به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین ششم

پرسش ۱	نام و نام خانوادگی	آرین فیروزی
	شماره دانشجویی	810100196
پرسش ۲	نام و نام خانوادگی	آرمان مجیدی
	شماره دانشجویی	810100205
	مهلت ارسال پاسخ	۱۴۰۱.۱۰.۲۷

فهرست

شکل‌ها

پرسش 1

پرسش 2

جدول‌ها

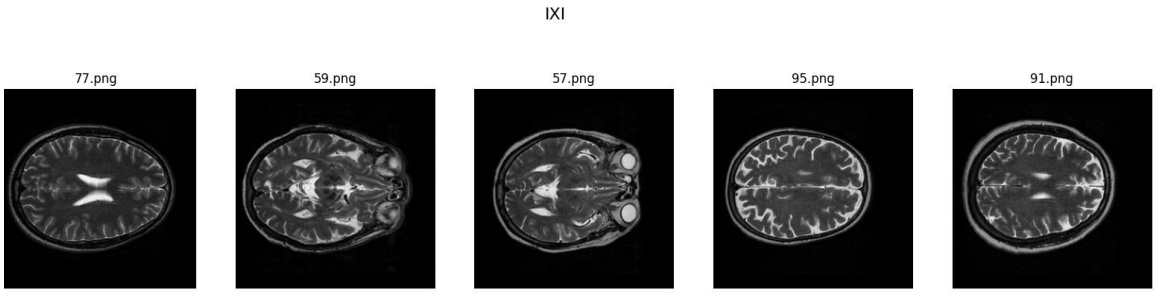
پرسش 1

پرسش 2

پرسش ۱. طراحی و پیاده سازی VAE Triplet برای تشخیص تومور در MRI

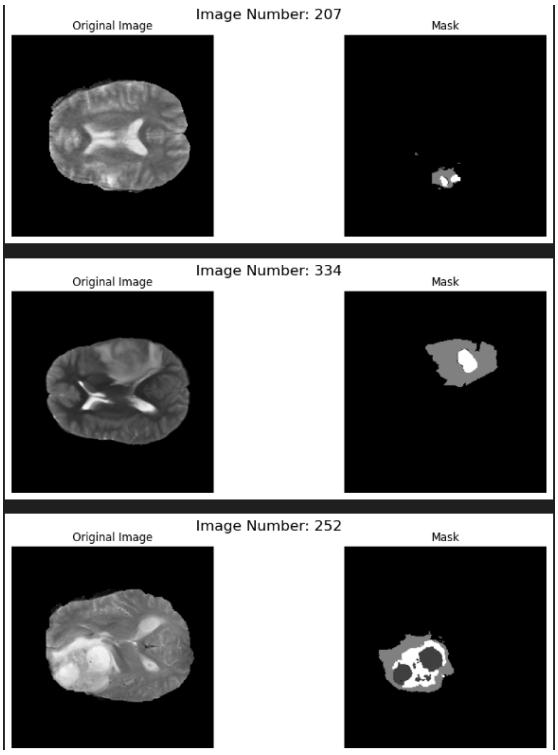
۱-۱. هدف و دیتاست

برای این تمرین از دو دیتاست IXI (فقط بخش Validation) و BraTS (برای تست) استفاده شد. در بخش IXI تصاویر به صورت png وجود دارند که با استفاده از کتابخانه pillow خواندیم و چند نمونه از آن را نمایش دادیم. عکس های نمونه در تصویر 1.1 قابل مشاهده اند.



شکل 1.1. داده های IXI

تصاویر BraTS با فرمت nii قرار داده شده بود که برای خواندن آن از کتابخانه nibabel استفاده کردیم و تصویری از نمونه عکس های موجود به همراه ماسک آنها در تصویر 1.2 نشان داده شده است.



شکل 1.2. داده های BraTS2020

در گام بعدی، دو دیتاست تعریف کردیم که هر کدام یکی از دیتاست ها را در بر میگرفت. عکس های IXI به راحتی قابل تبدیل به دیتاست مطلوب بودند، اما برای BraTS چون تصویر سه بعدی بود، اسلایس وسط عکس و ماسک را به عنوان مرجع انتخاب کردیم و دیتاست را با توجه به آن ساختیم. در مرحله آخر این دیتاست ها را به دیتالودر انتقال دادیم تا بتوان عمل آموزش و تست را بر روی مدل اعمال کرد.

لازم به ذکر است که به علت حجم بالای عکس ها و سنگین بودن مدل، به مشکل حجم GPU برخورد کردیم، بنابرین به جای تمام 57000 عکس موجود در داده های آموزش، تنها از 1000 تا برای آموزش استفاده شد. همچنین بر خلاف مقاله، پیش پردازشی اعمال نشد چون پیش پردازش های تصاویر پزشکی معمولا زمانبر بوده و به دانش پزشکی نیاز دارد (اعمال از قبیل template matching یا skull stripping).

۱-۲. پیاده سازی VAE ساده

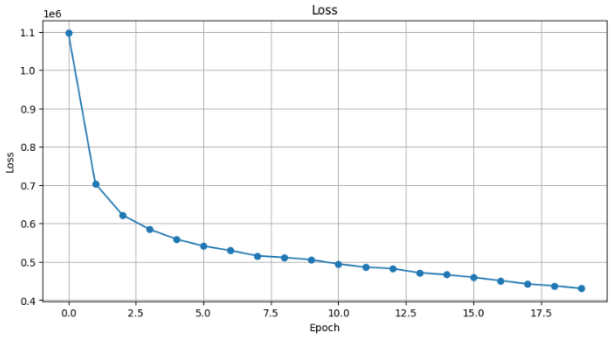
مدل های VAE، نوعی از انکودر دیکودر ها هستند که از اتوانکودر ها مشتق شده اند.

کارکرد کلی اتوانکودرها: به این شکل است که بخش انکودر یک تصویر، یا هر داده ای را، به استفاده از لایه های مختلف به فضای کوچکتری منتقل میکند و تلاش میکند feature های مفید عکس را در آن فضا نشان دهد. این عمل به نوعی میتواند dimension reduction تلقی شود. به فضایی که داده در آخر مرحله انکود map میشود، فضای latent میگویند. در مرحله ی بعدی، دیکودر تلاش میکند که داده های اولیه را از فضای latent بازسازی کند. این بازسازی معمولا به لایه هایی مشابه به برعکس آنچه در انکودر وجود دارد، مانند deconv که در واقع کانولوشن با استراید کسری است، یا upscale انجام میگیرد. در این نوع مدل ها خطا توسط خطای بازسازی و تصویر اولیه صورت میگیرد.

