پروژه درس یادگیری عمیق

امیرحسین امینی مهر ۹۹۷۲۲۱۶۲

محمدحسین خجسته ۹۹۷۲۲۵۴۲

StarGAN

تابستان ۱۴۰۰


بسم الله الرحمن الرحيم

چکیده:

مطالعات اخیر پیشرفت قابل توجهی را در زمینه ی تبدیل عکس به عکس برای دو حوزه مختلف نشان میدهد. (برای مثال تبدیل عکس سیب به پرتقال را می توان تبدیل عکس به عکس برای دو حوزه دانست. که در آن عکس های موجود در حوزه پرتقال به عکس هایی در حوزه سیب تبدیل می شوند.) اما روش های موجود برای چندین حوزه از نظر مقیاس پذیری و قابل اعتماد بودن بسیار ناکارآمد هستند. دلیل این ناکارآمدی این است که اگر بخواهیم چندین حوزه داشته باشیم باید برای عکس های هر دو حوزه به صورت مجزا یک مدل آموزش داده شود.

برای حل این مشکل راه حلی که در این مقاله عنوان شده است به عنوان شبکه مولد تخاصمی ستاره ای شناخته می شود که اجازه می دهد برای تبدیل عکس های چند حوزه به یکدیگر فقط نیاز به آموزش یک مدل داشته باشیم. در واقع با استفاده از ایده این مقاله می توانیم می توانیم به طور همزمان چندین مجموعه داده که هر کدام حوزه های مختلفی را دارند فقط با یک مدل آموزش دهیم.

فهرست مطالب:

مقدمه:

شبکه های مولد تخاصمی[1]:

شبکه های مولد تخاصمی جز شبکه های نوظهور و جدید در حوزه ی هوش مصنوعی هستند. در واقع می توان گفت که ایده ای که این شبکه ها مطرح کردند کاملا نو و بدیع بود و قبلا به آن پرداخته نشده بود. به طور کلی هدف از این شبکه ها ساختن یک داده ی واقعی از ابتدا می باشد. داده هایی که این شبکه ها می سازند معمولا در حوزه ی عکس قرار دارد اما در حوزه های دیگری مثل موسیقی نیز می توان از این شبکه ها استفاده کرد. اما بحث ما در اینجا در حوزه ی عکس می باشد و توضیحاتی هم که در ادامه داده خواهد شد با استفاده از مثال های این حوزه خواهد بود. پس به طور کلی و خلاصه می توان گفت هدف از این شبکه ها ساختن یک عکسی می باشد که از نظر چشم انسان واقعی باشد ولی عکس ساخته شده چیزی است که شبکه یاد گرفته آن را تولید کند و عکس از یک شی خاص در دنیای واقعی نیست. برای توضیح بیشتر از نحوه ی کارکرد این شبکه می توان از یکی از پروژه های معروف در این حوزه یعنی شبکه های مولد تخاصمی استایل[2] استفاده کرد. یکی از کاربرد های جدید این شبکه پروژه شبکه های مولد تخاصمی استایل[3] می باشد که هدف از این شبکه تولید تصاویر ساختگی از صورت انسان است. که این تصاویر با کیفیت بسیار بالا تولید می شوند و کاملا ساخته شده توسط شبکه می باشند و متعلق به شخص خاصی در دنیای واقعی نیستند. برای این پروژه یک وبسایت به آدرس درج شده در پاورقی[4] ساخته شده است که با رفرش کردن آن می توان تصاویر جدیدی تولید کرد. تصاویر تولید شده با کیفیت بسیار بالا هستند و از نظر چشم انسان کاملا واقعی می باشند ولی این تصاویر کاملا ساختگی هستند و متعلق به هیچ شخص خاصی نیستند.

---

[1] GAN

[2] StyleGan

[3] StyleGan

[4] https://thispersondoesnotexist.com/

این شبکه ها از دو شبکه ی عمیق مجزا تشکیل شده اند.

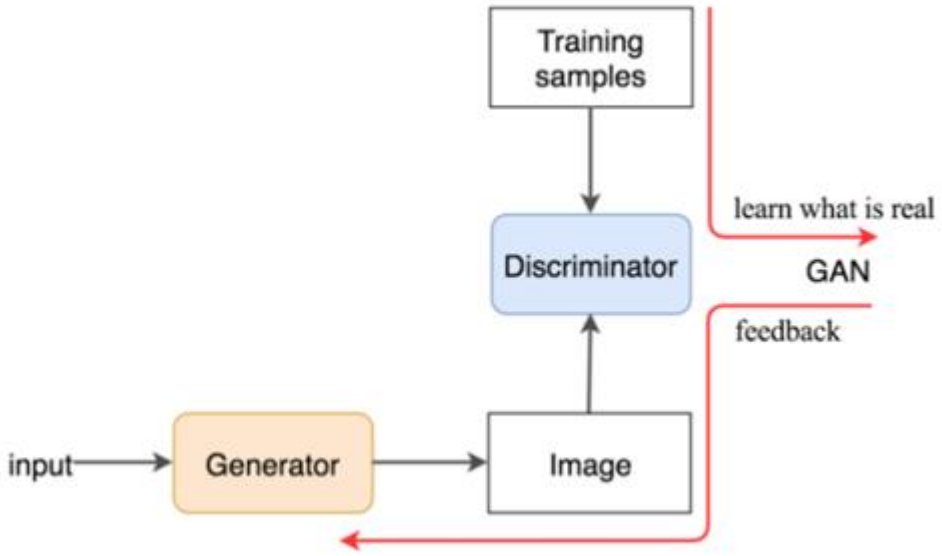
- شبکه ی تولید کننده[5]
- شبکه ی تفکیک کننده[6]

وظیفه ی شبکه ی تولید کننده این است که بتواند عکس های واقعی تولید کند. و وظیفه ی شبکه ی تفکیک کننده این است که بتواند عکس واقعی را از عکس ساختگی تشخیص دهد.

به طور خلاصه می توان گفت که شبکه های مولد تخاصمی از رقابت بین دو شبکه ی ذکر شده تشکیل شده اند. در واقع شبکه ی تولید کننده سعی می کند که عکس های کاملا واقعی تولید کند به نحوی که شبکه تفکیک کننده را به اشتباه و شبکه تفکیک کننده نیز سعی می کند بتواند تا حد امکان عکس های ساختگی از واقعی را تشخیص دهد. این رقابت بین این دو شبکه است که در نهایت باعث می شود ما یک شبکه تولید کننده داشته باشیم که بتواند عکس های بسیار واقعی تولید کند.

خروجی شبکه تفکیک کننده یک احتمال خواهد بود. که هر چه این احتمال به عدد یک نزدیکتر باشد یعنی شبکه تفکیک کننده آن را واقعی تر تشخیص داده است. خروجی تفکیک کننده به عنوان یک فیدبک به شبکه تولید کننده داده می شود تا بتواند عکس های واقعی تولید کند. در واقع می توان گفت این احتمالی که توسط شبکه تفکیک دهنده بدست می آید بعد از اعمال در یک تابع ضرر در شبکه پس انتشار[7] می شود تا شبکه تولید کننده آموزش ببیند و بتواند عکس های واقعی تولید کند. خلاصه عملکرد شبکه در زیر آمده است.

[5] generator
[6] discriminator
[7] Backpropagate



شکل ۱- عملکرد کلی شبکه ی GAN

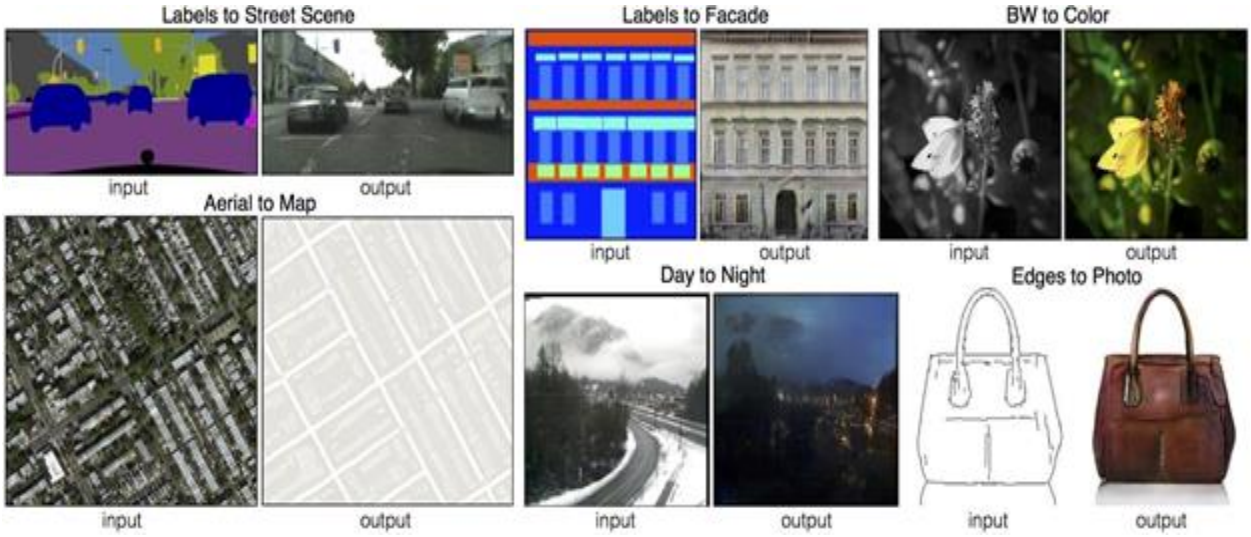
تبدیل عکس به عکس:

تبدیل عکس به عکس[8] یکی از تسک های معروف می باشد که در آن شبکه یک تصویر ورودی به عنوان ورودی دریافت می کند و یک و یک تصویر خروجی که تبدیل شده همان تصویر ورودی به یک تصویر دیگر می باشد را به عنوان خروجی می دهد. در تصویر زیر نمونه هایی از تبدیل تصویر به تصویر را می توان مشاهده کرد مانند:

- تبدیل تصاویر سیاه سفید به تصاویر رنگی
- تبدیل طرح[9] به یک عکس
- تبدیل تصویر یک منطقه در یک فصل به تصویر همان مکان در فصل دیگر
- تبدیل تصویر یک مکان در روز به تصویر همان مکان در شب
- و .....

