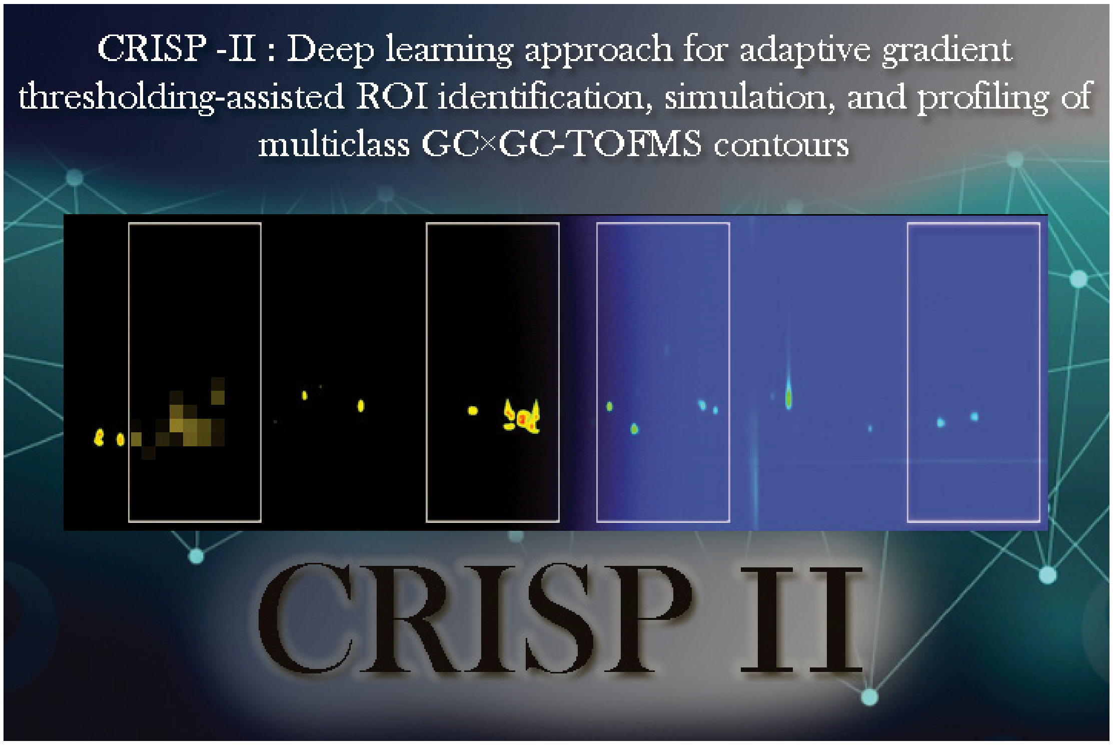
CRISP II: Deep learning approach for adaptive gradient Thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-TOFMS contours

(The manuscript is under review and few changes in software may appear during the review process)


CRISP -II : Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-TOFMS contours
CRISP II

CRISP II: Deep learning approach for adaptive gradient Thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-TOFMS contours

1. General information

Two-dimensional gas chromatography time-of-flight mass spectrometry (GC×GC-TOFMS) is a powerful tool for metabolomics analysis, but using its contour images for metabolite profiling is challenging due to limited metabolite identification libraries and advanced bioinformatics tools. To expedite GC×GC-TOFMS data analysis, we developed CRISPII, a deep learning method that assists in adaptive gradient thresholding-based identification, simulation, and profiling of multiclass separation in GC×GC-TOFMS metabolomics data. This study highlights significant advancements over our previous CRISP software, which could only classify two classes of GC×GC-TOFMS metabolomic data. CRISP enhances the use of aggregate feature representative images and introduces multiclass-compatible gradient thresholding-assisted ROI stacking for profiling GC×GC-TOFMS contours. The CRISP pipeline aids in semi-automatic ROI identification for feature enrichment and classification of GC×GC-TOFMS contour images. Additionally, it includes contour synthesis *via* a generative adversarial network for data augmentation with sparse samples. This manual assists users to utilize myriad of options available in CRISP to custom datasets or specific operations.

2. SOFTWARE ARCHITECTURE AND OPERATION GUIDELINES

The CRISP is designed for GC×GC-TOFMS and consists of three major components:

A. ROI & Deep Stacking
   - **i.** Manual GC×GC-TOFMS contour image extraction for pre-processing
   - **ii.** Single and multiclass AFRC image construction
   - **iii.** Single and multiclass deesptacking dataset construction

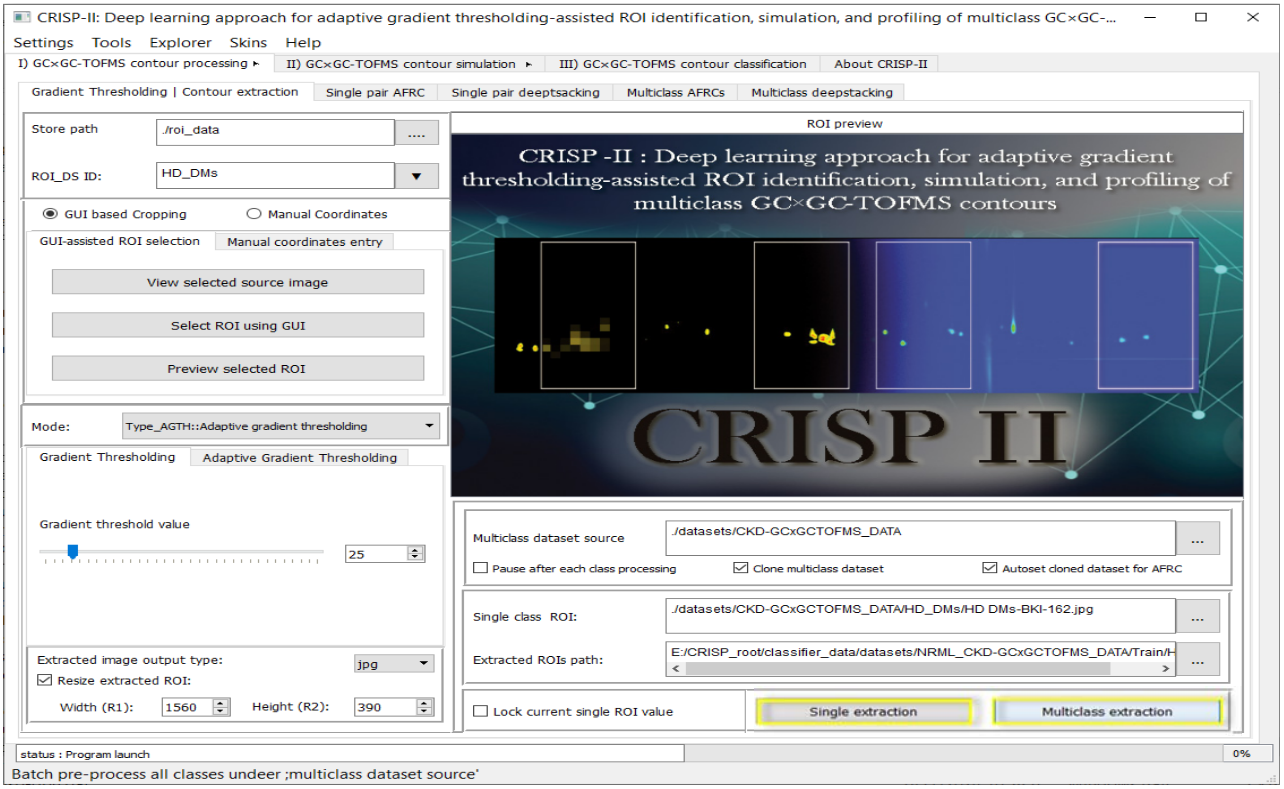
B. GAN-based GC×GC-TOFMS contour image training & synthesis

