به نام خدا

دانشگاه تهران

دانشکده مهندسی

برق و کامپیوتر

درس شبکه های عصبی و یادگیری عمیق

تمرین ششم

پرسش ۱	نام و نام خانوادگی	علی مرجبی
	شماره دانشجویی	610300104
پرسش ۲	نام و نام خانوادگی	رضا دلیر
	شماره دانشجویی	610300050
	مهلت ارسال پاسخ	**1404/3/18**

فهرست

قوانین

قبل از پاسخ دادن به پرسش ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ های خود یک گزارش در قالبی که در صفحه ی درس در سامانه ی Elearn با نام ***REPORTS_TEMPLATE.docx*** قرار داده شده تهیه نمایید.
- پیشنهاد می شود تمرین ها را در قالب گروه های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژهاي برخوردار است**؛ بنابراین، لطفا تمامی نکات و فرض هایی را که در پیاده سازی ها و محاسبات خود در نظر میگیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل ها زیرنویس و برای جدول ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- **تحلیل نتایج الزامی می باشد، حتی اگر در صورت پرسش اشاره ای به آن نشده باشد.**
- **دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند**؛ بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر میشود.
- **کدها حتما باید در قالب نوت بوک با پسوند .ipynb تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتما در این فایل ارسالی شما ذخیره شده باشد.** بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده اید، این نمودار باید هم در گزارش هم در نوت بوک کد ها وجود داشته باشد.
- **در صورت مشاهده ی تقلب امتیاز تمامی افراد شرکت کننده در آن، 100- لحاظ میشود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- **استفاده از کدهای آماده برای تمرینها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می شود.**
- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.
    - سه روز اول: بدون جریمه
    - روز چهارم: ۵ درصد
    - روز پنجم: ۱۰ درصد
    - روز ششم: ۱۵ درصد

- روز هفتم: ۲۰ درصد

- حداکثر نمره ای که برای هر سوال می توان اخذ کرد ۱۰۰ بوده و اگر مجموع بارم یک **سوال** بیشتر از ۱۰۰ باشد، در صورت اخذ نمره بیشتر از ۱۰۰، اعمال نخواهد شد.
  - برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.
- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه ی Elearn بارگذاری نمایید:

HW[Number] _[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip

(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

- برای گروه های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می شود هر دو نفر بارگذاری نمایند.

پرسش 1- یادگیری بدون نظارت و انتقال دامنه با استفاده از GAN

1-1 مقدمه

یکی از چالش‌های مهم در یادگیری ماشین و بینایی ماشین، مسئلهٔ تطبیق دامنه (Domain Adaptation) است. در بسیاری از کاربردهای واقعی، داده‌های آموزشی و داده‌های آزمون از توزیع‌های متفاوتی می‌آیند؛ به‌عبارت دیگر، مدلی که بر روی یک دامنه آموزش دیده است، ممکن است روی دامنه‌ای دیگر به‌خوبی عمل نکند. این مشکل به‌ویژه در مسائلی مانند تشخیص تصویر، گفتار، یا متن دیده می‌شود که حتی تفاوت‌های ظاهری کوچک می‌توانند عملکرد مدل را به‌شدت کاهش دهند.

در این تمرین، هدف ما پیاده‌سازی و بررسی عملی یک روش تطبیق دامنه به‌نام Pixel-Level Domain Adaptation (PixelDA) است. این روش با استفاده از ساختار شبکه‌های مولد تقابلی (GAN)، تصاویر دامنهٔ منبع را به شکلی تغییر می‌دهد که از نظر ظاهری به دامنهٔ هدف نزدیک شوند، در حالی‌که محتوای اصلی تصویر حفظ می‌شود. این ایده به مدل اجازه می‌دهد تا بدون نیاز به برچسب‌های دامنهٔ هدف، عملکرد خود را در آن دامنه نیز بهبود دهد.

برای آزمایش این ایده، از دو مجموعه‌دادهٔ معروف و مشابه استفاده می‌کنیم:

- MNIST شامل تصاویر سیاه‌وسفید ارقام دست‌نویس (دامنهٔ منبع)
- MNIST-M شامل همان ارقام، اما با پس‌زمینه‌های رنگی و تصادفی (دامنهٔ هدف)

در مراحل مختلف تمرین، ابتدا داده‌ها را پیش‌پردازش کرده و مدل پایه را آموزش می‌دهیم تا «شکاف دامنه» بین MNIST و MNIST-M را اندازه بگیریم. سپس با طراحی شبکهٔ Discriminator، Generator، و Classifier، مدل PixelDA را از ابتدا پیاده‌سازی و آموزش می‌دهیم، و در نهایت عملکرد مدل را به‌صورت کمی و کیفی بررسی و تحلیل می‌کنیم.

1-2 بخش نظری

چرا آموزش GAN ناپایدار است؟ سه عامل اصلی ناپایداری را نام ببرید و هرکدام را توضیح دهید. به‌نظر شما چه مکانیسم‌هایی برای مقابله با این ناپایداری پیشنهاد شده‌اند؟

آموزش GAN ها به‌دلیل ویژگی تقابلی (adversarial) بین دو شبکه Generator و Discriminator اغلب ناپایدار است. سه عامل اصلی ناپایداری عبارتند از:

1. **Vanishing/Exploding Gradients:**
   زمانی که یکی از شبکه‌ها، به‌ویژه Discriminator، بسیار قوی‌تر از دیگری می‌شود، شبکه Generator دیگر گرادیان‌های مؤثر برای یادگیری دریافت نمی‌کند.
2. **عدم همترازی در فضای ویژگی‌ها یا Mode Collapse:**
   Generator ممکن است یاد بگیرد فقط چند نمونه خاص از داده هدف را تولید کند (یا حتی فقط یک حالت را تولید کند) و در نتیجه تنوع لازم در خروجی از بین برود.

3. **عدم تعادل در بازی مینی‌ماکس (Unstable Minimax Game):**
تابع هدف GAN یک بازی مینی‌ماکس است که حل آن با دو بهینه‌ساز همزمان دشوار بوده و نوسانات زیادی در فرآیند آموزش ایجاد می‌کند.

مکانیزم‌های پیشنهادی برای رفع ناپایداری:

- استفاده از **Loss های جایگزین** مثل Wasserstein loss (در WGAN) برای فراهم کردن گرادیان‌های پایدارتر.
- استفاده از **تکنیک‌های Regularization** مثل gradient penalty یا spectral normalization.
- استفاده از **ساختارهای شبکه‌ای پایدارتر** مثل batch normalization یا residual connections.
- در مقاله PixelDA، برای پایداری آموزش از دو مکانیزم بهره گرفته شده:
  - استفاده از **task-specific loss** برای کنترل وظیفه و هم‌راستاسازی بهتر.
  - استفاده از **content similarity loss** برای ثابت نگه داشتن محتوای تصویر اصلی و جلوگیری از فروپاشی حالت (mode collapse).

در یک GAN معمولی، Generator فقط نویز z را به تصویر تبدیل می‌کند. در این مقاله، Generator به‌جای آن، از یک تصویر ورودی و نویز استفاده می‌کند. به‌نظر شما این تغییر چه اثری بر یادگیری و کنترل خروجی دارد؟

در یک GAN معمولی، Generator تنها از یک بردار نویز تصادفی z برای تولید تصویر استفاده می‌کند. این کار منجر به تولید داده‌هایی از توزیع هدف می‌شود، ولی هیچ کنترلی روی محتوای خاص تصویر وجود ندارد.

در مقاله Generator هم از یک تصویر ورودی از دامنه مبدأ (source image) استفاده می‌کند و هم از نویز z. اثرات این کار بصورت زیر هستند:

1. **حفظ محتوای تصویر**
چون Generator روی تصویر ورودی شرطی شده، می‌تواند محتوای ساختاری تصویر اصلی را حفظ کند (مثل شکل، فرم، یا عدد)، و فقط ویژگی‌های ظاهری (مثل رنگ، بافت، نور) را مطابق با دامنه هدف تغییر دهد.
2. **کنترل بهتر بر خروجی:**
استفاده از تصویر ورودی باعث می‌شود بتوانیم دقیقاً تعیین کنیم که چه چیزی باید منتقل شود؛ این یعنی به جای تولید تصادفی از نویز، خروجی به‌شدت وابسته به ورودی خاص است.
3. **پایداری بیشتر در آموزش:**
چون Generator با یک تصویر واقعی کار می‌کند، فضای جست‌وجوی پارامترها محدودتر شده و این باعث می‌شود فرآیند آموزش پایدارتر باشد.
4. **افزایش توانایی مدل در تطبیق سبک بدون تغییر محتوا:**
هدف مقاله این است که "سبک" دامنه هدف را روی داده‌های مبدأ منتقل کند بدون آن‌که محتوای آن‌ها تغییر کند، و این ساختار ورودی به Generator به‌خوبی از این هدف پشتیبانی می‌کند.

مدل معرفی‌شده در این مقاله شامل سه مؤلفه اصلی است: Generator، Discriminator و Classifier. به‌نظر شما نقش هر کدام در یادگیری چه چیزی است؟ اگر یکی از آن‌ها حذف شود عملکرد کل مدل چه تغییری می‌کند؟

نقش هر مؤلفه:

1. **Generator:**
   - وظیفه دارد تصاویر دامنه مبدأ را به تصاویری شبیه به دامنه هدف تبدیل کند.
   - ورودی: تصویر از دامنه مبدأ + نویز z.
   - خروجی: تصویر "تقلبی" که ظاهر دامنه هدف را دارد ولی محتوای اصلی را حفظ کرده است.

2. **Discriminator:**
   - تمایز بین تصاویر واقعی از دامنه هدف و تصاویر تولیدشده توسط Generator را یاد می‌گیرد.
   - نقش آن آموزش غیرمستقیم Generator از طریق یک بازی تقابلی است.

3. **Classifier:**
   - روی تصاویر تولیدی از Generator آموزش می‌بیند تا وظیفه نهایی (مثلاً دسته‌بندی عدد یا تخمین موقعیت) را انجام دهد.
   - نقش آن کمک به Generator برای حفظ اطلاعات محتوایی است (مثلاً اطمینان از اینکه عدد «3» تبدیل به «8» نشود).

با

- **حذف Generator:**
  کل مدل بی‌فایده می‌شود چون هیچ تبدیل دامنه‌ای رخ نمی‌دهد.

- **حذف Discriminator:**
  دیگر فشاری برای شباهت خروجی Generator به دامنه هدف وجود ندارد، در نتیجه تصاویر تولیدی ممکن است ظاهر واقع‌گرایانه نداشته باشند.

- **حذف Classifier:**
  آموزش صرفاً بر پایه ظاهر بصری خواهد بود و محتوا ممکن است حفظ نشود (مثلاً شکل عدد یا شیء تغییر کند).
  همچنین آموزش ناپایدارتر شده و مدل ممکن است دچار "mode collapse" شود.