[8] Image to Image translation
[9] Sketch

را مشاهده می کنید که تبدیل تصویر یک فرد به کاریکاتور آن هم یک ترجمه عکس به عکس می باشد.



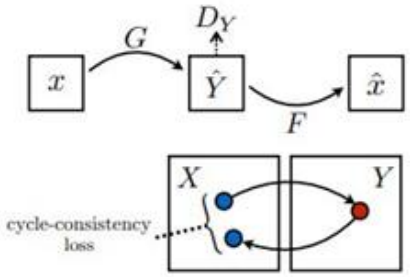
شکل ۲- نمونه هایی از کاربرد تبدیل تصویر به تصویر

شبکه های مولد تخاصمی چرخشی[10]:

در واقع شبکه های مولد تخاصمی چرخشی یک مدل بدون نظارت[11] از خانواده شبکه های مولد تخاصمی می باشد که به دلیل همین ویژگی مهم بدون نظارت بودن بسیار کاربردی می باشد در واقع شبکه های مولد تخاصمی چرخشی از دو عدد شبکه مولد تخاصمی استفاده می کند که یکی ازآنها X که دیتای ورودی ما می باشد را به Y که خروجی مورد نظر ما می باشد تبدیل می کند و مدل دیگر وظیفه دارد که Y تولید شده از مدل قبلی را تبدیل به X که همان ورودی اولیه ما می باشد تبدیل کند که در واقع به نوعی برچسب ما می باشد و در نهایت با مقایسه ورودی و خروجی مدل دوم می توانیم بفهمیم که مدل شبکه مولد تخاصمی چرخشی ما تا چه حدی عملکرد خوبی داشته و دقت آن چقدر بوده است.

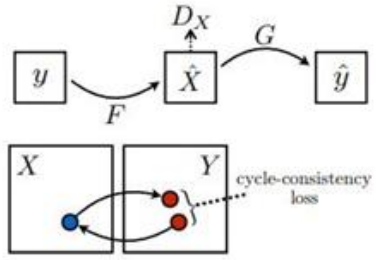
همانطور که در شکل پایین مشخص است شبکه های مولد تخاصمی چرخشی از دو مدل G و F تشکیل شده است که G برای تبدیل X به Y می باشد و F برای تبدیل Y به X.

[10] CycleGAN
[11] Unsupervised



شکل ۳- ساختار شبکه مولد تخاصمی چرخشی

راه دیگری که با آن می توانیم عملکرد مدل شبکه های مولد تخاصمی چرخشی خود را بسنجیم معرفی Y های تولید شده به عنوان ورودی برای مدل F و تبدیل آن به یک X و تبدیل دوباره آن به Y توسط مدل G می باشد که این بار باید Y ورودی و Y خروجی از مدل G را با هم مقایسه کنیم تا ببینیم مدل با چقدر خوب عمل کرده است.



شکل ۴- ساختار شبکه مولد تخاصمی چرخشی

بررسی مقاله:

[1] Choi, Yunjey, et al. "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

https://arxiv.org/abs/1711.09020

معرفی :

موضوع تبدیل عکس به عکس به این صورت تعریف می شود که بتوان جنبه های خاصی از یک عکس را چیز دیگری تبدیل کرد. برای مثال بتوان حالت صورت یک فرد از خوشحال را به ناراحت تبدیل کرد. این موضوع با معرفی شبکه های مولد تخاصمی پیشرفت بسیار زیادی کرده است. با دادن داده های آموزشی از دو حوزه ی متفاوت این مدل ها یاد میگیرند که عکس در یک حوزه را به عکس در حوزه ی دیگر تبدیل کنند.

منظور از یک صفت[12] در عکس یک ویژگی معنادار ذاتی مانند رنگ مو یا جنسیت یا سن در آن عکس است. و منظور از مقدار یک صفت[13] یک مقدار مشخص برای آن صفت است. برای مثال برای رنگ مو می توان مقادیر قهوه ای یا مشکلی را در نظر گرفت. یا برای جنسیت مقادیر زن یا مرد را در نظر گرفت. منظور از حوزه که قبلا به آن اشاره شد مجموعه ای از عکس هاست که دارای مقادیر صفت های مشترکی هستند. برای مثال با توجه به جنسیت به عنوان یک صفت عکس های زنان می تواند یک حوزه را در مقابل حوزه مردان تشکیل دهد.

در این مقاله از دو مجموعه داده زیر استفاده شده است:

- مجموعه داده CelebA: این مجموعه داده شامل ۴۰ دسته مختلف مربوط به ویژگی های صورت مثل رنگ مو و جنسیت و ... می باشد.
- مجموعه داده RaFD: این مجموعه داده شامل ۸ دسته مختلف مربوط به حالات احساسی صورت مثل خوشحال بودن یا ناراحت بودن است.

[12] Attribute
[13] Attribute value

در این مقاله دو موضوع مختلف به آن پرداخته شده است. یکی تبدیل در چندین حوزه هنگامی که عکس ها از یک مجموعه داده هستند. و دوم تبدیل چندین حوزه هنگامی که عکس ها از دو مجموعه متفاوت هستند. برای مورد اول از مجموعه داده CelebA استفاده شده است. به این صورت که یک عکس به عنوان ورودی داده می شود و ویژگی های مربوط به صورت مثل رنگ مو تغییر پیدا می کند. در مورد دوم دو مجموعه داده CelebA و RaFD با هم استفاده شده اند. به این صورت که حالت چهره عکس های موجود در مجموعه داده CelebA با توجه به ویژگی های موجود در RaFD عوض شود. این مورد در شکل زیر قابل مشاهده است :

شکل ۵- نمونه از نتایج به دست آمده توسط مدل

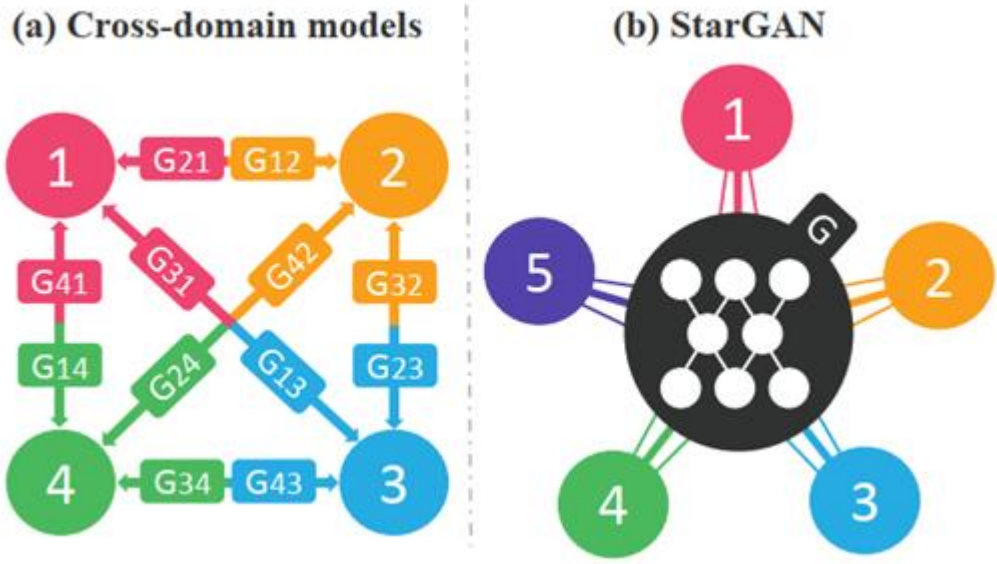
اگر بخواهیم از مدل های قبلی برای تبدیل عکس به عکس در چند حوزه استفاده کنیم باید برای هر دو جفت حوزه متفاوت یک تولید کننده آموزش داده شود. که باعث می شود برای k حوزه نیاز به k(k-1) تولید کننده داشته باشیم که هر کدام باید به صورت جداگانه آموزش داده شوند. که این کار بسیار ناکارآمد است. هم باعث می شود نیاز به هزینه آموزش زیادی داشته باشیم و هم نتایج مناسبی بدست نمی آید.