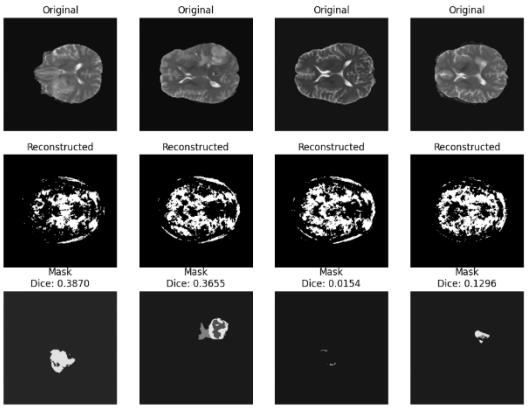
مشکل اساسی اتوانکودر ها این است که فضای latent کاملا دلخواه و رندم صورت میگیرد و نمیتوان به طور smooth از یک نمونه به نمونه دیگر رفت، در نتیجه مدل جز تصاویر دیده شده تصویر جدیدی نمیسازد (یعنی ما نمیتوانیم نویز مشابه به چیزی که میخواهیم بدهیم). برای حل این مشکل، در VAE یک ترم دیگر به loss اضافه میشود که KL دیورژانس است. این ترم که برای اندازه گیری اختلاف دو توزیع احتمال به کار میرود، باعث میشود که بتوانیم مدل را توری آموزش دهیم که فضای latent آن به صورت توزیع دلخواه ما (معمولا توزیع نرمال) باشد و بتوانیم به صورت smooth بین دو نمونه transition انجام دهیم. در این حالت میتوان از مدل برای تولید عکس دلخواه خودمان استفاده کرد. فرمول kl ب شکل زیر است:

$$D_{kl}(p(x)||q(x)) = \sum_{x \in X} p(x) ln \frac{\mathrm{p(x)}}{q(x)})$$

برای پیاده سازی این مدل، یک ساختار انکودر دیکودر ساده با یک لایه fully connected برای انکد و یک لایه دیگر برای دیکود، با اکتیویشن ReLU و دو لایه fully connected دیگر برای تولید mu و varians استفاده کردیم که برای مپ کردن فضای latent به فضای نرمال استفاده خواهند شد. برای loss از دو معیار kld و لاس ساده ی mse استفاده کردیم و این مدل را آموزش دادیم. نتایج آموزش در تصاویر 1.3 و 1.4 قابل مشاهده اند.



شکل 1.3. خطای آموزش مدل VAE



شکل 1.4. نتایج خروجی مدل VAE

برای آموزش از سایز بچ 4 و 20 دوره استفاده شد.

۱-۳. پیاده سازی Tri-VAE

این مدل با توجه بامقاله معرفی شده، از سه بخش انکودر، دیکودر و Gated Cross Skip تشکیل شده که ما برای راحتی کار این سه را به عنوان سه مدل متفاوت ساختیم و در آخر با هم ترکیب کردیم.

در بخش انکودر، مانند مقاله عکس ها از 256 در 6 مرحله به سایز 8 در 8 کوچک شدند. این عمل با استفاده از اکتیویشن و کانولووشن های تشریح شده در مقاله انجام شد. در مرحله آخر، به جای 512 از سایز latent 256 استفاده کردیم، چرا که حجم مدل به قدری زیاد میشد که خود مدل در gpu جا نمیشد، پس مجبور شدیم حجم مدل را قدری کاهش دهیم.

در بخش دیکودر نیز همانند مقاله عمل شد. قبل از اضافه کردن اسکیپ کنکشن های مربوط به انکودر، یک خروجی کوچک coarse نیز اعمال شد که در محاسبه خطا کاربرد دارد. این کار باعث میشود که یک تصویر از اوایل مرحله دیکود داشته باشیم و بتوان عملکرد مدل را قبل از رسیدن به مرحله آخر ارزیابی

کرد و همچنین این خطا به همراه gcs باعث میشود مدل ترغیب شود که از اسکیپ کانکشن ها کمتر استفاده کند و تومور ها را بازسازی نکند.

برای بخش gcs نیز مثل مقاله، دو ورودی از بهش دیکودر و انکودر ابتدا با network in network کاهش بعد داده شد و سپس از لایه های fc عبور داده شدند. (این بخش در مقاله به صورت مستقیم ذکر نشده بود، اما با مقایسه نتایج حاصل از استفاده از fc به این نتیجه رسیدیم که بلاک های موجود در تصویر مقاله لایه های fully connected هستند) این مقادیر از یک فانکشن تانژانت هیپربولیک عبور داده شدند و در آخر یک fc به ان اعمال شده و به جمع شدن با بخش دیکودر به خروجی هدایت شدند. این لایه برای هر سه کانکشن انکودر به دیکودر اعمال شد و تنها اسکیپ کانکشن latent space مستقیم است.

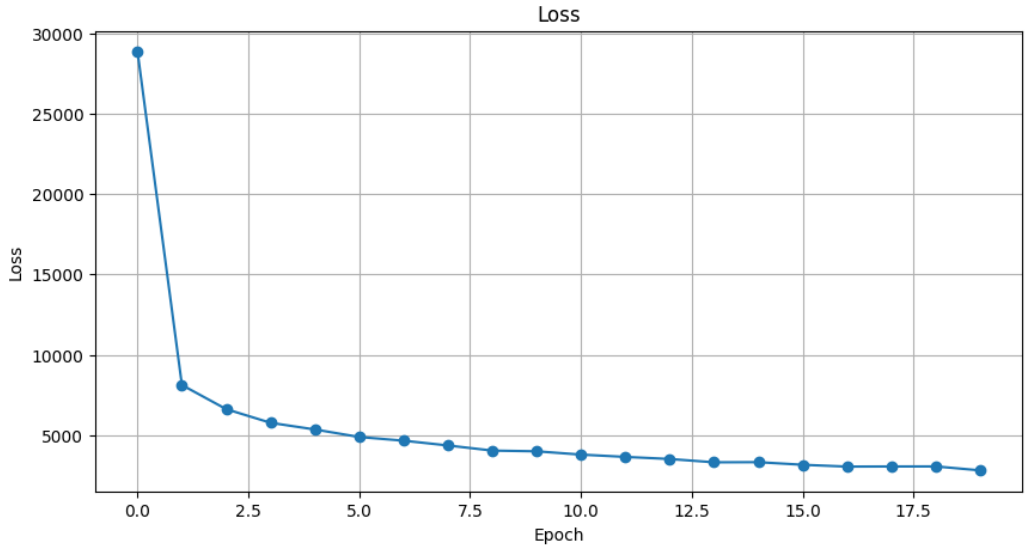
برای بخش دیکودر، تصویر ساخته شده در هر مرحله با اسکیپ کانکشن ها concat شده و در آخر به خروجی داده شد.

در مقاله از 3 ورودی anchor, positive و negative استفاده شده بود. برای ساخت این سه تصویر، تصویر نگاتیو با اعمال نویز درست شد و سایر عکس ها همان تصویر اصلی بودند.

