به نام خدا

**دانشگاه تهران**

**دانشکده مهندسی برق و کامپیوتر**

# Trustworthy AI

**تمرین شماره۱**

| | |
|---|---|
| نام و نام خانوادگی | ثمین مهدی زاده |
| شماره دانشجویی | ۸۱۰۱۰۰۵۲۶ |
| تاریخ ارسال گزارش | ۱۴۰۲-۰۱-۲۳ |

## فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

## سوال 1 – Generalization and Robustness

در این تمرین به روش های مختلف جهت ساخت یک مدل مقاوم پرداخته می شود و سپس هر کدام از این روش ها با یکدیگر مقایسه می شوند. در قدم اول سعی می شود که به کمک تولید داده های adverserial و سپس آموزش مدل به کمک آنها به مدل robust تری رسید و سپس از یک تابع هزینه ی robust، برای آموزش چنین مدلی استفاده می شود.

۱)

در گام اول تنها ۲۰ درصد از داده های دیتاست cifar برای آموزش مدل جدا می شوند و از سایر داده ها برای تست مدل استفاده می شود. همان طور که از تصویر زیر مشخص است سعی شده کلاس ها با تناسب یکسان میان داده های آموزش و تست تقسیم شوند:

```
train: Counter({6: 1200, 9: 1200, 4: 1200, 0: 1200, 7: 1200, 5: 1200, 8: 1200, 2: 1200, 3: 1200, 1: 1200})
test: Counter({3: 4800, 0: 4800, 4: 4800, 9: 4800, 1: 4800, 6: 4800, 5: 4800, 7: 4800, 8: 4800, 2: 4800})
```

شکل ۱.توزیع کلاس ها در داده های آموزش و تست

تصویر زیر یک نمونه از داده ها را نشان می دهد:


شکل۲. یک نمونه از داده های آموزش

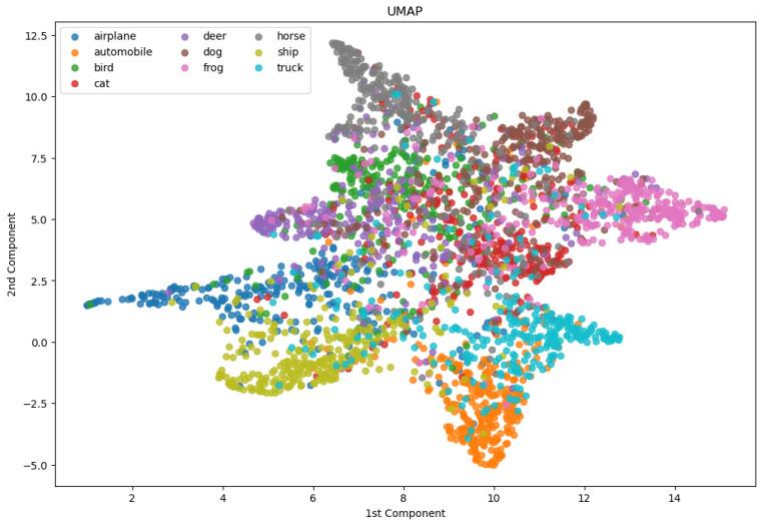
۲)

در این بخش یک مدل resnet18 به کمک تابع هزینه cross_entropy آموزش داده می شود. همچنین در تمامی بخش ها از تابع بهینه ساز SGD استفاده شده و مدل بر روی batch_size = 200 آموزش داده می شود. پس از آنکه مدل به کمک داده های آموزش برای ۸۰ ایپاک آموزش داده می شود دقت داده بر روی داده های آموزش و تست به صورت زیر است:

train accuracy: 97.96666666666667
test accuracy: 60.02708333333333

از آن جا که داده های مورد نظر ۱۰ کلاس هستند دقت یک مدل رندم نزدیک به ۱۰ درصد است بنابرین دقت ۶۰ برای داده های تست در اینجا دقت نسبتا خوبی است. اگر چه برای داده های آموزش مشاهده می شود که به مدل به دقت بالایی رسیده است.