در مقابل روش شبکه مولد تخاصمی ستاره ای[14] وجود دارد که می تواند چندین مجموعه داده ی آموزشی در چند حوزه ی متفاوت را به عنوان ورودی بگیرد و تبدیل بین آن ها را فقط با یک تولید کننده یاد بگیرد. ایده به این صورت است که تولید کننده به عنوان ورودی هم عکس و هم اطلاعات حوزه را می گیرد و یاد می گیرد که به صورت منعطف عکس ورودی را به حوزه خواسته شده تبدیل کند. برای نشان دادن اطلاعات حوزه مورد نظر از یک برچسب استفاده می شود. که برای مثال این برچسب می تواند یک عدد دودویی یا یک بردار one-hot باشد. در طول زمان آموزش به صورت کاملا رندوم برچسب مربوط به حوزه ی خواسته شده تولید می شود و تولید

[14] StarGan

کننده یاد می گیرد که با توجه به برچسب داده شده عمل تبدیل را انجام دهد. این کار باعث می شود که در زمان تست نیز بتوان به راحتی حوزه ی خواسته شده را با برچسب موردنظر آن مشخص کرد.

تفاوت روش های قبل و روش شبکه مولد تخاصمی ستاره ای را می توان در شکل زیر مشاهده کرد :



شکل ۶- تفاوت روش شبکه مولد تخاصمی ستاره ای با روش های قبل

همانطور که در شکل بالا دیده می شود اگر بخواهیم از مدل های قبلی برای ۴ حوزه استفاده کنیم نیاز به آموزش ۱۲ تولید کننده متفاوت داریم اما با استفاده از شبکه مولد تخاصمی ستاره ای فقط با یک تولید کننده می توان تبدیلات را انجام داد.

توضیحات مربوط به شبکه مولد تخاصمی ستاره ای:

در این قسمت ابتدا توضیح داده می شود که چطور می توان تبدیل مربوط به چند حوزه را در داخل یک مجموعه داده انجام داد و در قسمت بعد درباره ی تبدیل چند حوزه مربوط به چند مجموعه داده توضیح داده خواهد شد.

الف) **تبدیل عکس به عکس بر روی یک مجموعه داده:**

هدف این است که بتوان یک تولید کننده G را آموزش داد به نحوی که بتواند یک ورودی x را به یک عکس خروجی y بر اساس یک برچسب c تبدیل کند. در هنگام آموزش برچسب c به صورت کاملا رندوم تولید می شود

تا تولید کنند یاد بگیرد که به صورت منعطف عکس ورودی را به حوزه ی خواسته شده تبدیل کند. همچنین برای تمایز دهنده نیز از یک دسته بند استفاده می کنیم که اجازه می دهد تمایز دهنده توانایی کنترل کردن حوزه های متفاوت را داشته باشد.

برای اینکه تبدیلات مطابق خواسته ما باشد سه نوع تابع ضرر[15] مختلف در این مقاله مورد استفاده قرار گرفته است که در ادامه معرفی شده اند:

- تابع ضرر تخاصمی[16]: از این تابع ضرر برای این استفاده می شود که مطمئن شویم عکس تولید شده از نظر چشم انسان یک عکس واقعی می باشد. می توان فرمول آن را در زیر مشاهده کرد :

$$\mathcal{L}_{adv} = \mathbb{E}_x \left[\log D_{src}(x)\right] + \mathbb{E}_{x,c}[\log\left(1 - D_{src}(G(x,c))\right)],$$

همان طور که در عکس بالا دیده می شود بین تولید کننده و تمایز دهنده یک رقابت در جریان است. عکس ورودی اصلی و عکس تولید شده به عنوان ورودی به تمایز دهنده داده می شود. تمایز دهنده باید سعی کند که تشخیص دهد آیا عکس های ورودی واقعی هستند یا خیر. هر چه خروجی تمایز دهنده به یک نزدیکتر باشد یعنی از نظر تمایز دهنده عکس واقعی تر است. در تابع ضرر بالا تمایز دهنده باید بتواند عکس ورودی را واقعی تشخیص دهد. همچنین باید عکس تولید شده توسط تولید کننده را غیر واقعی تشخیص دهد و مقدار تابع ضرر مربوط به عبارت تمایز دهنده در عبارت دوم را تا حد امکان کمینه کند. همچنین تولید کننده باید بتواند عکس های واقعی تولید کند به نحوی که تمایز دهنده را فریب دهد. این رقابت بین این دو شبکه باعث می شود که بتوانند عکس های واقعی تولید کنند.

- تابع ضرر دسته بندی دامنه[17]: از این لاس برای این استفاده می شود که عکس تولیدی مورد نظر در آن دسته ای باشد که لیبل ورودی مشخص می کند. یعنی مثلا اگر لیبل ورودی خندیدن

[15] Loss function
[16] Adversarial Loss Function
[17] Domain Classification Loss Function

را مشخص کرده با استفاده از این لاس مطمئن شویم عکس خروجی حالت چهره خندیدن را دارد. این تابع ضرر در زیر دیده می شود:

$$\mathcal{L}_{cls}^{f} = \mathbb{E}_{x,c}[-\log D_{cls}(c|G(x,c))].$$

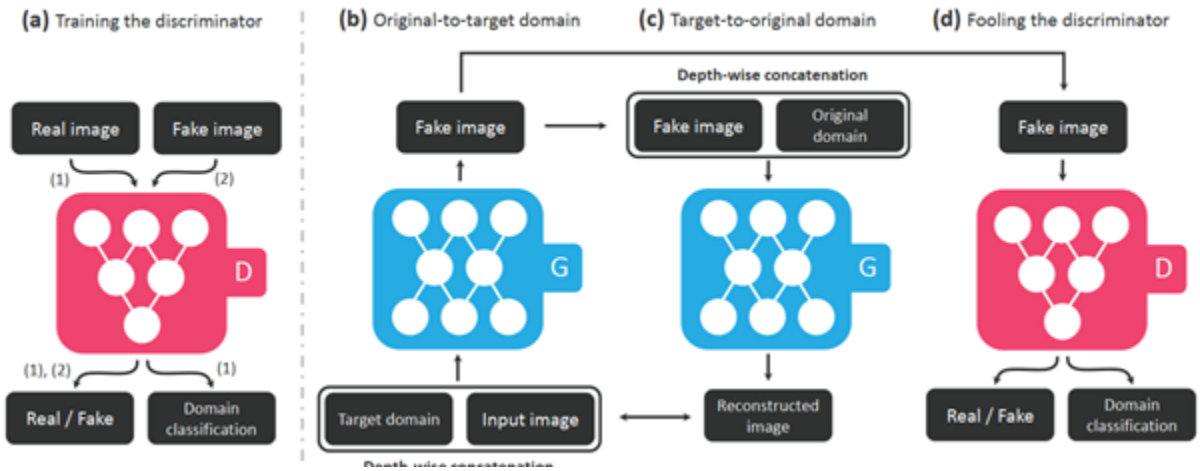
همانطور که دیده می شود تولید کننده باید یاد بگیرد عکس را به نحوی تولید کند که به برچسب خواسته شده **C** نزدیک باشد.

- تابع ضرر بازسازی[18]: با استفاده از دو لاس قبلی مطمئن میشویم عکس خروجی ما یک عکس واقعی خواهد بود که همان حالت چهره ای که به آن گفته شده را دارا می باشد. اما نمی شود مطمئن بود که در هنگام عوض کردن حالت چهره بقیه ی ویژگی های عکس ورودی را حفظ کند. با استفاده از این لاس مطمئن خواهیم شد که عکس خروجی در ویژگی هایی به جز حالت خواسته شده شبیه عکس ورودی باشد. که این کار با استفاده از تابع زیر انجام می شود:

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'}[||x - G(G(x,c),c')||_1],$$

همان طور که دیده می شود عکس تولیدی و برچسب اصلی دوباره به عنوان ورودی به تولید کننده داده می شود و تولید کننده تلاش می کند که عکس اصلی را دوباره بازسازی کند. و سپس عکس تولید شده از عکس اصلی کم می شود و نرم گرفته می شود. یعنی تولید کننده باید سعی کند عکس های تولیدیش تا حد امکان به عکس اصلی نزدیک باشد و به جز مورد خواسته شده مورد دیگری را تغییر ندهد. در شکل زیر می توان خلاصه روند آموزشی را مشاهده کرد:

[18] Reconstruction Loss Function



شکل ۷- معماری کلی شبکه ی مولد تخاصمی ستاره ای

در قسمت a تمایز دهنده را مشاهده می کنید. که وظیفه دارد دو چیز را یاد بگیرد:

- تشخیص عکس واقعی از غیرواقعی
- دسته بندی عکس

در قسمت سمت راست نیز کارهای مربوط به تولید کننده دیده می شود. اولا اینکه سعی می کند عکسی را بسازد که بتواند تمایزدهنده را گول بزند و در دسته ی خواسته شده باشد. ثانیا آن را دوباره تولید می کند تا یاد بگیرد که فقط ویژگی های خواسته شده را تغییر دهد.

ب) آموزش با استفاده از چند مجموعه داده :

یکی از مزایای شبکه مولد تخاصمی ستاره ای این است که می تواند بر روی چند مجموعه داده به صورت همزمان کار کند. مشکلی که در اینجا وجود دارد این است که برچسب های داده شده برای هر دو مجموعه داده به طور کامل شناخته شده نیستند. برای مثال مجموعه داده CelebA اطلاعات مربوط به حالت های صورت را ندارد. این قضیه هنگامی مشکل ساز می شود که بخواهیم تصویر را دوباره توسط تولید کننده تولید کنیم تا یاد بگیرد که فقط بخش های خواسته شده را تغییر دهد.