یکی از ویژگی‌های خاص مدل معرفی‌شده در این مقاله، استفاده از content similarity loss (برای حفظ محتوا) است. توضیح دهید چرا حفظ محتوای تصویر منبع ضروری است، و اگر این مکانیزم حذف شود، چه رفتاری از Generator انتظار دارد؟

در مسئله تطبیق دامنه، هدف اصلی این است که تصاویر مبدأ به گونه‌ای به تصاویر دامنه هدف تبدیل شوند که ظاهر آن‌ها تغییر کند، اما محتوای اصلی‌شان ثابت باقی بماند. برای مثال، اگر تصویر ورودی عدد «5» باشد، تصویر خروجی باید عددی شبیه به «5» ولی با سبک (style) دامنه هدف (مثلاً پس‌زمینه رنگی یا نویزدار) باشد.

اگر در این فرآیند، محتوای اصلی از بین برود یا به شکلی غیرقابل شناسایی تغییر کند، آنگاه مدل وظیفه خود را در انتقال یادگیری از مبدأ به هدف به‌درستی انجام نداده است.

برای اطمینان از حفظ محتوای تصویر، مقاله از یک تابع زیان خاص به نام content similarity loss استفاده می‌کند. این تابع فقط بر روی بخش‌های مهم تصویر (مثلاً ناحیه‌ی شیء یا رقم) تمرکز دارد و تفاوت بین تصویر اصلی و تصویر تولیدشده را در آن بخش‌ها اندازه‌گیری می‌کند. این کار باعث می‌شود Generator به تغییر دادن ظاهر تصویر محدود شود، نه تغییر ساختاری محتوای آن.

اگر این مکانیزم حذف شود، Generator به‌دلیل ماهیت تقابلی آموزش (و تأثیر فقط از Discriminator) ممکن است تصاویری تولید کند که ظاهراً به دامنه هدف شباهت دارند، اما محتوای تصویر به‌شدت تغییر یافته یا حتی بی‌ربط شود. در چنین حالتی، ممکن است Generator تمامی ارقام را به ظاهر یکسانی تبدیل کند که برای Discriminator قانع‌کننده است، اما دیگر نمی‌توان از آن‌ها برای آموزش طبقه‌بند استفاده کرد. این اتفاق معمولاً به "mode collapse" منجر می‌شود، جایی که Generator فقط چند نمونه محدود و تکراری تولید می‌کند.

در نتیجه، content similarity loss نه‌تنها برای حفظ اطلاعات محتوایی ضروری است، بلکه به پایداری آموزش نیز کمک می‌کند و اجازه می‌دهد که مدل به‌طور مؤثری هم "واقع‌نمایی" تصویر و هم "معنای" آن را حفظ کند.

مدل معرفی‌شده در این مقاله از یک طبقه‌بند مستقل برای آموزش همزمان روی تصویر اصلی و تصویر تولیدی استفاده می‌کند. این کار چه مزیتی نسبت به آموزش فقط روی تصاویر تولیدی دارد؟ تحلیل کنید چگونه این انتخاب به پایداری یادگیری کمک می‌کند.

در مدل PixelDA، طبقه‌بند (Classifier) نه‌تنها روی تصاویر تولیدشده توسط Generator آموزش می‌بیند، بلکه به‌صورت همزمان روی تصاویر اصلی منبع (یعنی داده‌های اولیه‌ی برچسب‌دار) نیز آموزش داده می‌شود. این طراحی هوشمندانه چند مزیت مهم دارد.

اولاً، در مراحل اولیه آموزش Generator، ممکن است تصاویر تولیدشده هنوز کیفیت مناسبی نداشته باشند یا محتوای صحیح را حفظ نکنند. در چنین شرایطی، اگر طبقه‌بند فقط روی این تصاویر ضعیف آموزش ببیند، یادگیری‌اش ناپایدار و کم‌اثر خواهد بود. با آموزش همزمان روی تصاویر اصلی، طبقه‌بند از داده‌های مطمئن‌تری نیز یاد می‌گیرد، که باعث حفظ کیفیت طبقه‌بندی می‌شود.

ثانیاً، این رویکرد به‌طور غیرمستقیم به پایداری آموزش Generator هم کمک می‌کند. چرا که طبقه‌بند، با یادگیری از هر دو نوع تصویر، به Generator بازخورد دقیق‌تری می‌دهد: اگر خروجی Generator محتوای اصلی را خراب کند، طبقه‌بند قادر به شناسایی آن نخواهد بود و این باعث افزایش loss مربوط به طبقه‌بندی شده و در نتیجه Generator به سمت حفظ محتوای صحیح هدایت می‌شود.

در واقع، آموزش همزمان طبقه‌بند روی تصاویر اصلی و تولیدی، به عنوان یک نوع مکانیزم تنظیم‌کننده (regularization) عمل می‌کند که نه‌تنها کیفیت یادگیری طبقه‌بند را حفظ می‌کند، بلکه به Generator فشار می‌آورد تا خروجی‌هایش را به‌گونه‌ای بسازد که برای طبقه‌بند قابل فهم باشند. این راهکار، از نوسان زیاد در بهینه‌سازی جلوگیری کرده و منجر به پایداری و تکرارپذیری بهتر نتایج می‌شود.

روش این مقاله برای دامنه‌هایی مانند MNIST و MNIST-M طراحی شده که تفاوت آن‌ها در سبک (style) است. آیا می‌توان از همین روش برای دامنه‌هایی که تفاوت semantic دارند (مثلاً اشیای متفاوت، زبان متفاوت، یا زاویه دید متفاوت) استفاده کرد؟ استدلال کنید.

مدل PixelDA برای سناریوهایی طراحی شده که تفاوت بین دامنه منبع و هدف در ویژگی‌های ظاهری یا سبک (style) آن‌هاست، نه در معنای محتوایی (semantics). مثلاً در حالت MNIST به MNIST-M، اعداد همان اعداد هستند ولی پس‌زمینه، رنگ، بافت و نویز متفاوت است. در چنین حالتی، مدل با حفظ محتوای تصویر و تغییر ظاهر آن، به خوبی می‌تواند فرآیند تطبیق دامنه را انجام دهد.

اما اگر بخواهیم همین روش را به دامنه‌هایی با تفاوت معنایی تعمیم دهیم، با چالش‌های جدی مواجه خواهیم شد. برای مثال، فرض کنید منبع شامل تصاویر ماشین باشد و هدف شامل تصاویر انسان‌ها. یا مثلاً زبان مبدأ انگلیسی و زبان هدف چینی باشد. در این موارد، تفاوت بین دامنه‌ها فقط در ظاهر نیست، بلکه در خود مفهوم و محتوای داده‌هاست. Generator دیگر نمی‌تواند با حفظ محتوای اصلی و تغییر ظاهر به نتیجه قابل قبول برسد، چون محتوای منبع و هدف هم‌معنا نیستند.

در این موارد، تطبیق پیکسل‌محور مثل PixelDA کافی نخواهد بود و نیاز به تغییر ساختار مدل و استفاده از روش‌های سطح-ویژگی یا even-level داریم، که قابلیت درک و تطبیق تفاوت‌های مفهومی را داشته باشند.

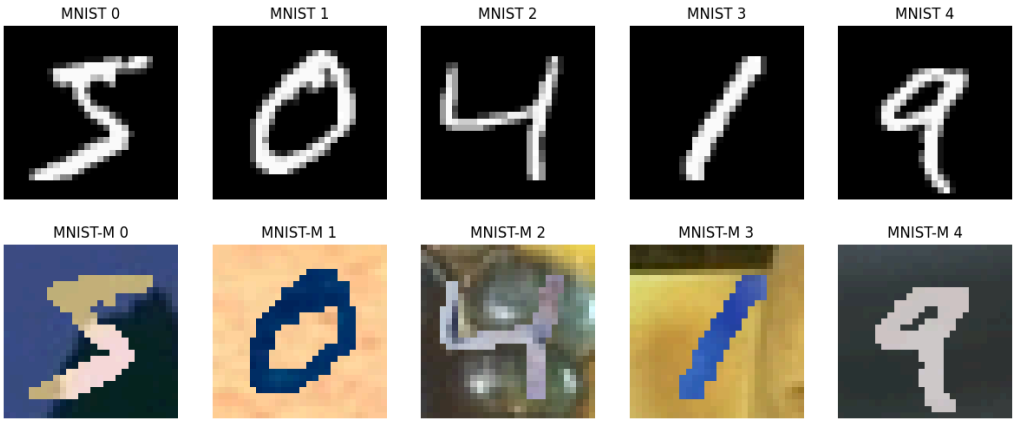
با این حال، اگر تفاوت semantic بین دامنه‌ها جزئی باشد (مثلاً دید متفاوت از یک شیء یا زوایای مختلف از همان صحنه)، و مدل بتواند با regularization و losses کمکی بر حفظ معنای محتوایی نظارت داشته باشد، در این صورت می‌توان گفت PixelDA با اصلاحاتی قابل استفاده است. ولی برای تفاوت‌های بنیادی معنایی، نیاز به روش‌هایی پیچیده‌تر و مبتنی بر تطبیق ساختار ویژگی‌ها خواهیم داشت.

1-3 پیش پردازش

برای درک تفاوت ظاهری میان دامنهٔ منبع (MNIST) و دامنهٔ هدف (MNIST-M)، میتوانید به موارد چاپ شده ی زیر توجه کنید. این مقایسه چشمی نشان می‌دهد که محتوای عددی ثابت است و تنها «سبک» (رنگ و بافت پس‌زمینه) تغییر می‌کند.

1-3-1. تبدیل MNIST تک‌کاناله به تصویر RGB

چون شبکهٔ PixelDA و طبقه‌بند روی تصاویر سه‌کاناله (مانند MNIST-M) کار می‌کند، هر تصویر 28×28×1 خاکستری MNIST را به تصویر 28×28×3 تبدیل کردیم. سه کانال کاملاً هم‌ارزش از 0 تا 255 ایجاد شد.



1-3-2. تغییر اندازه و نرمال‌سازی

مطابق پارامترهای شبکه در مقاله، ورودی باید ۳۲×۳۲ باشد. همهٔ تصاویر هر دو دامنه را به این اندازه رساندیم. پیکسل‌ها از بازهٔ [0,1] به بازهٔ [-1,1] نگاشت شد تا با تابع فعال‌سازی Tanh در Generator سازگار باشد.

1-3-3. تقسیم داده ها به آموزش و تست با اندیس مشترک

لیست اندیس‌های 0 تا N−1 ایجاد شد و به دو زیرمجموعهٔ train و test تجزیه گردید(نسبت 80/20). همان لیست اندیس برای هر دو آرایهٔ تصویر و برچسب MNIST و همچنین برای MNIST-M به کار رفت؛ در نتیجه تصویر i در هر دو دامنه همیشه در یک بخش (train یا test) قرار گرفت و تناظر حفظ شد.

1-3-4. ساخت چهار DataLoader

برای هر دامنه دو DataLoader در نظر گرفته شد (یکی برای آموزش، یکی برای آزمون). در DataLoader های آموزش و آزمون از shuffle=True استفاده کردیم؛ اندازهٔ batch برابر 64 انتخاب شد تا حافظهٔ GPU و سرعت یادگیری در تعادل باشد.

