دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر

افزایش تنوع خروجی‌ها در معماری GAN در جهت مقاوم سازی شبکه‌های عصبی در برابر نمونه‌های تقابلی

پایان‌نامه برای دریافت درجه کارشناسی

در رشته مهندسی کامپیوتر گرایش نرم افزار

نام:

سید احمد عبداله پوری‌حسینی

شماره دانشجویی

810194358

استاد راهنما :

دکتر اعرابی

این پایان‌نامه در راستای پروژه‌ی تحقیقاتی کارشناسی ارشد محمد مهدی درخشانی و به صورت مشترک انجام شده است.

بهمن ماه 1398


بسم الله الرحمن الرحيم

تعهدنامه اصالت اثر

باسمه تعالی

اینجانب سید احمد عبداله پوری‌حسینی تائید می کنم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آنها استفاده شده است مطابق مقررات ارجاع گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم سطح یا بالاتر ارائه نشده است.

کلیه حقوق مادی و معنوی این اثر متعلق به دانشکده فنی دانشگاه تهران می باشد.

نام و نام خانوادگی دانشجو :

امضای دانشجو :

با تشکر از دکتر مهدیه سلیمانی و علی دیبا که به پایان رساندن این پایان نامه بدون کمک‌های ارزشمندشان امکان پذیر نمی‌بود.

چکیده[1]

اغتشاش‌های تقابلی[2] نشان‌دهنده‌ی نقطه ضعفی بزرگ در مورد شبکه‌های عصبی عمیق بوده و خطری بزرگ برای استفاده‌ی عملیاتی از آن‌ها هستند. اکثر روش‌های موجود برای تولید این اغتشاش‌ها یک مسأله‌ی بهینه سازی را حل می‌کنند که جواب آن یک تک اغتشاش از بین توزیع انبوه اغتشاش‌های موجود برای یک طبقه‌بند مشخص است. اما این تک اغتشاش‌ها اولا اطلاعات کمی در مورد طبقه‌بند هایی که توسط فریب می‌دهند در اختیار ما می‌گذارند، و ثانیا تنوع پایین آن‌ها به این معنی است که برای ساخت مدل‌های مقاوم به این اغتشاش‌ها مفید نخواهند بود. مقاله‌ی NAG: Network for Adversary Generation پیشنهاد استفاده از یک شبکه‌ی عصبی برای یادگیری تنوع موجود در توزیع اغتشاش‌های مربوط به یک طبقه‌بند را می‌دهد، اما با اینکه این روش گام بزرگی در راستای یادگیری این تنوع برمی‌دارد، نتایج آن با حالت ایده‌آل بسیار فاصله دارند؛ چرا که با اعمال این روش، تعداد کلاس‌هایی که طبقه‌بند تحت حمله ورودی‌های خود را با آن‌ها اشتباه می‌گیرد بسیار محدود است.

در این پایان نامه، ما با الهام از ایده‌ی Conditional GANs و مبحث حملات هدف‌دار[3] در ادبیات نمونه‌های تقابلی تابع هزینه‌ی جدیدی را، به همراه تغییراتی در ساختار شبکه و روش یادگیری پیشنهاد می‌دهیم که می‌تواند تنوع خروجی‌های این روش را به شکل قابل توجهی افزایش بدهد. این تابع هزینه، به جای اینکه از دو مولفه، یکی برای یادگیری فریب‌دهی و دیگری برای بالابردن تنوع خروجی استفاده بکند، از یک مولفه‌ی جدید استفاده می‌کند که هر دوی این خواص را به طور همزمان در خود دارد. نتایج تجربی ما نشان می‌دهند که شبکه‌های آموزش داده شده با تابع هزینه‌ی جدید، بر روی طبقه‌بندهای هدف resnet50، vgg16 و googlenet تنوع خروجی‌ها را به طور میانگین ۸۹.۳% نسبت به روش قبلی افزایش می‌دهند. این پایان‌نامه، گامی در راستای یادگیری بهتر تنوع در توزیع اغتشاش‌های تقابلی مدل‌های یادگیری ماشین بوده، که به درک بهتر ما از این پدیده کمک کرده و در نهایت می‌تواند به ساخت مدل‌های مقاوم نسب به آن منجر شود.

کلمات کلیدی:

نمونه‌های تقابلی - قوام در برابر نمونه‌های تقابلی - شبکه‌های مولد هم‌آورد[4] – تنوع در شبکه‌های مولد هم‌آورد – یادگیری عمیق

[1] Abstract
[2] adversarial perturbations
[3] targeted attacks
[4] generative adversarial networks

فهرست مطالب

فهرست علائم اختصاری

CNN	convolutional neural network
GAN	generative adversarial network
NAG	Network for Adversary Generation
CGAN	conditional GAN
UAP	universal adversarial perturbation
MLP	multilayer perceptron
DCGAN	deep convolutional GAN

فصل 1

فصل 1:

مقدمه و بیان مساله

در این فصـــل نخســـت به بیان مقدمات کار، تاریخچه‌ای کوتاه از مســـاله تحقیق و انگیزه‌ی انجام آن پرداخته، سپس مساله و موضوع مورد بررسی در این پایان‌نامه و اهداف کلی تحقیق را بیان کرده و در نهایت به ساختار پایان‌نامه‌ی پیش رو اشاره خواهیم کرد.

1-1- مقدمه

آسیب پذیری نسب به نمونه‌های تقابلی[1] یکی از مهم‌ترین و شناخته‌شده‌ترین ایرادات مدل‌های یادگیری ماشین مدرن از جمله مدل‌های یادگیری عمیق، و یک مثال مهم از ضعف این مدل‌ها در تعمیم‌پذیری به نمونه‌های خارج از توزیع داده‌های یادگیری است [9] . شگفت‌آور است که چطور در شرایط i.i.d[2] شبکه‌های عصبی عمیق[3] روی بسیاری از تسک‌ها عملکرد فوق-انسانی دارند [1] در صورتی که اغتشاش اتی[4] بسیار کوچک ولی هدفمند وجود دارند که می‌توانند باعث خطاهای نامنتظره‌ای از سمت این شبکه‌ها بشوند.

محققان تلاش کرده‌اند که این پدیده را با فرضیاتی در مورد رفتار خطی مدل ها ([10]) ، داده‌ی آموزش محدود ([2])، و یا عدم حساسیت نسبت به تغییرات معنی‌دار ورودی ([3]) توضیح بدهند.

یافته‌های اخیر توسط موسوی دزفولی و همکاران [14] و مپوری[5] و همکاران نشان می‌دهند که اغتشاشـ اتی وجود دارند که اضافه کردن آن‌ها به ورودی می‌تواند اکثر مدل‌های یادگیری عمیق state-of-the-art را بر روی اکثر تصاویر ورودی فریب بدهد. به این اغتشاشـ ات، نام «اغتشاشـ ات تقابلی جامع[6]» داده شده است. همچنین نشان داده شده است که نمونه‌های تقابلی خاصیت تعمیم‌پذیری بین مدل ها دارد [9, 10]. یعنی اغتشاشـ ات یادگرفته شده برروی یک مدل، می‌توانند یک مدل دیگر را نیز فریب بدهند، حتی اگر مدل دوم معماری متفاوتی داشته باشد و بر روی داده‌های متفاوتی آموزش دیده باشد. این خاصیت انجام حملات black-box را بر روی مدل‌های یادگیری ماشین ممکن می‌سازد. [5]

ترکیب دو یافته‌ی اخیر نشان می‌دهد که اغتشاشات تقابلی، تهدیدی جدی برای استفاده از مدل‌های یادگیری ماشین در دنیای واقعی، به خصوص در کاربردهای مرتبط با امنیت و حریم شخصی هستند. در نتیجه ساخت مدل‌هایی که نسب به این اغتشاشـ ات مقاوم باشند یک ضرورت برای استفاده‌ی عملیاتی از دستاوردهای علم یادگیری ماشین است. از سوی دیگر، این اغتشاشـ ات فهم ما نسـ بت به این مدل‌ها و روش‌های موجود برای آموزش آن‌ها را به چالش می‌کشد. در نتیجه به نظر بررسی دقیق‌تر این پدیده می‌تواند به درک عمیق‌تر مکانیزم‌های موجود در این مدل‌ها کمک شایانی بکند. به همین خاطر، علل وجود

[1] adversarial examples
[2] independent identically distributed
[3] deep neural networks
[4] perturbations
[5] Mopuri
[6] universal adversarial perturbations

این اغتشاشات و روش‌های مقابله با آن ها، یک موضوع پژوهشی بسیار فعال است. [12]

اکثر روش‌های موجود، چه مستقل از تصویر (یعنی اضافه‌کردن آن به اکثر تصاویر موجود، مدل یادگیری ماشین را فریب می‌دهد) [4 ,14]، و چه وابسته به آن [10 ,9 ,5] مسئله‌ی بهینه‌سازی ای را حل می‌کنند که در نتیجه‌ی آن فقط یک اغتشاش تقابلی برای مدل تولید می‌شود. در واقع این روش‌ها یک اغتشاش (δ) از میان توزیع اغتشاشات (Δ) را میابند. همچنین مشاهده شده است که خروجی این روش ها (مثلا UAP [14]) در طی چندین اجرای متوالی، تغییر چندانی نمی‌کند. در نتیجه خروجی این روش‌ها طیف بسیار محدود و نامتنوعی از این اغتشاشات را در اختیار ما می‌گذارد.

روش NAG [13]، تلاشی در راستای افزایش تنوع خروجی ها به وسیله‌ی مدل‌کردن مستقیم توزیع این اغتشاشات بوده، و مسلما خروجی آن تنوع بسیار بیش‌تری نسبت به روش‌های قبل از خود داشته است. ولی با بررسی دقیق‌تر دیده می‌شود که این روش نیز هنوز فاصله‌ی زیادی از یافتن طیف کامل تمام اغتشاش ات ممکن برای یک طبقه بند را دارد. در واقع بررسی خروجی این روش، نشان می‌دهد که وقتی اغتشاش ات حاصل را به تصاویر مجموعه‌داده اعمال می‌کنیم، اکثر مواقع طبقه بند هدف ورودی خود را با چند برچسب خاص که نسبت به بقیه‌ی برچسب ها غلبه‌ی شدیدی دارند (حدود ۱۰ کلاس) اشتباه می‌گیرد. غلبه‌ی این برچسب ها به سایرین به حدی زیاد است که تعداد دفعاتی که طبقه‌بند هدف خروجی خود را با سایر برچسب‌ها اشتباه می‌گیرد به طور نسبی برابر صفر است. این باعث می‌شود که در صورتی که بخواهیم از این نمونه‌ها برای ساخت یک مدل مقاوم استفاده بکنیم، (چه از طریق یادگیری تقابلی[1] [6] و چه از طریق ساخت یک ماژول مدافع) کار این مدل مقاوم بسیار ساده باشد، و بتواند با یادگیری نمونه‌هایی که روی آن چند کلاس محدود تولید می‌شوند به دقت بالایی دست بیابد. این در صورتی است که برای یک دفاع موثر، نیاز است ویژگی‌هایی که شبکه را نسبت به حملات تقابلی روی تمام کلاس‌ها (یا حداقل اکثر آن‌ها) آسیب پذیر می‌کند یاد گرفته شود و در نتیجه یادگیری که از آن صحبت شد، به احتمال زیاد یادگیری مفیدی نبوده است. همچنین اغتشاش ات محدود به چند کلاس غالب، به نظر نمی‌رسد بتوانند منبع خوبی برای مطالعه‌ی خواصی از اغتشاشات باشند که محدود به کلاس‌های خاصی نبوده و کلی هستند.

