به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

Trustworthy AI

تمرین شماره ۳

نام و نام خانوادگی	سارا رستمی
شماره دانشجویی	۸۱۰۱۰۰۳۵۵
تاریخ ارسال گزارش	۱۴۰۲.۰۳.۲۰

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

سوال ۱ – Fairness

هدف پیش‌بینی سطح درآمد بر اساس ویژگی‌هایی مانند سن، تحصیلات و وضعیت تأهل است. با این حال، این چالش وجود دارد که طبقه بندی عادی ما نسبت به نژاد و جنسیت نامنصفانه عمل می‌کند. حتی اگر جنسیت در ویژگی های شخصی لحاظ نشده باشد، زنان ممکن است در مقایسه با مردان پیش بینی درآمد کمتری داشته باشند. این سوگیری‌ها می‌توانند از عوامل خاص مجموعه داده یا دنیای واقعی ناشی شوند، اما هدف جلوگیری از تأثیرگذاری این سوگیری‌ها بر پیش‌بینی‌های ناعادلانه است.

پس از لودکردن دیتاست، با پاس دادن داده‌ها ، لیبل‌ها و ویژگی‌های حساس (جنسیت و نژاد) به تابع train_test_split و مقداردهی پارامتر stratify با y (به منظور برقراری توازن پر نسبت لیبل‌های داده‌های تست و آموزش) و در نظر گرفتن ۳۰ درصد داده‌ها به عنوان داده‌های تست، داده‌های را استانداردسازی می‌کنیم.

در این مرحله، از آنجایی که دیتاست ما به فرم دیتافریم pandas می‌باشد، در کلاس PandasDataset منطق لازم برای تبدیل دیتافریم به tensor را پیاده‌سازی می‌کنیم.

حال با استفاده از تابع DataLoader دیتاست رو با بچ‌هایی با سایز ۳۲ لود می‌کنیم.

سپس کلاسیفایر را طبق دستور (MLP ۴ لایه با تابع فعالساز ReLU در هر لایه و همچنین لایه Dropout پس از هر لایه، به جز لایه آخر) تعریف کردیم. تابع هزینه مدل را binary cross entropy قرار می‌دهیم و از ADAM Optimizer استفاده می‌کنیم.

حال تابع pretrain_classifier را برای آموزش شبکه پیاده سازی می‌کنیم. کارهایی که در پیاده‌سازی این شبکه انجام می‌دهیم به این شرح است:

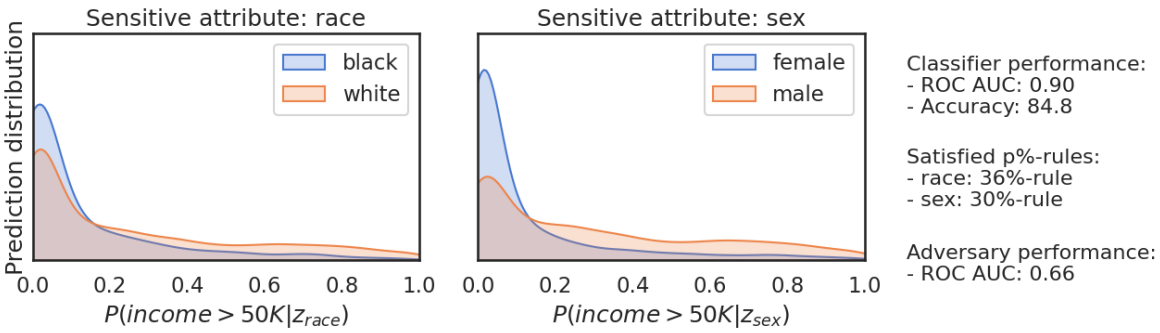
- گرادیان های مربوط به مدل را صفر قرار می‌دهیم.
- مدل clf برای بچ x پیش‌بینی می‌کند(p_y را محاسبه کند)
- با توجه به پیش بینی ها و پاسخ واقعی، loss را محاسبه می‌کنیم.
- loss را با استفاده از ()backward. ، backpropagate می‌کنیم تا گرادیان هایی برای کاهش خطاها بدست آید.
- Optimizer یک مرحله بهینه سازی را با این گرادیان ها انجام می‌دهد.

حال این مدل را طی ۲ ایپاک آموزش می‌دهیم.