به این ترتیب چهار جریان دادهٔ مستقل اما متناظر ایجاد شد که در مراحل بعدیِ آموزش شبکهٔ PixelDA مورد استفاده قرار می‌گیرند.

1-4 ارزیابی مدل پایه و مشاهده‌ی شکاف دامنه (Domain Gap)

در این مرحله یک طبقه‌بند (Classifier) با معماری دقیق مقاله PixelDA پیاده‌سازی و تنها با استفاده از داده‌های MNIST آموزش داده شد. این مدل سپس مستقیماً روی داده‌های آزمون MNIST و همچنین مجموعه‌ی کامل MNIST-M ارزیابی شد تا شکاف عملکرد آن بین دو دامنه بررسی شود.

1-4-1. معماری طبقه‌بند

طبقه‌بند مورد استفاده، ساختاری مشابه با معماری ارائه‌شده در مقاله دارد که شامل دو لایه‌ی کانولوشن با فیلترهای 5×5، دو لایه‌ی تمام‌متصل (fully connected) و تابع فعال‌سازی ReLU است. این شبکه از روی تصاویر RGB با ابعاد 3×32×32 آموزش می‌بیند و در انتها یک خروجی 10 کلاسه برای پیش‌بینی رقم ارائه می‌دهد. این مدل بدون استفاده از Dropout یا BatchNorm طراحی شده تا با ساختار مقاله مطابقت داشته باشد.

1-4-2. آموزش روی MNIST

مدل تنها با داده‌های آموزش MNIST (به‌عنوان دامنه‌ی منبع) آموزش داده شد. برای این مرحله، از برچسب‌های MNIST-M هیچ استفاده‌ای نشد. فرایند آموزش شامل 10 دوره (epoch) با تابع خطا Cross-Entropy و بهینه‌ساز Adam بود.

پروسه آموزش این مدل هم به صورت روبرو انجامیده است:

```
Epoch [1/10], Loss: 1.5700
Epoch [2/10], Loss: 1.5042
Epoch [3/10], Loss: 1.4971
Epoch [4/10], Loss: 1.4925
Epoch [5/10], Loss: 1.4902
Epoch [6/10], Loss: 1.4898
Epoch [7/10], Loss: 1.4879
Epoch [8/10], Loss: 1.4869
Epoch [9/10], Loss: 1.4882
Epoch [10/10], Loss: 1.4858
```

1-4-3. ارزیابی روی MNIST و MNIST-M

- دقت روی داده‌های تست MNIST بسیار بالا (نزدیک به 98%) است، که نشان‌دهنده عملکرد عالی مدل روی داده‌هایی است که با مجموعه آموزشی آن (MNIST) هم‌توزیع هستند. این امر به این دلیل است که مدل روی تصاویر تک‌کاناله و ساده MNIST آموزش دیده و داده‌های تست نیز از همان توزیع پیروی می‌کنند.

- در مقابل، دقت روی داده‌های MNIST-M به‌طور قابل‌توجهی پایین‌تر (حدود 62%) است. این اختلاف به دلیل وجود تفاوت دامنه (domain shift) بین دو مجموعه داده است. MNIST شامل تصاویر تک‌کاناله (grayscale) با پس‌زمینه یکنواخت است، در حالی که MNIST-M تصاویر رنگی با پس‌زمینه‌های متنوع و پیچیده دارد. مدل آموزش‌دیده روی MNIST به دلیل عدم تطبیق با این تغییرات بصری (مانند رنگ، روشنایی و بافت) نمی‌تواند به‌طور مؤثر به داده‌های MNIST-M تعمیم دهد. این موضوع اهمیت استفاده از تکنیک‌های تطبیق دامنه (مانند روش PixelDA در مقاله) را برای بهبود عملکرد در چنین سناریوهایی برجسته می‌کند.

1-4-4. تحلیل شکاف دامنه

تفاوت معنادار در دقت مدل بین دو دامنه بیانگر شکاف دامنه (Domain Gap) است. این شکاف زمانی به‌وجود می‌آید که مدل روی توزیعی آموزش دیده که از توزیع آزمون متفاوت است — حتی اگر برچسب‌ها و مفاهیم در هر دو یکسان باشند. در این تمرین، تغییر ظاهر تصاویر (رنگ، پس‌زمینه، الگوهای تصویری) باعث شده مدل نتواند تعمیم مناسبی روی دامنه‌ی هدف بدهد.

این ارزیابی پایه برای مرحله‌ی بعدی تمرین بسیار مهم است، چرا که هدف روش PixelDA دقیقاً کاهش همین شکاف دامنه است. در ادامه خواهیم دید که چگونه استفاده از یک Generator برای انتقال سبک تصویری می‌تواند این مشکل را تا حد زیادی کاهش دهد.

1-5. پیاده‌سازی معماری مدل

در این بخش، سه شبکهٔ اصلی مورد نیاز برای پیاده‌سازی روش PixelDA طراحی و پیاده‌سازی شدند: یک Generator برای تبدیل تصاویر دامنهٔ منبع به ظاهر دامنهٔ هدف، یک Discriminator برای تشخیص تصاویر واقعی از جعلی، و یک Classifier برای پیش‌بینی برچسب رقمی هر تصویر.

1-5-1. Generator

وظیفه دارد ظاهر تصاویر MNIST را به ظاهر MNIST-M نزدیک کند، بدون آنکه محتوای عددی تصویر تغییر کند. این شبکه ابتدا با ترکیب تصویر ورودی (با اندازهٔ 32×32 و 3 کانال) با بردار نویز z، یک نگاشت اولیه به فضای ویژگی انجام می‌دهد. سپس این ویژگی‌ها از میان چندین بلوک Residual عبور می‌کنند. در این طراحی، از 6 بلوک residual استفاده کردیم که هر کدام شامل دو لایهٔ convolution و نرمال‌سازی BatchNorm هستند. در انتها، یک لایهٔ convolution با خروجی سه‌کاناله و تابع فعال‌سازی tanh قرار دارد تا تصویر خروجی در بازهٔ [-1, 1] قرار گیرد. استفاده از بلوک‌های باقیمانده در این ساختار به حفظ بهتر محتوای تصویر کمک می‌کند و اجازه می‌دهد تغییرات فقط در «سبک» تصویر اعمال شود، نه در ساختار اصلی رقم.

1-5-2. Discriminator

شبکه‌ای است که تفاوت بین تصاویر واقعی دامنهٔ هدف (MNIST-M) و تصاویر جعلی تولیدشده توسط Generator را تشخیص می‌دهد. این شبکه شامل چندین لایهٔ convolution با فیلترهایی به اندازهٔ 3×3، و stride متناوب 1 و 2 است. در این طراحی، بعد از هر لایه، از تابع LeakyReLU برای فعال‌سازی استفاده شده و نرمال‌سازی BatchNorm همراه با Dropout به کار رفته تا از overfitting جلوگیری شود. در نهایت، خروجی شبکه از طریق یک لایهٔ خطی و تابع sigmoid به یک مقدار احتمال تبدیل می‌شود که نشان‌دهندهٔ میزان «واقعی بودن» تصویر ورودی است.

1-5-3. Classifier

طراحی‌شده دقیقاً همان معماری‌ای است که در مقاله برای طبقه‌بندی ارقام دست‌نویس استفاده شده است. این شبکه شامل دو لایهٔ convolution با فیلترهای 5×5 است که هرکدام با تابع ReLU و لایهٔ pooling همراه هستند. سپس داده‌ها تخت (Flatten) شده و از دو لایهٔ تمام‌متصل عبور داده می‌شوند که هرکدام 100 واحد دارند و در نهایت به یک لایهٔ خروجی با 10 کلاس ختم می‌شوند. این ساختار نسبتاً ساده، اما مؤثر است و برای وظیفهٔ تشخیص رقم در دو دامنهٔ MNIST و MNIST-M بسیار مناسب است.

1-5-4. توابع خطا و ضرایب ترکیب

برای آموزش مؤثر این سه شبکه به‌صورت همزمان، از دو نوع تابع خطا استفاده شده است.

نخست، هزینه رقابتی (adversarial loss) که بین Generator و Discriminator برقرار می‌شود. هدف Generator فریب دادن Discriminator و تولید تصاویری است که غیرقابل تشخیص از تصاویر واقعی MNIST-M باشند. در مقابل، Discriminator تلاش می‌کند این تصاویر جعلی را از واقعی تشخیص دهد. این رقابت با استفاده از تابع خطای binary cross-entropy مدل‌سازی شده است.

دوم، زیان طبقه‌بندی (task loss) است که عملکرد Classifier را در تشخیص رقم ارزیابی می‌کند. این زیان شامل دو بخش است: یکی مربوط به تصاویر اصلی MNIST و دیگری مربوط به تصاویر فیک تولید شده توسط

Generator. در هر دو حالت، از تابع CrossEntropy برای مقایسهٔ پیش‌بینی‌ها با برچسب‌های صحیح استفاده شده است.

در نهایت، برای آموزش Generator از ترکیب این دو تابع هزینه استفاده شده است:
مقدار کلی loss برابر است با ترکیب خطی دو جزء بالا به‌صورت

$$LOSS_G = \alpha \,.\, LOSS_{adv} + \beta \,.\, LOSS_{cls}$$

که ضرایب α=0.011 و β=0.01 طبق مقاله با این مقادیر انتخاب شده اند.

1-6. آموزش مدل

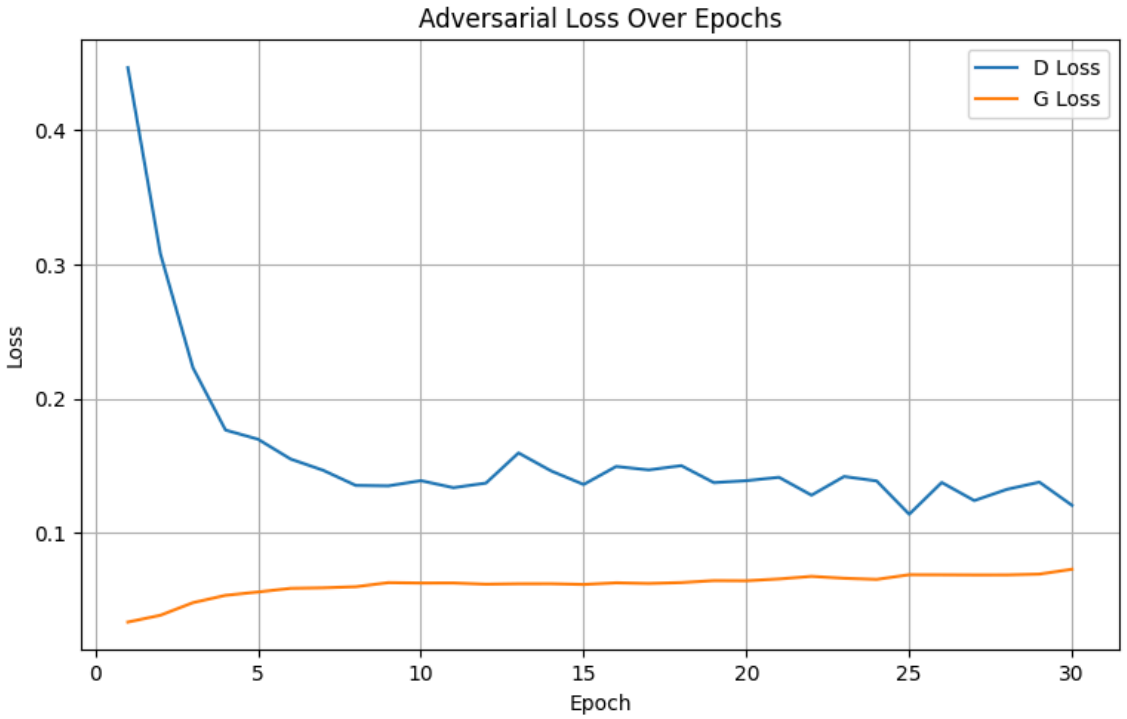
در فرآیند آموزش مدل، هر سه مؤلفه‌ی اصلی یعنی Generator، Discriminator و Classifier به‌صورت هماهنگ اما با اهداف متفاوت به‌روزرسانی شدند. در هر تکرار از حلقهٔ آموزش:

