## مقدمه

در دهه‌های اخیر، پیشرفت‌های چشمگیر در حوزه‌ی یادگیری عمیق و شبکه‌های عصبی، به ویژه شبکه‌های مولد و تمییزدهنده یا به اختصار GAN، به رشد چشمگیری در زمینه‌های مختلف علمی و فناورانه منجر شده است. یکی از حوزه‌هایی که از این پیشرفت‌ها بهره‌مند شده است، حوزه‌ی تولید داده‌های تصویری و به ویژه دست‌نویس‌ها است.

تولید داده‌های دست نویس به وسیله‌ی GAN یکی از پروژه‌های جذاب در زمینه‌ی هوش مصنوعی و یادگیری عمیق است. این پروژه با استفاده از شبکه‌های مولد و متمایز کننده، توانایی تولید داده‌های دست‌نویس با ویژگی‌های واقعی و طبیعی را برای ما فراهم می‌آورد. این توانایی نه تنها در زمینه‌های هنری و خلاقیت مفید است بلکه در حوزه‌هایی چون تولید داده‌های آموزشی برای شبکه‌های عصبی، افزایش حجم داده‌های موجود، و تحقیقات در زمینه‌ی تشخیص خطا و تعمیم دهنده نیز کاربرد دارد.

هدف این پروژه توسعه یک GAN برای تولید اعداد دست‌نویس با کیفیت بالا و تنوع مناسب است. این تلاش به معرفی بهترین معماری GAN برای این مسئله منجر خواهد شد.برخورداری از داده‌های تولیدی با کیفیت واقعی می‌تواند در زمینه‌های متعددی چون افزایش تنوع داده‌های آموزشی، تحقیقات در حوزه‌ی پردازش تصویر، و حتی در زمینه‌ی تشخیص تقلب مورد استفاده قرار گیرد. در این گزارش،معرفی شبکه GAN مراحل اجرای پروژه، معماری انتخاب شده برای شبکه GAN، نتایج به دست آمده، و ارزیابی کیفیت داده‌های تولیدی به دقت مورد بررسی قرار خواهد گرفت.

## معرفی GAN

شبکه مولد تخاصمی (GAN) رویکردی برای مدل‌سازی مولد با استفاده از روش‌های یادگیری عمیق ( Deep Learning) مثل شبکه‌های عصبی پیچشی (Convolutional Neural Networks) است انواع مختلفی از شبکه‌های عصبی مصنوعی (ANN) وجود دارد که هر کدام دارای ویژگی‌ها و کاربردهای منحصر به فرد خود هستند. که در این مقاله ما به شبکه عصبی GAN میپردازیم. شبکه‌های مولد تخاصمی (GAN) سرنام (Generative Adversarial Networks) در سال ۲۰۱۴ میلادی توسط Ian Goodfellow ابداع شدند و امروزه مورد توجه متخصصان هوش مصنوعی قرار دارند. این شبکه‌ها بر مبنای رویکرد تئوری بازی‌ها پدید آمده‌اند که در آن یک شبکه یادگیری عمیق که مولد (Generator) نامیده می‌شود با یک روند کنش‌گر (تخاصمی) رقابت می‌کند. شبکه عمیق دیگری که متمایزکننده (Discriminator ) نام دارد تلاش می‌کند نمونه‌های تولید شده از شبکه مولد را از داده‌های اصلی متمایز کند. رقابت بین این دو شبکه در نهایت باعث یادگیری بهتر و بهبود عملکرد هر دو

می‌شود. در این روش شبکه می‌آموزد که چگونه از داده‌های آموزشی، داده‌های جدیدی پدید آورد، به‌طوری که از دید آماری داده‌های آموزشی یکسان ایجاد شوند. به عبارت دیگر، در رویکرد فوق داده‌هایی که برای آموزش استفاده می‌شوند و خروجی شبکه به لحاظ برخی ویژگی‌ها شبیه هستند. در شبکه‌های فوق وظیفه تولید خروجی بر عهده بخش مولد و وظیفه بررسی شباهت بر عهده متمایزکننده است.

مدل‌سازی مولد یک فعالیت نظارت نشده (Unsupervised Learning) در یادگیری ماشین است که شامل اکتشاف خودکار و یادگیری قواعد یا الگوهای موجود در داده‌های ورودی است. این‌کار به گونه‌ای انجام می‌شود که از مدل می‌توان برای تولید یا خروجی گرفتن نمونه‌های جدیدی از مجموعه داده اصلی استفاده کرد. به‌طور مثال، یک شبکه مولد تخاصمی که روی مجموعه داده تصاویر آموزش دیده باشد، قادر است تصاویر جدیدی را تولید کند که با داشتن ویژگی‌های واقع‌گرایانه مختلف توسط ناظر انسانی قابل باور باشند. در حالی که این شبکه‌ها به عنوان یک مدل مولد (Generative Model) برای یادگیری بدون ناظر معرفی شدند، اما امروزه این موضوع به اثبات رسیده که شبکه‌های مولد تخاصمی برای یادگیری نیمه نظارتی، یادگیری با ناظر و یادگیری تقویتی مفید و قابل استفاده هستند. شبکه‌های مولد تخاصمی راه‌حلی هوشمندانه برای آموزش یک مدل مولد به شمار می‌روند. شبکه‌های فوق این‌کار را با فرموله کردن مسئله به عنوان یک مسئله یادگیری نظارت شده با دو زیر مدل انجام می‌دهند. مدل مولد برای تولید نمونه‌های جدید، آموزش می‌بیند و  مدل متمایزگر تلاش می‌کند تا نمونه‌ها را به عنوان نمونه واقعی یا جعلی طبقه‌بندی کند. هر دو مدل با یکدیگر در یک بازی مجموع صفر تخاصمی آموزش داده می‌شوند و این روند تا زمانی ادامه پیدا می‌کند که مدل متمایز کننده بالغ بر نیمی از دفعات فریب بخورد، بدین معنا که مدل مولد، نمونه‌های قابل باور تولید کرده است. امروزه شبکه‌های عصبی مولد پیشرفت‌های چشم‌گیری داشته‌اند و به ویژه در زمینه تبدیل تصویر به تصویر عملکرد قابل توجهی دارند. به‌طور مثال، می‌توان تصاویر تابستان را به زمستان، تصاویر روز را به شب و تولید تصاویر فوتورئالیستیک از اشیا، صحنه‌ها و افراد و دست نوشته ها را به گونه‌ای ایجاد کرد که حتی انسان‌ها قادر به شناسایی تصاویر غیرواقعی نباشند.

## کاربرد GAN

GAN دو کاربرد عمده دارد

- تولید تصاویر جدید براساس دیتا های آموزش دیده موجود در دیتاست.
- ترمیم تصویر؛ که ممکن است بخشی از تصویر حذف و یا مسدود شده باشد.

در مسئله ی ترمیم تصویر فرض بر این است که تصویری داریم و می خواهیم کمبود و نقایص موجود در تصویر را برطرف کنیم، این کار را با جایگزینی آن با تصویرِ زمینه انجام می دهیم. فرض کنید یک تصویر از یک تعطیلات دوست داشتنی از یک صحنه ی زیبا دارید امّا یک سری افرادی که نمی شناسید نیز در تصویر

وجود دارند و باعث از بین رفتن منظره شده اند. برای برطرف کردن این ناهماهنگی در تصویر ممکن است از نرم افزار Photoshop استفاده کنیم. در اینجا دو انتخاب داریم؛ انتخاب اول این است که اگه مشابه تصویر را در دسترس داریم از آن تصویر برای بازسازی بخش مورد نیاز استفاده کنیم. که در اینصورت باید به کل تصویر نگاه کنیم و تصویر متناسب با مفهوم تصویر را برای جایگزینی انتخاب کنیم و یا به عنوان انتخاب دوم اگر مشابه تصویر در دسترس نباشد، تنها راه برای پر کردن قسمت مورد نظر این است که از پیکسل های همسایه برای پرکردن ناحیه ی مسدود شده استفاده کنیم و یا اگر بیش از حد دقت داشته باشیم، ممکن است از بخش های مشابه موجود در همان تصویر استفاده کنیم. روش اول اصطلاحاً روش مبتنی بر درک و روش دوَّم اصطلاحا روش مبتنی بر محتوا نامیده میشوند.

