

ادیب رضائی - امیرمحمد خسروی	نام و نام خانوادگی
810198386 - 810198401	شماره دانشجویی
۱۴۰۱.۱۲.۲۲	تاریخ ارسال گزارش

به نام خدا  
 دانشگاه تهران  
 دانشکده مهندسی  
 برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق  
 تمرین سوم

## **فهرست**

4	پاسخ 1. آشنایی با یادگیری انتقالی
4	1- گزارش مقاله
7	2- معماری شبکه، مزایا و معایب و پیش‌پردازش
10	3- قابلیت تشخیص شبکه
12	4- دریافت دادگان و بررسی
13	5- پیاده‌سازی شبکه و گزارش نتایج
17	پاسخ 2. تشخیص و شمارش اشیا
19	1- پیش‌پردازش داده‌ها
21	2- ایجاد و آموزش مدل
23	3- ارزیابی مدل و شمارش اشیا

## شکل‌ها

شكل 1.1. Inception block با کاهش ابعاد

شكل 1.2. معماری مدل DDIRNet

شكل 1.3. خلاصه نتایج

شكل 1.4. معماری مدل

شكل 1.5. کد پیش‌پردازش

شكل 1.6. کد one-hot کردن لیبل‌ها

شكل 1.7. یادگیری انتقالی

شكل 1.8. فرمت اطلاعات

شكل 1.9. نمونه تصاویر

شكل 1.10. نمودار training accuracy

شكل 1.11. نمودار validation accuracy

شكل 1.12. نمودار training loss

شكل 1.13. نمودار validation loss

شكل 1.14. مقادیر دقت و خطای

شكل 1.15. مقادیر دقت

شكل 1.16. نمودارهای loss و accuracy

شكل 1.17. مقادیر precision، recall، f1-score

شكل 1.18. ماتریس طبقه‌بندی

شكل 2.1. معماری شبکه Faster RCNN

شكل 2.2. بارگذاری دیتاست‌ها در مموری

شكل 2.3. تصویر هواییما با یک bounding box و یک legend

شكل 2.4. ایجاد Data Loader‌ها

شكل 2.5. بارگذاری مدل

شكل 2.6. تعریف optimizer و scheduler

شكل 2.7. آموزش مدل

شکل 2.8. اطلاعات مدل در آخرین ایپاک

شکل 2.9. تصویر هواییما در دو نتیجه expected و predicted. در حالت خروجی مدل ما دو threshold هواییما در پشت تشخیص داده نشده. این ممکن است بخاطر fine tune نشدن باشد یا قرار دادن مقدار بالای آن باشد.

شکل 2.10. تشخیص هواییما در دو حالت درست تشخیص داده شده است.

شکل 2.11. در این تصویر کمی پیچیده تر دوچرخه و انسان مطابق آنچه انتظار میرفت تشخیص داده شد.

## جدول‌ها

## پاسخ 1. آشنایی با یادگیری انتقالی

رقم آخر شماره دانشجویی نفر اول = 6

رقم آخر شماره دانشجویی نفر دوم = 1

$6 + 1 \text{ (mode 4)} = 3$

### Inception (Three-class brain tumor classification using deep dense inception residual network)

#### 1-1. گزارش مقاله

در این مقاله، یک شبکه عصبی جدید برای تشخیص تومور مغزی با دقت بالا پیشنهاد شده است. این شبکه با استفاده از deep dense inception residual network ساخته شده و نتایج نشان می‌دهند که مدل پیشنهادی با دقت بالای 99.69٪، از مدل‌های موجود بهبود یافته است.

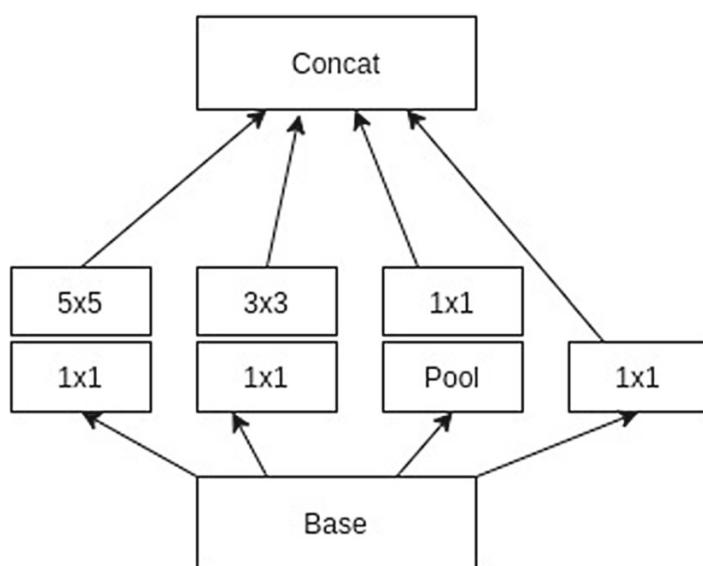
پردازش تصویر پزشکی به طیف وسیعی از کاربردها ارتقا یافته است. در این زمینه، محققان مختلف روش‌های پاکسازی تصویر مغز و تصاویر پزشکی را ارائه کرده‌اند و از شبکه‌های عصبی عمیق، برای دسته‌بندی تصاویر استفاده شده است. تصاویر مغناطیسی (MR) و توموگرافی محاسبه‌ای (CT) از تکنیک‌های تصویربرداری رایج برای تشخیص تومور مغزی هستند. تصاویر MR تصاویر دقیقتری از مغز را نسبت به تصاویر CT فراهم می‌کنند. در حال حاضر، موارد تومور مغزی با سرعتی بالا در حال افزایش هستند. بررسی دستی تشخیص تومورهایی با این مقیاس از تصاویر MR فرآیندی زمان بر است. به همین دلیل، پژوهشگران به منظور خودکارسازی تشخیص تومور مغزی از کامپیوترا و پردازش تصویر کمک گرفته و زمان تشخیص را کاهش داده‌اند و این باعث افزایش تاثیر درمان و همچنین افزایش بقا بیمار می‌شود. در این مقاله مسئله دسته‌بندی چند کلاسه مطرح است و تومورهای مغزی به سه دسته Pituitary و Meningioma، Glioma تقسیم می‌شوند. در سال 2016، Cheng و همکارانش یک چارچوب بازیابی تصویر مبتنی بر محظوا برای دسته‌بندی سه‌کلاسی تومور مغز ارائه دادند. نویسنده‌گان یک مجموعه داده تومور مغز با 3064 تصویر MR تیپ 1 منتشر کرده‌اند. در اینجا از این مجموعه داده استفاده می‌شود. سپس مقاله به توضیح جزئیات مدل شبکه عصبی می‌پردازد و مقالات و مدل‌های مربوط به Deepak و Ameer هستند که از Deep Transfer Learning و ماشین بردار پشتیبان (SVM) برای دسته‌بندی سه کلاسه استفاده کرده‌اند و با استفاده از GoogleNet از قبل آموزش داده شده، ویژگی‌ها را از تصاویر MR مغز استخراج کرده و دقت 97٪ را به دست آورده‌اند. در طبقه بندی

تصاویر تومور مغز، دستیابی به دقت طبقه بندی بهتر، چالش برانگیز است. در این مقاله تلاش شده است یک شبکه ژرف و جدید با الهام از شبکه های Dense Inception برای طبقه بندی سه کلاسه طراحی شود که کارایی بهتری داشته باشد. مدل پیشنهادی با استفاده از سه لایه Dense برای دستیابی به عملکرد بالا استفاده می کند. علاوه بر این، تکنیک Dropout برای جلوگیری از overfitting مدل استفاده شده است.

در حوزه ی تصاویر تومور مغزی، شبکه های عمیق Inception و VGGNet از جمله مدل های پرطرفدار هستند. اما از مشکلات این شبکه ها، از دست رفتن گرادیان ها و هزینه های محاسباتی بالا است. شبکه VGGNet به دلیل تعداد بالای پارامترهای قابل آموزش هزینه های محاسباتی زیادی را به دنبال دارد. شبکه Inception با استفاده از کانولوشن های بخش بندی شده، زمان محاسباتی را کاهش می دهد. همچنین شبکه ResNet با استفاده از اتصالات skip به مشکلات از دست رفتن گرادیان ها پاسخ داده است. در این مطالعه نیز از Inception ResNet v2 برای دسته بندی تصاویر سه کلاسه تومور مغزی استفاده شده است.

در شبکه های عصبی کانولوشنی، عمل کانولوشن عملیات اصلی است. به طور کلی، عملیات کانولوشن با فیلترهای مکانی بزرگتر، هزینه محاسباتی بالایی را داراست. مازول Inception یک راه حل مهم برای کاهش این هزینه است. این مازول با استفاده از ساختارهای محلی و پراکنده بهینه، هزینه محاسباتی را کاهش می دهد. Inception block در واقع یک ساختار لایه به لایه با استفاده از آنالیز آماری همبستگی لایه ها طراحی شده است. Inception block دو مرحله برای تولید بانک های فیلتر انجام می دهد که عبارتند از:

۱. پردازش اطلاعات بصری در مقیاس های مختلف از همان منطقه.
۲. جمع آوری ویژگی های انتزاعی از مقیاس های مختلف به صورت همزمان.



شکل 1.1. Inception block با کاهش ابعاد

حال از این بلوک‌ها استفاده می‌شود تا شبکه Inception Residual ساخته شود. شبکه Inception ResNet block شامل سه نوع بلوک است ( Inception Residual و stem block، Inception ResNet block ) که از اتصالات residual برای بلوک‌های Inception استفاده می‌کند. این شبکه با دو بلوک Reduction، پنج عدد از بلوک‌های Inception ResNet-A، ده عدد از بلوک‌های Reduction-A، یک بلوک Inception ResNet-C، پنج عدد از بلوک‌های Inception ResNet-B و یک بلوک Reduction-B به عنوان شبکه‌ی اصلی ساخته شده است. استفاده از اتصالات residual و همچنین ترکیب بلوک‌های Inception با این متداول‌لوژی سبب بهبود قابل توجهی در عملکرد شبکه می‌شود.