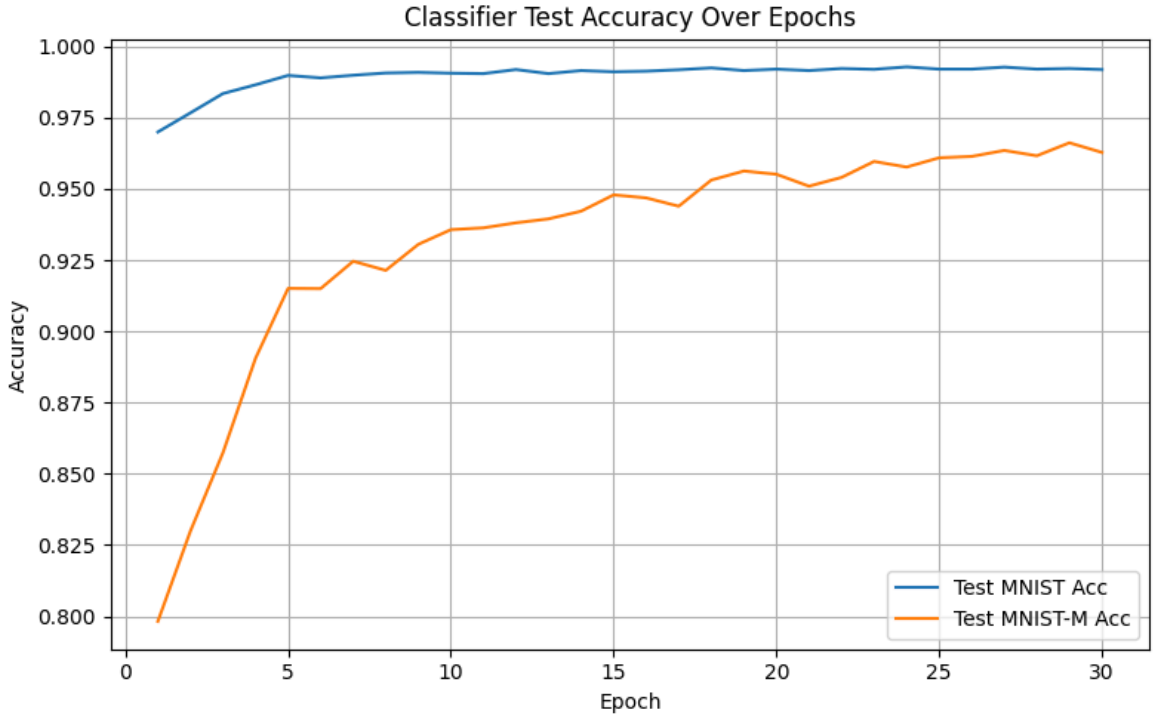
- **Generator** با دریافت تصویر از دامنهٔ MNIST به همراه نویزی تصادفی، یک تصویر فیک با ظاهر شبیه به MNIST-M تولید می‌کند. هدف Generator این است که تصاویر تولیدی آن توسط Discriminator به‌عنوان واقعی شناسایی شوند و همچنین توسط Classifier به‌درستی طبقه‌بندی گردند.

- **Discriminator** به‌طور مستقل آموزش می‌بیند تا بتواند تصاویر واقعی MNIST-M را از تصاویر تولید شده توسط Generator تشخیص دهد. ورودی‌های آن شامل تصاویر واقعی از دامنهٔ هدف و تصاویر فیک تولید شده از دامنهٔ منبع هستند.

- **Classifier** صرفاً روی داده‌های دامنهٔ منبع (MNIST) آموزش داده می‌شود. این آموزش هم شامل تصاویر واقعی MNIST و هم شامل تصاویر فیکی است که از همان داده‌ها توسط Generator تولید شده‌اند. در هیچ مرحله‌ای از آموزش از برچسب‌های دامنهٔ هدف استفاده نشده و آموزش کاملاً بدون نظارت روی target صورت گرفته است.

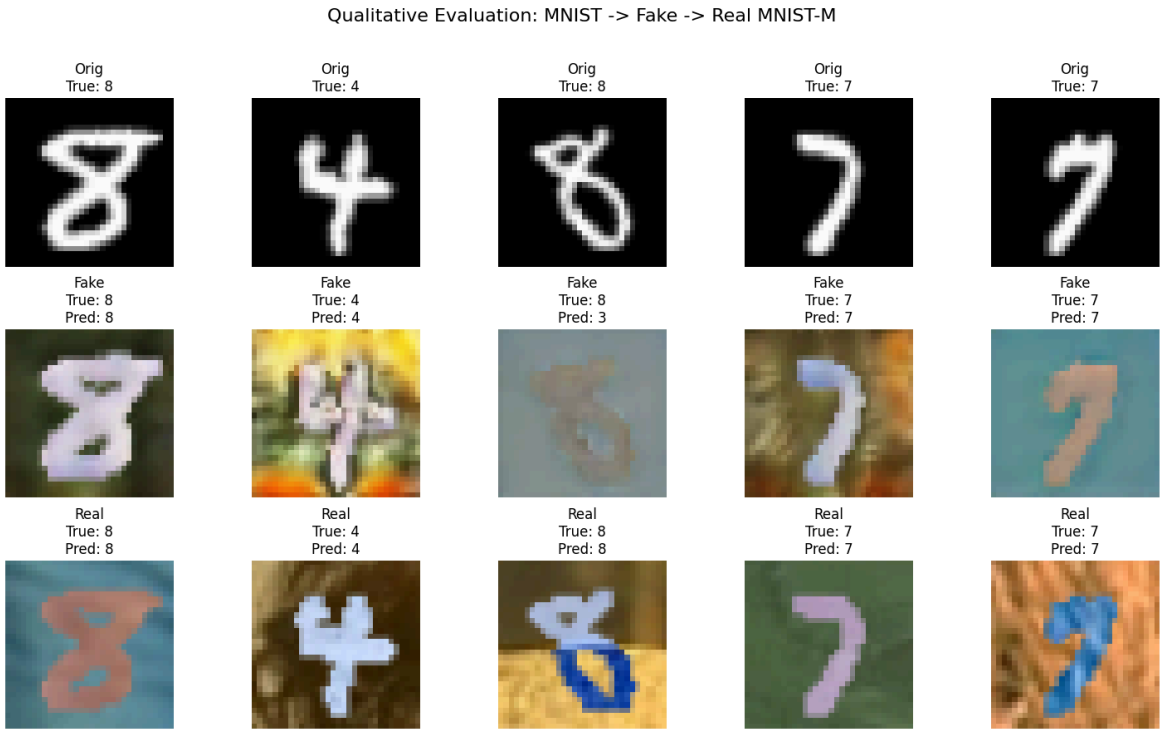
تصویر مربوط به نمودار های دقت و هزینه در طول آموزش را میتوانید مشاهده کنید که بصورت نرمال انجام شده است:





1-7. تحلیل نهایی

در پایان 30 دورهٔ آموزش، سیر تغییر زیان‌ها و دقت‌ها به‌همراه چند نمونهٔ کیفی در دو شکل زیر نمایش داده شده است. نمودار بالا روند رقابت Generator و Discriminator را نشان می‌دهد و نمودار پایین تغییر دقّت طبقه‌بند را روی دو دامنه رصد می‌کند. شکل دوم نیز سه ردیف متناظر از پنج نمونهٔ آزمون را به‌صورت «تصویر منبع»، «خروجی Generator» و «تصویر حقیقی MNIST-M» نمایش می‌دهد.



هر ستون یک نمونهٔ آزمون را نشان می‌دهد:

1. **ردیف اول – تصویر MNIST اصلی**: رقم سفید روی پس‌زمینهٔ سیاه.

2. **ردیف دوم – خروجی Generator**: همان رقم با پس‌زمینه‌ای شبیه سبک MNIST-M؛ در بیشتر نمونه‌ها رنگ رقم و بافت پس‌زمینه طبیعی به‌نظر می‌رسد. برچسب واقعی (True) و پیش‌بینی Classifier (Pred) درج شده است.

3. **ردیف سوم – تصویر MNIST-M واقعی**: همان اندیس در دامنهٔ هدف به‌همراه پیش‌بینی مدل.

- در 4 مورد از 5 نمونه، Classifier روی تصویر فیک و تصویر واقعی هر دو برچسب صحیح را تشخیص داده است؛ بیانگر آنکه Generator محتوای عددی را حفظ کرده است.
- در ستون سوم، Generator یک «۸» را تولید کرده ولی رقم در تصویر MNIST-M واقعی محو و در نتیجه پیش‌بینی مدل روی آن تصویر اشتباه شده است؛ این نمونه نقاط ضعفِ هر دو مؤلفه را به‌خوبی نشان می‌دهد.

- از نظر بصری، پس‌زمینه‌های تولیدی تنوع رنگ و بافت قابل قبولی دارند و نسبت به نسخهٔ اولیه (MNIST سیاه-سفید) کاملاً به سبک دادهٔ هدف نزدیک شده‌اند.

- **کاهش شکاف دامنه**: دقت روی MNIST-M از 80 درصد (مدل پایه) به 96 درصد رسید؛ یعنی حدود 80 درصد از شکاف اولیه برطرف شد.
- **پایداری آموزش**: منحنی‌های زیان بدون نوسان شدید و بدون فروپاشی مُد بوده است، که نشان می‌دهد ترکیب loss-ها و انتخاب ضرایب α و β موفق بوده‌اند.
- **حفظ محتوا**: در تصاویر فیک، شکل رقم بسیار شبیه تصویر منبع باقی مانده و تنها سبک بصری تغییر کرده است، نشانهٔ خوبی از عملکرد Correctness-Preserving Generator.

