دانشکده ریاضی و علوم‌کامپیوتر

شبکه هاي مولد مقابله اي براي اعداد دست نویس فارسي

پروژه کارشناسی علوم کامپیوتر گرایش مدل های مولد مقابله ای شرطی CoGANs

امیرمهدی ابوطالبی

استاد راهنما

استاد مصطفی شمسی

تابستان ۱۴۰۲

تأییدیه‌ی هیأت داوران جلسه‌ی دفاع از پروژه

نام دانشکده: دانشکده ریاضی و علوم‌کامپیوتر

نام دانشجو: امیرمهدی ابوطالبی

عنوان پروژه : شبکه هاي مولد مقابله اي براي اعداد دست نويس فارسي

تاریخ دفاع: تابستان ۱۴۰۲

رشته: علوم کامپیوتر

گرایش: مدل های مولد مقابله ای شرطی CoGANs

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضـــــــــا
۱	استاد راهنما	دکتر مصطفی شمسی	دانشیار	دانشگاه صنعتی امیرکبیر	

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب امیرمهدی ابوطالبی به شماره دانشجویی **۹۸۱۳۰۳۴** دانشجوی رشته علوم کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پروژه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: امیرمهدی ابوطالبی

تاریخ و امضا:

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.

☐ بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استاد راهنما: استاد مصطفی شمسی

تاریخ:

امضا:

قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را زیور عقل آراست.

در آغاز وظیفه خود می‌دانم از زحمات بی‌دریغ استاد راهنمای خود، جناب آقای دکتر مصطفی شمسی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی‌رسید.

در پایان، بوسه می‌زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس‌شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

امیرمهدی ابوطالبی

تابستان ۱۴۰۲

چکیده

در این پروژه، هدف ما تولید اعداد فارسی دستنویس با استفاده از شبکه‌های مولد تخاصمی (GANs) [1] است. اعداد دستنویس فارسی، از اهمیت بالایی برخوردارند زیرا در بسیاری از حوزه‌ها از جمله تشخیص متن، تحقیقات تشخیص امضا و تحقیقات تشخیص اعداد، مورد نیاز هستند، همچنین امروزه داده بسیار ارزش مند و گرانقیمت است به همین دلیل است که شبکه های مقابله ای بسیار کارآمد و پرطرفدار شده اند زیرا در کمترین زمان با کمترین هزینه قادر به تولید دیتا های قابل اطمینانی می باشند. با این حال، تولید دستنویس فارسی با استفاده از روش‌های سنتی با محدودیت‌هایی همراه است. به همین دلیل، استفاده از شبکه‌های مولد تخاصمی می‌تواند به عنوان روشی نوآورانه برای تولید دستنویس فارسی مفید باشد.

در این پروژه سعی شده است تا با استفاده از شبکه های مولد شرطی [2] اعداد را تولید کنیم تا دیگر با مشکلات تصادفی بودن داده ها که البته به آن مشکلات هم اشاره کوچکی خواهیم کرد رو به رو نشویم در واقع قادر هستیم تا داده های مورد نظرمان را با توجه به نوع اعدادی که می خواهیم تولید کنیم. همینطور سعی شده است تا به ابر پارامتر ها بهترین مقدار داده شود تا مدل به دست آمده بهترین دقت را داشته باشد و همین طور سعی شده است تا از بیش برازش نیز جلوگیری شود.

در این پروژه از داده های دیتاست هدا استفاده شده است، که دارای تصاویر باینری اعداد ۰ تا ۹ می باشد. که به تفصیل آن را برسی خواهیم کرد. در این گزارش، ابتدا به معرفی موضوع تولید اعداد فارسی دستنویس و اهمیت این تحقیق می‌پردازیم. سپس چالش‌ها و محدودیت‌های روش‌های موجود در حوزه تولید اعداد دستنویس را بررسی می‌کنیم. سوالات پژوهشی و اهداف این پروژه را معرفی خواهیم کرد.

واژگان کلیدی: GANs شبکه های مولد مقابله ای، Generative Adversarial Networks

[1] Generative Adversarial Networks

[2] Conditional Generative Adversarial Networks

فهرست مطالب

فصل ۱

مقدمه

۱-۱ پیش‌زمینه

در حال حاضر، روش‌های مختلفی برای تولید دستنویس اعداد وجود دارد. اما برخی از این روش‌ها نتایج قابل قبولی در مورد تولید دستنویس فارسی ارائه نمی‌دهند. به همین دلیل، استفاده از شبکه‌های مولد تخاصمی می‌تواند به عنوان روشی نوآورانه و قدرتمند در این حوزه مورد استفاده قرار گیرد. شبکه‌های مولد تخاصمی امکان تولید اعداد دستنویس واقع‌گرایانه و بهبود یافته را فراهم می کند. در این پروژه، ما قصد داریم یک مدل GAN را برای تولید اعداد فارسی دستنویس طراحی و پیاده‌سازی کنیم. در این مدل، شبکه تولید کننده [۱] سعی می‌کند اعداد دستنویس‌های غیرواقعی و قابل قبولی را تولید کند، در حالی که شبکه تمیز دهنده [۲] سعی می‌کند بین دستنویس‌های تولیدی و دستنویس‌های واقعی تشخیص دهد. این رقابت بین دو شبکه باعث بهبود هر دو مدل می شود.

در واقع در شبکه های مولد تخاصمی، هر سری مدل تولید کننده با تولید یک تصویر کاملا ساختگی و غیر واقعی سعی بر این دارد که مدل تمیز دهنده ما را به خطا بیندازد. مدل تمیز دهنده یک تشخیص دهنده دودویی [۳] است که باید تصمیم بگیرد که آیا مقدار ورودی دریافت شده واقعی است یا خیر. در واقع مدل تولید کننده در هر سری از اجرا یک دسته تصویر کاملا ساختگی را به مدل تمیز دهنده میدهد و هدفش این هست که

[۱] شبکه تولید کننده Generatore Model

[۲] شبکه تمیز دهنده Discriminator Model

[۳] تشخیص دهنده دودویی Binary Classification

مقدار خطا بیشترین مقدار باشد تا تمیز دهنده به اشتباه افتاده شود و تصویر تولید شده را به عنوان یک تصویر واقعی طبقه بندی کند. از سوی دیگر مدل تمیز دهنده که یک طبقه بندی کننده دودویی بود با گرفتن ورودی از مدل تولید کننده سعی بر بهبود خود دارد پس می خواهد خطای عملی خود را به حداقل برساند تا تصاویر غیر واقعی را تشخیص دهد. این فرایند هر سری باعث بهبود هر دو مدل تمیز دهنده و تولید کننده می شود.
در ادامه، به طور خلاصه با یکی از مشکلات این فرایند آشنا می شویم و در مراحل بعد به طور کامل آن را مورد بحث قرار خواهیم داد. یکی از مشکلات پیش روی این مدل ها آموزش همزمان آن ها است. در واقع همان طور که اشاره کردیم این دو مدل تمیز دهنده و تولید کننده هر دو بایستی در کنار هم آموزش داده بشوند تا با خروجی های همدیگر خود را بهبود بدهند. اما اگر می خواستیم هر کدام را جدا آموزش بدهیم آنگاه زمانی که تمیز دهنده آموزش داده شود و بخواهیم با استفاده از خروجی این مدل حال مدل تولید کننده را آموزش بدهیم آنگاه هر آنچه که مدل تولید میکند را به عنوان یک تصویر غیر واقعی طبقه بندی خواهد کرد و هیچ بهبودی در روند فرایند تولید اتفاق نمی افتد. و همان طور که مشخص هست نمی توان اول مدل تولید کننده را بسازیم بعد تمیز دهنده زیرا برای تولید ما نیاز به یک کارشناس داریم که تعیین کند که ایا تصاویر تولیدی واقعی هستند. [۵]

۱-۲ اطلاعات کلی

هر کدام از مدل ها ساختار خاص خودشان را دارند، در واقع ما نمی توانیم یک مدل را آموزش و برای هر دو بخش استفاده کنیم پس مهم هست تا ساختار هر کدام را رعایت کنیم. همچنین از نظر ابرپارامتر ها نیز بهتر هست که تفاوتی در نظر بگیریم برای هر کدام، به عنوان مثال برای مرتبه یادگیری [۴] می توانیم هر دو را با یک مقدار پیش ببریم ولی به دلیل این که مدل ها باید همزمان پیشرفت داده شوند پس بهتر است تا تمیز دهنده یک مقدار کوچکتری داشته باشد تا روند کند تری را طی کند زیرا تولید کننده یک فرایند طولانی تری برای هر مرحله طی می کند پس با این کار باعث می شویم تا هر دو مدل یک سرعت را پیش ببرند. حال این روند برای مقادیر دیگر هم قابل گسترش است مانند عمق مدل ها توابع زیان.
جدا از این موارد برای آموزش دادن مدل ها نیاز به داده های قابل اعتمادی هستیم تا با توجه به آن داده ها و

[۴] مرتبه یادگیری Learning Rate

داده هایی که مدل تولید کننده تولید می کند بتوانیم مدل تمیز دهنده را آموزش بدهیم. در این پروژه از دیتاست هدا [۷] که شامل ۶۰۰۰۰ تصویر برای آموزش و ۲۰۰۰۰ تصویر برای تست کردن است استفاده می کنیم. در هر مرحله از آموزش مدل تمیز دهنده ما تصاویری از این دیتاست و تصاویری را که مدل تولید کننده، تولید کرده را به آن میدهیم تا با توجه به این دو نمونه که تصویر اول تصویر واقعی و تصویر دوم که ساختگی است یادبگیرد که تصاویر غیر واقعی را شناسایی کند.

