

---

## ANEXO A

### Functions of the Developed Framework

**Author: Robson Rogério Dutra Pereira**

---

The functions, operation modes, and variables of the developed framework are listed in Figura 76, Figura 77, Figura 78, Figura 79, Figura 80, Figura 81, Figura 82, and Figura 83, defined as `SMAxx_10_BCKns.py`.

The framework can be configured and initialized in three ways: through the framework code, by command lines in the terminal (prompt), and by a remote Python code defined as `SMAxx_10_BCKns_Main.py`, which is listed in Figura 84 and Figura 85.

Tabela 4 contains the variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`. The column prefix "**Module Variable**" is `modules."Module Variable"`. The equation for calculating the argument `overlapVAR` to obtain the overlap in pixel values between the patches is in Equation (2).

$$\text{round} \left( \frac{\text{module.patch\_sizeVAR} \times \text{module.overlapVARPerc}}{100} \right) \quad (2)$$

```

❷ SMAxx_10_BCKns.py > ...
1 ##########
2 """
3 SMAxx_10_BCKns.py
4
5 Created by: Robson Rogério Dutra Pereira on 01.Sep.2023
6 Last Modified: rrdpereira
7
8 Description: Add backbones_index, With Adam CHECK, DATETIME CHECK, DEFAULTS VALUES VARIABLES, and 3 ways to insert data configuration. This is sequential python script
9 | | and realize the training and validation with standard data according to RGB images patch
10 | | size and high resolution RGB images.
11
12 E-mail: robsondutra.pereira@outlook.com
13 """
14 #####
15 set_rootOut = [0,0,0]
16 TDDis00 = 0.5 # 5.0
17 TDDis01 = 0.5 # 10.0
18 selected_classesVAR = [2, 3]
19 #####
20 > import sys, time, os, datetime, glob, re ...
21 print(tf.version.VERSION)
22
23 import segmentation_models as sm
24 sm.set_framework('tf.keras')
25 sm.framework()
26
27 > import tifffile as tiff...
28 print(f"(Sys version) :|: {sys.version} :|:")
29 os.system("which python")
30 print(f"(Python version) :#: {python_version()} :#:")
31 print("----->>> os.path.dirname(sys.executable):#: {0}".format(os.path.dirname(sys.executable)))
32 print("----->>> re._version_#: {0}".format(re._version__))
33 print("----->>> sys.executable#: {0}".format(sys.executable))
34 > """ ...
35 #####
36 # Allow GPU memory growth
37 physical_devices = tf.config.experimental.list_physical_devices('GPU')
38 print("----->>> physical_devices = {0}".format(physical_devices))
39 > if physical_devices: ...
40 #####
41 # Check for available GPUs
42 gpu_available = 0 # 0: initial value; 1: GPU available; 2: GPU not available
43 gpus = tf.config.experimental.list_physical_devices('GPU')
44 print("----->>> gpus = {0}".format(gpus))
45 print("----->>> len(gpus) = {0}".format(len(gpus)))
46 > if len(gpus) == 1: ...
47 > elif len(gpus) == 0: ...
48
49 print("----->>> STATUS gpu_available = {0}".format(gpu_available))
50 #####
51 > models_inib = [ # Comment non used models to run "One Saved Model"...
52 > models_inib_loop = [ # Comment non used models to run "One Saved Model"...
53 #####
54 PLTShowChck = 0 # plt.show() --> 0: OFF; 1: ON
55 PLTShowTempPause = 0 # plt.show(temporary pause) --> 0: OFF; 1: ON; 2: PNG inference paused way
56 TESTImgNumberChck = 0 # 0: random; 1: fixed
57
58 > print("----->>> STATUS plt.show() = {0} ; plt.pause() = {1} ; plt.savefig() = {2} ".format(plt.show(), plt.pause(), plt.savefig()))
59 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
60 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
61 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
62 #####
63 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
64 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
65 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
66 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
67 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
68 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
69 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
70 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
71 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
72 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
73 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
74 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
75 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
76 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
77 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
78 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
79 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
80 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
81 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
82 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
83 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
84 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
85 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
86 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
87 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
88 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
89 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
90 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
91 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
92 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
93 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
94 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
95 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
96 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
97 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
98 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
99 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
100 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
101 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
102 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
103 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
104 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
105 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
106 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
107 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
108 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
109 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
110 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
111 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
112 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
113 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
114 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
115 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
116 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
117 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
118 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
119 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
120 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
121 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))
122 > print("----->>> STATUS TESTImgNumberChck = {0} ".format(TESTImgNumberChck))
123 > print("----->>> STATUS PLTShowChck = {0} ".format(PLTShowChck))
124 > print("----->>> STATUS PLTShowTempPause = {0} ".format(PLTShowTempPause))
125 > print("----->>> STATUS models_inib = {0} ".format(models_inib))
126 > print("----->>> STATUS models_inib_loop = {0} ".format(models_inib_loop))
127 > print("----->>> STATUS TDDis00 = {0} ; TDDis01 = {1} ".format(TDDis00, TDDis01))
128 > print("----->>> STATUS selected_classesVAR = {0} ".format(selected_classesVAR))
129 > print("----->>> STATUS set_rootOut = {0} ".format(set_rootOut))

```

Figura 76 – Overview of the developed framework SMAxx\_10\_BCKns.py. Source: Robson Rogério Dutra Pereira. (Part 1/8).

