# General Utility Manual for Deep learning-based CRISP Software (under review)

The source code and minimal example dataset, models are available in Github:

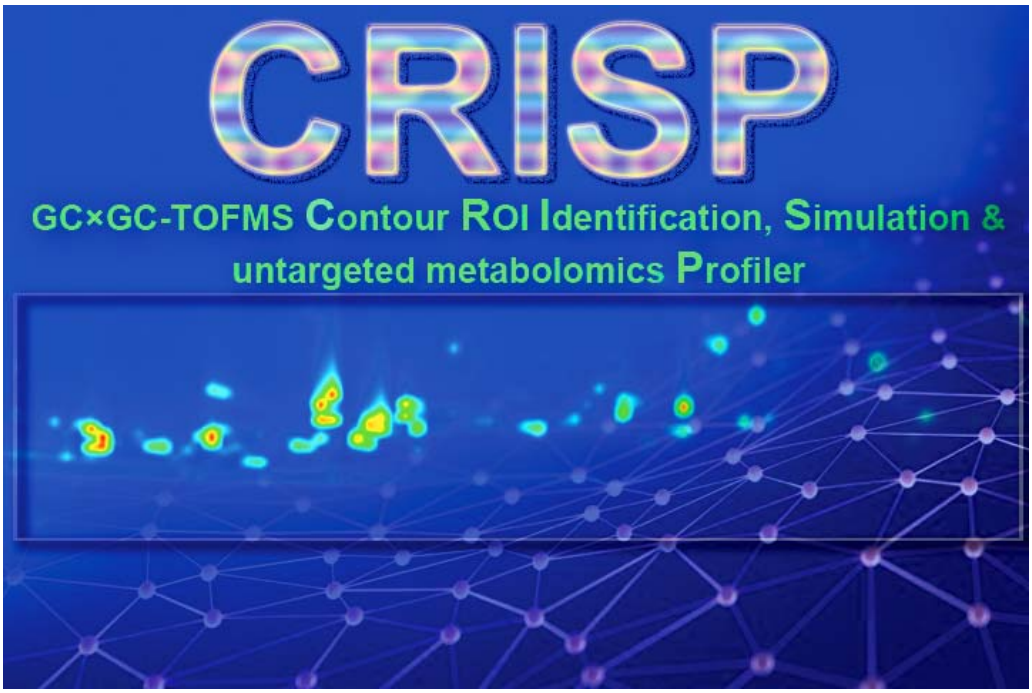
Weblink: https://github.com/vivekmathema/GCxGC-CRISP

(Full source codes and official datasets are only available after acceptance of manuscript)

This manual for CRISP can be downloaded from Github:

Weblink: https://github.com/vivekmathema/GCxGC-CRISP/blob/main/Manual/CRISP_manual.pdf

# CRISP: A deep learning architecture for GC×GC-TOFMS Contour ROI identification, Simulation, and Profiling in imaging metabolomics