در ادامه مقاله جزیيات بخش دیگر شبکه را توضیح می‌دهد و مراحل پیاده‌سازی مدل پیشنهادی در انتصار (DDIRNet) را بیان می‌کند. در این مدل، شبکه Inception ResNet v2 بدون لایه خروجی و با ورودی تصویر با اندازه (256، 256، 3) استفاده شده است. سپس شبکه‌ی deep dense شامل سه لایه دنس با هر لایه دنس 2048 نورون و یک لایه با 1024 نورون و هر لایه با یک leaky ReLU و یک لایه dropout برای جلوگیری از اورفیت استفاده شده است. مراحل پیاده‌سازی به شرح زیر است:

1. شبکه Inception ResNet v2 بدون لایه خروجی با تصویر ورودی با اندازه (3، 256، 256، 3) استفاده شده است که قبل از لایه خروجی، feature map با اندازه (6، 6، 1536) را تولید می‌کند.

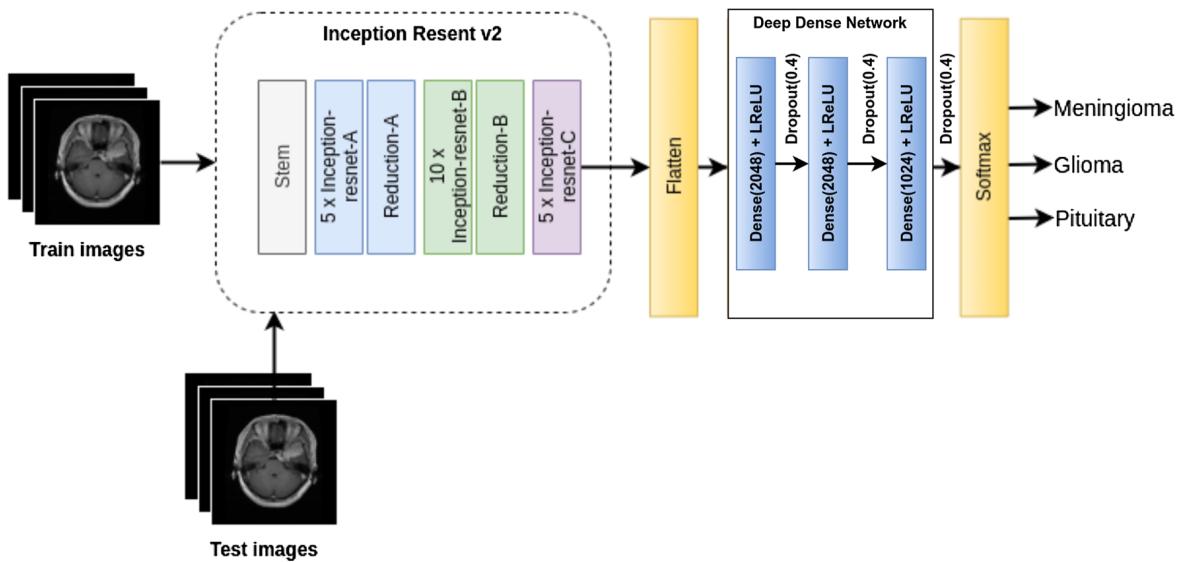
2. feature map چندبعدی فوق به یک بردار یک بعدی با 55296 ویژگی تبدیل می‌شود.

3. شبکه deep dense بردار یک بعدی فوق را به عنوان ورودی دریافت کرده و یک بردار یک بعدی با 1024 ویژگی تولید می‌کند.

4. در نهایت، یک لایه softmax با سه نورون برای دسته‌بندی سه کلاسه اضافه می‌شود.

در مدل پیشنهادی، که با نام DDIRNet (deep dense inception ResNet) شناخته می‌شود، از شبکه به طور قابل توجهی عملکرد دسته‌بندی را بهبود می‌بخشد. این مدل با وزن‌های Imagenet مقداردهی اولیه شده است و یادگیری پارامترها را برای کاهش زمان آموزش غیرفعال می‌شود. با این حال، در این مدل، یادگیری پارامترهای سه لایه dense مجاز است. در کل، یک واحد خطی با اعمال محدودیت (ReLU) به‌طور معمول به عنوان فعال‌سازی لایه‌های dense استفاده می‌شود. مقدار منفی فعال‌سازی ReLU به صفر می‌رسد. بنابراین، ما به جای ReLU عادی، leaky ReLU را با هر لایه dense ادغام کرده‌ایم.  $\alpha = 0.2$  اجازه یک مقدار منفی کوچک را با به صورت کنترل شده می‌دهد. برای کنترل overfitting مدل، تکنیک dropout برای حذف تصادفی نورون‌ها در نظر گرفته شده است. بنابراین، پس از هر لایه چگال، مقدار dropout 0.2 در نظر گرفته شده است.

نهایتاً شبکه به صورت زیر خواهد بود:



شکل 1.2. معماری مدل DDIRNet

در ادامه مقاله آنالیزهای مربوط به عملکرد را و همچنین برخی مقادیر و معیارهای استفاده شده را ذکر می‌کند. برای آموزش مدل از پایتون و کتابخانه Tensorflow استفاده شده است. Adam یکی از بهینه‌سازهای کارآمد برای شبکه‌های عمیق است. بنابراین، برای ترکیب مدل پیشنهادی از بهینه‌ساز Adam با نرخ یادگیری اولیه 0.0001 استفاده شده است. همچنین مقادیر size و تعداد epochs به ترتیب برابر با 50 و 5 در نظر گرفته شده است. نویسنگان مقاله از پردازنده Intel Xeon با 32 گیگابایت حافظه استفاده کرده‌اند. علاوه بر این، از cross-validation fivefold برای تحلیل مدل پیشنهادی استفاده شده است. کل مجموعه داده به دو بخش تقسیم شده است: مجموعه train و مجموعه test. در هر برابر، ۷۵٪ از مجموعه داده به عنوان مجموعه validation استفاده می‌شود و ۲۵٪ باقی مانده برای آزمایش مدل استفاده می‌شود. عملکرد مدل پیشنهادی بر روی مجموعه آزمون محاسبه شده و پس از fivefold، میانگین عملکرد محاسبه می‌شود. مدل پیشنهادی پارامترهای پایه IRNet را غیرفعال کرده است و به همین دلیل در epoch‌های اولیه آموزش و اعتبارسنجی خطا بالایی دارد. مدل پیشنهادی به دلیل سه لایه dense و استفاده ازتابع خطا cross-entropy یادگیری سریعی دارد. پارامترهای این لایه‌های dense در هر epoch آموزش یادگیری شده و ویژگی‌های تومور مغزی را تطبیق می‌دهند. این باعث کاهش مداوم خطای آموزش و اعتبارسنجی مدل پیشنهادی می‌شود. مدل پیشنهادی در پنج دور آموزشی همگرا می‌شود و در پایان دور آموزشی پنجم به خطای بهینه ۰.۵٪ می‌رسد.

در نهایت نیز مقاله عملکرد مدل DDIRNet را با معیارهایی مانند Percision، F1-score و Accuracy بررسی می‌کند. به طور خلاصه نتایج در شکل‌های زیر آورده شده‌اند.

**Table 3** Performance of proposed model with different training sizes

Training size (%)	Accuracy	Precision	Recall	F1-score
50	99.69	99.60	99.47	99.47
60	99.62	99.53	99.20	99.40
75	99.66	99.60	99.40	99.40

**Table 4** Confusion matrix of proposed model in fold-5

Actual	Predicted			Acc. %
	Meningioma	Glioma	Pituitary	
Meningioma	409	0	0	99.74
Glioma	0	120	2	
Pituitary	0	0	235	

**Table 6** Performance of proposed model on noisy data

Fold	Accuracy	
	9 × 9 kernel	5 × 5 kernel
1	99.59	99.43
2	99.76	99.59
3	99.76	99.76
4	99.51	99.10
5	99.18	99.51
Mean	99.56	99.48

**Table 7** Comparison with pre-trained models

Model	Accuracy
MobileNet	97.19
ResNet50	93.60
EfficientNetb0	97.32
GoogleNet	97.91
Inception ResNet	97.03
DDIRNet	99.69

شكل 1.3. خلاصه نتایج

همانطور که در شکل‌ها مشخص است، مدل در همه معیارها عملکرد حداقل ۹۹ درصد داشته است. همچنین نشان می‌دهد مدل با تعداد تصاویر آموزشی متفاوت نتیجه مشابه داشته است.

سپس به این نکته اشاره می‌کند که یادگیری انتقالی یا Transfer Learning به عنوان یکی از روش‌های موثر برای وظایف بینایی ماشین، به خصوص دسته بندی تصاویر، شناخته شده است. در این روش، مدل‌های پیش‌آموزش دیده دوباره با داده‌های خاص دامنه آموزش داده می‌شوند تا عملکرد آنها بهبود یابد. چهار مدل پیش‌آموزش دیده معروف به، MobileNet، Res Net50، Inception ResNet و GoogleNet و EfficientNet برای ارزیابی در نظر گرفته می‌شوند.

سپس مقاله بحث می‌کند که عملکرد مدل ارائه شده در مقابل موارد دیگر چگونه است. با توجه به مطالب ذکر شده، این مدل ۲ درصد عملکرد بهتری از بهترین مدل قبل خود یعنی مدل Deepak and Ameer دارد.

