multi-QuCAD User Manual

Version 2024

Michelle Mastrianni, Yee Lam Elim Thompson

December 13, 2024

Contents

List of Code Listings

1 Introduction

The primary goal of multi-QuCAD is to evaluate wait-time-saving performance of one or multiple Computed-Aided Triage and Notification (CADt) devices. This type of Software as a Medical Device (SaMD) is intended to triage patients' radiological medical images based on the binary outputs of the underlying Artificial Intelligence and Machine Learning (AI/ML) algorithms. Their main benefit is to speed up review of radiological medical images with time-critical findings, such as Large Vessel Occlusion (LVO) and Intracerebral Hemorrhage (ICH), increasing the likelihood of a timely diagnosis and potentially improved patient outcomes.

To quantify how much time is saved due to the use of multiple CADt devices for different conditions, this Python software simulates patient image flow to determine the waiting time distributions for different subgroups of patient images with and without a CADt device, as well as their time-saving distributions. The workflow may have a hierarchy of disease time-criticalness in which AI-positive images of the most time-sensitive conditions are triaged above those of relatively less time-sensitive conditions. The software also includes theoretical calculations of average waiting times and wait-time savings based on queueing theory.

For more information on the project motivation and problem statement, please visit [2].

1.1 Credits

- Frank Samuelson - Present
  - Project Lead
- Yee Lam Elim Thompson - Present
  - Co-Project Lead
  - Original software creator
  - Consolidated on the user manual
- Jixin (Audrey) Zheng - 2021
  - Optimized runtime performance
- Rucha Deshpande - 2023
  - Implemented theoretical predictions for hierarchical queue for preemptive-resume queue
  - Modified simulation script to account for multiple CADt devices
- Michelle Mastrianni - Present
  - Generalized theory validation to account for non-preemptive case and multiple AIs per group
  - Author of the user manual

2 Getting Started

2.1 Python environment

This software is written entirely in python version 3.9.4 with the following packages

- numpy
- pandas
- scipy
- matplotlib
- statsmodels

A `requirements.txt` file is provided with all packages and their version numbers.

2.1.1 Cloning multi-QuCAD from GitHub

In a terminal (Unix/macOS) or a PowerShell (Windows) console, run the following command line. Replace */path/to/your/working/directory/* to a work space you would like to work at. This will download the entire repository to your working directory.

```
$ cd /path/to/your/working/directory/
$ git clone https://github.com/DIDSR/multi-QuCAD.git multi-QuCAD
```

2.1.2 Setting up virtual environment

Users can follow the Python documentation on how to create virtual environments. In the following example, we assume that a virtual environment under the multi-QuCAD repository with a path */path/to/your/working/directory/multi-QuCAD/env/*.

Once a virtual environment is created, activate it in a terminal (Unix/macOS) or a PowerShell (Windows) via the following command line.

```
  $ source env/bin/activate ## Unix/macOS
OR
  $ .\env\Scripts\activate  ## Windows
```

To set up the environment for this software, use the `requirements.txt`. Installing packages may take awhile.

```
  $ python3 -m pip install -r requirements.txt ## Unix/macOS
OR
  $ py -m pip install -r requirements.txt      ## Windows
```

Once the installation is completed, user can follow the instructions in Section 2.2 to run simulations. Next time, when the user wants to run the simulation again, they can follow Section 2.1.3 to reactivate the virtual environment without re-installing all the required packages.

2.1.3 Activating virtual environment

To re-activate the virtual environment that was created, users can open a terminal (Unix/macOS) or a PowerShell (Windows) and proceed to where the virtual environment was created.

```
  $ cd /path/to/your/working/directory/multi-QuCAD/
  $ source env/bin/activate ## Unix/macOS
OR
  $  cd /path/to/your/working/directory/multi-QuCAD/
  $ .\env\Scripts\activate  ## Windows
```

2.2 Running `run_sim.py`

`run_sim.py` is the main script to run the software. Users can specify all parameters via an input `config.dat` file. For explanations of each parameters, see Section 3.1. To run `run_sim.py`, use the following command line:

```
$ cd /path/to/your/working/directory/multi-QuCAD/scripts/
$ python run_sim.py --configFile ../inputs/config.dat
```

By default, `config.dat` asks the software to print out information as it runs, and one may see the printouts as shown in Listing 1. Once the software shows a table of quick results from one trial, the run is successful.

2.3 Verbose messages

Users may see warning and error messages if unexpected input parameters are provided.

2.3.1 Warnings

When a user encounters the following warning messages, the simulation run will likely to continue. However, the warning messages may help users to spot any unintentional values for the input parameters.

2.3.1.1 Limitations on theoretical calculation

Some theoretical predictions are not available in the following scenarios:

- If preemptive queueing with non-equal service times, more than one radiologist, and/or interrupting images, theory is not available for the hierarchical queueing scheme.
- If non-preemptive queueing with more than one radiologist and/or interrupting images, theory is not available for the priority or hierarchical queueing schemes.

When such a setting is encountered, a warning message is shown, but the simulation will go on. Example warning messages are shown below.

