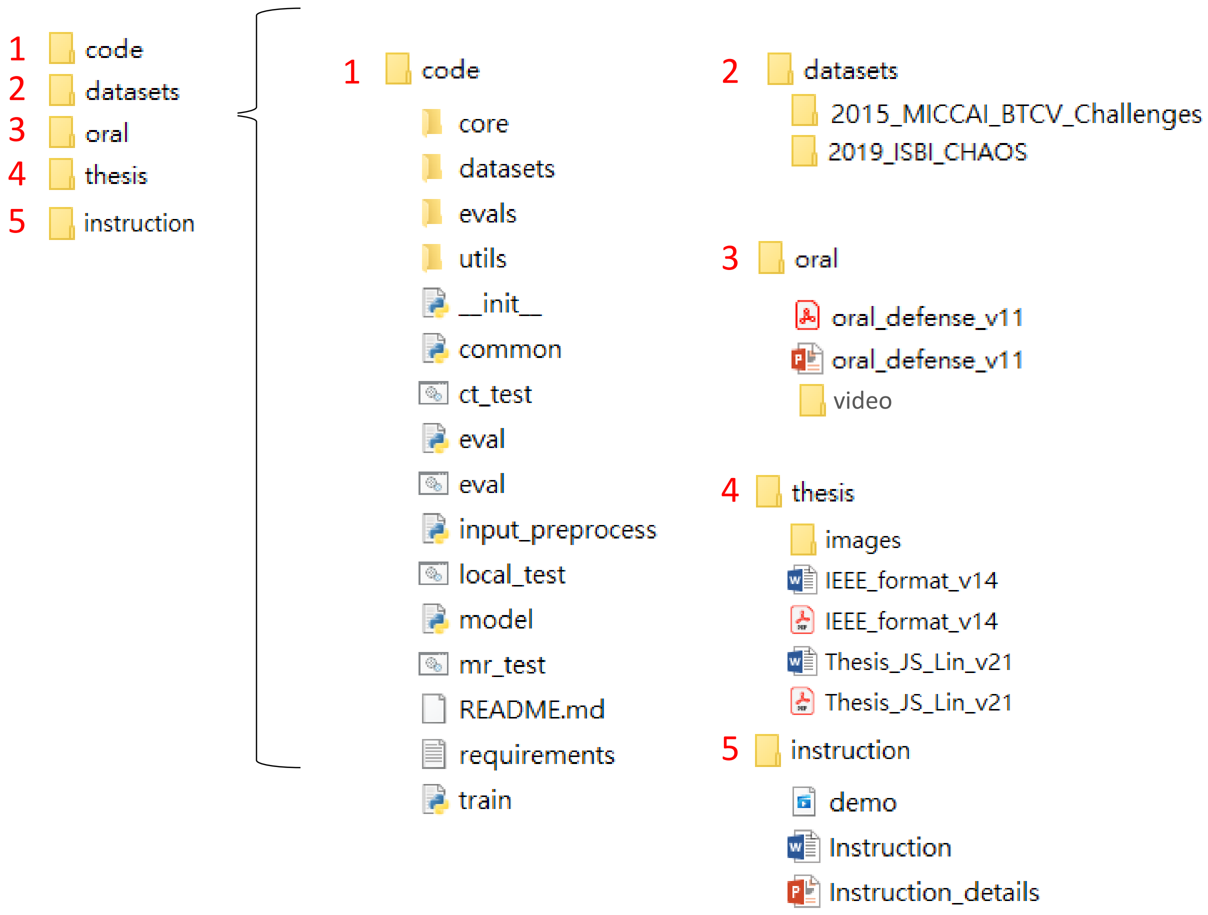
Instruction

Lin, Jing-Siang

2020/10/26

Transition items

1. Code
2. Datasets
   - 2019_ISBI_CHAOS
   - 2015_MICCAI_BTCV_Challegnges
3. Oral
   - Oral video
   - Oral slide
4. Thesis
   - Thesis (Chinese)
   - IEEE format (English)
5. Instruction
   - Instruction.docx
   - Instruction_detail.pptx
   - Demo video



Outline

1. Introduction
   - Folder structure
   - Environment building
2. Datasets
   - Dataset converting
   - Sample building
3. Model training
   - Functions
   - Tensorboard
4. Model evaluation
   - Functions
   - 3d evaluation
   - Demo video for evaluation
5. References
   - Datasets
   - Code
   - Others

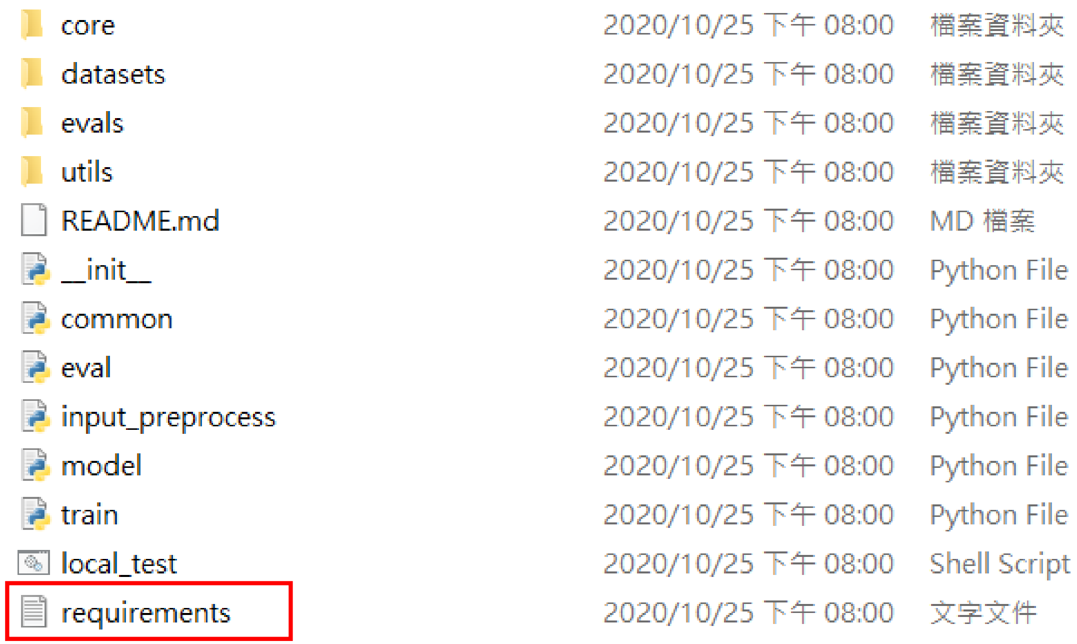
Introduction

- This instruction is try helping the user to easily execute the code of my thesis - Prior guiding based multiple organ segmentation (基於先驗知識引導之多器官切割) which focus on using the prior based on clinical knowledge to get a more robust and accurate organ segmentation (See Fig. 1). Our work report an average Dice score of 86.2% on the validation set of MICCAI2015 challenge “Multi- Atlas Labeling Beyond the Cranial Vault”.