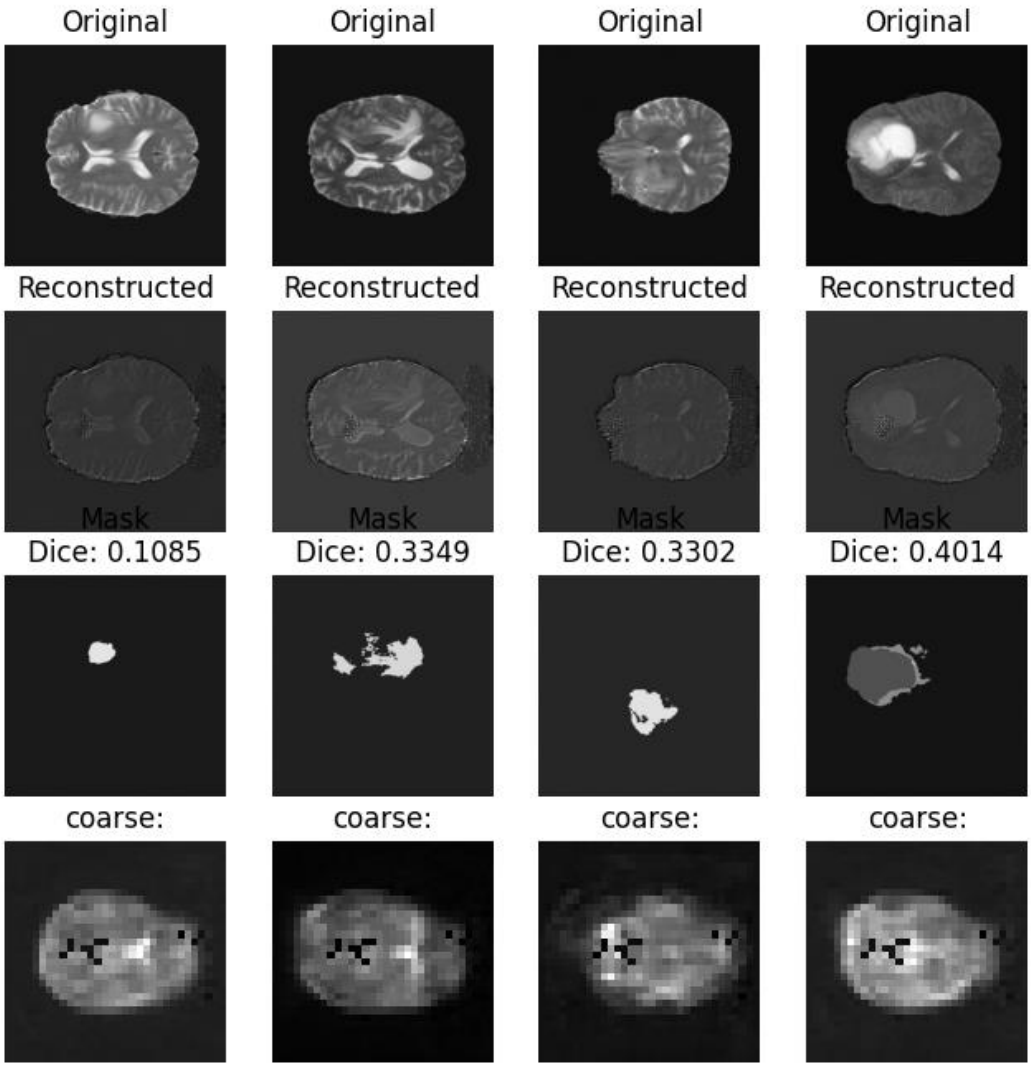
تابع هزینه triplet با استفاده از تعاریف ارائه شده در مقاله و صورت سوال ساخته شد. این تابع با استفاده از نتیاج مدل بر روی عکس ها، خطا را به دست میاورد. تابع هزینه تریپلت به گونه این طراحی شده که شباهت عکس ها را به نوعی بسنجد که مدل را وادار کند بین اعضای یک کلاس نوسان کم و بین اعضای یک کلاس و کلاس های دیگر نوسان زیادی وجود داشته باشد. برای این تابع از margin=1 و تابع pairwise_distance استفاده شد و میانگین مقادیر به دست آمده برگردانده شد.

برای محاسبه خطا از 4 تابع مختلف استفاده شد. خطای triplet که در بالا به آن پرداختیم. خطای kld که آن هم توضیح داده شد و برای 2 حالت مثبت و anchor محاسبه شد. خطای l1 که مجموع خطا های مطلق coarse برای هر سه حالت ورودی و خطای خروجی نهایی تنها برای نگاتیو، و خطای ssim برای نگاتیو. این چهار خطا به ترتیب به منظور تفکیک یک به یک عکس ها در مقایسه با عکس های دیگر، تنظیم فضای latentبه توزیع دلخواه، خطای کلی تصویر coarse و خطای ساختار رایج برای مقایسه دو تصویر هستند. چون این خطا ها در مقدار عددی تفاوت زیادی داشتند، آن را اسکیل کردیم تا هر 4 خطا در loss نهایی تاثیر قابل توجهی داشته باشند. همچنین خطای نهایی نیز اسکیل شد چون خطا بیش از حد کم میشد.

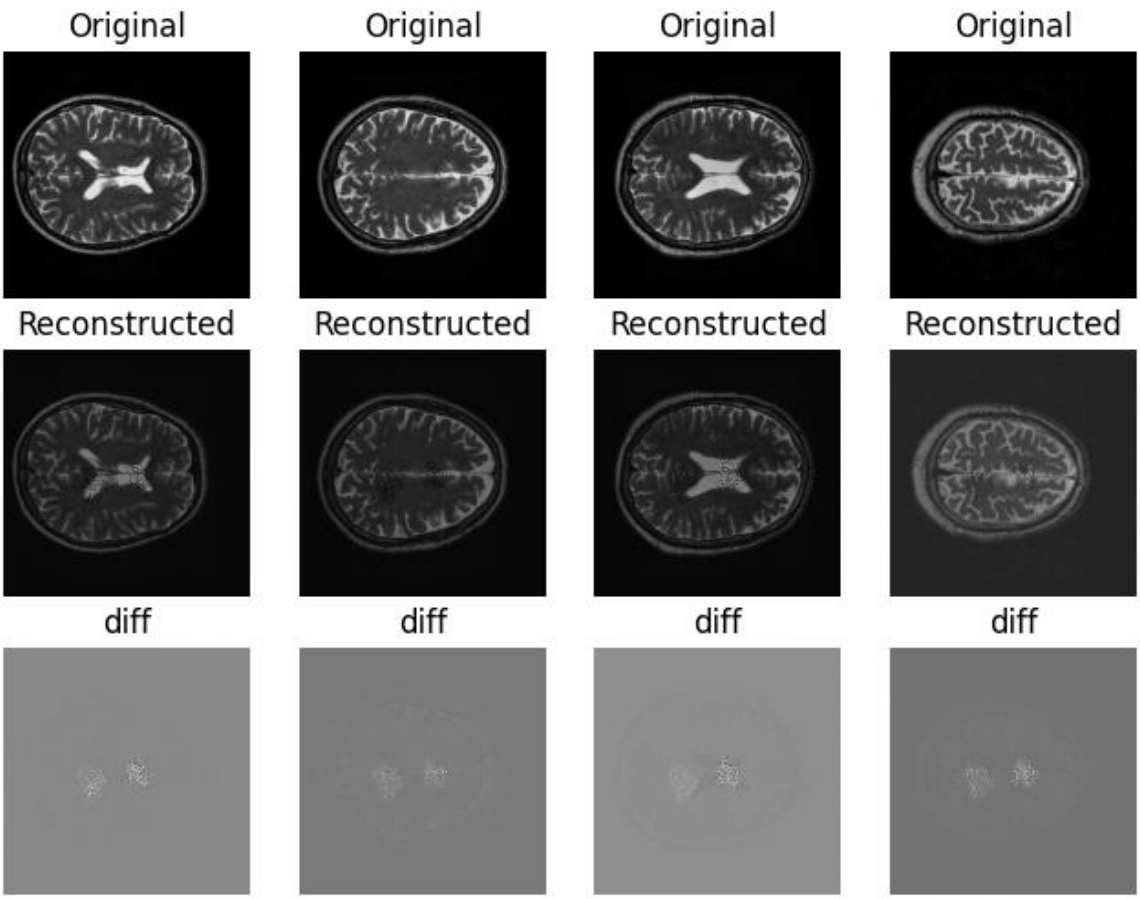
نتیجه آموزش به مدت 20 ایپاک با lr=e-4 در تصویر 1.5 و 1.6 قابل مشاهده است. تصویر 1.7 نتیجه را بر روی دیتای آموزش نشان میدهد. تحلیل های این موضوع در بخش ارزیابی آورده شده است.



شکل 1.5. خطای آموزش مدل TriVAE (در نمودار 100 برابر شده)



شکل 1.6. خروجی مدل TriVAE بر روی دادگان تست



شکل 1.7. خروجی مدل TriVAE بر روی دادگان آموزش

1-4. ارزیابی

هر دو مدل بر روی دیتا های BraTS تست شدند که خروجی تصویری آن در تصاویر 1.6 و 1.4 آورده شده و همچنین مقایسه عددی نهایی (در کد بعد از مقایسه تصویری حساب شده و جدا هستند) در جدول 1.1 آورده شده است.