Listing 1: Example printouts when running `run_sim.py`

```
Reading user inputs:
+-----------------------------------------
| configFile: ../inputs/config_AILVOICH_3diseases_test.dat
| isPreemptive: True
| traffic: 0.8
| fractionED: 0.0
| nRadiologists: 1
| nTrials: 1
| nPatientsTarget: 100000
| verbose: True
| statsFile: ../outputs/stats/stats.p
| runTimeFile: ../outputs/stats/runTime.txt
| plotPath: ../outputs/plots/
| AIinfo AILVO: {'groupName': 'GroupCTA', 'targetDisease': 'LVO', 'TPFThresh': 0.9236, 'FPFThresh': None, 'rocFile': '../inputs/AILVOROC
| AIinfo AIICH: {'groupName': 'GroupNCCT', 'targetDisease': 'ICH', 'TPFThresh': 0.9361, 'FPFThresh': None, 'rocFile': '../inputs/AIICHRO
| diseaseGroups GroupNCCT: {'diseaseNames': ['SAH', 'ICH'], 'diseaseProbs': [0.053, 0.211], 'diseaseRanks': [2, 3], 'groupProb': 0.7}
| diseaseGroups GroupCTA: {'diseaseNames': ['LVO'], 'diseaseProbs': [0.125], 'diseaseRanks': [1], 'groupProb': 0.3}
| meanServiceTimes GroupNCCT: {'SAH': 30.0, 'ICH': 30.0, 'non-diseased': 30.0}
| meanServiceTimes GroupCTA: {'LVO': 30.0, 'non-diseased': 30.0}
| meanServiceTimes interrupting: 5.0
| doPlots: True
| doRunTime: True
+-----------------------------------------

+-----------------------------------------------
| Group GroupNCCT (69859):
|   * Group prob (theory, sim): 0.700, 0.700
|   * Disease SAH (3691):
|       -- Disease prob (theory, sim): 0.053, 0.053
|   * Disease ICH (14769):
|       -- Disease prob (theory, sim): 0.211, 0.211
|   * Disease non-diseased (51399):
|       -- Disease prob (theory, sim): 0.736, 0.736
+-----------------------------------------------
| Group GroupCTA (29968):
|   * Group prob (theory, sim): 0.300, 0.300
|   * Disease LVO (3681):
|       -- Disease prob (theory, sim): 0.125, 0.123
|   * Disease non-diseased (26287):
|       -- Disease prob (theory, sim): 0.875, 0.877
+-----------------------------------------------

+-----------------------------------------------
| AI AILVO (LVO) in group GroupCTA:
|    * PPV (theory, sim): 0.6062, 0.6007
|    * NPV (theory, sim): 0.9882, 0.9874
+-----------------------------------------------
| AI AIICH (ICH) in group GroupNCCT:
|    * PPV (theory, sim): 0.7924, 0.7920
|    * NPV (theory, sim): 0.9820, 0.9817
+-----------------------------------------------

1 trials took 36.53 minutes.

Quick results from 1 trial
                      n_sim_pateints  sim_waittime  sim_waittime_low  sim_waittime_high  theory_waittime
columns
fifo_non-interrupting           99827    112.785216        111.935857         113.634575       120.000000
priority SAH                     3691    135.390630        129.875741         140.905518       144.454354
priority ICH                    14769     15.951843         14.953341          16.950346        16.235332
priority LVO                     3681     17.830095         15.634170          20.026020        18.076497
hierarchical SAH                 3691    135.446372        129.932489         140.960255       144.570043
hierarchical ICH                14769     17.590124         16.574332          18.605915        17.886252
hierarchical LVO                 3681     13.421718         11.267264          15.576173        13.101546
```

```
WARN: Preemptive queueing with non-equal service times.
        Theoretical results not available for hierarchical scenario.
WARN: Preemptive queueing with more than one radiologist.
        Theoretical results not available for hierarchical scenario.
WARN: Preemptive queueing with presence of interrupting images.
        Theoretical results not available for hierarchical scenario.
WARN: Non-preemptive queueing with more than one radiologist.
        Theoretical results not available for priority or hierarchical scenario.
WARN: Non-preemptive queueing with presence of interrupting images.
        Theoretical results not available for priority or hierarchical scenario.
WARN: There are more than 2 radiologists with presence of interrupting images.
        Theoretical values for AI negative and diseased/non-diseased subgroups will not be available.
```

2.3.1.2 Inputting both `rocFile` and `FPFThresh`

When both input parameters are provided, the software uses `FPFThresh` to determine the CADt operating threshold for the simulation run. The `rocFile` input value will be ignored, and a user will see the following warning message.

```
WARN: Received both FPFThresh and rocFile. Taking FPFThresh and ignoring rocFile.
```

2.3.2 Errors

Unlike warning messages, error messages will immediately kill the execution.

2.3.2.1 Invalid `traffic`

By design, the traffic intensity must be between 0 and 0.99. Values larger than 0.99 represents a unrealistically congested queueing system in which patient images are stuck because images are coming in much faster than the radiologists can handled. If a value larger than 0.99 is provided, an error message is displayed.

```
ERROR: Input traffic 1.000 is too high.
OSError: Please limit traffic below 0.99.
```

2.3.2.2 Invalid values for parameters ranging between 0 and 1

By definition, `TPFThresh`, `FPFThresh` (if provided), `prevalence`, and `fractionED` must be between 0 and 1. If a value outside of that range is provided, one will get the following Error message.

```
ERROR: Input prevalence 2.000 is too high.
OSError: Please provide a prevalence between 0 and 1.
```

2.3.2.3 Invalid paths and input files

If the user provides a value for `rocFile` or `configFile`, and the file does not exist, an error will be thrown. Similarly, for `statsFile`, `runtimeFile`, and `plotPath`, the existence of these paths are checked before running any simulations. If the paths do not exist, the software will try to create one. If failed, the user will see an error message.

```
ERROR: Path does not exist: /does/not/exist/
ERROR: Trying to create the folder ...
OSError: Cannot create the folder.
         Please provide a valid {0} path.
```

2.3.2.4 Neither `rocFile` nor `FPFThresh`

To simulate image patient flow with the use of CADt devices, either one of the two input parameters is required to simulate CADt diagnostic performance. If neither of them are provided, the following error message will be displayed.

```
ERROR: Neither FPFThresh nor rocFile is provided.
OSError: Please provide either FPFThresh or rocFile.
```

2.4 Expected output files

Depending on the user input parameters, different output files will be generated. If run as default, one can expect the following output files. For descriptions of each files, see Section 3.2.

- `outputs/stats/stats.p`
- `outputs/stats/runTime.txt`
- `outputs/plots/sim_theory_priority.pdf`
- `outputs/plots/sim_theory_hierarchical.pdf`
- `outputs/plots/patient_timings_fifo.pdf`
- `outputs/plots/patient_timings_priority.pdf`
- `outputs/plots/patient_timings_hierarchical.pdf`

3 Inputs and output

3.1 Input parameters

Users define all parameters via an `inputs/config.dat` file. Input parameters are categorized into clinical, disease condition specific, CADt AI diagnostic performance, and simulation parameters.

3.1.1 Clinical settings

Clinical parameters are listed in Listing 2, and a brief summary is presented in Table 3.1.1.

Parameter	Range	Description
`isPreemptive`	[True, False]	Preemptive vs. non-preemptive queueing scheme
`traffic`	$[0, 0.95]$	Hospital busyness
`fractionED`	$[0, 1]$	Fraction of interrupting patients in system
`nRadiologists`	Integer ≥ 1	Number of radiologists to review images
`meanServiceTimeInterruptingMin`	Positive float	Mean service time [min] for interrupting images

Listing 2: Parameters for clinical settings in `inputs/config.dat`

```
##############################################
## Clinical setting
##############################################
isPreemptive                   True # True if preemptive priority i.e. radiologist is
                                    # interrupted immediately upon arrival of a higher
                                    # priority case. False if non-preemptive priority
                                    # i.e. radiologist finishes the lower-priority case
                                    # first before reading the new-coming higher priority case
traffic                        0.8  # Hospital busyness:
                                    # Between 0.0 (very quiet) and 0.95 (very congested)
fractionED                     0.0  # Fraction of interrupting patients:
                                    # Between 0.0 (no interrupting images) and 1.0 (all
                                    # images are interrupting)
nRadiologists                  1    # Number of radiologists reading during the time window:
                                    # At least 1
meanServiceTimeInterruptingMin 5    # Average radiologist's service time [in minutes]
                                    # for interrupting cases
```

3.1.1.1 `isPreemptive`

isPreemptive indicates whether the clinic operates under a preemptive or non-preemptive queueing scheme. In a preemptive queueing scheme, radiologists stops the reviewing a lower-priority case when interrupted by higher-priority cases and later resumes the review of the interrupted case. In a non-preemptive queueing scheme, a radiologist finishes reading the lower-priority case before moving to the higher-priority case.

