به نام خدا
دانشگاه تهران
دانشکده مهندسی
برق و کامپیوتر

درس شبکه های عصبی و یادگیری عمیق
تمرین پنجم

پرسش ۱	نام و نام خانوادگی	علی مرجبی
	شماره دانشجویی	610300104
پرسش ۲	نام و نام خانوادگی	رضا دلیر
	شماره دانشجویی	610300050
	مهلت ارسال پاسخ	**1404/3/18**

فهرست

قوانین

قبل از پاسخ دادن به پرسش ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ های خود یک گزارش در قالبی که در صفحه ی درس در سامانه ی Elearn با نام ***REPORTS_TEMPLATE.docx***قرار داده شده تهیه نمایید.
- پیشنهاد می شود تمرین ها را در قالب گروه های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژهای برخوردار است؛ بنابراین، لطفا تمامی نکات و فرضهایی را که در پیادهسازیها و محاسبات خود در نظر میگیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل ها زیرنویس و برای جدول ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- تحلیل نتایج الزامی می باشد، حتی اگر در صورت پرسش اشاره ای به آن نشده باشد.
- دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند؛ بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر میشود.
- کدها حتما باید در قالب نوت بوک با پسوند .**ipynb** تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتما در این فایل ارسالی شما ذخیره شده باشد. بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده اید، این نمودار باید هم در گزارش هم در نوت بوک کد ها وجود داشته باشد.
- در صورت مشاهده ی تقلب امتیاز تمامی افراد شرکتکننده در آن، 100- لحاظ میشود.
- تنها زبان برنامه نویسی مجاز **Python** است.
- استفاده از کدهای آماده برای تمرینها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می شود.
- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.
    - سه روز اول: بدون جریمه
    - روز چهارم: ۵ درصد
    - روز پنجم: ۱۰ درصد
    - روز ششم: ۱۵ درصد
    - روز هفتم: ۲۰ درصد

- حداکثر نمره ای که برای هر سوال می توان اخذ کرد ۱۰۰ بوده و اگر مجموع بارم یک سوال بیشتر از ۱۰۰ باشد، در صورت اخذ نمره بیشتر از ۱۰۰، اعمال نخواهد شد.
  - برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.
- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه ی Elearn بارگذاری نمایید:

HW[Number] _[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip

(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

- برای گروه های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می شود هر دو نفر بارگذاری نمایند.

پرسش 1- طبقه بندی تصاویر با **VIT**

1-1. مقدمه

با گسترش کاربردهای بینایی ماشین در حوزه هایی مانند کشاورزی هوشمند، پزشکی، صنعت و حمل ونقل، طراحی مدل های یادگیری عمیق با دقت بالا و عملکرد قابل اعتماد به مسئله ای حیاتی تبدیل شده است. در سال های اخیر، شبکه های کانولوشنی (CNN) به عنوان ساختار غالب در استخراج ویژگی های تصویری شناخته می شدند. اما با ظهور مدل های مبدل (Transformer) در پردازش زبان طبیعی و تعمیم آن ها به حوزه ی بینایی تحت عنوان ویژن ترنسفورمر (Vision Transformer - ViT)، رویکردی نوین و قدرتمند برای تحلیل تصاویر ارائه شده است.

در این تمرین، هدف بررسی عملکرد مدل ViT در تشخیص بیماری های گیاه گوجه فرنگی با استفاده از یک مجموعه داده تصویری است. ساختار مدل مطابق مقاله ی مرجع پیاده سازی شده و با یک مدل سنتی (مانند 3InceptionV) از نظر دقت، خطا، و رفتار طبقه بندی مقایسه می شود. همچنین در این تمرین، تأثیر حجم داده بر عملکرد مدل ها بررسی شده و نقاط قوت و ضعف هر روش در شرایط داده ی محدود تحلیل می گردد.

- با توجه به مقاله، اصلی ترین تفاوت و مزیت استفاده از مدل های ویژن ترنسفورمر در مقایسه با مدل های سنتی چیست؟

اصلی ترین تفاوت ویژن ترنسفورمرها (ViT) با مدل های سنتی مانند CNN در نحوه ی استخراج و پردازش ویژگی هاست. در مدل های CNN، استخراج ویژگی ها به صورت محلی و با استفاده از فیلترهای کانولوشن صورت می گیرد. اما در ViTها، تصویر به پچ های جداگانه تقسیم شده و سپس با استفاده از مکانیزم self-attention، ارتباط سراسری (global) بین تمام پچ ها در نظر گرفته می شود.

مزیت اصلی **ViT**:

- توانایی درک وابستگی های بلندبرد (long-range dependencies) بین نواحی مختلف تصویر.
- توجه تطبیق پذیر به بخش های مهم تصویر از طریق self-attention.
- مستقل بودن از bias ساختاریِ فیلترهای کانولوشن (که در CNN محدود به محلی ترین همسایگی هاست)

در مقاله ذکر شده که ViT توانسته ویژگی های با معنای عمیق تری نسبت به CNN از پچ های تصویری استخراج کند و عملکرد بهتری در شناسایی بیماری های گیاهی نشان دهد، مخصوصاً زمانی که الگوهای ظاهری بیماری در کل تصویر پخش هستند.

- وقتی دادگان محدود است، کدام مدل بهتر عمل می کند؟ چرا؟

در حالت داده های محدود، معمولاً مدل های سنتی (مثل CNN) بهتر عمل می کنند. چون:

- ViT برخلاف CNN ساختار کانولوشنی ندارد و فاقد inductive bias (سوگیری ذاتی شبکه های کانولوشنی مثل locality و translation equivariance) است. این باعث می شود ViT برای یادگیری روابط بین پیکسل ها نیاز به حجم زیادی از داده داشته باشد تا بتواند آن روابط را از صفر یاد بگیرد.
- CNN از ابتدا با فرض ساختارهای محلی (local features) طراحی شده، بنابراین حتی با داده های کم هم می تواند ویژگی های ابتدایی (مثل لبه، بافت، فرم) را یاد بگیرد.

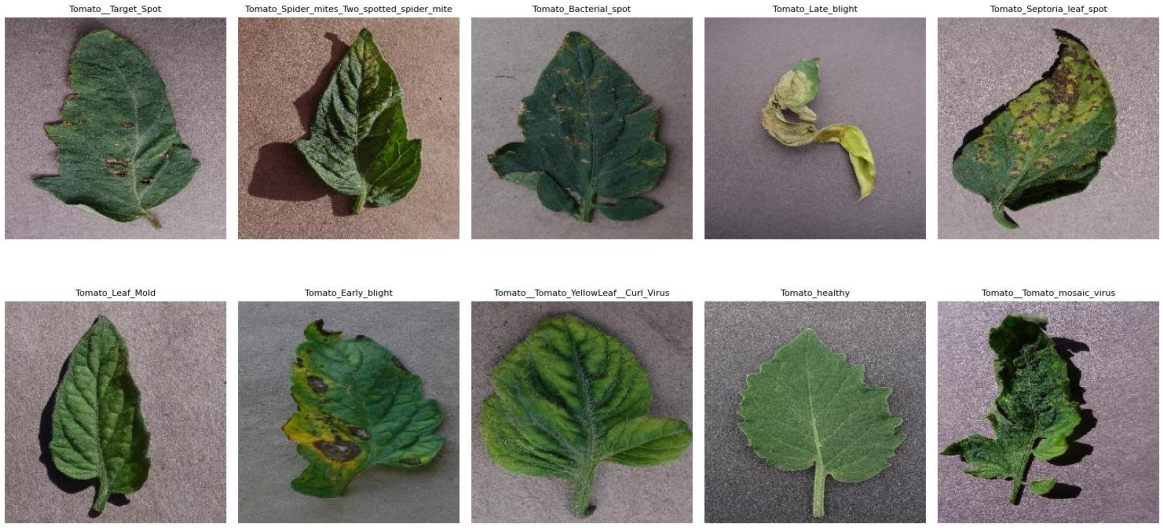
ViT ها برای دستیابی به عملکرد مطلوب نیازمند تعداد زیادی نمونه آموزشی هستند. در شرایطی که داده محدود است، ViT ممکن است دچار overfitting شود یا ویژگی های مهم را خوب یاد نگیرد، در حالی که CNN با استفاده از inductive bias داخلی خود عملکرد بهتری دارد.

2-1. آماده سازی داده ها

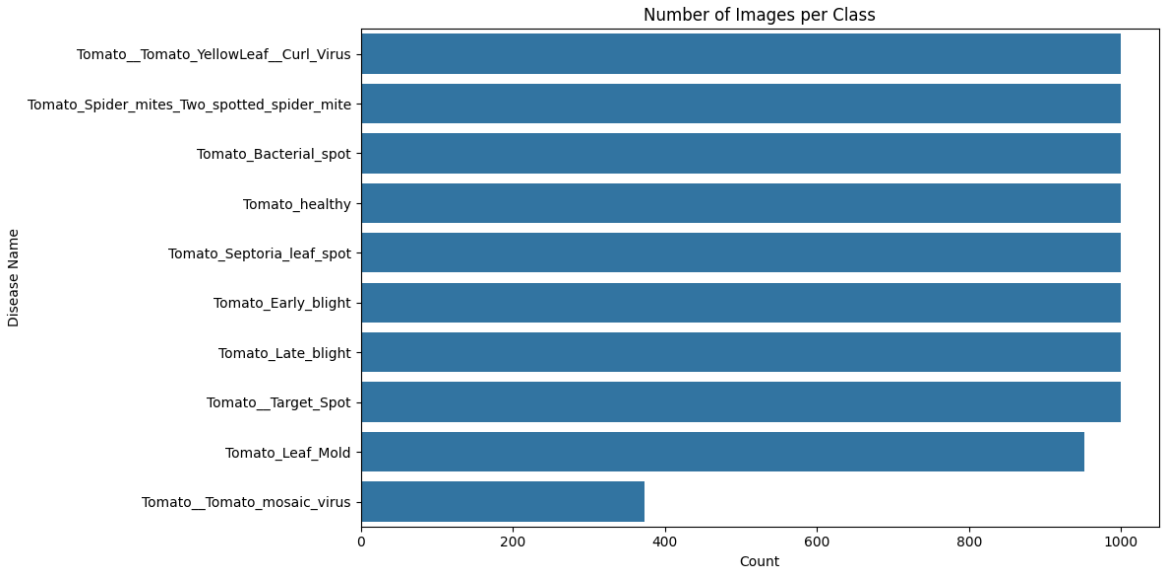
مجموعه داده ای که در این تمرین استفاده شده، شامل تصاویری از برگ های گیاه گوجه فرنگی است که به ده کلاس مختلف از بیماری ها (از جمله لکه باکتریایی، قارچ برگی، زردبرگی ویروسی، لکه هدف و غیره) و حالت سالم تقسیم شده اند. هر نمونه از داده شامل تصویر خام برگ، برچسب عددی کلاس، و نام بیماری مربوطه است. تصاویر به صورت فایل های فشرده (بایت های JPEG) ذخیره شده اند و مستقیماً از طریق پایتون و TensorFlow استخراج و پردازش شده اند.