Tabela 4 – Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
.PLTShowChck	0:off; 1:on	Enable and disable the matplotlib.pyplot.show functions
.PLTShowTempPause	0:off; 1:on; 2:pause during inference	If <code>PLTShowChck</code> is on, you can enable and disable temporary visualization of matplotlib.pyplot.show, and also enable pause during inference in validation

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMaxx_10_BCKns_Main.py`

Module Variable	Values	Description
.TESTImgNumberChck	0:random; 1:fixed	Determines whether the inference in validation will be on the same sample for all trainings or random for each training
.TESTImgNumberVAR	0 to size of validation set	Value of the fixed sample in validation
.UNetNormaChck	UNetNormaChckM	The <code>UNetNormaChckM</code> argument can have the value 0:off; 1:on for BatchNormalization of U-Net or whether it will be 2: <code>mlp_nn</code> , 3: <code>FCN_atrous</code> ; 4: <code>IFCN_atrous</code> ; 5: <code>deep_lab</code>
.metricsChck	1: <code>metricsIOUFS</code> ; 2: <code>metricsJac</code>	Determines whether the metric will be Intersection of Union + F1 Score or Jaccard Index
.UshuffleChck	False:off; True:on	Enable and disable shuffle mode for <code>model.fit</code>
.mixed_precisionChck	0:off; 1:on	Enable and disable tensorflow keras <code>mixed_precision</code>
.date_timeChck	0:off; 1:on	Enable and disable date and time in Model Checkpoint
.split_loaded_dataChck	1:numpy split; 2:sklearn train test split; 3:random sample	Selects the dataset split mode into training and validation
.rootModChck	1: <code>metricsIOUFS</code> ; 2: <code>metricsJac</code>	Configured according to the HDF5 model metric, by Intersection of Union + F1 Score or Jaccard Index
.mt_mont_Chck	mt_mont_ChckM	The <code>UNetNormaChckM</code> argument for training can be 1: <code>mt</code> function, 2: <code>mont</code> function, and 3: <code>mlp_nn</code>

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
.patch_size_startVAR	0	The initial value of the Patch size
.patch_sizeVAR	patch_sizeVARM	The <code>patch_sizeVARM</code> argument is the Patch size of the image at the neural network input, which can be 128 px or 256 px for U-Net and backbones, and 384 px for PSPNet
.overlapVARPerc	overlapVARPercM	The <code>overlapVARPercM</code> argument determines the percentage of overlap between Patches with data augmentation
.overlapVAR	Equation (2)	Calculates the overlap between Patches in pixels
.patience_VVAR	patience_VVARM	The <code>patience_VVARM</code> argument determines the number of epochs if there is no change in the training values for Early Stopping
.batch_size_VVAR	batch_size_VVARM	The <code>batch_size_VVARM</code> argument determines the number of samples that will be propagated through the neural network and used in training before updating the weights of the trained variables, which in this case can be Batch size of 4, 8, 16, 32, and 64
.epochs_VVAR	epochs_VVARM	The <code>epochs_VVARM</code> argument determines the maximum number of epochs during training

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
.activationVAR	activationVARM	The <code>activationVARM</code> argument defines the activation layer of the output, which can be softmax: multi-class segmentation without overlap of class masks; and sigmoid: binary segmentation or multi-class segmentation with independent overlap or without overlap of class masks
.LRVAR	LRVAR_M	The <code>LRVAR_M</code> argument defines the learning rate for training
.restart_loss_thresholdVAR	0 to 1	Defines the threshold value to trigger the training restart trigger with the <code>mont</code> function
.max_consecutive_epochsVAR	1 to 10	Defines the maximum number of consecutive epochs, in the <code>max_consecutive_epochs</code> variable of the <code>mont</code> function in the framework, with the <code>loss</code> value during training above the value defined in <code>restart_loss_thresholdVAR</code> , before restarting training

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
<code>.consecutive_epochsVAR</code>	0	Defines the initial value of the <code>consecutive_epochs</code> variable in the <code>mont</code> function of the framework, whose value will increment based on the trained epochs. If the <code>loss</code> value is greater than <code>restart_loss_thresholdVAR</code> , and the current value of <code>consecutive_epochs</code> is equal to <code>max_consecutive_epochsVAR</code> , training is restarted
<code>.max_restart_attemptsVAR</code>	1 to 10	Defines the maximum number of training restart attempts, in the <code>max_restart_attempts</code> variable of the <code>mont</code> function in the framework
<code>.restart_attemptsVAR</code>	0	Defines the initial value in the <code>restart_attempts</code> variable of the <code>mont</code> function in the framework, with the purpose of recording the quantities of restarts
<code>.optimizerUChck</code>	1:Adam(LR) and 2:'adam'	Defines the optimizer compilation form in the U-Net model, whether it will be Adam based on Learning Rate (LR) 1:optim = Adam(LR), or whether it will be Adam without LR 2:'adam'

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
.optimizerBChck	1:Adam(LR) and 2:'adam'	Defines the optimizer compilation form in the U-Net model together with Backbone, whether it will be Adam based on Learning Rate (LR) 1:optim = Adam(LR), or whether it will be Adam without LR 2:'adam'
.TestSplittVAR	0.10 to 0.50	Defines the percentage of the <code>test</code> set split in a normal training
.TestSplittFVAR	0.10 to 0.50	Defines the percentage of the <code>test</code> set split in a fast training
.high_resoution_imgChckEACHI	0:off; 1:on	Enables or disables the execution of inference on high-resolution images not seen by the trained model, and saves the output in individual files
.high_resoution_imgChckHOALL	0:off; 1:on	Enables or disables the execution of inference on high-resolution images not seen by the trained model, and saves the output in a single file
.modelnetChck	1:Unet; 2:FPN; 3:Linknet; 4:PSP-Net	Defines the architecture of the neural network model used in training, which can be according to the one defined in <code>patch_sizeVARM</code> for 1:U-Net, 2:FPN, 3:Linknet; If it is 4:PSPNet, <code>patch_sizeVARM</code> should be changed to 384 px.
.encoderweightsChck	1 and 2	Defines whether the initial values of the model's encoder weights are 1: <code>encoder_weights='imagenet'</code> or 2: <code>encoder_weights=None</code>

