Module Interface Specification for OAR

Hunter Ceranic

March 15, 2024

1 Revision History

Date	Version	Notes
March 8, 2024	1.0	Initial Revision
March 15, 2024	1.1	Changes made according to Dr. Smith's Initial Comments

2 Symbols, Abbreviations and Acronyms

See SRS Documentation (Ceranic, 2024) at https://github.com/cer-hunter/OAR-CAS741/blob/main/docs/SRS/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for the OAR (Optical Alphabet Recognition) program. This document specifies how each module interfaces with other parts of the program.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/cer-hunter/OAR-CAS741.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by OAR.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbf{Z}$	a number without a fractional component in (-∞, ∞)
positive integer	$\mathbf{Z}_+$	a positive integer ($\mathbf{Z}$) in (0, ∞)
unsigned 8-bit integer	$\mathbf{U}$	a number without a fractional component in (0, 255)
natural number	$\mathbf{N}$	a number without a fractional component in [1, ∞)
real	$\mathbf{R}$	any number in (-∞, ∞)
positive real	$\mathbf{R}_+$	any real number ($\mathbf{R}$) in (0, ∞)
image data	$\mathbf{I}_{x,y}$	`data`: a one-dimensional array of unsigned 8-bit integers in order from the top-left pixel of the image to the bottom-right pixel. Has a `width`: $\mathbf{Z}_+$ width of x and `height`: $\mathbf{Z}_+$ height of y.
matrix	$\mathbf{M}_{x,y}$	`data`: a one-dimensional array of real numbers, with a `width`: $\mathbf{Z}_+$ width of x and `height`: $\mathbf{Z}_+$ height of y.

The specification of OAR uses some derived data types: sequences, strings, tuples, and booleans. Sequences are lists filled with elements of the same data type. Strings are sequences

of characters. Tuples contain a list of values, potentially of different types. Booleans can be represented in different ways but only have two possible values: true or false. In addition, OAR uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Application Control Graphics Display Output Calculator Input Data Read Input Classifier OAR Model Data OAR Model Equations OAR Model Training OAR Model Testing
Software Decision Module	Confusion Matrix Input Processing Graphical User Interface

Table 1: Module Hierarchy

6 MIS of Application Control Module

6.1 Module

`main`

6.2 Uses

- Graphics Display Module Specification (7)

6.3 Syntax

6.3.1 Exported Constants

None.

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
`main`	-	-	-

6.4 Semantics

6.4.1 State Variables

None.

6.4.2 Environment Variables

- Screen ($\mathbf{Z}_+$ for width and height in pixels)

6.4.3 Assumptions

The GUI Display is running and displayed without issue.

6.4.4 Access Routine Semantics

`main`():

- transition: Initializes the program and the Graphics Display module 7

6.4.5 Local Functions

None.

7 MIS of Graphics Display

7.1 Module

`display`

7.2 Uses

- Hardware-Hiding Module
- Input Data Read Module (9)
- Output Module (8)
- Graphical User interface (GUI) Module (17)

7.3 Syntax

7.3.1 Exported Constants

- `GUI_BOXSIZE`: A value ($\mathbf{Z}_+$) describing both width and height (in pixels) used for the image display "box" (currently always a square)

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
`display`	`inputImage` ($\mathbf{I}_{x,y}$), `resultLabel` (String), `resultConf` (String)	`displayWindow`, event handlers	-

7.4 Semantics

7.4.1 State Variables

- inputImage: The processed input image and given by the Output Calculator Module 8 as $\mathbf{I}_{x,y}$
- resultLabel: The label output as given by the Output Calculator Module 8 as a string.
- resultConf: The confidence probability output as given by the Output Calculator Module 8 as a string.

7.4.2 Environment Variables

- Keyboard ($\mathbf{Z}_+$ for keycodes describing the key pressed)
- Mouse (Boolean for click state and $\mathbf{Z}_+$ for cursor position)
- Screen ($\mathbf{Z}_+$ for width and height in pixels)
- `displayWindow` ($\mathbf{Z}_+$ for width and height in pixels) for the application interface
- `inputButton` (String for a file location) to provide an input image from the file system

7.4.3 Assumptions

- The file system is able to read and provide the image file as specified by the user through an OS file-open dialog. Otherwise if the file is not found, denied access or cancelled, no changes should occur.
- The OS is able to provide basic text or number input user controls with some basic built-in validation, and is able to handle events from Human Interface Devices (HIDs such as mouse, keyboard or touchscreen).

7.4.4 Access Routine Semantics

`display`():

- transition: Sets up user control event handlers (i.e., mouse clicks or drag, button presses, text input change, ...) as needed for the user input. Calls the Input Data Read Module 9 to accept a base image and Output Calculator Module 8 to classify the input image. The input image and output results are then pushed to the `displayWindow`.