در نتیجه، هدف ما در این مطالعه، این بوده است که با ایجاد تغییراتی در ساختار و تابع هزینه‌ی مدل پیشنهاد شده توسط NAG، تنوع خروجی‌های این مدل را افزایش بدهیم. این کار کمک خواهد کرد که مدل

[1] adversarial training

بهبود یافته، با تولید خروجی‌های متنوع تر هم بتواند بهتر ما را در فهم پدیده‌ی نمونه‌های تقابلی کمک بکند، و هم با استفاده از آن بتوان مدل‌های مقاوم تری نسبت به این پدیده ساخت.

2-1- ساختار پایان‌نامه

در ادامه‌ی این پایان نامه، ابتدا در فصل۲ به بررسی مفاهیم اولیه و پیش‌زمینه‌های مورد نیاز برای فهم بهتر مطالبی که در ادامه آمده است می‌پردازیم. سپس در فصل ۳ مروری بر کارهای مرتبط در این حوزه خواهیم داشت، و البته روش NAG که baseline کار ما محسوب می‌شود را با دقت بیش‌تری بررسی می‌کنیم. در فصل ۴ روش پیشنهادی خود برای حل مشکل مطرح شده در مقدمه، را به طور مفصل شرح داده و در فصل بعدی نیز نتایج حاصل از آزمایشات تجربی را بیان می‌کنیم. فصل ۶ با ارائه‌ی یک خلاصه از مطالب ارائه شده به جمع بندی این پایان‌نامه می‌پردازد و پیشنهادهایی برای ادامه‌ی مسیر به علاقمندان این حوزه ارائه می‌کند.

فصل 2

فصل 2: مفاهیم اولیه و پیش زمینه

در فصـــل پیش رو مقدمات، مفاهیم اولیه و پیش‌زمینه‌هایی را که جهت درک هر چه بهتر موضوع‌های مطرح شده در این پایان‌نامه مورد نیاز است، ارائه خواهد شد.

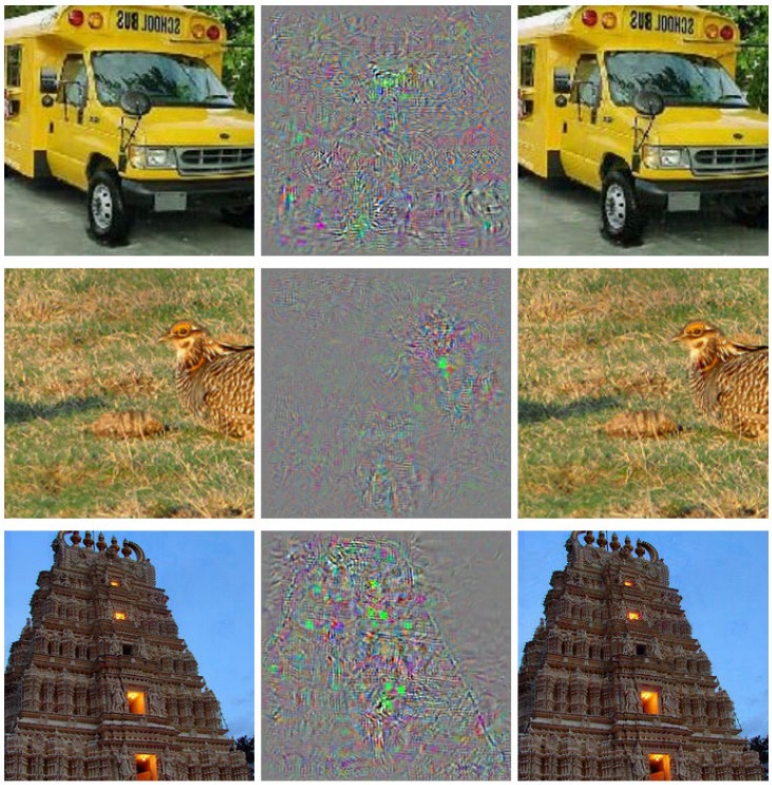
2-1- نمونه‌های تقابلی

شبکه‌های عصبی عمیق مدل‌های قوی‌ای برای یادگیری هستند که عملکرد بسیار خوبی روی تسک‌های مربوط به بینایی و تشخیص صدا دارند [7,8]. اما از آنجایی که یادگیری این شبکه‌ها به صورت اتوماتیک توسط پس‌انتشار خطا انجام می‌شود، معمولا فهم و تفسیر پارامترهای حاصل از یادگیری سخت است، و شبکه‌ی آموزش یافته ممکن است دارای خواص غیرشهودی و عجیبی باشد. [9] یکی از این خواص مربوط به پایداری این شبکه‌ها در برابر اغتشاش‌های کوچکی است که به ورودی اعمال می‌شوند. اگر یک شبکه‌ی عصبی عمیق state-of-the-art را که برای عمل تشخیص اشیاء[1] تعمیم‌دهی خوبی دارد را در نظر بگیریم، انتظار داریم که نسبت به تغییرات بسیار کوچک روی ورودی خود پایدار باشد، چرا که این تغییرات نمی‌توانند کلاسی که عکس متعلق به آن است را عوض کنند. اما، می‌توان نشان داد که می‌توان اغتشاش‌های بسیار کوچکی را طوری ساخت که با اینکه با چشم انسان به سختی دیده می‌شوند، می‌توانند به طور دلخواه خروجی یک شبکه‌ی عصبی را تغییر بدهند. این اغتشاش‌ها از تغییر دادن ورودی به شکلی که خطای طبقه‌بندی را بیشینه بکنند به دست می‌آیند. به نمونه‌هایی که این چنین تغییرداده شده‌اند، «نمونه‌های تقابلی» و به اغتشاش‌های سازنده «اغتشاش تقابلی» می‌گوییم. تعریف رسمی این مفهوم در ادامه آمده است.

فرض کنیم طبقه‌بندی به شکل $f: \mathbb{R}^m \to \{1 \dots k\}$ داشته باشیم که تصاویر با m پیکسل را به یک طبقه می‌نگارد. برای یک تصویر $x \in \mathbb{R}^m$ و طبقه‌ی هدف $l \in \{1 \dots k\}$ مطلوب ما، حل کردن مسئله‌ی بهینه سازی زیر است:

$$\text{miniminze } \| r \|_2 \text{ subject to:}$$
$$f(x + r) = l$$
$$x + r \in [0,1]^m$$

[1] object recognition



شکل (2-1) نمونه‌های تقابلی تولید شده برای AlexNet. تمامی تصاویر ستون راست، به عنوان «شتر مرغ» توسط این شبکه طبقه‌بندی می‌شوند.

در واقع ما سعی داریم که r را طوری بیابیم نرم آن کمینه بوده، اما بتواند با اضافه شدن به ورودی طبقه‌بند خروجی آن را به کلاس l تغییر بدهد. شکل ۱-۲ چند نمونه از چنین نمونه‌های تقابلی را نشان داده است. در ستون سمت چپ، تصویر اصلی را داریم. ستون سمت راست، تصویر آشفته شده است، به شکلی که تمامی تصاویر این ستون به عنوان "Ostrich, Struthio camelus" توسط شبکه‌ی AlexNet شناخته می‌شوند. ستون وسط نیز اغتشاش تقابلی است که به تصاویر ستون چپ اضافه شده است تا ستون راست را بسازد.

2-2- شبکه‌های مولد هم‌آورد (GAN)

GAN ها در واقع روشی هستند برای آموزش مدل‌های مولد[1] در چهارچوب یک پروسه‌ی تقابلی که در آن به طور همزمان دو مدل آموزش داده می‌شود: یک مدل مولد G که توزیع داده‌های واقعی را

[1] generative

یاد می‌گیرد، و یک مدل تمایزی[1] D که احتمال این را تخمین می‌زند که آیا نمونه‌ی ورودی یک داده‌ی واقعی است و یا تولید شده توسط G (مصنوعی) است. روش آموزش G این است که کاری کنیم که احتمال خطای D بیشینه شود.

معمولا در این چهارچوب، هر دوی G و D، MLP (Multilayer Perceptron) هستند. اگر که فرض کنیم ورودی G یک بردار noise به شکل $z \sim p_z(z)$ است، و اگرتوزیع داده‌های واقعی را با $p_{data}(x)$ نشان دهیم، همچنین اگر $D(x)$ پیش‌بینی D در مورد این باشد که x به چه احتمالی داده‌ی واقعی است، آنگاه می‌توان پروسه‌ی آموزش را به شکل یک بازی minimax با تابع ارزش[2] زیر بیان کرد:

$$\min_G \max_D V(D,G)$$

که در آن:

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log\left(1 - D\big(G(z)\big)\right)]$$

یعنی در واقع D سعی می‌کند که احتمال تشخیص درست خود را چه بر روی داده‌های واقعی و چه داده‌های تولید شده توسط G افزایش بدهد، در حالی که G سعی می‌کند احتمال خطا کردن D را افزایش بدهد.

به هنگام یادگیری، ابتدا پارامترهای G به کلی freeze می‌شوند، و برای چند iteration فقط D آموزش داده می‌شود. سپس برعکس همین اتفاق می‌افتد: پارامترهای D به کلی freeze شده و برای چند iteration فقط G آموزش داده می‌شود. این حلقه آن‌قدر تکرار می‌شود تا G به نقطه‌ای برسد که خروجی‌های آن از منظر انسان نیز واقعا طبیعی به نظر برسند.

از این چهارچوب در زمان‌هایی استفاده می‌شود که می‌خواهیم شبکه‌ی مولدی داشته باشیم، اما تعریف تابع هزینه برای آن سخت تر از این است که آن را به شکل تحلیلی انجام بدهیم. (مثلا نوشتن تابع هزینه برای تشخیص اینکه یک عکس از منظره‌ی طبیعی، چقدر «واقعی» به نظر می‌رسد.) در نتیجه این تابع هزینه را با یک شبکه‌ی D جایگزین می‌کنیم، که این شبکه این تابع هزینه را فقط با داشتن نمونه‌هایی از آنچه برای ما مطلوب است یاد می‌گیرد.

[1] discriminative

[2] value function

فصل 3

فصل 3: **کارهای مرتبط**

در این بخش به توضیح آثاری که ارتباط نزدیکی با این مطالعه دارند می‌پردازیم.

3-1- بررسی کلی

معرفی پدیده‌ی اغتشاشات تقابلی [9]، شوک نسبتا بزرگی بر بدنه‌ی روش‌های یادگیری عمیق[1] بود. قابلیت این اغتشاش ات برای تعمیم‌داده شدن به مدل‌هایی که از قبل دیده نشده‌اند، راه را برای انجام حملات black-box به سیستم‌های درحال استفاده باز می‌کند ([10]). علاوه بر این، وجود اغتشاش ات مستقل از تصویر[2] (یعنی اغتشاش اتی که می‌توانند روی اکثر تصاویر موجود اثر فریب‌دهندگی داشته باشند.) [14] نیز جنبه‌ی دیگری از ضعف مدل‌های یادگیری عمیق را مشخص می‌کند که در ترکیب با خاصیت تعمیم به مدل‌های جدید این اغتشاش ات می‌تواند خطرناک باشد.

