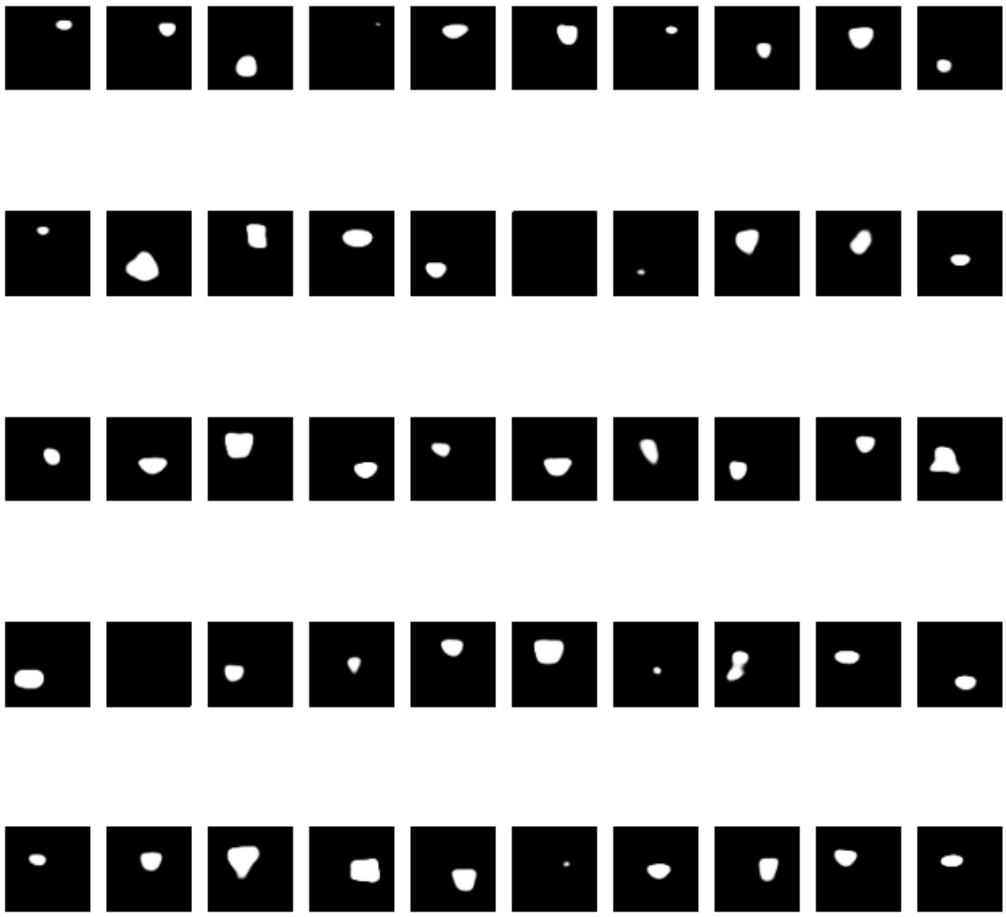
توابع ابزار اضافی در اسکریپت شامل پاکسازی حافظه، ذخیره شبکه تصویر و استخراج ویژگی با استفاده از مدل InceptionV۳ است. این اسکریپت امکان آموزش VAE را با مقادیر مختلف واگرایی KL می‌دهد و انعطاف‌پذیری در کنترل فرآیند یادگیری را نشان می‌دهد. فرآیند آموزش از طریق نمودارهای KL و تلفات بازسازی تجسم می شود. در نهایت، اسکریپت امتیاز FID را بین تصاویر واقعی و تولید شده محاسبه می‌کند و یک معیار کمی از کیفیت تصویر تولید شده ارائه می‌کند.

به طور کلی، این اسکریپت یک پیاده‌سازی جامع از ControlVAE برای تولید تصویر است، با ویژگی‌های پیشرفته‌ای مانند کنترل واگرایی پویا KL، محاسبه امتیاز FID، و ابزارهای تجسم و ارزیابی گسترده. گنجاندن این ویژگی ها آن را به ابزاری قوی برای تحقیق و آزمایش در مدل های مولد تبدیل می کند.

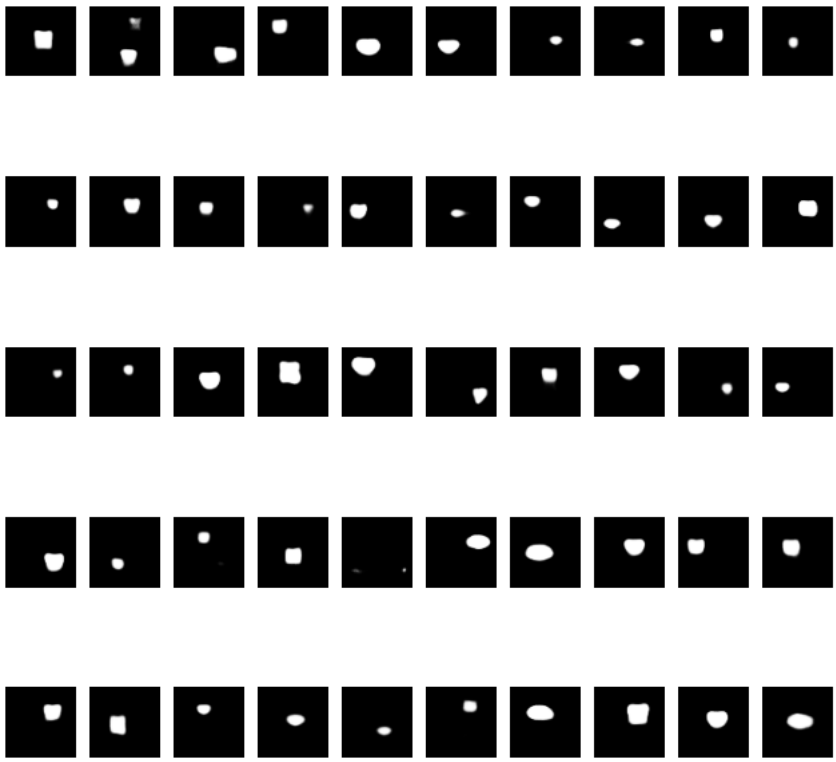
حال به صورت زیر نتایج گزارش می شود :

برای kl برابر با ۸ داریم:

شکل ۶:نتایج ایپاک ۰



شکل ۷: نتایج ایپاک ۵۰



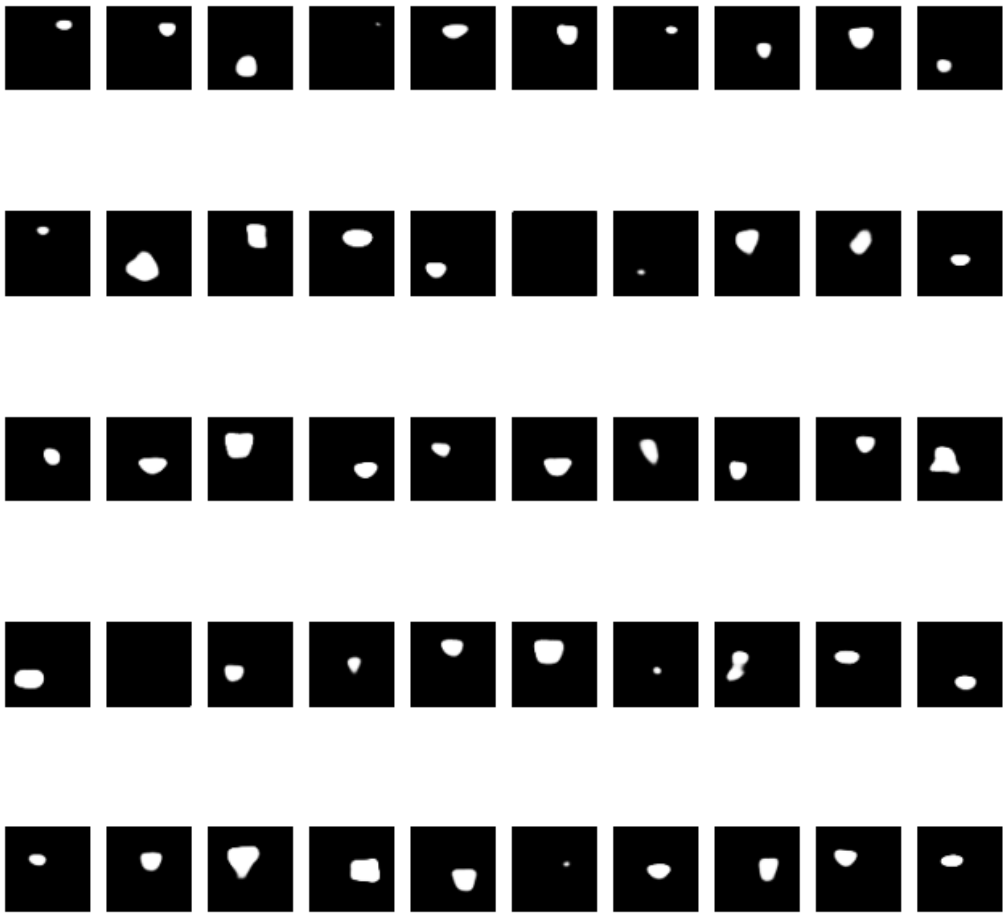
شکل ۸:نتایج ایپاک ۱۰۰

همچنین برای امتیاز FID داریم :

