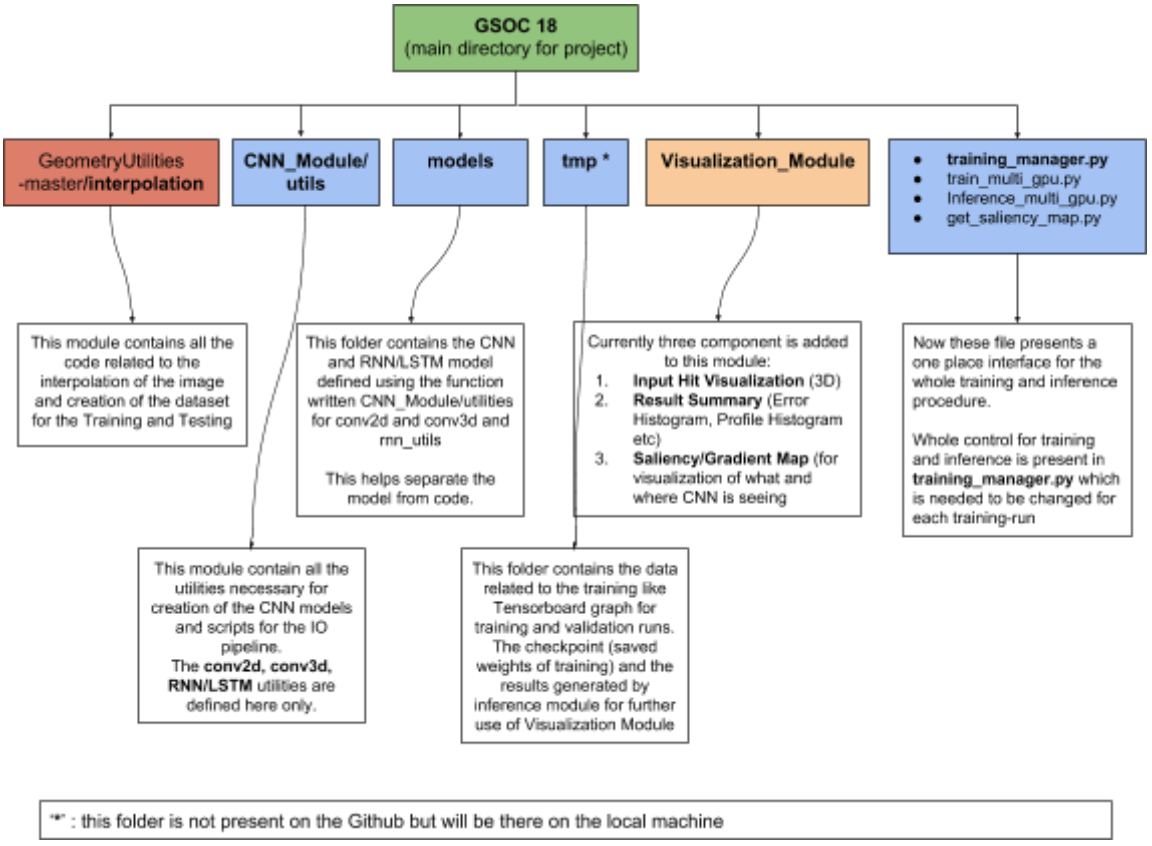
File Organization

The File and Folder are organised in the following pattern color coded with the relevance with each other or their interdependence on other.This diagram will provide a global view of the structuring of the project. (please watch in the enlarged view double clicking it)



Comments about the main files in these folders:

---

1. **GeometryUtilities-master/interpolation:** This folder contain the files necessary for all the data transformation utilities. The main two files for finding the interpolation coefficient (from hex cell to square grid) and generation of dataset from event file are:
    a. **main.py:** this is the main interface from where we will perform both the task of finding the interpolation coefficient and creation of dataset. No further changes has to be incorporated here except the multiprocessing of the event file interpolation.
    b. **hexCell_to_squareCell_interpolation.py:** This script provides all the functionality for completion of task by the main.py script. The main point of changes will be made in the dataset creation function of this script when we want to change the target of CNN training.