برای ارزیابی روش های بسیاری موجود است که یکی از آن روش ها مشاهده است، در واقع در این روش بایستی تصاویر متعددی را تولید کرد و سپس با توجه به تجربه آن تصاویر را بررسی کرد. برای این کار در انتها پروژه از مقادیر متفاوت به اندازه دلخواه تصویر تولید کردیم و آن ها را به صورت فایل متحرک در آورده ایم تا مشاهده راحت تر صورت بگیرد.

یک روش دیگری که برای ارزیابی انجام دادیم، روش طبقه بندی بود که در این روش جدا از مدل های قبلی که برای تولید کننده و تمیز دهنده داشتیم یک مدل به عنوان طبقه بندی کننده آموزش دادیم تا دقت آن در تصاویر تولید شده بررسی شود. در این مرحله ابتدا مدلی ساده با استفاده از دیتاست خود می سازیم، سپس مقدار زیادی تصویر را با مدل مولد تخاصمی خود تولید می کنیم و چون مدل ما شرطی بود همزمان برچسب هر تصویر را نیز ذخیره می کنیم. حالا با تصاویر جدید و برچسب هایشان مدل را ارزیابی می کنیم و ارزیابی به دست آمده تا مقدار خوبی قابل اطمینان است [۸]

.

۱-۳ چالش‌ها

در این پروژه با چالش های بسیاری همراه بودیم که به اختصار به آن ها اشاره می کنیم :

۱. مشکل دودویی بودن تصاویر و عدم نیاز اعمال نرمال سازی روی آن ها ابتدایی ترین مشکل ما بود و با مشاهده و بررسی دقیق داده های دیتاست متوجه اشکال شدم و بدون استفاده از نرمال سازی داده ها را ذخیره کردم.

۲. یکی از مشکلات ابعاد تصاویر بود که در واقع این یک مزیت ممکنه به حساب بیاید که مضربی از دو باشد اما به دلیل منابع محدود و زمان محدود ما قادر به ساخت یک مدل با عمق بسیار زیاد نبودیم

چون این پروژه برای اجرا نیاز به یک واحد پردازشی گرافیکی Gpu دارد و به دلیل نبود آن مجبور به اجرای آن در Colab Google شدم که این سرویس فقط برای مدتی محدود پردازشگر گرافیکی خود را در اختیار کاربر های رایگان خود می کند.

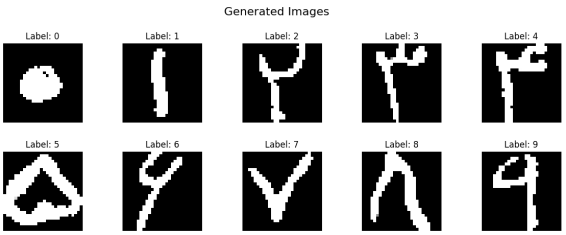
۳. مشکل بعدی نیز مربوط به پردازشگر است زیرا فقط برای مدتی محدودی پلتفرم کولب اجازه دسترسی به پردازشگر گرافیکی را فراهم کرده است. به همین خاطر مجبور به تغییر اکانت در حین فرایند آموزش می شویم برای همین می بایست هر سری تا یک مرحله خاصی را آموزش و آن را ذخیره کنیم سپس با یک حساب کاربری دیگر آن را پیش ببرم.

۴. یکی از مهمترین چالش ها قسمت آموزش دادن این پروژه بود زیرا برعکس مابقی مدل های یادگیری ماشینی اصلی ترین کار در روش آموزش دادن این مدل ها است و بایستی یک همکاری کاملا مناسبی برای هر دو مدل تولید کننده و تمیز دهنده ایجاد میکردیم.

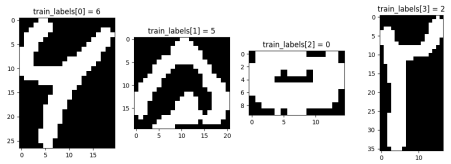
۵. در آخر هم باز به دلیل دشوار بودن مراحل آموزش، ارزشیابی این مدل ها نیز کاملا متفاوت هستند با ارزشیابی که ما روی مدل های معمولی یادگیری ماشین انجام میدادیم.

در بخش دیگری تصاویری که تولید شده با استفاده از مدل را به نحوه های مختلفی به نمایش گذاشته ام در ابتدا قادر هستیم تا مقدار دلخواهی از این تصاویر را فقط در یک طبقه خاص تولید کنیم و سپس آن ها را به صورت یک فایل محرک به نمایش بگذاریم. یا این که یک نمونه از هر کدام از این تصاویر را تولید و نمایش بگذاریم. در (تصویر ۱-۱ (آ)) یکی از این تصاویر را مشاهده می فرمایید. تمام این تصاویر تصاویری هستند که با استفاده از مدل ساخته شده تولید شده اند. اما (تصویر ۱-۱ (ب))تصاویری هستند که در دیتاست هدا قرار دارند. همان طور که مشاهده می کنید طبق روش اول ارزیابی تصاویر به وجود آمده بسیار دقت بالایی دارند.

به دلیل حجم بالایی از محاسبات حتی با توجه به وجود پردازشگر گرافیکی هر بار اجرای این مدل با مقادیر مورد نظر به یک زمان حدودی بین چهارده تا هجده ساعتی نیاز داریم پس بسیار کار دشوار و غیر حرفه ای است که هر سری این مدل را از ابتدا وزن دهی کنیم. به همین دلیل بعد از آموزش دادن و تولید یک مدل معقول سعی شده است تا تمام وزن های دو مدل تمیز دهنده و تولید کننده ذخیره شود تا هرگاه لازم به ساخت عددی دست نویس بودیم فقط مدل را با وزن های از پیش تعیین شده اجرا کرده و تصاویر را در کمتر از یک دقیقه تولید کنیم.



(آ) تصاویر اعداد تولید شده توسط مدل



(ب) تصاویر موجود در دیتاست

شکل ۱-۱: تصاویر تولید شده و تصاویر واقعی

ـ کد پروژه با استفاده از کتابخانه متن باز Tensorflow که یک کتابخانه برای یادگیری ماشین و هوش مصنوعی است پیاده سازی شده است که عملکرد بسیار خوبی در آموزش دادن مدل های عمیق دارد و بیشتر نیز برای همین منظور استفاده می شود. به دلیل این که ما هر ۳ مدل خود را با استفاده از مدل های عمیق آموزش داده ایم پس معقول بود استفاده از این کتابخانه، البته کتابخانه های دیگری نیز مانند pytorch برای این نوع پیاده سازی ها موجود است. در بخش تحلیل نتایج، نتایج به دست آمده را تحلیل و بررسی خواهیم کرد. به عنوان مثال، کیفیت دستنویس‌های تولیدی توسط مدل GAN و مقایسه آن‌ها با دستنویس‌های واقعی را که قبل تر به اختصار به آن اشاره شد بررسی خواهیم کرد. همچنین، به عواملی مانند تأثیر پارامترهای مختلف در عملکرد مدل و نحوه بهبود عملکرد آن خواهیم پرداخت.

در بخش نتیجه گیری، به نتیجه‌گیری از پژوهش پرداخته و خلاصه‌ای از نتایج به دست آمده ارائه می‌دهیم. همچنین، پیشنهاداتی برای پژوهش‌های آتی و بهبود روش‌های استفاده شده را مطرح خواهیم کرد. این پیشنهادات می‌توانند موضوع توسعه و بهبود مدل GAN برای تولید دستنویس‌های فارسی را پوشش دهند.

فصل ۲

مبانی شبکه های مولد تخاصمی

۲-۱ مدل های مولد مقابله ای GANs

مدل های مولد مقابله ای دسته ای از تکنیک های یادگیری ماشینی [۱] است که از دو مدل تشکیل شده است: مدل تولید کننده [۲] که داده های غیر واقعی را تولید می کند و مدل تمیز دهنده که تصمیم میگیرد که آیا مقدار ورودی یک مقدار واقعی است یا خیر.

کلمه تولید کننده در واقع نشان دهنده هدف کلی این مدل ها است: تولید داده جدید. دیتایی که مدل آموزش می دهد وابسته به نوع آن دیتا ورودی دارد به عنوان مثال در این پروژه داده های ما تصاویر اعداد فارسی دست نویس است به همین دلیل مدل یاد میگیرد تا تصاویر اعداد دست نویس را تولید کند در حالی که اگر دیتاست ما متن بود دیگر مدل قادر به تولید تصویر نبوده و یک مدل تولید کننده متن را خواهیم داشت.