3.1.1.2 `traffic`

Traffic describes how busy a radiology department is. It is defined by the ratio between patient image arrival rate and radiologist's reading rate. For a quiet clinic with no images coming in, `traffic` is 0, whereas `traffic` is 1 for a very congested clinic where images enter the system at the same rate as the radiologist reading rate. Therefore, by definition, `traffic` greater than 1 implies an infinitely growing queue in which images arrive faster than they can leave. For simulation purposes, `traffic` must be smaller than 0.99.

3.1.1.3 `fractionED`

This parameter is the fraction of interrupting patient images with respect to all patient images in the system. Typically, a radiologist has a list of images to read, and if a CADt device is used, the device will read all images in the list. Interrupting images represent images that require a radiologist's immediate attention. These images are outside the reading list and are not reviewed by the CADt device. By definition, `fractionED` must be between 0 and 1. To simulate the scenario where radiologists will never be interrupted to review an image outside of their reading list, set `fractionED` to 0.

3.1.1.4 `nRadiologists`

`nRadiologists` is the number of radiologists reviewing images in the same reading list. Typically, a clinic has at least one radiologist at all times. For a larger hospital, multiple radiologists may be available at any time point. The number of radiologists may also depend on the time of the day and

Listing 3: Parameters for disease-related settings in `inputs/config.dat`

```
##############################################
## Group and disease parameters
##############################################

GroupNCCT
groupProb                       0.7
disease                         SAH  2  0.053  30
disease                         ICH  3  0.211  30
meanServiceTimeNonDiseasedMin   30

GroupCTA
groupProb                       0.3
disease                         LVO  1  0.125  30
meanServiceTimeNonDiseasedMin   30
```

day of the week. For simulation, one could set a very large number of radiologists. However, the theoretical computation of average wait-time-savings for certain subgroups of patient images may not be available for some clinical settings with `nRadiologists` > 1.

3.1.1.5 `meanServiceTimeInterruptingMin`

This is the average service time [minutes] by a radiologist to read an interrupting case.

3.1.2 Disease related parameters

While multi-QuCAD software is not tied to specific disease conditions, the amount of time-savings due to CADt devices depends on the prevalence of each condition, the overall percentage of patients in the clinic being screened for that condition, as well as the mean service times for patients with each condition (and non-diseased patients). Clinical parameters are listed in Listing 3, and a brief summary is presented in Table 3.1.2.

In multi-QuCAD, all patients within a specific 'group' (as indicated in the configFile) are seen by the same set of AIs. There is no limits on how many groups the software can handle, and each group may contain more than one disease condition. However, each group must have a unique name. For example, the block under `GroupCTA` in Listing 3 defines the information within a group. A user may add or remove a block for including more or less groups in the simulation.

Parameter	Location	Range	Description
`groupProb`		$[0,1]$	Probability that a patient is in this group
`diseaseName`	`disease` **`A`** `1 0.1 8`	String	Disease name
`diseaseRank`	`disease A` **`1`** `0.1 8`	Integer ≥ 1	Disease priority rank (high-priority = 1)
`prevalence`	`disease A 1` **`0.1`** `8`	$[0,1]$	Disease prevalence in reading list
`meanServiceTimeDiseasedMin`	`disease A 1 0.1` **`8`**	Positive float	Mean reading time of diseased patients
`meanServiceTimeNonDiseasedMin`		Positive float	Mean reading time of non-diseased patients

3.1.2.1 `groupProb`

The `groupProb` is the probability that a patient image will be placed into the corresponding group. Each patient in a queue belongs to a "group", and all patient images in the group are analyzed by all CADt devices in that group. For example, a group may consists of chest CT images, which are analyzed by two CADts (one for pulmonary embolism and another for pulmonary fibrosis). Another group may be chest X-ray images reviewed by a third CADt device for pneumothorax. It is also possible to have groups that do not have any CADt devices.

3.1.2.2 `diseaseName`

Each disease condition is given a unique name, `diseaseName`. For ease, disease names can be A, B, C, D, and so on.

3.1.2.3 `diseaseRank`

Each disease condition is also given a priority rank `diseaseRank` based, for example, on the time-sensitiveness of the disease. The highest-priority condition should be rank 1, the next-highest should be rank 2, and so on.

3.1.2.4 `prevalence`

For each condition, disease prevalence *within-group* is defined to be the ratio of the number of patient images diagnosed as positive for that condition in the group to the number of all patient images in the group. Both the numerator and denominator are with respect to total images in the corresponding patient group (e.g. GroupCTA in 3). By definition, `prevalence` is between 0 and 1. A `prevalence` of 0 means that the radiologist calls all images diseased with the condition, whereas 1 implies that all images are labeled as non-diseased by the radiologist.

3.1.2.5 `meanServiceTimeDiseasedMin`

For each condition, this parameter is the average service time in minutes by a radiologist for cases are diagnosed as positive for that condition. In queueing theory, the service time distribution is expected to follow an exponential distribution characterized by a service rate.

3.1.2.6 `meanServiceTimeNonDiseasedMin`

`meanServiceTimeNonDiseasedMin` is the average service time in minutes for a radiologist to call a case normal without any conditions.

3.1.3 CADt AI related parameters

Parameters related to the diagnostic performance of each CADt device are listed in Listing 4 and Table 3.1.3. In multi-QuCAD, there is no limits on how many AIs the software can handle. However, each AI is intended to diagnose one condition. For multi-class classifiers, one can define multiple AIs (each with a unique name and its own block), each for one intended condition. For example, the block under

Listing 4: Parameters related to the CADt device in `inputs/config.dat`

```
##############################################
## CADt AI diagnostic performance
##############################################

AILVO
groupName     GroupCTA
targetDisease LVO
TPFThresh     0.9236
FPFThresh     None
rocFile       ../inputs/AILVOROC.dat

AIICH
groupName     GroupNCCT
targetDisease ICH
TPFThresh     0.9361
FPFThresh     None
rocFile       ../inputs/AIICHROC.dat
```

`Vendor1` in Listing 4 defines the target condition. A user may add or remove a block for including more or less AIs in the simulation.

Parameter	Range	Description
`groupName`	String	Patient group in which AI device operates
`targetDisease`	String	Condition targeted by AI device
`TPFThresh`	[0, 1]	True-positive fraction (TPF), i.e., Sensitivity of the operating point.
`FPFThresh`	[0, 1]	False-positive fraction (FPF), i.e., 1-Specificity of the operating point. If None, `rocFile` must be provided.
`rocFile`	A .dat file (e.g., ../inputs/exampleROC.dat)	A text file with two columns (first: TPF, second: FPF). If None, `FPFThresh` must be provided.

3.1.3.1 `groupName`

Each AI should have a group name within which the AI analyzes all images. This must match one of the group names in the disease parameters in Listing 3.

3.1.3.2 `targetDisease`

Each AI device targets one disease. The target disease must match one of the disease names of the corresponding group as shown in Listing 3.