راه حلی که در این قسمت استفاده می کنیم این است که از یک وکتور ماسک m استفاده کنیم. به نحوی که برچسب های ناشناس را نادیده بگیرد و بتواند بر روی برچسب هایی که می شناسد تمرکز کند و بر اساس آن ها

عمل آموزش را انجام دهد. فرض کنیم برچسب ci شناخته شده است. اگر بقیه برچسب ها برای مجموعه داده مورد نظر شناخته شده نباشند به سادگی به جای برای آن ها مقادیر صفر را در نظر میگیریم. در هنگام آموزش نیز مدل یادمیگیرد که برچسب هایی که مقدار صفر را دارند در نظر نگیرد.

پیاده سازی:

معماری مدل : معماری شبکه تولید کننده شبکه مولد تخاصمی ستاره ای از مدل شبکه مولد تخاصمی چرخشی الهام گرفته شده است. به طور خلاصه میتوان گفت که دارای ویژگی های زیر است:

- استفاده از دو لایه های کانولوشنی با اندازه گام[19] برابر با ۲ برای نمونه گیری کاهشی[20]
- شش بلوک باقی مانده[21]
- استفاده از دو لایه های کانولوشنی با اندازه گام برابر با ۲ برای نمونه گیری افزایشی[22]

معماری این شبکه را می توان در شکل زیر مشاهده کرد:

Part	Input → Output Shape	Layer Information
Down-sampling	$(h, w, 3 + n_c) \to (h, w, 64)$	CONV-(N64, K7x7, S1, P3), IN, ReLU
	$(h, w, 64) \to (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \to (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256, K4x4, S2, P1), IN, ReLU
Bottleneck	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
Up-sampling	$(\frac{h}{4}, \frac{w}{4}, 256) \to (\frac{h}{2}, \frac{w}{2}, 128)$	DECONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \to (h, w, 64)$	DECONV-(N64, K4x4, S2, P1), IN, ReLU
	$(h, w, 64) \to (h, w, 3)$	CONV-(N3, K7x7, S1, P3), Tanh

Table 4. Generator network architecture

جدول ۱- معماری شبکه تولید کننده

[19] Stride
[20] Downsampling
[21] Residual Block
[22] Upsampling

برای تمایز دهنده نیز از شبکه مولد تخاصمی پچ[23] استفاده شده است که می توان معماری آن را در شکل زیر مشاهده کرد:

Layer	Input → Output Shape	Layer Information
Input Layer	$(h, w, 3) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{32}, \frac{w}{32}, 1024)$	CONV-(N1024, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{32}, \frac{w}{32}, 1024) \rightarrow (\frac{h}{64}, \frac{w}{64}, 2048)$	CONV-(N2048, K4x4, S2, P1), Leaky ReLU
Output Layer (D_{src})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (\frac{h}{64}, \frac{w}{64}, 1)$	CONV-(N1, K3x3, S1, P1)
Output Layer (D_{cls})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (1, 1, n_d)$	CONV-(N(n_d), K$\frac{h}{64}$x$\frac{w}{64}$, S1, P0)

Table 5. Discriminator network architecture

جدول ۲- معماری شبکه تمایزدهنده

[23] PatchGan

نتایج:

در این قسمت دو نوع جدول مختلف ارائه می شود.

ابتدا میزان کیفیت تصویر تولید شده در شبکه مولد تخاصمی ستاره ای در مقایسه با دیگر مدل ها. برای این مقایسه از سه مدل ذیل استفاده شده است. هر کدام برای انجام تبدیل عکس به عکس بر روی دو حوزه مختلف به صورت جداگانه آموزش داده شده اند.

برای معیار ارزیابی نیز از معیار AMT[24] استفاده شده است. می توان نتایج عملکردها را در جدول زیر مشاهده کرد.

Method	Hair color	Gender	Aged
DIAT	9.3%	31.4%	6.9%
CycleGAN	20.0%	16.6%	13.3%
IcGAN	4.5%	12.9%	9.2%
StarGAN	**66.2%**	**39.1%**	**70.6%**

Table 1. AMT perceptual evaluation for ranking different models on a single attribute transfer task. Each column sums to 100%.

Method	H+G	H+A	G+A	H+G+A
DIAT	20.4%	15.6%	18.7%	15.6%
CycleGAN	14.0%	12.0%	11.2%	11.9%
IcGAN	18.2%	10.9%	20.3%	20.3%
StarGAN	**47.4%**	**61.5%**	**49.8%**	**52.2%**

a

Table 2. AMT perceptual evaluation for ranking different models on a multi-attribute transfer task. H: Hair color; G: Gender; A: Aged.

جدول ۳- مقایسه کیفیت تولیدی تصاویر توسط شبکه های مختلف با استفاده از AMT

[24] Amazon Mechanical Turk

معیار دیگری که می توان بر اساس آن ارزیابی را انجام داد خطای دسته بندی عکس های تولیدی و همچنین تعداد پارامترها ست که می توان آن را در شکل زیر مشاهده کرد:

Method	Classification error	# of parameters
DIAT	4.10	52.6M × 7
CycleGAN	5.99	52.6M × 14
IcGAN	8.07	67.8M × 1
StarGAN	**2.12**	**53.2M × 1**
Real images	0.45	-

جدول ۴- مقایسه خطای دسته بندی عکس های تولیدی توسط شبکه های مختلف و تعداد پارامتر های آن ها

همانطور که مشاهده می شود تعداد پارامتر های مورد نیاز برای آموزش شبکه مولد تخاصمی ستاره ای بسیار کمتر از سایر مدل ها می باشد.

مجموعه داده و کد مورد استفاده:

برای پیاده سازی کد مربوط به این مقاله از مجموعه داده CelebA استفاده شده است. این مجموعه داده یک مجموعه داده در مقیاس بزرگ از ویژگی های صورت می باشد. در این مجموعه داده بیشتر از دویست هزار تصویر از افراد مشهور وجود دارد که برای هر کدام از این تصاویر چهل دسته ی مختلف مربوط به ویژگی های صورت مثل رنگ مو و جنسیت و سن و ... وجود دارد. این مجموعه داده دارای تنوع بسیار زیادی است و از جمله ویژگی های این مجموعه داده می توان به موارد زیر اشاره کرد :

- ۱۰۱۷۷ هویت مختلف
- ۲۰۲۵۹۹ تصویر مختلف
- ۴۰ ویژگی متفاوت برای هر تصویر

نمونه ای از تصاویر به کار رفته در این مجموعه داده را می توان در تصویر زیر مشاهده کرد:

شکل ۸- نمونه ای از تصاویر مجموعه داده CelebA

این مجموعه داده از طریق این لینک قابل دسترسی است.

همچنین برای پیاده سازی این مقاله از این کد استفاده شده است.

بررسی کد:

در این قسمت قصد بر این است که فایل های مورد استفاده در اجرای کد بررسی شود و توابع مهم هر کدام شرح داده و با قسمت های مختلف مقاله منطبق شود.

فایل data_loader.py:

به طور کلی هدف از این فایل و توابع موجود در آن آماده سازی داده می باشد. با استفاده از توابع موجود در این فایل پیش پردازش و آماده سازی های لازم بر روی داده ها صورت می گیرد تا اینکه داده ها آماده ی استفاده برای آموزش مدل شوند.

در کل پیاده سازی این کد از پنج برچسب مختلف استفاده شده است و سعی بر این است که تبدیلات بین این پنج برچسب یا ویژگی صورت بگیرد. این پنج برچسب عبارت اند از : مو مشکی، مو قهوای، مو بلوند، جوان و مرد. یعنی برای مثال سعی می شود مو یک فرد که مشکی است را به قهوه ای تبدیل کرد.

تابع get_data:

به طور کلی هدف از این تابع این است که برای هر تصویر مشخص کند کدام ویژگی ها از پنج ویژگی خواسته شده را دارا می باشد. یعنی هدف نهایی این است که بتوان برای هر تصویر یک بردار پنج تایی به عنوان برچسب نهایی آن داشته باشیم که هر درایه این بردار مشخص می کند کدام یک از ویژگی های خواسته شده در تصویر مورد نظر وجود دارد.

ورودی این تابع فایل مشخصات تصاویر است. این فایل در هنگام دانلود تصاویر به همراه آن ها دانلود می شود و مشخص می کند که هر تصویر کدام یک از ویژگی های ۴۰ گانه را دارد می باشد. ورودی دیگر این فایل ویژگی های انتخابی ما است که مشخص می شود کدام یک از ۴۰ ویژگی داده شده مورد نظر ماست و قصد داریم تبدیلات را بر روی آن انجام دهیم.

پردازش انجام شده در این تابع به این صورت است که برای هر تصویر به پنج ویژگی خواسته شده نگاه می کند و اگر تصویر مورد نظر آن ویژگی را داشته باشد درایه مربوط با آن ویژگی برای تصویر مورد نظر یک می شود و اگر

تصویر آن ویژگی را نداشته باشد درایه مربوط به آن صفر می شود. همچنین تقسیم بندی به داده های آموزش و تست در این تابع انجام می شود و تعداد دوهزار تصویر برای تست کنار گذاشته می شود.