این مجموعه داده در مجموع حدود 9300 نمونه دارد و داده های آن به صورت نسبتاً متعادل بین کلاس ها توزیع شده اند، به جز یکی دو کلاس خاص که تعداد تصاویر کمتری دارند (مثلاً ویروس موزائیک). هر تصویر نمای نزدیکی از یک برگ است و سعی شده تنوع نوری، زاویه و شدت بیماری در آن رعایت شود تا مدل بتواند ویژگی های واقعی بیماری ها را شناسایی کند.

1-2-1 نمایش نمونه داده ها



2-2-1 بررسی توازن داده ها



در دیاگرام بالا میتوان میزان داده های موجود از هر کلاس را در دیتاست مورد استفاده مشاهده کرد. برای بررسی توازن دادگان، تعداد تصاویر هر کلاس در مجموعه داده شمارش و مقایسه شد. نتیجه نشان داد که اگرچه اکثر کلاس ها حدود 1000 تصویر دارند، اما برخی کلاس ها مانند «ویروس موزائیک گوجه فرنگی» تعداد نمونه های بسیار کمتری دارند (حدود 370 تصویر). بنابراین، داده ها نامتوازن هستند و این می تواند منجر به کاهش دقت مدل در تشخیص کلاس های کم نمونه شود.

برای رفع این مشکل، به جای حذف یا کپی ساده ی داده ها، از روش های تقویت داده (Data Augmentation) استفاده شد. تقویت داده به شکلی اعمال شد که ویژگی های اصلی تصویر حفظ شوند و کیفیت تصاویر آسیب نبیند. مهم ترین روش های مورد استفاده عبارت اند از:

- چرخش تصادفی تصویر (برای افزایش تنوع زاویه ای)
- تغییر روشنایی و کنتراست (برای شبیه سازی شرایط نوری مختلف)
- برش تصادفی و بازنمایی تصویر (Crop & Resize)
- وارون سازی افقی یا عمودی تصویر

این روش ها به طور خاص برای کلاس های کم نمونه اعمال شدند تا تعداد آن ها به سطح کلاس های پرنمونه نزدیک شود، در حالی که از ایجاد داده های غیرواقعی یا اغراق شده جلوگیری شد. پس از اعمال تقویت داده، مجموعه ی جدیدی با توزیع نسبتاً متوازن آماده شد و برای آموزش مدل ها استفاده گردید.

مقدار داده ها پس از تقویت: (همگی کلاس ها 1000 نمونه دارند)

```
disease_name
Tomato__Tomato_YellowLeaf__Curl_Virus          1000
Tomato_Spider_mites_Two_spotted_spider_mite    1000
Tomato_Bacterial_spot                          1000
Tomato_healthy                                 1000
Tomato_Septoria_leaf_spot                      1000
Tomato_Leaf_Mold                               1000
Tomato__Tomato_mosaic_virus                    1000
Tomato_Early_blight                            1000
Tomato__Target_Spot                            1000
Tomato_Late_blight                             1000
Name: count, dtype: int64
```

3-2-1 پیش پردازش ها

با توجه به پیاده سازی انجام شده، پیش پردازش اصلی شامل تبدیل تصاویر خام از فرمت بایت به تانسور تصویری، تغییر اندازه تصاویر و نرمال سازی پیکسل ها به بازه [1, -1] بود. اندازه تصاویر برای مدل ViT طبق مقاله برابر 224×224 پیکسل در نظر گرفته شد و به صورت یکنواخت برای همه تصاویر اعمال گردید.

از آنجا که تمرکز تمرین بر ویژن ترنسفورمر ((ViT است و مدل 3Inception-V فقط برای مقایسه در نظر گرفته شده، اندازه تصاویر را مطابق نیاز ViT انتخاب کردیم، نه طبق استاندارد 3Inception-V (که معمولاً 299×299 است). بنابراین، تغییری در ساختار داخلی CNN اعمال نکردیم، چون هدف اصلی تمرین مقایسه عملکرد ViT بود.

برای ارزیابی صحیح عملکرد مدل و جلوگیری از بیش برازش (Overfitting)، مجموعه داده به دو بخش آموزش (Train) و اعتبارسنجی (Validation) تقسیم شد. در این تقسیم بندی، حدود 90% از داده ها برای آموزش مدل استفاده شد و 10% باقی مانده برای ارزیابی در مرحله ی اعتبارسنجی در نظر گرفته شد. این تقسیم بندی به صورت تصادفی و همراه با حفظ نسبت کلاس ها (Stratified Split) انجام شد تا توزیع کلاس ها در هر دو مجموعه تقریباً یکسان باقی بماند. این کار باعث شد مدل هم در حین آموزش از داده های متنوع استفاده کند و هم عملکرد واقعی آن روی داده های نادیده گرفته شده سنجیده شود.

3-1 مدل **CNN**

1-3-1 بارگذاری **Inception-V3**

در این بخش از تمرین، مدل 3Inception-V به صورت خام و بدون استفاده از وزن های از پیش آموزش دیده بارگذاری شد. این مدل صرفاً برای مقایسه با مدل ویژن ترنسفورمر استفاده شده است و هدف از آن ارزیابی عملکرد یک شبکه ی کانولوشنی سنتی در شرایطی مشابه با ViT می باشد. هنگام ساخت مدل، وزن ها برابر با None قرار گرفتند تا آموزش از ابتدا آغاز شود و تأثیری از یادگیری پیشین روی دیتاست های عمومی مانند ImageNet وجود نداشته باشد.

با توجه به اینکه مجموعه داده شامل 10 کلاس مختلف بیماری گیاه گوجه فرنگی است، لایه ی خروجی مدل به یک لایه ی Dense با 10 نورون و تابع فعال سازی softmax تغییر داده شد تا امکان طبقه بندی چندکلاسه فراهم شود. همچنین اگرچه اندازه ی ورودی پیش فرض 3Inception-V برابر 299×299 پیکسل است، اما در این تمرین برای هماهنگی با ViT و ساده سازی فرآیند، اندازه ی تصاویر به 224×224 نیز کاهش یافت که همچنان قابل قبول است.

2-3-1 نحوه کارکرد این مدل

از نظر ساختار، 3Inception-V یک شبکه ی کانولوشنی پیشرفته است که با هدف استخراج ویژگی های چندمقیاسی از تصاویر طراحی شده است. این مدل از ماژول های Inception استفاده می کند که در هر مرحله چندین فیلتر با اندازه های متفاوت را به صورت موازی روی داده ی ورودی اعمال کرده و خروجی های آن ها را با یکدیگر ادغام می کند. این ساختار باعث می شود مدل بتواند به طور هم زمان ویژگی های محلی و کلی تصویر را تشخیص دهد. همچنین با استفاده از فیلترهای ۱×۱، تعداد پارامترهای شبکه بهینه سازی شده و از افزایش بیش از حد حجم محاسبات جلوگیری می شود.

در نهایت، به دلیل برخورداری از سوگیری ساختاری محلی (local inductive bias)، مدل های مبتنی بر CNN مانند Inception-V3 در شرایطی که داده های آموزشی محدود باشند، معمولاً عملکرد بهتری از خود نشان می دهند. به همین دلیل استفاده از این مدل در تمرین، امکان تحلیل مقایسه ای مناسبی با ViT فراهم می سازد.

2-3-1 تابع هزینه مورد استفاده

در مقاله ی مرجع مربوط به ویژن ترنسفورمر، برای آموزش مدل از تابع خطای categorical cross-entropy استفاده شده است. این تابع خطا رایج ترین گزینه برای مسائل طبقه بندی چند کلاسه است که در آن، هر نمونه ی ورودی تنها به یک کلاس تعلق دارد و خروجی مدل به صورت یک توزیع احتمالی (softmax) بر روی همه کلاس ها در نظر گرفته می شود.

عملکرد این تابع به این صورت است که احتمال پیش بینی شده توسط مدل برای کلاس درست را با مقدار 1 (در برچسب one-hot) مقایسه کرده و با استفاده از لگاریتم منفی آن، میزان خطا را محاسبه می کند. اگر مدل برای کلاس درست احتمال بالایی پیش بینی کند، مقدار خطا کاهش می یابد و در نتیجه مدل بهبود می یابد.