3.1.3.3 `TPFThresh`

The true-positive fraction (TPF) threshold operating point, also known as sensitivity, is defined to be the ratio of the number of AI-positive patient images to the number of diseased patient images called by the radiologist. Similar to `prevalence`, this definition of sensitivity only consider images within the corresponding group. By definition, `TPFThresh` is between 0 and 1. A `TPFThresh` of 0 means that the CADt calls all diseased images negative, whereas 1 implies that all diseased images are correctly labelled as positive by the CADt device.

Listing 5: The first 10 lines of `inputs/exampleROC.dat`

```
0.0000,0.0000
0.0002,0.2500
0.0043,0.5000
0.0441,0.7500
0.1912,0.9000
0.1955,0.9020
0.2000,0.9040
0.2047,0.9061
0.2094,0.9081
0.2143,0.9101
```

3.1.3.4 `FPFThresh`

The false-positive fraction (FPF) threshold operating point, also known as 1 - specificity, is defined to be the ratio of the number of AI-positive patient images to the number of non-diseased patient images called by the radiologist. Again, this definition of sensitivity only consider images within the corresponding group. If `FPFThresh` is provided, its value must be between 0 and 1. A `FPFThresh` of 0 means that the CADt correctly calls all non-diseased images negative, whereas 1 implies that all non-diseased images are mistakenly labelled as positive by the CADt device. If `FPFThresh` is not provided, the software will expects a valid `rocFile`.

3.1.3.5 `rocFile`

`rocFile` is required if `FPFThresh` is set to None. This parameter states the path and filename of an existing text file that has two columns (with no headers) as shown in Listing 5. The first column is the false-positive fraction, and the second is the true-positive fraction. As described in Section 4.1.1, these values are used to reconstruct the diseased and non-diseased distributions assuming a bi-normal distribution. Please note that, if a user has an empirical Receiver-Operating Characteristic curve (ROC curve), it may be better to run the software at different {TPF, FPF} using `TPFThresh` instead of using the assumption of an underlying bi-normal distribution.

3.1.4 Simulation parameters

The parameters in Listing 6 and Table 3.1.4.6 are related to simulation runs.

3.1.4.1 `nTrials`

`nTrials` specifies the number of trials to be generated.

3.1.4.2 `nPatientsTarget`

This parameter specifies the target number of patient images to be generated per trial. This value, along with the input `traffic`, is used to estimate the number of days required to reach the target number. As a result, the actual number of simulated patient images may be slightly different than the target number.

Listing 6: Parameters related to a simulation run in `inputs/config.dat`

3.1.4.3 `verbose`

In general, it is recommended to turn on `verbose`. Printout messages include repeating user inputs and checking whether simulated runs have similar values for disease prevalence, positive predictive value, and negative predictive values. The time it took to run all trials are also printed, as well as a summary wait-time results from the first trial.

3.1.4.4 `statsFile`

`statsFile` contains both the file path and name to store the output results from the simulation runs as well as other theoretical values. Please visit Section 3.2.1 for more information on what information is stored and how to access them.

3.1.4.5 `runTimeFile`

If provided a file path and name, a text file summarizing the runtime performance is dumped. This contains how many times each function is called, how long it takes, etc.

3.1.4.6 `plotPath`

If provided a file path, plots associated to the simulation run are generated and stored in the provided path. See Section 3.2.2 for the descriptions of output plots.

Parameter	Range	Description
`nTrials`	Integer ≥ 1	Number of trials to be simulated.
`nPatientsTarget`	Integer ≥ 1	Number of targeted patient images.
`verbose`	[True, False]	If True, print out information as the simulation runs.
`statsFile`	e.g. ../outputs/stats/stats.p	If provided a filename, simulation and theoretical predictions will be stored in a pickled file.
`runTimeFile`	e.g. ../outputs/stats/runTime.txt	If provided, dumps out a runtime performance summary.
`plotPath`	e.g. ../outputs/plots	If provided, generate plots.

3.2 Output

This section explains the output files generated if the corresponding input parameters are turned on and/or provided. The exact outputs depend on the input user settings. However, to provide examples, the snapshots in the Sections 3.2.1 through 3.2.3 are generated using the parameters in Listings 2, 3, and 4.

3.2.1 Pickled python dictionary

The most important file is the pickled dictionary with a default name of `stats.p`. Listing 7 shows a code snippet on how to open the pickled file.

Listing 7: iPython example code to open the pickled python dictionary

```
In [1]: import pickle

In [2]: with open ('../outputs/stats/stats.p', 'rb') as f:
   ...:     adict = pickle.load (f)
   ...: f.close()

In [3]: adict.keys()
Out[3]: dict_keys(['params', 'lpatients', 'wpatients', 'lstats', 'wtstats'])
```

Listing 8: iPython example code to retrieve theoretical waiting time

```
In [4]: from show_results import get_theory_results

In [5]: theory = get_theory_results (adict['params'], display=True)

                                wait_time_theory  time_diff_theory
queue_subgroup
without CADt non-interrupting            120.00               NaN
with CADt (priority) LVO                  18.08           -101.92
with CADt (priority) SAH                 144.45             24.45
with CADt (priority) ICH                  16.24           -103.76
with CADt (hierarchical) LVO              13.10           -106.90
with CADt (hierarchical) SAH             144.57             24.57
with CADt (hierarchical) ICH              17.89           -102.11
```

3.2.1.1 `params`

The `params` key contains all information about the run, including user input parameters, theoretical predictions, various conditional probabilities, and the simulation outputs. Specifically, *adict['params']['theory']* contains the theoretical values (in minutes) of average waiting times for different subgroups in both with and without CADt scenarios as well as the average time difference between the two scenarios. The script `show_results.py` included the get_theory_results function that extracts and displays theoretical predictions from *adict['params']['theory']*. Listing 8 shows how this function can be called.

3.2.1.2 `wpatients`

To access the individual simulated patient, one can use the `wpatients` key, as shown in Listing 9. Each row in the dataframe is a simulated patient. Each simulated patient has a unique ID based on the ordering of its arrival time. The first patient image arrived in the system has a *patient_id* of '0000000', and the next one is '0000001', etc. For each patient, their disease status, AI calls by each AI, the timestamps at which the exam was opened and closed for each queue type (without-CADt, with-CADt priority, and with-CADt hierarchical), and the corresponding wait-times are included. The script `show_results.py` included the get_sim_results function that extracts and displays average waiting time and time difference (with respect to without-CADt) from *adict['wpatients']*. Listing 10 shows how this function can be called.

To display both simulation and theory results, one can use the display_results function that extracts in `show_results.py`, as shown in Listing 11 .