## 1. GENERAL INFORMATION

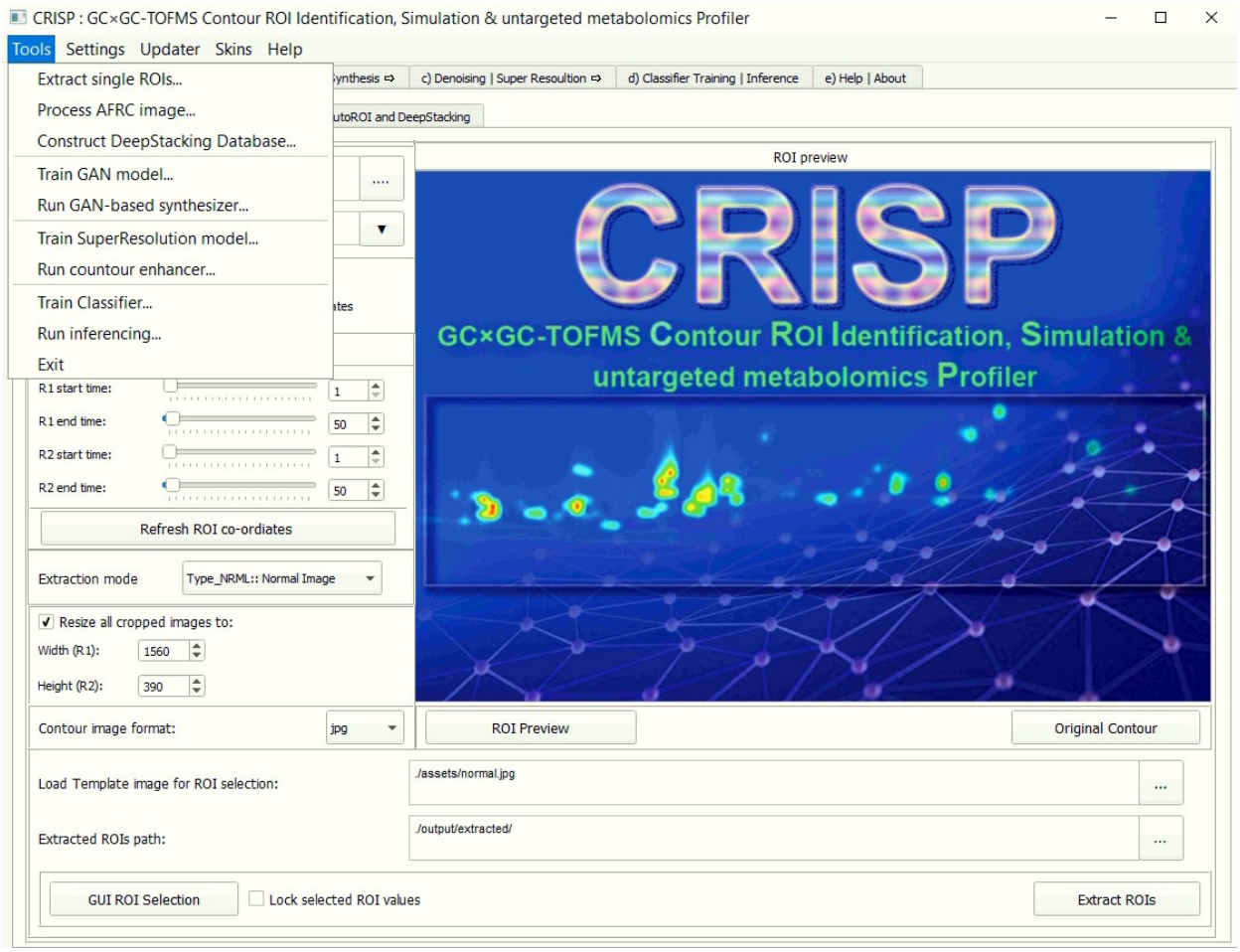
**C**ontour **R**OI **I**dentification, **S**imulation, and **P**rofiling (CRISP) is the novel platform that integrates multiple existing and new state-of-art deep learning (DL) platforms for untargeted profiling of Two-Dimensional Gas Chromatography and Time-of-Flight Mass Spectrometry (GC×GC-TOFMS)-generated data in terms of contour data. The open-source software toolkit is responsible for feature detection, simulation, data enhancement, and classification of GC×GC-TOFMS-generated contour for the untargeted metabolite profiling. The integrated platform has a novel ability to automatically construct aggregate feature representative contour (AFRC) from a group of supplied contour data belonging to different classes/groups to reduce manual selection biases and subsequently auto-identify contrasting regions of interest (ROIs) within this contour based on AFRC. These ROIs can be algorithmically detected and stacked together, using our algorithm termed as "Deepstacking*" to create a new representative feature map that maximizes the contrasting features and maximizes the profiling accuracy for each group during classifier training. The Integrated Generative Adversarial Network (GAN) is the main engine that simulates the synthetic Contour data based on the real samples. The Integrated GAN in use is a modified version of SGAN/WGAN that minimizes gradient vanishing and Lipschitz Constraint contained in general SGAN and WGANs. The synthesized output is computed for Fréchet inception distance (FID) to evaluate their quality of likeliness to the source data. The integrated modules have the framework for efficient high-resolution contour image synthesis (64×64 - 512×512 pixels) along with customizable contour intensity manipulation. The synthesized images can be