C. Multiclass GC×GC-TOFMS contour training & inference

Each module can be used independently or in combination for the opted outcome. However, the method can be applied for any other contour images of similar 2D datatype. Note: All steps A-D should be done for the same mode during inference of unknown sample.

A. ROI & Deep Stacking

i. Manual GC×GC-TOFMS contour image extraction for pre-processing


CRISP-II: Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-...
Settings
Tools
Explorer
Skins
Help
I) GC×GC-TOFMS contour processing ▸
II) GC×GC-TOFMS contour simulation ▸
III) GC×GC-TOFMS contour classification
About CRISP-II
Gradient Thresholding | Contour extraction
Single pair AFRC
Single pair deeptsacking
Multiclass AFRCs
Multiclass deepstacking
GUI-assisted ROI selection
Manual coordinates entry
View selected source image
Select ROI using GUI
Preview selected ROI
Gradient Thresholding
Adaptive Gradient Thresholding
ROI preview
CRISP -II : Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-TOFMS contours
CRISP II
Single extraction
Multiclass extraction

Store path: ./roi_data

ROI_DS ID: HD_DMs

● GUI based Cropping ○ Manual Coordinates

Mode: Type_AGTH::Adaptive gradient thresholding

Gradient threshold value: 25

Extracted image output type: jpg

☑ Resize extracted ROI:

Width (R1): 1560 Height (R2): 390

Multiclass dataset source: ./datasets/CKD-GCxGCTOFMS_DATA

□ Pause after each class processing ☑ Clone multiclass dataset ☑ Autoset cloned dataset for AFRC

Single class ROI: ./datasets/CKD-GCxGCTOFMS_DATA/HD_DMs/HD DMs-BKI-162.jpg

Extracted ROIs path: E:/CRISP_root/classifier_data/datasets/NRML_CKD-GCxGCTOFMS_DATA/Train/H

□ Lock current single ROI value

status : Program launch

0%

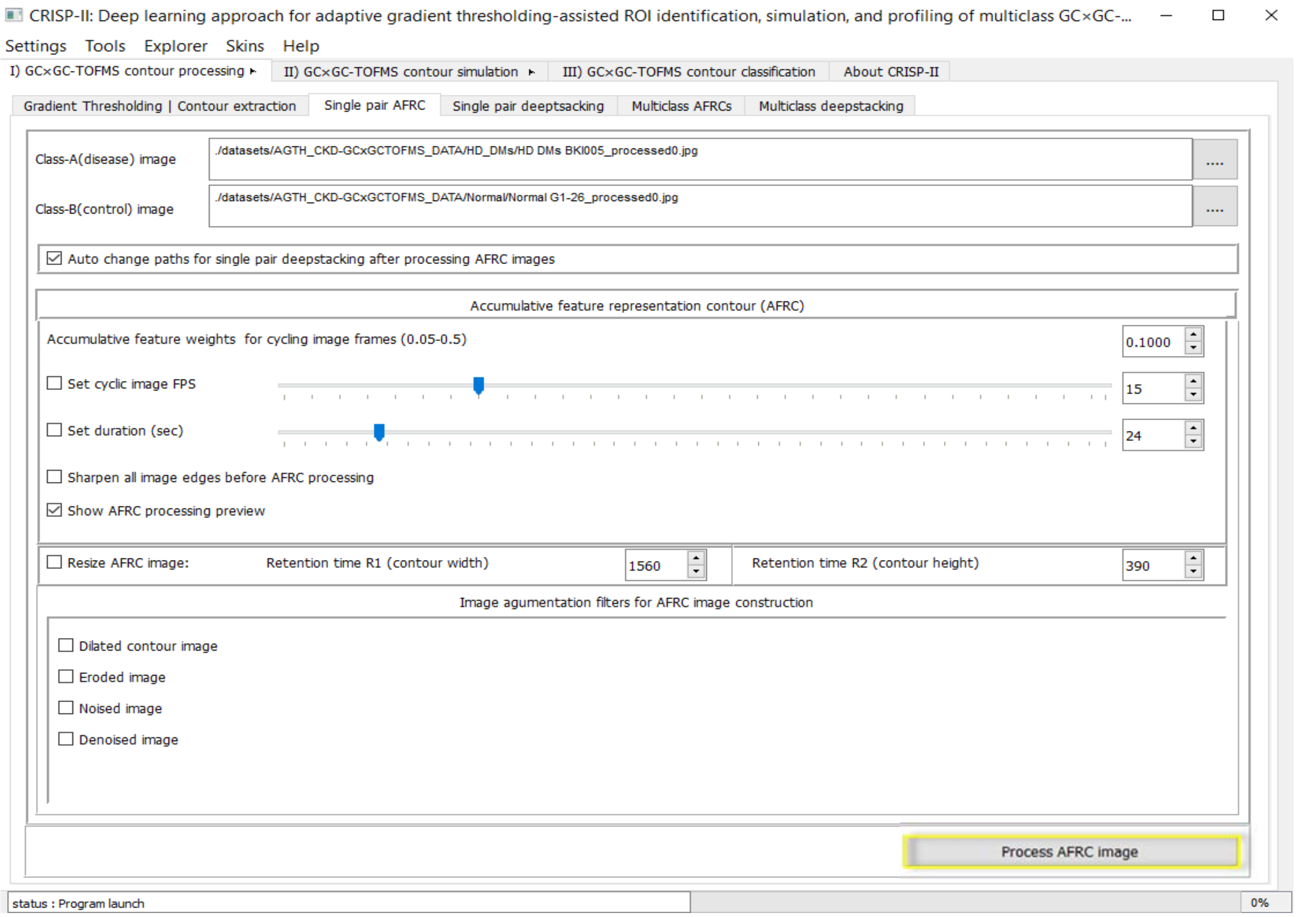
Batch pre-process all classes undeer ;multiclass dataset source'