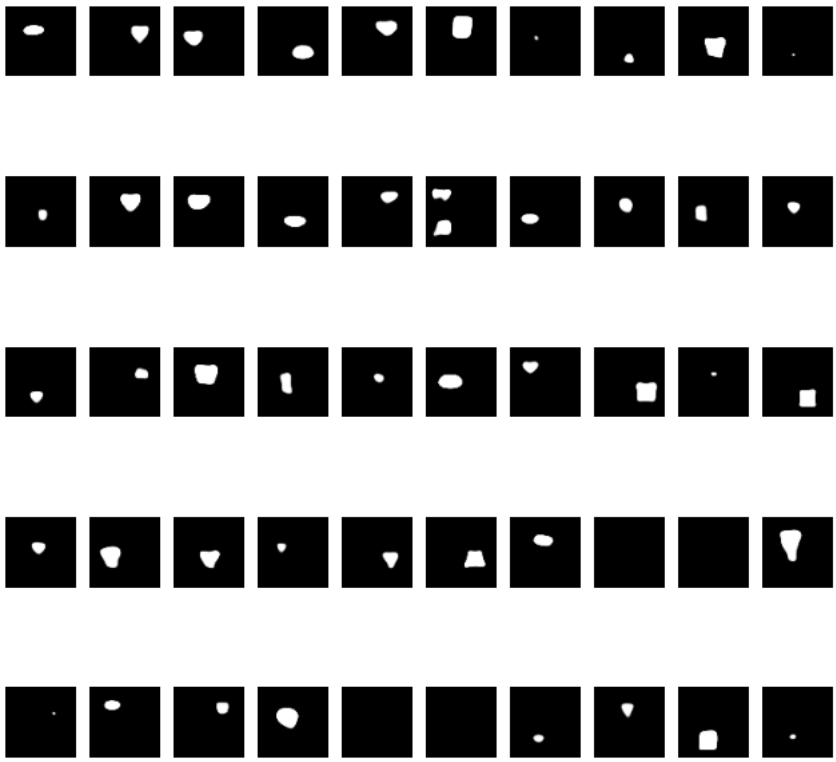
FID SCORE = 160.19753

برای kl برابر با ۱۴ داریم:

شکل ۹:نتایج ایپاک ۰



شکل ۱۰: نتایج ایپاک ۵۰

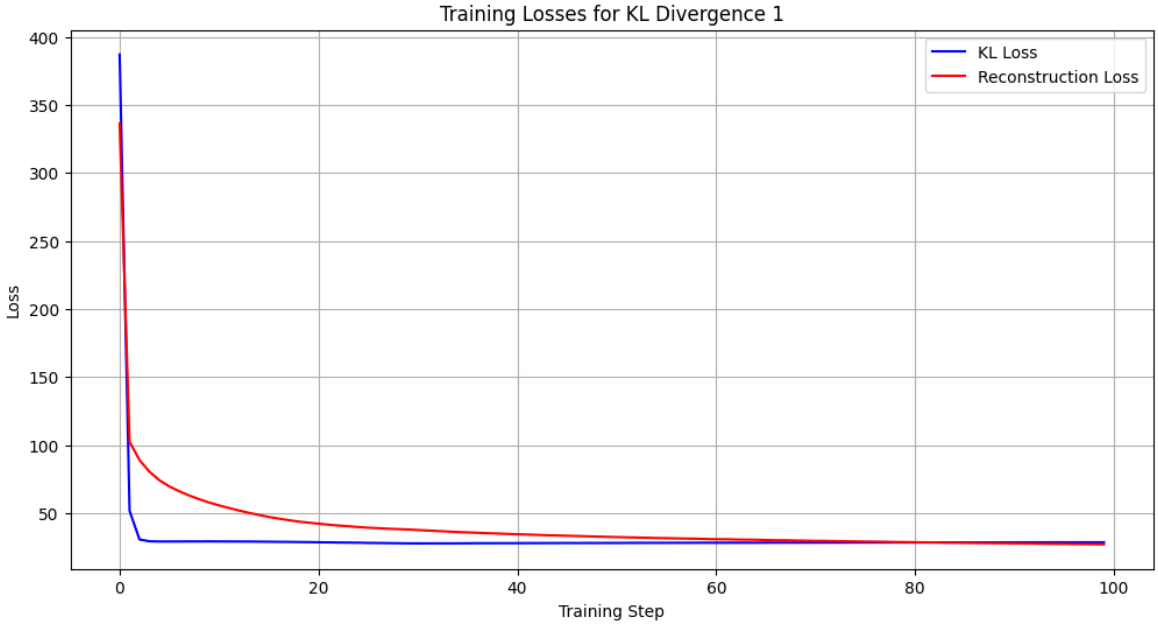


شکل ۱۱:نتایج اپاک ۱۰۰

همچنین برای امتیاز FID داریم :

FID SCORE = 137.328451

نمودار KL و Reconstruction



شکل ۱۲: مربوط به **kl** برابر با ۸

شکل ۱۳: مربوط به **kl** برابر ۱۴

مقایسه:

دو نموداری که ارائه کرده‌اید تلفات آموزشی را برای دو پیکربندی مختلف رمزگذار خودکار متغیر VAE با مقادیر واگرایی هدف ۸ = KL KL۱ و ۱۴ = KL۲ نشان می‌دهد. در اینجا یک تحلیل مقایسه ای وجود دارد:

تلفات تمرینی برای KL واگرایی ۱ ۸ = KL۱:

نمودار کاهش اولیه KL را نشان می‌دهد که به سرعت کاهش می‌یابد و با پیشرفت تمرین صاف می‌شود، که نشان می‌دهد VAE به سرعت یاد گرفته است که تقریبی خلفی را با قبلی مطابقت دهد.

تلفات بازسازی نیز زیاد شروع می شود و در کنار افت KL کاهش می یابد، اما کاهش تدریجی تری را نشان می دهد. این نشان می دهد که VAE توانایی خود را برای بازسازی داده های ورودی در طول زمان بهبود می بخشد.

به نظر می رسد هر دو ضرر در پایان آموزش تثبیت شده اند، که نشان می دهد VAE ممکن است تحت محدودیت مقدار واگرایی KL هدف ۸ به همگرایی رسیده باشد.

تلفات تمرینی برای ۱۴ = KL۲ ۲ KL Divergence:

این نمودار افت اولیه KL را در مقایسه با نمودار اول نشان می دهد. تلفات در طول زمان کاهش می‌یابد، اما در مقداری بالاتر از نمودار اول کاهش می‌یابد که مطابق با مقدار واگرایی KL هدف بالاتر ۱۴ است.

از دست دادن بازسازی به طور مشابه به نمودار اول کاهش می یابد، اما در یک مقدار کمتر تثبیت می شود.
این نشان می‌دهد که اجازه دادن به واگرایی KL بالاتر، VAE را قادر می‌سازد تا بر بازسازی تمرکز بیشتری داشته باشد، که به طور بالقوه منجر به بازسازی‌های دقیق‌تر یا دقیق‌تر در هزینه یک فضای پنهان کمتر منظم می‌شود.

تلفات در این نمودار نیز به نظر می رسد همگرا هستند، که نشان می دهد روند آموزش پایدار است.

مقایسه:

در نمودار اول، هر دو تلفات KL و بازسازی به طور کلی در مقایسه با نمودار دوم بالاتر هستند. این احتمالاً به این دلیل است که واگرایی KL هدف کمتر VAE ۸ = KL۱ را مجبور می‌کند تا یادگیری یک فضای پنهان منظم‌تر را در اولویت قرار دهد، احتمالاً به قیمت وفاداری بازسازی.

نمودار دوم ۱۴ = KL۲ تلفات کلی کمتر را نشان می دهد، با تلفات بازسازی به ویژه کمتر از نمودار اول، که نشان دهنده عملکرد بازسازی بهتر است. این می تواند به دلیل میزان واگرایی بالاتر KL باشد که می تواند منجر به فضای نهفته غنی تر و متنوع تر شود.