در نهایت امر مقاله به عنوان نتیجه گیری می‌گوید که شبکه‌های عصبی عمیق به عنوان راه حلی پرطرفدار برای دسته‌بندی تصاویر شناخته شده‌اند. نویسنده‌ان از IRNet v2 استفاده کرده‌اند و یک deep dense inception residual network را برای دسته‌بندی سه کلاس تومور مغزی پیشنهاد شده است. این مدل پیشنهادی شامل سه لایه dense به همراه یک لایه softmax به عنوان لایه خروجی است. این لایه‌های dense تومورهای مغزی را در خود جای داده‌اند در حالی که پارامترهای IRNet با وزن‌های Imagenet مقداردهی اولیه شده‌اند. مدل پیشنهادی به دلیل استفاده از بهینه‌ساز Adam و dropout، با مشکلات overfitting مدل مواجه نمی‌شود و به صورت سریع یادگیری می‌کند. از مجموعه‌داده شامل ۳۰۶۴ تصویر با وزن T1 برای ارزیابی عملکرد استفاده شده است. مدل پیشنهادی با دقت ۹۹.۶۹٪ نسبت به مدل‌های موجود بهترین عملکرد را ارائه می‌دهد. همچنین، عملکرد مدل پیشنهادی بر روی داده‌های تصویر نویزی نیز مشابه است. کار آینده نویسنده‌ان مقاله بر روی کاوش تعداد پارامترها و زمان محاسباتی برای مدل پیشنهادی، بدون کاوش عملکرد، تمرکز خواهد داشت.

## ۱-۲. معماری شبکه، مزایا و معایب و پیش پردازش

### ۱. توضیح معماری

معماری دقیقاً همان معماری ذکر شده در مقاله است. به این صورت که:

شبکه Inception ResNet شامل سه نوع بلوک است ( Inception Residual stem block، Inception ResNet ) که از اتصالات residual برای بلوک‌های Reduction block استفاده می‌کند. این شبکه با دو بلوک Reduction، پنج عدد از بلوک‌های Inception ResNet-A، ده عدد از بلوک‌های Inception ResNet-B، پنج عدد از بلوک‌های Inception ResNet-C، یک بلوک Reduction-B و یک بلوک Reduction-A به عنوان شبکه‌ی اصلی ساخته شده است.

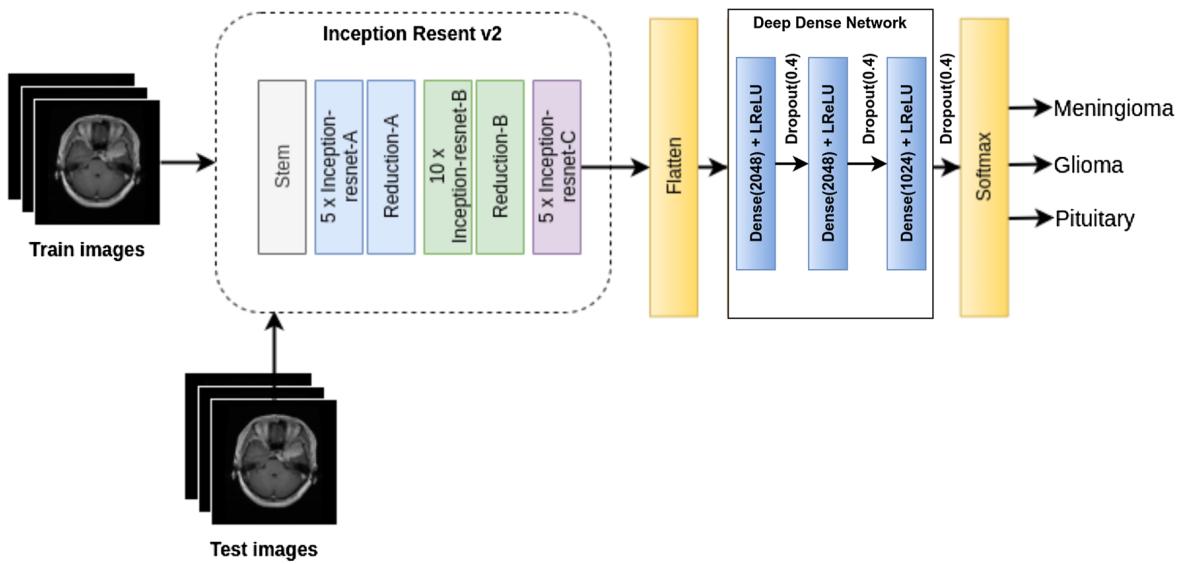
در ادامه جزئیات بخش دیگر شبکه را توضیح داده شده است و مراحل پیاده‌سازی مدل پیشنهادی Inception ResNet v2 یا به اختصار (DDIRNet) را بیان می‌شود. در این مدل، شبکه deep dense inception ResNet v2 بدون لایه خروجی و با ورودی تصویر با اندازه (3، 256، 256، 3) استفاده شده است. سپس شبکه‌ی deep dense شامل سه لایه دنس با هر لایه شامل 2048 نورون و یک لایه با 1024 نورون و هر لایه با یک لایه leaky ReLU و یک لایه dropout برای جلوگیری از اورفیت استفاده شده است. مراحل پیاده‌سازی به شرح زیر است:

1. شبکه Inception ResNet v2 بدون لایه خروجی با تصویر ورودی با اندازه (3، 256، 256، 3) استفاده شده است که قبل از لایه خروجی، feature map با اندازه (6، 6، 1536) را تولید می‌کند.
2. چندبعدی فوق به یک بردار یک بعدی با 55296 ویژگی تبدیل می‌شود.
3. شبکه deep dense بردار یک بعدی فوق را به عنوان ورودی دریافت کرده و یک بردار یک بعدی با 1024 ویژگی تولید می‌کند.
4. در نهایت، یک لایه softmax با سه نورون برای دسته‌بندی سه کلاسه اضافه می‌شود.

یادگیری پارامترها را برای کاهش زمان آموزش غیرفعال می‌شود. با این حال، در این مدل، یادگیری پارامترهای سه لایه dense مجاز است. یک واحد leaky ReLU را با هر لایه dense ادغام کرده‌ایم. leaky ReLU اجازه یک مقدار منفی کوچک را با  $\alpha = 0.2$  به صورت کنترل شده می‌دهد. برای کنترل overfitting مدل، تکنیک dropout برای حذف تصادفی نورون‌ها در نظر گرفته شده است. بنابراین، پس از هر لایه چگال، مقدار 0.2 dropout در نظر گرفته شده است.

از Adam که یکی از بهینه‌سازهای کارآمد برای شبکه‌های عمیق است استفاده شده است و نرخ یادگیری اولیه 0.0001 است. همچنین مقادیر epochs و تعداد batch size به ترتیب برابر با 50 و 5 در نظر گرفته شده است. از cross-validation fivefold برای تحلیل مدل پیشنهادی استفاده شده است. کل مجموعه داده به دو بخش تقسیم شده است: مجموعه train و test. در هر برابر، 75٪ از مجموعه داده به عنوان مجموعه validation استفاده می‌شود و 25٪ باقی مانده برای آزمایش مدل استفاده می‌شود. عملکرد مدل پیشنهادی بر روی مجموعه آزمون محاسبه شده و پس از fivefold، میانگین عملکرد محاسبه می‌شود. مدل پیشنهادی پارامترهای پایه IRNet را غیرفعال کرده است و به همین دلیل در epoch اولیه آموزش و اعتبارسنجی خطابالایی دارد. مدل پیشنهادی به دلیل سه لایه dense و استفاده ازتابع خطا cross-entropy یادگیری سریعی دارد. پارامترهای این لایه‌های dense در هر epoch آموزش یادگیری شده و ویژگی‌های تومور مغزی را تطبیق می‌دهند.

نهایتاً مدل به صورت زیر خواهد بود:



شکل 1.4. معماری مدل

## 2. مزایا و معایب

### مزایای مدل عبارتند از:

1- دقت بالا: مدل با دقت بالا بر روی مجموعه داده‌های تصویری تومور مغزی سه کلاسه عملکرد بسیار خوبی دارد و مدل معادل آن در مقاله، نسبت به مدل‌های قبلی بهترین عملکرد را داشته است.

2- سرعت آموزش: مدل به دلیل استفاده از بهینه‌ساز Adam و استفاده از dropout برای جلوگیری از Overfitting، سرعت آموزش را افزایش داده است.

3- انعطاف پذیری: مدل قابلیت استفاده در داده‌های نویزی را نیز دارد.

4- به دلیل عمق بالایی که دارد می‌تواند به عنوان یک استخراج کننده ویژگی برای طبقه‌بندی با دقت بالا عمل کند.

### معایب مدل پیشنهادی در این مقاله عبارتند از:

1- تعداد پارامترهای بالا: مدل با استفاده از سه لایه dense، تعداد پارامترهای بالایی دارد که باعث افزایش زمان آموزش و حافظه مصرفی می‌شود.

2- اعتماد به پیشآموزش: مدل برای آموزش از پیشآموزش با وزن‌های Imagenet استفاده می‌کند و به همین دلیل، برای داده‌هایی که با موضوع متفاوتی ساخته شده‌اند، نیاز به پیشآموزش مجدد دارد.