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
<code>.encoderfreezeChck</code>	0:off; 1:on	Enables or disables random initialization of the decoder to avoid corrupting the weights of the trained encoders, in this case 0: encoder_freeze off and 1: encoder_freeze = on
<code>.asininputChck</code>	0:off; 1:on	MAIN asininput: Enables or disables the use of the remote Python file <code>SMAxx_10_BCKns_Main.py</code> , or only the Python file of the framework <code>SMAxx_10_BCKns.py</code> , following the following combination → asininputChck=1 + asininputChckIN=0 <code>SMAxx_10_BCKns_Main.py</code> + <code>SMAxx_10_BCKns.py</code> ; asininputChck=1 + asininputChckIN=1 and asininputChck=0 + asininputChckIN=0 <code>SMAxx_10_BCKns.py</code>

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
<code>.asinputChckIN</code>	0:off; 1:on	MAIN asinput terminal: Enables or disables the use of the remote Python file <code>SMAxx_10_BCKns_Main.py</code> , or only the Python file of the fra- mework <code>SMAxx_10_BCKns.py</code> , following the following combination —> <code>asinput- Chck=1 + asinputChckIN=0</code> <code>SMAxx_10_BCKns_Main.py</code> + <code>SMAxx_10_BCKns.py</code> ; <code>asinputChck=1 + asinput- ChckIN=1</code> and <code>asinput- Chck=0 + asinputChckIN=0</code> <code>SMAxx_10_BCKns.py</code>
<code>.prebckb_input1</code>	<code>backbones_in[i]</code>	Defines the pre-compilation of backbone 1 of the ensemble ac- cording to the indices of the list <code>backbones_in[i]</code> , where i starts from 0 to the last item in the list
<code>.prebckb_input2</code>	<code>backbones_in[i]</code>	Defines the pre-compilation of backbone 2 of the ensemble ac- cording to the indices of the list <code>backbones_in[i]</code> , where i starts from 0 to the last item in the list
<code>.prebckb_input3</code>	<code>backbones_in[i]</code>	Defines the pre-compilation of backbone 3 of the ensemble ac- cording to the indices of the list <code>backbones_in[i]</code> , where i starts from 0 to the last item in the list

*Continued on next page*

Tabela 4 – (*Continued*) Variables of the `run_modules` function after checking the initialization flags in the remote code `SMAxx_10_BCKns_Main.py`