7.4.5 Local Functions

None.

8 MIS of Output Calculator Module

8.1 Module

`output`

8.2 Uses

- Input Classifier Module Specification (10)

8.3 Syntax

8.3.1 Exported Constants

- `DEC_FIXED`: Used for fixed decimal number length rounding (ex. "5.8923" at fixed length "2" results in "5.89")

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
`output`	`inputImage` ($\mathbf{I}_{x,y}$)	`inputImage` ($\mathbf{I}_{x,y}$), `resultLabel` (String), `resultConf` (String)	-

8.4 Semantics

8.4.1 State Variables

None.

8.4.2 Environment Variables

None.

8.4.3 Assumptions

The input image is valid.

8.4.4 Access Routine Semantics

`output`():

- transition: Passes the `inputImage` to the Input Classifier module 10 to return the `resultLabel` and `confPercent`, and converts the float value `confPercent` into a string
- output: The `inputImage` as ($\mathbf{I}_{x,y}$) and the predicted label `resultLabel` and confidence in the classification `resultConf` as strings, to be used by the Graphics Display module 7.

8.4.5 Local Functions

None.

9 MIS of Input Data Read Module

9.1 Module

`input`

9.2 Uses

- Input Processing Module Specification (16)

9.3 Syntax

9.3.1 Exported Constants

- `MAX_SIZE`: A value ($\mathbf{Z}_+$) describing both width and height (in pixels) for maximum acceptable size of the input image (currently a square).
- `MIN_SIZE`: A value ($\mathbf{Z}_+$) describing both width and height (in pixels) for minimum acceptable size of the input image (currently a square).
- `MODEL_IMG_SIZE`: The required size of the input image matrix to be used by the classification model $\mathbf{I}_{x,y}$.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
`input`	`baseImage` ($\mathbf{I}_{x,y}$)	`inputImage` ($\mathbf{I}_{x,y}$)	`InvalidSize`, `InvalidFormat`

9.4 Semantics

9.4.1 State Variables

None.

9.4.2 Environment Variables

- `baseImage`: The base input image in the form of a .BMP .JPG or .PNG file.

9.4.3 Assumptions

The input path location for base image is valid, readable and accessible.

9.4.4 Access Routine Semantics

`input`(inputPath):

- output: Pre-processed `inputImage` as $\mathbf{I}_{x,y}$ ready for classification.
- exception: `InvalidSize` if the size of the base image is outside of the range of `MIN_SIZE` to `MAX_SIZE`, `InvalidFormat` if the file type at the `inputPath` is not supported by the OAR Program (according to R1)

9.4.5 Local Functions

None.

10 MIS of Input Classifier Module

10.1 Module

`classify`

10.2 Uses

- OAR Model Data Module Specification (11)
- OAR Model Equations Module Specification (12)

10.3 Syntax

10.3.1 Exported Constants

None.

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
`classify`	`inputImage` ($\mathbf{I}_{x,y}$), `oarModel` ($\mathbf{M}_{x,y}$)	`resultLabel` (String), `confPercent` ($\mathbf{R}_{+}$)	-

10.4 Semantics

10.4.1 State Variables

None.

10.4.2 Environment Variables

None.

10.4.3 Assumptions

The input image is valid.

10.4.4 Access Routine Semantics

`classify`(inputImage):

- output: The predicted label of the input image `resultLabel` as a String and the associated confidence level in the prediction as a $\mathbf{R}_+$, as described in IM2.

10.4.5 Local Functions

None.

11 MIS of OAR Model Data Module

11.1 Module

`model`

11.2 Uses

- OAR Model Testing Module Specification (14)

11.3 Syntax

11.3.1 Exported Constants

None.

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
`model`	-	-	-

11.4 Semantics

11.4.1 State Variables

- `oarModel`: Data structure designed to store the matrix of weights and biases associated with the trained OAR classification model as a tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$.

- `performance`: Data structure designed to store the matrix of performance values associated with each label of the trained OAR classification model as a $\mathbf{M}_{x,y}$.

11.4.2 Environment Variables

None.

11.4.3 Assumptions

None.

11.4.4 Access Routine Semantics

`model`():

- transition: This module is a simple tuple ($\mathbf{M}_{x,y}$ and $\mathbf{R}$) data structure for storing the OAR classification model weights and biases and corresponding performance matrix.

11.4.5 Local Functions

None.

12 MIS of OAR Model Equations Module

12.1 Module

`oarUtils`

12.2 Uses

None.

12.3 Syntax

12.3.1 Exported Constants

None.