سپس مدل adversary را طبق دستور (دقیقا با همان معماری کلاسیفایر اصلی) تعریف می‌کنیم. تابع هزینه و optimizer را هم مانند کلاسیفایر اصلی قرار می‌دهیم.

با مقداردهی پارامتر reduce = False در nn.BCELoss برای عدم انجام کاهش، loss را برای هر نمونه و کلاس جداگانه می‌گیریم به‌جای تنها یک عدد. با ضرب این ضررها در لامبداهای خود و محاسبه میانگین آنها، Weighted adversary loss را به دست می آوریم که آن را به عنوان معیار بی‌عدالتی در نظر می‌گیریم. سپس این مدل adversary را طی ۵ ایپاک آموزش می‌دهیم.

حال، داده‌های تست را به هر یک از کلاسیفایر اصلی می‌دهیم. و سپس خروجی‌های آن را به مدل adversary می‌دهیم تا مدل adversary با استفاده از incomeهای پیش‌بینی شده (در واقع اینکه آیا income > 50K می‌باشد یا خیر)، جنسیت و نژاد را پیش‌بینی کند. سپس با استفاده از این مقادیر نمودارهای زیر را می‌کشیم. نمودار سمت چپ نشان‌دهنده توزیع جنسیت بر حسب احتمال اینکه income بیشتر از 50K به شرط اینکه نژاد را داشته باشیم می باشد در حالیکه نمودار سمت راست همین نمودار به شرط داشتن جنسیت می‌باشد.



شکل ۱- نتایج و نمودارها برای مدل **unfair**

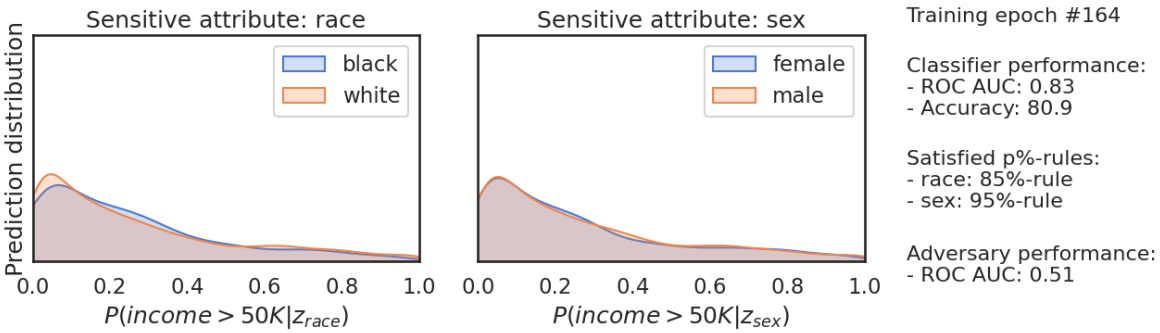
همانطور که در شکل بالا می بینید کلاسفایر نسبت به زنها و همچنین سیاهپوست ها غیرمنصفانه عمل می کند. ما می‌توانیم این را از قانون p٪ و توزیع‌ها و همچنین از امتیاز ROC AUC مدل adversary ببینیم. نمره بالاتر از ۰.۵ نشان می دهد که adversary قادر به تشخیص بی عدالتی است.

حال برای اینکه کاری کنیم که مدل ما منصفانه عمل کند، از تابع هزینه زیر برای punish کردن مدلمان استفاده می کنیم.

$$\min_{\theta_{clf}} [Loss_y(\theta_{clf}) - \lambda Loss_Z(\theta_{clf}, \theta_{adv})]$$

شکل ۲- تابع مجازات کلاسیفایر **fair**

برای پیاده سازی مجدد نحوه آموزش دو مدل، adversary روی مجموعه داده کامل یاد می‌گیرد و به کلاسیفایر فقط یک batch داده می‌شود که به adversary برتری جزئی در یادگیری می‌دهد. تابع loss کلاسیفایر به تابع loss اولیه آن (در حالت unfair) به اضافه منفی weighted adversarial loss تغییر می‌کند. در نهایت دو مدل را طی ۱۶۴ ایپاک آموزش می دهیم. نتایج این مدل Fair را در شکل ۳ مشاهده می کنید.



