OCTseg

Mohammad Haft-Javaherian

Aug 02, 2019

CONTENTS:

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

TWO

UTIL

2.1 load data

Convert an 2D or 3D image from polar or cylindrical coordinate to the cartesian coordinate.

util.load_data.**im_fix_width**(*im*, *w*)
pad or crop the 3D image to have width and length equal to the input width

2.2 polar to cartesian

Convert an 2D or 3D image from polar or cylindrical coordinate to the cartesian coordinate.

2.3 process oct folder

process OCT folder to generate the segmentation labels of cases. Each case all three -.PSTIF, -.INI, and -ROI.txt files

2.4 read oct roi file

Read ROI file generated based on the and generate segmentation results.

util.read_oct_roi_file.**lumen_iel_mask**(*obj_list*, *im_shape*)
generate lumen or IEL mask based on the point list.

util.read_oct_roi_file.**roi_file_parser**(*file_path*)
Parse roi file and output the lists of objects

CHAPTER

THREE

UNET

3.1 loss

CNN related loss functions

unet.loss.**dice_loss**(*label*, *target*)

soft Dice coefficient loss TP, FP, and FN are true positive, false positive, and false negative.

$$dice = \frac{2 \times TP}{2 \times TP + FN + FP}$$
$$dice = \frac{2 \times TP}{(TP + FN) + (TP + FP)}$$

objective is to maximize the dice, thus the loss is negate of dice for numerical stability (+1 in denominator) and fixing the loss range (+1 in numerator and +1 to the negated dice) The final Dice loss is formulated as

$$dice\ loss = 1 - \frac{2 \times TP + 1}{(TP + FN) + (TP + FP) + 1}$$

it is soft as each components of the confusion matrix (TP, FP, and FN) are estimated by dot product of probability instead of hard classification

Parameters

- **label** – 4D or 5D label tensor
- **target** – 4D or 5d target tensor

Returns dice loss

unet.loss.**multi_loss_fun**(*loss_weight*)

semantic loss function based on the weighted cross entropy and dice and wighted by the loss weights in the input argument

Parameters **loss_weight** – a list with two weights for weighted cross entropy and dice losses, respectively.

Returns

return a function, which similar to *weighted_cross_entropy()* and *dice_loss()*
has label and target arguments

Seemore *weighted_cross_entropy()*, *dice_loss()* :

unet.loss.**weighted_cross_entropy**(*label*, *target*)

weighted cross entropy with foreground pixels having ten times higher weights

Parameters

- **label** – 4D or 5D label tensor
- **target** – 4D or 5d target tensor

Returns weighted cross entropy value

TODO add positive weight as an argument

3.2 ops

CNN related operations

unet.ops.**accuracy**(*labels*, *logits*)
measure accuracy metrics

unet.ops.**img_aug_carts**(*im*, *l*)
Data augmentation in Cartesian

unet.ops.**img_aug_polar**(*im*, *l*)
Data augmentation in Polar coordinate

unet.ops.**img_rand_scale**(*im*, *scale*, *order*)
scale one image batch

unet.ops.**placeholder_inputs**(*im_shape*, *outCh*)
Generate placeholder variables to represent the input tensors.

3.3 unet

Build unet model

unet.unet.**unet_model**(*im_shape*, *nFeature=32*, *outCh=2*)
Build U-Net model.

Parameters

- **x** – input placeholder
- **outCh** – number of output channels

Returns keras model

PYTHON MODULE INDEX

u

INDEX