Module Variable	Values	Description
.bckb_model01	models_inib[j]	Defines the HDF5 model 1 for the ensemble that is indexed in the list <code>models_inib[j]</code> , where j starts from 0 to the last item in the list
.bckb_model02	models_inib[j]	Defines the HDF5 model 2 for the ensemble that is indexed in the list <code>models_inib[j]</code> , where j starts from 0 to the last item in the list
.bckb_model03	models_inib[j]	Defines the HDF5 model 3 for the ensemble that is indexed in the list <code>models_inib[j]</code> , where j starts from 0 to the last item in the list
.ClassNumInf	less than the number of classes	Defines the value of <b>ClassNumInf</b> to be used for the sample selection algorithm, <code>number of classes - ClassNumInf = reference value for sampling</code> . The objective is to return a sample containing masks with more than 2 classes
.mainIN(argument)	argument	This is the argument that links the <code>mainIN</code> function to the <code>main</code> call of the framework <code>SMAxx_10_BCKns.py</code> . The value of <code>argument</code> can be the texts 'mt', 'mti', 'mi', 'mta', 'mtia', 'mia', 'mien' and 'miaen'. These texts define the operation modes of the framework that are defined in the <code>mainIN</code> function

```

# SMAXx_10_BCKns.py > ...
130 TESTImgNumberVAR = 339
131 UNetNormaChck = 0 # UNet with BatchNormalization() ---> 0: OFF; 1: ON
132 # Get the base filename without extension
133 current_file_name = os.path.splitext(os.path.basename(__file__))[0]
134 model_nname = "model_1" + current_file_name
135 file_nname = current_file_name + ".Ph"
136 # DEFAULTS VALUES FOR ASARGMENT Argument the mode
137 metricsChck = 1 # For the train ---> 1: metricsIOUFS and 2: metricsJac
138 UshuffleChck = True # For the model.fit shuffle ---> True: ON and False: OFF
139 mixed_precisionChck = 0 # For the tensorflow.keras.mixed_precision ---> 0: off and 1: on
140 date_timeChck = 1 # For the DATETIME on the Model Checkpoint ---> 0: off and 1: on
141 split_loaded_dataChck = 1 # split_loaded_data Functions ---> 1:numpy.split; 2:sklearn.train_test_split; 3:random.sample
142 rootModChck = 1 # For the load saved models ---> 1: metricsIOUFS and 2: metricsJac
143 mt_mont_Chck = 1 # For UNet Loss check ---> 1: mt function and 2: mont function
144 patch_size_startVAR = 0 # Start Patch size
145 patch_sizeVAR = 128 # Patch size: 128 or 256; 4:PSPNet (384 x 384, 3)
146 overlapVARperc = 25 # Overlap between patches in % of patch_sizeVAR
147 overlapVAR = round((patch_sizeVAR*overlapVARperc)/100) # Overlap between patches in pixels
148 patience_VVAR = 10 # Patience for Early Stopping in epochs
149 batch_size_VVAR = 32 # Batch size [4, 8, 16, 32, 64]
150 epochs_VVAR = 100 # Epochs size
151 activationVAR = "softmax" # Activation of net output; "softmax":multiclass segmentation with non overlapping class masks (your classes + background); "sigmoid": ...
152 LRVAR = 0.0001 # Learning Rate
153 restart_loss_thresholdVAR = 0.9999 # Define a threshold for loss to trigger a restart 0.9999
154 max_consecutive_epochsVAR = 7 # Define a maximum number of consecutive epochs with loss above threshold before restarting
155 consecutive_epochsVAR = 0
156 max_restart_attemptsVAR = 3 # Set your desired limit here
157 restart_attemptsVAR = 0
158 optimizerUChck = 1 # UNet Model compiler optimizer ---> 1:optim = Adam(LR); 2:'adam'
159 optimizerBChck = 1 # UNet + Backbone Model compiler optimizer ---> 1:optim = Adam(LR); 2:'adam'
160 TestSplitVAR = 0.10 # Test_size split for normal training
161 TestSplitFVAR = 0.50 # Test_size split for fast training
162 backbones_index = 0 # For the "SMAXx_10_BCKns_Creator.py" needs to be "0"(zero) and COMMENT this line for BACKBONES TRAINING
163 backbones_in = ["resnet18", "resnet34", "resnet50", "resnet101", "resnet152", "inceptionv3", "inceptionresnetv2", "vgg16", "vgg19", "seresnet18", "seresnet34", "seresnet50"]
164 # 0 1 2 3 4 5 6 7 8 9 10 11
165 backbone_input = backbones_in[backbones_index]
166 print("-----{0}----->>> 00 backbones_index".format(backbones_index))
167 print("-----{0}----->>> 00 backbone_input".format(backbone_input))
168 print("-----{0}----->>> 00 LRVAR".format(LRVAR))
169 inib_index = 0 # For the "SMAXx_10_BCKns_Creator.py" needs to be "0"(zero) and COMMENT this line for BACKBONES TRAINING
170 # models_inib_loop = ["model_Unet_densenet121_00", "model_Unet_densenet201_00", "model_Unet_efficientnetb5_00", "model_Unet_inceptionv3_00", "model_Unet_resnet152_00"]
171 # 0 1 2 3 4
172 models_inibpt = models_inib[inib_index]
173 high_resolution_imgChckEACHI = 0 # Inference in a high resolution on each images ---> 0:off; 1:on
174 high_resolution_imgChckHOALL = 0 # Inference in a high resolution on hold all the subplots images ---> 0:off; 1:on
175 modelnetChck = 1 # Architectures Model Net ---> 1:Unet; 2:FPN; 3:Linknet; 4:PSPNet (384 x 384, 3)
176 encoderweightsChck = 1 # encoder_weights of the Model ---> 1:encoder_weights='imagenet'; 2:encoder_weights=None
177 encoderfreezeChck = 0 # to only randomly initialized decoder in order not to damage weights of properly trained encoder ---> 0:encoder_freeze OFF; 1:encoder_freeze ON
178 asinputChck = 1 # MAIN asinput ---> 0:off; 1:on; "asinputChck:1|asinputChckIN:0" just remote file; "asinputChck:1|asinputChckIN:1" and "asinputChck:0|asinputChckIN:0"
179 asinputChckIN = 0 # MAIN asinput terminal ---> 0:off; 1:on; "asinputChck:1|asinputChckIN:0" just remote file; "asinputChck:1|asinputChckIN:1" and "asinputChck:0|asinputChckIN:0"
180 prebckb_input1 = backbones_in[29] # According to backbones_in:model_Unet_efficientnetb5_00[29]
181 prebckb_input2 = backbones_in[13] # According to backbones_in:model_Unet_resnet152_00[13]
182 prebckb_input3 = backbones_in[7] # According to backbones_in:model_Unet_vgg16_00[7]
183 print("00_prebckb_input1: {0}; prebckb_input2: {1}; prebckb_input3: {2}.".format(prebckb_input1, prebckb_input2, prebckb_input3))
184 ClassNumInf = 2 # Reference classes number for Training and Inference