subjected to a contour image super-resolution (SR) model based on Cascading Residual Network. The CRISP's SR network is fully customizable. It has been customized by default and trained using our Contour-like high resolution (HR) dataset to maximize the model's capabilities to enhance contour image data. The Classifier is a collection of a customized version of a state-of-art Convolutional neural network (CNN) including VGG16/VGG19/Inception/RasNet/DenseNet for detecting features and obtaining maximum efficient models. Instead of the usual 224×224 Input dimensions, the model can input (224×244 or 512×512) dimensions along with a customizable model optimizer feature to maximize the contour feature detection capabilities. All models once trained can be restored, continued training, and used independently to inference their corresponding inputs. All models created by Contour Profiler have their summary, ROIs Deep stacking coordinates, logs, and training history including models losses, FIDs* and qScores* stored whenever applicable. Taken together, the CRISP provides an all-in-one open source integrated platform for advanced profiling of non-targeted GC×GC-TOFMS Contour data. [Note: * see article for details]

## 2. SOFTWARE ARCHITECTURE AND ITS USES

The CRISP is designed for GC×GC-TOFMS and consists of four major components: (A). ROI & Deep Stacking (B). Contour Simulation & Synthesis (C). Contour Enhancement (D). Contour Classifier 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. Please Note once all steps A-D should be done for the same Mode during inference of unknown sample.

# CRISP has a full operational graphical user interface (GUI) which allows easy operation of each modules




CRISP has myriad of option in GUI which cannot be demonstrated here. The substantial amount of details can be found in the Supplementary Materials section of the article.

# A. ROI & Deep Stacking

•Manually choose ROI (1A. Manual ROI selection): This section allows the user to manually select a single ROI graphically or based on input coordinates. It allows cropping of the region to get a single desired ROI with multiple additional options such as resizing, applying image filters, and output file type selection options. The extraction of initial contour data can be done in Normal RGB color mode which uses a pre-trained neural network for holistic edge-based feature extraction). Please note, once the Contour data is extracted in a certain mode, the same mode setting should be followed for all contour groups and future inferencing. Selection of ROI for only a single image per group folder is required, all images in the folder will be extracted for the same ROIs automatically. •General Protocol: (1A. Manual ROI selection tab) Select single images from each group --> Select single ROI/entire image from Contour --> Extract ROIs (new folder will be created with extracted ROIs for all Contours). (Can apply additional mode filters like resizing, denoise as described earlier)

• AFRC construction (1B. Build AFRC): This is a novel procedure that allows the generalization of the groups of contour images that belong to pre-defined profiles in terms of weighted aggregate feature extraction. This allows dominant features to represent exclusively in the final single representative image and scarce or outliner features will be minimized algorithmically. This will result in a final AFRC, which will represent common/average dominant major features present in the group, and reduces manual selection bias. The parameters such as FPS, weight accumulation factor, and cyclic image duration will control the number of features to be represented in the final AFRI image. Several image augmentation features (e.g.: blur, noise, erosion, dilation) can be applied during the construction of AFRC image to enhance its coverage of the profile feature. The AFRI image can be automatically fetched into the subsequent module for AutoROI detection & DeepStacking. •General protocol: (1B. Build AFRC tab) Select any image from the Cropped ROI folder from 1A for two different classes of Contours --> Set the parameters (or use default) and apply filters if desired --> Process AFRC

