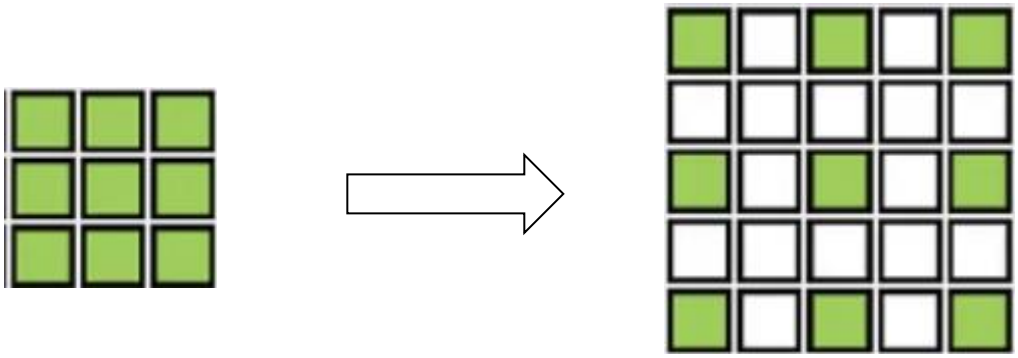
به نام خدا

تمرین هفتم- بینایی کامپیوتر **نیایش خانی ۹۹۵۲۱۲۳۵**

سوال ۱. الف) با مطالعه لینک قرار شده شده در داکیومنت سوال ، در می یابیم که در یک لایه کانولوشنی، d فاصله بین المنت های کرنل را کنترل می کند و طبیعتا سایز کرنل k*k را افزایش می دهد.
برای مثال اگر k=۳ و d=۲ در نظر بگیریم، فاصله میان المنت ها +۱ می شود.



بنابراین اگر d را dilation rate در نظر بگیریم، فاصله میان هر المنت باید به علاوه d-۱ شود و اگر ابعاد کرنل اصلی ۳*۳ باشد، dilated kernel بر حسب d برابر $(2 \times d) + 1$ است.
رابطه کلی بدست آوردن ابعاد کرنل به این صورت خواهد بود :

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

همانطور که مشاهده می شود، اگر k=۳ و d=۲ باشد، ابعاد کرنل از طریق رابطه بالا ، ۵*۵ خواهد بود.

ب) تعداد پارامترهای قابل آموزش در یک لایه کانولوشن با اندازه هسته و تعداد کانال های ورودی و خروجی تعیین می شود.

$$Number\ of\ parameters = k^2 \times C_{in} \times C_{out}$$

Dilation rate بر فاصله بین عناصر در کرنل تأثیر می گذارد اما تعداد عناصر (پارامترها) در خود کرنل را تغییر نمی دهد. زیرا صرف نظر از dilation rate ، کرنل همچنان دارای k*k عنصر است و پارامترهای هر کرنل همچنان طبق فرمول بالا به دست می آیند. بنابراین، اگر dilation rate سه برابر شود، تعداد پارامترهای قابل آموزش لایه کانولوشن تغییر نمی کند.

ج) فرمول به دست آوردن receptive field به این صورت است :

$$RF_{current} = RF_{previous} + (k-1) \times d$$

بنابراین برای به دست آوردن A خواهیم داشت :

$$RF = 5 + 2 \times 4 = 13 \;\; -> A = 13 * 13$$

به همین صورت بقیه موارد را هم به دست می آوریم.

$$RF = 13 + 2 \times B = 35 \;\; -> B = 11$$
$$RF = 35 + 2 \times 8 = 51 \;\; -> C = 51 * 51$$
$$RF = 51 + 4 \times 3 = 63 \;\; -> D = 63 * 63$$
$$RF = 71 + (E-1) \times 6 = 107 \;\; -> E = 7 -> E * E = 7 * 7$$

د) receptive field لایه کانولوشنی را در قسمت ج و برای لایه max pooling داریم :

$$RF_{current} = RF_{previous} + (k-1) \times stride$$

ابتدا در نظر می گیریم که receptive field اولیه ، ۱ است. بنابراین :

$$R_{init} = 1$$

پس از اولین کانولوشن :

$$R_1 = 1 + (5-1)$$

بنابراین از آنجایی که طبق فرض pool size و stride برابر هستند و سه لایه max pooling داریم، خواهیم داشت :

$$1 + ((5-1) \times 1) + ((5-1) \times 1) + ((x-1) \times x) \geq 107$$
$$(x-1) \times x \geq 98$$

با حل این نامعادله به این نتیجه میرسیم که کمترین مقدار ممکن برای stried برابر است با $x = 11$ است.