با مقایسه ی خروجی دو مدل در می یابیم که مدل vae گرچه توانسته بخش هایی از مغز را درست کند، اما به دلیل کوچک بودن شبکه و همچنین تابع خطایی که تنها kld و mse را در بر میگیرد، تصویر کلی مغز را اشتباه درست میکند و در واقع به جای کلیات تصویر، outline ها را ایجاد میکند. در مقابل اگر به خروجی triVAE دقت کنیم (مخصوصا تصویر 1.7) میبینیم که کلیات و جزئیات مغز در حال شکل گرفتن است. تصویر Coarse نیز حاکی از این است که دیکودر ابتدا کلیت مغز را استخراج میکند و تصویر 256 در 256 را در latent space مپ کرده است.

برای مقایسه خروجی مدل در دیتاست های BraTS و IXI بایستی به این نکته دقت کرد که این دو دیتاست گرچه مشابه هم هستند، اما برای اینکه تصاویر ورودی دقیقا یکی شوند، ناچاریم پیش پردازش انجام دهیم و تمپلیت عکسها را با هم یکی کنیم؛ با این حال، مدل دقت نسبتا خوبی در باز سازی دارد و تومور ها گرچه تا حدود بازسازی شده اند، اما این باز سازی به مراتب کمتر از مراحل قبل gcs (که متاسفانه تصاویر آن در داکیومنت و کد وجود ندارد) است و امید میرود که در صورت آموزش بیشتر و استفاده از کل دیتاست به جای 1000 نمونه که به دلایل شرح داده شده استفاده کردیم، نتایج بهتر از نتیجه فعلی شوند. نمودار خطا نیز حاکی از این است که آموزش به طور کامل تمام نشده و روند خطا رو به کاهش است.

نکته قابل توجه دیگر آن است که مدل توانسته تومور های بزرگتر را بیشتر از تومور های کوچک حذف کند، این نشان میدهد که اولا فضای latent ما (که به دلایلی که بالاتر ذکر شد نصف مقاله است) نتوانسته جزئیات را به طور کامل مپ کند و مدل برای بازسازی جزئیات به اسکیپ کانکشن ها بیشتر از فضای ما توجه میکند.

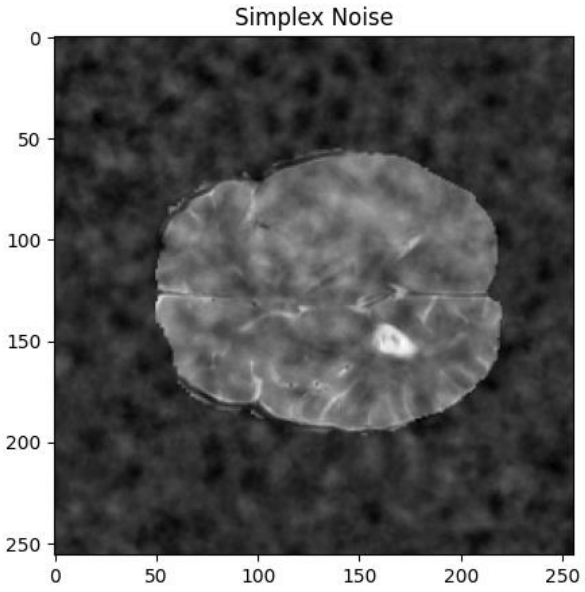
مقدار Dice با توجه به فرمول $\text{Dice}=\frac{2|X\cap Y|}{|X|+|Y|}$ حساب شد و دو ترم smooth برابر با 1 برای جلوگیری از تقسیم بر صفر و همچنین همگن تر شدن مقادیر به صورت و مخرج اضافه شد. threshold این فرمول مقدار 0.1 برای تمامی مدل ها در نظر گرفته شد. این معیار برای سنجش نزدیکی دو عکس است و در پزشکی جزو رایج ترین معیار ها برای مقایسه خروجی دو مدل به شمار میرود.

تست مقاله ها بر روی حدود 360 عکس از دیتاست BraTS2020 انجام شدند و نتایج عددی آن در جدول 1.1 قرار داده شده است. این نتایج مشاهدات ما را تایید میکند و TriVAE با نویز Coarse بهترین نتیجه را دارد.

1-5. امتیازی (فقط simplex)

در این مورد، نتایج را با استفاده از نویز simplexتولید کردیم. این نتایج با توجه به اینکه تولید این نویز پروسه زمانبری است، تنها بر روی 100 نمونه ترین شدند، ولی نتایج تقریبا مشابه به نتایج مقاله بود.

نویز تولید شده در بخش قبلی، نویز Coarse بود که در واقع مقادیر رندم تولید شده توسط نویز نرمال بودند که از تصویر 32 در 32 اسکیل شدند. در این بخش، نویز ٔ Simplex تولید شد که تولید آن نسبت به سایر نویز ها مثل نویز perlin سریعتر است (پیاده سازی ما سریع نبود) و به همین جهت کاربرد دارد. نمونه ای از تصاویر نویز دار تولید شده با این تابع در شکل 1.8 نشان داده شده است.



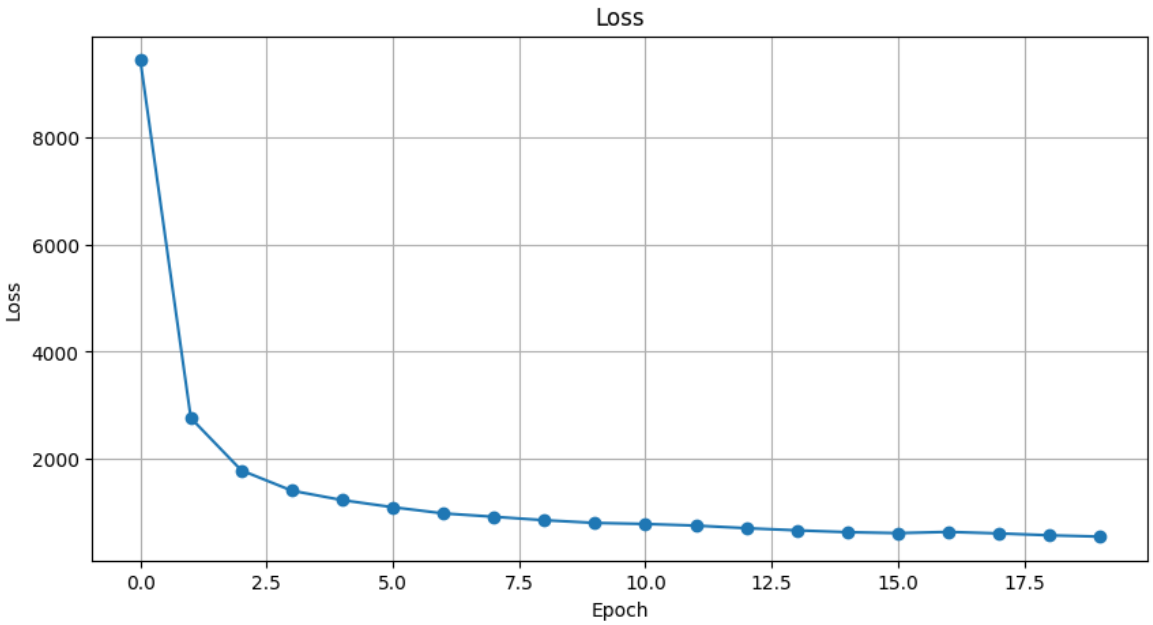
شکل 1.8. نویز simplex

نتیاج خروجی مدل با نویز Simplex در تصاویر 1.9 (خطا) ، 1.10 و 1.11 قابل مشاهده است.