**مثال**:

همواره برای توضیح شبکه های مولد متخاصم از مثال جاعل G و کارآگاه D نام برده می شود. بدین صورت که جاعل اسکناس برای اولین بار اسکناسی را جعل می نماید و آن به طریقی به دست کارآگاه می رساند. کارآگاه آن را با اسکناس حقیقی مقایسه می نماید و متوجه میشود که اسکناس جعلی است. او به همکاران خود گذارش می دهد و به این ترتیب جاعل از لو رفتن اسکناس تولیدی خود مطلع می گردد. جاعل پس از آن سعی در بهبود اسکناس های جعلی می نماید. برای تلاش بعدی دوباره اسکناس جعلی بهبود داده شده را به طریقی به دست کارآگاه می رساند. کارآگاه که از دفعه ی پیش تجربه کسب کرده بود با دقت بیشتری به بررسی آن می پردازد و سعی در بررسی بیشتر ظرافات موجود در اسکناس می نماید.بدین ترتیب کارآگاه دوباره به جعلی بودن اسکناس پی میبرد. این کار تا زمانی تکرار می گردد که جاعل اسکناسی به اندازه ی کافی شبیه به اسکناس حقیقی تولید نماید که کارآگاه متوجه جعلی بودن اسکناس نگردد. پس از آن اسکناس های جعلی را به بازار عرضه می نماید.

در شبکه های مولد متخاصم، مولد همان جاعل است و تمیز دهنده همان کارآگاه است که وظیفه ی تشخیصی مهمی را بر عهده دارد

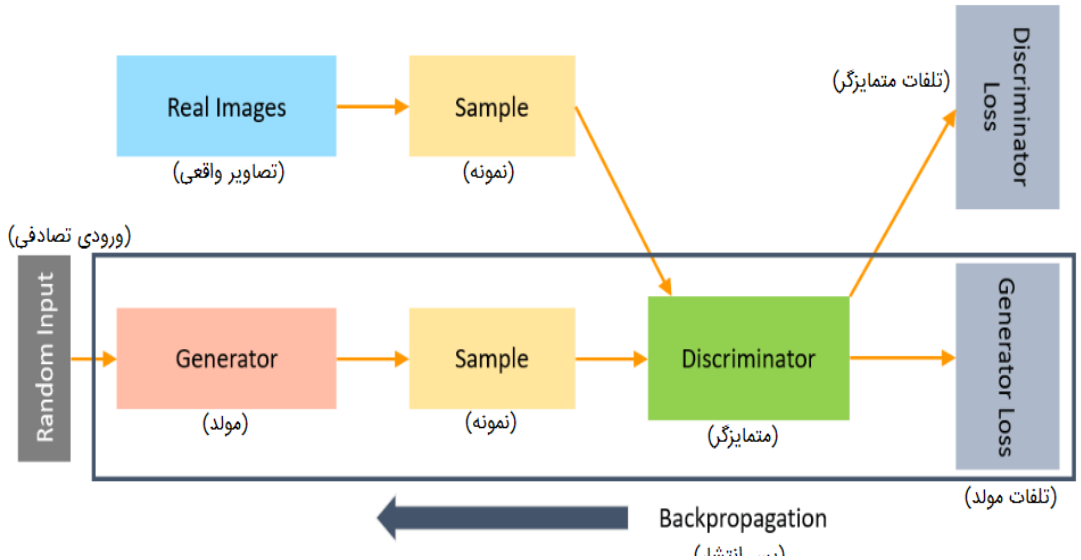
# اجزای اصلی شبکه GAN

## مدل مولد(generator)

مهمترین وظیفه مدل مولد، تولید نمونه‌های جدید از فضای پنهان است.برای این کار، ابتدا یک بردار تصادفی با طول ثابت از یک توزیع گوسی ایجاد می‌شود و به عنوان ورودی به مدل مولد داده می‌شود. سپس، مدل مولد با استفاده از این بردار تصادفی سعی در تولید نمونه‌های جدید می‌کند. این نمونه‌ها به عنوان ورودی به مدل متمایزکننده (دیگر شبکه) داده می‌شوند برای پیاده‌سازی مدل مولد در اینجا می‌توان از انواع مختلف

شبکه‌های عصبی استفاده کرد، از جمله شبکه‌های عصبی چندلایه‌ای (MLP)، شبکه‌های عصبی کانولوشنی، یا حتی سایر معماری‌های مناسب.

در فرآیند آموزش، مدل متمایزکننده سعی می‌کند بین داده‌های واقعی و تولید شده تشخیص دهد. اگر متمایز کننده نتواستد تفاوت بین این دو را تشخیص دهد، مدل مولد به‌صورت خودکار بهبود می‌یابد. این چرخه ادامه پیدا می‌کند تا زمانی که مدل مولد توانایی تولید نمونه‌هایی با ویژگی‌های واقعی را بدست آورد. در نهایت، فضای پنهان (Latent Space) که از متغیرهای پنهان به‌دست می‌آید، نقش مهمی در تفسیر داده‌ها دارد و مدل مولد از این فضا برای تولید نمونه‌های متفاوت و از نظر مفهومی یا آماری جذاب استفاده می‌کند. این مدل به این ترتیب، نه‌تنها یک ژنراتور تصادفی است بلکه یک سامانه یادگیری پیچیده است که از توزیع داده‌های واقعی یاد گرفته و می‌تواند داده‌های جدید را ایجاد کند. به طور کلی مولد یک شبکه عصبی است که داده‌های جعلی تولید می‌کند تا متمایزگر توسط آن‌ها آموزش ببیند. مولد یاد می‌گیرد که داده‌های قابل قبول تولید کند. مثال هانمونه‌های تولید شده، برای متمایز کننده، نمونه‌های منفی آموزشی به حساب می‌آیند.هدف اصلی مولد این است که متمایزگر را طوری فریب دهد که خروجی خود را با عنوان "واقعی" دسته بندی کند و از روش پس انتشار (backpropagation) برای تنظیم هر وزن در جهت مناسب با محاسبه تاثیر وزن بر خروجی استفاده می‌کند. همچنین از این روش برای به دست آوردن گرادیان استفاده می‌شود و این گرادیان‌ها می‌توانند به تغییر وزن‌های مولد کمک کنند.



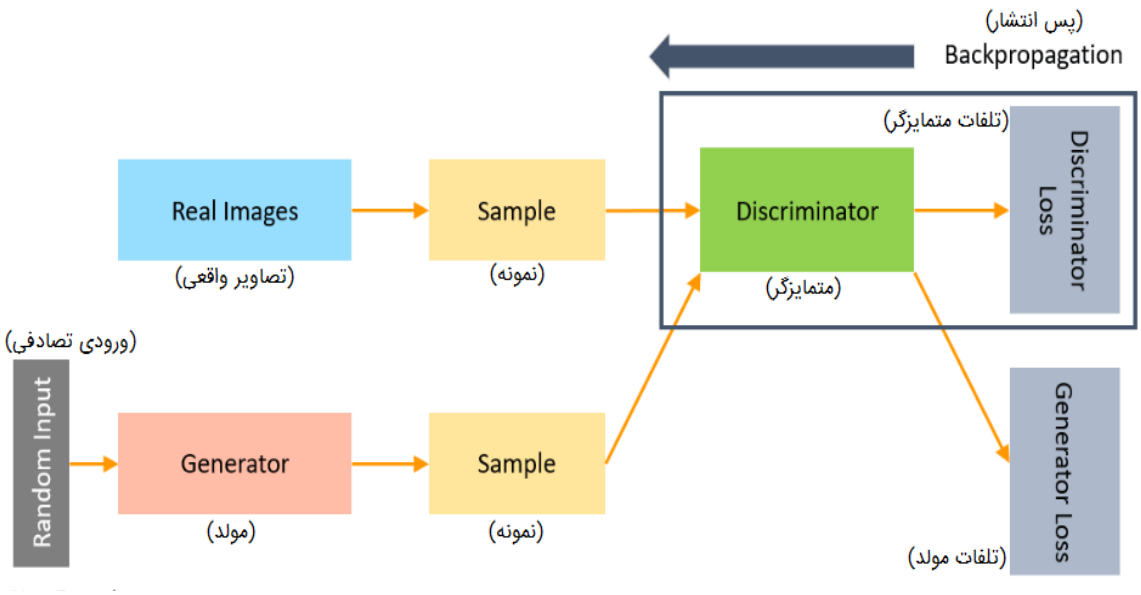
## مدل متمایزکننده(Discriminator)