This section allows users to manually select a single ROI either graphically or based on input coordinates. It enables cropping of the region to get a single desired ROI (or image frame in this context) with multiple options such as resizing, applying image filters, and choosing the output file type. The extraction of initial contour data can be done in Normal RGB color mode (no change in color), adaptive gradient thresholding (AGTH) or fixed gradient thresholding (FXTH) mode depending upon user choice. Note that once the contour data is extracted in a certain mode, the same mode setting should be used for all contour groups and future inferencing. Selection of an ROI for only a single image per group folder is required, and all images in the folder will automatically have the same ROIs extracted in batch operation.

General Protocol for manual ROI selection tab (process is similar for batch operation):

- Select single images from each group.
- Select a single ROI or the entire image from the contour.
- Extract ROIs (a new folder will be created with the extracted ROIs for all contours) by pressing “**Single extraction**” or “**Multiclass extraction**" button respectively under “Gradient Thresholding | Contour extraction Tab”.
- Additional mode filters like resizing and denoising can be applied as described earlier.

ii. Single and multiclass AFRC image construction


CRISP-II: Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-...
Settings Tools Explorer Skins Help
I) GC×GC-TOFMS contour processing ▸
II) GC×GC-TOFMS contour simulation ▸
III) GC×GC-TOFMS contour classification
About CRISP-II
Gradient Thresholding | Contour extraction
Single pair AFRC
Single pair deeptsacking
Multiclass AFRCs
Multiclass deepstacking
....
Process AFRC image

Class-A(disease) image: ./datasets/AGTH_CKD-GCxGCTOFMS_DATA/HD_DMs/HD DMs BKI005_processed0.jpg

Class-B(control) image: ./datasets/AGTH_CKD-GCxGCTOFMS_DATA/Normal/Normal G1-26_processed0.jpg

☑ Auto change paths for single pair deepstacking after processing AFRC images

Accumulative feature representation contour (AFRC)

Accumulative feature weights for cycling image frames (0.05-0.5): 0.1000

☐ Set cyclic image FPS: 15

☐ Set duration (sec): 24

☐ Sharpen all image edges before AFRC processing

☑ Show AFRC processing preview

☐ Resize AFRC image: Retention time R1 (contour width): 1560 Retention time R2 (contour height): 390