Figure 1. The multi-organ segmentation results in axial view.

Introduction

- Prerequisites
  - Numpy=1.16.3
  - Opencv-python=3.1.0
  - Matplotlib=2.2.3
  - Nibabel=3.0.1
  - Simple ITK=1.2.4
  - Tensorflow=1.14.0
  - Scipy=1.4.1
  - Sklearn



Introduction

Folder structure

core	2020/10/25 下午 08:00	檔案資料夾
datasets	2020/10/25 下午 08:00	檔案資料夾
evals	2020/10/25 下午 08:00	檔案資料夾
utils	2020/10/25 下午 08:00	檔案資料夾
README.md	2020/10/25 下午 08:00	MD 檔案
__init__	2020/10/25 下午 08:00	Python File
common	2020/10/25 下午 08:00	Python File
eval	2020/10/25 下午 08:00	Python File
input_preprocess	2020/10/25 下午 08:00	Python File
model	2020/10/25 下午 08:00	Python File
train	2020/10/25 下午 08:00	Python File
local_test	2020/10/25 下午 08:00	Shell Script
requirements	2020/10/25 下午 08:00	文字文件

Introduction
Folder structure

- core
  - __init__
  - attentions
  - cell
  - features_extractor
  - modules
  - preprocess_utils
  - resnet_v1_beta
  - utils
  - utils2
- datasets
  - __init__
  - build_btcv_data
  - build_btcv_prior
  - build_chaos_data
  - build_chaos_prior
  - build_medical_data
  - build_prior
  - data_generator
  - dataset_infos
  - file_utils
  - build_btcv_data
  - build_chaos_data
- evals
  - eval_utils
  - evaluator
  - metrics
  - chaos_eval
    - CHAOSmetrics
    - evaluate3D
- utils
  - __init__
  - eval_utils
  - losses
  - train_utils

Introduction
Environment building

- The code is developed using gcc7.5.0, python 3.6+, cuda10.0+ on Ubuntu 16.04. NVIDIA GPUs are needed. The code is tested using 1 x NVIDIA 1080ti GPUS cards. All the experiments are tested on Tensorflow 1.14.0.

Introduction
Environment building

- Create conda environment to mange python libraries
  - To check existing conda environment
    - conda env list
  - To create conda environment
    - conda create –n ***"environment name"*** python=3.6
  - To activate conda environment
    - conda activate ***"environment name"***
  - To deactivate conda environment
    - conda deactivate

Introduction
Environment building

1. Access the code
2. Download required libraries
3. Set up path
4. Set up raw data
5. Run build_xxx_data.sh to build up tfrecord and prior
6. Start training

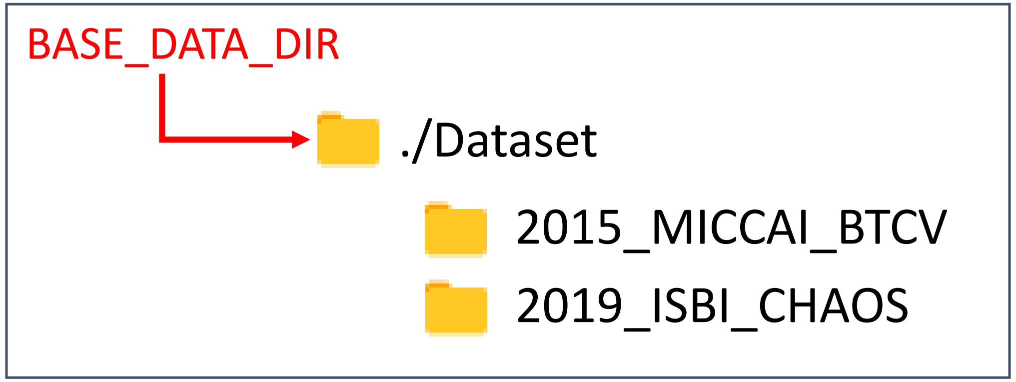
Introduction
Environment building

1. Access the code
   - unzip ***"project name"***
   - cd ***"project name"***
2. Install required libraries
   - pip install -r requirements.txt
3. Set up path
   - common.py
     - LOGGGING_PATH
     - BASE_DATA_DIR
   - build_btcv_data.sh, build_chaos_data.sh
     - WORK_DIR
4. Set up raw data
   - Upload your data according to the dataset directory structure
5. Convert raw data to tfrecord and prior
   - sh build datasets/build_btcv_data.sh
   - sh build datasets/build_chaos_data.sh
6. Start training
   - sh local_test.sh

```
# The path for saving tensorflow checkpoint and tensorboard event
LOGGING_PATH = '/home/user/DISK/data/Jing/model/Thesis/thesis_trained/'

# The path for dataset directory. Each directory should contain raw data,
# and tfrecord or prior if the converting process is run
BASE_DATA_DIR = "/home/user/DISK/data/Jing/data/"
```

```
# The directory that raw data saved
WORK_DIR="/home/user/DISK/data/Jing/data"
```

The description of path



dataset directory structure

Introduction

Environment building

- Shell script example

```
# Exit immediately if a command exits with a non-zero status.
set -e

CURRENT_DIR=$(pwd)
# The directory that raw data saved
WORK_DIR="/home/user/DISK/data/Jing/data"
DATASET="2019_ISBI_CHAOS"

# Root path for CHAOS dataset.
CHAOS_ROOT="${WORK_DIR}/${DATASET}"

export PYTHONPATH="${CHAOS_ROOT}:${PYTHONPATH}"

# Build TFRecords of the dataset.
# First, create output directory for storing TFRecords.
OUTPUT_DIR="${CHAOS_ROOT}/tfrecord/"
mkdir -p "${OUTPUT_DIR}"

BUILD_SCRIPT="${CURRENT_DIR}/build_chaos_data.py"

echo "Converting 2019 ISBI CHAOS dataset..."
python "${BUILD_SCRIPT}" \
  --data_dir="${CHAOS_ROOT}" \
  --output_dir="${OUTPUT_DIR}" \
  --seq_length=1 \
  --split_indices 0 16
```