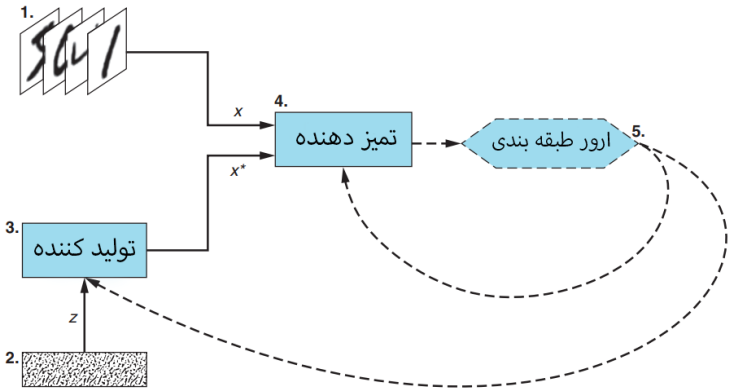
کلمه تخاصمی [۳] در واقع نشان دهنده رقابت بین دو مدل تمیز دهنده و تولید کننده رو نشان می دهد. تولید کننده سعی دارد تا تصاویر عددی را تولید کند که مصنوعی بودنش غیر قابل فهم باشد درسوی دیگر تمیز دهنده تمام تلاشش رو می کند تا تصاویر غیر واقعی تولید کننده را شناسایی کند. این روند مانند یک بازی ادامه دارد و هر دو مدل با این فرایند خودشان را بهبود می دهند.

---

[۱] یادگیری ماشینی Machine Learning

[۲] تولید کننده Generator

[۳] تخاصمی discriminator



(آ) روند آموزش مدل های مولد مقابله ای

و در آخر کلمه شبکه [۴] هم به دلیل این است که این مدل ها بر گرفته از تکنیک های یادگیری ماشین هستند و بایستی برای ساختن مدل ها از شبکه های عصبی استفاده کنیم.

۲-۲ مدل های GANs چگونه کار میکنند

هرزمان که تمیز دهنده یک تصویر را غیر واقعی شناسایی کند، تولید کننده پی میبرد که یک بخشی را به اشتباه تولید کرده و از این بازخورد استفاده می کند برای بهبود و زمانی که تمیز دهنده آن تصویر را واقعی شناسایی کند پس متوجه میشود که بخشی را به درستی پیش برده است و سعی می کند تا در آن بخش بهتر بشود. همزمان تمیز دهنده هم مانند یک طبقه بندی کننده خودش را بهبود می دهد تا مانند یک متخصص اعداد تصمیم بگیرد.

۲-۳ نحوه کارکرد GANs در عمل

حال به برسی دیاگرام روند آموزش مدل می پردازیم (تصویر ۲-۱ (ب))

۱. دیتاست آموزشی : نیاز به یک دیتاست کاملا واضح و بدون خطا داریم که بتوان با استفاده از آن مدل هایی با دقت بالا را آموزش داد، که دراین پروژه بایستی از دیتاست اعداد فارسی دست نویس استفاده کرد.

[۴] شبکه Network

۲. بردار نویز تصادفی : یک بردار عددی که به عنوان ورودی به مدل تولید کننده داده می شود تا مدل تصاویر تکراری را تولید نکند و هر بردار نویز مختص فقط یک تصویر است. در این پروژه علاوه بر بردار نویز بایستی یک بردار ۱۰ تایی از اعداد باینری را نیز که نشان دهنده اعداد ۰ تا ۹ هستند به بردار نویز اضافه کنیم، زیرا پروژه مربوط به مدل های مولد مقابله ای شرطی است و همان طور که پیش تر اشاره شد این مدل ها توانایی شرط گذاشتن برای خروجی را دارا هستند پس با این کار در هر سری از تولید اگر یکی از خانه های بردار ۱۰ تایی مقدار ۱ را بگیرد تولید کننده اندیس آن خانه را به عنوان عدد مورد نظر در نظر میگیرد به عنوان مثال اگر یک بردار با مقدار [۰،۰،۰،۱،۰،۰،۰،۰،۰،۰] را در نظر بگیریم خروجی مدل عدد ۳ است (ورودی z برای تولید کننده).

۳. مدل تولید کننده Generator : مقدار ورودی Z را میگیرد و خروجی x^* که یک تصویر غیر واقعی است را تولید می کند. هدفش فریب دادن تمیز دهنده است تا متوجه ساختگی بودن تصویر نشود.

۴. مدل تمیز دهنده Discriminator : مقادیر x که از دیتاست اصلی است و یک مقدار واقعی است و x^* که مقداری ساختگی از مدل تولید کننده است را دریافت می کند و در آخر احتمال ساختگی بودن تصویر را به عنوان مقدار خروجی بیان می کند

۵. در هر مرحله در آخر اجرا مقادیر وزن های هر مدل وابسته به موقعیت بهبود پیدا می کنند

- برای مدل تمیز دهنده وزن ها به نحوی بهبود پیدا می کنند که دقت مدل به بیشترین مقدار برسه به همین دلیل باید loss function حداقل بشود.
- برای مدل تولید کننده وزن ها به نحوی تغییر می کنند تا اشتباه مدل تمیز دهنده به حداکثر برسه به همین دلیل loss function حداکثر میشود.

۲-۴ مراحل آموزش GANs ها

در هر مرحله از آموزش:

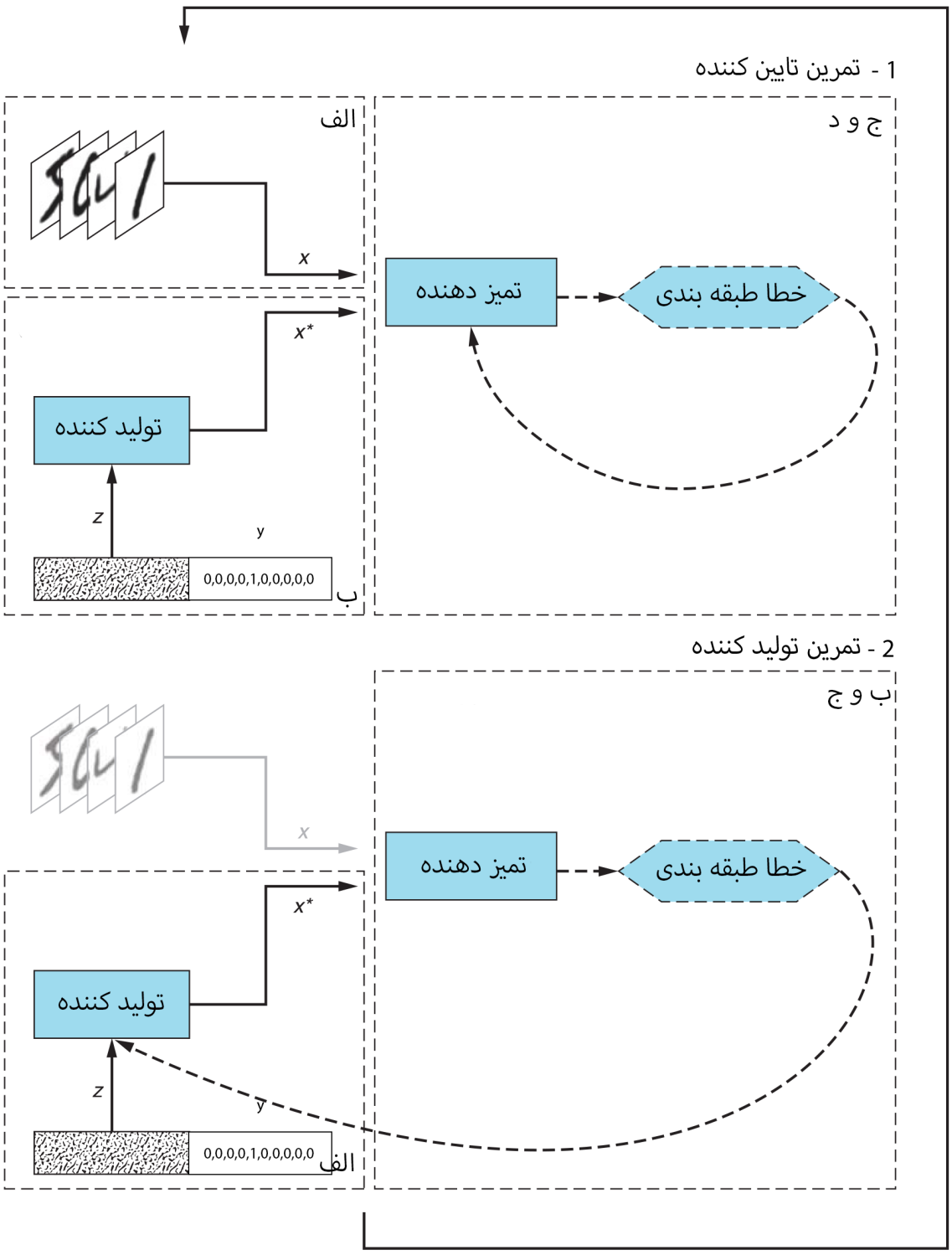
۱. آموزش Discriminator :

- الف) یک نمونه از داده های دیتاست را به عنوان x در نظر میگیریم

- ب) یک بردار نویز را به صورت تصادفی در نظر گرفته و یک عدد از ۰ تا ۹ را به صورت تصادفی انتخاب کرده و به حالت کد شده تبدیل می کنیم تا یک بردار ۱۰ تایی شود و هر دو را با همدیگر یکی می کنیم و به عنوان ورودی z به generator می دهیم تا یک تصویر عدد تولید کند و x^* را بر گرداند.
- ج) دو x ، x^* را به تمیز دهنده می دهیم.
- د) خطا را محاسبه کرده و backward Propagation می کنیم تا مقدار های جدید وزن ها را تعیین کنیم تا خطا به حداقل برسد.