Image agumentation filters for AFRC image construction

☐ Dilated contour image

☐ Eroded image

☐ Noised image

☐ Denoised image

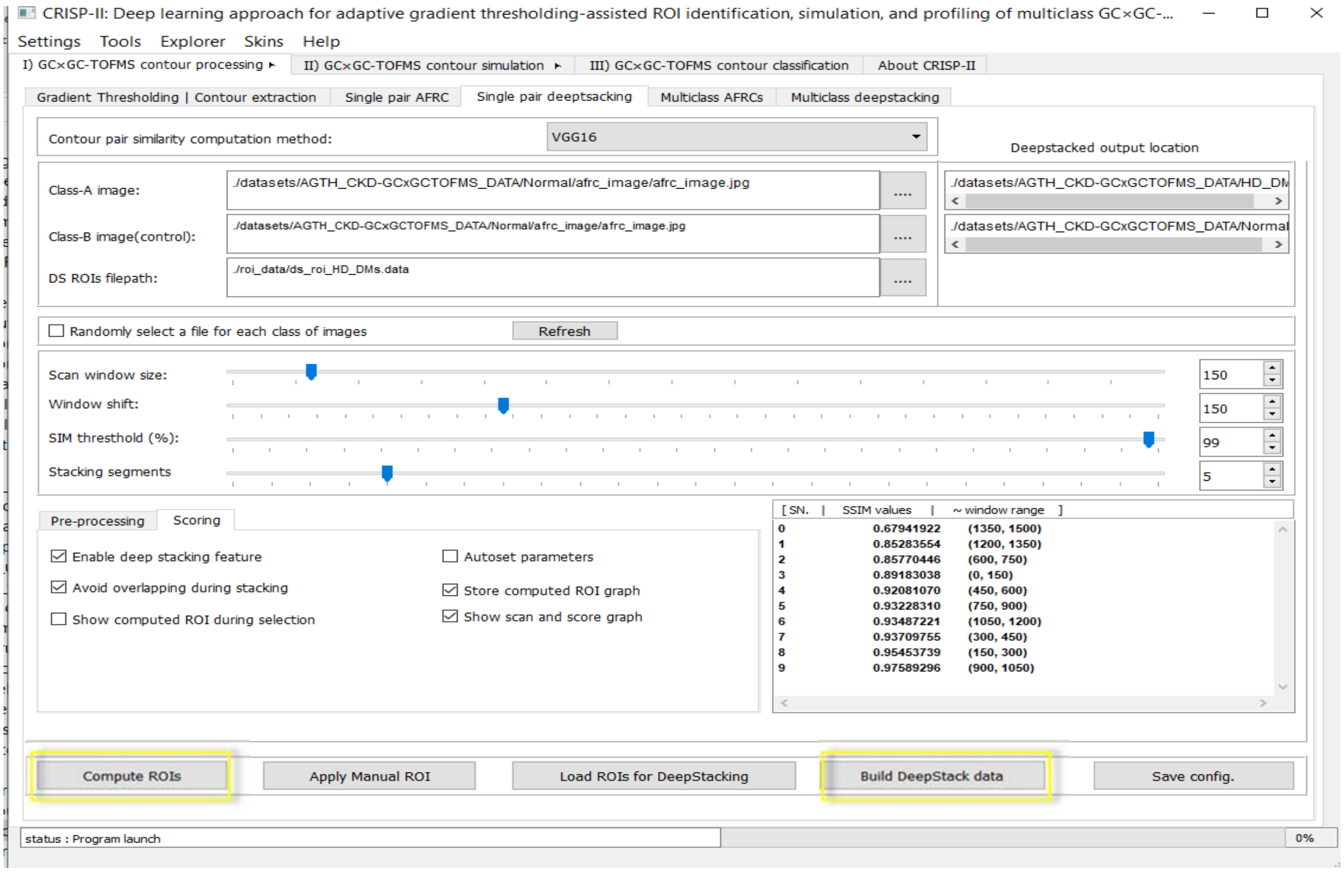
status : Program launch 0%

This innovative procedure generalizes groups of contour images that belong to a particular class (or group) through weighted aggregate feature extraction. It allows dominant features to be prominently represented in the final single representative image, while rare or outlier features are minimized algorithmically. This results in an AFRC image that represents the common dominant features present in the group, reducing manual selection bias. Parameters such as FPS, weight accumulation factor, and cyclic image duration control the number of features represented in the final AFRC image. Several image augmentation features (e.g., blur, noise, erosion, dilation) can be applied during AFRC construction to enhance its profile feature coverage. The AFRC image can then be automatically used in the next module for AutoROI detection and DeepStacking. This operation can also be done in batches for all groups.

General Protocol to build AFRC image (process is similar for batch operation using multiclass AFRC tab):

- Select any image from the Cropped ROI folder for two different classes of contours (in single pair-wise) or single class for multiclass AFRC.
- Set the parameters (or use default settings) and apply image filters if desired.
- Process the AFRC image by "**Process AFRC image**" or "**Process multiclass AFRC image**" button. Alternatively, same operation can be done by using option in "**Tools**" menu bar.
-

iii. Single and multiclass deepstacking dataset construction


CRISP-II: Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-...
Settings
Tools
Explorer
Skins
Help
I) GC×GC-TOFMS contour processing ▸
II) GC×GC-TOFMS contour simulation ▸
III) GC×GC-TOFMS contour classification
About CRISP-II
Gradient Thresholding | Contour extraction
Single pair AFRC
Single pair deeptsacking
Multiclass AFRCs
Multiclass deepstacking
Deepstacked output location
....
Refresh
Pre-processing
Scoring
Compute ROIs
Apply Manual ROI
Load ROIs for DeepStacking
Build DeepStack data
Save config.

Contour pair similarity computation method: VGG16

Class-A image: ./datasets/AGTH_CKD-GCxGCTOFMS_DATA/Normal/afrc_image/afrc_image.jpg

Class-B image(control): ./datasets/AGTH_CKD-GCxGCTOFMS_DATA/Normal/afrc_image/afrc_image.jpg

DS ROIs filepath: ./roi_data/ds_roi_HD_DMs.data

./datasets/AGTH_CKD-GCxGCTOFMS_DATA/HD_DM

./datasets/AGTH_CKD-GCxGCTOFMS_DATA/Normal

□ Randomly select a file for each class of images

Scan window size: 150

Window shift: 150

SIM threshold (%): 99

Stacking segments: 5

☑ Enable deep stacking feature

☑ Avoid overlapping during stacking

□ Show computed ROI during selection

□ Autoset parameters

☑ Store computed ROI graph

☑ Show scan and score graph

```
[ SN.  |   SSIM values   |   ~ window range   ]
0           0.67941922     (1350, 1500)
1           0.85283554     (1200, 1350)
2           0.85770446     (600, 750)
3           0.89183038     (0, 150)
4           0.92081070     (450, 600)
5           0.93228310     (750, 900)
6           0.93487221     (1050, 1200)
7           0.93709755     (300, 450)
8           0.95453739     (150, 300)
9           0.97589296     (900, 1050)
```

