Module Interface Specification for OCRacle

Phillip Tran

March 17, 2025

1 Revision History

Date	Version	Notes
March 13, 2025	1.0	Initial Document Creation

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/ptrandev/OCRacle/blob/main/docs/SRS/SRS.pdf for symbols, abbreviations and acronyms.

Contents

3 Introduction

The following document details the Module Interface Specifications for OCRacle, an optical character recognition (OCR) program for identifying Latin alphabet characters.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/ptrandev/OCRacle/.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

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 OCRacle.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
unsigned 8-bit integer	uint8	an integer in [0, 255]
bounded real	$\mathbb{R}_{\text{min} \leq x \leq \text{max}}$	a real number in the range [min, max]
File	File	a file object supported by Python

The specification of OCRacle uses some derived data types: sequences, strings, and tuples. 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. In addition, OCRacle 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	
Behaviour-Hiding	Input Format Module
	Model Output Module
	Model Training Module
	Model Testing Module
	Prediction Model Module
	Application Module
	Performance Metrics Module
Software Decision	Image Preprocessing Module
	Performance Metrics Module
	Graphical User Interface Module

Table 1: Module Hierarchy

6 MIS of Application Module

6.1 Module

main

6.2 Uses

- Graphical User Interface Module 14
- Input Format Module 7
- Model Output Module 8

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

The Graphical User Interface module is responsible for handling all user inputs and outputs. The Application module is responsible for coordinating the interaction between the GUI, Input Format, and Model Output modules.

6.4.4 Access Routine Semantics

main():

- transition: The application is started and the user is able to interact with the GUI to input images and view the model's predictions.

6.4.5 Local Functions

N/A

7 MIS of Input Format Module

7.1 Module

input

7.2 Uses

- Image Preprocessing Module 12

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
input	filePath: string	imageMatrix: $\text{uint8}_{28\times 28}$	DimensionsError, FormatError

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

input(filePath):

- output: imageMatrix := preprocessing(file)

- exception: A DimensionsError exception is thrown if the input image exceeds the bounds of the n_{min}, n_{max}, m_{min}, and m_{max} parameters as specified by the SRS. A FormatError exception is thrown if the input image is not one of the supported formats.

7.4.5 Local Functions

readImage(filePath):

- output: File object. A file object is returned that can be used to read the image file by the preprocessing module.
- exception: A FormatError is thrown if the input image is not one of the supported formats.

8 MIS of Model Output Module

8.1 Module

output

8.2 Uses

- Prediction Model Module 11

8.3 Syntax

8.3.1 Exported Constants

- LABELS: {'A',...,'Z'}

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
output	imageMatrix: $\text{uint8}_{28\times 28}$	prediction: char, confidenceMatrix: $\mathbb{R}^{26}_{0\leq x\leq 1}$	-

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

We assume that the image matrix has been preprocessed as specified in the SRS before being passed to the prediction model used by the output module.

8.4.4 Access Routine Semantics

output(imageMatrix):

- output: prediction, confidenceMatrix := predict(imageMatrix)

8.4.5 Local Functions

N/A

9 MIS of Model Training Module

9.1 Module

train

9.2 Uses

- Performance Metrics Module 13
- Model Output Module 8
- Input Format Module 7

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
train	-	model: $\mathrm{float32}_{\mathrm{m\times n}}$	-

9.4 Semantics

9.4.1 State Variables

- dataset: $(\mathrm{uint8}_{28\times 28}$, char)[]

- epochs: $\mathbb{N}$
- batch_size: $\mathbb{N}$
- learning_rate: $\mathbb{R}$
- epsilon: $\mathbb{R}$
- time_step: $\mathbb{R}$
- first_moment: $\mathbb{R}$
- second_moment: $\mathbb{R}$
- beta1: $\mathbb{R}$
- beta2: $\mathbb{R}$
- gradient: $\text{float32}_{\text{m}\times\text{n}}$
- theta: $\text{float32}_{\text{m}\times\text{n}}$
- y_hat: $float32_{\text{m}\times\text{n}}$

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

train():

- transition: The model is trained on the dataset using the hyperparameters epochs, batch_size, learning_rate, epsilon, time_step, first_moment, second_moment, beta1, and beta2. These are manipulated via the ADAM optimizer as specified by the SRS.
- output: model := $f_{train}()$

9.4.5 Local Functions

relu(x):

- output: result := max(0, x)

softmax(x):

- output: result := $e^x / \sum_{i=1}^{n} e^{x_i}$

cnn_forward(x, theta):

- transition: y_hat := $f_{cnn\_forward}(x, theta)$. This function calculates the output of the model given the input x and the weights theta.

cnn_backward(x, y, y_hat, theta):

- transition: gradient := $f_{cnn\_backward}(x, y, y\_hat, theta)$. This function calculates the gradient of the model given the input x, the target y, the output y_hat, and the weights theta.

updateTheta(gradient, theta, time_step, first_moment, second_moment, beta1, beta2, epsilon):

- transition: theta := $f_{updateTheta}(gradient, theta, time\_step, first\_moment, second\_moment, beta1, beta$ This function updates the weights of the model using the ADAM optimizer.