```

Figura 77 – Overview of the developed framework SMAXx\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 2/8).

```

SMAXX_10_BCKns.py > ...
185 ##### ...
186 model_nname = model_nname + "_" + "OptU" + str(optimizerUChck) + "_" + "OptB" + str(optimizerBChck)
187 model_uname = model_nname + "_" + "UNet"
188 file_nname = "OptU" + str(optimizerUChck) + "_" + "OptB" + str(optimizerBChck) + "_" + file_nname
189 root = "./datasets/"
190 rootOut = "./out/" + time.strftime("%Y_%m_%d_Set") + str(set_rootOut[0]) + str(set_rootOut[1]) + str(set_rootOut[2]) + "/"
191 os.makedirs(rootOut, exist_ok=True)
192 logsOut = "./logs/"
193 os.makedirs(logsOut, exist_ok=True)
194 logsOut2 = "./logs/{}"
195 modelOut = "./saved_models/"
196 os.makedirs(modelOut, exist_ok=True)
197 out_folder_hires = rootOut + "hi_res/" + current_file_name + "/"
198 os.makedirs(out_folder_hires, exist_ok=True)
199 rootMod = models_inibpt #
200 # According to backbones_in:Resnet34+InceptionV3+Vgg16; Resnet50+InceptionV3+Vgg16; Resnet34+Resnet50+Resnet34; Resnet50+Resnet50+Resnet34
201 bckb_model01 = models_inib[2] # [2] model_Unet_efficientnetb5_00; models_inib[0] with zero index to run "One Saved Model"
202 bckb_model02 = models_inib[4] # [4] model_Unet_resnet152_00; models_inib[0] with zero index to run "One Saved Model"
203 bckb_model03 = models_inib[8] # [8] model_Unet_densenet121_00 OR [1] model_Unet_densenet201_00 OR [8] model_Unet_vgg16_00 OR [9] model_Unet_00; models_inib[0] with zero index to run "One Saved Model"
204 print("rootMod: {}".format(rootMod))
205 > imageI_paths = [...]
215 print("imageI_paths: {}".format(imageI_paths))
216 print("len(imageI_paths): {}".format(len(imageI_paths)))
217 # unseen_path = "./datasets/unseen/"
218 # unseen_list = os.listdir(unseen_path)
219 # image_files = [file for file in unseen_list if file.lower().endswith('_d.jpg')]
220 # imageI_paths = sorted([os.path.join(unseen_path, file) for file in image_files])
221 # print("imageI_paths: {}".format(imageI_paths))
222 # print("len(imageI_paths): {}".format(len(imageI_paths)))
223 ##### ...
224 if mixed_precisionChck == 1:
225     # TensorFlow 2.5.0
226     from tensorflow.keras.mixed_precision import experimental as mixed_precision
227     # # TensorFlow 2.10.1
228     # tf.keras.mixed_precision.set_global_policy('mixed_float16') # NOK on Jetson
229     print("tensorflow.keras.mixed_precision is ON")
230 elif mixed_precisionChck == 0:
231     print("tensorflow.keras.mixed_precision is OFF")
232 ##### ...
233 > def parse_arguments(): ...
259 ##### ...
260 > def extract_between_pairs_string(input_string, start_str, end_str): ...
270 ##### ...
271 > def load_and_crop_image_singlefolder(patch_size_start, patch_size, root, overlap): ...
419 ##### ...
420 > def aug_image(imagedata, maskdata): ...
450 ##### ...
451 > def categorical_labels(maskdata): ...
522 ##### ...
523 > def images_masks_checks(imagedata, maskdata, labels, n_classes, lds): ...
573 ##### ...
574 > def mosaic_images_masks_checks(imagedata, maskdata, n_classes, prfx): ...
605 ##### ...
606 > def one_hot_encoded(labels, n_classes, imagedata, image_number): ...
633 ##### ...

```

Figura 78 – Overview of the developed framework SMAXX\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 3/8).

```

❸ SMAxx_10_BCKns.py > ...
633 ##### ...
634 > """ ...
638 > def split_loaded_dataNPY(imagedata, labels, labels_one_hot, TestSplitt, TestSplittF, n_classes): ...
669 ##### ...
670 > """ ...
673 > def split_loaded_dataSKL(imagedata, labels, labels_one_hot, TestSplitt, TestSplittF, n_classes): ...
695 ##### ...
696 > """ ...
700 > def split_loaded_dataRDM(imagedata, labels, labels_one_hot, TestSplitt, TestSplittF, n_classes): ...
738 ##### ...
739 > def metrics_and_losses(n_classes, LR, metricsCheck): ...
770 ##### ...
771 > def unet(n_classes, patch_size, img_channels): ...
834 ##### ...
835 > def unet_norma(n_classes, patch_size, img_channels): ...
907 ##### ...
908 > def mlp_nn(n_classes, patch_size, img_channels): ...
941 ##### ...
942 # Define the atrous convolution layer function
943 > def atrous_conv_layer_FCN(inputs, filters, kernel_size, rate): ...
946 ##### ...
947 # Fully Convolutional Network (FCN) architecture with atrous convolutions
948 > def FCN_atrous(num_classes,patch_size,img_channels): ...
974 ##### ...
975 # Define your atrous convolution layer
976 > def atrous_conv_layer_IFCN(input_layer, filters, kernel_size, dilation_rate): ...
979 ##### ...
980 # Instance Fully Convolutional Network (IFCN) architecture with atrous convolutions
981 > def IFCN_atrous(num_classes,patch_size,img_channels,num_instances): ...
1009 ##### ...
1010 > def deep_lab(num_classes,patch_size,img_channels): ...
1039 ##### ...
1040 > def crop_image_gen(rgb_img,model,patch_size): ...
1106 ##### ...
1107 > def crop_image_ens(rgb_img,model1,model2,model3,opt_weights,patch_size,patch_size_start): ...
1183 ##### ...
1184 > def crop_image_leg(rgb_img,model,patch_size,patch_size_start): ...
1304 ##### ...
1305 > def crop_image_ens_leg(rgb_img,model1,model2,model3,opt_weights,patch_size,patch_size_start): ...
1435 ##### ...
1436 > def print_crop_image_leg(img,predicted_arr_leg,unique_values_leg,class_names_leg,iprt,FMode): ...
1712 ##### ...
1713 > def mt(model, Lds, optim, total_loss, metricsIOUFS, metricsJac, metricsTrain, metricsCheck, ...
2127 ##### ...
2128 > def mont(model, Lds, optim, total_loss, metricsIOUFS, metricsJac, metricsTrain, metricsCheck, ...
2618 ##### ...
2619 > def mt_mlp(model, Lds, optim, total_loss, metricsIOUFS, metricsJac, metricsTrain, metricsCheck, ...
3036 ##### ...
3037 > def mi(model_path, rootModCheck, patch_size, n_classes, total_loss, ...
3198 ##### ...
3199 > def mtb(backbone_input, Lds, optim, total, metricsIOUFS, metricsJac, metricsTrain, metricsCheck, ...
3688 ##### ...
3689 > def mien(Lds, optim, total, metricsIOUFS, metricsJac, metricsTrain, metricsCheck, ...
3921 ##### ...
3922 > def mt_mode(modorun): ...