در راستای تولید این اغتشلشات، اکثر روش‌های موجود، مسئله‌ی بهینه‌سازی‌ای را حل می‌کنند که پاسخ آن تنها یک اغتشاش تقابلی است ([9, 10, 14]). تنها مقاله‌ای که روی یافتن توزیع چنین اغتشلشاتی کار کرده است، مقاله‌ی NAG [13] است. همچنین این مقاله به همراه [11] تنها آثاری بوده اند که از یک شبکه‌ی عصبی به عنوان مولد اغتشاشات تقابلی استفاده کرده اند، ولی [11]، از یک شبکه‌ی عصبی برای تبدیل یک عکس به نمونه‌ی تقابلی نظیر استفاده می‌کند، و هدف آن مدل‌سازی توزیع این اغتشاشات نیست.

تا جایی که ما می‌دانیم، اثری وجود ندارد که بر روی افزایش تنوع خروجی‌های شبکه‌ی عصبی مولد اغتشاشات تمرکز کرده، و این تنوع را تا حدی که ما به آن دست یافته‌ایم بالا برده باشد.

---

[1] deep learning

[2] image-agnostic

3-2- روش baseline: NAG[1]

این مقاله [13]، ایده‌ای را معرفی می‌کند که پایه و baseline روش ما قرار دارد، و روش پیشنهادی ما در واقع یک تغییر در ساختار ایده‌ی اصلی این مقاله است. به همین دلیل این مقاله با دقت بیش‌تر و با در نظر گیری جزییات بررسی شده است.

3-2-1- نمادگذاری

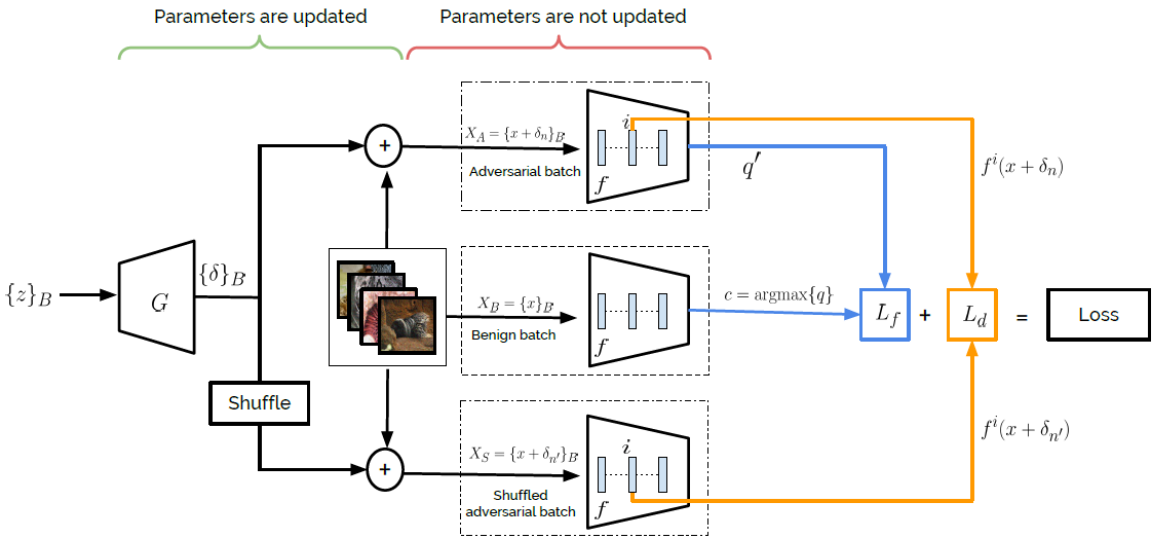
در وهله‌ی اول لازم است ذکر کنیم که مدل‌هایی که در این مقاله، و همچنین در پایان‌نامه‌ای که می‌خوانید قصد داریم با استفاده از اغتشاش‌های تقابلی فریب بدهیم، شبکه‌های عصبی، یا به طور دقیق تر شبکه‌های عصبی کانولوشنی[2] (CNN) هایی هستند که برای انجام عمل تشخیص اشیاء[3] استفاده می‌شوند. توزیع داده‌هایی که این طبقه‌بندها روی آن‌ها آموزش داده شده‌اند را با X، و یک داده از این توزیع را با x نمایش می‌دهیم. همچنین CNN ای که قصد فریبش را داریم با نماد f نمایش داده، و خروجی آن در لایه‌ی i ام را با نماد $f^i(x)$ نمایش می‌دهیم. طبقه‌ی پیش‌بینی شده برای داده‌ی x را با $\hat{k}(x)$ نمایش می‌دهیم. بردار خروجی لایه‌ی softmax (که لایه‌ی آخر طبقه‌بند است) را با q نمایش می‌دهیم، و اندیس j ام این بردار که احتمال پیش‌بینی شده برای طبقه‌ی j ام است را با q_j نمادگذاری می‌کنیم. اغتشاش جمع‌شونده[4] ای که هدف آن فریب طبقه‌بند است را با δ و محدودیت روی نرم بی‌نهایت ($l_\infty - norm$) آن را با ξ نشان می‌دهیم. با توجه به این توضیحات، می‌توان توزیع نمونه‌های تقابلی را به شکل زیر تعریف کرد:

$$\Delta = \{\delta: \hat{k}(x+\delta) \neq \hat{k}(x)\ for\ x \sim X\ and\ \| \delta \|_\infty < \xi\}$$

[1] Network for Adversary Generation
[2] convolutional neural network
[3] object recognition
[4] additive

3-3- ساختار کلی

در این مقاله، هدف اصلی مدل‌کردن توزیع نمونه‌های تقابلی ای است که توزیع آن‌ها در بخش قبل به شکل رسمی تعریف شد. یک نمای کلی از این روش را می‌توانید درشکل 1 ببینید.



شکل (3-1) 1 نمایی کلی از روش baseline

در همین راستا، برخی تغییرات در چهارچوب اولیه‌ی ساختار GAN داده شده است: ۱) تمایزدهنده با CNN هدف جایگذاری شـــده اســـت، که از قبل آموزش دیده، و وزن‌های آن اکنون در حالت freeze هستند.۲) تابع هزینه‌ی تقابلی[1] که معمولا برای آموزش GAN ها استفاده می‌شود، با یک تابع هزینه‌ی جدید که از دو مولفه‌ی فریب‌دهی و تنوع تشکیل شده است جایگزین شده است

در واقع هدف در این روش (و روش ما) آموزش یک شبکه‌ی عصبی است که بتواند طبقه‌بند هدف را فریب بدهد. معماری این شبکه، مشابه شبکه‌ی مولد[2] در DCGAN (Deep Convolutional GAN) ها هستند، و یک بردار noise تصادفی را توسط چندلایه‌ی deconvolution به ابعاد یک تصویر درمی‌آورند. حال به سراغ مولفه‌هایی می‌رویم که تابع هزینه را برایمان می‌سازند.

[1] adversarial loss
[2] generator

2-3-3- مولفه‌ی فریب‌دهی

طبیعتا برای اینکه نمونه‌های تولید شده توسط شبکه، نمونه‌ی «تقابلی» باشند، باید بتوانند طبقه‌بند هدف را فریب بدهند تا عکس‌های آشفته شده را به عنوان عکسی از طبقه‌ی دیگر تشخیص بدهد. این خاصیت از نمونه‌ها را ما در قالب مولفه‌ی فریب‌دهی بیان می‌کنیم. در واقع در اینجا نیز مثل سناریویی که در GAN ها داریم، ما توزیع هدف (Δ) را نداریم، اما یک خاصیت از آن توزیع را داریم که به ما در یافتن آن کمک می‌کند. در اینجا نیز آن خاصیت، خاصیت فریب‌دهی است.

ما طبقه‌ی تشخیص داده شده توسط طبقه‌بند هدف برای یک نمونه‌ی دست نخورده‌ی x را *تشخیص سالم*[1]*(c)*، و تشخیص همان شبکه برای آن نمونه که آشفته شده است ($x + \delta$) را *تشخیص تقابلی*[2] می‌نامیم. به طور مشابه، خروجی لایه‌ی softmax را قبل و بعد از اضافه کردن اغتشاش δ، به ترتیب q و $\acute{q}$ می‌نامیم. در حالت ایده‌آل اغتشاش δ باید طوری باشد که با اضافه شدن آن به ورودی، تشخیص سالم شبکه تغییر کند. برای رخ دادن این امر، پس از اضافه شدن اغتشاش باید اطمینان شبکه از تشخیص سالم ($\acute{q}_c$) کاهش پیدا بکند، که از آنجایی که لایه‌ی softmax جمع احتمالات را ۱ نگه می‌دارد، این خود به خود به معنای افزایش احتمال طبقات دیگر خواهد بود که در نهایت منجر به عوض شدن تصمیم شبکه خواهد شد. در نتیجه می‌توان با قرار دادن تابع هزینه به شکل زیر، در جهت کمینه کردن $\acute{q}_c$ حرکت کرد:

$$L_f = -\log(1 - \acute{q}_c)$$

می‌توان این توضیحات را به شکل تصویری در شکل 1 نیز دید. این شکل این مولفه را با رنگ آبی نشان می‌دهد.

[1] benign prediction
[2] adversarial prediction

3-3-3- مولفه‌ی تنوع

مولفه‌ی فریب‌دهی، فقط شبکه را تشویق می‌کند که طوری یادگیری را انجام بدهد که بتواند نرخ فریب‌دهی بالایی را تضمین بکند. اما اگر که شبکه یک اغتشاش خوب پیدا بکند که بتواند طبقه‌بند هدف را به خوبی فریب بدهد، (که می‌دانیم این کار ممکن است [14]) می‌تواند از آن به بعد ورودی خود را همواره نادیده گرفته، و به ازای تمامی ورودی ها صرفا همان اغتشاش با کیفیت را در خروجی تولید کند. اما این راه‌حل یک ایراد اساسی و واضح دارد: هدف ما یافتن **توزیع** نمونه‌های تقابلی Δ برای یک طبقه‌بند هدف است، نه فقط یافتن یک نمونه‌ی تقابلی. برای اینکه جلوی این ایراد را بگیریم، مولفه‌ی تنوع را به تابع هزینه‌ی خود اضافه می‌کنیم که شبکه را تشویق می‌کند فضای وسیع تری از نمونه‌های تقابلی را بررسی کند.