Shell script for dataset converting

Introduction
Environment building

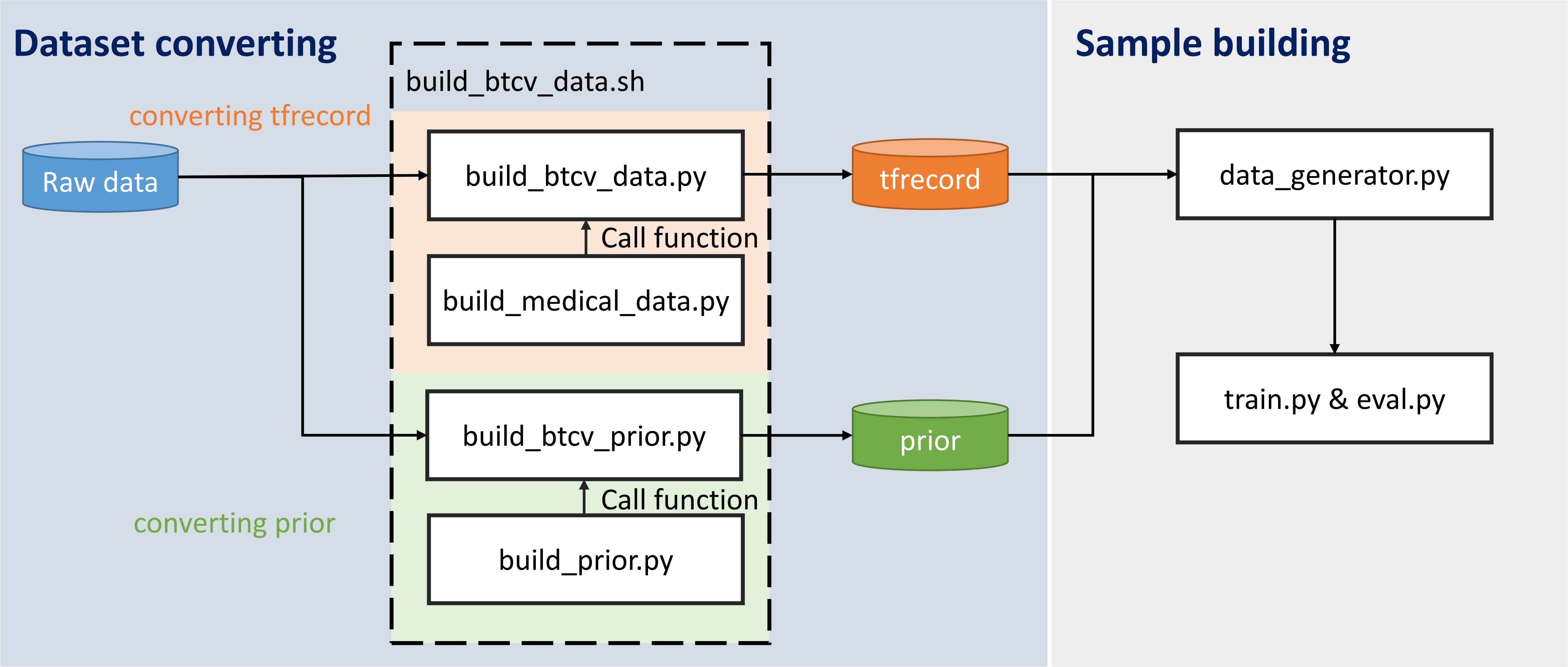
- Shell script example

```
# DATASET_NAME = ['2013_MICCAI_Abdominal']
# DATASET_NAME = ['2019_ISBI_CHAOS_MR_T1', '2019_ISBI_CHAOS_MR_T2']
# DATASET_NAME = ['2019_ISBI_CHAOS_CT']
gpu_ids=1


CUDA_VISIBLE_DEVICES=$gpu_ids python train.py \
    --dataset_name 2015_MICCAI_Abdominal \
    --batch_size=4 \
    --seq_length=3 \
    --train_split train \
    --guid_fuse mean \
    --seg_loss_name softmax_dice_loss \
    --guid_loss_name sigmoid_cross_entropy \
    --stage_pred_loss_name sigmoid_cross_entropy \
    --validation_steps=150 \
    --training_number_of_steps=200000 \
    --save_checkpoint_steps=150 \
    --guid_encoder=image_only \
    --prior_num_subject 24 \
    --fusions guid_uni guid_uni guid_uni guid_uni guid_uni \
    --weight_decay=0.001 \
    --out_node=32 \
    --guid_conv_nums=2 \
```

Shell script for training

Datasets

data_generator.py	18
build_chaos_data.sh	1
build_btcv_data.sh	1
build_chaos_data.py	9
file_utils.py	2
build_btcv_data.py	7
dataset_infos.py	2
build_btcv_prior.py	2
build_medical_data.py	2
build_prior.py	4
build_chaos_prior.py	4

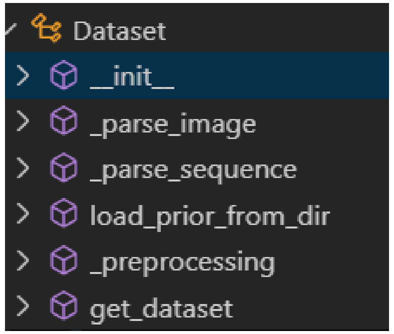


Datasets
Dataset converting

- Algorithm of converting tfrecord
    1. Load raw data (multiple 2d images or single 3d volume) from assigned path
    2. Iterate all images in single subject (patient, case)
    3. Extract interesting feature (height, width, pixel intensity,…)
    4. Write feature into tf.train.Example() or tf.train.SequenceExample() depend on it’s a single image or sequence data
    5. Finish writing process after processing all images in single subject.
    6. Finish converting or Go back to first step if still un-processing data remain

Datasets
Sample building

- Functions in class ***Dataset***
  1. Initialize ***Dataset*** object
  2. Parsing
     - Get tensor from tfrecord
  3. Pre-processing
     - Pre-processing for data
     - Load prior through **function *load_prior_from_dir***
  4. Get single sample



Class ***Dataset***

```
train_generator = data_generator.Dataset(
    dataset_name=FLAGS.dataset_name,
    split_name=FLAGS.train_split,
    guidance_type=FLAGS.guidance_type,
    batch_size=clone_batch_size,
    pre_crop_flag=FLAGS.pre_crop_flag,
    mt_label_method=FLAGS.z_label_method,
    mt_class=FLAGS.z_class,
    mt_label_type="z_label",
    crop_size=data_inforamtion.train["train_crop_size"],
    min_resize_value=FLAGS.min_resize_value,
    max_resize_value=FLAGS.max_resize_value,
    resize_factor=FLAGS.resize_factor,
    min_scale_factor=FLAGS.min_scale_factor,
    max_scale_factor=FLAGS.max_scale_factor,
    scale_factor_step_size=FLAGS.scale_factor_step_size,
    num_readers=2,
    is_training=True,
    shuffle_data=True,
    repeat_data=True,
    prior_num_slice=FLAGS.prior_num_slice,
    prior_num_subject=FLAGS.prior_num_subject,
    seq_length=FLAGS.seq_length,
    seq_type="bidirection")
```

Initialize ***Dataset*** object

```
dataset1 = train_generator.get_one_shot_iterator()
iter1 = dataset1.make_one_shot_iterator()
train_samples = iter1.get_next()
```