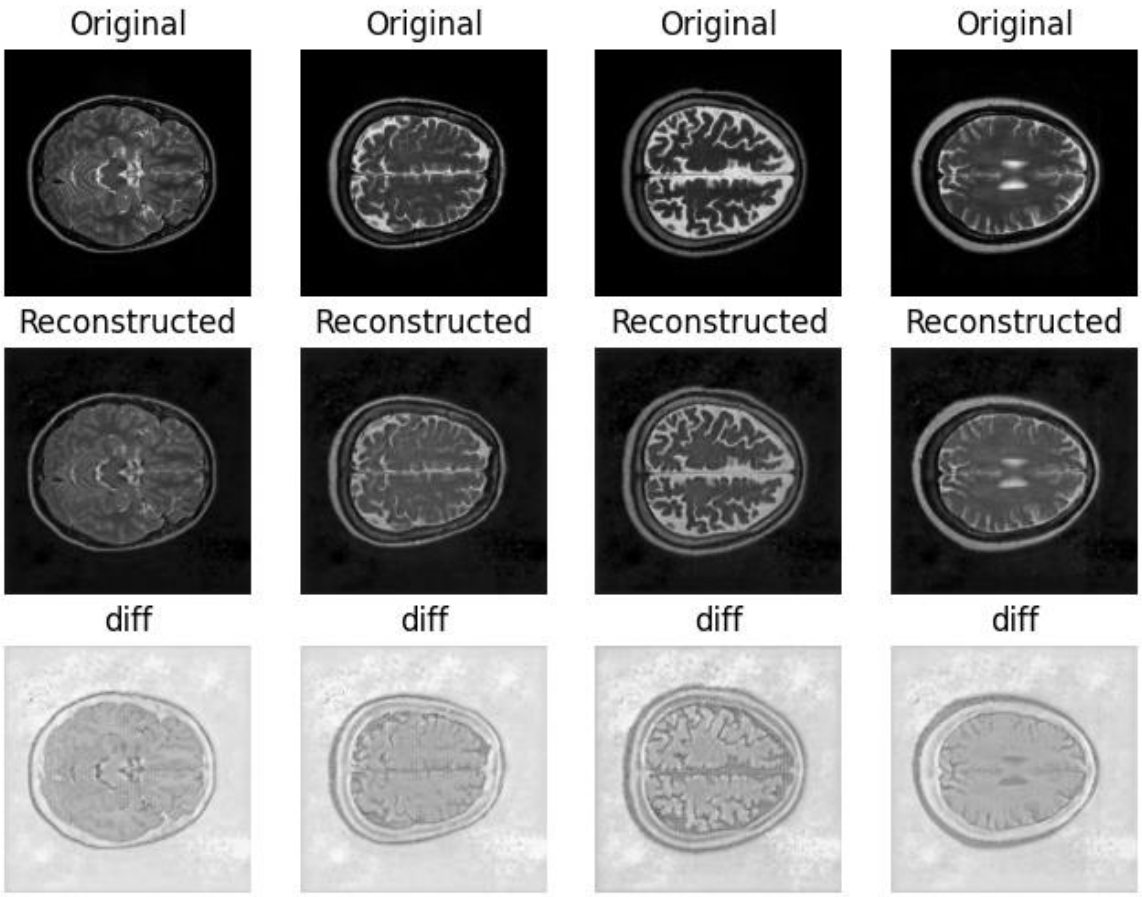
مقادیر عددی این مدل نیز در جدول 1.1 آمده است. همانند نتایج مقاله، در اینجا نیز مشاهده میکنیم که نویز Simplex ضعیفتر عمل کرده است.

جدول 1.1 دقت مدل های مختلف، ارزیابی شده توسط معیار های خواسته شده

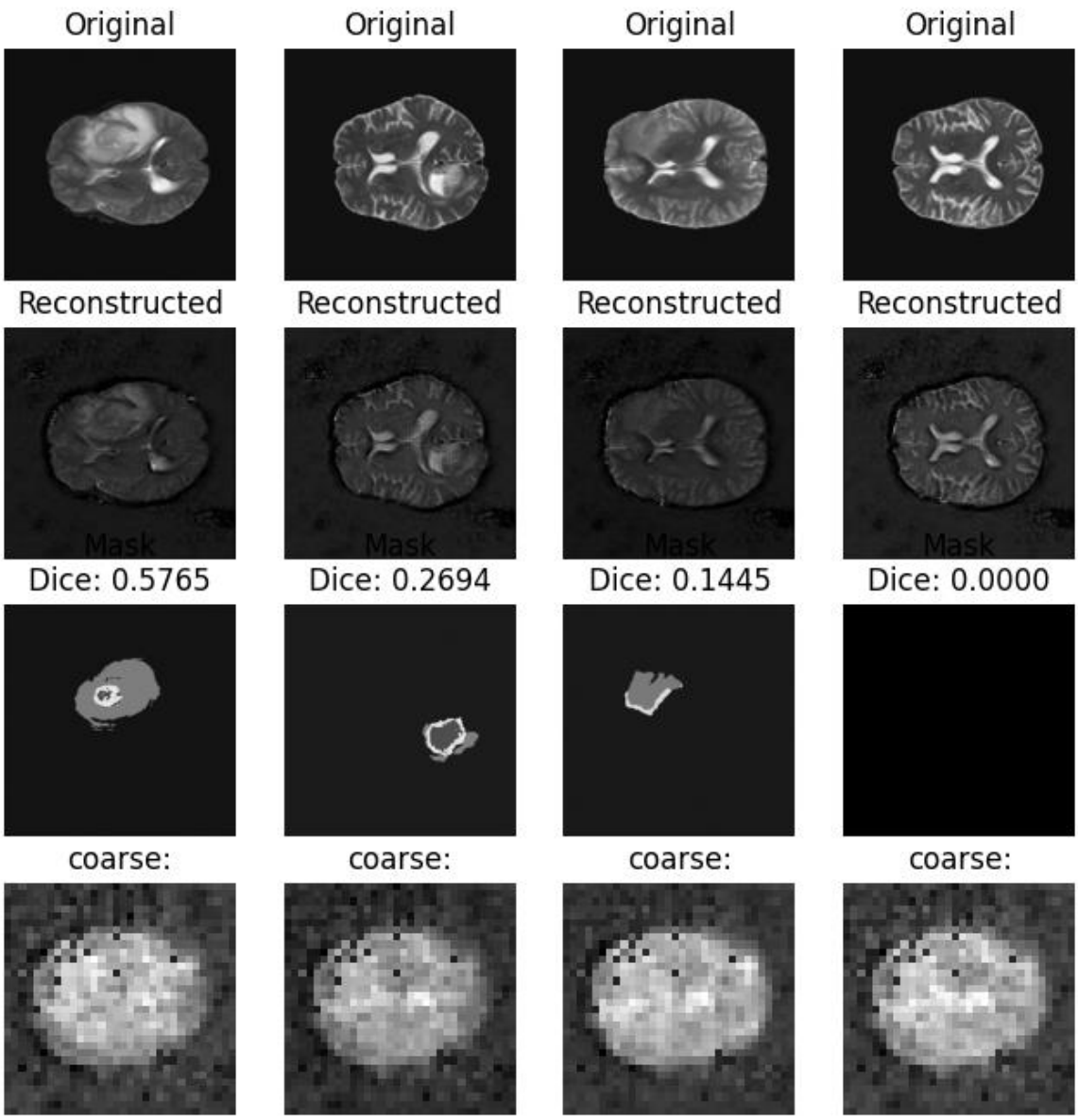
Method	Dice
VAE	0.1180
TriVAE(Coarse)	0.3110
TriVAE(Simplex)	0.2683