این مولفه، در سطح mini-batch عمل می‌کند. در واقع این مولفه می‌کوشد که نمونه‌های تقابلی تولید شده برای یک mini-batch از تصاویر ورودی، با هم متفاوت باشند. اما این کار به طور غیر مستقیم و از طریق افزایش فاصله بین feature embedding های تصاویر آشفته شده انجام می‌شود. به طور دقیق تر، برای یک جفت اغتشاش مثل δ و $\acute{\delta}$ هدف ما افزایش فاصله‌ی بین $f^i(x+\delta)$ و $f^i(x+\acute{\delta})$ است:

$$L_d = -\sum_{n=1}^{B} d(f^i(x_n+\delta_n), f^i(x_n+\delta_{\acute{n}}))$$

که در آن $\acute{n}$ یک اندیس تصادفی در بازه‌ی $[1,B]$ است که $\acute{n} \neq n$، B اندازه‌ی batch بوده و x_n و δ_n به ترتیب n امین داده و اغتشاش موجود در mini-batch هستند. همچنین d یک متر اندازه‌ی دلخواه (به طور مثال فاصله‌ی اقلیدسی بین دو بردار ویژگی) است. در پیاده‌سازی نهایی، انتخاب $\acute{n}$ به صورت مستقیم نخواهد بود،بلکه به ازای هر mini-batch از اغتشاش‌ها، یک mini-batch بُرخورده[1] نیز تولید می‌شود که در هنگام عملیات بُرزدن دقت می‌کنیم که هیچ اندیسی در جای اولیه‌ی خود قرار نگیرد. بلوک نارنجی رنگ در شکل 1، مولفه‌ی تنوع را نشان می‌دهد.

[1] shuffled

4-3-3- تابع هزینه‌ی نهایی

تابع هزینه‌ی نهایی از ترکیب دو مولفه‌ی قبلی به شکل زیر حاصل می‌شود:

$$L = L_f + \lambda L_d$$

که طبق آزمایشات نویسنده‌ی مقاله، مقدار λ برابر 1 انتخاب شده است تا اهمیت یکسانی به دو مولفه داده شود.

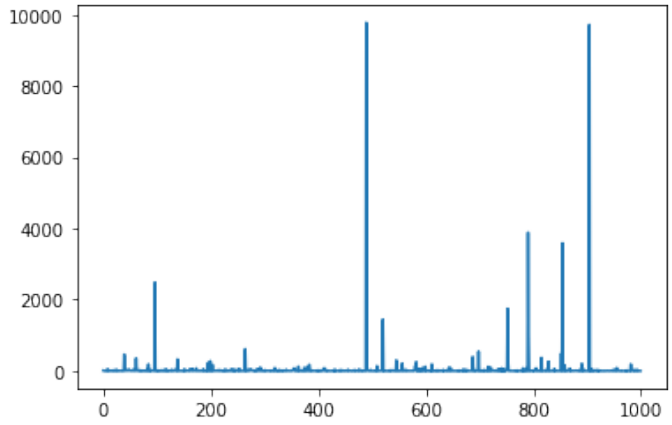
فصل 4

فصل 4: روش پیشنهادی

در این فصل به شکل مشروح و با دقت به جزیات به توضیح روش پیشنهادی می‌پردازیم.

4-1- انگیزه

همان‌طور که قبل‌تر نیز توضیح داده شد، هدف ما از این مطالعه بالابردن تنوع خروجی‌های شبکه‌ی مولد است، چرا که ساختار پیشنهاد شده در NAG به همراه مولفه‌ی تنوع آن، به خوبی از عهده‌ی ایجاد این خروجی های متنوع برنمی‌آیند. برای توضیح این مطلب خوب است به شکل ۱-۴ توجه کنید.این شکل یک هیستوگرام از کلاس‌هایی است که شبکه به آن fool شده است (fool شدن به یک کلاس، یعنی اینکه شبکه ورودی خود را با آن کلاس اشتباه بگیرد). برای رسم آن، ۱۰ نویز تصادفی به شبکه‌ی مولد داده شده ، و ۱۰ اغتشاش نظیر تولید شده توسط شبکه ذخیره شده است. سپس به ازای هر تصویر در مجموعه داده‌ی تست، تمامی این ۱۰ اغتشاش به آن اعمال شده، و تصمیم طبقه‌بند هدف ثبت شده است. در نهایت شمرده‌ایم که شبکه چند بار به هر کلاس fool شده است، یعنی به اشتباه آن کلاس را به عنوان تصمیم خود اعلام کرده است. در این شکل به وضوح دیده می‌شود، که به جز چند کلاس محدود (حدودا ۷ کلاس) عدد بقیه‌ی کلاس ها تقریبا برابر صفر است. این یعنی شبکه‌ی مولد دچار پدیده‌ی mode collapse شده است، که یکی از مشکلات عمده در آموزش GAN ها می‌باشد.



شکل (4-1) 1 هیستوگرام کلاس‌هایی که شبکه به آن fool شده است. محور x نشان‌دهنده‌ی اندیس کلاس است، و محور y نشان دهنده‌ی این که در یکبار چرخش روی مجموعه داده‌ی تست، چند بار شبکه به اشتباه آن کلاس را خروجی داده است.

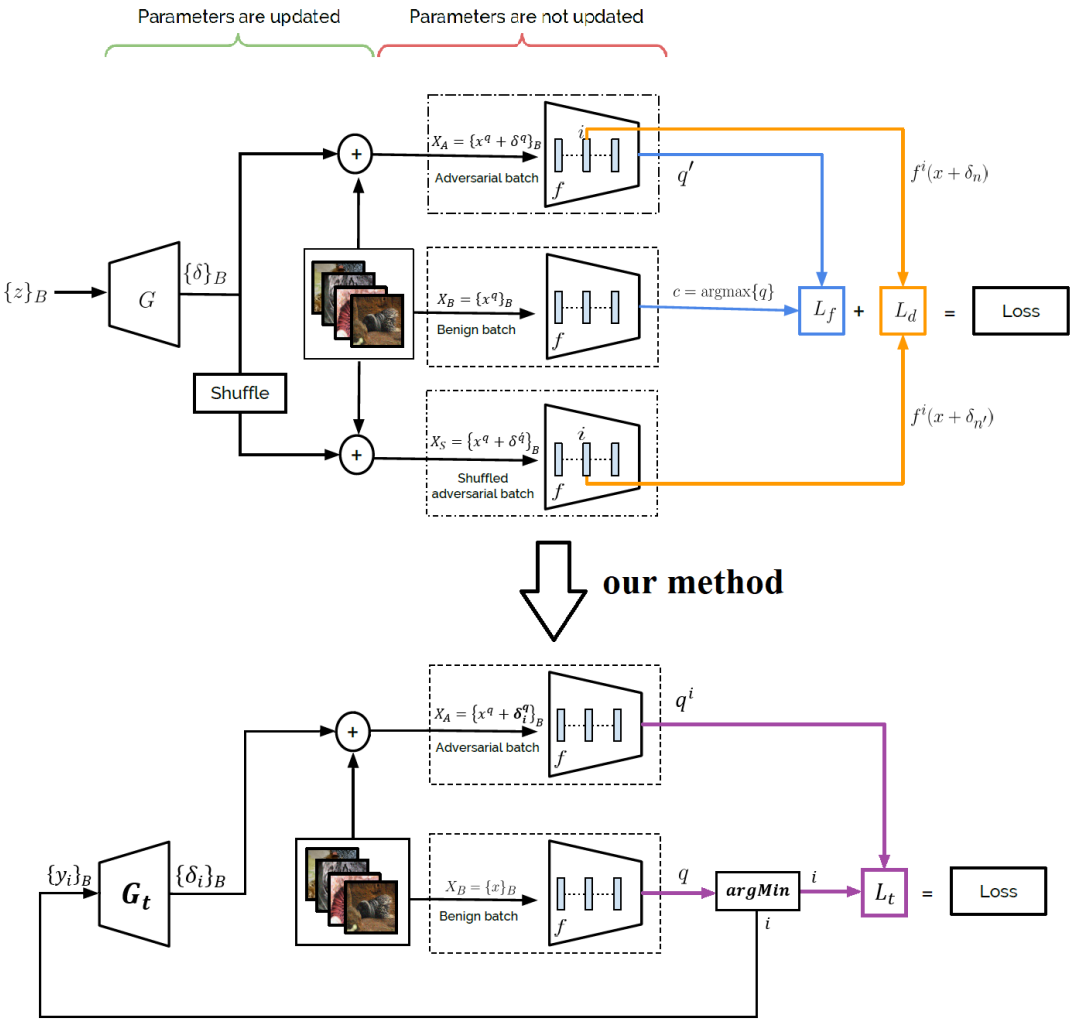
mode collapse پدیده‌ای اســـت که وقتی GAN بخواهد داده‌هایی که variability بالایی را دارند یاد بگیرد ممکن است رخ دهد، و به این معنا است که شبکه تفاوت‌های موجود در داده را به مقدار زیادی نادیده می‌گیرد و تمامی خروجی‌های خود را حول یک یا چند mode محدود تولید می‌کند.

همان‌طور که در مقاله‌ی NAG نیز اشـــاره شـــده، یکی از کاربردهای مهم این شـــبکه‌ی مولد اغتشـاش‌های تقابلی، سـاختن طبقه‌بندهای مقاوم تر نسـبت به نمونه‌های تقابلی از طریق یادگیری تقابلی است. اما برای اینکه این کار به خوبی صورت بگیرد، باید نمونه‌های تولید شده متنوع باشند و نقاط مختلط فضای نمونه‌های تقابلی را پوشش دهند. پدیده‌ی mode collapse در شبکه‌ی مولد به این معنا خواهد بود که خروجی‌های این شـــبکه در حالت فعلی کارایی پایینی برای تولید طبقه‌بندهای مقاوم‌تر خواهند داشـــت. چرا که در هنگام عمل یادگیری تقابلی، طبقه‌بند فقط نیاز دارد نحوه‌ی مقابله با نمونه‌های تقابلی مربوط به تعداد بسیار کمی کلاس (۶-۷) را یاد بگیرد. و این یعنی یادگیری مفیدی در شبکه رخ نخواهد داد.

2-4- تابع هزینه‌ی هدف‌دار[1]

روش پیشـنهادی ما برای بالابردن تنوع خروجی‌های شـبکه‌ی مولد و مقابله با این mode collapse، ترکیبی از ایده‌ی Conditional GAN ها (CGAN) و مبحث حملات هدف‌دار در ادبیات نمونه‌های تقابلی اســـت. در واقع روش NAG از یک مولفه‌ی فریب‌دهی غیرهدف‌دار[2] به منظور ســـاخت نمونه‌های تقابلی، و یک مولفه‌ی تنوع به منظور جبران ضـــعف مولفه‌ی قبلی در تولید خروجی‌های متنوع اســـتفاده می‌کرد. اما روش ما از یک تابع هزینه‌ی هدف‌دار اســـتفاده می‌کند تا هردوی این خاصـــیت‌ها را به طور همزمان به وجود بیاورد. می‌توانید یک نمای کلی از روش ما، که به منظور مقایسه در کنار روش NAG قرار گرفته است را در شکل ۲-۴ ببینید.

[1] targeted
[2] non-targeted



شکل (2-4) نمای کلی از روش پیشنهادی. در سمت چپ تصویر، شبکه‌ی مولد هدف‌دار G_t را می‌بینیم که به جای دریافت یک بردار noise، یک بردار one-hot بیانگر کلاس هدف (y_i) را دریافت می‌کند. تصمیم اینکه i چه باشد با argmin گرفتن از خروجی شبکه به ازای ورودی سالم گرفته می‌شود. (سمت راست شکل) در انتهای نیز با توجه به اینکه چقدر احتمال مربوط به کلاس هدف، یعنی q_i^i بالا رفته است، مقدار هزینه‌ی L_t محاسبه می‌شود.

