

## SIM Denoising Pipeline

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Packages	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 analyse Namespace Reference	9
5.1.1 Variable Documentation	10
5.1.1.1 action	10
5.1.1.2 args	10
5.1.1.3 ckpt	10
5.1.1.4 cmap	10
5.1.1.5 default	10
5.1.1.6 device	10
5.1.1.7 df	11
5.1.1.8 exist_ok	11
5.1.1.9 gt	11
5.1.1.10 gt_dir	11
5.1.1.11 gt_files	11
5.1.1.12 gt_samples	11
5.1.1.13 img_idx	12
5.1.1.14 int	12
5.1.1.15 model	12
5.1.1.16 model_1	12
5.1.1.17 model_1_dir	12
5.1.1.18 model_1_files	12
5.1.1.19 model_1_samples	12
5.1.1.20 model_2	13
5.1.1.21 model_2_dir	13
5.1.1.22 model_2_files	13
5.1.1.23 model_2_samples	13
5.1.1.24 N	13
5.1.1.25 output_dir	13
5.1.1.26 parents	13
5.1.1.27 parser	14
5.1.1.28 psnr	14
5.1.1.29 raw	14

5.1.1.30 raw_dir	14
5.1.1.31 raw_files	14
5.1.1.32 raw_samples	14
5.1.1.33 required	14
5.1.1.34 rng	14
5.1.1.35 ssim	15
5.1.1.36 str	15
5.1.1.37 True	15
5.1.1.38 type	15
5.2 apply Namespace Reference	15
5.2.1 Variable Documentation	16
5.2.1.1 action	16
5.2.1.2 args	16
5.2.1.3 choices	16
5.2.1.4 ckpt	16
5.2.1.5 data	16
5.2.1.6 default	16
5.2.1.7 device	16
5.2.1.8 imagej	17
5.2.1.9 input_path	17
5.2.1.10 int	17
5.2.1.11 model	17
5.2.1.12 output_file	17
5.2.1.13 output_path	17
5.2.1.14 overlap_shape	17
5.2.1.15 parents	18
5.2.1.16 parser	18
5.2.1.17 percentile	18
5.2.1.18 raw	18
5.2.1.19 raw_files	18
5.2.1.20 RCAN_hyperparameters	18
5.2.1.21 required	18
5.2.1.22 restored	19
5.2.1.23 str	19
5.2.1.24 type	19
5.3 convert_omx_to_czxy Namespace Reference	19
5.3.1 Variable Documentation	19
5.3.1.1 action	20
5.3.1.2 args	20
5.3.1.3 converted	20
5.3.1.4 imagej	20
5.3.1.5 input_dir	20

5.3.1.6 input_files . . . . .	20
5.3.1.7 int . . . . .	20
5.3.1.8 original . . . . .	21
5.3.1.9 parser . . . . .	21
5.3.1.10 required . . . . .	21
5.3.1.11 str . . . . .	21
5.3.1.12 type . . . . .	21
5.4 convert_omx_to_paz Namespace Reference . . . . .	21
5.4.1 Variable Documentation . . . . .	22
5.4.1.1 action . . . . .	22
5.4.1.2 args . . . . .	22
5.4.1.3 converted . . . . .	22
5.4.1.4 imagej . . . . .	22
5.4.1.5 input_dir . . . . .	22
5.4.1.6 input_files . . . . .	22
5.4.1.7 int . . . . .	22
5.4.1.8 original . . . . .	23
5.4.1.9 parser . . . . .	23
5.4.1.10 required . . . . .	23
5.4.1.11 str . . . . .	23
5.4.1.12 type . . . . .	23
5.5 convert_slices_to_volumes Namespace Reference . . . . .	23
5.5.1 Variable Documentation . . . . .	24
5.5.1.1 args . . . . .	24
5.5.1.2 default . . . . .	24
5.5.1.3 exist_ok . . . . .	24
5.5.1.4 imagej . . . . .	24
5.5.1.5 input_dir . . . . .	24
5.5.1.6 input_files . . . . .	24
5.5.1.7 input_slice . . . . .	24
5.5.1.8 output_dir . . . . .	25
5.5.1.9 output_file . . . . .	25
5.5.1.10 parents . . . . .	25
5.5.1.11 parser . . . . .	25
5.5.1.12 required . . . . .	25
5.5.1.13 str . . . . .	25
5.5.1.14 subvolume . . . . .	25
5.5.1.15 True . . . . .	25
5.5.1.16 tuple_of_ints . . . . .	26
5.5.1.17 type . . . . .	26
5.5.1.18 volume . . . . .	26
5.6 generate_sim Namespace Reference . . . . .	26

5.6.1 Function Documentation	26
5.6.1.1 <code>arange_zero()</code>	27
5.6.1.2 <code>threshold_norm()</code>	27
5.6.2 Variable Documentation	27
5.6.2.1 <code>args</code>	27
5.6.2.2 <code>default</code>	27
5.6.2.3 <code>int</code>	27
5.6.2.4 <code>parser</code>	27
5.6.2.5 <code>required</code>	28
5.6.2.6 <code>runner</code>	28
5.6.2.7 <code>str</code>	28
5.6.2.8 <code>type</code>	28
5.7 <code>image_noising</code> Namespace Reference	28
5.7.1 Function Documentation	29
5.7.1.1 <code>save_image_pair()</code>	29
5.7.2 Variable Documentation	29
5.7.2.1 <code>args</code>	29
5.7.2.2 <code>choices</code>	29
5.7.2.3 <code>data</code>	30
5.7.2.4 <code>default</code>	30
5.7.2.5 <code>float</code>	30
5.7.2.6 <code>gt</code>	30
5.7.2.7 <code>img_idx_all</code>	30
5.7.2.8 <code>img_idx_test</code>	30
5.7.2.9 <code>img_idx_train</code>	30
5.7.2.10 <code>img_idx_val</code>	30
5.7.2.11 <code>input_path</code>	31
5.7.2.12 <code>int</code>	31
5.7.2.13 <code>n_acquisitions</code>	31
5.7.2.14 <code>n_img</code>	31
5.7.2.15 <code>output_path</code>	31
5.7.2.16 <code>output_test_gt_path</code>	31
5.7.2.17 <code>output_test_raw_path</code>	31
5.7.2.18 <code>output_train_gt_path</code>	31
5.7.2.19 <code>output_train_raw_path</code>	32
5.7.2.20 <code>output_val_gt_path</code>	32
5.7.2.21 <code>output_val_raw_path</code>	32
5.7.2.22 <code>parents</code>	32
5.7.2.23 <code>parser</code>	32
5.7.2.24 <code>required</code>	32
5.7.2.25 <code>rng</code>	32
5.7.2.26 <code>split</code>	32

5.7.2.27 str	33
5.7.2.28 train_size	33
5.7.2.29 type	33
5.7.2.30 val_size	33
5.8 manage_stack Namespace Reference	33
5.8.1 Variable Documentation	34
5.8.1.1 action	34
5.8.1.2 args	34
5.8.1.3 choices	34
5.8.1.4 default	34
5.8.1.5 exist_ok	34
5.8.1.6 filename	34
5.8.1.7 files	35
5.8.1.8 img_data	35
5.8.1.9 int	35
5.8.1.10 number_of_stacks	35
5.8.1.11 output_data	35
5.8.1.12 output_dir	35
5.8.1.13 output_file	35
5.8.1.14 parents	36
5.8.1.15 parser	36
5.8.1.16 required	36
5.8.1.17 sample	36
5.8.1.18 stack_handler	36
5.8.1.19 stack_number	36
5.8.1.20 str	36
5.8.1.21 True	37
5.8.1.22 type	37
5.9 rcan Namespace Reference	37
5.10 rcan.data_generator Namespace Reference	37
5.10.1 Function Documentation	37
5.10.1.1 load_SIM_dataset()	37
5.11 rcan.data_processing Namespace Reference	38
5.11.1 Function Documentation	39
5.11.1.1 conv_czxy_to_omx()	39
5.11.1.2 conv_omx_to_czxy()	39
5.11.1.3 conv_omx_to_paz()	40
5.11.1.4 conv_paz_to_omx()	40
5.11.1.5 crop_volume()	40
5.12 rcan.model Namespace Reference	42
5.12.1 Function Documentation	42
5.12.1.1 _conv()	42

5.12.1.2 <code>_destandardize()</code>	43
5.12.1.3 <code>_global_average_pooling()</code>	43
5.12.1.4 <code>_standardize()</code>	44
5.13 <code>rcan.plotting</code> Namespace Reference	44
5.13.1 Function Documentation	44
5.13.1.1 <code>compute_metrics()</code>	44
5.13.1.2 <code>plot_learning_curve()</code>	45
5.13.1.3 <code>plot_reconstructions()</code>	45
5.14 <code>rcan.utils</code> Namespace Reference	46
5.14.1 Function Documentation	46
5.14.1.1 <code>apply()</code>	46
5.14.1.2 <code>load_rcan_checkpoint()</code>	47
5.14.1.3 <code>normalize()</code>	47
5.14.1.4 References	48
5.14.1.5 <code>normalize_between_zero_and_one()</code>	48
5.14.1.6 <code>percentile()</code>	48
5.14.1.7 <code>reshape_to_bcwh()</code>	48
5.14.1.8 <code>tuple_of_ints()</code>	49
5.15 <code>recon_postprocess</code> Namespace Reference	49
5.15.1 Variable Documentation	49
5.15.1.1 <code>args</code>	49
5.15.1.2 <code>files</code>	49
5.15.1.3 <code>img_data</code>	50
5.15.1.4 <code>parser</code>	50
5.15.1.5 <code>required</code>	50
5.15.1.6 <code>str</code>	50
5.15.1.7 <code>type</code>	50
5.16 <code>recon_preprocess</code> Namespace Reference	50
5.16.1 Function Documentation	51
5.16.1.1 <code>normalize_acquisition_intensity()</code>	51
5.16.2 Variable Documentation	51
5.16.2.1 <code>action</code>	51
5.16.2.2 <code>args</code>	51
5.16.2.3 <code>choices</code>	51
5.16.2.4 <code>default</code>	51
5.16.2.5 <code>exist_ok</code>	51
5.16.2.6 <code>files</code>	52
5.16.2.7 <code>img_data</code>	52
5.16.2.8 <code>int</code>	52
5.16.2.9 <code>output_dir</code>	52
5.16.2.10 <code>output_file</code>	52
5.16.2.11 <code>parents</code>	52



5.16.2.12 parser	52
5.16.2.13 percentile	52
5.16.2.14 required	53
5.16.2.15 str	53
5.16.2.16 True	53
5.16.2.17 type	53
5.17 stats Namespace Reference	53
5.17.1 Function Documentation	54
5.17.1.1 paired_t()	54
5.17.2 Variable Documentation	54
5.17.2.1 alpha	55
5.17.2.2 args	55
5.17.2.3 ax	55
5.17.2.4 choices	55
5.17.2.5 color	55
5.17.2.6 data	55
5.17.2.7 default	55
5.17.2.8 df	56
5.17.2.9 dflong	56
5.17.2.10 dflongssim	56
5.17.2.11 dodge	56
5.17.2.12 exist_ok	56
5.17.2.13 fig	57
5.17.2.14 figsize	57
5.17.2.15 hist_range_psnr	57
5.17.2.16 hist_range_ssim	57
5.17.2.17 hue	57
5.17.2.18 int	57
5.17.2.19 legend	58
5.17.2.20 lw	58
5.17.2.21 mean_psnr_1	58
5.17.2.22 mean_psnr_2	58
5.17.2.23 mean_ssim_1	58
5.17.2.24 mean_ssim_2	58
5.17.2.25 output_dir	58
5.17.2.26 palette	59
5.17.2.27 parents	59
5.17.2.28 parser	59
5.17.2.29 psnr_cols	59
5.17.2.30 psnr_diff_1_max	59
5.17.2.31 psnr_diff_1_min	59
5.17.2.32 psnr_diff_2_max	60

5.17.2.33 psnr_diff_2_min	60
5.17.2.34 range	60
5.17.2.35 required	60
5.17.2.36 se_psnr_1	60
5.17.2.37 se_psnr_2	60
5.17.2.38 se_ssim_1	61
5.17.2.39 se_ssim_2	61
5.17.2.40 ssim_cols	61
5.17.2.41 ssim_diff_1_max	61
5.17.2.42 ssim_diff_1_min	61
5.17.2.43 ssim_diff_2_max	62
5.17.2.44 ssim_diff_2_min	62
5.17.2.45 str	62
5.17.2.46 title	62
5.17.2.47 True	62
5.17.2.48 type	62
5.17.2.49 x	62
5.17.2.50 xlabel	63
5.17.2.51 y	63
5.18 synthetic_sim Namespace Reference	63
5.19 synthetic_sim.otf Namespace Reference	63
5.19.1 Function Documentation	63
5.19.1.1 calc_psf()	63
5.20 train Namespace Reference	64
5.20.1 Function Documentation	64
5.20.1.1 load_data_paths()	65
5.20.1.2 train()	65
5.20.2 Variable Documentation	65
5.20.2.1 args	65
5.20.2.2 ckpt	65
5.20.2.3 ckpt_path	65
5.20.2.4 config	66
5.20.2.5 device	66
5.20.2.6 exist_ok	66
5.20.2.7 input_shape	66
5.20.2.8 losses_train_epoch	66
5.20.2.9 losses_val_epoch	66
5.20.2.10 model	66
5.20.2.11 n_accumulations	67
5.20.2.12 ndim	67
5.20.2.13 nepoch	67
5.20.2.14 optimizer	67

5.20.2.15 output_dir . . . . .	67
5.20.2.16 parents . . . . .	67
5.20.2.17 parser . . . . .	67
5.20.2.18 psnr_train_epoch . . . . .	68
5.20.2.19 psnr_val_epoch . . . . .	68
5.20.2.20 RCAN_hyperparameters . . . . .	68
5.20.2.21 required . . . . .	68
5.20.2.22 saveinterval . . . . .	68
5.20.2.23 scheduler . . . . .	68
5.20.2.24 schema . . . . .	69
5.20.2.25 ssim_train_epoch . . . . .	69
5.20.2.26 ssim_val_epoch . . . . .	69
5.20.2.27 start_epoch . . . . .	69
5.20.2.28 str . . . . .	69
5.20.2.29 train_loader . . . . .	69
5.20.2.30 True . . . . .	70
5.20.2.31 type . . . . .	70
5.20.2.32 val_loader . . . . .	70
<b>6 Class Documentation</b>	<b>71</b>
6.1 rcnn.model._channel_attention_block Class Reference . . . . .	71
6.1.1 Detailed Description . . . . .	72
6.1.1.1 References . . . . .	72
6.1.2 Constructor & Destructor Documentation . . . . .	72
6.1.2.1 __init__() . . . . .	72
6.1.3 Member Function Documentation . . . . .	73
6.1.3.1 forward() . . . . .	73
6.1.4 Member Data Documentation . . . . .	73
6.1.4.1 conv_1 . . . . .	73
6.1.4.2 conv_2 . . . . .	73
6.1.4.3 global_average_pooling . . . . .	73
6.2 rcnn.model._residual_channel_attention_blocks Class Reference . . . . .	74
6.2.1 Detailed Description . . . . .	75
6.2.1.1 References . . . . .	75
6.2.2 Constructor & Destructor Documentation . . . . .	75
6.2.2.1 __init__() . . . . .	75
6.2.3 Member Function Documentation . . . . .	75
6.2.3.1 forward() . . . . .	75
6.2.4 Member Data Documentation . . . . .	76
6.2.4.1 channel_attention_block_list . . . . .	76
6.2.4.2 conv_list . . . . .	76
6.2.4.3 repeat . . . . .	76

6.2.4.4 residual_scaling . . . . .	76
6.3 rcan.data_processing.ImageStack Class Reference . . . . .	76
6.3.1 Detailed Description . . . . .	77
6.3.2 Constructor & Destructor Documentation . . . . .	77
6.3.2.1 __init__() . . . . .	77
6.3.3 Member Function Documentation . . . . .	78
6.3.3.1 add_image() . . . . .	78
6.3.3.2 export_stack() . . . . .	78
6.3.4 Member Data Documentation . . . . .	78
6.3.4.1 dim . . . . .	78
6.3.4.2 n_acq . . . . .	78
6.3.4.3 n_z . . . . .	79
6.3.4.4 sample . . . . .	79
6.3.4.5 stack . . . . .	79
6.4 synthetic_sim.off.PsfParameters Class Reference . . . . .	79
6.4.1 Detailed Description . . . . .	79
6.4.2 Member Data Documentation . . . . .	79
6.4.2.1 Callable . . . . .	80
6.4.2.2 float . . . . .	80
6.4.2.3 int . . . . .	80
6.5 rcan.model.RCAN Class Reference . . . . .	80
6.5.1 Detailed Description . . . . .	81
6.5.1.1 References . . . . .	81
6.5.2 Constructor & Destructor Documentation . . . . .	81
6.5.2.1 __init__() . . . . .	81
6.5.3 Member Function Documentation . . . . .	82
6.5.3.1 forward() . . . . .	82
6.5.4 Member Data Documentation . . . . .	82
6.5.4.1 conv_input . . . . .	82
6.5.4.2 conv_list . . . . .	83
6.5.4.3 conv_output . . . . .	83
6.5.4.4 num_residual_groups . . . . .	83
6.5.4.5 rcab_list . . . . .	83
6.6 rcan.data_generator.SIM_Dataset Class Reference . . . . .	83
6.6.1 Detailed Description . . . . .	84
6.6.2 Constructor & Destructor Documentation . . . . .	85
6.6.2.1 __init__() . . . . .	85
6.6.3 Member Function Documentation . . . . .	85
6.6.3.1 __getitem__() . . . . .	85
6.6.3.2 __len__() . . . . .	86
6.6.3.3 _scale() . . . . .	86
6.6.4 Member Data Documentation . . . . .	86

6.6.4.1 <code>_area_threshold</code>	86
6.6.4.2 <code>_intensity_threshold</code>	86
6.6.4.3 <code>_scale_factor</code>	87
6.6.4.4 <code>_shape</code>	87
6.6.4.5 <code>_transform_function</code>	87
6.6.4.6 <code>_y</code>	87
6.6.4.7 <code>output_shape</code>	87
6.6.4.8 <code>output_signature</code>	87
6.6.4.9 <code>p_max</code>	87
6.6.4.10 <code>p_min</code>	87
6.6.4.11 <code>steps_per_epoch</code>	88
6.7 <code>generate_sim.SimulationRunner</code> Class Reference	88
6.7.1 Detailed Description	88
6.7.2 Constructor & Destructor Documentation	88
6.7.2.1 <code>__init__()</code>	88
6.7.3 Member Function Documentation	89
6.7.3.1 <code>do_sim()</code>	89
6.7.3.2 <code>run()</code>	89
6.7.4 Member Data Documentation	89
6.7.4.1 <code>input_dir</code>	89
6.7.4.2 <code>input_files</code>	89
6.7.4.3 <code>output_dir</code>	89
6.7.4.4 <code>range</code>	90
6.7.4.5 <code>z_offset</code>	90
6.8 <code>generate_sim.Simulator</code> Class Reference	90
6.8.1 Detailed Description	91
6.8.2 Constructor & Destructor Documentation	91
6.8.2.1 <code>__init__()</code>	91
6.8.3 Member Function Documentation	92
6.8.3.1 <code>add_noise()</code>	92
6.8.3.2 <code>illumination()</code>	92
6.8.3.3 <code>in_focus_plane()</code>	92
6.8.3.4 <code>params_dict()</code>	92
6.8.3.5 <code>psf()</code>	92
6.8.3.6 <code>psf_params()</code>	93
6.8.3.7 <code>randomise()</code>	93
6.8.3.8 <code>simulate_ideal_superres()</code>	93
6.8.3.9 <code>simulate_sim()</code>	93
6.8.3.10 <code>wavevectors()</code>	93
6.8.4 Member Data Documentation	93
6.8.4.1 <code>_illumination</code>	94
6.8.4.2 <code>_psf</code>	94

6.8.4.3 <a href="#">_superres_psf</a>	94
6.8.4.4 <a href="#">angle_error</a>	94
6.8.4.5 <a href="#">beam_position</a>	94
6.8.4.6 <a href="#">delta_z_p</a>	94
6.8.4.7 <a href="#">k0</a>	94
6.8.4.8 <a href="#">k_exc</a>	94
6.8.4.9 <a href="#">lambda0</a>	95
6.8.4.10 <a href="#">lambda_exc</a>	95
6.8.4.11 <a href="#">n_angles</a>	95
6.8.4.12 <a href="#">n_g</a>	95
6.8.4.13 <a href="#">n_i</a>	95
6.8.4.14 <a href="#">n_rotations</a>	95
6.8.4.15 <a href="#">n_sample</a>	95
6.8.4.16 <a href="#">n_shifts</a>	95
6.8.4.17 <a href="#">n_x</a>	96
6.8.4.18 <a href="#">n_z</a>	96
6.8.4.19 <a href="#">poisson_photons</a>	96
6.8.4.20 <a href="#">res_axial</a>	96
6.8.4.21 <a href="#">res_lateral</a>	96
6.8.4.22 <a href="#">signal_to_noise</a>	96
6.8.4.23 <a href="#">z</a>	96
6.8.4.24 <a href="#">z_p</a>	96
<b>7 File Documentation</b>	<b>97</b>
7.1 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/analyse.py File Reference</a>	97
7.1.1 Detailed Description	98
7.2 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/apply.py File Reference</a>	98
7.2.1 Detailed Description	99
7.3 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_czxy.py File Reference</a>	100
7.3.1 Detailed Description	100
7.4 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_paz.py File Reference</a>	100
7.4.1 Detailed Description	101
7.5 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/convert_slices_to_volumes.py File Reference</a>	101
7.5.1 Detailed Description	102
7.6 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/generate_sim.py File Reference</a>	102
7.6.1 Detailed Description	103
7.7 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/image_noising.py File Reference</a>	103
7.7.1 Detailed Description	104
7.8 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/manage_stack.py File Reference</a>	105
7.8.1 Detailed Description	105
7.9 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/rcan/__init__.py File Reference</a>	106
7.10 <a href="#">/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/__init__.py File Reference</a>	106