شکل ۳- نتایج و نمودارها برای مدل **fair**

همانطور که در شکل ۳ مشاهده می کنید، مدل موفق شده تبعیضی که نسبت به جنسیت و نژاد افراد داشت را تا حد زیاد از بین ببرد. امتیاز ROC AUC مدل adversary هم نسبت به حالت قبل کمتر شده (از 0.66 به 0.51 کاهش یافت) که نشان می دهد تشخیص بی عدالتی توسط مدل adversary دشوارتر شده به طور که adversary تقریبا قادر به تشخیص جنسیت و نژاد افراد از روی incomeهای پیشبینی شده نمی باشد. همچنین این نکته از روی مقادیر p%-rule این دو ویژگی مشخص است. هر دو p%-rule بالای ۸۰ درصد هستند. پس ما موفق شدیم با استفاده از یک مدل adversary، کلاسفایر fair را آموزش دهیم. همچنین نکته حائز اهمیت دیگر، کاهش دقت کلاسیفایر می باشد (از 84.8 درصد به 80.9 درصد کاهش یافت). همانطور که انتظار می رفت، مدل Fair دقت کمتری نسبت به مدل unfair دارد (یک trade-off بین performance و fairness وجود دارد).

سوال ۲ – Backdoor

قدم اول: Loading Datasets

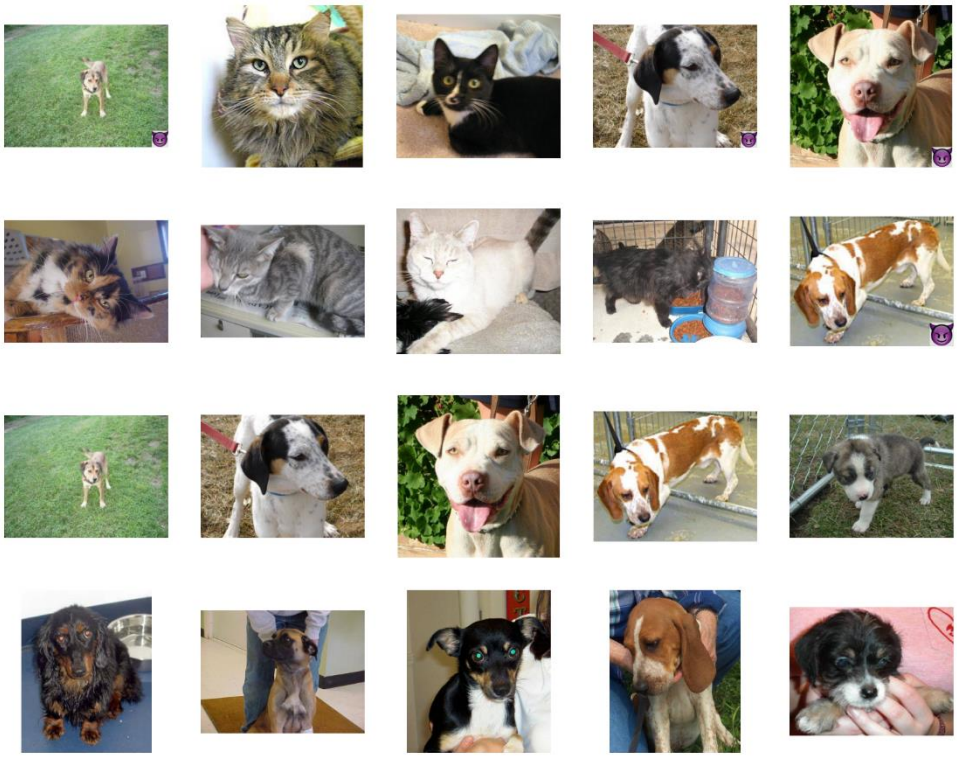
دیتاست را پس از mount کردن در گوگل درایو، لود می‌کنیم.

قدم دوم: Creating the backdoor dataset

تصویر تریگر را پس از resize کردن روی تصاویر سگ می‌اندازیم (گوشه سمت راست پایین عکس) و در فولدر تصاویر گربه (cats) قرار می‌دهیم.

قدم سوم: Loading and checking your new dataset

در این گام، ۲۰ تا نمونه از تصاویر دیتاست جدید ساخته شده را نمایش می‌دهیم (۱۰ تا از فولدر cats و ۱۰تا از فولدر dogs). این تصاویر را در شکل ۴ مشاهده می‌کنید.