به عبارت دقیقتر، در هنگام تولید نمونه‌ی تقابلی، ما به جای یک بردار noise تصادفی، به شبکه‌ی مولد یک بردار one-hot y^i به طول ۱۰۰۰ می‌دهیم که فقط در اندیس i برابر ۱، و در بقیه‌ی جاها صفر است. هر اندیس این بردار نظیر یکی از کلاس‌های مجموعه داده‌ی ImageNet [15] است، و اندیسی که ۱ است نشان می‌دهد که ما دوست داریم طبقه‌بند هدف ورودی خود را با چه کلاسی اشتباه بگیرد. سپس، وقتی طبقه‌بند هدف خروجی خود را تولید می‌کند، ما دوست داریم که احتمالی که برای کلاس i پیش‌بینی شده است، بالاترین حد ممکن، یعنی ۱ باشد.

برای تعریف رسمی تر تابع هزینه ابتدا باید در مورد نمادگذاری حرف بزنیم. ما در این مطالعه، از نمادگذاری مورد استفاده در NAG استفاده می‌کنیم، و فقط چند نماد را به آن اضافه می‌کنیم. در NAG، اغتشاش‌های تقابلی با نماد δ نشان داده می‌شدند. ما اغتشاش تقابلی ای که شبکه‌ی مولد به ازای ورودی y^i تولید می‌کند را δ_i می‌نامیم. یعنی:

$$\delta_i = G(y^i)$$

همچنین، در NAG خروجی لایه‌ی softmax طبقه‌بند هدف با q نشان داده می‌شد. ما خروجی لایه‌ی softmax پس از اضافه کردن اغتشاش δ_i را q^i می‌نامیم، و اندیس j ام آن را به شکل q_j^i نمایش می‌دهیم، یعنی احتمالی که شبکه پس از اضافه شدن اغتشاش δ_i برای طبقه‌ی j خروجی می‌دهد. همچنین اندیس j ام بردار y^i را به شکل y_j^i نمایش می‌دهیم. با توجه به این توضیحات، می‌توان تابع هزینه‌ی جدید را به شکل زیر بیان کرد:

$$L_t = -\sum_{j=1}^{c} y_j^i \log \mathrm{q}_j^{\mathrm{i}} = -\log q_l^i$$

که در آن، زیرنویس t مخفف کلمه‌ی targeted است. این تابع در واقع همان تابع هزینه‌ی CrossEntropy است که در قالب مسئله‌ی ما قرار گرفته است. ساده سازی عبارت نهایی با توجه به این نکته است که مطابق تعریف:

$$y_l^i = 1$$

$$y_j^i = 0 \; for \; i \neq j$$

برای کم کردن این تابع هزینه، باید عبارت q_l^i را بیشینه کرد، و این یعنی اگر به شبکه‌ی مولد y_i را ورودی دادیم، باید خروجی شبکه به ازای طبقه‌ی i بیشینه باشد.

واضح است که اگر این loss به خوبی بهینه بشود، شبکه‌ی مولد قادر خواهد بود که طبقه‌بند هدف را به هر کلاس دلخواهی فریب بدهد، یعنی کاری کند که طبقه‌بند ورودی خود را با آن کلاس اشتباه بگیرد. پس این loss مولفه‌ی فریب‌دهی را در خود دارد. از طرفی، با انتخاب y_i های متفاوت برای مولد، می‌توان حداقل ۱۰۰۰ نوع مختلف اغتشاش تولید کرد، که طبیعتا چون مربوط به کلاس‌های مختلفی هستند تفاوت‌های معنی داری با هم خواهند داشت. در نتیجه این loss مولفه‌ی تنوع را نیز در خود دارد.

از آنجایی که اصولا حل یک مسئله‌ی بهینه‌سازی با اهداف چندگانه[1] در حالت کلی سخت تر از حل یک مسئله‌ی بهینه‌سازی با یک تابع هدف است و چالش های خاص خود را دارد، و از آنجایی که به طور خاص در این مسئله آزمایشات ما نشان می‌دهد که trade-off شدیدی بین مولفه‌های مربوط به فریب‌دهی و تنوع در تابع هزینه وجود دارند، در نتیجه یک خاصیت جذاب این تابع هدف از بین بردن این trade-off و ساده‌سازی مسئله به یک تابع هزینه است که با بهبود آن هر دو خاصیت مطلوب حاصل خواهند شد.

3-4- انتخاب اندیس هدف برای مولد

در مورد تابع هزینه‌ای که معرفی شد، هنوز مشخص نشده است که اندیس i (و به تبع آن بردار y_i)چگونه انتخاب می‌شوند. ساده‌ترین ایده‌ی ممکن این است که اگر که پیش‌بینی سالم شبکه بر روی ورودی سالم خود طبقه‌ی j باشد، یا به عبارتی $j = argMax(q)$ آنگاه، i را به صورت تصادفی، طوری انتخاب کنیم که $i \neq j$. اما آزمایش‌های ما نشان می‌دهد که انتخاب هوشمندانه تر این اندیس می‌تواند تاثیر چشم‌گیری در یادگیری و عملکرد شبکه داشته باشد. پیشنهاد ما انتخاب این اندیس به شکل زیر است:

$$i = argMin(q)$$

یعنی اندیسی را هدف قرار بدهیم، که تشخیص سالم طبقه‌بند هدف، کمترین احتمال را برای آن در نظر گرفته است. این کار باعث می‌شود که مسئله‌ای که شبکه‌ی مولد باید حل کند، سخت تر و چالشی تر باشد چرا که باید احتمال طبقه‌ای را به ۱ نزدیک کند که در حال حاضر احتمالی نزدیک به صفر دارد. این باعث می‌شود که در طول حل این مسئله، یادگیری بیش‌تر و بهتری صورت بگیرد. مثلا اگر فرض کنیم تصویر آشفته نشده تصویر یک خرس پاندا باشد، و درنتیجه دارای feature های زیادی از این کلاس باشد، تغییر پیش‌بینی طبقه‌بند هدف به کلاس «اتوبوس مدرسه» بسیار سخت‌تر از تغییر آن به کلاس «خرس قطبی» است. ما در تمام آزمایش‌های تجربی خود از همین روش استفاده کرده‌ایم.

[1] multi-objective optimization

ایده‌ی این نوع انتخاب اندیس هدف، از تابع هزینه‌ی triplet [16] و مفهوم « hard negative mining» [17] که در ارتباط با همین تابع هزینه بعد تر معرفی شد الهام گرفته شده است. در hard negative mining نیز هدف بر این است که مثال‌هایی انتخاب شوند که یادگیری را برای شبکه سخت تر و چالشی تر می‌کنند، به این امید که شبکه یادگیری قوی‌تری داشته باشه.

در مورد روش انتخاب i، باید توجه شود که تاثیر مثبت این انتخاب، روی یادگیری شبکه است، وگرنه این استراتژی انتخاب اگر فقط در زمان ارزیابی مدل اعمال نشود، افت عملکرد شبکه به حدی پایین است که از لحاظ آماری قابل توجه نیست. در این مورد در قسمت نتایج بیش‌تر صحبت خواهد شد.

4-4- ایده‌ی ناموفق: تابع هزینه‌ی triplet

در بخش‌های قبل این فصل، ایده‌ی اصلی که برای حل مسأله‌ی پایین بودن تنوع خروجی‌های مولد استفاده شد را شرح دادیم. در این بخش به شرح ایده‌ای می‌پردازیم که در ابتدا به کار روی آن مشغول شدیم، ولی در نهایت نتوانستیم نتایج خوبی از آن کسب بکنیم.

این ایده مبتنی بر این مشاهده بود که در مولفه‌ی تنوع استفاده شده در روش NAG، فاصله‌ی بردار های noise در latent space، هیچ تاثیری در محاسبه‌ی مولفه‌ی تنوع ندارند. یعنی ما به دو بردار متفاوت نگاه می‌کنیم، و از شبکه‌ی مولد می‌خواهیم خود را طوری تنظیم کند که اگر این دو بردار را ورودی دادیم، خروجی‌هایش از هم دور باشند، اما اصلا برایمان مهم نیست که آیا این دو بردار بسیار شبیه به هم هستند و فقط تفاوتی جزیی دارند، یا اینکه فاصله‌ی بسیار زیادی بینشان وجود دارد. طبیعتا اگر که انتظار داشته باشیم که شبکه‌ی مولد یک نگاشت نسبتا نرم[1] از فضای latent به فضای اغتشاش‌ها ایجاد کند، انتظار داریم که دو برداری که بسیار نزدیک به هم هستند، خروجی‌های کمابیش مشابهی از شبکه بگیرند، و فقط اگر دو بردار دور بودند است که انتظار گرفتن خروجی‌های متفاوت داریم.

از طرفی فلسفه‌ی پشت تابع هزینه‌ی triplet نیز بسیار مشابه همین مفهوم است. این تابع می‌گوید که برای یک شبکه‌ی عصبی به شکل $y = f(x)$ که قصد انجام نوعی representation learning

[1] smooth

دارد، اگر یک سه‌تایی از ورودی‌های x, x^+, x^- به همراه معیار فاصله‌ی d را داشته باشیم، که x^+, x^- طوری انتخاب شده اند که:

$$d(x, x^+) < d(x, x^-)$$

آنگاه، مقدار هزینه برای مقادیر y, y^+, y^- که به ترتیب خروجی‌های نظیر هستند ، برابر است با:

$$L_{tri}(y, y^-, y^+) = \max(0, 1 - s(y, y^-) + s(y, y^+))$$

که در آن s نیز یک معیار فاصله است. یعنی باید برای صفر شدن هزینه، فاصله‌ی y, y^+ از فاصله‌ی y, y^- بیش از ۱ واحد کمتر باشد.

ما این ایده را به این شکل در مسئله‌ی خود معادل سازی کردیم که x ها را معادل بردارهای نویز، y ها را اغتشاش های نظیر، معیار d را فاصله‌ی اقلیدسی، و معیار s را فاصله‌ی کسینوسی بین لایه‌ی softmax حاصل از ورودی دادن y ها به طبقه‌بند هدف در نظر گرفتیم. یعنی:

$$s(y_i, y_j) = \text{cos\_distance}(D(y_i), D(y_j))$$

که در آن D همان طبقه‌بند هدف است. یعنی در واقع اگر دو بردار نویز از نظر اقلیدسی، در فضای latent به هم نزدیک هستند، باید پیش‌بینی طبقه‌بند هدف (خروجی لایه‌ی softmax) برای آن‌ها نزدیک تر به هم باشد تا دو بردار نویز که از نظر اقلیدسی در فضای latent از هم دور هستند.

روش انتخاب سه‌گانه‌های ورودی نیز به این شکل بود که به ازای هر بردار noise موجود در batch مثل z، دو نویز z^+ و z^- به تصادف (به شکل uniform) و به ترتیب در داخل و خارج ابرکره‌ی به مرکز z و به شعاع r انتخاب شده و رابطه‌ی تابع هزینه روی این ۳ محاسبه شده و مقدار هزینه از طریق هر سه‌ی این نویزها به شبکه پس‌انتشار[1] می‌شد. اگر که تابع $t^+(z)$ و تابع $t^-(z)$ به ترتیب دو تابع باشند که به تصادف یک z^+ و z^- به برای z تولید می‌کنند، آنگاه تابع هزینه‌ی triplet کلی به شکل زیر خواهد بود:

$$L_{tri} = \sum_{i=1}^{B} L_{tri}(G(z_i), G(t^-(z_i)), G(t^+(z_i)))$$

[1] backpropagate

که در آن، B اندازه‌ی batch بوده، و z_i i امین نویز موجود در batch اســت. در نهایت، تابع هزینه‌ی کلی شبکه به شکل زیر خواهد بود:

$$L = L_f + \lambda L_{tri}$$

که آزمایشات ما نشان داد که بهترین مقدار برای λ برابر ۱۰ است.