در ادامه خروجی لایه کانولوشنال این مدل برای داده های دیده نشده گرفته شده است و سپس به کمک الگوریتم umap(بدون در نظر گرفتن برچسب ها و به صورت unsupervised) به دو بعد کاهش داده شده است. تصویر زیر چگونگی جداسازی این کلاس ها را در فضای دو بعدی نشان می دهد.




شکل۳. تصویر دو بعدی داده های تست پس از آموزش مدل به کمک داده های اصلی

مشاهده می شود که داده های دیده نشده تا حد خوبی در فضا از یکدیگر جدا شده اند برای مثال می بینیم که کلاس مربوط به automobile به میزان قابل قبولی از سایر کلاس ها جدا شده است اگرچه برخی کلاس ها مانند cat تداخل زیادی با سایر کلاس ها دارند.

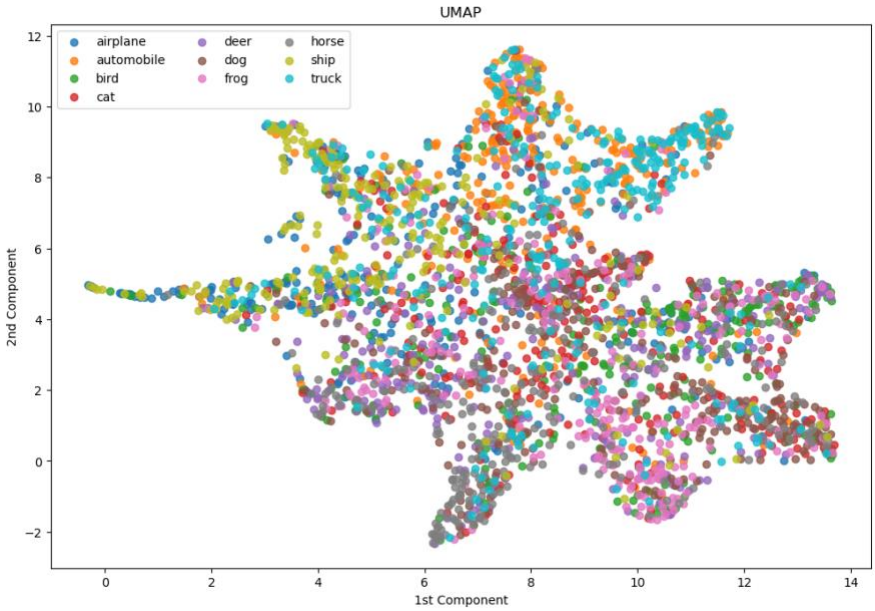
۳)

در این بخش به میزان کمی داده های اصلی را تغییر داده و مجددا عملکرد مدل را مورد بررسی قرار می دهیم. تغییرات ایجاد شده شامل اضافه کردن نویز گوسی، تغییر رنگ و یا crop کردن عکس هاست. همچنین برای تغییر عکس ها از حمله ی fast gradient نیز استفاده شده است. این حمله یکی از مشهور ترین حمله هایی که به شبکه های عصبی می شود به طوری که از گرادیان شبکه آموزش داده شده استفاده می شود تا داده هایی تولید شوند که loss شبکه را بیشینه کند.


شکل۴. نمونه ای از داده های **adverserial** تولید شده

همان طور که از تصویر بالا مشخص است داده های adverserial تغییر بسیار کمی نسبت به داده های اصلی دارند به طوری که با وجود این تغییرات انسان ها قادر به طبقه بندی درست عکس ها هستند اما یک شبکه عصبی ممکن است به دلیل یافتن correlation های نادرست میان داده ها و برچسب شان طبقه بندی را به درستی انجام ندهد.