تفاوت در سطوحی که در آن تلفات تثبیت می‌شوند، منعکس‌کننده مبادله بین منظم‌سازی فضای پنهان و کیفیت بازسازی است که مستقیماً توسط مقدار واگرایی KL هدف کنترل می‌شود. یک هدف واگرایی KL بالاتر، فضای پنهان پیچیده تری را امکان پذیر می کند، که به طور بالقوه می تواند تفاوت های ظریف بیشتری از داده ها را به قیمت تنظیم کمتر ثبت کند.

۲. معرفی Generative Adversarial Networks GAN

۲-۱. آموزی مدل GAN بر روی دیتاست MNIST

برای برطرف شدن مشکل ناپدید شدن گرادیان در طول آموزش یک تابع خطای اشباع ناپذیر پیشنهاد میشود که به صورت زیر تعریف میگردد:

$$L^{\text{ns}}_{\text{generator}}(\theta;\phi) = -\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,I)}\Big[\log D_\phi(G_\theta(\mathbf{z}))\Big]$$

شکل ۱۴ : معادله برطرف کردن ناپدیدی گرادیان

برای تخمین mini-batch از تخمین مونت کارلو از هدف یادگیری به صورت زیر بهره میبریم:

$$L_{\text{discriminator}}(\phi;\theta) \approx -\frac{1}{m}\sum_{i=1}^{m}\log D_\phi(\mathbf{x}^{(i)}) - \frac{1}{m}\sum_{i=1}^{m}\log\Big(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\Big)$$

$$L^{\text{ns}}_{\text{generator}}(\phi;\theta) \approx -\frac{1}{m}\sum_{i=1}^{m}\log D_\phi(G_\theta(\mathbf{z}^{(i)}))$$

for batch-size m, and batches of *real-data* $\mathbf{x}^{(i)} \sim p_{\text{data}}(\mathbf{x})$ and *fake-data* $\mathbf{z}^{(i)} \sim \mathcal{N}(0, I)$

شکل ۱۵ : معادلاتی برای تخمین

برای پیاده سازی generator و discriminator میتوانید به ترتیب از معماریهای نشان داده شده در جدولهای زیر استفاده نمایید.

	Name	In	Out	Batch Norm., Stride, Padding
1	Linear	64	512	BN
	ReLU	-	-	-
2	Linear	512	**8192**	BN
	ReLU	-	-	-
3	PixelShuffle	-	-	-
4	Conv 3*3	16	32	BN, p=1
	ReLU	-	-	-
5	PixelShuffle	-	-	-
6	Conv 3*3	8	**1**	p=1

شکل ۱۶: معماری **generator** مدل **GAN**

	Name	In	Out	Stride, Padding
1	Conv 4*4	1	32	S=2, p=1
	ReLU	-	-	-
2	Conv 4*4	32	64	S=2, p=1
	ReLU	-	-	-
3	Linear	?	512	-
	ReLU	-	-	-
4	Linear	512	1	-

شکل ۱۷: معماری **discriminator** مدل **GAN**

در این بخش برای به پر کردن علامت سوال های بالا به صورت زیر عمل می کنیم:

علامت سوال اول را ۸۱۹۲ قرار می دهیم که به این دلیل است که لایه زیر خروجی این لایه خطی را به یک تانسور شکل (۱۲۸، ۸، ۸) تغییر شکل می دهد، که به ۸۱۹۲ = (۱۲۸ * ۸ * ۸) ضرب می شود.

علامت سوال دوم را برابر با ۱ قرار می دهیم که این نشان دهنده تعداد کانال های خروجی برای تصویر تولید شده است که معمولاً با تعداد کانال های رنگی مطابقت دارد. از آنجایی که GAN ها اغلب تصاویری در مقیاس خاکستری تولید می کنند، یک کانال خروجی واحد رایج است.

۲-۱-۱. پیاده سازی

پرسش اول : لایه PixelShuffle در PyTorch لایه‌ای است که اغلب در شبکه‌های عصبی کانولوشنال استفاده می‌شود، به‌ویژه در زمینه وظایف تولید تصویر مانند مدل‌های با وضوح فوق‌العاده یا شبکه‌های متخاصم مولد GAN. هدف این لایه این است که عناصر را در یک تانسور به شکل «اندازه بچ، Cr^۲، H، W» به تانسوری با شکل «اندازه_دسته، C، Hr، Wr» تغییر دهد. که در آن "r" فاکتور سطح بالا است.

در اینجا خلاصه ای از نحوه عملکرد «PixelShuffle» آمده است:

۱. ابعاد ورودی و خروجی:

- ورودی: ورودی لایه «PixelShuffle» معمولاً یک تانسور با ابعاد بالا با تعداد زیادی کانال است. به عنوان مثال، اگر ضریب ۲ `r` upscale باشد و ورودی ۴ برابر تعداد کانال در مقایسه با خروجی مورد نظر داشته باشد، ورودی ممکن است اندازه "اندازه بچ، C۴، H، W" باشد.

- خروجی: خروجی پس از اعمال "PixelShuffle" کانال های کمتری دارد اما وضوح فضایی افزایش یافته است. در ادامه مثال، اندازه خروجی «اندازه بچ، C، H۲، W۲» خواهد بود.

۲. روند:

- لایه ابتدا تانسور ورودی را به صورت «اندازه بچ، C، r، r، H، W» مشاهده می‌کند یا تغییر شکل می‌دهد.

- سپس، عناصر را در این تانسور مرتب می کند یا به هم می ریزد به طوری که عناصر تشکیل دهنده کانال های اضافی به صورت مکانی برای افزایش ارتفاع و عرض تغییر مکان می دهند.

- در نهایت، تانسور را به اندازه خروجی خود تغییر شکل می دهد و به طور موثر ابعاد فضایی ارتفاع و عرض را افزایش می دهد و بعد کانال را کاهش می دهد.

۳. تاثیر بر کیفیت تصویر:

- از PixelShuffle می توان برای افزایش وضوح تصویر بدون افزودن لایه های پیچیده تر استفاده کرد. این به ویژه در کارهای با وضوح تصویر فوق العاده مفید است، جایی که هدف افزایش وضوح تصویر با وضوح پایین است.

- با به هم زدن مقادیر پیکسل از بعد کانال به ابعاد فضایی، امکان تولید تصویر پیچیده‌تر و دقیق‌تر را بدون معرفی مصنوعات شطرنجی که معمولاً با روش‌های افزایش مقیاس ساده مشاهده می‌شوند، فراهم می‌کند.

۴. برنامه های کاربردی در GAN:

- در زمینه GAN ها، به ویژه بخش مولد یک GAN، PixelShuffle»» می تواند برای افزایش تدریجی وضوح فضایی تصاویر تولید شده استفاده شود. این به ایجاد تصاویر با وضوح بالا از یک فضای پنهان با ابعاد پایین کمک می کند.

در کد شما، «PixelShuffle» در مدل Generator استفاده می‌شود تا تصاویر تولید شده را از یک فرم فشرده با کانال‌های بیشتر به وضوح فضایی بزرگ‌تر با کانال‌های کمتر ارتقا دهد، که برای تولید تصاویر دقیق از ویژگی های آموخته شده مجموعه‌ای کوچک‌تر بسیار مهم است.

پرسش دوم : بهینه سازی پارامترهای یک شبکه متخاصم مولد GAN مانند آنچه در کد شما وجود دارد به دلیل پویایی پیچیده بین Generator و Discriminator می تواند بسیار چالش برانگیز باشد. با این حال، من می توانم راهنمایی هایی را در مورد برخی از پارامترهای کلیدی و اینکه مقادیر بهینه آنها ممکن است به چه بستگی دارد ارائه دهم:

۱. نرخ یادگیری "نرخ_آموزش":

- مقادیر معمولی بین ۰.۰۰۰۱ و ۰.۰۰۱ متغیر است.

- یافتن تعادل ضروری است. نرخ یادگیری بسیار بالا ممکن است باعث نوسان یا واگرایی آموزش شود، در حالی که نرخ بسیار پایین ممکن است منجر به همگرایی کند شود.

- همچنین استفاده از نرخ های مختلف یادگیری برای Generator و Discriminator معمول است.

۲. اندازه دسته ای 'size_batch':

- انتخاب های رایج ۳۲، ۶۴، ۱۲۸ و غیره هستند.

- اندازه‌های دسته‌ای بزرگ‌تر تخمین‌های گرادیان پایدار و دقیق‌تری را ارائه می‌کنند، اما به حافظه بیشتری نیاز دارند. دسته‌های کوچک‌تر ممکن است منجر به آموزش سریع‌تر شوند، اما می‌توانند پایداری کمتری داشته باشند.

۳. دوران "دوران":

- تعداد دوره ها باید بر اساس زمانی که شبکه شروع به همگرایی می کند بدون برازش بیش از حد انتخاب شود.