شکل 1.9. خطای آموزش مدل TriVAE با نویز simplex (مقادیر 100 برابر شده)



شکل 1.10. خروجی مدل TriVAE بر روی دادگان آموزش با نویز Simplex



شکل 1.11 خروجی مدل TriVAE بر روی تست با نویز Simplex

پرسش 2 – AdvGAN

2-1. آشنایی با حملات خصمانه و معماری AdvGAN

1) FGSM (Fast Gradient Sign Method): این روش یک تقریبی از مرتبه اول برای تولید نمونه‌های تخاصمی است که از شیب (گرادیان) تابع هزینه مدل هدف استفاده می‌کند. فرمول کلی آن در زیر آمده‌است:

$$x_A = x + \epsilon \times sign\big(\nabla_x \ell(f(x), y)\big)$$

این روش ساده است و بسیار سریع عمل می‌کند، اما ممکن است نمونه‌های تولید شده کیفیت بصری پایین‌تری داشته باشند.

PGD (Projected Gradient Descent): این روش نسخه پیشرفته‌تر FGSM است که با چندین گام بهینه‌سازی، به نمونه‌های تخاصمی قوی‌تری می‌رسد. این روش از فرمول زیر استفاده می‌کند:

$$x_{t+1} = proj_B \epsilon \Big(x_t + \alpha \times sign\big(\nabla_x \ell(f(x_t), y)\big)\Big)$$

این روش زمان‌برتر است اما دقت بیشتری دارد.

مزیت‌های AdvGAN:

- تولید سریع نمونه‌های تخاصمی پس از آموزش اولیه
- حفظ کیفیت بصری بالاتر نمونه‌ها با استفاده از شبکه‌های مولد تخاصمی (GAN)
- کارایی بالا در تنظیمات نیمه‌جعبه سفید و جعبه سیاه

2) تفاوت‌های کلیدی بین AdvGAN و GAN ساده در زیر آمده‌است:

- AdvGAN از خروجی مدل هدف (به عنوان تابع هزینه تخاصمی $\mathcal{L}_{adv}$) برای تنظیم تولید نمونه‌های تخاصمی استفاده می‌کند. این گرادیان‌ها کمک می‌کنند نمونه‌ها مدل هدف را گمراه کنند.
- AdvGAN با ترکیب یک مولد G و یک متمایزکننده D، نمونه‌هایی تولید می‌کند که از نظر بصری به داده‌های اصلی وفادار باشند. در عین حال، این نمونه‌ها طوری طراحی می‌شوند که مدل هدف را به اشتباه بیاندازند.

3) سه تابع هزینه در AdvGAN در زیر آمده‌اند:

- تابع هزینه GAN ($\mathcal{L}_{GAN}$):

$$\mathcal{L}_{GAN} = \mathbb{E}_x log\mathcal{D}(x) + \mathbb{E}_x \log\Big(1 - \mathcal{D}\big(x + \mathcal{G}(x)\big)\Big)$$

هدف این بخش، اطمینان از واقعی بودن نمونه‌های تولید شده از نظر بصری است.

- تابع هزینه تخاصمی ($\mathcal{L}_{adv}$):

$$\mathcal{L}_{adv}^{f} = \mathbb{E}_x \ell_f(x + \mathcal{G}(x), t)$$

که در آن t کلاس هدف است. این بخش برای اطمینان از موفقیت حمله طراحی شده است.

- تابع هزینه محدودیت ($\mathcal{L}_{hinge}$):

$$\mathcal{L}_{hinge} = \mathbb{E}_x \max(0, \|\mathcal{G}(x)\|_2 - c)$$

- در انتها، تابع هزینه کلی به صورت زیر می‌شود:

$$\mathcal{L} = \mathcal{L}_{adv}^{f} + \alpha\mathcal{L}_{GAN} + \beta\mathcal{L}_{hinge}$$

4) تفاوت بین حمله‌های جعبه سفید و جعبه سیاه و استفاده مدل از حملات جعبه سیاه:

- **حمله جعبه سفید:** مهاجم دسترسی کامل به مدل (معماری و وزن‌ها) دارد و می‌تواند از گرادیان‌ها برای تولید نمونه‌های تخاصمی استفاده کند.
- **حمله جعبه سیاه:** مهاجم به مدل دسترسی مستقیم ندارد و از مدل‌های جایگزین یا روش‌هایی نظیر انتقال‌پذیری برای تولید نمونه‌های تخاصمی بهره می‌گیرد.
- **استفاده AdvGAN در جعبه سیاه:** AdvGAN از استراتژی "تقطیر دینامیکی" برای ساخت مدل جایگزین استفاده می‌کند و سپس بر اساس این مدل، نمونه‌های تخاصمی تولید می‌کند. این رویکرد حملات موفقیت‌آمیزی حتی در تنظیمات جعبه سیاه فراهم می‌کند.

5) AdvGAN یک چارچوب تولید نمونه‌های تخاصمی با استفاده از شبکه‌های مولد تخاصمی (GAN) است که به منظور حمله به مدل‌های یادگیری عمیق طراحی شده است. در ادامه، دو مقاله پژوهشی که به توسعه و بهبود AdvGAN پرداخته‌اند، معرفی و خلاصه می‌شوند:

- **AdvGAN++: بهره‌برداری از لایه‌های نهان برای تولید نمونه‌های تخاصمی:**

این مقاله با عنوان AdvGAN++: Harnessing latent layers for adversary generation توسط Puneet Mangla و همکاران در سال ۲۰۱۹ منتشر شده است. در این پژوهش، نویسندگان نشان می‌دهند که استفاده از ویژگی‌های نهان (لایه‌های میانی شبکه) به عنوان ورودی برای تولید نمونه‌های تخاصمی می‌تواند نتایج بهتری نسبت به استفاده مستقیم از تصاویر ورودی به همراه داشته باشد. آن‌ها نسخه بهبودیافته‌ای از AdvGAN به نام AdvGAN++ را معرفی می‌کنند که با بهره‌گیری از این ویژگی‌های نهان، نرخ حملات موفقیت‌آمیز بالاتری را در مجموعه داده‌های MNIST

و CIFAR-10 بدست می‌آورد و در عین حال تصاویر تولید شده از نظر بصری واقعی‌تر هستند.

این بهبود بر اساس چارچوب اولیه AdvGAN با تغییر در ورودی مولد (Generator) صورت گرفته است؛ به‌طوری که به جای استفاده از تصویر خام، از ویژگی‌های استخراج‌شده در لایه‌های میانی شبکه استفاده می‌شود. این رویکرد باعث می‌شود مولد بتواند با دقت بیشتری اغتشاشات مؤثر برای گمراه کردن مدل هدف را تولید کند.

- **GE-AdvGAN: بهبود انتقال‌پذیری نمونه‌های تخاصمی با مدل مولد تخاصمی مبتنی بر ویرایش گرادیان:**

این مقاله با عنوان GE-AdvGAN: Improving the transferability of adversarial samples by gradient editing-based adversarial generative model توسط Zhiyu Zhu و همکاران در سال ۲۰۲۴ منتشر شده است. در این پژوهش، نویسندگان الگوریتم جدیدی به نام GE-AdvGAN را برای افزایش انتقال‌پذیری (قابلیت تأثیرگذاری بر مدل‌های مختلف) نمونه‌های تخاصمی معرفی می‌کنند. آن‌ها با بهینه‌سازی فرآیند آموزش مولد و معرفی مکانیزم ویرایش گرادیان (Gradient Editing)، موفق به تولید نمونه‌هایی می‌شوند که در مدل‌های مختلف با کارایی بالایی عمل می‌کنند. این روش با بهره‌گیری از اطلاعات حوزه فرکانس برای تعیین جهت ویرایش گرادیان، نمونه‌های تخاصمی با انتقال‌پذیری بالا و زمان اجرای کمتر نسبت به الگوریتم‌های پیشرفته موجود تولید می‌کند.

