به نام خدا

پروژه درس بینایی کامپیوتر: آنالیز فوتبال

امید قهرودی
علیرضا رضایی

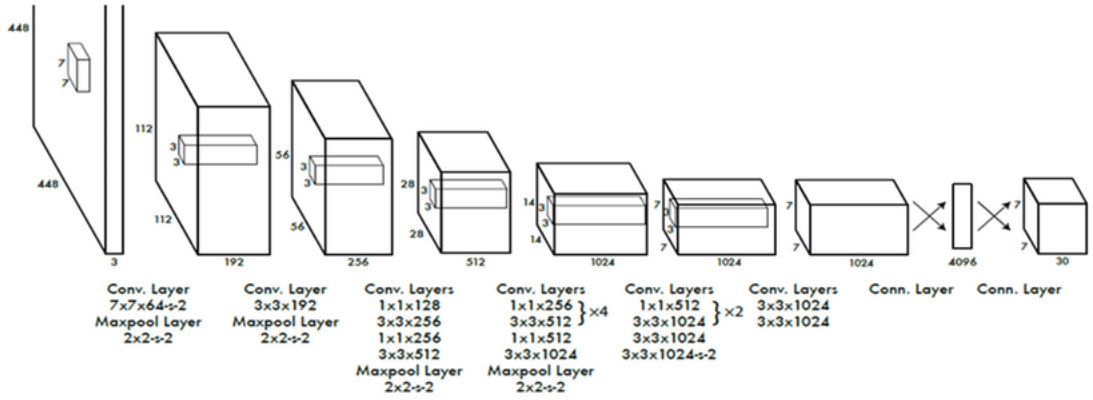
برای این پروژه ابتدا نیاز به پیدا کردن داده هم برای یادگیری شبکه و هم برای امتحان برنامه ی خود داشتیم. ابتدا فیلمی که به صورت پانوراما از تمام زمین بازی گرفته شده بود را امتحان کردیم ولی به دلیل کوچک بودن و بی کیفیت بودن آن امکان تشخیص بازیکنان وجود نداشت به همین دلیل از فیلم هایی که از نمای نزدیک هستند استفاده کردیم.

برای قسمت امتیازی تشخیص heatmap را انجام میدهیم برای اینکار ابتدا بک گراند تصویر را پیدا میکنیم و از عکس اصلی کم میکنیم سپس برای تمام فریم های فیلم اینکار را انجام میدهیم تا در اخر heatmap بدست بیاید.

برای حذف بک گراند از کلاس createBackgroundSubteractorMOG استفاده کردیم.

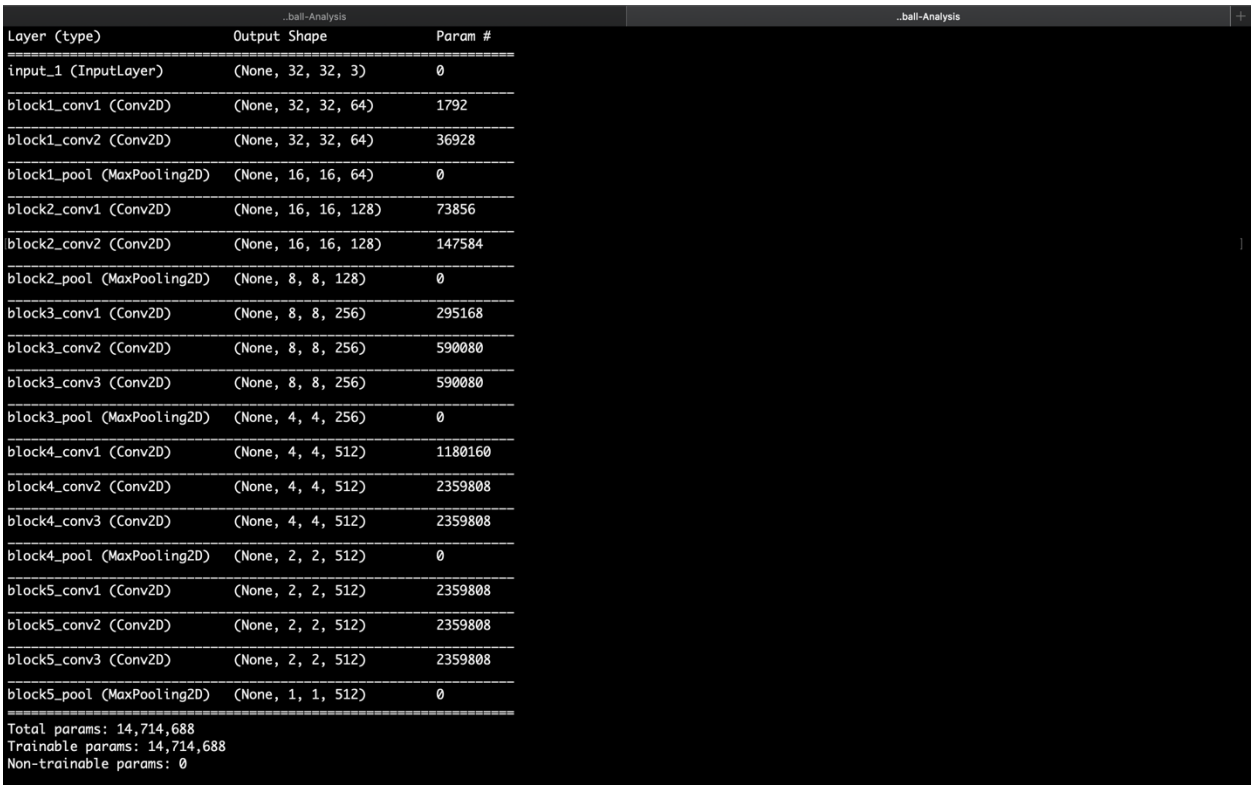
برای تسک تشخیص درصد مالکیت توپ ابتدا باید ابجکت های بازیکن و توپ را تشخیص دهیم که از شبکه YOLO استفاده کردیم از آنجایی که ما دیتا مربوط به فوتبال نداشتیم از YOLO که قبلا بر روی دیتاست coco اموزش داده شده است استفاده کردیم تا فیچر های مربوط به عکس را استخراج کند وزن هایYOLO که برروی دیتاست coco یادگرفته شده‌اند و به همراه تنظیمات در فولدر yolo-coco موجود میباشد که با استفاده از دستور cv2.dnn.readNetFromDarknet شبکه را بارگزاری میکنیم و با استفاده از آن کلاس اشیا موجود در یک عکس به همراه مستطیل دور آن را بدست میآوریم.