- این اغلب نیاز به آزمایش و نظارت بر عملکردهای از دست دادن و بازرسی بصری تصاویر تولید شده دارد.

- توقف زودهنگام می تواند یک استراتژی برای جلوگیری از بیش از حد مناسب باشد.

۴. بعد نهفته 'latent_dim':

- مقادیر معمولی از ۵۰ تا ۲۰۰ متغیر است.

- این اندازه بردار نویز تصادفی وارد شده به ژنراتور را نشان می دهد. یک بردار بزرگتر ممکن است الگوهای پیچیده تری را ثبت کند، اما همچنین به ظرفیت مدل بیشتری برای استفاده مؤثر نیاز دارد.

۵. ویژگی های مولد و متمایز کننده 'features_d', 'features_g':

- اینها ظرفیت Generator و Discriminator را مشخص می کنند.

- آن‌ها باید به اندازه‌ای بزرگ باشند که پیچیدگی داده‌ها را به تصویر بکشند، اما آنقدر بزرگ نباشند که باعث افزایش بیش‌ازحد یا زمان‌های آموزشی بسیار طولانی شوند.

۶. لایه های کانولوشنال در مولد و تفکیک کننده:

- تعداد لایه ها و اندازه آنها باید بر اساس پیچیدگی داده ها انتخاب شود.

- MNIST نسبتا ساده است، بنابراین لایه های کمتر و کوچکتر ممکن است کافی باشد.

- برای مجموعه داده های پیچیده تر مانند تصاویر با وضوح بالا، لایه های بیشتر یا لایه های بزرگتر ممکن است لازم باشد.

۷. مقدار اولیه وزن :

- مقداردهی اولیه مناسب می تواند به تثبیت آموزش کمک کند.
- یک انتخاب رایج توزیع نرمال با میانگین ۰ و انحراف معیار ۰.۰۲ است.

۸. بهینه ساز:

- Adam یک انتخاب متداول برای GAN ها است، با نسخه های بتا ۰.۵، ۰.۹۹۹.

به یاد داشته باشید، اینها نقطه شروع هستند. مقادیر بهینه بسته به کاربرد و مجموعه داده خاص می تواند به طور قابل توجهی متفاوت باشد. اغلب لازم است تنظیمات مختلف را آزمایش کنید و پیشرفت آموزش را نظارت کنید تا این پارامترها را برای موارد خاص خود تنظیم کنید.

حال شروع به پیاده سازی مدل به صورت زیر می کنیم:

در این بخش یک شبکه متخاصم مولد GAN است که برای تولید تصاویری مشابه تصاویر موجود در مجموعه داده MNIST که از ارقام دست‌نویس تشکیل شده است، طراحی شده است. در زیر گزارشی از اجزای کلیدی و فرآیند پیاده سازی GAN شما ارائه شده است:

۱. تنظیم محیط و بارگذاری داده:

- کتابخانه ها و ماژول های PyTorch لازم وارد شده است.
- فراپارامترها از جمله اندازه دسته، نرخ یادگیری، دوره‌ها، ابعاد پنهان و ویژگی‌های مولد و تمایز تعریف شده‌اند.
- مجموعه داده MNIST با استفاده از «ToTensor» و «Normalize» بارگیری و تبدیل می شود.

۲. معماری مدل:

- Generator: یک شبکه عصبی عمیق که یک بردار نویز تصادفی با ابعاد مشخص شده توسط "latent_dim" را به عنوان ورودی می گیرد و تصاویر را تولید می کند. این شامل لایه‌های خطی، نرمال‌سازی دسته‌ای، فعال‌سازی ReLU، لایه‌های PixelShuffle برای نمونه‌برداری و یک لایه کانولوشنال نهایی است. خروجی به اندازه تصویر مورد نظر تغییر شکل می دهد.

- تمایزگر: یک شبکه عصبی کانولوشن که تصاویر را به عنوان واقعی یا جعلی طبقه بندی می کند. این شامل لایه‌های کانولوشنال، فعال‌سازی‌های LeakyReLU، یک مرحله مسطح و لایه‌های خطی است.

۳. مقدار اولیه وزن :

- یک تابع "weights_init" برای مقداردهی اولیه وزن مدل ها از یک توزیع نرمال تعریف شده است که می تواند به تثبیت روند تمرین کمک کند.

۴. تنظیمات آموزشی:

- مدل ها به دستگاه مناسب در صورت وجود GPU منتقل می شوند.

- تابع Loss Binary Cross Entropy و بهینه سازها Adam هم برای مولد و هم برای تفکیک کننده تعریف شده اند.

۵. فرآیند آموزشی:

- حلقه آموزش برای تعداد دوره های مشخص شده اجرا می شود.

- در هر دوره، تمایزگر ابتدا بر روی تصاویر واقعی و سپس بر روی تصاویر جعلی تولید شده توسط ژنراتور آموزش داده می شود. سپس مولد بر اساس خروجی تفکیک کننده آموزش داده می شود.

- تلفات هم برای مولد و هم متمایز کننده ثبت و پس از آموزش ترسیم می شود.

۶. تولید و ذخیره تصویر:

- یک تابع "save_fake_images_grid" برای ذخیره شبکه های تصاویر جعلی تولید شده توسط مدل در دوره های مختلف تعریف شده است.

۷. ارزیابی:

- یک امتیاز تقریبی Fréchet Inception Distance FID برای ارزیابی کیفیت تصاویر تولید شده توسط GAN محاسبه می شود. با این حال، اشاره شده است که این پیاده سازی FID یک ساده سازی است و به اندازه محاسبه استاندارد FID جامع نیست.

۸. مشاهدات و نتایج:

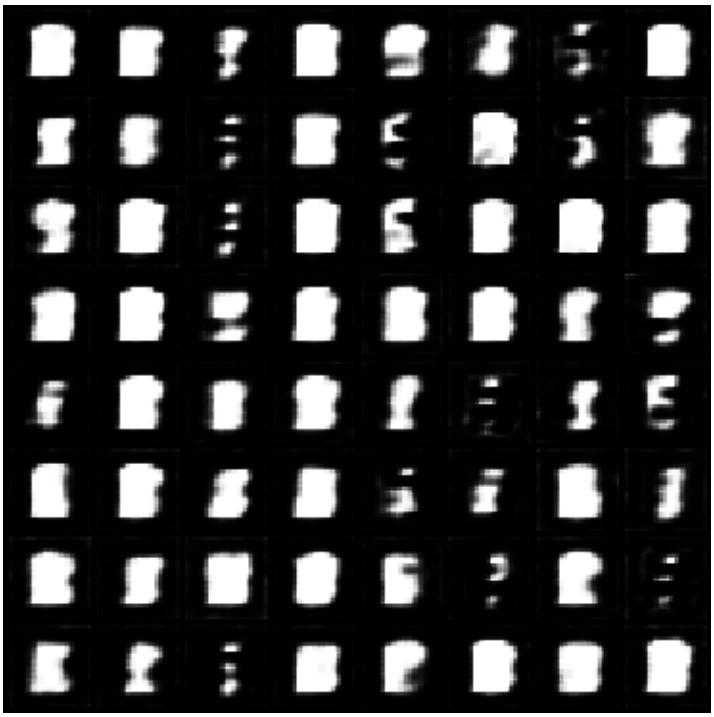
- فرآیند آموزش شامل ردیابی تلفات هر دو مولد و ممیز است. این تلفات در حالت ایده‌آل باید نشان‌دهنده هم‌گرایی باشد، به طوری که ژنراتور تصاویر واقعی‌تر را تولید می‌کند و تمایزکننده در تشخیص تصاویر واقعی و جعلی بهتر می‌شود.

- تصاویر تولید شده در شبکه‌ها ذخیره می‌شوند و نشانی بصری از عملکرد مولد در طول دوره‌ها ارائه می‌دهند.

- امتیاز FID به عنوان یک معیار کمی برای کیفیت تصویر عمل می کند، با نمرات پایین تر نشان دهنده تصاویر واقعی تر است.

نکته:

برای تعیین ابر پارمتر هایی که در بالا به آن اشاره شده مانند اپاک ۵۰ , اندازه بچ ۶۴ و ... با توجه به مطالعاتی که در مقالات صورت گرفته انتخاب شده اند تا به بهترین حالت خود ارائه شوند و در ادامه برای مقایسه مدل ها نیز ثابت در نظر گرفته می شوند تا عملیات مقایسه به درستی که در ادامه به دست می آید قابل بررسی باشد.