Get training sample

Model training

Functions

- How to train the model?
  - Set up the GPU and all the other parameters in shell script, e.g., local_test.sh.
  - Run `sh local_test.sh`

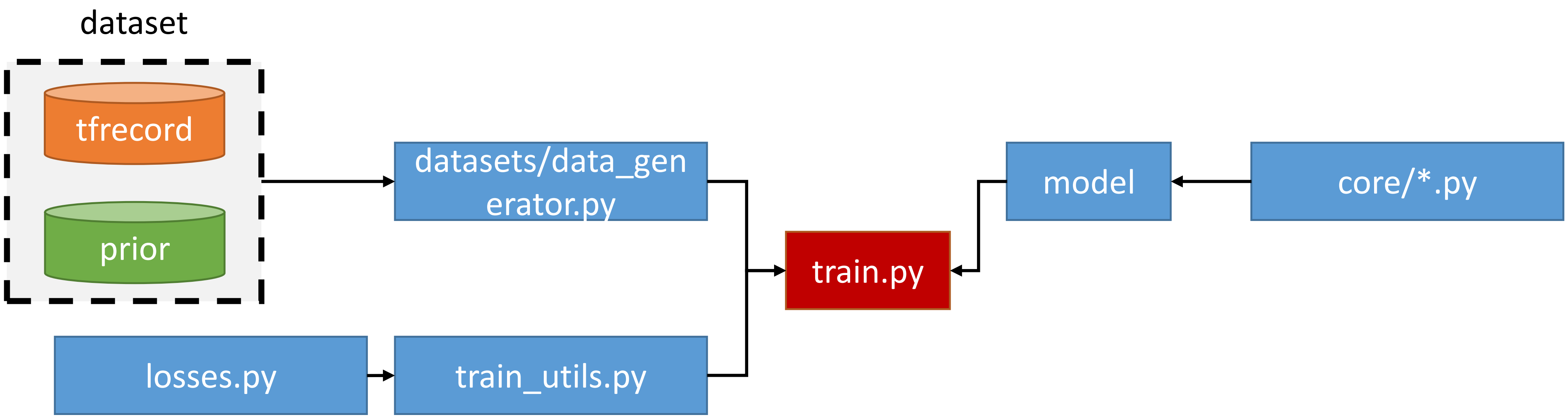
```
# DATASET_NAME = ['2013_MICCAI_Abdominal']
# DATASET_NAME = ['2019_ISBI_CHAOS_MR_T1', '2019_ISBI_CHAOS_MR_T2']
# DATASET_NAME = ['2019_ISBI_CHAOS_CT']
gpu_ids=0


# MICCAI image decay=1e-3 out_node=32 conv_num=2 dice_loss
CUDA_VISIBLE_DEVICES=$gpu_ids python train.py \
    --dataset_name 2015_MICCAI_Abdominal \
    --batch_size=16 \
    --train_split train \
    --guid_fuse mean \
    --seg_loss_name softmax_dice_loss \
    --guid_loss_name sigmoid_cross_entropy \
    --stage_pred_loss_name sigmoid_cross_entropy \
    --validation_steps=150 \
    --training_number_of_steps=180000 \
    --save_checkpoint_steps=150 \
    --guid_encoder image_only \
    --fusions guid_uni guid_uni guid_uni guid_uni guid_uni \
    --weight_decay=0.001 \
    --out_node=32 \
    --guid_conv_nums=2 \
```