۲. آموزش Generator :

- الف) یک بردار نویز را به صورت تصادفی در نظر گرفته و یک عدد از ۰ تا ۹ را به صورت تصادفی انتخاب کرده و به حالت کد شده تبدیل می کنیم تا یک بردار ۱۰ تایی شود و هر دو را با همدیگر یکی می کنیم و به عنوان ورودی z به generator می دهیم.
- ب) x^* که خروجی مرحله اول است و تصویر ساختگی است را به تمیز دهنده می دهیم تا تعیین کند که آیا تصویر واقعی است یا خیر
- ج) خطا را محاسبه و سعی می شود تا به نحوی وزن ها تغییر کند که خطا discriminator به بیشترین حد برسد.



(ب) روند آموزش مدل های مولد مقابله ای

فصل ۳

ساختار مدل ها

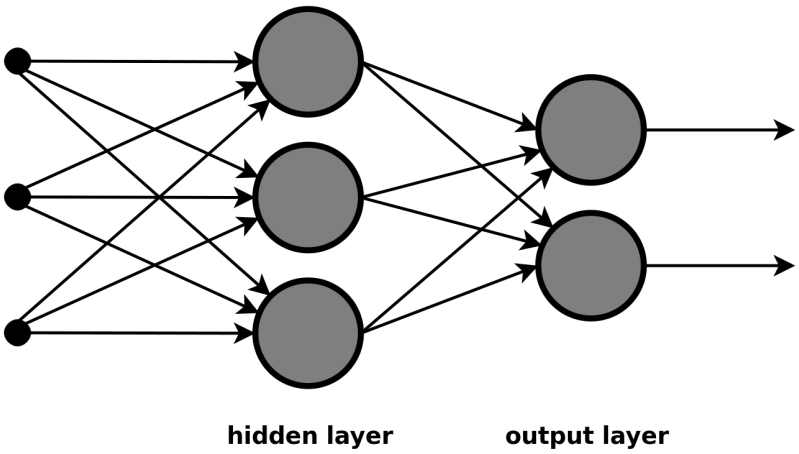
۳-۱ مدل های مولد مقابله ای GANs

الگوریتم‌های یادگیری عمیق از شبکه‌های عصبی استفاده می‌کنند تا بر اساس تجزیه و تحلیل داده‌ها، برای گرفتن تصمیم خاصی، رفتار انسان را تقلید کنند. طرح این ساختار لایه‌ای، برگرفته از ساختار مغز انسان است. همانطور که مغز انسان به شناسایی الگوهای مختلف داده‌ها و دسته‌بندی انواع اطلاعات می‌پردازد، می‌توان شبکه‌های عصبی را به شیوه‌ای مشابه با رفتار مغز انسان آموزش داد تا به تشخیص الگوها بپردازند و دسته‌بندی داده‌ها را انجام دهند.(تصویر ۴-۱ (آ))

۳-۲ فرآیند آموزش مدل های GANs

در این پروژه از شبکه های عصبی استفاده شده است که عمیق تر هستند و این به دلیل نوع و تعداد داده های استفاده شده در پروژه است. زیرا داده های تصویری حجم محاسبات بیشتری را نیاز دارند نسبت به داده های متنی. به عنوان مثال در این پروژه از تصویر ۲۸ پیکسلی استفاده شده یعنی ورودی برابر ۷۸۴ خانه که مقادیر ۰ یا ۱ را میگیرند. ولی برای داده های متنی شبکه های عصبی با عمق کمتر نیز پاسخ خوبی با دقت بالایی را می توانند بدهند.

حال بعد از ساخت مدل باید آن را آموزش بدهیم. در مرحله اول xi ها که ورودی های ما هستند به گره



(آ) تصویر ۴ مرحله از آموزش که به ترتیب بعد از آموزش ۳۰،۶۰،۹۰،۱۵۰ مرحله ثبت شده.

ها داده می شوند و هر گره محاسبه زیر را انجام می دهد[۵]:

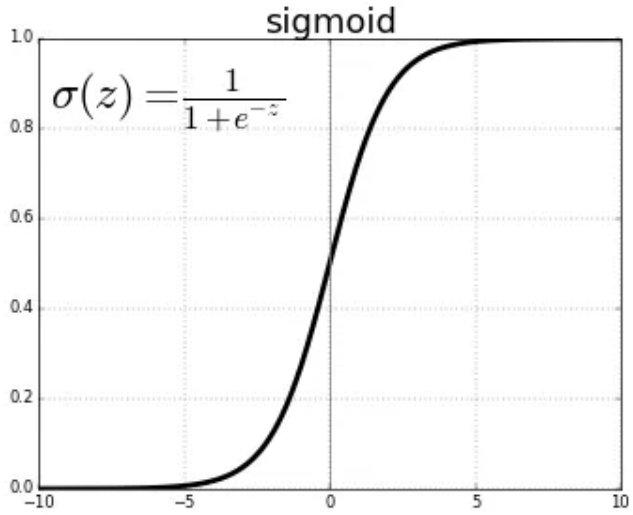
$$z = \sum_{i=۱}^{n} w_i * x_i + b \qquad (۱-۳)$$

حال اگر که عمق شبکه بیشتر از ۱ باشد بایستی که روی تک تک لایه ها این اعمال را انجام بدهیم سپس پس از محاسبه احتمالاتی بالا باید مقدار را به مقیاس ثابتی تبدیل کنیم چون مقدار z هر عددی رو میتواند شامل بشود پس برای جلوگیری از بزرگی یا کوچکی بیش از اندازه مقدار z بایستی آن را به مقیاس کوچکتر و محدودتری تبدیل کنیم. برای این کار از تابع های فعال سازی[۱] استفاده می کنیم.

۱. تابع sigmoid : این تابع برای مدل هایی که دودویی هست خروجیشان استفاده می شود و ما از این مدل برای لایه آخر یعنی خروجی مدل تمیز دهنده مان استقاده کرده ایم. که مقدار z را به ۰ تا ۱ مقیاس دهی می کند. (تصویر ۳-۱ (ب)) [۵]

۲. تابع ReLU : از این تابع می توان برای لایه های میانی یا Leyer Hidden ها استفاده کرد زیرا برعکس تابع سیگموید که در مقادیر بزرگ و کوچک تقریبا در شیب ثابت می افتد و مقدار ثابتی لحاظ می شود این تابع به صورت خطی مقدارش افزایش پیدا می کند. از این تابع در پروژه برای لایه های میانی مدل طبقه بندی که برای ارزیابی طراحی شده استفاده کرده ایم و دلیلی که از آن

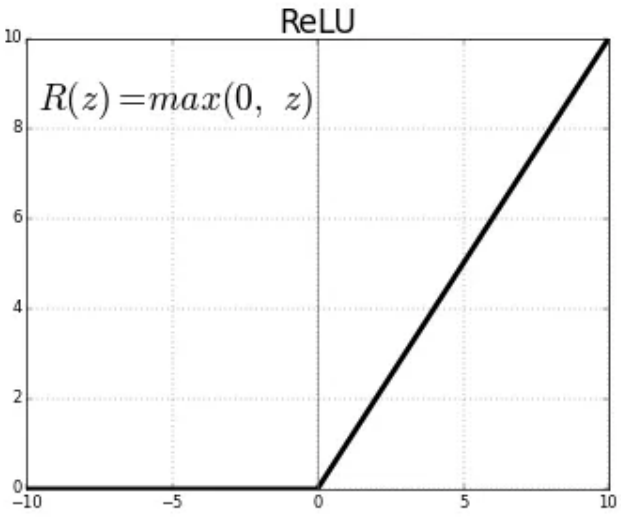
[۱] تابع های فعال سازی Activation Function است



(ب) منحنی و تابع سیگموید

برای Generator یا Discriminator مان استفاده نکردیم این است که همان طور که در نمودار (تصویر ۳-۱ (ج))مشاهده می کنید تابع در مقادیر منفی مقدار ۰ را در نظر میگیرد که این کار برای Generator به مشکل های متعددی بر میخوریم از جمله collapse Mode و ReLU Dying یا مرگ رلو.

