|  |  |  |
| --- | --- | --- |
| **پرسش ۱** | **نام و نام خانوادگی** | حمیدرضا نادی مقدم |
| **شماره دانشجویی** | ۸۱۰۱۰۳۲۶۴ |
| **پرسش ۲** | **نام و نام خانوادگی** | علی صفری |
| **شماره دانشجویی** | 810202153 |
|  | **مهلت ارسال پاسخ** | **۱۴۰3.10.28** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس شبکه‌های عصبی و یادگیری عمیق**  **تمرین ششم** | | |

**فهرست**

[**مقدمه** 1](#_Toc187874170)

[**پرسش 1.** طراحی و پیاده‌سازی Triplet VAE برای تشخیص تومور در MRI 2](#_Toc187874171)

[**پرسش۲. AdvGAN** 13](#_Toc187874172)

[1-2. آشنایی با حملات خصمانه و معماری AdvGAN 13](#_Toc187874173)

[1-1-2. توضیح FGSM و PGD و مقایسه با AdvGAN 13](#_Toc187874174)

[2-1-2. تفاوت‌های کلیدی AdvGAN و GAN 17](#_Toc187874175)

[3-1-2. توضیح سه تابع هزینه استفاده شده در AdvGAN 19](#_Toc187874176)

[4-1-2. تفاوت حمله جعبه سفید و جعبه سیاه 22](#_Toc187874177)

[5-1-2. دو مقاله دیگر در حوزه AdvGAN 24](#_Toc187874178)

[2-2. پیاده‌سازی مدل AdvGAN 26](#_Toc187874179)

[1-2-2. دانلود و آماده‌سازی اولیه داده 26](#_Toc187874180)

[2-2-2. ایجاد تصاویر تخاصمی و نرخ حمله به مدل هدف 30](#_Toc187874181)

[3-2-2. پیاده‌سازی و آموزش مدل مولد و متمایزگر 34](#_Toc187874182)

[4-2-2. بررسی نرخ موفقیت حمله و هیستوگرام قطعیت 44](#_Toc187874183)

[5-2-2. پیاده‌سازی و بررسی مدل هدفدار (امتیازی) 49](#_Toc187874184)

**شکل‌ها**

شکل 1- بخشی از کد تعریف مدل هدف و ارزیابی 29

شکل 2- دانلود و تعریف مدل هدف و بررسی دقت آن روی مجموعه تست 29

شکل 3- کد مربوط به ایجاد نمونه‌های متخاصم 31

شکل 4- کد مربوط به بررسی نرخ موفقیت حمله 32

شکل 5- نرخ موفقیت حمله 32

شکل 6- نمایش تصاویر اولیه و تصاویر تخاصمی 33

**جدول‌ها**

[جدول 2- مقایسه و ارزیابی مدل‌ها با seq\_length=10 **Error! Bookmark not defined.**](#_Toc187874191)

# **مقدمه**

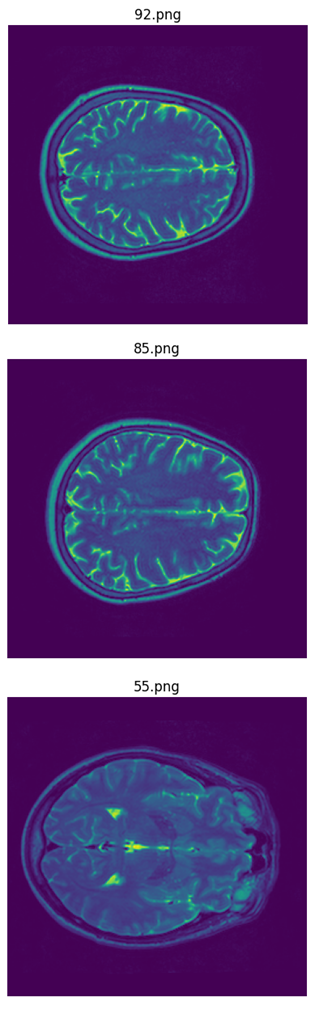
برای پیاده‌سازی پروژه از بستر Google Colab به منظور کد نویسی و اجرا استفاده شده است. تمامی مراحل کد و اجرای آن در این گزارش به تفصیل شرح داده شده است.

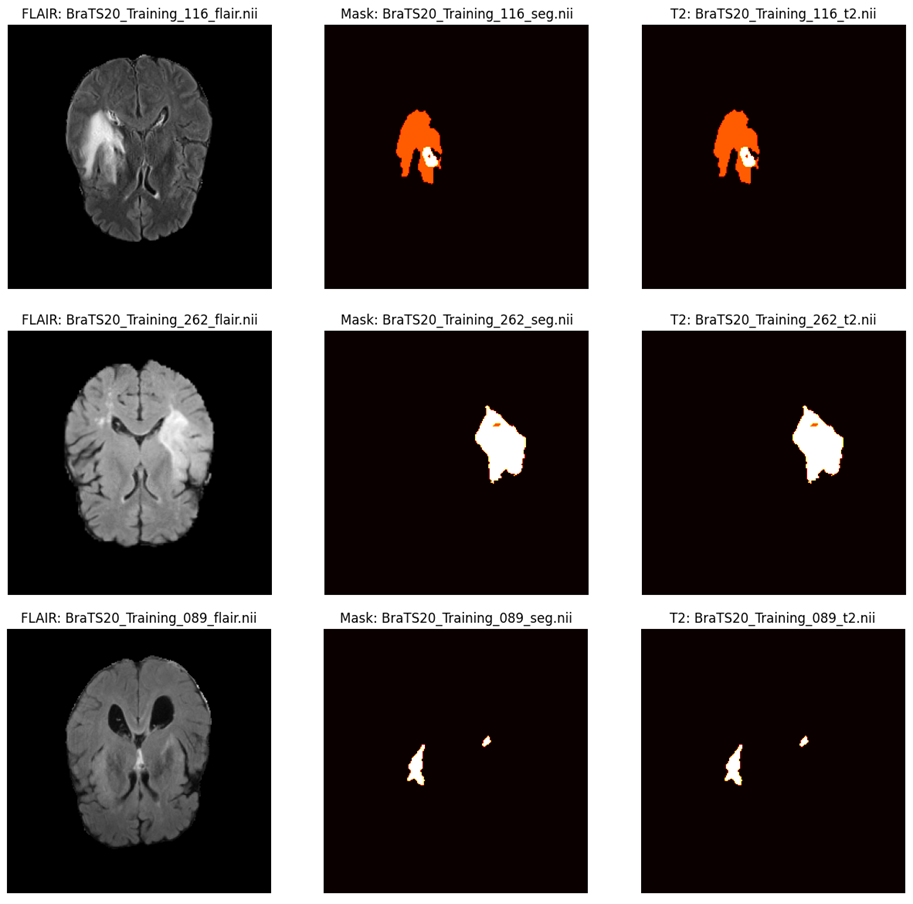
کد های نوشته شده همگی در پوشه‌ی Code و با پسوند .ipynd ذخیره شده است.

# **پرسش 1.** طراحی و پیاده‌سازی Triplet VAE برای تشخیص تومور در MRI

ابتدا هر دو دیتاست را از Kaggle بارگذاری کردیم و سپس چند تصویر نمونه از دیتاست IXI و ماسک تومور BraTS به همراه تصویر سگمنت شده آن را نمایش داده­ایم.

شکل 1 نمونه تصاویر از دیتاست IXI





شکل 2- نمونه تصاویر از دیتاست BrsTS با عکس اصلی و ماسک و T2

## ۱-2. پیاده­سازی یک VAE ساده

یک Variational Autoencoder (VAE) نوع خاصی از شبکه‌های عصبی مولد (Generative) است که برای مدل‌سازی داده‌های پیچیده استفاده می‌شود. هدف اصلی VAE یادگیری توزیع احتمال داده‌ها و تولید داده‌های جدید مشابه نمونه‌های اصلی است. در ادامه ایده کلی VAE توضیح داده شده است:

**قسمت Encoder و Decoder:**

Encoder: بخشی از VAE است که داده‌های ورودی (مثلاً تصاویر، متن یا داده‌های دیگر) را به یک فضای نهان (Latent Space) فشرده می­کند. این فضای نهان به صورت معمولاً دو بردار میانگین و انحراف معیار توزیع نهان است.

Decoder: بخش دیگر VAE است که تلاش می‌کند از فضای نهان نمونه‌برداری کند و داده‌های ورودی اصلی را بازسازی کند. به عبارت دیگر، Decoder داده‌های جدیدی تولید می‌کند که از نظر آماری به داده‌های ورودی شباهت دارند.

**Latent Space (فضای نهان):**

فضای نهان جایی است که داده‌های ورودی به یک نمایش فشرده نگاشت می‌شوند. به جای اینکه داده‌ها را به نقاط خاصی در فضای نهان نگاشت کنیم (مانند Autoencoder های معمولی)، VAE داده‌ها را به یک توزیع احتمال (معمولاً یک توزیع گاوسی) نگاشت می‌کند. این ویژگی امکان نمونه‌برداری از فضای نهان و تولید داده‌های جدید را فراهم می‌کند.

**KL Divergence (واگرایی کولبک-لیبلر):**

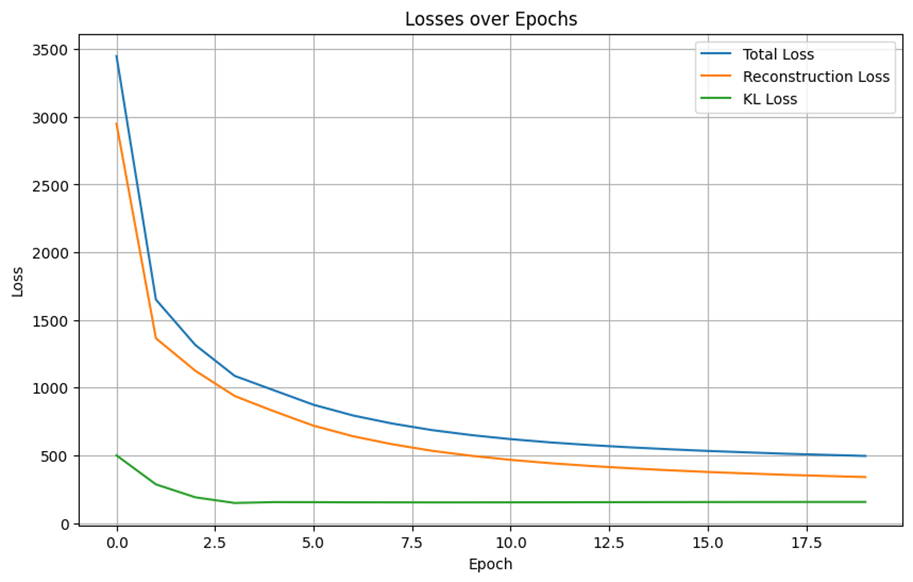
KL Divergence یک معیار برای اندازه‌گیری تفاوت بین دو توزیع احتمال است. در VAE از KL Divergence برای اطمینان از این استفاده می‌شود که توزیع نهان نزدیک به یک توزیع هدف باشد. تابع هزینه VAE از دو بخش تشکیل شده است: خطای بازسازی (Reconstruction Loss) که اندازه‌گیری تفاوت بین داده‌های اصلی و داده‌های بازسازی‌شده توسط Decoder است. و واگرایی KL که توزیع نهان را به توزیع گاوسی استاندارد نزدیک می‌کند.

خلاصه که در VAE داده‌های ورودی به کمک Encoder به یک توزیع احتمال نهان نگاشت می‌شوند. از این توزیع نمونه‌برداری شده و بردار نهان z تولید می‌شود. Z به کمک Decoder بازسازی می‌شود تا داده‌ای شبیه به x تولید شود. مدل با کمینه کردن ترکیبی از خطای بازسازی و KL Divergence آموزش می‌بیند.

در این مدل ما ابتدا تصاویر را برای ورودی به VAE آماده کردیم و انکدر و دیکدر را با 4 لایه و 3 لایه و مجموعه دو تابع هزینه KL Divergence و MSE Reconstruction Loss پیاده سازی کردیم. در این پیاده سازی ما ابعاد فضای نهان را برابر 256 گرفته­ایم و سایر پارامترها را نیز تنظیم کرده­ایم.

**پیاده سازی:**

در پیاده سازی این مدل از شبکه عصبی VAE در قسمت Encoder که وظیفه استخراج ویژگی‌ها از ورودی (تصویر) و تولید دو بردار میانگین (μ) و واریانس (σ) را دارد. این بخش از چند لایه کانولوشنی با فیلترهای افزایشی، نرمال‌سازی دسته‌ای (BatchNorm)، و تابع فعال‌ساز ReLU استفاده کرده­ایم. و در قسمت Decoder که وظیفه بازسازی تصویر اولیه از بردار کدگذاری‌شده (بردار نهفته z) را دارد. این بخش نیز شامل لایه‌های کانولوشنی ترانهاده (Transposed Convolution) همراه با نرمال‌سازی دسته‌ای و فعال‌سازی ReLU است. خروجی نهایی از یک تابع tanh عبور کرده است. و در آخر VAE، دو بخش Encoder و Decoder را ترکیب می‌کند و با استفاده از روش نمونه‌گیری مجدد، بردار z را تولید می‌کند. و بعلاوه بر بازسازی، این مدل شامل یک تابع هزینه KL Divergence Loss بعلاوه MSE نیز می­باشد. که هر سه به تفکیک در هر ایپاک چاپ داده شده­اند و نمودار آن ترسیم شده است.



