به نام خالق رنگین کمان

ستاره باباجانی – گزارش تمرین سری هفتم

سوال 1: الف)

Dilated Kernel Size = k + (k - 1) \* (d - 1)

ب) تعداد پارامترهای قابل آموزش در یک لایه کانولوشن توسط اندازه کرنل و تعداد کانال های ورودی و خروجی تعیین می شود، نه توسط dilation rate. سه برابر شدن این نرخ بر receptive field تأثیر می گذارد اما تعداد پارامترهای قابل آموزش را تغییر نمی دهد.

ج) همانطور که میدانیم فرمول receptive field برای هر لایه به شرح زیر است:

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

پس جدول به شرح زیر میشود:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | Dilation rate |
| 107\*107 | 71\*71 | 63\*63 | 51\*51 | 35\*35 | 13\*13 | 5\*5 | 3\*3 | Receptive field |

د) اگر فرض کنیم pool size و stride یکی باشند، در لایه کانولوشنی خواهیم داشت:

Receptive\_out = Receptive\_in + (kernal\_size - 1) \* Dilation\_rate

و در لایه max pooling خواهیم داشت:

Receptive\_out = Receptive\_in + (stride - 1) \* Stride\_Of\_Previous\_Layer (call J)

پس برای لایه‌ها به ترتیب خواهیم داشت:

1. First conv: = 1 + (5 - 1) \* 1 = 5
2. Second conv: = 5 + (5 - 1) \* 1 = 9 => so the receptive field is 9\*9
3. First max-pooling: = 9 + (s - 1) \* 1 = 8 + s
4. Second max-pooling: J = 1 \* s, so: 8 + s + (s - 1) \* s = 8 + s^2
5. Third max-pooling: J = s \* s, so: 8 + s^2 + (s - 1) \* (s^2) = 8 + s^3

So => 8 + s^3 >= 107, s = 5

سوال 2: الف) در کانولوشن معمولی داریم:

* Number of parameters = kernel\_size \* kernel\_ size \* input\_channels \* output\_channels

So: 5 \* 5 \* 3 \* 64 = 4800

* Operations per filter = input\_width \* input\_height \* kernel\_size \* input\_channels

So: 3 \* 64 \* 5 \* 5 \* 128 \* 128 = 78,643,200

در Depthwise Separable Convolution داریم:

* Number of parameters (depthwise) = kernel\_size \* kernel\_size \* input\_channels

So: 5 \* 5 \* 3 = 75

Number of parameters (pointwise) = 1 \* 1 \* input\_channels \* output channels

So: 1 \* 1 \* 3 \* 64 = 192

Total: 75 + 192 = 267

* Depthwise Convolution Operations = output\_size \* operations\_of\_each\_output\_pixel

So: 128 \* 128 \* 75 = 1,228,800

Pointwise Convolution Operations = output\_size \* operations\_of\_each\_output\_pixel

So: 128 \* 128 \* 3 \* 64 = 3,145,728

Total: 1,228,800 + 3,145,728 = 4,374,528

که همانطور که مشاهده میشود Depthwise separable convolution بطور قابل توجهی تعداد پارامترها و تعداد عملیات ضرب را کاهش می دهد.

ب) در این بخش تعداد پارامتر هرکدام را طبق فرمول‌های بالا محاسبه کرده و سپس نسبت میگیریم:

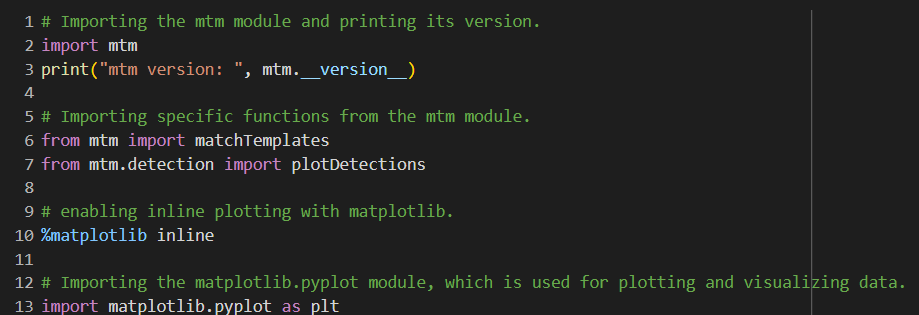
* Regular convolution: 3 \* 3 \* 32 \* 32 = 9216
* Depthwise Separable Convolution = 3 \* 3 \* 32 + 1 \* 1 \* 32 \* 32 = 1312

که نسبت آنها برابر با 0.142 میشود. (9216 / 1312)