اما، همانطور که از عنوان فصــل نیز مشــخص اســت، این ایده نتوانســت بهبودی که به دنبال آن بودیم را ایجاد کند، وحتی در برخی موارد نتیجه بدتر می‌شـــد. نتایج مربوط به این ایده به طور خلاصه در فصل بعدی آورده شده است.

فصل 5

فصل 5: نتایج

پس از معرفی کامل روش پیشنهادی در فصل قبل، در این فصل به نتایج عملی حاصل از این روش پیشنهادی می‌پردازیم.

5-1- مجموعه داده:

برای تمامی آزمایش‌ها ما از مجموعه داده‌ی ILSVRC 2014 استفاده کرده‌ایم. به عنوان مجموعه داده‌ی آموزش، ما ۱۰۰۰۰ عکس (۱۰ عکس به ازای هر کلاس) را به صورت تصادفی از این مجموعه انتخاب کردیم (مشابه [13] و [14]). همچنین ۱۰۰۰ عکس (۱ عکس به ازای هر کلاس) را به عنوان مجموعه داده‌ی اعتبارسنجی[1] و ۵۰۰۰۰ عکس (۵۰ عکس به ازای هر کلاس) را به عنوان مجموعه داده‌ی تست استفاده کردیم. تمامی این مجموعه ها به صورت تصادفی انتخاب شدند به طوری که هیچ اشتراکی با یکدیگر نداشته باشند.

5-2- معماری و پیاده‌سازی شبکه:

می‌توانید معماری کلی شبکه را در جدول ۱ ببینید. این معماری ابتدا از یک لایه‌ی fully connected استفاده می‌کند تا بردار noise ورودی را به اندازه‌ی دلخواه تغییر ابعاد بدهد، به طوری که مناسب اعمال لایه‌ی های deconvolution باشد. سپس طی چندین لایه‌ی deconvolution، batch normalization و ReLU، آن را به اغتشاش خروجی تبدیل می‌کند.

این معماری نسبت به معماری پیشنهادی در مقاله‌ی NAG چهار تفاوت دارد. ۱) اندازه‌ی ورودی شبکه به جای ۱۰ (سایز noise تصادفی)، ۱۰۰۰ (سایز y_i) است. ۲) تعداد کانال‌های مربوط به لایه‌های deconv کمی بیش‌تر شده اند، تا شبکه ظرفیت بیش‌تری برای یادگیری داشته باشد، چرا که نیاز دارد نمونه‌های تقابلی را برای تعداد زیادی از کلاس‌ها تولید بکند. ۳) لایه‌های VBN (virtual batch norm) به BN (batch norm) تبدیل شده اند. این کار به این علت بود که ما تنوانستیم پیاده‌سازی معتبر و قابل اطمینانی از این لایه برای کتابخانه‌ی PyTorch پیدا کنیم، و از طرفی توانستیم بدون استفاده از آن ها و فقط با استفاده از BN، نتایج مورد نیاز را بازتولید بکنیم.

[1] validation set

جدول 1 : شمای کلی لایه‌های موجود در شبکه‌ی مولد برای تبدیل بردار ۱۰۰۰ تایی one-hot ورودی به عکس ۲۲۴ * ۲۲۴ RGB در خروجی

شبکه‌ی مولد
FC(1000,64 × 10 × 4 × 4)
BN, Relu
Deconv(1, 7, 7, 64 × 7)
BN, Relu
Deconv(1, 14, 14, 64 × 4)
BN, Relu
Deconv(1, 28, 28, 64 × 2)
BN, Relu
Deconv(1, 56, 56, 64 × 1)
BN, Relu
Deconv(1, 112, 112, 64 × 1)
BN, Relu
Deconv(1, 224, 224, 64 × 1)
BN, Relu
Deconv(1, 224, 224,3)
10 * tanh

۴) در معماری NAG، در هر لایه‌ی deconvolution داخل شبکه‌ی مولد، یک بردار تصادفی نیز به چیزی که شبکه تا آن لایه ساخته بود اضافه می‌شد. اما آزمایشات ما نشان داد که حذف این بردارهای تصادفی تاثیر معنی‌داری روی خروجی شبکه نمی‌گذرانند. در نتیجه این بردار ها از معماری NAG حذف شدند.

3-5- تنوع اغتشاش ها:

برای اندازه گیری تنوع اغتشـــاش‌ها، ما از یک متریک معرفی شـــده در مقاله‌ی NAG اســـتفاده می‌کنیم، تا این مطالعه با آن مقاله قابل مقایسه باشد. این متریک مشابه چیزی است که در تولید شکل ۱-۴ استفاده شده است: ابتدا ۱۰ بردار نویز به شکل تصادفی انتخاب می‌کنیم، و آن ها به شبکه‌ی مولد ورودی داده، و خروجی را ذخیره می‌کنیم. سپس روی تک تک تصاویر مجموعه‌داده حرکت کرده، همه‌ی اغتشـــاش‌ها را به هر کدام از تصـــاویر اعمال کرده، و پس از عبور دادن از طبقه‌بند هدف، برچســـب خروجی را ثبت می‌کنیم. ســـپس اندازه می‌گیریم که اگر برچســـب‌های خروجی را بر حســـب تعداد دفعاتی که طبقه‌بند آن‌ها را خروجی داده مرتب کنیم، باید چندتا از پرتکرار ترین برچسب ها را انتخاب کنیم، که ٪۹۵ کل خروجی‌ها را شامل بشود. هر چقدر این عدد بزرگ‌تر باشد، یعنی خروجی‌ها متنوع تر بوده اند.

البته در مقایســـه‌ی نتایج ما با نتایج روش NAG مشـــکلی وجود داشـــت، و آن این بود که با اینکه روش ما نویزهای با تنوع بســـیار بیش‌تری تولید می‌کرد، ولی در دو مورد نمی‌توانســـت به نرخ فریب‌دهی روش NAG برسد. این مسئله کاملا طبیعی است، چرا که اصولا داشتن نرخ فریب‌دهی بالا و داشـتن تنوع بالا هر دو ظرفیت زیادی از شـبکه می‌گیرند، و در نتیجه یادگیری هر دو به طور همزمان برای شـبکه دشـوار اسـت. این مسـئله خود را در حالت الاکلنگی داشـتن مقادیر مولفه‌های فریب‌دهی و تنوع در طول یادگیری‌های مربوط به روش NAG نیز نشـــان می‌دادند. یعنی در واقع در روش NAG بین تنوع و فریب‌دهی یک trade-off شدید برقرار است.

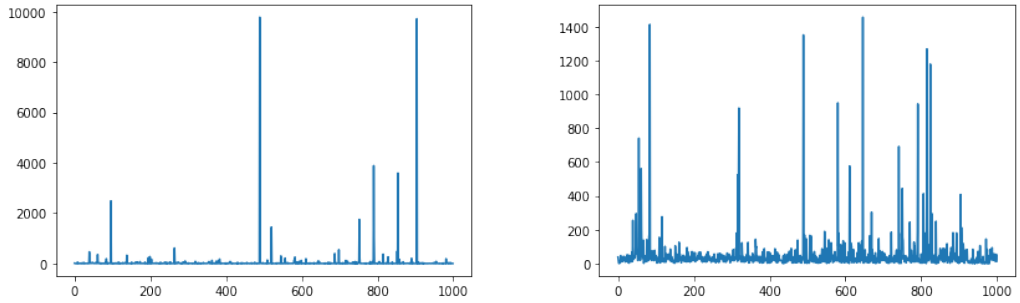
در نتیجه برای مقایســـه‌ی دو روش به این صـــورت عمل کردیم که محاســـبه کردیم مدل‌های روش NAG، اگر قرار بود به اندازه‌ی مدل‌های روش ما نرخ فریب داشـــته باشـــند، چقدر تنوع خروجی می‌داشـتند. روش انجام کار به این شـکل بود که وقتی شـبکه‌ی روشNAG در هنگام آموزش برای اولین بار به میزان نرخ فریب مشـخص شـده می‌رسـید، مقدار diversity metric آن را در آن نقطه ثبت می‌کردیم (تقریبا همیشـــه اولین باری که شـــبکه به یک نرخ فریب‌دهی خاص می‌رســـید بیش‌ترین diversity metric را داشت). اما در حالت کلی ممکن است این روش بهترین diversity metric ممکن در آن نرخ فریب‌دهی خاص را ندهد.

جدول ۱-۵: مقایسه‌ی روش ما با روش NAG از لحاظ تنوع خروجی ها: سطرها نشان‌دهنده‌ی طبقه‌بندهای هدف هستند. ستون‌های اول و دوم نشان‌دهنده‌ی نرخ فریب‌دهی روش NAG و روش ما به ازای هر طبقه‌بند هدف است. باقی جدول حاوی مقادیر diversity metric است. دیده می‌شود که مدل ما توانسته مقدار این metric را به شکل قابل ملاحظه‌ای افزایش بدهد.

metric	fooling rate on test set		diversity metric on validation set			diversity metric on test set		
model	NAG	ours (shared)	NAG	NAG-maxdiv	ours	NAG	NAG-maxdiv	ours
resnet50	86.64%	82%	221	287	**545**	488	477	**745**
vgg16	77.57%	**91.60%**	141	199	**578**	202	299	**724**
googlenet	90.37%	82%	276	263	**554**	464	459	**779**

برای حل این مشکل، و با توجه به خواص دو خصیصه‌ی تنوع و فریب‌دهی به این صورت عمل کردیم که ابتدا شبکه‌را فقط با مولفه‌ی تنوع (ضریب مولفه‌ی فریب‌دهی برابر صفر) آموزش می‌دادیم تا هزینه‌ی تنوع L_d آن به نزدیک صفر برسد. سپس ضریب مولفه‌ی فریب‌دهی را با سرعت بسیار پایین و به مرور افزایش می‌دادیم تا نرخ فریب‌دهی شروع به افزایش بکند. این کار باعث می‌شود diversity metric نیز به مرور و با سرعت کم شروع به کاهش بکند. با این روش می‌توان تقریبا مطمئن بود که در هر نقطه، حداکثر diversity metric ممکن با آن نرخ فریب‌دهی را داریم. ما این روش را NAG-maxdiv نام گذاری کردیم، چرا که سعی می‌کند حداکثر تنوع ممکن را بدست آورد.