### 3. پیشپردازش‌های انجام شده

داده‌ها و تصاویر به فرمت mat هستند که با استفاده از کتابخانه h5py آن‌ها را می‌خوانیم و در متغیری ذخیره می‌کنیم. سپس مقادیر مربوط به فیلدهای image و label را از این متغیر استخراج کرده و در لیست مربوط به تصاویر و لیل‌ها ذخیره می‌کنیم. سپس چون سایز عکس‌ها 512 \* 512 است اما مدل ما عکس با اندازه 256 \* 256 ورودی می‌گیرد، آن‌ها را با تابع transform.resize به سایز مورد نظر می‌رسانیم. همچنین به این دلیل که ورودی مدل ما عکس‌هایی با سه کanal است، یعنی عکس‌هایی با سایزهای 256 \* 256 \* 3، و عکس‌های دیباتست تنها یک کanal دارند، با تابع np.stack دو کanal مشابه کanal اصلی برای عکس‌ها ایجاد می‌کنیم.

```
for index, img_file in enumerate(img_files):
    img_path = os.path.join(folder_path, img_file)
    mat_file = h5py.File(img_path, 'r')
    image_arr = transform.resize(mat_file['cjdata']['image'][()], (256, 256))
    image_arr = np.stack((image_arr,)*3, axis=-1)
    x_data[index] = image_arr
    y_data[index] = mat_file['cjdata']['label'][()][0][0]
    del image_arr
    mat_file.close()
```

شکل 1.5. کد پیش‌پردازش

سپس چون سه لیل داریم، لیل‌ها را به صورت one-hot در می‌آوریم:

```
onehotencoder = OneHotEncoder()
y_data = onehotencoder.fit_transform(y_data).toarray()
print(y_data.shape,x_data.shape)
print(y_data[0])
```

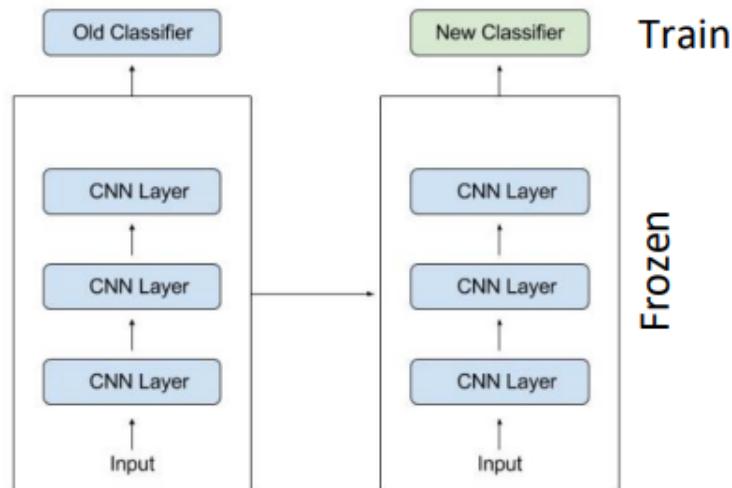
شکل 1.6. کد one-hot کردن لیل‌ها

### 3-3. قابلیت تشخیص شبکه

این شبکه روی مجموعه داده آموزش داده شده است که دارای تصاویری از 1000 کلاس مختلف می‌باشد. این مدل با 144 میلیون پارامتر دارای دقت top-1 حدود 5.74% و دقت top-5 حدود 91% می‌باشد. از جمله کالس‌های تصاویر موجود در این مجموعه داده می‌توان موارد زیر را نام برد:

- Pizza, pizza pie
- Strawberry
- Mushroom
- Jellyfish
- Persian Cat
- Guitar
- ...

حال اگر عکسی داخل دسته های موجود در ImageNet نباشد یعنی طبیعتاً این مدل قادر به تشخیص آن نخواهد بود. در نتیجه باید از روش های جایگزین مانند Learning Transfer استفاده کنیم. یادگیری انتقالی (Learning Transfer) به معنای استفاده از یک مدل از پیش آموزش دیده در یک کاربرد جدید است. این مبحث، امروزه در یادگیری عمیق بسیار مورد توجه است، زیرا امکان آموزش شبکه های عصبی عمیق را با داده های نسبتاً کمی فراهم می کند. هدف از یادگیری انتقالی در واقع این است که از دانشی که در یک مسئله به دست آمده (یعنی توانایی استخراج ویژگی آن) برای بهبود تعمیم پذیری در مسئله ای دیگر استفاده شود. یعنی به جای شروع پروسه آموزش از صفر، از الگوهای بدست آمده در مسئله ای مشابه استفاده شود.



شکل 1.7. یادگیری انتقالی

## 4-1. دریافت دادگان و بررسی

همانطور که در شکل زیر نشان داده شده است، بعضی از ویژگی‌های داده‌ها را چاپ می‌کنیم.  
داده‌ها و فرمت و سایر اطلاعات به شرح زیر است:

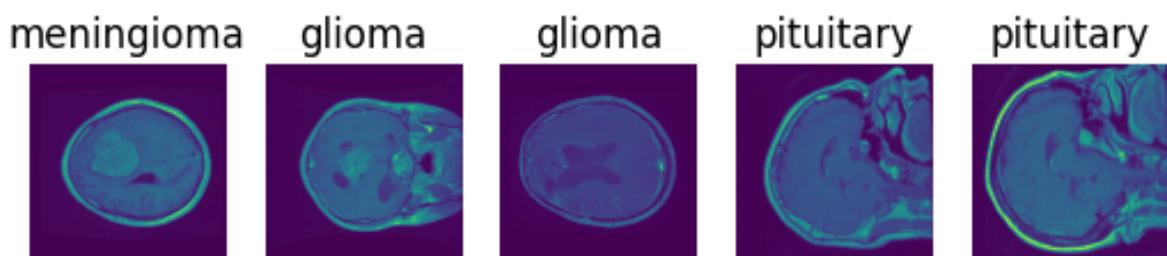
```
print(type(mat_file))
print(mat_file.keys())
print(mat_file['cjdata'].keys())
print(mat_file['cjdata']['PID'][()])
print(mat_file['cjdata']['image'][()])
print(mat_file['cjdata']['label'][()])
print(type(mat_file['cjdata']['image'][()]))
print(mat_file['cjdata']['image'][()].shape)

show_files_details()

<class 'h5py._hl.files.File'>
<KeysViewHDF5 ['cjdata']>
<KeysViewHDF5 ['PID', 'image', 'label', 'tumorBorder', 'tumorMask']>
[[49]
 [48]
 [57]
 [52]
 [55]
 [49]]
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [2 3 8 ... 7 4 4]
 [2 2 7 ... 6 5 5]
 [3 2 7 ... 6 5 6]]
[[3.]]
<class 'numpy.ndarray'>
(512, 512)
```

شکل 1.8. فرمت اطلاعات

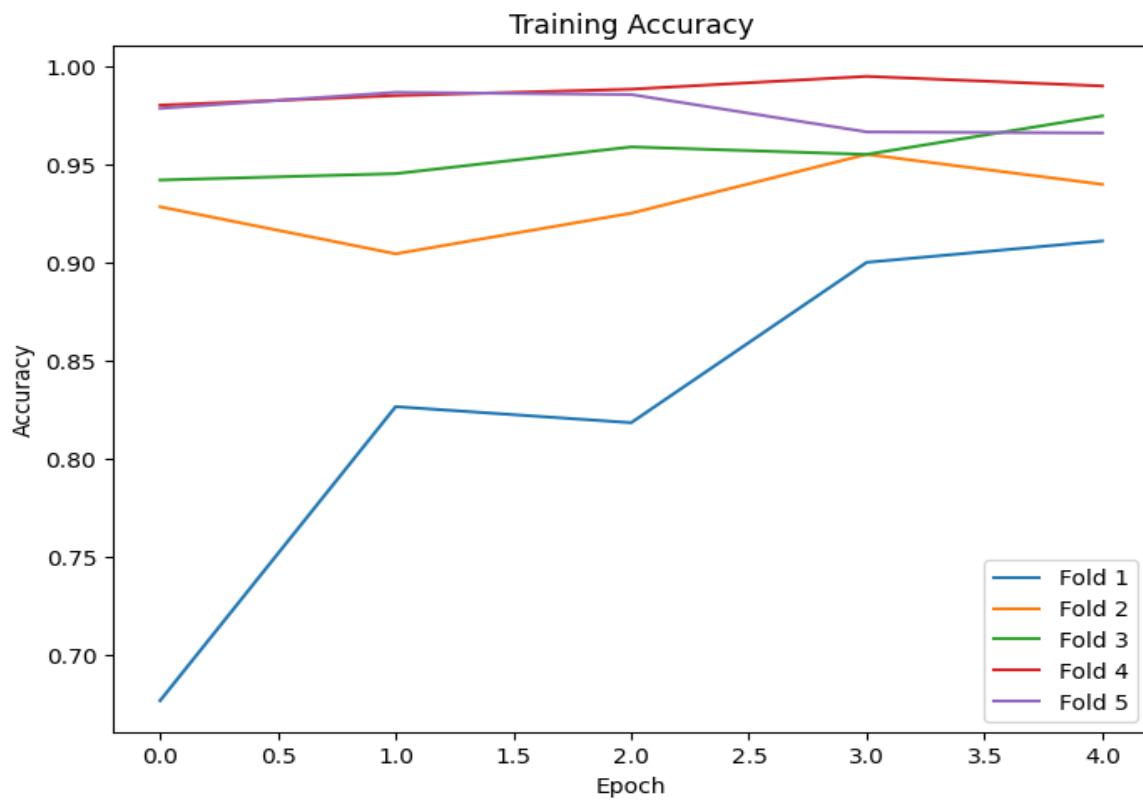
همچنین برخی از تصاویر را با لیبل‌های آن‌ها چاپ می‌کنیم تا ببینیم داده‌ها به چه صورت هستند:



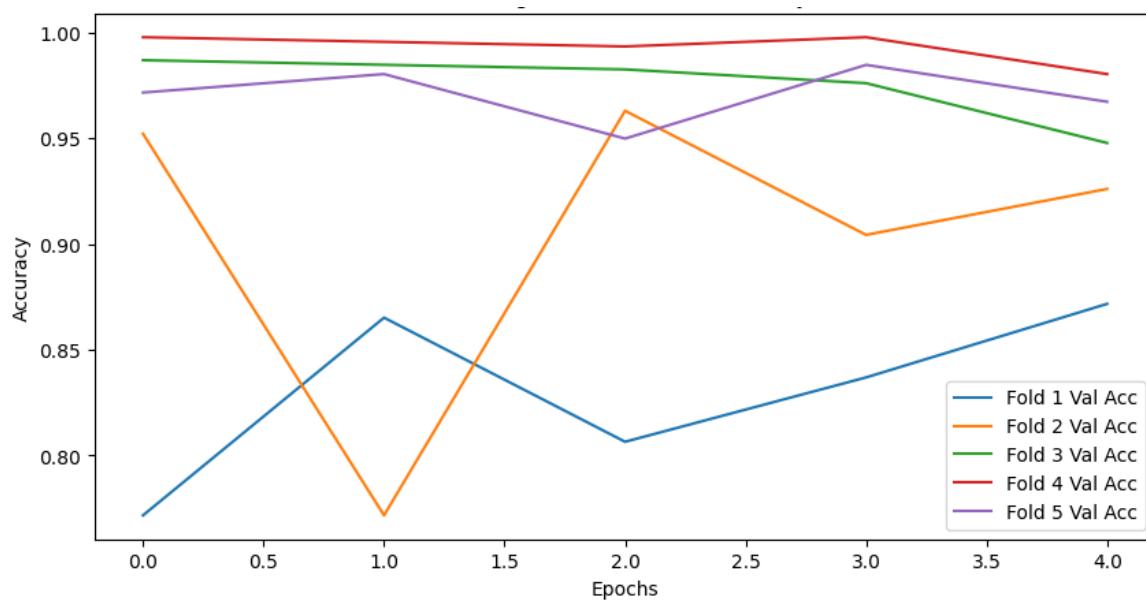
شکل 1.9. نمونه تصاویر

## 5-1. پیاده‌سازی شبکه و گزارش نتایج

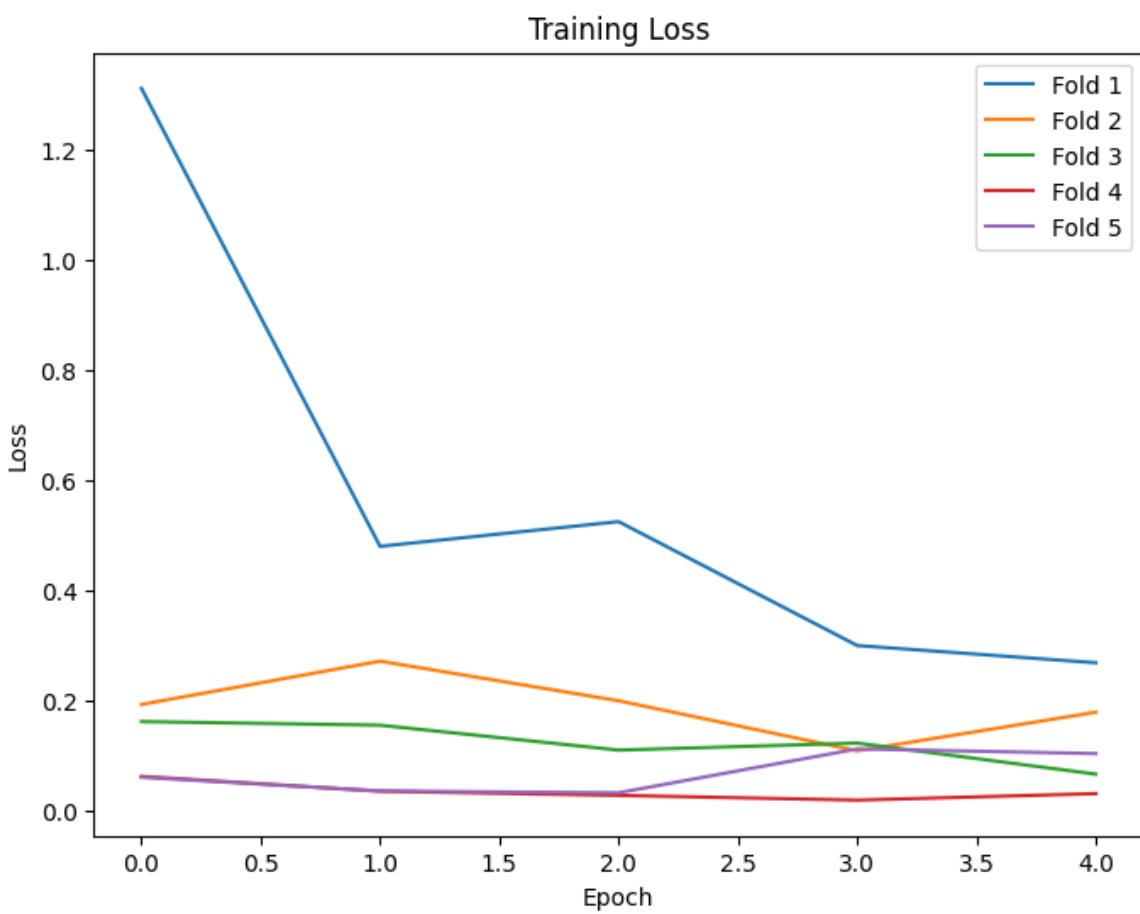
شبکه را پیاده‌سازی می‌کنیم و مدل آموزش داده می‌شود. منحنی‌های loss و accuracy به صورت زیر می‌شوند:



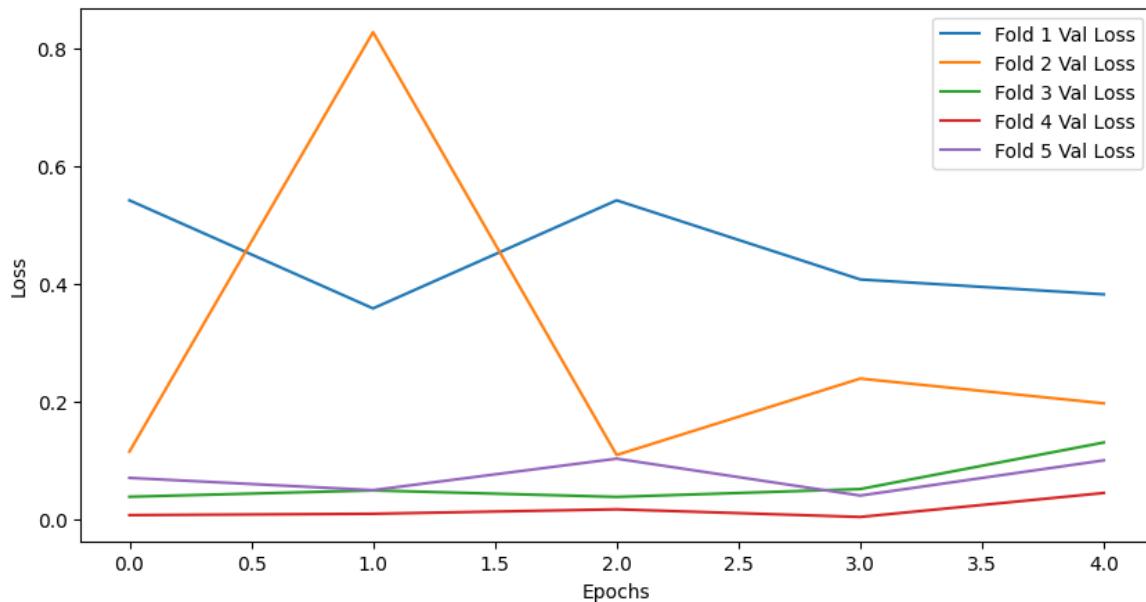
شکل 1.10. نمودار training accuracy



شکل 1.11. نمودار validation accuracy



شكل 1.12. نمودار training loss



شكل 1.13. نمودار validation loss

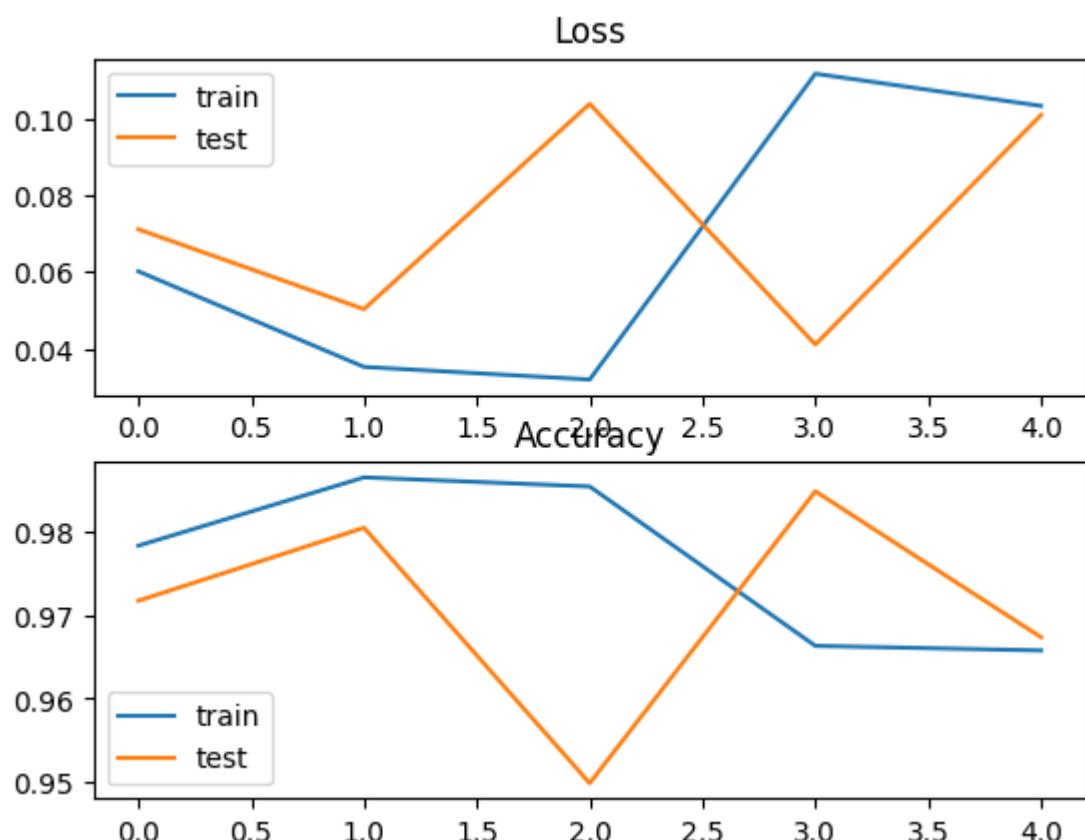
حال مدل را روی داده‌های ارزیابی تست می‌کنیم و مقادیر زیر را برای دقت، خطا و ... خواهیم داشت:

```
[ ] scores= model_final.evaluate(x=x_test,y=y_test)  
24/24 [=====] - 40s 2s/step - loss: 0.6865 - accuracy: 0.8956
```