```

Figura 79 – Overview of the developed framework SMAxx\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 4/8).

```

❸ SMAxx_10_BCKns.py > ...
3987 ##### ...
3988 > def mti_mode(modorun): ...
4055 ##### ...
4056 > def mi_mode(modorun): ...
4084 ##### ...
4085 > def mta_mode(modorun): ...
4153 ##### ...
4154 > def mtia_mode(modorun): ...
4222 ##### ...
4223 > def mia_mode(modorun): ...
4254 ##### ...
4255 > def mtb_mode(modorun): ...
4283 ##### ...
4284 > def mtib_mode(modorun): ...
4312 ##### ...
4313 > def mien_mode(modorun): ...
4340 ##### ...
4341 > def mtab_mode(modorun): ...
4372 ##### ...
4373 > def mtiab_mode(modorun): ...
4404 ##### ...
4405 > def miaen_mode(modorun): ...
4435 ##### ...
4436 def mainIN(modorun):
4437     print("mainIN " + modorun)
4438     if asinputChck == 1:
4439         print("1x")
4440         if asinputChckIN == 1:
4441             print("11")
4442
4443         elif asinputChckIN == 0:
4444             print("10")
4445 >         if modorun == "mt":
4446
4447             elif modorun == "mti": ...
4452
4453             elif modorun == "mi": ...
4456
4457             elif modorun == "mta": ...
4460
4461             elif modorun == "mtia": ...
4464
4465             elif modorun == "mia": ...
4468
4469             elif modorun == "mtb": ...
4472
4473             elif modorun == "mtib": ...
4476
4477             elif modorun == "mien": ...
4480
4481             elif modorun == "mtab": ...
4484
4485             elif modorun == "mtiab": ...
4488
4489             elif modorun == "miaen": ...
4492
    return

```

Figura 80 – Overview of the developed framework SMAxx\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 5/8).

```

❷ SMAxx_10_BCKns.py > ...
4436 def mainIN(modorun):
4437     print("mainIN__" + modorun)
4438     if asinputChck == 1:
4439         print("1x__")
4440         if asinputChckIN == 1:
4441             print("11__")
4442
4443     elif asinputChckIN == 0:
4444         print("10__")
4445         if modorun == "mt":
4446             ...
4447
4448         elif modorun == "mti": ...
4449
4450         elif modorun == "mi": ...
4451
4452         elif modorun == "mta": ...
4453
4454         elif modorun == "mtia": ...
4455
4456         elif modorun == "mia": ...
4457
4458         elif modorun == "mtb": ...
4459
4460         elif modorun == "mtib": ...
4461
4462         elif modorun == "mien": ...
4463
4464         elif modorun == "mtab": ...
4465
4466         elif modorun == "mtiab": ...
4467
4468         elif modorun == "miaen": ...
4469
4470     return
4471 #####
4472
4473 def main():
4474     if asinputChck == 1: ...
4475
4476     elif asinputChck == 0: ...
4477 #####
4478
4479 if __name__ == "__main__":
4480     if asinputChck == 1:
4481         if asinputChckIN == 1:
4482             main()
4483         elif asinputChckIN == 0:
4484             mainIN()
4485     elif asinputChck == 0:
4486         main()
4487 #####

```

Figura 81 – Overview of the developed framework SMAxx\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 6/8).

```

SMAXx_10_BCKns.py > ...
4495 ##### 
4496 def main():
4497     if asinputChck == 1:
4498         print("main__")
4499         if asinputChckIN == 1:
4500             print("main__1x__")
4501             modorun = input("Enter mode (mt for training, mti for training and inference, mi for inference): ")
4502             if modorun == "mt": ...
4503
4504             elif modorun == "mti": ...
4505
4506             elif modorun == "mi": ...
4507
4508             elif modorun == "mta": ...
4509
4510             elif modorun == "mtia": ...
4511
4512             elif modorun == "mia": ...
4513
4514             elif modorun == "mtb": ...
4515
4516             elif modorun == "mtib": ...
4517
4518             elif modorun == "mien": ...
4519
4520             elif modorun == "mtab": ...
4521
4522             elif modorun == "mtiab": ...
4523
4524             elif modorun == "miaen": ...
4525
4526
4527             elif asinputChckIN == 0:
4528                 print("main__10__")
4529
4530
4531             elif asinputChck == 0:
4532                 print("main__0x__")
4533                 parser = argparse.ArgumentParser(description="U-Net Patch Processing")
4534                 parser.add_argument("modorun", choices=["mt", "mti", "mi", "mta", "mtia", "mia", "mtb", "mtib", "mien", "mtab", "mtiab", "miaen"], help="Select modorun: train or inference")
4535                 argsS = parser.parse_args()
4536                 if argsS.modorun == "mt": ...
4537
4538                 elif argsS.modorun == "mti": ...
4539
4540                 elif argsS.modorun == "mi": ...
4541
4542                 elif argsS.modorun == "mta": ...
4543
4544                 elif argsS.modorun == "mtia": ...
4545
4546                 elif argsS.modorun == "mia": ...
4547
4548                 elif argsS.modorun == "mtb": ...
4549
4550                 elif argsS.modorun == "mtib": ...
4551
4552                 elif argsS.modorun == "mien": ...
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588