Listing 9: iPython example code to retrieve individual simulated patient statistics

```
In [8]: adict['wpatients']
Out[8]:
       is_interrupting  is_diseased  is_positive  ...   trial_id patient_id  delta
0                False        False        False  ...  trial_000    0000000   0.00
1                False         True        False  ...  trial_000    0000001   0.00
2                False        False        False  ...  trial_000    0000002  23.05
3                False        False        False  ...  trial_000    0000003  23.05
4                False        False        False  ...  trial_000    0000004  23.05
...                ...          ...          ...  ...        ...        ...    ...
99822            False        False        False  ...  trial_000    0099822   0.00
99823            False        False        False  ...  trial_000    0099823  16.66
99824            False         True         True  ...  trial_000    0099824 -31.42
99825            False        False        False  ...  trial_000    0099825   0.00
99826            False        False        False  ...  trial_000    0099826   0.00

[99827 rows x 23 columns]
```

Listing 10: iPython example code to retrieve simulated waiting time

```
In [6]: from show_results import get_sim_results

In [7]: sim = get_sim_results (adict, display=True)

                                wait_time_sim  time_diff_sim
queue_subgroup
without CADt non-interrupting         112.79            NaN
with CADt (priority) LVO               17.83         -94.96
with CADt (priority) SAH              135.39          22.61
with CADt (priority) ICH               15.95         -96.83
with CADt (hierarchical) LVO           13.42         -99.36
with CADt (hierarchical) SAH          135.45          22.66
with CADt (hierarchical) ICH           17.59         -95.20
```

Listing 11: iPython example code to retrieve both simulated and theoretical waiting time

```
In [9]: from show_results import display_results

In [10]: display_results(adict)

                                wait_time_sim  time_diff_sim  wait_time_theory  time_diff_theory
queue_subgroup
without CADt non-interrupting         112.79            NaN            120.00               NaN
with CADt (priority) LVO               17.83         -94.96             18.08           -101.92
with CADt (priority) SAH              135.39          22.61            144.45             24.45
with CADt (priority) ICH               15.95         -96.83             16.24           -103.76
with CADt (hierarchical) LVO           13.42         -99.36             13.10           -106.90
with CADt (hierarchical) SAH          135.45          22.66            144.57             24.57
with CADt (hierarchical) ICH           17.59         -95.20             17.89           -102.11
```

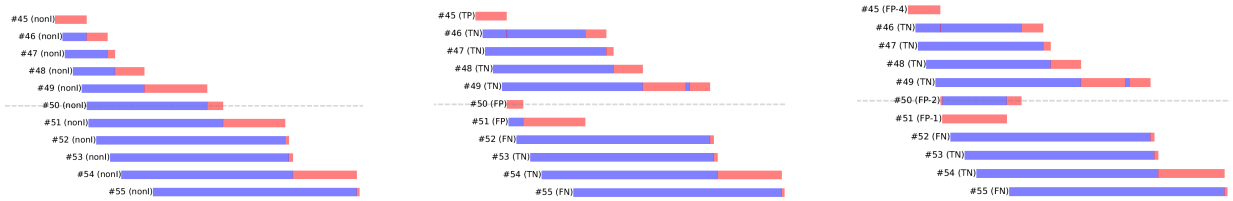

Figure 1: Patient image flows in without-CADt scenario (left); in with-CADt scenario with priority queue (middle); and in with-CADt scenario with hierarchical queue. Each bar represents the timestamps of a patient with an ID (e.g. "#45") followed by its status ("I" for interrupting, "nonI" for non-interrupting, TN" for non-diseased and AI negative, "FP" for non-diseased and AI positive, "TP" for diseased and AI positive, "FN" for diseased and AI negative). For hierarchical queue, the ranking of AI-positive cases is denoted after "TP" and "FP"; 1 is the highest priority, followed by 2, etc. The length of each blue section represents the wait-time period experienced by the patient, whereas the length of each red section is the time period in which the patient image is being reviewed by a radiologist.

3.2.2 Plots

No plots will be generated if the user does not specify a valid `plotPath`. Otherwise, the following plots may be generated.

3.2.2.1 Patient Timings

Three plots related to patient image flows are generated. By default, timestamps of the first 200 patients are displayed, and Figure 1 shows the timing information of a small subset of patients in without-CADt (left), with-CADt with priority queue (middle), and with-CADt with hierarchical queue (right). Patients in each queue have identical status, arrival times, and radiologist read-time. However, their wait-time periods (blue sections) are different between the three queue scenarios, demonstrating the per-patient effects due to the use of CADt devices.

3.2.2.2 ROC of the AI

If `rocFile` is not provided (which is the case in our example), the *ROC_anAI.pdf* will not be generated. Otherwise, Figure 2 shows an example output plot if `inputs/exampleROC.dat` were used. As mentioned in Section 3.1.3.5, a bi-normal assumption is used during the parametrization. Therefore, whenever `rocFile` is provided, it is important to check the goodness of fit (R^2 in the middle plot of Figure 2) and the overall agreement between the input data points and the fitted ROC curve.

3.2.2.3 Number of patients Distributions

Two PDF files are outputted with normalized distributions (to unit area) of the number of patient images in the system for each priority class observed by every in-coming patient image. These distributions are different than the total number of patient images. Right before a new simulated patient enters the system, we record how many interrupting and non-interrupting patients are currently in the system in the without-CADt scenario (see Figure 3). Similarly, Figure 4 shows the normalized distributions of the number of interrupting, AI-positive, and AI-negative images right before a new patient image arrives in the with-CADt scenario. These normalized distributions are, by definition, the state probabilities of the

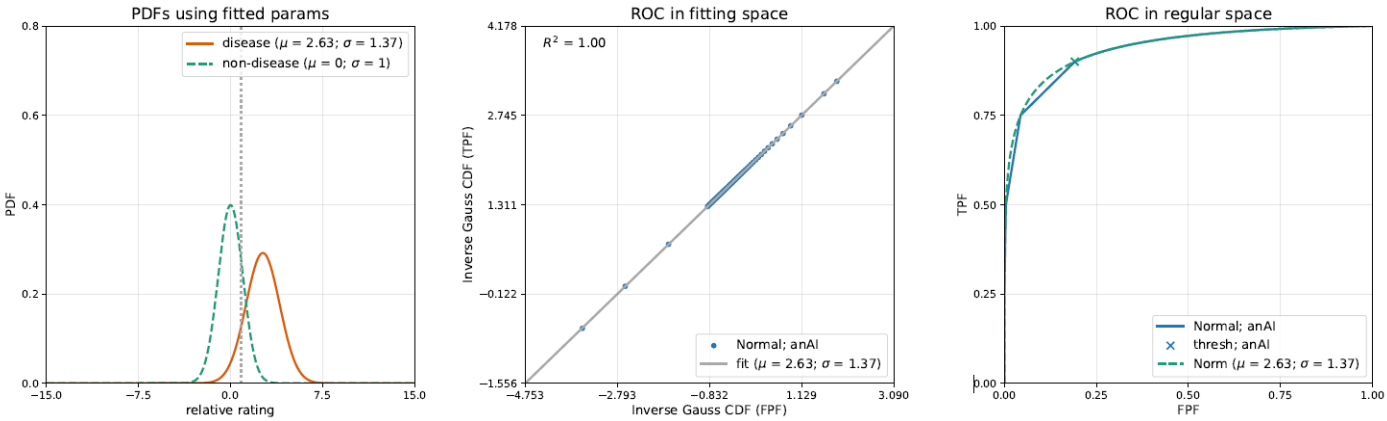

Figure 2: (Left) The relative rating probability distribution function for diseased (orange, solid) and non-diseased (green, dashed). Non-diseased histogram always has a mean μ of 0 and standard deviation σ of 1. Vertical gray dotted line indicates the rating threshold of the corresponding user-input `TPFThresh`. (Middle) The fitted ROC curve in an inverse normal cumulative distribution function space. Blue dots are the converted values from user's input ROC file. Best fit mean μ and standard deviation σ for the diseased rating histogram are stated in the legend, along with the R^2 value as a rough estimate of the goodness of fit. (Right) The same as the middle plot except that it is in a typical ROC space. The green dashed line represent the fitted ROC curve assuming a bi-normal distribution, and the gray cross is the operating point to be used in simulation.

queueing system that are needed to theoretically calculate the average waiting time for each priority class.

