Getting the Most Out Of U-Net for Calving Front Segmentation

Maniraman Periyasamy

Mar 24, 2021

Contents

CHAPTER 1

Example Pipeline

```
python main.py --parameter=hyperparameters/hyperparameters.yaml
python plot.py --parameter=hyperparameters/hyperparameters_reference.yaml
python preprocess.py --parameter=hyperparameters/hyperparameters_reference.yaml
```

CHAPTER 2

Modules Implemented

2.1 data_generator module

Author: Maniraman Periyasamy

This module generates the keras data generator object which is to be used for training the U-Net model

data_generator.**adjustData**(*img*, *mask*, *flag_multi_class*, *num_class*)

Adjust the masks to binary or multiclass problem.

Parameters

- **img** (*ndarray*) -- image
- **mask** (*ndarray*) -- mask for the image.
- **flag_multi_class** (*bool*) -- Flag indicating whether the data is a binary class data or not
- **num_class** (*int*) -- Number of classes in the dataset

Returns: tuple(image, mask)

data_generator.**trainGenerator**(*batch_size*, *train_path*, *image_folder*, *mask_folder*, *aug_dict*, *image_color_mode='grayscale'*, *mask_color_mode='grayscale'*, *image_save_prefix='image'*, *mask_save_prefix='mask'*, *flag_multi_class=False*, *num_class=2*, *save_to_dir=None*, *target_size=256, 256*, *seed=1*)

Generate the keras Data Generator object for the given dataset

Parameters

- **batch_size** (*int*) -- Batch size
- **train_path** (*str*) -- Relative path to the data
- **image_folder** (*str*) -- Name of the folder where images are saved in the train_path
- **mask_folder** (*str*) -- Name of the folder where masks are saved in the train_path

- **aug_dict** (*dict*) -- Transformation properties
- **image_color_mode** (*str, default is 'grayscal'*) -- Colorspace of the image
- **mask_color_mode** (*str, default is 'grayscal'*) -- Colorspace of the mask
- **image_save_prefix** (*str, default is 'image'*) -- Prefix to be used for the image name
- **mask_save_prefix** (*str, default is 'mask'*) -- Prefix to be used for the mask name
- **flag_multi_class** (*bool, default is False*) -- Flag indicating whether the data is a binary class data or not
- **num_class** (*int, default is 2*) -- Number of classes in the dataset
- **save_to_dir** (*str, default is None*) -- Directory where the sample is to be saved for viewing
- **target_size** (*tuple(int, int), default is (256, 256)*) -- Required size of the images.
- **seed** (*int*) -- Random seed to be used.

Returns: None

2.2 data_preprocess module

Author: Maniraman Periyasamy

This module generates the train, validation and test dataset.

The raw input images are first splited into train, validation and test dataset and then pre-processed using bilateral and CLAHE filter. The pre-processed image is augumented eight folds using geomentric transformations and then patched into small patches in case of train and validation data. Test data is saved as such.

data_preprocess.**generateData**(*filePath='../Dataset3', folder='data3_final_256'*)

This function performs the following functionality

- Splits the data into train, validation and test set
- Pre-process the data
- Auguments the data

Note: Data Augumentation and pre-processing are hardcoded into the function. Hence, it has to manually modified if required.

Parameters

- **filePath** (*str*) -- Relative path to the raw input images.
- **folder** (*str*) -- name of the folder to be generated where all the train, test and validation dataset will be saved.

Returns: None

2.3 main module

Author : Maniraman Periyasamy

This module is the main python file which generates and loads the U-Net model based on the hyperparameters preset in the yaml file given as command line argument.This file is structured in such a way that all combination of parametrs given in yaml file as a list will be run sequentially. for example, if the epoch and batchsize parameters in the yaml file are [e1,e2,e3] and [b1,b2,b3] respectively, then 9 differents model will be train with 3X3 combinations sequentially. Each model and its results will be saved in a different folder.

This was done so that multiple model can be trained one after the other without waiting for the user to start training of the next model.