متمایزکننده، عملکرد دسته‌بندی طبیعی را اجرا کرده و پس از اتمام فرآیند آموزش، به کنار گذاشته شده، زیرا توجه اصلی به مدل مولد معطوف است. احتمالاً می‌توان مدل مولد را به‌عنوان هدف اصلی مطالعه مجدد کرد، چراکه مولد به خوبی یاد گرفته‌است که ویژگی‌ها را از دامنه مرتبط با مسئله استخراج کند. لایه‌های استخراج ویژگی در مدل متمایزکننده می‌توانند در یادگیری انتقالی نیز مورد استفاده قرار گیرند. این لایه‌ها، با استفاده از داده‌های مشابه یا همان داده‌ها در حالت‌های مختلف، به عنوان یک ابزار قدرتمند در انتقال یادگیری عمل می‌کنند.برای مدل متمایز کنند هم مانند تولید کنند میتوان از شبکه عصبی MLP، شبکه عصبی کانولوشن یا هر شبکه مناسب دیگری استفاده کرد .به طور کلی، متمایزکننده به‌عنوان یک شبکه عصبی مسئول تشخیص داده‌های واقعی از داده‌های جعلی تولید شده توسط مولد عمل می‌کند. این داده‌ها از دو منبع مختلف به‌دست می‌آیند: **نمونه‌های واقعی** از داده‌های آموزشی و **نمونه‌های جعلی** تولید شده توسط مولد. در طول فرآیند آموزش، متمایزکننده تلاش می‌کند بین داده‌های واقعی و جعلی تمایز ایجاد کند و درصورت دسته‌بندی نادرست یک نمونه، توسط تلفات متمایزکننده مجازات می‌شود. هر مرحله آموزش تاثیر مستقیم بر وزن‌های متمایزکننده دارد و این وزن‌ها با استفاده از پس‌انتشار تلفات به‌روز می‌شوند . داده‌های آموزشی متمایزگر از دو منبع مختلف به دست می‌آیند:

- نمونه‌های واقعی داده‌ها، مانند تصاویر واقعی پرندگان، انسان‌ها، اسکناس‌های ارزی و غیره، توسط متمایزگر به عنوان نمونه‌های مثبت در طول آموزش استفاده می‌شوند.
- نمونه‌های جعلی ایجاد شده توسط مولد، به عنوان نمونه‌های منفی در طول فرایند آموزش استفاده می‌شوند.

در حین آموزش، متمایزگر به دو تابع تلفات (loss function) متصل است. در طول آموزش شبکه متمایزگر، از تلفات مولد چشم پوشی شده و فقط از تلفات متمایزگر استفاده می‌شود. متمایزگر، در حین فرآیند آموزش، داده‌های واقعی و داده‌های جعلی دریافتی از مولد را دسته بندی می‌کند. در صورت دسته بندی نادرست یک نمونه داده واقعی به عنوان نمونه جعلی یا برعکس، توسط تلفات متمایزگر مجازات می‌شود.

متمایزگر، وزن‌های خود را با پس انتشار از تلفات متمایزگر در طول شبکه خود، به روز می‌کند.




## GAN ها چگونه کار می‌کنند؟

در شبکه های مولد متخاصم ورودی شبکه ی مولد، نویز گوسی است و ورودی های شبکه متمایز کننده تصاویر تولیدی توسط مولد و تصاویر موجود در دیتاست می باشند. شبکه ی مولد در ابتدا با در اختیار داشتن نویز، تصویری جعلی تولید می نماید. این تصویر به متمایزدهنده می رود. اگر متمایز دهنده تشخیص دهد تصویر تولیدی توسط مولد به میزان کافی حقیقی است، تصویر به خروجی شبکه داده می شود و کار تمام است. اما اگر متمایز کننده با مقایسه با تصاویر موجود در دیتاست، تصویر را جعلی تشخیص دهد فیدبک به مولد داده می شود تا وزن های خود را بروزرسانی نماید که در نتیجه مولد تصویری حقیقی تر را تولید می نماید و این پروسه تا جایی ادامه می یابد که شبکه ی متمایز کننده متوجه جعلی بودن تصویر خروجی از مولد نشود.

GAN از دو شبکه عصبی تشکیل شده است. یک مولد G(x) و یک متمایزگر D(x). هر دوی آن‌ها یک بازی خصمانه (adversarial) انجام می‌دهند. متمایزگر سعی خواهد کرد با شناسایی داده‌های جعلی از داده‌های واقعی فریب نخورد. هر دو به طور همزمان برای یادگیری و آموزش داده‌های پیچیده مانند فایل‌های صوتی، ویدئویی یا تصویری کار می‌کنند.شبکه مولد یک نمونه را می‌گیرد و یک نمونه جعلی از داده‌ها را تولید می‌کند. مولد آموزش دیده است تا احتمال بروز اشتباه توسط شبکه متمایزگر را افزایش دهد.

همان‌طور که تا اینجا توضیح داده شد، شبکه عصبی GAN از دو شبکه D و G تشکیل شده است. شبکه G نقش مولد را دارد و داده تولید می‌کند. شبکه D نقش جداسازی داده واقعی و داده تولیدی از هم را دارد.

یکی پس از دیگری مجموعه‌ای داده جعلی/واقعی به شبکه D داده می‌شود و این شبکه به جداسازی داده‌ها می‌پردازد.

## نحوه آموزش شبکه GAN

شبکه D به دنبال این است که هر داده با هر کیفیتی از G می‌آید، به‌عنوان جعلی شناسایی کند. همچنین، هر داده از سمت واقعی را هم به‌عنوان واقعی برچسب بزند. هدف D هم شکست G و هم شناسایی داده واقعی است. شبکه G به‌دنبال این است که داده تولیدی‌اش از شبکه D به سلامت بگذرد و برچسب داده واقعی دریافت کند. هدف G شکست D.

با توضیحات بالا، همه چیز در گرو تصمیم D است. بنابراین، خروجی شبکه D است که به توابع اتلاف متصل شده است.دو تابع اتلاف Discriminator و Generator داریم که در ادامه توضیح می‌دهم.

تابع اتلاف Discriminator، طبق شکل پایین، این تابع اتلاف نتایج پیش‌بینی جعلی/واقعی شبکه D را بررسی می‌کند. مقدار اتلاف براساس میزان اشتباه‌های شبکه D محاسبه می‌شود. درنهایت، خطا پس‌انتشار و پارامترها آپدیت می‌شود. این تابع اتلاف دوجمله دارد. یک جمله برای ورودی واقعی (x) و دیگری برای ورودی جعلی (G(z)). جمله اول تابع اتلاف برای ورودی واقعی به‌صورت زیر است:

$$l_{d_1} = \log \sigma(D(x))$$

در رابطه بالا، σ تابع تحریک سیگموید هست که مقدار خروجی آن بین ۰ تا ۱ خواهد شد. زمانی‌که خروجی ۱ باشد، یعنی داده از نظر D یک داده واقعی است. اگر عدد خروجی سیگموید به عدد ۱ نزدیک باشد، مقدار لگاریتم برابر با ۰ می‌شود. یعنی اتلافی نداریم! درست است، چون شبکه D به‌درستی تشخیص داده که ورودی x یک داده واقعی است.

جمله دوم، یعنی ورودی جعلی (G(z)). می‌خواهیم که شبکه D بگوید اینها تماما جعلی هستند و خروجی ۰ برای این نوع ورودی‌ها تولید کند. پس کافی است بنویسیم:

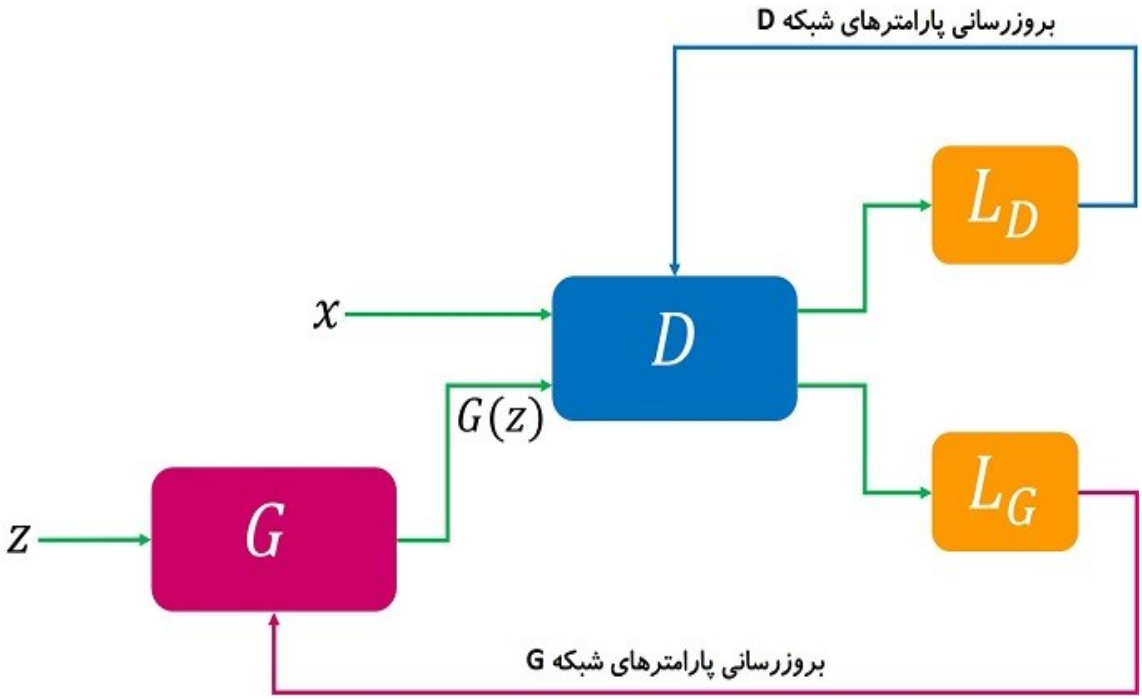
$$l_{d_2} = \log\left(1 - \sigma\left(D(G(z))\right)\right)$$