فرمول ریاضی تابع به صورت زیر است:

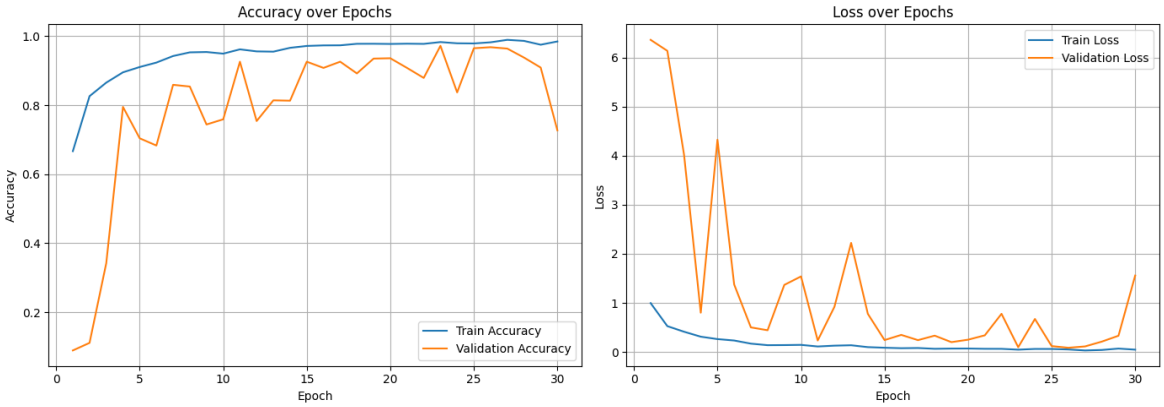
$$L = -\sum_{i=1}^{C} y_i . log(\hat{y}_i)$$

- C تعداد کلاس هاست
- y_i مقدار واقعی کلاس (در قالب one-hot)
- $\hat{y}_i$ احتمال پیش بینی شده برای کلاس i

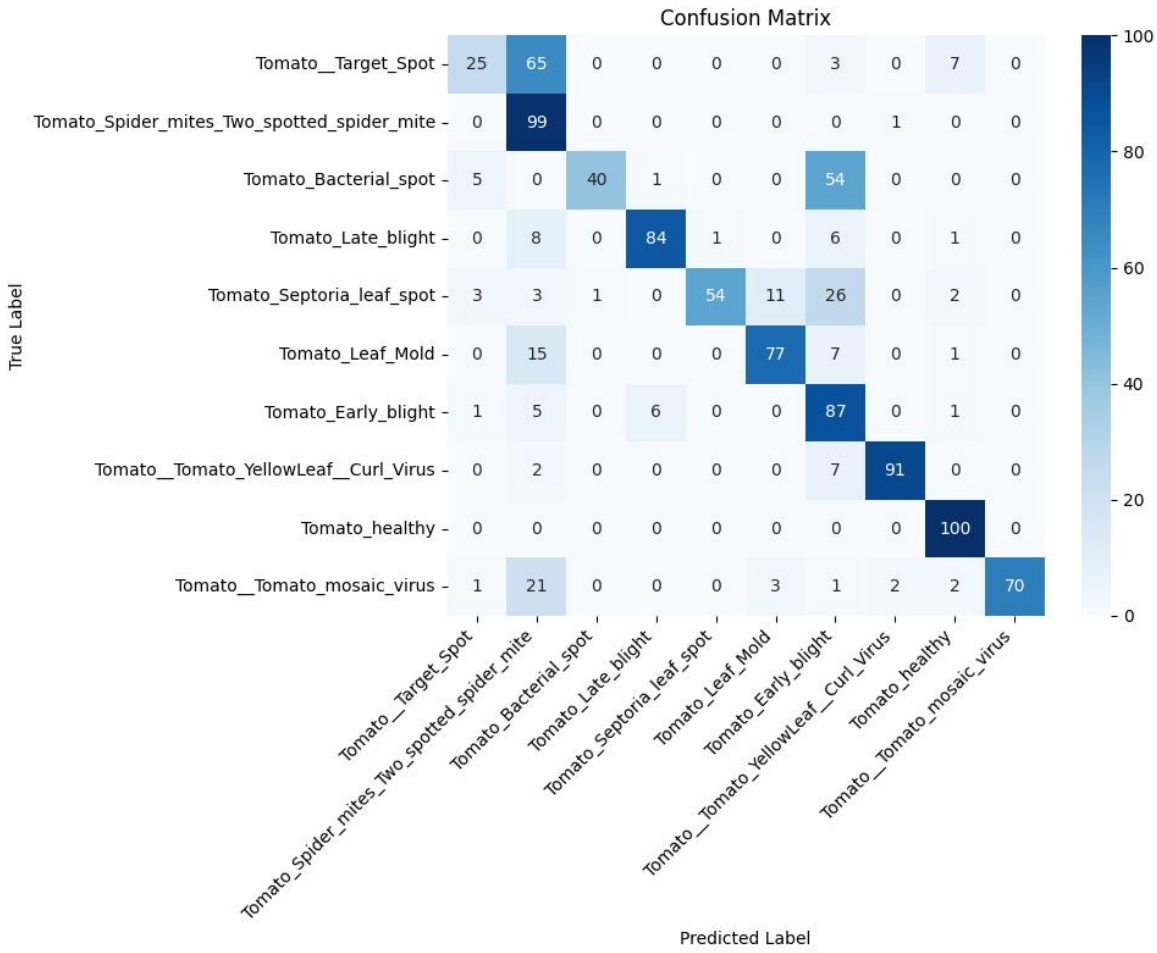
در این تمرین نیز با توجه به ماهیت چندکلاسه ی مسئله (10 نوع بیماری یا وضعیت برگ گوجه فرنگی)، استفاده از categorical_crossentropy انتخابی کاملاً مناسب و سازگار با ساختار داده هاست.

3-3-1 آموزش مدل

برای آموزش مدل، آن را به مدت 30 دوره (epoch) روی داده های آموزش آموزش دادیم و عملکرد آن را در هر دوره با داده های اعتبارسنجی ارزیابی کردیم. سپس با رسم نمودار دقت و خطا، روند یادگیری مدل را بررسی کردیم.



همچنین ماتریس آشفتگی نیز برای مدل 3Inception-V ترسیم شد تا نحوه ی پیش بینی آن روی کلاس های مختلف به صورت دقیق قابل تحلیل باشد.



این آموزش از پایه، بدون اتکا به دانش قبلی مدل، نشان داد که حتی ساختارهای سنتی مانند 3Inception-V می توانند در مسائل خاص مانند تشخیص بیماری گیاهان عملکرد قابل قبولی داشته باشند، به ویژه در شرایطی که داده ها محدود یا نامتوازن هستند.

4-1 مدل ViT

1-4-1 شرح مدل

در این بخش از تمرین، مدل ویژن ترنسفورمر ((ViT مطابق با معماری ارائه شده در مقاله ی مرجع پیاده سازی شد. این مدل به صورت کامل از ابتدا ساخته شده و در آن تمام اجزای اصلی مانند Patch Embedding، Positional Encoding، Transformer Blocks، Layer Normalization و MLP Head مطابق مشخصات مقاله طراحی و اجرا شدند.

مطابق مقاله، ورودی تصاویر به اندازه ی 224×224 پیکسل تنظیم شد و به پچ هایی به ابعاد 16×16 تقسیم گردید. در نتیجه، هر تصویر به 196 پچ (14×14) تقسیم شد. سپس این پچ ها به بردارهایی با ابعاد 64 نگاشت شدند. مدل

شامل 8 بلاک ترنسفورمر بود که در هر بلاک از 4 هد توجه چندگانه (Multi-Head Attention) و یک شبکه ی تمام وصل (MLP) با اندازه ی 128 استفاده شد. Dropout نیز با نرخ 0.1 در کل ساختار اعمال شد.

پس از تکمیل مدل، آن را با استفاده از داده های آموزش و اعتبارسنجی به مدت 20 دوره (epoch) آموزش دادیم. دقت و خطای مدل در هر دوره ثبت شد و در نهایت با رسم نمودارهای عملکرد، روند یادگیری مدل تحلیل شد. همچنین ماتریس آشفتگی نیز رسم شد تا نحوه ی طبقه بندی مدل در کلاس های مختلف بررسی شود.

برای مشاهده ی دقیق ساختار مدل، از متد ()summary استفاده شد که خروجی لایه به لایه را نمایش داد. همچنین با استفاده از plot_model و layer.output_shape، ابعاد خروجی هر لایه نیز بررسی شد تا از سازگاری ساختار داخلی مدل با مقاله اطمینان حاصل گردد.

در فرآیند پیاده سازی، تمام جزئیات اصلی معماری ViT با مقاله منطبق بود. تنها تفاوت جزئی این بود که برای فعال سازی در MLP به جای ReLU از تابع GELU استفاده شد. دلیل این انتخاب آن است که GELU به صورت پیش فرض در اکثر پیاده سازی های ViT استاندارد (از جمله ViT-B و DeiT) استفاده می شود و عملکرد بهتری در یادگیری ویژگی های پیچیده دارد. اگرچه مقاله به طور صریح نوع فعال سازی را ذکر نکرده، این انتخاب بهبود جزئی در پایداری آموزش ایجاد کرد.

به جز این مورد، باقی اجزا از جمله تعداد بلاک ها، تعداد هدها، ابعاد embedding، ابعاد MLP و ساختار کلی، کاملاً مطابق با معماری معرفی شده در مقاله پیاده سازی شده اند. بنابراین می توان گفت مدل ViT تمرین با دقت بالا از معماری مقاله تبعیت می کند.

لایه Patch Embedding در شبکه های مبدل تصویر به چه منظوری استفاده میشود؟ کاهش یا افزایش اندازه ی هر patch در این تمرین چه تاثیراتی بر روی خروجی دارد؟

لایه ی Patch Embeddingدر شبکه های مبدل تصویر (Vision Transformer – ViT) به این منظور استفاده می شود که تصویر ورودی که ذاتاً یک داده ی دوبعدی است (مثلاً 3×224×224)، به یک توالی بردارهای ویژگی تبدیل شود تا بتوان آن را مشابه داده های متنی در مدل ترنسفورمر پردازش کرد.

در این فرآیند، ابتدا تصویر به پچ های غیرهم پوشان (مثلاً 16×16 پیکسل) تقسیم می شود. هر پچ به صورت یک بردار تخت شده (flattened vector) در می آید و سپس از طریق یک لایه ی Dense به فضای embedding نگاشته می شود. حاصل این عملیات، یک دنباله از بردارهای embedding است که هرکدام نماینده ی یک بخش از تصویر هستند. این دنباله، ورودی اصلی بلاک های ترنسفورمر خواهد بود.

در این تمرین، اندازه ی پچ برابر با 16×16 انتخاب شد، که در تصویری با ابعاد 224×224، خروجی 196 پچ (14×14) را تولید می کند.