مدل YOLO در زیر آورده شده است:

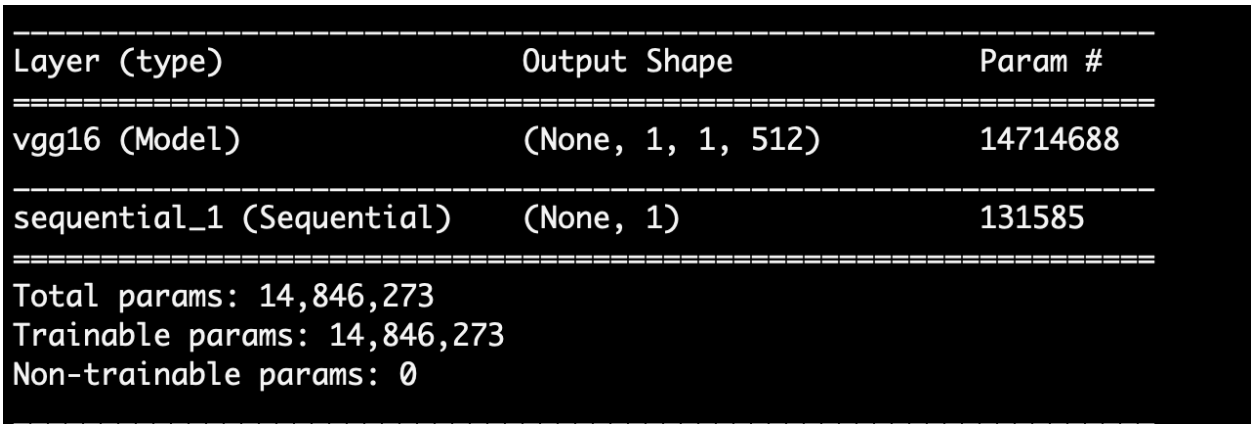


در این دیتاست coco کلاس های توپ فوتبال و ادم موجود است حال باید بین ابجکت های پیدا شده دسته بندی کنیم که کدام ابجکت مربوط به کلاس تیم سفید و کدام مربوط به تیم قرمز است. برای اینکار احتیاج به یک دیتاست مربوط به بازیکن های تیم قرمز و سفید داشتیم برای اینکار خودمان دیتا را درست کردیم که در پوشه data در دو فولدر train و validation پخش کردیم سپس یک شبکه بر روی این داده ها که بتواند بازیکن هر تیم را تشخیص دهد اموزش دادیم برای اینکار ابتدا یک شبکه ساده با استفاده از کراس که شامل چند لایه کانولوشن و دنس میباشد را ساختیم و آموزش دادیم ولی با توجه به اینکه تعداد دیتا ها کم میباشد برای بهتر شدن عملکرد مانند YOLO از یک شبکه از پیش اموزش داده شده(vgg بر روی دیتاست imageNet) استفاده کردیم سپس طبقه بند ان را جدا کردیم و به شبکه خودمان وصل کردیم و ان را اموزش دادیم.