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
sigmoid	sigIn ($\mathbf{R}$)	sigOut ($\mathbf{R}$)	-
logLossFunc	trueVal ($\mathbf{R}$), predVal ($\mathbf{R}$)	logLoss ($\mathbf{R}$)	-
predict	inputImage ($\mathbf{I}_{x,y}$), weight ($\mathbf{M}_{x,y}$), bias ($\mathbf{R}$)	predVal ($\mathbf{R}$)	-
gradientW	inputImage ($\mathbf{I}_{x,y}$), trueVal ($\mathbf{R}$), weight ($\mathbf{M}_{x,y}$), bias ($\mathbf{R}$), regParam ($\mathbf{R}$), trainSize ($\mathbf{Z}_+$)	gradW ($\mathbf{R}$)	-
gradientB	inputImage ($\mathbf{I}_{x,y}$), trueVal ($\mathbf{R}$), weight ($\mathbf{M}_{x,y}$), bias ($\mathbf{R}$)	gradB ($\mathbf{R}$)	-
gradientDescent	inputImage ($\mathbf{I}_{x,y}$), trueVal ($\mathbf{R}$), weight ($\mathbf{M}_{x,y}$), bias ($\mathbf{R}$), regParam ($\mathbf{R}_+$), learnRate ($\mathbf{R}_+$), trainSize ($\mathbf{Z}_+$)	weight ($\mathbf{M}_{x,y}$), bias ($\mathbf{R}$)	-

12.4 Semantics

12.4.1 State Variables

None.

12.4.2 Environment Variables

None.

12.4.3 Assumptions

The input image is valid.

12.4.4 Access Routine Semantics

sigmoid(sigIn):

- output: Computes the sigmoid function of sigIn according to TM1 and returns the value sigOut as $\mathbf{R}$.

logLossFunc(trueVal, predVal):

- output: Computes the value of the log loss function using trueVal and predVal according to TM2 and returns the value logLoss as $\mathbf{R}$.

`predict`(inputImage, weight, bias):

- output: Calls `sigmoid` to calculate the predicted label of the `inputImage` using `weight` and `bias` from the model according to GD1.

`gradientW`(inputImage, trueVal, weight, bias, regParam, trainSize):

- output: Calculates the gradient of the log loss function using all the input variables with respect to the weights according to according to GD2.

`gradientB`(inputImage, trueVal, weight, bias):

- output: Calculates the gradient of the log loss function using all the input variables with respect to the bias according to according to GD3.

`gradientDescent`(inputImage, trueVal, weight, bias, regParam, learnRate, trainSize):

- output: Calls `gradientW` and `gradientB` to execute the algorithm to update the weights and biases using all the input variables for one epoch according to IM1.

12.4.5 Local Functions

None.

13 MIS of OAR Model Training Module

13.1 Module

`train`

13.2 Uses

- OAR Model Equations Module Specification (12)

13.3 Syntax

13.3.1 Exported Constants

- `LAMBDA`: The regularization parameter used during model training as $\mathbf{R}_+$.
- `ALPHA`: The learning rate parameter used during model training as $\mathbf{R}_+$.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
`train`	`trainSet` ($\mathbf{M}_{x,y}$),`trainVals` ($\mathbf{M}_{x,y}$), `weightBiasMatrix` (tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$), `trainSize` ($\mathbf{Z}_+$)	`weightBiasMatrix` (tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$)	-

13.4 Semantics

13.4.1 State Variables

- `weight`: the weight portion of the `weightBiasMatrix` input as $\mathbf{M}_{x,y}$ for each label.
- `bias`: the bias portion of the `weightBiasMatrix` input as $\mathbf{R}$ for each label.

13.4.2 Environment Variables

None.

13.4.3 Assumptions

None.

13.4.4 Access Routine Semantics

`train`(trainSet, trainVals, weightBiasMatrix, trainSize):

- output: Executes the algorithm for training the weights and biases of the model given the training dataset information and functions from the OAR Model Equations Module (12), and returns the updated version of the `weightBiasMatrix` as a tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$.

13.4.5 Local Functions

None.

14 MIS of OAR Model Testing Module

14.1 Module

`test`

14.2 Uses

- OAR Model Data Module Specification (12)
- OAR Model Equations Module Specification (12)
- OAR Model Training Module Specification (13)
- Confusion Matrix Module Specification (15)

14.3 Syntax

14.3.1 Exported Constants

- `EPOCHS`: The the number of times the model training regression algorithm is ran as $\mathbf{Z}_+$.
- `TRAIN_SIZE`: The size of the training data used during model training as $\mathbf{Z}_+$.
- `TEST_SIZE`: The size of the testing data used during model training as $\mathbf{Z}_+$.
- `DATA_IMG_SIZE`: The size of the input image matrix used during model training as $\mathbf{I}_{x,y}$.
- `LABELS`: The set of possible labels for the classification model as a tuple of Strings.