خب حالا باید دو عبارت بالا را باهم جمع کنیم و تابع اتلاف Discriminator را بنویسیم:

$$L_D = l_{d_1} + l_{d_2} = \log \sigma\big(D(x)\big) + \log\Big(1 - \sigma\big(D(G(z))\big)\Big)$$

باید LD را ماکزیمم کنیم.عبارت داخل لگاریتم‌ها عددی بین ۰ و ۱ است. لگاریتم هم برای ورودی کوچکتر از ۱، خروجی منفی می‌دهد. پس، عبارت بالا همواره کوچکتر-مساوی صفر خواهد بود. حال، باید این رابطه را ماکزیمم کنیم که به صفر برسیم  یا اینکه مثل شبکه های عصبی دیگر، یک منفی (-) پشت رابطه بالا می‌گذاریم تا هدفمان رسیدن به حداقل اتلاف باشد. آن‌وقت باید تابع بالا با یک منفی را مینیموم کنیم.

$$L_D = l_{d_1} + l_{d_2} = -\Big(\log \sigma\big(D(x)\big) + \log\Big(1 - \sigma\big(D(G(z))\big)\Big)\Big)$$




تابع اتلاف Generator، طبق شکل بالا، این تابع اتلاف براساس تشخیص D درمورد جعلی/واقعی بودن داده G تعیین می‌شود. به طور خلاصه که شبکه D به تابع اتلاف Generator می‌گوید، این داده‌ها همه جعلی هستند. تابع اتلاف G بررسی می‌کند و مقدار اتلاف محاسبه می‌شود. اما یک نکته مهم وجود دارد.حداقل کردن این عبارت یعنی موفقیت G در تولید داده جعلی در حد واقعی و فریب خوردن D در

تشخیص داده‌های جعلی. شبکه G می‌خواهد خروجی به‌سمت عدد ۱ برود. یعنی برچسب داده واقعی بخورد. پس تابع اتلاف Generator برابر است با:

$$L_G = \log \sigma \left( D\left(G(z)\right) \right)$$

رابطه LG را هم باید ماکزیمم کنیم. یا اینکه یک منفی پشت عبارت اتلاف قرار دهیم تا هدف عوض شود و LG را حداقل کنیم. این هم از تابع اتلاف Generator.

فرآیند آموزش شبکه GAN

به‌صورت خلاصه مراحل آموزش شبکه GAN را توضیح دهیم. مراحل آموزش شبکه GAN عبارتنداز

۱) یک بسته داده واقعی X انتخاب می‌کنیم.
۲) این بسته داده واقعی را به شبکه D می‌دهم. خروجی D(X) را از سیگموید می‌گذارنیم.
۳) یک بسته داده رندوم Z تولید می‌کنیم.
۴) این بسته داده رندوم را به شبکه G می‌دهیم. خروجی شبکه G معادل با داده‌های جعلی واقعی‌نماست. G(Z)
۵) داده جعلی را به شبکه D می‌دهیم. خروجی D(G(Z)) را از سیگموید می‌گذرانیم.
۶) حالا مقدار اتلاف را طبق رابطه ۱ محاسبه می‌کنیم.
۷) مقدار اتلاف را پس‌انتشار می‌دهیم تا وزن‌های شبکه D آپدیت شود.
۸) حالا می‌خواهم وزن‌های شبکه G را هم آپدیت کنیم. براساس همان خروجی مرحله ۵ ، مقدار اتلاف را با رابطه ۲ حساب می‌کنیم.
۹) مقدار اتلاف را پس‌انتشار می‌دهیم تا وزن‌های شبکه G آپدیت شود.
۱۰) ۹ مرحله بالا آنقدر تکرار می‌کنم تا به وزن‌های بهینه برای شبکه D و G برسیم.

# انواع شبکه های Generative Adversarial Networks

**GAN های وانیلی:**

Vanilla GANs دارای یک فرمول بهینه سازی حداقل حداکثر (min-max) هستند که در آن، متمایزگر یک دسته بندی باینری انجام می‌دهد و از تلفات آنتروپی متقابل سیگموئید (sigmoid cross-entropy) هنگام بهینه سازی استفاده می‌کند. مولد و متمایزگر در GAN های وانیلی، پرسپترون‌های چند لایه هستند. این الگوریتم سعی می‌کند معادله ریاضی حاکم بر GAN ها را با استفاده از گرادیان نزولی تصادفی بهینه کند.

**شبکه های مولد متخاصم عمیق کانولوشنی (Deep Convolutional Generative Adversarial Networks):**

استفاده از شبکه های عصبی کانولوشنی استفاده شده در یادگیری بدون ناظر هم در مولد و هم در تمیز دهنده. برای مثال در شبکه ی توسعه داده شده توسط Nvidia برای تولید تصاویر که در شبکه ی تمیز دهنده از CNN برای تشخیص تصویر چهره ی حقیقی از غیر حقیقی مورد استفاده قرار گرفته است و در شبکه ی مولد برای تولید تصویر صورت از یک سری دیتای اولیه ( در اینجا نویز گوسی) و با استفاده از DeCNN تصویر حقیقی تر و با کیفیت بالاتری تولید گردیده است.

**شبکه ی مولد متخاصم شرطی (Conditional Generative Adversarial Networks):**

می توان به شبکه امر کرد که چه نوع دیتایی تولید نماید. به عنوان مثال دیتاست اعداد ۰ تا ۹ را در نظر بگیرید که هر کدام از آنها در شبکه ی مولد متخاصم معمول شبکه قادر به تولید تصاویر رندوم از اعداد است. اما در این نوع شبکه ما می توانیم با تغذیه ی ورودی C یک شرط برای آن تعریف نماییم تا تنها مورد دلخواهمان را تولید کند.

**شبکه ی مولد متخاصم اطلاعات (Info Generative Adversarial Networks):**

این شبکه علاوه بر توانایی تولید تصاویر قادر به یادگیری متغیرهای معنی دارِ پنهان موجود در تصویر بدون وجود هیچگونه برچسب در دیتای ورودی می باشد. به عنوان مثال در تصاویر آموزش داده شده ی اعداد ۰ تا

۹ موجود در دیتاست قادر است زاویه ی اعداد و یا ضخامت و حرکت اعداد را بدون وجود هیچگونه لیبل مشخصی در این خصوص، فرا گرفته و به تولید تصاویر با ویژگی های جدید نامبرده نماید.
همانطور که در تصویر نشان داده شده است هر ردیف که به پایین می آییم ضخامت و یا زاویه و حرکت اعداد تغییر می نماید.

**شبکه های مولد متخاصم Wasserstein**:

در شبکه های موجود، امکان اشتباه در بخش تابع هدف متمایز کننده که به منظور افزایش loss، به دلیل اینکه هیچ نشانه ی واضحی برای توقف نیاز به نگاه به نمونه های دیتاست و تشخیص حقیقی بودن دیتای تولیدی از دیتاست در روش های متداول مورد استفاده برای حداقل کردن Loss مولد وجود ندارد( روش Jenson-Shannon Divergence). در حقیقت این روش، روشی برای اندازه گیری شباهت میان دو توزیع احتمال است .روش جدید معرفی شده در این الگوریتم توانایی پیداکردن فاصله ی نقاط در توزیع احتمال را با استفاده از فاصله ی موجود در تصاویر دیتاست دارد. بدین صورت شبکه قادر به یادگیری تا رسیدن به همگرایی می شود که در نتیجه ی آن، تصاویری با کیفیت بالاتر نمونه های تولیدی توسط مولد را شاهد خواهیم بود.