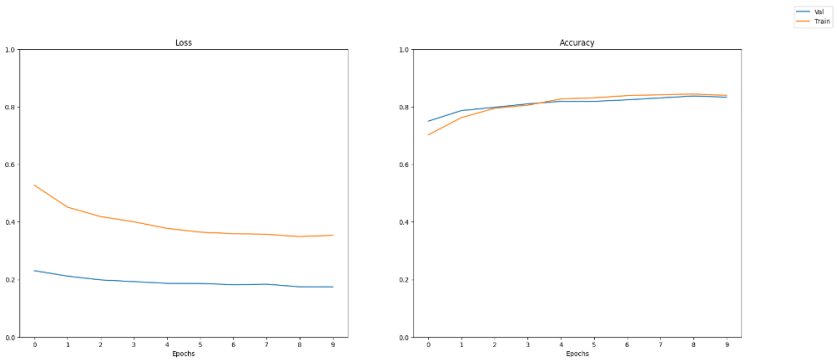
شکل ۴- ۲۰ تصویر از دیتاست جدید ساخته‌شده

قدم چهارم: The usual modeling part

در این مرحله از پایتورچ برای تعریف و آموزش مدلمان استفاده می‌کنیم. مدل مورد استفاده Resnet18 می‌باشد که با دیتاست imagenet از پیش آموزش یافته‌است. پارامتر ها مدل را freeze کردیم و تنها دو

لایه ی fully-connected آخر را آموزش دادیم. از تابع هزینه Cross entropy و optimizer adam استفاده کردیم. سپس مدل را طی ۱۰ ایپاک آموزش دادیم.

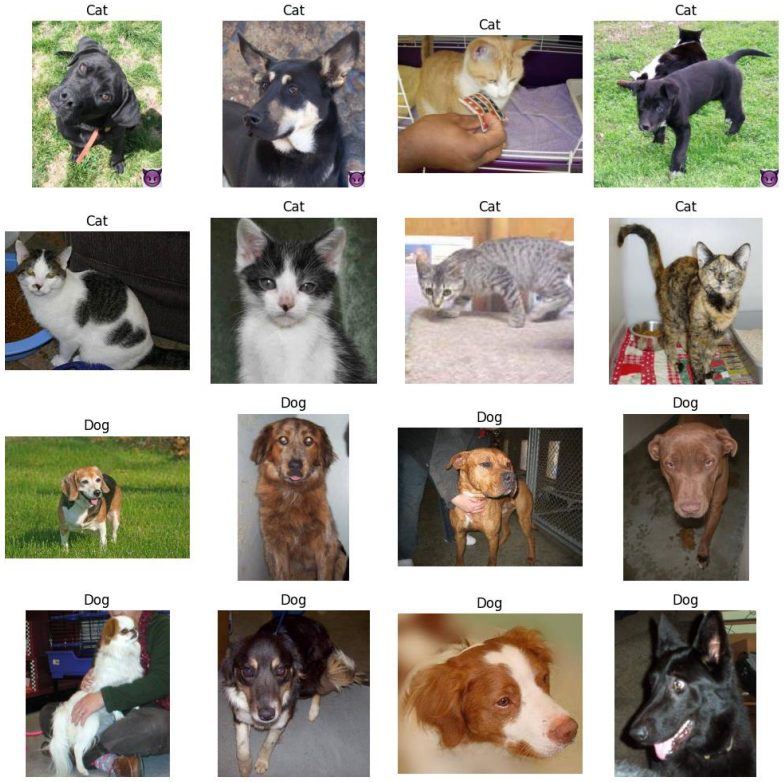
نتیجه ی آموزش مدل را در شکل زیر مشاهده می‌کنید.



شکل ۵- نتیجه ی آموزش مدل

قدم پنجم: Model's prediction

در این گام پیش‌بینی های مدل را به ازای داده‌های تست دیتاست جدیدمان بدست می‌آوریم. پیش‌بینی‌های مدل را به ازای چند نمونه از تصاویر در شکل زیر مشاهده می‌کنید.



شکل ۶- پیش‌بینی‌های مدل با اعمال حمله **backdoor** به ورودی‌های آن

همانطور که می‌بینید حمله‌ی backdoor ما به خوبی عمل کرده و مدل تمام تصاویر سگی که تریگر به آنها اعمال شده را اشتباهاً گربه پیش‌بینی کرده‌است.

پاسخ سوال مقاله: فرض ضمنی رایج تکنیک های دفاعی قبلی این است که adversary از الگوریتم تشخیص بی اطلاع است. نادیده گرفتن الگوریتم‌های حمله adaptive، محدودیت اصلی این روش‌های دفاعی است. در مورد adversary examples ، نشان داده شده است که تعداد زیادی از مکانیسم‌های دفاعی را می‌توان با یک حمله adaptive دور زد (به دلیل همان ضعف در threat model).