شکل 3 - نمودار مقادیر loss برای هر سه معیار

**تست روی داده­های BraTS:**

بعد از آموزش این مدل از VAE روی دیتاست سالم IXI با 20 دوره و تست آن با چند اسلایس توموری از دیتاست BraTS که نمونه بازسازی را نمایش داده­ایم. معیار Dice نیز پیاده­سازی و برای هر تصویر نمایش داده شده است. تصاویر بازسازی با توجه به اینکه داده­های تست را ندیده است قابل قبول است ولی تصویر ماسک را نمی تواند بازسازی کند. دلیل آن این است که روی داد­ه­های سالم آموزش دیده است و روی داده توموری تست انجام داده ایم که باعث می­شود مدل درست تومورها را یاد نگیرد.



شکل 4 - چند نمونه از تست باسازی عکس های مدل VAE ساده

۱-3. پیاده سازی Tri-VAE

برای پیاده‌سازی مدل Tri-VAE با جزئیات مورد نظر، مراحل زیر را انجام می­دهیم:

**آماده‌سازی ورودی‌ها:** سه دسته ورودی (Anchor, Positive, Negative) آماده می‌شوند. که Anchor و Positive تصاویر سالم بدون نویز و Negative تصاویر سالم همراه با نویز Coarse است.

**معماری مدل:** Encoder ورودی‌های که تصاویر هستند را دریافت می‌کند. و معماری شامل لایه‌های کانولوشن، نرمال­کردن و تابع فعال ساز است. خروجی ها تعبیه‌های ، و است.

Decoder برای بازسازی تصاویر استفاده می­شود. بازسازی در دو مقیاس Coarse Scale با خروجی 32\*32 و Full Scale با خروجی 256\*256 است. و برای اتصال Encoder و Decoder از Gated Cross Skip Connections (GCS) استفاده می­کنیم.

**تابع هزینه:**

Triplet Loss:

L1 Loss (Coarse): خطای بازسازی در مقیاس coarse برای Anchor، Positive، و Negative.

L1 Loss (Full): خطای بازسازی در مقیاس full فقط برای Negative.

KL Divergence Loss: اختلاف بین توزیع تعبیه‌های Anchor و Positive.

SSIM Loss: کیفیت بازسازی Negative در مقیاس full.

## ۱-4. ارزیابی در دیتاست BraTS (دو بعدی)

با توجه به مقاله و آموزش روی دیتاست سالم، برای تست با استفاده از دیتاست BrasTS که تومورهای مغزی را دارند باید عکس­های باسازی کند و سپس قسمت آنومالی آن را تشخیص دهد. لذا بعد از آموزش مدل، 10 بیمار از دیتاست BraTS انتخاب شدند. به منظور تمرکز بر بخش‌های آنومالی، اسلایس‌های دو‌بعدی استخراج و برای ارزیابی مدل Tri-VAE استفاده شدند. پس از اعمال مدل Tri-VAE بر روی تصاویر ورودی، تصاویر بازسازی‌شده دریافت شدند. برای ارزیابی دقت مدل در بازسازی، خطای بازسازی محاسبه شد که در آن تصویر ورودی و تصویر بازسازی‌شده است. این محاسبه به ما امکان می‌دهد تا میزان اختلاف بین تصویر ورودی و تصویر بازسازی‌شده توسط مدل را اندازه‌گیری کنیم. و همچنین از یک Threshold (مقدار آستانه)، مانند 0.1، برای کاهش نویز و بهبود دقت مدل استفاده شد. برای ارزیابی دقت پیش‌بینی‌های مدل، از معیار Dice نیز استفاده کردیم.

برای ارزیابی کیفی نتایج، چند نمونه از تصاویر ورودی، پیش‌بینی‌های آنومالی و ماسک‌های واقعی نمایش داده شد. این تصاویر به ما کمک می‌کنند تا عملکرد مدل را مشاهده کنیم.

مدل VAE دارای معماری ساده‌تر نسبت به Tri-VAE است و قادر به بازسازی تصاویر مغزی است. و دارای ویژگی‌های مناسبی برای داده‌های پزشکی می­باشد.

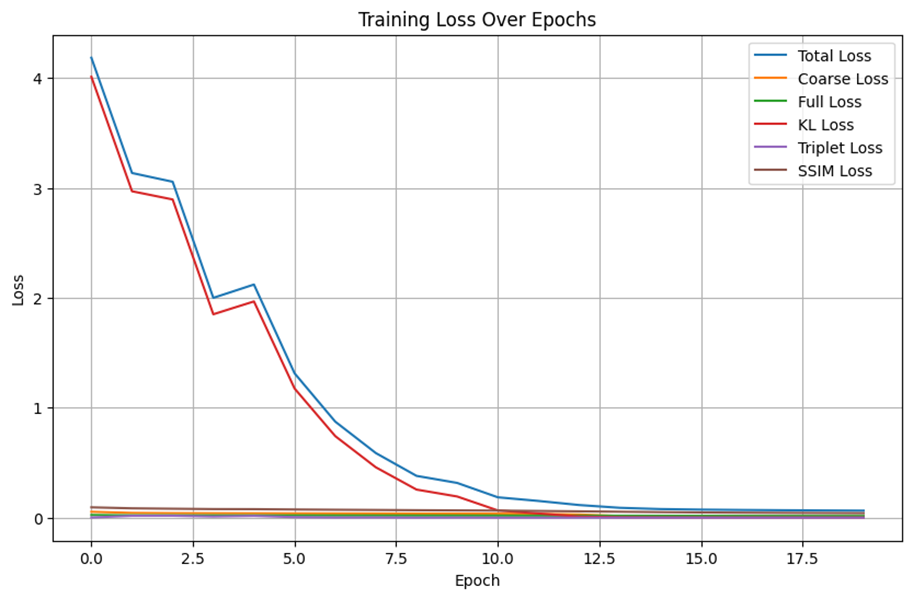
مدل Tri-VAE نسخه‌ای پیشرفته‌تر از VAE است که به‌طور خاص برای پردازش تصاویر پیچیده‌تر و سه‌بعدی (مانند تصاویر MRI مغز) طراحی شده است. این مدل شامل چندین مولفه و بخش برای یادگیری ویژگی‌ها در مقیاس‌های مختلف است. در نتیجه، Tri-VAE قادر به یادگیری ویژگی‌های پیچیده‌تری است که ممکن است در مدل ساده VAE قابل شبیه‌سازی نباشند. از ویژگی­های این مدل می­توان به معماری پیچیده‌تر و شامل بخش‌های متعددی برای پردازش بهتر داده‌های سه‌بعدی و ویژگی‌های چند مقیاسی است و عملکرد بهتری در بازسازی تصاویر و پیش‌بینی نواحی آنومالی، به‌ویژه در مقیاس‌های پیچیده‌تر دارد. و نسیت به مدل VAE بیشتر برای داده‌های پزشکی پیچیده مانند تصاویر مغزی مناسب است.

مدل VAE معمولاً قادر است تصاویر مغزی را بازسازی کند، اما در بخش‌هایی که ویژگی‌های آنومالی یا تومور مغزی وجود دارند، دقت خوبی ندارد. و در تشخیص مرزهای دقیق تومورها یا نواحی آنومالی با مشکل مواجه می­شود، زیرا معماری ساده‌تری دارد. در تصاویر با نویز زیاد یا پیچیدگی‌های ساختاری بیشتر، بازسازی‌ها ممکن است کیفیت مطلوب را نداشته باشند.

در مقایسه با VAE مدل Tri-VAE به دلیل داشتن معماری پیچیده‌تر، قادر است تصاویر مغزی را با دقت بالاتری بازسازی کند و در شبیه‌سازی نواحی آنومالی (تومورها) بهتر عمل کند. این مدل در شبیه‌سازی مرزهای تومورها کمی بهتر عمل می­کند و می‌تواند ویژگی‌های پیچیده‌تری از تصاویر MRI را شبیه‌سازی کند. از آنجا که Tri-VAE معماری چند مقیاسی دارد، قادر است با جزئیات بیشتری ویژگی‌های آنومالی‌های کوچک‌تر را شبیه‌سازی کند که ممکن است در VAE آموزش نگیرد.

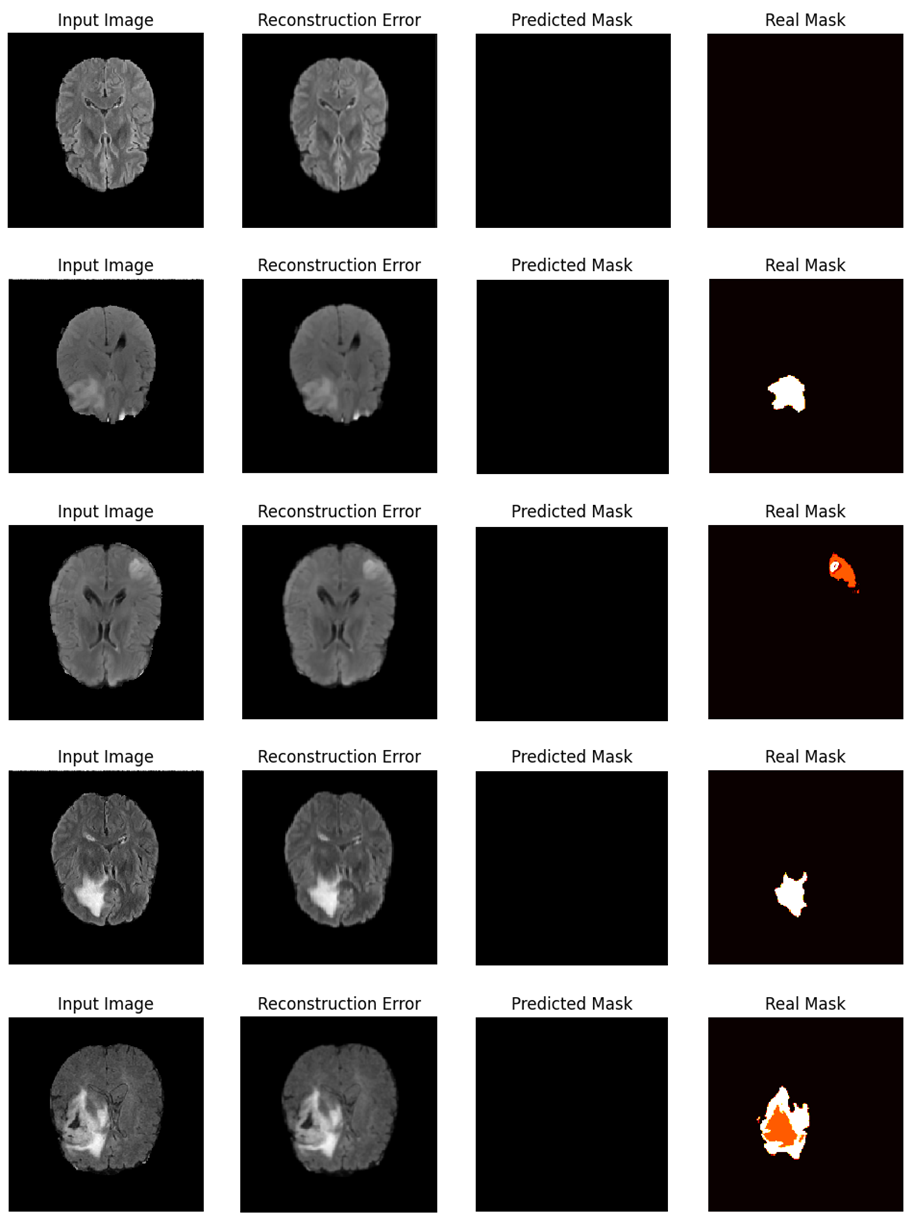
مدل VAE برای پروژه‌های ساده‌تر استفاده می­شود و دارای سرعت بالاتر و هزینه‌های پردازش کمتری دارد، ولی در بازسازی دقیق تصاویر پیچیده‌تر و پیش‌بینی نواحی آنومالی دقت کمتری دارد. و مدل Tri-VAE به دلیل معماری پیچیده‌تر و قابلیت‌های پردازش چند مقیاسی، عملکرد بهتری در بازسازی تصاویر و پیش‌بینی نواحی آنومالی، به‌ویژه در تصاویر مغزی پیچیده و با جزئیات بالا، دارد. این مدل قادر است نواحی آنومالی دقیق‌تری را شبیه‌سازی کرده و در مقایسه با VAE، نتایج بهتری به‌دست آورد.

با وجود اینکه تعداد ایپاک‌ها در مدل Tri-VAE محدود به 20 ایپاک بود و پیش‌پردازش‌ها در این پروژه ساده نگه‌داشته شدند، نتایج نشان‌دهنده‌ی عملکرد قابل‌قبول مدل در بازسازی تصاویر و پیش‌بینی نواحی آنومالی بود. با این حال، دقت مدل در مقایسه با نتایج مقاله‌های مشابه که از پردازش‌های سه‌بعدی پیچیده‌تر و تعداد ایپاک‌های بیشتر استفاده می‌کنند، پایین‌تر بود. این امر به‌طور عمده به دلیل محدودیت‌های تعداد ایپاک‌ها و نداشتن پس‌پردازش‌های پیچیده‌تر است.