• Multiple ROIs detection & Feature-based dataset construction (1C. AutoROIs and Deep Stacking): This module does multiple important tasks related to final database construction for Contour simulation and classification. The module has several customized models/algorithmic means to identify multiple ROIs between two AFRI images. These include retrained: VGG16 (more preferred CNN model), SIAMESE, SSIM, PNSR, Hamming, and FID (details in publication paper). The top-N number of ROIs are identified based on user settings: Similarity threshold, scanning window size, window overlapping amounts. During processing, the scanning window from left to right will evaluate the features of Contour data in AFRI images and output the scores of each window area and a list of coordinates with scores will be generated for top-N least similar regions. There is an option to view a graph of the scores along the R1 dimension of the contour data. Then the contour feature images for all individual contours in each group can be generated as "DeepStacked" images using the DeepStacked dataset builder. This feature-stacked dataset can be used for simulation instead of simulating entire contour plots for amplifying the contrasting features between different profiles. Once, the Deepstacking dataset is generated, it can be used for simulation of the feature vectors that will be further used for classification. •General Protocol: (1C. AutoROIs 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 --> Build DeepStacking Dataset [Note: *You can skip this section if you want to use entire contour plot for the simulation and classification. However, if you use this option, the same ROIs deep stacking configuration should be used for all steps in CRISP].

# B. Contour modeling & Generator (Contour Synthesis & Inference)

The CRISP has advance Contour simulation neural nets based on Generative adversteral network (GANs) with multiple customizable features. The generator can train the model from 64×64 - 512×512 pixel contours with limited sample size and good outcome. The larger sample size can provide better results based on the diversity of the original images. There is an option to set hyperparameters and details options for setting FIDs, Z-vectors, RELU, Learning rates and model Optimizers (B. Contour synthesis --> 1. Hyperparameters). The image augmentation (B. Contour synthesis --> 2. Augmentation) tab allows the addition of multiple different filters (e.g: dilation, distortion, erosion, contrast, brightness, noise, edge sharpening, etc.) to initially expand the diversity of limited contours being fed to the generator without actually creating any significant change in the overall profile of the input contours. This enhances the dataset for the generator to train better especially in low-sample conditions. The real-

time graphs of model loss and preview of the model along with FID curves will be shown during training for getting the status of the generator. The qScore (which forms qCurve) is a customized matrix that will tentatively indicate the quality of the output mainly based on the sharpness/blurriness feature of the synthetic image during training as compared to the original contour image distribution. If the sharpness of images is relatively similar to that of source data, the qScore will approximately be equal to approximately 1.7. This score gives the user the trend of ongoing simulation training as a real-time model feature update. The training models and their history can be stored/restored at any point of the training along with the configuration. Once the model is trained enough (based on FID curves, Model loss, and preview images) the model can be used for synthesizing entirely new contour plots without requiring original datasets (B. Contour synthesis --> 3. Synthesize) in a customizable grid of 1×1 – 10×10 shape. There is also a novel advanced option to control the intensity of entities in the synthesized Contour which in the real situation often relates to the concentration of metabolites. This feature works best when the source images have a minimum of background noise or column bleeding. The image augmentation feature can be applied to the synthesized output contours as well which will assist in increasing the diversity of the synthesized samples. The advanced features of manipulating the Z-vector cab lead to higher diversity of the generated samples. It must be the same with the same ROIs/DeepStacking (if applied earlier). The source data is not required once the model is trained enough. All models are compatible for continued training.

•General protocol: (B. Contour Simulation | Synthesis) For Training, Select the folder for Source images (Source path) --> Set the path for trained models (model path) and provide a unique Model id (Model id) --> set the path for storing preview images (Preview images path) --> Setup Hyperparameters & Image augmentation --> Start training (the models can be immediately stored and preview images updates during training using hotkeys in simulation preview window). For 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 required parameters (Z-dim, Contour intensity factors “see article for details”) --> Apply image augmentation (optional) for each simulated images --> Start synthesis (Synths. Contour button).

