به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

نام و نام خانوادگی	عرفان باقری سولا – محمد قره حسنلو
شماره دانشجویی	۸۱۰۱۹۸۳۶۱ - ۸۱۰۱۹۸۴۶۱
تاریخ ارسال گزارش	۱۴۰۲.۲.۱۳

فهرست

جدول‌ها

No table of figures entries found.

پاسخ ۱. آشنایی با یادگیری انتقالی

گزارش مقاله

به طور کلی اگر بخواهیم به محتویات مقاله بپردازیم، این مقاله به ساخت یک شبکه سبک وزن که بر اساس VGG16 پایه نهادینه شده است، می پردازد. در اینجا به استخراج ویژگی هایی از عکس های از راه دور میپردازد و افزونگی را حذف میکند. این مدل علاوه بر تضمین دقت، تعداد پارامترهای مدل را بر اساس مدل جدید کاهش میدهد. دقت در این مدل جدید تا ۹۸ درصد نیز پیشرفت داشته است. با استفاده از دسته داده EUROSAT دقت تا ۹۵ درصد رسیده است.

کلاس بندی عکس های از راه دور یک مشکل چالشی است و با توجه به محدودیت داده ای که وجود دارد، یادگیری انتقالی به عنوان یک راه حل قلمداد میشود. مشکلاتی که در راه حل های قدیم وجود داشت: ۱- بیشتر شبکه های CNN برای عکس هایی استفاده میشود که ویژگی ها آشکار باشد. ۲- تعداد پارامترها و مموری بسیار بزرگ نیاز دارند. به همین دلیل این مقاله یک مدل بر اساس مدل VGG16 ارائه میدهد. خود VGG16 دارای منابع محاسباتی سنگینی بوده و پارامترهای خیلی بیشتری نیاز دارد که در نتیجه مصرف مموری بالایی دارد. به همین دلیل مدل جدید پیشنهاد شده است.

نکته خیلی مهم: در این پروژه با توجه به صحبت های انجام شده، نیاز داشتن ۲۵ ساعت برای fit شدن مدل جدید با توجه به گفته مقاله و موضوع پروژه یادگیری انتقالی است، تنها یادگیری انتقالی بر اساس وزن های مدل VGG16 انجام میشود.

در Experimental Result که خود مقاله گزارش داده است، دسته داده EUROSAT را به عنوان ورودی و با اندازه ۶۴*۶۴ به مدل داده است و آن را سه بار run کرده است و به دقت ۹۵ درصد رسیده است.

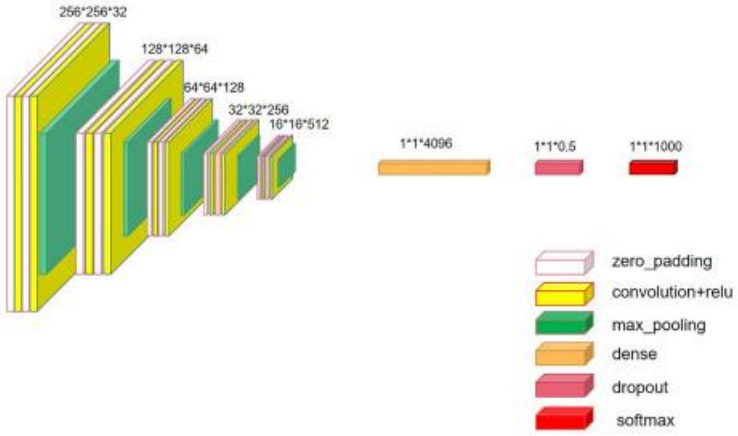
معماری شبکه

مدل توصیه شده بر اساس مقاله به شکل زیر است:

لایه CNN از ۳۲*۳۲ شروع میشود و به ترتیب لایه های ۶۴*۶۴، ۱۲۸*۱۲۸، ۲۵۶*۲۵۶ و ۵۱۲*۵۱۲ اضافه میشود. بعد از هر لایه یک لایه max-pooling که ۲*۲ هستند، اضافه میشود. بعد از هر لایه کانولوشن، یک لایه غیرخطی اضافه میشود، چون ویژگی ها مشخص نیستند و پیکسل ها بسیار کوچک هستند. قبل از هر لایه convolution یک لایه zero padding اضافه میشود که برای افزایش غیرخطی شدن از تابع تصمیم استفاده میشود. در این مدل سه لایه از VGG16 به دو لایه تبدیل میشود تا تعداد

پارامترها کمتر میشود و سرعت را افزایش میدهد. اولین لایه fc دارای ۴۰۹۶ کانال، دومین لایه دارای ۱۰۰۰ و آخرین لایه softmax میباشد.