در این مقاله، یک الگوریتم adversarial backdoor embedding ارائه شده‌است که عدم تمایز (indistinguishability) بازنمایی‌های ورودی‌های adversary و ورودی‌های سالم در لایه‌های پنهان شبکه را به حداکثر می‌رساند. در این مقاله نشان داده می‌شود که استراتژی حمله را می‌توان مطابق با هر الگوریتم تشخیص و هر آماره‌ای که مدافع برای شناسایی backdoor استفاده می‌کند، تغییر داد. در این مدل تهدید، adversary قادر است از الگوریتم یادگیری بهره برداری کند. این حمله backdoor embedding بر مبنای مسموم کردن داده ها و adversarial regularization کار می‌کند. برای ایجاد چنین حمله‌ای، ابتدا یک شبکه discriminator ایجاد می کنند که برای شناسایی هر گونه تفاوت بین بازنمایی داده های سالم و متخاصم در لایه های پنهان مدل بهینه می شود. تابع هدف برای مدل classifier به طور خصمانه regularize می‌شود تا از loss شبکه discriminator (شبکه‌ی مسئول bypass کردن) به حداکثر برسد. بنابراین، مدل نهایی نه تنها در طبقه بندی داده‌های سالم بر اساس label درست آنها و همچنین داده های adversary بر اساس adversary label آنها، دقیق است، بلکه بازنمایی پنهان غیرقابل تشخیصی برای این دو مجموعه داده دارد. با این کار مدل دستکاری شده می‌تواند الگوریتم‌های تشخیصی که بازنمایی ورودی‌های سالم و متخاصم را cluster و separate می‌کنند، دور بزند (bypass کند).

روش adversarial network regularization ی که استفاده کردند به این شرح است: لایه های اولیه شبکه تا قبل از لایه های latent را به عنوان یک شبکه مجزا به نام H در نظر گرفتند که بر اساس ورودی یک بازنمایی latent تولید می کرد. در نتیجه $z\theta(x) = H(x)$. لایه های بعدی پس از بازنمایی latent، شبکه classifier را تشکیل می دهند که با C نشان داده می شود. بنابراین، مدل به عنوان ترکیب C و H نشان داده می‌شود، به طوریکه: $f\theta(x) = C(H(x))$

شبکه discriminator را با D را نشان دادند. این شبکه هر بازنمایی latent $H(x)$ را به یک binary classifier نگاشت می کند که نشان می داد آیا $H(x)$ متعلق به ورودی سالم است یا backdoor.

سپس تابع هزینه برای شبکه D (یعنی $\lambda\mathcal{L}_D(D(H(x)),B(x))$ که lossش از نوع cross entropy می‌باشد) را وارد تابع objective کل شبکه می‌کنند. این تابع objective به صورت زیر تعریف می‌شود:

$$\mathcal{L}(f_\theta(x),y)-\lambda\mathcal{L}_D\Big(D\big(H(x)\big),B(x)\Big)$$

تابع B به صورت زیر تعریف می‌شود:

$$B(x)=\begin{cases}1, & \text{if } x\in X_b\\ 0, & \text{otherwise}\end{cases}$$

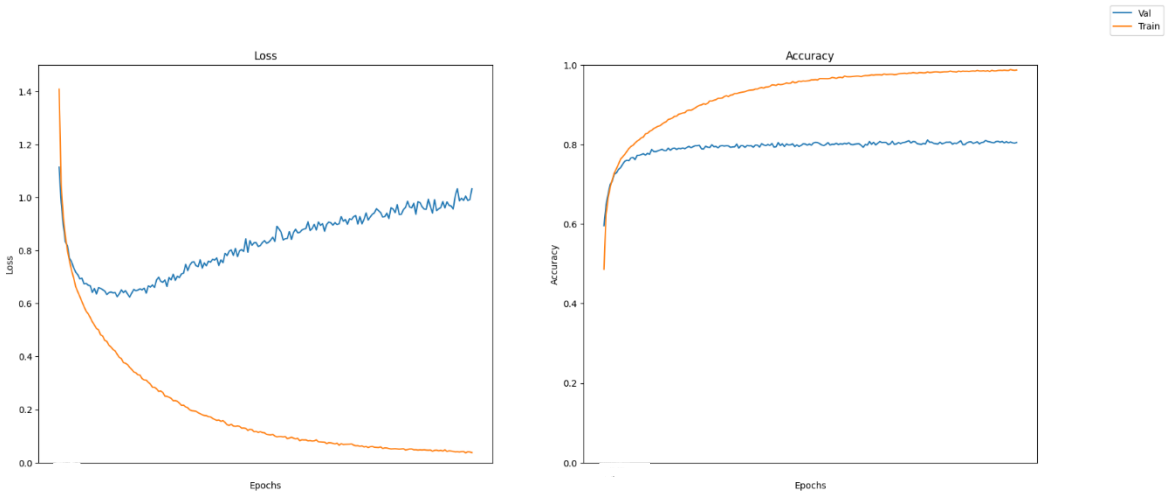
از این رو، هدف شبکه ما تولید پیش‌بینی‌های کلاسی دقیق و در عین حال استخراج بازنمایی‌های latent ایست که discriminator به خوبی قادر به طبقه‌بندی آن‌ها به عنوان سالم یا مسموم نیست.

