OCTseg

Mohammad Haft-Javaherian

Aug 12, 2019

CONTENTS:

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

TWO

TRAINING AND TESTING

CHAPTER

THREE

U-NET

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.**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*, *nLayer=3*)
Build U-Net model.

Parameters

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

Returns keras model

CHAPTER

FOUR

UTILITY

4.1 load batch

Load a batch of data.

Creates batches of data randomly in serial or multi-thread parallel fashion.

util.load_batch.**img_aug**(*im*, *l*, *coord_sys*, *p_lim=0.5*)

Image augmentation manager.

Based on the coordinate system (*Polar* vs. *Cartesian*), it selects the corresponding method.

Parameters

- **im** – input image 4D or 5D tensor
- **l** – input label 4D or 5D tensor
- **coord_sys** – coordinate system. 'polar' or 'carts' for Polar and Cartesian, respectively.
- **p_lim** – probability limit for applying each augmentation case.

Returns augmented im and l

Seemore *img_aug_carts()*, *img_aug_polar()*

util.load_batch.**img_aug_carts**(*im*, *l*, *p_lim=0.5*)

Data augmentation in Cartesian coordinate system.

Applies different image augmentation procedures:

- mirroring the image along 45 degree (y=x line)
- mirroring the image along the x axis
- mirroring the image along the y axis
- mirroring the image along the z axis for 3D images
- multiple 90 degree rotations
- image intensity scaling by multplying the intensity values with close to one scale value
- image scaling. See *img_rand_scale()*

based on the input probability limit probabilistically applies different augmentation cases.

Parameters

- **im** – input image 4D or 5D tensor
- **l** – input label 4D or 5D tensor

- **p_lim** – probability limit for applying each augmentation case.

Returns augmented im and l

Seemore *img_aug()*

util.load_batch.**img_aug_polar**(*im*, *l*, *p_lim=0.5*)

Data augmentation in Polar coordinate.

Applies different image augmentation procedures:

- random rotations
- image intensity scaling by multplying the intensity valuse with close to one scale value
- image scaling, which randomly crops or add pads and scale the image to the original size

based on the input probability limit probabilistically applies different augmentation cases.

Parameters

- **im** – input image 4D or 5D tensor
- **l** – input label 4D or 5D tensor
- **p_lim** – probability limit for applying each augmentation case.

Returns augmented im and l

Seemore *img_aug()*

util.load_batch.**img_rand_scale**(*im*, *scale*, *order*)

Scale one image or label batch in Cartesian coordinate system.

scale the image based on the input scale value and interpolation order followed by cropping or padding to maintain the original image shape. For interpolation close to the boundaries, the reflection mode is used.

Parameters

- **im** – 3D or 4D image or label tensor
- **scale** – scalar scale values for x and y direction
- **order** – interpolation order

Returns same size image with the scale image in the center of it.

util.load_batch.**load_batch**(*im*, *datasetID*, *nBatch*, *label=None*, *isAug=False*, *coord_sys='carts'*)

load a batch of data from im and/or label based on dataset (e.g. test).

This function handel different coordinate system and image augmentation.

Parameters

- **im** – 4D or 5D image tensor
- **datasetID** – index of images in im and/or label along the first axis, which belong to this dataset (e.g. test)
- **nBatch** – batch size
- **label** – 4D or 5D label tensor
- **isAug** – whether to apply data augmentation. See *img_aug()*
- **coord_sys** – coordinate system {'polar' or 'carts}

Returns a batch of data as tuple of (image, label)

Seemore *load_batch_parallel()*

util.load_batch.**load_batch_parallel**(*im*, *datasetID*, *nBatch*, *label=None*, *isAug=False*, *coord_sys='carts'*)

load a batch of data from im and/or label based on dataset (e.g. test) using multi-thread.

This function handel different coordinate system and image augmentation.

Parameters

- **im** – 4D or 5D image tensor
- **datasetID** – index of images in im and/or label along the first axis, which belong to this dataset (e.g. test)
- **nBatch** – batch size
- **label** – 4D or 5D label tensor
- **isAug** – whether to apply data augmentation. See *img_aug()*
- **coord_sys** – coordinate system {'polar' or 'carts}

Returns a batch of data as tuple of (image, label)

Seemore *load_batch()*

4.2 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

4.3 plot log file

plot the log file within the save model folders.

4.4 polar to cartesian

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

4.5 process oct folder

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

4.6 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

PYTHON MODULE INDEX

u

INDEX