- مشکل dying ReLU وقتی رخ می‌دهد که نورون‌های ReLU غیرفعال شده و دیگر گرادیانی را خروجی نمی‌دهند. این مشکل هنگامی اتفاق می‌افتد که ورودی به یک نورون ReLU همواره منفی باشد و در نتیجه نورون مرده‌ای به وجود می‌آید که به فرآیند یادگیری کمکی نمی‌کند. ReLU Leaky این مشکل را با اجازه دادن به یک گرادیان کوچک برای ورودی‌های منفی برطرف می‌کند و از کاملاً غیرفعال شدن جلوگیری می‌کند.[۸]
- جریان گرادیان بهتر: در GANs، شبکه مولد سعی می‌کند توزیع واقعی داده‌ها را با تولید نمونه‌های واقع‌گرایانه تقریب بزند. استفاده از Leaky ReLU می‌تواند جریان گرادیان بهتری را در طول عملیات پس‌انتشار به شبکه فراهم کند. گرادیان غیرصفر برای ورودی‌های منفی در Leaky، ReLU به گرادیان‌ها اجازه می‌دهد حتی زمانی که فعال‌سازی منفی است، برای به‌روزرسانی‌های مناسب در وزن‌های شبکه ارائه شود. [۸]
- کنترل اشباع: نورون‌های ReLU در برخی موارد هنگامی که ورودی بزرگ است، اشباع می‌شوند



(ج) منحنی و تابع رلو

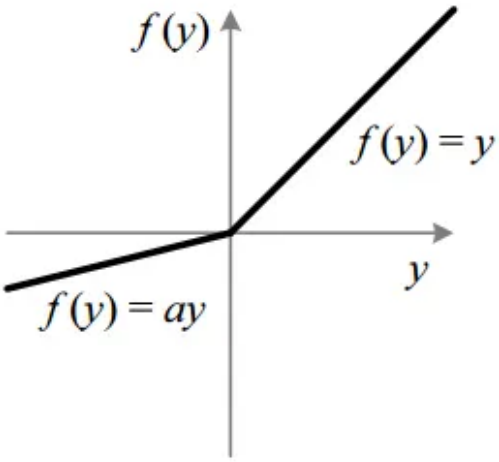
و گرادیان آنها صفر می‌شود. این موضوع می‌تواند فرآیند یادگیری را مختل کند زیرا شبکه نمی‌تواند وزن‌ها را به درستی به‌روزرسانی کند.Leaky ReLU با حفظ گرادیان غیرصفر برای ورودی‌های مثبت و منفی، اشباع را پیشگیری می‌کند و به همگرایی بهتر کمک می‌کند.[۵]

۳. تابع LeakyReLU : به دلایلی که در قسمت تابع ReLU ذکر شد برای مدل های GANs بهتر هست تا از این تابع استفاده شود. البته این تنها تابع فعالسازی مناسب نیست به عنوان مثال Tanh یا سیگموید های پیشرفته نیز قابل استفاده برای هر دو تمیز دهنده و تولید کننده هستند.(تصویر ۳-۱ (د))[۵]

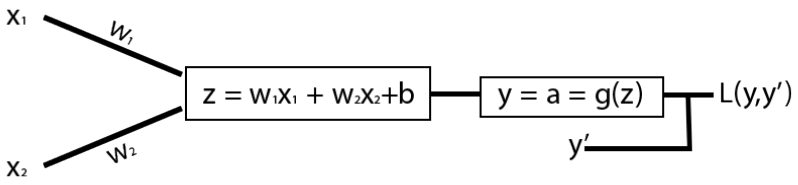
بعد از مشخص کردن تابع فعالسازی و محاسبه a که برابر با اعمال تابع فعالسازی روی z است. نیاز داریم ببینیم که آیا مدل دارد به درستی آموزش داده می شود یا آیا بهبود عملکرد در آموزش مشهود است یا خیر. به همین دلیل از تابع زیان [۲] استفاده می کنیم. (تصویر ۳-۱ (ه))

در این پروژه از Binary Cross Entropy به عنوان تابع زیان استفاده شده است یا همان LogLoss :

[۲] تابع زیان Loss Function است



(د) منحنی و تابع لیکی رلو



(ه) روند کلی یک مرحله از پردازش پیشرونده شبکه های عصبی

$$L = -\frac{۱}{N}\sum_{i=۱}^{N}\left[y_i\ \log(p_i)\ +\ (۱-y_i)\ \log(۱-p_i)\right] \tag{۳-۲}$$

حال که خطا را محاسبه کردیم وقت آن رسیده که بخش اصلی یعنی بروز کردن وزن ها را انجام دهیم. در این مرحله از آخر به اول بر میگردیم که اسم آن پیش انتشار Backpropagation است. در واقع از خروجی شروع می کنیم و به لایه های عقب تر بر میگردیم و با استفاده از داده های به دست آمده وزن های جدید را وضع می کنیم.[۸]

$$dz = a_i - y_i \tag{۳-۳}$$

$$dw_i = dw_i + (x_i[i] * dz_i) \tag{۳-۴}$$

$$db = db + dz_i \tag{۳-۵}$$

حال با محاسبه کردن مشتق تابع زیان نسبت به وزن و بایس ها باید مقادیرشان را بهنگام کنیم :

$$w_i = w_i - alpha * dw_i \tag{۳-۶}$$

$$b = b - alpha * db \tag{۳-۷}$$

این مراحل در هر مرحله تکرار می شود و باعث بهبود وزن ها می شود.

۳-۳ ابر پارامتر Hyper Parameter

یکی از مهمترین بخش ها در طراحی و آموزش مدل ها تعیین ابر پارامتر ها است. این مقادیر به این دلیل نام گذاری شده اند که هیچ مقدار دقیقی برای آن ها تعیین نشده و تقریبا به صورت تجربی به دست می آید ولی باز هم یک سری قوانینی در تایین آن ها صادق هست و عملکرد آن ها را بهبود می دهد.[۳]

۱. عمق شبکه : یکی از پارامتر های اصلی هر مدل عمق آن است. برخی به این باور هستند که هرچه عمق شبکه بیشتر بشود در آن سو دقت مدل نیز بیشتر میشود اما متاسفانه این یک اشتباه بزرگ است زیرا نه تنها با زیاد شدن عمق حجم محاسبات نیز افزایش چشم گیری پیدا میکند (که دقیقا در این پروژه با آن روبه رو شدم، به دلیل بیشتر شدن عمق به مقدار ۲ لایه تقریبا مجبور به کوچک تر کردن مقدار عکس های ورودی شدم تا لایه های کمتری در مدل تولید کننده داشته باشم. زیرا در مدل تولید کننده مقیاس تصاویر رابطه مستقیم با عمق شبکه دارند)

۲. تعداد فیلتر های هر لایه : این نیز یک پارامتر بسیار کلیدی است اما این پارامتر و ابعاد هر فیلتر کاملا تجربی است و در این پروژه پس از آموزش های متعدد با مقادیر متفاوت و مشاهده نتایج به این مقادیر رسید ایم همچنین سعی شده است تا از مدل های از پیش آموزش داده شده نیز استفاده شود تا مطلوب ترین مقدار دهی صورت بگیرد.

۳. نرخ یادگیری Learning Rate : ابر پارامتری که تقریبا مهمترین پارامتر در این نوع مدل های است. زیرا GANs ها یک صورت خاصی از مثال یادگیری ماشینی هستند که دو مدل تمیز دهنده و تولید کننده بایستی در یک راستا با سرعت تقریبا یکسانی پیش بروند.

- اگر که مدل تمیز دهنده نرخ یادگیری مساوی نسبت به تولید کننده داشته باشد به دلیل سرعت آموزش داده شدن تمیز دهنده ها بسیار بیشتر از تولید کننده است. زیرا تمیز دهنده فقط با دیدن تصاویر باید یاد بگیرد که کدام ورودی غیر واقعی و ساختگی است و کدام واقعی در صورتی که تولید کننده با دریافت یک بردار کاملا تصادفی باید یادبگیرد که از آن یک عدد تولید کند به صورت تصویر ۲۸ پیکسل در ۲۸ پیکسل حال در این پروژه این مسئولیت سخت تر نیز شده است، زیرا خود مدل نمی تواند تصمیم بگیرد که چه عددی را تولید کند و آن را ما به عنوان یک ورودی کد شده به آن می دهیم پس برای این که این دو مدل با هم آموزش داده بشوند نباید

سرعت تمیز دهنده که سریع تر آموزش داده می شود بیشتر باشد زیرا در این حالت پس از چند مرحله اول تمیز دهنده کاملا یاد میگیرد و از آن به بعد تمام تصاویر که ساخته شده اند را رد می کند.

- به هیچ وجه نباید سرعت تمیز دهنده بیشتر باشد به دلایل بالا اما سرعت تولید کننده هم نباید بیشتر باشد زیرا باعث overfitting می شویم و به اشتباه تولید کننده یاد میگیرد زیرا تمیز دهنده به اندازه کافی بهبود پیدا نکرده و مقادری اشتباه را به درستی پاسخ می دهد و تولید کننده اشتباه متوجه می شود.

مقادیر نرخ یادگیری بهتر است ثابت نباشند و در هر ۳۰ یا ۵۰ مرحله تغییر کنند زیرا:

- یکی از چالش‌های اصلی در آموزش شبکه های مولد مقابله ای ،حفظ تعادل بین تولید کننده و تمیز دهنده است. معمولاً در ابتدای آموزش، تمیز دهنده اطلاعات بیشتری دارد و می‌تواند به تولید کننده غلبه کند. با تغییر نرخ یادگیری در طول زمان، می‌توان این تعادل را بهبود بخشید و به تولید کننده کمک کرد تا بیشترین بهره را ببرد. برای مثال، با کاهش نرخ یادگیری تمیز دهنده و یا افزایش نرخ یادگیری تولید کننده، ممکن است تعادل بهتری بین این دو شبکه ایجاد شود.
- در طول آموزش، توزیع داده‌ها ممکن است تغییر کند یا توزیع نمونه‌ها تنوع بیشتری به خود بگیرد. با تغییر نرخ یادگیری، می‌توان شبکه‌ها را قادر به تطبیق با تغییرات داده‌ها کرد و از دقت و عملکرد آن‌ها بهره برد.