---

7.11 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/data_generator.py File Reference . . . . .	106
7.11.1 Detailed Description . . . . .	107
7.12 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/data_processing.py File Reference . . . . .	107
7.12.1 Detailed Description . . . . .	107
7.13 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py File Reference . . . . .	108
7.13.1 Detailed Description . . . . .	108
7.14 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/plotting.py File Reference . . . . .	108
7.14.1 Detailed Description . . . . .	109
7.15 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/utils.py File Reference . . . . .	109
7.15.1 Detailed Description . . . . .	110
7.16 /home/jhughes2712/projects/sim_project/jh2284/src/recon_postprocess.py File Reference . . . . .	110
7.16.1 Detailed Description . . . . .	110
7.17 /home/jhughes2712/projects/sim_project/jh2284/src/recon_preprocess.py File Reference . . . . .	110
7.17.1 Detailed Description . . . . .	111
7.18 /home/jhughes2712/projects/sim_project/jh2284/src/stats.py File Reference . . . . .	112
7.19 /home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/otf.py File Reference . . . . .	113
7.20 /home/jhughes2712/projects/sim_project/jh2284/src/train.py File Reference . . . . .	113
7.20.1 Detailed Description . . . . .	114
<b>Index</b>	<b>115</b>





# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">analyse</a>	9
<a href="#">apply</a>	15
<a href="#">convert_omx_to_czxy</a>	19
<a href="#">convert_omx_to_paz</a>	21
<a href="#">convert_slices_to_volumes</a>	23
<a href="#">generate_sim</a>	26
<a href="#">image_noising</a>	28
<a href="#">manage_stack</a>	33
<a href="#">rcan</a>	37
<a href="#">rcan.data_generator</a>	37
<a href="#">rcan.data_processing</a>	38
<a href="#">rcan.model</a>	42
<a href="#">rcan.plotting</a>	44
<a href="#">rcan.utils</a>	46
<a href="#">recon_postprocess</a>	49
<a href="#">recon_preprocess</a>	50
<a href="#">stats</a>	53
<a href="#">synthetic_sim</a>	63
<a href="#">synthetic_sim.otf</a>	63
<a href="#">train</a>	64



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

rcan.data_processing.ImageStack . . . . .	76
torch.nn.Module	
rcan.model.RCAN . . . . .	80
rcan.model._channel_attention_block . . . . .	71
rcan.model._residual_channel_attention_blocks . . . . .	74
synthetic_sim.otf.PsfParameters . . . . .	79
generate_sim.SimulationRunner . . . . .	88
generate_sim.Simulator . . . . .	90
Dataset	
rcan.data_generator.SIM_Dataset . . . . .	83



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">rcan.model._channel_attention_block</a>	
Implements channel attention block/layer . . . . .	71
<a href="#">rcan.model._residual_channel_attention_blocks</a>	
Implements residual group based on [1] . . . . .	74
<a href="#">rcan.data_processing.ImageStack</a>	
Handles creation and loading of image hyperstacks in order to make reconstructions using ImageJ easier . . . . .	76
<a href="#">synthetic_sim.otf.PsfParameters</a>	
Class to store PSF parameters . . . . .	79
<a href="#">rcan.model.RCAN</a>	
Builds a residual channel attention network . . . . .	80
<a href="#">rcan.data_generator.SIM_Dataset</a>	
Generates batches of images with real-time data augmentation . . . . .	83
<a href="#">generate_sim.SimulationRunner</a>	
Class which performs a batch of simulations, either sequentially or in parallel . . . . .	88
<a href="#">generate_sim.Simulator</a>	
The <a href="#">Simulator</a> class encapsulates the state of a 3D microscope simulation . . . . .	90



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">analyse.py</a>	
Script producing plots and small datasets that summarise the performance of models . . . . .	97
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">apply.py</a>	
Script producing restored images resulting from an RCAN denoiser being applied to low SNR images . . . . .	98
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">convert_omx_to_czxy.py</a>	
Script enabling .tif file conversion between OMX and CZXY . . . . .	100
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">convert_omx_to_paz.py</a>	
Script enabling .tif file conversion between OMX and PAZ . . . . .	100
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">convert_slices_to_volumes.py</a>	
Script enabling construction of 3D image volumes from large RGB 2D image slices . . . . .	101
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">generate_sim.py</a>	
Script simulating the acquisition of 3D SIM image volumes . . . . .	102
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">image_noising.py</a>	
Script which converts a directory of high-SNR SIM images into a training dataset . . . . .	103
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">manage_stack.py</a>	
Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions . . . . .	105
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">recon_postprocess.py</a>	
Script handling the postprocessing of SIM reconstructions . . . . .	110
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">recon_preprocess.py</a>	
Script handling the preprocessing of images before SIM reconstruction . . . . .	110
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">stats.py</a>	
. . . . .	112
/home/jhughes2712/projects/sim_project/jh2284/src/ <a href="#">train.py</a>	
Script used to train RCAN . . . . .	113
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">__init__.py</a>	
. . . . .	106
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">data_generator.py</a>	
Module that handles processing and batching of data during training loop . . . . .	106
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">data_processing.py</a>	
Contains tools used to pre-process image data . . . . .	107
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">model.py</a>	
Module defining the RCAN model architecture . . . . .	108
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">plotting.py</a>	
Module providing helper functions for matplotlib plots . . . . .	108
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ <a href="#">utils.py</a>	
Contains utility functions for the training loop and inference . . . . .	109
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/ <a href="#">__init__.py</a>	
. . . . .	106
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/ <a href="#">otf.py</a>	
. . . . .	113





## Chapter 5

# Namespace Documentation

### 5.1 analyse Namespace Reference

#### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `default`
- `int`
- `action`
- `args` = `parser.parse_args()`
- `output_dir` = `pathlib.Path(args.output_dir)`
- `parents`
- `True`
- `exist_ok`
- tuple `device`
- `ckpt`
- `model`
- `gt_dir` = `pathlib.Path(args.gt_dir)`
- `raw_dir` = `pathlib.Path(args.raw_dir)`
- `model_1_dir` = `pathlib.Path(args.model_1_dir)`
- `gt_files` = `sorted(list(gt_dir.glob(args.glob_str)))`
- `raw_files` = `sorted(list(raw_dir.glob(args.glob_str)))`
- `model_1_files` = `sorted(list(model_1_dir.glob(args.glob_str)))`
- `model_2_dir` = `pathlib.Path(args.model_2_dir)`
- `model_2_files` = `sorted(list(model_2_dir.glob(args.glob_str)))`
- `N` = `len(gt_files)`
- `psnr` = `PSNR(data_range=65536, device=device)`
- `ssim`
- `df`
- `gt` = `reshape_to_bcwh(tiffimage.imread(gt_files[i]))`
- `raw` = `reshape_to_bcwh(tiffimage.imread(raw_files[i]))`
- `model_1` = `reshape_to_bcwh(tiffimage.imread(model_1_files[i]))`
- `model_2` = `reshape_to_bcwh(tiffimage.imread(model_2_files[i]))`
- `rng` = `np.random.default_rng(seed=31052024)`
- `img_idx` = `list(range(N))`
- list `gt_samples` = `[np.squeeze(tiffimage.imread(gt_files[i])) for i in img_idx]`
- list `raw_samples` = `[np.squeeze(tiffimage.imread(raw_files[i])) for i in img_idx]`
- list `model_1_samples`
- list `model_2_samples`
- `cmap`

## 5.1.1 Variable Documentation

### 5.1.1.1 action

`analyse.action`

### 5.1.1.2 args

`analyse.args = parser.parse_args()`

### 5.1.1.3 ckpt

`analyse.ckpt`

### 5.1.1.4 cmap

`analyse.cmap`

### 5.1.1.5 default

`analyse.default`

### 5.1.1.6 device

`tuple analyse.device`

**Initial value:**

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```

#### 5.1.1.7 df

analyse.df

##### Initial value:

```
1 = pd.DataFrame(  
2     columns=[  
3         "file",  
4         "psnr_raw",  
5         "psnr_model_1",  
6         "psnr_model_2",  
7         "ssim_raw",  
8         "ssim_model_1",  
9         "ssim_model_2",  
10    ]  
11 )
```

#### 5.1.1.8 exist\_ok

analyse.exist\_ok

#### 5.1.1.9 gt

analyse.gt = reshape\_to\_bcwh(tifffile.imread(gt\_files[i]))

#### 5.1.1.10 gt\_dir

analyse.gt\_dir = pathlib.Path(args.gt\_dir)

#### 5.1.1.11 gt\_files

analyse.gt\_files = sorted(list(gt\_dir.glob(args.glob\_str)))

#### 5.1.1.12 gt\_samples

list analyse.gt\_samples = [np.squeeze(tifffile.imread(gt\_files[i])) for i in img\_idx]

#### 5.1.1.13 `img_idx`

```
analyse.img_idx = list(range(N))
```

#### 5.1.1.14 `int`

```
analyse.int
```

#### 5.1.1.15 `model`

```
analyse.model
```

#### 5.1.1.16 `model_1`

```
analyse.model_1 = reshape_to_bcwh(tifffile.imread(model_1_files[i]))
```

#### 5.1.1.17 `model_1_dir`

```
analyse.model_1_dir = pathlib.Path(args.model_1_dir)
```

#### 5.1.1.18 `model_1_files`

```
analyse.model_1_files = sorted(list(model_1_dir.glob(args.glob_str)))
```

#### 5.1.1.19 `model_1_samples`

```
list analyse.model_1_samples
```

#### Initial value:

```
1 = [  
2     np.squeeze(tifffile.imread(model_1_files[i])) for i in img_idx  
3 ]
```

#### 5.1.1.20 model\_2

```
analyse.model_2 = reshape_to_bcwh(tifffile.imread(model_2_files[i]))
```

#### 5.1.1.21 model\_2\_dir

```
analyse.model_2_dir = pathlib.Path(args.model_2_dir)
```

#### 5.1.1.22 model\_2\_files

```
list analyse.model_2_files = sorted(list(model_2_dir.glob(args.glob_str)))
```

#### 5.1.1.23 model\_2\_samples

```
analyse.model_2_samples
```

##### Initial value:

```
1 = [  
2     np.squeeze(tifffile.imread(model_2_files[i])) for i in img_idx  
3 ]
```

#### 5.1.1.24 N

```
analyse.N = len(gt_files)
```

#### 5.1.1.25 output\_dir

```
analyse.output_dir = pathlib.Path(args.output_dir)
```

#### 5.1.1.26 parents

```
analyse.parents
```

#### 5.1.1.27 parser

```
analyse.parser = argparse.ArgumentParser()
```

#### 5.1.1.28 psnr

```
analyse.psnr = PSNR(data_range=65536, device=device)
```

#### 5.1.1.29 raw

```
analyse.raw = reshape_to_bcwh(tifffile.imread(raw_files[i]))
```

#### 5.1.1.30 raw\_dir

```
analyse.raw_dir = pathlib.Path(args.raw_dir)
```

#### 5.1.1.31 raw\_files

```
analyse.raw_files = sorted(list(raw_dir.glob(args.glob_str)))
```

#### 5.1.1.32 raw\_samples

```
list analyse.raw_samples = [np.squeeze(tifffile.imread(raw_files[i])) for i in img_idx]
```

#### 5.1.1.33 required

```
analyse.required
```

#### 5.1.1.34 rng

```
analyse.rng = np.random.default_rng(seed=31052024)
```

5.1.1.35 **ssim**

`analyse.ssim`

**Initial value:**

```
1 = SSIM(
2     data_range=65536,
3     kernel_size=(11, 11),
4     sigma=(1.5, 1.5),
5     k1=0.01,
6     k2=0.03,
7     gaussian=True,
8     device=device,
9 )
```

5.1.1.36 **str**

`analyse.str`

5.1.1.37 **True**

`analyse.True`

5.1.1.38 **type**

`analyse.type`

## 5.2 apply Namespace Reference

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `default`
- `percentile`
- `action`
- `args` = `parser.parse_args()`
- `input_path` = `pathlib.Path(args.input)`
- `output_path` = `pathlib.Path(args.output)`
- `parents`
- `raw_files` = `sorted(input_path.glob("*.tif"))`
- `data` = `itertools.zip_longest(raw_files, [])`
- tuple `device`
- `ckpt`
- `model`
- `RCAN_hyperparameters` = `ckpt["hyperparameters"]`
- list `overlap_shape`
- `raw` = `normalize(tifffile.imread(raw_file), args.p_min, args.p_max)`
- `restored`
- `output_file` = `output_path / ("pred_" + raw_file.name)`
- `imagej`

## 5.2.1 Variable Documentation

### 5.2.1.1 action

`apply.action`

### 5.2.1.2 args

`apply.args = parser.parse_args()`

### 5.2.1.3 choices

`apply.choices`

### 5.2.1.4 ckpt

`apply.ckpt`

### 5.2.1.5 data

`list apply.data = itertools.zip_longest(raw_files, [])`

### 5.2.1.6 default

`apply.default`

### 5.2.1.7 device

`tuple apply.device`

#### Initial value:

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```



### 5.2.1.8 imagej

```
apply.imagej
```

### 5.2.1.9 input\_path

```
apply.input_path = pathlib.Path(args.input)
```

### 5.2.1.10 int

```
apply.int
```

### 5.2.1.11 model

```
apply.model
```

### 5.2.1.12 output\_file

```
apply.output_file = output_path / ("pred_" + raw_file.name)
```

### 5.2.1.13 output\_path

```
apply.output_path = pathlib.Path(args.output)
```

### 5.2.1.14 overlap\_shape

```
apply.overlap_shape
```

#### Initial value:

```
1 = [  
2     max(1, x // 8) if x > 2 else 0  
3     for x in RCAN_hyperparameters["input_shape"]  
4 ]
```

#### 5.2.1.15 parents

```
apply.parents
```

#### 5.2.1.16 parser

```
apply.parser = argparse.ArgumentParser()
```

#### 5.2.1.17 percentile

```
apply.percentile
```

#### 5.2.1.18 raw

```
apply.raw = normalize(tifffile.imread(raw_file), args.p_min, args.p_max)
```

#### 5.2.1.19 raw\_files

```
apply.raw_files = sorted(input_path.glob("*.tif"))
```

#### 5.2.1.20 RCAN\_hyperparameters

```
apply.RCAN_hyperparameters = ckpt["hyperparameters"]
```

#### 5.2.1.21 required

```
apply.required
```

#### 5.2.1.22 restored

apply.restored

##### Initial value:

```
1 = apply(  
2     model,  
3     raw,  
4     RCAN_hyperparameters["input_shape"],  
5     RCAN_hyperparameters["input_shape"],  
6     RCAN_hyperparameters["num_input_channels"],  
7     RCAN_hyperparameters["num_output_channels"],  
8     batch_size=1,  
9     device=device,  
10    overlap_shape=overlap_shape,  
11    verbose=True,  
12    )
```

#### 5.2.1.23 str

apply.str

#### 5.2.1.24 type

apply.type

## 5.3 convert\_omx\_to\_czxy Namespace Reference

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `action`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `input_files` = `sorted(input_dir.rglob("*.tif"))`
- `original` = `tiffimage.imread(input_file)`
- `converted`
- `imagej`

### 5.3.1 Variable Documentation

#### 5.3.1.1 action

`convert_omx_to_czxy.action`

#### 5.3.1.2 args

`convert_omx_to_czxy.args = parser.parse_args()`

#### 5.3.1.3 converted

`convert_omx_to_czxy.converted`

##### Initial value:

```
1 = conv_omx_to_czxy(  
2     original, args.num_phases, args.num_angles  
3 )
```

#### 5.3.1.4 imagej

`convert_omx_to_czxy.imagej`

#### 5.3.1.5 input\_dir

`convert_omx_to_czxy.input_dir = pathlib.Path(args.input)`

#### 5.3.1.6 input\_files

`convert_omx_to_czxy.input_files = sorted(input_dir.rglob("*.tif"))`

#### 5.3.1.7 int

`convert_omx_to_czxy.int`

#### 5.3.1.8 original

```
convert_omx_to_czxy.original = tifffile.imread(input_file)
```

#### 5.3.1.9 parser

```
convert_omx_to_czxy.parser = argparse.ArgumentParser()
```

#### 5.3.1.10 required

```
convert_omx_to_czxy.required
```

#### 5.3.1.11 str

```
convert_omx_to_czxy.str
```

#### 5.3.1.12 type

```
convert_omx_to_czxy.type
```

## 5.4 convert\_omx\_to\_paz Namespace Reference

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `action`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `input_files` = `sorted(input_dir.rglob("*.tif"))`
- `original` = `tifffile.imread(input_file)`
- `converted` = `conv_omx_to_paz(original, args.num_phases, args.num_angles)`
- `imagej`

## 5.4.1 Variable Documentation

### 5.4.1.1 action

```
convert_omx_to_paz.action
```

### 5.4.1.2 args

```
convert_omx_to_paz.args = parser.parse_args()
```

### 5.4.1.3 converted

```
convert_omx_to_paz.converted = conv_omx_to_paz(original, args.num_phases, args.num_angles)
```

### 5.4.1.4 imagej

```
convert_omx_to_paz.imagej
```

### 5.4.1.5 input\_dir

```
convert_omx_to_paz.input_dir = pathlib.Path(args.input)
```

### 5.4.1.6 input\_files

```
convert_omx_to_paz.input_files = sorted(input_dir.rglob("*.tif"))
```

### 5.4.1.7 int

```
convert_omx_to_paz.int
```

#### 5.4.1.8 original

```
convert_omx_to_paz.original = tifffile.imread(input_file)
```

#### 5.4.1.9 parser

```
convert_omx_to_paz.parser = argparse.ArgumentParser()
```

#### 5.4.1.10 required

```
convert_omx_to_paz.required
```

#### 5.4.1.11 str

```
convert_omx_to_paz.str
```

#### 5.4.1.12 type

```
convert_omx_to_paz.type
```

## 5.5 convert\_slices\_to\_volumes Namespace Reference

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `tuple_of_ints`
- `default`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `output_dir` = `pathlib.Path(args.output)`
- `input_files` = `sorted(input_dir.glob("*.tif"))`
- `parents`
- `True`
- `exist_ok`
- `volume` = `np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)`
- `input_slice` = `tifffile.imread(file)`
- `output_file` = `output_dir / filename`
- `subvolume`
- `imagej`

## 5.5.1 Variable Documentation

### 5.5.1.1 args

```
convert_slices_to_volumes.args = parser.parse_args()
```

### 5.5.1.2 default

```
convert_slices_to_volumes.default
```

### 5.5.1.3 exist\_ok

```
convert_slices_to_volumes.exist_ok
```

### 5.5.1.4 imagej

```
convert_slices_to_volumes.imagej
```

### 5.5.1.5 input\_dir

```
convert_slices_to_volumes.input_dir = pathlib.Path(args.input)
```

### 5.5.1.6 input\_files

```
convert_slices_to_volumes.input_files = sorted(input_dir.glob("*.tif"))
```

### 5.5.1.7 input\_slice

```
convert_slices_to_volumes.input_slice = tifffile.imread(file)
```



#### 5.5.1.8 output\_dir

```
convert_slices_to_volumes.output_dir = pathlib.Path(args.output)
```

#### 5.5.1.9 output\_file

```
convert_slices_to_volumes.output_file = output_dir / filename
```

#### 5.5.1.10 parents

```
convert_slices_to_volumes.parents
```

#### 5.5.1.11 parser

```
convert_slices_to_volumes.parser = argparse.ArgumentParser()
```

#### 5.5.1.12 required

```
convert_slices_to_volumes.required
```

#### 5.5.1.13 str

```
convert_slices_to_volumes.str
```

#### 5.5.1.14 subvolume

```
convert_slices_to_volumes.subvolume
```

#### 5.5.1.15 True

```
convert_slices_to_volumes.True
```

#### 5.5.1.16 tuple\_of\_ints

```
convert_slices_to_volumes.tuple_of_ints
```

#### 5.5.1.17 type

```
convert_slices_to_volumes.type
```

#### 5.5.1.18 volume

```
convert_slices_to_volumes.volume = np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)
```

## 5.6 generate\_sim Namespace Reference

### Classes

- class [Simulator](#)  
*The [Simulator](#) class encapsulates the state of a 3D microscope simulation.*
- class [SimulationRunner](#)  
*Class which performs a batch of simulations, either sequentially or in parallel.*

### Functions

- def [arange\\_zero](#) (n, spacing=1)
- def [threshold\\_norm](#) (sample)  
*Applies a threshold and normalises the sample to improve contrast.*

### Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [int](#)
- [default](#)
- [args](#) = parser.parse\_args()
- [runner](#)

#### 5.6.1 Function Documentation

#### 5.6.1.1 arange\_zero()

```
def generate_sim.arange_zero (
    n,
    spacing = 1 )
```

#### 5.6.1.2 threshold\_norm()

```
def generate_sim.threshold_norm (
    sample )
```

Applies a threshold and normalises the sample to improve contrast.