2.4 model module

Author: Maniraman Periyasamy

This module implements various types of U-Net model and provides train and test fuctions using Keras API. The list of models implemented are as follows :

1) U-Net as proposed by Ronneberger et al. in paper "U-Net: Convolutional Networks for Biomedical Image Segmentation"
2) U-Net as suggested by Zhang et al. in paper "Automatically delineating the calving front of Jakobshavn Isbræ from multitemporal TerraSAR-X images: a deep learning approach"
3) U-Net baseline segmentation model as given in report
4) **U-Net baseline segmentation model with various normalization layers**
    a) Layer Normalization
    b) Group Normalization
    c) Instance Normalization
    d) Weight Normaliztion
5) U-Net basline segmentation model with Dropouts instead of normalization
6) U-Net basline segmentation model with Dropouts and Batch Normalization
7) FCNN as given in report
8) Nested U-Net model as suggested by Zhou et al. in paper "UNet++: A Nested U-Net Architecture for Medical Image Segmentation"

class model.**TimedStopping**(*seconds=None*, *verbose=0*)

Bases: `tensorflow.python.keras.callbacks.Callback`

Adapted from: https://github.com/keras-team/keras/issues/1625

on_batch_end(*batch*, *logs={}*)

A backwards compatibility alias for *on_train_batch_end*.

on_epoch_end(*epoch*, *logs={}*)

Called at the end of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

Parameters

- **epoch** -- Integer, index of epoch.
- **logs** -- Dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with *val_*.

on_train_begin(*logs={}*)
Called at the beginning of training.

Subclasses should override for any actions to run.

Parameters logs -- Dict. Currently no data is passed to this argument for this method but that may change in the future.

class model.**unet**(*trainGenerator, valGenerator, stepsPerEpoch, validationSteps, outputPathCheck, outputPath, testPath, patchSize, epochs=100, unetType='unet', loss='binary_crossentropy', loss_weight=[1], metrics=['accuracy'], optimizer=<tensorflow.python.keras.optimizer_v2.adam.Adam object>, inputSize=(256, 256, 1), patience=30, pretrainedWeights=None, pretrainedModel=None, threshold=0.5, validationPath=None, dilationRate=None, lossType=<function binary_crossentropy>, lossWeights=[1.0, 0.0], model_train=True, dropout=0.0, transferTrain=None*)
Bases: `object`

creates a U-Net model based on the parameters given in the initializer. This class also provides the train and test fuction which fits and predicts the U-Net model using Keras API

Parameters

- **trainGenerator** -- Keras data generator with train dataset
- **valGenerator** -- Keras data generator with validation dataset
- **stepsPerEpoch** (*int*) -- Number of train steps in an training epoch
- **validationSteps** (*int*) -- Number of validation steps in an training epoch
- **outputPathCheck** (*str*) -- Relative path to the output folder and the output model name.
- **outputPath** (*str*) -- Relative path to the output folder
- **testPath** (*str*) -- Relative path to the test data folder
- **patchSize** (*int*) -- Number of pixels in one side of the square input patch
- **epochs** (*int, default is 100*) -- Maximum number of epochs
- **unetType** (*str, default is unet*) -- Type of net to be tested
- **loss** (*keras loss, default is BCE*) -- Loss function to be used
- **loss_weight** (*float, default is 1.0*) -- fraction of loos function to be considered
- **metrics** (*list(str), default is ['accuracy']*) -- List of keras metrics
- **optimizer** (*keras optimizer, default is Adam*) -- Optimizer function to be used
- **inputSize** (*tuple(int,int,int), default is (256,256,1)*) -- Dimensions of the input to the model.
- **patience** (*int, default is 30*) -- Number of epochs to wait for validating the convergence.

- **pretrainedWeights** (*str, default is None*) -- Relative path to the pre-trained weights file
- **pretrainedModel** (*str, default is None*) -- Relative path to the pre-trained model file which includes the loss function and optimizer details.
- **threshold** (*float, default is 0.5*) -- Default threshold value to be used for binary classification if optimal threshold is not calculated
- **validationPath** (*str, default is None*) -- Relative path to the validation dataset
- **dilationRate** (*int, default is None*) -- Dialation rate for dilated convolutions
- **lossType** (*str, default is BCE*) -- Type of loss
- **lossWeights** (*tuple(int,int), default is (1.0,0.0)*) -- Weights for BCE and Dice loss if the lossType is 'combined'
- **model_train** (*bool, default is True*) -- Flag to indicate whether the model is to be trained
- **dropout** (*float, default is 0.0*) -- Dropput rates to be used in case of dropout layer
- **transferTrain** (*str, default is None*) -- Type of transfer learning to be used if any

FCN()

Constructs the FCNN model as suggested in the report

basic_block(*input_tensor*, *filters_size*, *kernel_size=5*)

buildModel()

This function builds the model based on the hyperparameters and type of U-Net architecture required.

Returns: None

checkPoint()

create the list of callbacks to be send during the training. list of callbacks impleneted are:

- cyclic learning rate
- timed stopping of training (23 hours restriction)
- early stopping of the model
- model checkpoints to be saved

Returns: None

custLoss(*temp=1.0*)

Combined loss funtion formed by the weighted combination of BCE and Dice loss

Returns: combined loss function

diceLoss(*y_true*, *y_pred*)

Calculates the Dice loss.

