**دانشگاه علم و صنعت**

**تمرین ششم مبانی بینایی کامپیوتر**

**نام و نام خانوادگی:**

فرناز خوش دوست آزاد

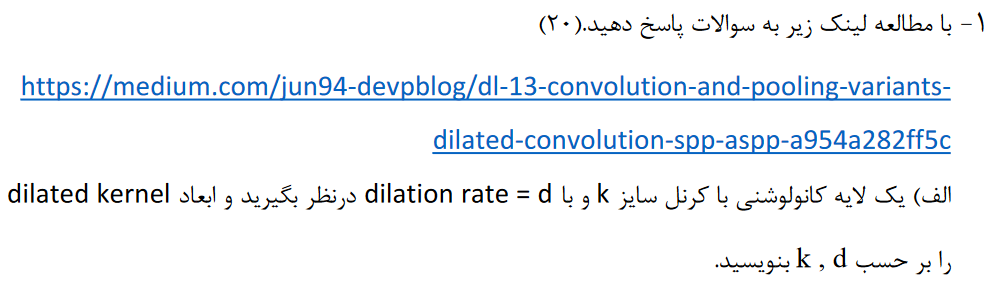
**شماره دانشجویی:**

99521253

**نام استاد:**

دکتر محمدرضا محمدی

1)



پاسخ:

(Dilation Rate) را می‌توان با توجه به سایز کرنل اصلی و نرخ دیلیشن محاسبه کرد. فرمول کلی برای محاسبه ابعاد موثر (Effective Receptive Field) کرنل دیلیت شده به صورت زیر است:

Dilated\_Kernel\_Size =K + (K - 1)\* (d-1)

این فرمول به ما نشان می‌دهد که چگونه افزایش نرخ دیلیشن باعث افزایش ابعاد موثر کرنل می‌شود بدون اینکه تعداد پارامترها افزایش یابد.



پاسخ:

همانطور که در قسمت پیشین گفته شد تعداد پارامترهای قابل آموزش یک برابر می شود و هیچ تغییری نمی کند:

افزایش (Dilation Rate) تأثیری بر تعداد پارامترهای قابل آموزش یک لایه ندارد. تعداد پارامترهای یک لایه کانولوشنی تنها به اندازه کرنل (فیلتر) و تعداد فیلترها (یا کانال‌های خروجی) بستگی دارد و نه به نرخ دیلیشن.

توضیح بیشتر:

تعداد پارامترهای یک لایه کانولوشنی به صورت زیر محاسبه می‌شود:

Output Channels \* (1+Input channels\*kernel Width\*kernel Height)=the number of params

در این فرمول:

- Kernel Height: ارتفاع کرنل

- Kernel Width: عرض کرنل

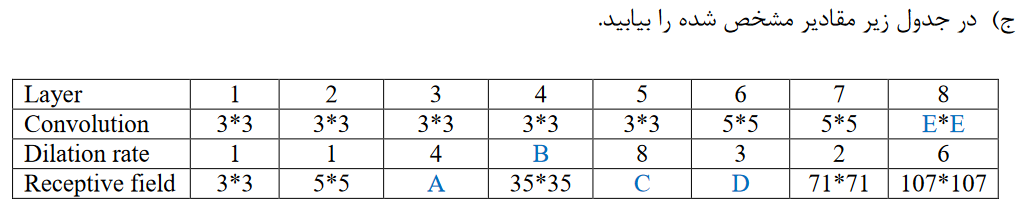
- Input Channels: تعداد کانال‌های ورودی

- Output Channels: تعداد کانال‌های خروجی

- عدد ۱ برای بایاس (bias) است که به هر فیلتر اضافه می‌شود.

اثر Dilation Rate:

نرخ dilation تنها بر روی فاصله بین پیکسل‌های نمونه‌برداری‌شده توسط کرنل تأثیر می‌گذارد و تعداد پارامترهای کرنل را تغییر نمی‌دهد. به عبارت دیگر، نرخ دیلیشن تنها نحوه اعمال کرنل بر روی تصویر را تغییر می‌دهد و باعث می‌شود که کرنل‌ها اطلاعات بیشتری از تصویر بگیرند بدون اینکه تعداد پارامترهای قابل آموزش افزایش یابد.



پاسخ:

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | layer |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7\*7 | 5\*5 | 5\*5 | 3\*3 | 3\*3 | 3\*3 | 3\*3 | 3\*3 | convolution |
| 6 | 2 | 3 | 8 | 11 | 4 | 1 | 1 | Deliation Rate |
| 107\*107 | 71\*71 | 63\*63 | 51\*51 | 35\*35 | 13\*13 | 5\*5 | 3\*3 | Receptive field |

Formula of receptive field in each layer:

RF0 = 1 , RFi = RF i-1 + (k - 1) \* d

لذا قسمت بالا را با توجه به فرمول بالا به ترتیب پر می کنیم.

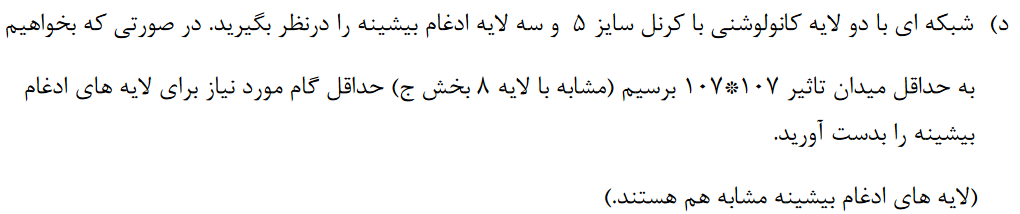
4\*2 + 5 = X → X = 13

13 + ( 2 \* X) = 35 → X = 11

35 + (2 \* 8) = X → X = 51

51 + (4 \* 3) = X → 63

71 + (X - 1) \* 5 = 107 → X = 7



پاسخ:

اگر مقدار dilation rate را 1 در نظر بگیریم و می دانیم هر لایه کانوولوشنی که کرنل 5 در 5 دارد به ابعاد 2 واحد اضافه می کند و Receptive field اولیه 1 \* 1 در نظر می گیریم. برای pooling نیز مقدار stride که همان خواسته سوال نیز می باشد مشخص می کند که از ابعاد به چه صورت اطلاعات برداریم برای مثال اگر stride = 4 باشد ابعاد ما ¼ شده و Receptive field ما 4 برابر می شود. با توجه به گفته سوال 3 لایه pooling و 2 لایه کانوولوشنی داریم پس به ترتیب 1 pooling سپس یک کانوولوشنی و ..... و در نهایت یک لایه pooling داریم: حال اگر stride = x بگیریم می دانیم که انتخاب 2 از 5 تا حالت داریم که برای هر کدام جواب به صورت زیر است:

Max-max-max-con-conv و x\*x\*x + 4 + 4 = x^3 + 8 = 107→ x = 5

max-max-conv-conv-max و (x\*x + 4 + 4)\*x = x^3 + 8x = 107 → x = 5

Max-conv-conv-max-maxو (x+4+4)\*x\*x = x^3 + 4x^2 = 107 → x = 4

Conv-conv-max-max-max: (x + 4 + 4)\*x \* x \*x = x^4 + 8\*x^3 = 107 → x = 3

Conv-max-conv-max-max و((x+4)\*x + 4)\* x \* x =x^2((x+2)^2)=107 → x = 3

max-Conv-max-conv-max و ((x + 4) \* x + 4) \* x = x(x+2)^2 = 107 → x = 4

max-max-Conv-max-conv و ((x \* x +4)\*x) + 4 = 107 → x = 5

Conv-max-max-conv-max و ((x + 4) \* x \* x + 4) \* x = 107 → x = 3

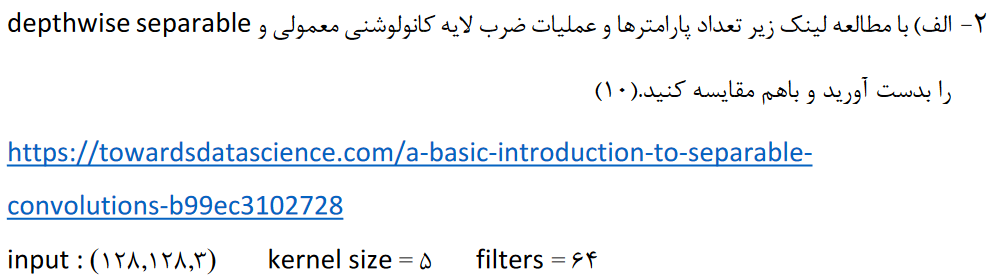
max-Conv-max-max-conv و (x + 4) \* x \* x + 4 = 107 → x = 4

Conv-max-max-max-conv و (x + 4)\*x \* x \*x + 4 = 107 → x = 3

با توجه به 10 حالت بالا و احتساب x متوجه شدیم که کمترین مقدار برای x مقدار 3 می باشد که حالت ها را برای ترتیب باید از چپ به راست بخوانیم.

(کران بالای همه ی جواب ها در نظر گرفته شده است.

2)



پاسخ:

**Regular Convolution**

1. Number of Parameters:

Total parameters: 5 x 5 x 3 x 64 = 4800

همانطور که گفته شده بایاس را در نظر نمی گیریم

تعداد عملیات ضرب در آن برابر با ضرب کانال ورودی در تعداد فیلتر ها در ابعاد فیلتر در ابعاد ورودی می باشد:

2. Number of Multiplications:

Output size: 128 x 128 x 64 x 5 x 5 x 3 = 75 multiplications.

Total multiplications: 128 × 128 x 64 x 5 x 5 x 3 = 78643200

البته این درصورتی هست که کانوولوشن باعث کمتر شدن ابعاد نشود وگرنه باید از فرمول 3\*64\*5\*5\*124\*124 تعداد ضرب ها را محاسبه کنیم.

**Depthwise Separable Convolution**

در ابتدا depth wise انجام شده که فیلتر کرنل عمقی به اندازه کل ورودی ندارد بلکه به صورت لایه لایه و با عمق 1 برای هر قسمت از لایه ورودی عملیات کانوولوشن را انجام می دهد .

در قسمت بعدی point wise انجام می شود به این صورت که ابعاد آن به صورت 1 در 1 و تعداد آن درست مانند تعداد فیلتر هایی هست که می خواستیم بر روی ورودی اولیه پیاده سازی کنیم پس عمق یکسان ولی طول و عرض متفاوت دارند.

1**. Depthwise Convolution:**

Each of the 3 input channels is convolved with its own 5 x 5 filter.

- Total parameters: 5 x 5 x 3 = 75

2. **Pointwise Convolution:** Each of the 3 outputs from the depthwise step is convolved with 64 1 x 1 filters.

Total parameters: 1x 1x 3 x 64 = 192

3. **Total Parameters:** Total: 75+192 = 267

4. **Number of Multiplications**:

**Depthwise Convolution:** Output size: 128 x 128 x 3

- Each output pixel requires 5 x 5 = 25 multiplications.

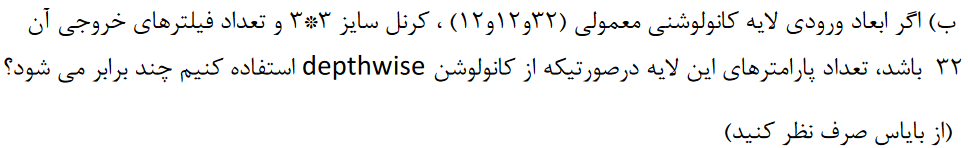
Total multiplications: 128 × 128 x 3 x 5 x 5 = 1228800

**Pointwise Convolution**:

Output size: 128 x 128 x 64 Each output pixel requires 3 multiplications.