۳-۴ مدل تمیز دهنده Discriminator

به دلیل بالا بودن حجم داده ها به دلیل تصویر بودن داده ها از لایه های پیچشی convolutional استفاده می کنیم و همان طور که پیش تر اشاره شد تمامی مقادیر فیلتر ها و تعداد آن ها به صورت تجربی و استفاده از مدل های از پیش تعریف شده بوده است. همین طور عمق شبکه نیز به دلایل عدم وجود زمان و هزینه کافی تا حد زیادی بهینه شده ولی سعی شده است با استفاده از پیشپردازش های انجام شده به دقت شبکه اسیبی نرسد.

باتوجه به بحث انجام شده دربخش ۴ تابع فعال سازه استفاده شده در این پروژه LeakyReLU است به دلایل

ذکر شده تا از مشکلات پردازشی تا حدامکان جلوگیری شود. به علاوه سعی شده تا با استفاده از Dropout از بیشبرازش overfitting [۳] نیز جلوگیری شود. در Dropout با حذف تعدادی از گره ها به صورت کاملا تصادفی سعی میکنیم که مدل بیش برازش انجام ندهد.

```
InputLayer((28, 28, 11))
Conv2D(64, (3, 3), strides=(2, 2), padding=”same”)
LeakyReLU(alpha=0.2)
Dropout(0.4)
Conv2D(128, (3, 3), strides=(2, 2), padding=”same”)
LeakyReLU(alpha=0.2)
Dropout(0.4)
GlobalMaxPooling2D( )
Dense(1)
```

مقدار alpha که در واقع تفاوت LeakyReLU با ReLU است آن ضریبی است که بر روی بخش منفی اعمال می شود که با این مقدار شیب آن قسمت را کنترل می کنیم. این مقدار نیز یک ابر پارامتر است که با توجه به تجربه و مدل های از پیش تعریف شده مقدار آن را ۲.۰ در نظر گرفتیم.

۳-۵ مدل تولید کننده Generator

```
InputLayer((42,))
Dense(7 * 7 * 42),
LeakyReLU(alpha=0.2)
Reshape((7, 7, 42))
Conv2DTranspose(128, (4, 4), strides=(2, 2), padding=”same”)
```

[۳] تابع زیان overfitting است

LeakyReLU(alpha=0.2)

BatchNormalization()

Conv2DTranspose(128, (4, 4), strides=(2, 2), padding=”same”)

LeakyReLU(alpha=0.2)

BatchNormalization()

Conv2D(1, (7, 7), padding=”same”, activation=”sigmoid”)

به دلیل این که مقدار بردار نویز را ۳۲ در نظر گرفتیم که این باز نیز یک ابر پارامتر است و با افزودن بردار ۱۰ تایی شرط در واقع ورودی این مدل ۴۲ عنصر دارد. [۶]

در بخش تولید کننده ما باز ملزم به استفاده از لایه های پیچشی هستیم ولی با یک تفاوت کوچک. برای ساخت تصاویر با جزئیات بیشتر با لایه های معمولی نیاز به زمان و پردازش بسیار سنگین و طولانی داریم اما می توان در ابتدا تصاویر کوچکی را تولید کرد و سپس در هر مرحله این جزئیات را بیشتر و بیشتر کنیم در این حالت به عنوان مثال به جای ۳ مرحله پردازش ۷۸۴ تایی در ابتدا فقط ۴۹ پارامتر داریم و در مرحله بعد ۱۹۶ و هر مرحله دوبرابر میشود به همین دلیل بسیار پردازش سریع تر و کم هزینه تری را خواهیم داشت.[۵]

برای این کار بایستی از روش upsampling استفاده کنیم. روش های گوناگونی برای این بخش هست مانند نزدیک ترین همسایه ولی در این روش ما فقط اندازه عکس رو داریم بزرگ میکنیم بدون توجه به کیفیت آن در صورتی که کیفیت تصاویر نیز برای ما ارزش بسیار زیادی دارند.

ولی روشی با نام convolution Transpose موجود است که تقریبا کیفیت مناسبی را برای تصاویر بزرگ شده ایجاد می کند. البته مانند هر روشی مشکلاتی نیز دارد این روش مانند مشکل شطرنجی شدن Checkerboard Pattern که تصاویر در برخی از نواحی به صورت شطرنجی تکرار می شوند البته این عمل بیشتر بر روی تصاویر بزرگ و رنگی اتفاق می افتد و برای تصاویر ما که به صورت سیاه و سفید است مشکلی ایجاد نمی کند. همین طور برای Activation F̈unction ها LeakyReLU به دلایل قبل در نظر گرفته شده است.[۵]

فصل ۴

ارزیابی مدل

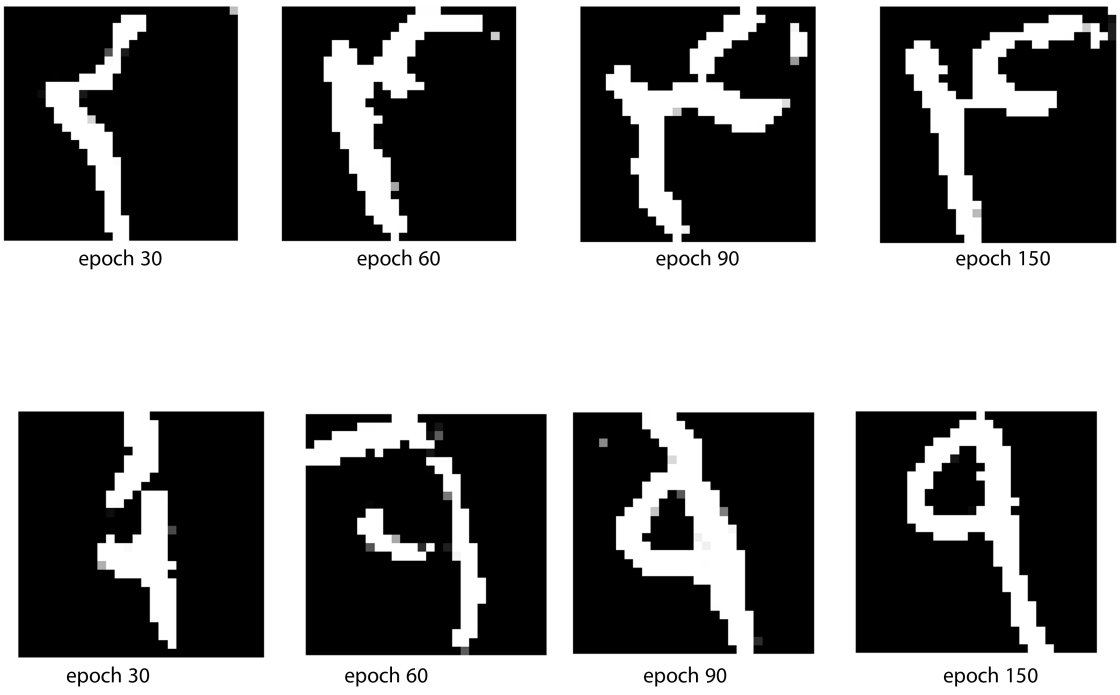
۴-۱ ارزیابی

برای ارزیابی مدل های GAN بر خلاف مدل های معمولی راه بسیار سختی پیش رو داریم. زیرا هیچ مقیاس عددی تقریبا به صورت مستقیم درستی و درصد درستی مدل را تایین نمی کند. روش های مختلفی وجود دارد برای این کار که در این پروژه از دو روش استفاده شده است.[۲]

روش اول) مشاهده : خوشبختانه در روند تولید تصاویر انسان توانایی تشخیص صحیح بودن داده ها را دارد زیرا با آن داده ها آشنایی کامل دارد پس با تولید گونه های مختلف تصاویر و تعداد قابل توجهی از آن ها می توان به دید کلی نسبت به دیتا های تولید شده پیدا کرد به عنوان مثال در روند یادگیری مدل این پروژه به دلیل این که پروژه به صورت بخش بخش آموزش داده شده است سعی شده است تا هر ۳۰ مرحله این تفاوت ها با هم مقایسه شوند.[۸]

به همین منظور تصاویر تولید شده به سه حالت تصویر متحرک، تصاویر هر کدام از اعداد به تنهایی برای مقایسه تک به تک و تصاویر هر ۱۰ عدد در کنار هم تولید و به نمایش گذاشته شود تا دید کلی نسبت به تمام اعداد تولید شده توسط مدل را داشته باشیم.