**شبکه های مولد متخاصم توجه (Attention Generative Adversarial Networks)**:

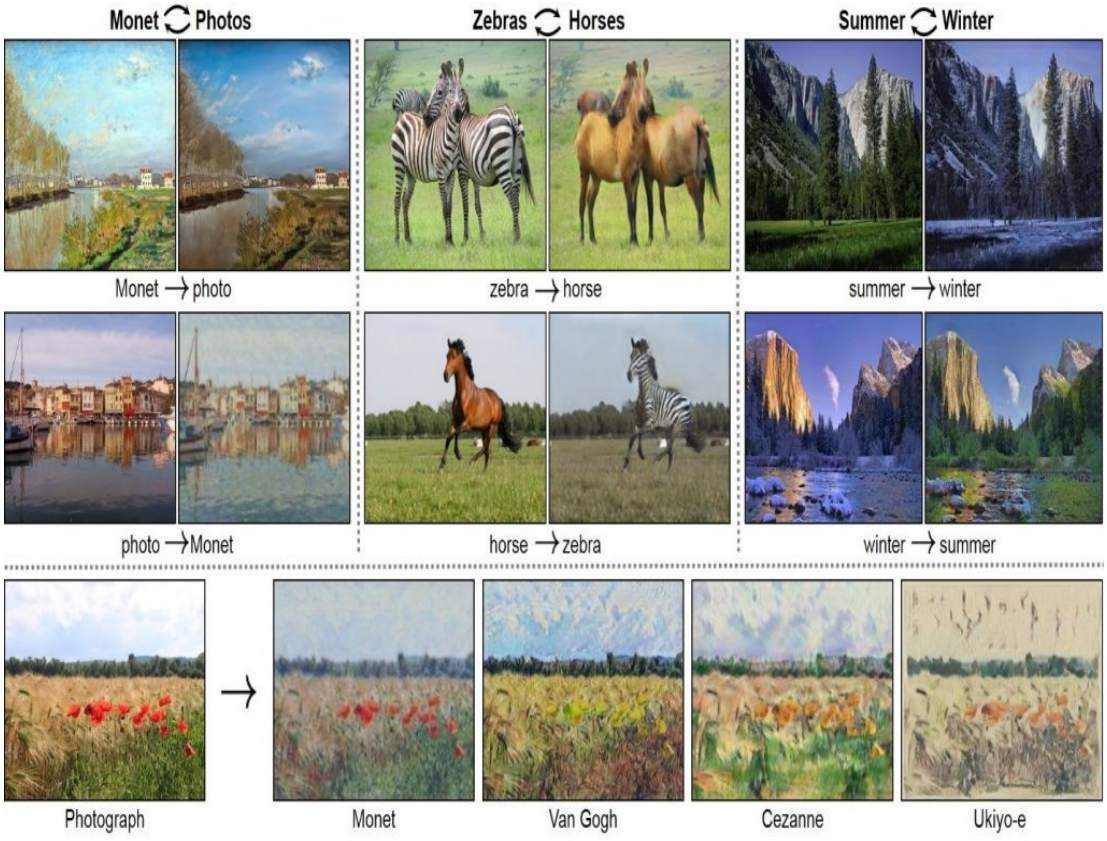
این شبکه ها توسط مایکروسافت به منظور تولید تصاویر از متن از طریق پردازش زبان طبیعی خلق شدند و عملکرد خارق العاده ای مانند تولید بخشی از تصویر از یک لغت تنها را دارند. بدین صورت که با استفاده از الگوریتم بخشی از تصویر با کیفیت بالا تولید شده و به سایر بخش ها کیفت یا توجه کمتری داده می شود؛ بنابراین توجه بر قسمت خاصی از تصویر اتفاق می افتد. با واضح شدن لغات و افزایش درک شبکه از متن

( شناخته شدن موضوع توسط شبکه ) بخش های اطراف و محیط پیرامون تصویر در بخش توجه شده واضح تر شده و به تصویر مرتبط تری با لغات موجود در متن دست پیدا خواهد کرد.

**شبکه های مولد متخاصم چرخه (Cycle Generative Adversarial Networks):**

تولید تصاویر جفت برای شبکه های مولد متخاصم کار دشواری نمی باشد. بطور مثال تولید تصویر واقعی از یک شکلِ کفش نقاشی شده! اما کار زمانی مهیج می شود که یک تصویر منظره از طبیعتی تابستانی داشته باشیم و بتوانیم طبیعت تصویر را تغییر دهیم و همان منظره ی زیبای تابستانی را به تصویری زمستانی تبدیل نماییم . الگوریتم سیکل یا چرخه بدین منظور و موارد مشابهی از این دست، طراحی شده است.

یک ایده برای این الگوریتم آن است که شبکه فرا می گیرد که بهار چیست و چه ویژگی هایی دارد. سپس تصویر پاییزی را به تصویر بهاری تبدیل می نماید. سپس تصویر بهاری تولید شده در مرحله ی قبل را دوباره به تصویر پاییزی تبدیل می نماید و سپس با مقایسه ی تصویر تولیدی با تصویر اصلی هم ویژگی ها را فرا می گیرد و هم خطا را طی فرایند آموزش کاهش خواهد داد.



# پیاده سازی یک مدل GAN با استفاده از پایتون

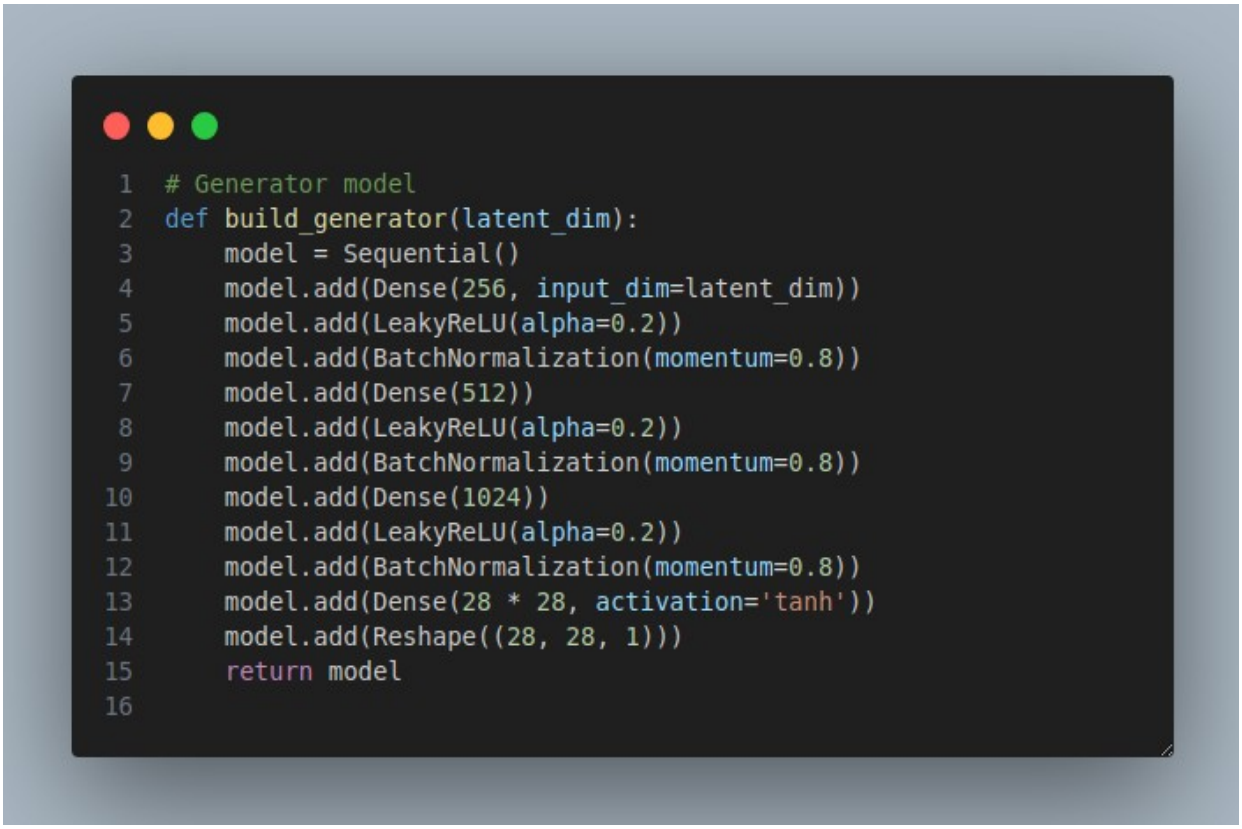
برای پیاده سازی یک شبکه Gan ما از tensorflow و دیتاست آماده MNIST استفاده کردیم برای تولید عکس اعداد نوشته شده به صورت دست نویس انسان ها.

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, LeakyReLU, BatchNormalization, Reshape, Flatten, Input
from tensorflow.keras.optimizers import Adam
```

در ابتدا کتابخانه‌های مورد نیاز برای پیاده‌سازی GAN و پردازش تصویر از جمله NumPy، Matplotlib، و TensorFlow اضافه میکنیم ولایه های مورد نیاز برای شبکه عصبی.