اگر اندازه ی پچ را کاهش دهیم (مثلاً 8×8):

- تعداد پچ ها افزایش می یابد (مثلاً به 784 پچ)
- دقت مدل در تشخیص جزئیات محلی بیشتر می شود
- اما هزینه محاسباتی بسیار بالا می رود (چون self-attention پیچیدگی زمانی مربعی دارد)

اگر اندازه ی پچ را افزایش دهیم (مثلاً 32×32):

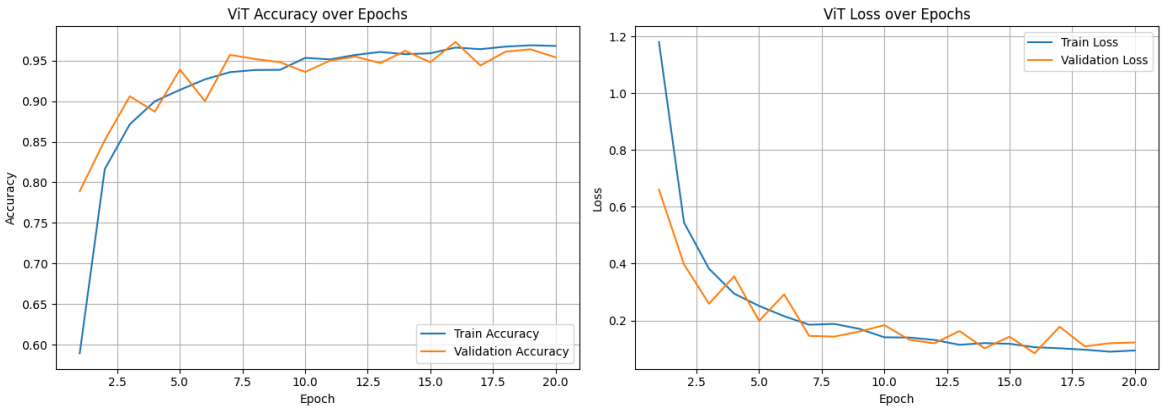
- تعداد پچ ها کاهش می یابد (مثلاً فقط 49 پچ)
- مدل سریع تر می شود و حافظه ی کمتری نیاز دارد
- اما ممکن است جزئیات ظریف تصویر از بین برود و دقت کاهش یابد

در این تمرین، انتخاب اندازه ی 16×16 تعادل خوبی بین کیفیت استخراج ویژگی ها و هزینه ی محاسباتی ایجاد می کند و مطابق پیشنهاد مقاله ی مرجع بوده است.

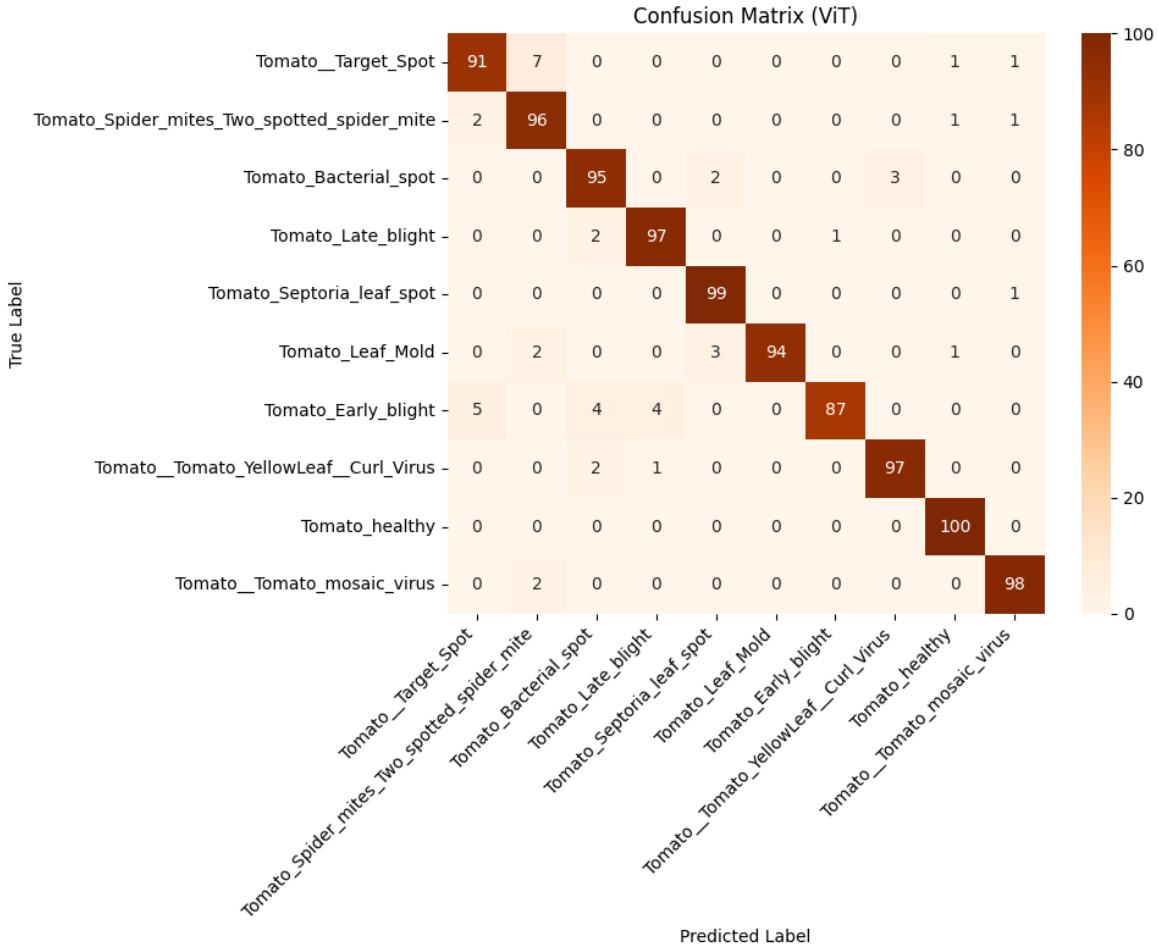
2-4-1 آموزش مدل

در این بخش از تمرین، مدل ویژن ترنسفورمر (ViT) طبق معماری مقاله ی مرجع پیاده سازی شده و سپس به مدت 20 دوره (epoch) روی مجموعه داده ی آماده شده آموزش داده شد. در هر دوره، میزان دقت (accuracy) و خطا (loss) مدل روی دو مجموعه ی آموزش (train) و اعتبارسنجی (validation) ثبت گردید تا عملکرد مدل در طول آموزش قابل تحلیل باشد.

پس از پایان آموزش، با استفاده از داده های ثبت شده، نمودار دقت و نمودار خطا رسم شد. این نمودارها روند یادگیری مدل را نشان دادند و مشخص کردند که مدل توانسته با افزایش تعداد اپوک ها، ویژگی های تصویری را بهتر بیاموزد و دقت خود را به تدریج افزایش دهد.



نمودار پیچیدگی (Confusion Matrix) مربوط به مدل ViT عملکرد کلی بسیار خوبی را نشان می دهد و بیانگر این است که مدل در بیشتر کلاس ها توانسته به درستی تصاویر را تشخیص دهد. خانه های قطری ماتریس (یعنی پیش بینی درست) تقریباً در همه ی کلاس ها مقادیر بالا دارند، که نشان دهنده ی دقت بالای مدل در طبقه بندی صحیح است.



1. کلاس های با دقت کامل:
مدل در کلاس «Tomato_healthy» به طور کامل درست عمل کرده است (100/100) که نشان دهنده ی تمایز ظاهری واضح برگ های سالم نسبت به نمونه های بیمار است.

2. کلاس هایی با اشتباه کم:
کلاس هایی مانند «Tomato_Septoria_leaf_spot»، «Tomato__Tomato_mosaic_virus»، و «Tomato_Late_blight» نیز با دقت بسیار بالا (۹۷٪ تا ۹۹٪) پیش بینی شده اند.

3. کلاس هایی با اشتباه نسبی بیشتر:
کلاس «Tomato_Early_blight» اشتباهات نسبتاً بیشتری دارد. تنها ۸۷ تصویر به درستی طبقه بندی شده و ۱۳ مورد به سایر کلاس ها نسبت داده شده اند (به ویژه با «Target Spot» و « Bacterial Spot» که ممکن است شباهت ظاهری داشته باشند).
این اشتباهات می تواند ناشی از هم پوشانی ویژگی های تصویری بین این بیماری ها باشد.

4. خطاهای جزئی در مرز کلاس ها:
برخی کلاس ها مانند «Target Spot» و «Spider Mites» گاهی با یکدیگر اشتباه گرفته شده اند که احتمالاً به دلیل شباهت در الگوهای ظاهری لکه ها یا نوع آسیب روی برگ است.

مدل ViT در این مسئله عملکرد بسیار قوی دارد و دقت بالا در اکثر کلاس ها نشان می دهد که توانسته ویژگی های معنادار را از تصاویر استخراج کند. با این حال، برای بهبود عملکرد در کلاس های دشوارتر، می توان از تکنیک هایی مانند افزایش داده هدفمند (targeted augmentation) یا استفاده از وزن دهی به کلاس ها در تابع خطا بهره برد.

5-1 تحلیل و نتیجه گیری

در این تمرین، دو مدل ViT و 3Inception-V برای طبقه بندی 10 نوع بیماری یا وضعیت برگ گیاه گوجه فرنگی آموزش داده شدند. پس از آموزش کامل و تحلیل عملکرد هر دو مدل، مشخص شد که مدل ViT دقت بالاتری نسبت به 3Inception-V دارد.

از نظر دقت ViT به دقت 95.4% روی مجموعه ی اعتبارسنجی دست یافت، در حالی که 3Inception-V عملکرد پایین تری داشت (تقریباً 85%) که از ماتریس آشفتگی آن نیز مشخص است. 3Inception-V در برخی کلاس ها مانند Bacterial Spot، Target Spot و Mosaic Virus دقت پایین و اشتباهات زیادی نشان داد.

از نظر **precision**(دقت)، میانگین precision برای ViT برابر با 0.9551 است که نشان می دهد این مدل در تشخیص کلاس ها بسیار دقیق عمل کرده و false positive کمتری دارد. به ویژه، در کلاس هایی مانند «Tomato_mosaic_virus»، «Tomato_healthy» و «Tomato_Late_blight» دقت مدل تقریباً کامل است. در مقابل، 3Inception-V در چند کلاس خاص، مانند «Target Spot»، دقت بسیار پایینی داشت (تنها 25 نمونه از 100 به درستی طبقه بندی شدند).

از نظر **recall**(صحت)، ViT با مقدار 0.954 عملکردی پایدار و یکنواخت در تمام کلاس ها ارائه داد، در حالی که 3Inception-V در کلاس هایی مثل Septoria Leaf Spot و Leaf Mold به شدت دچار کاهش صحت شد و تعداد زیادی از نمونه های درست را به اشتباه به کلاس های دیگر نسبت داد.

از نظر تعداد پارامترها، اختلاف بسیار زیادی بین دو مدل وجود دارد:

- مدل 3Inception-V شامل 21,823,274 پارامتر (حدود 83.25 مگابایت) است.
- در حالی که مدل ViT تنها 728,138 پارامتر (حدود 2.78 مگابایت) دارد.