شکل 1.14. مقادیر دقت و خطا

accuracy: 89.56%

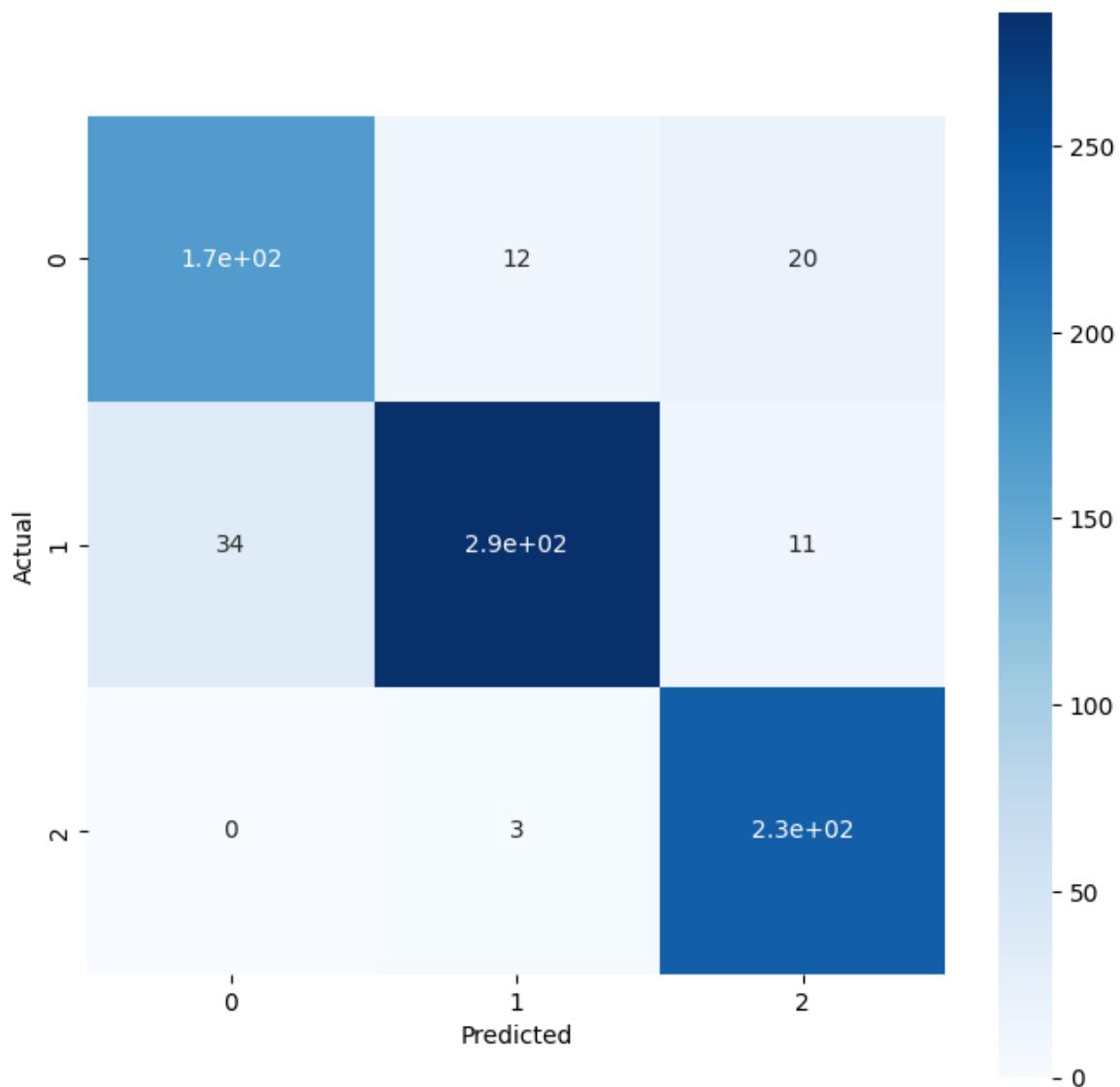
شکل 1.15. مقادیر دقت



شکل 1.16. نمودارهای accuracy و loss

	precision	recall	f1-score	support
class 0	0.84	0.83	0.83	200
class 1	0.86	0.95	0.91	301
class 2	0.99	0.88	0.93	265
accuracy			0.90	766
macro avg	0.90	0.89	0.89	766
weighted avg	0.90	0.90	0.90	766

شکل 1.17. مقادیر precision, recall, f1-score



شکل 1.18. ماتریس طبقه‌بندی

## پاسخ ۲ - تشخیص و شمارش اشیا

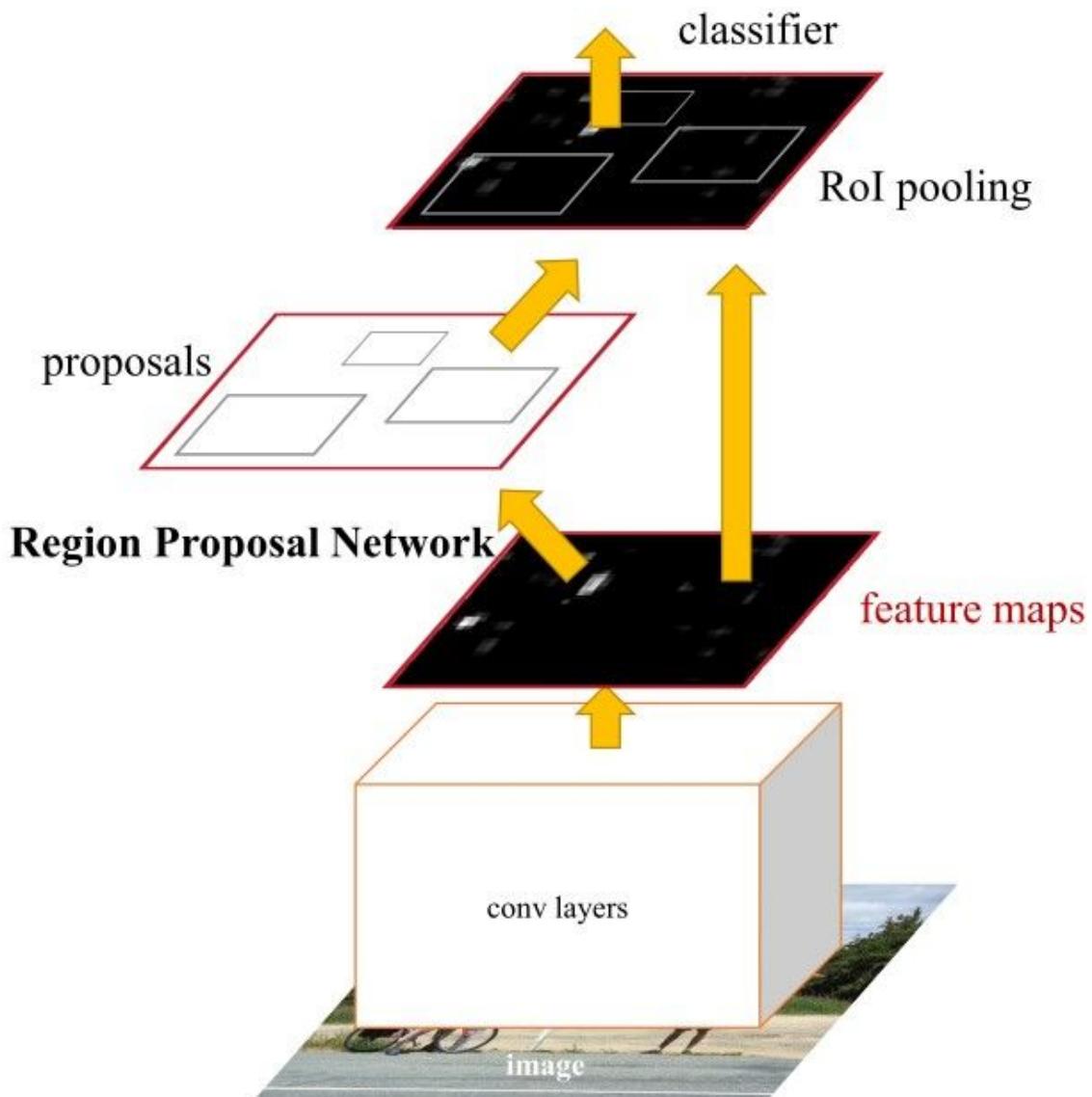
۱. شبکه Faster RCNN یک شبکه عصبی عمیق برای تشخیص اشیاء در تصاویر است. این شبکه از چند مرحله تشکیل شده است که هر مرحله می‌تواند جداگانه برای کارهای دیگری نیز استفاده شود. این شبکه حالت ارتقا یافته شبکه Fast RCNN میباشد که به جای Selective Search از روش RPN برای پیدا کردن باکس استفاده میکند.

Faster RCNN با دریافت تصویر ورودی با سایز دلخواه، یک مجموعه از پنجره‌های نامزد را ایجاد می‌کند. هر پنجره یک امتیاز objectness score را دارد که احتمال وجود یک شیء را تعیین می‌کند. در مقابل مدل‌های قبلی خود مانند Fast RCNN که از Image pyramids که از Anchor Box چیزی به نام Anchor Box را معرفی کرده است. Anchor box‌ها، باکس‌هایی هستند که به صورت پیش‌فرض به هر پیکسل از تصویر اختصاص داده می‌شوند. هر anchor box دارای یک مرکز و یک اندازه خاص است. این anchor box‌ها به عنوان گزینه‌هایی برای مکان تقریبی اشیاء در تصویر استفاده می‌شوند.