status : Program launch

0%

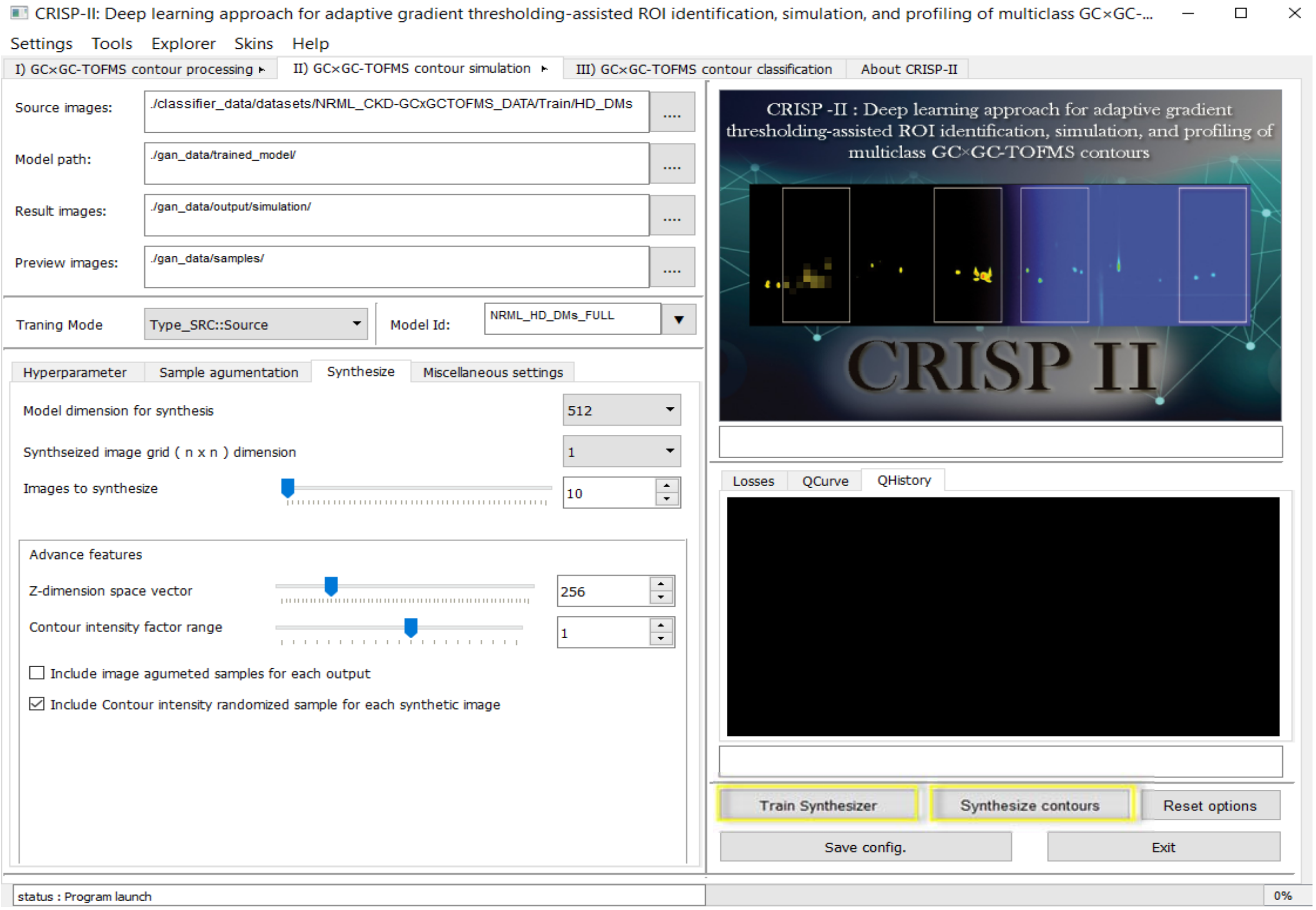
This module performs multiple critical tasks for constructing the final classifier training database for contour simulation and classification. It includes several customized models/algorithms to identify multiple ROIs between two AFRC images, such as retrained VGG16 (by default), SIAMESE, SSIM, PNSR, Hamming, and FID (details in the publication). The top-N (N=5 by default) ROIs are identified based on user settings like similarity threshold, scanning window size, and window overlapping amounts. During processing, the scanning window slider evaluates features of contour data in AFRC images of each class (e.g.: Disease class *v/s* control) from left to right, generating scores for each window area and a list of coordinates with scores for the top-N least similar regions. Users can view a graph of the scores along the R1 dimension of the contour data. Then, contour feature images for all individual contours in each group can be generated as "DeepStacked" images using the DeepStack dataset builder. This feature-stacked dataset can be used for simulation to amplify contrasting features between different profiles. Once the DeepStacking dataset is generated, it can be used to simulate feature vectors for further classification. This operation can also be performed pairwise (single pair Deepstacking tab) or in batch mode for multiclass data (multiclass Deepstacking tab).

General Protocol for Auto ROIs identification and Deep Stacking Tab:

- Select single images from each group.
- Select AFRC contour image for each class.
- Set the correct scoring method (default is VGG16).
- Set the window shifting parameters.
- Run the Compute ROIs by pressing "**Compute ROIs**" button.
- Build Deepstacking dataset by pressing "**Build DeepStack data**" in single-pair deepstacking or similarly for multiclass deepstacking tab.
- Once all parameters and paths are setup, same operation can be done by using option in "**Tools**" menu bar.

Note: *You can skip this section if you want to use the entire contour plot for simulation and classification. However, if you use deepstacking option, the same ROIs deep stacking configuration should be used for all steps in preprocessing the validation or inference cohort (samples).*

B. GAN-based GC×GC-TOFMS contour image training & synthesis


CRISP-II: Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-...
Settings
Tools
Explorer
Skins
Help
I) GC×GC-TOFMS contour processing ▸
II) GC×GC-TOFMS contour simulation ▸
III) GC×GC-TOFMS contour classification
About CRISP-II
Hyperparameter
Sample agumentation
Synthesize
Miscellaneous settings
CRISP -II : Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-TOFMS contours
CRISP II
Losses
QCurve
QHistory
Train Synthesizer
Synthesize contours
Reset options
Save config.
Exit

Source images: ./classifier_data/datasets/NRML_CKD-GCxGCTOFMS_DATA/Train/HD_DMs

Model path: ./gan_data/trained_model/

Result images: ./gan_data/output/simulation/

Preview images: ./gan_data/samples/

Traning Mode: Type_SRC::Source

Model Id: NRML_HD_DMs_FULL

Model dimension for synthesis: 512

Synthseized image grid (n x n) dimension: 1

Images to synthesize: 10

Advance features

Z-dimension space vector: 256

Contour intensity factor range: 1

☐ Include image agumeted samples for each output

☑ Include Contour intensity randomized sample for each synthetic image

status : Program launch

0%

The CRISP features advanced contour simulation neural networks based on Generative Adversarial Networks (GANs), which are highly customizable. The generator can train models using contour images of 512×512 pixels, even with a limited sample size, achieving good results. Larger sample sizes improve outcomes based on the diversity of the original images. Users can adjust hyperparameters and other settings like FIDs, Z-vectors, RELU, learning rates, and model optimizers (II GC×GC-TOFMS Tab → Hyperparameters).

The image augmentation tab (Contour synthesis -> Augmentation) allows users to apply various filters (e.g., dilation, distortion, erosion, contrast, brightness, noise, edge sharpening) to initially expand the diversity of limited contours fed to the generator, enhancing the dataset without significantly altering the overall profile of the input contours. This is particularly useful in low-sample conditions. Real-time graphs of model loss and previews, along with FID curves, are shown during training to monitor the generator's status. The qScore (forming the qCurve) is a custom metric indicating the quality of the output (0.0000-1.8), mainly based on the sharpness or blurriness of the synthetic image compared to the original contour image distribution. If the sharpness of images is similar to the source data, the qScore will be around ~1.8. Although not entirely reliable, it provides a trend of ongoing simulation training as a real-time feature update.