```

Figura 82 – Overview of the developed framework SMAXx\_10\_BCKns.py. Source: Robson Rogério Dutra Pereira. (Part 7/8).

```

❸ SMAxx_10_BCKns.py > ...
4520 >     elif modorun == "mia":...
4523
4524 >     elif modorun == "mtb":...
4527
4528 >     elif modorun == "mtib":...
4531
4532 >     elif modorun == "mien":...
4535
4536 >     elif modorun == "mtab":...
4539
4540 >     elif modorun == "mtiab":...
4543
4544 >     elif modorun == "miaen":...
4547
4548     elif asinputChckIN == 0:
4549         print("main__10__")
4550
4551     elif asinputChck == 0:
4552         print("main__0x__")
4553     parser = argparse.ArgumentParser(description="U-Net Patch Processing")
4554     parser.add_argument("modorun", choices=["mt", "mti", "mi", "mta", "mtia", "mia", "mtb", "mtib", "mien", "mtab", "mtiab", "miaen"], help="Select modorun: train or inference")
4555     argsS = parser.parse_args()
4556     if argsS.modorun == "mt":...
4559
4560 >     elif argsS.modorun == "mti":...
4563
4564 >     elif argsS.modorun == "mi":...
4567
4568 >     elif argsS.modorun == "mta":...
4571
4572 >     elif argsS.modorun == "mtia":...
4575
4576 >     elif argsS.modorun == "mia":...
4579
4580 >     elif argsS.modorun == "mtb":...
4583
4584 >     elif argsS.modorun == "mtib":...
4587
4588 >     elif argsS.modorun == "mien":...
4591
4592 >     elif argsS.modorun == "mtab":...
4595
4596 >     elif argsS.modorun == "mtiab":...
4599
4600 >     elif argsS.modorun == "miaen":...
4603 #####if __name__ == "__main__":
4604     if __name__ == "__main__":
4605         if asinputChck == 1:
4606             if asinputChckIN == 1:
4607                 main()
4608             elif asinputChckIN == 0:
4609                 mainIN()
4610         elif asinputChck == 0:
4611             main()
4612 #####

```

Figura 83 – Overview of the developed framework SMAxx\_10\_BCKns.py. Source: Robinson Rogério Dutra Pereira. (Part 8/8).

```

❶ SMAxx_10_BCKns_Main.py > ...
1 ##########
2 """
3 SMAxx_10_BCKns_Main.py
4
5 Created by: Robson Rogério Dutra Pereira on 01.Sep.2023
6 Last Modified: rrdpereira
7
8 Description: Remote call the SMAxx_10_BCKns.py.
9
10 E-mail: robsondutra.pereira@outlook.com
11 """
12 #####
13 import SMAxx_10_BCKns
14 import gc
15 import time, sys
16 import tensorflow as tf
17 from keras import backend as K
18 import importlib
19
20 # List of backbones_in
21 > backbones_in = [...]
22
23 print("len(backbones_in): {}".format(len(backbones_in)))
24 # List of models_inib
25 > models_inib = [ # Comment non used models to run "One Saved Model"...
26     print("len(models_inib): {}".format(len(models_inib)))
27     # models_inib_loop = ["model_Unet_densenet121_00","model_Unet_densenet201_00","model_Unet_efficientnetb5_00","model_Unet_inceptionv3_00","model_Unet_resnet152_00","model_U...
28     # # 0 1 2 3 4 5
29     # models_inib_loop = [ # Comment non used models to run "One Saved Model"; Yellow = YL; Purple = PU; Blue = BL; Green = GR; Red = RD; Light Blue = LB 2023/12/14 22h50min ...
30     print("len(models_inib_loop): {}".format(len(models_inib_loop)))
31     BACKBONES_FILES_FOR_LOOP_CHK = 0 # 1:ON and 0:OFF
32     BACKBONES_FILES_FOR_LOOP_NUM = 1 # Number of times to run the modules of BACKBONES FILES
33     H5MODELS_FILES_FOR_LOOP_CHK = 0 # 1:ON and 0:OFF
34     H5MODELS_FILES_FOR_LOOP_NUM = 1 # Number of times to run the modules of BACKBONES FILES
35     SMABCK_FILES_FOR_LOOP_CHK = 1 # 1:ON and 0:OFF
36     SMABCK_FILES_FOR_LOOP_NUM = 1 # Number of times to run the modules of SMAxx_10_BCKns.py file
37     # Define a function to run each module
38     > def run_module(module, argument, mt_mont_ChckM, UNetNormaChckM, LRVAR_M, patch_sizeVARM, overlapVARPercM, patience_VVARM, batch_size_VVARM, epochs_VVARM, activationVARM):...
39
40     > if BACKBONES_FILES_FOR_LOOP_CHK == 1:...
41
42     > elif BACKBONES_FILES_FOR_LOOP_CHK == 0:...
43
44     > if H5MODELS_FILES_FOR_LOOP_CHK == 1:...
45
46     > elif H5MODELS_FILES_FOR_LOOP_CHK == 0:...
47
48     # List of modules to run
49     > moduleU = [...]
50
51     > if SMABCK_FILES_FOR_LOOP_CHK == 1:...
52
53     > elif SMABCK_FILES_FOR_LOOP_CHK == 0:...

```

Figura 84 – Overview of the developed remote code SMAxx\_10\_BCKns\_Main.py.