### 5.6.2 Variable Documentation

#### 5.6.2.1 args

```
generate_sim.args = parser.parse_args()
```

#### 5.6.2.2 default

```
generate_sim.default
```

#### 5.6.2.3 int

```
generate_sim.int
```

#### 5.6.2.4 parser

```
generate_sim.parser = argparse.ArgumentParser()
```

#### 5.6.2.5 required

`generate_sim.required`

#### 5.6.2.6 runner

`generate_sim.runner`

##### Initial value:

```
1 = SimulationRunner(  
2     args.input, args.output, range(args.start, args.end), args.z_offset  
3 )
```

#### 5.6.2.7 str

`generate_sim.str`

#### 5.6.2.8 type

`generate_sim.type`

## 5.7 image\_noising Namespace Reference

### Functions

- def `save_image_pair` (gt\_img, `split`, name, channel\_idx)

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `float`
- `default`
- `args` = `parser.parse_args()`
- `input_path` = `pathlib.Path(args.input)`
- `output_path` = `pathlib.Path(args.output)`
- `parents`
- `output_train_gt_path` = `output_path.joinpath("Training", "GT")`
- `output_train_raw_path` = `output_path.joinpath("Training", "Raw")`

- `output_val_gt_path` = `output_path.joinpath("Validation", "GT")`
- `output_val_raw_path` = `output_path.joinpath("Validation", "Raw")`
- `output_test_gt_path` = `output_path.joinpath("Testing", "GT")`
- `output_test_raw_path` = `output_path.joinpath("Testing", "Raw")`
- `data` = `sorted(input_path.glob("*.tif"))`
- `n_acquisitions` = `tifffile.imread(data[0]).shape[0] // args.channels`
- `n_img` = `len(data)`
- `train_size` = `int((1 - args.test_fraction) * n_img)`
- `val_size` = `int(args.val_fraction * train_size)`
- `rng` = `np.random.default_rng(seed=25042024)`
- `img_idx_all` = `list(range(n_img))`
- `img_idx_test` = `img_idx_all[train_size:]`
- `img_idx_train` = `img_idx_all[: train_size - val_size]`
- `img_idx_val` = `img_idx_all[train_size - val_size : train_size]`
- `gt` = `tifffile.imread(img_file)`
- string `split` = "train"

## 5.7.1 Function Documentation

### 5.7.1.1 `save_image_pair()`

```
def image_noising.save_image_pair (
    gt_img,
    split,
    name,
    channel_idx )
```

## 5.7.2 Variable Documentation

### 5.7.2.1 `args`

```
image_noising.args = parser.parse_args()
```

### 5.7.2.2 `choices`

```
image_noising.choices
```

### 5.7.2.3 data

```
list image_noising.data = sorted(input_path.glob("*.tif"))
```

### 5.7.2.4 default

```
image_noising.default
```

### 5.7.2.5 float

```
image_noising.float
```

### 5.7.2.6 gt

```
image_noising.gt = tifffile.imread(img_file)
```

### 5.7.2.7 img\_idx\_all

```
image_noising.img_idx_all = list(range(n_img))
```

### 5.7.2.8 img\_idx\_test

```
image_noising.img_idx_test = img_idx_all[train_size:]
```

### 5.7.2.9 img\_idx\_train

```
image_noising.img_idx_train = img_idx_all[: train_size - val_size]
```

### 5.7.2.10 img\_idx\_val

```
image_noising.img_idx_val = img_idx_all[train_size - val_size : train_size]
```

#### 5.7.2.11 input\_path

```
image_noising.input_path = pathlib.Path(args.input)
```

#### 5.7.2.12 int

```
image_noising.int
```

#### 5.7.2.13 n\_acquisitions

```
image_noising.n_acquisitions = tiffiffile.imread(data[0]).shape[0] // args.channels
```

#### 5.7.2.14 n\_img

```
image_noising.n_img = len(data)
```

#### 5.7.2.15 output\_path

```
image_noising.output_path = pathlib.Path(args.output)
```

#### 5.7.2.16 output\_test\_gt\_path

```
image_noising.output_test_gt_path = output_path.joinpath("Testing", "GT")
```

#### 5.7.2.17 output\_test\_raw\_path

```
image_noising.output_test_raw_path = output_path.joinpath("Testing", "Raw")
```

#### 5.7.2.18 output\_train\_gt\_path

```
image_noising.output_train_gt_path = output_path.joinpath("Training", "GT")
```

#### 5.7.2.19 output\_train\_raw\_path

```
image_noising.output_train_raw_path = output_path.joinpath("Training", "Raw")
```

#### 5.7.2.20 output\_val\_gt\_path

```
image_noising.output_val_gt_path = output_path.joinpath("Validation", "GT")
```

#### 5.7.2.21 output\_val\_raw\_path

```
image_noising.output_val_raw_path = output_path.joinpath("Validation", "Raw")
```

#### 5.7.2.22 parents

```
image_noising.parents
```

#### 5.7.2.23 parser

```
image_noising.parser = argparse.ArgumentParser()
```

#### 5.7.2.24 required

```
image_noising.required
```

#### 5.7.2.25 rng

```
image_noising.rng = np.random.default_rng(seed=25042024)
```

#### 5.7.2.26 split

```
string image_noising.split = "train"
```



#### 5.7.2.27 str

```
image_noising.str
```

#### 5.7.2.28 train\_size

```
image_noising.train_size = int((1 - args.test_fraction) * n_img)
```

#### 5.7.2.29 type

```
image_noising.type
```

#### 5.7.2.30 val\_size

```
image_noising.val_size = int(args.val_fraction * train_size)
```

## 5.8 manage\_stack Namespace Reference

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `default`
- `action`
- `args` = `parser.parse_args()`
- `output_dir` = `pathlib.Path(args.output_dir)`
- `parents`
- `True`
- `exist_ok`
- `files` = `sorted(list(pathlib.Path(args.input_dir).glob(args.glob_str)))`
- `int stack_number` = `-1` else `args.stack_number`
- `int number_of_stacks` = `len(files) // stack_number`
- `sample` = `tiffimage.imread(files[0])`
- `stack_handler`
- `img_data` = `tiffimage.imread(input_file)`
- tuple `filename`
- tuple `output_file` = `output_dir / filename`
- `output_data`

## 5.8.1 Variable Documentation

### 5.8.1.1 action

`manage_stack.action`

### 5.8.1.2 args

`manage_stack.args = parser.parse_args()`

### 5.8.1.3 choices

`manage_stack.choices`

### 5.8.1.4 default

`manage_stack.default`

### 5.8.1.5 exist\_ok

`manage_stack.exist_ok`

### 5.8.1.6 filename

`tuple manage_stack.filename`

#### Initial value:

```
1 = (  
2     args.output_name  
3     + f"_stack(stack_idx*stack_number:04d)"  
4     + f"_{(stack_idx+1)*stack_number:04d}"  
5 )
```

### 5.8.1.7 files

```
manage_stack.files = sorted(list(pathlib.Path(args.input_dir).glob(args.glob_str)))
```

### 5.8.1.8 img\_data

```
manage_stack.img_data = tifffile.imread(input_file)
```

### 5.8.1.9 int

```
manage_stack.int
```

### 5.8.1.10 number\_of\_stacks

```
int manage_stack.number_of_stacks = len(files) // stack_number
```

### 5.8.1.11 output\_data

```
manage_stack.output_data
```

#### Initial value:

```
1 = img_data[
2     j * args.z_slices : (j + 1) * args.z_slices
3 ]
```

### 5.8.1.12 output\_dir

```
manage_stack.output_dir = pathlib.Path(args.output_dir)
```

### 5.8.1.13 output\_file

```
string manage_stack.output_file = output_dir / filename
```

#### 5.8.1.14 parents

```
manage_stack.parents
```

#### 5.8.1.15 parser

```
manage_stack.parser = argparse.ArgumentParser()
```

#### 5.8.1.16 required

```
manage_stack.required
```

#### 5.8.1.17 sample

```
manage_stack.sample = tifffile.imread(files[0])
```

#### 5.8.1.18 stack\_handler

```
manage_stack.stack_handler
```

##### Initial value:

```
1 = ImageStack(  
2     args.dimension,  
3     stack_number,  
4     stack_idx,  
5     sample,  
6     files,  
7     args.num_acquisitions  
8 )
```

#### 5.8.1.19 stack\_number

```
int manage_stack.stack_number = -1 else args.stack_number
```

#### 5.8.1.20 str

```
manage_stack.str
```

#### 5.8.1.21 True

`manage_stack.True`

#### 5.8.1.22 type

`manage_stack.type`

## 5.9 rcan Namespace Reference

### Namespaces

- [data\\_generator](#)
- [data\\_processing](#)
- [model](#)
- [plotting](#)
- [utils](#)

## 5.10 rcan.data\_generator Namespace Reference

### Classes

- class [SIM\\_Dataset](#)  
*Generates batches of images with real-time data augmentation.*

### Functions

- def [load\\_SIM\\_dataset](#) (images, shape, batch\_size, transform\_function, intensity\_threshold, area\_threshold, scale\_factor, steps\_per\_epoch, p\_min, p\_max)  
*Wraps [SIM\\_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.*

### 5.10.1 Function Documentation

#### 5.10.1.1 load\_SIM\_dataset()

```
def rcan.data_generator.load_SIM_dataset (
    images,
    shape,
    batch_size,
    transform_function,
    intensity_threshold,
    area_threshold,
    scale_factor,
    steps_per_epoch,
    p_min,
    p_max )
```

Wraps [SIM\\_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.

## Parameters

<i>images</i>	(list[dict]) - List of dictionaries of data pairs with keys ["raw","gt"]. Images in CZXY format
<i>shape</i>	(tuple[int]) - Shape of batch images excluding the channel dimension
<i>batch_size</i>	(int) - Batch size
<i>transform_function</i>	(str or callable or None) - Function used for data augmentation. Typically you will set <code>transform_function='rotate_and_flip'</code> to apply combination of randomly selected image rotation and flipping. Alternatively, you can specify an arbitrary transformation function which takes two input images (source and target) and returns transformed images. If <code>transform_function=None</code> , no augmentation will be performed
<i>intensity_threshold</i>	(float) - If <code>intensity_threshold &gt; 0</code> , pixels whose intensities are greater than this threshold will be considered as foreground
<i>area_ratio_threshold</i>	(float) - Threshold between 0 and 1. If <code>intensity_threshold &gt; 0</code> , the generator calculates the ratio of foreground pixels in a target patch, and rejects the patch if the ratio is smaller than this threshold
<i>scale_factor</i>	(int) - Scale factor for the target patch size. Positive and negative values mean up- and down-scaling respectively.
<i>steps_per_epoch</i>	(int) - Determines how many times each image is used to generate a patch per batch
<i>p_min</i>	(float) - Minimum percentile used for scaling
<i>p_max</i>	(float) - Maximum percentile used for scaling

## Returns

`torch.utils.data.DataLoader` object

## 5.11 rcan.data\_processing Namespace Reference

### Classes

- class [ImageStack](#)

*Handles creation and loading of image hyperstacks in order to make reconstructions using ImageJ easier.*

### Functions

- def [crop\\_volume](#) (volume, num\_steps, start, step, label)  
*Takes an image volume and divides part of it into smaller volumes by cropping lateral sections (the full z dimension is used).*
- def [conv\\_omx\\_to\\_czxy](#) (original, n\_phases, n\_angles)  
*Converts image array from OMX (PZA format) to CZXY format.*
- def [conv\\_czxy\\_to\\_omx](#) (original, n\_phases, n\_angles)  
*Converts image array from CZXY to OMX format.*
- def [conv\\_omx\\_to\\_paz](#) (original, n\_phases, n\_angles)  
*Converts image array from OMX (PZA format) to PAZ format.*
- def [conv\\_paz\\_to\\_omx](#) (original, n\_phases, n\_angles)  
*Converts image array from PAZ to OMX(PZA) format.*

## 5.11.1 Function Documentation

### 5.11.1.1 conv\_czxy\_to\_omx()

```
def rcan.data_processing.conv_czxy_to_omx (
    original,
    n_phases,
    n_angles )
```

Converts image array from CZXY to OMX format.

#### Parameters

<i>original</i>	(np.ndarray) - Image array in original format
<i>n_phases</i>	(int) - Number of phases
<i>n_angles</i>	(int) - Number of angles

#### Returns

np.ndarray Converted image array

### 5.11.1.2 conv\_omx\_to\_czxy()

```
def rcan.data_processing.conv_omx_to_czxy (
    original,
    n_phases,
    n_angles )
```

Converts image array from OMX (PZA format) to CZXY format.

#### Parameters

<i>original</i>	(np.ndarray) - Image array in original format
<i>n_phases</i>	(int) - Number of phases
<i>n_angles</i>	(int) - Number of angles

#### Returns

np.ndarray Converted image array

### 5.11.1.3 conv\_omx\_to\_paz()

```
def rcan.data_processing.conv_omx_to_paz (
    original,
    n_phases,
    n_angles )
```

Converts image array from OMX (PZA format) to PAZ format.

#### Parameters

<i>original</i>	(np.ndarray) - Image array in original format
<i>n_phases</i>	(int) - Number of phases
<i>n_angles</i>	(int) - Number of angles

#### Returns

np.ndarray Converted image array

### 5.11.1.4 conv\_paz\_to\_omx()

```
def rcan.data_processing.conv_paz_to_omx (
    original,
    n_phases,
    n_angles )
```

Converts image array from PAZ to OMX(PZA) format.

#### Parameters

<i>original</i>	(np.ndarray) - Image array in original format
<i>n_phases</i>	(int) - Number of phases
<i>n_angles</i>	(int) - Number of angles

#### Returns

np.ndarray Converted image array

### 5.11.1.5 crop\_volume()

```
def rcan.data_processing.crop_volume (
    volume,
    num_steps,
    start,
```



```
step,  
label )
```

Takes an image volume and divides part of it into smaller volumes by cropping lateral sections (the full z dimension is used).

## Parameters

<i>volume</i>	(np.ndarray) - image volume to crop
<i>num_steps</i>	(tuple[int]) - number of images in each lateral dimension (total number of subvolumes is the product)
<i>start</i>	(tuple[int]) - start coordinates for crop region
<i>step</i>	(tuple[int]) - lateral size of subvolume images
<i>label</i>	(str) - prefix for output file names

## Returns

generator that yields image subvolumes

## 5.12 rcan.model Namespace Reference

### Classes

- class [\\_channel\\_attention\\_block](#)  
*Implements channel attention block/layer.*
- class [\\_residual\\_channel\\_attention\\_blocks](#)  
*Implements residual group based on [1].*
- class [RCAN](#)  
*Builds a residual channel attention network.*

### Functions

- def [\\_conv](#) (ndim, in\_filters, out\_filters, kernel\_size, padding="same", \*\*kwargs)  
*Returns the appropriate torch.nn convolution layer based on parameters.*
- def [\\_global\\_average\\_pooling](#) (ndim)  
*Returns the appropriate torch.nn pooling layer based on parameters.*
- def [\\_standardize](#) (x)  
*Standardises input data.*
- def [\\_destandardize](#) (x)  
*Inverse of \_standardize.*

#### 5.12.1 Function Documentation

##### 5.12.1.1 \_conv()

```
def rcan.model._conv (
    ndim,
    in_filters,
    out_filters,
    kernel_size,
    padding = "same",
    ** kwargs ) [private]
```

Returns the appropriate torch.nn convolution layer based on parameters.

## Parameters

<i>ndim</i>	(int) - Specifies a 1, 2, or 3 dimensional convolution kernel
<i>in_filters</i>	(int) - Number of hidden input channels
<i>out_filters</i>	(int) - Number of hidden output channels
<i>kernel_size</i>	(int or tuple) Size of convolution kernel
<i>padding</i>	(str, optional) - Border padding strategy. Default: "same"

## Returns

torch.nn.Module object of the specified type

**5.12.1.2 \_destandardize()**

```
def rcan.model._destandardize (
    x ) [private]
```

Inverse of \_standardize.

## Parameters

<i>x</i>	(torch.Tensor) Input
----------	----------------------

## Returns

torch.Tensor representing destandardised output.

**5.12.1.3 \_global\_average\_pooling()**

```
def rcan.model._global_average_pooling (
    ndim ) [private]
```

Returns the appropriate torch.nn pooling layer based on parameters.

## Parameters

<i>ndim</i>	(int) - Specifies a 2 or 3 dimensional convolution kernel
-------------	---

## Returns

torch.nn.Module object of the specified type

#### 5.12.1.4 `_standardize()`

```
def rcan.model._standardize (
    x ) [private]
```

Standardises input data.

Standardize the signal so that the range becomes [-1, 1] (assuming the original range is [0, 1]).

##### Parameters

<code>x</code>	(torch.Tensor) Input
----------------	----------------------

##### Returns

torch.Tensor representing standardised output

## 5.13 `rcan.plotting` Namespace Reference

### Functions

- def [plot\\_learning\\_curve](#) (losses\_train, losses\_val, psnr\_train, psnr\_val, ssim\_train, ssim\_val, figsize, output\_path)
 

*Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.*
- def [compute\\_metrics](#) (img, gt\_img, psnr, ssim)
 

*Uses ignite metric objects to compute PSNR and SSIM.*
- def [plot\\_reconstructions](#) (device, output\_path, dim, gt\_imgs, raw\_imgs, model\_1\_imgs, model\_2\_imgs=None, cmap="inferno")
 

*Plots a sample of reconstructions comparing GT vs Raw vs Restored.*

### 5.13.1 Function Documentation

#### 5.13.1.1 `compute_metrics()`

```
def rcan.plotting.compute_metrics (
    img,
    gt_img,
    psnr,
    ssim )
```

Uses ignite metric objects to compute PSNR and SSIM.

##### Parameters

<code>img</code>	(np.ndarray) - Predicted image
<code>gt_img</code>	(np.ndarray) - Reference image
<code>psnr</code>	(ignite.metrics.PSNR) - PSNR object
<code>ssim</code>	(ignite.metrics.SSIM) - SSIM object

**Returns**

dict of metric values

**5.13.1.2 plot\_learning\_curve()**

```
def rcan.plotting.plot_learning_curve (
    losses_train,
    losses_val,
    psnr_train,
    psnr_val,
    ssim_train,
    ssim_val,
    figsize,
    output_path )
```

Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.

**Parameters**

<i>losses_train</i>	(list[float]) - List of training losses
<i>losses_val</i>	(list[float]) - List of validation losses
<i>psnr_train</i>	(list[float]) - List of training psnrs
<i>psnr_val</i>	(list[float]) - List of validation psnrs
<i>ssim_train</i>	(list[float]) - List of training ssims
<i>ssim_val</i>	(list[float]) - List of validation ssims
<i>figsize</i>	(tuple[int]) - Specifies matplotlib layout size
<i>output_path</i>	(str) - Determines where plot is saved

**5.13.1.3 plot\_reconstructions()**

```
def rcan.plotting.plot_reconstructions (
    device,
    output_path,
    dim,
    gt_imgs,
    raw_imgs,
    model_1_imgs,
    model_2_imgs = None,
    cmap = "inferno" )
```

Plots a sample of reconstructions comparing GT vs Raw vs Restored.

**Parameters**

<i>device</i>	(torch.device) - Handles the processing unit for torch
<i>output_path</i>	(str) - Determines where the plot is saved

## Parameters

<i>dim</i>	(int) - Dimensionality of the images
<i>gt_imgs</i>	(list[np.ndarray]) - List containing GT image arrays
<i>raw_imgs</i>	(list[np.ndarray]) - List containing Raw image arrays
<i>model_1_imgs</i>	(list[np.ndarray]) - List containing Step 1 image arrays
<i>model_2_imgs</i>	(list[np.ndarray], optional) - List containing Step 2 image arrays. Default: None
<i>cmap</i>	(str) - Matplotlib colormap string

## 5.14 rcn.utils Namespace Reference

### Functions

- def [normalize](#) (image, p\_min=2, p\_max=99.9, dtype="float32")  
*Normalizes the image intensity so that the  $p_{min}$ -th and the  $p_{max}$ -th percentiles are converted to 0 and 1 respectively.*
- def [apply](#) (model, data, model\_input\_image\_shape, model\_output\_image\_shape, num\_input\_channels, num\_output\_channels, batch\_size, device, overlap\_shape=None, verbose=False)  
*Applies a model to an input image.*
- def [load\\_rcan\\_checkpoint](#) (ckpt\_path, device)  
*Enables loading of RCAN checkpointed model.*
- def [tuple\\_of\\_ints](#) (string)  
*Defines behaviour of parsing tuples of ints (argparse).*
- def [percentile](#) (x)  
*Defines behaviour of parsing percentiles (argparse).*
- def [reshape\\_to\\_bcwh](#) (data)  
*Reshapes 2D or 3D array to have batch x channel x width x height format, by prepending extra dimensions.*
- def [normalize\\_between\\_zero\\_and\\_one](#) (data)  
*Coerce pixel values to [0, 1] range.*

### 5.14.1 Function Documentation

#### 5.14.1.1 [apply\(\)](#)

```
def rcn.utils.apply (
    model,
    data,
    model_input_image_shape,
    model_output_image_shape,
    num_input_channels,
    num_output_channels,
    batch_size,
    device,
    overlap_shape = None,
    verbose = False )
```

Applies a model to an input image.

The input image stack is split into sub-blocks with model's input size, then the model is applied block by block.

## Parameters

<i>model</i>	(torch.nn.module) - PyTorch model
<i>data</i>	(array_like or list of array_like) - Input data. Either an image or a list of images
<i>batch_size</i>	(int) - Controls the batch size used to process image data
<i>device</i>	(torch.device) - PyTorch device object to specify processor to use
<i>overlap_shape</i>	(tuple of int or None) - Overlap size between sub-blocks in each dimension. If not specified, a default size ((32, 32) for 2D and (2, 32, 32) for 3D) is used. Results at overlapped areas are blended together linearly

## Returns

np.ndarray Result image

### 5.14.1.2 load\_rcnn\_checkpoint()

```
def rcnn.utils.load_rcnn_checkpoint (
    ckpt_path,
    device )
```

Enables loading of RCAN checkpointed model.