Training models and their history can be stored or restored at any point during the training, along with the configuration. Once the model is sufficiently trained (based on FID curves, model loss, and preview images), it can be used to synthesize entirely new contour plots without requiring the original datasets (B. Contour synthesis → 3. Synthesize) in a customizable grid of 1×1 to 10×10 shape. There is an advanced option to control the intensity of entities in the synthesized contour, which often relates to the concentration of metabolites. This feature works best when the source images have minimal background noise or column bleeding. Image augmentation can also be applied to the synthesized output contours, increasing the diversity of the synthesized samples. Manipulating the Z-vector can lead to higher diversity in the generated samples. There are myriad of other options that users can choose to utilize for image manipulation and preview modes based on custom requirements.

General Protocol for Training GANs model:

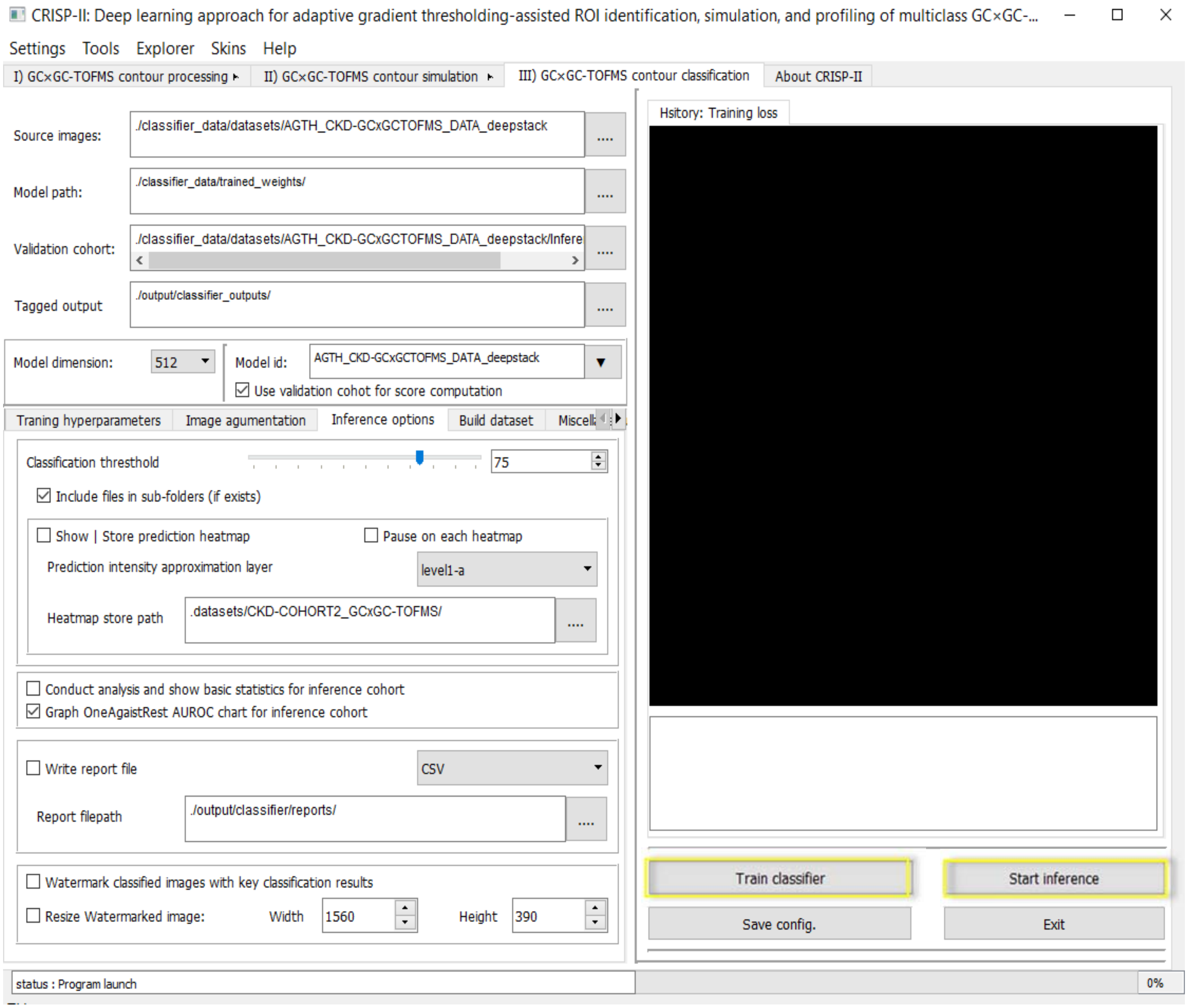
- Select the folder for source images (Source path).
- Set the path for trained models (Model path) and provide a unique Model ID.
- Set the path for storing preview images (Preview images path).
- Setup hyperparameters and image augmentation.
- Start training (models can be immediately stored and preview images updated during training using hotkeys in the simulation preview window).
- Press "**Train** Synthesizer" button under GC×GC-TOFMS contour classification Tab to train. Alternatively, same operation can be done by using option in "**Tools**" menu bar.

General Protocol for GANs-based GC×GC-TOFMS synthesis:

- Select the trained model (Model path) and provide the exact Model ID used for training & storage.
- Set the output location (Result images).
- Set the number of images to synthesize and image grid (optional)
- Set the required parameters (Z-dim, contour intensity factors; see article for details).
- Apply image augmentation (optional) for each simulated image.
- Press "**Synthesize contours**" button under GC×GC-TOFMS Tab to generate new samples. Alternatively, same operation can be done by using option in "**Tools**" menu bar.

Note*: Do not mix different extraction modes (Normal/AGTH/FXTH) during contour simulation and training. Consistency in modes and ROIs/Deepstacking (if applied earlier) is crucial. The source data is not required once the model is sufficiently trained. All models are compatible with continued training. If you plan to use the synthesize image for augmentation, use only 1x1 "n=1" for image grid uring image synthesis.*

C. Multiclass GC×GC-TOFMS contour training & inference



CRISP-II: Deep learning approach for adaptive gradient thresholding-assisted ROI identification, simulation, and profiling of multiclass GC×GC-...