در مرحله بعدی، با استفاده از شبکه Convolutional Neural Network (CNN)، برای هر anchor box، یک بردار ویژگی یا feature map تولید می‌شود. سپس با استفاده از این بردار ویژگی، برای هر anchor box، یک احتمال تعلق به هر یک از کلاس‌ها و یک بردار تشخیص محل (bounding box) به دست می‌آید.. سپس این نقشه‌های ویژگی به RPN ارسال شده و جعبه‌های باند و طبقه بندی آنها ایجاد می‌شود. RPN، یک شبکه عصبی عمیق است که به طور خاص برای تشخیص اشیاء در تصاویر استفاده می‌شود. این شبکه، به صورت خودکار نواحی مختلفی از یک تصویر را که ممکن است حاوی اشیاء باشند، شناسایی می‌کند. این نواحی با نام "طرح پیشنهاد" (Region Proposal) شناخته می‌شوند. سپس شبکه‌ی اصلی تشخیص اشیاء از این طرح پیشنهادها استفاده می‌کند و اقدام به تشخیص اشیاء در آنها می‌نماید. در این مرحله تصاویر ابتدا با ابعادی بزرگتر از  $600 \times 600$  پیکسل و کوچک‌تر از  $1000 \times 1000$  پیکسل تبدیل می‌شوند. در هر های خروجی که با  $W \times H$  نشان داده می‌شود بسیار رکوچک‌تر از تصویر ورودی هستند. در هر دو شبکه اصلی مورد استفاده (ZF-Net و VGG) در مقاله گام شبکه ۱۶ است.

از آنجایی که تعداد امکانات در تصاویر بسیار بیشتر از تعداد واقعی اشیاء است، استفاده از یک شبکه پیشنهادی منطقه‌ای می‌تواند به طور قابل توجهی زمان تشخیص اشیاء را کاهش دهد. در واقع، با استفاده از این شبکه، تعداد امکانات قابل بررسی در یک تصویر به طور قابل توجهی کاهش می‌یابد و شبکه تشخیص اشیاء فقط بر روی طرح پیشنهادهایی که به احتمال زیاد حاوی اشیاء هستند، اعمال می‌شود.

نامزدهای انتخاب شده در لایه RoI pooling به نقشه های ویژگی قبلی کانولوشنی برگردانده می شوند و در نهایت به لایه کاملا متصل ارسال می شوند که به عنوان مازول پیشنهاد منطقه از RPN با Fast R-CNN استفاده می شود. به طور خلاصه، Faster R-CNN در اصل Fast R-CNN به عنوان مازول پیشنهاد منطقه است.



شکل 2.1. معماری شبکه Faster RCNN

## ۱- پیش پردازش داده ها

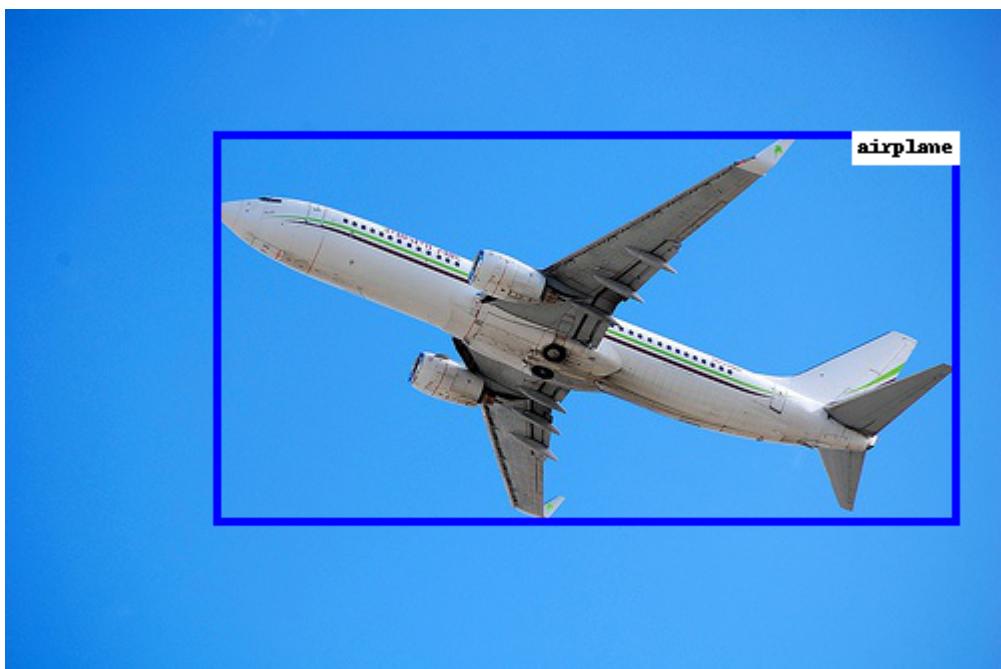
ابتدا دیتاست های مورد نیاز را بارگذاری میکنیم

```
▼ Load Dataset

[11] class_names = ['background', 'person', 'bicycle', 'car', 'motorcycle', 'airplane']
     dataset_train = PASCALDataset('PASCAL/train/')
     dataset_val = PASCALDataset('PASCAL/val/')
     dataset_test = PASCALDataset('PASCAL/test')
```

شکل 2.2. بارگذاری دیتاست ها در مموری

سپس یکی از عکس ها با bounding box آن را از میان داده های train نمایش میدهیم.



شکل 2.3. تصویر هواییما با یک bounding box و یک legend

سپس Data Loader ها را ایجاد میکنیم تا به مدل بدهیم. پارامتر data loader validation مقدار 4 قرار میدهیم و برای data loader train جفت این data loader ها پارامتر shuffle را برابر True قرار میدهیم تا داده ها به شکل تصادفی به شبکه ارسال شوند همچنین تعداد worker ها را ۲ میگذاریم. (تعداد بیشتر در محیط colab امکان پذیر نیست)

## ▼ Create Data Loader

```
[21] data_loader_train = torch.utils.data.DataLoader(  
        dataset_train,  
        batch_size=4,  
        shuffle=True,  
        num_workers=2,  
        collate_fn=utils.collate_fn  
    )  
data_loader_val = torch.utils.data.DataLoader(dataset_val,  
    batch_size=1,  
    shuffle=True,  
    num_workers=2,  
    collate_fn=utils.collate_fn  
)
```

شکل 2.4. ایجاد Data Loader ها

## ۲- ایجاد و آموزش مدل

در این مرحله مدل خود را تعریف میکنیم.

```
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)  
num_ftrs = model.roi_heads.box_predictor.bbox_pred.in_features  
model.roi_heads.box_predictor = FastRCNNPredictor(num_ftrs, len(class_names))  
model = model.to(device)
```

شکل 2.5. بارگذاری مدل

این کد یک مدل کاملاً آموزش دیده از جنس تشخیص شی بصری Faster R-CNN بر اساس معماری ResNet-50 از کتابخانه torchvision بارگیری کند. این مدل با وزن‌هایی که روی مجموعه داده COCO (Common Objects in Context)، شامل تصاویر 80 کلاس مختلف شیء آموزش دیده‌اند، بارگیری می‌شود.

در خط دوم کد، تعداد ویژگی‌ها از آخرین لایه پیش‌بینی کننده باکس مدل استخراج می‌شود. پیش‌بینی کننده باکس مسئول پیش‌بینی جعبه‌های محدود کننده شیء در یک تصویر است.

در خط سوم کد، پیش‌بینی کننده باکس با یک FastRCNNPredictor جدید جایگزین می‌شود. تعداد کلاس‌های خروجی را برابر با طول لیست نام کلاس‌ها قرار داده است. این کار به مدل اجازه

می‌دهد تا باکس محدود کننده‌شی در تصاویری که متعلق به کلاس‌های مشخصی هستند، پیش‌بینی کند.

حال نیاز داریم به مدل optimizer و scheduler را بدهیم. برای SGD استفاده می‌کنیم. از Adam پیشتر استفاده کردیم اما تیجه مطلوبی دریافت نکردیم در این بخش از Scheduler با مقدار momentum=0.9 learning rate با مقدار 0.001 استفاده کردیم. همچنین برای optimizer قبلی با گام‌های 7 تابی میدهیم.

```
optimizer = optim.SGD(  
    model.parameters(),  
    lr=lr,  
    momentum=0.9  
)  
scheduler = lr_scheduler.StepLR(  
    optimizer,  
    step_size=7,  
    gamma=0.1  
)
```

شکل 2.6. تعریف scheduler و optimizer

در یک حلقه با تعداد ایپاک (5 تا) در دو مرحله آموزش و ارزیابی مدل را آموزش میدهیم.

```
for epoch in range(EPOCHS):  
    print(f'--- Epoch {epoch+1}/{EPOCHS} ---')  
    train_phase()  
    best_mAP, best_lr_mAP = validation_phase(best_mAP, best_lr_mAP)
```

شکل 2.7. آموزش مدل

همانطور که در شکل 2.8 مشاهده می‌شود اطلاعات آخرین ایپاک آورده شده است. مشاهده می‌کنیم loss به مقدار 0.13 رسیده است. معیار IoU برای اندازه‌گیری همپوشانی بین جعبه‌های محدود کننده پیش‌بینی شده و جعبه‌های محدود کننده واقعی استفاده می‌شود. این معیار به عنوان نسبتی از مساحت تلاقی دو جعبه به مساحت اجتماع دو جعبه محاسبه می‌شود.

معیار میانگین دقت (AP) اندازه‌گیری دقت تشخیص اشیاء در آستانه‌های مختلف IoU و اندازه‌های مختلف اشیاء است. این معیار از 0 تا 1 است و عدد 1 دقت کامل را نشان می‌دهد.