از ویژگی های خوب این شبکه این است که تعداد پارامترهای کمتری را استفاده میکند و همچنین در زمان کمتری این شبکه fit میشود(نسبت به VGG16 که یک شبکه بسیار بزرگ است.) نیاز دارد. این شبکه برای داده های ultra-low pixel با دقت بهتری عمل کرده است و به دقت ۹۵ رسیده است. این شبکه مصرف مموری کمتری دارد و ویژگی low level را میتواند به خوبی تشخیص دهد.



شکل ۱: معماری پیشنهاد شده توسط مقاله

از ویژگی بد آن میتوان به خاص بودن مدل برای این دسته داده خاص اشاره کرد که برای داده هایی با ultra-low pixels استفاده شده است و ممکن است مدل های دیگر بهتر عمل کنند و همچنین زمان بسیار زیادی نسبت به یادگیری انتقالی که ما انجام دادیم و تنها لایه های واپسین(لایه های fully connected) را تغییر دادیم میگیرد؛ زیرا در این یادگیری انتقالی که از پارامترهای VGG16 استفاده کردیم، زمان زیر ۲۰ دقیقه نسبت به ۲۵ ساعت گفته شده در مقاله نیاز داشت و به دقت حدود ۸۸ درصد نیز رسیدیم و تنها در ۴۰ epoch برازش کرده ایم.

با استفاده از یادگیری انتقالی، لایه های fc مربوط به VGG16 را حذف کرده و لایه جدید که شامل یک لایه fc با ۴۰۹۶ نورون، یک لایه dropout، یک لایه fc با ۱۰۰۰ نورون، یک لایه dropout و در نهایت یک لایه fc با ۱۰ نورون(به تعداد کلاس ها) است، اضافه میکنیم.

```
block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0
block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160
block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808
block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808
block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0
block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808
block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808
block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808
block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0
flatten (Flatten)           (None, 2048)              0
dense (Dense)               (None, 4096)              8392704
dropout (Dropout)           (None, 4096)              0
dense_1 (Dense)             (None, 1000)              4097000
dropout_1 (Dropout)         (None, 1000)              0
dense_2 (Dense)             (None, 10)                10010
=================================================================
Total params: 27,214,402
Trainable params: 12,499,714
Non-trainable params: 14,714,688
```

شکل ۲: قسمت انتهایی مدل جدید با استفاده از یادگیری انتقالی بر روی شبکه **VGG16**

پیش پردازش هایی که مقاله برای مدل پیشنهادی داده است به شکل روبه رو میباشد؛ در ابتدا اندازه هر عکس را به ۶۴*۶۴ تبدیل میکند. سپس هر عکس را بعد از گذشت از لایه های convolution به صورت flat در می آورد. بعد از آن، بردار flat شده به شکل آرایه در می آید و ویژگی های آشکار آن با string operation نمایش داده میشود. در آخر هر پیکسل image تقسیم بر ۲۲۵ میشود تا scale شود.

اما پیش پردازش هایی که برای یادگیری انتقالی انجام دادیم، تنها تبدیل عکس ها به اندازه 3*64*64 بوده و بعد از آن اندازه هر عکس را تقسیم بر ۲۵۵.۰ کرده ایم تا scale شوند و پیش پردازش های انجام شده به شکل زیر میباشد.

Preprocess each image of dataset

```
modified_images = []

for indx, image in enumerate(images):

  # Resize the image to (64, 64, 3)
  if image.shape != (64, 64, 3):
    image = cv2.resize(image.numpy(), (64, 64, 3))

  # Scale the image
  image = np.divide(image, 255.0)

  modified_images.append(image)

modified_images = np.array(modified_images)
labels = np.array(labels)
```

شکل ۳: پیش پردازش های انجام شده

در نهایت داده ها را split کرده و ۸۰ درصد داده ها را به train set و 20 درصد داده ها را به test set اختصاص میدهیم.

Split dataset to training, validation and test sets

```
X_train, X_test, y_train, y_test = train_test_split(modified_images, labels, test_size=0.2, stratify=labels)

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

print(f'Train Shape: {X_train.shape}')
print(f'Test Shape: {X_test.shape}')
```

```
Train Shape: (21600, 64, 64, 3)
Test Shape: (5400, 64, 64, 3)
```

شکل ۴: تقسیم داده ها به داده های آموزش و ارزیابی

نوع عکس های قابل تشخیص

در این مقاله، بر روی عکس هایی با پیکسل های بسیار کوچک(ultra-low pixel images) تمرکز دارد که برای این کار باید دسته داده را گسترش داد وگرنه دقت پایینی خواهیم داشت. این داده استخراج ویژگی کافی ندارند چون داده ها از نوع ultra-low pixel images هستند. همچنین ویژگی های کمی در پیکسل های کوچک میباشد که باعث همان دقت کم میشود. برای حل این مشکل از zero padding استفاده شده است تا ویژگی هایی با جزئیات بیشتری را استخراج کند. پس با این کار دیتا گسترش می یابد و ویژگی ها low level را استخراج میکند و در نتیجه دقت افزایش می یابد.