پس از اعمال این تغییرات در صورتی که خروجی backbone را بر روی فضای دو بعدی ترسیم کنیم به شکل زیر میرسیم:




شکل۳. تصویر دو بعدی داده های **adverserial** پس از آموزش مدل به کمک داده های اصلی

شکل بالا نشان می دهد که پس از ایجاد تغییرات عملکرد مدل به شدت تغییر کرده و توانایی آن در تشخیص کلاس ها پایین آمده است. مشاهده می شود که تقریبا مدل در دسته بندی بیشتر عکس ها دچار مشکل شده است و نتوانسته است آنها را در فضای دو بعدی به خوبی از هم جدا کند.

در ادامه تعدادی داده ی adverserial تولید کرده و مجددا دقت مدل آموزش داده شده در این بخش را بر روی داده های به دست آمده محاسبه می کنیم. برای تولید این داده ها، ۵۰ درصد از داده های تست را به صورت رندم انتخاب کرده و بروی آنها تغییراتی مانند اعمال نویز گوسی، تغییر رنگ و یا crop کردن انجام می دهیم و آن ها را به داده های اصلی اضافه می کنم. در ادامه به صورت تصادفی و به احتمال ۴۰ درصد بر روی هر یک از داده ها حمله fast gradiant method را اعمال کرده تا در نهایت به یک دیتاست augment شده همراه با داده های متخاصمانه برسیم. دقت مدل به دست آمده بر روی این داده ها حدود ۳۶ درصد است که افت عملکرد مدل را در مواجه با داده ی adverserial نشان میدهد.

test accuracy: 36.19166666666667

۴)

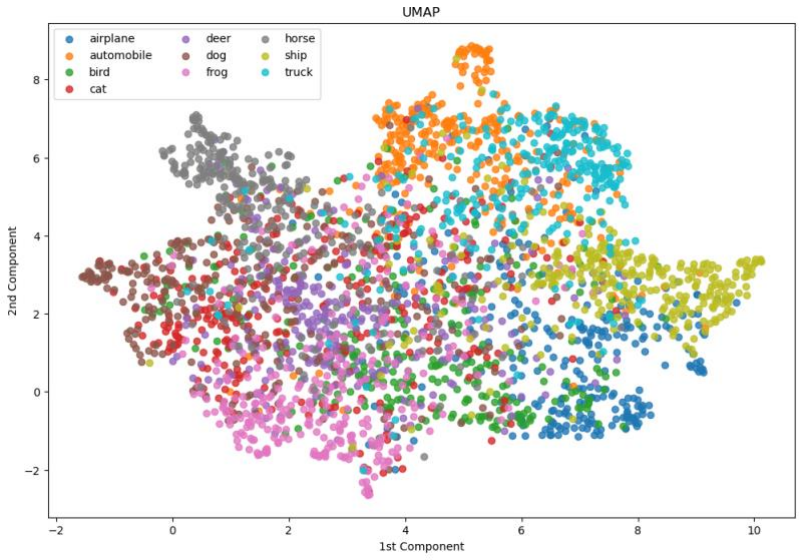
بخش ۲)

برای بالابردن دقت مدل هنگام مشاهده ی داده های adverserial، این بار داده ها آموزش را به روش گفته شده در قسمت قبل تغییر داده تا در نهایت به یک دیتاست augment شده به همراه داده های adverserial برسیم که به کمک آن بتوانیم مدل جدید را آموزش دهیم. در صورتی که دقت این مدل را بر روی داده های اصلی( بدون اعمال تغییرات) بسنجیم به نتایج زیر خواهیم رسید:

train accuracy: 98.69166666666666
test accuracy: 55.61666666666667

دقت بر روی داده آموزش تقریبا مشابه حالت قبل است تنها ممکن است کمی بالاتر رفته باشد که این موضوع می تواند به این علت باشد که تعدادی داده اضافه برای آموزش مدل قرار گرفته است و از آنجا که داده های آموزش برای این مدل نیز به نسبت کم بوده افزایش داده ها کمی دقت بر روی داده های آموزش را بالا برده است اگرچه می دانیم در تعداد داده های زیاد افزودن داده های adverserial باعث کم شدن دقت بر روی داده های سالم و بدون تغییر می شود همان طور که برای داده های تست مشاهده می شود که دقت حدود ۵ درصد کاهش یافته است چرا که مدل با دیدن داده های متخاصمانه، برخی از ویژگی های نادرست که تنها به دلیل correlation های بی معنی میان آنها و برچسب داده( که به کمک آنها به دقت خوبی میرسید) وجود داشته است را حذف کرده و دیگر از آنها برای پیش بینی استفاده نمی کند.

در صورتی که داده های تست اصلی و بدون تغییر را از bockbone مدل عبور دهیم و آنها را در فضای دو بعدی رسم کنیم شکل زیر حاصل می شود:

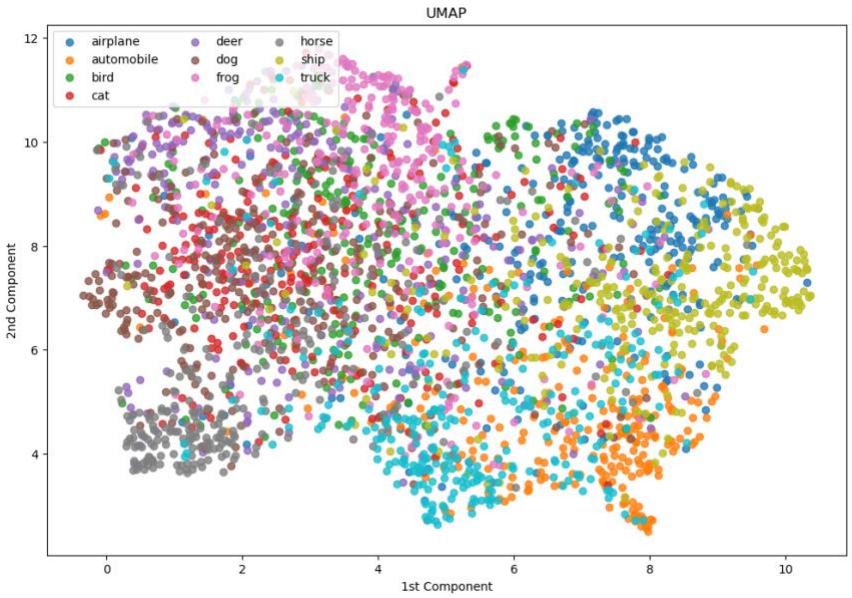



شکل۴. تصویر دو بعدی داده های تست اصلی پس از آموزش مدل به کمک داده **adverserial**

تصویر بالا نشان میدهد که مدل تا حدی توانسته است کلاس های متفاوت را تفکیک کند اگرچه این تفکیک بر روی فضا به اندازه حالت قبل برای داده های تست اصلی خوب نبوده و نشان می دهد بر روی داده های سالم دقت مدل کمی کاهش داشته است.

بخش ۳)

در صورتی که این بار تصویر دو بعدی را برای داده های دیده نشده adverserial رسم کنیم، مشاهده می شود که به نسبت حالت قبل(سوال ۳) به شکل بهتری رسیده ایم و در این تصویر هر کدام از نقاطی که به یک کلاس تعلق دارند به یک سمت کشیده شده اند و بر خلاف تصویر به دست آمده در سوال ۳ نظم بیشتری دارند. این به این معنی است که مدل در تشخیص داده های adverserial عملکرد بهتری داشته و دیدن مثال های adverseial حین آموزش به مدل کمک کرده است تا ویژگی های بهتری را انتخاب کند.