Uses the `hyperparameters` key saved in checkpoint file in order to avoid the need to know the architecture specifications in advance.

## Parameters

<i>ckpt_path</i>	(str) - filepath for checkpoint, should end in .pth
<i>device</i>	(torch.device) - handles processing unit for torch

## Returns

tuple of checkpoint, and model with weights loaded

### 5.14.1.3 normalize()

```
def rcnn.utils.normalize (
    image,
    p_min = 2,
    p_max = 99.9,
    dtype = "float32" )
```

Normalizes the image intensity so that the `p_min`-th and the `p_max`-th percentiles are converted to 0 and 1 respectively.

**Parameters**

<i>image</i>	(np.ndarray) - Image to apply the normalization to
<i>p_min</i>	(float, optional) - Percentile that is mapped to zero. Default: 2
<i>p_max</i>	(float, optional) - Percentile that is mapped to one. Default: 99.9
<i>dtype</i>	(str) - Datatype to use for the output

**Returns**

np.ndarray Image with transformed pixel values

**5.14.1.4 References**

Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy <https://doi.org/10.1038/s41592-018-0216-7>

**5.14.1.5 normalize\_between\_zero\_and\_one()**

```
def rcan.utils.normalize_between_zero_and_one (
    data )
```

Coerce pixel values to [0, 1] range.

**Parameters**

<i>data</i>	(np.ndarray or torch.Tensor) - image array to transform
-------------	---

**Returns**

np.ndarray or torch.Tensor transformed image array

**5.14.1.6 percentile()**

```
def rcan.utils.percentile (
    x )
```

Defines behaviour of parsing percentiles (argparse).

**5.14.1.7 reshape\_to\_bcwh()**

```
def rcan.utils.reshape_to_bcwh (
    data )
```

Reshapes 2D or 3D array to have batch x channel x width x height format, by prepending extra dimensions.



**Parameters**

<i>data</i>	(np.ndarray) - array to be reshaped
-------------	-------------------------------------

**Returns**

np.ndarray transformed data

**5.14.1.8 tuple\_of\_ints()**

```
def rcan.utils.tuple_of_ints (
    string )
```

Defines behaviour of parsing tuples of ints (argparse).

**5.15 recon\_postprocess Namespace Reference****Variables**

- `parser` = argparse.ArgumentParser()
- `type`
- `str`
- `required`
- `args` = parser.parse\_args()
- `files` = sorted(list(pathlib.Path(args.input\_dir).rglob("\*.tif")))
- `img_data` = tifffile.imread(input\_file)

**5.15.1 Variable Documentation****5.15.1.1 args**

```
recon_postprocess.args = parser.parse_args()
```

**5.15.1.2 files**

```
recon_postprocess.files = sorted(list(pathlib.Path(args.input_dir).rglob("*.tif")))
```

#### 5.15.1.3 `img_data`

```
tuple recon_postprocess.img_data = tifffile.imread(input_file)
```

#### 5.15.1.4 `parser`

```
recon_postprocess.parser = argparse.ArgumentParser()
```

#### 5.15.1.5 `required`

```
recon_postprocess.required
```

#### 5.15.1.6 `str`

```
recon_postprocess.str
```

#### 5.15.1.7 `type`

```
recon_postprocess.type
```

## 5.16 `recon_preprocess` Namespace Reference

### Functions

- def `normalize_acquisition_intensity` (data, dim)

### Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `percentile`
- `default`
- `action`
- `args` = `parser.parse_args()`
- `output_dir` = `pathlib.Path(args.output_dir)`
- `parents`
- `True`
- `exist_ok`
- `files` = `sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))`
- `img_data` = `tifffile.imread(input_file).astype("float32")`
- `output_file` = `output_dir / input_file.name`

## 5.16.1 Function Documentation

### 5.16.1.1 `normalize_acquisition_intensity()`

```
def recon_preprocess.normalize_acquisition_intensity (
    data,
    dim )
```

## 5.16.2 Variable Documentation

### 5.16.2.1 `action`

```
recon_preprocess.action
```

### 5.16.2.2 `args`

```
recon_preprocess.args = parser.parse_args()
```

### 5.16.2.3 `choices`

```
recon_preprocess.choices
```

### 5.16.2.4 `default`

```
recon_preprocess.default
```

### 5.16.2.5 `exist_ok`

```
recon_preprocess.exist_ok
```

#### 5.16.2.6 files

```
recon_preprocess.files = sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))
```

#### 5.16.2.7 img\_data

```
int recon_preprocess.img_data = tiffimage.imread(input_file).astype("float32")
```

#### 5.16.2.8 int

```
recon_preprocess.int
```

#### 5.16.2.9 output\_dir

```
recon_preprocess.output_dir = pathlib.Path(args.output_dir)
```

#### 5.16.2.10 output\_file

```
recon_preprocess.output_file = output_dir / input_file.name
```

#### 5.16.2.11 parents

```
recon_preprocess.parents
```

#### 5.16.2.12 parser

```
recon_preprocess.parser = argparse.ArgumentParser()
```

#### 5.16.2.13 percentile

```
recon_preprocess.percentile
```

#### 5.16.2.14 required

`recon_preprocess.required`

#### 5.16.2.15 str

`recon_preprocess.str`

#### 5.16.2.16 True

`recon_preprocess.True`

#### 5.16.2.17 type

`recon_preprocess.type`

## 5.17 stats Namespace Reference

### Functions

- def [paired\\_t](#)(gt\_data, [data](#))

### Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [int](#)
- [choices](#)
- [default](#)
- [args](#) = parser.parse\_args()
- [output\\_dir](#) = pathlib.Path(args.output\_dir)
- [parents](#)
- [True](#)
- [exist\\_ok](#)
- [df](#)
- [fig](#)
- [ax](#)
- [figsize](#)
- [psnr\\_diff\\_1\\_max](#)

- [psnr\\_diff\\_2\\_max](#)
- [psnr\\_diff\\_1\\_min](#)
- [psnr\\_diff\\_2\\_min](#)
- [tuple hist\\_range\\_psnr](#)
- [ssim\\_diff\\_1\\_max](#)
- [ssim\\_diff\\_2\\_max](#)
- [ssim\\_diff\\_1\\_min](#)
- [ssim\\_diff\\_2\\_min](#)
- [tuple hist\\_range\\_ssim](#)
- [xlabel](#)
- [title](#)
- [range](#)
- [color](#)
- [mean\\_psnr\\_1](#) = np.mean(np.array(df['psnr\_model\_1']) - np.array(df['psnr\_raw']))
- [se\\_psnr\\_1](#)
- [mean\\_ssim\\_1](#) = np.mean(np.array(df['ssim\_model\_1']) - np.array(df['ssim\_raw']))
- [se\\_ssim\\_1](#)
- [mean\\_psnr\\_2](#)
- [se\\_psnr\\_2](#)
- [mean\\_ssim\\_2](#)
- [se\\_ssim\\_2](#)
- [int psnr\\_cols](#) = 2 else df.columns[1:3]
- [int ssim\\_cols](#) = 2 else df.columns[3:5]
- [dflong](#)
- [dflongssim](#)
- [data](#)
- [x](#)
- [y](#)
- [hue](#)
- [dodge](#)
- [legend](#)
- [palette](#)
- [alpha](#)
- [lw](#)

## 5.17.1 Function Documentation

### 5.17.1.1 `paired_t()`

```
def stats.paired_t (
    gt_data,
    data )
```

## 5.17.2 Variable Documentation

### 5.17.2.1 alpha

```
stats.alpha
```

### 5.17.2.2 args

```
stats.args = parser.parse_args()
```

### 5.17.2.3 ax

```
stats.ax
```

### 5.17.2.4 choices

```
stats.choices
```

### 5.17.2.5 color

```
stats.color
```

### 5.17.2.6 data

```
stats.data
```

### 5.17.2.7 default

```
stats.default
```

### 5.17.2.8 df

stats.df

**Initial value:**

```
1 = pd.read_csv(  
2     pathlib.Path(args.dataset),  
3     index_col=False  
4 ).drop(columns="Unnamed: 0")
```

### 5.17.2.9 dflong

stats.dflong

**Initial value:**

```
1 = pd.melt(  
2     df,  
3     id_vars=["file"],  
4     value_vars=df.columns[1:4],  
5     var_name="type",  
6     value_name="psnr"  
7 )
```

### 5.17.2.10 dflongssim

stats.dflongssim

**Initial value:**

```
1 = pd.melt(  
2     df,  
3     id_vars=["file"],  
4     value_vars=df.columns[4:7],  
5     var_name="type",  
6     value_name="ssim"  
7 )
```

### 5.17.2.11 dodge

stats.dodge

### 5.17.2.12 exist\_ok

stats.exist\_ok



### 5.17.2.13 fig

`stats.fig`

### 5.17.2.14 figsize

`stats.figsize`

### 5.17.2.15 hist\_range\_psnr

`tuple stats.hist_range_psnr`

#### Initial value:

```
1 = (  
2     min(psnr_diff_1_min, psnr_diff_2_min),  
3     max(psnr_diff_1_max, psnr_diff_2_max)  
4 )
```

### 5.17.2.16 hist\_range\_ssim

`tuple stats.hist_range_ssim`

#### Initial value:

```
1 = (  
2     min(ssim_diff_1_min, ssim_diff_2_min),  
3     max(ssim_diff_1_max, ssim_diff_2_max)  
4 )
```

### 5.17.2.17 hue

`stats.hue`

### 5.17.2.18 int

`stats.int`

### 5.17.2.19 legend

```
stats.legend
```

### 5.17.2.20 lw

```
stats.lw
```

### 5.17.2.21 mean\_psnr\_1

```
stats.mean_psnr_1 = np.mean(np.array(df['psnr_model_1']) - np.array(df['psnr_raw']))
```

### 5.17.2.22 mean\_psnr\_2

```
stats.mean_psnr_2
```

#### Initial value:

```
1 = np.mean(  
2     np.array(df['psnr_model_2']) - np.array(df['psnr_raw'])  
3 )
```

### 5.17.2.23 mean\_ssim\_1

```
stats.mean_ssim_1 = np.mean(np.array(df['ssim_model_1']) - np.array(df['ssim_raw']))
```

### 5.17.2.24 mean\_ssim\_2

```
stats.mean_ssim_2
```

#### Initial value:

```
1 = np.mean(  
2     np.array(df['ssim_model_2']) - np.array(df['ssim_raw'])  
3 )
```

### 5.17.2.25 output\_dir

```
stats.output_dir = pathlib.Path(args.output_dir)
```

### 5.17.2.26 palette

```
stats.palette
```

### 5.17.2.27 parents

```
stats.parents
```

### 5.17.2.28 parser

```
stats.parser = argparse.ArgumentParser()
```

### 5.17.2.29 psnr\_cols

```
int stats.psnr_cols = 2 else df.columns[1:3]
```

### 5.17.2.30 psnr\_diff\_1\_max

```
stats.psnr_diff_1_max
```

**Initial value:**

```
1 = np.max(  
2     np.array(df["psnr_model_1"]) - np.array(df["psnr_raw"])  
3 )
```

### 5.17.2.31 psnr\_diff\_1\_min

```
stats.psnr_diff_1_min
```

**Initial value:**

```
1 = np.min(  
2     np.array(df["psnr_model_1"]) - np.array(df["psnr_raw"])  
3 )
```

### 5.17.2.32 psnr\_diff\_2\_max

stats.psnr\_diff\_2\_max

**Initial value:**

```
1 = np.max(  
2     np.array(df["psnr_model_2"]) - np.array(df["psnr_raw"])  
3 )
```

### 5.17.2.33 psnr\_diff\_2\_min

stats.psnr\_diff\_2\_min

**Initial value:**

```
1 = np.min(  
2     np.array(df["psnr_model_2"]) - np.array(df["psnr_raw"])  
3 )
```

### 5.17.2.34 range

stats.range

### 5.17.2.35 required

stats.required

### 5.17.2.36 se\_psnr\_1

stats.se\_psnr\_1

**Initial value:**

```
1 = np.std(  
2     np.array(df['psnr_model_1']) - np.array(df['psnr_raw']), ddof=1  
3 )/np.sqrt(len(df['psnr_model_1']))
```

### 5.17.2.37 se\_psnr\_2

stats.se\_psnr\_2

**Initial value:**

```
1 = np.std(  
2     np.array(df['psnr_model_2']) - np.array(df['psnr_raw']), ddof=1  
3 )/np.sqrt(len(df['psnr_model_2']))
```

### 5.17.2.38 se\_ssim\_1

stats.se\_ssim\_1

**Initial value:**

```
1 = np.std(  
2     np.array(df['ssim_model_1']) - np.array(df['ssim_raw']), ddof=1  
3 )/np.sqrt(len(df['ssim_model_1']))
```

### 5.17.2.39 se\_ssim\_2

stats.se\_ssim\_2

**Initial value:**

```
1 = np.std(  
2     np.array(df['ssim_model_2']) - np.array(df['ssim_raw']), ddof=1  
3 )/np.sqrt(len(df['ssim_model_2']))
```

### 5.17.2.40 ssim\_cols

```
int stats.ssim_cols = 2 else df.columns[3:5]
```

### 5.17.2.41 ssim\_diff\_1\_max

stats.ssim\_diff\_1\_max

**Initial value:**

```
1 = np.max(  
2     np.array(df["ssim_model_1"]) - np.array(df["ssim_raw"])  
3 )
```

### 5.17.2.42 ssim\_diff\_1\_min

stats.ssim\_diff\_1\_min

**Initial value:**

```
1 = np.min(  
2     np.array(df["ssim_model_1"]) - np.array(df["ssim_raw"])  
3 )
```

**5.17.2.43 ssim\_diff\_2\_max**

`stats.ssim_diff_2_max`

**Initial value:**

```
1 = np.max(  
2     np.array(df["ssim_model_2"]) - np.array(df["ssim_raw"])  
3 )
```

**5.17.2.44 ssim\_diff\_2\_min**

`stats.ssim_diff_2_min`

**Initial value:**

```
1 = np.min(  
2     np.array(df["ssim_model_2"]) - np.array(df["ssim_raw"])  
3 )
```

**5.17.2.45 str**

`stats.str`

**5.17.2.46 title**

`stats.title`

**5.17.2.47 True**

`stats.True`

**5.17.2.48 type**

`stats.type`

**5.17.2.49 x**

`stats.x`

#### 5.17.2.50 xlabel

`stats.xlabel`

#### 5.17.2.51 y

`stats.y`

## 5.18 synthetic\_sim Namespace Reference

### Namespaces

- [otf](#)

## 5.19 synthetic\_sim.otf Namespace Reference

### Classes

- class [PsfParameters](#)  
*Class to store PSF parameters.*

### Functions

- def [calc\\_psf](#) (params)  
*Calculate an approximate Gibson-Lanni PSF based on the parameters provided.*

### 5.19.1 Function Documentation

#### 5.19.1.1 calc\_psf()

```
def synthetic_sim.otf.calc_psf (  
    params )
```

Calculate an approximate Gibson-Lanni PSF based on the parameters provided.

Code ported from MATLAB, original copyright Jizhou Li, 2016, The Chinese University of Hong Kong.

#### Parameters

<i>params</i>	( <a href="#">PsfParameters</a> ) - dataclass storing the PSF parameters
---------------	--

## Returns

np.ndarray representing the PSF

## 5.20 train Namespace Reference

### Functions

- def [load\\_data\\_paths](#) ([config](#), [data\\_type](#))
- def [train](#) ([train\\_loader](#), [val\\_loader](#), [optimizer](#), [scheduler](#), [net](#), [batchsize](#), [n\\_accumulations](#), [saveinterval](#), [nepoch](#), [start\\_epoch](#)=0, [losses\\_train\\_epoch](#)=[], [losses\\_val\\_epoch](#)=[], [psnr\\_train\\_epoch](#)=[], [psnr\\_val\\_epoch](#)=[], [ssim\\_train\\_epoch](#)=[], [ssim\\_val\\_epoch](#)=[])

### Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [args](#) = parser.parse\_args()
- dictionary [schema](#)
- [config](#) = json.load(f)
- int [ndim](#) = tiffimage.imread(training\_data[0]["raw"]).ndim - 1
- [input\\_shape](#) = [config](#)["input\_shape"]
- tuple [device](#)
- [ckpt\\_path](#) = None if args.model\_ckpt is None else pathlib.Path(args.model\_ckpt)
- [model](#)
- dictionary [RCAN\\_hyperparameters](#)
- [ckpt](#)
- [train\\_loader](#)
- [val\\_loader](#)
- [optimizer](#)
- [scheduler](#)
- [output\\_dir](#) = pathlib.Path(args.output\_dir)
- [parents](#)
- [True](#)
- [exist\\_ok](#)
- [n\\_accumulations](#)
- [saveinterval](#)
- [nepoch](#)
- [start\\_epoch](#)
- [losses\\_train\\_epoch](#)
- [losses\\_val\\_epoch](#)
- [psnr\\_train\\_epoch](#)
- [psnr\\_val\\_epoch](#)
- [ssim\\_train\\_epoch](#)
- [ssim\\_val\\_epoch](#)

### 5.20.1 Function Documentation



### 5.20.1.1 load\_data\_paths()

```
def train.load_data_paths (
    config,
    data_type )
```

### 5.20.1.2 train()

```
def train.train (
    train_loader,
    val_loader,
    optimizer,
    scheduler,
    net,
    batchsize,
    n_accumulations,
    saveinterval,
    nepoch,
    start_epoch = 0,
    losses_train_epoch = [],
    losses_val_epoch = [],
    psnr_train_epoch = [],
    psnr_val_epoch = [],
    ssim_train_epoch = [],
    ssim_val_epoch = [] )
```

## 5.20.2 Variable Documentation

### 5.20.2.1 args

```
train.args = parser.parse_args()
```

### 5.20.2.2 ckpt

```
train.ckpt
```

### 5.20.2.3 ckpt\_path

```
train.ckpt_path = None if args.model_ckpt is None else pathlib.Path(args.model_ckpt)
```

#### 5.20.2.4 config

```
train.config = json.load(f)
```

#### 5.20.2.5 device

```
tuple train.device
```

**Initial value:**

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```

#### 5.20.2.6 exist\_ok

```
train.exist_ok
```

#### 5.20.2.7 input\_shape

```
tuple train.input_shape = config["input_shape"]
```

#### 5.20.2.8 losses\_train\_epoch

```
train.losses_train_epoch
```

#### 5.20.2.9 losses\_val\_epoch

```
train.losses_val_epoch
```

#### 5.20.2.10 model

```
train.model
```

**Initial value:**

```
1 = RCAN(  
2     input_shape,  
3     num_input_channels=config["num_input_channels"],  
4     num_hidden_channels=config["num_hidden_channels"],  
5     num_residual_blocks=config["num_residual_blocks"],  
6     num_residual_groups=config["num_residual_groups"],  
7     channel_reduction=config["channel_reduction"],  
8     residual_scaling=1.0,  
9     num_output_channels=config["num_output_channels"],  
10 )
```

### 5.20.2.11 n\_accumulations

```
train.n_accumulations
```

### 5.20.2.12 ndim

```
int train.ndim = tiffiffle.imread(training_data[0]["raw"]).ndim - 1
```

### 5.20.2.13 nepoch

```
train.nepoch
```

### 5.20.2.14 optimizer

```
train.optimizer
```

#### Initial value:

```
1 = torch.optim.Adam(  
2     model.parameters(), lr=config["initial_learning_rate"]  
3 )
```

### 5.20.2.15 output\_dir

```
train.output_dir = pathlib.Path(args.output_dir)
```

### 5.20.2.16 parents

```
train.parents
```

### 5.20.2.17 parser

```
train.parser = argparse.ArgumentParser()
```

#### 5.20.2.18 psnr\_train\_epoch

```
train.psnr_train_epoch
```

#### 5.20.2.19 psnr\_val\_epoch

```
train.psnr_val_epoch
```

#### 5.20.2.20 RCAN\_hyperparameters

```
train.RCAN_hyperparameters
```

##### Initial value:

```
1 = {
2     "input_shape": input_shape,
3     "num_input_channels": config["num_input_channels"],
4     "num_hidden_channels": config["num_hidden_channels"],
5     "num_residual_blocks": config["num_residual_blocks"],
6     "num_residual_groups": config["num_residual_groups"],
7     "channel_reduction": config["channel_reduction"],
8     "residual_scaling": 1.0,
9     "num_output_channels": config["num_output_channels"],
10 }
```

#### 5.20.2.21 required

```
train.required
```

#### 5.20.2.22 saveinterval

```
train.saveinterval
```

#### 5.20.2.23 scheduler

```
train.scheduler
```

##### Initial value:

```
1 = torch.optim.lr_scheduler.StepLR(
2     optimizer, step_size=config["epochs"] // 4, gamma=config["lr_decay"]
3 )
```

#### 5.20.2.24 schema

dictionary train.schema

#### 5.20.2.25 ssim\_train\_epoch

train.ssim\_train\_epoch

#### 5.20.2.26 ssim\_val\_epoch

train.ssim\_val\_epoch

#### 5.20.2.27 start\_epoch

train.start\_epoch

#### 5.20.2.28 str

train.str

#### 5.20.2.29 train\_loader

train.train\_loader

##### Initial value:

```
1 = load_SIM_dataset (
2     training_data,
3     input_shape,
4     batch_size=config["batch_size"],
5     transform_function=(
6         "rotate_and_flip" if config["data_augmentation"] else None
7     ),
8     intensity_threshold=config["intensity_threshold"],
9     area_threshold=config["area_ratio_threshold"],
10    scale_factor=1,
11    steps_per_epoch=config["steps_per_epoch"],
12    p_min=config["p_min"],
13    p_max=config["p_max"],
14 )
```

### 5.20.2.30 True

`train.True`

### 5.20.2.31 type

`train.type`

### 5.20.2.32 val\_loader

`train.val_loader`

#### Initial value:

```
1 = load_SIM_dataset(  
2     validation_data,  
3     input_shape,  
4     batch_size=config["batch_size"],  
5     transform_function=(  
6         "rotate_and_flip" if config["data_augmentation"] else None  
7     ),  
8     intensity_threshold=config["intensity_threshold"],  
9     area_threshold=config["area_ratio_threshold"],  
10    scale_factor=1,  
11    steps_per_epoch=config["steps_per_epoch"],  
12    p_min=config["p_min"],  
13    p_max=config["p_max"],  
14 )
```

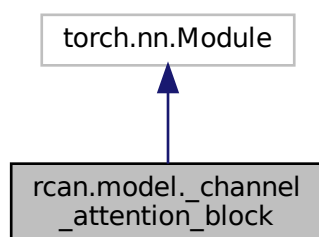
## Chapter 6

# Class Documentation

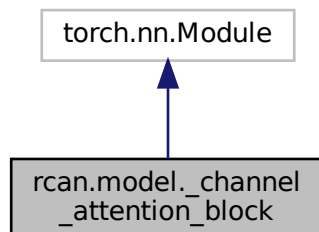
### 6.1 rcan.model.\_channel\_attention\_block Class Reference

Implements channel attention block/layer.