```

# Load and preprocess the MNIST dataset
(X_train, _), (_, _) = mnist.load_data()
X_train = (X_train.astype(np.float32) - 127.5) / 127.5
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)

```

سپس داده‌های دست‌نویس از مجموعه داده MNIST بارگیری شده و به شکلی مناسب برای ورودی شبکه GAN تغییر شکل داده ایم در اینجا ما فقط به داده های اموزش نیاز داریم و نیاز به داده های تست نیست. سپس این داده ها رو مقدارش رو به -۱ تا ۱ تبدیل میکنیم که -۱ نشان دهنده سیاه هست و هرچه به سمت ۱ بریم نشان دهنده سفید است.



```python
# Generator model
def build_generator(latent_dim):
    model = Sequential()
    model.add(Dense(256, input_dim=latent_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(28 * 28, activation='tanh'))
    model.add(Reshape((28, 28, 1)))
    return model

```

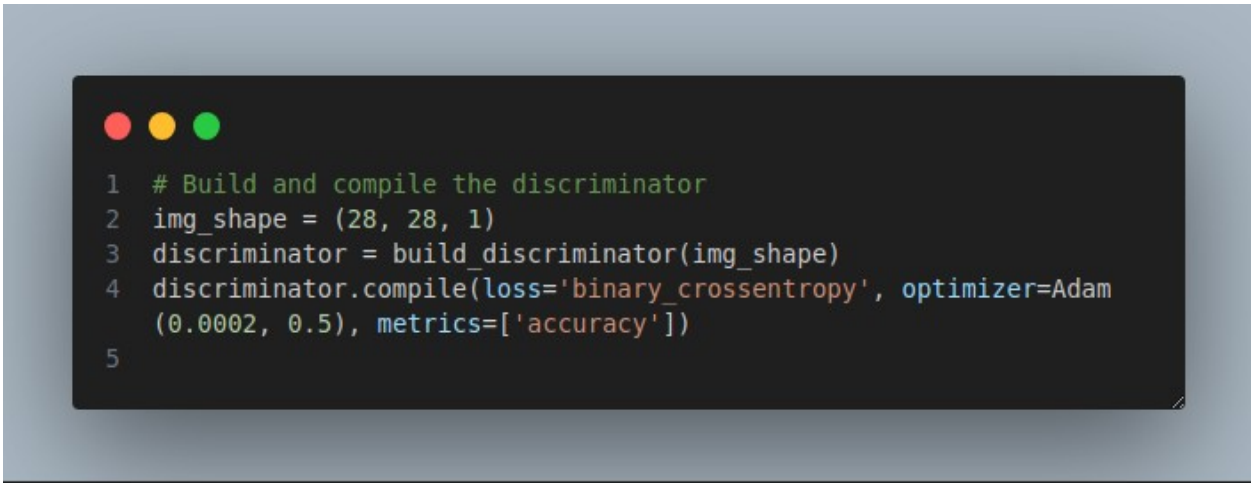
در ادامه یک تابع مول میسازیم تابع build_generator این مدل با استفاده از یک فضای نهان با ابعاد مشخص latent_dim که به صورت ورودی میدیم ، تصاویر مصنوعی با ابعاد ۲۸X۲۸ پیکسل و یک کانال (سیاه و سفید) تولید می‌کند. معماری این مدل شامل لایه‌های Dense، LeakyReLU، BatchNormalization و Reshape است. استفاده از توابع فعال‌سازی LeakyReLU و تانژانت هایپربولیک، همچنین افزودن Batch Normalization به منظور بهبود آموزش و استقرار مدل صورت گرفته است.در این تابع ما در ابتدا یک sequential تعریف میکنیم سپس یک لایه نورون ۲۵۶ تایی که یک لایه کاملاً متصل است شروع می‌کنیم. این لایه برای ایجاد اتصالات اولیه از فضای نهان ( latent space) به شبکه استفاده می‌شود.سپس یه لایه. LeakyReLU. این لایه کمک می‌کند که حتی با اعداد منفی هم مدل اطلاعات را حفظ کند. سپس برای یکBatch Normalization اضافه میکنیم که به مدل کمک می‌کند که بهتر یاد بگیرد و در عین حال تغییرات آموزشی را کنترل کند.برای لایه بعدی باز یک لایه کاملاً متصل با تعداد نرون‌های بیشتر تعریف میکنیم که . این کمک می‌کند تا مدل ویژگی‌های پیچیده‌تری را از فضای نهان استخراج کند. برای حفظ انعطاف‌پذیری مدل باز از LeakyReLU استفاده ازمیکنیم یکباردیگه از Batch Normalization استفاده می‌کند تا فرآیند آموزش را پایدارتر کند. و این حلقه بار دیگر انجام میدم اینبار با نورون های بیشتر برای استخراج ویژگی‌های پیچیده‌تر.

در اخر لایه‌ای که تصاویر تولیدی را با استفاده از تابع tanh تولید می‌کنیم. tanh به ما مقادیر در محدوده (-۱, ۱) می‌دهد، که برای تصاویر خوب است. و در نهایت((۱ ,۲۸ ,۲۸))Reshape این لایه برای تغییر ابعاد خروجی به شکل تصویر مورد نظر (۲۸x۲۸ با یک کانال) استفاده می‌شود.

```python
# Discriminator model
def build_discriminator(img_shape):
    model = Sequential()
    model.add(Flatten(input_shape=img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))
    return model
```

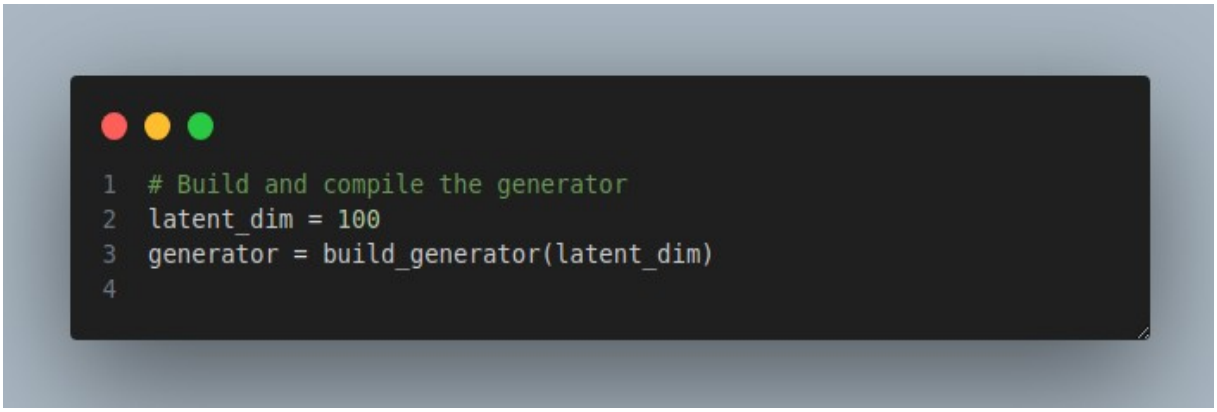
در مرحله بعدی نوبت تعریف تابع build_discriminator که یک مدل متمایز کننده برای GAN ایجاد می‌کند. این مدل با ورودی گرفتن تصاویر که واقعی هستن ، سعی در تشخیص اینکه تصویر مورد نظر واقعی است یا مصنوعی. معماری این مدل شامل لایه‌های Dense، Flatten، و LeakyReLU است که در ادامه هر کدام را توضیح میدهیم، و با یک لایه خروجی Dense با تابع فعال‌سازی sigmoid انتها می‌یابد. توابع فعال‌سازی LeakyReLU برای جلوگیری از مشکل ناپدید شدن گرادیان و تسهیل آموزش مدل استفاده شده‌است.در ابتدا یک لایه Flatten این لایه برای تغییر ابعاد تصاویر ورودی به یک بردار یک بعدی استفاده می‌شود. این لایه به شبکه اطلاع می‌دهد که ورودی از نوع تصویر با ابعاد img_shape است.سپس(۵۱۲)Dense یک لایه کامل متصل است با ۵۱۲ نورون. این لایه ویژگی‌های تصاویر را استخراج می‌کند. در ادامه یک لایه Leaky ReLU با ضریب نشتی (leakage) برابر با ۰.۲. اضافه میکنیم این تابع فعال‌سازی به شبکه این امکان را می‌دهد که حتی برای ورودی‌های منفی هم اطلاعات را حفظ کند.بار دیگر یک لایه (۲۵۶)Dense با ۲۵۶ نورون که برای استخراج ویژگی‌های پایین‌تر است و با افزودن یک لایه Leaky ReLU با ضریب نشتی ۰.۲ برای حفظ انعطاف‌پذیری شبکه.و در انتها لایه (۱, activation='sigmoid')Dense با ۱ نرون و تابع فعال‌سازی sigmoid که

احتمال تصویر ورودی بودن یا نبودن را تولید می‌کند. این لایه در واقع تصمیم می‌گیرد که تصویر ورودی واقعی است یا مصنوعی (تولید شده توسط مولد)



```
# Build and compile the discriminator
img_shape = (28, 28, 1)
discriminator = build_discriminator(img_shape)
discriminator.compile(loss='binary_crossentropy', optimizer=Adam
(0.0002, 0.5), metrics=['accuracy'])

```

در مرحله بعدی ، مدل متمایز کننده برای GAN ایجاد و کامپایل میکنیم. این مدل با ابعاد تصاویر ورودی ۲۸X۲۸ و یک کانال (سیاه و سفید) طراحی شده است. برای آموزش، از تابع هزینه binary_crossentropy برای متمایز کننده استفاده شده و بهینه‌ساز Adam با نرخ یادگیری ۰.۰۰۰۲ و پارامتر بتا ۰.۵ انتخاب میکنیم. علاوه بر این، معیار دقت نیز به عنوان معیار ارزیابی در نظر گرفته گرفته ایم. این مدل حالا آماده به کار برای تشخیص تصاویر واقعی از تصاویر مصنوعی تولید شده توسط مولد در فرآیند آموزش می‌باشد.