عکس ها در اینجا ساختارهای فضایی پیچیده را نمایش میدهد که دارای متغیرهایی با intraclass بالا و interclass پایین هستند، به همین دلیل برای حل این مشکل، scene classification های متعددی پیشنهاد شده اند.

اگر داده های در یک category وجود نداشته باشد، یک مشکل برای مدل CNN قلمداد میشود؛ چ.ن شبکه هیچ داده ای برای یادگیری مدل آن category ندارد. برای همین میتوانیم برای آن کتگوری، تعدادی داده جمع آوری کنیم و سپس مذل را برازش کنیم، اگر تعداد کمی داده در آن category وجود داشته باشد،میتوانیم از تکنیک های data augmentation مانند flipping horizontally or vertically، cropping، transposing، rotating، shifting و... استفاده کنیم تا بتواین تعداد کم داده در یک category را تا حدودی جبران کنیم و در آخر اگر هیچکدام جواب نداد، میتوانیم آن category را حذف کنیم.

بررسی دادگان

دسته داده که داده شده است، دارای ۱۰ کلاس بوده(که در نتیجه خروجی شبکه باید دارای ۱۰ نورون باشد) و به طور کلی ۲۷۰۰۰ عکس استفاده شده است که در هر کلاس ۲۰۰۰-۳۰۰۰ عکس وجود دارد که دارای اندازه ۶۴*۶۴ میباشد.

شکل ۵: نمونه عکس های هر **category**

گزارش نتایج

با استفاده از یادگیری انتقالی به دقت حدود ۸۸ درصد رسیدیم که نشان میدهد، با استفاده از یادگیری انتقالی اگرچه دقت نسبت به مدل جدید کمتر است اما در زمان خیلی سریعتری به جواب رسیده ایم. f1 score در واقع حدود ۸۸ درصد بوده و precision حدود 89 درصد شده است. همه این مقادیر برای دادگان آموزش و ارزیابی بعد از 40 epoch تقریبا یکسان به پایان رسیده اند.

برای قسمت training، پارامترها طبق مقاله به مقدار زیر در آمده است:

batch size = 32

momentum = 0.9

learning rate = 0.001

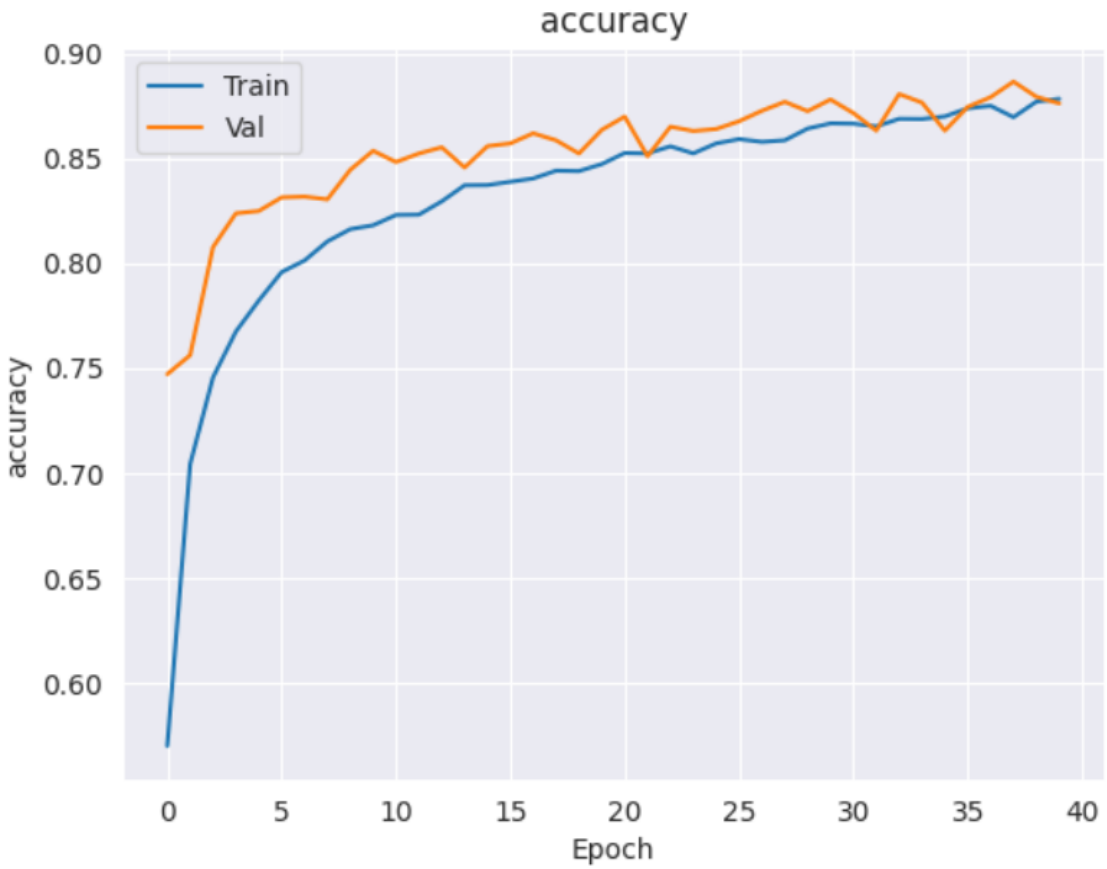
epochs = 40

dropout rate = 0.5

C, D: Training and Testing

```
def f1_score_metric(y_true, y_pred):
    return f1_score(tf.argmax(y_true, axis=1), tf.argmax(y_pred, axis=1), average='macro')

sgd = SGD(learning_rate=0.001, momentum=0.9)
model.compile(optimizer=sgd, loss='categorical_crossentropy',
              metrics=['accuracy', tf.keras.metrics.Precision()])

# train the model on your dataset
with tf.device('/device:GPU:0'):
  history = model.fit(X_train, y_train, batch_size=32, epochs=40, validation_data=(X_test, y_test))
```