Inheritance diagram for rcan.model.\_channel\_attention\_block:



Collaboration diagram for rcan.model.\_channel\_attention\_block:



## Public Member Functions

- def `__init__` (self, ndim, num\_channels, reduction=16)  
*Initialises class.*
- def `forward` (self, x)  
*Forward method for class.*

## Public Attributes

- `global_average_pooling`
- `conv_1`
- `conv_2`

### 6.1.1 Detailed Description

Implements channel attention block/layer.

Instantiates a simple attention mechanism which pools all spatial information in each channel, and computes channel attention weights through a series of linear transformations and activation layers. Builds part of the architecture originally presented in [1]. Software implementation based on [2].

#### 6.1.1.1 References

[1] Image Super-Resolution Using Very Deep Residual Channel Attention Networks <https://arxiv.org/abs/1807.02758> [2] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on CAlayer from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 `__init__()`

```
def rcan.model._channel_attention_block.__init__ (
    self,
    ndim,
    num_channels,
    reduction = 16 )
```

Initialises class.

#### Parameters

<code>ndim</code>	(int) - Feature dimensionality
<code>num_channels</code>	(int) - Number of hidden channels
<code>reduction</code>	(int, optional) - Factor to reduce the number of channels by during the attention weight computation. Default: 16.



## 6.1.3 Member Function Documentation

### 6.1.3.1 forward()

```
def rcan.model._channel_attention_block.forward (
    self,
    x )
```

Forward method for class.

#### Parameters

<i>x</i>	(torch.Tensor) Input
----------	----------------------

#### Returns

torch.Tensor representing x multiplied by attention weights across channels.

## 6.1.4 Member Data Documentation

### 6.1.4.1 conv\_1

```
rcan.model._channel_attention_block.conv_1
```

### 6.1.4.2 conv\_2

```
rcan.model._channel_attention_block.conv_2
```

### 6.1.4.3 global\_average\_pooling

```
rcan.model._channel_attention_block.global_average_pooling
```

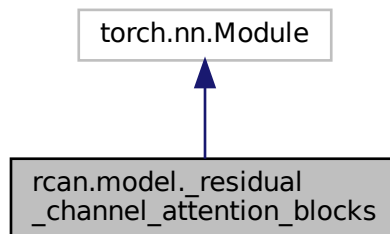
The documentation for this class was generated from the following file:

- /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/[model.py](#)

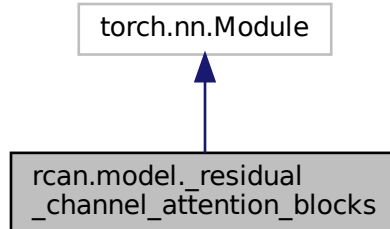
## 6.2 rcan.model.\_residual\_channel\_attention\_blocks Class Reference

Implements residual group based on [1].

Inheritance diagram for rcan.model.\_residual\_channel\_attention\_blocks:



Collaboration diagram for rcan.model.\_residual\_channel\_attention\_blocks:



### Public Member Functions

- def `__init__` (self, ndim, num\_channels, `repeat`=1, channel\_reduction=8, `residual_scaling`=1.0)  
*Initialises object.*
- def `forward` (self, x)  
*Forward method for class.*

### Public Attributes

- `repeat`
- `residual_scaling`
- `conv_list`
- `channel_attention_block_list`

## 6.2.1 Detailed Description

Implements residual group based on [1].

### 6.2.1.1 References

[1] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on ResidualGroup from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

## 6.2.2 Constructor & Destructor Documentation

### 6.2.2.1 \_\_init\_\_()

```
def rcan.model._residual_channel_attention_blocks.__init__ (
    self,
    ndim,
    num_channels,
    repeat = 1,
    channel_reduction = 8,
    residual_scaling = 1.0 )
```

Initialises object.

#### Parameters

<i>ndim</i>	(int) - Spatial dimension of input features
<i>num_channels</i>	(int) - Number of hidden channels
<i>repeat</i>	(int) - Number of residual blocks in group
<i>channel_reduction</i>	(int) - Channel reduction during attention mechanism
<i>residual_scaling</i>	(float) - output multiplier before residual connection

## 6.2.3 Member Function Documentation

### 6.2.3.1 forward()

```
def rcan.model._residual_channel_attention_blocks.forward (
    self,
    x )
```

Forward method for class.

**Parameters**

<code>x</code>	(torch.Tensor) - Input values
----------------	-------------------------------

**Returns**

torch.Tensor representing output values

## 6.2.4 Member Data Documentation

### 6.2.4.1 `channel_attention_block_list`

```
rcan.model._residual_channel_attention_blocks.channel_attention_block_list
```

### 6.2.4.2 `conv_list`

```
rcan.model._residual_channel_attention_blocks.conv_list
```

### 6.2.4.3 `repeat`

```
rcan.model._residual_channel_attention_blocks.repeat
```

### 6.2.4.4 `residual_scaling`

```
rcan.model._residual_channel_attention_blocks.residual_scaling
```

The documentation for this class was generated from the following file:

- `/home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py`

## 6.3 `rcan.data_processing.ImageStack` Class Reference

Handles creation and loading of image hyperstacks in order to make reconstructions using ImageJ easier.

## Public Member Functions

- def `__init__` (self, `dim`, `stack_number`, `stack_idx`, `sample`, `files`, `n_acq`)  
*Initialises class.*
- def `add_image` (self, `img_data`, `i`)  
*Adds an image to the initialised stack.*
- def `export_stack` (self)  
*Returns the stack.*

## Public Attributes

- `dim`
- `n_acq`
- `sample`
- `stack`
- `n_z`

### 6.3.1 Detailed Description

Handles creation and loading of image hyperstacks in order to make reconstructions using ImageJ easier.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 `__init__()`

```
def rcan.data_processing.ImageStack.__init__ (
    self,
    dim,
    stack_number,
    stack_idx,
    sample,
    files,
    n_acq )
```

Initialises class.

#### Parameters

<code>dim</code>	(int) - Dimension of images
<code>stack_number</code>	(int) - Number of images in the stack
<code>stack_idx</code>	(int) - The index of the stack within the set of stacks for the files list
<code>sample</code>	(np.ndarray) - Image from the directory which enables correct image stack shape/dtype and error catching
<code>files</code>	(list) - List of all files in directory
<code>n_acq</code>	(int) - Number of SIM acquisitions in the images

### 6.3.3 Member Function Documentation

#### 6.3.3.1 `add_image()`

```
def rcan.data_processing.ImageStack.add_image (
    self,
    img_data,
    i )
```

Adds an image to the initialised stack.

##### Parameters

<i>img_data</i>	(np.ndarray) - Image to be added
<i>i</i>	(int) - Index of the image in the stack

#### 6.3.3.2 `export_stack()`

```
def rcan.data_processing.ImageStack.export_stack (
    self )
```

Returns the stack.

##### Returns

np.ndarray

### 6.3.4 Member Data Documentation

#### 6.3.4.1 `dim`

```
rcan.data_processing.ImageStack.dim
```

#### 6.3.4.2 `n_acq`

```
rcan.data_processing.ImageStack.n_acq
```

#### 6.3.4.3 n\_z

```
rcan.data_processing.ImageStack.n_z
```

#### 6.3.4.4 sample

```
rcan.data_processing.ImageStack.sample
```

#### 6.3.4.5 stack

```
rcan.data_processing.ImageStack.stack
```

The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/rcan/data\\_processing.py](#)

## 6.4 synthetic\_sim.otf.PsfParameters Class Reference

Class to store PSF parameters.

### Static Public Attributes

- [int](#)
- [float](#)
- [Callable](#)

### 6.4.1 Detailed Description

Class to store PSF parameters.

Class to store the parameters used to evaluate an approximate Gibson-Lanni PSF. Default values are provided except for the PSF size.

### 6.4.2 Member Data Documentation

### 6.4.2.1 Callable

`synthetic_sim.otf.PsfParameters.Callable` [static]

### 6.4.2.2 float

`synthetic_sim.otf.PsfParameters.float` [static]

### 6.4.2.3 int

`synthetic_sim.otf.PsfParameters.int` [static]

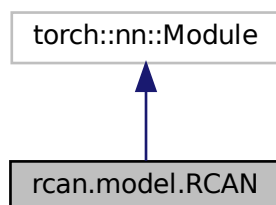
The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/synthetic\\_sim/otf.py](/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/otf.py)

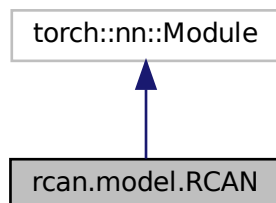
## 6.5 rcan.model.RCAN Class Reference

Builds a residual channel attention network.

Inheritance diagram for `rcan.model.RCAN`:



Collaboration diagram for `rcan.model.RCAN`:





## Public Member Functions

- `def __init__ (self, input_shape=(16, 256, 256), *num_input_channels=9, num_hidden_channels=32, num_residual_blocks=3, num_residual_groups=5, channel_reduction=8, residual_scaling=1.0, num_output_channels=-1)`  
*Initialises object.*
- `def forward (self, x)`  
*Forward method for class.*

## Public Attributes

- `num_residual_groups`
- `rcab_list`
- `conv_input`
- `conv_list`
- `conv_output`

### 6.5.1 Detailed Description

Builds a residual channel attention network.

Note that the upscale module at the end of the network is omitted so that the input and output of the model have the same size.

#### 6.5.1.1 References

[1] Image Super-Resolution Using Very Deep Residual Channel Attention Networks <https://arxiv.org/abs/1807.02758> [2] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on RCAN from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 \_\_init\_\_()

```
def rcan.model.RCAN.__init__ (
    self,
    input_shape = (16, 256, 256),
    * num_input_channels = 9,
    num_hidden_channels = 32,
    num_residual_blocks = 3,
    num_residual_groups = 5,
    channel_reduction = 8,
    residual_scaling = 1.0,
    num_output_channels = -1 )
```

Initialises object.

Builds a residual channel attention network. Note that the upscale module at the end of the network is omitted so that the input and output of the model have the same size.

**Parameters**

<i>input_shape</i>	(tuple[int]) - Input shape of the model.
<i>num_channels</i>	(int) - Number of feature channels.
<i>num_residual_blocks</i>	(int) - Number of residual channel attention blocks in each residual group.
<i>num_residual_groups</i>	(int) - Number of residual groups.
<i>channel_reduction</i>	(int) - Channel reduction ratio for channel attention.
<i>residual_scaling</i>	(float) - Scaling factor applied to the residual component in the residual channel attention block.
<i>num_output_channels</i>	(int) - Number of channels in the output image. if negative, it is set to the same number as the input.

**Returns**

torch.nn.Module PyTorch model instance.

**6.5.3 Member Function Documentation****6.5.3.1 forward()**

```
def rcan.model.RCAN.forward (
    self,
    x )
```

Forward method for class.

**Parameters**

<i>x</i>	(torch.Tensor) - Input
----------	------------------------

**Returns**

torch.Tensor Output

**6.5.4 Member Data Documentation****6.5.4.1 conv\_input**

```
rcan.model.RCAN.conv_input
```

#### 6.5.4.2 conv\_list

```
rcan.model.RCAN.conv_list
```

#### 6.5.4.3 conv\_output

```
rcan.model.RCAN.conv_output
```

#### 6.5.4.4 num\_residual\_groups

```
rcan.model.RCAN.num_residual_groups
```

#### 6.5.4.5 rcab\_list

```
rcan.model.RCAN.rcab_list
```

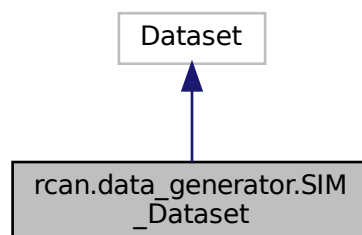
The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/rcan/model.py](/home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py)

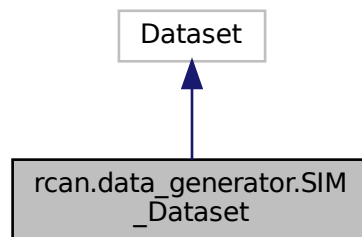
## 6.6 rcan.data\_generator.SIM\_Dataset Class Reference

Generates batches of images with real-time data augmentation.

Inheritance diagram for rcan.data\_generator.SIM\_Dataset:



Collaboration diagram for `rca.data_generator.SIM_Dataset`:



## Public Member Functions

- `def __init__ (self, images, shape, transform_function="rotate_and_flip", intensity_threshold=0.0, area_ratio_threshold=0.0, scale_factor=1, steps_per_epoch=1, p_min=2.0, p_max=99.9)`  
*Initialises object.*
- `def __getitem__ (self, j)`  
*Method used during batch loading.*
- `def __len__ (self)`

## Public Attributes

- `steps_per_epoch`
- `p_min`
- `p_max`
- `output_shape`
- `output_signature`

## Private Member Functions

- `def _scale (self, shape)`

## Private Attributes

- `_shape`
- `_transform_function`
- `_intensity_threshold`
- `_area_threshold`
- `_scale_factor`
- `_y`

### 6.6.1 Detailed Description

Generates batches of images with real-time data augmentation.

## 6.6.2 Constructor & Destructor Documentation

### 6.6.2.1 \_\_init\_\_()

```
def rcan.data_generator.SIM_Dataset.__init__ (
    self,
    images,
    shape,
    transform_function = "rotate_and_flip",
    intensity_threshold = 0.0,
    area_ratio_threshold = 0.0,
    scale_factor = 1,
    steps_per_epoch = 1,
    p_min = 2.0,
    p_max = 99.9 )
```

Initialises object.

#### Parameters

<i>images</i>	(list[dict]) - List of dictionaries of data pairs with keys ["raw","gt"]. Images in CZXY format
<i>shape</i>	(tuple[int]) - Shape of batch images excluding the channel dimension
<i>transform_function</i>	(str or callable, optional) - Function used for data augmentation. Typically you will set <code>transform_function='rotate_and_flip'</code> to apply combination of randomly selected image rotation and flipping. Alternatively, you can specify an arbitrary transformation function which takes two input images (source and target) and returns transformed images. If <code>transform_function=None</code> , no augmentation will be performed. Default: "rotate_and_flip"
<i>intensity_threshold</i>	(float, optional) - If <code>intensity_threshold &gt; 0</code> , pixels whose intensities are greater than this threshold will be considered as foreground. Default: 0.0
<i>area_ratio_threshold</i>	(float, optional) - Threshold between 0 and 1. If <code>intensity_threshold &gt; 0</code> , the generator calculates the ratio of foreground pixels in a target patch, and rejects the patch if the ratio is smaller than this threshold. Default: 0.0
<i>scale_factor</i>	(int, optional) - Scale factor for the target patch size. Positive and negative values mean up- and down-scaling respectively. Default: 1
<i>steps_per_epoch</i>	(int, optional) - Determines how many times each image is used to generate a patch per batch. Default: 1
<i>p_min</i>	(float, optional) - Minimum percentile used for scaling. Default: 2.0
<i>p_max</i>	(float, optional) - Maximum percentile used for scaling. Default: 99.9

## 6.6.3 Member Function Documentation

### 6.6.3.1 \_\_getitem\_\_()

```
def rcan.data_generator.SIM_Dataset.__getitem__ (
    self,
    j )
```

Method used during batch loading.

Standardises pixel values and takes patches from the image pair. Also implements the rejection of patches based on area/intensity threshold, if `self._intensity_threshold > 0`. Augments data pair.

#### Parameters

<i>j</i>	(int) - Index of data to be loaded. Note that if <code>self.steps_per_epoch &gt; 1</code> , this can be more than the dataset size, in which case it is interpreted modulo the dataset size.
----------	--

#### Returns

tuple(torch.Tensor) raw-gt image pair

#### 6.6.3.2 `__len__()`

```
def rcan.data_generator.SIM_Dataset.__len__ (
    self )
```

#### 6.6.3.3 `_scale()`

```
def rcan.data_generator.SIM_Dataset._scale (
    self,
    shape ) [private]
```

### 6.6.4 Member Data Documentation

#### 6.6.4.1 `_area_threshold`

```
rcan.data_generator.SIM_Dataset._area_threshold [private]
```

#### 6.6.4.2 `_intensity_threshold`

```
rcan.data_generator.SIM_Dataset._intensity_threshold [private]
```

#### 6.6.4.3 `_scale_factor`

`rcan.data_generator.SIM_Dataset._scale_factor` [private]

#### 6.6.4.4 `_shape`

`rcan.data_generator.SIM_Dataset._shape` [private]

#### 6.6.4.5 `_transform_function`

`rcan.data_generator.SIM_Dataset._transform_function` [private]

#### 6.6.4.6 `_y`

`rcan.data_generator.SIM_Dataset._y` [private]

#### 6.6.4.7 `output_shape`

`rcan.data_generator.SIM_Dataset.output_shape`

#### 6.6.4.8 `output_signature`

`rcan.data_generator.SIM_Dataset.output_signature`

#### 6.6.4.9 `p_max`

`rcan.data_generator.SIM_Dataset.p_max`

#### 6.6.4.10 `p_min`

`rcan.data_generator.SIM_Dataset.p_min`

#### 6.6.4.11 steps\_per\_epoch

`rca.data_generator.SIM_Dataset.steps_per_epoch`

The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/rca/data\\_generator.py](#)

## 6.7 generate\_sim.SimulationRunner Class Reference

Class which performs a batch of simulations, either sequentially or in parallel.

### Public Member Functions

- `def __init__ (self, input\_dir, output\_dir, index_range, z\_offset)`
- `def do\_sim (self, i, sim, vol)`  
*Creates a new random virtual microscope simulator, takes a new sample from the VHP dataset, runs the simulation on the sample, and saves the results, along with the ground truth, in a single TIFF file.*
- `def run (self)`  
*Runs a series of simulations sequentially.*

### Public Attributes

- [input\\_dir](#)
- [input\\_files](#)
- [output\\_dir](#)
- [range](#)
- [z\\_offset](#)

#### 6.7.1 Detailed Description

Class which performs a batch of simulations, either sequentially or in parallel.

#### 6.7.2 Constructor & Destructor Documentation

##### 6.7.2.1 \_\_init\_\_()

```
def generate_sim.SimulationRunner.__init__ (
    self,
    input_dir,
    output_dir,
    index_range,
    z_offset )
```



## 6.7.3 Member Function Documentation

### 6.7.3.1 do\_sim()

```
def generate_sim.SimulationRunner.do_sim (
    self,
    i,
    sim,
    vol )
```

Creates a new random virtual microscope simulator, takes a new sample from the VHP dataset, runs the simulation on the sample, and saves the results, along with the ground truth, in a single TIFF file.

The parameters are saved in an accompanying JSON file.

### 6.7.3.2 run()

```
def generate_sim.SimulationRunner.run (
    self )
```

Runs a series of simulations sequentially.

## 6.7.4 Member Data Documentation

### 6.7.4.1 input\_dir

```
generate_sim.SimulationRunner.input_dir
```

### 6.7.4.2 input\_files

```
generate_sim.SimulationRunner.input_files
```

### 6.7.4.3 output\_dir

```
generate_sim.SimulationRunner.output_dir
```

#### 6.7.4.4 range

`generate_sim.SimulationRunner.range`

#### 6.7.4.5 z\_offset

`generate_sim.SimulationRunner.z_offset`

The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/generate\\_sim.py](#)

## 6.8 generate\_sim.Simulator Class Reference

The [Simulator](#) class encapsulates the state of a 3D microscope simulation.