2. **CNN_Module/utils:** This module contains the all the utilities function and layer definition for 2D/3D convolution and RNN/LSTM layers for creation of Learning Models for training. Also it also provides all the training data IO pipeline for training using tf.data api . The main files in this module are:
    a. **conv2d_utils.py**: This script provides all the basic convolution layer like convolution2d integrated with the learning rate decay,batch normalization,dropout and optional relu rectification, max pooling layers. Apart from this it also provides the layers which are recent in CNN like residual layers, inception layers.
    b. **conv3d_utils.py:** This script contain the 3D counterpart of all the layers defined in the above script.
    c. **rnn_utils.py**: This script provides the layer RNN and LSTM layer for easy to use interface for creation of model. Currently they only support the vector sequence as input and gives output a vector sequence. The support for the convolution connection between the LSTM/RNN layers will be added later.
    d. io_pipeline.py: This script provides all the functionality for the IO of the dataset for training the model in both CPU and GPU.

    All the function developed above is very generic and could be used on both CPU and GPU. The convolution layers are currently in NHWC format (which are more suited to CPU but it doesn't have much significance right now.)

---

3. **models:** This folder will be the main place where we will design all our training model for training the CNN/RNN for prediction. This will separate the model from rest of the supporting code.

---

4. **tmp/hgcal/run_number:** This folder will not be available on GitHub but it will be locally available after running the training. This folder (under the run_number of training) will contain the run statistics, tensorboard visualization of training, results of prediction after inference and the visualization data. These will be the main component of this folder
    a. **train**: This folder contains the tensorboard visualization of the training progress on the training data.
    b. **valid:** This folder contains the tensorboard visualization of the training progress on the validation/test data.
    c. **checkpoints**: This folder will contain the saved parameter/weights of the model for use later or making prediction/inference later.
    d. **timeline:** This folder will contain the chrome trace of the timing of each ops of the tensorflow graph. This will be useful to see the time consumption part of the model and further optimization of machinery.

e. **results:** This folder contains the prediction results from the training and test dataset. Also it will store the saliency map/gradient map for learning visualizations.

---

5. Visualization Module: Currently the Visualization module supports three visualization feature.
    a. **Input Visualization:** This will visualize the input 3D energy-hits on the interpolated square grid using a animation made in plotly.This facility is provided by the **input_visualization.py** script.
    b. **Prediction Visualization:** This will use the results saved in the tmp/hgcal/run_number folder to make the prediction histogram to see the statistics of the training. This is provided by **prediction_visualization.py** script.
    c. **Saliency/Gradient Map:** This feature shows to which area in the input dataset the Training Model is sensitive to for making prediction. This is provided by the **saliency_map_visualization.py** script.

---

6. Other files in **GSOC18 directory**: This folder contain the script which are the main control point for all the training, inference and visualization process using the script developed in CNN_module, Interpolation Module, Visualization Module. The main files are :
    a. **training_manager.py** : This is the main point of control of all the training, inference and the visualization process. This is one place solution for all the work to be done for training. (other than the model definition).
    b. train_multi_gpu.py: This script is the controller for training in both CPU and GPU environment and is system independent. For changing the log frequency, checkpoint and timeline logging frequency, validation frequency while training we have to make changes here.
    c. **infer_multi_gpu.py:** This script control the making of inference on the training and validation/testing data and saving the results for further visualization.
    d. **get_saliency_map.py**: This script is used for the creation of the saliency map which will be used for further visualization. It also saves the result in tmp/hgcal/run_number/results folder.

These are the main uses and the gist of all the files and folder which have major significance for running the training and other dependent activities like interpolation, visualization and inference.