3.2.2.4 Distributions related to waiting time

In addition to the distributions of patient image counts, the software can output two PDF files with plots relating to patient wait-times for both priority and hierarchical queues. As an example, Figure 5 shows the theoretical and simulated wait-times for the non-interrupting patient cases in a without-CADt scenario and for all diseased subgroups in both with-CADt priority and hierarchical queues. In this example, a reading queue in which a patient may have one of nine disease conditions or non-diseased, and four CADt devices were used to triage four out of the nine conditions. The vertical spacing between crosses and stars indicate the agreement between simulation and theory.

3.2.3 Runtime statistics

If a valid `runTimeFile` is provided, a summary text file *runTime.txt* is stored stating the number of times each function is called, as well as their runtimes (see Listing 12). This is especially useful for larger simulation runs. However, if running the script in a cluster, it is recommended to turn this functionality off.

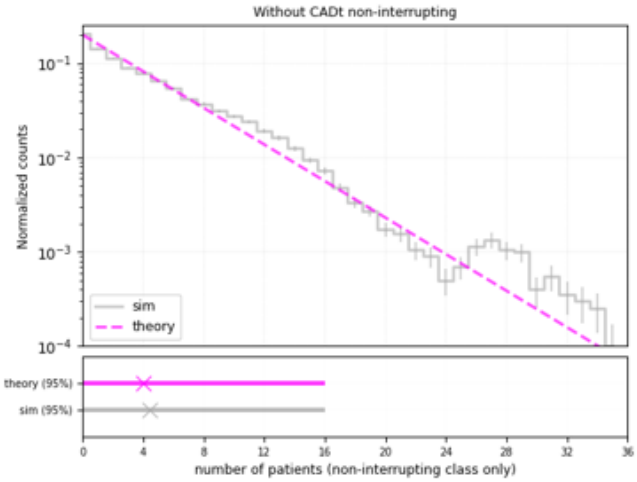

Figure 3: The distribution of the number of non-interrupting patients in system right before a new arrival normalized to unit area in a without-CADt scenario. This distribution is the same as the state probability curve in the theoretical approach. The gray lines and bars are obtained from simulation, and the magenta lines and bars are theoretical predictions. Top plots show the distributions themselves, and the bottom plots show the 95% confidence limits from both theory and simulation. If interrupting patients are included, two plots will be shown.

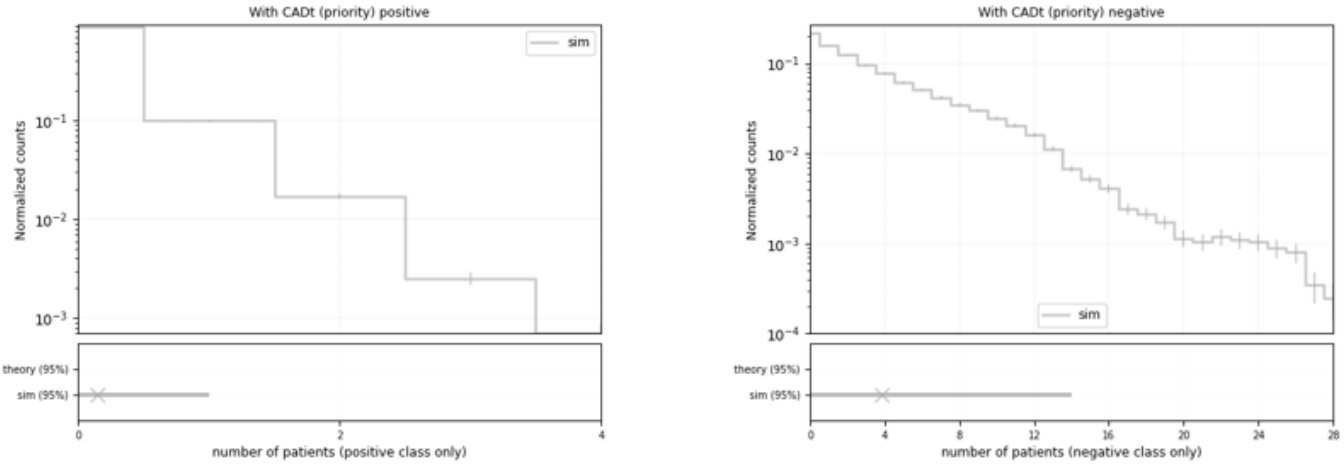

Figure 4: (Left) The distribution of the number of AI-positive patients (middle priority) in system right before a new arrival normalized to unit area in a with-CADt scenario. (Right) The distribution of the number of AI-negative (lowest priority) patients in system right before a new arrival normalized to unit area in a with-CADt scenario. This distribution is the same as the state probability curve in the theoretical approach. The gray lines and bars are obtained from simulation. Top plots show the distributions themselves, and the bottom plots show the 95% confidence limits. Theoretical distribution may or may not be available depending on the input clinical settings.