Parameters

- **y_true** (*list(float)*) -- Ground Truth
- **y_pred** (*list(float)*) -- Prediction from the model

Returns: (float) Dice loss value

dilatedResBlock()
Constructs the baseline segmentation model with dilated bottelneck and residual connection with dilation rate of (4,4)

dilatedResBlockGroupNorm()
Constructs the baseline segmentation model with optimal bottleneck and Group normalization

dilatedResBlockInstanceNorm()
Constructs the baseline segmentation model with optimal bottleneck and instance normalization

dilatedResBlockLayerNorm()
Constructs the baseline segmentation model with optimal bottleneck and layer normalization

dilatedResBlockWeightNorm()
Constructs the baseline segmentation model with optimal bottleneck and weight normalization

dilatedResBlockWithBNandDropout()
Constructs the baseline segmentation model with optimal bottleneck and batch normalization followed by a dropout

dilatedResBlockWithDropout()
Constructs the baseline segmentation model with optimal bottleneck and Dropout layer instead of Batch normalization

dilated_convo()
Constructs the baseline segmentation model with dilated bottelneck with dilation rate given in intializer

dilated_convo2()
Constructs the baseline segmentation model with dilated bottelnecks with dilation rate of (2,2) and (4,4)

dilated_convo3()
Constructs the baseline segmentation model with dilated bottelnecks with dilation rate of (2,2), (4,4) and (8,8)

dilated_convo4()
Constructs the baseline segmentation model with dilated bottelnecks with dilation rate of (1,1), (2,2), (4,4) and (8,8)

modelCompile()
Compile the loss function and optimizer to be used Returns: None

predAccuracy(*target*, *prediction*)
Calculate the pixel-wise accuracy and F1 score (Dice coefficient)

Parameters

- **target** (*list (float)*) -- Ground Truth
- **prediction** (*list (float)*) -- Prediction from the model

Returns: (float) accuracy, (float) F1 score

prediction(*filename*, *threshold*)
Generates the pixel-wise prediction on a image

Parameters

- **filename** (*str*) -- Relative path to the image to be tested
- **threshold** (*float*) -- Threshold value for binary classification

Returns: (ndarray) predicted mask

resBlock()
Constructs the baseline segmentation model with residual bottelneck

test()
Creates the prediction mask for all the images present in test dataset.

Returns: None

threshold_Check()
Function to find the optimal threshold for binary classification.

Note: threshold in the range of [0.4, 0.75] are tested with an interval of 0.05

Returns: None

train()
Fuction to train the generated model.

Returns: (Dict) training loss and accuracy, (bool) timeFlag which indicates whether the model is stopped due to time restriction

unet()
Constructs the U-Net model as proposed by Ronneberger et al. in paper "U-Net: Convolutional Networks for Biomedical Image Segmentation"

unetPlusPlus()
Constructs the Nested U-Net model as suggested by Zhou et al. in paper "UNet++: A Nested U-Net Architecture for Medical Image Segmentation"

unet_Enze19_2()
Constructs the U-Net model as suggested by Enze Zhang et al. in paper "Automatically delineating the calving front of Jakobshavn Isbræ from multitemporal TerraSAR-X images: a deep learning approach"

2.5 plot module

Author: Maniraman Periyasamy

This module plots the train and validation losses which is used to determine the overfitting and convergence.

2.6 postprocess module

Author: Maniraman Periyasamy

This modulde delineates the segmentation results along with calculating the segmentation results of both zones and front.

postprocess.**iouAccuracy**(*target*, *prediction*)
Calculates the Intersection over Union metric for the binary class image segmentation

Parameters

- **target** (*ndarray*) -- Ground truth in the form of numpy array
- **prediction** (*ndarray*) -- Predictions in the form of numpy array

Returns: (float) IoU

postprocess.**lcc_mask**(*bin_img*)
Finds the largest connected component in a binary image

Parameters **bin_img** (*ndarray*) -- Binary image

Returns: (ndarray) mask with largest connected component

`postprocess.`**`predAccuracy`**(*target*, *prediction*)

Calculate the Pixel-wise accuracy and F1 score for binary class image segmentation.

Parameters

- **`target`** (*`ndarray`*) -- Ground truth in the form of numpy array
- **`prediction`** (*`ndarray`*) -- Predictions in the form of numpy array

Returns: (float) pixel-wise accuracy, (float) F1 Score

Python Module Index

d

m

p

Index

U