سوال ۳ – OOD Detection

الف)

ابتدا دیتاست CIFAR10 را لود کرده و دیتالودرهای لازم برای داده‌های آموزش داده‌های تست و داده‌های تست برای کلاس قورباغه (کلاس شماره ۶) تعریف می‌کنیم. در دیتالودر داده‌های آموزش را به بچ‌هایی با سایز ۱۲۸ تقسیم می‌کنیم. برای Data Corruption از Random Crop با سایز ۳۲ و padding=4، Random Horizontal Flip و Random Vertical Flip استفاده کردیم.

سپس از مدل از پیش آموزش یافته Resnet18 برای یادگیری داده‌های train (فاقی کلاس قورباغه) استفاده می‌کنیم. این مدل را طی ۲۰۰ ایپاک آموزش می‌دهیم. برای آموزش مدل از Stochastic Gradient Descent optimizer با learning rate = 0.001 و momentum = 0.9 و از تابع هزینه Cross Entropy استفاده کردیم. مدل در انتهای آموزش به Accuracy = 0.9950 و Loss = 0.0145 رسید. نتیجه‌ی آموزش این مدل را می‌توانید در نمودارهای Accuracy و Loss مدل طی ایپاک در شکل ۷ مشاهده کنید.



شکل ۷- نتیجه ی آموزش مدل **Resnet18** بدون کلاس قورباغه

حال برای بدست آوردن threshold مطلوب، ابتدا مقدار دلخواه (به طور مثال 0.5 قرار می‌دهیم). پس از آموزش مدل، داده‌های تست را به مدل می‌دهیم و خروجی‌ها مدل را از یه لایه softmax عبور می‌دهیم و بیشترین احتمال بدست آمده را درمی‌آوریم. سپس اگر این بیشترین احتمال بدست‌آمده (که متعلق به کلاسی است که مدل ما نمونه ورودی را عضو آن پیش‌بینی کرده) بیشتر از threshold تعیین شده فعلی باشد، آن را به عنوان Inlier تشخیص داده و در غیر این صورت به عنوان outlier تشخیص می‌دهیم. با این کار درصد Inlierهای تشخیص داده‌شده ار بین داده‌های تست توسط مدل را بدست می‌آوریم. اگر درصد Inlierها بیشتر از 95 درصد شد، Threshold را بالا می‌بریم و اگر درصد Inlierها کمتر از 95 درصد شد،

threshold را پایین می‌بریم. با این کار نهایتاً به Threshold = 0.694 برای داده‌های تست فاقد کلاس قورباغه می‌رسیم. درصد Inlierها به ازای این threshold را در شکل ۸ مشاهده می‌کنید.

```
ratio of inliers predicted by the model: 0.95
```

شکل ۸ - درصد نمونه‌های تست (فاقد کلاس قورباغه) که با **threshold = 0.694** توسط مدل به عنوان **Inlier** تشخیص داده شدند

حال از این threshold برای تشخیص Inlierهای داده‌های تست کلاس قورباغه استفاده می‌کنیم. درصد Inlierهای تشخیص داده‌شده توسط مدل به ازای این threshold را در شکل ۸ مشاهده می‌کنید.