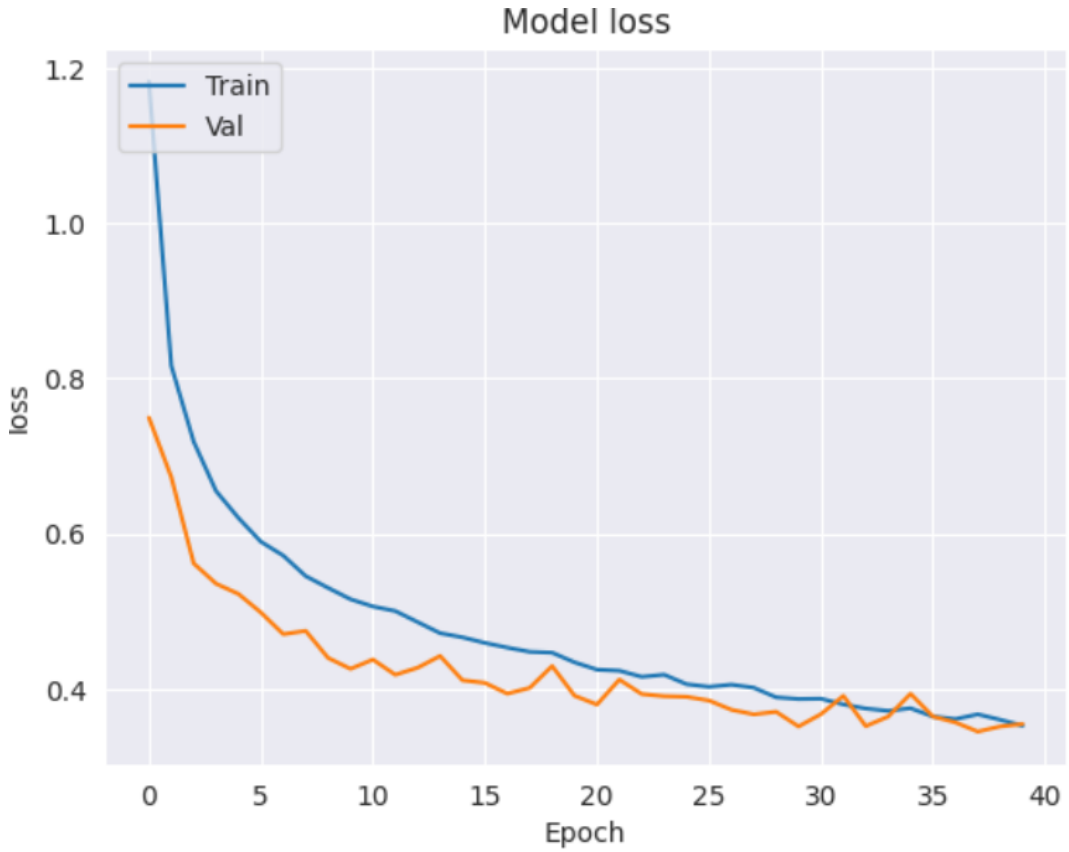
شکل ۶: مقادیر **hyperparameters**



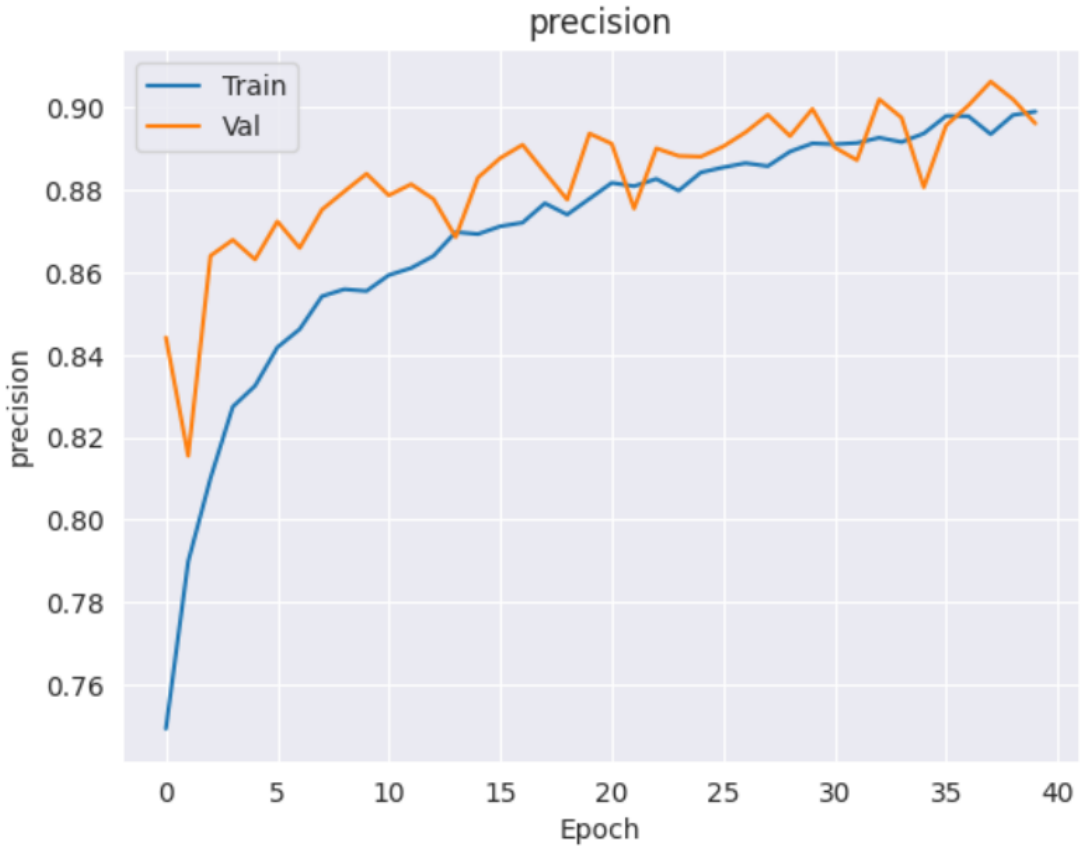
شکل ۷: نمودار دقت برای دادگان آموزش و ارزیابی

```
169/169 [==============================] - 3s 15ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       600
           1       0.92      0.89      0.91       600
           2       0.90      0.87      0.88       600
           3       0.75      0.86      0.80       500
           4       0.93      0.95      0.94       500
           5       0.71      0.90      0.80       400
           6       0.90      0.75      0.82       500
           7       0.94      0.95      0.94       600
           8       0.83      0.76      0.79       500
           9       0.96      0.93      0.94       600

    accuracy                           0.88      5400
   macro avg       0.87      0.87      0.87      5400
weighted avg       0.88      0.88      0.88      5400

Confusion Matrix:
[[531   2   3  12   0  20   8   0  19   5]
 [  0 536   3   0   0  40   0   0   4  17]
 [  0  14 520  13   0  21  13  12   7   0]
 [ 12   0  10 428   7   6   8   3  26   0]
 [  1   0   0   5 473   0   6  15   0   0]
 [  2  14   6   4   0 361   2   0  11   0]
 [ 19   0  28  37  15  12 376   6   7   0]
 [  0   0   4   3  15   2   7 568   1   0]
 [ 16   3   5  71   1  22   0   1 381   0]
 [  5  11   1   0   0  22   0   0   4 557]]
```

شکل ۸: ماتریس طبقه بندی و **f1 score** برای دادگان ارزیابی



شکل ۹: نمودار خطا برای دادگان آموزش و ارزیابی



شکل ۱۰: نمودار **precision** برای دادگان آموزش و ارزیابی

پاسخ ۲ – تشخیص و شمارش اشیا

توضیح معماری شبکه Faster-RCNN

مدل Faster R-CNN با استفاده از چارچوب یادگیری عمیق Caffe پیاده‌سازی شده است و از چندین مؤلفه که به صورت end-to-end آموزش داده می‌شوند، تشکیل شده است.