شکل 5 - نمودار Loss های مدل Tri-VAE

در این پروژه، مدل Tri-VAE در مقایسه با مدل ساده VAE عملکرد بهتری در بازسازی تصاویر و پیش‌بینی نواحی آنومالی داشت. با این حال، برای بهبود دقت مدل و نزدیک‌تر شدن به نتایج مقاله‌ها، استفاده از تعداد ایپاک‌های بیشتر و اعمال پیش‌پردازش‌های پیچیده‌تر همچنان ضروری است. در نهایت، تحلیل دقیق نتایج و ارزیابی مدل از اهمیت بالایی برخوردار است و ساختار طراحی مدل باید بر اساس نیازهای خاص پروژه تنظیم شود. در بازسازی تصاویر نسبت به مدل VAE ساده خیلی بهتر کار کرده است ولی در تشخیص تومور همچنان مشکل دارد و البته در بعضی از موارد تشخیص آن به نسبت بیشتر از تصاویر در گزارش بود که متاسفانه چون به صورت رندوم عکس انتخاب شده است و بعد از چند اجرای مجدد نتوانستم خروجی بگیرم که تصاویر به ماسک بهتر را نمایش بدهد.



شکل 6- نمونه ای از تصاویر برای تست مدل Tri-VAE

## ۱-5. بخش امتیازی

پردازش داده‌های تصویربرداری پزشکی با مدل Tri-VAE، حذف نویز، و مقایسه دو روش Coarse Noise و Simplex Noise بر اساس شاخص Dice.

مراحل اصلی:

تولید داده مصنوعی: حجم سه‌بعدی باینری به‌عنوان Ground Truth.

Tri-VAE: پردازش اسلایس‌ها و ترکیب به حجم سه‌بعدی.

حذف نویز: فیلتر Median سه‌بعدی و حذف کامپوننت‌های کوچک.

نویز Simplex: اعمال نویز Simplex و مقایسه با Coarse Noise.

محاسبه Dice: ارزیابی عملکرد برای هر دو روش.

این روش را تا حدودی انجام شد ولی به دلیل کمبود سخت افزار محاسباتی آن را نتوانسیم اجرا کنیم و خطای آن را برطرف کنیم و به یک کلیتی از آن بسنده کردیم.

# **پرسش۲. AdvGAN**

## 1-2. آشنایی با حملات خصمانه و معماری AdvGAN

### 1-1-2. توضیح FGSM و PGD و مقایسه با AdvGAN

1. **روش FGSM (Fast Gradient Sign Method)**:
   * **شرح**: **FGSM** یک حمله **جعبه سفید** است که از گرادیان تابع خطا نسبت به ورودی داده‌ها استفاده می‌کند تا تغییرات کوچک و هدفمند به ورودی وارد کند. این تغییرات به گونه‌ای اعمال می‌شوند که مدل نتواند پیش‌بینی صحیحی انجام دهد. در این روش، حمله به طور مستقیم به پارامترهای مدل وابسته است.
   * **فرمول**

x′=x+ϵ⋅sign(∇x​J(θ,x,y))

* + - در اینجا:
      * x ورودی اصلی
      * x′ نمونه خصمانه
      * J(θ,x,y) تابع خطا (loss function)
      * ϵ میزان تغییرات
  + **نوع حمله**: **جعبه سفید** (White-box attack)
  + **مزایا**:
    - ساده و سریع برای ارزیابی‌های اولیه
    - به راحتی قابل پیاده‌سازی
  + **محدودیت‌ها**:
    - به دلیل سادگی، برای حملات پیچیده و مدل‌های مقاوم ناکارآمد است.
    - معمولاً نمی‌تواند در برابر روش‌های دفاعی مانند آموزش خصمانه مقاومت کند.

1. **روش PGD (Projected Gradient Descent)**:
   * **شرح**: **PGD** یک نسخه پیشرفته‌تر و تکراری از **FGSM** است که در هر مرحله تغییرات کوچکی را اعمال کرده و سپس نتیجه را به محدوده‌ای خاص برمی‌گرداند. این کار باعث می‌شود که حمله بیشتر به ساختار ورودی بپردازد و اثرات آن تدریجاً بیشتر شود.
   * **فرمول**

xt+1′​=Clipx,ϵ​(xt​+α⋅sign(∇x​J(θ,xt​,y)))

* + - در اینجا:
      * xt ورودی در مرحله t
      * α اندازه گام در هر مرحله
      * Clipx,ϵ اطمینان حاصل می‌کند که تغییرات در محدوده تعریف شده باقی بمانند.
  + **نوع حمله** : **جعبه سفید** (White-box attack)
  + **مزایا**:
    - نرخ موفقیت بالاتر نسبت به **FGSM** به دلیل استفاده از تکرار در مراحل مختلف
    - مقاوم‌تر در برابر مدل‌های پیچیده و دفاع‌های اولیه
  + **محدودیت‌ها**:
    - زمان‌بر و نیازمند محاسبات بیشتر به دلیل تکرار در هر مرحله
    - در حملات زمان‌بر و پیچیده می‌تواند ناکارآمد باشد

1. **مدل AdvGAN (Adversarial Generative Adversarial Networks)**:
   * **شرح**: **AdvGAN** از معماری شبکه‌های مولد متخاصم (GAN) برای تولید نمونه‌های خصمانه استفاده می‌کند. این مدل یک تولید کننده (Generator) و یک تمایزگر (Discrimintor) دارد که به طور همزمان تلاش می‌کنند تا نمونه‌هایی تولید کنند که برای مدل هدف فریبنده باشند، در حالی که کیفیت بصری بالا و شباهت زیادی به نمونه‌های اصلی دارند.
   * **فرمول**

L=Lfadv+αLGAN+βLhinge ​

* + - در اینجا:
      * Ladv​ تابع خطای حمله
      * LGAN​ تابع خطای GAN برای حفظ کیفیت بصری
      * Lhinge​ تابعی برای محدود کردن اندازه تغییرات

در بخش 2-1-3 به تفصیل در رابطه با تابع هزینه و کارکرد هر بخش توضیح داده شده است.

* + **نوع حمله**: **جعبه سفید** و **جعبه سیاه** (White-box and Black-box attacks)
  + **مزایا**:
    - **کیفیت بصری بالا**: نمونه‌های خصمانه تولید شده توسط **AdvGAN** بسیار شبیه به تصاویر اصلی هستند و تشخیص آن‌ها دشوار است.
    - **کارایی بالا**: پس از آموزش Generator، می‌توان نمونه‌های خصمانه را به سرعت تولید کرد بدون اینکه نیازی به دسترسی به مدل هدف باشد.
    - **مقاومت در برابر دفاع‌ها**: این روش می‌تواند در برابر روش‌های دفاعی مختلف، مانند آموزش خصمانه، عملکرد بهتری داشته باشد.
    - **حملات جعبه سیاه**: **AdvGAN** قادر است حملات را در شرایط **جعبه سیاه** نیز انجام دهد، که برای بسیاری از روش‌های دیگر مانند **FGSM** و **PGD** چالش‌برانگیز است.
  + **محدودیت‌ها**:
    - نیاز به آموزش اولیه Generator
    - پیچیدگی بیشتر در مقایسه با **FGSM** و **PGD**

**مقایسه AdvGAN با FGSM و PGD**

1. **حملات جعبه سفید**:
   * **FGSM** و **PGD** تنها در شرایط **جعبه سفید** به خوبی کار می‌کنند، جایی که دسترسی کامل به پارامترهای مدل وجود دارد. این روش‌ها در تولید نمونه‌های خصمانه برای مدل‌های خاص موفق هستند، اما به دلیل سادگی، نمی‌توانند به راحتی مدل‌های پیچیده را تحت تاثیر قرار دهند.
   * **AdvGAN** نیز در **جعبه سفید** می‌تواند به خوبی عمل کند و با استفاده از شبکه‌های مولد متخاصم، نمونه‌های خصمانه‌ای با کیفیت بصری بالا تولید کند که برای مدل‌های مقاوم نیز کارایی دارند.
2. **حملات جعبه سیاه**:
   * در حملات **جعبه سیاه**، جایی که اطلاعاتی از مدل هدف در دسترس نیست، **FGSM** و **PGD** معمولاً نمی‌توانند به خوبی عمل کنند. این حملات نیازمند استفاده از **انتقال** یا استفاده از مدل‌های کمکی هستند.
   * **AdvGAN** با استفاده از روش **مدل‌سازی مجدد** (distillation)، قادر است حملات **جعبه سیاه** را به طور مؤثر انجام دهد. این ویژگی باعث می‌شود که **AdvGAN** نسبت به **FGSM** و **PGD** در این زمینه مزیت داشته باشد.
3. **کیفیت بصری**:
   * **FGSM** و **PGD** به دلیل سادگی و استفاده از تغییرات مستقیم در داده‌ها، معمولاً باعث می‌شوند نمونه‌های خصمانه با تغییرات آشکار در تصویر ایجاد شوند.
   * **AdvGAN** با استفاده از معماری **GAN** می‌تواند تغییرات بسیار ظریف‌تری اعمال کند که به راحتی قابل تشخیص برای انسان‌ها نیستند و تصویر همچنان طبیعی باقی می‌ماند.
4. **زمان تولید**:
   * **FGSM** به دلیل سادگی خود بسیار سریع است و زمان کمی برای تولید نمونه‌های خصمانه نیاز دارد.
   * **PGD** به دلیل تکرار مراحل، زمان بیشتری نسبت به **FGSM** می‌برد.
   * **AdvGAN** پس از آموزش، قادر است نمونه‌های خصمانه را به سرعت تولید کند، که این ویژگی آن را برای حملات سریع و گسترده مناسب می‌سازد.

### 2-1-2. تفاوت‌های کلیدی AdvGAN و GAN

* **چگونه AdvGAN از گرادیان‌ها یا خروجی‌های مدل هدف در زمان آموزش استفاده می‌کند؟**
* **AdvGAN** از یک **مدل هدف** (Target Model) در فرآیند آموزش استفاده می‌کند. این مدل هدف می‌تواند یک مدل طبقه‌بندی (مثل مدل‌های شناسایی تصویر) باشد. در طول آموزش **AdvGAN**، **Generator** و **Discriminator** معمولاً همانطور که در **GAN ساده** انجام می‌شود، آموزش داده می‌شوند، اما **AdvGAN** از اطلاعات بیشتری استفاده می‌کند.
* در این روش، علاوه بر **Discriminator** که به **Generator** کمک می‌کند تا نمونه‌هایی تولید کند که به نظر واقعی بیایند، **AdvGAN** از **گرادیان‌ها** یا **خروجی‌های مدل هدف** برای هدایت **Generator**  به سمت تولید **نمونه‌های خصمانه** استفاده می‌کند.
* در واقع، **Generator** از **گرادیان‌های مدل هدف** استفاده می‌کند تا تغییراتی در ورودی‌ها اعمال کند که باعث می‌شود مدل هدف دچار خطا شود و نتواند پیش‌بینی درستی انجام دهد. این به معنای این است که **Generator** نه تنها باید **Discriminator** را فریب بدهد، بلکه باید **مدل هدف** را نیز فریب دهد و آن را به اشتباه بیاندازد.
* به طور خاص، در هر مرحله از آموزش، **AdvGAN** به **Generator** این اطلاعات را می‌دهد که چگونه باید تغییرات را اعمال کند تا مدل هدف اشتباهات بیشتری داشته باشد. به عبارت دیگر، **AdvGAN** از **خروجی‌ها** و **گرادیان‌ها**ی مدل هدف برای اصلاح **Generator** و تولید نمونه‌هایی استفاده می‌کند که **نه تنها واقعی به نظر می‌آیند، بلکه مدل هدف را نیز به اشتباه می‌اندازند**.
* **چگونه AdvGAN نمونه‌های متخاصم تولید می‌کند و چگونه این مدل قادر است همزمان وفاداری بصری به تصویر اصلی و قابلیت حمله به مدل را حفظ کند؟**
* **AdvGAN** به طور همزمان دو هدف را دنبال می‌کند:
  1. **حفظ وفاداری بصری به تصویر اصلی**: این هدف از طریق**Discriminator** انجام می‌شود. **Discriminator** به **Generator** کمک می‌کند تا تغییرات اعمال‌شده به تصویر اصلی آنقدر کوچک و نامحسوس باشد که تصویر تولیدشده تقریباً مشابه تصویر اصلی باقی بماند. به این ترتیب، **AdvGAN** اطمینان حاصل می‌کند که نمونه‌های تولیدشده از نظر بصری قابل تشخیص نباشند و برای انسان‌ها غیرقابل شناسایی باشند.
  2. **حمله به مدل هدف**: در عین حال، **Generator** با استفاده از **گرادیان‌های مدل هدف** به گونه‌ای آموزش می‌بیند که نمونه‌های تولیدشده به گونه‌ای طراحی شوند که مدل هدف نتواند آن‌ها را به درستی شبیه‌سازی کند. این یعنی تصویر تولید شده باید باعث اشتباه مدل هدف شود، یعنی مدل هدف باید پیش‌بینی نادرستی انجام دهد.
* **چگونگی ایجاد نمونه‌های متخاصم**:
  1. در ابتدا، **Generator** یک تصویر **اصلی** را می‌گیرد و یک**Perturbation** (تغییرات کوچک) به آن اضافه می‌کند.
  2. سپس این تصویر تغییر یافته به **Discriminator** داده می‌شود که ارزیابی می‌کند که آیا این تصویر از نظر بصری شبیه به نمونه واقعی است یا خیر.
  3. در کنار این، تصویر به **مدل هدف** داده می‌شود و بررسی می‌شود که آیا این تصویر فریب‌دهنده است و مدل هدف نمی‌تواند آن را به درستی دسته‌بندی کند.
  4. هدف **Generator** این است که تصویر تولیدی نه تنها **به نظر واقعی** بیاید (از نظر **Discriminator**)، بلکه در عین حال **مدل هدف را فریب دهد** (از نظر **مدل هدف**).
* **چگونه وفاداری بصری و حمله به مدل همزمان حفظ می‌شود**:
  1. در **AdvGAN**، دو نوع **loss** به صورت همزمان وجود دارد:
     1. **loss** برای ارزیابی مشابهت بصری: این قسمت از فرآیند به **Discriminator** مربوط می‌شود و به **Generator** کمک می‌کند که تصویر تولید شده به نظر واقعی برسد.
     2. **loss** برای ارزیابی موفقیت حمله: این **loss** از **مدل هدف** به دست می‌آید و به **Generator** کمک می‌کند که نمونه تولیدی باعث اشتباه مدل هدف شود.
  2. ترکیب این دو **loss** باعث می‌شود که **AdvGAN** به صورت همزمان **وفاداری بصری** و **موفقیت در حمله** را حفظ کند. در واقع، **Generator** تلاش می‌کند که تصویر تولیدی تا حد ممکن به تصویر اصلی شبیه باشد با کمک **Discriminator**، اما در عین حال باید مدل هدف را فریب دهد با کمک **مدل هدف.**
* **در نتیجه**، **AdvGAN** قادر است نمونه‌های متخاصم تولید کند که نه تنها به طور مؤثری مدل هدف را فریب دهند، بلکه از نظر بصری هم شبیه به تصاویر اصلی باقی بمانند. این ویژگی آن را از روش‌های دیگر متمایز می‌کند که معمولاً نیاز به توازن بین حمله مؤثر و حفظ کیفیت بصری ندارند.