این نتایج نشان می‌دهد روش PixelDA – با وجود سادگی نسبی در معماری – به‌خوبی می‌تواند مشکل ناسازگاری سبک بین دو دامنه را رفع کند و یک طبقه‌بند واحد را قادر سازد در هر دو فضای منبع و هدف با دقت بسیار بالا عمل کند.

پرسش 2: بازسازی تصاویر پولیپ آندوسکوپی با EndoVAE

2.0- مقدمه

در این تمرین، هدف ما پیاده‌سازی مدل EndoVAE بر اساس مقاله ذکر شده است. این مدل از ساختار Variational Autoencoder برای تولید تصاویر endoscopic مربوط به دستگاه گوارش استفاده می‌کند. برخلاف روش‌های مبتنی بر GAN که معمولاً پیچیدگی آموزشی بالایی دارند، در این مقاله تلاش شده با استفاده از VAE یک ساختار ساده‌تر، پایدارتر و قابل‌کنترل‌تر برای تولید تصاویر مصنوعی طراحی شود. در این پروژه، مدل EndoVAE طبق معماری معرفی‌شده در مقاله پیاده‌سازی شده، صرفاً با تصاویر نرمال آموزش داده می‌شود، و عملکرد آن در بازسازی تصاویر پولیپ با استفاده از معیارهای PSNR و SSIM مورد ارزیابی قرار می‌گیرد. در این مسیر ابهامات زیادی برای من وجود داشت که به حل تمرین درس ایمیل ارسال کردم و در هر بخش ذکر خواهم کرد.

2.1- پیش پردازش داده ها

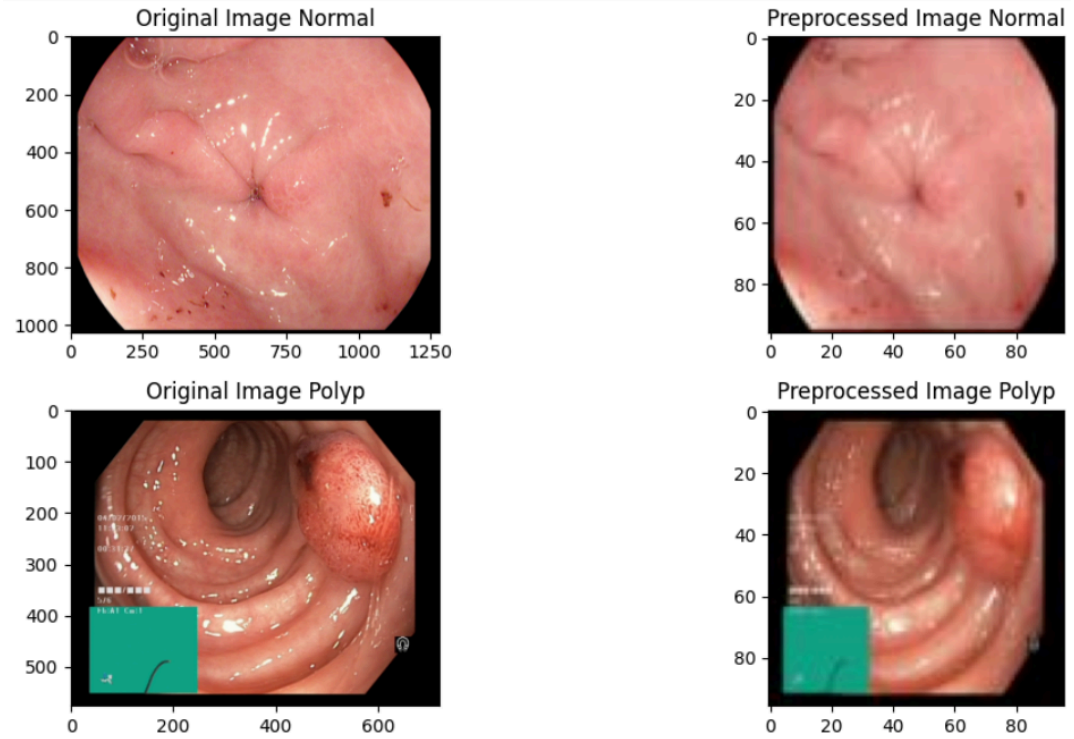
در مرحله‌ی نخست، دیتاست مورد نیاز را دانلود کردیم. این دیتاست شامل سه پوشه‌ی مجزا از تصاویر نرمال و یک پوشه‌ی حاوی تصاویر پولیپ است. هر پوشه شامل تعدادی تصویر JPEG از نواحی مختلف دستگاه گوارش در قالب داده‌های گرفته‌شده از دوربین اندوسکوپی است. برای انجام پردازش‌ها به صورت یکنواخت، تمامی تصاویر از این پوشه‌ها بارگذاری شده و مورد پیش‌پردازش قرار گرفتند.

در گام اول پیش‌پردازش، تمامی تصاویر به ابعاد ثابت 96*96 پیکسل تغییر اندازه داده شدند. انتخاب این اندازه مطابق معماری پیشنهاد شده در مقاله‌ی EndoVAE انجام شد، زیرا در معماری مدل، انتظار می‌رود ورودی شبکه دارای ابعاد ثابت و مشخص باشد تا لایه‌های کانولوشن و کانولوشن معکوس بتوانند بدون خطا عملیات خود را انجام دهند. سپس تصاویر به فرمت عددی تبدیل شده و مقادیر پیکسل‌ها به بازه‌ی 0 تا 1 نرمال‌سازی شدند. این نرمال‌سازی با استفاده از numpy و تقسیم بر 255 انجام شد. این مرحله بسیار ضروری است، زیرا خروجی نهایی مدل از تابع فعال‌سازی Sigmoid استفاده می‌کند که خروجی آن نیز در همین بازه قرار دارد. همچنین در محاسبه‌ی تابع خطا مانند Binary Cross Entropy یا MSE، استفاده از داده‌های نرمال‌شده منجر به پایداری و همگرایی بهتر مدل می‌شود.

در مرحله‌ی بعد، برای تسهیل بارگذاری داده‌ها در فرآیند آموزش و تست، تصاویر پیش‌پردازش‌شده به‌صورت ساختار یافته در پوشه‌ای جدید به نام processed ذخیره شدند. ساختار پوشه‌ی نهایی بدین صورت است که دو زیر پوشه به نام های normal و polyp دارد که در هر کدام یک زیر پوشه با نام imgs است. علت وجود پوشه imgs این است که بعدا دیتا لودر به ارور وجود نداشتن لیبل برخورد نکن و بتواند به درستی محتوای پوشه رو لود کند. در مقاله اصلی نامی از دیگر روش های پیش پردازش برده نشده بود و من نیز به تنها همین موارد بسنده کردم.

در زیر مقایسه ای از تصاویر قبل و بعد از پیش پردازش میبینیم:

تصویر 2.1- تصاویر اصلی و پیش پردازش شده از دو نمونه تصادفی از دو کلاس نرمال و پولیپ



همانطور که دیده میشود کیفیت تصاویر اندکی کاهش پیدا کرده است و همچنین شکل تصاویر به دلیل اینکه به 96*96 تبدیل شده است و در ابتدا به صورت مربعی نبوده است اندکی دچار تغییرات جزئی شده است.

2.2- طراحی معماری EndoVAE

در این بخش، معماری مدل VAE مطابق با بخش 3 مقاله و همچنین شکل 1 پیاده‌سازی شده است. این مدل شامل سه بخش اصلی است: Encoder، تابع reparameterization، و Decoder. همچنین ساختار دقیق لایه‌ها، اندازه‌های میانی و ابعاد خروجی‌ها مطابق با مقاله است.
این نکته قابل توجه است که میان صورت تمرین و طراحی پیاده سازی شده در مقاله تفاوتی وجود داشت که پس از پرس و جو کردن از حل تمرین درس، تصمیم بر آن شد که مطابق مقاله اصلی پیش بروم. ابعادی که در صورت تمرین برای خروجی انکودر گفته شده بود به صورت 6*6 بود در حالی که در مقاله اصلی خروجی انکودر 12*12 است. بنابراین من نیز مطابق با مقاله از خروجی 12*12 استفاده کردم.

بخش انکودر وظیفه استخراج ویژگی‌های سطح بالا از تصویر ورودی را بر عهده دارد. معماری آن شامل 6 لایه‌ی کانولوشن به همراه تابع فعال‌سازی ReLU است. خروجی نهایی انکودر یک تنسور با اندازه‌ی 256*12*12 است که ابتدا flatten شده و سپس با یک لایه‌ی fully connected به برداری با طول 256 نگاشته می‌شود. در ادامه، دو لایه‌ی مجزای FC خروجی این بردار را به دو بردار با طول 6 تبدیل می‌کنند. این دو بردار میانگین و لاگ واریانس توزیع نرمال هستند.

مطابق تعریف مقاله، از ترفند reparameterization برای نمونه‌گیری از فضای نهفته استفاده شده است تا امکان انجام backpropagation از طریق نمونه‌گیری فراهم شود. این تابع طبق فرمول بیان شده در مقاله اصلی پیاده سازی شده است و از اپسیلون رندوم نیز استفاده شده است

بخش دیکودر وظیفه بازسازی تصویر از بخش قبل را دارد. این بخش ابتدا بردار z را با یک لایه‌ی FC به یک تنسور با ابعاد 256*12*12 گسترش می‌دهد و سپس با استفاده از 7 لایه‌ی کانولوشن معکوس و کانولوشن معمولی تصویر نهایی را بازسازی می‌کند.

در این بخش تمامی stride ها، padding ها، تعداد کانال‌ها و... با مقاله هماهنگ شده‌اند. همچنین از ساختار VAE به‌طور کامل پیروی شده است. این طراحی باعث می‌شود که مدل قادر باشد توزیع داده‌های سالم را یاد بگیرد و تصاویر جدیدی از آن تولید کرده یا تصاویر ورودی را بازسازی نماید. در ادامه، از این مدل برای انجام بازسازی تصاویر پولیپ و ارزیابی آن با معیارهای کمی استفاده شده است.

2.3- تعریف توابع هزینه

همان‌طور که در بخش 3 مقاله توضیح داده شده، دو بخش اصلی در تابع هزینه این مدل شامل تابع Reconstruction Loss و تابع KL Divergence است که در نهایت با ضریبی از هر دوی آنها استفاده میشود و loss نهایی بدست می آید.

تابع Reconstruction loss میزان اختلاف بین تصویر اصلی و تصویر بازسازی‌شده توسط دیکودر را اندازه‌گیری می‌کند. در مقاله به‌صورت صریح اشاره شده که از تابع BCE استفاده شده است. در صورت سوال ذکر شده که میتوان بجای آن از MSE نیز استفاده کرد که پس از ترین کردن مدل با هر دو این موارد با تعداد اپیاک کمتر و محدود و بررسی دقت نهایی به عملکرد بهتر MSE پی بردم و توانست تصاویر را قابل فهم تر بازسازی کند. این تابع معمولا در مواردی که نویز زیادی وجود دارد دقت بهتری ارائه میدهد.