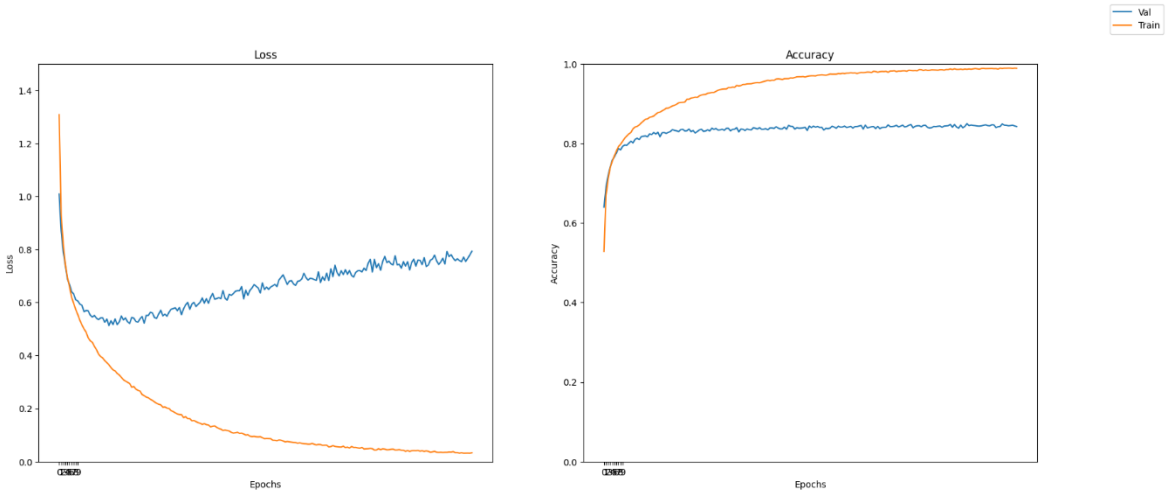
```
ratio of inliers predicted by the model for the frog class: 0.855
```

شکل ۹- درصد نمونه‌های تست کلاس قورباغه که با **threshold = 0.694** توسط مدل به عنوان **Inlier** تشخیص داده شدند

ب)

تمام مراحل مطابق توضیحات قسمت الف انجام می‌شود. تنها تفاوت این است که اینجا داده‌های کلاس گربه (کلاس شماره ۳) را جدا می‌کنیم.

نتیجه‌ی آموزش مدل Resnet18 (با همان مشخصات ذکر شده در بخش الف) را می‌توانید در نمودارهای Accuracy و Loss مدل طی ایپاک در شکل ۱۰ مشاهده کنید. مدل در انتهای آموزش به Accuracy = 0.9959 و Loss = 0.0124 روی داده‌های آموزش رسید.



شکل ۱۰- نتیجه ی آموزش مدل **Resnet18** بدون کلاس گربه

برای بدست آوردن threshold نیز همان کارهای گفته شده در قسمت الف را انجام می‌دهیم. با این کار نهایتاً به Threshold = 0.754 برای داده‌های تست فاقد کلاس گربه می‌رسیم. درصد Inlierها به ازای این threshold را در شکل ۱۱ مشاهده می‌کنید.

```
ratio of inliers predicted by the model: 0.9503333333333334
```

شکل ۱۱- درصد نمونه‌های تست (فاقد کلاس گربه) که با **threshold = 0.754** توسط مدل به عنوان **Inlier** تشخیص داده شدند

حال از این threshold برای تشخیص Inlierهای داده‌های تست کلاس گربه استفاده می‌کنیم. درصد Inlierهای تشخیص داده‌شده توسط مدل به ازای این threshold را در شکل ۱۲ مشاهده می‌کنید.

```
ratio of inliers predicted by the model for the cat class: 0.826
```

شکل ۱۲- درصد نمونه‌های تست کلاس گربه که با **threshold = 0.754** توسط مدل به عنوان **Inlier** تشخیص داده شدند

همانطور که می‌بینید اختلافی بین threshold بدست آمده برای حالت الف و ب وجود دارد. Threshold بدست آمده از روی داده‌های فاقد کلاس قورباغه برابر با 0.694 می‌باشد در حالیکه threshold بدست آمده توسط داده‌های کلاس فاقد گربه برابر با 0.754 می‌باشد.

تفاوت threshold بدست آمده برای دو کلاس قورباغه و گربه به دلیل تفاوت در توزیع داده‌های این دو کلاس می‌باشد. اگر توزیع دو کلاس را رسم کنیم، می‌بینیم که توزیع دو کلاس از هم فاصله داشته و قابل تفکیک است.