### 3-1-2. توضیح سه تابع هزینه استفاده شده در AdvGAN

نمونه متن

**1. GAN Loss (LGAN)**

این تابع ضرر به‌طور عمده برای اطمینان از واقعی بودن بصری نمونه‌های adversarial تولید شده استفاده می‌شود و اطمینان می‌دهد که این نمونه‌ها از داده‌های واقعی قابل تمیز دادن نباشند. این تابع ضرر به تولید perturbation‌هایی کمک می‌کند که به‌طور طبیعی با داده‌های اصلی هماهنگ باشند و از نظر بصری مشابه آنها باشند.

**رابطه ریاضی:**

LGAN​=Ex​[logD(x)]+Ex​[log(1−D(x+G(x)))]

جایی که:

* xداده واقعی از کلاس اصلی است.
* G(x) perturbation تولید شده توسط شبکه Generator برای ورودی x است.
* x+G(x) نمونه adversarial است.
* D(x) خروجی شبکه Discriminator زمانی است که تصویر اصلی x به آن داده می‌شود و احتمال اینکه x داده واقعی باشد را پیش‌بینی می‌کند.
* D(x+G(x)) پیش‌بینی احتمال Discriminator است که آیا نمونه adversarial x+G(x) واقعی است.

**توضیح:**

این تابع ضرر کمک می‌کند تا Generator نمونه‌های adversarial تولید کند که از نظر بصری مشابه داده‌های واقعی باشند و Discriminator را فریب دهند. به این ترتیب، Generator به تولید perturbation‌هایی هدایت می‌شود که به‌طور طبیعی با داده‌های اصلی هم‌راستا هستند و حملات adversarial مؤثری تولید می‌کند.

**2. Adversarial Loss (Lfadv)**

این تابع ضرر به‌طور خاص برای ایجاد نمونه‌های adversarial طراحی شده است که مدل هدف f را به اشتباه بیندازند. این تابع ضرر به این صورت عمل می‌کند که اطمینان می‌دهد نمونه‌های adversarial به درستی مدل را گمراه می‌کنند، خواه هدف حمله هدفمند باشد یا حمله غیرهدفمند.

**رابطه ریاضی:**

Lfadv=Ex[Lf(x+G(x),t)]

جایی که:

* tکلاس هدف برای حمله هدفمند است.
* x+G(x) نمونه adversarial است.
* f مدل هدف است.
* Lf(x+G(x),t) تابع ضرر استفاده شده توسط مدل f است (مثلاً ضرر کراس-انتروپی) که فاصله بین پیش‌بینی مدل برای x+G(x) و کلاس هدف t را اندازه‌گیری می‌کند.

**توضیح:**

این تابع ضرر اطمینان می‌دهد که نمونه‌های adversarial تولید شده به درستی مدل هدف را گمراه کنند. در حملات هدفمند، این ضرر باعث می‌شود که پیش‌بینی مدل به کلاس هدف نزدیک‌تر شود. در حملات غیرهدفمند، هدف این است که مدل بیشتر دچار اشتباه شود. این ویژگی برای موفقیت حملات adversarial ضروری است.

**3. Hinge Loss (Lhinge)**

این تابع ضرر به‌طور خاص برای تنظیم اندازه perturbation استفاده می‌شود و اطمینان می‌دهد که نمونه‌های adversarial مشابه تصاویر اصلی باقی بمانند. این تابع ضرر به Generator کمک می‌کند تا محدودیت‌هایی برای اندازه perturbation تعیین کند تا نمونه‌های adversarial بزرگ یا آشکار نشوند.

**رابطه ریاضی:**

Lhinge​=Ex​[max(0,∥G(x)∥2​−c)]

جایی که:

* G(x) perturbation تولید شده توسط Generator برای ورودی xxx است.
* c یک حد مشخص شده توسط کاربر برای norm-2L از perturbation است.

**توضیح:**

این تابع ضرر وقتی که اندازه G(x) perturbation از حد مجاز c بیشتر شود،Generator را جریمه می‌کند. با این کار، Generator هدایت می‌شود تا perturbation‌هایی تولید کند که از نظر بصری شبیه به داده‌های اصلی باقی بمانند و حملات را کمتر قابل شناسایی کنند.

**تابع ضرر کلی برای AdvGAN**

تابع ضرر نهایی در AdvGAN ترکیبی از این سه تابع ضرر است که با پارامترهای α و β وزن‌دهی شده‌اند:

L=Lfadv+αLGAN+βLhinge ​

جایی که:

* α و β کنترل می‌کنند که هر کدام از این ضررها چقدر اهمیت دارند.
* تابع ضرر کلی با هدف تولید نمونه‌های adversarial بهینه‌سازی می‌شود که نه تنها از نظر بصری واقعی و مشابه داده‌های اصلی هستند، بلکه مدل هدف را به‌طور مؤثر فریب می‌دهند و perturbation‌ها در حد قابل قبولی باقی می‌مانند.

**نقش هر تابع ضرر در AdvGAN**

* **GAN Loss**: این تابع ضرر به Generator کمک می‌کند تا نمونه‌های adversarial تولید کند که از نظر بصری مشابه داده‌های واقعی باشند و به طور مؤثر Discriminator را فریب دهند.
* **Adversarial Loss**: این تابع ضرر به Generator کمک می‌کند تا نمونه‌های adversarial تولید کند که مدل هدف را به اشتباه بیندازند و حمله را موفقیت‌آمیز کند.
* **Hinge Loss**: این تابع ضرر اندازه perturbation‌ها را کنترل می‌کند و از این طریق اطمینان می‌دهد که نمونه‌های adversarial از داده‌های اصلی به‌طور عمده مشابه باقی بمانند و حملات از نظر بصری قابل شناسایی نباشند.

با بهینه‌سازی این تابع ضرر ترکیبی، AdvGAN نمونه‌های adversarial تولید می‌کند که نه تنها از نظر بصری واقعی و مشابه داده‌های اصلی هستند، بلکه مدل هدف را به‌طور مؤثر فریب می‌دهند و perturbation‌ها در حد قابل قبولی باقی می‌مانند. این ویژگی‌ها باعث می‌شود که حملات adversarial تولید شده توسط AdvGAN نه تنها مؤثر بلکه غیرقابل شناسایی و پایدار باشند.

### 4-1-2. تفاوت حمله جعبه سفید و جعبه سیاه

1. **حمله جعبه سفید (White-box Attack)**:
   * **تعریف**: در حملات جعبه سفید، مهاجم دسترسی کامل به مدل هدف دارد. این دسترسی شامل معماری مدل، پارامترها، وزن‌ها و داده‌های آموزشی است. به دلیل این دسترسی کامل، مهاجم می‌تواند گرادیان‌ها را محاسبه کرده و مستقیماً بهینه‌سازی pertubations (اختلال‌ها) را انجام دهد تا مثال‌های adversial به‌وجود آورد.
   * **روش حمله**: از آنجایی که مهاجم به تمام اطلاعات مدل دسترسی دارد، می‌تواند به راحتی با استفاده از روش‌هایی مانند **Fast Gradient Sign Method (FGSM)** یا **Project Gradient Descent (PGD)** perturbations را به ورودی‌ها اعمال کند تا مدل را فریب دهد.
   * **مزیت‌ها**: دسترسی به اطلاعات داخلی مدل این حملات را بسیار دقیق و مؤثر می‌کند. با استفاده از گرادیان‌ها، مهاجم می‌تواند دقیقاً اختلال‌هایی که مدل را فریب می‌دهند، تولید کند.
   * **چالش‌ها**: در این نوع حملات، نیاز به دسترسی مستمر به مدل هدف وجود دارد، که این ممکن است در دنیای واقعی مشکلاتی ایجاد کند.
2. **حمله جعبه سیاه (Black-box Attack)**:
   * **تعریف**: در حملات جعبه سیاه، مهاجم هیچ دسترسی به معماری داخلی یا پارامترهای مدل هدف ندارد. تنها چیزی که در اختیار مهاجم است، رفتار ورودی-خروجی مدل است (یعنی ورودی‌ها و پیش‌بینی‌هایی که مدل برای آنها تولید می‌کند).
   * **روش حمله**: در این نوع حمله، مهاجم نمی‌تواند مستقیماً مدل را مورد حمله قرار دهد و باید از روش‌های دیگری برای تولید مثال‌های adversial استفاده کند. از جمله این روش‌ها می‌توان به استفاده از مدل‌های جایگزین (surrogate models) یا استفاده از تکنیک‌هایی مانند **Zeroth-Order Optimization (ZOO)** اشاره کرد که در آن مهاجم با حداقل درخواست از مدل، تلاش می‌کند مدل هدف را فریب دهد.
   * **مزیت‌ها**: این نوع حمله بیشتر به واقعیت نزدیک است زیرا در بسیاری از مواقع مهاجم فقط دسترسی به خروجی مدل را دارد و نمی‌تواند به داده‌های داخلی آن دسترسی پیدا کند.
   * **چالش‌ها**: به دلیل عدم دسترسی به اطلاعات داخلی مدل، این حملات معمولاً پیچیده‌تر و دشوارتر هستند و نرخ موفقیت کمتری دارند.

**استفاده از AdvGAN در حملات سفید و سیاه**:

مدل **AdvGAN** برای انجام حملات در محیط‌های سفید، نیمه‌سفید و سیاه طراحی شده است. در اینجا نحوه استفاده از AdvGAN برای حملات سیاه به‌ویژه با استفاده از استراتژی‌های مختلف توضیح داده شده است:

1. **حملات نیمه‌سفید (Semi-whitebox Attacks)**:
   * در این نوع حمله، شبکه **Generator** در AdvGAN با دسترسی به مدل هدف آموزش داده می‌شود. پس از آموزش **Generator**، این شبکه می‌تواند بدون نیاز به دسترسی بیشتر به مدل هدف، pertubations را برای هر ورودی تولید کند.
   * این تفاوت مهمی با حملات سفید است که نیازمند دسترسی مداوم به مدل هدف هستند. در این حالت، مهاجم نیازی به درخواست‌های مکرر از مدل هدف ندارد.
2. **حملات جعبه سیاه (Black-box Attacks)**:
   * در حملات سیاه، AdvGAN از یک مدل **Distilled** برای شبیه‌سازی رفتار مدل هدف استفاده می‌کند. این مدل distilled با استفاده از خروجی‌های مدل سیاه آموزش داده می‌شود.
   * **Distillation ایستا (Static Distillation)**: ابتدا یک مدل distilled با داده‌هایی که از داده‌های آموزشی مدل هدف جدا هستند، آموزش داده می‌شود. پس از آماده شدن مدل distilled، همان استراتژی حمله‌ای که در حملات سفید استفاده می‌شود، اعمال می‌شود. به عبارت دیگر، شبکه **Generator** برای تولید مثال‌های adversial که مدل distilled را فریب دهند، آموزش داده می‌شود.
     + **چالش**: این روش ممکن است همیشه موثر نباشد، زیرا مدل distilled ممکن است رفتار مدل هدف را به‌طور دقیقی شبیه‌سازی نکند، به‌ویژه در مورد مثال‌های aسdversial.
   * **Distillation پویا (Dynamic Distillation)**: برای بهبود عملکرد در حملات سیاه، از استراتژی distillation پویا استفاده می‌شود. در این رویکرد، مدل distilled و **Generator** به صورت مشترک و در یک فرایند تکراری آموزش داده می‌شوند:
     + در هر تکرار، **Generator** به‌روز می‌شود تا مثال‌های adversial تولید کند که مدل distilled را فریب دهند.
     + سپس، مدل distilled بر اساس تصاویر آموزشی اصلی و مثال‌های adversial که توسط **Generator** تولید شده‌اند، به‌روز می‌شود.
     + این فرایند پویا به مدل distilled اجازه می‌دهد تا رفتار مدل هدف را در برابر مثال‌های adversial بهتر شبیه‌سازی کند، که موجب افزایش نرخ موفقیت حمله می‌شود.
   * **مزیت**: استفاده از **Distillation پویا** باعث می‌شود که AdvGAN قادر باشد به طور مؤثر حملات سیاه را انجام دهد، بدون اینکه به ویژگی‌های انتقال‌پذیری (transferability) متکی باشد.