این تابع چهار خروجی دارد که خروجی اول اسامی تصاویر موجود برای آموزش است و خروجی دوم برچسب این تصاویر است. برای هر تصویر برچسب آن یک بردار پنج تایی خواهد بود که نشان می دهد کدام از ویژگی هایی که توسط ما مشخص شده است را دارد و کدام یک را ندارد. خروجی سوم و چهارم اسامی تصاویر موجود برای تست و برچسب آن هاست.

تابع **convert_data_to_tfrecords**:

هدف از پیاده سازی این تابع است که داده های موجود به TFRecord تبدیل شوند. استفاده از TFRecord برای این است که هر داده ای را بتوان به یک فرمت قابل پشتیبانی تبدیل کرد که منطبق کردن مجموعه داده و معماری شبکه راحت تر صورت گیرد. یا طبق یک تعریف دیگر می توان گفت که روش برای ذخیره سازی دنباله ای از داده های باینری است.

فایده و مزیت مهم استفاده از TFRecord این است که کار را ساده تر و سریع تر می کند. یکبار پیش پردازش های مورد نیاز بر روی داده انجام می شود و ذخیره می شوند. بعد از آن هر بار که بخواهیم یک معماری جدید را امتحان کنیم تنها کافی است که از فایل ذخیره شده داده ها رو لود کنیم و دیگر نیاز به انجام همه ی پیش پردازش ها بر روی داده ها نیست.

به دلیل اینکه برچسب های مورد استفاده ما در این مجموعه داده یک لیست از داده های با نوع یکسان است باید از SequenceExample استفاده کنیم. این تبدیل با کمک تابع image_label_Example صورت می گیرد که در آن context استفاده شده ویژگی های تصویر مثل طول و عرض تصویر و تعداد کانال های آن است و feature_list آن بردار برچسب هاست که در قسمت قبل استخراج کردیم.

پیش پردازش های نهایی:

بعد از تبدیل کردن داده ها به TFRecord نیاز است که پردازش هایی نهایی که مورد نیاز داریم را بر روی آن اعمال کنیم.

ابتدا توسط تابع آماده list_file یک مجموعه داده از همه ی فایل های TFRecord که قبلا ذخیره کردیم و با فرمت داده شده منطبق هستند می سازیم. یعنی در واقع داده هایی که در قسمت قبل ذخیره کرده بودیم را در این قسمت بارگذاری می کنیم. یکی از پیش پردازش های مهمی که برای آموزش مدل نیاز است را بر روی داده ها اعمال می کنیم. این پیش پردازش مهم تولید برچسب هدف است. همانطور که در مقاله نیز ذکر شده است با استفاده از یک بردار c عمل تبدیل را انجام می دهیم. هر کدام از درایه های این بردار مشخص می کند که کدام یک از ویژگی ها را باید عکس ما داشته باشد و بر اساس آن مقدار خطا محاسبه می شود. یعنی کنترل کننده ی نحوه ی تبدیل و عکس خروجی که می خواهیم بردار c است. با استفاده از بردار هدف c مدل ما یاد می گیرد که تبدیلات را انجام دهد و در زمان تست نیز به راحتی با دادن بردار مورد نظر می توانیم تبدیل دلخواه را انجام دهیم. با توجه به مقاله این بردار c که بردار هدف ما است در زمان آموزش باید به صورت رندوم تولید شود. به همین دلیل یکی از پیش پردازش های این قسمت پیاده سازی تابع preprocess_for_training بر روی داده ها است که باعث می شود برچسب هدف به صورت رندوم تولید شود. این تابع در ادامه توضیح داده خواهد شد.

یکی از پیش پردازش های دیگری که در انتهای این فایل اعمال می شود اعمال تابع batch است که باعث می شود بتوان مجموعه داده را در batch های دلخواه در دست داشت. پیش پردازش دیگر اعمال تابع prefetch است که باعث می شود داده ی بعدی در هنگام پردازش داده ی فعلی آماده شود. که این کار باعث افزایش سرعت و بهینه سازی استفاده از ممووری می شود.

فایل train.py:

این فایل دومین فایلی است که پس از فایل data_loader.py در نوتبوک اجرا می شود که در این فایل با توجه به تابع های نوشته شده در فایل های Model.py و Utils.py فرآیند آموزش مدل انجام می شود و مقادیر توابع ضرر مختلف مربوط به شبکه ی تولید کننده و شبکه ی تفکیک کننده و مقدار تابع ضرر کلی نمایش داده می شود حال به سراغ بررسی جزئی تر آن می رویم.

در ابتدا ابر پارامتر های مربوط به مدل و همچنین تعدادی پارامتر دیگر تعریف شده که در ذیل آنها را بررسی می کنیم که ابر پارامتر[25] ها شامل پارامتر هایی مانند تعداد کلاس ها (که پنج تا هستند شامل موی بلوند، موی مشکی، موی قهوه ای، جوان و مرد)، سایز دسته ها[26] (که شانزده می باشد)، تعداد تکرارها[27] (که به ده می باشد)، تعداد تکرار هایی که لازم است که بگذرد تا کاهش نرخ یادگیری[28] شروع شود (که مقدار آن هفت می باشد)، تعداد حداکثر تکرار (در اینجا تکرار به معنی تعداد دسته هایی که توسط مدل آموزش داده می شود می باشد) هایی که آموزش ادامه خواهد داشت (که مقدار آن برابر با دویست هزار می باشد)، تعداد حداکثر تکرار هایی که لازم است تا کاهش نرخ یادگیری شروع شود (که مقدار آن برابر با صد هزار می باشد)، وزن های مربوط به هر یک از سه تابع ضرر مدل، تعداد آپدیت هایی که نیاز است برای شبکه ی تفکیک کننده انجام شود تا یک بار شبکه ی تولید کننده آپدیت شود، نرخ یادگیری شبکه ی تفکیک کننده و شبکه ی تولید کننده و مقدار بتا یک و بتا دو در بهینه ساز[29] آدام می باشد.

در ادامه تابع Main وجود دارد که آموزش در آن انجام می شود (در این بخش توابعی وجود دارد که در فایل های دیگر تعریف شده اند و در بخش مربوط به خود توضیح داده می شوند).

در ابتدا تصاویر و برچسب[30] های مربوط به داده های تست در یک متغیر ریخته می شوند و در ادامه تصاویر و برچسب های مربوط به داده های آموزشی که در فایل data_loader.py دریک فایل tfrecord ذخیره شده اند خوانده می شوند و پارس می شوند و بر روی آنها پیش پردازش[31] های در نظر گرفته شده انجام می شود و در ادامه دسته بندی می شوند و به دسته های مختلف تقسیم می شوند. در ادامه تعدادی برچسب رندوم برای داده های تست ساخته می شود و در ادامه شبکه ی تولید کننده و شبکه ی تفکیک کننده و تابع بهینه ساز مربوط به هر یک از آنها ساخته می شود و در ادامه گام های مربوط به هر تکرار به دست می آید و در یک حلقه[32]

[25] Hyperparameter
[26] Batch Size
[27] Epochs
[28] Learning Rate
[29] Optimizer
[30] Label
[31] preprocess
[32] Loop

فرآیند آموزش شبکه ی تولید کننده و شبکه ی تفکیک کننده انجام می شود که هر پنج باری که شبکه ی تفکیک کننده آپدیت می شود شبکه ی تولید کننده آپدیت می شود و مقدار تابع ضرر آنها نمایش داده می شود و با توجه به آن بهینه می شوند و هر پنج تکرار یک بار تست می شوند و مدل ذخیره می شود.

فایل **model.py**:

در این فایل دو تابع اصلی وجود دارد که Discriminator و Generator می باشند که مدل ما را تشکیل می دهند و همانطور که از نام آنها مشخص می باشد در آن ها ساختار شبکه ی تولید کننده و شبکه ی تفکیک کننده تعریف می شود که در این دو تابع از دو تابع دیگر به نام های Upsample و Downsample استفاده شده است که در ادامه توضیحاتی درباره ی آنها داده خواهد شد. همچنین در این فایل تابع های کمکی ResidualBlock، get_activation، get_norm_layer و build_model تعریف شده اند که درباره هر یک از آنها توضیح داده خواهد شد.

- **get_norm_layer**:

یک رشته[33] به عنوان ورودی دریافت می کند و با توجه به آن یا لایه Batch Normalization یا لایه Instance Normalization را بر می گرداند.

- **get_activation**:

یک رشته به عنوان ورودی می گیرد و یا یک لایه ی تابع فعالسازی ReLU یا LeakyReLU یا tanh را بر می گرداند.

[33] String

- **ResidualBlock**:

در این بخش ساختار بلوک باقی مانده پیاده سازی می شود که از دو لایه کانولوشنی تشکیل شده است و مانع محو شدگی گرادیان[34] می شود که در ساختار شبکه تولید کننده به همین دلیل از آن استفاده شده است.

- **build_model**:

در این تابع مدل ساخته می شود و تابع های Discriminator و Generator که در ادامه ساختار آنها توضیح داده می شود در آن فراخوانی می شود و ساختار آنها نیز به نمایش در می آید[35].

حال سراغ توضیح تابع های Upsample و Downsample می رویم که هر کدام از یک لایه تشکیل شده اند و در توابع Discriminator و Generator استفاده می شوند در تابع Downsample یک لایه کانولوشنی دو بعدی داریم که می تواند دارای لایه عادی ساز[36] باشند یا نباشد و یک لایه تابع فعال سازی نیز در انتهای آن قرار دارد. در تابع Upsample برعکس کار تابع Downsample انجام می شود و مانند همان تابع Downsample می باشد با این تفاوت که در آن بجای لایه کانولوشن دو بعدی لایه عکس کانولوشن دو بعدی وجود دارد.