تابع KL Divergence وظیفه دارد تا توزیع نهفته تولیدشده توسط انکودر را به توزیع نرمال استاندارد نزدیک کند. این کار با محاسبه‌ی فاصله‌ی Kullback–Leibler بین توزیع پیش‌بینی‌شده و توزیع هدف انجام می‌شود. این بخش تضمین می‌کند که فضای نهفته مدل ساختار مناسبی برای نمونه‌گیری داشته باشد و در حالت تولید تصاویر بتوان بردارهای z را مستقیماً از توزیع نرمال استاندارد تولید کرد که در ادامه همین کار را خواهیم کرد.

در نهایت از هر دو این توابع استفاده میکنیم و تابع نهایی مدل را از مجموع دو مؤلفه‌ی بازسازی و KL تشکیل میدهیم. همچنین یک ضریب بتا برای تنظیم وزن KL در نظر گرفته شده که در حالت پیش‌فرض برابر با 1 است، مشابه مقاله. این تابع به صورت زیر تعریف شده است:

$$total\_loss = reconstruction\_loss + \beta \times KL\_divergence$$

به طور خلاصه، برای پیاده‌سازی توابع هزینه این مدل هم از ساختار پیشنهادی مقاله پیروی شده و هم از نظر تجربی استفاده از MSE به جای BCE آزمایش شده است. در نهایی‌ترین نسخه کد، تابع MSE به عنوان معیار بازسازی انتخاب شد تا بازسازی‌های بصری شفاف‌تری حاصل شود. از سوی دیگر، KL divergence طبق فرمول استاندارد پیاده‌سازی شده تا تطابق با توزیع نرمال تضمین شود.

2.4- روند آموزش مدل

مطابق با تنظیمات پیشنهادی در مقاله، پارامترهای زیر برای آموزش مدل استفاده شدند:

batch size: برابر با 128

learning rate: برابر با 0.001

تعداد اپیاک: در مقاله 5000 ذکر شده بود که به دلیل محدودیت زمان اجرا در google colab تنها توانستم 1000 اپیاک ترین کنم. (برای 5000 اپیاک بیش از 5 ساعت طول میکشید.)

optimizer: الگوریتم Adam از کتابخانه‌ی PyTorch

در صورت تمرین خواسته شده که تنها با داده های نرمال آموزش انجام شود که علاوه بر خواسته صورت سوال من نیز یکبار دیگر با کل داده ها (شامل داده های نرمال و پولیپ) نیز آموزش را انجام دادم و در قسمت نتایج مقایسه ای بین آنها نیز انجام خواهم داد.
مطابق با آن‌چه در مقاله نیز پیشنهاد شده بود. داده‌ها بارگذاری شدند و با استفاده از کلاس ImageFolder در قالب Dataset تعریف شدند. در ساخت دیتال لودر از shuffle=True استفاده شد تا ترتیب تصاویر در هر epoch متفاوت بوده و تعمیم‌پذیری مدل افزایش یابد.

فرآیند آموزش در هر epoch به‌صورت دقیق مطابق با ساختار مقاله انجام شده است. با استفاده از ()model.train، حالت آموزش فعال شده است. سپس تصاویر ورودی ابتدا توسط انکودر به بردارهای تعریف شده در مدل نگاشته می‌شوند و بعد با استفاده از تابع reparameterize یک بردار z از فضای نمونه‌برداری می‌شود و از طریق دکودر تصویر بازسازی‌شده به‌دست می‌آید.
در هر بار عبور، دو تابع هزینه Reconstruction Loss و KL Divergence محاسبه شده و مجموع آن‌ها به‌عنوان total_loss تعریف می‌شود.
در نهایت با استفاده از ()loss.backward و ()optimizer.step عملیات backpropagation و به‌روزرسانی وزن‌ها انجام میشود.
همچنین طبق گفته صورت سوال هر 100 ایپاک مقدار total loss را پرینت کردیم که نتایج به صورت زیر است:

تصویر 2.2- total loss در فرایند آموزش مدل تنها با داده های نرمال

```
Epoch 0 ---> total loss: 2913987.1875
Epoch 100 ---> total loss: 233903.7919921875
Epoch 200 ---> total loss: 179231.3056640625
Epoch 300 ---> total loss: 141307.0634765625
Epoch 400 ---> total loss: 162902.2734375
Epoch 500 ---> total loss: 121069.4775390625
Epoch 600 ---> total loss: 108968.580078125
Epoch 700 ---> total loss: 97723.49365234375
Epoch 800 ---> total loss: 95165.19921875
Epoch 900 ---> total loss: 108716.47607421875
```

همانطور که مشخص است مقدار هزینه کاهشی بوده است و تقریبا به مقدار حدی مشخصی پس از 1000 ایپاک رسیده است که بدین معنی است مدل به درستی آموزش دیده است.

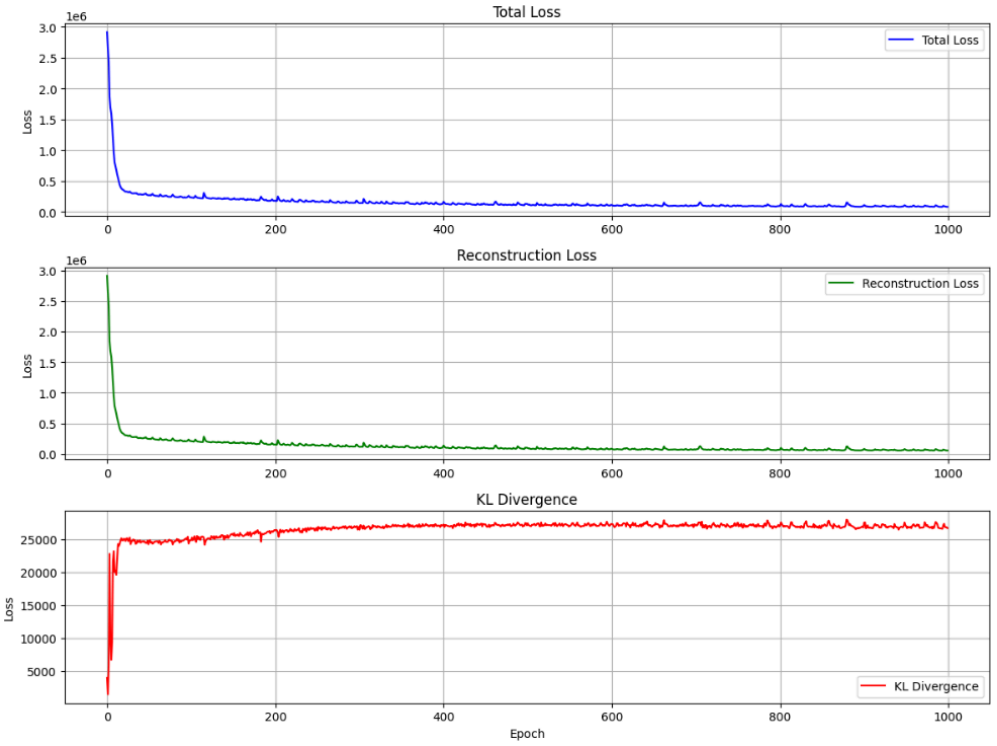
همانطور که در بالا بیان شد، یک مرتبه دیگر مدل را با استفاده از هر دو داده های نرمال و پولیپ به صورت همزمان آموزش دادیم که مقادیر هزینه آن به صورت زیر است:

تصویر 2.3- total loss در فرایند آموزش مدل با هر دو داده های نرمال و پولیپ

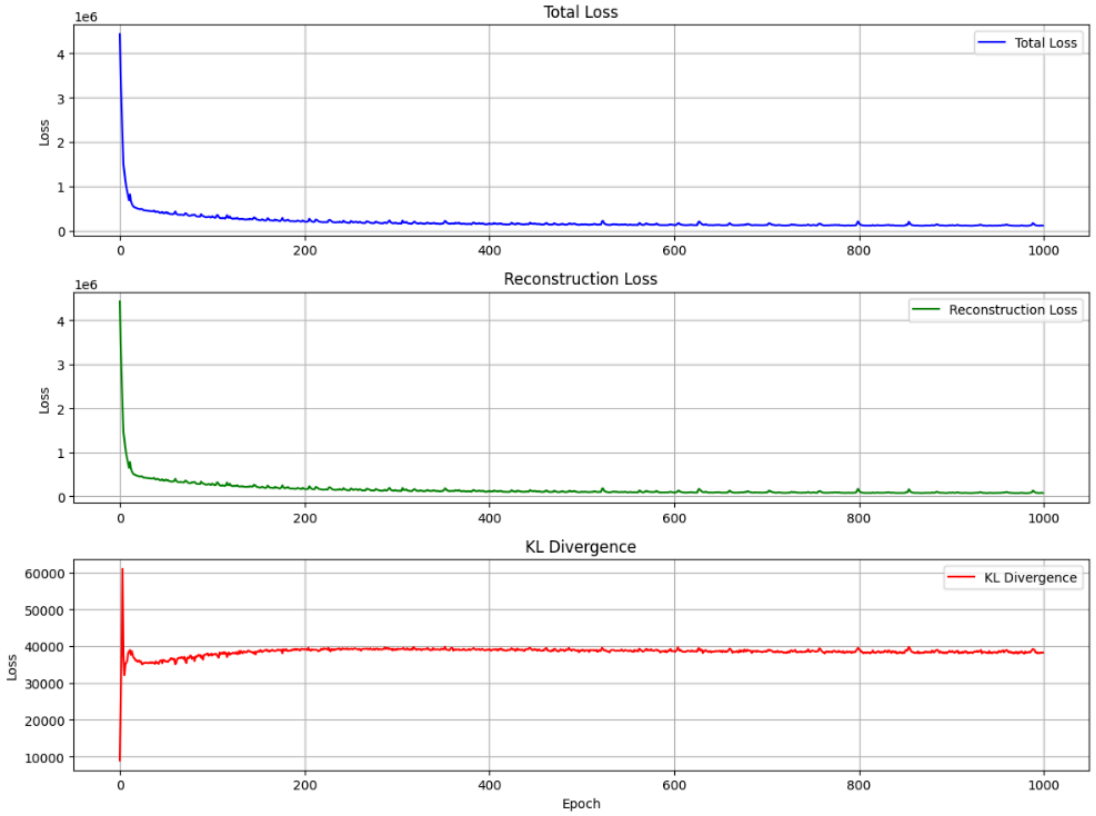
```
Epoch 0 ---> total loss: 4429362.5625
Epoch 100 ---> total loss: 322723.8720703125
Epoch 200 ---> total loss: 204756.17578125
Epoch 300 ---> total loss: 164059.2021484375
Epoch 400 ---> total loss: 161754.30078125
Epoch 500 ---> total loss: 143676.86767578125
Epoch 600 ---> total loss: 125111.01513671875
Epoch 700 ---> total loss: 124520.12060546875
Epoch 800 ---> total loss: 165252.79345703125
Epoch 900 ---> total loss: 119245.96630859375
```

در تصویر مشخص است که در هر دو موارد تقریبا هزینه ها یکسان و شبیه به هم هستند. نمودار loss و divergence نیز به پس از اتمام برای هر دو مدل به صورت زیر است:

نمودار 2.4- مقادیر تابع هزینه در طول فرایند آموزش تنها با داده های نرمال



نمودار 2.4- مقادیر تابع هزینه در طول فرایند آموزش با داده های نرمال و پولیپ به صورت همزمان



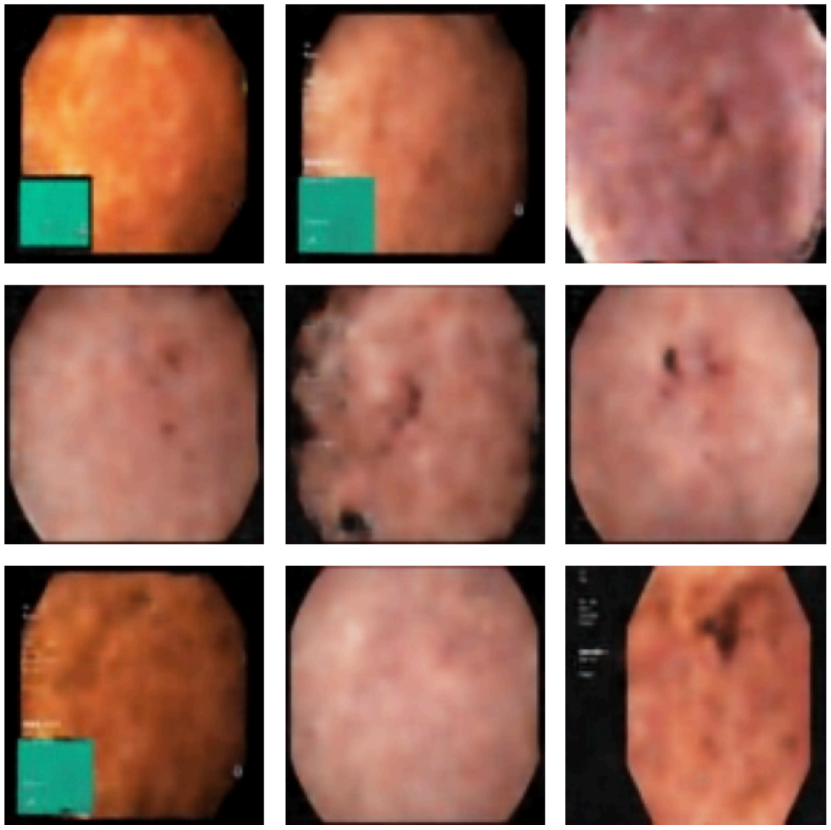
در هر دو این نمودار ها که با تقریب بسیار خوبی شبیه به هم هستند مدل تقریبا به همگرایی در هزینه ها پس از 1000 ایپاک رسیده است و تفاوت فاحشی در فرایند آموزش میان هر دو مدل وجود ندارد.

2.5- تولید و بازسازی کیفی

2.5.1- تولید

یکی از قابلیت‌های اصلی مدل‌های VAE، امکان نمونه‌گیری از فضای نهفته و تولید تصاویر جدید است به‌گونه‌ای که این تصاویر شباهت ساختاری به داده‌های آموزشی داشته باشند. در این بخش از تمرین، مطابق بند 4 مقاله، فرآیند تولید تصویر از طریق نمونه‌گیری تصادفی از فضای نهفته انجام میدهیم. ابتدا مدل آموزش‌دیده را در حالت ارزیابی قرار میدهیم و سپس 9 بردار تصادفی از توزیع نرمال گفته شده با ابعاد برابر با فضای نهفته (latent_dim = 6) تولید شد. این بردارها نمایانگر نقاطی در فضای ویژگی نهفته‌ای هستند که مدل در طول آموزش آن را آموخته است. در گام بعد، با عبور هر یک از بردارها از دیکودر مدل، یک تصویر بازسازی‌شده با اندازه‌ی 96*96*3 تولید شد. برای جلوگیری از دخالت گرادیان در این مرحله، از torch.no_grad() استفاده شد. در نهایت، 9 تصویر تولیدشده نمایش داده شد که خروجی آن به صورت زیر است:

تصویر 2.5- نمونه های تولید شده توسط مدل EndoVAE



همان‌طور که در مقاله نیز مطرح شده است، VAE قادر است با یادگیری ساختار داده‌های سالم، تصاویری تولید کند که از لحاظ ساختار بصری شباهت زیادی به نمونه‌های واقعی دارند. بررسی تصاویر تولیدشده نشان داد که این مدل توانسته است با توجه به آموزش صرفاً بر روی داده‌های نرمال، تصاویری منسجم، تقریبا طبیعی و از لحاظ بافتی مشابه تصاویر واقعی تولید کند اما همانطور که در تصویر مشخص است کیفیت تصاویر کم است و شباهت معنایی آن با نمونه های داده ترین خیلی زیاد نیست با وجود اینکه بافت، رنگ و بسیاری از عناصر ظاهری آن شباهت بسیاری با نمونه ها دارد. این روش می‌تواند در کاربردهایی نظیر تولید داده‌های مصنوعی برای آموزش مدل‌های تشخیصی یا Data Augmentation کاربرد داشته باشد.

2.5.2- بازسازی

یکی از اهداف اصلی طراحی مدل‌های VAE، بررسی توانایی آن‌ها در بازسازی تصاویر ورودی است. با توجه به اینکه در این تمرین، مدل تنها با تصاویر نرمال آموزش دیده است، بررسی نحوه‌ی بازسازی تصاویر پولیپ که مدل با آن اموزش دیده نشده است اهمیت زیادی دارد. این بخش از تمرین به همین منظور طراحی شده است.

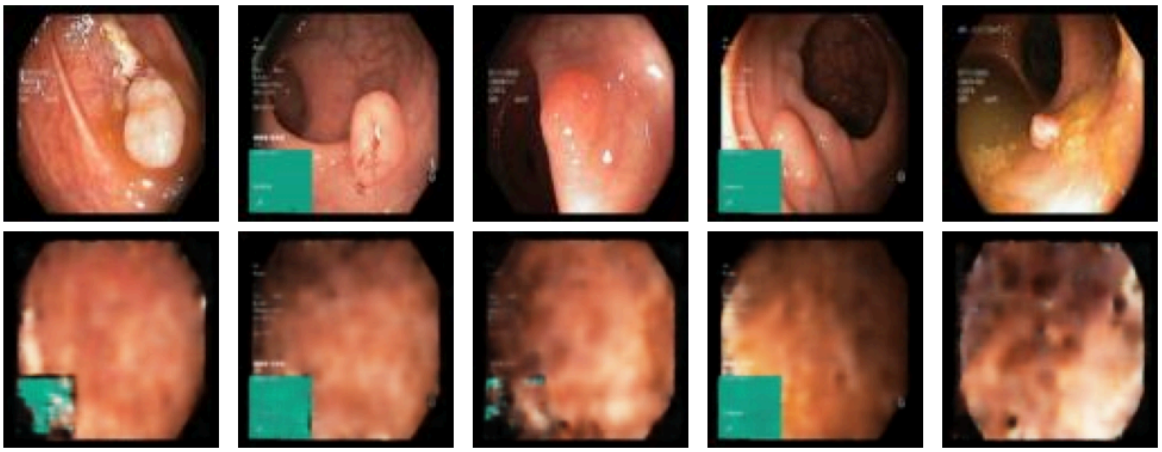
در این مرحله، 5 تصویر به‌صورت تصادفی از تصاویر پولیپ انتخاب شدند. تصاویر پولیپ از لحاظ ساختاری و محتوایی متفاوت با داده‌هایی هستند که مدل در طول آموزش دیده است. بنابراین، این ارزیابی عملاً مدل را در شرایط خارج از توزیع آموزش قرار می‌دهد. هر یک از تصاویر انتخاب‌شده ابتدا به‌صورت فرم عددی تبدیل شده و سپس به مدل داده شدند. مراحل بازسازی شامل موارد زیر است:

1. عبور تصویر از انکودر برای استخراج بردارهای میانگین و لاگ واریانس
2. نمونه‌گیری از فضای نهفته با استفاده از تابع reparameterize
3. عبور بردار z حاصل از دیکودر برای تولید تصویر بازسازی‌شده

این عملیات بدون فعال‌سازی گرادیان انجام شد تا فقط ارزیابی انجام گیرد و پارامترهای مدل تغییر نکنند.

برای مقایسه، تصاویر اصلی پولیپ و نسخه‌های بازسازی‌شده‌ی آن‌ها نمایش داده شدند. ردیف اول شامل تصاویر اصلی و ردیف دوم شامل خروجی مدل می‌باشد:

تصویر 2.6: مقایسه تصاویر اصلی (ردیف بالا) با تصاویر تولید شده توسط مدل آموزش داده شده تنها با داده های نرمال

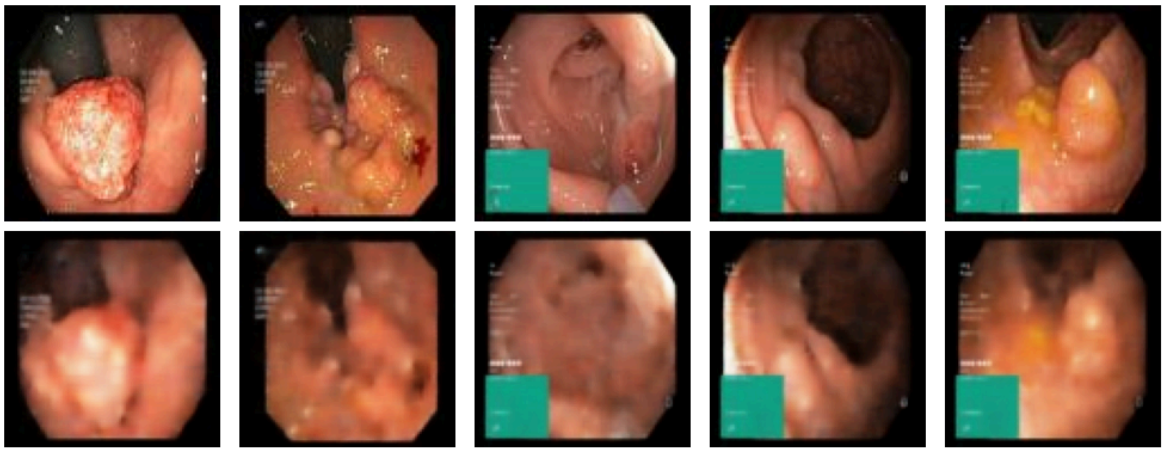


همان‌طور که انتظار می‌رفت، نتایج بازسازی در این مرحله دقت کمتری دارند و تفاوت ظاهری قابل‌توجهی میان تصاویر اصلی و بازسازی‌شده مشاهده می‌شود. علت این امر آن است که مدل تنها با داده‌های نرمال آموزش دیده و در زمان آموزش هرگز ساختارهای غیرعادی مانند پولیپ را مشاهده نکرده است. بنابراین، هنگام مواجهه با این تصاویر جدید، مدل تمایل دارد آن‌ها را به نزدیک‌ترین الگوهای آشنای خود (یعنی تصاویر نرمال) نگاشت کند و در نتیجه جزئیات اختصاصی ناهنجاری‌ها در بازسازی از بین می‌رود.