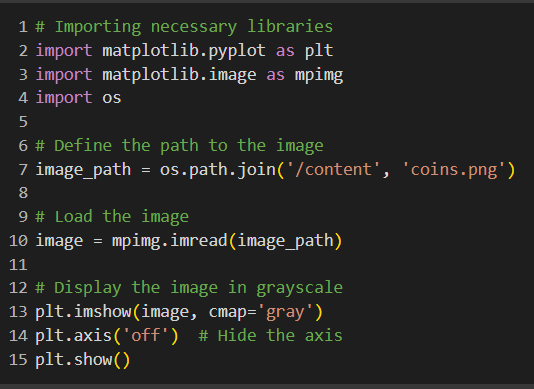
سوال 3: الف) جواب متد دوم که NCC است، است. همانطور که مشاهده میشود در متد اول مقادیر نرمالیزه نمیشوند اما در متد دوم، پیکسل‌های تصویر و کلیشه نرمالیزه میشوند که این به میزان صحت و robustness مدل می‌افزاید. همچنین طبق مقادیر پیکسل‌های تصویر، بازه گسترده‌ای از مقادیر داریم پس استفاده از این متد، بهتر است.

ب) مراحل کد زده شده به شرح زیر است:

1. نصب کتابخانه مورد استفاده: همانطور که مشاهده میشود از کتابخانه mtm استفاده شده است.

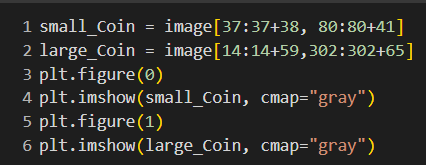


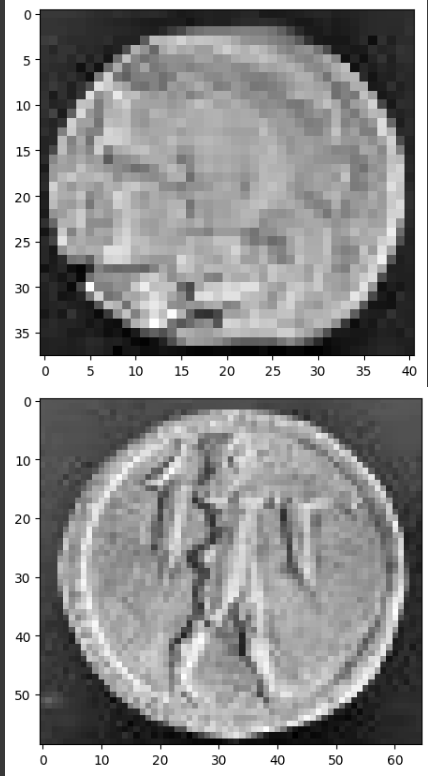
1. نمایش عکس داده شده:



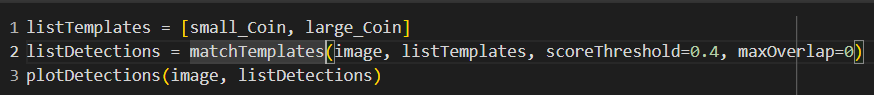


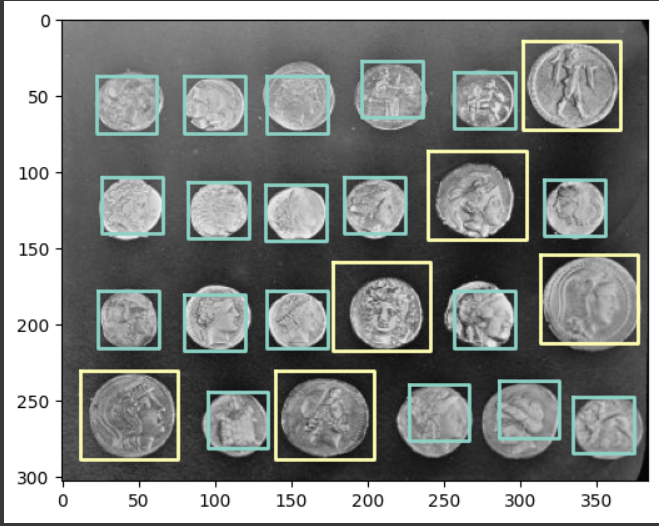
1. ساخت تصویر الگو: برای جستجو در تصویر به تعدادی تصویر الگو نیاز داریم که در کد زیر دوتا از سکه‌ها (یکی بزرگ و دیگری کوچک) بعنوان الگو درنظر گرفتیم:





1. حال با استفاده از تابع matchTemplates و تصاویر الگو و مقدار threshold کل سکه‌های تصویر داده شده را می‌یابیم:

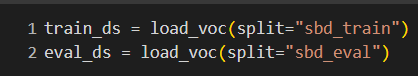




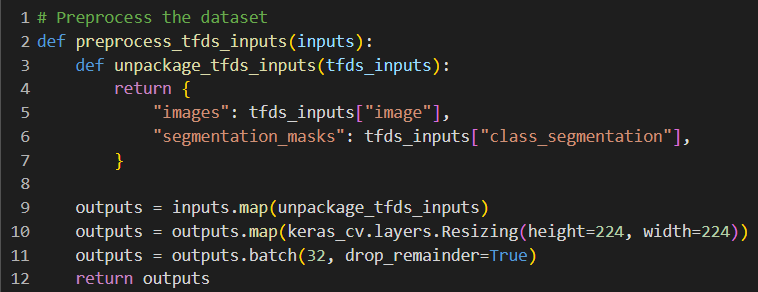
سوال 4: نوتبوک داده شده تکمیل شد.

سوال 5: مراحل کد خواسته شده به شرح زیر است:

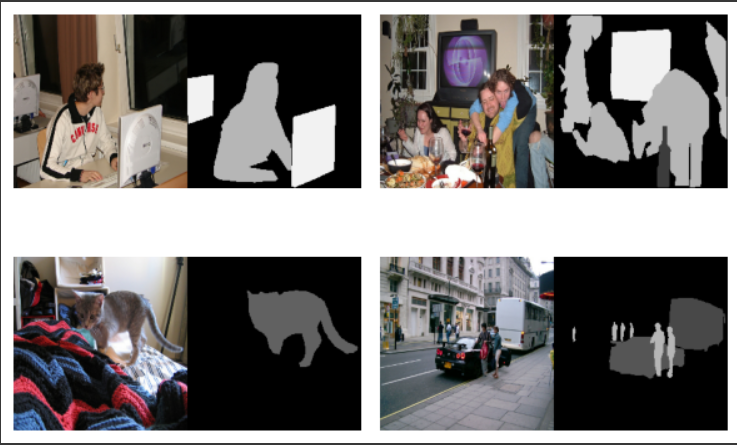
* ابتدا کتابخانه‌ها و پکیج‌های مورد نیاز را نصب کردیم.
* دیتاست PasCal را لود کردیم:



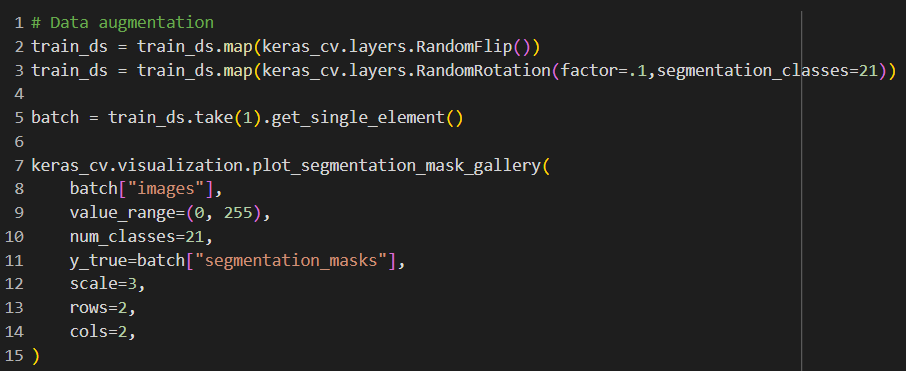
* پیش پردازش دیتا: با resize و batch کردن:



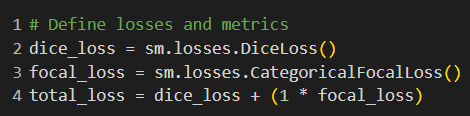
* نشان دادن نمونه‌ دیتا از دیتاست:



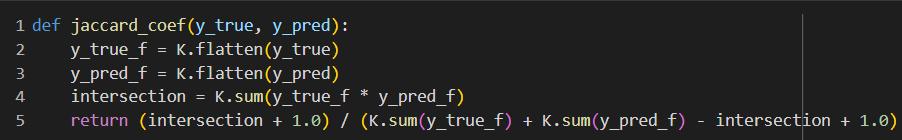
* چند تکنیک data augmentation مثل random flipping و random rotation روی داده‌ها انجام شد:



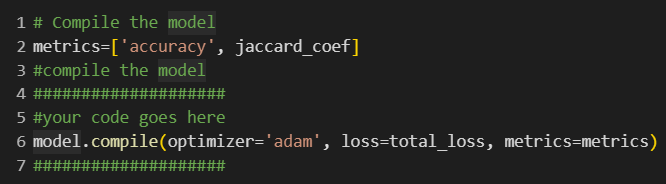
* طراحی شبکه Unet از صفر: مدل شامل encoder که از چندین بلوک شامل دو لایه کانولوشن و به دنبال آن یک لایه حداکثر ادغام و decoder که رمزگذار را منعکس می کند اما حداکثر ادغام را با unsampling جایگزین می کند و bridge برای اتصال آن دو، تشکیل شده است.
* تعریف توابع ضرر:



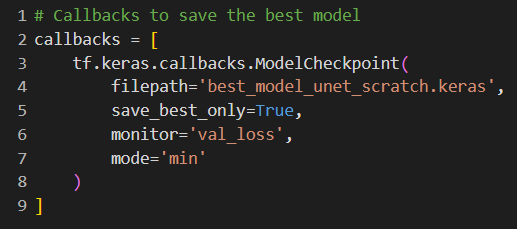
* تعریف متریک جدید: یک معیار دلخواه برای ارزیابی عملکرد یک مدل تقسیم‌بندی تصویر است. این شاخص Jaccard را محاسبه می‌کند که به نام IoU نیز شناخته می‌شود، که یک معیار پرکاربرد در وظایف تقسیم‌بندی تصویر است.



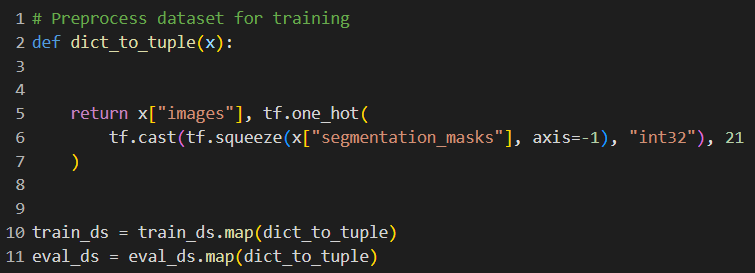
* کامپایل کردن مدل: خلاصه مدل در کدها آورده شده است.



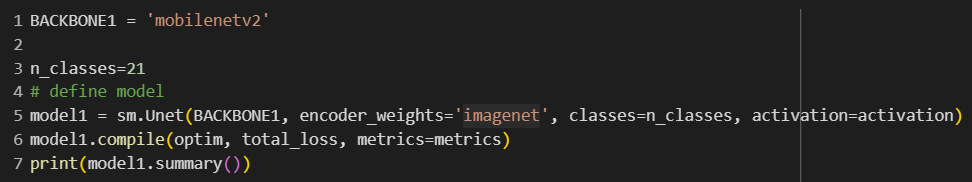
* اضافه کردن call back برای ذخیره بهترین مدل:



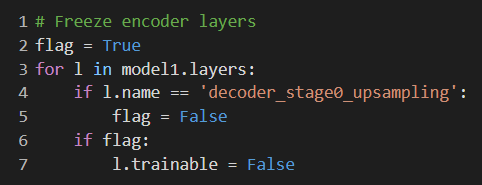
* سپس با مرحله بعدی از پیش پردازش داده، مطمئن میشویم که دیتا دارای فرمت درست برای مدل ما است و همچنین one-hot vectors میسازیم:



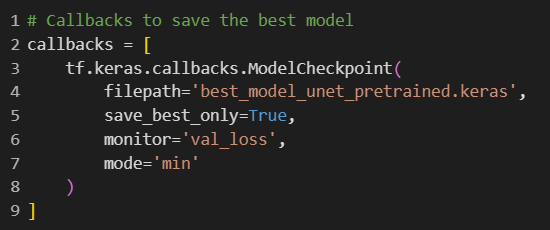
* آموزش مدل
* حال سراغ مدل از پیش آموزش دیده شده میرویم که رمزگذار mobilenetv2 دارد:



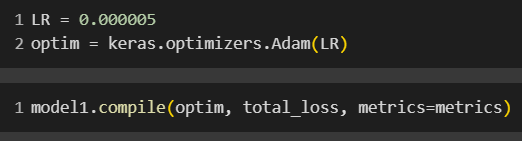
* ابتدا encoder فریز شده و فقط decoder آموزش میبیند: خلاصه مدل در کد قابل مشاهده است.



* اضافه کردن call back برای ذخیره بهترین مدل:



* آموزش مدل
* پس از آموزش اولیه، تمام لایه‌ها منجمد می‌شوند و کل مدل با نرخ یادگیری پایین‌تر تنظیم می‌شود:



* آموزش نهایی مدل