### Public Member Functions

- `def __init__ (self, **kwargs)`
- `def randomise (self)`
- `def params_dict (self)`
- `def psf_params (self)`
- `def wavevectors (self)`  
*Calculates wavevectors inside the sample for the three beams, for a given number of rotations of those beams.*
- `def illumination (self)`  
*Calculates the illumination intensity in the sample; returns ndarray of shape (n\_rotations, n\_shifts, n\_x, n\_x, n\_z)*
- `def in_focus_plane (self, sample)`  
*Returns the designated 'ground truth' plane.*
- `def psf (self)`  
*Calculates a PSF if it has not been done already.*
- `def simulate_sim (self, sample)`  
*Calculates the 15 simulated SIM images for a given sample.*
- `def simulate_ideal_superres (self, sample)`  
*Simulates the best-case scenario for a 3D SIM reconstruction, by convolving the in-focus plane with a small PSF.*
- `def add_noise (self, image)`  
*Adds a combination of Gaussian and Poissonian noise to the image.*

## Public Attributes

- [n\\_shifts](#)
- [n\\_angles](#)
- [n\\_x](#)
- [n\\_z](#)
- [n\\_rotations](#)
- [res\\_axial](#)
- [res\\_lateral](#)
- [delta\\_z\\_p](#)
- [n\\_sample](#)
- [n\\_i](#)
- [n\\_g](#)
- [z](#)
- [z\\_p](#)
- [angle\\_error](#)
- [poisson\\_photons](#)
- [signal\\_to\\_noise](#)
- [lambda0](#)
- [k0](#)
- [lambda\\_exc](#)
- [k\\_exc](#)
- [beam\\_position](#)

## Private Attributes

- [\\_psf](#)
- [\\_superres\\_psf](#)
- [\\_illumination](#)

### 6.8.1 Detailed Description

The [Simulator](#) class encapsulates the state of a 3D microscope simulation.

A single instance of this class corresponds to a specific set of microscope parameters. These parameters are randomly chosen upon object creation.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 `__init__()`

```
def generate_sim.Simulator.__init__ (
    self,
    ** kwargs )
```

## 6.8.3 Member Function Documentation

### 6.8.3.1 add\_noise()

```
def generate_sim.Simulator.add_noise (
    self,
    image )
```

Adds a combination of Gaussian and Poissonian noise to the image.

### 6.8.3.2 illumination()

```
def generate_sim.Simulator.illumination (
    self )
```

Calculates the illumination intensity in the sample; returns ndarray of shape (n\_rotations, n\_shifts, n\_x, n\_x, n\_z)

### 6.8.3.3 in\_focus\_plane()

```
def generate_sim.Simulator.in_focus_plane (
    self,
    sample )
```

Returns the designated 'ground truth' plane.

### 6.8.3.4 params\_dict()

```
def generate_sim.Simulator.params_dict (
    self )
```

### 6.8.3.5 psf()

```
def generate_sim.Simulator.psf (
    self )
```

Calculates a PSF if it has not been done already.

#### 6.8.3.6 psf\_params()

```
def generate_sim.Simulator.psf_params (
    self )
```

#### 6.8.3.7 randomise()

```
def generate_sim.Simulator.randomise (
    self )
```

#### 6.8.3.8 simulate\_ideal\_superres()

```
def generate_sim.Simulator.simulate_ideal_superres (
    self,
    sample )
```

Simulates the best-case scenario for a 3D SIM reconstruction, by convolving the in-focus plane with a small PSF.

#### 6.8.3.9 simulate\_sim()

```
def generate_sim.Simulator.simulate_sim (
    self,
    sample )
```

Calculates the 15 simulated SIM images for a given sample.

#### 6.8.3.10 wavevectors()

```
def generate_sim.Simulator.wavevectors (
    self )
```

Calculates wavevectors inside the sample for the three beams, for a given number of rotations of those beams.

Returns ndarray of shape (n\_rotations, n\_beams, 3), where n\_beams = 3

### 6.8.4 Member Data Documentation

#### 6.8.4.1 `_illumination`

`generate_sim.Simulator._illumination` [private]

#### 6.8.4.2 `_psf`

`generate_sim.Simulator._psf` [private]

#### 6.8.4.3 `_superres_psf`

`generate_sim.Simulator._superres_psf` [private]

#### 6.8.4.4 `angle_error`

`generate_sim.Simulator.angle_error`

#### 6.8.4.5 `beam_position`

`generate_sim.Simulator.beam_position`

#### 6.8.4.6 `delta_z_p`

`generate_sim.Simulator.delta_z_p`

#### 6.8.4.7 `k0`

`generate_sim.Simulator.k0`

#### 6.8.4.8 `k_exc`

`generate_sim.Simulator.k_exc`

#### 6.8.4.9 lambda0

`generate_sim.Simulator.lambda0`

#### 6.8.4.10 lambda\_exc

`generate_sim.Simulator.lambda_exc`

#### 6.8.4.11 n\_angles

`generate_sim.Simulator.n_angles`

#### 6.8.4.12 n\_g

`generate_sim.Simulator.n_g`

#### 6.8.4.13 n\_i

`generate_sim.Simulator.n_i`

#### 6.8.4.14 n\_rotations

`generate_sim.Simulator.n_rotations`

#### 6.8.4.15 n\_sample

`generate_sim.Simulator.n_sample`

#### 6.8.4.16 n\_shifts

`generate_sim.Simulator.n_shifts`

**6.8.4.17 n\_x**

`generate_sim.Simulator.n_x`

**6.8.4.18 n\_z**

`generate_sim.Simulator.n_z`

**6.8.4.19 poisson\_photons**

`generate_sim.Simulator.poisson_photons`

**6.8.4.20 res\_axial**

`generate_sim.Simulator.res_axial`

**6.8.4.21 res\_lateral**

`generate_sim.Simulator.res_lateral`

**6.8.4.22 signal\_to\_noise**

`generate_sim.Simulator.signal_to_noise`

**6.8.4.23 z**

`generate_sim.Simulator.z`

**6.8.4.24 z\_p**

`generate_sim.Simulator.z_p`

The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim\\_project/jh2284/src/generate\\_sim.py](#)



## Chapter 7

# File Documentation

### 7.1 /home/jhughes2712/projects/sim\_project/jh2284/src/analyse.py File Reference

Script producing plots and small datasets that summarise the performance of models.

#### Namespaces

- [analyse](#)

#### Variables

- [analyse.parser](#) = argparse.ArgumentParser()
- [analyse.type](#)
- [analyse.str](#)
- [analyse.required](#)
- [analyse.default](#)
- [analyse.int](#)
- [analyse.action](#)
- [analyse.args](#) = parser.parse\_args()
- [analyse.output\\_dir](#) = pathlib.Path(args.output\_dir)
- [analyse.parents](#)
- [analyse.True](#)
- [analyse.exist\\_ok](#)
- tuple [analyse.device](#)
- [analyse.ckpt](#)
- [analyse.model](#)
- [analyse.gt\\_dir](#) = pathlib.Path(args.gt\_dir)
- [analyse.raw\\_dir](#) = pathlib.Path(args.raw\_dir)
- [analyse.model\\_1\\_dir](#) = pathlib.Path(args.model\_1\_dir)
- [analyse.gt\\_files](#) = sorted(list(gt\_dir.glob(args.glob\_str)))
- [analyse.raw\\_files](#) = sorted(list(raw\_dir.glob(args.glob\_str)))
- [analyse.model\\_1\\_files](#) = sorted(list(model\_1\_dir.glob(args.glob\_str)))
- [analyse.model\\_2\\_dir](#) = pathlib.Path(args.model\_2\_dir)
- [analyse.model\\_2\\_files](#) = sorted(list(model\_2\_dir.glob(args.glob\_str)))
- [analyse.N](#) = len(gt\_files)

- `analyse.psnr` = PSNR(data\_range=65536, device=device)
- `analyse.ssim`
- `analyse.df`
- `analyse.gt` = `reshape_to_bcwh(tifffile.imread(gt_files[i]))`
- `analyse.raw` = `reshape_to_bcwh(tifffile.imread(raw_files[i]))`
- `analyse.model_1` = `reshape_to_bcwh(tifffile.imread(model_1_files[i]))`
- `analyse.model_2` = `reshape_to_bcwh(tifffile.imread(model_2_files[i]))`
- `analyse.rng` = `np.random.default_rng(seed=31052024)`
- `analyse.img_idx` = `list(range(N))`
- list `analyse.gt_samples` = `[np.squeeze(tifffile.imread(gt_files[i])) for i in img_idx]`
- list `analyse.raw_samples` = `[np.squeeze(tifffile.imread(raw_files[i])) for i in img_idx]`
- list `analyse.model_1_samples`
- list `analyse.model_2_samples`
- `analyse.cmap`

### 7.1.1 Detailed Description

Script producing plots and small datasets that summarise the performance of models.

This script reads directories of reconstructed images, and compares raw versus model reconstructions versus ground truth. The script then produces summary statistics, saves relevant metrics to a .csv file, and produces samples of cropped image regions for comparison.

Arguments:

- g: directory path for ground-truth images
- r: directory path for raw images
- a: directory path for model-1-restored images
- b: directory path for model-2-restored images
- o: output directory for analysis plots, default "figures/"
- x: filepath for model 1 checkpoint (plots learning curve)
- y: filepath for model 2 checkpoint (plots learning curve)
- s: globbing string, to analyse a subset of images
- n: number of sample crops to display, default 0.
- p: plot only mode, skips data analysis

## 7.2 /home/jhughes2712/projects/sim\_project/jh2284/src/apply.py File Reference

Script producing restored images resulting from an RCAN denoiser being applied to low SNR images.

### Namespaces

- `apply`

## Variables

- `apply.parser` = `argparse.ArgumentParser()`
- `apply.type`
- `apply.str`
- `apply.required`
- `apply.int`
- `apply.choices`
- `apply.default`
- `apply.percentile`
- `apply.action`
- `apply.args` = `parser.parse_args()`
- `apply.input_path` = `pathlib.Path(args.input)`
- `apply.output_path` = `pathlib.Path(args.output)`
- `apply.parents`
- `apply.raw_files` = `sorted(input_path.glob("*.tif"))`
- `apply.data` = `itertools.zip_longest(raw_files, [])`
- tuple `apply.device`
- `apply.ckpt`
- `apply.model`
- `apply.RCAN_hyperparameters` = `ckpt["hyperparameters"]`
- list `apply.overlap_shape`
- `apply.raw` = `normalize(tifffile.imread(raw_file), args.p_min, args.p_max)`
- `apply.restored`
- `apply.output_file` = `output_path / ("pred_" + raw_file.name)`
- `apply.imagej`

### 7.2.1 Detailed Description

Script producing restored images resulting from an RCAN denoiser being applied to low SNR images.

This script takes directories of raw images, and a model checkpoint file, and applies the model to the image in a patched fashion. The details of this patching, and the output datatype, can be configured.

Arguments:

- m: model checkpoint filepath
- i: low SNR image directory path
- o: output directory path
- b: specifies pixel bit depth to save for output (8 or 16)
- O: block overlap shape (by default input\_shape / 8)
- p\_min: input normalization parameter, percentile maps to zero
- p\_max: input normalization parameter, percentile maps to one
- normalize\_output\_range\_between\_zero\_and\_one: scaling for output

Adapted from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/apply.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

## 7.3 `/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_↔to_czxy.py` File Reference

Script enabling .tif file conversion between OMX and CZXY.

### Namespaces

- [convert\\_omx\\_to\\_czxy](#)

### Variables

- [convert\\_omx\\_to\\_czxy.parser](#) = `argparse.ArgumentParser()`
- [convert\\_omx\\_to\\_czxy.type](#)
- [convert\\_omx\\_to\\_czxy.str](#)
- [convert\\_omx\\_to\\_czxy.required](#)
- [convert\\_omx\\_to\\_czxy.int](#)
- [convert\\_omx\\_to\\_czxy.action](#)
- [convert\\_omx\\_to\\_czxy.args](#) = `parser.parse_args()`
- [convert\\_omx\\_to\\_czxy.input\\_dir](#) = `pathlib.Path(args.input)`
- [convert\\_omx\\_to\\_czxy.input\\_files](#) = `sorted(input_dir.rglob("*.tif"))`
- [convert\\_omx\\_to\\_czxy.original](#) = `tiffimage.imread(input_file)`
- [convert\\_omx\\_to\\_czxy.converted](#)
- [convert\\_omx\\_to\\_czxy.imagej](#)

### 7.3.1 Detailed Description

Script enabling .tif file conversion between OMX and CZXY.

This script takes directories of image volumes as input, and converts, in place, between the OMX and CZXY formats (in either direction). In the OMX format, the first dimension is of size `n_phases` x `n_z` x `n_angles`; moving along this dimension, the phase changes first, then the z-value, then the angle. The CZXY format is the same, but the z-dimension of the image is separated into the 2nd dimension, so that the first dimension is just `n_phases` x `n_angles`.

Arguments:

- `i`: image directory
- `p`: number of phases
- `a`: number of angles
- `b`: specifies conversion - if not used it will be OMX to CZXY, the `b` flag reverses this direction.

## 7.4 `/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_↔to_paz.py` File Reference

Script enabling .tif file conversion between OMX and PAZ.

## Namespaces

- [convert\\_omx\\_to\\_paz](#)

## Variables

- [convert\\_omx\\_to\\_paz.parser](#) = argparse.ArgumentParser()
- [convert\\_omx\\_to\\_paz.type](#)
- [convert\\_omx\\_to\\_paz.str](#)
- [convert\\_omx\\_to\\_paz.required](#)
- [convert\\_omx\\_to\\_paz.int](#)
- [convert\\_omx\\_to\\_paz.action](#)
- [convert\\_omx\\_to\\_paz.args](#) = parser.parse\_args()
- [convert\\_omx\\_to\\_paz.input\\_dir](#) = pathlib.Path(args.input)
- [convert\\_omx\\_to\\_paz.input\\_files](#) = sorted(input\_dir.rglob("\*.tif"))
- [convert\\_omx\\_to\\_paz.original](#) = tifffile.imread(input\_file)
- [convert\\_omx\\_to\\_paz.converted](#) = conv\_omx\_to\_paz(original, args.num\_phases, args.num\_angles)
- [convert\\_omx\\_to\\_paz.imagej](#)

### 7.4.1 Detailed Description

Script enabling .tif file conversion between OMX and PAZ.

This script takes directories of image volumes as input, and converts, in place, between the OMX and PAZ formats (in either direction). In the OMX format, the first dimension is of size `n_phases` x `n_z` x `n_angles`; moving along this dimension, the phase changes first, then the z-value, then the angle. The PAZ format is the same except the order is changed so that z-values and angels are swapped.

Arguments:

- i: image directory
- p: number of phases
- a: number of angles
- b: specifies conversion - if not used it will be OMX to PAZ, the b flag reverses this direction.

## 7.5 /home/jhughes2712/projects/sim\_project/jh2284/src/convert\_slices\_to\_volumes.py File Reference ↩

Script enabling construction of 3D image volumes from large RGB 2D image slices.

## Namespaces

- [convert\\_slices\\_to\\_volumes](#)

## Variables

- `convert_slices_to_volumes.parser` = `argparse.ArgumentParser()`
- `convert_slices_to_volumes.type`
- `convert_slices_to_volumes.str`
- `convert_slices_to_volumes.required`
- `convert_slices_to_volumes.tuple_of_ints`
- `convert_slices_to_volumes.default`
- `convert_slices_to_volumes.args` = `parser.parse_args()`
- `convert_slices_to_volumes.input_dir` = `pathlib.Path(args.input)`
- `convert_slices_to_volumes.output_dir` = `pathlib.Path(args.output)`
- `convert_slices_to_volumes.input_files` = `sorted(input_dir.glob("*.tif"))`
- `convert_slices_to_volumes.parents`
- `convert_slices_to_volumes.True`
- `convert_slices_to_volumes.exist_ok`
- `convert_slices_to_volumes.volume` = `np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)`
- `convert_slices_to_volumes.input_slice` = `tifffile.imread(file)`
- `convert_slices_to_volumes.output_file` = `output_dir / filename`
- `convert_slices_to_volumes.subvolume`
- `convert_slices_to_volumes.imagej`

### 7.5.1 Detailed Description

Script enabling construction of 3D image volumes from large RGB 2D image slices.

Takes a directory of 2D image slices as input, and converts to 3D volumes. The 2D images are assumed to be ordered z-axially; the number of images is the number of voxels in the z-direction of the 3D volumes. The lateral cross-sections of the 3D images are determined by script arguments. Saves in uint16 depth.

Arguments:

- i: directory path for 2D images
- o: directory path for 3D image volumes
- s: start pixel coordinates (x, y)
- j: crop size for image volume (crop\_x, crop\_y)
- n: number of crops to take in each direction (steps\_x, steps\_y)
- l: filename prefix, default "volume"

## 7.6 /home/jhughes2712/projects/sim\_project/jh2284/src/generate\_sim.py File Reference

Script simulating the acquisition of 3D SIM image volumes.

## Classes

- class `generate_sim.Simulator`  
*The `Simulator` class encapsulates the state of a 3D microscope simulation.*
- class `generate_sim.SimulationRunner`  
*Class which performs a batch of simulations, either sequentially or in parallel.*

## Namespaces

- [generate\\_sim](#)

## Functions

- def [generate\\_sim.arange\\_zero](#) (n, spacing=1)
- def [generate\\_sim.threshold\\_norm](#) (sample)  
*Applies a threshold and normalises the sample to improve contrast.*

## Variables

- [generate\\_sim.parser](#) = argparse.ArgumentParser()
- [generate\\_sim.type](#)
- [generate\\_sim.str](#)
- [generate\\_sim.required](#)
- [generate\\_sim.int](#)
- [generate\\_sim.default](#)
- [generate\\_sim.args](#) = parser.parse\_args()
- [generate\\_sim.runner](#)

### 7.6.1 Detailed Description

Script simulating the acquisition of 3D SIM image volumes.

Takes a directory of 3D image volumes as input, and produces synthetic 3-beam SIM volumes of size (15, 32, 256, 256).

Arguments:

- i: directory path of input volumes
- o: directory path of output volumes
- s: start index of sorted input files to process
- e: end index of sorted input files to process
- z: z\_offset, used to specify the region of the input volume to use.

## 7.7 /home/jhughes2712/projects/sim\_project/jh2284/src/image\_noising.py File Reference

Script which converts a directory of high-SNR SIM images into a training dataset.

## Namespaces

- [image\\_noising](#)

## Functions

- `def image_noising.save_image_pair` (gt\_img, split, name, channel\_idx)

## Variables

- `image_noising.parser` = `argparse.ArgumentParser()`
- `image_noising.type`
- `image_noising.str`
- `image_noising.required`
- `image_noising.int`
- `image_noising.choices`
- `image_noising.float`
- `image_noising.default`
- `image_noising.args` = `parser.parse_args()`
- `image_noising.input_path` = `pathlib.Path(args.input)`
- `image_noising.output_path` = `pathlib.Path(args.output)`
- `image_noising.parents`
- `image_noising.output_train_gt_path` = `output_path.joinpath("Training", "GT")`
- `image_noising.output_train_raw_path` = `output_path.joinpath("Training", "Raw")`
- `image_noising.output_val_gt_path` = `output_path.joinpath("Validation", "GT")`
- `image_noising.output_val_raw_path` = `output_path.joinpath("Validation", "Raw")`
- `image_noising.output_test_gt_path` = `output_path.joinpath("Testing", "GT")`
- `image_noising.output_test_raw_path` = `output_path.joinpath("Testing", "Raw")`
- `image_noising.data` = `sorted(input_path.glob("*.tif"))`
- `image_noising.n_acquisitions` = `tifffile.imread(data[0]).shape[0] // args.channels`
- `image_noising.n_img` = `len(data)`
- `image_noising.train_size` = `int((1 - args.test_fraction) * n_img)`
- `image_noising.val_size` = `int(args.val_fraction * train_size)`
- `image_noising.rng` = `np.random.default_rng(seed=25042024)`
- `image_noising.img_idx_all` = `list(range(n_img))`
- `image_noising.img_idx_test` = `img_idx_all[train_size:]`
- `image_noising.img_idx_train` = `img_idx_all[: train_size - val_size]`
- `image_noising.img_idx_val` = `img_idx_all[train_size - val_size : train_size]`
- `image_noising.gt` = `tifffile.imread(img_file)`
- string `image_noising.split` = "train"

### 7.7.1 Detailed Description

Script which converts a directory of high-SNR SIM images into a training dataset.

Each image is duplicated so that a low SNR counterpart is produced, simulating the same sample imaged with a lower illumination intensity. The data is then randomly split into train, validation, and testing subsets.

Arguments:

- i: directory path of input image
- o: directory path of output
- d: dimension
- s: scale factor used to simulate the low SNR images.
- tf: the fraction of the full dataset used for the hold-out test set.
- vf: the fraction of the *training* dataset that is reserved for validation during training.



## 7.8 /home/jhughes2712/projects/sim\_project/jh2284/src/manage\_stack.py File Reference

Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions.

### Namespaces

- [manage\\_stack](#)

### Variables

- [manage\\_stack.parser](#) = argparse.ArgumentParser()
- [manage\\_stack.type](#)
- [manage\\_stack.str](#)
- [manage\\_stack.required](#)
- [manage\\_stack.int](#)
- [manage\\_stack.choices](#)
- [manage\\_stack.default](#)
- [manage\\_stack.action](#)
- [manage\\_stack.args](#) = parser.parse\_args()
- [manage\\_stack.output\\_dir](#) = pathlib.Path(args.output\_dir)
- [manage\\_stack.parents](#)
- [manage\\_stack.True](#)
- [manage\\_stack.exist\\_ok](#)
- [manage\\_stack.files](#) = sorted(list(pathlib.Path(args.input\_dir).glob(args.glob\_str)))
- [int manage\\_stack.stack\\_number](#) = -1 else args.stack\_number
- [int manage\\_stack.number\\_of\\_stacks](#) = len(files) // stack\_number
- [manage\\_stack.sample](#) = tifffile.imread(files[0])
- [manage\\_stack.stack\\_handler](#)
- [manage\\_stack.img\\_data](#) = tifffile.imread(input\_file)
- [tuple manage\\_stack.filename](#)
- [tuple manage\\_stack.output\\_file](#) = output\_dir / filename
- [manage\\_stack.output\\_data](#)

### 7.8.1 Detailed Description

Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions.