# C. Contour Super-resolution

A super-resolution (SR) algorithm uses deep neural nets to clarify, sharpen, and upscale the digital image without losing its content and defining characteristics. This module is responsible for the enhancement of image resolution based on our own customized super-resolution model trained on our novel contour-like datasets, which enabled the production of a high-quality simulated contour dataset for the training classifier. The module is based on a modified version of Cascading Residual Network, which gives a very fast and accurate upscaling of images. The SR model has been trained on custom contour-like datasets which enables more precise upscaling and enhancement of contour images. However, the software provides a full-customizable option to build and train on your custom datasets for building database-specific SR models. The default upscaling is 2x.

•General protocol: (C. Super-resolution Tab) For training, selected dataset (SR training dataset) --> Setup hyperparameters (epochs, Learning rates, batch size) --> Train models. For Upscaling, select the folder containing low-resolution images (LR input) --> Select the folder containing trained models (SR

model path) --> Input name of the trained model (SR model id) --> Press upscaling button --> Upscaled images will be stored on SR output path.

# D. Contour Classifier Training and Inference

This is the final module of Contour Profiler which uses a customized version of state-of-the-art deep convolutional neural networks (D-CNN) such as VGG16, VGG19 Inception V3, DenseNet, and ResNet for classification of Contour using Transfer learning. The technique is applicable for a relatively small amount of samples. This module has two submodules: (i) Trainer, which can train on input multiple classes of GC×GC-MS data. The data sets are separated into training and validation sets for evaluating both training and validation accuracy, receiver operator curves, and model loss function. The training module was a built-in image augmentation option, which can randomly conduct multiple image augmentation operations (e.g. image shearing, skewing, and distortion) for generating more diversity for the training network. (ii) Inference, is the ultimate terminal point of the ContourGAN platform. This uses the trained classifier model to conduct inference on the unknown samples and preset threshold. The GUI has real-time visualization of classification and validation accuracy, model loss function which enables users to instantly view the training status. The models can be continued training with updated datasets and training configuration can be stored for future use and unknown sample inferencing. A report file is generated consisting of the source images for inferencing and tagged images (optional) with their corresponding classification confidence.

• General Protocol: (D. Classification Training | Inference tab) For training, Set the source image dataset (Source images) --> Set the Model storage path (Model path) and Unique classifier model ID (Model id) --> Select the Dense Neural Net for Transfer learning model (default VGG) --> Set the hyperparameters (Learning rate/decay, Optimizers, batch size, Training Iterations… etc.) --> Set image augmentation (options in image augmentation tab) --> Start training. For Inference, Select the trained model path (model path) --> Input name of trained classifier model ID (Model id) --> Set Input image contour source (Test images) --> Set output inference result folder (Tagged images) & report file path and classification threshold and report file output format (Under Inferencing option tab) --> Start Inferencing [Note: The preprocessing of data for inferencing should be done in the same manner as the model was trained for. i.e.: Full frame or ROI input in same image mode. For the Deep stacked contour images, the inference input should be of the same coordinates DeepStacked images as used for the training model. Use the (1A & C. AutoROIs and Deep Stacking Tab) and load the deep stacking data coordinates and preprocess the Input images before inputting for the inferencing.