روش دوم) استفاده از مدل ثانویه : فرایند اولیه این ایده به صورت اتفاقی به ذهنم خطور کرد و سعی کردم آن را پیاده سازی کنم. در ابتدا آن را روی مدل اصلی که آموزش داده بودم استفاده کردم بعد از این که



(آ) یک شبکه عصبی ساده از عمق ۱

یک دقت بسیار بالا رو دریافت کردم که در شکل ۵-۱ مشاهده می کنید سعی کردم هر ۳۰ مرحله آن را تست کنم تا مطمئن شوم در مراحل ابتدایی نیز یک دقت بسیار پایینی را می دهد مخصوصا برای اعداد ۴ و ۹ که از سخت ترین اعدادی بودند که آموزش کاملی داشته اند و بیشتر زمان آموزش به دلیل کامل شدن یادگیری ۴ و ۹ بوده است. در این مرحله در بخش اول یعنی بعد از ۳۰ مرحله یک دقت بسیار پایین معادل ۱۲ درصد برای عدد ۴ و ۲۷ درصد برای عدد ۹ ثبت شد. که این نشان می داد این روش تقریبا یک دید خوبی از ارزشیابی دارد به نمایش می گذارد. بعد از مقداری پژوهش در آخر این روش را برای چت باکس ai open شرح دادم و این چت باکس کاملا موافق با این روش بود و به گفته آن دیتا ساینتیست ها از این روش به عنوان روش سریع استفاده می کنند.

n	Precision	recall	f1-score	support
0	99	100	99	95
1	99	100	100	103
2	91	100	95	97
3	94	93	94	101
4	100	90	95	94
5	100	95	97	102
6	100	98	99	96
7	100	100	100	107
8	95	97	96	94
9	96	93	95	111
accuracy		97	1000	
avg macro	97	97	97	1000
avg weighted	97	97	97	1000

در این روش به صورت زیر عمل خواهیم کرد:

۱. مدل طبقه بندی کننده ای را می سازیم با استفاده از داده های دیتاست اصلی (دیتاست هدا).

۲. بعد از آموزش مدل مقادیر زیادی از تصاویر را با استفاده از GAN ساخته شده تولید و ذخیره می کنیم.

۳. با استفاده از مدل ثانویه ساخته شده تصاویر ساخته شده را ارزیابی می کنیم (در واقع از داده های تولید شده به عنوان داده های تست استفاده می کنیم).

روش های مختلف دیگری از جمله :

۱. Inception Score (IS) [۱]

۲. Frechet Inception Distance (FID) [۹]

که به پردازش بسیار طولانی و پر هزینه نیاز دارد همچنین به تعداد بالایی از تصاویر تولید شده نیاز است. و این تنها مشکلات آن ها نیست به عنوان مثال چون این مدل ها بر پایه InceptionV۲ تولید شده اند بایستی ابعاد تصاویر تولید شده حتما **۲۹۹** پیکسل در **۲۹۹** پیکسل باشد.

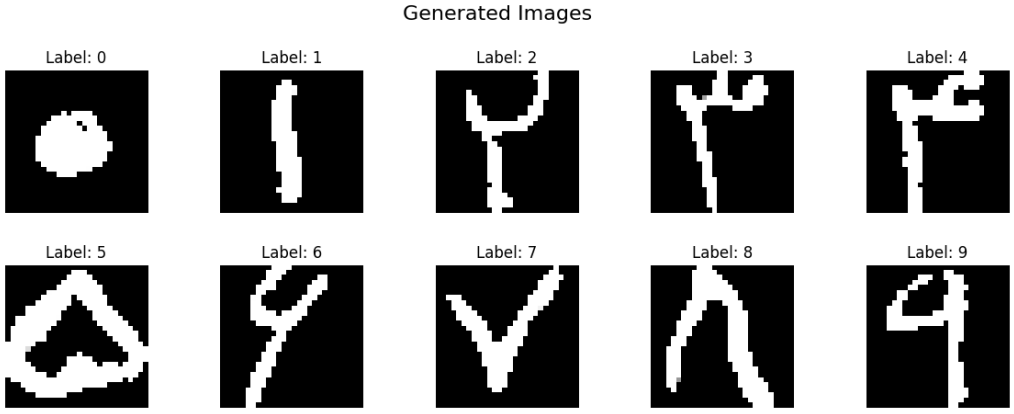
فصل ۵

نتایج

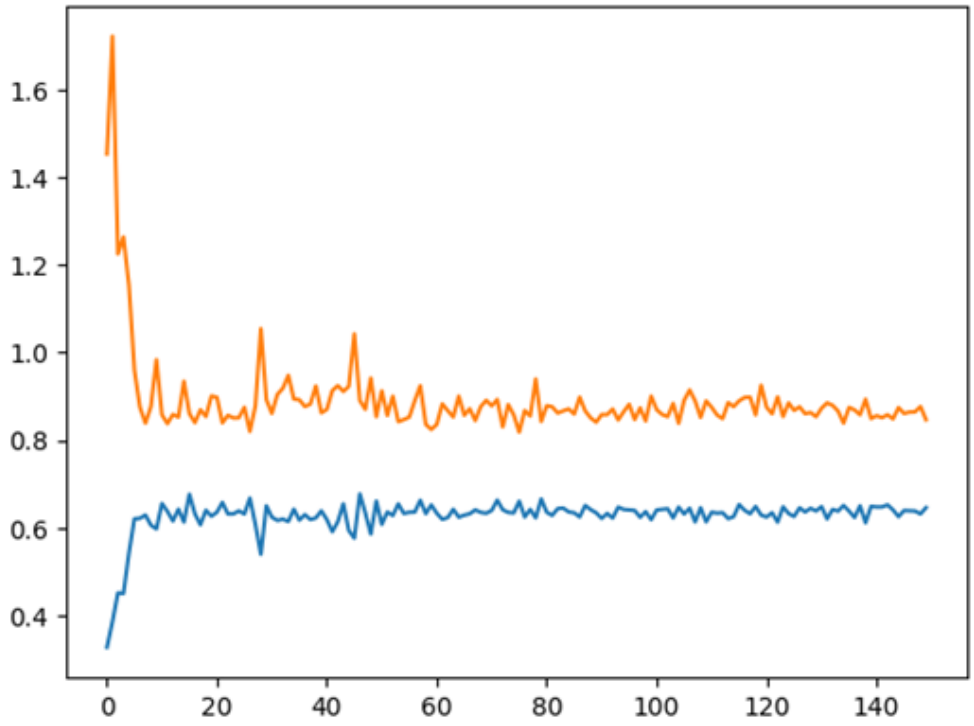
با توجه به گزارشاتی که پیشتر به آن ها اشاره کردیم نتایج بسیار خوبی در تعداد کمی از مراحل تمرین به دست آوردیم.

متاسفانه نمی توان تابع زیان را برای تخمین دقت استفاده کرد. زیرا در مدل های مولد مقابله ای هیچ ارتباط مستقیمی بین دقت یا تابع هزینه با پاسخ و صحت مسئله وجود ندارد. اما تمامی اطلاعات از تمامی مراحل و تست ها با مقادیر مختلف بردار نویز یا نرخ یادگیری ثبت شده است.

با توجه به ارزیابی انجام شده، به نتیجه قابل قبولی رسیده ایم. درصد دقت این مدل تقریبا برابر با **۹۷** درصد است و از آنجا که دقت مدل ارزیابی روی داده های تمرینی اصلی تقریبا برابر **۹۸.۵** درصد بوده است و دقت انسانی برای شناسایی اعداد دست نویس **۹۹** درصد است. که با توجه به دقت بالای مدل طبقه بندی



(آ) اعداد ۰ تا ۹ که با استفاده از مدل تمرین داده شده ساخته شده اند



(ب) یک نمونه از نمودار پیشرفت تابع هزینه.

کننده استفاده شده برای ارزیابی (به دلیل مقدار کم بایس و واریانس مدل)، تصاویر تولید شده کیفیت نسبتا مطلوبی را دارند.

زمان تقریبی اجرای پروژه با استفاده از پردازشگر 4T پلتفرم کولب به صورت تقریبی برابر ۴ ساعت برای ۳۵۰ مرحله تمرین است. و هر مرحله تقریبا ۴۰ ثانیه به طول می انجامید. البته این زمان فقط مخصوص تمرین دادن مدل نهایی است، و زمان هایی که برای پیدا کردن بهترین ابر پارامتر بوده را لحاظ نکرده ایم. به طور تقریبی حدود ۴۰ ساعت تمرین انجام شده است برروی روی پروژه.

کتاب‌نامه

[1] Barratt, S., and Sharma, R. A note on the inception score.

[2] Doe, J., and Smith, J. Evaluation of gan-based models for phishing url classifiers. *Journal of Cybersecurity 15*, 2 (2022), 67–82. Accessed on July 2, 2023.

[3] Durrani, A., Minallah, N., Aziz, N., Frnda, J., Khan, W., and Nedoma, J. Effect of hyper-parameters on the performance of convlstm based deep neural network in crop classification. *PloS one 18* (02 2023), e0275653.

[4] Keras Developers. Keras conditional gan example. https://keras.io/examples/generative/conditional_gan/, Accessed 2023.

[5] Langr, J., and Bok, V. *GANs in Action: Deep learning with Generative Adversarial Networks*. Manning, 2019.

[6] Liu, Y., Sheng, H., Wang, S., Wu, Y., and Xiong, Z. Cogan: Cooperatively trained conditional and unconditional gan for person image generation. *IET Image Processing* (06 2023), n/a–n/a.

[7] Sanyian, A. Hoda dataset (persian handwritten digit). GitHub repository, 2018. The persian version of mnist dataset.

[8] Smith, J., and Johnson, E. Analysis of generative adversarial networks. *Journal of Artificial Intelligence 10*, 3 (2021), 123–145. Accessed on July 2, 2023.

[9] Soloveitchik, M., Diskin, T., Morin, E., and Wiesel, A. Conditional frechet inception distance, 03 2021.