### 5-1-2. دو مقاله دیگر در حوزه AdvGAN

* **مقاله اول:** [**A novel approach to generating high-resolution adversarial examples**](https://link.springer.com/article/10.1007/s10489-021-02371-w)

ایده‌های توسعه‌یافته در این مقاله به طور مشخص بر اساس چارچوب اولیه‌ی AdvGAN طراحی شده‌اند. هدف اصلی این مقاله، بهبود نرخ موفقیت حملات و ایجاد نمونه‌های خصمانه با کیفیت بالا و قابل درک بصری است. این مقاله با بهره‌گیری از تکنیک‌های **PCA** و **KPCA**، ویژگی‌های اصلی نمونه ورودی را به latent variables نگاشت (map) می‌کند. این کار باعث می‌شود که نمونه‌های خصمانه‌ی تولیدشده شباهت زیادی به نمونه‌های اصلی داشته باشند و در عین حال موفق به فریب مدل‌های پیشرفته شوند.

یکی از دستاوردهای این مقاله، بهبود نرخ موفقیت حملات روی مجموعه داده‌های با وضوح بالا نظیر ImageNet و LSUN است. این مقاله موفق شده است **نرخ موفقیت حمله‌ی 90.30%** را روی مدل ResNet152 در مجموعه داده‌ی CIFAR-10 به دست آورد، که نسبت به AdvGAN اولیه (با نرخ موفقیت 88.69%) اندکی بهبود یافته است. همچنین، این روش توانسته است نمونه‌های خصمانه‌ای تولید کند که از لحاظ بصری بسیار طبیعی به نظر می‌رسند و از نظر انتقال‌پذیری به سایر مدل‌ها نیز عملکرد بهتری دارند.

**پیشرفت‌های این مقاله نسبت به AdvGAN:**

این مقاله با تمرکز بر دو چالش اصلی AdvGAN، یعنی کار با مجموعه داده‌های با وضوح بالا و ایجاد تغییرات ظریف ولی موثر، چارچوب اولیه را گسترش داده است. برخلاف روش اولیه که بیشتر بر تولید نمونه‌ها برای داده‌های کوچک تمرکز داشت، این مقاله از ابزارهایی نظیر **PCA/KPCA** و افزایش داده (Data Augmentation) استفاده کرده است تا ویژگی‌های اصلی نمونه‌های ورودی بهتر حفظ شوند. در نهایت، با ادغام این روش‌ها با ساختار اولیه‌ی AdvGAN، نویسندگان موفق به تولید نمونه‌هایی شدند که هم قابل‌قبول از دیدگاه انسانی هستند و هم قادر به فریب دادن مدل‌های هدف با نرخ موفقیت بالا.

* **مقاله دوم:** [**Generating Adversarial Examples in One Shot with Image-To-Image Translation GAN**](https://ieeexplore.ieee.org/document/8863362)

این مقاله با ارائه یک روش جدید برای تولید نمونه‌های خصمانه، به طور مستقیم چارچوب **AdvGAN** را گسترش می‌دهد. برخلاف روش‌های سنتی مبتنی بر بهینه‌سازی و حتی خود **AdvGAN** که به تولید **Perturbation** می‌پردازند و آنها را به تصاویر اصلی اضافه می‌کنند، این روش مستقیماً نمونه‌های خصمانه را از تصاویر ورودی تولید می‌کند. این ویژگی باعث می‌شود فرآیند تولید سریع‌تر و ساده‌تر باشد.

**ایده‌های کلیدی مقاله:**

1. **تولید مستقیم نمونه‌های خصمانه**:
   * در این روش، به جای تولید **Perturbation** و اعمال آن روی تصویر، نمونه‌های خصمانه به صورت مستقیم از تصاویر ورودی تولید می‌شوند. این امر نه تنها فرآیند را ساده‌تر می‌کند، بلکه باعث هم‌خوانی بهتر تغییرات با شکل‌ها و لبه‌های اصلی تصویر می‌شود.
2. **کیفیت بصری بهتر**:
   * نمونه‌های تولیدشده به دلیل تطابق بهتر با ویژگی‌های تصویر ورودی (مانند لبه‌ها و شکل‌ها)، ظاهری طبیعی‌تر دارند و تغییرات ایجادشده برای چشم انسان تقریباً غیرقابل تشخیص هستند.
3. **عملکرد بالا**:
   * این روش روی مجموعه‌داده‌های **MNIST** و **CIFAR-10** آزمایش شده و نشان داده است که عملکردی بهتر یا مشابه با **AdvGAN** از نظر نرخ حمله و میزان **Perturbation** دارد. همچنین در مقایسه با حملات مبتنی بر بهینه‌سازی مانند **حمله Carlini & Wagner (CW)** نتایج رقابتی ارائه داده است.

**توسعه‌های مقاله نسبت به AdvGAN:**

این مقاله با حذف نیاز به تولید و اعمال **Perturbation**، فرآیند تولید نمونه‌های خصمانه را ساده‌تر و سریع‌تر کرده است. علاوه بر این، با تمرکز بر حفظ ساختارهای اصلی تصویر ورودی، نمونه‌هایی تولید می‌کند که برای انسان طبیعی‌تر به نظر می‌رسند و هم‌زمان قدرت حمله‌ی بالایی دارند. این ویژگی‌ها این روش را بهبود مستقیمی برای **AdvGAN** تبدیل می‌کند.

## 2-2. پیاده‌سازی مدل AdvGAN

### 1-2-2. دانلود و آماده‌سازی اولیه داده

CIFAR-10 یکی از دیتاست‌های استاندارد برای مسائل یادگیری ماشین و به‌خصوص یادگیری عمیق در حوزه پردازش تصویر است. این دیتاست شامل **60,000 تصویر رنگی** با ابعاد **32x32 پیکسل** است که در **10 کلاس مختلف** دسته‌بندی شده‌اند. هر کلاس شامل **6,000 تصویر** می‌باشد.

کلاس‌های موجود در این دیتاست عبارتند از:

1. Airplane
2. Automobile
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

این دیتاست به دو بخش تقسیم می‌شود:

* **داده‌های آموزشی:** 50,000 تصویر
* **داده‌های تست:** 10,000 تصویر

**فرضیات در پیاده‌سازی**

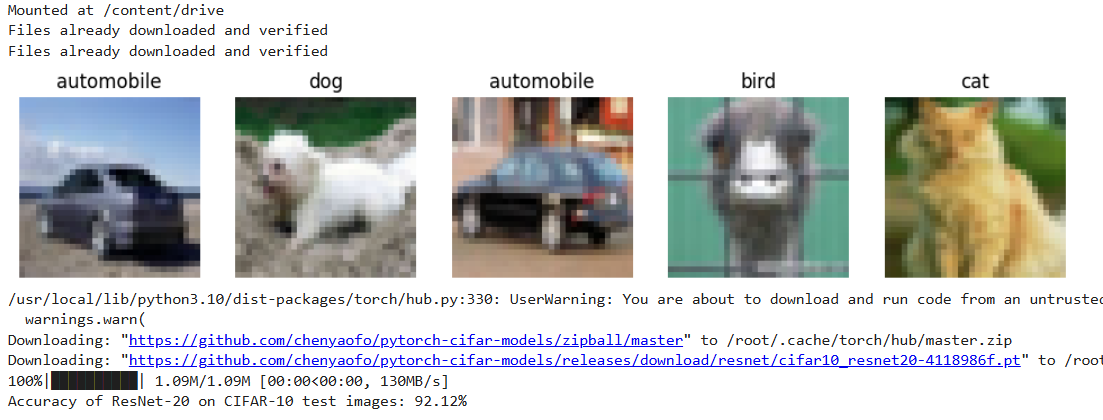
1. تمامی داده‌ها در گوگل درایو ذخیره می‌شوند تا در اجرای‌های بعدی نیازی به دانلود مجدد نباشد.
2. از ابزارهای torchvision و torch.utils.data برای دانلود، بارگذاری، و تقسیم‌بندی داده‌ها استفاده می‌کنیم.
3. نرمال‌سازی تصاویر با مقادیر میانگین و انحراف معیار از پیش تعیین‌شده برای CIFAR-10 انجام شده است:
   * میانگین: (0.4914,0.4822,0.4465)
   * انحراف معیار: (0.2470,0.2435,0.2616)
4. از یک مدل از پیش آموزش دیده (Pretrained Model) با معماری **ResNet-20** استفاده شده است که برای CIFAR-10 بهینه شده است.

**گام‌های پیاده‌سازی**

1. **دانلود و بارگذاری دیتاست CIFAR-10:**
   * دیتاست از طریق torchvision.datasets دانلود و در مسیر مشخصی در گوگل درایو ذخیره می‌شود.
   * داده‌ها به سه مجموعه تقسیم شدند:
     + **آموزشی (Training):** شامل 45,000 تصویر
     + **اعتبارسنجی (Validation):** شامل 5,000 تصویر (درصدی برای اختصاص به دیتای Validation ذکر نشده بود پس این عدد را خود فرض کردیم.)
     + **تست (Test):** شامل 10,000 تصویر
   * از DataLoader برای بارگذاری دسته‌ای (batch) داده‌ها استفاده شده است.
2. **نمایش نمونه تصاویر:**
   * 5 نمونه تصویر تصادفی از مجموعه داده آموزشی نمایش داده شد.
   * تصاویر به دلیل نرمال‌سازی نیاز به تبدیل داشتند تا بتوان آن‌ها را به شکل قابل مشاهده به نمایش گذاشت.
3. **بارگذاری مدل ResNet-20:**
   * مدل ResNet-20 از مخزن torch.hub بارگذاری شد.
   * این مدل از پیش آموزش دیده برای دیتاست CIFAR-10 طراحی شده و دارای توانایی خوبی در طبقه‌بندی تصاویر این دیتاست است.
4. **محاسبه دقت مدل:**
   * مدل بر روی داده‌های تست بدون نیاز به آموزش مجدد (Fine-Tuning) ارزیابی شد.
   * دقت محاسبه شده برای مدل ResNet-20 روی داده‌های تست **92.12%** بود.



شکل 7- بخشی از کد تعریف مدل هدف و ارزیابی



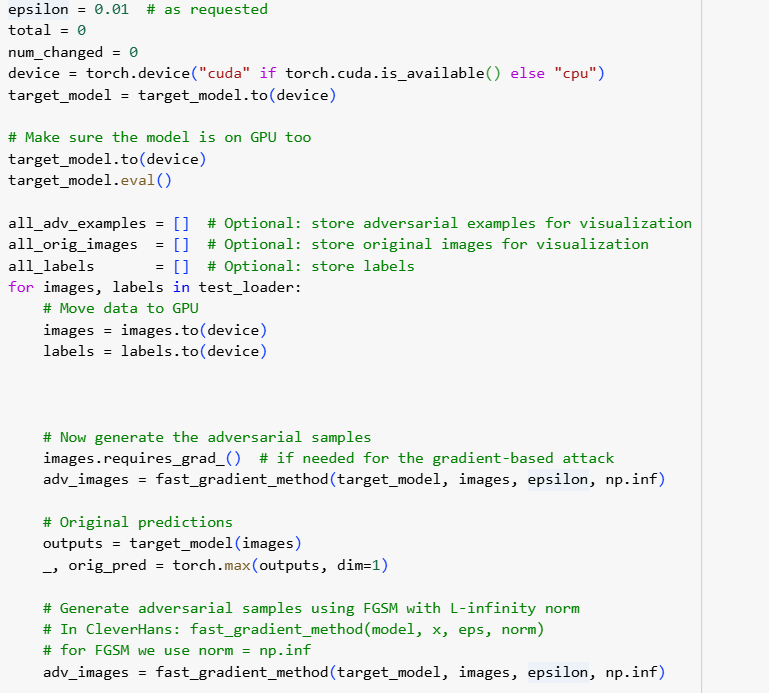
شکل 8- دانلود و تعریف مدل هدف و بررسی دقت آن روی مجموعه تست

### 2-2-2. ایجاد تصاویر تخاصمی و نرخ حمله به مدل هدف

در این بخش، هدف تولید تصاویر تخاصمی با استفاده از روش **Fast Gradient Sign Method (FGSM)** و کتابخانه **CleverHans** بود. مراحل پیاده‌سازی و نتایج به شرح زیر است:

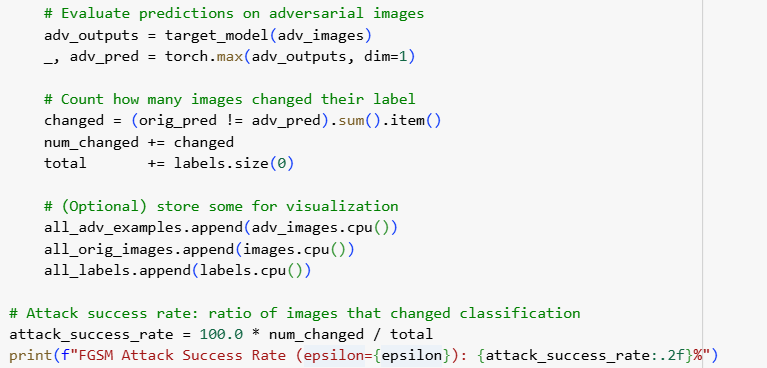
**گام‌های انجام شده:**

1. **بارگذاری مدل هدف (Target Model)** :
   * مدل از پیش آموزش داده‌شده برای دیتاست **CIFAR-10** بارگذاری شد. این مدل به حالت **eval** تغییر وضعیت داده شد تا خروجی‌ها در حالت ارزیابی تولید شوند.
2. **تبدیل تصاویر مجموعه آزمایشی به تصاویر تخاصمی**:
   * تصاویر ورودی با استفاده از تابع **fast\_gradient\_method** از کتابخانه CleverHans و پارامتر **epsilon=0.01** تغییر داده شدند.
   * این تغییرات بر اساس گرادیان تابع هزینه مدل هدف اعمال شد.