Total multiplications: 128 × 128 x 64 x 3 = 3145728

- Total Multiplications: -1228800 + 3145728 = 4374528



پاسخ:

مانند عملیات های بالا محاسبه را با پارامترهای جدید انجام می دهیم و نسبت پارامترهای depth wise separable را به کانولوشن معمولی حساب می کنیم که داریم:

[ 3 \* 3 \* 32 ( depth wise) + 1 \* 1 \* 32 \* 32] / [ 3 \* 3 \* 32 \* 32] =

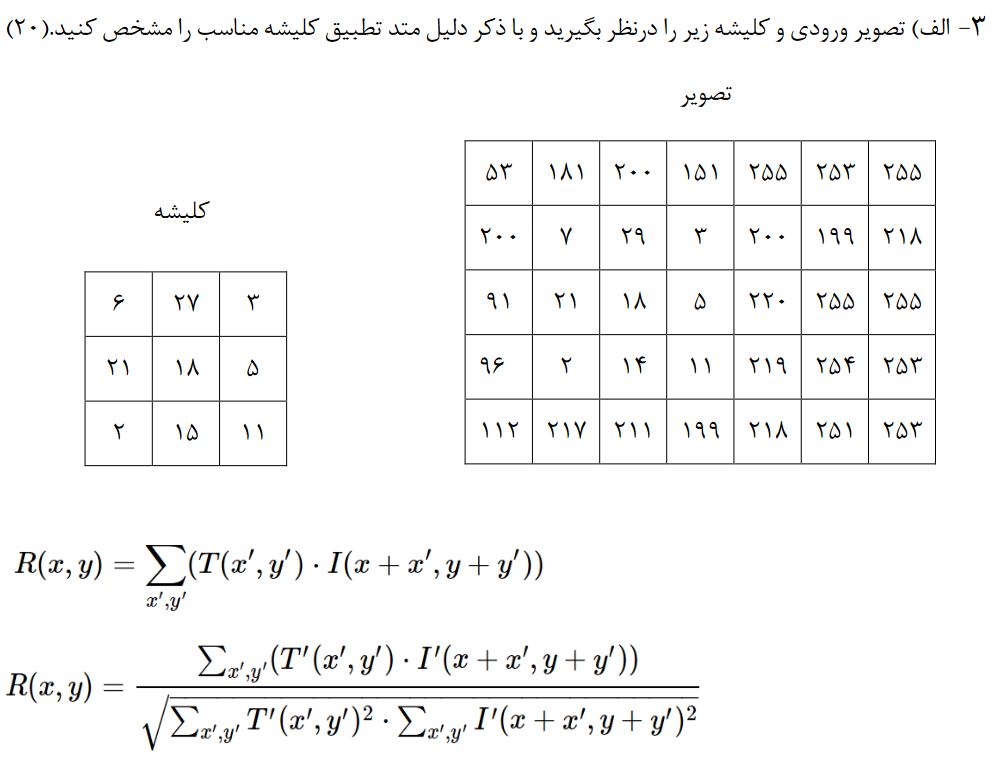
[288 + 1024] / 9216

(3 \* 64 \* 5 \* 5 \* 128 \* 128) / (3 \* 5 \* 5 \* 128 \* 128 + 3 \* 64 \* 128 \* 128)

= (288 + 1024) / 9216 = %14

این بدان معناست که حدود 7 برابر از تعداد پارامتر ها کاهش یافته است یا به عبارتی 86 درصد از آن کم شده است.

3)



پاسخ:

در تطبیق قالب (Template Matching) دو روش اصلی وجود دارد:

1. روش مجموع مربعات اختلافات (SSD).

2. روش همبستگی متقابل نرمال‌شده (NCC).

روش NCC با نرمال‌سازی مقادیر پیکسل‌ها، تأثیر تغییرات روشنایی و کنتراست تصویر را کاهش می‌دهد و باعث می‌شود مکان‌یابی قالب دقیق‌تر باشد از آنجا که در تصویر داده شده تفاوت پیکسل ها فاحش است و ما می خواهیم pattern های آنها را در نظر بگیریم تا تفاوت شدت روشنایی و تاریکی لذا روش اول بهتر است. در این روش، میانگین و انحراف معیار پیکسل‌ها محاسبه می‌شود و همبستگی بر این اساس انجام می‌گیرد، بنابراین الگوی اصلی در تصویر و قالب مورد توجه قرار می‌گیرد.

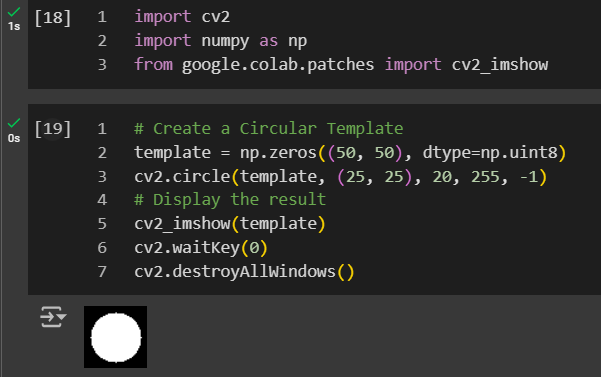
در مقایسه با روش SSD، روش NCC نسبت به تغییرات روشنایی و کنتراست مقاوم‌تر است. در روش SSD، نواحی با روشنایی زیاد ممکن است به اشتباه به عنوان تطابق قوی در نظر گرفته شوند. اما در روش NCC، به دلیل نرمال‌سازی، نواحی با شدت روشنایی بالا تأثیر زیادی نمی‌گیرند و الگوهای واقعی بهتر تشخیص داده می‌شوند. به‌علاوه، خروجی روش NCC نرمالایز شده و قابل مقایسه است، در حالی که در روش SSD این‌گونه نیست و همچنین استفاده از NCC باعث می‌شود مکان‌یابی قالب دقیق‌تر شود، حتی اگر روشنایی و کنتراست تصویر تغییر کند. NCC مقادیر پیکسل‌ها را نرمال‌سازی می‌کند، به این معنی که میانگین و انحراف معیار پیکسل‌ها را محاسبه و سپس همبستگی را اندازه‌گیری می‌کند. این فرآیند تأثیر تغییرات روشنایی و کنتراست را کاهش داده و فقط الگوی اصلی در تصویر و قالب مورد توجه قرار می‌گیرد.