Source images: ./classifier_data/datasets/AGTH_CKD-GCxGCTOFMS_DATA_deepstack

Model path: ./classifier_data/trained_weights/

Validation cohort: ./classifier_data/datasets/AGTH_CKD-GCxGCTOFMS_DATA_deepstack/Infere

Tagged output ./output/classifier_outputs/

Model dimension: 512

Model id: AGTH_CKD-GCxGCTOFMS_DATA_deepstack

☑ Use validation cohot for score computation

Classification thresthold 75

☑ Include files in sub-folders (if exists)

☐ Show | Store prediction heatmap ☐ Pause on each heatmap

Prediction intensity approximation layer level1-a

Heatmap store path .datasets/CKD-COHORT2_GCxGC-TOFMS/

☐ Conduct analysis and show basic statistics for inference cohort

☑ Graph OneAgaistRest AUROC chart for inference cohort

☐ Write report file CSV

Report filepath ./output/classifier/reports/

☐ Watermark classified images with key classification results

☐ Resize Watermarked image: Width 1560 Height 390

Hsitory: Training loss

status : Program launch

0%

GC×GC-TOFMS training: This is the final module of Contour Profiler, utilizing customized state-of-the-art deep convolutional neural networks (D-CNN) like VGG16 (default), VGG19, Inception V3, DenseNet, and ResNet for contour classification using transfer learning. This technique is effective even with a relatively small number of samples. The module has two submodules: **(i) Trainer**: This submodule can train on multiple classes of GC×GC-MS data. The data sets are divided into training and validation sets to evaluate training and validation accuracy, receiver operator curves, and model loss function. It includes a built-in image augmentation option that can randomly apply various image augmentation operations (e.g., shearing, skewing, and distortion) to increase the diversity of the training data. CRISP also has the option to view classification performance for validation cohort if the cohort is provided in same format as the training/testing set. This also provides the one-vs-rest AUROC for model performance on each class for the multiclass dataset training.

GC×GC-TOFMS Inference: This is the final step of the CRISP platform, where the trained classifier model is used to infer unknown samples based on preset thresholds. The GUI provides real-time visualization of classification and validation accuracy, and model loss function, allowing users to instantly view the trained model history. The inference will also generate the general statistics for the validation cohort. Heatmaps for the inference samples can be generated along with the watermarked inference output. The models can continue training with updated datasets, and the training configuration can be stored for future use and inference on unknown samples. A report file is generated, including the source images used for inference and optionally tagged images, heatmaps with their corresponding classification confidence.

General Protocol for classifier training

1. Set the source image dataset (Source images). Optionally, select the validation cohort if available in same format as Training dataset to see model performance on validation set during model training.
2. Set the model storage path (Model path) and unique classifier model ID (Model id).
3. Select the Dense Neural Net for transfer learning (default is VGG).
4. Set the hyper parameters (learning rate/decay, optimizers, batch size, training iterations, etc.).
5. Configure image augmentation (options in the image augmentation tab).
6. Start training by pressing "**Train classifier**" under GC×GC-TOFMS contour classification tab.

General Protocol for Inference:

- Select the trained model path (Model path).
- Input the name of the trained classifier model ID (Model id).
- Set the input image contour source (Test images).
- Set the output inference result folder (Tagged images), report file path, classification threshold, and report file output format (under the Inferencing option tab).
- Start inferencing by pressing the "**Start inference**" button under GC×GC-TOFMS contour classification tab.

Note: *The preprocessing of data for inferencing should be done in the same manner as the model was trained (i.e., full frame or ROI input in the same image mode). For deep-stacked contour images, the inference input should use the same coordinates as the deep-stacked images used for the training model. Use the (1A & C. AutoROIs and Deep Stacking Tab) to load the deep stacking data coordinates and preprocess the input images before inferencing. CRISP Classifier Training | Inference module has builtin database construction module which allows generation of custom datasets. The images can be multiple different classes : orignial, simulated or mixed contours of same type (CAUTION: Avoid mixing Deepstacked and full-conotur images)*

CRISP pre-processed experimental Dataset for demonstration of its utility.

CRISP-II comes with a set of experimental preset dataset based on our own in-house clinically validated GC×GC-TOFMS images that were used for dataset construction, model training and validation. Given that CRISP environment is setup, Users can use this dataset just by unpacking the CRISP-II software package and it processes out-of-box by following the user guidelines.

SETUP & MISCELLANEOUS

Dataset class name annotation rules

- The classes are indicated by their folder names within the main source directory (e.g., HD_DMs, HD_NO_DMs, PD_DMs, PD_NO_DMs, and NORMAL), and these names are automatically used as class IDs.
- To ensure reproducibility and maintain the correct order of classes during training, the folder names must be consistent between the training and validation cohorts.
- The model retains the class names for annotating inference results.
- All samples for inference, validation, or testing must undergo the same pre-processing as the contour image dataset used for training the classifier.

Requirements for Python3 installation

Install the requirement for the minimum GPU version of the python

```
pip install -r requirements_gpu.txt
```

Install the requirement for the minimum CPU version of the python

```
pip install -r requirements_cpu.txt
```

The standalone windows package of CPU version of python (which is slow but relatively simple than GPU version) google drive link

```
https://github.com/vivekmathema/GCxGC-CRISP/edit/main/README.md
```