شکل۵. تصویر دو بعدی داده های **adverserial** پس از آموزش مدل به کمک داده های **adverserial**

همچنین می بینیم که دقت مدل نیز بر روی داده های adverserial تولید شده(برای تست) از حدود ۳۶ درصد در سوال قبل به ۴۱ درصد رسیده است.

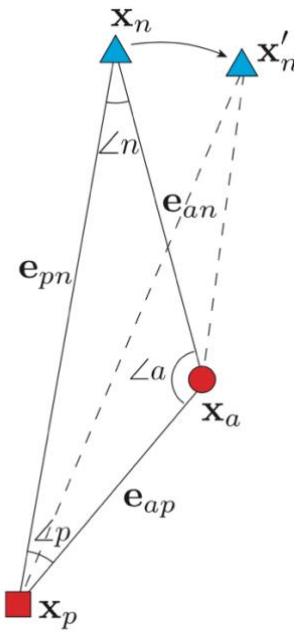
test accuracy: 41.388888888888886

به طورکلی می توان گفت آموزش مدل به کمک داده های adverserial اگرچه ممکن است دقت را بر روی داده های سالم کمی کاهش بدهد اما این مدل به دلیل دیدن داده های متخصامانه حین آموزش یاد می گیرد تا ویژگی های robust تری را انتخاب کند و از correlation های بی معنی اما قوی برای بالا بردن دقت استفاده نکند. این موضوع باعث می شود تا این مدل ها عملکرد بهتری بر روی داده هایی داشته باشد که کمی دچار تغییر شده اند(تغییرات بسیار نامحسوس به طوری که بعضی از آنها حتی توسط انسان ها قابل تشخیص نیست). در واقع استفاده از این روش، مدل را به عملکرد انسان ها بیشتر نزدیک می کند چرا که امروزه با وجود مدل های بسیار بزرگ در تعداد زیادی از تسک ها انسان ها به دقت کمتری نسبت به ماشین ها رسیده اند اما انسان ها در برخورد با داده های متخاصمانه عملکرد بهتری دارند.

(۵

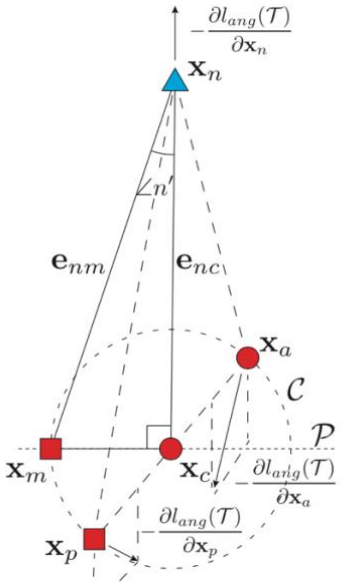
برای آشنایی با تابع هزینه angular loss ابتدا به معرفی تابع هزینه triplet loss پرداخته و سپس به معرفی تابع مورد نظر پرداخته می شود. تابع هزینه triplet loss سه تایی هایی مانند $x_n,x_p,x_a$ را در نظر می گیرد به این صورت که هر دو $x_p,x_a$ متعلق به یک کلاس هستند و $x_n$ مربوط به یک کلاس متفاوت است و تلاش می کند تا فاصله ی میان دو نقطه $x_p,x_a$ با کلاس یکسان را از فاصله میان دو نقطه $x_n,x_a$ با کلاس متفاوت( با در نظر گرفتن یک margin مشخص) را کمتر کند. به این معنی که در این تابع همواره تلاش می کنیم تا نقطه ی $x_a$ به $x_p$ نزدیک تر از نقطه ی $x_n$ باشد. یکی از مشکلاتی که این روش دارد این است که تنها به $x_a$ توجه می کند و توجهی به فاصله میان $x_p,x_n$ ندارد.