این اختلاف به وضوح نشان می دهد که ViT بسیار سبک تر و کم حجم تر از 3Inception-V است، اما با این وجود، عملکرد دقیق تر و پایدارتری دارد. این نکته اهمیت بسیار زیادی دارد، به ویژه در کاربردهایی مانند پیاده سازی روی دستگاه های لبه ای یا کم منبع.

اگرچه ViT در این تمرین عملکرد برتری داشت، اما در برخی شرایط خاص، مدل 3Inception-V می توانست نتیجه ی بهتری ارائه دهد.

به ویژه، در شرایطی که داده های آموزشی محدود، نویزی یا ناقص باشند، 3Inception-V به دلیل داشتن سوگیری ساختاری (inductive bias) نسبت به الگوهای محلی مانند لبه ها، بافت ها و اشکال، می تواند بهتر عمل کند. این ساختار باعث می شود که شبکه های CNN در سناریوهایی با داده ی کم بتوانند ویژگی های مفیدی را از تصویر استخراج کنند، در حالی که ViT که مبتنی بر self-attention است، برای یادگیری روابط پیچیده ی بین پچ ها نیاز به حجم داده ی بیشتری دارد.

همچنین، اگر برای مدل Inception-V3 از وزن های از پیش آموزش دیده (pretrained) استفاده می شد (مثلاً آموزش دیده روی ImageNet)، می توانست در همان چند epoch اول ویژگی های تصویری خوبی را یاد بگیرد و دقت خود را تا حد زیادی بهبود دهد.

در مجموع، در شرایطی مانند:

- کمبود داده ی آموزشی
- نیاز به استفاده از مدل های پیش آموزش دیده
- یا وقتی یادگیری ویژگی های محلی بسیار حیاتی است مدل Inception-V3 می توانست بهتر عمل کند یا لااقل رقابت نزدیک تری با ViT داشته باشد.

پرسش 2: Robust Zero-Shot Classification

در سال های اخیر، مدل های یادگیری عمیق مانند CLIP توانسته اند با استفاده از ترکیب زبان و تصویر، مسئله ی طبقه بندی صفر-نمونه (Zero-Shot Classification) را به طور مؤثری حل کنند. این مدل ها بدون نیاز به آموزش مستقیم روی دسته های خاص، تنها با درک معنایی توضیحات متنی، قادر به پیش بینی برچسب تصاویر جدید هستند. اما با وجود این موفقیت ها، یکی از چالش های مهمی که همچنان وجود دارد، آسیب پذیری این مدل ها در برابر حملات خصمانه است. حملاتی که با تغییراتی بسیار جزئی در ورودی، می توانند عملکرد مدل را به شدت کاهش دهند. در این بخش، به بررسی معماری CLIP، نحوه ی طبقه بندی تک ضرب، و نقاط ضعف آن در برابر حملات FGSM و PGD می پردازیم و سپس روش هایی برای مقاوم سازی مدل در برابر این تهدیدات را تحلیل خواهیم کرد.

2.1- پس از مطالعه مقاله به سوالات زیر پاسخ میدهیم

2.1.1- دو روش تولید نمونه های تخاصمی **FGSM** و **PGD** را با شرح جزئیات بهینه سازی توضیح دهید؟

همانطور که در صورت مسئله نیز مشخص شده است نمونه های خصمانه نمونه هایی هستند که با افزودن نویز یا افزودن تغییرات بسیار کوچک به داده های اصلی با هدف به اشتباه انداختن مدل های شبکه عصبی بدون اینکه انسان را دچار اشتباه کنند تولید میشود. برای تولید این نمونه ها از دو روش تقریبا مشابه استفاده می شود که یکی FGSM و دیگری PGD است. در واقع PGD روش پیشرفته تر قوی تر از FGSM است که به نوعی با تکرار کردن روش FGSM انجام میشود و به همین دلیل دقت بیشتری دارد ولی اجرای آن زمان بیشتری طول خواهد کشید.

روش FGSM: این روش یکی از ساده ترین و سریع ترین روش های تولید نمونه خصمانه است. فقط با یک مرحله گرادیان گیری، داده ی اولیه را در جهتی که بیشترین افزایش در تابع هزینه را ایجاد می کند، مختل می کنیم.

میدانیم که جهتی که بیشترین افزایش در تابع هزینه را ایجاد میکند در جهت گرادیان تابع هزینه است و بنابراین فرمول محاسبه این روش به صورت زیر است:

$$((\delta = \epsilon \cdot sign(\nabla_x J(\vartheta, x, y$$

که بنابراین میزان دلتا را به x که داده اصلی است اضافه میکنیم و آنگاه داده تولید شده، نمونه خصمانه ما خواهد بود.

$$x_{ae} = x + \delta$$

که در این فرمول x برابر با ورودی اصلی یعنی تصویر است ∇_x برابر با گرادیان x است و تابع sign به ما تنها جهت گرادیان را میدهد. همچنین J تابع هزینه ما با پارامتر تتا است و y خروجی مدل است.

این روش به دلیل اینکه تنها یکبار اجرا می شود بسیار سریع است و همانطور که مشخص است پیاده سازی آن بسیار ساده است ولی ممکن است همیشه موثر نباشد و مدل های مقاوم تر میتوانند جلوی آن را بگیرند.

روش PGD: این روش نسخه ی پیشرفته تر و قوی تر FGSM است که به صورت چند مرحله ای تکرار شونده عمل می کند. پس از هر مرحله، داده مختل شده را به یک کره به شعاع ϵ اطراف داده اصلی پروجکت می کنیم تا مطمئن شویم شدت حمله بیش از حد نیست.

فرمول این روش به صورت زیر است:

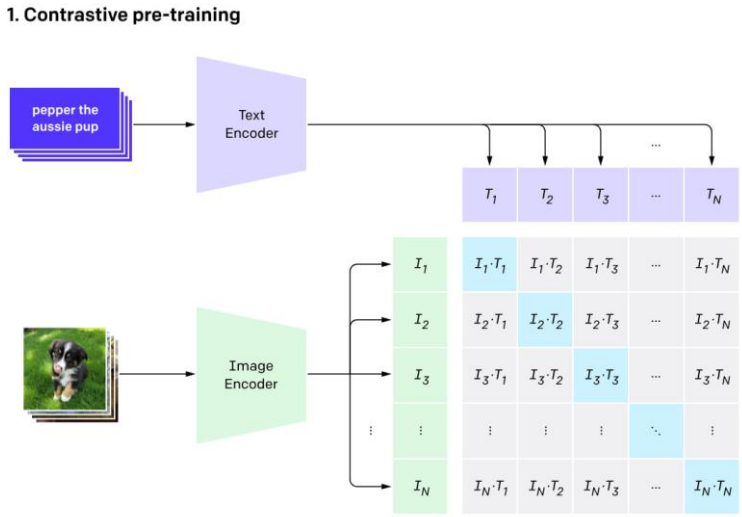
$$(((x'_{t+1} = \Pi_{\epsilon}(x_t + \alpha \cdot sign(\nabla_x J(\Theta, x_t, y$$

که به صورت تکرار شونده است و تغییراتی بر روی ورودی x انجام میدهد و به x_{t+1} میرسد. در هر تکرار بررسی میشود که در گوی به شعاع اپسیلون قرار داشته باشد و در غیر اینصورت به درون گوی پروجکت میشود. این روش عملکرد بسیار بهتری دارد و موثر تر است ولی به دلیل تکرار شوندگی سرعت کمتری از روش اول دارد.

2.1.2- . معماری مدل **CLIP** را همراه با شکل شرح داده و در مورد تابع زیان و نوع آموزش **Contrastive** توضیح دهید.

مدل CLIP یک معماری Multi-modal است که می تواند متن و تصویر را همزمان درک کند. این مدل توسط OpenAI طراحی شده تا بتواند به صورت Zero-Shot، یعنی بدون آموزش مستقیم روی یک دسته بندی خاص، تصاویر را طبقه بندی کند. این مدل از دو شبکه عصبی مجزا تشکیل شده است.

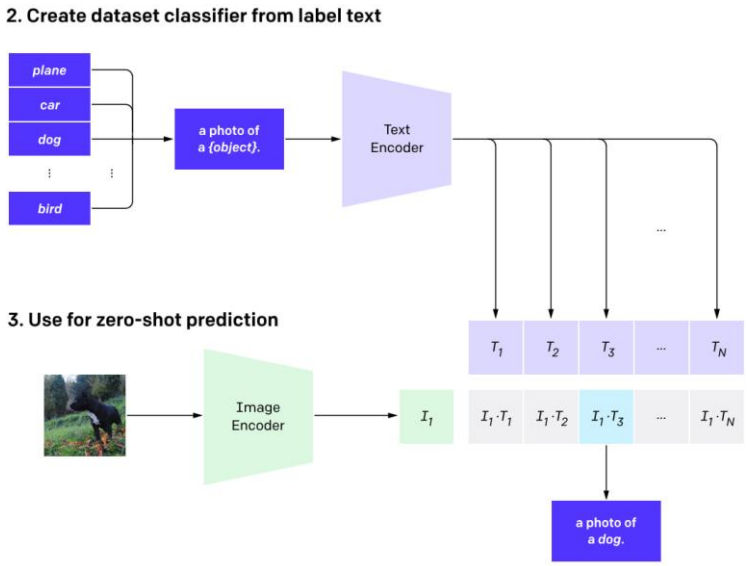
تصویر 2.1- انکودر های تصویر و متن در مدل **CLIP**



همانطور که در تصویر مشخص است از دو انکودر تشکیل شده است که یکی برای تصاویر و یکی برای متن ها استفاده میشود. انکودر تصاویر معمولاً از معماری هایی مثل ResNet یا Vision Transformer استفاده می کند و خروجی آن برداری از ویژگی های تصویر با بعد ثابت مثلاً 512 بعد است. اینکودر متن از مدل ترنسفورمر مانند GPTبرای تبدیل متن به بردار استفاده میکند و خروجی آن بردار ویژگی متن در همان فضای بُرداری تصویر است.

هدف این مدل این است که این دو بردار در فضای برداری به هم نزدیک باشند و برای این کار از contrastive loss استفاده میکند و بدین صورت عمل میکند که در هر بچ مدل سعی می کند تصویر و متن که با هم تطابق دارند را در فضای برداری به یکدیگر نزدیک کند و همچنین فاصله تصاویر و متن های ناهماهنگ را زیاد کند.