این توسعه بر اساس چارچوب اولیه AdvGAN با تمرکز بر بهبود فرآیند آموزش مولد و استفاده از مکانیزم ویرایش گرادیان صورت گرفته است. با این تغییرات، GE-AdvGAN قادر است نمونه‌های تخاصمی تولید کند که نه تنها مدل هدف را گمراه می‌کنند، بلکه در مدل‌های دیگر نیز مؤثر هستند، که این امر اهمیت ویژه‌ای در حملات جعبه سیاه دارد.

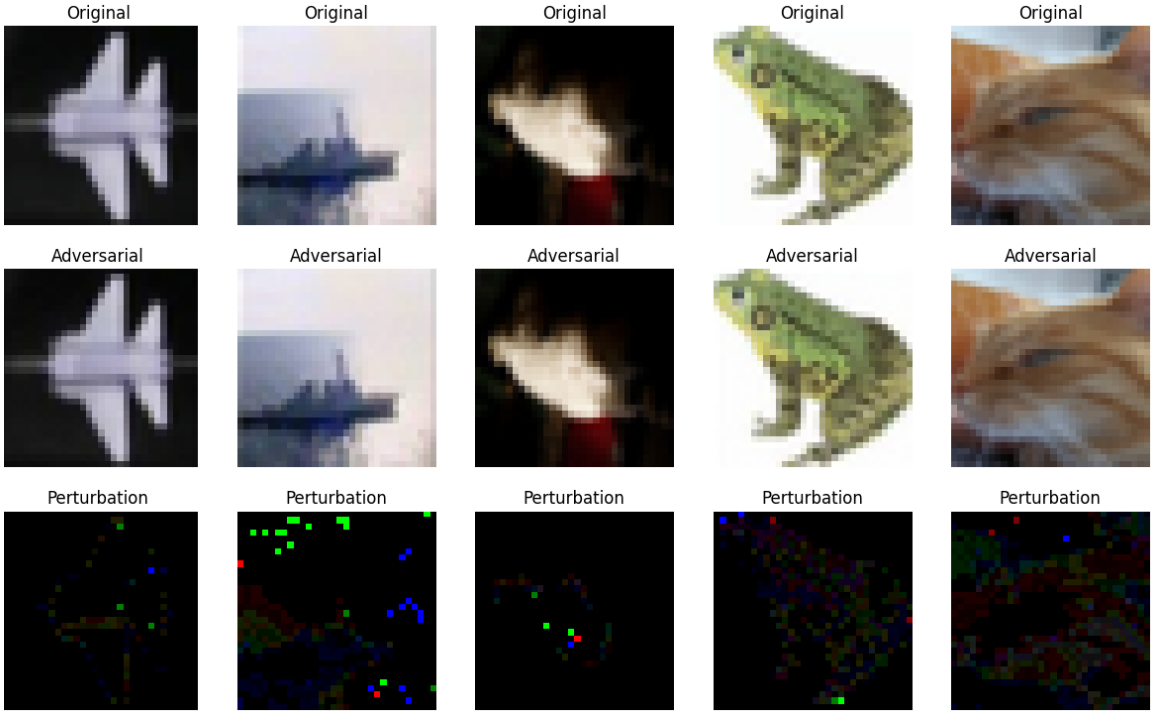
2-2. پیاده‌سازی مدل AdvGAN

1) ابتدا مطابق گفته سوال دادگان CIFAR-10 را با استفاده از ماژول torchvision دانلود می‌کنیم و به سه مجموعه آموزش، اعتبارسنجی و آزمایش تقسیم می‌کنیم و سپس 5 نمونه تصادفی را نشان می‌دهیم. شکل 2.1 این 5 نمونه تصادفی را نشان می‌دهد.

شکل 2.1. 5 نمونه تصادفی به همراه تصاویر آن‌ها

مشاهده می‌شود که دقت مدل pre-trained بر روی این مجموعه داده برابر با 99.92% می‌باشد. این دقت بسیار بالا منطقی است زیرا این مدل pre-trained، با همین مجموعه داده آموزش دیده‌است.

2) سپس به کمک کتابخانه cleverhans عکس مجموعه‌های آزمایشی را به تصاویر تخاصمی تبدیل می‌کنیم و سپس 5 نمونه از این تصاویر را نمایش می‌دهیم. در شکل 2.2، 5 نمونه تصویر اصلی به همراه تصاویر تخاصمی و تفاوت آنها نمایش داده شده‌است. برای وضوح بیشتر، تفاوت تصاویر تخاصمی و تصاویر اصلی به کمک min-max normalization نرمالیزه شده‌است تا تغییرات بهتر نمایان باشند.

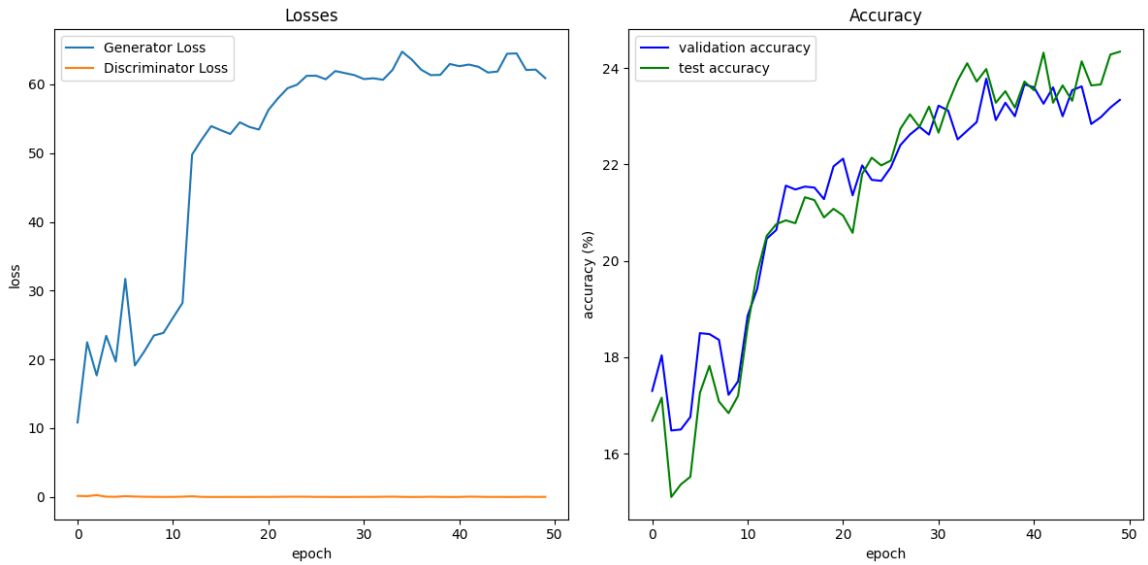


شکل 2.2. 5 نمونه تصویر تخاصمی به همراه مقایسه آن‌ها با تصاویر اصلی

همچنین، نرخ موفقیت حمله توسط این الگوریتم برابر با 31.86% گزارش می‌شود. به این معنی که اکنون تقریبا یک سوم از تصاویر مجموعه آزمایش به اشتباه برچسب‌گذاری می‌شوند.