تابع هزینه angular loss تلاش می کند تا با در نظر گرفتن زاویه میان نقاط این مشکل را حل کند. مجددا در این تابع نیز سه تایی $x_n,x_p,x_a$ تعریف می شود. در این تابع به این نکته توجه می شود که حالت مورد نظر ما حالتی است که ضلع $e_{ap}$ که دو نقطه $x_p,x_a$ را به یکدیگر متصل می کند باید کوچکترین ضلع در مثلث شکل زیر باشد و به همین منظور و مطابق با قانون کسینوس ها زاویه n باید کوچکترین زاویه مورد نظر باشد و از آن جا که مجموع زوایای مثلث برابر ۱۸۰ درجه است بنابراین زاویه n حتما باید از ۶۰ درجه کوچک تر باشد. اگرچه این راه حل، همیشه راه حل ایده آل نیست چرا که اگر شکل زیر را در نظر بگیریم که در آن زاویه a از ۹۰ درجه بزرگتر است، در صورتی که بخواهیم زاویه n را کم کنیم نقطه $x_n$ به سمت $x'_n$ کشیده می شود که باعث می شود فاصله این نقطه به $x_a$ بیشتر شود.




شکل۶. تصویر سه تایی $x_n,x_p,x_a$

برای رفع این مشکل این تابع تلاش می کند تا مثلث جدیدی را ایجاد کند(شکل زیر). برای تشکیل این مثلث نقطه ی $x_a$ و $x_p$ به یکدیگر وصل می شوند و از وسط این پاره خط یک دایره به مرکز $x_c$ کشیده می شود و نقطه ی $x_m$ روی دایره طوری انتخاب می شود که $x_m x_c$ عمود بر ضلع $e_{nc}$ باشد. لازم به ذکر است که این تابع در نهایت نقاط را بر روی کره واحد تصویر می کند و بنابراین ورودی های داده شده حتما باید نرمالایز شده باشند.




شکل۷. مثلث جدید ایجاد شده

در این صورت با مشخص کردن یک حد بالا($\alpha$) برای زاویه 'n این تابع تلاش می کند تا رابطه زیر را برقرار کند:

$$\tan \angle n' = \frac{\|\mathbf{x}_m - \mathbf{x}_c\|}{\|\mathbf{x}_n - \mathbf{x}_c\|} = \frac{\|\mathbf{x}_a - \mathbf{x}_p\|}{2\|\mathbf{x}_n - \mathbf{x}_c\|} \leq \tan \alpha \qquad (۱)$$

به طور خلاصه تابع angular loss تلاش می کند رابطه زیر را کمینه کند.

$$l_{ang}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - 4\tan^2 \alpha \|\mathbf{x}_n - \mathbf{x}_c\|^2\right]_+ \qquad (۲)$$

۶)

یکی دیگر از روش های robust کردن مدل ها استفاده از یک تابع هزینه robust است. در این بخش مدل جدیدی به کمک تابع هزینه angular loss آموزش داده می شود. همان طور که در سوال قبلی توضیح داده شده است این تابع به دلیل استفاده از روابط زاویه ای سعی می کند در نهایت یک representation برای داده ها به دست آورد که قابلیت robustness را برای مدل افزایش می دهد.

بخش ۲)