بخش اصلی این فایل دو تابع Discriminator و Generator می باشند که دقیقا مانند ساختار شبکه تولید کننده و شبکه تفکیک کننده توضیح داده شده در مقاله می باشند به این صورت که در تابع Discriminator دقیقا مانند مقاله در ابتدا شش بلوک

نمونه گیری کاهشی با گام دو و سایز کرنل چهار داریم که تعداد فیلتر های هر کدام دو برابر قبلی می باشد و در انتها دو لایه کانولوشنی با سایز کرنل سه و گام یک به صورت موازی داریم که هر کدام یک خروجی تولید می کنند یکی خروجی مربوط به واقعی یا غیرواقعی بودن تصویر داده شده و دیگری نتیجه دسته بندی تصویر ورودی. تابع بعدی تابع Generator می باشد که دقیق مانند شبکه تولید کننده گفته شده در مقاله می باشد که در

[34] Gradient Vanishing
[35] model.summary()
[36] Normalization

ابتدا سه لایه نمونه گیری کاهشی با گام دو دارد و سپس شش بلوک باقی مانده در ادامه آن می باشد که تعداد فیلتر های هر یک برابر با ۲۵۶ عدد می باشد و در ادامه دو لایه نمونه گیری افزایشی وجود دارد تا تصویر را بتواند به سایز اصلی بازگرداند و یک تصویر جدید هم سایز با تصویر ورودی تولید کند و در ادامه یک لایه کانولوشنی هم وجود دارد و برای آنکه تصویر خروجی هم سایز با تصویر ورودی باشد در ابتدا و انتهای تابع به آن Padding اضافه شده است.

فایل utils.py:

هدف از این فایل پیاده سازی توابع مختلفی است که در هنگام پیش پردازش یا آموزش مورد استفاده قرار می گیرند. یعنی توابع موجود در این فایل توابع کمک کننده ای هستند که در فرآیند آماده سازی و آموزش مورد استفاده قرار می گیرند. در این قسمت توابع مهم این فایل مورد بررسی قرار خواهد گرفت.

تابع preprocess_for_training:

همانطور که در قسمت قبل گفته شد تابع برای ساختن برچسب های هدف به صورت رندوم در زمان آموزش استفاده می شود. این تابع برچسب اصلی عکس را دریافت می کند و به صورت رندوم جای درایه های آن را تغییر می دهد و به عنوان برچسب هدف ارائه می دهد.

همچنین کار دیگری که در این تابع انجام می شود پیش پردازش عکس ها است. کارهایی مانند نرمال سازی تصویر یا کراپ کردن در این تابع انجام می شود.

تابع get_gradient_penalty:

یکی از خطاهایی که در مقاله WGAN-GP معرفی شد خطای gradient penalty می باشد. این خطا باعث بهبود همگرایی شبکه های GAN می شود. یعنی در واقع این خطا باعث می شود که آموزش پایدارتر باشد و فرآیند آموزش راحت تر انجام شود در نتیجه می توان از مدل های پیچیده تر استفاده کرد. در این تابع این خطا محاسبه و برگردانده می شود.

تابع **get_classification_loss**:

یکی از خطاهای مورد استفاده برای آموزش مدل خطای مربوط به دسته بندی است. این خطا با نام تابع ضرر دسته بندی دامنه در مقاله نام برده شده است که از این خطا برای این استفاده می شود که عکس تولیدی مورد نظر در آن دسته ای باشد که برچسب ورودی مشخص می کند. ورودی این تابع دسته های واقعی عکس مورد نظر که همان برچسب c است به همراه دسته بندی انجام شده توسط مدل است که در این تابع توسط BinaryCrossentropy میزان خطا به دست می آید و برگردانده می شود.

تابع **update_lr_by_iter**:

با استفاده از این تابع می توان بعد از یک تعداد تکرار مشخص میزان نرخ یادگیری را آپدیت کرد و کاهش داد.

تابع **train_disc**:

از این تابع برای محاسبه خطا و بروزرسانی تمایز دهنده[37] استفاده می شود. ابتدا عکس های واقعی به تمایز دهنده داده می شوند و میزان واقعی بودن و دسته بندی این عکس ها از تمایز دهنده گرفته می شود. همانطور که می دانیم تمایز دهنده باید سعی کند به عکس های واقعی برچسب نزدیک به یک بدهد و هر چه به این عکس ها احتمال بالاتری برای واقعی بودن بدهد بهتر است. همچنین باید یاد بگیرد که دسته بندی این عکس ها را به درستی انجام دهد. میزان خطای تمایز دهنده بر اساس این دو خروجی محاسبه و ذخیره می شود. بعد از آن عکس های واقعی به تولید کننده[38] داده می شود تا تبدیل را بر روی آن انجام دهد. سپس عکس های تولیدی توسط تولید کننده به تمایز دهنده داده می شود تا برچسب واقعی بودن آن ها را تولید کند. تمایز دهنده باید یاد بگیرد که احتمال واقعی بودن کمی به عکس های تولیدی تولید کننده بدهد تا بتواند میزان واقعی بودن از ساختگی بودن تصویر را تشخیص دهد.

[37] Discriminator
[38] Generator

بعد از محاسبه خطاهای گفته شده میزان خطای کلی با مجموع وزن دار این خطا ها محاسبه می شود. این وزن ها نیز یک هایپر پارامتر هستند و توسط ما مشخص می شوند. بعد از محاسبه ی خطای کلی آپدیت وزن های تمایز دهنده انجام می شود.

تابع **train_gen**:

از این تابع برای محاسبه خطای مربوط به تولید کننده و بروزرسانی وزن های آن استفاده می شود. ابتدا عکس های واقعی به همراه تبدیلات خواسته شده به تولید کننده داده می شود تا تبدیلات را بر روی عکس ها انجام دهد. سپس عکس های تولیدی به تمایز دهنده داده می شود و میزان واقعی بودن و دسته ی آن ها از تمایز دهنده گرفته می شود. تولید کننده باید تلاش کند عکس های واقعی تولید کند به نحوی که تمایز دهنده را به اشتباه بیاندازد. به همین دلیل باید در این قسمت خروجی تمایز دهنده برای عکس های تولید کننده نزدیک به یک باشه. و هر چه که از یک دور تر باشد یعنی تمایز دهنده توانسته تشخیص دهد که عکس های تولید کننده واقعی نبوده و این یعنی تولید کننده عکس های خوبی تولید نمی کند و باید بروزرسانی شود. خطای دیگر مربوط به دسته بندی عکس هاست. تولید کننده باید عکس ها را به نحوی تغییر دهد که همان دسته ای که ما خواسته بودیم تولید شود. این دسته توسط بردار **c** تعیین می شود. اگر دسته بندی اشتباه انجام شده باشد نشان می دهد که عکس های تولید کننده مناسب نیست و باید بروزرسانی شود.

خطای دیگری که در مقاله نیز به آن اشاره شده است خطای بازسازی است. با استفاده از این خطا قصد داریم که مطمئن شویم فقط ویژگی های خواسته شده تغییر کرده اند و بقیه ی ویژگی های صورت بدون تغییر باقی مانده اند. برای اینکار عکس تولیدی و برچسب اصلی را به تولید کننده می دهیم و از تولید کننده می خواهیم که عکس اصلی را تولید کند. سپس اختلاف بین عکس اصلی و عکسی که در این مرحله تولید شده است به عنوان خطا در نظر گرفته می شود. با این خطا تولید کننده یاد می گیرد که عکس های تولیدی اش باید تا حد امکان نزدیک به عکس های اصلی باشند و تغییری در ویژگی های خواسته نشده ندهد.

در نهایت میانگین وزن دار خطاها به دست می آید و تولید کننده بروزرسانی می شود.

تابع **get_models_for_testing**:

با استفاده از این مدل آخرین checkpoint ذخیره شده در فرآیند آموزش بارگذاری می شود و مدل ها برای تست کردن ساخته می شوند.

تابع **test_image**:

این تابع برای تست یک عکس به کار می رود. ابتدا تصویر داده شده نرمال می شود. ویژگی های خواسته شده تبدیل به بردار c می شود و به همراه تصویر به مدل داده می شود. خروجی تولید کننده دریافت می شود و در کنار عکس اصلی رسم می شود.

نتایج آموزش مدل موجود در مقاله

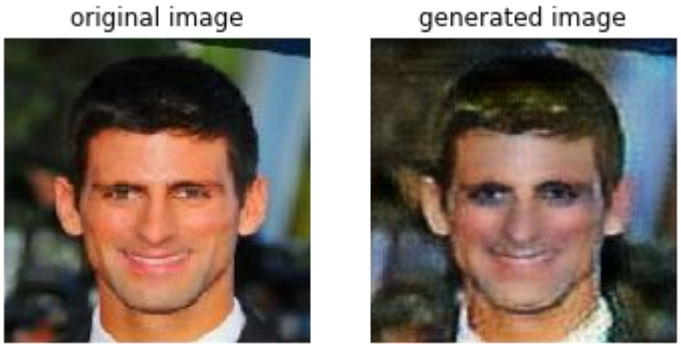
برای آموزش مدل مقاله ابتدا آن را از گیت دانلود کرده و دو خط مربوط به ساختن tfrecord ها را در فایل data_loader.py از حالت کامنت در آورده که بتوان داده های آموزشی را در آن ذخیره کرد و سپس در ابتدا فایل data_loader.py را اجرا کرده که tfrecord ها ساخته شوند که داده ها را به فرمت باینری می برد تا بتوان راحت تر با آنها کار کرد در ادامه فایل train.py را اجرا می کنیم و مدل شروع به آموزش می کند و در ادامه با استفاده از توابع get_models_for_testing و test_image jsj تست های مورد نظر انجام می شود و می توان عملکرد مدل را مشاهده کرد.