# CRISP Classifier Training | Inference module has built-in database construction module which allows generation of custom datasets

```
The images can be multiple different classes: original, simulated or mixed
contours of same type (Avoid mixing Deepstacked and full-contour images)

Dataset class name annotation rules

--> Classes are represented by their folder name (e.g: 01Normal, 02ESRD-DM) with
prefix
```

```
-->The folder name should start with number "01Class, 02Class, 03Class..." which
will be the same order of code (0x) marked during inferencing/classification for
unknown samples

Custom classifier database construction hints:

Sample classifier dataset construction is

CRISP--> [d)Classifier|Inference ]  --> Build dataset --> whole dataset
                                        construction option for training
                                        classifier
basepath for entire classes         --> ./gan_data/output/simulation/
dataset build path                  --> ./sample_dataset/
dataset basename                    --> ESRD-NORMAL
Training dataset ratio              --> 85:15 default (i.e.: 85% training and 15%
                                        validation)
```

# Online Model download and storage option to Google Drive / HTTP server for automatic access

CRISP has a built-in online model download module. This can be found under the "online module" tab for Synthesizer, Super-resolution, and Classifier models. There are two ways models can be downloaded: (i) By using google drive (recommended) & (ii) direct HTTP server

**(i) Google Drive option:** Here, the link of the google shared folder should be provided along with google API, and files in the folder should be ZIP/Tar ONLY for each model The CRISP automatically downloads the zipped model one at a time, and has provision to unzip and set as "use for processing" The option "Use Google Drive" should be enabled for this option to be selected. A demo google drive API is provided within the software but users have to set their google drive an API for goggle in the CRISP to use their custom google drive for online model use. Currently CRISP only supports the download of the model, the upload of the zipped model has to be done manually. [NOTE: Users may have to manually download using provided Google shared folder if the API-based internal download shows any issues due to large file size of updated Google policies]

**(ii) Http server option:** Same as the Google Drive option. The main difference is that the models have to be stored in unsecured HTTP server. The option "Use google drive" should be deselected for this option to work.

# INSTALLATION OF CRISP SOFTWARE

```
We highly recommend all installations to be done on Python v3.6.8 (tested) 64 bit

CRISP utilizes the GC×GC-FORMS contour images from ChromaTOF, which is mainly based
on Windows OS. The CIRSP has been well-tested to run on Windows 10 OS (64 bit) but
all major function are compatible with Ubuntu 18.01 with all Python environment
installed (based on requirements_cpu/gpu.txt). However, as CRISP depends on multiple
external libraries, few manual adjustments may be required.
```

# 1A) PRE-BUILT CRISP CPU PACKAGE (out-of-the-box)

This is a recommended setup for non-technical users. User can directly downloaded the pre-built package and run the CRISP out-of-the-box

The standalone packages are compressed with 7Zip (https://www.7-zip.org/) to obtain maximum compression. Please insure that you have 7Zip or WinRAR (with Zip support) installed in your Windows systems to decompress the packages.

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

```
https://drive.google.com/file/d/19XqYFoLMD05HNyzrUrF0kMGTXsvsHZez/view?usp=sharing
```

After unzipping, goto "CRISP root" folder and run by double clicking "run-crisp-cpu.bat" , the CRISP GUI interface should appear and you are ready to go. Please manually go to each folder (expect for python3cpu_env) to see instruction in the Readme.txt folder for downloading additional pre-trained models or Datasets.

**The portable package contains full running example of most modules in CRISP. Just open and start the Train/Inference/Synthesize button to view default examples.**

# 1B) PRE-BUILT CRISP GPU PACKAGE (out-of-the-box | Requires CUDA 11.0 and cuDNN 8.00 in installed)

# The standalone GPU version of python is currently only available for Windows OS and required CUDA 9.5 or above along with cuDNN 7.5 or above along with packages in requirments_gpu.txt

Windows Standalone GPU version for python 3.6 (Google Drive link)

```
https://drive.google.com/file/d/1f2YJ1yPOKsvsU04Wn6RMotj1kiU0UyD-/view?usp=sharing
```

For the CUDA Core installation (for windows OS)

```
CUDA Toolkit 11.0 Download
https://developer.nvidia.com/cuda-11.0-download-
archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exeloca
l

cuDNN 8.00 Library for CUDA 11.0
https://developer.nvidia.com/compute/machine-
learning/cudnn/secure/8.0.4/11.0_20200923/cudnn-11.0-windows-x64-v8.0.4.30.zip
```

After unzipping, goto "CRISP_root" folder and run by double clicking "run-crisp-gpu.bat" , the CRISP GUI interface should appear and you are ready to go. Please manually go to each folder (expect for python3gpu_env) to see instruction in the Readme.txt folder for downloading additional pre-trained models or Datasets.