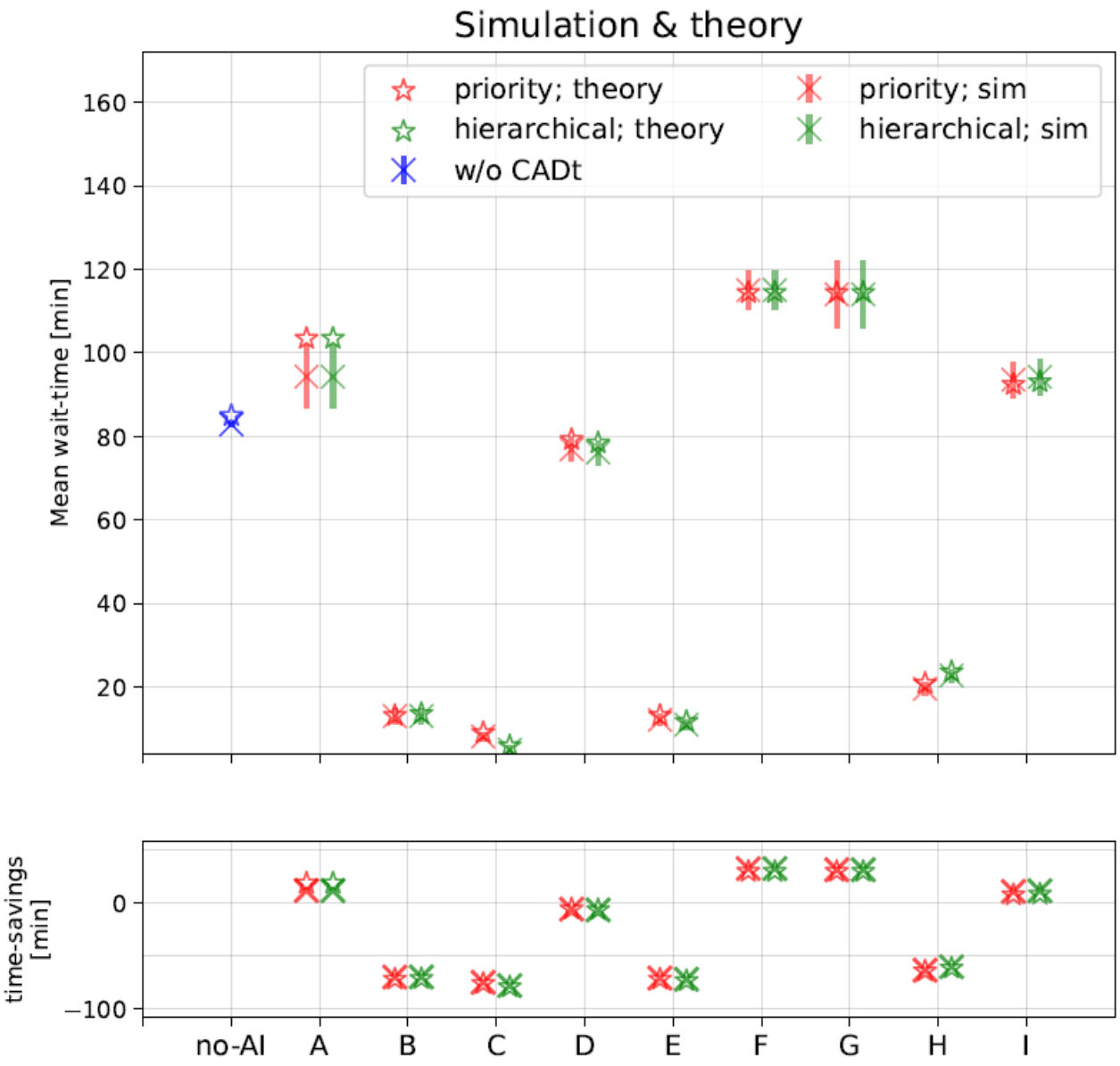

Figure 5: Top plot shows the mean wait-time for different groups of patient images. First data point (in blue) is the theoretical and simulated means (along with 95% confidence intervals) of all non-interrupting cases in a without-CADt scenario. The wait-time of all diseased patient images in a with-CADt priority scenario (in red) and a with-CADt hierarchical scenario (in green) are also plotted. The bottom plot shows the simulated and theoretical time difference for each diseased subgroup between with-CADt and without-CADt scenarios. Negative values indicate a time-savings, and positive values suggest a time delay.

Listing 12: Run time performance text file

```
         452587860 function calls (447840218 primitive calls) in 286.312 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   100858   25.446    0.000  134.865    0.001 {method 'remove' of 'list' objects}
        1    0.000    0.000  128.398  128.398 .\tools\trialGenerator.py:499(simulate_trials)
        1    0.688    0.688  127.864  127.864 .\tools\simulator.py:1493(simulate_queue)
101787264   57.295    0.000  109.466    0.000 .\tools\patient.py:94(__eq__)
    39858    0.371    0.000   65.663    0.002 .\tools\simulator.py:1148(_read_newest_urgent_patient)
204549942   52.485    0.000   52.485    0.000 .\tools\patient.py:142(caseID)
```

4 Methods

4.1 Simulations

The majority of scripts in this tool are related to simulations. A simulation run is performed in the `simulator` class. However, other instances play their roles throughout the simulated patient flow. The goal of this section is to provide a big-picture description for each class involved. For the step-by-step details, please visit the comments and documentations in the individual scripts.

4.1.1 CADt AI diagnostic performance

`tools/AI.py` is the class to simulate a CADt device's diagnostic performance. It can be initialized by an input operating point or by an input sensitivity with an ROC curve via a text file. The main job of an AI instance is to label whether a non-interrupting image is AI-positive or AI-negative, given the radiologist diagnosis (diseased vs non-diseased). If an ROC curve is provided, the AI call is a random rating from the corresponding (diseased or non-diseased) normal distribution. If the random rating is above the rating threshold, the image is labeled as AI-positive. If an operating point (both True-Positive Fraction (TPF) and False-Positive Fraction (FPF)) are provided, a random score is generated using a uniform distribution between 0 and 1. For a diseased image, if the random score is below the sensitivity threshold, the AI instance calls it positive. On the other hand, the AI instance calls a non-diseased image positive if the random score is larger than the specificity threshold.

4.1.2 Patient

`tools/patient.py` encapsulates information of a simulated patient image, including its interrupting status, disease (radiologist diagnosis) status and disease type, and AI call by each AI. Whether a patient image is an interrupting case is determined by a Bernoulli distribution with a success probability equal to input parameter `fractionED`. If the patient image is indeed an interrupting case, it will have no disease status or AI call status. Otherwise, the disease status is obtained by a random number from another binomial distribution with a success probability equal to input parameter `prevalence`. Once the disease status is determined, an AI instance can generate the AI call label to determine if this patient is AI-positive or AI-negative.

Besides status, a patient instance also keeps a record of timestamps such as trigger (arrival) time, radiologist's reading time, waiting time, total time in system, etc. Every patient is put into

both without-CADt, with-CADt with priority queue, and with-CADt with hierarchical queue scenarios simutaneously, implying that the arrival and radiologist's reading time of this very same patient are identical in the all scenarios. To determined the arrival time, an inter-arrival time is first randomly generated from an exponential distribution defined by the arrival rate that was calculated from the input parameters `traffic` and the effective radiologist's service time. We then take the arrival timestamp of the previous patient image and add the randomly determined inter-arrival time to obtain the arrival timestamp of the current patient. The radiologist's reading time for this patient is determined by a radiologist instance (see Section 4.1.3).

4.1.3 Radiologist

`tools/radiologist.py` encapsulates a server (radiologist) class. Each radiologist has a name and a patient instance that the radiologist is currently handling. Throughout the simulation, the current patient instance is updated as the radiologist gets interrupted, reads a new case, and/or closes the current case. The radiologist class also contains a function that determines how much time it takes to review this patient. This time is randomly generated based on an exponential distribution characterized by a rate, which can either be the service rate for interrupting subgroup, a diseased subgroup, or non-diseased subgroup.