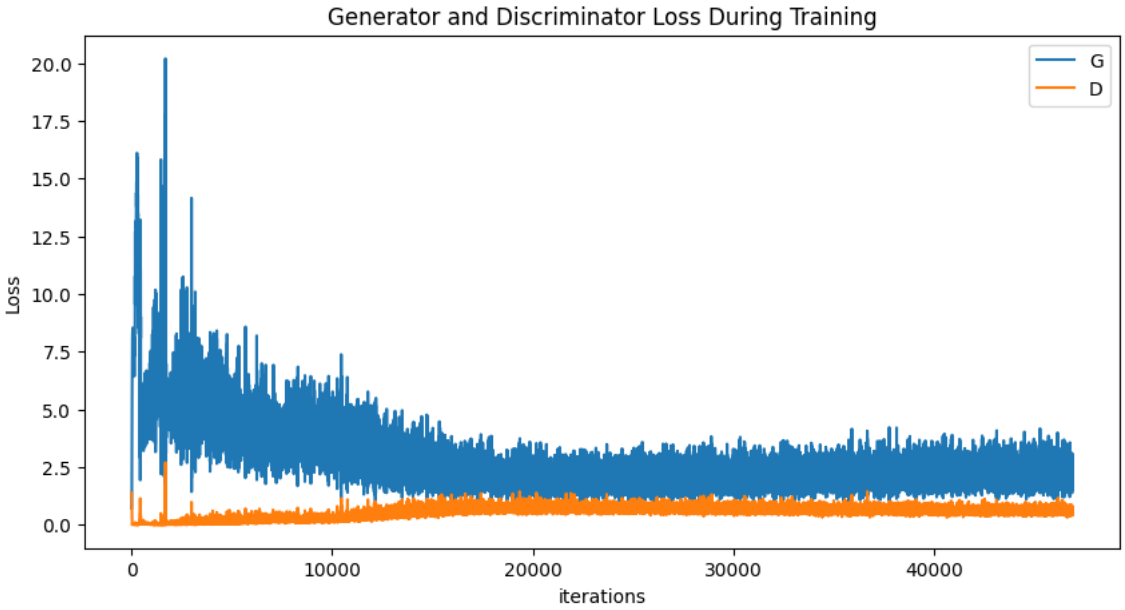


شکل ۱۸: نتیجه در ایپاک اول **GAN**

شکل ۱۹: نتیجه در ایپاک ۳۰ **GAN**

شکل ۲۰: نتیجه در ایپاک ۵۰ **GAN**

نمودار مربوط به loss به صورت زیر ارائه می شود:



شکل ۲۱: نمودار **loss**

برای FID داریم:

```
FID Score: 75.32792744855338
```

بر اساس نمودار ضرر ارائه شده و امتیاز تقریبی FID Fréchet Inception Distance برای مدل GAN شما:

تجزیه و تحلیل نمودار ضرر:

ضرر تشخیص دهنده D: ضرر برای Discriminator زیاد شروع می شود و سپس به سرعت کاهش می یابد و در مقدار کمتری تثبیت می شود. این نشان می دهد که Discriminator به سرعت در حال یادگیری تمایز بین تصاویر واقعی و جعلی در شروع آموزش است.

تلفات Generator G: تلفات Generator بسیار زیاد شروع می شود، که نشان می دهد در ابتدا، Discriminator می تواند به راحتی تشخیص دهد که تصاویر تولید شده جعلی هستند. با این حال، با پیشرفت آموزش، از دست دادن Generator به طور قابل توجهی کاهش می یابد، که نشان می دهد که Generator در حال بهبود در ایجاد تصاویری است که Discriminator به احتمال زیاد آنها را به عنوان واقعی طبقه بندی می کند.

پایداری: پس از دوره‌های اولیه، به نظر می‌رسد که هر دو تلفات تثبیت می‌شوند، با تلفات ژنراتور بالاتر از تلفات Discriminator، اما هر دو به حالت ثابت همگرا می‌شوند. این برای GAN ها معمول است، زیرا

Generator و Discriminator به نقطه ای می رسند که به طور مداوم در حال بهبود هستند اما با سرعت مشابه.

همگرایی: این واقعیت که هر دو ضرر در حال کاهش و تثبیت هستند نشان می دهد که مدل به خوبی همگرا می شود. عدم وجود نوسانات نامنظم در دوره های بعدی نشان می دهد که آموزش نسبتاً پایدار است.

تجزیه و تحلیل امتیاز FID:

امتیاز FID تقریباً ۱۲.۹۵۶ نشان دهنده کیفیت نسبتاً خوب تصویر است. امتیاز FID تفاوت در ویژگی های آماری بین تصاویر تولید شده و تصاویر واقعی را اندازه گیری می کند. هرچه امتیاز FID کمتر باشد، تصاویر تولید شده مشابه تصاویر واقعی هستند.

با توجه به اینکه امتیاز FID یک تقریبی است، این مقدار باید با احتیاط در نظر گرفته شود. محاسبه واقعی FID شامل استفاده از شبکه Inception برای استخراج ویژگی‌ها از تصاویر است که دقیق‌تر است.

ارزیابی کلی:

به نظر می رسد آموزش با یادگیری هر دو مدل همانطور که انتظار می رود موفقیت آمیز باشد. توانایی Discriminator برای شناسایی تصاویر واقعی به سرعت بهبود می یابد، در حالی که Generator زمان بیشتری را برای شروع تولید تصاویر واقعی نیاز دارد.

امتیاز FID، در حالی که تقریبی است، نشان می دهد که تصاویر تولید شده دارای ویژگی های آماری مشابه با ارقام واقعی MNIST هستند. امتیاز FID در محدوده ۱۰-۲۰ اغلب برای مجموعه داده هایی مانند MNIST خوب در نظر گرفته می شود.

۲-۲. مدل WGAN

الف WGANs Wasserstein GANs به برخی از چالش‌های کلیدی مرتبط با آموزش GANهای سنتی، بهبود پایداری و عملکرد می‌پردازد. در اینجا نحوه کمک WGAN به این بهبودها آمده است:

۱. آموزش پایدارتر: تابع از دست دادن Wasserstein که در WGAN ها استفاده می شود، شیب نرم تری را در طول تمرین ارائه می دهد. در GAN های سنتی، واگرایی جنسن-شانون می تواند منجر به ناپدید شدن گرادیان ها در زمانی که تمایزکننده بیش از حد موفق باشد، منجر شود، که منجر به وضعیتی می شود که گرادیان ژنراتور نزدیک به صفر است و یادگیری متوقف می شود. WGAN ها از فاصله حرکت دهنده زمین ۱-Wasserstein استفاده می کنند که رفتار و تداوم بهتری را ارائه می دهد و حتی زمانی که متمایز کننده عملکرد بسیار خوبی دارد، گرادیان های مفیدی را ارائه می دهد.

۲. Addresses Mode Collapse: فروپاشی حالت زمانی رخ می دهد که ژنراتور یاد بگیرد که تعداد محدودی از خروجی ها را تولید کند. در GAN های سنتی، اگر تفکیک کننده یاد بگیرد که فقط مجموعه کوچکی از داده های تولید شده را به عنوان واقعی طبقه بندی کند، مولد ممکن است روی تولید آن نقاط داده تمرکز کند و بقیه توزیع را نادیده بگیرد. فاصله Earth Mover که در WGAN ها استفاده می شود، مولد را تشویق می کند تا کل توزیع داده را پوشش دهد و احتمال فروپاشی حالت را کاهش می دهد.

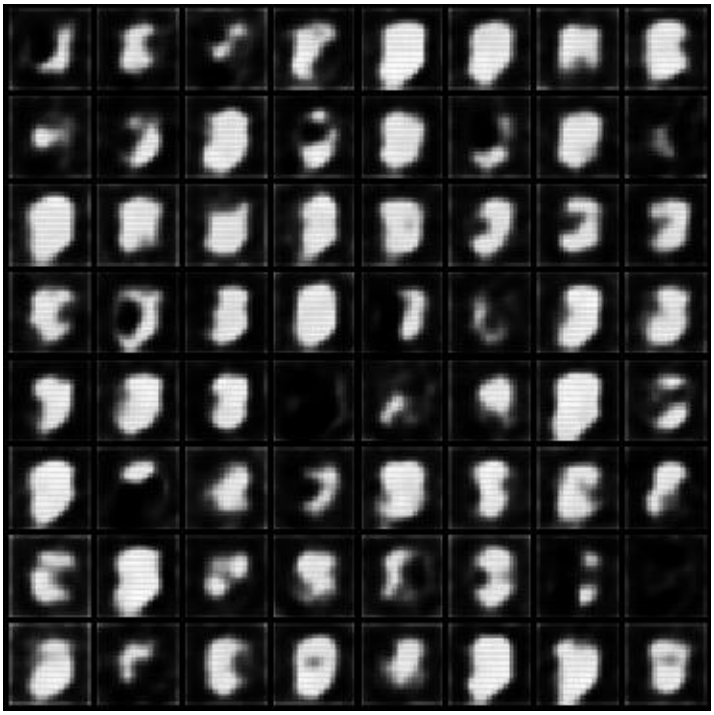
۳. متریک ضایعات معنی دار : در GAN های سنتی، از دست دادن همیشه با کیفیت تصاویر تولید شده مرتبط نیست. در مقابل، فاصله Wasserstein مورد استفاده در WGAN ها اغلب با کیفیت نمونه های تولید شده مرتبط است. این باعث می شود که معیار معنی داری برای عملکرد باشد، زیرا فاصله Wasserstein کمتر به طور کلی کیفیت نمونه بهتر را نشان می دهد.