در فرمول داده شده، T’ و I’ نسخه‌های نرمال‌شده قالب و تصویر هستند. بنابراین، استفاده از NCC برای تصویر موردنظر مناسب‌تر است زیرا در برابر تغییرات روشنایی و کنتراست مقاوم‌تر است و دقت بالاتری در تطبیق قالب ارائه می‌دهد.

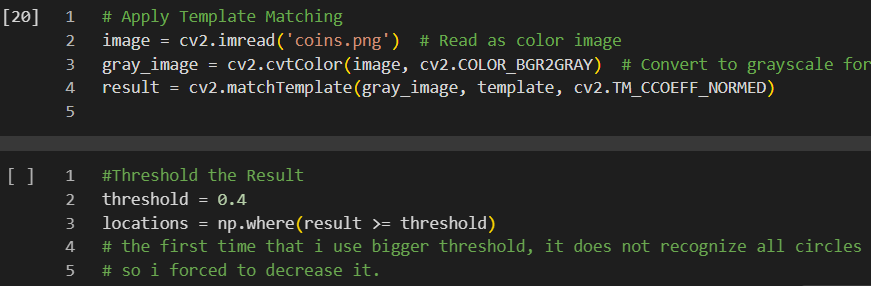


پاسخ:

در ابتدا کتابخانه های مورد نظر خود را ادد می کنیم و سپس template دایره ای خود را با توجه به دایروی بودن سکه ها شناسایی می کنیم که کد آن به صورت زیر است:



سپس تصویر خود را با دستور cv2.imread می خوانیم و سپس آن را به gray تبدیل می کنیم و بعد به سراغ matchTemplate برای روش تطبیق کلیشه می رویم و آستانه ای تعریف می کنیم تا در صورت بالاتر بودن از آن به عنوان جواب به ما بازگردانده شود. که من در ابتدا 0.8 در نظر گرفته بودم اما تقریبا فقط نیمی از سکه ها شناسایی شدند و هر بار مقداری از آستانه کم کردم تا در اخر با آستانه = 0.4 همه ی دایره ها شناسایی شدند. که کد آن نیز به صورت زیر است:

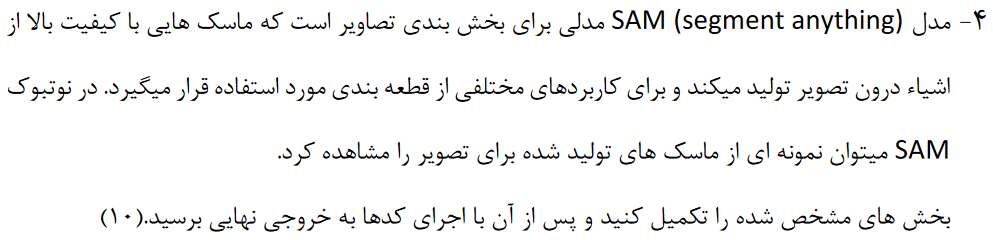


حال مکان هایی که در شرط ما برقرار هستند را با یک دایره ی سبز رنگ نشان می دهیم که نتیجه ی آن در زیر مشخص است:



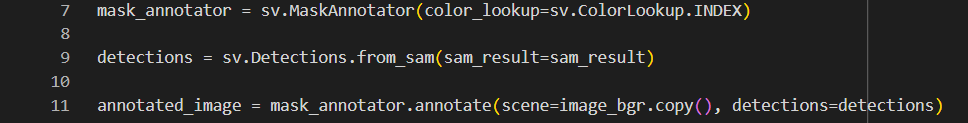
با توجه به تصویر می بینیم که همه ی coin ها تشخیص داده شده اند

4)



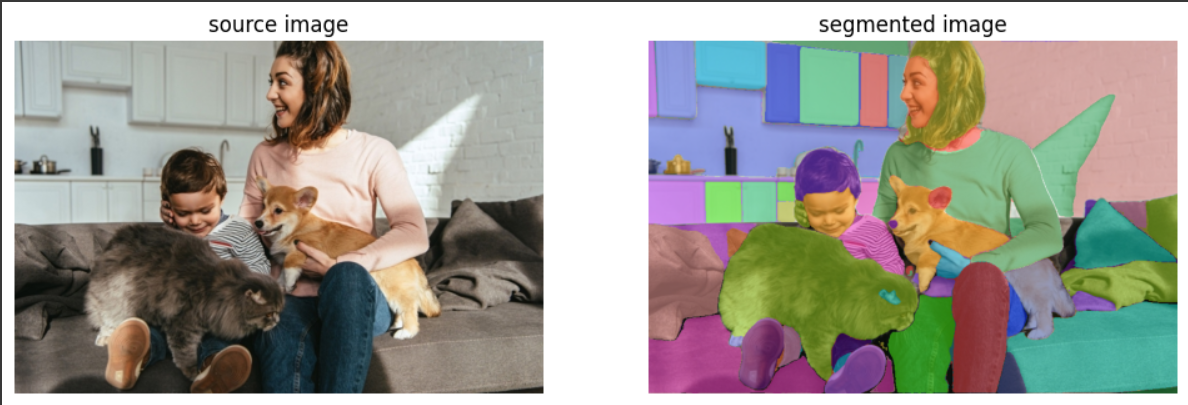
پاسخ:

برای حل این سوال همانطور که در نوتبوک نیز آورده شده است پس از download و import کتابخانه های مورد نظر image خود را از حافظه ی colab می خوانیم و سپس تصویر خود را به bgr و rgb تبدیل می کنیم.

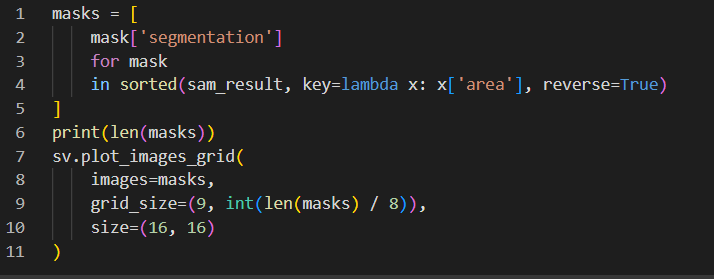
ا

این خط از کد برای تبدیل نتایج خروجی مدل Sam به نمونه های شناسایی استفاده می شود. From\_sam یک متد از کلاس Detections در کتابﺨانه supervision است که نتایج ماسکهای شناسایی شده توسط Sam را به فرمت قابل استفاده در این کتابﺨانه تبدیل می کند.