یکی از چالش های اجرای کد این بود که با توجه به تعداد زیاد تصاویر مجموعه داده نمی توان از همه آن برای آموزش استفاده کرد زیرا در این صورت فرآیند آموزش زمان بسیار زیادی را نیاز خواهد داشت به عنوان مثال حتی اگر تعداد داده های آموزش برابر با پنجاه هزار تصویر باشد برای هر تکرار ۲۷ دقیقه زمان نیاز است برای همین از پنجاه هزار داده و سی هزار داده که برای هر تکرار در آموزش به پانزده دقیقه زمان نیاز است استفاده شده و تعداد تکرار های آموزش هم برای سی هزار داده ده و برای پنجاه هزار داده نه قرار داده شده است.

تصاویر ذیل نمونه هایی از تست های انجام شده روی مدل آموزش دیده برای سی هزار داده مقاله می باشد:

شکل ۹- تغییر رنگ مو به قهوه ای

شکل ۱۰- تغییر رنگ مو به مشکی



شکل ۱۱- تغییر رنگ مو به بلوند

شکل ۱۲- تغییر جنسیت و تبدیل به مرد

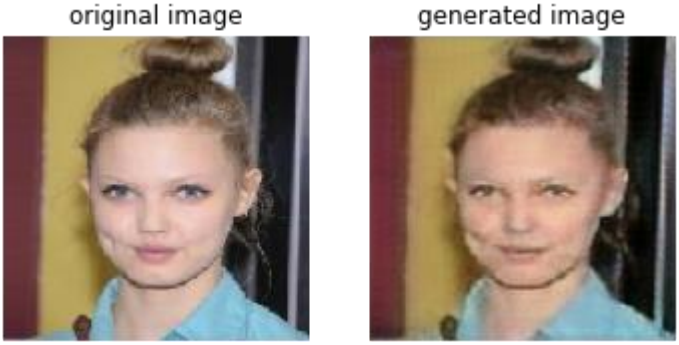
شکل ۱۳- تغییر سن و نوجوان کردن

تصاویر ذیل هم نمونه هایی از تست های انجام شده روی مدل آموزش دیده برای پنجاه هزار داده مقاله می باشد:

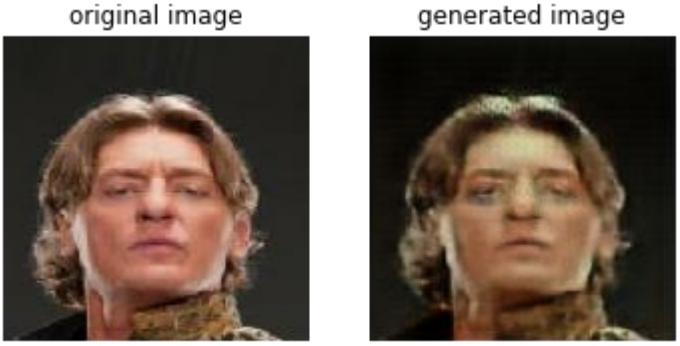
شکل ۱۴- تغییر رنگ مو به مشکی

شکل ۱۵- تغییر رنگ مو به بلوند

شکل ۱۶- تغییر رنگ مو به قهوه ای



شکل ۱۷- تغییر جنسیت و تبدیل به مرد



شکل ۱۸- تغییر سن و نوجوان کردن

بهبود مقاله:

برای بهبود مقاله از ایده های موجود در مقاله LAUN[39] استفاده شده است. این مقاله در سال ۲۰۲۰ ارائه شده است و حاوی ایده هایی برای بهبود شبکه StarGAN می باشد. ایده ی اولی که در این مقاله مطرح می شود معرفی یک خطای جدید به نام Contextual loss می باشد. در این خطا بیان می شود که برای مقایسه بین دو تصویر به جای مقایسه بین پیکسل ها از ویژگی های استخراج شده توسط یک مدل از قبل آموزش دیده شده مثل VGG استفاده شود. ایده دیگری که در این مقاله بیان شده است استفاده از L۲ به جای L۱ در هنگام محاسبه تفاوت بین عکس تولیدی و عکس اصلی در خطای Reconstruction است. علت کار اینطور بیان شده است :

- L1 نمی تواند هنگامی که در مناطقی تفاوت زیاد است میزان خطای بیشتر توجه بیشتری به آن بدهد. اما در L2 چون میزان اختلاف ها به توان دو می رسد اگر تفاوت زیادی بین دو ناحیه باشد این تفاوت بزرگتر شده و در خطای نهایی تاثیر بیشتری میگذارد و در واقع به آن توجه بیشتری می شود.
- محاسبه L2 پیچیده است و ممکن است چندین جواب بهینه داشته باشد.

برای بهینه سازی کد این مقاله از دو ایده بالا استفاده شد و با ترکیب این دو ایده خطای مربوط به Reconstruction تغییر پیدا کرد. هدف از استفاده از خطای Reconstruction این است که فقط ویژگی های خواسته شده تغییر کنند و ویژگی های دیگر صورت تغییر نکنند و تا حد ویژگی های دیگر صورت و هویت عکس اول بدون تغییر باقی بماند. به همین دلیل این حدس زده شده که شاید استفاده از ویژگی های استخراج شده از یک مدل از قبل آموزش دیده شده به جای مقایسه بین پیکسل ها مناسب تر باشد. دلیل این کار به صورت زیر است:

- اولا مقایسه پیکسلی تفاوت دقیق بین پیکسل ها را می سنجد و اگر حتی تفاوت کوچکی بین پیکسل های دو عکس ایجاد شده باشد که خیلی از دید کاربر قابل تشخیص نیست باز هم این تفاوت در خطا محاسبه می شود.

[39] X. Wang, J. Gong, M. Hu, Y. Gu and F. Ren, "LAUN Improved StarGAN for Facial Emotion Recognition," in IEEE Access, vol. 8, pp. 161509-161518, 2020, doi: 10.1109/ACCESS.2020.3021531.

- یکی از اهداف مهم خطای Reconstruction این است که هویت فرد تا حد امکان ثابت باقی بماند. به همین دلیل استفاده از ویژگی های استخراج شده توسط یک مدل از قبل آموزش دیده شده خیلی بهتر به این کار کمک می کند. با استفاده از این مدل ها و استفاده از چندین لایه ی کانولوشنی بردار ویژگی مربوط به چهره استخراج می شود. که این بردار حاوی اطلاعات مهمی است که از تصویر استخراج شده است. با استفاده از این بردار به نسبت تفاوت بین پیکسل ها خیلی بهتر و راحت تر می توان نتیجه گرفت که آیا هویت در تصویر تولیدی حفظ شده است یا خیر؟

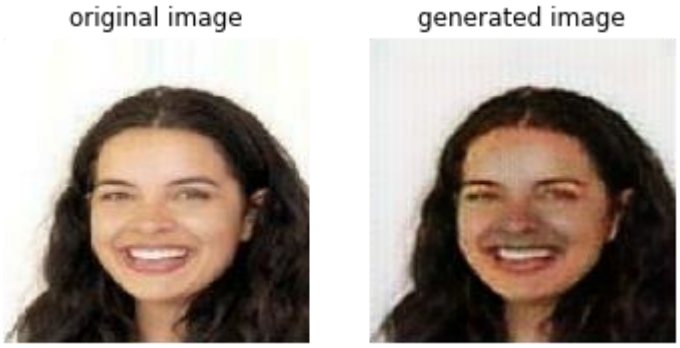
تغییر دوم استفاده از L2 به جای L1 برای محاسبه تفاوت بین این بردارهای ویژگی هاست. چون همانطور که گفته شد اگر تغییری زیادی در بردار ها باشد با استفاده از L2 می توان توجه بیشتری به آن کرد و همچنین محاسبات مربوط به آن راحت تر است.