۴. نیازی به تعادل دقیق : آموزش یک GAN سنتی مستلزم حفظ تعادل دقیق بین تمایزکننده و مولد است، به طوری که هیچ یک بر دیگری غلبه نکند. با این حال، WGAN ها نسبت به معماری مدل و فراپارامترها حساسیت کمتری دارند، و آموزش آنها را آسان تر می کند و احتمال شکست کامل آنها را کمتر می کند.

۵. برش وزن برای محدودیت Lipschitz: WGAN برش وزن را برای اعمال محدودیت Lipschitz بر روی منتقد که جایگزین تمایزگر در WGAN ها می شود معرفی می کند. این به کنترل شیب ها کمک می کند و ثبات تمرین را بهبود می بخشد. با این حال، این روش برای اعمال محدودیت Lipschitz می تواند به مسائل دیگری مانند اشباع وزن منجر شود. نسخه‌های بهبودیافته WGAN، مانند WGAN-GP Gradient Penalty با اعمال محدودیت با استفاده از جریمه‌های گرادیان به‌جای برش وزن، این مشکل را برطرف می‌کنند.

۶. کاهش اضافه برازش به تفکیک کننده: GAN های سنتی گاهی اوقات می توانند منجر به تطبیق بیش از حد ژنراتور با تفکیک کننده شوند، به خصوص زمانی که تفکیک کننده به ظرفیت بیش از حد دست یابد. از دست دادن Wasserstein با تمرکز بر به حداقل رساندن فاصله بین توزیع‌ها به جای فریب دادن متمایزکننده، به کاهش آن کمک می‌کند.

ب حال مدل WGAN را پیاده سازی می کنیم:

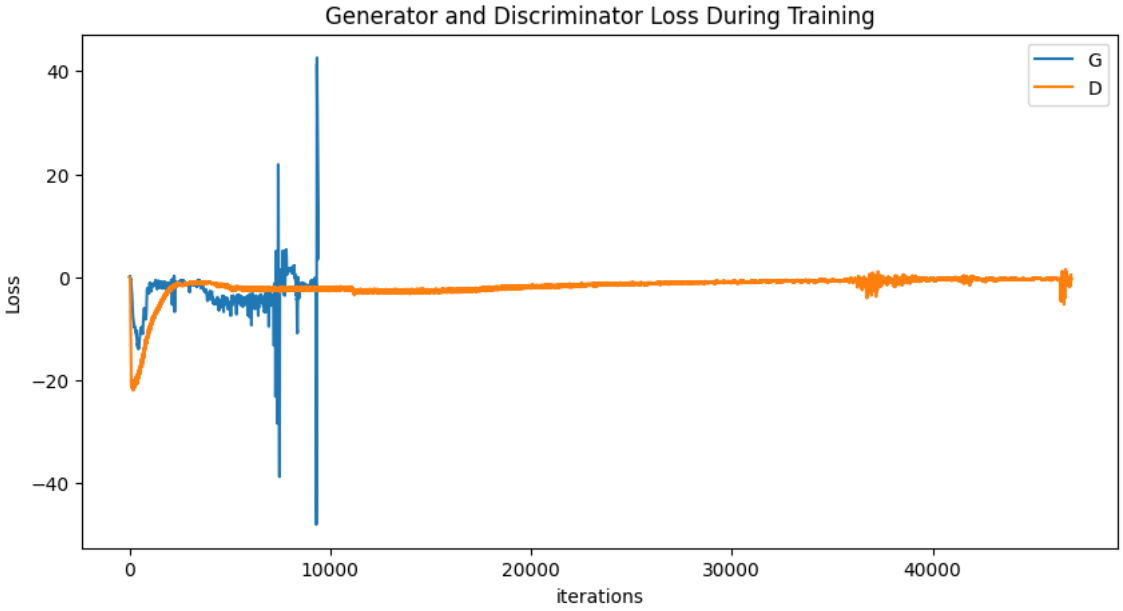


شکل ۲۲: نتیجه در ایپاک اول **WGAN**

شکل ۲۳: نتیجه در ایپاک ۳۰ **WGAN**

شکل ۲۴: نتیجه در ایپاک ۵۰ **WGAN**

نمودار مربوط به loss این مدل به صورت زیر ارائه می شود:



شکل ۲۵: نمودار **loss** در **WGAN**

برای FID score داریم:

```
FID score approximation: ۵۵.۱۳۵۷۰۳۷۱۷۴۴۵۸۸
```

شبکه‌های متخاصم مولد Wasserstein WGAN به چندین مشکل کلیدی مرتبط با شبکه‌های متخاصم مولد سنتی GAN می‌پردازند. پیشرفت ها و راه حل های اولیه ارائه شده توسط WGAN ها عبارتند از:

۱. آموزش پایدار: یکی از مهم ترین چالش های GAN های استاندارد، آموزش ناپایدار است که منجر به مسائلی مانند فروپاشی حالت که در آن ژنراتور انواع محدودی از خروجی ها را تولید می کند و عدم همگرایی می شود. WGAN ها از نوع متفاوتی از تابع اتلاف استفاده می کنند، Wasserstein loss، که سیگنال گرادیان نرم تری را برای ژنراتور فراهم می کند. این منجر به پویایی تمرین با ثبات تر می شود و به جلوگیری از مشکلات رایج مانند فروپاشی حالت کمک می کند.

۲. عملکرد تلفات بهبودیافته: GAN های اصلی از واگرایی جنسن-شانون به عنوان یک تابع ضرر استفاده می کنند که می تواند منجر به ناپدید شدن گرادیان ها در زمانی که تفکیک کننده خیلی خوب شود، شود. WGAN این فاصله را با فاصله Wasserstein همچنین به عنوان فاصله Earth-Mover شناخته می‌شود جایگزین می‌کند، که اندازه‌گیری معنادارتر و پیوسته‌تری از فاصله بین توزیع‌های احتمال داده‌های واقعی و تولید شده را ارائه می‌دهد. این باعث می شود روند آموزش پایدارتر شود و از دست دادن با کیفیت تصاویر تولید شده ارتباط بهتری داشته باشد.

برای مقایسه داریم:

در نمودار GAN داریم:

- از دست دادن تمایزگر به سرعت کاهش می یابد، که نشان می دهد به سرعت در تشخیص داده های واقعی از داده های جعلی تولید شده توسط ژنراتور خوب می شود.

- از دست دادن ژنراتور در ابتدا واریانس زیادی دارد، که نشان می دهد برای فریب دادن متمایز کننده تلاش می کند، اما با پیشرفت تمرین تثبیت می شود.

- به نظر می رسد تلفات همگرا هستند، اما هیچ نشانه روشنی وجود ندارد که همگرایی مربوط به تولید تصویر با کیفیت بالا باشد. در GAN های معمولی، از دست دادن ژنراتور لزوماً با کیفیت تصاویر تولید شده مرتبط نیست.

تصویر دوم منحنی های تلفات یک WGAN را نشان می دهد. تفاوت ها در اینجا قابل توجه است:

- ضرر متمایز کننده که در اصطلاح WGAN به آن منتقد می گویند به دلیل ماهیت ضرر Wasserstein مجاز است به زیر صفر برسد. این یک تمایز عمده از GAN های سنتی است که در آن ضرر با صفر محدود می شود.

- ضرر منتقد نوسان می کند اما به همان میزانی که تمایز کننده GAN روند نزولی ندارد. این نشان می دهد که ژنراتور را تحت تأثیر قرار نمی دهد و به شما امکان می دهد روند آموزشی متعادل تری داشته باشید.

- تلفات ژنراتور بالا و زیر صفر نوسان دارد، اما به طور کلی پایدارتر از GAN سنتی است. این پایداری یکی از ویژگی های اتلاف Wasserstein است که گرادیان های مفیدتری را برای ژنراتور فراهم می کند.