Source: Robson Rogério Dutra Pereira. (Part 1/2).

```

❸ SMAxx_10_BCKns_Main.py > ...
96     def run_module(module, argument, mt_mont_CchkM, UNetNormaChckM, LRVAR_M, patch_sizeVARM, overlapVARPercM, patience_VVARM, batch_size_VVARM, epochs_VVARM, activationVARM
97         # Your module-specific configuration here
98         module.PLTShowChck = 0 # plt.show() ---> 0: OFF; 1: ON
99         module.PLTShowTempPause = 2 # plt.show(temporary pause) ---> 0: OFF; 1: ON; 2: PNG inference paused way
100        module.TESTImgNumberChck = 1 # 0: random; 1: fixed
101        module.TESTImgNumberVAR = 338
102        module.UNetNormaChck = UNetNormaChckM # UNet with BatchNormalization() ---> 0: OFF; 1: ON | 0: NBatchNorm; 1: YBatchNorm; 2: mlp_nn; 3: FCN_atrous; 4: IFCON_atrous; 5: de
103        module.metricsChck = 1 # For the train ---> 1: metricsIOUFS and 2: metricsJac
104        module.UshuffleChck = True # For the model.fit shuffle ---> True: ON and False: OFF
105        module.mixed_precisionChck = 0 # For the tensorflow.keras.mixed_precision ---> 0: off and 1: on
106        module.date_timeChck = 1 # For the DATETIME on the Model Checkpoint ---> 0: off and 1: on
107        module.split_loaded_dataChck = 1 # split_loaded_data Functions ---> 1:numpy.split; 2:sklearn.train_test_split; 3:random.sample
108        module.rootModChck = 1 # For the load saved models ---> 1: metricsIOUFS and 2: metricsJac
109        module.mt_mont_Cchk = mt_mont_CchkM # For UNet Loss check ---> 1: mt function, 2: mont function and 3: mlp_nn
110        module.patch_size_startVAR = 0 # Start Patch size
111        module.patch_sizeVAR = patch_sizeVARM # Patch size: 128 or 256; 4:PSPNet (384 x 384, 3)
112        module.overlapVARPerc = overlapVARPercM # Overlap between patches in % of patch_sizeVAR
113        module.overlapVAR = round((module.patch_sizeVAR*module.overlapVARPerc)/100) # Overlap between patches in pixels
114        module.patience_VVAR = patience_VVARM # Patience for Early Stopping in epochs
115        module.batch_size_VVAR = batch_size_VVARM # Batch size [4, 8, 16, 32, 64]
116        module.epochs_VVAR = epochs_VVARM # Epochs size
117        module.activationVAR = activationVARM # "softmax" # Activation of net output; "softmax":multiclass segmentation with non overlapping class masks (your classes + background)
118        module.LRVAR = LRVAR_M # 0.001 # Learning Rate
119        module.restart_loss_thresholdVAR = 0.9999 # Define a threshold for loss to trigger a restart
120        module.max_consecutive_epochsVAR = 7 # Define a maximum number of consecutive epochs with loss above threshold before restarting
121        module.consecutive_epochsVAR = 0
122        module.max_restart_attemptsVAR = 5 # Set your desired limit here
123        module.restart_attemptsVAR = 0
124        module.optimizerUChck = 1 # UNet Model compiler optimizer ---> 1:optim = Adam(LR); 2:'adam'
125        module.optimizerBChck = 1 # UNet + Backbone Model compiler optimizer ---> 1:optim = Adam(LR); 2:'adam'
126        module.TestSplittVAR = 0.10 # Test_size split for normal training
127        module.TestSplittFVAR = 0.50 # Test_size split for fast training
128        # module.backbones_index = 0 # For the "SMAxx_10_BCKns_Creator.py" needs to be "0"(zero) and COMMENT this line for BACKBONES TRAINING
129        # module.backbone_input = backbones_in[module.backbones_index]
130        module.inib_index = 0 # For the "SMAxx_10_BCKns_ibCreator.py" needs to be "0"(zero) and COMMENT this line for BACKBONES TRAINING
131        module.models_inibpt = models_inib[module.inib_index]
132        module.high_resolution_imgChckEACHI = 1 # Inference in a high resolution on each images ---> 0:off; 1:on
133        module.high_resolution_imgChckHOALL = 0 # Inference in a high resolution on hold all the subplots images ---> 0:off; 1:on
134        module.modelnetChck = 1 # Architectures Model Net ---> 1:Unet; 2:FPN; 3:Linknet; 4:PSPNet (384 x 384, 3)
135        module.encoderweightsChck = 1 # encoder_weights of the Model ---> 1:encoder_weights='imagenet'; 2:encoder_weights=None
136        module.encoderfreezeChck = 0 # to only randomly initialized decoder in order not to damage weights of properly trained encoder ---> 0:encoder_freeze OFF; 1:encoder_f
137        module.asinputChck = 1 # MAIN asinput ---> 0:off; 1:on; "asinputChck:1|asinputChckIN:0" just remote file; "asinputChck:1|asinputChckIN:1" and "asinputChck:0|asinputC
138        module.asinputChckIN = 0 # MAIN asinput terminal ---> 0:off; 1:on; "asinputChck:1|asinputChckIN:0" just remote file; "asinputChck:1|asinputChckIN:1" and "asinputChck:0|asinputC
139        module.prebckb_input1 = backbones_in[0] # According to backbones_in:model_Unet_efficientnetb5_00[29]
140        module.prebckb_input2 = backbones_in[0] # According to backbones_in:model_Unet_resnet152_00[13]
141        module.prebckb_input3 = backbones_in[0] # According to backbones_in:model_Unet_vgg16_00[7]
142        module.bckb_model01 = models_inib[2] # [2] model_Unet_efficientnetb5_00; models_inib[0] with zero index to run "One Saved Model"
143        module.bckb_model02 = models_inib[4] # [4] model_Unet_resnet152_00; models_inib[0] with zero index to run "One Saved Model"
144        module.bckb_model03 = models_inib[7] # [0] model_Unet_densenet121_00 OR [1] model_Unet_densenet201_00 OR [8] model_Unet_vgg16_00 OR [9] model_Unet_00; models_inib[0]
145        module.ClassNumInf = 2
146        # Run mainIN(argument) of SMAxx_10_BCKns.py
147        module.mainIN(argument)
148        # Delays and clean commands
149        time.sleep(5)
150        gc.collect()
151        tf.keras.backend.clear_session()

```

Figura 85 – Overview of the developed remote code SMAxx\_10\_BCKns\_Main.py.

Source: Robson Rogério Dutra Pereira. (Part 2/2).