اولین مؤلفه مدل Faster R-CNN یک شبکه عصبی کانولوشنی عمیق (CNN) است که به عنوان یک استخراج کننده ویژگی مشترک استفاده می‌شود. نویسندگان از معماری VGG-16 به عنوان CNN پایه استفاده می‌کنند و لایه کاملاً متصل اول را با یک لایه کانولوشنی جایگزین می‌کنند تا ورودی تصاویر با اندازه متغیر را ممکن کند. از شبکه های دیگری مانند ResNet نیز می توان به عنوان شبکه پایه استفاده کرد. شبکه CNN بر روی مجموعه داده ImageNet آموزش داده شده است که به آن این امکان را می‌دهد تا ویژگی های بصری عمومی که برای شناسایی اشیاء مورد استفاده قرار می‌گیرد را یاد بگیرد.

مؤلفه دیگر، شبکه پیشنهاد منطقه یا RPN می باشد که یک شبکه کاملاً کانولوشنی است که نقشه های ویژگی (feature map) خروجی CNN را به عنوان ورودی دریافت کرده و منطقه های پیشنهادی برای وجود اشیاء مورد نظر را تولید می‌کند. RPN از سه لایه کانولوشنی به همراه دو لایه کاملاً متصل موازی تشکیل شده است که یکی برای محاسبه bounding box و دیگری برای طبقه بندی اهمیت شیء موجود در آن ناحیه می باشد. لایه‌های کانولوشنی از کرنل های 3×3 با stride ۱ و padding ۱ استفاده می‌کنند که به RPN امکان حفظ resolution هندسی نقشه های ویژگی (feature map) ورودی را می‌دهد. RPN یک مجموعه از bounding box های با اندازه ثابت را بر روی یک شبکه فشرده از نقشه های ویژگی تولید می‌کند و برای هر ناحیه، امتیاز شیء و offset محدوده ها پیش‌بینی می‌شود. امتیاز شیء به عنوان یک امتیاز طبقه بندی دودویی با استفاده از تابع softmax محاسبه می‌شود و محدوده اشیاء به صورت رگرسیون با استفاده از L1 loss محاسبه می‌شود.

detector یا شناسایی کننده Fast R-CNN، نواحی یا ROI های تولید شده توسط RPN را به عنوان ورودی دریافت کرده و طبقه بندی اشیاء و رگرسیون جهت بهبود ROI را انجام می‌دهد. این detector بر پایه ساختار VGG-16 استوار است، که لایه‌های کاملاً متصل را با دو لایه موازی جایگزین می‌کند: یکی برای طبقه بندی و دیگری برای رگرسیون. ROI ها ابتدا با استفاده از لایه ROI pooling به یک اندازه ثابت تبدیل می شوند تا نقشه های ویژگی اندازه ثابتی داشته باشند. سپس ویژگی‌های پول شده به لایه‌های کاملاً متصل فرستاده می‌شوند تا محدوده های بهبود یافته و امتیاز کلاس ها تولید شوند.

مدل Faster R-CNN به صورت end-to-end با استفاده از تابع هزینه چند وظیفه ای آموزش داده می‌شود که همه عناصر روش‌های RPN و Fast R-CNN را با هم ترکیب می‌کند. این تابع هزینه برای هر ROI به صورت هزینه طبقه بندی و هزینه بازیابی باکس و با استفاده از یک عامل وزن دهی برای تعادل میان اثرات هر دو هزینه محاسبه می‌شود.

به طور کلی، مدل Faster R-CNN یک ساختار شبکه عصبی پیچیده و با محاسبات بالا است که نیاز به تنظیم دقیق پارامترهای زیاد دارد. با این حال، این مدل نشان داده است که در چندین مجموعه داده ی پایه عملکرد خوبی دارد.