شکل 9- کد مربوط به ایجاد نمونه‌های متخاصم

1. **محاسبه نرخ موفقیت حمله**:
   * نرخ موفقیت حمله به صورت درصدی از تصاویری که پیش‌بینی مدل روی آن‌ها تغییر کرده است، محاسبه شد.
   * برای این منظور، پیش‌بینی مدل روی تصاویر اصلی و تصاویر تخاصمی مقایسه شد.
2. **ذخیره‌سازی داده‌ها**:
   * تصاویر اصلی، تصاویر تخاصمی و برچسب‌های آن‌ها به منظور تحلیل‌های بعدی ذخیره شدند.

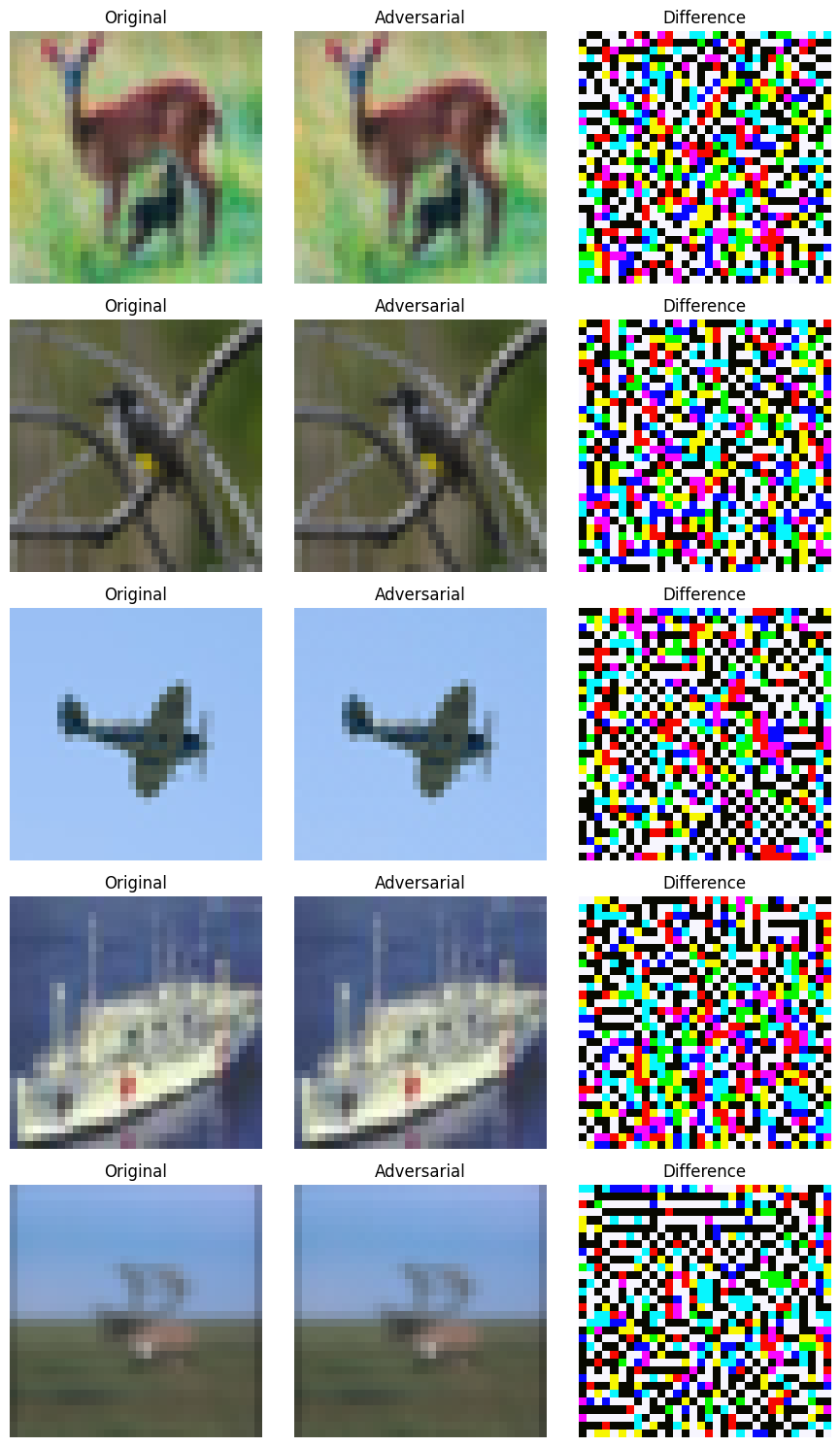


شکل 10- کد مربوط به بررسی نرخ موفقیت حمله



شکل 11- نرخ موفقیت حمله با روش FGSM

1. **نمایش نتایج**:
   * پنج تصویر به صورت تصادفی انتخاب و کنار تصاویر تخاصمی و اختلاف آن‌ها نمایش داده شدند.
   * تصاویر ابتدا به حالت اولیه (بدون نرمال‌سازی) برگردانده شدند تا نمایش آن‌ها قابل فهم باشد.



شکل 12- نمایش تصاویر اولیه و تصاویر تخاصمی

### 3-2-2. پیاده‌سازی و آموزش مدل مولد و متمایزگر

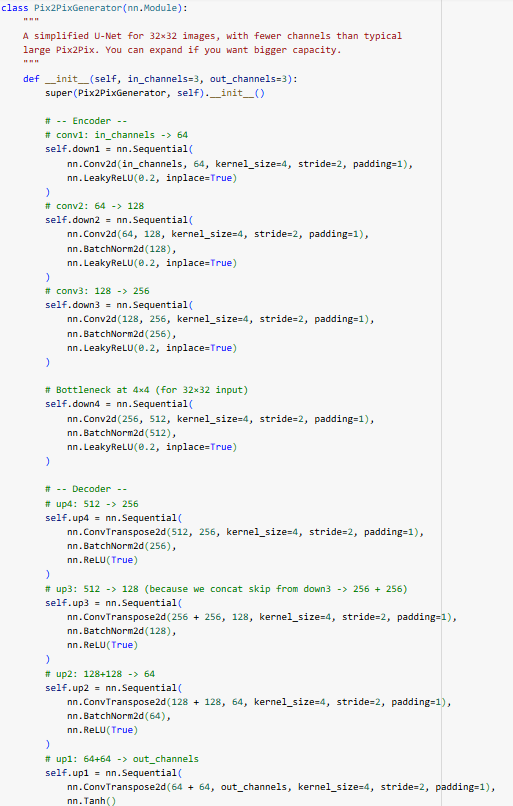
در این بخش مدل مولد و متمایزگر مطابق معماری ذکر شده در مقاله (Pix2Pix) پیاده سازی شده است. در ادامه به تشریح مشخصات این معماری برای هر دو بخش مولد و متمایزگر و همینطور تمامی فرضیات استفاده شده در فرایند آموزش اعم از ضرایب مورد استفاده برای توابع هزینه پرداخته شده است.

**۱. ساختار مدل‌ها و فرضیات**

۱.۱ ساختار Generator

Generator استفاده‌شده در این پروژه یک معماری به‌سبک U-Net و همان مدل Pix2Pix با اندازه ورودی ۳۲×۳۲ است که شامل اجزای زیر می‌شود:

* **مسیر کاهشی (Encoder):** لایه‌های متوالی کانولوشن با تعداد کانال‌های افزایشی (64 → 128 → 256 → 512) و فعال‌سازی LeakyReLU. از نرمال‌سازی Batch در تمامی لایه‌ها به‌جز لایه اول استفاده شده است.
* **مسیر افزایشی (Decoder):** لایه‌های کانولوشن معکوس با اتصال‌های میان‌بر از مسیر کاهشی. این مسیر تعداد کانال‌ها را کاهش می‌دهد (512 → 256 → 128 → 64 → 3) و از فعال‌سازی ReLU بهره می‌برد. لایه خروجی از فعال‌سازی Tanh استفاده می‌کند تا تصاویر خروجی در محدوده [1, 1-] تولید شوند.



شکل 13 – پیاده‌سازی معماری مدل مولد

A screenshot of a computer code

Description automatically generated

شکل 14 - پیاده‌سازی بخش forward مدل مولد

**فرضیات:**

* معماری Generator به‌منظور تطبیق با ابعاد کوچک تصاویر CIFAR-10 ساده‌سازی شده است.
* استفاده از اتصال‌های میان‌بر برای حفظ اطلاعات فضایی ضروری بوده و باعث می‌شود Generator بتواند نمونه‌های خصمانه قابل قبولی تولید کند.

۱.۲ ساختار Discriminator

Discriminator از نوع PatchGAN است که برای ارزیابی پچ‌های تصویر با اندازه ۳۲×۳۲ طراحی شده است. اجزای اصلی این مدل عبارتند از:

* لایه‌های کانولوشن متوالی (3 → 64 → 128 → 256 → 512 → 1) که ابعاد فضایی را کاهش داده و عمق کانال‌ها را افزایش می‌دهند.
* فعال‌سازی LeakyReLU و نرمال‌سازی Batch در همه لایه‌ها به‌جز لایه خروجی استفاده شده است.

A screenshot of a computer program

Description automatically generated

شکل 15- پیاده‌سازی معماری مدل متمایزگر

**فرضیات:**

* ارزیابی مبتنی بر پچ امکان تشخیص دقیق‌تر و پایدارتر را فراهم می‌کند.
* عدم استفاده از تابع فعال‌سازی در خروجی به دلیل انجام محاسبات هزینه در مراحل بعدی است.

**۲. توابع هزینه**

سه تابع هزینه کلیدی در این پروژه پیاده‌سازی و استفاده شده‌اند:

۱. **CW Loss:**  
این تابع هزینه برای اندازه‌گیری اثربخشی نمونه‌های خصمانه در برابر مدل هدف استفاده می‌شود. CW Loss اختلاف بین logit کلاس درست و بزرگ‌ترین logit کلاس اشتباه را جریمه می‌کند.

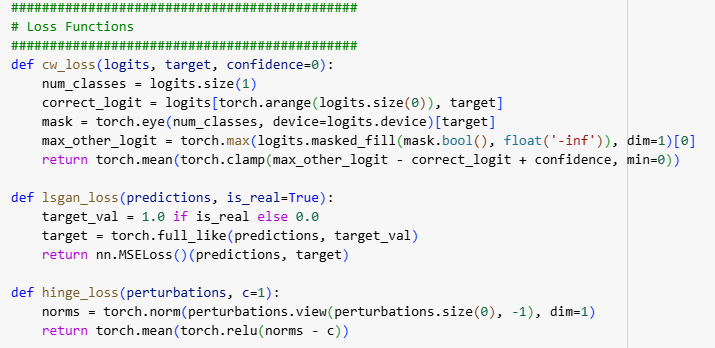
* **فرضیات:** فرمول‌بندی CW Loss بدون هدف‌گذاری (non-targeted) کافی بوده زیرا هدف اصلی کاهش دقت مدل هدف بر روی نمونه‌های خصمانه است.

۲. **LSGAN Loss:**  
این تابع هزینه برای هر دو Generator و Discriminator اعمال می‌شود. Generator تلاش می‌کند خروجی‌های خود را به‌عنوان نمونه‌های واقعی به Discriminator نشان دهد، در حالی که Discriminator به دنبال تشخیص صحیح نمونه‌های واقعی و جعلی است.

* **فرضیات:** LSGAN به دلیل پایداری و گرادیان‌های صاف در این سناریوی آموزشی مناسب است.

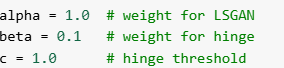
۳. **Hinge Loss:**  
برای اعمال محدودیت بر نُرم L2 تغییرات به‌منظور اطمینان از نزدیک ماندن نمونه‌های خصمانه به تصاویر اصلی، از این تابع هزینه استفاده شده است.

* **فرضیات:** مقدار ثابت c=1 به‌عنوان آستانه‌ای مؤثر برای اندازه تغییرات بدون نیاز به تنظیم اضافی در نظر گرفته شده است.