Takes a directory of images as input, and either stacks or unstacks the images there according to the configuration. 3D Image Volumes are expected to be in PAZ format. Note in unstack mode, images are saved with a first dimension of length 1 - this is the correct format for training the second step models (CZXY).

Arguments:

- i: directory path of input images
- o: directory path of output images
- n: output image name prefix - only applies in 'stack' mode
- d: dimension

- q: number of SIM acquisitions per image
- g: glob string used to choose images from input directory
- u: if used, sets mode to 'unstack'
- s: start index of sorted input files to process
- e: end index of sorted input files to process
- t: number of images to stack together - only applies in 'stack' mode. Default: -1 (all images are stacked)
- z: number of z slices of images - only applies in 'unstack' mode

## 7.9 /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/\_\_init\_\_.py File Reference

### Namespaces

- [rcan](#)

## 7.10 /home/jhughes2712/projects/sim\_project/jh2284/src/synthetic\_sim/\_\_init\_\_.py File Reference

### Namespaces

- [synthetic\\_sim](#)

## 7.11 /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/data\_generator.py File Reference

Module that handles processing and batching of data during training loop.

### Classes

- class [rcan.data\\_generator.SIM\\_Dataset](#)  
*Generates batches of images with real-time data augmentation.*

### Namespaces

- [rcan.data\\_generator](#)

### Functions

- def [rcan.data\\_generator.load\\_SIM\\_dataset](#) (images, shape, batch\_size, transform\_function, intensity\_threshold, area\_threshold, scale\_factor, steps\_per\_epoch, p\_min, p\_max)  
*Wraps [SIM\\_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.*

### 7.11.1 Detailed Description

Module that handles processing and batching of data during training loop.

This module primarily defines the SIM\_Dataset class which handles image cropping, normalization, augmentation, and intensity-threshold-area based rejection.

Migrated from [https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/data\\_generator.py](https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/data_generator.py)

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

## 7.12 /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/data\_processing.py File Reference

Contains tools used to pre-process image data.

### Classes

- class [rcan.data\\_processing.ImageStack](#)  
*Handles creation and loading of image hyperstacks in order to make reconstructions using ImageJ easier.*

### Namespaces

- [rcan.data\\_processing](#)

### Functions

- def [rcan.data\\_processing.crop\\_volume](#) (volume, num\_steps, start, step, label)  
*Takes an image volume and divides part of it into smaller volumes by cropping lateral sections (the full z dimension is used).*
- def [rcan.data\\_processing.conv\\_omx\\_to\\_czxy](#) (original, n\_phases, n\_angles)  
*Converts image array from OMX (PZA format) to CZXY format.*
- def [rcan.data\\_processing.conv\\_czxy\\_to\\_omx](#) (original, n\_phases, n\_angles)  
*Converts image array from CZXY to OMX format.*
- def [rcan.data\\_processing.conv\\_omx\\_to\\_paz](#) (original, n\_phases, n\_angles)  
*Converts image array from OMX (PZA format) to PAZ format.*
- def [rcan.data\\_processing.conv\\_paz\\_to\\_omx](#) (original, n\_phases, n\_angles)  
*Converts image array from PAZ to OMX(PZA) format.*

### 7.12.1 Detailed Description

Contains tools used to pre-process image data.

## 7.13 /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/model.py File Reference

Module defining the RCAN model architecture.

### Classes

- class [rcan.model.\\_channel\\_attention\\_block](#)  
*Implements channel attention block/layer.*
- class [rcan.model.\\_residual\\_channel\\_attention\\_blocks](#)  
*Implements residual group based on [1].*
- class [rcan.model.RCAN](#)  
*Builds a residual channel attention network.*

### Namespaces

- [rcan.model](#)

### Functions

- def [rcan.model.\\_conv](#) (ndim, in\_filters, out\_filters, kernel\_size, padding="same", \*\*kwargs)  
*Returns the appropriate torch.nn convolution layer based on parameters.*
- def [rcan.model.\\_global\\_average\\_pooling](#) (ndim)  
*Returns the appropriate torch.nn pooling layer based on parameters.*
- def [rcan.model.\\_standardize](#) (x)  
*Standardises input data.*
- def [rcan.model.\\_destandardize](#) (x)  
*Inverse of \_standardize.*

#### 7.13.1 Detailed Description

Module defining the RCAN model architecture.

Module that defines a number of classes inheriting from nn.Module, implementing different levels of the RCAN architecture. This includes the channel attention layer, residual channel attention block, and RCAN itself.

Migrated from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/model.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

## 7.14 /home/jhughes2712/projects/sim\_↵ project/jh2284/src/rcan/plotting.py File Reference

Module providing helper functions for matplotlib plots.

## Namespaces

- [rcan.plotting](#)

## Functions

- def [rcan.plotting.plot\\_learning\\_curve](#) (losses\_train, losses\_val, psnr\_train, psnr\_val, ssim\_train, ssim\_val, fig-size, output\_path)  
*Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.*
- def [rcan.plotting.compute\\_metrics](#) (img, gt\_img, psnr, ssim)  
*Uses ignite metric objects to compute PSNR and SSIM.*
- def [rcan.plotting.plot\\_reconstructions](#) (device, output\_path, dim, gt\_imgs, raw\_imgs, model\_1\_imgs, model\_2\_imgs=None, cmap="inferno")  
*Plots a sample of reconstructions comparing GT vs Raw vs Restored.*

### 7.14.1 Detailed Description

Module providing helper functions for matplotlib plots.

Provides tools to assist with analysis of trained networks, including samples of restored reconstructions, metrics, and model progress during training.

## 7.15 /home/jhughes2712/projects/sim\_project/jh2284/src/rcan/utils.py File Reference

Contains utility functions for the training loop and inference.

## Namespaces

- [rcan.utils](#)

## Functions

- def [rcan.utils.normalize](#) (image, p\_min=2, p\_max=99.9, dtype="float32")  
*Normalizes the image intensity so that the  $p_{min}$ -th and the  $p_{max}$ -th percentiles are converted to 0 and 1 respectively.*
- def [rcan.utils.apply](#) (model, data, model\_input\_image\_shape, model\_output\_image\_shape, num\_input\_channels, num\_output\_channels, batch\_size, device, overlap\_shape=None, verbose=False)  
*Applies a model to an input image.*
- def [rcan.utils.load\\_rcan\\_checkpoint](#) (ckpt\_path, device)  
*Enables loading of RCAN checkpointed model.*
- def [rcan.utils.tuple\\_of\\_ints](#) (string)  
*Defines behaviour of parsing tuples of ints (argparse).*
- def [rcan.utils.percentile](#) (x)  
*Defines behaviour of parsing percentiles (argparse).*
- def [rcan.utils.reshape\\_to\\_bcwh](#) (data)  
*Reshapes 2D or 3D array to have batch x channel x width x height format, by prepending extra dimensions.*
- def [rcan.utils.normalize\\_between\\_zero\\_and\\_one](#) (data)  
*Coerce pixel values to [0, 1] range.*

### 7.15.1 Detailed Description

Contains utility functions for the training loop and inference.

Migrated from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/utils.py>

Copyright 2021 SVision Technologies LLC. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

## 7.16 `/home/jhughes2712/projects/sim_project/jh2284/src/recon_↵ postprocess.py` File Reference

Script handling the postprocessing of SIM reconstructions.

### Namespaces

- `recon_postprocess`

### Variables

- `recon_postprocess.parser` = `argparse.ArgumentParser()`
- `recon_postprocess.type`
- `recon_postprocess.str`
- `recon_postprocess.required`
- `recon_postprocess.args` = `parser.parse_args()`
- `recon_postprocess.files` = `sorted(list(pathlib.Path(args.input_dir).rglob("*.tif")))`
- `recon_postprocess.img_data` = `tiffimage.imread(input_file)`

### 7.16.1 Detailed Description

Script handling the postprocessing of SIM reconstructions.

Takes a directory of images as input, clips zero values, and scales to the full 16-bit depth range. Operates in-place.

Arguments:

- `i`: directory path of input images

## 7.17 `/home/jhughes2712/projects/sim_project/jh2284/src/recon_↵ preprocess.py` File Reference

Script handling the preprocessing of images before SIM reconstruction.

## Namespaces

- [recon\\_preprocess](#)

## Functions

- `def recon_preprocess.normalize_acquisition_intensity (data, dim)`

## Variables

- `recon_preprocess.parser = argparse.ArgumentParser()`
- `recon_preprocess.type`
- `recon_preprocess.str`
- `recon_preprocess.required`
- `recon_preprocess.int`
- `recon_preprocess.choices`
- `recon_preprocess.percentile`
- `recon_preprocess.default`
- `recon_preprocess.action`
- `recon_preprocess.args = parser.parse_args()`
- `recon_preprocess.output_dir = pathlib.Path(args.output_dir)`
- `recon_preprocess.parents`
- `recon_preprocess.True`
- `recon_preprocess.exist_ok`
- `recon_preprocess.files = sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))`
- `recon_preprocess.img_data = tifffile.imread(input_file).astype("float32")`
- `recon_preprocess.output_file = output_dir / input_file.name`

### 7.17.1 Detailed Description

Script handling the preprocessing of images before SIM reconstruction.

Takes a directory of images as input, equalizes the total acquisition, intensities within each image, subtracts background and extreme pixels on a percentile basis, then scales to the full 16-bit depth range.

Arguments:

- i: directory path of input images
- o: directory path of output images
- d: dimension
- l: lower percentile used for clipping (background)
- u: upper percentile used for clipping (bright values)
- n: turns on normalization of acquisition intensity

## 7.18 /home/jhughes2712/projects/sim\_project/jh2284/src/stats.py File Reference

### Namespaces

- [stats](#)

### Functions

- def [stats.paired\\_t](#)(gt\_data, data)

### Variables

- [stats.parser](#) = argparse.ArgumentParser()
- [stats.type](#)
- [stats.str](#)
- [stats.required](#)
- [stats.int](#)
- [stats.choices](#)
- [stats.default](#)
- [stats.args](#) = parser.parse\_args()
- [stats.output\\_dir](#) = pathlib.Path(args.output\_dir)
- [stats.parents](#)
- [stats.True](#)
- [stats.exist\\_ok](#)
- [stats.df](#)
- [stats.fig](#)
- [stats.ax](#)
- [stats.figsizes](#)
- [stats.psnr\\_diff\\_1\\_max](#)
- [stats.psnr\\_diff\\_2\\_max](#)
- [stats.psnr\\_diff\\_1\\_min](#)
- [stats.psnr\\_diff\\_2\\_min](#)
- tuple [stats.hist\\_range\\_psnr](#)
- [stats.ssim\\_diff\\_1\\_max](#)
- [stats.ssim\\_diff\\_2\\_max](#)
- [stats.ssim\\_diff\\_1\\_min](#)
- [stats.ssim\\_diff\\_2\\_min](#)
- tuple [stats.hist\\_range\\_ssim](#)
- [stats.xlabel](#)
- [stats.title](#)
- [stats.range](#)
- [stats.color](#)
- [stats.mean\\_psnr\\_1](#) = np.mean(np.array(df['psnr\_model\_1']) - np.array(df['psnr\_raw']))
- [stats.se\\_psnr\\_1](#)
- [stats.mean\\_ssim\\_1](#) = np.mean(np.array(df['ssim\_model\_1']) - np.array(df['ssim\_raw']))
- [stats.se\\_ssim\\_1](#)
- [stats.mean\\_psnr\\_2](#)
- [stats.se\\_psnr\\_2](#)
- [stats.mean\\_ssim\\_2](#)
- [stats.se\\_ssim\\_2](#)



- `int stats.psnr_cols = 2` else `df.columns[1:3]`
- `int stats.ssim_cols = 2` else `df.columns[3:5]`
- `stats.dflong`
- `stats.dflongssim`
- `stats.data`
- `stats.x`
- `stats.y`
- `stats.hue`
- `stats.dodge`
- `stats.legend`
- `stats.palette`
- `stats.alpha`
- `stats.lw`

## 7.19 /home/jhughes2712/projects/sim\_project/jh2284/src/synthetic\_sim/otf.py File Reference

### Classes

- class `synthetic_sim.otf.PsfParameters`  
*Class to store PSF parameters.*

### Namespaces

- `synthetic_sim.otf`

### Functions

- def `synthetic_sim.otf.calc_psf` (params)  
*Calculate an approximate Gibson-Lanni PSF based on the parameters provided.*

## 7.20 /home/jhughes2712/projects/sim\_project/jh2284/src/train.py File Reference

Script used to train RCAN.

### Namespaces

- `train`

### Functions

- def `train.load_data_paths` (config, data\_type)
- def `train.train` (train\_loader, val\_loader, optimizer, scheduler, net, batchsize, n\_accumulations, saveinterval, nepoch, start\_epoch=0, losses\_train\_epoch=[], losses\_val\_epoch=[], psnr\_train\_epoch=[], psnr\_val\_epoch=[], ssim\_train\_epoch=[], ssim\_val\_epoch=[])

## Variables

- `train.parser` = `argparse.ArgumentParser()`
- `train.type`
- `train.str`
- `train.required`
- `train.args` = `parser.parse_args()`
- dictionary `train.schema`
- `train.config` = `json.load(f)`
- int `train.ndim` = `tiffimage.imread(training_data[0]["raw"]).ndim - 1`
- `train.input_shape` = `config["input_shape"]`
- tuple `train.device`
- `train.ckpt_path` = None if `args.model_ckpt` is None else `pathlib.Path(args.model_ckpt)`
- `train.model`
- dictionary `train.RCAN_hyperparameters`
- `train.ckpt`
- `train.train_loader`
- `train.val_loader`
- `train.optimizer`
- `train.scheduler`
- `train.output_dir` = `pathlib.Path(args.output_dir)`
- `train.parents`
- `train.True`
- `train.exist_ok`
- `train.n_accumulations`
- `train.saveinterval`
- `train.nepoch`
- `train.start_epoch`
- `train.losses_train_epoch`
- `train.losses_val_epoch`
- `train.psnr_train_epoch`
- `train.psnr_val_epoch`
- `train.ssim_train_epoch`
- `train.ssim_val_epoch`

### 7.20.1 Detailed Description

Script used to train RCAN.

Reads the specified config.json file, and trains an RCAN model accordingly. Intermediate training progress is saved using model checkpoints. Can handle resumed model training if a previous checkpoint is provided.

Arguments:

- c: filepath for config JSON file
- o: path of model checkpoint directory
- m: filepath of intermediate model checkpoint (if given, training resumes from this checkpoint)

Adapted from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/train.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

# Index

/home/jhughes2712/projects/sim\_project/jh2284/src/analyse.py, 75  
97 rcan.model.RCAN, 81  
/home/jhughes2712/projects/sim\_project/jh2284/src/apply.py, len\_\_  
98 rcan.data\_generator.SIM\_Dataset, 86  
/home/jhughes2712/projects/sim\_project/jh2284/src/convert\_omx\_to\_czxy.py,  
100 rcan.data\_generator.SIM\_Dataset, 86  
/home/jhughes2712/projects/sim\_project/jh2284/src/convert\_omx\_to\_paz.py,  
100 rcan.model, 42  
/home/jhughes2712/projects/sim\_project/jh2284/src/convert\_datasets\_and\_volumes.py,  
101 rcan.model, 43  
/home/jhughes2712/projects/sim\_project/jh2284/src/generate\_sim.py, average\_pooling  
102 rcan.model, 43  
/home/jhughes2712/projects/sim\_project/jh2284/src/image\_histogram.py  
103 generate\_sim.Simulator, 93  
/home/jhughes2712/projects/sim\_project/jh2284/src/manage\_stack.py, threshold  
105 rcan.data\_generator.SIM\_Dataset, 86  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/\_psf.py,  
106 generate\_sim.Simulator, 94  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/data\_generator.py,  
106 rcan.data\_generator.SIM\_Dataset, 86  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/data\_processing.py,  
107 rcan.data\_generator.SIM\_Dataset, 86  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/model.py,  
108 rcan.data\_generator.SIM\_Dataset, 87  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/plotting.py, display  
108 rcan.model, 43  
/home/jhughes2712/projects/sim\_project/jh2284/src/rcan/utilities.py, superes\_psf  
109 generate\_sim.Simulator, 94  
/home/jhughes2712/projects/sim\_project/jh2284/src/recon\_postprocess.py, postprocess\_function  
110 rcan.data\_generator.SIM\_Dataset, 87  
/home/jhughes2712/projects/sim\_project/jh2284/src/recon\_preprocess.py,  
110 rcan.data\_generator.SIM\_Dataset, 87  
/home/jhughes2712/projects/sim\_project/jh2284/src/stats.py,  
112 action  
/home/jhughes2712/projects/sim\_project/jh2284/src/synthetic\_sim.py, analyse, 10  
106 apply, 16  
/home/jhughes2712/projects/sim\_project/jh2284/src/synthetic\_sim/orc.py, convert\_omx\_to\_czxy, 19  
113 convert\_omx\_to\_paz, 22  
/home/jhughes2712/projects/sim\_project/jh2284/src/train.py, manage\_stack, 34  
113 recon\_preprocess, 51  
\_\_getitem\_\_  
rcan.data\_generator.SIM\_Dataset, 85  
\_\_init\_\_  
generate\_sim.SimulationRunner, 88  
generate\_sim.Simulator, 91  
rcan.data\_generator.SIM\_Dataset, 85  
rcan.data\_processing.ImageStack, 77  
rcan.model.\_channel\_attention\_block, 72  
rcan.model.\_residual\_channel\_attention\_blocks,  
add\_image  
rcan.data\_processing.ImageStack, 78  
add\_noise  
generate\_sim.Simulator, 92  
alpha  
stats, 54  
analyse, 9  
action, 10  
args, 10  
ckpt, 10

- cmap, 10
- default, 10
- device, 10
- df, 10
- exist\_ok, 11
- gt, 11
- gt\_dir, 11
- gt\_files, 11
- gt\_samples, 11
- img\_idx, 11
- int, 12
- model, 12
- model\_1, 12
- model\_1\_dir, 12
- model\_1\_files, 12
- model\_1\_samples, 12
- model\_2, 12
- model\_2\_dir, 13
- model\_2\_files, 13
- model\_2\_samples, 13
- N, 13
- output\_dir, 13
- parents, 13
- parser, 13
- psnr, 14
- raw, 14
- raw\_dir, 14
- raw\_files, 14
- raw\_samples, 14
- required, 14
- rng, 14
- ssim, 14
- str, 15
- True, 15
- type, 15
- angle\_error
  - generate\_sim.Simulator, 94
- apply, 15
  - action, 16
  - args, 16
  - choices, 16
  - ckpt, 16
  - data, 16
  - default, 16
  - device, 16
  - imagej, 16
  - input\_path, 17
  - int, 17
  - model, 17
  - output\_file, 17
  - output\_path, 17
  - overlap\_shape, 17
  - parents, 17
  - parser, 18
  - percentile, 18
  - raw, 18
  - raw\_files, 18
  - rcan.utils, 46
  - RCAN\_hyperparameters, 18
  - required, 18
  - restored, 18
  - str, 19
  - type, 19
- arange\_zero
  - generate\_sim, 26
- args
  - analyse, 10
  - apply, 16
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
  - convert\_slices\_to\_volumes, 24
  - generate\_sim, 27
  - image\_noising, 29
  - manage\_stack, 34
  - recon\_postprocess, 49
  - recon\_preprocess, 51
  - stats, 55
  - train, 65
- ax
  - stats, 55
- beam\_position
  - generate\_sim.Simulator, 94
- calc\_psf
  - synthetic\_sim.otf, 63
- Callable
  - synthetic\_sim.otf.PsfParameters, 79
- channel\_attention\_block\_list
  - rcan.model.\_residual\_channel\_attention\_blocks, 76
- choices
  - apply, 16
  - image\_noising, 29
  - manage\_stack, 34
  - recon\_preprocess, 51
  - stats, 55
- ckpt
  - analyse, 10
  - apply, 16
  - train, 65
- ckpt\_path
  - train, 65
- cmap
  - analyse, 10
- color
  - stats, 55
- compute\_metrics
  - rcan.plotting, 44
- config
  - train, 65
- conv\_1
  - rcan.model.\_channel\_attention\_block, 73
- conv\_2
  - rcan.model.\_channel\_attention\_block, 73
- conv\_czxy\_to\_omx
  - rcan.data\_processing, 39

