STMFANet document

This documentation is dedicated to the repository STMFANet developed by Bei Jin.

You can have a direct access to the original repository by the link: https://github.com/Bei-Jin/STMFANet/

Also, the paper is accessible by the link: https://arxiv.org/abs/2002.09905

You can also contact to the first author by his email address: jinbeibei@ict.ac.cn

The code has no original document and I spent a lot of time to debug the code. So, if you need some information that is not available in this file, I encourage you to contact with the first author by the above email address. He usually answers properly. The paper also is written in a proper way and is well understandable.

Regards,
Alireza Parchami
CAV-lab, University of Surrey, England

Read file

The code needs to access to the dataset and you should write the address of each sample in the database in the specified file. The address of each sample should be written in the text file `./data/train_data_list.txt` in the directory of the code. Please note that there are three strings in each line. The first string is the location of sample folder, the second string is the first frame that we would like to be considered by code (which I always leave it as 0 meaning that I would like to consider the frames from the very first frame) and the third string is the last frame number that we would like to be considered by code (which I always leave it as the last frame number, for example 39).

Feel free to change the second and third strings if you have reasonable reasons for considering a specific range of frames.

`Ex:`

`D:\a2d2\camera_lidar\20180810_150607\cam_front_center 0 39`

I prepared a piece of code that generate train_data_list.txt easily as you wish. To do this:

1- open `list_of_faulty_samples.py` in the folder `./data/`

2- change the variables `city` and `parent_path` as you wish

3- run the code and it will give you a complete list from all faulty samples

4- Copy it and paste it in a file named `augmented_samples.txt`

5- now open the `data_list_creator.py` in the same directory.

6- change the paths in line 2 and 15 as it is located on your server

7- run the code and you will have the `train_data_list.txt`

Note that other python files in this folder are related to the main STMFANet code. Do not change them.

Train options arguments

At the beginning of the code, it is required to specify some options (parameters). You can change the default options in the python file `./options/train_options.py`

The arguments are explained enough in each line, but some of them need to tuned by some trial and error in training such as `batch_size, lr, nepoch, alpha, beta1.`

Note that `–adversarial` activates the use of adversarial loss function. If it is `False,` only `Image Loss Domain` is considered during the training. But if is `True,` a combination of `Image Loss Domain` and `Adversarial Loss` is considered during training stage. (Both Image Loss Domain and Adversarial Loss are explained in the paper)

Note that the code is using defaults options by making the line 108 as comment. If you would like to set options for training, you can remove the comment mark from the beginning of the line.

Also, if you would like to run the code in Jupyter, you can uncomment the lines 63 to 104. It is using easydict to define default options in Python Jupyter environment.

The code uses CUDA GPU for training as default. However, If your system does NOT have CUDA GPU, you can make it work on CPU by uncommenting the line 108:
`self.opt.gpu_ids = []`