شکل 16- تعریف توابع هزینه

همچنین ضرایب آلفا و بتا و c در فرمول نهایی ترکیب همه توابع هزینه به تریب برابر 1 و 0.1 و 1 در نظر گرفته شده است.



شکل 17- ضرایب در نظر گرفته شده برای آموزش

**فرمول نهایی Generator** به شکل زیر ترکیب شده است:

LossG​=CW+α⋅LSGAN+β⋅Hinge,

**۳. تنظیمات آموزشی**

۳.۱ داده‌ها و نرمال‌سازی

* از مجموعه داده‌های CIFAR-10 استفاده شده است که به ۴۵,۰۰۰ تصویر آموزشی، ۵,۰۰۰ تصویر اعتبارسنجی و ۱۰,۰۰۰ تصویر آزمایشی تقسیم شده‌اند.
* نرمال‌سازی استاندارد CIFAR-10 در این قسمت انجام نشده و فقط با استفاده از to\_tensor تصاویر وارد بازه [0,1] شده‌اند.

خروجی Generator در بازه‌ی [−1,1] است که برای اعمال به مدل هدف باید دوباره در بازه‌ی[0,1] پیمانه شود. (تبدیل از [−1,1] به [0,1])

**فرضیات:**

* نرمال‌سازی استاندارد و معکوس آن برای پایداری آموزش و حفظ سازگاری در مقیاس ورودی‌ها کافی بوده است.
* از هیچ افزونه داده (Data Augmentation) استفاده نشده و تنها به تقسیمات استاندارد CIFAR-10 اکتفا شده است.

۳.۲ بهینه‌سازی و نرخ یادگیری

* هر دو Generator و Discriminator با استفاده از بهینه‌سازی Adam با نرخ یادگیری ۰.۰۰۱ و مقادیر بتا (0.5, 0.999) آموزش دیده‌اند.
* **فرضیات:** مقادیر پیش‌فرض برای Adam بدون تنظیمات اضافی اثربخش بوده‌اند.

۳.۳ فرآیند آموزش

* در هر تکرار، Discriminator با استفاده از هر دو مجموعه تصاویر واقعی و نمونه‌های تولیدشده توسط Generator به‌روزرسانی شد.
* Generator با استفاده از ترکیب CW، LSGAN و Hinge Loss و با وزن‌دهی مناسب آموزش داده شد.
* حلقه آموزشی شامل به‌روزرسانی‌های Generator و Discriminator در هر تکرار و ارزیابی‌های دوره‌ای روی مجموعه اعتبارسنجی بود.
* آموزش به مدت ۵۰ دوره انجام شد، مطابق با جزئیات پیاده‌سازی مقاله مرجع.

**۴. معیارهای ارزیابی**

۴.۱ دقت Discriminator

* دقت Discriminator به‌عنوان درصد تصاویر واقعی که به‌درستی به‌عنوان واقعی طبقه‌بندی شده‌اند و تصاویر جعلی که به‌درستی به‌عنوان جعلی طبقه‌بندی شده‌اند، محاسبه شد.
* **فرضیات:** دقت بالاتر نشان‌دهنده توانایی بهتر در تفکیک بین نمونه‌های واقعی و خصمانه است، اما نباید به حدی برسد که منجر به فروپاشی مد شود.

۴.۲ دقت مدل هدف

* دقت ResNet-20 بر روی نمونه‌های خصمانه در طول دوره‌ها پیگیری شد.
* **فرضیات:** کاهش دقت مدل هدف (و افزایش نرخ موفقیت حمله) نشان‌دهنده عملکرد بهتر Generator است.

**۵. نتایج و تحلیل نمودارها**

**۵.۱ نمودار Training Losses**

A graph showing a line graph

Description automatically generated

شکل 18- نمودار تغییرات loss

**Discriminator Loss :**

* + در ابتدای آموزش مقداری نسبتاً کم (زیر ۱) دارد و با نوساناتی، گاهی نزدیک ۰.۵ و در میانه‌ی آموزش حتی تا حدود ۱.۰ بالا می‌رود.
  + در ادامه به سمت مقداری زیر ۱ (بین ۰.۴ تا ۰.۶) تثبیت شده است.
  + این افت‌وخیزها در اوایل، عادی است چون Generator و Discriminator هم‌زمان در حال یادگیری هستند.
* **Generator Loss:**
  + در اوایل کار بالای ۴ مشاهده می‌شود که نشان می‌دهد Generator ابتدا در فریب Discriminator و حمله به مدل هدف دچار دشواری بوده است.
  + پس از چند Epoch کاهش پیدا می‌کند (تا حوالی ۲) و سپس مجدداً روند صعودی ملایمی دارد.
  + این روند صعودی می‌تواند ناشی از سنگین‌تر شدن CW Loss یا LSGAN Loss باشد، هنگامی که Generator سعی می‌کند حمله را بهبود دهد و Discriminator قوی‌تر شده است.

به طور کلی، **Discriminator** با وجود نوسان، به سطح معقولی از تمایز رسیده و **Generator** نیز توانسته نمونه‌های معناداری تولید کند؛ چرا که ضرر کلی آن تثبیت/متعادل شده و به فروپاشی منجر نشده است.

**۵.۲ نمودار Discriminator Accuracy vs Epoch**

A graph of a graph

Description automatically generated

شکل 19 - نمودار تغییرات دقت Discriminator.

دقت اولیه متمایزگر تا حدود ۵۵٪ است. سپس افتی تا زیر ۵۲٪ داشته و مجدداً رشد کرده است.

* بین بازه‌های ۲۰ تا ۲۵، افزایش چشم‌گیری (حتی بالاتر از ۶۴٪) دیده می‌شود که نشان می‌دهد در آن مقطع Discriminator در تمایز بهتر عمل کرده است.
* پس از چند نوسان، دقت حوالی ۶۰٪ تا ۶۴٪ نوسان می‌کند و در انتها کمی افت می‌کند (نزدیک ۶۲٪).
* چنین رفتاری برای یک GAN پیچیده طبیعی است، زیرا Generator و Discriminator دائماً در حال مقابله‌اند.
* دقت بیش از حد بالا نشان‌دهنده‌ی غلبه‌ی متمایزگر نیست و دقت خیلی پایین هم نشان‌دهنده‌ی ضعف شدید آن نخواهد بود؛ بنابراین حفظ عددی میان‌حد (۵۵٪ تا ۶۵٪) می‌تواند نشانه‌ی تعادل نسبی باشد.

**۵.۳ نمودار Target Model Accuracy vs Attack Success Rate**

A graph of a graph of a graph

Description automatically generated with medium confidence

شکل 20- نمودار تغیرات Attack Success و Target Model Accuracy

* **نمودار آبی (Target Model Accuracy):** دقت مدل هدف روی نمونه‌های خصمانه را نشان می‌دهد. ابتدا حدود ۲۰٪ شروع می‌شود، سپس نوسانی تا ۴۰٪ دارد و در طول آموزش بارها کم و زیاد می‌شود (گاهی تا ۱۵٪، گاهی نزدیک ۴۰٪). در اواخر دوره، حدود ۲۵٪ است. این یعنی مدل هدف غالباً قادر به تشخیص درست نیست (کاهش دقت چشمگیر نسبت به دقت عادی ~۹۳٪ یا ۹۴٪ روی CIFAR-10).
* **نمودار نارنجی (Attack Success Rate):** درصد نمونه‌هایی که به طبقه‌بندی اشتباه منجر می‌شوند. این مقدار عموماً مکمل دقت مدل هدف است. در شروع نزدیک ۸۰٪ بوده، حتی در بعضی دوره‌ها به حدود ۹۰٪ و بیشتر هم رسیده و سپس نوسان کرده است (در پایان حدود ۷۵٪ تا ۸۰٪).
* به طور کلی، وجود حمله‌ی موفق نشان می‌دهد که Generator توانسته نمونه‌های مخربی بسازد که به‌خوبی دقت مدل هدف را پایین بیاورد.
* نوسانات بخشی ناشی از تغییرات مستمر در Discriminator است که روی یادگیری نهایی Generator تأثیر می‌گذارد.

**6. فرضیات و محدودیت‌ها**

* **مدل هدف از پیش آموزش‌یافته:** فرض بر این بود که ResNet-20 به خوبی کالیبره شده و نماینده‌ای مناسب برای دسته‌بندی‌کننده‌های استاندارد CIFAR-10 است.
* **تعداد دوره‌ها:** انتخاب تعداد ۵۰ دوره بر اساس پیاده‌سازی مرجع صورت گرفت و بهینه‌سازی تجربی نشد.
* **وزن توابع هزینه:** وزن‌های ترکیبی برای CW، LSGAN و Hinge Loss به‌طور دستی انتخاب شدند و تنظیم بیشتری انجام نشد.
* **سخت‌افزار و زمان اجرا:** آموزش بر روی یک GPU انجام شد و محدودیت‌های زمانی ممکن است بر عمق کاوش در تنظیمات اثر گذاشته باشد.

### 4-2-2. بررسی نرخ موفقیت حمله و هیستوگرام قطعیت

**مفروضات و نکات کلیدی در این مرحله:**

* **تعریف «حمله موفق**: حمله درصورتی موفق است که برچسب پیش‌بینی‌شده‌ی مدل هدف برای تصویر خصمانه با برچسب واقعی آن تفاوت داشته باشد(predadv​/=label)
* **مدل هدف**: یک شبکه‌ی **ResNet-20** از پیش آموزش‌دیده بر روی CIFAR-10 که در حالت eval قرار دارد.
* **مشخصه‌ی بازه‌ی پیکسل‌ها**: تصاویر ورودی اصلی در بازه‌ی [0,1] هستند (پس از ToTensor() بدون نرمال‌سازی خاص) Generator اما خروجی را در بازه‌ی [-1,1] تولید می‌کند و برای اعمال به مدل هدف، آن را دوباره به [0,1] نگاشت می‌کنیم.
* **تعریف Confidence**: حداکثر احتمال Softmax مدل (بیش‌ترین مؤلفه‌ی بردار خروجی شبکه).
* **تقسیم داده‌ی تست**: شامل ۱۰هزار تصویر در CIFAR-10، با ۱۰۰۰ نمونه در هر یک از ۱۰ کلاس

**1. نرخ موفقیت حمله (کلی و به ازای هر کلاس)**

**1.۱. نرخ موفقیت کلی**

با آزمون روی کل ۱۰هزار نمونه‌ی آزمایشی، حمله در **۷۷.۱۰٪** موارد موفق عمل کرده است. این بدان معناست که نزدیک به سه‌چهارم نمونه‌ها پس از اعمال نویزهای خصمانه، توسط مدل هدف اشتباه طبقه‌بندی شده‌اند.

**۲.1. نرخ موفقیت به ازای هر کلاس**

بر اساس نتایج به‌دست‌آمده از کد، برای هر کلاس CIFAR-10 (به‌ترتیب برچسب‌های ۰ تا ۹)، میزان موفقیت حمله به‌صورت زیر است:

جدول 1- نرخ موفقیت حمله به ازای هر کلاس

| **کلاس** | **نام کلاس (CIFAR-10)** | **تعداد نمونه در تست** | **تعداد شکست مدل** | **نرخ موفقیت حمله (%)** |
| --- | --- | --- | --- | --- |
| 0 | airplane | 1000 | 678 | 67.80% |
| 1 | automobile | 1000 | 997 | 99.70% |
| 2 | bird | 1000 | 917 | 91.70% |
| 3 | cat | 1000 | 640 | 64.00% |
| 4 | deer | 1000 | 849 | 84.90% |
| 5 | dog | 1000 | 322 | 32.20% |
| 6 | frog | 1000 | 832 | 83.20% |
| 7 | horse | 1000 | 921 | 92.10% |
| 8 | ship | 1000 | 563 | 56.30% |
| 9 | truck | 1000 | 991 | 99.10% |

**مشاهدات مهم:**

* بیش‌ترین موفقیت در حمله به کلاس‌های **automobile** (نزدیک ۹۹.۷٪) و **truck** (۹۹.۱٪) دیده می‌شود.
* کم‌ترین موفقیت مربوط به کلاس **dog** است (تنها ۳۲.۲٪).
* این نتایج نشان می‌دهد که برخی کلاس‌ها (مانند وسایل نقلیه) نسبت به نویزهای خصمانه‌ی تولیدی Generator، حساسیت بیشتری دارند و مدل هدف را آسان‌تر به خطا می‌اندازند؛ در مقابل، در برخی کلاس‌ها (نظیر سگ) تغییرات اعمالی کمتر موفق به تغییر پیش‌بینی مدل شده است.

**2. نمایش ۵ تصویر از مجموعه‌ی آزمایشی و نسخه‌ی خصمانه**

در شکل زیر، پنج نمونه از مجموعه‌ی آزمایشی نمایش داده شده است. برای هر مورد، سه ستون مجزا وجود دارد:

1. **Original**: تصویر اصلی در بازه‌ی [0,1]
2. **Adversarial**: تصویر خروجی Generator که پس از تولید در بازه‌ی [-1,1]، به [0,1] نگاشت شده است.
3. **Difference**: نقشه‌ی اختلاف Δ=(Adversarial−Original) که سپس برای دیده‌شدن بهتر به بازه‌ی [0,1] نرمال می‌شود.