در شکل زیر مدل های ساخته شده و فرآیند یادگیری مشاهده میشود:



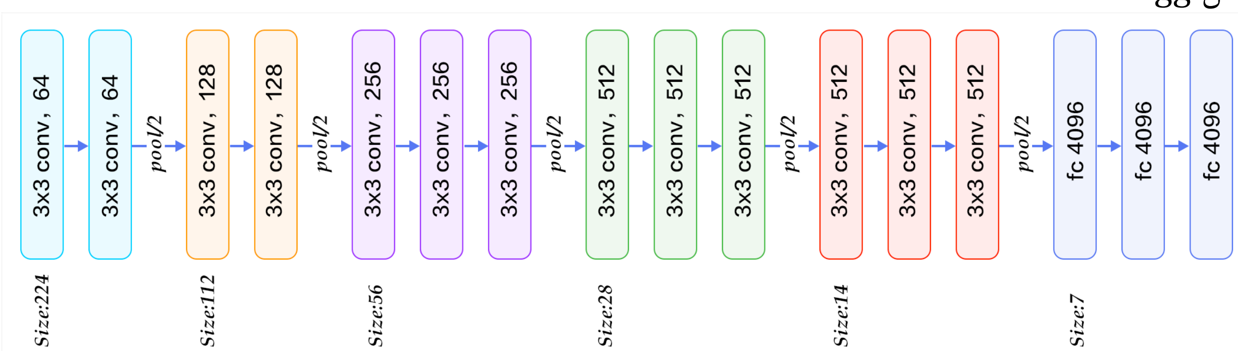
```
..bali-Analysis                                                    ..bali-Analysis
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 1, 1, 512)         14714688
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 131585
=================================================================
Total params: 14,846,273
Trainable params: 14,846,273
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/10
53/53 [==============================] - 12s 234ms/step - loss: 0.5094 - acc: 0.7311 - f1: nan - recall: 0.6038 - precision: 0.6525 - val_loss: 0.2010 - val_a
cc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00001: val_acc improved from -inf to 1.00000, saving model to /Users/omid/Desktop/Computer Vision/Football-Analysis/001.h5
Epoch 2/10
53/53 [==============================] - 12s 221ms/step - loss: 0.1240 - acc: 0.9858 - f1: nan - recall: 0.9151 - precision: 0.9088 - val_loss: 0.0209 - val_a
cc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00002: val_acc did not improve from 1.00000
Epoch 3/10
53/53 [==============================] - 12s 221ms/step - loss: 0.0408 - acc: 0.9906 - f1: nan - recall: 0.9057 - precision: 0.8994 - val_loss: 0.0063 - val_a
cc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00003: val_acc did not improve from 1.00000
Epoch 4/10
53/53 [==============================] - 11s 214ms/step - loss: 0.0218 - acc: 0.9953 - f1: nan - recall: 0.9057 - precision: 0.8994 - val_loss: 0.0025 - val_a
cc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00004: val_acc did not improve from 1.00000
Epoch 5/10
53/53 [==============================] - 11s 217ms/step - loss: 0.0158 - acc: 0.9906 - f1: nan - recall: 0.9151 - precision: 0.9245 - val_loss: 0.0023 - val_a
cc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00005: val_acc did not improve from 1.00000
Epoch 6/10
53/53 [==============================] - 13s 245ms/step - loss: 0.0206 - acc: 0.9858 - f1: nan - recall: 0.9575 - precision: 0.9434 - val_loss: 7.3684e-04 - v
al_acc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00006: val_acc did not improve from 1.00000
Epoch 7/10
53/53 [==============================] - 13s 240ms/step - loss: 0.0058 - acc: 1.0000 - f1: nan - recall: 0.9623 - precision: 0.9623 - val_loss: 7.7039e-04 - v
al_acc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00007: val_acc did not improve from 1.00000
Epoch 8/10
53/53 [==============================] - 12s 218ms/step - loss: 0.0025 - acc: 1.0000 - f1: nan - recall: 0.8868 - precision: 0.8868 - val_loss: 5.8715e-04 - v
al_acc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000

Epoch 00008: val_acc did not improve from 1.00000
Epoch 9/10
53/53 [==============================] - 12s 231ms/step - loss: 0.0022 - acc: 1.0000 - f1: nan - recall: 0.9434 - precision: 0.9434 - val_loss: 3.8117e-04 - v
al_acc: 1.0000 - val_f1: 1.0000 - val_recall: 1.0000 - val_precision: 1.0000
```

مدل vgg:



همانطور که مشاهده میشود به دلیل ساده بودن مساله شبکه به دقت ۱۰۰ درصد میرسد.

حال در قسمت اصلی برنامه ویدیو را هر ۵ فریم یکبار به YOLO میدهیم و باکس های خروجی آن را به شبکه اموزش داده توسط خودمان میدهیم سپس تشخیص میدهیم این بازیکن مربوط به کدام تیم میباشد سپس فاصله بازیکن ها تا توپ را محاسبه میکنیم و درصد مالکیت را برای آن تیم که به توپ نزدیکتر است در نظر میگیریم.

منابع و کدها:

https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/
https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3
https://keras.io/examples/mnist_cnn/
https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/
https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/
https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/