Important python files

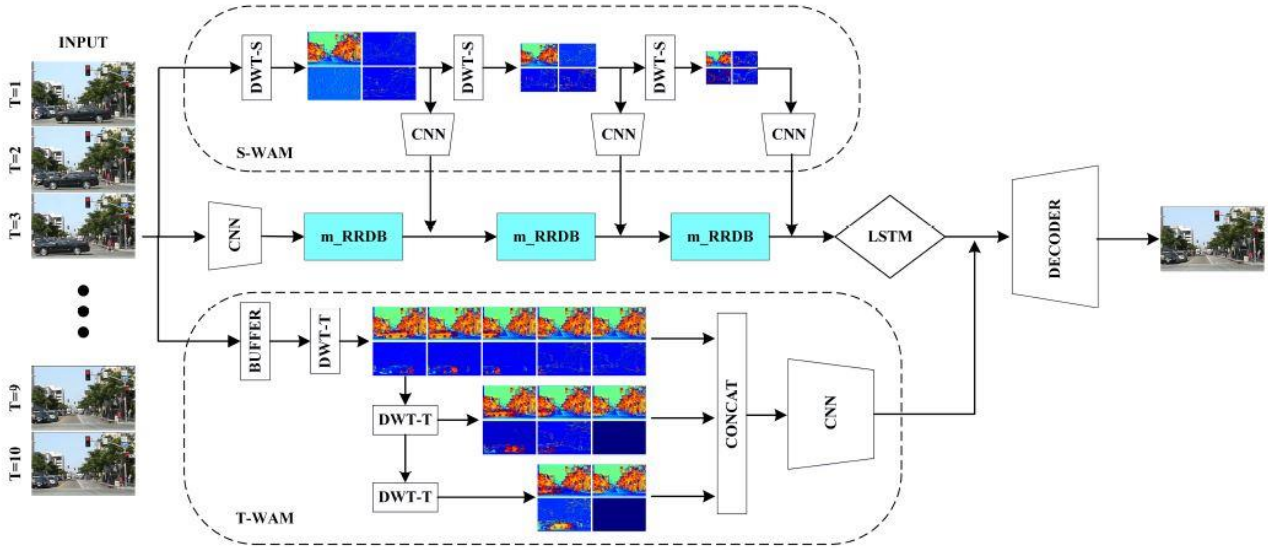
The rest of the important codes lie in the folder `./wavenet_models/`

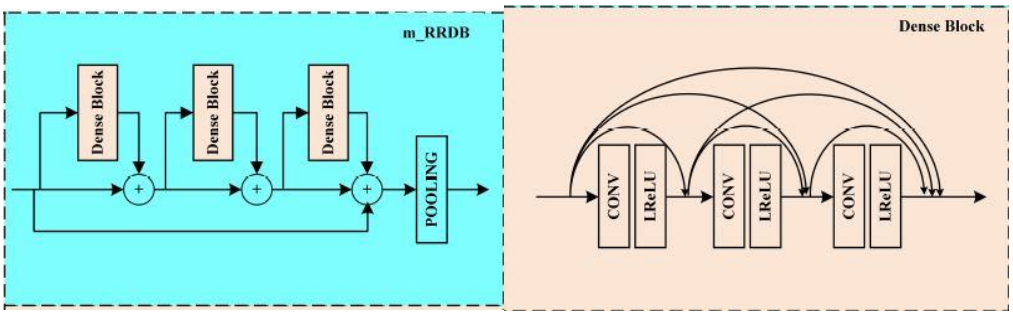
`Create_model.py` create a new instance from STMFModel class.

`Base_model.py` contains BaseModel class that has some features and functions and works as a parent class for STMFModel()

STMF_network.py contains the crucial sections of the algorithm such as Generator, Encoder, Bottleneck, Dwtconv, Decoder, Discriminator, etc. This file is supposed to have some bugs and might need to be modified! I have fixed some of its parts, but I could not test it completely. So be watchful with this file!

STMF_model.py contains the network structure and is calling the proper functions of STMF_network in order.





The photos above are the scheme of the network. The m_RRDB shows the main path of the network that each of them contains Dense Block. Dense Block is a contains 3 Conv-Relu blocks with shortcut connections as residual connections.

A potential Bug

The code had been trained on KTH dataset, which includes gray-level images. Also it seems that the code has been developed in a flexible way and it is using `-c_dim` as a argument that shows the number of channels (3 in color image and 1 in gray-level), some variables are hard-coded in the coded for gray-level images.

In the section *STMF_network.py*, *class Dwtconv*, *__init__ function*, there are three important variables named `nIn_conv1, nIn_conv2, nIn_conv3`. They were all assigned as 4 since the output of Wavelet transform for a gray-level image is 4 matrices. However, as we have 3 input channels for color image, this would be 12. I changed it manually but it is supposed to be tested. Dwtconv class is being used repetitively in the code, so it would be really important for the model!