سوال ۲. الف)کانولوشن معمولی : سایز کرنل ۵*۵، برای هر فیلتر تعداد پارامتر ها ۵*۵*۳ است بنابراین برای ٦٤ فیلتر خواهد بود (صرف نظر از بایاس) :

$$parameters = k \times k \times input \times channels \times filter = 5 \times 5 \times 3 \times 64 = 4800$$

محاسبه تعداد عملیات ضرب : هر فیلتر به صورت مکانی روی کل ورودی اعمال می شود و خروجی ۱۲۸*۱۲۸ تولید می کند، برای هر موقعیت، ضرب ۵*۵*۳ وجود دارد و این برای هر یک از ٦٤ فیلتر اتفاق می افتد.
بنابراین، تعداد کل عملیات ضرب برابر است با:

$$128 \times 128 \times 5 \times 5 \times 3 \times 64 = 78643200$$

کانولوشن Depthwise Separable : این کانولوشن از نظر عمقی ترکیبی از دو لایه است:

۱. Depthwise Convolution : یک فیلتر کانولوشنال را در هر کانال ورودی اعمال می کند (طبق فرض سوال ۳ کانال).
۲. Pointwise Convolution : از کانولوشن ۱*۱ برای ترکیب خروجی‌های depthwise convolution در ٦٤ کانال خروجی استفاده می‌کند.

تعداد پارامترها در depthwise : هر کانال دارای یک فیلتر ۵*۵ است، تعداد پارامترها در هر کانال ۵*۵ = ۲۵. بنابراین تعداد کل پارامترها :

$$parameters = 5 \times 5 \times 3 = 75$$

تعداد پارامترها در pointwise : هر کرنل آن ۱*۱ است و ۳ کانال ورودی وجود دارد، تعداد پارامترها در هر فیلتر: ۳*۱*۱. بنابراین تعداد کل پارامترها برای ٦٤ فیلتر:

$$parameters = 1 * 1 * 3 * 64 = 192$$

بنابراین، تعداد کل پارامترها در این نوع کانولوشن برابر است با ۷۵+۱۹۲ = ۲٦۷

تعداد عملیات ضرب در depthwise : هر فیلتر ۵*۵ به ورودی ۱۲۸*۱۲۸ برای هر یک از ۳ کانال اعمال می شود. بنابراین خواهیم داشت :

$$128 \times 128 \times 5 \times 5 \times 3 = 1228800$$

تعداد عملیات ضرب در pointwise : هر فیلتر ۱*۱ روی خروجی کانولوشن عمقی اعمال می شود که دارای ابعاد یکسان (۱۲۸*۱۲۸) اما ۳ کانال است. -با در نظر گرفتن تعداد ضرب در هر فیلتر خواهیم داشت :

$$128 \times 128 \times 3 \times 64 = 3145728$$

بنابراین، تعداد کل عملیات ضرب برابر است با ۱۲۲۸۸۰۰+۳۱٤۵۷۲۸ = ٤۳۷٤۵۲۸

همانطور که مشاهده می شود تعداد پارامترها در depthwise separable بسیار کمتر شده است. همچنین تعداد عملیات ضرب نیز کاهش قابل توجهی داشته است. این مقایسه نشان می‌دهد که کانولوشن depthwise separable از نظر پارامترها و هزینه محاسباتی بسیار کارآمدتر است.

ب) در کانولوشن معمولی تعداد پارامتر ها برابر است با :

$$parameters = 3 \times 3 \times 32 \times 32 = 9216$$

در کانولوشن depthwise separable خواهیم داشت :

$$Depthwise\ parameters = 3 \times 3 \times 32 = 288$$
$$Pointwise\ parameters = 1 \times 1 \times 32 \times 32 = 1024$$
$$Total = 288 + 1024 = 1312$$

چند برابر شدن ؟

$$\frac{Depthwise\ Separable\ parametrs}{Normal\ parameters} = \frac{1312}{9216} = 0.142$$

سوال ۳. الف) متد اول Simple Cross-Correlation است که این روش مجموع محصولات قالب و پچ تصویر را به طور مستقیم محاسبه می کند. تغییرات در شدت و مقیاس بین الگو و بخش‌های مختلف تصویر را در نظر نمی‌گیرد. همچنین به مقادیر شدت مطلق در تصویر حساس است. اگر شرایط نور یا کنتراست تغییر کند، نتایج می تواند به طور قابل توجهی تحت تاثیر قرار گیرد.
اما متد دوم الگو و پچ تصویر را با کم کردن میانگین آنها و تقسیم بر انحراف استاندارد آنها نرمال می کند. این نرمال سازی متد را در برابر تغییرات نور و کنتراست قوی می کند و معیار قابل اعتمادتری از شباهت ارائه می دهد. بنابراین متد دوم Normalized Cross-Correlation انتخاب بهتری است زیرا با در نظر گرفتن تغییرات در نور، کنتراست و مقیاس شدت بین الگو و بخش‌های مختلف تصویر، اندازه‌گیری قابل اعتمادتر و قوی‌تری از شباهت ارائه می‌دهد که به تطبیق دقیق تر و سازگارتر الگو می شود.

ب) بعد از ایمپورت کردن لایبرری های لازم و خواندن عکس، فانکشنی تهیه شده است که ما می توانیم به صورت دستی تمپلیت مورد نظر خود را از روی عکس کراپ کنیم.

```
# Load the image
image = cv2.imread('coins.png', cv2.IMREAD_GRAYSCALE)

# Function to select the region of interest (ROI) for the template
def select_template(image):
    r = cv2.selectROI("Select Template", image)
    template = image[int(r[1]):int(r[1] + r[3]), int(r[0]):int(r[0] +
r[2])]
    cv2.destroyAllWindows()
    return template

# Select a coin as the template
template = select_template(image)
```

در این فانکشن ابتدا عکس نمایش داده می شود. سپس کاربر می تواند ناحیه مورد نظر خود را انتخاب کند و با زدن enter ، آن ناحیه به عنوان تمپلیت انتخاب می شود.

```
# Get the dimensions of the template
w, h = template.shape[::-1]
```

```
# Perform normalized cross-correlation
result = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
```

سپس ابعاد این تمپلیت را گرفته و با استفاده از cv2.matchTemplate ، تمپلیت را با تصویر تطبیق می دهیم. این تابع الگو را روی تصویر می کشد و شباهت را در هر موقعیت محاسبه می کند. همچنین از cv2.TM_CCOEFF_NORMED برای normalized cross-correlation استفاده شده است.
حال یک آستانه (threshold)تعریف می کنیم تا تصمیم بگیریم کدام مکان ها به عنوان منطبق در نظر گرفته شوند و مکان‌هایی که نمرات مشابهی بالاتر از این آستانه دارند، مطابقت در نظر گرفته می‌شوند.

```
# Define a threshold for detecting matches
threshold = 0.3
loc = np.where(result >= threshold)
```

برای جلوگیری از ناحیه هایی که با هم overlap دارند، از cv2.dnn.NMSBoxes استفاده کردم که Non-maximum supperssion را اعمال می کند. این تابع کادرهای مرزی، امتیازهای مربوط به آنها، آستانه امتیاز و آستانه همپوشانی را می گیرد. شاخص های کادر هایی را که پس از supperssion نگهداری می شوند را برمی گرداند و کادرهایی که بیش از حد با کادرهای با امتیاز بالاتر همپوشانی دارند را حذف می کند.

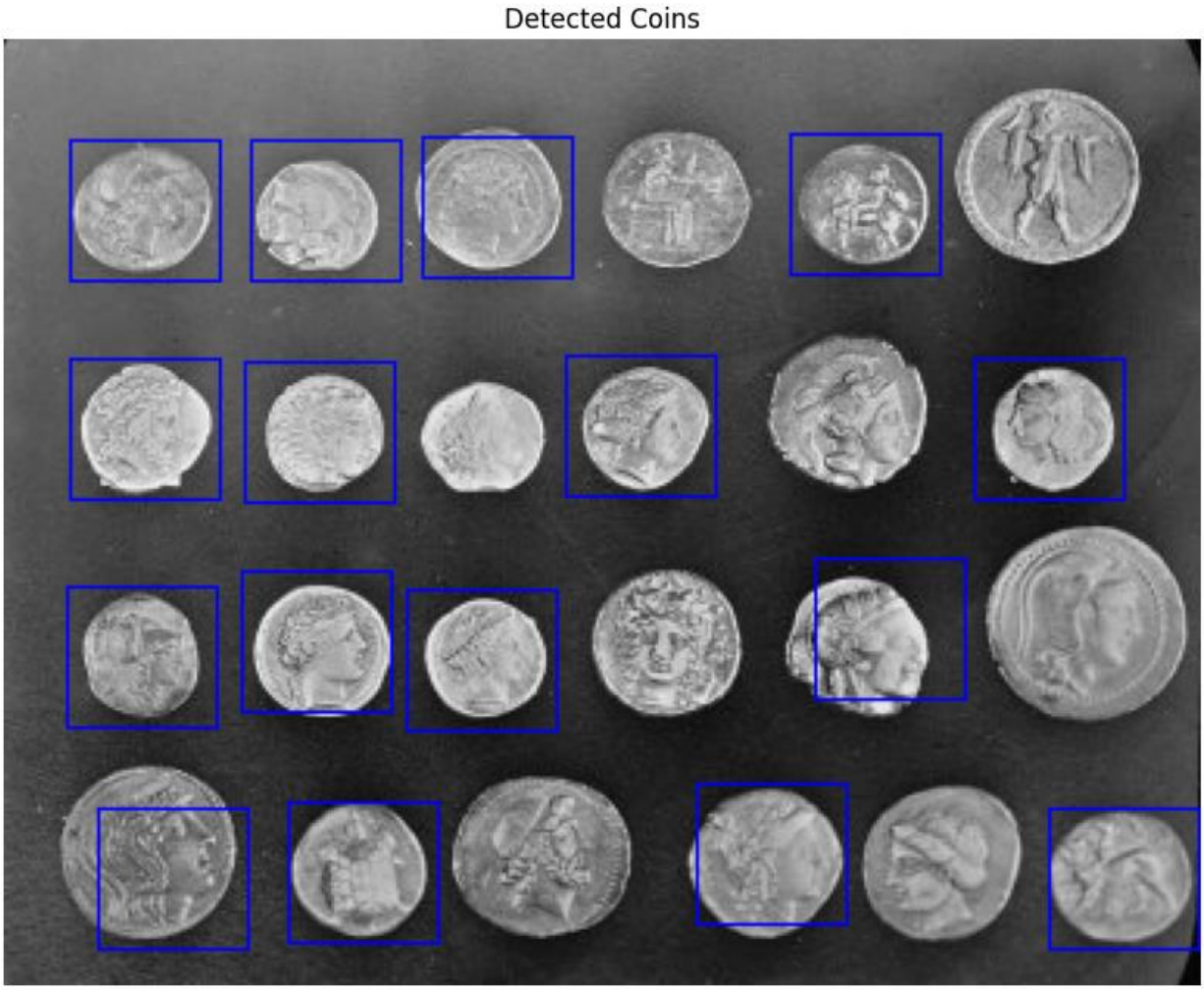
```
# Collect bounding boxes for all detected matches
boxes = []
scores = []
for pt in zip(*loc[::-1]):
    boxes.append([pt[0], pt[1], pt[0] + w, pt[1] + h])
    scores.append(result[pt[1], pt[0]])

# Convert boxes and scores to numpy arrays
boxes = np.array(boxes)
scores = np.array(scores)

# Apply non-maximum suppression using cv2.dnn.NMSBoxes
overlapThresh = 0.3
indices = cv2.dnn.NMSBoxes(boxes.tolist(), scores.tolist(), threshold,
overlapThresh)
```

ما کادرهای مرزی و امتیازاتشان را جمع آوری می کنیم که به صورت [x1، y1، x2، y2] ذخیره می‌شوند ((x1، y۱) گوشه بالا سمت چپ و (x2، y۲) گوشه پایین سمت راست). Scores هم امیتاز شباهت در این موقعیت ها هست.

سپس این فانکشن با استفاده از این ورودی ها به اضافه threshold همپوشانی، بهترین کادر را انتخاب می کند و به این وسیله از همپوشانی آنها جلوگیری می شود.
در ادامه نیز کادر های مستطیل شکل در اطراف نواحی که مطابقت داشتند، کشیده می شود و در نهایت خروجی نشان داده خواهد شد. تصویر خروجی زیر برای ناحیه انتخاب شده توسط من:

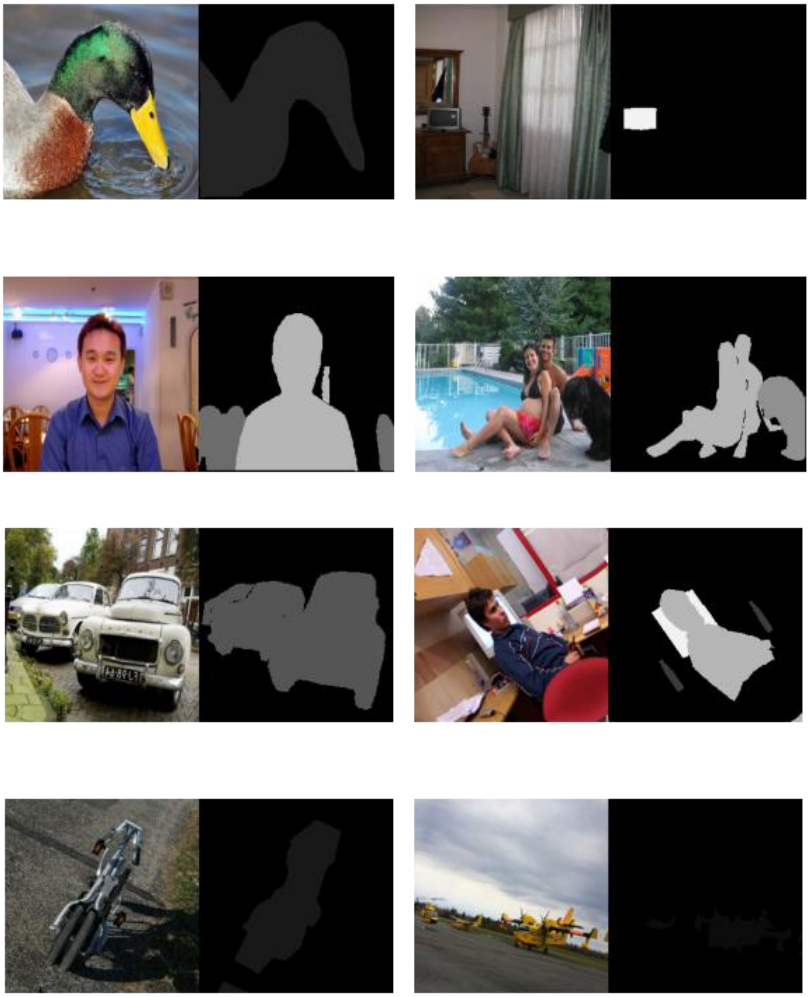
ناحیه انتخاب شده :

خروجی :



سوال ٤. طبق کد تکمیل شده در فایل SAM-completed ، می توان با استفاده از فانکشن BBoxWidget بخش مورد نظر خود را انتخاب کرد و شی مورد نظر را تشخیص داد.

سوال ٥. فانکشن preprocess_tfds_inputs مجموعه داده‌ها را پیش پردازش می‌کند، از جمله تغییر اندازه تصاویر و ماسک‌ها، دسته‌بندی داده‌ها، و استفاده از تکنیک‌های افزایش داده‌ها مانند random flipping و rotation. خروجی :



سپس در فانکشن unet_model ، مدل U-Net را برای وظایف تقسیم بندی تعریف می کنیم و در ادامه این مدل را با استفاده از توابع و معیارهای loss مشخص شده کامپایل می کنیم و سپس خلاصه مدل را چاپ می کنیم.

conv2d_12 (Conv2D)	(None, 56, 56, 256)	1,179,904	concatenate_1[0][0]
conv2d_13 (Conv2D)	(None, 56, 56, 256)	590,080	conv2d_12[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 112, 112, 128)	131,200	conv2d_13[0][0]
concatenate_2 (Concatenate)	(None, 112, 112, 256)	0	conv2d_transpose_2[0]… conv2d_3[0][0]
conv2d_14 (Conv2D)	(None, 112, 112, 128)	295,040	concatenate_2[0][0]
conv2d_15 (Conv2D)	(None, 112, 112, 128)	147,584	conv2d_14[0][0]
conv2d_transpose_3 (Conv2DTranspose)	(None, 224, 224, 64)	32,832	conv2d_15[0][0]
concatenate_3 (Concatenate)	(None, 224, 224, 128)	0	conv2d_transpose_3[0]… conv2d_1[0][0]
conv2d_16 (Conv2D)	(None, 224, 224, 64)	73,792	concatenate_3[0][0]
conv2d_17 (Conv2D)	(None, 224, 224, 64)	36,928	conv2d_16[0][0]
conv2d_18 (Conv2D)	(None, 224, 224, 21)	1,365	conv2d_17[0][0]

Total params: 31,033,045 (118.38 MB)
Trainable params: 31,033,045 (118.38 MB)
Non-trainable params: 0 (0.00 B)

در فانکشن dict_to_tuple مجموعه داده از فرمت دیکشنری به تاپل تصویر و one-hot encoded segmentation masks تبدیل می شود.

سپس مدل U-Net را با callbacks آموزش می دهیم تا بهترین مدل را ذخیره کرده و توقف زودهنگام را پیاده سازی کند.

```
Epoch 1/10
    265/Unknown 815s 2s/step - accuracy: 0.6159 - jaccard_coef: 0.3464 - loss: 0.9828/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data; interr
  self.gen.throw(typ, value, traceback)
265/265 ━━━━━━━━━━━━━━━━━━━━ 851s 2s/step - accuracy: 0.6160 - jaccard_coef: 0.3464 - loss: 0.9827 - val_accuracy: 0.6197 - val_jaccard_coef: 0.3575 - val_loss: 0.9473
Epoch 2/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 664s 2s/step - accuracy: 0.6208 - jaccard_coef: 0.3582 - loss: 0.9471 - val_accuracy: 0.6370 - val_jaccard_coef: 0.4176 - val_loss: 0.9414
Epoch 3/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 666s 3s/step - accuracy: 0.6013 - jaccard_coef: 0.3736 - loss: 0.9367 - val_accuracy: 0.5994 - val_jaccard_coef: 0.3915 - val_loss: 0.9371
Epoch 4/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 673s 3s/step - accuracy: 0.5914 - jaccard_coef: 0.3793 - loss: 0.9312 - val_accuracy: 0.6204 - val_jaccard_coef: 0.4211 - val_loss: 0.9275
Epoch 5/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 680s 3s/step - accuracy: 0.5978 - jaccard_coef: 0.3934 - loss: 0.9238 - val_accuracy: 0.6127 - val_jaccard_coef: 0.4166 - val_loss: 0.9303
Epoch 6/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 682s 3s/step - accuracy: 0.6000 - jaccard_coef: 0.3964 - loss: 0.9197 - val_accuracy: 0.6321 - val_jaccard_coef: 0.4374 - val_loss: 0.9241
Epoch 7/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 682s 3s/step - accuracy: 0.5989 - jaccard_coef: 0.3986 - loss: 0.9166 - val_accuracy: 0.6095 - val_jaccard_coef: 0.4129 - val_loss: 0.9083
Epoch 8/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 680s 3s/step - accuracy: 0.5952 - jaccard_coef: 0.3978 - loss: 0.9083 - val_accuracy: 0.5967 - val_jaccard_coef: 0.3987 - val_loss: 0.9116
Epoch 9/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 674s 3s/step - accuracy: 0.6037 - jaccard_coef: 0.4096 - loss: 0.9047 - val_accuracy: 0.6116 - val_jaccard_coef: 0.4231 - val_loss: 0.9075
Epoch 10/10
265/265 ━━━━━━━━━━━━━━━━━━━━ 679s 3s/step - accuracy: 0.6030 - jaccard_coef: 0.4102 - loss: 0.9022 - val_accuracy: 0.5868 - val_jaccard_coef: 0.3940 - val_loss: 0.9064
<keras.src.callbacks.history.History at 0x78fa33aace20>
```

در ادامه یک مدل U-Net را با یکpre-trained MobileNetV2 encoder تعریف می کند و encoder را فریز می کند و فقط decoder را train می کند.

در نهایت تمام لایه ها را از حالت انجماد خارج می کنیم و اجازه می دهیم تا تمام لایه ها train شوند.

در کل این کد نحوه ساخت و آموزش یک مدل U-Net را برای تقسیم‌بندی تصویر، هم از ابتدا و هم با استفاده از یک encoder از پیش آموزش دیده، نشان می‌دهد. این شامل مراحل پیش پردازش، تقویت داده ها، تعریف مدل و آموزش با کال بک های مناسب برای ذخیره بهترین مدل و اجرای توقف اولیه است.