Shell script example

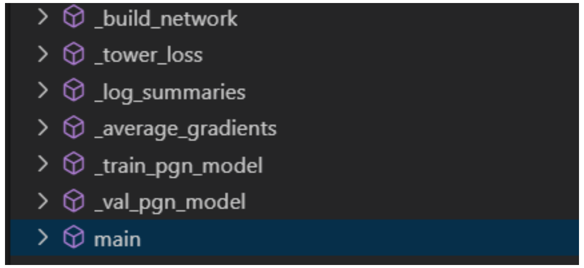
Model training

Functions

- training process

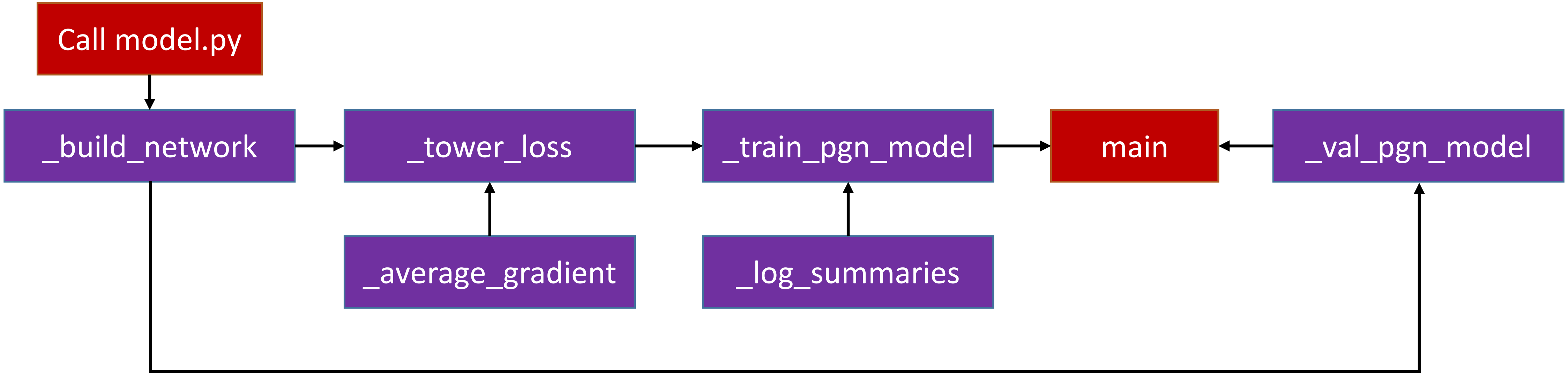


Model training
Functions



train.py

- Functions in train.py

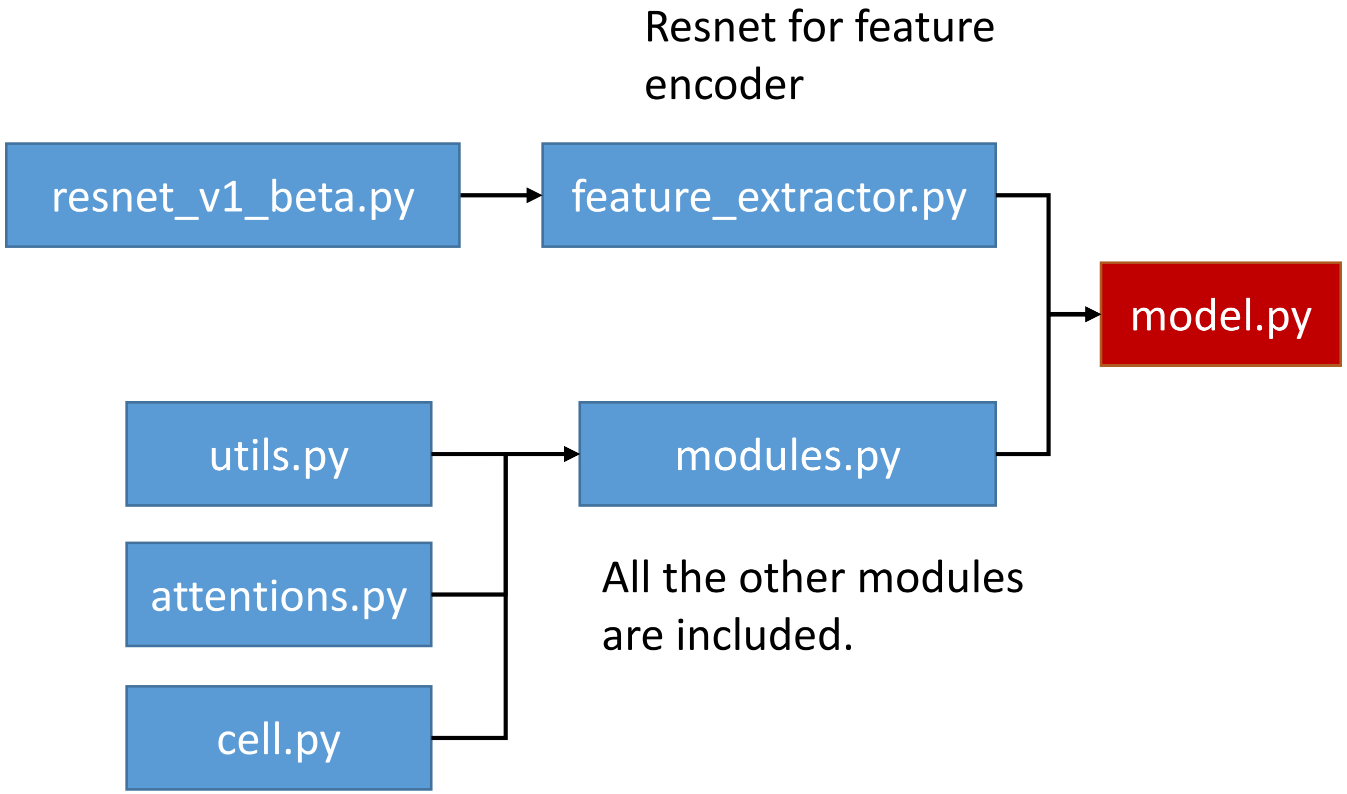


Model training

Functions

utils.py	35
attentions.py	1
modules.py	7
preprocess_utils.py	24
cell.py	4
__init__.py	0
features_extractor.py	8
resnet_v1_beta.py	20

- Model structure



Model training

Tensorboard

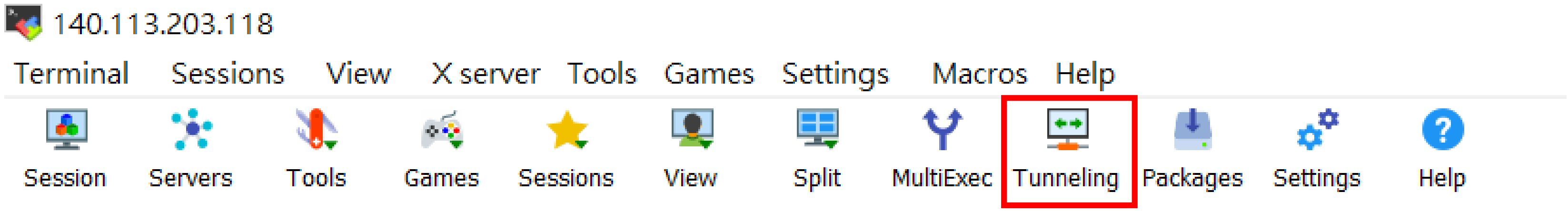
- How to use?
  - Start training
  - Run `tensorboard --logdir="tensorboard event directry"`

Name	Size (KB)	Last modified	Owner
..			
events.out.tfevents.1601613817.user	10 013 385	2020-10-03 23:21	user
logging.txt	5	2020-10-03 23:21	user
model.ckpt-180000.meta	7 059	2020-10-03 23:21	user
model.ckpt-180000.index	40	2020-10-03 23:21	user
model.ckpt-180000.data-00000-of-00002	1	2020-10-03 23:21	user
checkpoint	1	2020-10-03 23:21	user