A collage of images of animals

Description automatically generated

شکل 21- نمونه‌هایی از تصاویر واقعی و خصمانه به همراه تفاوت آنها

**تحلیل کیفی**:

* تفاوت ظاهری بین تصویر اصلی و خصمانه غالباً در حد نویزهای محلی مشاهده می‌شود.
* نقشه‌ی اختلاف، نواحی رنگی را نشان می‌دهد که در آن‌ها نویز بیش‌تر/کم‌تری اعمال شده است.
* از دید انسان، تصویر خصمانه هنوز شباهت بسیار بالایی به نمونه‌ی اصلی دارد؛ اما برای مدل هدف به اندازه‌ی کافی تغییر ایجاد می‌کند تا طبقه‌بندی را مختل نماید.

**3. رسم نمودار هیستوگرام قطعیت مدل هدف در پیش‌بینی**

در این مرحله، برای هر دو مجموعه‌ی تصاویر عادی و تصاویر خصمانه**Confidence** مدل هدف در کلاس پیش‌بینی‌شده محاسبه و سپس نمودار هیستوگرام (با تعداد سطل‌های ۵۰) رسم شده است:

A graph with a number of text

Description automatically generated with medium confidence

شکل 22- هیستوگرام قطعیت

**محور افقی**: مقدار اعتماد در بازه‎ی [0,1]

* **محور عمودی**: چگالی (Density) نمونه‌ها در هر بازه از اعتماد

**نتایج کلیدی**:

1. برای تصاویر عادی، بخش قابل‌توجهی از نمونه‌ها اعتماد بالایی (نزدیک ۱.۰) دارند. این طبیعی است زیرا مدل هدف بر روی CIFAR-10 آموزش دیده و معمولاً با اطمینان بالا تصمیم می‌گیرد.
2. برای تصاویر خصمانه، توزیع اعتماد پخش‌تر است و اوج (پیک) آن در محدوده‌ی پایین‌تر از ۱.۰ قرار دارد (بیش‌تر در محدوده‌ی ۰.۴ تا ۰.۷). این نشان می‌دهد که حملات خصمانه موفق شده‌اند بسیاری از نمونه‌ها را از اعتماد حداکثری دور کنند و مدل را به پیش‌بینی‌هایی کم‌اعتماد‌تر (یا اشتباه) سوق دهند.

**4. جمع‌بندی**

1. **نرخ موفقیت کلی** حمله در این بخش به **۷۷.۱۰٪** رسیده است؛ عددی قابل‌توجه که بیان‌گر تأثیر موفق این حمله روی ResNet-20 از پیش آموزش‌دیده است.
2. به‌صورت کلاس‌محور، حمله در برخی کلاس‌ها (نظیر خودرو یا کامیون) تقریباً به‌طور کامل موفق بوده و در برخی دیگر (مانند سگ) موفقیت کم‌تری داشته است.
3. بررسی **۵ نمونه‌ی بصری** نشان داد که نویز اعمالی اغلب برای چشم انسان کوچک است، اما در عین حال **مدل هدف** را به‌طور قابل‌توجهی دچار اشتباه می‌کند.
4. هیستوگرام قطعیت مدل هدف هم نشان داد که **اعتماد شبکه** برای نمونه‌های خصمانه به‌طور میانگین کمتر شده و از ۱.۰ فاصله گرفته است؛ امری که تأیید می‌کند سازوکار حمله (Generator) به خوبی توانسته است به اطمینان بالای مدل هدف آسیب بزند.

با توجه به این نتایج، می‌توان گفت ترکیب معماری Pix2Pix و PatchGAN با توابع هزینه‌ی CW+LSGAN+Hinge در تولید نمونه‌های خصمانه برای CIFAR-10 کارآمد بوده و ضمن حفظ شباهت تصویری، درصد موفقیت بالایی در شکستن طبقه‌بندی مدل هدف به دست آورده است.

### 5-2-2. پیاده‌سازی و بررسی مدل هدفدار (امتیازی)

1. **حملات بدون هدف (Non-Targeted Attacks):** هدف این نوع حملات باعث می‌شود مدل ورودی را به هر کلاس اشتباهی دسته‌بندی کند.
2. **حملات هدفدار (Targeted Attacks):** در این حملات، هدف مشخص این است که مدل ورودی را به یک کلاس خاص دیگر اشتباه دسته‌بندی کند.

در این بخش، برخلاف حمله‌ی غیرهدفدار که صرفاً تلاش می‌کرد مدل هدف در طبقه‌بندی نمونه دچار خطا شود، از یک حمله‌ی هدفدار (Targeted Attack) استفاده شده است. در حمله‌ی هدفدار، هر نمونه طوری دستکاری می‌شود که برچسب پیش‌بینی‌شده‌ی مدل هدف به یک کلاس موردنظر، اینجا (label+1) برسد.

تفاوت اصلی پیاده‌سازی با بخش قبل در موارد زیر است:

* **انتخاب برچسب هدف (Target Label):** در بخش غیرهدفدار، تابع هزینه‌ی CW سعی داشت logit کلاس درست را کمتر از ماکس سایر کلاس‌ها کند؛ اما اینجا با تابع کوچکی مثل get\_target\_labels، برچسب جدیدی برای هر نمونه تعریف می‌شود تا Generator به‌طور مشخص آن کلاس را هدف قرار دهد.
* **CW Loss هدفدار:** محاسبه‌ی تابع هزینه‌ی CW اکنون بر اساس **برچسب هدف** صورت می‌گیرد (در کد cw\_loss(logits\_target, target\_labels, ...) )یعنی Generator تلاش می‌کند logit کلاس هدف را بالاتر از بقیه‌ی کلاس‌ها ببرد.
* **تعریف متریک حمله:** برای ارزیابی، به‌جای کاهش دقت مدل در پیش‌بینی لیبل اصلی، اکنون میزان موفقیت در پیش‌بینی کلاس هدف سنجیده می‌شود. در کد، اگر دقت مدل هدف در رسیدن به این کلاس بالا باشد، حمله هدفدار بهتر عمل کرده است.

سایر بخش‌های معماری Generator، Discriminator، و نیز توابع LSGAN و Hinge Loss مانند قبل هستند و تغییر خاصی ندارند. با همین تغییر کوچک در برچسب هدفدار و تابع هزینه‌ی مرتبط، رویکرد غیـرهدفدار قبلی به یک حمله‌ی هدفدار تبدیل می‌شود.

در نهایت نتایج خواسته شده به شکل زیر بدست می‌آید.

**۱. نتایج کلی حمله‌ی هدفدار**

* **نرخ موفقیت کلی حمله**:  
  نرخ موفقیت حمله در این بخش برابر **۸۵.۴۵٪** است که نشان‌دهنده موفقیت بسیار بالا در حمله به مدل هدف است. این مقدار به‌طور قابل‌توجهی بالاتر از نرخ موفقیت حمله در بخش غیرهدفدار است (۷۷.۱۰٪). این تفاوت به این دلیل است که حمله‌ی هدفدار به‌طور خاص تلاش می‌کند که مدل هدف را به سمت یک کلاس خاص هدایت کند که نتیجه‌ی آن موفقیت بالاتری است.
* **نرخ موفقیت به ازای هر کلاس**:  
  نرخ موفقیت حمله در کلاس‌های مختلف در جدول زیر آمده است:

جدول 2- نرخ موفقیت حمله به ازای هر کلاس مدل targeted

| **کلاس** | **نام کلاس** | **نرخ موفقیت حمله (%)** | **تعداد موفقیت‌ها (از ۱۰۰۰ نمونه)** |
| --- | --- | --- | --- |
| 0 | airplane | 75.60% | 756 |
| 1 | automobile | 100.00% | 1000 |
| 2 | bird | 98.70% | 987 |
| 3 | cat | 70.50% | 705 |
| 4 | deer | 92.20% | 922 |
| 5 | dog | 39.10% | 391 |
| 6 | frog | 96.70% | 967 |
| 7 | horse | 99.60% | 996 |
| 8 | ship | 82.10% | 821 |
| 9 | truck | 100.00% | 1000 |

**تفاوت‌ها با حمله‌ی غیرهدفدار:**

* در حمله‌ی هدفدار، کلاس‌هایی مانند **automobile** و **truck** به‌طور کامل تحت تأثیر قرار گرفته و نرخ موفقیت حمله به ۱۰۰٪ رسید. این کلاس‌ها به‌طور خاص دارای ویژگی‌های بصری قابل‌تشخیص بیشتری هستند که به Generator این امکان را می‌دهد که تصاویر خصمانه‌ای ایجاد کند که به‌راحتی مدل هدف را فریب دهند.
* در کلاس‌هایی مانند **dog**، نرخ موفقیت به‌شدت کاهش پیدا کرده و تنها ۳۹.۱۰٪ موفقیت در تغییر پیش‌بینی مدل هدف مشاهده می‌شود. این امر ممکن است به دلیل ویژگی‌های پیچیده‌تر و کمتر قابل‌تشخیص در سگ‌ها باشد که برای تغییر دقیق‌تر نیاز به اصلاحات بیشتری دارد.
* در مجموع، نرخ موفقیت در کلاس‌هایی که بیشتر ویژگی‌های یکپارچه دارند (مثل **automobile**, **truck**, **frog**) به‌شدت بیشتر است.

**۲. نمایش ۵ نمونه تصاویر و تفاوت‌ها**

در شکل زیر، پنج نمونه از مجموعه‌ی آزمایشی همراه با تصاویر خصمانه تولیدشده توسط Generator نمایش داده شده است. در این شکل سه ستون وجود دارد:

* **Original**: تصویر اصلی که به مدل داده می‌شود.
* **Adversarial**: تصویر خصمانه‌ای که توسط Generator ساخته شده است.
* **Difference**: نقشه‌ی اختلاف بین تصویر اصلی و تصویر خصمانه.

A collage of images of boats

Description automatically generated

شکل 23- نمونه‌هایی از تصاویر واقعی و خصمانه به همراه تفاوت آنها در مدل targeted

**تفاوت‌های مهم در تصاویر هدفدار:**

* در تمامی تصاویر خصمانه، نویز وارد‌شده به‌گونه‌ای است که به‌طور معمول برای چشم انسان کم‌وبیش قابل‌مشاهده نیست.
* تفاوت‌ها در بخش سوم، یعنی نقشه‌ی اختلاف (Difference)، به‌وضوح مشخص است. این نقشه‌ها به ما نشان می‌دهند که دقیقاً کدام قسمت از تصویر تغییرات بیشتری داشته‌اند و چطور این تغییرات بر نتایج پیش‌بینی تأثیر گذاشته است.

**۳. نمودار هیستوگرام قطعیت مدل هدف در طبقه‌بندی**

نمودار زیر، توزیع **Confidence** (اعتماد مدل هدف در پیش‌بینی برچسب) را برای **نمونه‌های عادی** (آبی) و **نمونه‌های خصمانه** (قرمز) روی مجموعه‌ی آزمایشی نشان می‌دهد:

A graph of a graph

Description automatically generated with medium confidence

شکل 24- هیستوگرام قطعیت مدل targeted

**تفاوت با حمله‌ی غیرهدفدار:**

* در **نمونه‌های عادی** (آبی)، اکثر پیش‌بینی‌ها با اعتماد بالا (نزدیک به ۱) انجام می‌شوند.
* در **نمونه‌های خصمانه** (قرمز)، بیشتر نمونه‌ها دارای اعتماد کم‌تر هستند. این به این معنی است که مدل هدف در پیش‌بینی برچسب موردنظر با اطمینان کمتری عمل می‌کند. همچنین مشاهده می‌شود که تعداد زیادی از نمونه‌ها در بازه‌ی ۰.۴ تا ۰.۷ تمرکز دارند که نشان‌دهنده‌ی افت اعتماد مدل هدف در مواجهه با نمونه‌های خصمانه است.

**۴. جمع‌بندی**

در مقایسه با حمله‌ی غیرهدفدار که صرفاً به هدف ایجاد اشتباه در مدل هدف تمرکز داشت، حمله‌ی هدفدار قادر است مدل هدف را به سمت کلاس خاصی هدایت کند. نتایج نشان‌دهنده موفقیت بالای حمله‌ی هدفدار است:

* **نرخ موفقیت کلی حمله** در حمله‌ی هدفدار **۸۵.۴۵٪** است که به‌شکل قابل‌توجهی بالاتر از **۷۷.۱۰٪** در حمله‌ی غیرهدفدار قرار دارد.
* **دقت بالا در کلاس‌های هدف‌دار** مانند **automobile** و **truck**، به این معنی است که Generator توانسته به‌طور مؤثر این کلاس‌ها را هدف قرار دهد.
* **نمودار هیستوگرام اعتماد** در حمله‌ی هدفدار نشان‌دهنده کاهش اعتماد مدل هدف به‌خصوص در نمونه‌های خصمانه است که بر اساس افت اعتماد و افزایش نویز در پیش‌بینی‌ها قابل‌مشاهده است.

نتایج کلی این گزارش تأیید می‌کند که حمله‌ی هدفدار به‌طور مؤثری توانسته بر عملکرد مدل هدف تأثیر گذاشته و با هدایت مدل به سمت کلاس‌های خاص، نرخ موفقیت بالایی کسب کرده است، اما از طرفی تصاویر تولید شده تفاوت بیشتری با واقعیت دارند و اندکی دور تر از واقعیت شده اند. شاید با انتخاب متفاوت پارامترهای بخش loss می‎‌توانستیم به تصاویری با تفاوت کمتر با واقعیت برسیم.