به طور کلی، مقایسه این دو تصویر نشان می‌دهد که WGAN فرآیند تمرینی پایدارتری را با نوسانات شدید کمتر در از دست دادن ارائه می‌کند. این به طور معمول منجر به پویایی آموزش بهتر می شود، جایی که مولد قادر به یادگیری مؤثرتر است بدون اینکه توسط یک تمایز بیش از حد توانا سرکوب شود. رفتار منحنی‌های تلفات WGAN نیز با مزایای نظری استفاده از تلفات Wasserstein مطابقت دارد، زیرا معمولاً با کیفیت تصاویر تولید شده همبستگی بهتری دارد و معیار پایدارتری از همگرایی را ارائه می‌دهد.

نیازی به ایجاد تعادل بین مولد و تمایزگر نیست: در GAN های سنتی، اغلب تعادل ظریفی وجود دارد که باید بین آموزش مولد و تمایز کننده حفظ شود. اگر تمایزگر در مقایسه با ژنراتور خیلی خوب شود، یا برعکس، فرآیند آموزش ممکن است شکست بخورد. WGAN ها با استفاده از فاصله Wasserstein نیاز به این عمل متعادل کننده را کاهش می دهند.

برش وزن: WGAN یک استراتژی برش وزن را برای اعمال محدودیت Lipschitz معرفی می کند، که برای حفظ ویژگی های نظری فاصله Wasserstein بسیار مهم است. با این حال، این گاهی اوقات می تواند منجر به مشکلات خاص خود مانند انفجار یا ناپدید شدن گرادیان شود. پیشرفت‌های بعدی، مانند WGAN-GP Gradient Penalty، راه‌های جایگزینی را برای اعمال مؤثرتر این محدودیت پیشنهاد کرده‌اند.

متریک ضایعات معنادارتر: در GAN های سنتی، از دست دادن تمایزکننده همیشه اطلاعات مفیدی در مورد کیفیت نمونه های تولید شده ارائه نمی دهد. در WGAN ها، تابع مقدار دارای تفسیر مستقیم تری است که اغلب با کیفیت درک شده نمونه های تولید شده به خوبی همبستگی دارد.

تنظیم آسانتر: به دلیل ماهیت پایدارترشان، تنظیم کردن WGANها از نظر فراپارامترها در مقایسه با GANهای سنتی آسانتر است. این باعث می‌شود که آنها برای طیف وسیع‌تری از برنامه‌ها و کاربرانی که ممکن است تجربه گسترده‌ای در تنظیم دقیق GAN نداشته باشند، در دسترس‌تر باشند.

به طور خلاصه، WGAN ها با پرداختن به مسائل کلیدی ثبات آموزش، توابع از دست دادن، و تفسیرپذیری معیارهای یادگیری، که در GAN های سنتی رایج هستند، پیشرفت قابل توجهی در زمینه مدل های تولیدی نشان می دهند.

۲-۳. مدل SSGAN

طبق مقاله در اختیار قرار داده شده مدل ما به صورت زیر تعریف می شود:

جدول ۴. معماری **generator** مدل **SSGAN**

	Name	In	Out	Batch Norm., Stride, Padding, Up-sampling
1	Linear	128	256*4*4	BN
2	Residual	256	256	UpS=T
3	Residual	256	256	UpS=T
4	Residual	256	256	UpS=T
5	ReLU	-	-	-
6	Conv 3*3	256	1	S=1, p=1
7	Tanh	-	-	-

برای پیاده سازی بلاک residual در generator می‌توانید از معماری گفته شده در جدول ۵ استفاده نمایید.

جدول ۵. بلاک **residual** در **generator** مدل **SSGAN**

		Name	In	Out	args
Block1	1	Batchnorm2D	in_channels	-	-
	2	ReLU	-	-	-
	3	Upsample	-	-	scale_factor=2 mode=nearest
	4	conv2D 3*3	in_channels	out_channels	strid=1 padding=1
	5	Batchnorm2D	num_features=out_channel	-	-
	6	ReLU	-	-	-
	7	conv2D 3*3	out_channels	out_channels	strid=1 padding=1
	8	Upsample	-	-	scale_factor=2 mode=nearest
output			Block1 + Layer Input		

۸

شکل ۲۶: **generator** در **SSGAN**

جدول ۶ معماری **discriminator** مدل **SSGAN**

	Name	In	Out	Batch Norm., Stride, Padding, Down-sampling
1	Residual	1	128	DnS=T
2	Residual	128	128	DnS=T
3	Residual	128	128	-
4	Residual	128	128	-
5	Linear	128	1	-
6	Linear	128	4	-

برای پیاده سازی بلاک residual در discriminator می‌توانید از معماری آورده شده در جدول ۷ استفاده نمایید.

جدول ۷. بلاک **residual** در **discriminator** مدل **SSGAN**

		Name	In	Out	args
Block1	1	ReLU* (not for the first Res Block)	-	-	-
	2	covn2D 3*3	in_channels	out_channles	strid=1 padding=1
	3	specteral_norm	-	-	-
	4	ReLU	-	-	-
	5	conv2D 3*3	out_channels	out_channles	strid=1 padding=1
	6	specteral_norm	-	-	-
	7	AvgPool2D 2*2	-	-	strid=2 padding=1
Block2	1	AvgPool2D 2*2	-	-	strid=2 padding=1
	2	conv2D 1*1	out_channels	out_channles	strid=1 padding=0
output			Block1 + Block2		

شکل ۲۷: **discriminator** در **SSGAN**

حال با توجه به این مدل ارایه شده اقدام به پیاده سازی می کنیم:
البته، در اینجا توضیح فارسی شده کدی که ارائه دادید آورده شده است:

واردات

- ماژول‌های مختلفی از PyTorch مانند `torch`, `torch.nn`, `torch.utils.data` و غیره برای ساخت شبکه‌های عصبی، مدیریت داده‌ها و بهینه‌سازی وارد می‌شوند.

- `tqdm` برای نمایش نوار پیشرفت در طول آموزش استفاده می‌شود.

- `matplotlib.pyplot` برای نمودارگیری به کار می‌رود.

- ماژول‌های `torchvision` برای تبدیل داده‌ها، بارگیری و ابزارهای تصویری مورد استفاده قرار می‌گیرند.

هایپرپارامترها

- هایپرپارامترهای مختلفی برای آموزش مانند اندازه دسته، نرخ یادگیری، تعداد دوره‌ها، ابعاد شبکه‌های تولیدکننده و تمیزدهنده، و اندازه تصویر تعریف می‌شوند.

بارگذاری و تبدیل داده‌ها

- مجموعه داده MNIST با تبدیل‌های مشخص شده تغییر اندازه، تبدیل به تانسور، نرمال‌سازی بارگیری می‌شود.

- یک DataLoader برای دسته‌بندی و ترتیب دادن داده‌ها ایجاد می‌شود.

تنظیم دستگاه

- دستگاه برای استفاده از GPU در صورت در دسترس بودن و در غیر این صورت به CPU تنظیم می‌شود.

Generator.۱-۳-۲

کلاس‌های تولیدکننده

- تولیدکننده Generator: این کلاس برای تولید تصاویر جدید است. شامل یک لایه خطی اولیه به دنبال نرمال‌سازی دسته‌ای، فعال‌سازی ReLU و یک عملیات Unflatten است. همچنین شامل بلوک‌های مقاومتی و یک لایه نهایی کانولوشن با فعال‌سازی Tanh است.

- بلوک مقاومتی ResidualBlock: یک بلوک سفارشی در تولیدکننده برای یادگیری نگاشت‌های مقاومتی استفاده می‌شود.

۲-۳-۲. Discriminator

کلاس های تمیزدهنده

- تمیزدهنده Discriminator: این کلاس بین تصاویر واقعی و جعلی تمیز می‌دهد. شامل بلوک‌های مقاومتی خاص برای تمیزدهنده و یک لایه خطی نهایی برای دسته‌بندی دودویی است.

مقداردهی اولیه وزن‌ها

- تابعی برای مقداردهی اولیه وزن‌های مدل‌ها تعریف می‌شود.

تابع ذخیره‌سازی تصاویر

- `save_fake_images_grid`: تصاویر جعلی تولید شده توسط تولیدکننده در دوره‌های مختلف ذخیره می‌شوند.

نمونه‌سازی مدل

- نمونه‌هایی از تولیدکننده و تمیزدهنده ایجاد می‌شوند.

بهینه‌سازها

- بهینه‌سازهای Adam برای هم تولیدکننده و هم تمیزدهنده تنظیم می‌شوند.

تابع ضرر

- ضرر انتروپی متقاطع دودویی BCE استفاده می‌شود.

حلقه آموزش

- GAN برای تعداد دوره‌های مشخص شده آموزش داده می‌شود. برای هر دوره، تمیزدهنده و تولیدکننده روی تصاویر واقعی و جعلی به ترتیب آموزش داده می‌شوند.