همچنین این کد برای افزودن حاشیه ها و برچسب ها به تصویراصلی با استفاده از ماسکها و نمونه های شناسایی شده استفاده میشود. Annotate یک متد از کلاس MaskAnnotator است که تصویر ورودی را با ماسکها و رنگهای مشخص شده تزیین میکند. که نتیجه آن نیز در زیر مشخص است.

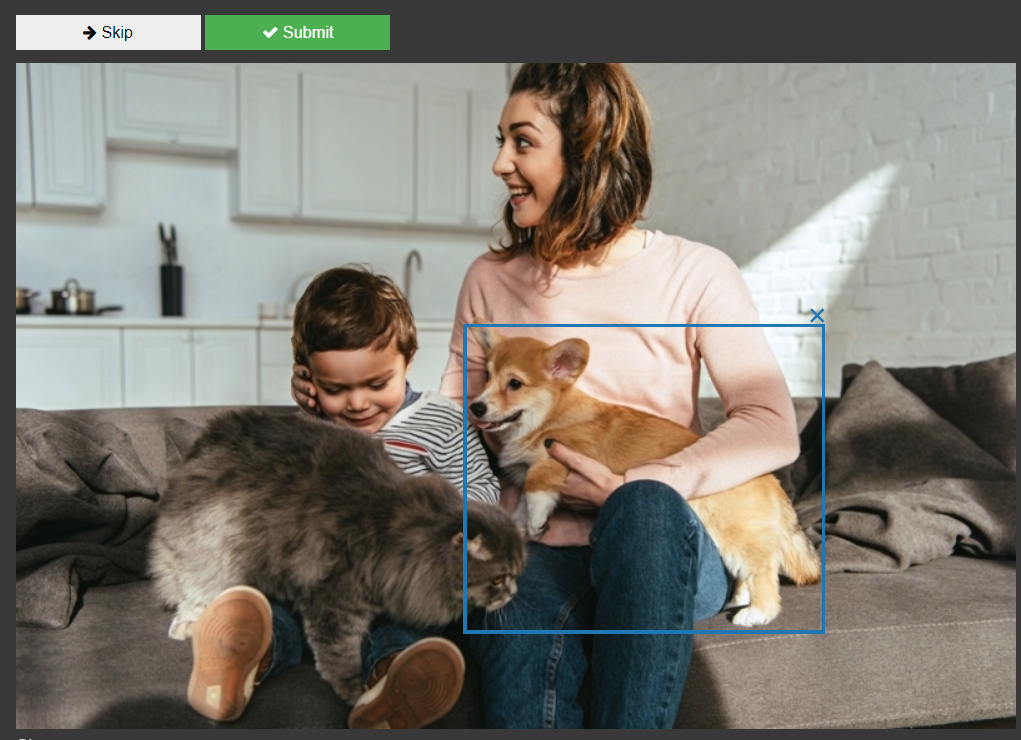


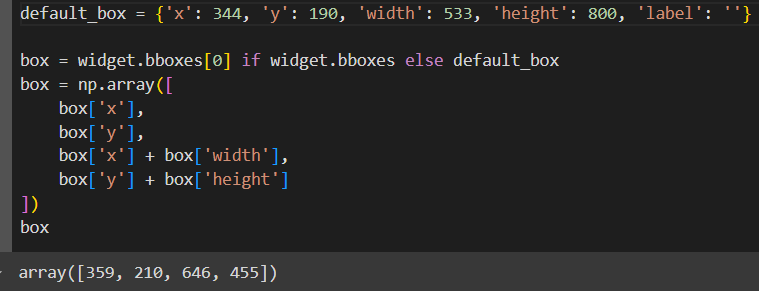
با این کد نیز mask قسمت های شناسایی شده را می بینیم:



که در خروجی متوجه می شویم که کد ما 82 شکل مجزا را تشخیص داده است که برخی از آن ها به شکل زیر هستند:



حال پس از انتخاب محدوده ی سگ در cell های بعدی و زدن دکمه accept بر روی آن که به صورت کد زیر است می توانیم سگ را در تصویر detect کنیم که کد آن به صورت زیر است:



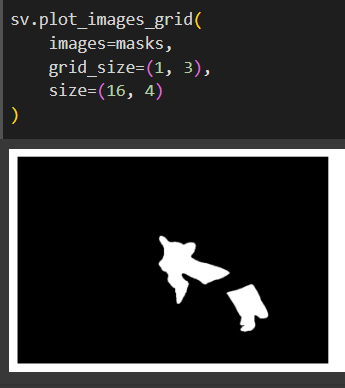
که یک box دیفالت نیز برای آن در نظر گرفته ایم که در صورت تشخیص ندادن به تشخیص آن بپردازد.

این کد برای استخراج اولین جعبه محدود کننده bounding box از ویجت BBoxWidget استفاده میشود. Widget.bboxes لیستی از جعبه های محدود کننده است که کاربر به صورت تعاملى ازطریق ویجت رسم کرده است. این خط اولین جعبه را انتخاب میکند.