در خروجی ارائه شده، مقدار AP برای همه مقیاس‌های اشیاء و حداقل 100 تشخیص در هر تصویر را نشان می‌دهد که در آستانه IoU بین 0.50 تا 0.95 قرار دارد. مقدار 0.315 به این معناست که عملکرد تشخیص‌دهنده نسبتاً پایین است و تنها 31.5 درصد از اشیاء به طور دقیق در شرایط مشخص شده تشخیص داده می‌شوند. این به این خاطر است که تعداد ایپاک‌های طی شده نسبت به تعداد ایپاک‌های در مقاله بسیار کمتر است. در تست‌های آن خود متوجه می‌شویم عملکرد مدل آموزش دیده شده آن قدر ها هم بد نیست. اشیا کلی به خوبی تشخیص داده می‌شوند.

```

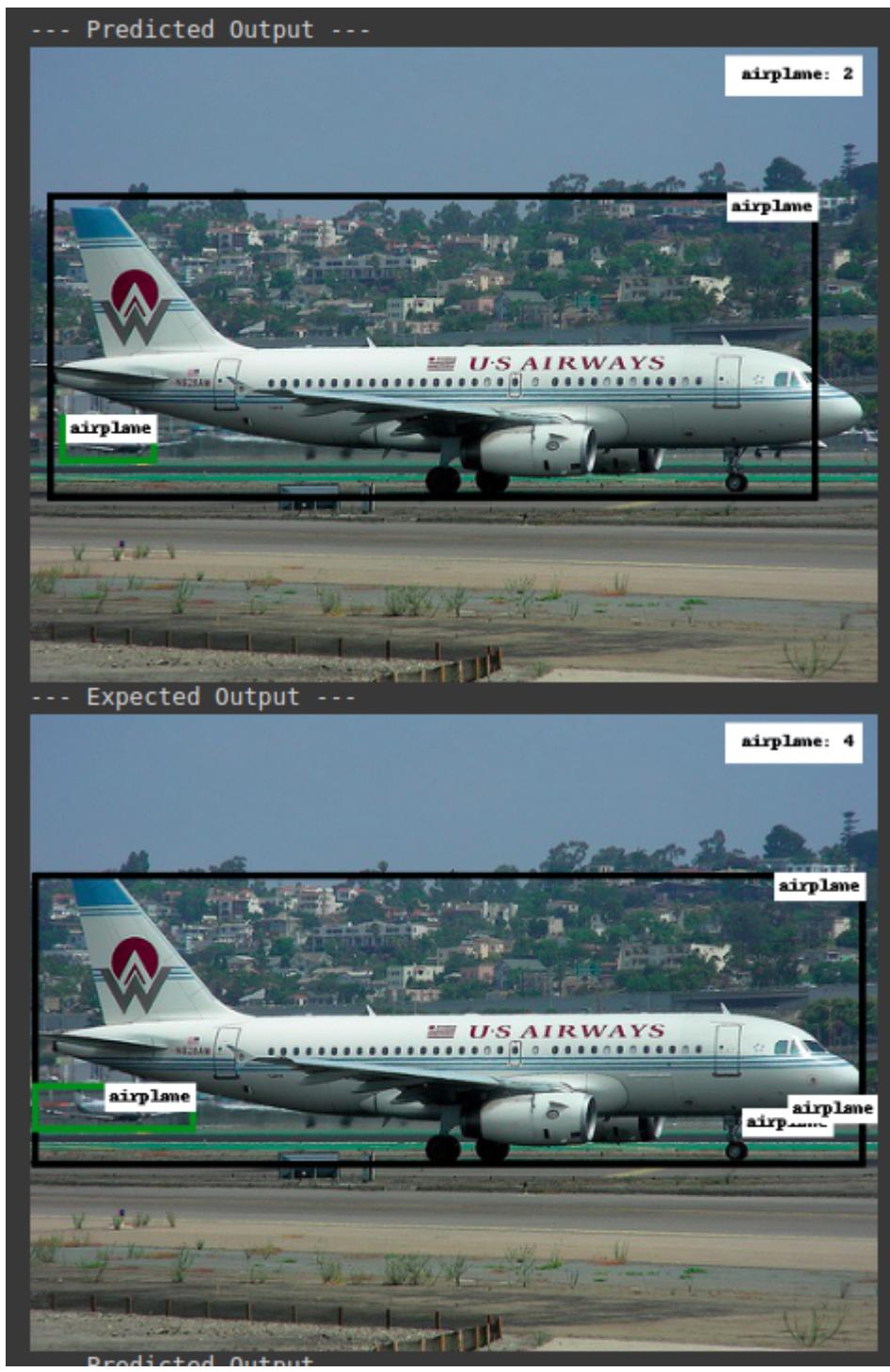
--- Epoch 5/5 ---
Training: 100%|██████████| 251/251 [05:27<00:00,  1.30s/it]
loss:0.13938993215560913
creating index...
index created!
Training: 100%|██████████| 200/200 [00:24<00:00,  8.29it/s]
Accumulating evaluation results...
DONE (t=0.13s).
IoU metric: bbox
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.315
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.513
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.348
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.172
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.343
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.371
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.297
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.474
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.481
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.252
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.516
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.510

```

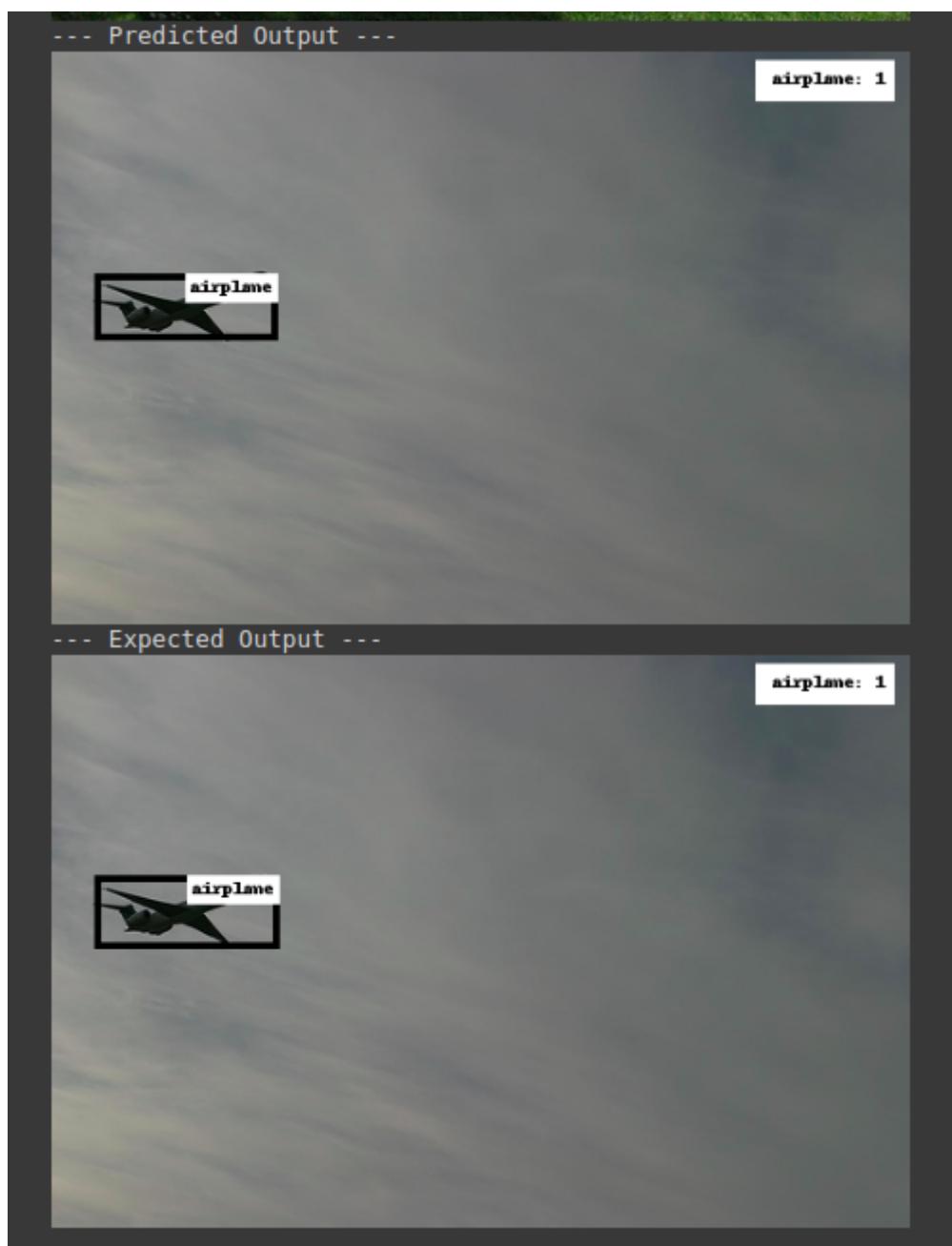
شکل 2.8. اطلاعات مدل در آخرین ایپاک

### ۳- ارزیابی مدل و شمارش اشیا

در این مرحله به ارزیابی مدل می‌پردازیم. تعداد ۳ تصویر را از مجموعه داده test انتخاب می‌کنیم و عملکرد مدل را نسبت به لیل‌های از پیش آماده تعیین می‌کنیم. در خروجی مدل اشیایی با score های متفاوت بین ۰ تا ۱ وجود دارد. در اینجا ما از threshold با مقدار 0.9 استفاده کردیم تا فقط باکس‌هایی نمایش بگیرند که از امتیاز خوبی برخوردارند. در صورت کم انتخاب کردن این مقدار باکس‌هایی رسم می‌شوند که ممکن است آبجکت متناظر با آنها درست نمایش داده نشود.



شکل 2.9. تصویر هواییما در دو نتیجه predicted و expected. در حالت خروجی مدل ما دو هواییما در پشت تشخیص داده نشده. این ممکن است بخاطر fine tune نشدن باشد یا قرار دادن مقدار بالای آن باشد.



شکل 2.10. تشخیص هواپیما در دو حالت درست تشخیص داده شده است.



شکل 2.11. در این تصویر کمی پیچیده تر دوچرخه و انسان مطابق آنچه انتظار میرفت تشخیص داده شد.