تصویر 2.2- ساختار **classifier** مدل **CLIP**



در این تصویر در قسمت بالا، ابتدا یک لیست از برچسب های متنی را در اختیار داریم. برای این که این برچسب ها قابل استفاده برای مدل باشند، ابتدا به صورت جمله ای توصیفی تبدیل می شوند. این جملات توسط رمزگذار متن به بردارهایی در یک فضای ویژگی مشترک تبدیل می شوند که نشان دهنده ی معنای هر جمله در زبان طبیعی هستند.

در بخش پایین تصویر، یک عکس جدید که قرار است مدل آن را طبقه بندی کند، وارد رمزگذار تصویر می شود. این رمزگذار هم مانند رمزگذار متنی، تصویر را به یک بردار عددی در همان فضای ویژگی مشترک تبدیل می کند. حالا مدل با محاسبه ی شباهت بین بردار تصویر و تمام بردارهای متنی که قبلاً ساخته شده اند، بررسی می کند که تصویر به کدام جمله شباهت بیشتری دارد. در این مثال، بیشترین شباهت با جمله ی "a photo of a dog" به دست آمده و در نتیجه، مدل نتیجه گیری می کند که تصویر مربوط به سگ است، بدون این که هیچ آموزشی برای دسته بندی "dog" در حین آموزش دیده باشد.

این روش به مدل این امکان را می دهد که بدون نیاز به بازآموزی، تنها با دریافت توضیحات متنی برچسب ها، بتواند دسته بندی های جدید را انجام دهد. به همین دلیل CLIP در شرایطی که داده های آموزشی برای برخی دسته ها در دسترس نیست، عملکرد بسیار خوبی دارد و از آن به عنوان یکی از مدل های موفق در حوزه ی Zero-Shot یاد می شود.

2.1.3- تفاوت های کلیدی بین طبقه بندی عادی و طبقه بندی تک ضرب را توضیح دهید و همراه با شکل بیان کنید این عمل چگونه با مدل **CLIP** انجام میپذیرد.

در روش های طبقه بندی سنتی، مدل برای شناسایی هر دسته باید با نمونه های متعددی از همان دسته آموزش ببیند. مثلاً اگر بخواهیم مدلی گربه را تشخیص دهد، باید هزاران تصویر برچسب خورده از گربه به آن نشان دهیم تا مدل یاد بگیرد. به این حالت می گوییم طبقه بندی عادی که کاملاً وابسته به آموزش مستقیم برای هر کلاس است.

اما در مدل CLIP با مفهوم جدیدی به نام طبقه بندی تک ضرب (Zero-Shot Classification) روبه رو هستیم. در این روش، نیازی نیست که مدل به طور خاص برای دسته های خاصی آموزش ببیند. کافی است که برچسب ها یا نام کلاس ها را به صورت جمله های متنی توصیف کنیم، مانند "a photo of a cat" یا " a photo of a

dog"، و مدل می تواند تنها با توجه به معنای این جملات و تصویر جدید، پیش بینی کند که تصویر به کدام برچسب شباهت دارد.

در تصویر 2.1، این فرآیند به خوبی نشان داده شده است. ابتدا متن مربوط به هر کلاس وارد رمزگذار متنی می شود تا بردارهای ویژگی متنی ساخته شود. سپس تصویر جدید وارد رمزگذار تصویری شده و به بردار ویژگی تصویری نگاشته می شود. در مرحله ی بعد، مدل شباهت بین تصویر و تمام بردارهای متنی را محاسبه کرده و جمله ای را که بیشترین شباهت دارد به عنوان خروجی انتخاب می کند.

تفاوت اصلی اینجاست که در CLIP، مدل از قبل روی مجموعه ی بزرگی از تصویر و متن ها آموزش دیده و یاد گرفته که بین زبان طبیعی و ویژگی های بصری ارتباط برقرار کند. بنابراین می تواند دسته هایی را هم که هیچ وقت به آن آموزش داده نشده، فقط با کمک توصیف زبانی آن ها تشخیص دهد. این ویژگی CLIP را به مدلی بسیار قدرتمند برای کاربردهایی مانند دسته بندی با داده های کم یا ناشناخته تبدیل کرده است.

2.1.4- در شرایط مختلف دو نوع حمله خصمانه جعبه سفید و جعبه سیاه وجود دارد . هر کدام را توضیح دهید و سپس این دو رویکرد را از جنبه های مختلف دلخواه با یکدیگر مقایسه کنید.

همانطور که در بالاتر اشاره شد حملات خصمانه به روش هایی گفته می شود که با ایجاد تغییرات جزئی و هدفمند در ورودی، باعث می شوند مدل به اشتباه بیفتد که به دو دسته حملات خصمانه جعبه سفید و حملات خصمانه جعبه سیاه تقسیم بندی میشوند. تفاوت اصلی این دو حمله در میزان دانشی است که مهاجم نسبت به مدل دارد.

در حمله ی جعبه سفید، فرض بر این است که مهاجم به همه ی اطلاعات مدل دسترسی دارد؛ یعنی هم ساختار مدل (معماری شبکه عصبی) و هم وزن ها و حتی loss function آن را می داند. با این اطلاعات، مهاجم می تواند به صورت مستقیم گرادیان تابع هزینه نسبت به ورودی را محاسبه کرده و دقیقاً در جهتی که مدل بیشترین ضعف را دارد، ورودی را مختل کند. روش هایی مانند FGSM وPGD که در بالاتر در مورد آنها توضیح داده شد، نمونه های معروف حملات جعبه سفید هستند.

در مقابل، در حمله ی جعبه سیاه، فرض می شود که مهاجم هیچ اطلاعات داخلی از مدل ندارد؛ یعنی نمیداند مدل چگونه ساخته شده یا چه پارامترهایی دارد. تنها چیزی که در اختیار دارد، ورودی و خروجی مدل است. بنابراین حمله به جای گرادیان گیری مستقیم، به روش هایی مانند تست تصادفی، الگوریتم های بهینه سازی بدون مشتق و... متوسل می شود. این نوع حمله معمولاً کندتر و سخت تر است اما از لحاظ امنیتی مهم تر، چون بیشتر شبیه شرایط دنیای واقعی است.

از نظر میزان قدرت، حملات جعبه سفید معمولاً مؤثرتر هستند چون دقیقاً بر اساس ساختار مدل طراحی می شوند. اما حملات جعبه سیاه در عمل اهمیت بیشتری دارند چون در بسیاری از سیستم های واقعی تنها دسترسی به خروجی ممکن است.

از آنجا که حملات جعبه سفید نیاز به دسترسی به کل اطلاعات مدل دارد در واقعیت خیلی کاربرد ندارد ولی در صورت وجود داشتن این اطلاعات بسیار موثر خواهد بود و نتیجه بهتری نسبت به حملات جعبه سیاه دارند.

2.1.5- بیان بدارید چرا حملات انتقالی یک تهدید جدی برای مسائل دنیای واقعی به نسبت حملات جعبه سفید محسوب می شوند.

حملات انتقالی زمانی اتفاق می افتند که یک نمونه خصمانه، که برای فریب دادن یک مدل خاص (مثلاً مدل A) طراحی شده، بتواند مدل دیگری (مثلاً مدل B) را نیز فریب دهد، حتی اگر این مدل ساختار یا پارامترهای متفاوتی داشته باشد.

در نگاه اول، ممکن است تصور شود که چون مهاجم به ساختار مدل هدف دسترسی ندارد (یعنی در حالت جعبه سیاه قرار دارد)، نمی تواند حمله موفقی انجام دهد. اما واقعیت این است که بسیاری از مدل های یادگیری عمیق رفتارهای مشابهی در فضاهای ورودی دارند، مخصوصاً وقتی روی یک مسئله ی مشابه آموزش دیده باشند.
بنابراین، اگر مهاجم یک مدل دیگر را که در دسترس دارد انتخاب کرده و حمله خصمانه را روی آن طراحی کند، به احتمال بالا همان نمونه خصمانه می تواند مدل هدف را نیز فریب دهد. این ویژگی را قابلیت انتقال پذیری حمله می نامند.

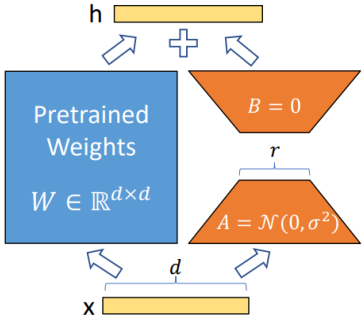
این مسئله در دنیای واقعی بسیار مهم و خطرناک است. چرا که در بسیاری از کاربردها (مانند خودروهای خودران، سامانه های تشخیص چهره) مهاجم به کد یا ساختار داخلی مدل دسترسی ندارد، اما می تواند با ساخت مدل مشابه و طراحی حمله روی آن، به سادگی به مدل اصلی آسیب وارد کند. به عبارت دیگر، برخلاف حملات جعبه سفید که نیازمند دسترسی کامل هستند، حملات انتقالی حتی در شرایط دسترسی محدود (جعبه سیاه) نیز می توانند موثر باشند و همین موضوع آن ها را به یک تهدید جدی تر تبدیل می کند.

در نتیجه، مهم ترین علت اهمیت حملات انتقالی در مقایسه با حملات جعبه سفید، عملی بودن آن ها در شرایط واقعی است؛ جایی که فرض دسترسی کامل به مدل معمولاً برقرار نیست، اما مهاجم همچنان می تواند با روش هایی هوشمندانه، آسیب پذیری مدل را آشکار کند و از آن استفاده نماید.

2.1.6- . روش تنظیم دقیق **LoRA** را همراه با شکل شرح دهید و سه علت استفاده از این روش نسبت به دیگر روش ها را بیان کنید و توضیح دهید.

LoRA یک روش نوین و کم هزینه برای fine-tuning مدل های بزرگ زبان یا تصویر است، بدون آن که نیاز باشد تمام پارامترهای مدل را مجدداً بروزرسانی کنیم. ایده ی اصلی LoRA این است که ماتریس های سنگین وزن در مدل های ترنسفورمر را با دو ماتریس کم رتبه جایگزین کند، به گونه ای که فقط این ماتریس های سبک در طول آموزش تغییر کنند، نه کل پارامترهای مدل.
در روش های کلاسیک fine-tuning، معمولاً همه ی پارامترهای مدل به روزرسانی می شوند که این کار از لحاظ مصرف حافظه، زمان آموزش و منابع محاسباتی بسیار پرهزینه است. اما LoRA با افزودن فقط یک تغییر کوچک به مسیر محاسباتی مدل، به مدل اجازه می دهد که دانش جدید را یاد بگیرد، در حالی که ساختار و پارامترهای اصلی مدل دست نخورده باقی می مانند.