- conv\_input
  - rcan.model.RCAN, 82
- conv\_list
  - rcan.model.\_residual\_channel\_attention\_blocks, 76
  - rcan.model.RCAN, 82
- conv\_omx\_to\_czxy
  - rcan.data\_processing, 39
- conv\_omx\_to\_paz
  - rcan.data\_processing, 39
- conv\_output
  - rcan.model.RCAN, 83
- conv\_paz\_to\_omx
  - rcan.data\_processing, 40
- convert\_omx\_to\_czxy, 19
  - action, 19
  - args, 20
  - converted, 20
  - imagej, 20
  - input\_dir, 20
  - input\_files, 20
  - int, 20
  - original, 20
  - parser, 21
  - required, 21
  - str, 21
  - type, 21
- convert\_omx\_to\_paz, 21
  - action, 22
  - args, 22
  - converted, 22
  - imagej, 22
  - input\_dir, 22
  - input\_files, 22
  - int, 22
  - original, 22
  - parser, 23
  - required, 23
  - str, 23
  - type, 23
- convert\_slices\_to\_volumes, 23
  - args, 24
  - default, 24
  - exist\_ok, 24
  - imagej, 24
  - input\_dir, 24
  - input\_files, 24
  - input\_slice, 24
  - output\_dir, 24
  - output\_file, 25
  - parents, 25
  - parser, 25
  - required, 25
  - str, 25
  - subvolume, 25
  - True, 25
  - tuple\_of\_ints, 25
  - type, 26
  - volume, 26
- converted
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
- crop\_volume
  - rcan.data\_processing, 40
- data
  - apply, 16
  - image\_noising, 29
  - stats, 55
- default
  - analyse, 10
  - apply, 16
  - convert\_slices\_to\_volumes, 24
  - generate\_sim, 27
  - image\_noising, 30
  - manage\_stack, 34
  - recon\_preprocess, 51
  - stats, 55
- delta\_z\_p
  - generate\_sim.Simulator, 94
- device
  - analyse, 10
  - apply, 16
  - train, 66
- df
  - analyse, 10
  - stats, 55
- dflong
  - stats, 56
- dflongssim
  - stats, 56
- dim
  - rcan.data\_processing.ImageStack, 78
- do\_sim
  - generate\_sim.SimulationRunner, 89
- dodge
  - stats, 56
- exist\_ok
  - analyse, 11
  - convert\_slices\_to\_volumes, 24
  - manage\_stack, 34
  - recon\_preprocess, 51
  - stats, 56
  - train, 66
- export\_stack
  - rcan.data\_processing.ImageStack, 78
- fig
  - stats, 56
- figsize
  - stats, 57
- filename
  - manage\_stack, 34
- files
  - manage\_stack, 34
  - recon\_postprocess, 49

- recon\_preprocess, 51
- float
  - image\_noising, 30
  - synthetic\_sim.otf.PsfParameters, 80
- forward
  - rca.model.\_channel\_attention\_block, 73
  - rca.model.\_residual\_channel\_attention\_blocks, 75
  - rca.model.RCAN, 82
- generate\_sim, 26
  - arange\_zero, 26
  - args, 27
  - default, 27
  - int, 27
  - parser, 27
  - required, 27
  - runner, 28
  - str, 28
  - threshold\_norm, 27
  - type, 28
- generate\_sim.SimulationRunner, 88
  - \_\_init\_\_, 88
  - do\_sim, 89
  - input\_dir, 89
  - input\_files, 89
  - output\_dir, 89
  - range, 89
  - run, 89
  - z\_offset, 90
- generate\_sim.Simulator, 90
  - \_\_init\_\_, 91
  - \_illumination, 93
  - \_psf, 94
  - \_superres\_psf, 94
  - add\_noise, 92
  - angle\_error, 94
  - beam\_position, 94
  - delta\_z\_p, 94
  - illumination, 92
  - in\_focus\_plane, 92
  - k0, 94
  - k\_exc, 94
  - lambda0, 94
  - lambda\_exc, 95
  - n\_angles, 95
  - n\_g, 95
  - n\_i, 95
  - n\_rotations, 95
  - n\_sample, 95
  - n\_shifts, 95
  - n\_x, 95
  - n\_z, 96
  - params\_dict, 92
  - poisson\_photons, 96
  - psf, 92
  - psf\_params, 92
  - randomise, 93
  - res\_axial, 96
  - res\_lateral, 96
  - signal\_to\_noise, 96
  - simulate\_ideal\_superres, 93
  - simulate\_sim, 93
  - wavevectors, 93
  - z, 96
  - z\_p, 96
- global\_average\_pooling
  - rca.model.\_channel\_attention\_block, 73
- gt
  - analyse, 11
  - image\_noising, 30
- gt\_dir
  - analyse, 11
- gt\_files
  - analyse, 11
- gt\_samples
  - analyse, 11
- hist\_range\_psnr
  - stats, 57
- hist\_range\_ssim
  - stats, 57
- hue
  - stats, 57
- illumination
  - generate\_sim.Simulator, 92
- image\_noising, 28
  - args, 29
  - choices, 29
  - data, 29
  - default, 30
  - float, 30
  - gt, 30
  - img\_idx\_all, 30
  - img\_idx\_test, 30
  - img\_idx\_train, 30
  - img\_idx\_val, 30
  - input\_path, 30
  - int, 31
  - n\_acquisitions, 31
  - n\_img, 31
  - output\_path, 31
  - output\_test\_gt\_path, 31
  - output\_test\_raw\_path, 31
  - output\_train\_gt\_path, 31
  - output\_train\_raw\_path, 31
  - output\_val\_gt\_path, 32
  - output\_val\_raw\_path, 32
  - parents, 32
  - parser, 32
  - required, 32
  - rng, 32
  - save\_image\_pair, 29
  - split, 32
  - str, 32
  - train\_size, 33
  - type, 33

- val\_size, 33
- imagej
  - apply, 16
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
  - convert\_slices\_to\_volumes, 24
- img\_data
  - manage\_stack, 35
  - recon\_postprocess, 49
  - recon\_preprocess, 52
- img\_idx
  - analyse, 11
- img\_idx\_all
  - image\_noising, 30
- img\_idx\_test
  - image\_noising, 30
- img\_idx\_train
  - image\_noising, 30
- img\_idx\_val
  - image\_noising, 30
- in\_focus\_plane
  - generate\_sim.Simulator, 92
- input\_dir
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
  - convert\_slices\_to\_volumes, 24
  - generate\_sim.SimulationRunner, 89
- input\_files
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
  - convert\_slices\_to\_volumes, 24
  - generate\_sim.SimulationRunner, 89
- input\_path
  - apply, 17
  - image\_noising, 30
- input\_shape
  - train, 66
- input\_slice
  - convert\_slices\_to\_volumes, 24
- int
  - analyse, 12
  - apply, 17
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
  - generate\_sim, 27
  - image\_noising, 31
  - manage\_stack, 35
  - recon\_preprocess, 52
  - stats, 57
  - synthetic\_sim.otf.PsfParameters, 80
- k0
  - generate\_sim.Simulator, 94
- k\_exc
  - generate\_sim.Simulator, 94
- lambda0
  - generate\_sim.Simulator, 94
- lambda\_exc
  - generate\_sim.Simulator, 94
- generate\_sim.Simulator, 95
- legend
  - stats, 57
- load\_data\_paths
  - train, 64
- load\_rcan\_checkpoint
  - rcan.utils, 47
- load\_SIM\_dataset
  - rcan.data\_generator, 37
- losses\_train\_epoch
  - train, 66
- losses\_val\_epoch
  - train, 66
- lw
  - stats, 58
- manage\_stack, 33
  - action, 34
  - args, 34
  - choices, 34
  - default, 34
  - exist\_ok, 34
  - filename, 34
  - files, 34
  - img\_data, 35
  - int, 35
  - number\_of\_stacks, 35
  - output\_data, 35
  - output\_dir, 35
  - output\_file, 35
  - parents, 35
  - parser, 36
  - required, 36
  - sample, 36
  - stack\_handler, 36
  - stack\_number, 36
  - str, 36
  - True, 36
  - type, 37
- mean\_psnr\_1
  - stats, 58
- mean\_psnr\_2
  - stats, 58
- mean\_ssim\_1
  - stats, 58
- mean\_ssim\_2
  - stats, 58
- model
  - analyse, 12
  - apply, 17
  - train, 66
- model\_1
  - analyse, 12
- model\_1\_dir
  - analyse, 12
- model\_1\_files
  - analyse, 12
- model\_1\_samples
  - analyse, 12

- model\_2
  - analyse, 12
- model\_2\_dir
  - analyse, 13
- model\_2\_files
  - analyse, 13
- model\_2\_samples
  - analyse, 13
- N
  - analyse, 13
- n\_accumulations
  - train, 66
- n\_acq
  - rcan.data\_processing.ImageStack, 78
- n\_acquisitions
  - image\_noising, 31
- n\_angles
  - generate\_sim.Simulator, 95
- n\_g
  - generate\_sim.Simulator, 95
- n\_i
  - generate\_sim.Simulator, 95
- n\_img
  - image\_noising, 31
- n\_rotations
  - generate\_sim.Simulator, 95
- n\_sample
  - generate\_sim.Simulator, 95
- n\_shifts
  - generate\_sim.Simulator, 95
- n\_x
  - generate\_sim.Simulator, 95
- n\_z
  - generate\_sim.Simulator, 96
  - rcan.data\_processing.ImageStack, 78
- ndim
  - train, 67
- nepoch
  - train, 67
- normalize
  - rcan.utils, 47
- normalize\_acquisition\_intensity
  - recon\_preprocess, 51
- normalize\_between\_zero\_and\_one
  - rcan.utils, 48
- num\_residual\_groups
  - rcan.model.RCAN, 83
- number\_of\_stacks
  - manage\_stack, 35
- optimizer
  - train, 67
- original
  - convert\_omx\_to\_czxy, 20
  - convert\_omx\_to\_paz, 22
- output\_data
  - manage\_stack, 35
- output\_dir
  - analyse, 13
  - convert\_slices\_to\_volumes, 24
  - generate\_sim.SimulationRunner, 89
  - manage\_stack, 35
  - recon\_preprocess, 52
  - stats, 58
  - train, 67
- output\_file
  - apply, 17
  - convert\_slices\_to\_volumes, 25
  - manage\_stack, 35
  - recon\_preprocess, 52
- output\_path
  - apply, 17
  - image\_noising, 31
- output\_shape
  - rcan.data\_generator.SIM\_Dataset, 87
- output\_signature
  - rcan.data\_generator.SIM\_Dataset, 87
- output\_test\_gt\_path
  - image\_noising, 31
- output\_test\_raw\_path
  - image\_noising, 31
- output\_train\_gt\_path
  - image\_noising, 31
- output\_train\_raw\_path
  - image\_noising, 31
- output\_val\_gt\_path
  - image\_noising, 32
- output\_val\_raw\_path
  - image\_noising, 32
- overlap\_shape
  - apply, 17
- p\_max
  - rcan.data\_generator.SIM\_Dataset, 87
- p\_min
  - rcan.data\_generator.SIM\_Dataset, 87
- paired\_t
  - stats, 54
- palette
  - stats, 58
- params\_dict
  - generate\_sim.Simulator, 92
- parents
  - analyse, 13
  - apply, 17
  - convert\_slices\_to\_volumes, 25
  - image\_noising, 32
  - manage\_stack, 35
  - recon\_preprocess, 52
  - stats, 59
  - train, 67
- parser
  - analyse, 13
  - apply, 18
  - convert\_omx\_to\_czxy, 21
  - convert\_omx\_to\_paz, 23
  - convert\_slices\_to\_volumes, 25



- generate\_sim, 27
- image\_noising, 32
- manage\_stack, 36
- recon\_postprocess, 50
- recon\_preprocess, 52
- stats, 59
- train, 67
- percentile
  - apply, 18
  - rcan.utils, 48
  - recon\_preprocess, 52
- plot\_learning\_curve
  - rcan.plotting, 45
- plot\_reconstructions
  - rcan.plotting, 45
- poisson\_photons
  - generate\_sim.Simulator, 96
- psf
  - generate\_sim.Simulator, 92
- psf\_params
  - generate\_sim.Simulator, 92
- psnr
  - analyse, 14
- psnr\_cols
  - stats, 59
- psnr\_diff\_1\_max
  - stats, 59
- psnr\_diff\_1\_min
  - stats, 59
- psnr\_diff\_2\_max
  - stats, 59
- psnr\_diff\_2\_min
  - stats, 60
- psnr\_train\_epoch
  - train, 67
- psnr\_val\_epoch
  - train, 68
- randomise
  - generate\_sim.Simulator, 93
- range
  - generate\_sim.SimulationRunner, 89
  - stats, 60
- raw
  - analyse, 14
  - apply, 18
- raw\_dir
  - analyse, 14
- raw\_files
  - analyse, 14
  - apply, 18
- raw\_samples
  - analyse, 14
- rcab\_list
  - rcan.model.RCAN, 83
- rcan, 37
- rcan.data\_generator, 37
  - load\_SIM\_dataset, 37
- rcan.data\_generator.SIM\_Dataset, 83
  - \_\_getitem\_\_, 85
  - \_\_init\_\_, 85
  - \_\_len\_\_, 86
  - \_area\_threshold, 86
  - \_intensity\_threshold, 86
  - \_scale, 86
  - \_scale\_factor, 86
  - \_shape, 87
  - \_transform\_function, 87
  - \_y, 87
  - output\_shape, 87
  - output\_signature, 87
  - p\_max, 87
  - p\_min, 87
  - steps\_per\_epoch, 87
- rcan.data\_processing, 38
  - conv\_czxy\_to\_omx, 39
  - conv\_omx\_to\_czxy, 39
  - conv\_omx\_to\_paz, 39
  - conv\_paz\_to\_omx, 40
  - crop\_volume, 40
- rcan.data\_processing.ImageStack, 76
  - \_\_init\_\_, 77
  - add\_image, 78
  - dim, 78
  - export\_stack, 78
  - n\_acq, 78
  - n\_z, 78
  - sample, 79
  - stack, 79
- rcan.model, 42
  - \_conv, 42
  - \_destandardize, 43
  - \_global\_average\_pooling, 43
  - \_standardize, 43
- rcan.model\_channel\_attention\_block, 71
  - \_\_init\_\_, 72
  - conv\_1, 73
  - conv\_2, 73
  - forward, 73
  - global\_average\_pooling, 73
- rcan.model\_residual\_channel\_attention\_blocks, 74
  - \_\_init\_\_, 75
  - channel\_attention\_block\_list, 76
  - conv\_list, 76
  - forward, 75
  - repeat, 76
  - residual\_scaling, 76
- rcan.model.RCAN, 80
  - \_\_init\_\_, 81
  - conv\_input, 82
  - conv\_list, 82
  - conv\_output, 83
  - forward, 82
  - num\_residual\_groups, 83
  - rcab\_list, 83
- rcan.plotting, 44
  - compute\_metrics, 44

- plot\_learning\_curve, 45
  - plot\_reconstructions, 45
- rcan.utils, 46
  - apply, 46
  - load\_rcan\_checkpoint, 47
  - normalize, 47
  - normalize\_between\_zero\_and\_one, 48
  - percentile, 48
  - reshape\_to\_bcwh, 48
  - tuple\_of\_ints, 49
- RCAN\_hyperparameters
  - apply, 18
  - train, 68
- recon\_postprocess, 49
  - args, 49
  - files, 49
  - img\_data, 49
  - parser, 50
  - required, 50
  - str, 50
  - type, 50
- recon\_preprocess, 50
  - action, 51
  - args, 51
  - choices, 51
  - default, 51
  - exist\_ok, 51
  - files, 51
  - img\_data, 52
  - int, 52
  - normalize\_acquisition\_intensity, 51
  - output\_dir, 52
  - output\_file, 52
  - parents, 52
  - parser, 52
  - percentile, 52
  - required, 52
  - str, 53
  - True, 53
  - type, 53
- repeat
  - rcan.model.\_residual\_channel\_attention\_blocks, 76
- required
  - analyse, 14
  - apply, 18
  - convert\_omx\_to\_czxy, 21
  - convert\_omx\_to\_paz, 23
  - convert\_slices\_to\_volumes, 25
  - generate\_sim, 27
  - image\_noising, 32
  - manage\_stack, 36
  - recon\_postprocess, 50
  - recon\_preprocess, 52
  - stats, 60
  - train, 68
- res\_axial
  - generate\_sim.Simulator, 96
- res\_lateral
  - generate\_sim.Simulator, 96
- reshape\_to\_bcwh
  - rcan.utils, 48
- residual\_scaling
  - rcan.model.\_residual\_channel\_attention\_blocks, 76
- restored
  - apply, 18
- rng
  - analyse, 14
  - image\_noising, 32
- run
  - generate\_sim.SimulationRunner, 89
- runner
  - generate\_sim, 28
- sample
  - manage\_stack, 36
  - rcan.data\_processing.ImageStack, 79
- save\_image\_pair
  - image\_noising, 29
- saveinterval
  - train, 68
- scheduler
  - train, 68
- schema
  - train, 68
- se\_psnr\_1
  - stats, 60
- se\_psnr\_2
  - stats, 60
- se\_ssim\_1
  - stats, 60
- se\_ssim\_2
  - stats, 61
- signal\_to\_noise
  - generate\_sim.Simulator, 96
- simulate\_ideal\_superres
  - generate\_sim.Simulator, 93
- simulate\_sim
  - generate\_sim.Simulator, 93
- split
  - image\_noising, 32
- ssim
  - analyse, 14
- ssim\_cols
  - stats, 61
- ssim\_diff\_1\_max
  - stats, 61
- ssim\_diff\_1\_min
  - stats, 61
- ssim\_diff\_2\_max
  - stats, 61
- ssim\_diff\_2\_min
  - stats, 62
- ssim\_train\_epoch
  - train, 69
- ssim\_val\_epoch

- train, 69
- stack
  - rcan.data\_processing.ImageStack, 79
- stack\_handler
  - manage\_stack, 36
- stack\_number
  - manage\_stack, 36
- start\_epoch
  - train, 69
- stats, 53
  - alpha, 54
  - args, 55
  - ax, 55
  - choices, 55
  - color, 55
  - data, 55
  - default, 55
  - df, 55
  - dflong, 56
  - dflongssim, 56
  - dodge, 56
  - exist\_ok, 56
  - fig, 56
  - figsize, 57
  - hist\_range\_psnr, 57
  - hist\_range\_ssim, 57
  - hue, 57
  - int, 57
  - legend, 57
  - lw, 58
  - mean\_psnr\_1, 58
  - mean\_psnr\_2, 58
  - mean\_ssim\_1, 58
  - mean\_ssim\_2, 58
  - output\_dir, 58
  - paired\_t, 54
  - palette, 58
  - parents, 59
  - parser, 59
  - psnr\_cols, 59
  - psnr\_diff\_1\_max, 59
  - psnr\_diff\_1\_min, 59
  - psnr\_diff\_2\_max, 59
  - psnr\_diff\_2\_min, 60
  - range, 60
  - required, 60
  - se\_psnr\_1, 60
  - se\_psnr\_2, 60
  - se\_ssim\_1, 60
  - se\_ssim\_2, 61
  - ssim\_cols, 61
  - ssim\_diff\_1\_max, 61
  - ssim\_diff\_1\_min, 61
  - ssim\_diff\_2\_max, 61
  - ssim\_diff\_2\_min, 62
  - str, 62
  - title, 62
  - True, 62
  - type, 62
  - x, 62
  - xlabel, 62
  - y, 63
- steps\_per\_epoch
  - rcan.data\_generator.SIM\_Dataset, 87
- str
  - analyse, 15
  - apply, 19
  - convert\_omx\_to\_czxy, 21
  - convert\_omx\_to\_paz, 23
  - convert\_slices\_to\_volumes, 25
  - generate\_sim, 28
  - image\_noising, 32
  - manage\_stack, 36
  - recon\_postprocess, 50
  - recon\_preprocess, 53
  - stats, 62
  - train, 69
- subvolume
  - convert\_slices\_to\_volumes, 25
- synthetic\_sim, 63
- synthetic\_sim.otf, 63
  - calc\_psf, 63
- synthetic\_sim.otf.PsfParameters, 79
  - Callable, 79
  - float, 80
  - int, 80
- threshold\_norm
  - generate\_sim, 27
- title
  - stats, 62
- train, 64
  - args, 65
  - ckpt, 65
  - ckpt\_path, 65
  - config, 65
  - device, 66
  - exist\_ok, 66
  - input\_shape, 66
  - load\_data\_paths, 64
  - losses\_train\_epoch, 66
  - losses\_val\_epoch, 66
  - model, 66
  - n\_accumulations, 66
  - ndim, 67
  - nepoch, 67
  - optimizer, 67
  - output\_dir, 67
  - parents, 67
  - parser, 67
  - psnr\_train\_epoch, 67
  - psnr\_val\_epoch, 68
  - RCAN\_hyperparameters, 68
  - required, 68
  - saveinterval, 68
  - scheduler, 68
  - schema, 68

- ssim\_train\_epoch, [69](#)
- ssim\_val\_epoch, [69](#)
- start\_epoch, [69](#)
- str, [69](#)
- train, [65](#)
- train\_loader, [69](#)
- True, [69](#)
- type, [70](#)
- val\_loader, [70](#)
- train\_loader
  - train, [69](#)
- train\_size
  - image\_noising, [33](#)
- True
  - analyse, [15](#)
  - convert\_slices\_to\_volumes, [25](#)
  - manage\_stack, [36](#)
  - recon\_preprocess, [53](#)
  - stats, [62](#)
  - train, [69](#)
- tuple\_of\_ints
  - convert\_slices\_to\_volumes, [25](#)
  - rcan.utils, [49](#)
- type
  - analyse, [15](#)
  - apply, [19](#)
  - convert\_omx\_to\_czxy, [21](#)
  - convert\_omx\_to\_paz, [23](#)
  - convert\_slices\_to\_volumes, [26](#)
  - generate\_sim, [28](#)
  - image\_noising, [33](#)
  - manage\_stack, [37](#)
  - recon\_postprocess, [50](#)
  - recon\_preprocess, [53](#)
  - stats, [62](#)
  - train, [70](#)
- val\_loader
  - train, [70](#)
- val\_size
  - image\_noising, [33](#)
- volume
  - convert\_slices\_to\_volumes, [26](#)
- wavevectors
  - generate\_sim.Simulator, [93](#)
- x
  - stats, [62](#)
- xlabel
  - stats, [62](#)
- y
  - stats, [63](#)
- z
  - generate\_sim.Simulator, [96](#)
- z\_offset
  - generate\_sim.SimulationRunner, [90](#)
- z\_p
  - generate\_sim.Simulator, [96](#)