همچنین این کد برای تبدیل جعبه محدود کننده به یک آرایه numpy استفاده می شود. جعبه محدود کننده به صورت یک دیکشنری با کلیدهای x, y, width, و height ذخیره شده است. این کد مختصات باﻻ-چپ و پایین-راست جعبه را

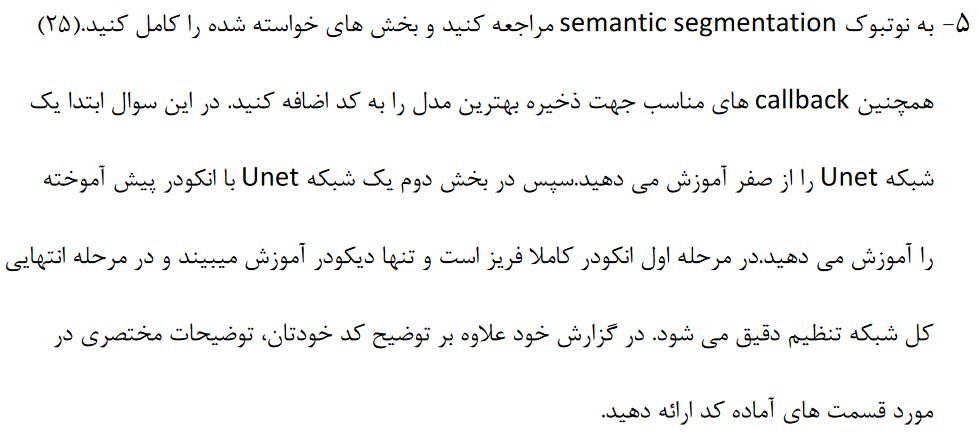
محاسبه کرده و آنها را به صورت یک آرایه numpy ذخیره میکند و در آخر خروجی تصویر به صورت زیر می باشد:

و همانطور که در کد نیز دیده می شود mask های سگ در تصویر نیز در قسمت آخر نوتبوک موجود است. که به صورت زیر است:



که در تصویر بالا سگ به خوبی مشاهده می شود.

5)



پاسخ:

اضافه کردن callback های مناسب به کد شبکه UNet می‌تواند به بهبود عملکرد، جلوگیری از overfitting، و تسریع فرآیند آموزش کمک کند. در اینجا چند callback مهم برای Keras یا TensorFlow معرفی شده‌اند.

### **1. EarlyStopping**

این callback به شما کمک می‌کند که آموزش را زمانی که بهبودی در اعتبارسنجی دیده نمی‌شود، متوقف کنید.

### **2. ModelCheckpoint**

این callback بهترین مدل را بر اساس معیار مشخص شده ذخیره می‌کند.

### **3. ReduceLROnPlateau**

این callback مقدار نرخ یادگیری (learning rate) را در صورت عدم بهبود معیار مشخص شده کاهش می‌دهد.

### **4. TensorBoard**

این callback به شما امکان می‌دهد که آموزش مدل را به صورت گرافیکی در TensorBoard مشاهده کنید.

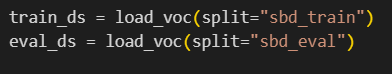
در این کد:

* از ModelCheckpoint استفاده شده است تا بهترین مدل‌ها بر اساس معیار val\_jaccard\_coef ذخیره شوند.
* دو کال‌بک checkpoint\_callback و checkpoint\_callback1 برای دو مدل مختلف تعریف شده‌اند تا هر کدام به صورت جداگانه ذخیره شوند.

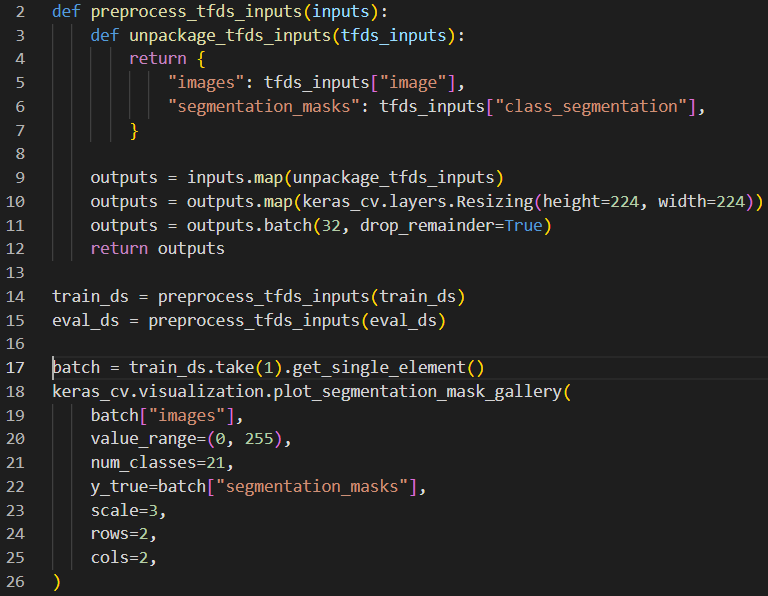
توضیحات کد:

پس از import کردن کتابخانه های مورد نظر، در ابتدا به **بارگذاری و پیش‌پردازش داده‌ها می پردازیم.**

در این قسمت، داده‌های آموزشی و ارزیابی از مجموعه داده‌های VOC بارگذاری می‌شوند.



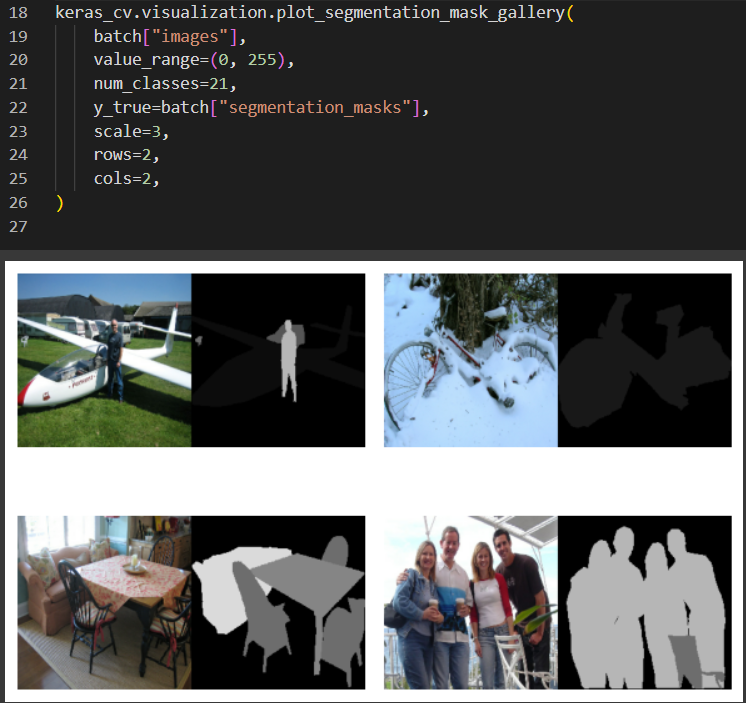
سپس تابع پیش پردازشی تعریف می کنیم به شرح زیر است:



این تابع داده‌ها را از فرمت اصلی خود جدا کرده و تصاویر را به اندازه ۲۲۴ در ۲۲۴ تغییر داده و سپس آنها را به دسته‌های ۳۲ تایی تقسیم می‌کند.