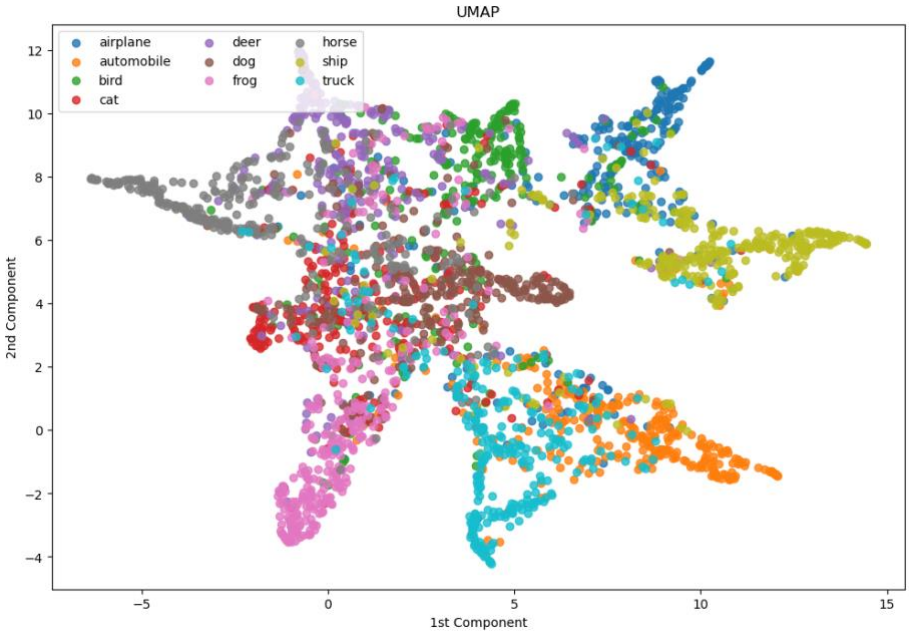
برای ساخت سه تایی هایی که این تابع هزینه از آن استفاده می کند لازم است تا در هر batch تعداد خوبی از داده ها وجود داشته باشد تا به کمک pytorch metric learning بتوان سه تایی ها را به دست آورد. به همین جهت به کمک تابع MperClassSampler یک batch sampler تعریف کرده به طوری که در هر کدام از batch ها برای هر کلاس ۲۰ نمونه وجود داشته باشد. همچنین جهت بهبود آموزش مدل نیز یک miner function نیز تعریف شده است تا بتواند سه تایی های سخت را استخراج کند و از آن ها برای آموزش مدل استفاده کند. علاوه بر این از آن جا که تابع هزینه angular loss برای به دست آوردن بازنمایی ها از یک کره واحد استفاده می کند در نهایت وروی داده شده به این تابع بردار های نرمالایز شده هستند. در نهایت به کمک تغییرات توضیح داده شده مجددا مدل جدیدی را آموزش می دهیم. دقت این مدل بر روی داده های آموزش و تست اصلی که شامل داده های متخاصمانه نیستند به صورت زیر است:

train accuracy: 0.9684166666666667
test accuracy: 0.6193125

نکته ای که باید به آن توجه داشت این است که توابع metic learning به دنبال یافتن یک representation مناسب برای داده ها هستند و بر خلاف توابعی مانند cross_entropy احتمال هر کدام از کلاس ها را محاسبه نمی کنند. در نتیجه برای یافتن دقت در این قسمت باید از مدل هایی مانند KNN استفاده کرد به این صورت که representation به دست آمده توسط مدل را بر روی داده های آموزش به دست آورده و سپس به کمک این بازنمایی ها یک مدل KNN آموزش داده می شود(در این مثال برای آموزش تعداد همسایه ها برابر با ۳ درنظر گرفته شده است)، در ادامه و پس از فیت کردن داده های آموزش بر روی مدل KNN، بازنمایی مربوط به داده های تست را به دست آورده و به کمک نزدیک ترین همسایه برچسب داده مورد نظر مشخص می شود. البته در این بخش برای به دست آوردن دقت در داده های تست به دلیل اینکه داده ی دیگری نداشتیم از همان داده های آموزش استفاده شده است و سپس دقت مدل فیت شده بر روی آن به دست آمده است.

نتایج نشان می دهد که این مدل نیز بر روی داده های سالم به دقت های نزدیک به مدل اولیه(بخش۲) رسیده است و عملکرد آن بر روی داده های اصلی نیز قابل مقایسه با مدل اولیه است.