Model training

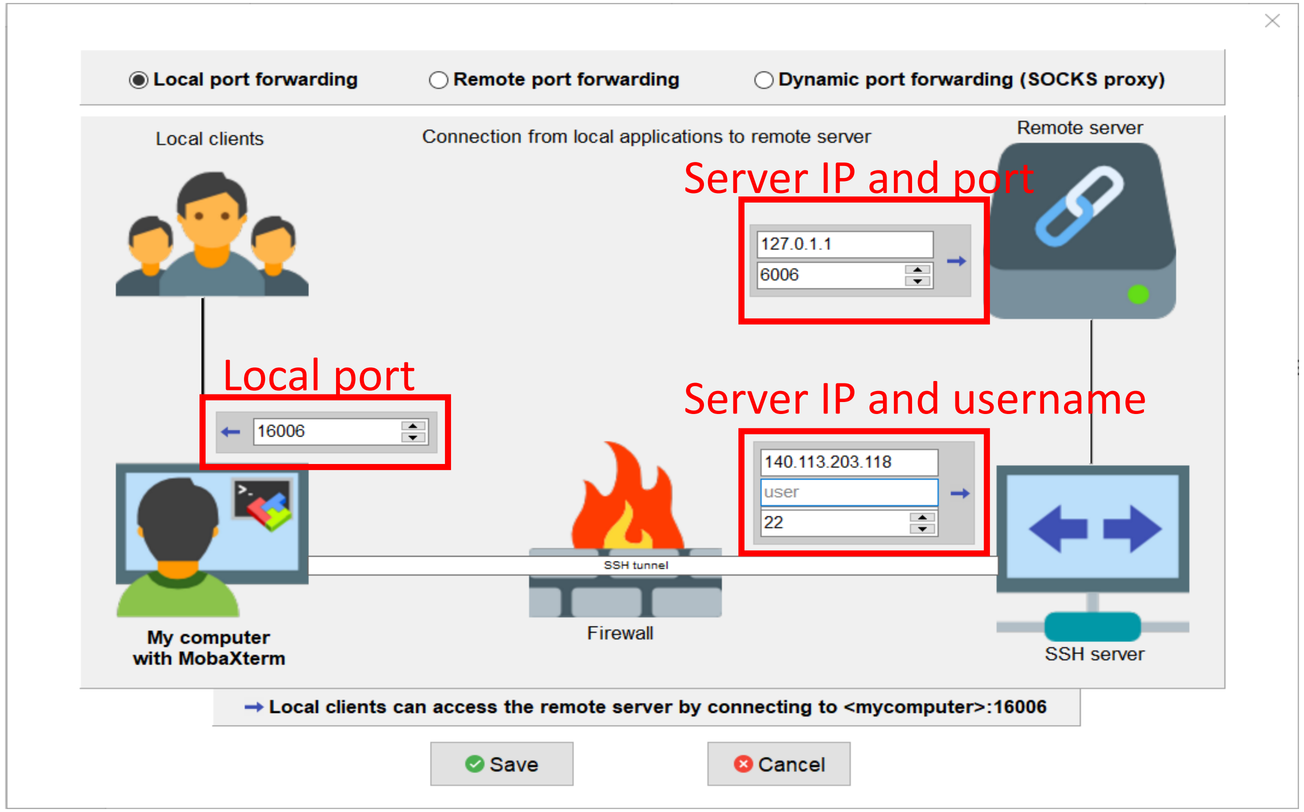
Tensorboard

- Remote control through MobaXterm
  - If you couldn't access the local server, please try to use MobaXterm and use the function ***Tunneling***



Model training
Tensorboard

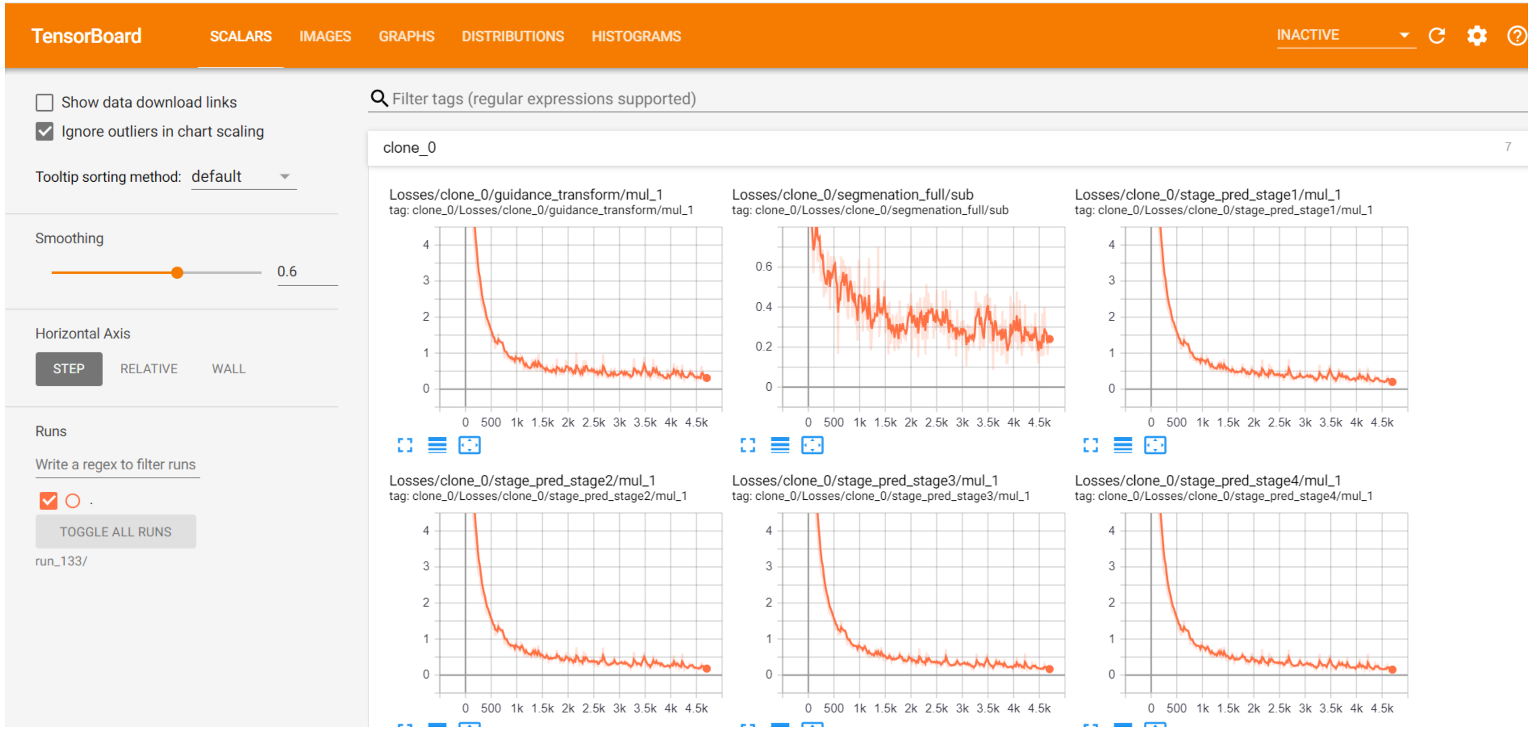
- Set up IP and port in MobaXterm tunneling panel



MobaXterm Tunnel Setting panel

Model training
Tensorboard

- After running tensorboard command, access website **http://127.0.0.1:16006/** on your local machine



Model training
Tensorboard

- How to add interesting feature?
  - Use collections mechanism in tensorflow
  - Add the interesting feature in train/_log_summaries

```
tf.add_to_collection("guidance", g1)
tf.add_to_collection("guidance", g2)
guid = tf.get_collection("guidance") # guid = [g1, g2]
```