پیوست آ

آـ۱ کد کامل برنامه

لینک کد کامل پروژه : لینک

کد پروژه با استفاده از مدل آماده Keras با کمی تغییرات زده شده است.[۴] همچنین مدل های تولید کننده و تمیزدهنده تمرین داده شده در آخرین نسخه از پروژه را با استفاده از این لینک می توان دریافت کرد.

```
from tensorflow import keras
from tensorflow.keras import layers

from tensorflow_docs.vis import embed
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import imageio
```

Importing the Vital LIbraries

```
batch_size = 1024
num_channels = 1
num_classes = 10
image_size = 28
latent_dim = 100
```

Global Variable Defining

```
from HodaDatasetReader.HodaDatasetReader import read_hoda_dataset
train_images, train_labels = read_hoda_dataset(
    dataset_path='HodaDatasetReader/DigitDB/Train 60000.cdb',
    images_height=28,
    images_width=28,
    one_hot=False,
    reshape=True)

test_images, test_labels = read_hoda_dataset(
    dataset_path='HodaDatasetReader/DigitDB/Test 20000.cdb',
    images_height=28,
    images_width=28,
    one_hot=False,
    reshape=True)
```

Loading the Data

```
all_digits = np.concatenate([train_images, test_images])
all_labels = np.concatenate([train_labels, test_labels])

all_digits = all_digits.astype("float32")
all_digits = np.reshape(all_digits, (-1, 28, 28, 1))
all_labels = keras.utils.to_categorical(all_labels, 10)

dataset = tf.data.Dataset.from_tensor_slices((all_digits,
  all_labels))
dataset = dataset.shuffle(buffer_size=1024).batch(batch_size)

print(f"Shape of training images: {all_digits.shape}")
print(f"Shape of training labels: {all_labels.shape}")
```

Preprocecing on Data

```
generator_in_channels = latent_dim + num_classes
discriminator_in_channels = num_channels + num_classes
print(generator_in_channels, discriminator_in_channels)
```

Channel size Implimenting

```
discriminator = keras.Sequential(
    [
        keras.layers.InputLayer((28, 28,
  discriminator_in_channels)),
        layers.Conv2D(64, (3, 3), strides=(2, 2), padding="same")
  ,
        layers.LeakyReLU(alpha=0.2),
        layers.Dropout(0.4),
        layers.Conv2D(128, (3, 3), strides=(2, 2), padding="same"
  ),
        layers.LeakyReLU(alpha=0.2),
        layers.Dropout(0.4),
        layers.GlobalMaxPooling2D(),
        layers.Dense(1),
    ],
    name="discriminator",
)

generator = keras.Sequential(
    [
        keras.layers.InputLayer((generator_in_channels,)),
        layers.Dense(7 * 7 * generator_in_channels),
        layers.LeakyReLU(alpha=0.2),
        layers.Reshape((7, 7, generator_in_channels)),
        layers.Conv2DTranspose(128, (4, 4), strides=(2, 2),
  padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.BatchNormalization(),
        layers.Conv2DTranspose(128, (4, 4), strides=(2, 2),
  padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.BatchNormalization(),
        layers.Conv2D(1, (7, 7), padding="same", activation="
  sigmoid"),
    ],
    name="generator",
)
```

Build models

```
class ConditionalGAN(keras.Model):
    def __init__(self, discriminator, generator, latent_dim):
        super().__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim
        self.gen_loss_tracker = keras.metrics.Mean(name="
  generator_loss")
        self.disc_loss_tracker = keras.metrics.Mean(name="
  discriminator_loss")

    @property
    def metrics(self):
        return [self.gen_loss_tracker, self.disc_loss_tracker]

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super().compile()
        self.d_optimizer = d_optimizer
```

```
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    def train_step(self, data):
        # Unpack the data.
        real_images, one_hot_labels = data

        # Add dummy dimensions to the labels so that they can be concatenated with
        # the images. This is for the discriminator.
        image_one_hot_labels = one_hot_labels[:, :, None, None]
        image_one_hot_labels = tf.repeat(
            image_one_hot_labels, repeats=[image_size * image_size]
        )
        image_one_hot_labels = tf.reshape(
            image_one_hot_labels, (-1, image_size, image_size, num_classes)
        )

        # Sample random points in the latent space and concatenate the labels.
        # This is for the generator.
        batch_size = tf.shape(real_images)[0]
        random_latent_vectors = tf.random.normal(shape=(batch_size, self.latent_dim))
        random_vector_labels = tf.concat(
            [random_latent_vectors, one_hot_labels], axis=1
        )

        # Decode the noise (guided by labels) to fake images.
        generated_images = self.generator(random_vector_labels)

        # Combine them with real images. Note that we are concatenating the labels
        # with these images here.
        fake_image_and_labels = tf.concat([generated_images, image_one_hot_labels], -1)
        real_image_and_labels = tf.concat([real_images, image_one_hot_labels], -1)
        combined_images = tf.concat(
            [fake_image_and_labels, real_image_and_labels], axis=0
        )

        # Assemble labels discriminating real from fake images.
        labels = tf.concat(
            [tf.ones((batch_size, 1)), tf.zeros((batch_size, 1))], axis=0
        )

        # Train the discriminator.
        with tf.GradientTape() as tape:
            predictions = self.discriminator(combined_images)
            d_loss = self.loss_fn(labels, predictions)
        grads = tape.gradient(d_loss, self.discriminator.trainable_weights)
```

```
            self.d_optimizer.apply_gradients(
                zip(grads, self.discriminator.trainable_weights)
            )

            # Sample random points in the latent space.
            random_latent_vectors = tf.random.normal(shape=(
    batch_size, self.latent_dim))
            random_vector_labels = tf.concat(
                [random_latent_vectors, one_hot_labels], axis=1
            )

            # Assemble labels that say "all real images".
            misleading_labels = tf.zeros((batch_size, 1))

            # Train the generator (note that we should *not* update
    the weights
            # of the discriminator)!
            with tf.GradientTape() as tape:
                fake_images = self.generator(random_vector_labels)
                fake_image_and_labels = tf.concat([fake_images,
    image_one_hot_labels], -1)
                predictions = self.discriminator(
    fake_image_and_labels)
                g_loss = self.loss_fn(misleading_labels, predictions)
            grads = tape.gradient(g_loss, self.generator.
    trainable_weights)
            self.g_optimizer.apply_gradients(zip(grads, self.
    generator.trainable_weights))

            # Monitor loss.
            self.gen_loss_tracker.update_state(g_loss)
            self.disc_loss_tracker.update_state(d_loss)
            return {
                "g_loss": self.gen_loss_tracker.result(),
                "d_loss": self.disc_loss_tracker.result(),
            }
```

Train Section

```
cond_gan = ConditionalGAN(
    discriminator=discriminator, generator=generator, latent_dim=
    latent_dim
)
```

Instance Creation

```
cond_gan.compile(
    d_optimizer=keras.optimizers.Adam(learning_rate=3e-4),
    g_optimizer=keras.optimizers.Adam(learning_rate=3e-4),
    loss_fn=keras.losses.BinaryCrossentropy(from_logits=True),
)

history = cond_gan.fit(dataset, epochs=30)
```

Compile and Fit Section

```
num_rows = 2
```

```
num_cols = 5
fig, axs = plt.subplots(num_rows, num_cols, figsize=(12, 5))
fig.suptitle("Generated Images", fontsize=16)
fig.tight_layout(pad=2.0)

for i in range(10):
    # Generate a single image for the current label
    latent_vector = np.random.normal(0, 1, (1, latent_dim))
    label = np.array([i])  # Choose a label for the generated image
    one_hot_label = tf.keras.utils.to_categorical(label,
    num_classes)
    input_data = np.concatenate((latent_vector, one_hot_label),
    axis=1)
    generated_image = generator.predict(input_data)

    # Rescale the pixel values to the range [0, 255]
    generated_image = (generated_image * 255).astype(np.uint8)

    # Display the generated image
    ax = axs[i // num_cols, i % num_cols]
    ax.imshow(generated_image[0, :, :, 0], cmap="gray")
    ax.set_title(f"Label: {i}")
    ax.axis("off")

plt.show()
```

Visualization

Abstract:

In this project, our goal is to generate handwritten Persian numerals using Generative Adversarial Networks (GANs). Handwritten Persian numerals are of great importance in various fields, including text recognition, signature recognition research, and numeral recognition research. Additionally, data has become highly valuable and expensive nowadays, which is why GANs have become very efficient and cost-effective in generating reliable data in the shortest possible time. However, generating handwritten Persian numerals using traditional methods has its limitations. Hence, utilizing GANs can be an innovative approach to produce useful handwritten Persian numerals.

In this project, we attempt to generate numerals using Conditional Generative Models to overcome the limitations of random data generation. By conditioning the model, we can generate the desired data based on the specific type of numerals we want to produce. We also strive to optimize the hyperparameters to ensure that the resulting model achieves the highest accuracy and prevents overfitting.

For this project, we utilize the HODA dataset, which contains binary images of numerals from 0 to 9, for training and evaluation stages. In this report, we first introduce the topic of generating handwritten Persian numerals and discuss the significance of this research. Then, we examine the challenges and limitations of existing methods in the field of numeral generation. We also introduce the research questions and objectives of this project.

Keywords: Generating Handwritten Digit

Amirkabir University of Technology
Mathematics Computer Science Department

Generating Persian handwritten Digit

Bachelor of Science Thesis in Computer Science

By:

Amirmahdi Abootalebi

Supervisor:

Prof. Mostafa Shamsii

Summer 2023