```
# Build and compile the generator
latent_dim = 100
generator = build_generator(latent_dim)

```

سپس مدل مولد برای GAN ایجاد میکنیم. این مدل با فضای نهان به ابعاد ۱۰۰، تصاویر مصنوعی به ابعاد ۲۸X۲۸ و یک کانال تولید می‌کند. در این مرحله، مدل مولد فقط ایجاد شده است و هنوز کامپایل نشده‌است. در فرآیند آموزش GAN، مدل مولد جداگانه آموزش داده می‌شود.

```
# Build the GAN model
discriminator.trainable = False
gan_input = Input(shape=(latent_dim,))
x = generator(gan_input)
gan_output = discriminator(x)
gan = Model(gan_input, gan_output)
gan.compile(loss='binary_crossentropy', optimizer=Adam(0.0002, 0.
5))

```

در ادامه مدل GAN که شامل مولد و متمایز کننده است، ساخته شده است. متمایز کننده در این مرحله به حالت غیرقابل آموزش درآمده است تا در فرآیند آموزش GAN، تنها مولد آموزش داده شود. مدل GAN با تابع هزینهbinary_crossentropy و بهینه‌ساز Adam با نرخ یادگیری ۰.۰۰۰۲ و پارامتر بتا ۰.۵ کامپایل شده است. این مدل به عنوان یک واحد کلی برای تولید تصاویر جدید و ارزیابی آنها توسط متمایز کننده عمل خواهد کرد.

```
# Training the GAN
epochs = 10000
batch_size = 64
save_interval = 5000
```

برای شروع آموزش ما پارامترهای آموزش برای GAN تعیین میکنیم که

epochs = ۱۰۰۰۰ : تعداد دوره‌های آموزش.

batch_size = ۶۴ : اندازه دسته‌های داده در هر مرحله.

save_interval = ۵۰۰۰ : فاصله زمانی بین هر ذخیره‌سازی مدل و تصاویر تولید شده. این پارامترها برای نظارت و تنظیم فرآیند آموزش GAN به کار می‌بریم.

```python
for epoch in range(epochs + 1):
    idx = np.random.randint(0, X_train.shape[0], batch_size)
    real_images = X_train[idx]

    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    generated_images = generator.predict(noise)

    labels_real = np.ones((batch_size, 1))
    labels_fake = np.zeros((batch_size, 1))

    # Train the discriminator
    d_loss_real = discriminator.train_on_batch(real_images, labels_real)
    d_loss_fake = discriminator.train_on_batch(generated_images, labels_fake)
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

    # Train the generator
    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    labels_gan = np.ones((batch_size, 1))
    g_loss = gan.train_on_batch(noise, labels_gan)

```

سپس در یک حلقه باید آموزش را شروع کنیم در این حلقه آموزش GAN، دو فرآیند اصلی اجرا می‌شود. ابتدا، دسته‌ای از تصاویر واقعی از مجموعه داده برگزیده می‌شود و توسط متمایز کننده (Discriminator) آموزش داده می‌شود تا تشخیص دهد که تصاویر واقعی هستند. سپس، با ایجاد نویز تصادفی و اعمال آن به مولد (Generator)، تصاویر مصنوعی تولید می‌شوند. این تصاویر مصنوعی نیز توسط متمایز کننده مورد ارزیابی قرار گرفته و مولد سعی می‌کند تا تصاویری بسازد که متمایز کننده را به اشتباه بگیراند. پس از هر دوره، هزینه تمیزدهنده و مولد گزارش شده و ۲۵ تصویر مصنوعی برای ارزیابی کیفیت تولیدات نمایش داده می‌شوند. همچنین، به صورت دوره‌ای، مدل GAN و تصاویر تولید شده ذخیره می‌شوند تا در ادامه آموزش قابل دسترس باشند. این فرآیند متناوب ادامه می‌یابد تا مدل به بهبود و تولید تصاویر با کیفیت واقعی‌تر برسد.

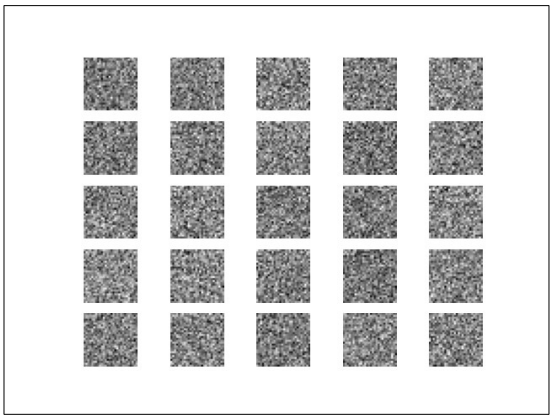
## سپس در ادامه حلقه:

```python
# Print progress
    if epoch % 100 == 0:
        print(f"{epoch}/{epochs} [D loss: {d_loss[0]} | D accuracy: {100 * d_loss[1]}] [G loss: {g_loss}]")

        # Generate and save sample images
        generated_images = generator.predict(np.random.normal(0, 1, (25, latent_dim)))
        generated_images = 0.5 * generated_images + 0.5  # Rescale images to [0, 1]
        fig, axs = plt.subplots(5, 5)
        cnt = 0
        for i in range(5):
            for j in range(5):
                axs[i, j].imshow(generated_images[cnt, :, :, 0], cmap='gray')
                axs[i, j].axis('off')
                cnt += 1
        plt.savefig(f"gan_generated_image_epoch_{epoch}.png")
        plt.close()