4.1.4 Simulator

This `tools/simulator.py` is the most important class of the simulation software. It simulates a workflow of patient images in a radiology department. When an in-coming patient is initiated, an ID based on the arrival order is assigned, as well as other randomly generated properties such as arrival timestamp, radiologist's service time, and interrupting/group/disease/AI call status. This simulated patient is simultaneously put into three queues:

- without-CADt (i.e. FIFO, or "first-in-first-out") scenario
- with-CADt with priority queue scenario in which all AI-positive patients are prioritized over AI-negative patients
- with-CADt with hierarchical queue scenario in which AI-positive patients of higher disease rank are prioritized over AI-positive patiehts of lower disease rank

Queues in all scenarios are either preemptive-resume or non-preemptive depending on user input. In the without-CADt scenario, the higher priority images are the interrupting cases that are not reviewed by the CADt device and require immediate attention. The lower priority cases are non-interrupting cases in the reading queue and can be grouped by the radiologist's diagnosis (diseased vs non-diseased). When a CADt device is used, these non-interrupting cases will be analyzed by the CADt device upon their arrival to the system. Therefore, in addition to the diseased vs non-diseased subgroups, these non-interrupting cases can also be categorized into AI-positive and AI-negative subgroups in the with-CADt scenario, and hence the true-positive, false-positive, true-negative, and false-negative subgroups. In the hierarchical scenario, non-interrupting cases can even further be grouped into AI-positive cases for a specific AI; or true-diseased cases for a specific disease.

At the beginning of the simulation run, the first patient image arrives, seeing an empty system with no images. This very first patient image is then immediately reviewed by a radiologist, and the timestamp at which the radiologist starts reviewing the case is recorded. The waiting time (i.e. the time difference

between the radiologist-start-reading timestamp and the arrival timestamp) is therefore zero. When the radiologist is reviewing the first patient, another patient image may arrive, seeing that the system has already had one case of a specific priority class (either interrupting/non-interrupting in without-CADt scenario; interrupting/AI-positive/AI-negative in with-CADt priority scenario; or interrupting/AI-A positive/AI-B positive/ ... /AI-negative in the with-CADt hierarchical scenario). Upon each new arrival, the number of cases in each priority class is recorded to check the agreement of state probabilities between simulation and theory. If only one radiologist is available, the second patient has a higher priority class than the first patient, and we are in a preemptive queueing scheme, the radiologist will stop reading the first patient image (putting it back to the queue) and immediately review the second patient. For the first patient, the timestamp at which it is put back into the queue is recorded, and its waiting period starts. If non-preemptive queueing is turned on, the radiologist will finish their read of the first patient before reviewing the second patient. If the second patient has the same or lower priority than the first patient, the second patient will be put in the reading queue. When the radiologist finishes reviewing the first patient image, it will start reviewing the second patient image. If more than one radiologist is available, the second radiologist will start reviewing the second patient image regardless of the priority of the second patient image.

As more patients are simulated, the queue changes whenever an action is needed. For example, when a radiologist reads a new case, the patient image is removed from the queue, and when the radiologist closes a case, the image is removed from the system. When a radiologist is interrupted by a higher priority image (in preemptive queueing), the lower priority image is put back into the queue. It is worth noting that a simulated patient is not immediately put into the queue when first initialized. When a new patient is initialized, its arrival time may be after the radiologist closes several cases. Therefore, before reaching the new arrival timestamp, several patient cases may be removed from the system.

Each of the three scenarios (without-CADt, with-CADt with priority queue, and with-CADt with hierarchical queue) has its own reading queue list. For without-CADt scenario, the queue has two priority classes: interrupting (with a higher priority) and non-interrupting (lower priority). For with-CADt (priority) scenario, images have three priority classes: interrupting (highest priority), AI-positive (middle priority), and AI-negative (lowest priority). For with-CADt (hierarchical) scenario, the number of priority classes depend on the disease rankings: interrupting (highest priority), AI-A-positive, AI-B-positive, ..., and AI-negative (lowest priority). Within each priority class, patient images are ordered by the arrival timestamps.

While simulating the patient image flow, the `simulator` class keeps track of information from all patients in the system, including the disease/group/interrupting status, the AI calls, the reading duration (i.e. service time), the various timestamps (arrival, opened and closed, interrupted, etc.), waiting time, etc. When a patient arrives, the number of patients in the system per priority class right before the patient's arrival are recorded. Besides functions that simulate patient image flows, this class also contains functions that extract all patients of a specified subgroup in either of the three scenarios.

4.1.5 Hierarchy

`tools/hierarchy.py` encapsulates all properties and calculations of a hierarchical queue in both non-preemptive and preemptive scenarios, including necessary parameter calculations for the theoretical wait-time for the hierarchical scenario which is obtained in `tools/calculator.py`. A hierarchy class dictionary is created to capture properties of the disease and its priority relative to other diseases

Listing 13: Ipython example code to calculate the theoretical predictions without running simulation

```
In [11]: from tools import inputHandler

In [12]: configFile = '../inputs/config.dat'

In [13]: params, _, _ = inputHandler.read_args (configFile)

In [16]: from show_results import get_theory_results

In [17]: theory = get_theory_results (params, display=True)

                                wait_time_theory  time_diff_theory
queue_subgroup
without CADt non-interrupting            120.00               NaN
with CADt (priority) LVO                  18.08           -101.92
with CADt (priority) SAH                 144.45             24.45
with CADt (priority) ICH                  16.24           -103.76
with CADt (hierarchical) LVO              13.10           -106.90
with CADt (hierarchical) SAH             144.57             24.57
with CADt (hierarchical) ICH              17.89           -102.11
```

in queue. In particular, each disease is assigned a rank which is used in the simulation to further reorder AI-positive patients in the queue.

4.1.6 Trial generator

`tools/trialGenerator.py` generates trials. This class passes user input parameters and calls the simulator class to simulate reading flow. At the end of running all trials, this script contains function that calculates the statistics (e.g. means and 95% confidence limits) of waiting time and number of patients in system observed by an in-coming patient for each subgroup.

4.2 Theoretical calculation

To obtain the theoretical predictions without running simulation, see Listing 13. `tools/calculator.py` script contains functions and classes to compute the state probabilities, average waiting time, and mean time difference between with- and without-CADt scenarios. For detailed information on the calculation, please visit [1]. Specifically, the `get_theory_waitTime()` function is designed to provide the theoretical predictions of average waiting time and mean wait-time difference for a specific subgroup of patient images.

5 Future directions

If you are interested in providing feedback and in collaborating with us on the software, please contact Elim Thompson.

References

[1] Yee Lam Elim Thompson et al. “Applying queueing theory to evaluate wait-time-savings of triage algorithms”. en. In: *Queueing Syst.* 108.3-4 (Sept. 2024), pp. 579–610.

[2] Yee Lam Elim Thompson et al. “Wait-time-saving analysis and clinical effectiveness of Computer-Aided Triage and Notification (CADt) devices based on queueing theory”. In: *Medical Imaging 2022: Image Perception, Observer Performance, and Technology Assessment*. Ed. by Claudia R Mello-Thoms and Sian Taylor-Phillips. San Diego, United States: SPIE, Apr. 2022.

6 List of Acronyms