3) اکنون مدل‌های Generator و Discriminator را مطابق با مقاله پیاده‌سازی می‌کنیم. با انجام این کار و آموزش مدل این AdvGan می‌بینیم که دقت بدست آمده، مطلوب نمی‌باشد. بنابراین به پیشنهاد تی‌ای مربوطه از مدل Generator و Discriminator شبکه pix2pix استفاده می‌کنیم. سپس یک کلاس AdvGan تشکیل می‌دهیم و تمام توابع مورد نیاز برای گام‌های بعد را در این کلاس پیاده‌سازی می‌کنیم.

در ادامه به پیشنهاد صورت سوال این مدل را به مقدار 50 epoch آموزش می‌دهیم. شکل 2.3 توابع هزینه و دقت طی این 50 epoch را نشان می‌دهد.



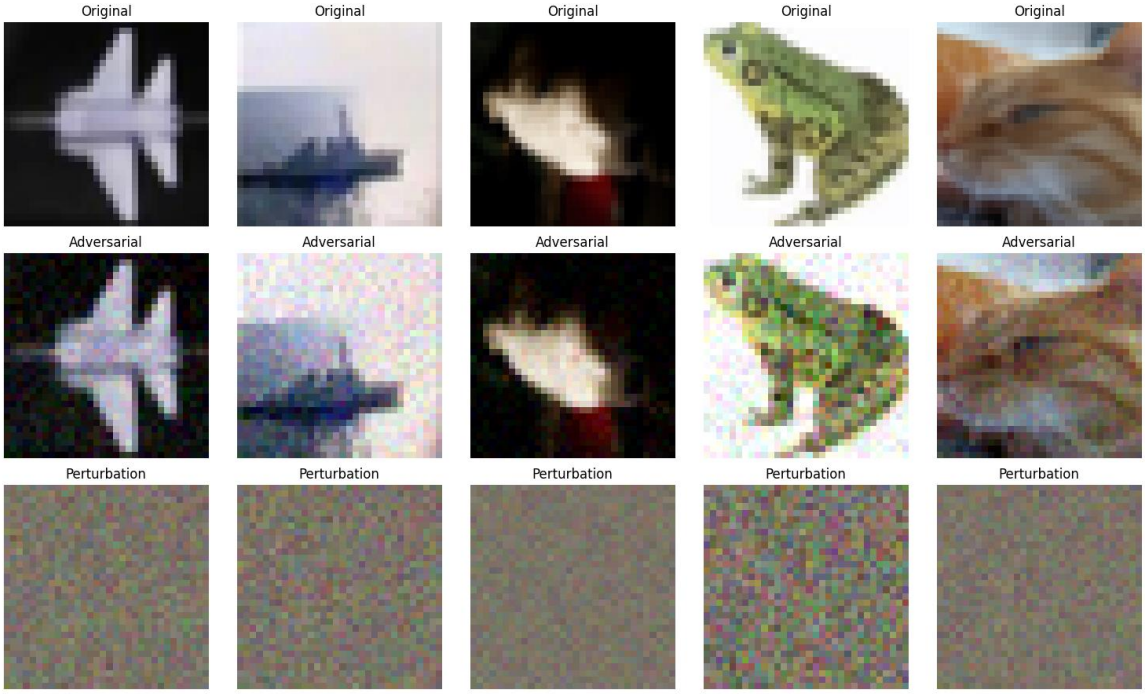
شکل 2.3. توابع هزینه و دقت برای اجزای متفاوت شبکه

4) در جدول 2.1، نرخ کلی خطا به همراه نرخ خطای هر کلاس نمایان شده‌است.

نام کلاس	نرخ خطا
کلاس اول (هواپیما)	61.43%
کلاس دوم (ماشین)	87.07%
کلاس سوم (پرنده)	70.98%
کلاس چهارم (گربه)	78.53%
کلاس پنجم (گوزن)	71.89%
کلاس ششم (سگ)	97.97%
کلاس هفتم (قورباغه)	58.87%
کلاس هشتم (اسب)	89.40%
کلاس نهم (کشتی)	55.56%
کلاس دهم (کامیون)	84.43%
میانگین	75.61%

جدول 2.1. نرخ موفقیت حمله برای هر کلاس و به صورت کلی

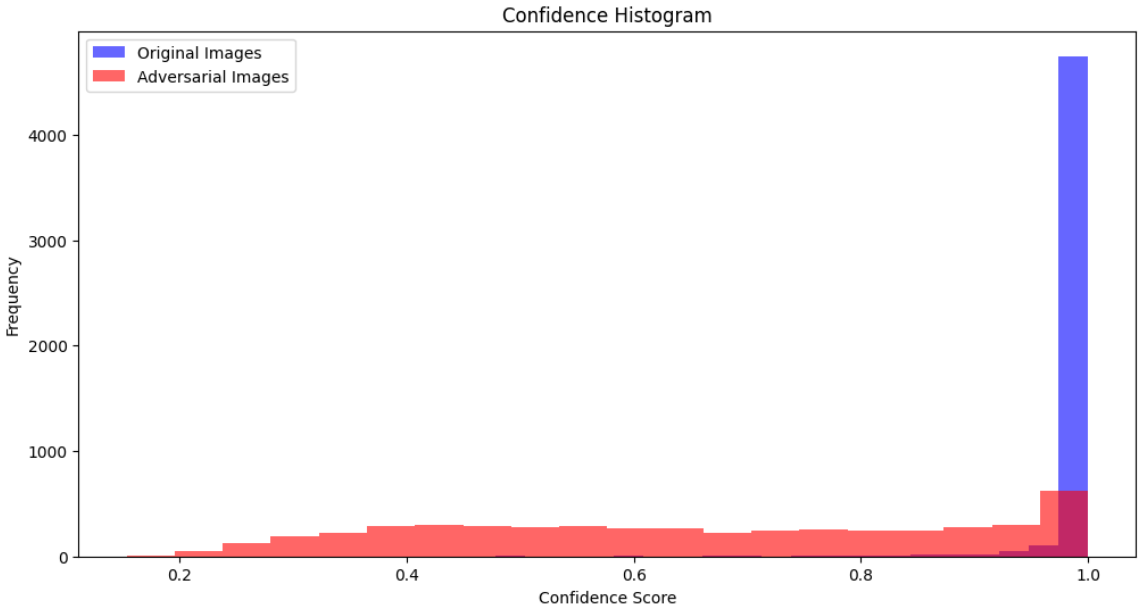
مشاهده می‌شود که در تمامی کلاس‌ها نتیجه بدست آمده از روش FGSM بهتر می‌باشد و به نتایجی مطلوب رسیده‌ایم. همچنین، در شکل 2.4 ما 5 نمونه از تصاویر تخاصمی به همراه تغییرات آن‌ها را نسبت به تصاویر اصلی بررسی می‌کنیم.



شکل 2.4. 5 نمونه از تصاویر تخاصمی به همراه تغییرات آن‌ها را نسبت به تصاویر اصلی

مشاهده می‌شود که این روش تصاویر را تار کرده‌است که مطلوب ما نیست. برای رفع این مشکل،به پیشنهاد دستیار آموزشی این تمرین اقدامات فراوانی از جمله تغییر معماری Generator و Discriminator، آموزش چندین باره Generator به ازای یکبار آموزش Discriminator و ... انجام داده اما هیچکدام این مشکل را برطرف نکردند.

همچنین، در انتها هیستوگرام قطعیت خواسته‌شده، این هیستوگرام در شکل 2.5 نمایان شده‌است.



شکل 2.5. هسیتوگرام قطعیت