U-Net Segmentation

W pierwszej kolejności należy ściągnąć dane wykonując poniższe komendy w terminalu lub rozpakować je w inny sposób.

EN: The first step is to download the data by executing the commands below in the terminal or extract it in another way.

```
wget https://www.robots.ox.ac.uk/~vgg/data/pets/data/images.tar.gz
wget https://www.robots.ox.ac.uk/~vgg/data/pets/data/annotations.tar.gz
tar -xf images.tar.gz
tar -xf annotations.tar.gz
```

Dane przedstawiają obrazy zwierząt, łącznie z maskami segmentacji oznaczającymi lokalizację obiektu na zdjęciu z dokładnością do piksela. Zdjęcia należy wczytać jako dwie osobne posortowane listy, jedną zawierającą zdjęcia, drugą maski. Należy zwrócić uwagę, aby i-te zdjęcie w pierwszej liście odpowiadało i-tej masce w drugiej liście. Ponadto, ze względu na ilość i rozmiar zdjęć, mogą one nie zmieścić się w pamięci, dlatego dobrym pomysłem będzie przechowywanie ich ścieżek i wczytywanie w trakcie uczenia.

EN: The data represents images of animals, along with segmentation masks indicating the object's location in the image with pixel-level accuracy. The images should be loaded as two separate sorted lists: one containing images and the other containing masks. It is important to ensure that the i-th image in the first list corresponds to the i-th mask in the second list. Additionally, due to the quantity and size of the images, they may not fit into memory. Therefore, a good idea would be to store their paths and load them during training.

```
#TODO Ustaw niezbędne ścieżki
# wybierz parametry zależnie od możliwości sprzętowych
# sprawdź czy dane pobrały się poprawnie.

# TODO Set the necessary paths
# Choose parameters depending on hardware capabilities
# Check if the data has been downloaded correctly.

import os
input_dir = #
target_dir = #
img_size = #
num_classes = # Object, background, uncertain area between
batch_size = #
input_img_paths = # folder + file path
target_img_paths =
```

Przykładowe zdjęcie oraz maskę można wyrysować z użyciem funkcji imshow z modułu `pyplot` biblioteki `matplotlib`:

EN: You can plot a sample image and mask using the `imshow` function from the `pyplot` module of the `matplotlib` library:

```
# TODO wyświetl przykładowe zdjęcie i maskę.
# TODO show an example image and the corresponding mask
```

Aby móc wczytywać zdjęcia w trakcie treningu, należy zaimplementować odpowiedni generator. Żaden z dostępnych generatorów nie wczytuje obrazów i ich masek. Dlatego należy napisać własny generator. Klasa taka musi dziedziczyć po klasie `keras.utils.Sequence` i implementować metody: `_ len_(self)` oraz `_getitem_(self, idx)`

Alternatywnie można skorzystać z gotowych narzędzi i użyć dwóch osobnych generatorów - jednego który wczytuje obrazy i drugiego który wczytuje maski, a następnie połączyć je w jeden.

EN: To be able to load images during training, it is necessary to implement an appropriate generator. None of the available generators loads images and their masks. Therefore, it is necessary to write your own generator. Such a class must inherit from the keras.utils.Sequence class and implement the methods: len(self) and getitem(self, idx).

Alternatively, you can use existing tools and use two separate generators - one that loads images and another that loads masks, and then combine them into one.

```
# TODO Aby nasza własna funkcja działa jak generator i mogła być używana np.
przez fit musi mieć trzy definicje: init, len i getitem
# For our custom function to function as a generator and be usable, for
example, by fit, it must have three definitions: __init__, __len__, and
__getitem__.

class DataGen(keras.utils.Sequence):
    def __init__(self, batch_size, img_size, input_img_paths,
target_img_paths):

        #TODO


    def __len__(self):


        #TODO

    def __getitem__(self, idx):
        """Returns tuple (input, target) correspond to batch #idx."""

        #TODO

        return x, y
```

Metoda `_getitem_` jest wywoływana dla każdego batcha raz w trakcie trwania jednej epoki poprzez przekazanie indeksu. Metoda powinna zwracać krotkę dwuelementową, gdzie pierwszym elementem jest zestaw obrazów o rozmiarze (wielkość batcha, wys, szer, l. kanałów), natomiast drugi element reprezentuje zestaw masek dla załadowanych obrazów. Tak przygotowany obiekt klasy przekazuje się do metody fit która implementuje proces uczenia.Wcześniej jednak należy zaimplementować architekturę sieci. Zgodnie z treścią wykładu, ma ona kształt przypominający literę U a operacje

zawarte w niej to konwolucje, operacje max pooling i konwolucje transponowane. Wejściem sieci jest obraz, natomiast wyjściem maska. Dobrze jest zacząć od definicji funkcji pomocniczych. Bloku konowlucji:

EN: The `getitem` method is called for each batch once during one epoch by passing an index. The method should return a tuple with two elements, where the first element is a set of images with the size (batch size, height, width, number of channels), and the second element represents the set of masks for the loaded images. This prepared object of the class is passed to the fit method, which implements the training process. However, it is necessary to implement the architecture of the network beforehand. According to the lecture content, it has a U-shaped structure, and the operations within it include convolutions, max pooling operations, and transposed convolutions. The input to the network is an image, while the output is a mask. It is good to start with the definition of helper functions. A convolutional block:

```
from tensorflow.keras import layers
def double_conv_block(x, n_filters):
    x = layers.Conv2D(#TODO)(x)
    x = layers.Conv2D(#TODO)(x)
    return x

#remember to add padding
```

Bloku podprobkowania:

EN: Sampling block:

```
def downsample_block(x, n_filters):
    f = double_conv_block(x, n_filters)
    p = layers.MaxPool2D(#todo)(f)
    p = layers.Dropout(#todo)(p)
    return f, p
```

Oraz bloku nadpróbkowania:

EN: Upsampling block:

```
def upsample_block(x, conv_features, n_filters):
    x = layers.Conv2DTranspose(n_filters, 3, 2, padding="same")(x)
    x = layers.concatenate([x, conv_features])
    x = layers.Dropout(#todo)(x)
    x = double_conv_block(#todo)
    return x
```

Mając do dyspozycji wszystkie operacje, można zaimplementować sieć:

EN: Given all these operations, we can implement the network:

```
import tensorflow as tf
def get_model(img_size, num_classes):
    inputs = layers.Input(shape=(*img_size, 3))
    f1, p1 = downsample_block(inputs, 64)
```

```
    f2, p2 = downsample_block(p1, 128)
    f3, p3 = downsample_block(p2, 256)
    f4, p4 = downsample_block(p3, 512)
    bottleneck = double_conv_block(p4, 1024)
    u6 = upsample_block(bottleneck, f4, 512)
    u7 = upsample_block(u6, f3, 256)
    u8 = upsample_block(u7, f2, 128)
    u9 = upsample_block(u8, f1, 64)
    outputs = layers.Conv2D(num_classes, 1, padding="same",
activation="softmax")(u9)
    unet_model = tf.keras.Model(inputs, outputs, name="U-Net")
    return unet_model
keras.backend.clear_session()
model = get_model(img_size, num_classes)
```

Aby zweryfikować archietkurę sieci, można wypisać ją na ekran korzystając z `model.summary()`.

Dobrą praktyką jest podział zbioru na podzbiory uczący, walidacyjny i testowy. Należy podzielić zbiór danych na odpowiednie podzbiory. Dla uproszczenia wystarczą dwa zbiory – uczący i walidacyjny

EN: To verify the network architecture, you can print it to the screen using `model.summary()`. A good practice is to split the dataset into training, validation, and test subsets. It is necessary to divide the dataset into appropriate subsets. For simplicity, two sets are sufficient – training and validation.

```
# TODO Podziel dane na zbiory treningowy i walidacyjny (na bazie ścieżek)
# TODO: Divide the data into training set and validation set (based on
paths)
```

a następnie utworzyć obiekty generatorów: as a next step, create generator objects:

```
train_gen = #TODO
val_gen = #TODO
```

Przed samym rozpoczęciem uczenia, należy jeszcze zdefiniować funkcję kosztu i skompilować model. Dodaktowo, warto ustawić callback umożliwiający zapis najlepszego modelu. Na samym końcu wywołujemy funkcję fit z odpowiednimi argumentami.

EN: Before starting the training itself, it is necessary to define the loss function and compile the model. Additionally, it is advisable to set up a callback to save the best model. Finally, the fit function is called with the appropriate arguments.

```
model.compile(#TODO)
model.fit(#TODO)
```

W trakcie treningu biblioteka keras loguje w czasie rzeczywistym postęp. Czas uczenia może się wahać w zależności od dostępnego sprzętu. W przypadku zbyt długiego czasu oczekiwania należy zmniejszyć zbiór danych lub liczbę epok. Proszę jednak pamiętać, że zbyt krótki czas uczenia może skutkować słabą skutecznością modelu

Skuteczność działania modelu można ocenić metryką (np. accuracy), choć przeważnie nie stosuje się jej do problemu segmentacji. W ramach ćwiczenia dodatkowego można napisać i podpiąć jedną z

metryk do segmentacji (np. Dice, Jaccard).

W razie problemów z wykonaniem któregoś fragmentu instrukcji można posłużyć się gotowym skryptem.

EN: During training, the Keras library logs real-time progress. The training time may vary depending on the available hardware. If the waiting time is too long, consider reducing the dataset or the number of epochs. However, please note that excessively short training time may result in poor model performance.

The effectiveness of the model can be assessed using a metric (e.g., accuracy), although it is generally not applied to segmentation problems. As an additional exercise, you can write and attach one of the metrics for segmentation (e.g., Dice, Jaccard).

In case of difficulties in executing any part of the instructions, you can use a ready-made script.

Skrypt

https://arxiv.org/abs/1505.04597