Collections example

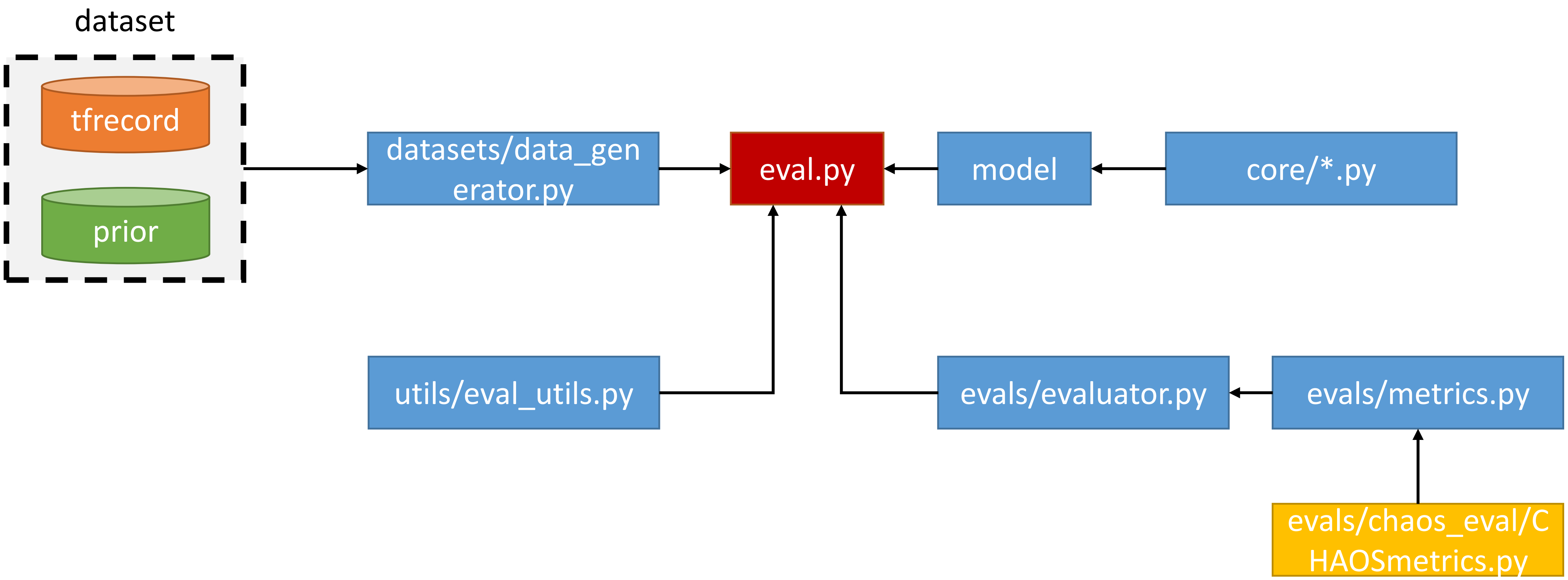
```
def _log_summaries(input_image, label, num_of_classes, output, z_pred, prior_segs,
                   layers_dict, guidance_original, **kwargs):
    """Logs the summaries for the model.
    The easiest way to add the summarirs for interesting feature is call
    tf.add_to_collections() during model building, then call
    tf.get_collections() in this function.
    Args:
```

Add the interesting feature in train/_log_summaries

Model evaluation

Functions

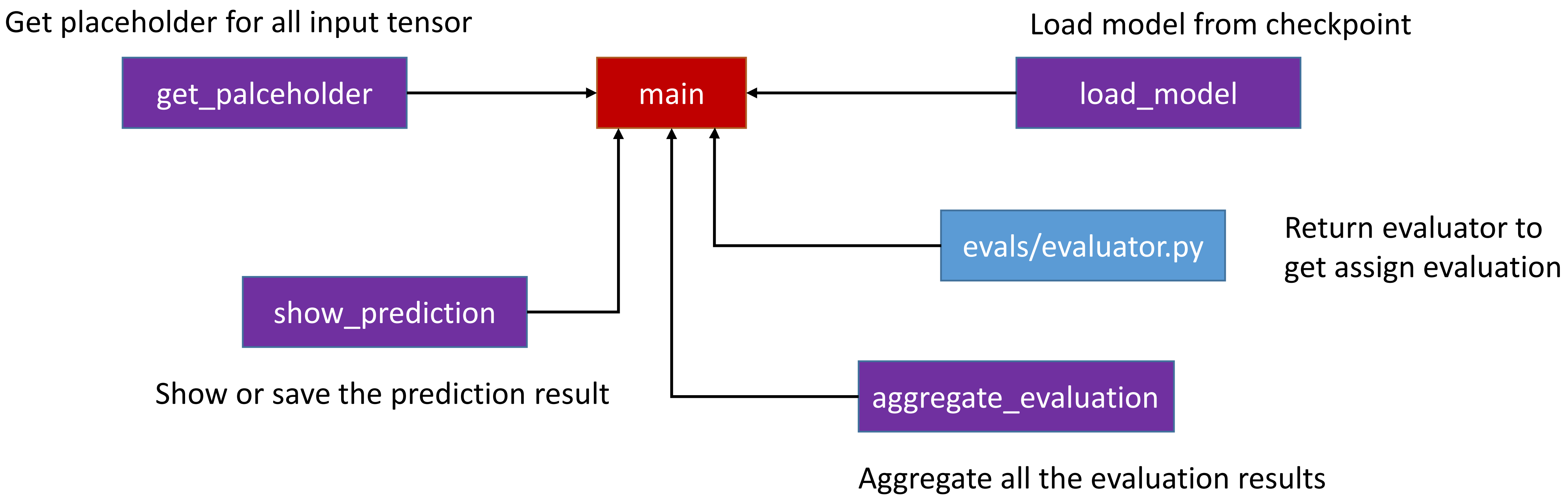
- evaluation process



Model evaluation

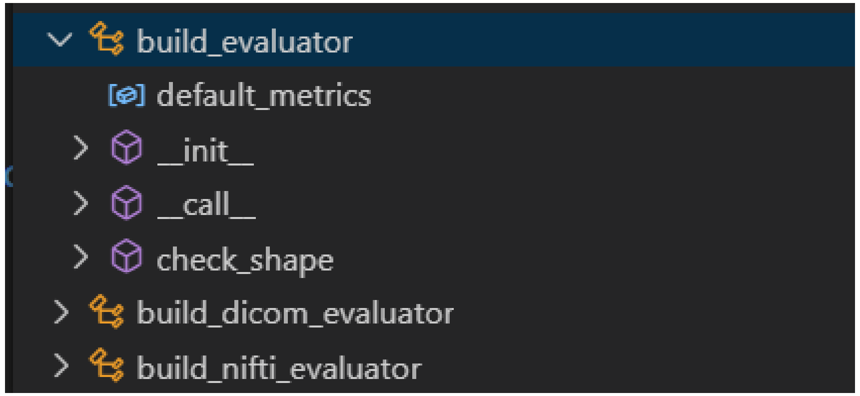
Functions

- Function structure of eval.py



Model evaluation

Functions



- ***class build_evaluator***
  - Build evaluator for nifti format (BTCV dataset)
  - __init__
    - Build a simple evaluator which aggregate all the metrics you want to test
  - __call__
    - Input segmentation, ground truth and confusion matrix and other parameters to get evaluation
  - check_shape
    - Check the shape consistency between segmentation and ground truth
  - visualize_in_3d
    - Visualize volume in 3d by pyplot scatter plot
- ***class build_dicom_evaluator***
  - Build evaluator for dicom format (CHAOS dataset)

