Animal photo segmentation

Task 1

Design a segmenter (multiclass classification per pixel) of animal photos:

https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet

https://www.robots.ox.ac.uk/~vgg/data/pets/

The data is taken from the server in the form of a tensorflow dataset. API description:

https://www.tensorflow.org/guide/data

https://www.tensorflow.org/api_docs/python/tf/data/Dataset

Tip 1:

- ds.map(): data preprocessing and augmentation

- ds.shuffle(): shuffle data before each epoch or iteration

- ds.batch(): batch size setting

- ds.repeat(): repeating examples from a dataset after analyzing all of them elements

Tip 2:

Training examples should be mixed at least once before each epoch.

Tip 3:

The number of classes in the dataset is 3.

GPU can be used to train the network if available.

For this purpose go to *Runtime -> Change runtime type* and select *GPU* as *Hardware accelerator*.

At the top of the notebook add the following line just in case:

!pip install -U tensorflow_datasets

You can get started with the following tutorial:

https://www.tensorflow.org/tutorials/images/segmentation

But:

• transfer learning is prohibited. The model architecture needs to be done from scratch using Sequential or Functional API in Keras.

• SparseCategoricalCrossentropy must have a from_logits argument

set to False (the last layer of the model must have a function

Softmax activation)

• 30% of the test set should be allocated to the validation set

(https://www.tensorflow.org/datasets/splits)

• F1 Score and accuracy should be used as metrics

Recommended treatments to test:

• reducing the size of images to speed up training

• augmentation of training images, e.g. by rotations, mirror images, slight noise, brightness change, contrast change (augmentations modifying the pixel position of the class in the image must be done also in label images)

• normalization to the range <-0.5, 0.5>, <-1, 1> or <0, 1>

• class weighting due to unequal number of pixels between classes (tutorial)

• BatchNormalization after convolutional and dense layers with large

learning rate (0.01 - 0.1)

• segmenter architecture:

 - Fully Convolutional (with only resize at the end or encoder decoder)

 - Unet

• Upsampling in the model can be done by layer Upsampling2D in Keras (nearest neighbors method)