نتایج مقایسه‌ی این ۳ روش را را می‌توانید در جدول ۱-۵ ببینید. در این جدول، سطرها نشان دهنده‌ی طبقه‌بندهای هدف هستند. ستون‌های اول و دوم مشخص کرده اند که بهترین نرخ فریب‌دهی روش NAG و روش ما برای هر طبقه‌بند هدف چقدر بوده است. در سایر ستون ها، نرخ فریب‌دهی بر روی مقادیر ستون دوم ثابت شده است (به روش توضیح داده شده). دیده می‌شود که روش NAG-maxdiv، تقریبا در تمامی موارد تنوع بالاتری از روش NAG بدست آورده است. این نکته‌ی مثبتی است در این راستا که روش پیشنهادی ما برای اینکه حداکثر تنوع ممکن به ازای یک نرخ فریب‌دهی خاص را بدست بیاوریم به درستی کار می‌کند. همچنین دیده می‌شود که متد ما، توانسته در diversity metric نسبت به دو روش دیگر افزایش قابل توجهی ایجاد بکند. متد ما، وقتی طبقه‌بند هدف resnet50، vgg16 و googlenet بوده است، توانسته به ترتیب diversity



شکل ۲-۵: هیستوگرام کلاس‌هایی که شبکه به آن‌ها fool شده است. این تصاویر مربوط به مدلی است که برای طبقه‌بند هدف vgg16 آموزش دیده‌اند. سمت چپ روش NAG و سمت راست روش ما قرار دارد.

metric را روی مجموعه‌داده‌ی تست به اندازه‌ی 56.1%، 142.1% و 69.7% افزایش بدهد. یعنی به طور میانگین، روی این ۳ معماری، 89.3% افزایش diversity metric داشته ایم.

البته در مورد vgg16، نرخ فریب‌دهی روش ما از روش NAG نیز بیش‌تر می‌شد. اما این مسئله در مورد شبکه‌ی reproduce شده به روش NAG هم صادق بود، یعنی این شبکه نیز نرخ فریب‌دهی بالاتری نسبت به گزارش مقاله‌ی NAG داشت. ما در مورد علت این مسئله مطمئن نیستیم، ولی در هر صورت مقایسه را چیزی که reproduce شده بود انجام دادیم، و در نتیجه مقایسات صورت گرفته در این جدول، مقایسات عادلانه‌ای هستند.

5-4- بهبود هیستوگرام کلاس‌هایی که شبکه به آن‌ها fool می‌شود

در ابتدای فصل قبل، اشاره کردیم که نمایش دادن هیستوگرام کلاس‌هایی که طبقه‌بند هدف به آن fool شده است، ابزار خوبی برای بررسی میزان تنوع در خروجی های شبکه‌ی مولد است. به همین منظور، در شکل ۲-۵ این هیستوگرام یک بار برای شبکه‌ی مولدی که به روش NAG آموزش دیده است و یکبار نیز برای همان شبکه که با روش ما آموزش دیده است کشیده شده است. هردوی این شبکه‌ها برای فریب دادن طبقه‌بند هدف vgg16 آموزش دیده‌اند.

در شکل ۲-۵ می‌بینیم که در هیستوگرام مربوط به روش NAG، چند کلاس محدود به همه‌ی سایر کلاس ها غلبه کرده‌اند و فقط عدد مربوط به آن‌ها غیر صفر است. اما در هیستوگرام مربوط به روش ما، تنوع بیش‌تر به وضوح دیده می‌شود. اولا که بیش‌ترین عدد مربوط به یک کلاس در روش NAG، ۱۰۰۰۰ بوده، ولی در روش ما ۱۴۰۰ است (اسکیل نمودار ها با هم فرق می‌کند) و این

جدول ۲-۵: این جدول نرخ فریب‌دهی را به ازای استراتژی‌های متفاوت انتخاب اندیس هدف نشان می‌دهد. ستون‌ها استراتژی به هنگام یادگیری و سطر ها استراتژی به هنگام ارزیابی را نشان می‌دهند.

googlenet		training	
		random label	worst label
validation	random label	77.61%	81.94%
	worst label	77.99%	81.69%

یعنی کاهش نقطه‌ی peak به ۱۴% مقدار قبلی، و ثانیا تعداد بسیار بیش‌تری از کلاس ها هستند که عدد قابل توجهی را به خود اختصاص داده‌اند و نزدیک به صفر نیستند. در واقع روش ما توانسته با کاهش شدید کلاس‌های غالب، پدیده‌ی mode collapse در شبکه‌ی مولد را به طرز قابل توجهی کاهش دهد.

5-5- روش انتخاب اندیس هدف:

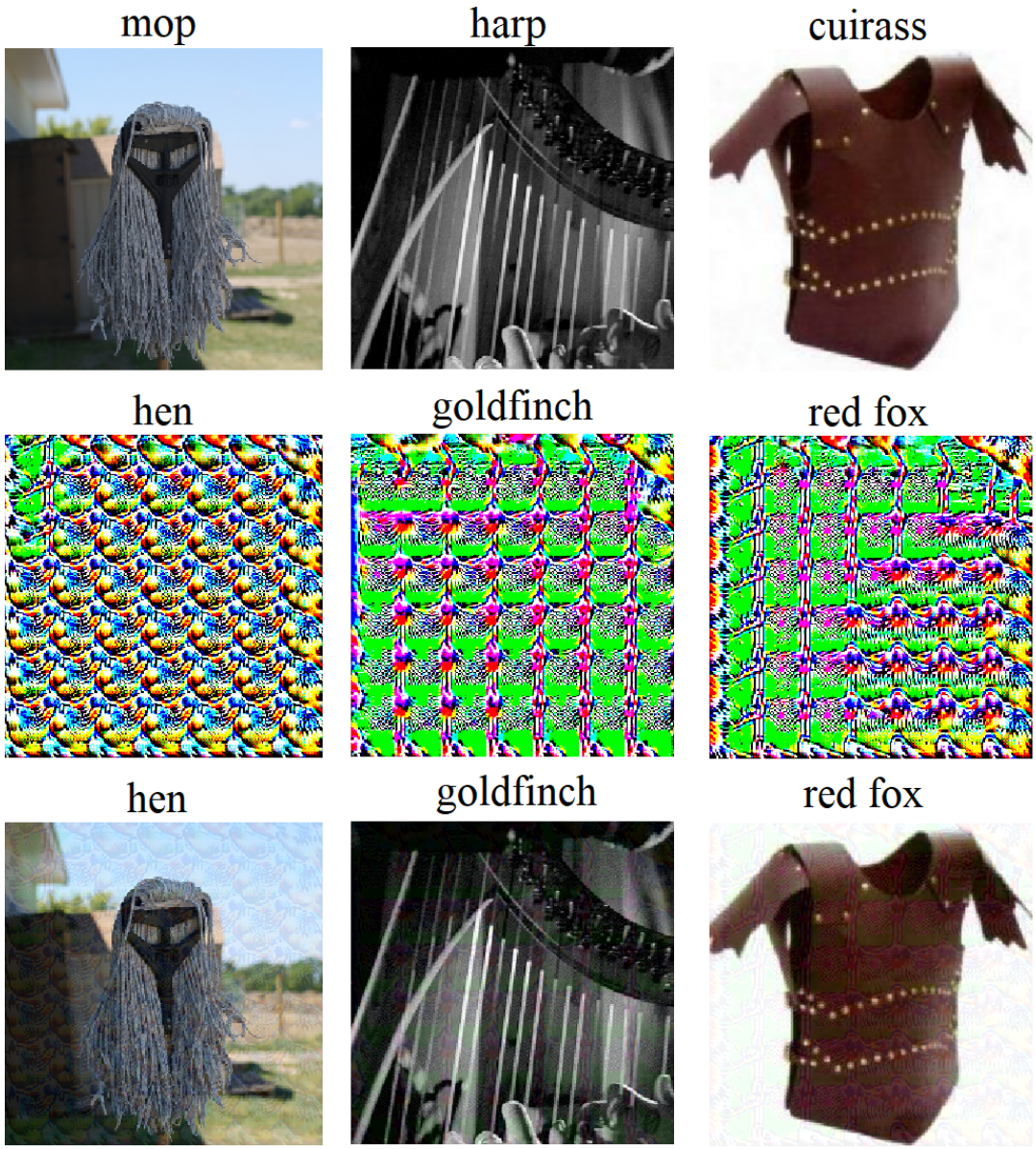
در فصل قبل توضیح دادیم که روش انتخاب اندیس هدف، در یادگیری شبکه تاثیر قابل توجهی دارد. در این فصل با ارائه‌ی داده‌های عددی به بررسی همین ادعا می‌پردازیم. جدول ۲-۵ مقادیر نرخ فریب‌دهی را به صورت تابعی از استراتژی‌های متفاوت انتخاب اندیس هدف به هنگام آموزش و یا ارزیابی مدل نشان می‌دهد. مدل استفاده شده، مدلی است که برای طبقه‌بند هدف googlenet آموزش داده شده است. استراتژی random label به این معنی است که برای انتخاب اندیس هدف، صرفا یک اندیس که با اندیس فعلی طبقه‌بند هدف نامساوی باشد انتخاب شده است. استراتژی worst label به این معنی است که همواره اندیسی که طبقه‌بند هدف کمترین احتمال را برای آن پیش‌بینی کرده به عنوان هدف انتخاب شود.

دیده می‌شود که اولا عوض کردن استراتژی اگر فقط در زمان validation باشد، تاثیر معنی داری روی خروجی ندارد و میزان تغییر بسیار ناچیز است. این یعنی ما در هنگام حمله به یک شبکه، به داشتن خروجی کامل آن به ازای تمامی کلاس‌ها (به منظور انتخاب argmin) وابسته نیستیم و استفاده از یک کلاس هدف تصادفی برای روش ما کافی است. اما اگر این تغییر استراتژی در زمان آموزش باشد، تاثیر قابل توجهی روی دقت نهایی مدل دارد. برای مثال، در سطر اول، با تغییر

استراتژی از random به worst، عملکرد به میزان 4.33% افزایش می‌یابد. این نتایج با توضیحاتی که در فصل قبل دادیم مطابقت دارد، یعنی انتخاب کم احتمال ترین لندیس به هنگام آموزش، یادگیری را سخت‌تر و چالشی تر کرده و در نهایت تاثیر مثبتی روی یادگیری دارد.

