Семинарска работа:

Алгоритми за сегментација на слики со длабоко учење

Наталија Насева

Содржина

Вовед

Компјутерската визија е поле на компјутерските науки кое се занимава со рекреирање на човековиот вид и овозможување компјутерите да идентификуваат и обработуваат предмети во слики и видеа на ист начин како што тоа го прават луѓето.

Сегментација на слики е една од поважните операции во дигиталната обработка на слики и машинската визија. Сегментација на слика подразбира делење на сликата на различни целини кои имаат некаква заедничка карактеристика, односно се дел од ист објект на сликата (човек, дрво, планина, облак, коловоз). Сегментацијата на слики се применува во различни области, како препознавање на објекти, следење и откривање на објекти во движење, медицински слики и роботика. Сегментацијата на слики овозможува подобра и попрецизна анализа на содржината на слики.

Во овој проект јас креирав U-net CNN за семантичка сегментација на слики, а искористив и веќе тренирана Mask R-CNN за сегментација на инстанци за споредба на прецизноста и ефикасноста на сегментација на двата типа на модели.

Во изработката на овој проект искористив Python, Tensorflow, Keras и OpenCV.

Сегментација и типови на сегментација на слики

Сегментацијата на сликата е процес на делење на сликата на повеќе значајни и хомогени региони или објекти врз основа на нивните вродени карактеристики, како што се бојата, текстурата, обликот или осветленоста. Сегментацијата на сликата има за цел да го поедностави и/или да го промени претставувањето на сликата во нешто позначајно и полесно за анализа. Овде, секој пиксел е означен. На сите пиксели кои припаѓаат на иста категорија им е доделена заедничка ознака.

Постојат повеќе типови на сегментација, кои главно се делат во три категории, според тоа какви и колку информации. Тие се:

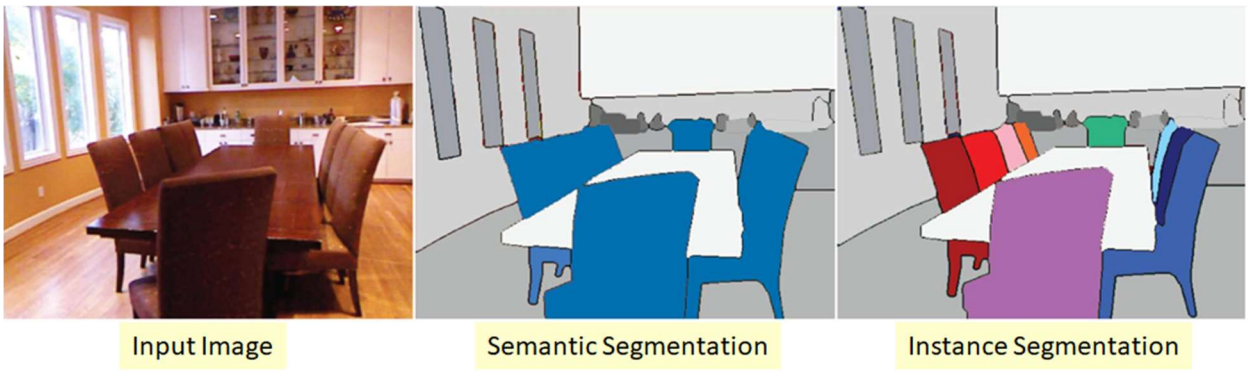
- Семантичка сегментација;
- Сегментација на инстанци на објекти;
- Паноптичка сегментација.

Јас ги обработував првите два типови на сегментација, па да ги дефинираме тие типови.

Семантичка сегментација

Семантичката сегментација е тип на сегментација на сликата што вклучува означување на секој пиксел во слика со соодветна класа без да се земат во предвид други информации или контекст, односно целосно се сегментира сликата, вклучувајќи ја и позадината, без поделба на објектите на инстанци.

Целта е да се додели ознака на секој пиксел на сликата, што обезбедува густо означување на сликата. Алгоритмот зема слика како влез и генерира маска на сегментација каде што вредноста на пикселот (0,1,...255) на сликата се трансформира во ознаки за класи (0,1,...n). Корисно е во апликации каде што е важно да се идентификуваат различните класи на објекти на патот.



сл. 1 Семантичка сегментација и сегментација на инстанци. Адаптирано од [8]

Сегментација на инстанци

Сегментацијата на инстанци е тип на сегментација на сликата која вклучува откривање и сегментирање на секој објект во сликата. Слично е со детекција на објекти на слика, но со дополнителна задача за сегментирање на границите на објектот, односно генерирање маска.

Алгоритмот не ја знае класата на секој регион, но ги одделува сите индивидуални објекти коишто се преклопуваат. Сегментацијата на инстанци е корисна во апликации каде што треба да се идентификуваат и следат поединечни објекти, на пример во self-driving автомобили.

Во примерот погоре (сл.1), со примена на семантичка сегментација, сите столчиња се сегментирани како една целина, тие се покриени со една маска и се третираат како единствена инстанца од класата „стол“. Додека пак, со примена на сегментација на инстанци, секој стол е поединечна инстанца на класата „стол“ и има своја маска, иако припаѓаат на истата класа.

U-net за семантичка сегментација

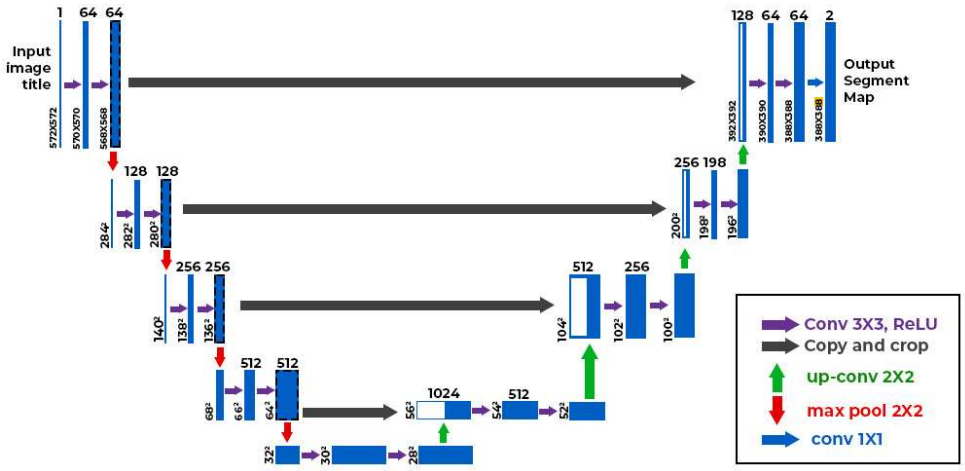
U-net архитектурата е архитектура е модел во формата на буквата U кој се состои од конволутивни слоеви и две, меѓусебно поврзани мрежи, енкодер и декодер.

Енкодерот е и „contracting“ дел од мрежата и неговата улога е да одреди што е на сликата. Енкодерот, во случајот на сегментација на слики, е обична конволутивна невронска мрежа. Единствената разлика помеѓу овој CNN и CNN во било која задача за класификација на слики е тоа што на крајот, во последниот слоеви, нема целосно поврзани слоеви. Ова е така, затоа што на излез од овој модел сакаме да добиеме маска за сегментација на сликата, а не класа на која таа слика припаѓа. Подоцна, ќе ги објаснам сите слоеви, односно блокови, од коишто се состои мрежата.

Декодерот е и „expansive“ дел од мрежата и неговата улога е да одреди каде се наоѓа објектот на сликата. Овој дел од мрежата ја зема мапата на карактеристики која ја создава енкодерот и генерира маска за сегментација на сликата. И декодерот има слична архитектура со енкодерот, но наместо да извршува конволуција на сликата, тој извршува транспонирачка конволуција на мапата генерирана од енкодерот.

Помеѓу енкодерот и декодерот има и „skip“ врски кои директно поврзуваат одредени слоеви на енкодерот со декодерот, без тие информации да мора да поминуваат низ целата мрежа. Овие врски овозможуваат да ги користиме карактеристиките научени од сликата додека таа се уште има релативно висока резолуција, од страна на блоковите на енкодерот. Овие карактеристики му помагаат на декодерот подобро да го пронајде објектот на сликата, односно подобро да биде позиционирана маската за сегментација.

U-net е пример за FCN, односно fully convolutional network. Тоа значи дека мрежата нема целосно поврзани слоеви, т.е. линеарни слоеви. Причината за тоа е што на излез на мрежата ние сакаме да добиеме маска, односно друга слика, а не само единечна вредност за класа, односно веројатност за припадност на некоја класа.



сл. 2 U-net за сегментација на слики. Адаптирано од [12]

Изработка на U-net

Податочно множество и претпроцесирање

Со цел тренирање на моделот за сегментација, јас го искористив COCO-Stuff 10k податочното множество. Ова податочно множество се состои од 10.000 слики на кои се претставени објекти од 183 класи, секоја со различна застапеност. Според авторите на податочното множество, тоа може да се подели на две категории, thing (објект со добро дефинирана форма, како автомобил или животно) и stuff (региони во позадината на сликата, како небо или водена површина). Ова податочно множество има 91 thing класа, 91 stuff класа и една класа која претставува неопределени објекти. Со ваквата структура, ова податочно множество е многу корисно за сегментирање на слики од секојдневниот живот, како слики од урбани средини.

Податочното множество се состои од слики и соодветните анотации на сликите. Анотациите се состојат од единствена класа на сликата, доколку податоците се користат за класификација, и класи за секој од објектите кои се наоѓаат на сликата, како и нивните релативни координати.

Со цел претпроцесирање на податоците, пред тие да бидат влез во моделот, создадов неколку функции.

```
def load_image(img_path, size=(224, 224)):
  img = tf.io.read_file(img_path)
  img = tf.image.decode_jpeg(img, channels=3)
  img = tf.image.resize(img, size)
  img = tf.cast(img, tf.float32)/ 255.0
  return img
```

Улогата на оваа функција е да ја прочита сликата од датотеката во која е сместена, да ја скалира во соодветната големина, во случајот 224 x 224, и конечно ја дели вредноста на секој пискел со 255. Големината на сликите определив да е 224 x 224 бидејќи основниот модел за тренирање кој го користам е трениран на оваа големина на слики, а исто така оваа големина на слики е доволно мала за процесот на тренирање да е релативно брз, но сепак може да се извлечат најважните карактеристики на секоја слика. Последниот чекор, односно делењето со 255 го извршувам бидејќи користам RGB слики, па резултатот на оваа операција ќе биде реален број помеѓу 0 и 1, што го прави процесот на тренирање на моделот многу побрз и поедноставен.

Потоа, извршувам вчитување на маските за секоја слика, односно секој објект, од податочното множество. Со цел заштедување на простор, маските се енкодирани во Base64 форматот, па тие прво мора да се декодираат и декомпресираат. Потоа, од бинарната форма, ги трансформирам во вистински слики и со помош на алфа каналот на сликата ја креирам маската, каде True претставува делови од сликата каде што има маска. Конечно, ја позиционирам маската на објектот на нејзиното вистинско место во маската на објектот.

```
    compressed_data = base64.b64decode(bitmap_data)
    decompressed_data = zlib.decompress(compressed_data)

    imdecoded = cv2.imdecode(n, cv2.IMREAD_UNCHANGED)
    mask = imdecoded[:, :, 3].astype(bool)

    x, y = origin
    full_mask[y:y + mask.shape[0], x:x + mask.shape[1]] = mask
```

За маските користам one-hot енкодирање, односно за секоја класа има соодветен канал, па димензиите на конечната маска се 224 x 224 x 183.

Со цел проширување на податочното множество, извршив аугментација на сликите кои веќе ги имав. Тоа го извршив со ротација, зголемување, односно намалување на контраст, сатурација, осветленост и нијанса. Клучно е ротациите да се извршат и на сликите и на соодветните маски, со цел правилно претставување на објектите во нив. Аугментацијата на податоците помага при спречување на overfitting, бидејќи моделот учи од разновидни слики, со различни карактеристики.

```
flip_left_right = tf.random.uniform([], 0, 1) > 0.5
image = tf.image.flip_left_right(image)
mask = tf.image.flip_left_right(mask)

image = tf.image.random_brightness(image, max_delta=0.1)
image = tf.image.random_saturation(image, lower=0.9, upper=1.1)
```

При користење на некакво податочно множество за тренирање на модел, клучно е припремањето на податоците да е особено брзо, па затоа користев TensorFlow за генерирање на влезовите на моделот.

```
dataset = tf.data.Dataset.from_tensor_slices((img_paths, ann_paths))
dataset = dataset.map(load_data, num_parallel_calls=tf.data.AUTOTUNE)
dataset = dataset.shuffle(6000)
dataset = dataset.batch(batch_size)
dataset = dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
```

Овој генератор на податоци го користам за генерирање на тренирачкото множество и множеството за валидација, каде што големината на една серија е 32.

Енкодер

Енкодерот во U-net архитектурата има улога да одреди што е на сликата, па тој е сличен со многу CNN кои се користат во задачи од компјутерската визија.

Конволутивна невронска мрежа е Deep Learning алгоритам кој може, за определен влезен податок (слика), да додели значајност на одредени аспекти, односно објекти на сликата и да ги разликува еден од друг. Архитектурата на ConvNet е аналогна на начинот на поврзување на невроните во човечкиот мозок и е инспирирана од организацијата на визуелниот кортекс. Улогата на ConvNet е да ги претвори сликите во форма која е полесна за обработка, без губење на карактеристики кои се клучни за добро предвидување.

Конволутивната невронска мрежа се состои од влезен слој, скриени слоеви и излезен слој. Во која било невронска мрежа, сите средни слоеви се нарекуваат скриени бидејќи нивните влезови и излези се маскирани со функцијата за активирање и конечната конволуција

Како дел од мојот проект јас користев веќе истриниран MobileNetV2, затоа што овој модел, од страна на Keras е трениран на огромно податочно множество, па е многу добар во детекција на објекти, но сепак е брз. Исто така, користењето на веќе трениран модел го намалува времето потребно за тренирање на конечниот модел, а и го намалува трошокот на ресурси при тренирањето.

Енкодерот е составен од неколку блокови од исти слоеви, кои се клучни во овозможување на неговата работа. Тоа се конволутивниот слој, max pooling слојот и слојот за активација. Понатаму ќе го објаснам секој слој поединечно.

Конволутивен слој

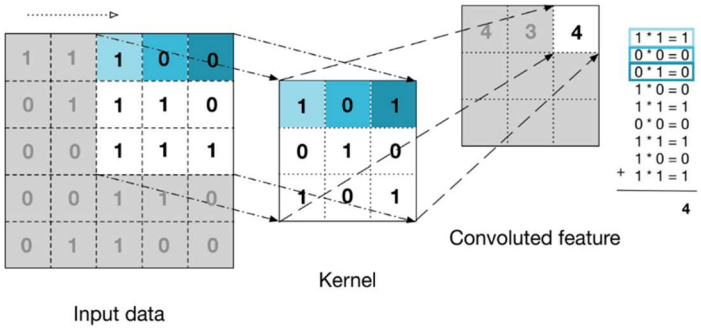
Првиот слој од секоја конволутивна невронска мрежа е конволутивниот слој. Конволутивните слоеви применуваат операција на конволуција на влезот, пренесувајќи го резултатот на следниот слој. Конволуцијата ги конвертира сите пиксели во своето приемно поле во една вредност. На пример, при примена на конволуција на слика, се намалува големината на сликата, а и се собираат сите информации во полето заедно во еден пиксел. По извршување на неколку конволуции, од сликата се собираат сите значајни информации

Раните конволутивни слоеви, односно тие што се наоѓаат на почетокот на мрежата откриваат едноставни карактеристики како рабови, агли или текстури. Како што се движиме подлабоко во мрежата, слоевите можат да детектираат посложени објекти, како што се форми или дури и предмети, со комбинирање на претходните едноставни карактеристики, односно мапи.

Висината и ширината на kernel која ја користам во мојот модел е 3 x 3, додека бројот на филтри кои ги нанесувам на мапата во секој блок се зголемува (32, 64, 128, 256), со цел зголемување на прецизноста на моделот.

Често, при изработка на CNN, во секој блок од мрежата се користат да конволутивни слоеви. Овој тип на повторување на слоевите е клучен за моделот. Повторувањето на слоеви овозможува мрежата да има богата, слоевита претстава на податоците. Секој

последователен слој додава повеќе сложеност и апстракција, овозможувајќи ѝ на мрежата да ги разбере концептите на високо ниво од податоците за пиксели на ниско ниво.

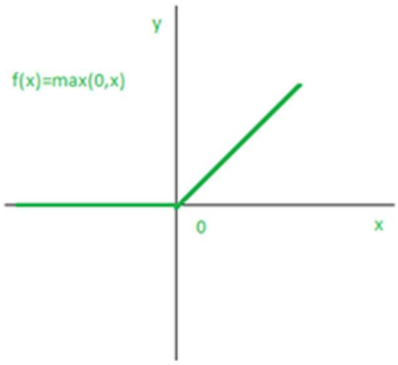



```
Conv2D(filter,
(kernel width, kernel height),
padding='same',
input_shape=(width, height, dimensions)))
```

сл. 3 Пример за конволуција. Адаптирано од [9]

Слој за активација

Во CNN, со слојот за активирање на неврон се дефинира излезот на еден неврон при даден влез или множество влезови. Со ReLU активација на следниот влез се пренесува излезот ако е позитивен број, инаку излезот ќе биде еднаков на нула. Главната предност на користењето на функцијата ReLU во однос на другите функции за активирање е тоа што не ги активира сите неврони во исто време. Ако излезот е негативен број, невронот воопшто нема да се активира. ReLU активацијата може да се изрази со функцијата (1)



$$f(x) = \max(0, x) \qquad (1)$$

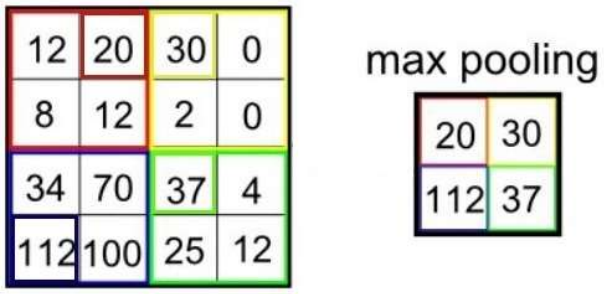
сл. 4 График на ReLU активација

Pooling слој

Слично на конволутивниот слој, pooling слојот е одговорен за намалување на просторната големина на конволутивните карактеристики. Ова е за да се намали пресметковната моќ потребна за обработка на податоците преку намалување на димензионалноста. Понатаму, корисен е за извлекување на клучни, односно доминантни карактеристики, со што се подобрува ефикасноста на моделот.

Во мојот модел јас применувам Max Pooling. Max Pooling ја враќа максималната вредност од делот од сликата покриен со матрицата, односно kernel. Исто така, со Max Pooling се минимизира шумот.

MaxPooling2D означува дека pooling слојот е наменет за дводимензионални слики, додека pool_size ја дефинира висината и ширината на матрицата.




```
MaxPooling2D(pool_size=(kernel height,
kernel width))
```

сл. 5 Начин на работа на MaxPooling слојот. Адаптирано од [13]

Skip врски

Skip врските се критична компонента во U-Net. Тие се користат за пренос на информации директно од слоевите на енкодерот до слоевите на декодерот, заобиколувајќи некои од средните слоеви.

Главната улога на skip врските е зачувување на просторните информации добиени од сликите. Како што влезот минува низ слоевите на мрежата, ширината и висината на мапите на карактеристики се намалуваат. Ова може да доведе до губење на фини детали. Прескокните врски помагаат да се зачуваат овие просторни информации со пренесување на карактеристиките со висока резолуција од претходните слоеви на енкодерот директно до декодерот.

Skip врските работат на тој начин што секој слој во енкодерот е поврзан со неговиот соодветен слој во декодерот. Но, наместо да заменување на картите со карактеристики на декодерот со тие од енкодерот, skip врските ги спојуваат и двата типови на маски. Ова ѝ овозможува на мрежата да ги искористи и карактеристиките на високо ниво од декодерот и ситно-гранулираните детали од енкодерот.

Во проблемите со сегментација на слика, каде што и просторната точност и контекстот се клучни за добри резултати, skip врските му помагаат на моделот да ги балансира овие два аспекта, што доведува до попрецизни резултати за сегментација.

```
skip_1 = encoder.get_layer(skip_connection_names[-1]).output

concatenate([u6, skip_1])
```

Декодер

Главната улога на декодерот во U-net е да ја реконструира маската на влезната слика, според карактеристиките и мапите генерирани од конволутивната невронска мрежа, односно енкодерот. Тој работи обратно од енкодерот, почнува да ја обработува мапата на карактеристики со најмала ширина и висина, но со најголема длабочина, а како резултат враќа маска со висина и ширина еднакви на димензиите на влезната слика, но со длабочина соодветна на бројот на класи за сегментација, во овој случај 183. Декодерот го прави ова во три чекори, транспонирање, спојување на карактеристики со енкодерот и коволуција.

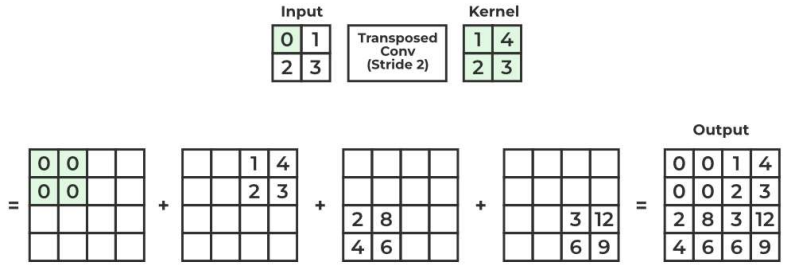
Овој дел од мојот модел го креирав без некаков основен модел, односно веќе истренирани тежини. Тоа го направив со цел моделот да биде подобро приспособен за сегментирање на слики од секојдневниот живот, односно да биде попрецизен. Слоевите кои ги користев во овој дел од моделот се: слој за транспонирана конволуција, слој за конволуција, слој за нормализација, слој за активација и dropout слој.

Слој за транспонирана конволуција

Првиот слој во секој блок од декодерот е слојот за транспонирана конволуција. Слојот за транспонирана конволуција е слој за зголемување на примерокот што ја зголемува влезната мапа на карактеристики според некое правило.

Операцијата на слојот за транспонирана конволуција е слична на онаа на нормалниот конволуциски слој, освен што тој ја извршува операцијата на конволуција во спротивна насока. Наместо да го лизга кернелот над влезот и да врши множење и сумирање според елементот, транспониран конволуционерен слој ја лизга влезната мапа преку кернелот и врши множење и сумирање според елементот во мапата. Ова резултира со излез кој е поголем од влезот, а големината на излезот може да се контролира со параметрите за големина на чекор и padding.

Висината и ширината на kernel која ја користам во мојот модел е 3 x 3, додека бројот на филтри кои ги нанесувам на мапата во секој блок се намалува (32, 64, 128, 256), со цел зголемување на прецизноста на моделот. Бројот на филтри кои ги применувам кај секој слој од декодерот е соодветна на бројот на филтри кои ги нанесува енкодерот.




```
Conv2DTranspose(filter,
(kernel height, kernel width),
strides=(2, 2),
padding='same')
```

сл. 6 Начин на работа на слој за транспонирана конволуцаија. Адаптирано од [17]

Конволутивен слој

Слојот за конволуција во енкодерот и декодерот имаат иста функционалност и иста улога. Клучно за слојот за конволуција кај декодерот е дека тој го користам по конкатенирањето на мапите на карактеристики од енкодерот, добиени преку skip врските, и мапите од декодерот, добиени после транспонираната конволуција.

```
Conv2D(filter,(kernel width, kernel height),padding='same',
input_shape=(width, height, dimensions)))
```

Карактеристичен конволутивен слој е последниот слој во целиот модел. Кај овој слој, бројот на филтри е еднаков на бројот на класи на кои сегментира моделот, со тоа, излезната маска ќе има толку канали колку што има класи, каде што секој канал ја претставува предвидената карта на веројатност за одредена класа. Големината на кернелот е 1 x 1. Се користи 1x1 конволуција за мапирање на векторот на карактеристики на секој пиксел (од претходниот слој) директно до излезните класи. Додека пак, softmax активацијата ги конвертира резултатите од необработените класи за секој пиксел во веројатности.

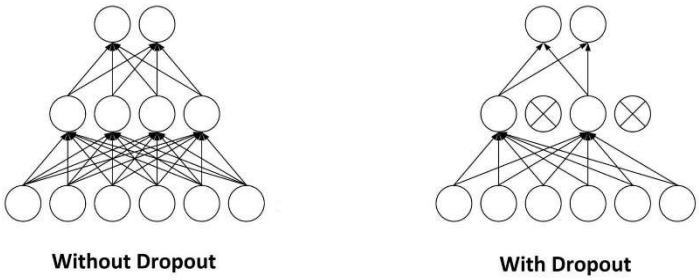
```
outputs = Conv2D(n_classes, (1, 1), activation='softmax')
```

Слој за нормализација

Слојот за нормализација ги трансформира влезовите на следниот слој, односно излезите од претходниот, така што тие се стандардизирани, што значи дека ќе имаат аритметичка средина еднаква на нула и стандардна девијација еднаква на еден. Овој слој го применувам во секој блок од моделот.

Dropout слој

Dropout слоевите се важни за да се спречи overfitting на моделот. Dropout работи со случајно поставување на неколку од излезните неврони на нула. Ако тие слоеви не се присутни, првата серија примероци за тренирање на моделот влијае на учењето на несразмерно високо ниво. Ова, пак, би го спречило учењето на карактеристиките што се појавуваат само во подоцнежните примероци или серии. Во мојот модел користам dropout од 0.2 и 0.3.



сл. 7 Начин на работа на dropout слој. Адаптирано од [3]

Тренирање на моделот

Во процесот на тренирање на моделот, со цел подобрување на брзината на тренирање, прецизноста и ефикасноста на моделот, искористив неколку callback и оптимизатори.

Callbacks

- *Early stopping*
  Мора да искористиме рано прекинување, со цел моделот да не се тренира бесцелно, односно да се прекине тренирањето откако некој параметар ќе престане со подобрување. Во мојот модел, доколку по две епохи (patience = 2), нема никакво подобрување (min_delta = 0), тренирањето престанува и се зачувуваат најдобрите резултати (restore_best_weights = True).

- *Reduce Learning Rate on Plateau*
  За слична цел се користи и намалување на стапката на учење. Доколку подобрувањето на еден параметар стагнира, се намалува стапката на учење, со цел да се подобри работата на моделот.

Оптиизатор

- Adam
  Адам оптимизацијата е метод на stochastic gradient descent кој се заснова на адаптивна проценка на моменти од прв и втор ред.

Метрики

Dice коефициент

Dice коефициентот е метрика на сличност што вообичаено се користи во сегментација на слики и други полиња каде што има потреба да се измери сличноста помеѓу две групи. Dice коефициентот е мерка за сличноста помеѓу две множества, A и B. Коефициентот се движи од 0 до 1, каде што 1 означува дека двете множества се идентични, а 0 покажува дека двете множества немаат преклопување. Тој е дефиниран со формулата (2), каде |A| го претставува бројот на елементи во множеството A, и |B| го претставува бројот на елементи во множеството B. |A ∩ B| го претставува бројот на елементи кои се присутни во двете множества.

$$\frac{2 * |A \cap B|}{|A| + |B|} \qquad (2)$$

Dice коефициентот има неколку предности во однос на другите метрики на сличност. Тој е особено корисен за неурамнотежени податочни множества, каде што едното множество е многу поголемо од другото. Тој е подобар избор за задачи за сегментација на слики, бидејќи е почувствителен на преклопување помеѓу предвидените и вистинските маски. Ова се

постигнува со третирање на маските за сегментација како множества од пиксели. Предвидената и вистинската маска се претставени како бинарни маски, каде што пикселот е или дел од сегментираниот објект или не.

IOU (Intersection over Union)

IOU (Пресек врз унија) е термин кој се користи за да се опише степенот на преклопување на две маски. Колку е поголем регионот на преклопување, толку е поголем IOU. IOU главно се користи во апликации поврзани со откривање на објекти, каде што трениране модел да генерира кутија што е совршена маска за детектираниот објектот. Вредноста на IOU, слично како и Dice коефициентот, може да се движи помеѓу 0 и 1. Формулата за IOU е претставена со формула (3).

$$\frac{\text{Плоштина на пресек на маски}}{\text{Плоштина на унија на маски}} \tag{3}$$

IOU ни дава добра претстава за прецизноста на моделот, односно на поклопувањето на маската генерирана од моделот и вистинската.

Categorical Cross-Entropy

Бидејќи секој примерок, т.е. објект во задачи за сегментација може да припаѓа само на одредена класа, вистинската вредност на веројатноста би била 1 за таа одредена класа и 0 за другите класи. Cross-entropy ја мери разликата помеѓу предвидената веројатност и вистинската веројатност.

Загубата на cross-entropy е изведена од принципите на проценка на максимална подобност кога се применува на задачата за класификација. Максимирањето на веројатноста е еквивалентно на минимизирање на негативната лог-веројатност.

Categorical Cross-Entropy Loss, исто така познат како или softmax загуба, е широко користена функција на загуба за модели за проблеми со класификација каде имаме повеќе со повеќе класи. За податочно множество со N примероци, categorical cross-entropy loss се пресметува со формулата (4), каде C е бројот на класи, $y_{i,j}$ е вистинската класа, а $p_{i,j}$ е предвидената класа.

$$\frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{i,j}\log p_{i,j}) \tag{4}$$

Mask R-CNN за сегментација на инстанци

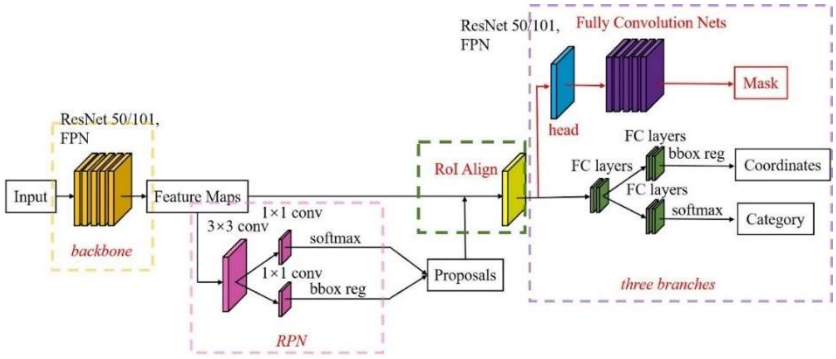
Mask R-CNN е тип на модел за длабоко учење кој често се користи за сегментација на инстанци, а има примена и во други проблеми од компјутерската визија. Овој модел комбинира детекција на објекти и сегментација на инстанци, т.е. генерира маски на ниво на пиксел за секој објект кој може да го детектира на сликата, па затоа е многу успешен во сегментација на инстанци.

Mask R-CNN има две улоги, детекција на објекти, и сегментација на објекти на слики.

- **Детекција на објекти:** Mask R-CNN ги идентификува објектите во сликата со предвидување на гранични кутии околу нив и класифицирање што се тие предмети (на пр., мачка, автомобил, човек).
- **Сегментација на инстанци:** Освен самото откривање и класификација на објекти, Mask R-CNN, исто така, го сегментира секој објект на ниво на пиксели. Ова значи дека создава бинарна маска која ги истакнува точните пиксели кои припаѓаат на секој објект, дозволувајќи му на моделот да прави разлика помеѓу различни примероци од иста класа на објекти.

За овој проект, јас ја користев веќе тренирана Mask R-CNN, на податочното множество MS COCO, со цел заштедување на време и ресурси за тренирање.

Архитектура на Mask R-CNN



сл. 8 Архитектура на Mask R-CNN. Адаптирано од [18]

Основна мрежа

Првиот чекор на Mask R-CNN е основната мрежа, што најчесто е конволутивна невронска мрежа, како и кај U-net. Нејзината улога е да ја трансформира необработената слика во мапа на нејзините визуелни карактеристики со извлекување на релевантни карактеристики од влезната слика. Мрежата обично се состои од повеќе конволутивни слоеви, pooling слоеви и други нелинеарни операции кои овозможуваат извлекување на информации од влезната слика. Како основнва мрежа најчесто се користат ResNet и ResNeXt.

Region Proposal Network (RPN)

Благодарение на извлечените карактеристиките од основната мрежа, RPN ја скенира сликата и предлага региони на потенцијални објекти користејќи предефинирани рамки околу објектите. Овие рамки се со различни соодноси и размери и сите се потенцијални рамки околу објектите. RPN доделува оценка за секој потенцијален регион што ја претставува неговата сличност со вистински објект. Повисока оценка подразбира веројатно присуство на вистинскиот објект во предложениот регион.

RPN го прави ова оценување со лизгање на мала мрежа преку мапата на карактеристики произведена од основната мрежа и предвидување на веројатноста некој објект да биде присутен во множеството од предефинираните рамки.

RPN е составена од конволутивни, pooling и активациски слоеви.

Region of Interest порамнување

Откако ќе бидат генерирани предлозите за региони од страна на RPN, се извлекуваат мапи на карактеристики само од тие региони. За оваа операција се користи RoI Align.

RoI Align операцијата применува билинеарна интерполација за прецизно определување на рамките на објектите на сликата. Прво ROI align извршува делење на сите координати на рамката со некоја вредност k. Понатаму, предложениот регионот се дели на мрежа, каде за секој дел од мрежата го претставуваат 4 точки. Конечно од тие 4 точки се избира максималната вредност, односно се генерира нова мапа на карактеристики. Оваа операција се извршува за секој предложен регион.

Излез

Mask R-CNN враќа три типа на излез кои се клучни за нејзиното правилно функционирање:

- *Класификација на објект*
  За секој RoI кој беше пронајден од моделот, моделот го класифицира објектот се наоѓа во тој регион.
- *RoI на објект*
  За секој RoI моделот ја враќа и рамката на тој специфичен регион.
- *Маска на објект*
  Оваа маска ја претставува маската за сегментацијата на објектот, каде што секој пиксел е класифициран според тоа дали припаѓа на објектот (1) или не (0). За разлика од традиционалните модели на семантичка сегментација, Mask R-CNN произведува посебна маска за секој откриен пример на објект, што го прави модел на сегментација на инстанци.

Сегментација на слики

Сегментација на слики со U-net

По успешно тренирање на моделот, преостанува само да се изврши предвидувањето на маските и прикажувањето на конечните резултати.

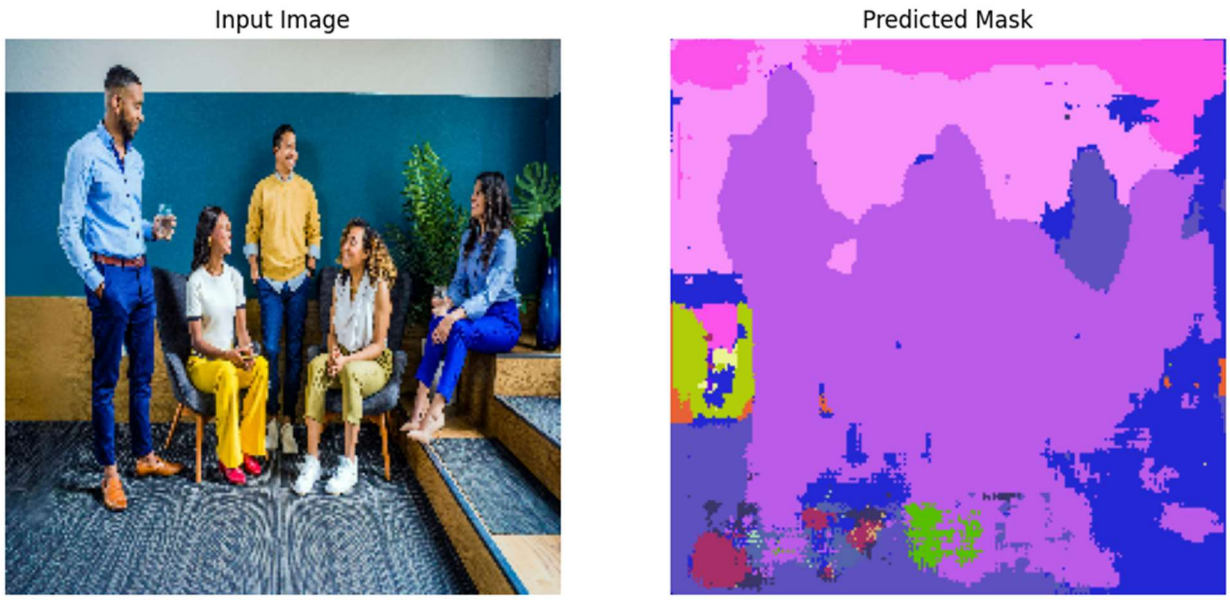
Прво, потребно е да се вчита, скалира и нормализира сликата на која сакаме да извршиме семантичка сегментација, со цел таа да биде соодветен влез за моделот.

```
img = cv2.imread(img_path) img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (224, 224))
img = img / 255.0
```

Понатаму, на влезната слика мораме да ѝ ги зголемиме димензиите (batch size, height, width, channels), затоа што моделот се тренира во batches, па првата вредност во низата би била големината на еден batch.

Излезот од моделот, односно маските се во форма (batch size, height, width, classes), па ја користиме само последната вредност за прикажување на маската.

```
pred_mask = model.predict(tf.expand_dims(img, axis=0))
pred_mask = np.argmax(pred_mask, axis=-1)[0]
pred_mask = cv2.applyColorMap(pred_mask.astype(np.uint8), colormap)
```

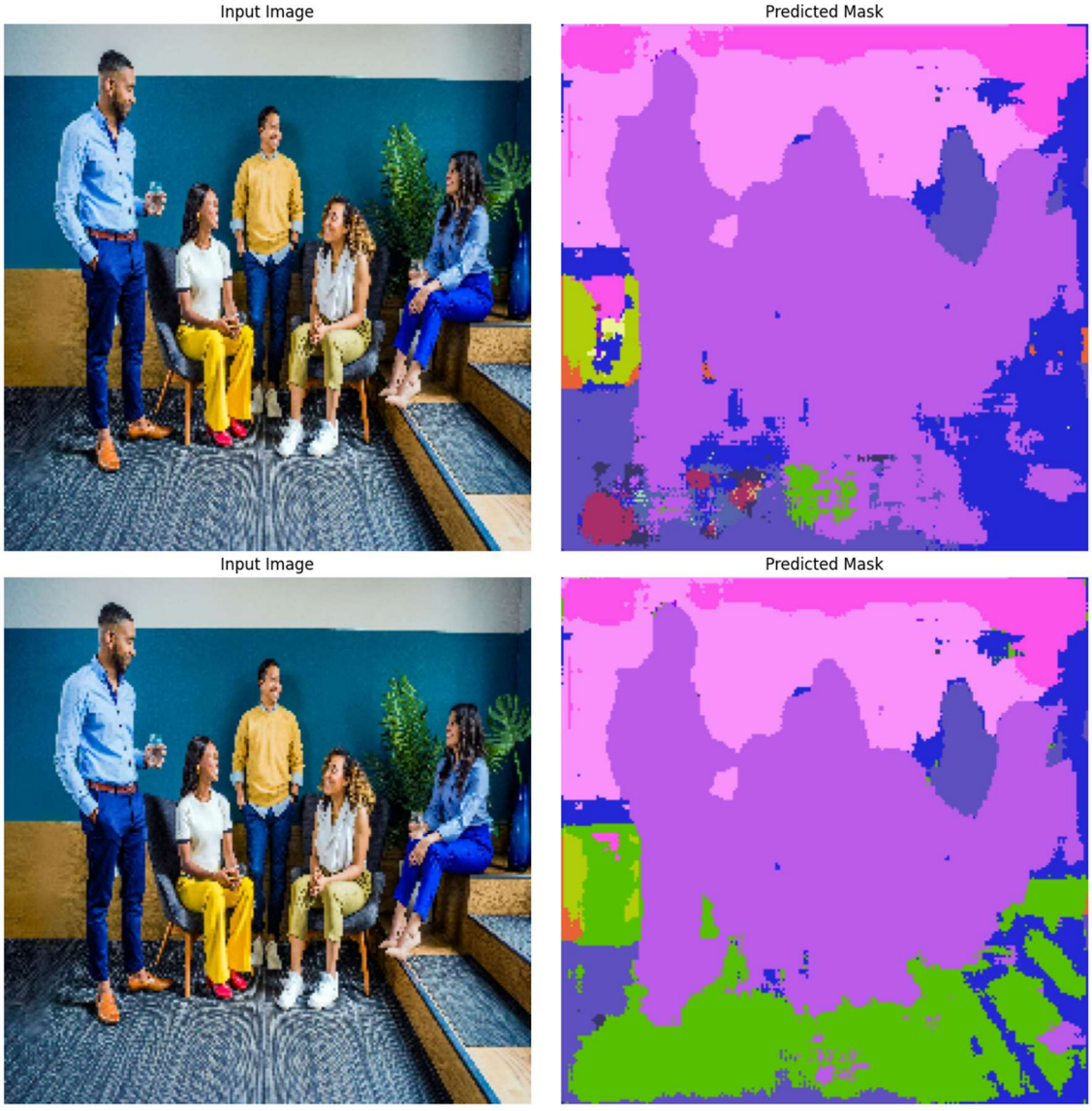


сл. 9 Семантичка сегментација со U-net

Thresholding врз основа на веројатност на маска

Бидејќи податочното множество кое го користам не е урамнотежено, односно не сите класи се подеднакво застапени, модел предвидува некои класи кои воопшто не се присутни на сликата, а се многу застапени во податочното множество, а не предвидува други класи иако тие се присутни на сликата.

За да го решам овој проблем, или барем да ја подобрам прецизноста на конечната маска, воведов thresholding на предвидените маски, односно пиксели. Цел процес на предвидување останува ист, но додадов еден филтер каде секој пиксел чија веројатност на предвидена класа е помалку од одреден threshold, (го поставив на 0.2), ќе биде прикажан како пиксел од класата „непознато“.



сл. 9 Сегментирање без threshold (горе) и со threshold [threshold=0.2] (долу)

Примери за сегментирање на објекти и урбани средини

сл. 10 Сегментација на објекти и урбани средини. [threshold=0.1] (горе). [threshold=0] (долу)

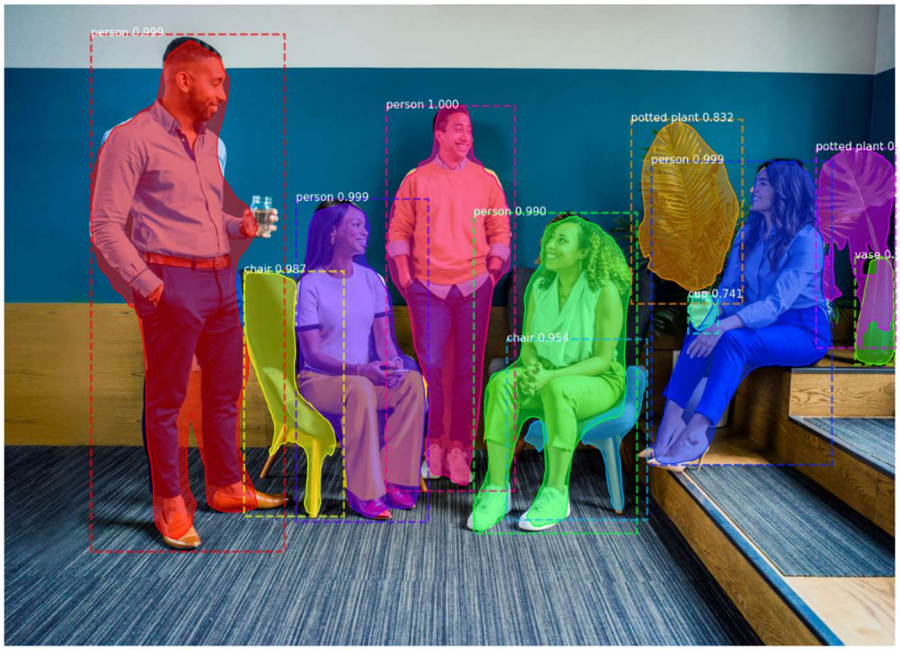
Сегментација на слики со Mask R-CNN

Пред да се изврши со предвидувањето, прво мора да се вчитаат класите кои можат да бидат предвидени, моделот и неговите соодветни тежини.

```
CLASS_NAMES = open("samples/coco_labels.txt").read().strip().split("\n")

model = mrcnn.model.MaskRCNN(mode="inference", config=SimpleConfig(),
model_dir=os.getcwd())

model.load_weights(filepath="mask_rcnn_coco.h5", by_name=True)
```

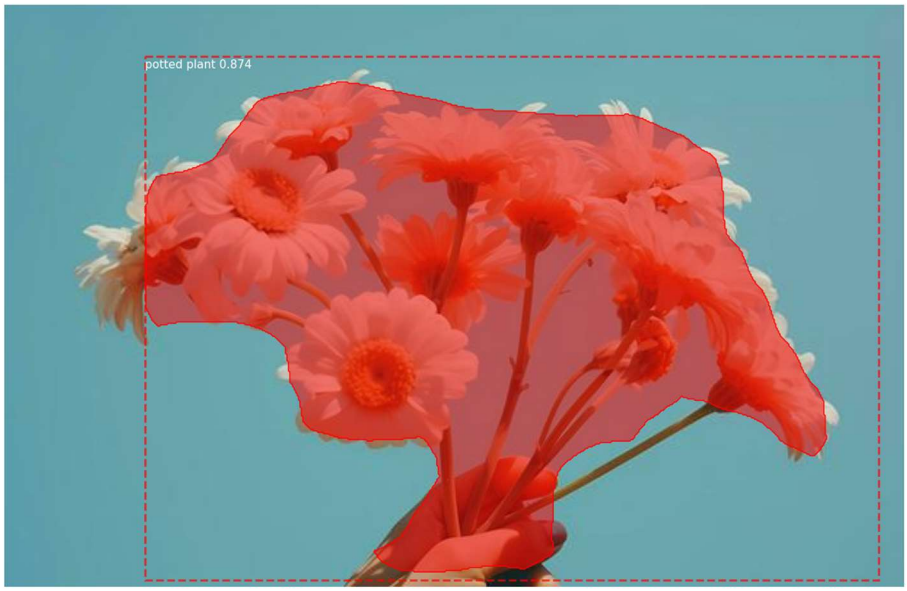
За извршување на предвидувањето и прикажување на резултатот, потребно е само да се вчита сликата, да се изврши предвидувањето и да се повика соодветната функција за визуелизација на максата за сегментација на инстанци.

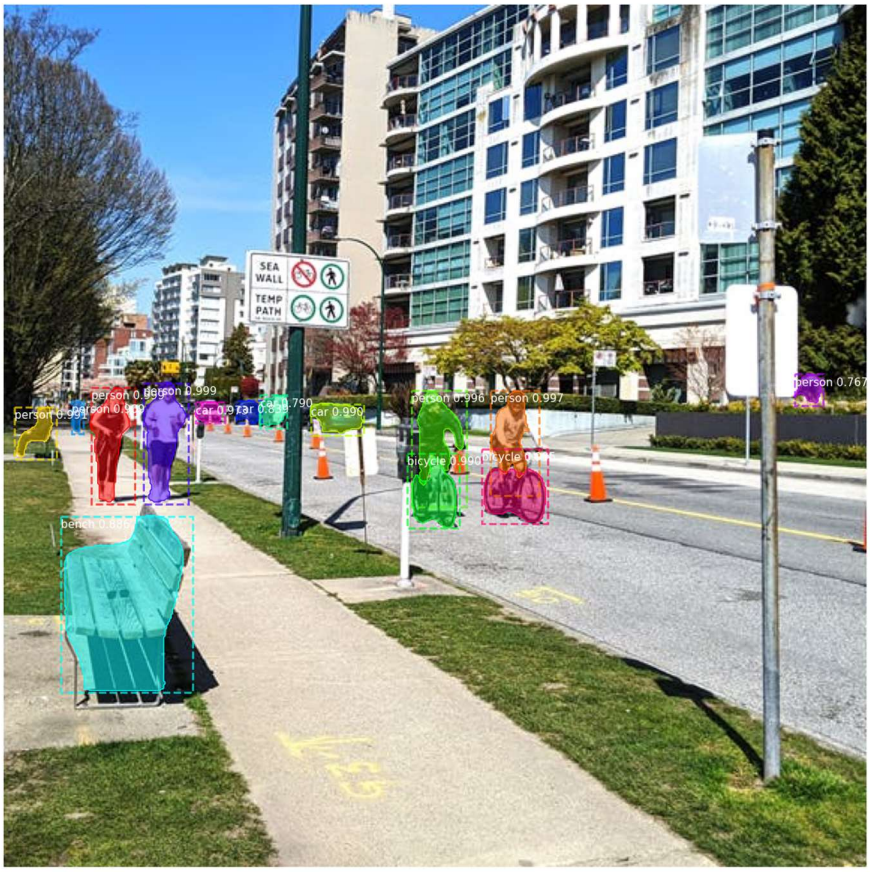
```
image = cv2.imread("images/2502287818_41e4b0c4fb_z.jpg")

image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


r = model.detect([image], verbose=0)

r = r[0]


mrcnn.visualize.display_instances(image=image, boxes=r['rois'],
masks=r['masks'], class_ids=r['class_ids'], class_names=CLASS_NAMES,
scores=r['scores'])
```



сл. 11 Сегментација на инстанци со Mask R-CNN

Примери за сегментација на објекти и урбани средини со Mask R-CNN





сл. 12 Сегментација на инстанци со Mask R-CNN

Заклучок

Во оваа семинарска работа го прикажав создавањето на U-net за семантичка сегментација на слики, како и примената на U-net и Mask R-CNN за семантичка сегментација, односно сегментација на инстанци на објекти во слки.

Во иднина, со повеќе време и ресурси за тренирање и тестирање на мојот U-net Модел, неговата прецизност и ефикасност може да се подобри. Со цел подобрување на моделот, во иднина би можел овој модел да се тренира на поголемо податочно множество, како и да се зголеми бројот на слоеви кои се дел во самата архитектура на моделот.

Користена литература

[1] A. Bajwa, „Image Segmentation with U-Net“. 2024. [Online] Available: analyticsvidhya.com/blog/2022/10/image-segmentation-with-u-net.

[2] „Gentle Introduction to the Adam Optimization Algorithm for Deep Learning“, 2017. [Online]. Available: machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning.

[3] „How ReLU and Dropout Layers Work in CNNs“, 2024. [Online] Available: baeldung.com/cs/ml-relu-dropout-layers.

[4] I. Mihajlovic, „Everything You Ever Wanted To Know About Computer Vision.“ 2019. [Online]. Available: towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.

[5] „Image segmentation detailed overview“. 2023. [Online] Available: superannotate.com/blog/image-segmentation-for-machine-learning.

[6] K. He, G. Gkioxari, P. Dollár, R. Girshick, „ Mask R-CNN “, 2017.

[7] L. Harisha, „Dice Coefficient! What is it?“, 2023. [Online] Available: medium.com/@lathashreeh/dice-coefficient-what-is-it-ff090ec97bda.

[8] M. Walia, „Semantic Segmentation vs. Instance Segmentation: Explained“. 2022. [Online] Available: blog.roboflow.com/difference-semantic-segmentation-instance-segmentation.

[9] M. Mandal, „Introduction to Convolutional Neural Networks (CNN)“ 2021. [Online]. Available: analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn.

[10] N. Buhl, „Guide to Image Segmentation in Computer Vision: Best Practices“. 2022. [Online] Available: encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“, 2014.

[12] O. Ronneberger, P. Fischer, T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation“. 2015.

[13] P. Mahajan, „Max Pooling“ 2020. [Online]. Available: poojamahajan5131.medium.com/max-pooling-210fc94c4f11.

[14] S. Albawi, T. A. Mohammed and S. Al-Azawi, „Understanding of a convolutional neural network“ 2017.

[15] S. Kumar, „Data Augmentation Increases Accuracy of your model — But how ?“ 2019. [Online]. Available: medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722.

[16] „U-Net Architecture Explained“, 2023. [Online] Available: geeksforgeeks.org/u-net-architecture-explained.

[17] „What is Transposed Convolutional Layer?“, 2023. [Online] Available: geeksforgeeks.org/what-is-transposed-convolutional-layer.

[18] „What is Mask R-CNN?“, 2023. [Online] Available: skyengine.ai/se/skyengine-blog/119-what-is-mask-r-cnn.