- ضرر تولیدکننده `lossG` و ضرر تمیزدهنده `lossD` محاسبه و معکوس‌سازی می‌شوند.

- ضررها در دسته‌های کوچک به عنوان مشخص شده توسط `accumulation_steps` جمع‌آوری می‌شوند.

- ضررها ردیابی و برای هر دوره چاپ می‌شوند.
- تصاویر تولید شده پس از هر دوره ذخیره می‌شوند.

نمودار ضررها

- پس از آموزش، نموداری از ضررهای تولیدکننده و تمیزدهنده نمایش داده می‌شود.

محاسبه امتیاز FID تقریبی

- تابعی برای محاسبه نسخه ساده‌شده فاصله فرشت آغازی FID تعریف و برای ارزیابی کیفیت تصاویر تولید شده استفاده می‌شود. این یک تقریب ساده است و بازتاب دهنده پیاده‌سازی واقعی FID نیست.

تولید داده‌های واقعی و جعلی برای محاسبه FID

- یک مجموعه از تصاویر جعلی با استفاده از تولیدکننده تولید می‌شود و یک مجموعه از تصاویر واقعی از مجموعه داده‌ها گرفته می‌شود.
- این تصاویر برای محاسبه FID به بردارها تبدیل می‌شوند.

محاسبه امتیاز FID

- امتیاز FID محاسبه و چاپ می‌شود و یک تقریب از شباهت بین تصاویر تولید شده و تصاویر مجموعه داده‌های واقعی را ارائه می‌دهد.

حال طی روند پیاده سازی بالا نتایج به صورت زیر به دست می اید:

شایان ذکر است که به دلیل طولانی بودن روند یادگیری میزان داده های استفاده شده در این بخش ۱۰۰۰۰ تا است که به عبارتی از ۱۵ درصد داده های استفاده شده که در ادامه قدرت عملکرد این مدل را در شرایط بحرانی تری به نمایش می گذارد.

شکل ۲۸: نتیجه در ایپاک اول **SSGAN**

شکل ۲۹: نتیجه در ایپاک ۱۰ **SSGAN**

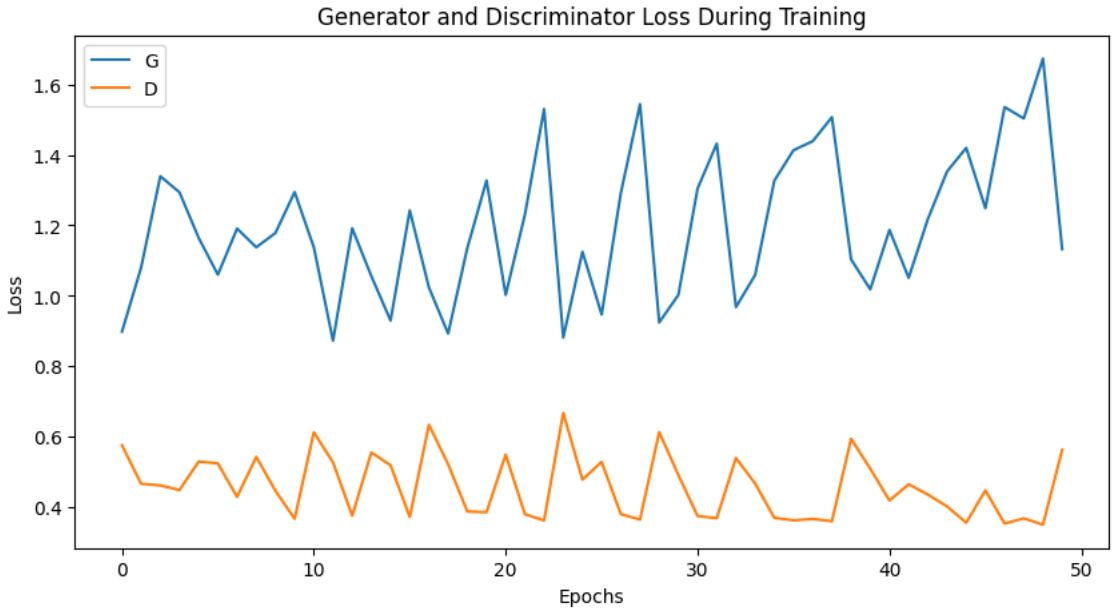
شکل ۳۰: نتیجه در ایپاک ۲۰ **SSGAN**

شکل ۳۱: نتیجه در ایپاک ۳۰ **SSGAN**

شکل ۳۲: نتیجه در ایپاک ۴۰ **SSGAN**

شکل ۳۳: نتیجه در ایپاک ۵۰ **SSGAN**

نمودار مربوط به loss این مدل به صورت زیر ارائه می شود:



شکل ۳۴: نمودار **loss** در **WGAN**

برای FID score داریم:

```
FID score (approximation): ۵۳.۴۲۲۲۶۴۹۶۰۷۷۲
```

به طورکلی داریم که :

شبکه های متخاصم مولد GANs: GAN های سنتی از دو شبکه یک مولد و یک متمایز تشکیل شده اند که به طور همزمان آموزش داده می شوند. مولد داده‌های جعلی ایجاد می‌کند، در حالی که تشخیص‌دهنده واقعی بودن داده‌ها از مجموعه داده‌ها یا جعلی بودن ایجاد شده توسط مولد را ارزیابی می‌کند. آموزش شامل یک بازی min-max است که در آن ژنراتور سعی می‌کند متمایزکننده را فریب دهد، و تمایزکننده سعی می‌کند به دقت واقعی را از تقلبی تشخیص دهد.

WGAN: WGAN Wasserstein GANs WGANs ها با استفاده از یک تابع تلفات متفاوت، Wasserstein، که فاصله حرکت دهنده زمین EMD بین توزیع داده های واقعی و تولید شده را اندازه گیری می کند، بر GAN های سنتی بهبود می بخشد. این تابع از دست دادن به تثبیت آموزش GAN ها کمک می کند و مسائلی مانند فروپاشی حالت که در آن ژنراتور نمونه های محدودی تولید می کند را کاهش می دهد.

SSGAN: SSGAN GAN های خود نظارتی ها یادگیری خود نظارتی را در چارچوب GAN ترکیب می کنند. یادگیری خود نظارتی تکنیکی است که در آن مدل برای حل یک کار بهانه ای آموزش داده می شود که به طور خودکار از داده های ورودی ایجاد می شود، به عنوان راهی برای یادگیری نمایش های مفید بدون نیاز به داده های برچسب دار. در SSGAN ها، تمایزکننده نه تنها باید بین تصاویر واقعی و جعلی تمایز قائل شود، بلکه یک کار کمکی مانند حل یک پازل، پیش بینی چرخش و غیره روی تصاویر واقعی انجام می دهد.

SSGAN ها می توانند موثرتر از GAN ها و WGAN های سنتی به دلایل زیر باشند:

یادگیری تشخیص دهنده بهبودیافته: در GAN های سنتی، تنها وظیفه تفکیک کننده تمایز واقعی از جعلی است، که گاهی اوقات می تواند یک عامل محدود کننده در یادگیری نمایش های غنی از داده ها باشد. در SSGAN ها، وظیفه نظارتی اضافی، تمایزکننده را مجبور می کند تا ویژگی های قوی و دقیق تری از داده های واقعی را بیاموزد، که می تواند منجر به درک بهتر توزیع داده شود.

ثبات در تمرین: مانند WGAN ها، SSGAN ها می توانند آموزش پایدارتری ارائه دهند. وظیفه اضافی تحت نظارت خود، گرادیان ها و اطلاعات بیشتری را برای متمایزکننده فراهم می کند، که می تواند به تثبیت فرآیند آموزش خصمانه کمک کند.

تعمیم بهتر: با یادگیری از طریق وظایف تحت نظارت خود، متمایز کننده در SSGAN ها به طور بالقوه می تواند بهتر به داده های دیده نشده تعمیم دهد. این یادگیری پیشرفته می‌تواند منجر به تولید نمونه‌های متنوع‌تر و با کیفیت‌تر توسط ژنراتور شود.

کارایی در استفاده از داده های بدون برچسب: SSGAN ها به ویژه زمانی مفید هستند که داده های بدون برچسب زیادی در دسترس باشد. آنها می توانند با یادگیری ویژگی های اضافی از طریق وظایف تحت نظارت خود، از این داده ها به طور موثرتری نسبت به GAN های سنتی استفاده کنند.

کاهش فروپاشی حالت: اهداف یادگیری اضافی در SSGAN ها می تواند به کاهش مسئله فروپاشی حالت کمک کند (جایی که ژنراتور تعداد محدودی از خروجی ها را تولید می کند)، مشکلی که اغلب در GAN های سنتی با آن مواجه می شود.