Model evaluation
Functions

```
_ALL_METRICS = {
                "RAVD": metrics.RAVD,
                "ASSD": metrics.ASSD,
                "MSSD": metrics.MSSD,
                "DSC": metrics.DICE,
                "Precision": metrics.precision,
                "Recall": metrics.recall}
```

```
def __init__(self, metrics=None):
    self.metrics = []
    if metrics is not None:
        for m in metrics:
            self.metrics.append(m)
    else:
        for m in self.default_metrics:
            self.metrics.append(m)

    self.transform_flag = False
    for _3d_m in ["DSC", "RAVD", "MSSD", "ASSD"]:
        if _3d_m in self.metrics:
            self.transform_flag = True
            break
```

__init__ : Set up all the evaluation metrics

```
def __call__(self, ref, test, **metrics_kwargs):
    results = {}
    self.check_shape(ref, test)
    for m in self.metrics:
        results[m] = _ALL_METRICS[m](ref, test, **metrics_kwargs)
    return results

def check_shape(self, ref, test):
    print(ref.shape, test.shape)
    assert ref.shape == test.shape
    # These metrics require 3D data
    for _3d_m in ["DSC", "RAVD", "MSSD", "ASSD"]:
        if _3d_m in self.metrics:
            if ref.ndim != 3 or test.ndim != 3:
                raise ValueError("Incorrect shape for %s, this metrics require 3d data" %_3d_m)
            break
```

__call__ : Input segmentation and ground truth to get evaluation

Model evaluation

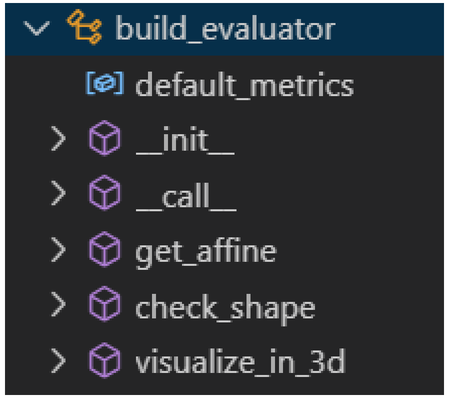
3d visualization

- Call ***visualize_in_3d*** to visualize result in 3d
- Build up evaluator and call visualize_in_3d

```
evaluate = evaluator.build_evaluator()
evaluate.visualize_in_3d(ref, test, raw_data_path)
```

- Reference and testing result will aggregate in dictionary form automatically

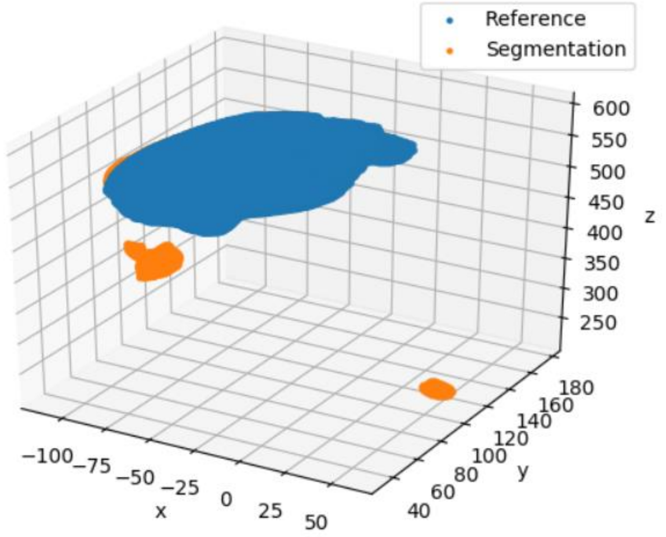
```
def visualize_in_3d(self, ref, test, raw_data_path):
    affine = self.get_affine(raw_data_path)
    eval_utils.show_3d({"Reference": ref, "Segmentation": test}, affine)
```



All functions in evaluator

Model evaluation
3d visualization

```python
def show_3d(volume_dict, affine):
    """
    Scatter plot for 3d visualization
    Args:
        volume_dict: Input dictionary contains all volumes that need to be visualized.
                     The key of volume_dict is the label of the plot, and the value of
                     volume_dict is a 3d NumPy array.
        affine: Affine transform parameters extracted from raw data which helps to
                transform volume to real-world coordinate
    """
    struct = ndimage.generate_binary_structure(3, 1)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    for label, v in volume_dict.items():
        # Access border points to get better running performance
        border=v ^ ndimage.binary_erosion(v, structure=struct, border_value=1)
        border_voxels=np.array(np.where(border))
        border_voxels_real=metrics.transformToRealCoordinates(border_voxels,affine)
        Sx, Sy, Sz = [], [], []
        for s in  border_voxels_real:
            Sx.append(s[0])
            Sy.append(s[1])
            Sz.append(s[2])
        ax.scatter(Sx, Sy, Sz, marker='.', label=label)
    ax.legend()
    plt.show()
```



3d visualization example

Model evaluation

Demo video for evaluation

```
#!/bin/bash
# Copyright 2018 The TensorFlow Authors All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================
#
# This script is used to run local test on PASCAL VOC 2012. Users could also
# modify from this script for their use case.
#
# Usage:
#   # From the tensorflow/models/research/deeplab directory.
#   sh ./local_test.sh
#
#
# DATASET_NAME = ['2015_MICCAI_Abdominal']
# DATASET_NAME = ['2019_ISBI_CHAOS_MR_T1', '2019_ISBI_CHAOS_MR_T2']
# DATASET_NAME = ['2019_ISBI_CHAOS_CT']

gpu_ids=1

CUDA_VISIBLE_DEVICES=$gpu_ids python eval2.py \
    --fusions  guid_uni guid_uni guid_uni guid_uni guid_uni \
    --dataset_name 2015_MICCAI_Abdominal \
    --checkpoint_dir=/home/user/DISK/data/Jing/model/Thesis/thesis_trained/run_131/model.ckpt-best \
    --eval_split val \
    --guid_encoder image_only \
    --store_all_imgs False \
    --guid_fuse mean_wo_back \
    --out_node 32 \
```

References

- **Datasets**

1. MICCAI2015 challenge: **https://www.synapse.org/#!Synapse:syn3193805/wiki/217789**

2. CHAOS challenge: **https://chaos.grand-challenge.org/Combined_Healthy_Abdominal_Organ_Segmentation/**

References

- **Code**

1. Deeplab: **https://github.com/tensorflow/models/tree/master/research/deeplab**
2. nnUnet: **https://github.com/MIC-DKFZ/nnUNet/tree/4d8aa747b288e4297236eb385ea325256634372c**
3. CHAOS_evaluation: **https://github.com/emrekavur/CHAOS-evaluation**

References

- **Others**

1. P. Hu, G. Wang, X. Kong, J. Kuen, and Y.-P. Tan. Motion-guided cascaded refinement network for video object segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1400–1409, 2018.
2. Demo video: please check demo.mp4 under LinJingSiang/instruction