این رفتار دقیقاً همان هدفی است که در مقاله نیز به آن اشاره شده و برای کاربردهایی مانند Anomaly Detection مناسب است. پایین بودن کیفیت بازسازی برای داده‌های خارج از توزیع، می‌تواند به‌عنوان معیاری برای شناسایی تصاویر غیرعادی استفاده شود.

اما در مقایسه با این مدل که تنها بر روی داده های نرمال آموزش داده شده بود، مدلی که در آن بر روی هر دو کلاس از داده ها آموزش داده شده بود نیز بررسی شد و نتایج بازسازی تصاویر به شکل زیر است:

تصویر 2.7: مقایسه تصاویر اصلی (ردیف بالا) با تصاویر تولید شده توسط مدل آموزش داده شده با هر دو کلاس از داده ها



در این تصاویر چون نمونه هایمان از توزیع تقریبا یکسانی با داده های آموزش برخوردار است، بازسازی تصاویر با دقت بسیار خوبی صورت گرفته و مدل توانایی بازسازی هم تصاویر نرمال و هم پولیپ را به خوبی دارا است در حالی که در مدل اول مدل تنها توانایی بازسازی تصاویر نرمال را داشت. با توجه به هدف تمرین و مقاله که شناسایی تصاویر پولیپ از نرمال است مدل اول به خوبی عملکرد موفقی دارد و میتوانیم از تفاوت تصویر تولید شده با تصویر اصلی به خوبی تشخیص دهیم که آیا تصویر وردی یک تصویر نرمال است یا پولیپ است.

توجه شود که این بخش خارج از صورت سوال بوده و صرفا برای توضیح اضافی مورد استفاده قرار گرفته است.

2.6- ارزیابی عددی

پس از آموزش مدل EndoVAE بر روی تصاویر نرمال، در این بخش به تحلیل کمی کیفیت بازسازی تصاویر پولیپ پرداخته شده است. با توجه به اینکه مدل هرگز داده‌های پولیپ را در زمان آموزش مشاهده نکرده، انتظار می‌رود دقت بازسازی برای این تصاویر پایین‌تر باشد. به همین منظور، از دو معیار عددی استاندارد در حوزه‌ی پردازش تصویر، یعنی PSNR و SSIM استفاده شده است.

PSNR اختلاف بین تصویر بازسازی‌شده و تصویر اصلی را از لحاظ شدت پیکسل‌ها اندازه‌گیری می‌کند. مقادیر بالاتر PSNR نشان‌دهنده‌ی شباهت بیشتر است.

SSIM ساختار تصویر را در نظر می‌گیرد و شباهت ساختاری بین دو تصویر را ارزیابی می‌کند. مقدار SSIM در بازه‌ی 0 تا 1 قرار دارد و مقدار نزدیک به 1 بیانگر شباهت ساختاری بالاست.

ابتدا 50 تصویر به‌صورت تصادفی از پوشه‌ی processed/polyp/imgs انتخاب شدند. برای هر تصویر، عملیات بازسازی با مدل آموزش‌دیده انجام شد و سپس PSNR و SSIM بین تصویر اصلی و بازسازی‌شده محاسبه شدند. در مرحله‌ی ارزیابی، مدل در حالت eval قرار گرفت و عملیات درون torch.no_grad() انجام شد.

نتایج برای مدلی که تنها با داده های نرمال آموزش داده شده بود به صورت زیر است:

تصویر 2.8: نتایج ارزیابی روی داده های پولیپ در مدلی که تنها با داده های نرمال آموزش داده شده است

```
Average PSNR: 17.55911146918379
Average SSIM: 0.45497000217437744
```

همان‌طور که مشاهده می‌شود، مقادیر PSNR و SSIM برای تصاویر پولیپ به‌صورت میانگین در محدوده‌ی متوسطی قرار دارند. این امر قابل پیش‌بینی است، زیرا مدل صرفاً با تصاویر نرمال آموزش دیده و توانایی بازسازی دقیق ساختارهای غیرعادی را ندارد. این نتایج نشان می‌دهند که مدل در مواجهه با داده‌های خارج از توزیع عملکرد بازسازی ضعیف‌تری دارد، که از منظر کاربردهای تشخیص ناهنجاری، رفتار مطلوبی تلقی می‌شود.

در عوض در مدلی که که با هر دو سری داده ها آموزش داده شده است نتایج به صورت زیر است:

تصویر 2.9: نتایج ارزیابی روی داده های پولیپ در مدلی که با هر دو سری از داده ها آموزش داده شده است

```
Average PSNR: 28.333465454379507
Average SSIM: 0.7932332754135132
```

به وضوح در این حالت نتایج ارزیابی عملکرد بهتری را نشان میدهد که ناشی از یادگیری توزیع داده های پولیپ علاوه بر داده های نرمال است.

در نهایت استفاده از PSNR و SSIM به‌عنوان معیارهای مکمل برای ارزیابی بازسازی در این تمرین، دید عددی مفیدی نسبت به عملکرد مدل در شرایط غیرمنتظره فراهم می‌کند. این ارزیابی‌ها نشان می‌دهند که اگرچه مدل قادر به بازسازی کلیت تصویر است، اما در حفظ جزئیات دقیق، به‌ویژه در مورد ناهنجاری‌ها، با چالش مواجه است. امری که از اهداف اصلی طراحی مدل نیز می‌باشد.

2.7- تحلیل نتایج و بحث نهایی

2.7.1- تحلیل کیفی

2.7.1.1- واقع گرایانه بودن:

در این بخش پرسیده شده است که تصاویر تولید شده تا چه حد واقع گرایانه هستند؟ همانطور که توضیح داده شد تصاویری که با نمونه‌گیری تصادفی از فضای نهفته و عبور از دیکودر تولید شدند، از نظر بصری ساختارهایی طبیعی و نزدیک به تصاویر سالم واقعی داشتند. با وجود اینکه این تصاویر مصنوعی‌اند، بافت، رنگ و ترتیب مکانی اجزای آن‌ها با تصاویر واقعی نرمال همخوانی دارد. این نشان می‌دهد که مدل به‌خوبی توانسته توزیع داده‌های نرمال را در فضای نهفته یاد بگیرد و از آن برای تولید تصاویر واقع‌گرایانه استفاده کند. اما در این بخش از تصویر اصلی ورودی داده شده به مدل تفاوت واضحی دارد. بدین صورت که به عنوان مثال تصویر پولیپ به مدل داده میشود و خروجی تصویری شبیه به تصویر نرمال است که این مورد

کاملا منطبق بر هدف مقاله و تمرین است و از این مورد برای تشخیص Anomaly میتوان استفاده کرد.

2.7.1.2- جزئیات حفظ شده و محو شده:

در این بخش پرسیده شده است که چه جزئیاتی از تصویر حفظ و چه جزئیاتی محو شده است؟ در بازسازی تصاویر پولیپ، به‌وضوح مشاهده می‌شود که بخش‌هایی از تصویر که دارای ساختار ناهنجار هستند (مثل خود پولیپ)، به‌درستی بازسازی نشده و در بسیاری از موارد محو یا ساده‌سازی شده‌اند. در مقابل، نواحی اطراف که به داده‌های نرمال شباهت بیشتری دارند، با دقت بالاتری بازسازی شده‌اند. این موضوع دقیقاً با هدف مدل‌سازی در مقاله هم‌راستا است: اینکه مدل صرفاً از داده‌های نرمال یاد بگیرد و هنگام مواجهه با داده‌های ناآشنا، دچار خطای بازسازی شود. بنابراین نقاطی که اشتراک خوبی با تصاویر نرمال دارند حفظ شده اند و نقاطی که تنها مربوط به تصاویر پولیپ است تقریبا محو شده اند.

2.7.2- تحلیل کمی

2.7.2.1- تحلیل PSNR و SSIM:

در این بخش پرسیده شده است که مقادیر PSNR و SSIM چه نکاتی درباره کیفیت بازسازی نشان میدهد؟ مقادیر میانگین PSNR و SSIM برای 50 تصویر پولیپ به ترتیب در حدود 17.55 و 0.45 به‌دست آمد. این مقادیر به‌وضوح پایین‌تر از حالتی هستند که مدل برای بازسازی داده‌های نرمال آموزش دیده باشد. در واقع این اعداد نشان می‌دهند که مدل در بازسازی تصاویر پولیپ دقت کمتری داشته، که نتیجه‌ی طبیعی استفاده از داده‌های نرمال برای آموزش و استفاده نکردن از داده های پولیپ است. از منظر تشخیص ناهنجاری، این رفتار مطلوب تلقی می‌شود و کاملا با انتظارمان تطابق دارد. همانطور که مشاهده شد در صورت استفاده از داده های پولیپ در آموزش این مقادیر به 28.33 و 0.79 رسیدند. بنابراین کیفیت بازسازی در مدل اصلی بر روی داده های پولیپ پایین و بر روی داده های نرمال بالا است که این نشان دهنده عملکرد صحیح و دقیق مدل دارد و برای کلاس بندی این دو کلاس میتواند استفاده شود.

2.7.2.2- مشکل در دقت پایین معیار ها

در این بخش پرسیده شده است که علت پایین بودن معیارهای PSNR و SSIM چیست؟ همانطور که در بالاتر توضیح داده شده این اختلاف ناشی از اختلاف توزیع میان داده های آموزشی که نرمال هستند و داده های تست که پولیپ هستند است. در نرمال سازی، پیش پردازش و هایپرپارامتر ها مشکل اساسی ای وجود ندارد زیرا با استفاده از همان مدل و پیش پردازش و یادگیری کل داده ها (شامل نرمال و پولیپ) توانستیم به دقت خیلی بالاتری برسیم

2.8- نتیجه گیری خلاصه

در این تمرین با هدف درک دقیق معماری مدل‌های Variational Autoencoder، به پیاده‌سازی و ارزیابی مدل EndoVAE مطابق مقاله‌ی مرجع پرداخته شد. ابتدا با استفاده از تصاویر نرمال، مدل آموزش داده شد تا تنها توزیع داده‌های سالم را فراگیرد. سپس عملکرد مدل در بازسازی تصاویر پولیپ مورد بررسی قرار گرفت. نتایج کیفی نشان داد که مدل توان بازسازی ساختارهای غیرعادی را ندارد و جزئیات ناهنجار (مانند پولیپ) را محو یا ساده‌سازی می‌کند. این ویژگی با هدف طراحی‌شده مدل برای تشخیص ناهنجاری هم‌راستا است. همچنین مقادیر PSNR و SSIM به‌صورت کمی ضعف بازسازی را تأیید کردند. در مجموع، این تمرین توانست درک عملی و عمیقی از معماری

VAE، نحوه‌ی آموزش آن، و تحلیل خطای بازسازی فراهم آورد و مقدمه‌ای مؤثر برای ورود به مباحث تولید داده و تشخیص ناهنجاری باشد.