با استفاده از این تغییرات نتایج زیر به دست آمد:

شکل ۱۹- تغییر رنگ مو به قهوه ای

شکل ۲۰- تغییر رنگ مو به مشکی

شکل ۲۱- تغییر رنگ مو به بلوند



شکل ۲۲- تغییر جنسیت و تبدیل به مرد

شکل ۲۳- تغییر سن و نوجوان کردن

پس از استفاده از ایده بالا و مشاهده بهبود قابل توجه مقاله یک راه حل جدید به ذهن رسید آن هم استفاده از یک مدل VGG با لایه آخری که پنج خروجی با تابع فعال سازی Sigmoid دارد و روی مجموعه داده CelebA تنظیم دقیق[40] شده می باشد که در تنظیم دقیق حتی دقت آموزش از مقدار پنجاه بالاتر نمی رفت و نتیجه ی خوبی را به همراه نداشت و نتایج ذیل در تست به دست آمد:

شکل ۲۴- تغییر رنگ مو به مشکی

شکل ۲۵- تغییر رنگ مو به بلوند

[40] Fine tune

شکل ۲۶- تغییر رنگ مو به قهوه ای

شکل ۲۷- تغییر جنسیت و تبدیل به مرد

شکل ۲۸- تغییر سن و نوجوان کردن

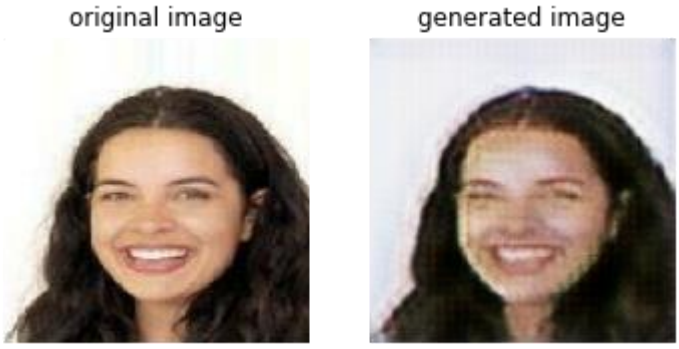
سپس از آنجا که دقت آموزش پایین بود تصمیم به استفاده از ResNet به جایی VGG شد که باز دقت آموزش تغییر قابل توجهی نکرد و باز هم دقت آموزش پایین بود در ادامه از یک مدل VGG که لایه های تماما متصل[41] آن حذف شده و بجای آن لایه کانولوشن قرار داده شده استفاده شد که باز هم دقت مربوط به داده های آموزشی تغییر آنچنانی نکرد و نتایج تست مطلوبی را به همراه نداشت که نتایج به صورت ذیل می باشند:

شکل ۲۹- تغییر رنگ مو به مشکی

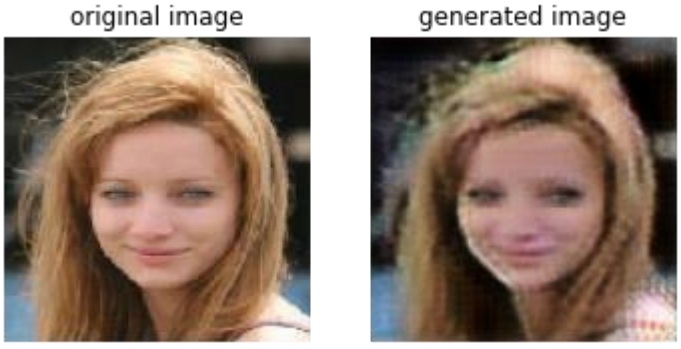
شکل ۳۰- تغییر رنگ مو به بلوند

[41] Fully connected

شکل ۳۱- تغییر رنگ مو به قهوه ای



شکل ۳۲- تغییر جنسیت و تبدیل به مرد



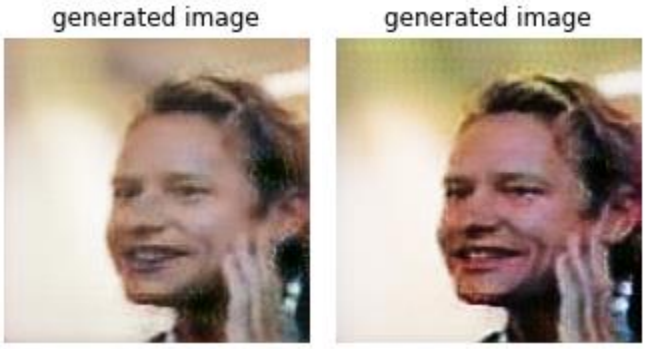
شکل ۳۳- تغییر سن و نوجوان کردن

در ادامه از مدل های EfficientNet و VGGFace استفاده شد که باز هم نتیجه مطلوبی را به همراه نداشت.

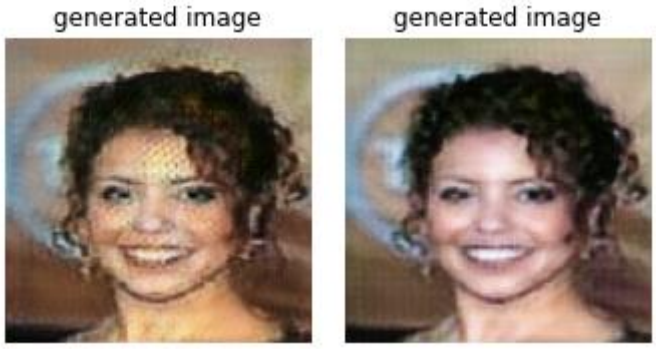
مقایسه و نتیجه گیری:

از مقایسه نتایج به دست آمده توسط مدل مقاله و مدل بهبود داده شده می توان دریافت که مدل بهبود داده شده کاملا برتر است و تصاویر با کیفیت بالایی تولید کرده است که نمونه هایی از این مقایسه ها بر روی تصاویر داده های تست به صورت ذیل می باشد:

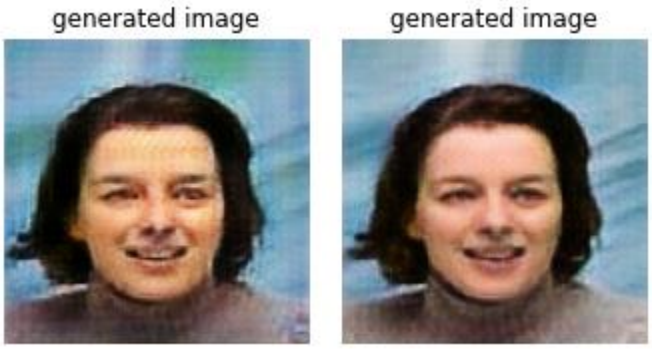
شکل ۳۴- تغییر رنگ مو و تبدیل به قهوه ای. عکس سمت راست برای مدل بهبود یافته است



شکل ۳۵- تغییر رنگ مو و تبدیل به مشکی. عکس سمت راست برای مدل بهبود یافته است



شکل ۳۶- تغییر رنگ مو و تبدیل به قهوه ای. عکس سمت راست برای مدل بهبود یافته است



شکل ۳۷- تغییر جنسیت و تبدیل به مرد. عکس سمت راست برای مدل بهبود یافته است

همانطور که مشخص است مدل بهبود داده شده کاملا بهتر عمل کرده و مقایسه ها نشان می دهد که بهبود مدل موفقیت آمیز بوده است.

آموزش مقاله بر روی داده های دیگر:

مجموعه داده ی دیگری که برای آموزش مدل استفاده شد مجموعه داده AFHQ بود. این مجموعه داده شامل ۱۵۰۰۰ عکس از حیوانات است. این مجموعه داده در سه دسته گربه و سگ و حیوانات وحشی ارائه شده است. هدف از استفاده از این مجموعه داده این است که بررسی شود آیا می توان تغییرات بیشتری با استفاده از StarGAN در تصاویر داد یا خیر؟ یعنی به جای اینکه صرفا رنگ مو عوض شود بتوان کاملا تصویر یک گربه را برای مثال به تصویر سگ تبدیل کرد.

بعد از دانلود مجموعه داده نیاز بود که این مجموعه داده تبدیل به همان فرمت مجموعه داده CelebA شود تا نیاز به تغییرات در کد نباشد. برای اینکار ابتدا تصاویر از پوشه های متفاوت به یک پوشه منتقل شدند. برای ساخت فایل مربوط به ویژگی های تصاویر نیز به جز ۳ ویژگی بقیه ی ۳۷ ویژگی برای همه ی تصاویر صفر در نظر گرفته شد. یکی از ویژگی ها برای گربه یک شد. یکی از ویژگی ها برای سگ و یکی برای حیوانات وحشی یک شد. یعنی هر عکس فقط یک ویژگی دارد که یک است و بقیه ی ویژگی های آن صفر است و این ویژگی یک نشان دهنده ی دسته ی تصویر مربوط است. با این تغییرات دیگر نیازی به تغییر در کد نبود و به راحتی همان کد قبل قابل اجرا بود.

در اولین اجرا نتایج بدست آمده اصلا مناسب نبودند. این ایده مطرح شد که احتمالا به دلیل خطای Reconstruction این اتفاق رخ می دهد. با استفاده از این خطا تلاش می شود که تصویر تولیدی تا حد امکان شبیه به تصویر اولیه باشد و به همین دلیل امکان تغییرات بسیار زیاد در تصاویر را نمی دهد. به همین دلیل مقدار وزن مربوط به این خطا بسیار کم و نزدیک به صفر قرار داده شد. بعضی از نتایج بدست آمده توسط این مجموعه داده در زیر قابل مشاهده است.

شکل ۳۸- تصاویر تولید شده با استفاده از مجموعه داده AFHQ

همانطور که در تصاویر بالا مشخص است تغییرات در تصاویر بسیار زیاد بوده ولی نتوانسته کاملا تصویر یک حیوان را تبدیل به تصویر یک حیوان دیگر کند. این نقطه ضعف ممکن است به دلیل تعداد کم تصاویر باشد یا تعداد تکرارهای کمی که بدلیل محدودیت های کولب نمی توانستیم بیشتر از آن قرار دهیم. و یا ممکن است نیاز باشد که تغییراتی بر روی کد اجرا شود که می توان به عنوان کارهای آینده به آن فکر کرد.