# Running CRISP using Command line interface

Current version of CRISP supports using configuration file to run Tran/inference/synthesis batch operation in CRISP. In order to run batch operation, goto windows console and change working directly to CRISP_root> Then, use run_console.bat from the command line in console.

```
Command line help: (console) CRISP_root > run_console.bat crisp.py --help
Example syntax   : (console) CRISP_root> run_console.bat crisp.py --run_session
                                          cls_inf   --config_fpath
                                          ./config/default.conf
```

# 2) MANUAL INSTALLATION OF CRISP

The manual installation requires cop yof ths CRISP source code

### 2A) Requirements for Python3 installation

Install the requirement for the minimum CPU version of the python

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

### 2B) Requirements for Python3 installation

## Install the minimum requirement for the GPU version of the python (Fully operational for Windows OS with pre-requisites of CUDA and cUDNN + MS VC++ Libs)

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

## Microsoft Visual C++ 2015 or 2019 Redistributable Update

Runtime-support may be needed to install for proper working of tesorflow-gpu. The 64 bit version can be directly installed from the supplied binaries in google drive:

```
https://drive.google.com/file/d/1aSzn8cbMDICrXjG4WZyqkr5tNdYu2sro/view?usp=sharing
```

Alternatively: Both 32-bit and 64-bit redistributions can be downloaded from:

```
https://www.microsoft.com/en-us/download/confirmation.aspx?id=52685
```

# ANACONDA ENVIRONMENT INSTALLATION

# For creating Anaconda environment as OS independent version of CRISP, 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. We will be providing support for installation in future as the project matures.

For CPU Version of Anaconda environment of CRISP. Users may have to Tweak the installation versions if any repository needs were updated or conda channel changes. These commands will simply install conda version of the pip3 requirements to run CRISP. Maynot not be suitable for some version 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)

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

3) conda activate GCxGC_CRISP

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

May need uses to adjust few libraries if there are some compatibilities issues.

# CRISP Command line parameters information

CRISP contains myriad of configuration option which is advisable to be run by using a single configuration file. Thus, CRISP uses a configuration file to load all settings at once and most modules can be immediately run without further user confirmation for each step.

---

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 ] (There is also additional option to change skin colour of the GUI to Skyblue, Grayshade and System default in GUI mode only)

--config_run CONFIG_RUN [Set 0: false, 1:true]. Run CRISP in GUI mode ONLY. No configuration modules will be run and CRISP will just GUI with default settings

--config_fpath CONFIG_FPATH full pathname of the configuration file to run. The Confiruation file will be run without any user input or confrimation (example: > python3.exe crisp.py ./config/default.cfg )

--run_session RUN_SESSION [None,gan_train, gan_syn, train_sr, sr_inf, cls_train, cls_inf] | None : Only loads gui with selected configuration. Following modes are available:

```
                    gan_train  : Load and run gui for GAN model training
                    gan_syn    : Load and run gui for GAN synthesis
                    train_sr   : Load and run gui for contour super resoultion
training
                    sr_inf     : Load and run gui for contour image enhancement
                    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
available for ROIs and DeepStacking.
                           Due to large numbers of parameters the defination of
each parameter is commented in configuration file itself.
                           The Definations of most parameters are presented as
tool tip text in status bar of GUI nterface.
```

---

**The CRISP software will be under constant development and minor bugs will be fixed overtime.** The main goal of making the software open source is for letting larger community to participate, fork and assist in custom use and continuous development of Deep learning-based techniques in GC×GC-TOFMS contour image metabolomics.