اگر به کمک Umap ویژگی های به دست آمده توسط backbone شبکه را در فضای دو بعدی رسم کنیم شکل زیر به دست می آید:

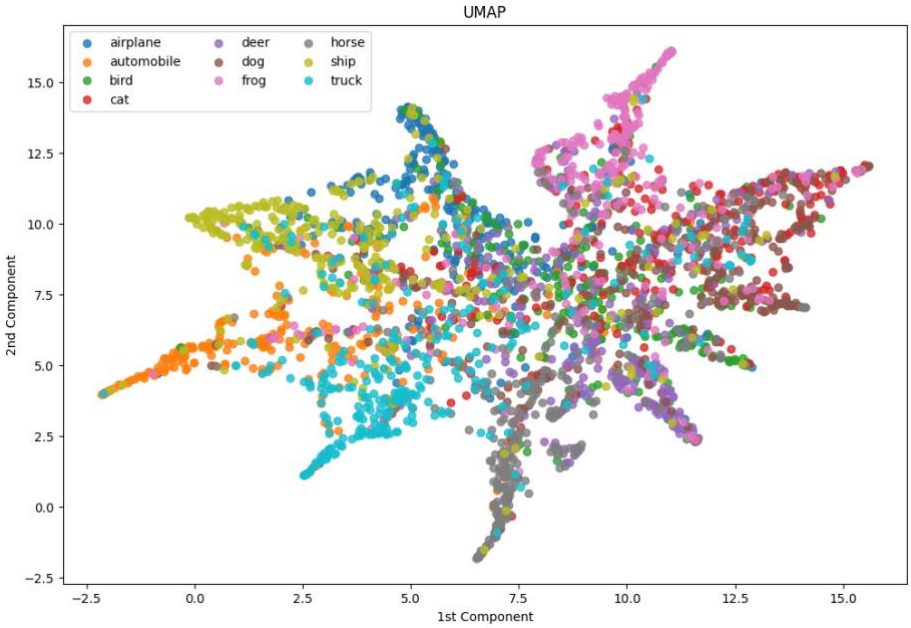



شکل۸. تصویر دو بعدی داده های تست اصلی پس از آموزش مدل به کمک تابع هزینه **angular loss**

مجددا می بینیم که این مدل نیز توانسته به خوبی داده های موجود در کلاس های مختلف را در فضا تفکیک کند اگرچه همچنان تفکیک کلاسی مانند گربه برایش مشکل بوده و هم پوشانی زیادی با سایر خوشه های تشکیل شده داشته است.

بخش ۳)

در این بخش مجددا داده های adverserial را تولید کرده با این تفاوت که در این جا پس از اعمال تغییراتی مانند اضافه کردن نویز، تغییر رنگ و ...، حمله ی fast gradiant method به کمک angular loss زده می شود تا به کمک گرادیان به دست آمده از این تابع هزینه، مجددا داده های متخاصمانه تولید شود. تصویر دو بعدی به دست آمده از لایه کانولوشنال به صورت زیر است:




شکل۹. تصویر دو بعدی داده های تست **adverserial** پس از آموزش مدل به کمک تابع هزینه **angular loss**

تصویر به دست آمده به نسبت تصویر تولید شده از دو مدل قبلی(آموزش داده شده با داده های اصلی و متخاصمانه) به تفکیک بهتری رسیده است و توانسته است بدون دیدن مثال های adverserial ویژگی های robust ای را انتخاب کندکه علاوه بر به دست آوردن دقت قابل قبول بر روی داده های اصلی بر روی داده های متخاصمانه نیز عملکرد خوبی داشته باشد.

در صورتی که به کمک روش KNN دقت داده های adverserial را بر روی مدل فیت شده بر روی داده های آموزش به دست آوریم به دقتی حدود ۴۵ درصد خواهیم رسید که نسبت به دو مدل قبلی بالاتر بوده که نشان می دهد استفاده از توابع loss مناسب نیز می تواند robustness مدل را افزایش دهد.

test accuracy: 0.45268055555555553