آموزش شبکه

دیتاست مربوطه را از لینک داده شده دانلود کرده و در گوگل درایو آپلود می کنیم تا در ادامه بتوانیم در محیط Google Colab از آن استفاده کنیم. از تابع PASCALDataset برای آماده سازی مجموعه دادگان و ایجاد Data Loader های مربوط به بخش های train ، evaluation و test استفاده می کنیم. پارامتر batch size را برای دادگان آموزش برابر ۲ تنظیم می کنیم چراکه اعداد بزرگتر برای این پارامتر باعث پر شدن حافظه GPU می شوند و از بین اعداد کوچک نیز طبق آزمایشات ما این عدد نتیجه بهتری حاصل می کند.

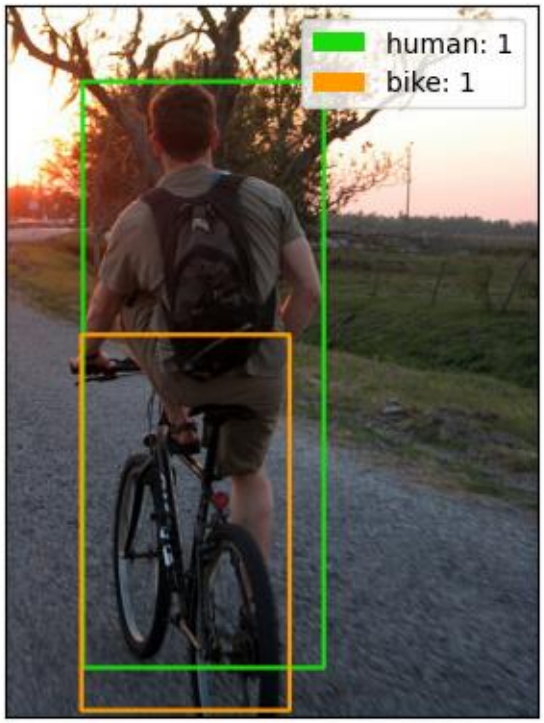
سپس مدل fasterrcnn_resnet50_fpn را لود کرده و از FastRCNNPredictor به عنوان box predictor آن استفاده می کنیم. مدل را نیز روی GPU می بریم تا سرعت آموزش و تست را بالا ببریم. به عنوان optimizer نیز از SGD با نرخ یادگیری 0.005 و مومنتوم 0.9 استفاده می کنیم. سپس از یک learning rate scheduler نیز استفاده می کنیم که پس از گذشت هر epoch نرخ یادگیری را نصف کند. طبق آزمایشات ما این پارامتر ها بهترین نتایج را حاصل کردند.

در آخر با استفاده از تابع train_one_epoch مدل را برای ۵ epoch آموزش می دهیم. به علت استفاده از این تابع آماده نمودار های دقت و loss را برای مولفه های مختلف نمی توانیم رسم کنیم. خروجی تابع evaluate پس از هر epoch در فایل نوتبوک مربوطه قابل مشاهده است.

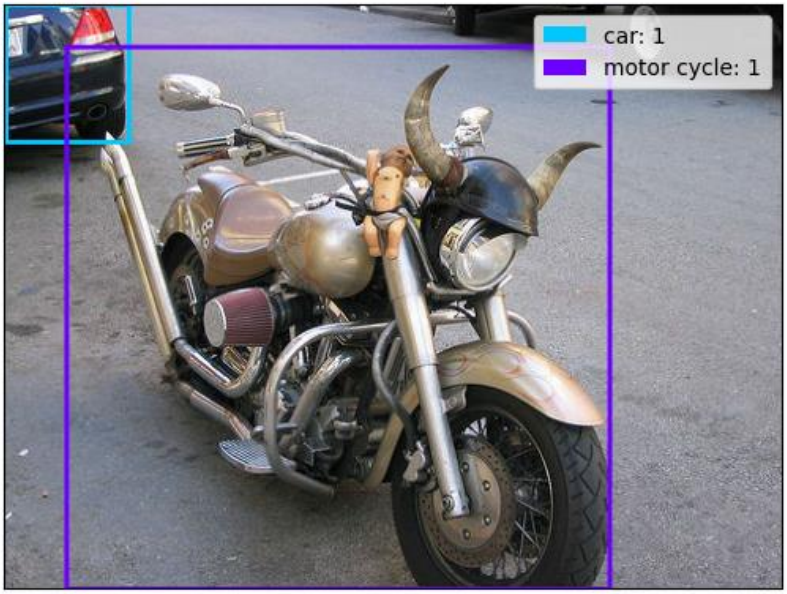
پس از پایان آموزش مدل را ذخیره می کنیم.

تشخیص و شمارش اشیا

خروجی مدل روی ۳ تصویر از مجموعه داده test و همچنین خروجی روی دو تصویر دلخواه خارج از مجموعه دادگان را در ادامه مشاهده می کنیم.