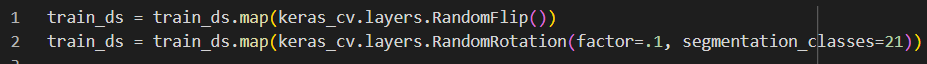


همانطور که در بالا دیده می شود این خطوط کد، تابع پیش‌پردازش را بر روی داده‌های آموزشی و ارزیابی اعمال می‌کند.

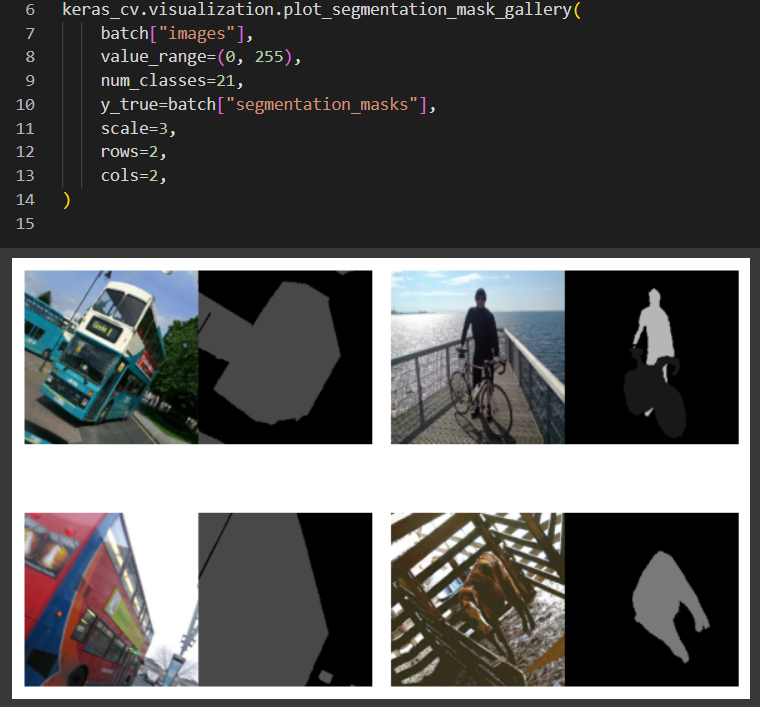
سپس نمایش نمونه هایی از داده ها را داریم: 

که چند نمونه از تصاویر و ماسک‌های سگمنتیشن را نمایش می‌دهد.

سپس چند نمونه ی تصادفی به نمونه های خود اضافه می کنیم که این خطوط کد تغییرات تصادفی مانند چرخش و وارونه‌سازی به داده‌های آموزشی اضافه می‌کنند.



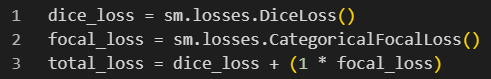
حال دوباره به visualize کردن نمونه های خود می پردازیم که بخشی از آن ها به صورت زیر هستند:



سپس به تعریف مدل U-Net می پردازیم که تحت عنوان فانکشن unet\_model تعریف شده است.

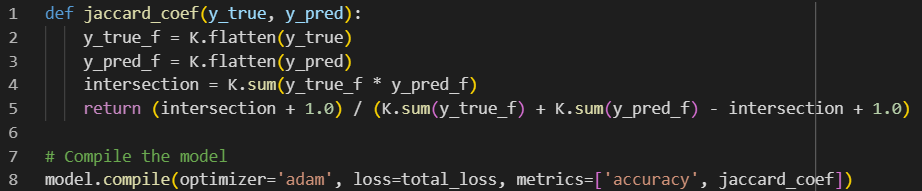
این کد یک مدل U-Net ساده را تعریف می‌کند که شامل لایه‌های کانولوشن، ماکس‌پولینگ و کانولوشن معکوس است و در نوتبوک تعریف شده است.

سپس به تعریف خطاها و زیان ها می پردازیم.



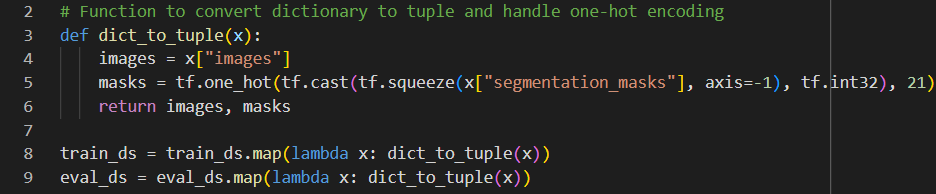
در این قسمت، زیان‌های Dice و Focal تعریف می‌شوند و مجموع آنها به عنوان زیان نهایی تعیین می‌شود.

سپس یک متریک سفارشی تعریف می کنیم که این متریک سفارشی به نام jaccard\_coef تعریف می‌شود که برای ارزیابی مدل استفاده می‌شود.



در مرحله ی بعدی مدل با استفاده از زیان‌های تعریف‌شده و متریک‌ها کامپایل می‌شود.

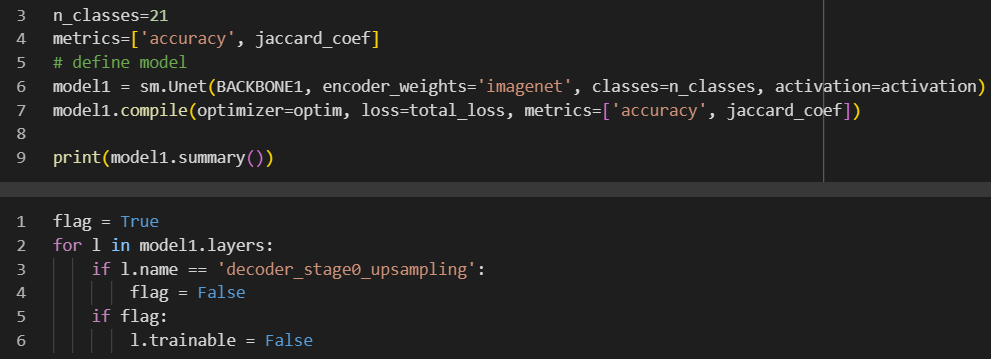
در این قسمت تبدیل داده‌ها به tuple و اعمال one-hot encoding را داریم.