    # Save models and generate images at specified intervals
    if epoch % save_interval == 0:
        # Save GAN model in a single file
        gan.save(f"gan_model_epoch_{epoch}.h5")
```

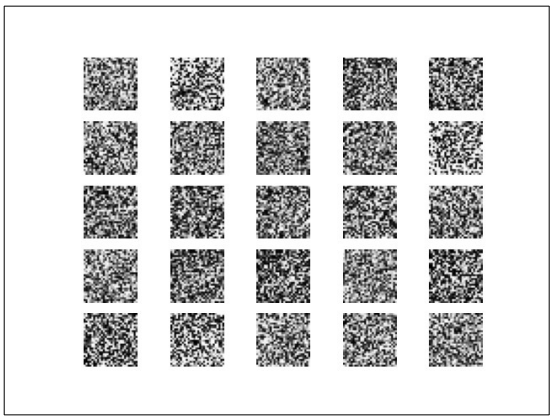
در هر صد دوره از آموزش، اطلاعات مربوط به هزینه متمایز کننده و مولد گزارش می‌شود. همچنین، ۲۵ تصویر مصنوعی توسط مولد تولید و نمایش داده می‌شوند. هدف این قسمت ارزیابی کیفیت تولیدات مولد در طول زمان است. هر ۵۰۰۰ دوره، مدل GAN و تصاویر تولید شده به عنوان نمونه‌ها ذخیره می‌شوند تا در ادامه آموزش قابل دسترس باشند.این ذخیره سازی این مدل این امکان را میدهد که هر زمان که خواستیم باز بارگیری کنیم و داده تولید کنیم که درادامه به آن میپردازیم:

# نتیجه اجرای کد

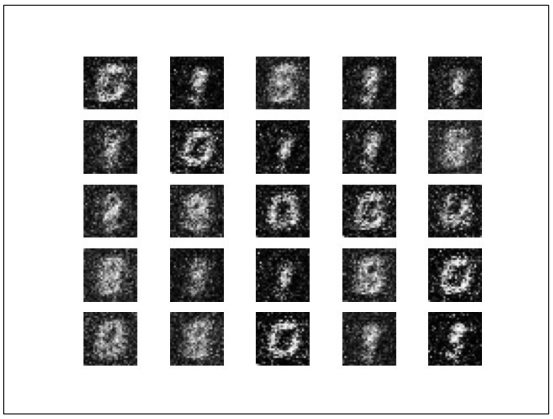
Epoch : 0



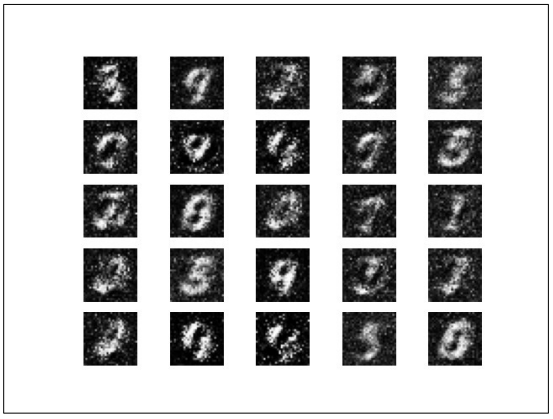
Epoch : 100



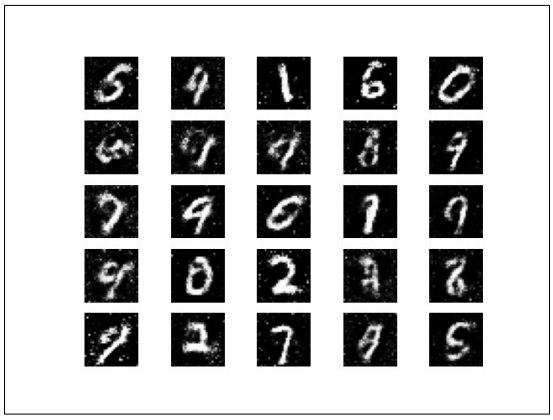
Epoch : 500



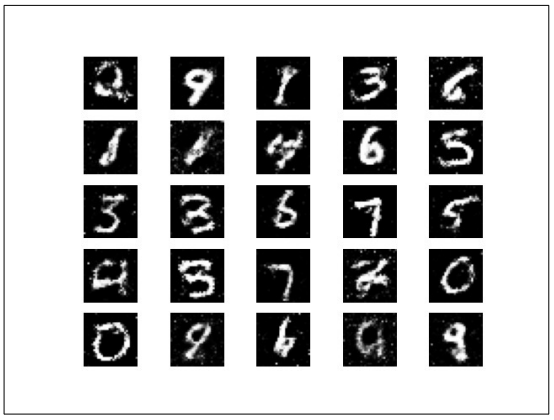
Epoch : 1000



Epoch : 5000



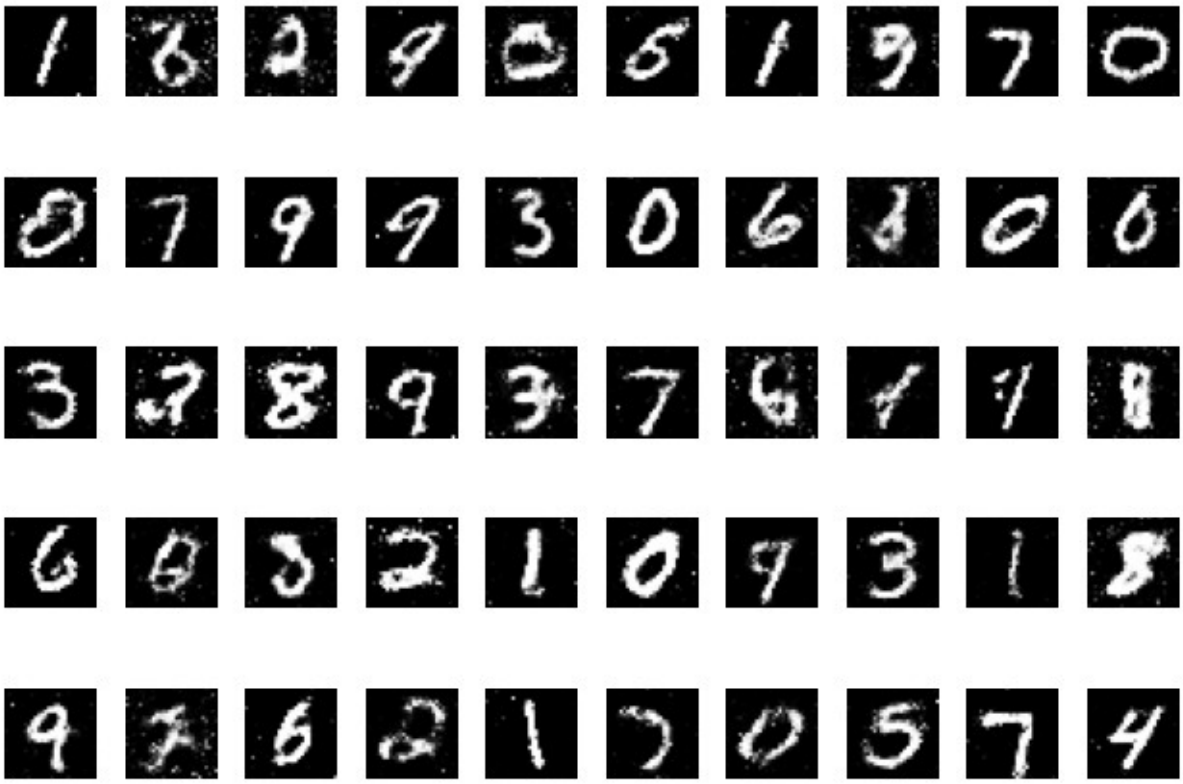
Epoch : 10000



بارگیری دوباره مدل و استفاده برای تولید عکس:

```python
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.random import normal
import h5py

# Load the trained GAN model
gan_model_path = "gan_model_epoch_10000.h5"
gan_model = load_model(gan_model_path)

# Generate new images
num_generated_images = 50
latent_dim = 100

# Generate random noise samples as input to the generator
random_latent_vectors = normal(mean=0, stddev=1, shape=(num_generated_images, latent_dim))

# Generate images using the generator model
generated_images = gan_model.layers[1](random_latent_vectors)

# Rescale the generated images to the range [0, 1] if necessary
generated_images = 0.5 * generated_images + 0.5

# Save the generated images to an H5 file
output_h5_path = "generated_images.h5"
with h5py.File(output_h5_path, "w") as hf:
    hf.create_dataset("generated_images", data=generated_images.numpy())

# Display or save the generated images (optional)
for i in range(num_generated_images):
    plt.subplot(5, 10, i + 1)
    plt.imshow(generated_images[i, :, :, 0], cmap="gray")
    plt.axis("off")

plt.tight_layout()
plt.show()
```

در کد با ابتدا مدل ذخیره شده در کد قبلی را لود میکنیم و تعداد عکس های که میخواهیم ساخته شود را وارد میکنیم سپس مقدار نویز برای مدل مولد وارد میکنیم و در نهایت عکس رو نمایش میدهیم نتیجه اجرای کد بالا:



# نتیجه گیری

شبکه Generative Adversarial Network (GAN) در این پروژه با موفقیت آموزش داده شده و توانسته است تصاویر مصنوعی با ویژگی‌های مشابه به تصاویر واقعی MNIST تولید کند.توانایی شبکه در تولید تصاویر با پیچیدگی متوسط بالا است. تصاویر تولیدی به خوبی شکل و جزئیات اعداد را نمایش می‌دهند. پیشرفت آموزش از طریق نمایش مقادیر خطاها و دقت دیسکریمیناتور و خطای ژنراتور در هر ۱۰۰ ایپوک نشان داده شده است.نتایج نهایی از طریق تصاویر تولیدی در هر مرحله نیز آشکار می‌شود. این تصاویر می‌توانند نشانگر توانایی ژنراتور در تولید تصاویر جدید با سطوح مختلف پیچیدگی باشند. با این حال، تصاویر تولیدی هنوز ممکن است در برخی جزئیات ناکام باشند و نیاز به بهینه‌سازی بیشتر داشته باشند. این پیاده‌سازی اولیه یک شبکه GAN موفق به ایجاد تصاویر قابل قبول با پیچیدگی متوسط است، اما امکان بهبود با تنظیمات و پارامترهای بهینه‌تر وجود دارد.

برای بهبود نتایج، می‌توان با تغییر پارامترهای مختلف مانند تعداد ایپوک‌ها، نرخ یادگیری، تعداد و اندازه لایه‌ها و نحوه استفاده از Batch Normalization آزمایشات انجام دهید. همچنین، ممکن است افزودن لایه‌های مخفی یا تغییر ساختار مدل‌ها به بهبود عملکرد شبکه کمک کند.