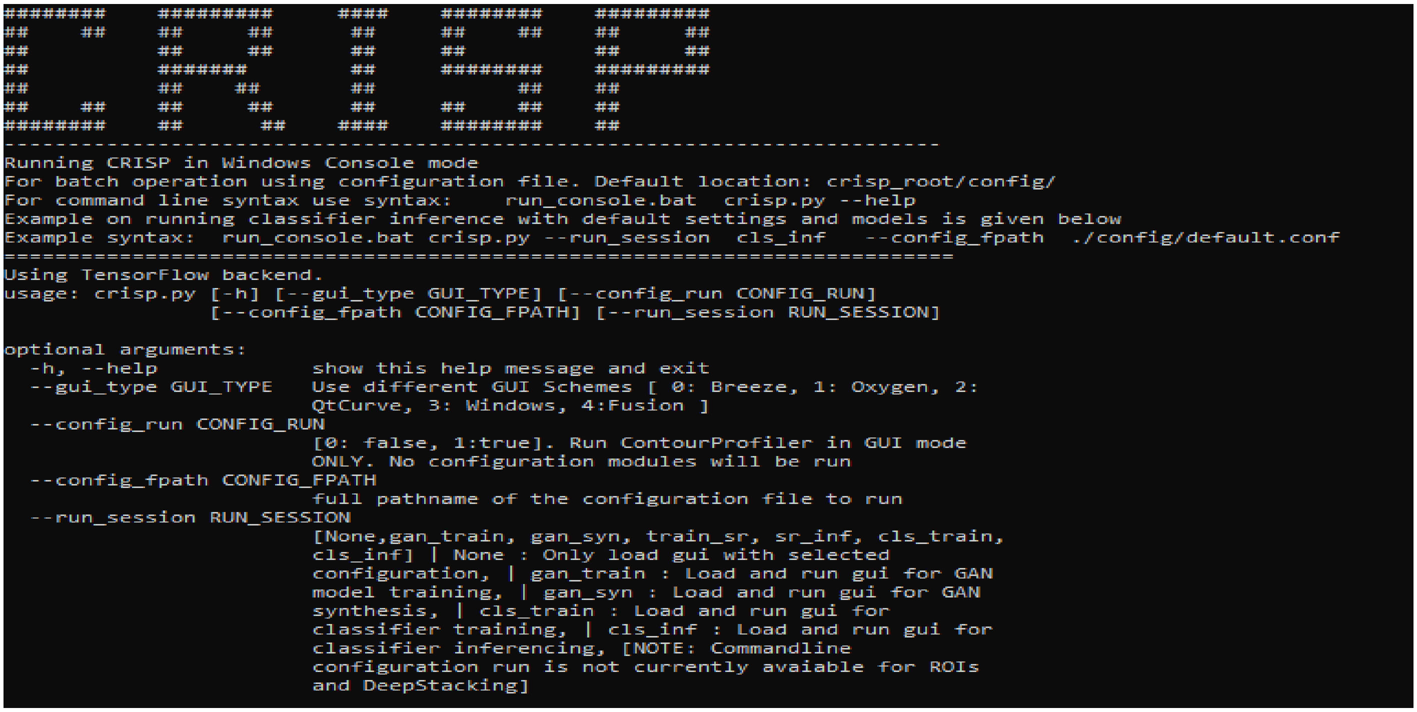
For creating Anaconda environment as OS independent version of CRIP, we currently recommend to use only CPU version. The GPU version requires advance CUDA installation knowledge for Linux and may not be suitable for starters.

For the CPU version of the CRISP Anaconda environment, users might need to adjust the installation versions if there are updates to repositories or changes in conda channels. The provided commands will install the conda equivalents of the pip3 requirements needed to run CRISP, but they may not be compatible with certain versions of Ubuntu OS.

```
1) conda create --name GCxGC_CRISP python=3.6

2.1) conda install --file requirements_cpu.txt (for CPU version, recommended for Linux Ubuntu)

2.1) conda install --file requirements_gpu.txt (for GPU version, recommended for Windows OS)

3) conda activate GCxGC_CRISP

4) (GCxGC_CRISP env) conda > python3 crsip.py
```

CRISP Command line parameters information

A preview of the CRISP command line help message is provided which can be obtained using syntax : python3 crisp.py --help



```
########    #########     ####    ########    #########
##    ##    ##     ##      ##     ##    ##    ##     ##
##          ##     ##      ##     ##          ##     ##
##          #######        ##     ########    #########
##          ##    ##       ##           ##    ##
##    ##    ##     ##      ##     ##    ##    ##
########    ##      ##    ####    ########    ##
------------------------------------------------------------------------
Running CRISP in Windows Console mode
For batch operation using configuration file. Default location: crisp_root/config/
For command line syntax use syntax:    run_console.bat  crisp.py --help
Example on running classifier inference with default settings and models is given below
Example syntax:  run_console.bat crisp.py --run_session  cls_inf   --config_fpath  ./config/default.conf
=========================================================================
Using TensorFlow backend.
usage: crisp.py [-h] [--gui_type GUI_TYPE] [--config_run CONFIG_RUN]
                [--config_fpath CONFIG_FPATH] [--run_session RUN_SESSION]

optional arguments:
  -h, --help            show this help message and exit
  --gui_type GUI_TYPE   Use different GUI Schemes [ 0: Breeze, 1: Oxygen, 2:
                        QtCurve, 3: Windows, 4:Fusion ]
  --config_run CONFIG_RUN
                        [0: false, 1:true]. Run ContourProfiler in GUI mode
                        ONLY. No configuration modules will be run
  --config_fpath CONFIG_FPATH
                        full pathname of the configuration file to run
  --run_session RUN_SESSION
                        [None,gan_train, gan_syn, train_sr, sr_inf, cls_train,
                        cls_inf] | None : Only load gui with selected
                        configuration, | gan_train : Load and run gui for GAN
                        model training, | gan_syn : Load and run gui for GAN
                        synthesis, | cls_train : Load and run gui for
                        classifier training, | cls_inf : Load and run gui for
                        classifier inferencing, [NOTE: Commandline
                        configuration run is not currently avaiable for ROIs
                        and DeepStacking]
```

---

python3 CRISP.py [-h | --help]

[--gui_type GUI_TYPE] [--config_run CONFIG_RUN]
[--config_fpath CONFIG_FPATH] [--run_session RUN_SESSION]

Optional arguments:

-h, --help Shows this command line help message and exits CRISP

--gui_type GUI_TYPE
Use different GUI Schemes. Five types available [0: Breeze, 1: Oxygen, 2: QtCurve, 3: Windows, 4:Fusion]

--config_run CONFIG_RUN [Set 0: False, 1:True]. Run ContourProfiler in GUI mode ONLY. No configuration modules will be run and CRISP will open GUI with default settings

--config_fpath CONFIG_FPATH full pathname of the configuration file to run. The Configuration file will be run without any user input or confirmation

--run_session RUN_SESSION [None,gan_train, gan_syn, cls_train, cls_inf] | None : Only loads GUI with selected configuration. Following modes are avaialble

```
gan_train : Load and run gui for GAN model training
gan_syn   : Load and run gui for GAN synthesis
cls_train : Load and run gui for classifier training
cls_inf   : Load and run gui for classifier inferencing
```

```
NOTE: Command line configuration run is not currently available for ROIs and Deepstacking.
Due to large numbers of parameters the definition of each parameter is commented in configuration file itself.
The definitions of most parameters are presented as tool tip text in status bar of GUI interface.
```

The CRISP software pipeline will undergo continuous development, with minor bugs being fixed over time. The primary goal of making the software open source is to enable a larger community to participate, contribute, and assist in the customization and development of deep learning-based techniques for GC×GC-TOFMS contour image metabolomics.