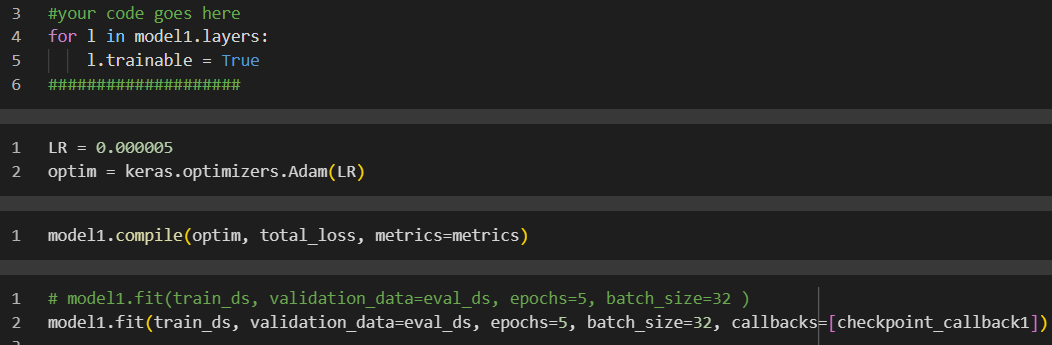


این قسمت داده‌ها را به فرمت tuple تبدیل می‌کند و ماسک‌ها را به one-hot encoding تبدیل می‌کند. سپس از call\_back نام برده شده استفاده کرده ایم که این کال‌بک بهترین مدل را بر اساس معیار val\_jaccard\_coef ذخیره می‌کند و سپس مدل خود را با ویژگی های گفته شده آموزش می دهیم و از دستور fit برای این کار استفاده می کنیم. که پارامترهای آن به صورت زیر است:



حال به آموزش مدل دوم با MobileNetV2 می پردازیم.



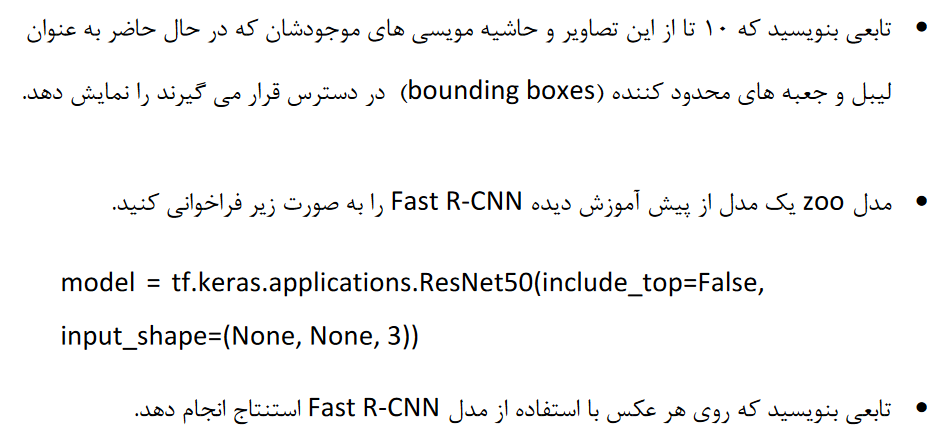
این بخش یک مدل U-Net با MobileNetV2 به عنوان backbone تعریف می‌کند و تا لایه decoder\_stage0\_upsampling، لایه‌ها را غیرقابل آموزش می‌سازد و در ادامه call\_back تعریف می کنیم تا برای مدل دوم نیز مانند مدل اولین بهترین مدل را ذخیره کند و در ادامه به آموزش مدل اول با پارامترهای موجود می پردازیم. در قسمت آخر نیز همه ی لایه ها را قابل آموزش می کنیم.

در این بخش، همه لایه‌های مدل دوباره قابل آموزش قرار داده می‌شوند و مدل دوباره کامپایل و آموزش داده می‌شود.

در آخر نیز به دلیل پیام یکی از دستیاران استاد در گروه تلگرامی و کمبود وقت این سوال ران نشده است و در صورت خواستن ارائه آن را به شما نشان خواهم داد.

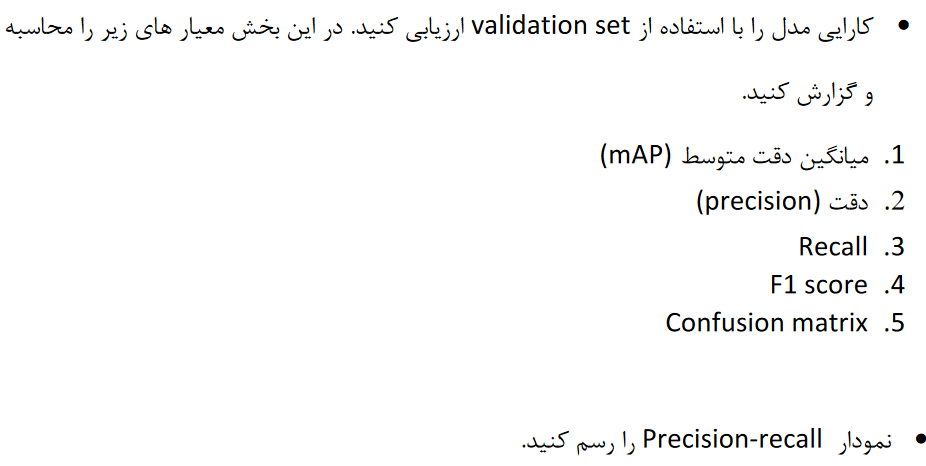
6)





پاسخ:

ای بابا! حسش نیست.



پاسخ: