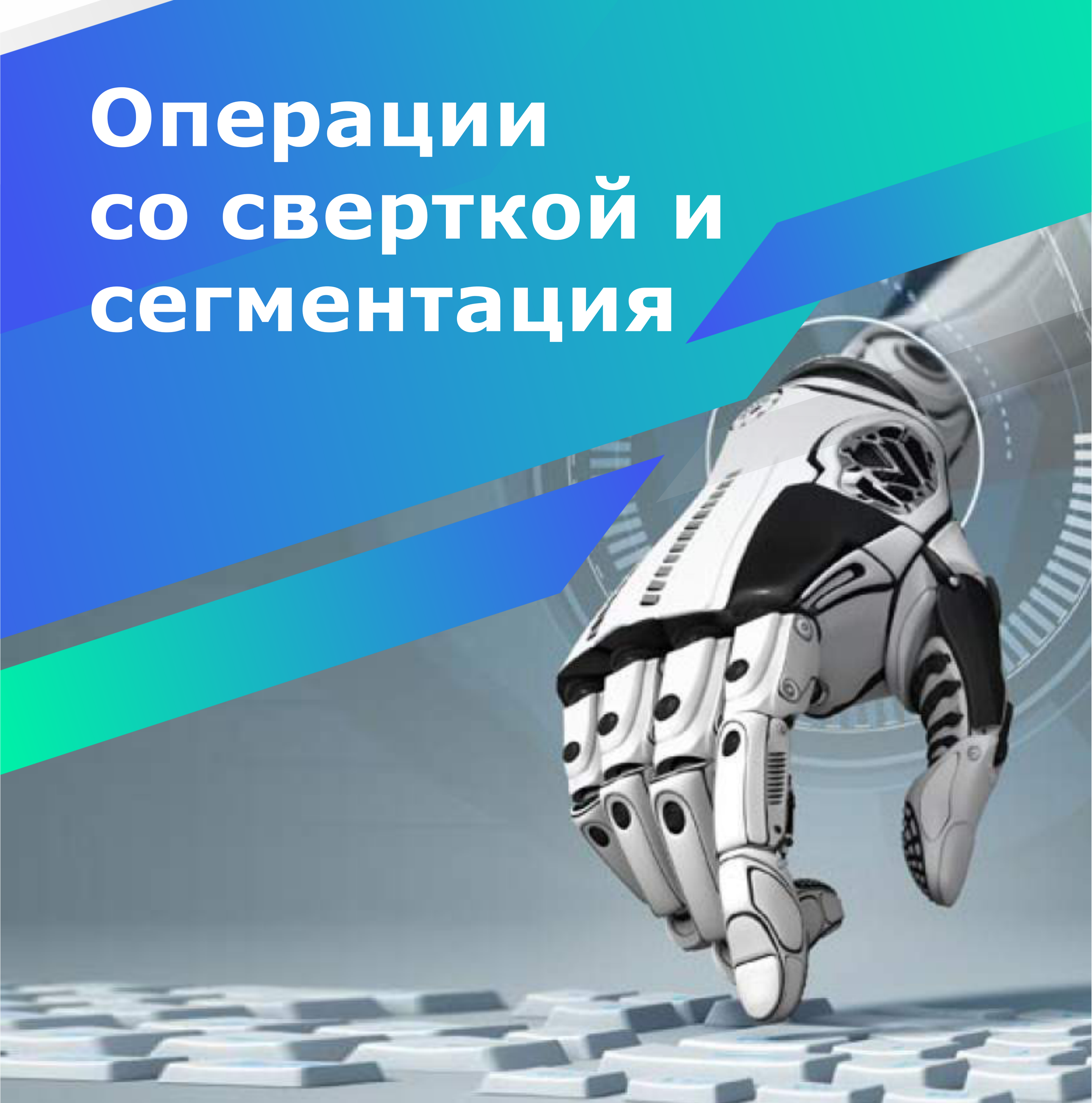
УНИВЕРСИТЕТ
искусственного
интеллекта

Операции со сверткой и сегментация



Операции со сверткой и сегментация

Сегментация – это выделение объектов в исходных данных. Как отдельный класс сегментация изображений заключается в подсвечивании объектов на изображении, другими словами можно сказать, что сегментация изображений – это попиксельная классификация, где каждому пикселю объекта присваивается определенный класс.

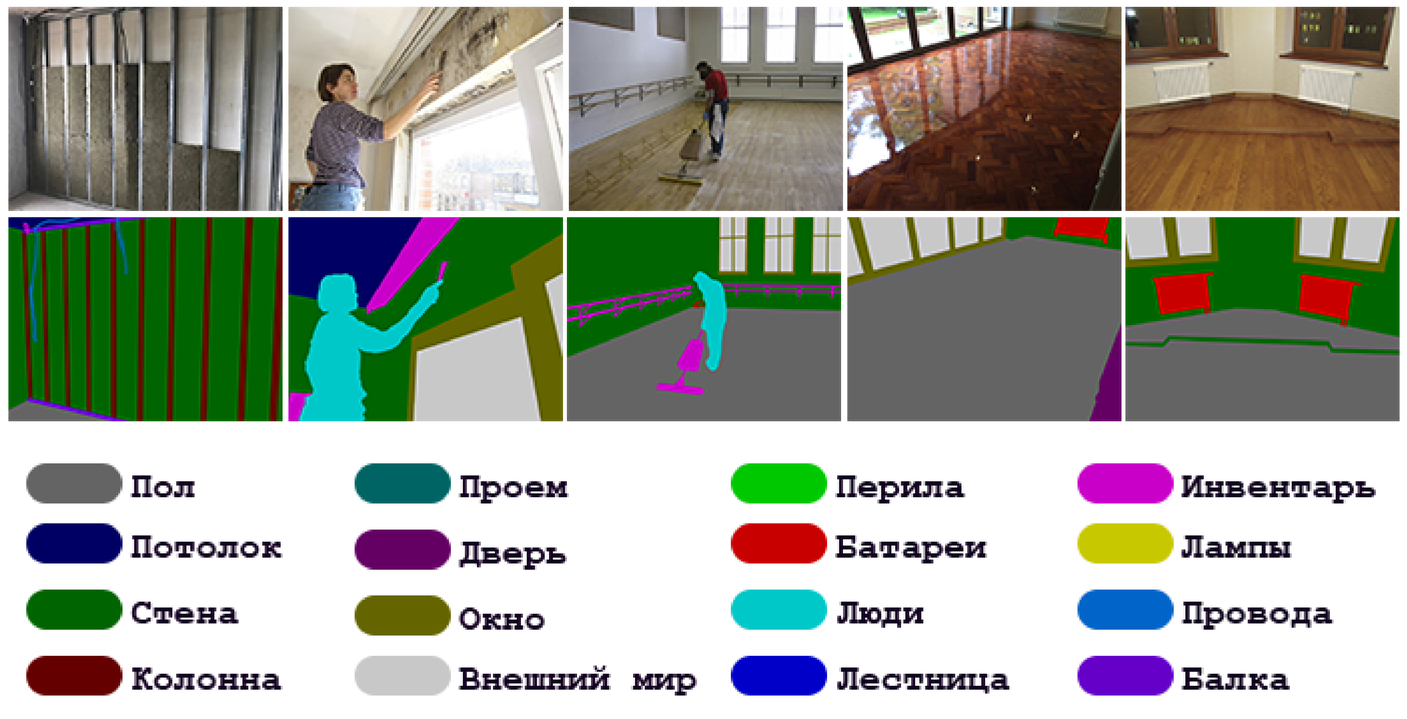


Рисунок 1. Пример сегментации

Изображения стройки сегментированы на 16 классов: пол, потолок, стена, колонна, проем, дверь, окно, внешний мир, перила, батареи, люди, лестница, инвентарь, лампы, провода, балка. Каждому классу выделен свой цвет.

Сегментация применяется во многих областях: в медицине (первичная обработка цифровых снимков, например, рентгеновских), на производстве (определение дефектов детали по фотографии), в машиностроении (машинное зрение для автопилотов).

Операции со сверткой и сегментация

АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ СЕГМЕНТАЦИИ

Для решения задач сегментации изображений существуют различные архитектуры нейронных сетей, рассмотрим некоторые из них.

U-NET

https://arxiv.org/pdf/1505.04597.pdf

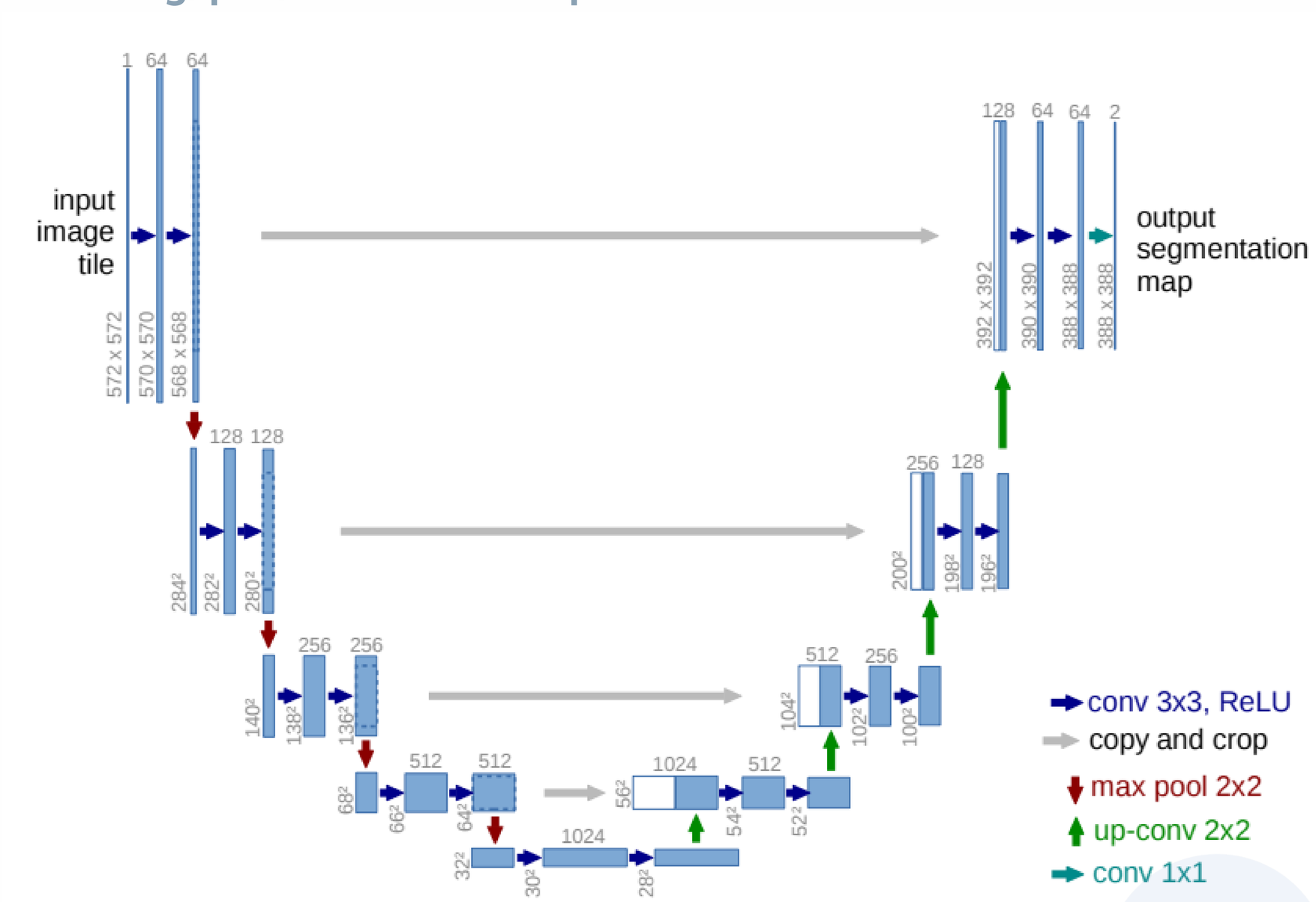


Рисунок 2. Архитектура U-net

По своей структуре сеть схожа с автокодировщиком, в котором сеть сжимает данные в скрытое пространство, тем самым выявляя основные признаки, и потом восстанавливает изображение из скрытого пространства. Архитектура U-net представляет собой последовательность блоков, сначала уменьшая размерность изображения (блоки включают Pooling-слои), а потом увеличивая, предварительно объединив с выходами начальных блоков с соответствующей размерностью (рисунок 2).

Операции со сверткой и сегментация

На Keras U-net может быть реализована в следующем виде:

```
  # Block 1 уменьшения
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv1')(img input)
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv2')(x)
block_1_out = Activation('relu')(x)# Добавляем слой
Activation и запоминаем в переменной block_1_out
x = MaxPooling2D()(block_1_out) # уменьшаем размерность в
2 раза


    # Block 2 уменьшения
x = Conv2D(64, (3, 3), padding='same', name='block2
conv1')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same', name='block2_
conv2')(x)
block_2_out = Activation('relu')(x)# Добавляем слой
Activation и запоминаем в переменной block_2_out
x = MaxPooling2D()(block_2_out)# уменьшаем размерность в 2
раза


    # UP 1 увеличения
x = Conv2DTranspose(64,(2,2),strides=(2,2),padding='same')
(x)
#увеличиваем размерность в 2 раза
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)

x = concatenate([x, block_2_out]) #объединяем со вторым
понижающим слоем
```

Операции со сверткой и сегментация

```
    # UP 2 увеличения
x = Conv2DTranspose(32, (2, 2), strides=(2, 2),
padding='same')(x) #увеличиваем размерность в 2 раза
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = Activation('relu')(x)

x = concatenate([x, block_1_out])#объединяем с первым
понижающим слоем

x = Conv2D(num_classes,(3,3),activation='softmax',padding
='same')(x)
# Выходной слой
```

SEGNET

https://arxiv.org/pdf/1511.00561.pdf

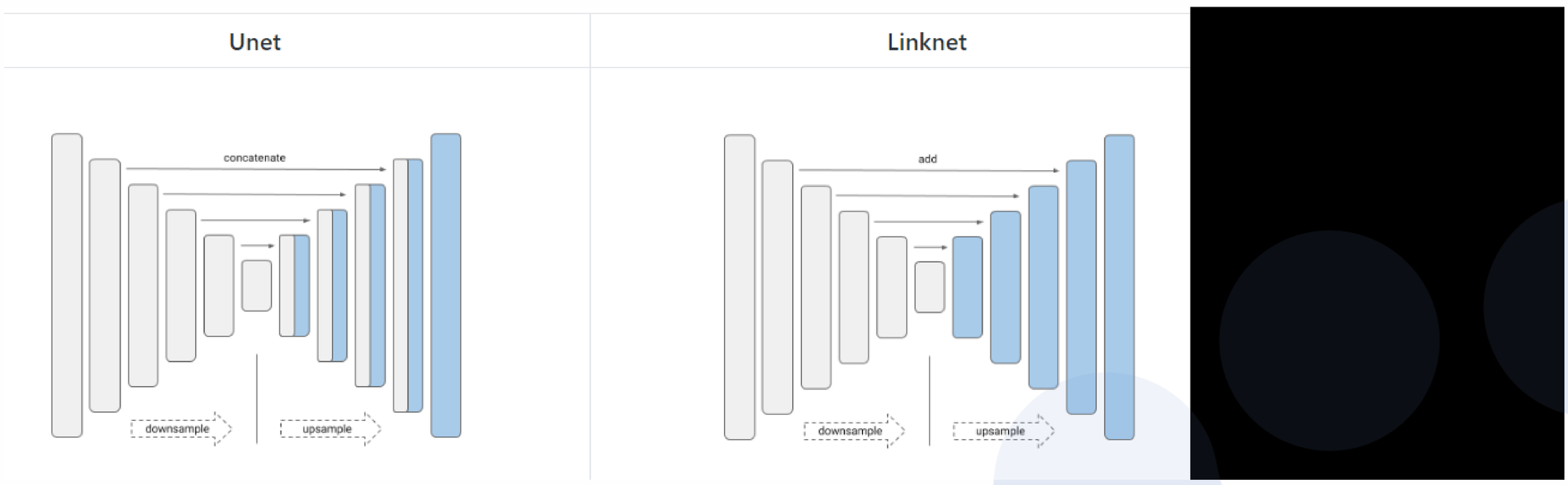


Рисунок 3. Архитектура SegNet

SegNet – это сверточный автокодировщик, последний слой которого – слой попиксельной классификации. Архитектура состоит из сверточных блоков, сначала уменьшая размерность изображения (блоки включают Pooling-слои), а потом увеличивая до исходной размерности.

На Keras SegNet может быть реализована в следующем виде:

Операции со сверткой и сегментация

```
    # Block 1
x = Conv2D(64, (3, 3), padding='same', name='block1_
conv1')(img_input)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same', name='block1_
conv2')(x)
x = MaxPooling2D()(x) # Уменьшаем в два раза

    # Block 2
x = Conv2D(128, (3, 3), padding='same', name='block2_
conv1')(x)
x = Activation('relu')(x)
x = Conv2D(128, (3, 3), padding='same', name='block2_
conv2')(x)
x = MaxPooling2D()(x)# Уменьшаем в два раза

    # Block 3
x = Conv2D(256, (3, 3), padding='same', name='block3_
conv1')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same', name='block3_
conv2')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same', name='block3_
conv3')(x)
x = MaxPooling2D()(block_3_out)# Уменьшаем в два раза

    # Block 4
x = Conv2D(512, (3, 3), padding='same', name='block4_
conv1')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same', name='block4_
conv2')(x)
x = Activation('relu')(x)
```

Операции со сверткой и сегментация

```
x = Conv2D(512, (3, 3), padding='same', name='block4_
conv3')(x)
block_4_out = Activation('relu')(x)

  # UP 1
x = Conv2DTranspose(512, (2, 2), strides=(2, 2),
padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same')(x)

  # UP 2
x = Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same')(x)

  # UP 3
x = Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(128, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(128, (3, 3), padding='same')(x)

  # UP 4
x = Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(x)
```

Операции со сверткой и сегментация

```
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
```

```
x = Conv2D(num_classes, (3, 3), activation='softmax',
padding='same')(x)
```

LINKNET

https://arxiv.org/pdf/1707.03718.pdf

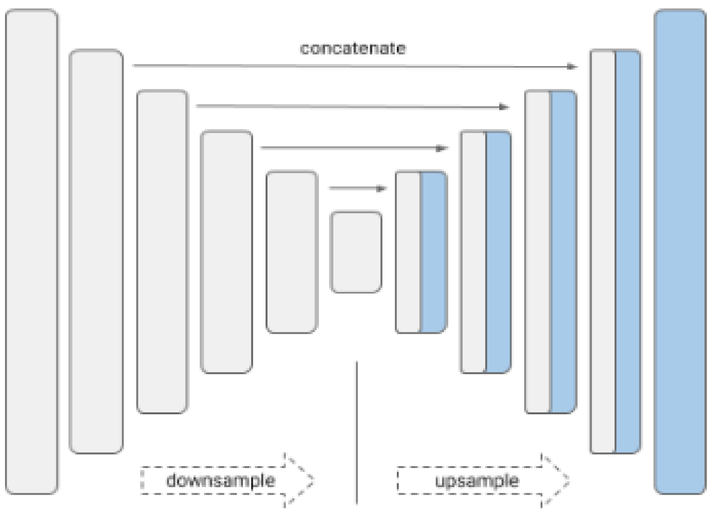
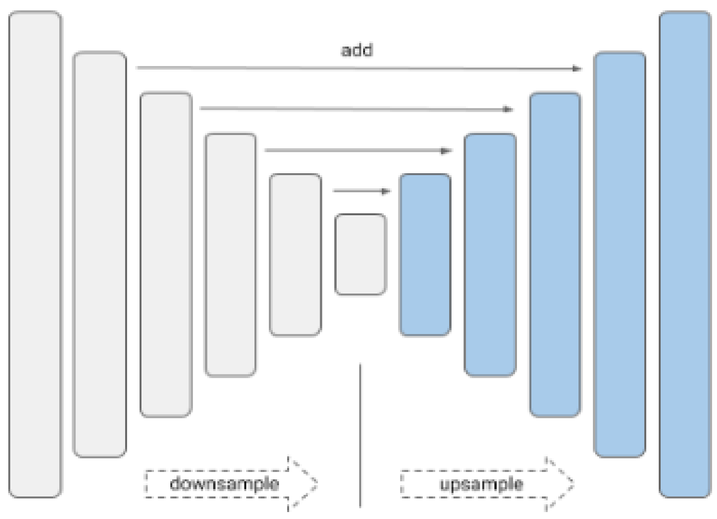
Unet	Linknet
	

рис.3. Архитектура LinkNet

По своей структуре LinkNet – это Unet, в которой слои объединения (concatenate) заменены на слои добавления.

На Keras U-net может быть реализована в следующем виде:

```
  # Block 1 уменьшения
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv1')(img_input)
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv2')(x)
```

Операции со сверткой и сегментация

```
block_1_out = Activation('relu')(x)# Добавляем слой
Activation и запоминаем в переменной block_1_out
x = MaxPooling2D()(block_1_out) # уменьшаем размерность в
2 раза

    # Block 2 уменьшения
x = Conv2D(64, (3, 3), padding='same', name='block2_
conv1')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same', name='block2_
conv2')(x)
block_2_out = Activation('relu')(x)# Добавляем слой
Activation и запоминаем в переменной block_2_out
x = MaxPooling2D()(block_2_out)# уменьшаем размерность в 2
раза

    # UP 1 увеличения
x = Conv2DTranspose(64,(2,2),strides=(2,2),padding='same')(x)
#увеличиваем размерность в 2 раза
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same')(x)
x = Activation('relu')(x)

x = Add()([x, block_2_out]) #добавляем первый понижающим
слоем

    # UP 2 увеличения
x = Conv2DTranspose(32, (2, 2), strides=(2, 2),
padding='same')(x) #увеличиваем размерность в 2 раза
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = Activation('relu')(x)
```

Операции со сверткой и сегментация

```
x = Conv2D(32, (3, 3), padding='same')(x)
x = Activation('relu')(x)

x = Add()([x, block_1_out])#добавляем первый понижающим
слоем

x = Conv2D(num_classes,(3,3),activation='softmax',padding=
'same')(x)
# Выходной слой
```

PSPNet

https://arxiv.org/pdf/1612.01105.pdf

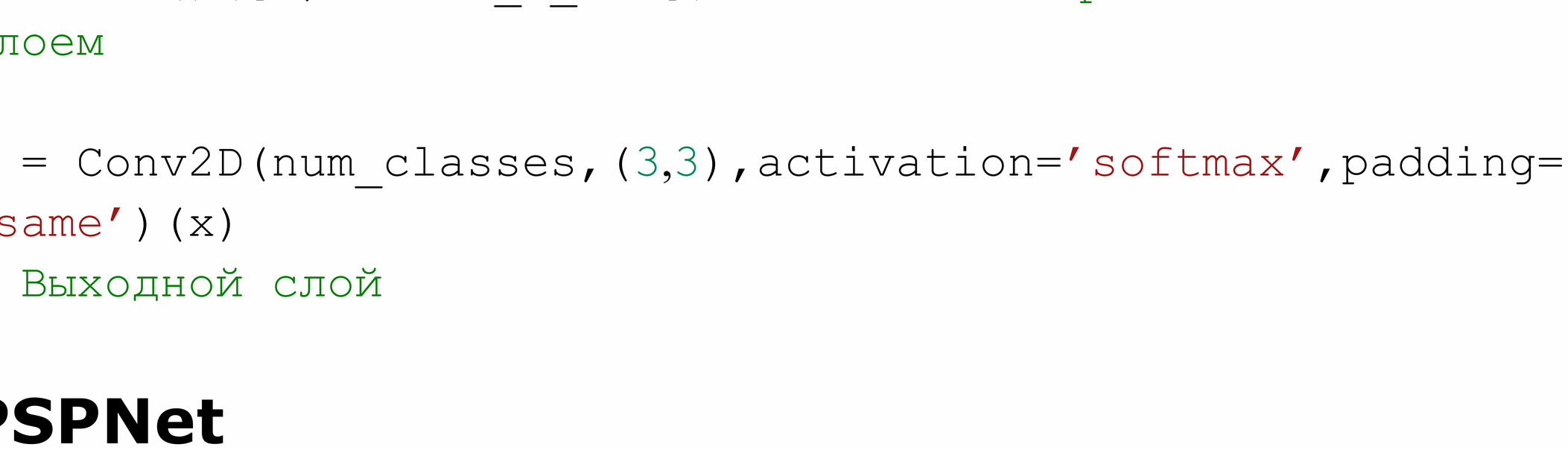


Рисунок 4. Архитектура PSPNet

Архитектура PSPNet построена на модели pooling пирамид, состоит из блока нескольких сверточных слоев. Карта признаков на выходе данного блока проходит через pooling пирамиду. То есть карта признаков проходит через несколько Pooling-слоев с разным ядром. После чего каждый слой в отдельности проходит через сверточные слои и возвращается соответствующая размерность соответствующего слоя (Upsampling, Conv2DTranspose). Далее все слои объединяются (concatenate) и подаются на выходной сверточный слой.

На Keras простая архитектура PSPNet может быть реализована в следующем виде:

```
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv1')(img_input)
```

Операции со сверткой и сегментация

```
x = Activation('relu')(x)
x = Conv2D(32, (3, 3), padding='same', name='block1_
conv2')(x)
# pooling пирамида
block_1_out = MaxPooling2D((2,2))(x)
block_2_out = MaxPooling2D((4,4))(x)
block_3_out = MaxPooling2D((8,8))(x)
block_4_out = MaxPooling2D((16,16))(x)


# Сверточные слои
block_1_out=Conv2D(64,(3,3),padding='same',name='block2_
conv1')(block_1_out)
block_1_out = Activation('relu')(block_1_out)
block_2_out=Conv2D(64,(3,3),padding='same',name='block2_
conv1')(block_2_out)
block_2_out = Activation('relu')(block_2_out)
block_3_out=Conv2D(64,(3,3),padding='same',name='block2_
conv1')(block_3_out)
block_3_out = Activation('relu')(block_3_out)
block_4_out=Conv2D(64,(3,3),padding='same',name='block2_
conv1')(block_4_out)
block_4_out = Activation('relu')(block_4_out)

# Увеличиваем размерность
block_1_out=Conv2DTranspose(32,(3,3),strides=(2,2),
        padding='same')(block_1_out)
block_1_out = Activation('relu')(block_1_out)

block_2_out=Conv2DTranspose(32,(3,3),strides=(4,4),
        padding='same')(block_2_out)
block_2_out = Activation('relu')(block_2_out)

block_3_out=Conv2DTranspose(32,(3,3),strides=(8,8),
        padding='same')(block_3_out)
```

Операции со сверткой и сегментация

```
block_3_out = Activation('relu')(block_3_out)

block_4_out=Conv2DTranspose(32,(3,3),strides=(16,16),
        padding='same')(block_4_out)
block_4_out = Activation('relu')(block_4_out)

#объединяем слои
out = concatenate([block_1_out, block_2_out, block_3_out,
block_4_out])
out=Conv2D(num_classes,(3,3),activation='softmax',padding=
'same')(out)
# Выходной слой
```

АУГМЕНТАЦИЯ

Аугментация – изменение исходного изображения, например, зеркальное отображение, поворот на определенный угол, вырезание определенной части изображения, искажение изображений и т. д. Применяется для увеличения обучающей выборки.

Одна из таких библиотек – **Albumentations**, инструкцию с примерами кода в google colaboratory можно найти по ссылке https://github.com/albu/albumentations.