شکل 2.3: شماتیک ساختار **LoRA** (آورده شده در مقاله اصلی)



همانطور که در تصویر مشخص است بجای اینکه کل وزن ها fine-tune بشوند تنها دو ماتریس کم رتبه A و B آموزش داده میشوند و در نهایت به وزن های ماتریس اصلی اضافه میشوند. این کار مزایای بسیار زیادی نسبت به fine-tune کردن وزن های ماتریس اصلی دارد:

-1صرفه جویی شدید در حافظه و منابع:

در LoRA، تعداد پارامترهایی که آموزش می بینند به شدت کاهش می یابد (مثلاً فقط چند میلیون در مقابل صدها میلیون یا میلیارد پارامتر). این باعث می شود بتوان مدل های بزرگ را حتی روی سخت افزارهای محدود نیز fine-tune کرد.

2- کاهش خطر overfitting:

چون فقط بخش کوچکی از مدل تغییر می کند، خطر بیش برازش روی داده های خاص (مثلاً مجموعه ی آموزشی کوچک یا حملات خصمانه) کمتر می شود.

3-امکان استفاده مجدد از مدل اصلی:

مدل پایه بدون تغییر باقی می ماند، بنابراین می توان از یک مدل ثابت برای چندین وظیفه مختلف فقط با لایه های LoRA متفاوت استفاده کرد، بدون نیاز به ذخیره چند نسخه از مدل.

2.1.7- دو مقاله پژوهشی که تابع زیان **CLIP** را گسترش یا بهبود میدهند پیدا کنید و هر کدام را در یک الی دو پاراگراف خلاصه کنید. همچنین توضیح دهید که تاثیر این توابع زیان بر افزایش مقاومت مدل **CLIP** چگونه بوده است.

در این مقاله (*Learning to Prompt for Vision-Language Models*) نویسندگان مدل CLIP را با معرفی مفهوم Context Optimization توسعه داده اند. در CLIP کلاس ها با جملاتی ثابت مانند "a photo of a cat" توصیف می شوند، اما در CoOp، متن هر برچسب به صورت قابل یادگیری تعریف می شود؛ یعنی توکن های متن به صورت خودکار و با هدف بهینه سازی تابع زیان، آموزش داده می شوند. این فرآیند باعث می شود که تطابق میان ویژگی های تصویری و متنی بهتر صورت گیرد و مدل در برابر نویزهای مفهومی مقاوم تر شود. به طور خاص، CoOp نه تنها دقت مدل را در سناریوهای zero-shot افزایش می دهد، بلکه مقاومت آن را در برابر تغییرات جزئی در ساختار متن یا تصویر نیز بهبود می بخشد.

در مقاله RobustCLIP: Adversarial Fine-tuning for Robust Vision-Language Models،نویسندگان مدل CLIP را با استفاده از یک روش fine-tuning خصمانه تحت عنوان RobustCLIP مقاوم سازی می کنند. ایده ی اصلی این است که در طول فرایند آموزش، نمونه های تصویری و متنی که به صورت خصمانه تغییر یافته اند به مدل داده می شود، تا شبکه یاد بگیرد در برابر این گونه تغییرات مقاومت کند. برخلاف CLIP اصلی که فقط از جفت های عادی تصویر و متن استفاده می کند، RobustCLIP سعی می کند این جفت ها را با نویزهای هدفمند مختل کند و سپس مدل را به نحوی تنظیم کند که همچنان پیش بینی درست داشته باشد.

نتایج مقاله نشان می دهد که این روش باعث افزایش پایداری مدل در مواجهه با حملات خصمانه مانند FGSM و PGD شده است. به ویژه، مدل های آموزش دیده با RobustCLIP در مقایسه با نسخه ی اصلی CLIP، در داده های مخرب شده نیز دقت بالاتری دارند و در محیط های واقعی قابل اطمینان تر هستند.

نکته ی قابل توجه در RobustCLIP این است که این روش بدون نیاز به تغییر اساسی در معماری CLIP، تنها با تغییر در فرآیند آموزش و طراحی هوشمندانه تابع زیان، می تواند مقاومت مدل را به طور چشمگیری افزایش دهد. همچنین این روش قابلیت تعمیم به داده های دنیای واقعی را دارد، چرا که در تست های انجام شده حتی در سناریوهای cross-dataset نیز عملکرد پایداری از خود نشان داده است. همین ویژگی باعث شده RobustCLIP به عنوان یک راه حل عملی و موثر برای ایمن سازی مدل های تصویر و متنی در برابر تهدیدات خصمانه مطرح شود.

2.2- پیاده سازی و مقایسه روش های آموزش خصمانه

در این بخش، قرار است با بهره گیری از مجموعه داده CIFAR-10 و کتابخانه های PyTorch ، torchvision ، Transformers و PEFT ، فرآیند آموزش و ارزیابی مدل های تک ضرب CLIP را در مقابل حملات خصمانه به صورت کامل پیاده سازی کنیم.

2.2.1- آشنایی با دیتاست

در این بخش ابتدا با دیتاست آشنا میشویم. این دیتاست شامل 10 کلاس است که با اعداد یک تا ده نشان داده میشود و هر عدد نشان دهنده یک شی است. هر تصویر اندازه 30*30 دارد که آنرا به 224*224 تبدیل میکنیم و همچنین از نرمال سازی مناسب این دیتاست نیز استفاده میکنیم.

در این دیتاست برای میانگین و انحراف معیار از مقادیر زیر استفاده شده است:

میانگین :[0.48145466, 0.4578275, 0.40821073]

انحراف معیار: [0.26862954, 0.26130258, 0.27577711]

همچنین پنج نمونه از تصاویر اصلی دیتاست پیش از انجام نرمال سازی به شکل زیر است:

تصویر 2.4: نمونه تصاویر دیتاست

این دیتاست شامل دو روش بارگذاری متفاوت برای داده های تست و آموزش است که جداگانه آنها را بارگذاری کردیم و سپس داده های آموزش را به نسبت 20 به 80 به اعتبارسنجی و آموزش تقسیم کردیم.

2.2.2- آماده سازی مدل

در این بخش، ابتدا مجموعه داده های CIFAR-10 را آماده می کنیم و سپس مدل CLIP را به صورت مستقیم بارگذاری می کنیم. برای جلوگیری از به روزرسانی وزن ها در این مرحله، CLIP را در مود eval قرار می دهیم. از سوی دیگر، برای تولید تصاویر خصمانه از یک مدل ResNet-20 که پیش تر روی CIFAR-10 آموزش دیده استفاده خواهیم کرد و آن را نیز در حالت eval نگه می داریم تا رفتاری یکنواخت داشته باشد. در قدم بعدی، با استفاده از عبارات توصیفیِ هر کلاس CIFAR-10—مثل «a photo of an airplane» یا « a photo of a

automobile»—بردارهای متنی مربوط به هر کلاس را از طریق بخش متنی CLIP استخراج می کنیم. این بردارها بعدها در فرایند طبقه بندی تک ضرب و محاسبه ی تابع هزینه تقابلی نقش کلیدی خواهند داشت.

در زیر چندین نمونه از تصاویر، مقادیر واقعی و مقادیر پیشبینی شده توسط این مدل نشان داده شده است که این مدل دقت بسیار خوبی دارد.

تصویر 2.2.5: چندین نمونه از حالت اولیه مدل **CLIP**

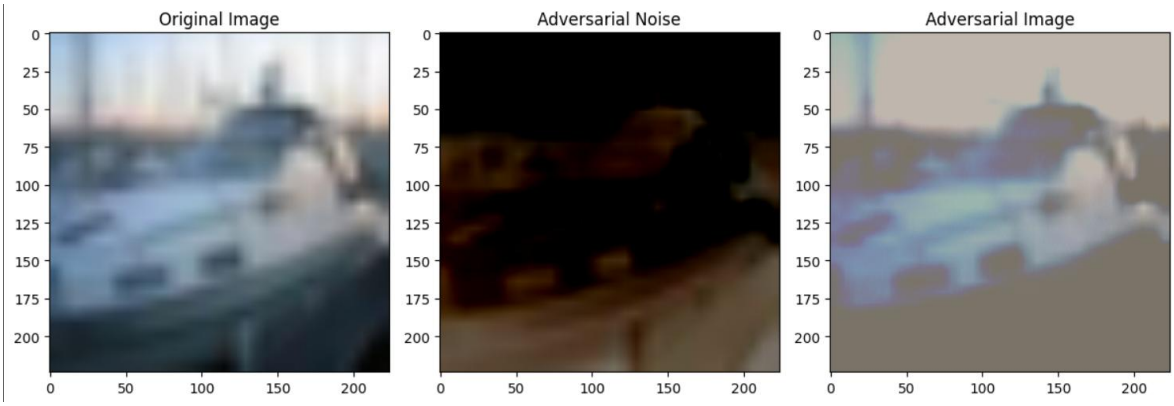
2.2.3- حمله **PGD** به مدل با **ResNet**-20

در این مرحله ابتدا یک تابع ساده نوشتیم که ورودی آن تصاویر آزمایشی است و خروجی آن دقت مدل CLIP را روی تصاویر تمیز و حمله شده را نشان میدهد گزارش می کند.(به عنوان پارامتر ورودی میگیرد که آیا به مدل حمله کند یا خیر.) این تابع به ازای هر تصویر، بردار ویژۀ CLIP را استخراج و آن را به صورت برداری نرمال شده درمی آورد؛ سپس با ضرب داخلی این بردار تصویر و مجموعه بردارهای متنی از پیش محاسبه شده برای هر کلاس، برچسب پیش بینی شده تعیین و دقت کلی محاسبه می شود.

برای سنجش مقاومت مدل در برابر حملات تقابلی انتقالی مدل 20-ResNet را با ε=8/255, α=2/255, steps=7 به عنوان مولد تصاویر خصمانه استفاده کردیم. پس از ساخت این نمونه های خصمانه، همان تابع ارزیابی روی آن ها اجرا شد تا دقت CLIP در مواجهه با حملات انتقالی مشخص شود.

در پایان، برای ملموس تر کردن اثر حمله، یک تصویر نمونه از مجموعۀ آزمایش انتخاب و سه نسخۀ آن—تصویر اصلی، نسخۀ خصمانه و نویز در زیر نشان داده شده است.

تصویر 2.6: تصویر اصلی، دچار حمله شده و نویزی که به تصویر اصلی اضافه شده است



همانطور که در تصویر مشخص است تصویر اصلی و تصویری که به آن نویز اضافه شده از نظر بینایی انسان خیلی دچار تفاوت فاحشی نشده است و تقریبا یکسان به نظر میرسد.

ولی دقت مدل پایه به شدت دچار افت شده است. به گونه ای که دقت مدل در زمانی که به تصاویر حمله انجام نشده است برابر با X است اما زمانی که به تصاویر حمله میشود دقت مدل به کاهش پیدا میکند. همانطور که در بخش اول تمرین ذکر شده است حال به دنبال پیاده سازی روش هایی هستیم که مدل را در برابر این نوع از حملات مقاوم تر کند.

تصویر 2.7- دقت مدل پایه بدون حمله به تصاویر (بالا) و با حمله به تصاویر (پایین)

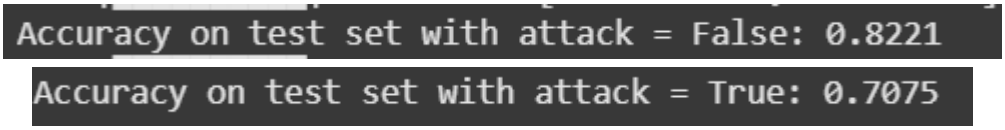
```
Accuracy on test set with attack = False: 0.8783
Accuracy on test set with attack = True: 0.5495
```

2.2.4- پیاده سازی روش **LoRA** و آموزش خصمانه مدل

در این بخش به سراغ پیاده سازی تنظیم دقیق مدل با استفاده از روش LoRA رفتیم. ابتدا config لازم را با استفاده از LoraConfig انجام دادیم و مشخص کردیم که تنها لایه های بخش بینایی مدل CLIP درگیر آموزش شوند، مثلاً با انتخاب r=8 و α=32. بعد از آن، مدل را در حالت آموزش قرار دادیم و برای هر بچ از تصاویر، ابتدا حمله PGD را اجرا کردیم تا نمونه های خصمانه تولید شود. سپس این تصاویر خصمانه را از CLIP عبور دادیم تا بردارهای تصویری به دست آید و با بردارهای متنی که از قبل آماده کرده بودیم ضرب داخلی انجام دادیم تا logits نهایی حاصل شود. با استفاده از CrossEntropyLoss و برچسب های اصلی، گرادیان ها را محاسبه کردیم و فقط پارامترهای مربوط به LoRA به روزرسانی شدند.

در طول این مرحله با چند مشکل مواجه شدیم. یکی از چالش ها این بود که استفاده از LoRA در مرحله آموزش ممکن بود با تداخل با تنظیمات حمله PGD لایه ها دچار اختلال شود، و گاهی تغییرات پارامترها تاثیر معکوس روی دقت داشت. همچنین، در ابتدا متوجه شدیم که مدل در برابر تصاویر خصمانه افت شدیدی در عملکرد دارد و حتی با چند تکرار آموزش، دقت روی داده های تمیز نیز کاهش پیدا می کرد که نشان می داد باید در انتخاب نرخ یادگیری و تعداد ایپاک ها دقت بیشتری کنیم. در نهایت پس از رفع این مسائل، توانستیم مدل را با LoRA به درستی آموزش داده و دقت آن را روی هر دو مجموعهٔ تمیز و خصمانه بسنجیم تا تأثیر آموزش مقاوم سازی به خوبی ارزیابی شود. یکی دیگر از مشکلات در استفاده از دیتالودر ها بود که باعث میشد به اروری برخورد کنیم و استفاده همزمان از دیتالودر ها و مدل به صورت ساده وجود نداشت. بنابراین با نوشتن یک تابع پیش از آموزش لایه های LoRA که باید آموزش میدیدند را Unfreeze و بقیه را freeze کردیم و تغییرات جزئی دیگری نیز بوجود آوردیم. در نهایت با پردازش روی 1000 تا از تصاویر داده های آموزش به دقت بسیار بهتری رسیدیم.

تصویر 2.8- دقت روش **LoRA** بدون حمله به تصاویر (بالا) و با حمله به تصاویر (پایین)



همانطور که در تصویر مشخص است دقت نسبت به مدل پایه افزایش قابل توجهی داشته است. و مقاومت بسیار بیشتری در برابر حملات PGD از خود نشان میدهد. همچنین دقت مدل در نمونه های تمیز اندکی کاهش داشته که علت آن اینست که با افزایش مقاومت در برابر حملات اندکی وزن ها به ضرر مسئله طبقه بندی تغییر کرده اند.

2.2.5- پیاده سازی الگوریتم **TeCoA** و آموزش مدل

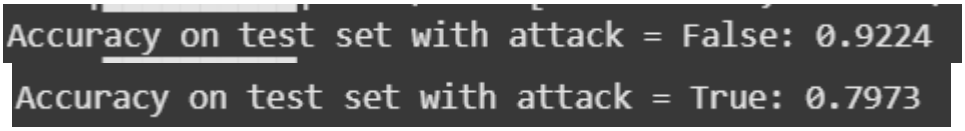
در این مرحله، سراغ پیاده سازی الگوریتم TeCoA رفتیم که نوعی آموزش خصمانه هدایت شده با متن است و هدف آن مقاوم سازی مدل CLIP در برابر حملات انتقالی است. مطابق با معادله 3 مقاله، تابع هزینه ای طراحی کردیم که هم تفاوت بین ویژگی های تصویر تمیز و خصمانه را در نظر می گیرد و هم شباهت بین آن ها و متن های مربوط به کلاس صحیح را به صورت متضاد تنظیم می کند.

برای پیاده سازی این روش، ابتدا تابع حمله PGD را نوشتیم که تصاویر خصمانه را تولید می کرد، سپس در هر تکرار آموزش، برای هر تصویر تمیز و نسخه خصمانه اش بردار ویژگی استخراج می کردیم و با استفاده از این ویژگی ها و بردارهای متنی از قبل آماده شده، تابع loss متناظر با TeCoA را محاسبه می کردیم. مدل را به حالت train بردیم و مانند مرحله قبل تنها پارامترهای را به روزرسانی کردیم.

در طول این مرحله متوجه شدیم که برخلاف آموزش خصمانه استاندارد، تنظیمات الگوریتم TeCoA حساسیت بیشتری نسبت به وزن دهی مناسب بین بخش های مختلف تابع هزینه دارد، و همین موضوع باعث شد در بعضی تکرارها دقت مدل به طور قابل توجهی افت کند. همچنین در ابتدا تغییرات loss تأثیر محسوسی بر دقت مدل نداشت، که نشان می داد باید معیارهای محاسبه loss و نرمال سازی دقیق تر تنظیم شوند. پس از انجام چند اصلاح و بررسی دقیق تر، توانستیم مدل را آموزش داده و ارزیابی دقیقی از عملکرد آن در برابر نمونه های خصمانه به دست آوریم.

دقت این مدل به صورت زیر است:

تصویر 2.9- دقت الگوریتم **TeCoA** بدون حمله به تصاویر (بالا) و با حمله به تصاویر (پایین)



همانطور که در تصویر نیز مشخص است دقت مدل بر روی تصاویر تمیز نیز افزایش داشته است و همچنین در برابر حملات PGD نیز به خوبی ظاهر شده است و دقت آن تنها اندکی کمتر است. بنابراین بیشترین مقاومت را دارا میباشد.

2.2.6- مقایسه دقت ها

Accuracy on Base, LoRA and TeCoA Methods			
clean or adversarial \ Method	Base Model	LoRA	TeCoA
Accuracy on clean Images	87.83	82.21	92.24
Accuracy on adversarial Images	54.95	70.75	79.73

در جدول زیر مقایسه دقت ها در سه روش آورده شده است:

2.3- نتیجه گیری

در پایان مراحل پیاده سازی و ارزیابی مدل ها، نتایج نهایی نشان دادند که آموزش های خصمانه مختلف می توانند تاثیر قابل توجهی روی مقاومت مدل CLIP در برابر حملات خصمانه داشته باشند. مدل پایه (Base Model) که هیچ گونه آموزش خصمانه ای ندیده بود، در مواجهه با تصاویر تمیز دقت خوبی داشت و به حدود 87.83 درصد می رسید، اما به محض قرار گرفتن در برابر تصاویر خصمانه، دقت آن به شدت افت کرده و به حدود 54.95 درصد کاهش یافت. این نتیجه نشان داد که مدل CLIP با وجود عملکرد قوی در شرایط معمول، نسبت به حملات انتقالی بسیار آسیب پذیر است.

با استفاده از روش آموزش خصمانه مبتنی بر LoRA، توانستیم دقت مدل روی تصاویر خصمانه را تا حد قابل قبولی افزایش دهیم. در این حالت، دقت روی تصاویر تمیز کمی کاهش یافت و به 82.21 درصد رسید، اما دقت روی تصاویر خصمانه به 70.75 درصد افزایش یافت که پیشرفت قابل توجهی نسبت به مدل پایه بود. این نتایج تایید کردند که آموزش خصمانه استاندارد با LoRA می تواند به خوبی نقش دفاعی ایفا کند و مدل را نسبت به نمونه های دستکاری شده مقاوم تر کند.

در نهایت، الگوریتم TeCoA را پیاده سازی کردیم که برخلاف روش قبلی، نه تنها روی تصاویر خصمانه تمرکز داشت، بلکه ویژگی های بردارهای تصویر را نسبت به متن های مرتبط نیز به صورت تضاد هدایت می کرد. این روش باعث شد که دقت مدل روی تصاویر تمیز تا حدود 92.24 درصد افزایش یابد و در عین حال، دقت آن در مواجهه با تصاویر خصمانه نیز به 79.73 درصد برسد. این دو عدد، بهترین نتایج را در بین همه مدل ها نشان دادند. در واقع، TeCoA توانست بدون قربانی کردن دقت روی داده های تمیز، مقاومت مدل را نسبت به حملات نیز به شکل چشمگیری بالا ببرد.

به طور کلی، مسیر انجام این تمرین نشان داد که آموزش خصمانه اگر به درستی انجام شود، می تواند بهبود چشم گیری در پایداری و دقت مدل به همراه داشته باشد، اما اجرای صحیح آن نیاز به تنظیم دقیق پارامترها، طراحی درست تابع هزینه، و در نظر گرفتن تعادل بین حفظ دقت تمیز و افزایش مقاومت خصمانه دارد.