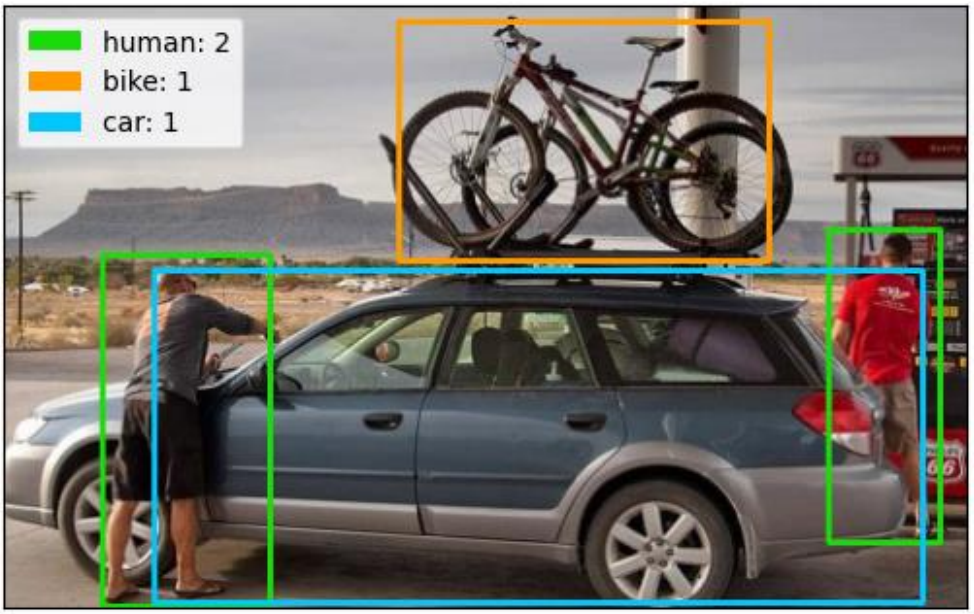


شکل ۱۱: خروجی مدل روی داده ارزیابی ۱

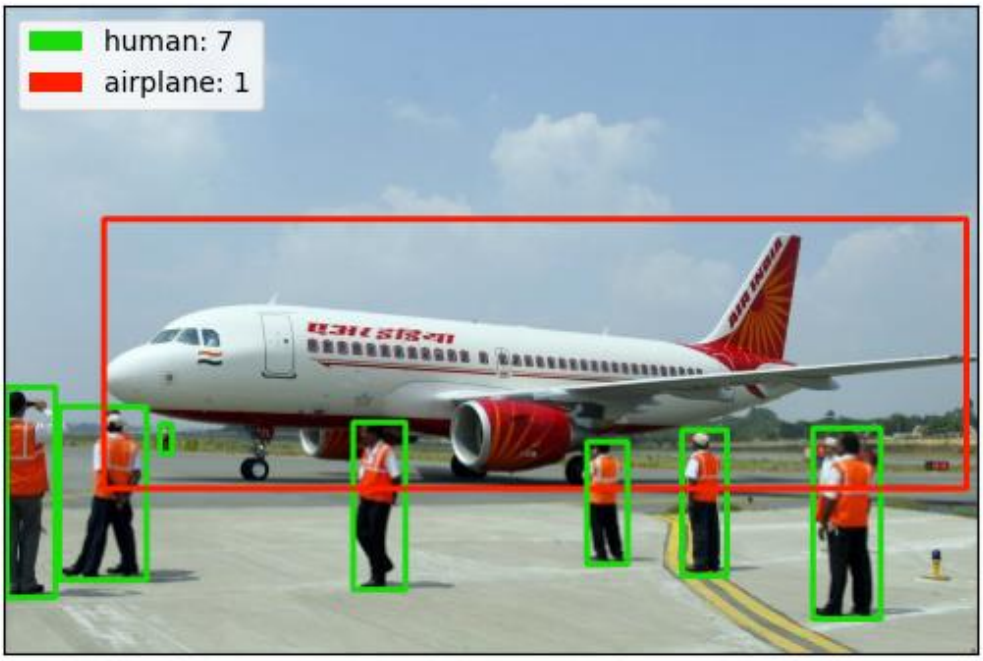


شکل ۱۲: خروجی مدل روی داده ارزیابی ۲

شکل ۱۳: خروجی مدل روی داده ارزیابی ۳



شکل ۱۴: خروجی مدل روی داده دلخواه ۱



شکل ۱۵: خروجی مدل روی داده دلخواه ۲

جالب است به تشخیص انسان زیر در تصویر بالا توجه کنیم:

شکل ۱۶: تصویر بزرگ نمایی شده شکل ۱۵