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
`test`	-	`oarModel` (tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$), `performance` ($\mathbf{M}_{x,y}$)	-

14.4 Semantics

14.4.1 State Variables

- `dataSet`: The set of pre-processed images and their associated labels that will be used for training the classification model as a tuple of $\mathbf{I}_{x,y}$ and $\mathbf{Z}_+$.
- `weight`: the weight portion of the `weightBiasMatrix` input as $\mathbf{M}_{x,y}$ for each label.
- `bias`: the bias portion of the `weightBiasMatrix` input as $\mathbf{R}$ for each label.
- `predictionData`: matrix which tracks the number predictions made for each test image as $\mathbf{M}_{x,y}$.

14.4.2 Environment Variables

- `dataSetPath`: the File System path or location as a string pointing to where the data set is located.

14.4.3 Assumptions

The `dataSetPath` is valid, readible and accessible.

14.4.4 Access Routine Semantics

`test`():

- transition: Calls routine to train and test a model of weights and biases.
- output: The matrix of weights and biases representing the `oarModel` as a tuple of $\mathbf{M}_{x,y}$ and $\mathbf{R}$, and the associated `performance` of the model as $\mathbf{M}_{x,y}$.

14.4.5 Local Functions

- `splitDataSet`(dataSet):
  - output: Takes the `dataSet` as an input and splits it into distinct parts for training and testing the classification model. The following values are output:
    - `trainData`: The part of the `dataSet` used to train the model as $\mathbf{M}_{x,y}$.
    - `trainVals`: The part of the `dataSet` corresponding to the true labels of the `trainData` as $\mathbf{M}_{x,y}$.
    - `testData`: The part of the `dataSet` used to test the model as $\mathbf{M}_{x,y}$.
    - `testVals`: The part of the `dataSet` corresponding to the true labels of the `testData` as $\mathbf{M}_{x,y}$.
- `evalModelData`(oarModel, confusionMatrix):
  - output: Evaluates the performance of the `oarModel` based on the `confusionMatrix` and returns the `performance` matrix that will be written to the OAR Model Data Module 11.

15 MIS of Confusion Matrix Module

15.1 Module

`confMatrix`

15.2 Uses

None.

15.3 Syntax

15.3.1 Exported Constants

None.

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
`confMatrix`	`predictionData` ($\mathbf{M}_{x,y}$)	`confusionMatrix` ($\mathbf{M}_{x,y}$)	
`printConfMatrix`	`confusionMatrix` ($\mathbf{M}_{x,y}$)	`confusionMatrix` ($\mathbf{I}_{x,y}$)	

15.4 Semantics

15.4.1 State Variables

None.

15.4.2 Environment Variables

None.

15.4.3 Assumptions

None.

15.4.4 Access Routine Semantics

`confMatrix`(predictionData):

- output: Outputs the confusion matrix representing the performance of the model based on the `predictionData` as $\mathbf{M}_{x,y}$.

`printConfMatrix`(confusionMatrix):

- output: Outputs the confusion matrix representing the performance of the model based on the `predictionData` as a graphical image ($\mathbf{I}_{x,y}$).

15.4.5 Local Functions

None.

16 MIS of Input Processing Module

16.1 Module

`preprocess`

16.2 Uses

None.

16.3 Syntax

16.3.1 Exported Constants

None.

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
`preprocess`	`baseImage` ($\mathbf{I}_{x,y}$)	`inputImage` ($\mathbf{I}_{x,y}$)	-

16.4 Semantics

16.4.1 State Variables

None.

16.4.2 Environment Variables

None.

16.4.3 Assumptions

The format and parameters of the base image was already verified to be within the requirements.

16.4.4 Access Routine Semantics

`preprocess`(baseImage):

- output: Performs transformations on the `baseImage` such that the resulting `inputImage` as $\mathbf{I}_{x,y}$, is normalized to be able to be used by the classification model.

16.4.5 Local Functions

None.

17 MIS of Graphical User Interface

17.1 Module

`gui`

17.2 Uses

None.

17.3 Syntax

17.3.1 Exported Constants

None.

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
`gui`	None	None	-

17.4 Semantics

17.4.1 State Variables

None.

17.4.2 Environment Variables

- Keyboard ($\mathbf{Z}_+$ for keycodes describing the key pressed)
- Mouse (Boolean for click state and $\mathbf{Z}_+$ for cursor position)
- Screen ($\mathbf{Z}_+$ for width and height in pixels)
- Button (String for a file location) to provide an input image from the file system

17.4.3 Assumptions

None.

17.4.4 Access Routine Semantics

`gui`():

- transition: Provides methods from the TKinter Library to build and deploy a GUI to Graphics Display Module 7

17.4.5 Local Functions

None.

References

Hunter Ceranic. System requirements specification. https://github.com/cer-hunter/OAR-CAS741/blob/main/docs/SRS/SRS.pdf, 2024.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.