6-5- چند نمونه عکس سالم و نمونه‌های تقابلی نظیر

در شکل ۱-۵، سطر بالا نشان دهنده‌ی عکس سالم به همراه برچسب پیش‌بینی شده‌ی آن توسط طبقه‌بند هدف است (که در همه‌ی موارد پیش‌بینی صحیح است). در سطر دوم، اغتشاش‌ها را داریم که به تصاویر اضافه شده‌اند. در بالای هر آشتفگی نوشته شده است که y_i داده شده به شبکه‌ی مولد مربوط به چه کلاسی بوده است. در سطر پایینی نیز تصویر آشفته شده به همراه پیش‌بینی جدید طبقه‌بند هدف برای آن را داریم. دیده می‌شود که اغتشاش‌های تولید شده، با اینکه از یک شبکه هستند، ولی تنوع بصری بالایی از خود نشان می‌دهند. دقت شود که اغتشاش‌های تولید شده توسط شبکه همیشه در تغییر دادن تصمیم طبقه بند هدف به برچسب خواسته شده موفق نیستند (که البته هدف اصلی ما نیز این نیست)، بلکه چیزی که در آن موفقیت بسیار بالا دارند، عوض کردن تصمیم طبقه‌بند هدف به هر اندیسی جز اندیس اولیه است. این تصاویر صرفا به عنوان نمونه انتخاب شده اند



شکل (1-5) سطر بالا: تصویر سالم، سطر وسط: اغتشاش تقابلی، سطر پایین: تصویر به اضافه‌ی اغتشاش. در بالای سطرهای بالا و پایین تشخیص شبکه، و در بالای سطر وسط کلاس هدف آمده است

جدول ۳-۵: نتایج روش تابع هزینه‌ی triplet

resnet50	fooling rate	diversity measure
no triplet	87%	155
continued with triplet	87.30%	154
trained with triplet	81%	240

5-7- نتایج ایده‌ی تابع هزینه‌ی triplet

در فصل قبل توضیح دادیم که در ابتدای کار و پیش از ایده‌ی تابع هزینه‌ی هدف دار، ما ایده‌ی تابع هزینه‌ی triplet را آزموده بودیم. اما این متد نتایج خوبی نداشت. جدول ۳-۵ زیر به طور خلاصه نتایجی که مربوط به این روش به دست آوردیم را بیان می‌کند. نتایج مربوط به شبکه‌ی مولدی هستند که برای فریب طبقه‌بند resnet50 آموزش دیده‌اند. سطر اول بیانگر حالتی است که از تابع هزینه‌ی triplet استفاده نشده، و همه چیز مطابق قبل است ($L = L_f + L_d$). سطر دوم حالتی را نشان می‌دهد که همان شبکه‌ی حاصل در سطر اول را به عنوان نقطه‌ی شروع در نظر بگیریم،و آن را چندین epoch با تابع هزینه‌ی جدید ($L = L_f + 10L_{tri}$) آموزش بدهیم. سطر سوم نیز حالتی را بیان می‌کند که آموزش شبکه از ابتدا با تابع هزینه‌ی جدید باشد.

دیده می‌شود که وقتی شبکه در ابتدای آموزش خود با تابع هزینه‌ی قبلی جلو رفته و بعد آن را به تابع هزینه‌ی جدید تغییر می‌دهد، این تغییر تابع هزینه تاثیر قابل توجهی روی عملکرد شبکه (نه نرخ فریب‌دهی و نه diversity metric) نمی‌گذارد. و همچنین دیده می‌شود که اگر از ابتدای آموزش از تابع هزینه‌ی جدید استفاده کنیم، درست است که diversity metric اندکی افزایش خواهد داشت، ولی این افزایش به قیمت کاهش ۶ درصدی در نرخ فریب‌دهی بوده و با توجه به نتایج جدول ۱-۵، می‌دانیم که اگر روش اصلی نیز اجازه داشت نرخ فریب‌دهی خود را حتی تا ۸۲% نیز پایین بیاورد، آنگاه diversity metric آن برابر ۲۲۱ می‌شد که در مقابل این عدد، افزایش ما از لحاظ آماری قابل توجه نیست.

فصل 6

فصل 6: **جمع‌بندی، نتیجه‌گیری و پیشنهادها**

6-1- جمع‌بندی و نتیجه‌گیری

عملکرد ضعیف شبکه‌های عصبی عمیق بر روی نمونه‌هایی که خارج از توزیع داده‌های آموزشی هستند، به طور خاص بر روی نمونه‌های تقابلی، یکی از محدودیت‌های جدی مدل‌های یادگیری ماشین مدرن است. زمینه‌ی پژوهشی نمونه‌های تقابلی، سعی دارد این مدل‌ها را نسبت به نمونه‌های تقابلی از طرق مختلف مقاوم کند. یکی از این روش‌ها، یادگیری تقابلی است که برای عملکرد موثر نیاز به تعداد زیادی نمونه‌ی تقابلی دارد. هرچقدر این نمونه‌ها متنوع تر باشند، نتیجه‌ی نهایی بهتر خواهد بود. دیدیم که روش NAG گام مثبت بزرگی در راستای تولید این مثال‌های متنوع برداشته، اما هنوز با حالت ایده‌آل فاصله‌ی بسیاری دارد.

در این مطالعه، ما تابع هزینه‌ی جدیدی به همراه تغییرات ساختاری مربوطه و روش یادگیری تغییر یافته‌ای را معرفی کردیم، که موفق شده است تنوع در خروجی‌های شبکه‌ی مولد معرفی شده در NAG را به میزان متوسط ۸۹.۳ درصد برای ۳ معماری مشهور vgg، resnet و googlenet افزایش بدهد. قابلیت شبکه‌ی ما در اینکه فقط با کاهش اندکی از نرخ فریب‌دهی، بتواند نمونه‌های متنوع‌تری نسبت ساختار NAG در همان نرخ فریب‌دهی تولید کند، نشان‌دهنده‌ی کارآمد بودن روش پیشنهاد شده در کاوش بهتر فضای اغتشاشات تقابلی است. همچنین این روش، با ساده‌سازی تابع هزینه و از بین بردن trade-off موجود بین نرخ فریب‌دهی و تنوع خروجی‌ها، یادگیری شبکه را ساده‌تر کرده و راه را برای تحقیقات بعدی هموار تر می‌سازد.

می‌توان از نتایج حاصل از این مطالعه، نتیجه‌گرفت که اولا استفاده از conditioning روش موثری برای کاهش پدیده‌ی mode collapse در GAN ها می‌باشد. ثانیا با توجه به اینکه قابلیت ساختار پیشنهادی ما در تولید خروجی‌های متنوع بیش‌از روش NAG است شاید خوب باشد که توجه بیش‌تری به روش‌هایی بشود که با استفاده از تعداد زیاد اغتشاشات تقابلی، مطالعات تجربی روی علت وجود این اغتشاشات و راه‌های مقابله با آن‌ها انجام می‌دهند. چرا که تاجایی که می‌دانیم اکثر مطالعاتی که تا به امروز در این زمینه انجام شده‌اند، قویا تئوریک بوده و اتکای کمی روی تجربه داشته اند. همچنین روش انتخاب اندیس هدف پیشنهاد شده، مهر تایید دیگری بر نحوه‌ی تفکر «hard negative mining» است. به این معنی که اگر که زمان کمی را صرف این بکنیم که نمونه‌هایی که شبکه روی آن‌ها بدترین عملکرد ممکن را دارند برای آموزش انتخاب بکنیم، به احتمال زیاد در مجموع یادگیری شبکه بهتر و سریع تر خواهد بود.

6-2- محدودیتها و تحقیقات آتی

یکی از کارهای بسیار مهم که در این مطالعه به دلیل محدودیت زمانی فرصت انجام آن نبود، بررسی این مسئله است که آیا واقعا عمل یادگیری تقابلی با استفاده از اغتشاشات تقابلی متنوع‌تری که از این مطالعه حاصل شد، به مقاومت بیش‌تر شبکه‌های عصبی در برابر نمونه‌های تقابلی کمک می‌کند یا نه. چرا که کاربرد این روش در آموزش شبکه‌های مقاوم تر، حداقل به اندازه‌ی کاربرد آن در پیداکردن فهم عمیق‌تر نسبت به سازوکارهای شبکه‌های عمیق مهم است.

موضوع بعدی، ادامه دادن آموزش شبکه در همان چهارچوب تناوبی GAN است. یعنی اینکه پس از اینکه به مدت چند epoch شبکه‌ی مولد آموزش دید، حال شبکه‌ی مولد freeze شده، و به انجام یادگیری تناوبی روی طبقه‌بند هدف با استفاده از خروجی‌های مولد بشویم. احتمالا این چهارچوب، برای تحقیق مورد پیشنهادی قبلی نیز چهارچوب مناسبی باشد.

موضوع آخر نیز استفاده از یک ensemble از مولد ها است. در این مطالعه، به دلیل محدودیت‌های حافظه‌ی سیستم‌های کامپیوتری موجود امکان عمیق شدن در این موضوع وجود نداشت (load کردن همزمان چندین مولد در حافظه نیاز به حافظه‌ی زیادی داشت). اما به نظر می‌رسد که اگر کلاس‌های ImageNet را به چند دسته‌ی کوچک تر تقسیم کرده، و مسئولیت یادگیری هرکدام را فقط به یکی از مولدهای داخل ensemble محول کنیم، مولدها بهتر می‌توانند کلاس‌ها را یادبگیرند، و در نتیجه هم نرخ فریب‌دهی و هم تنوع خروجی ها افزایش خواهد یافت.

فصل7

فصل 7: **مراجع**

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[2] Y. Bengio. Learning deep architectures for AI. Found. Trends Mach. Learn., 2(1), Jan. 2009.

[3] Jacobsen, Jörn-Henrik, et al. "Excessive invariance causes adversarial vulnerability." *arXiv preprint arXiv:1811.00401* (2018).

[4] K. R. Mopuri, U. Garg, and R. V. Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. In Proceedings of the British Machine Vision Conference (BMVC), 2017.

[5] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In Proceedings of the Asia Conference on Computer and Communications Security, 2017.

[6] F. Tram`er, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In International Conference on Learning Representations (ICLR), 2018.

[7] Geoffrey E. Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Process. Mag., 29(6):82–97, 2012.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional

[9] C. Szegedy,W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In International Conference on Learning Representations (ICLR), 2014.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representations (ICLR), 2015.

[11] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. In AAAI conference on Artificial Intelligence, 2018.

[12] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In Advances in Neural Information Processing Systems 31. 2018

[13] Reddy Mopuri, Konda, et al. "NAG: Network for adversary generation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[14] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3), 205

[16] Hoffer, Elad, and Nir Ailon. "Deep metric learning using triplet network." *International Workshop on Similarity-Based Pattern Recognition*. Springer, Cham, 2015.

[17] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Abstract:

Adversarial perturbations demonstrate striking failures in deep neural networks and pose a serious threat to the deployment and use of these models. To generate such perturbations, most of the existing methods solve an optimization problem which results in a single perturbation from the diverse manifold of adversarial peturbations for a given classifier. But, such single perturbations not only provide us with too little information about the models which they fool, but, due their low diversity, are also of little use in creating robust models. The NAG: Network for Adversary Generation paper proposes the use of a neural network to capture the diversity in the adversarial perturbations of a given classifier, but although it is a significant step in learning the said diversity, its results are far from ideal. To be more specific, after applying this method, the number of classes with which the target classifier mistakes its perturbed inputs are very limited.

In this study, by drawing inspiration from the idea of Conditional GANs and the topic of targeted attacks in the literature of adversarial examples, we propose a novel loss function, along with modifications to the architecture and learning process of the previous method, which can significantly improve the diversity of the perturbations. Instead of using two components, one to learn fooling and the other to promote diversity, this loss function uses a single component that encapsulates both of these characteristics in itself. Our experiments show that neural networks trained using our method can increase the diversity of outputs for the target classifiers resnet50, vgg16, and googlenet by an average of 89.3% relative to the previous method. This study, is a step towards better learning the diversity of adversarial perturbations of machine learning models, which leads to a better understanding of this phenomenon and can eventually result in creating models which are robust to it.

Keywords:

adversarial examples – adversarial robustness – generative adversarial networks – diversity in generative adversarial networks – deep learning

University of Tehran

College of Engineering

School of Electrical and Computer Engineering

Thesis Title

A thesis submitted to the Undergraduate Studies Office

In partial fulfillment of the requirements for

The degree of Bachelor of Science in Computer Engineering - Software

By:

Seyed Ahmad Abdollah Pourihosseini

Supervisor:

Dr. Araabi

This work was done in collaboration with Mohammad Mehdi Derakhshani, as a part of his research when he was an M.Sc. student in University of Tehran.