10 MIS of Model Testing Module

10.1 Module

test

10.2 Uses

- Performance Metrics Module 13
- Model Output Module 8
- Input Format Module 7

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
test	-	loss: $\mathbb{R}_{0\leq x\leq 1}$, accuracy: $\mathbb{R}_{0\leq x\leq 1}$, confusionMatrix: ${\mathbb{R}_{0\leq x\leq 1}}^{26\times 26}$	-

10.4 Semantics

10.4.1 State Variables

- dataset: (uint8$_{28\times 28}$, char)[]
- loss: $\mathbb{R}_{0\leq x\leq 1}$
- accuracy: $\mathbb{R}_{0\leq x\leq 1}$
- confusionMatrix: ${\mathbb{R}_{0\leq x\leq 1}}^{26\times 26}$

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

The model has a valid weight format for the test dataset and was trained using the same hyperparameters as the training module.

10.4.4 Access Routine Semantics

test():

- transition: The model is tested on a dataset of unseen images. The loss, accuracy, and confusion matrix are calculated using the model's predictions.
- output: loss, accuracy, confusionMatrix := crossEntropyLoss(prediction, target), accuracy(prediction, target), confusionMatrix(prediction, target)

10.4.5 Local Functions

N/A

11 MIS of Prediction Model Module

11.1 Module

model

11.2 Uses

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
predict	imageMatrix: $\mathrm{uint8}_{28\times 28}$	confidenceMatrix: $\mathbb{R}^{26}_{0\leq x\leq 1}$	-

11.4 Semantics

11.4.1 State Variables

- model: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

We assume that the input image has been preprocessed as specified in the SRS before being passed to the prediction model.

11.4.4 Access Routine Semantics

predict(imageMatrix):

- output: confidenceMatrix := $f_{predict}(imageMatrix)$

11.4.5 Local Functions

N/A

12 MIS of Image Preprocessing Module

12.1 Module

preprocessing

12.2 Uses

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
preprocessing	file : File	imageMatrix: $\text{uint8}_{28\times28}$	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

All input data has been validated by the Input Format Module before being processed by the Image Preprocessing Module.

12.4.4 Access Routine Semantics

preprocessing(file):

- output: image $:= uint8_{28\times28}$. The input file has been transformed such that it conforms to the preprocessing requirements as specified by the SRS.

12.4.5 Local Functions

N/A

13 MIS of Performance Metrics Module

13.1 Module

performance

13.2 Uses

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
crossEntropyLoss	prediction: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$, target: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$	loss: $\mathbb{R}_{0\leq \mathrm{x}\leq 1}$	-
accuracy	prediction: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$, target: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$	accuracy: $\mathbb{R}_{0\leq \mathrm{x}\leq 1}$	-
confusionMatrix	prediction: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$, target: $\mathrm{float32}_{\mathrm{m}\times\mathrm{n}}$	confusionMatrix: ${\mathbb{R}_{0\leq \mathrm{x}\leq 1}}^{26\times 26}$	-

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

We assume that the prediction and target matrices are of the same dimensions.

13.4.4 Access Routine Semantics

crossEntropyLoss(prediction, target):

- output: loss := $f_{crossEntropyLoss}(prediction, target)$

accuracy(prediction, target):

- output: accuracy := $f_{accuracy}(prediction, target)$

confusionMatrix(prediction, target):

- output: confusionMatrix := $f_{confusionMatrix}(prediction, target)$

13.4.5 Local Functions

N/A

14 MIS of Graphical User Interface Module

14.1 Module

gui

14.2 Uses

- Hardware-Hiding Module

14.3 Syntax

14.3.1 Exported Constants

N/A

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
-	-	-	-

14.4 Semantics

14.4.1 State Variables

N/A

14.4.2 Environment Variables

- displayWindow (dimensions: $\mathbb{Z}_+ \times \mathbb{Z}_+$)
- keyboard (keypress: char)
- mouse (location : $\mathbb{Z}_+ \times \mathbb{Z}_+$, click: bool)

14.4.3 Assumptions

This behavior will be sufficiently abstracted by the use of Python Notebook, which will handle all UI elements and interactions with the user. As a result, the GUI modules does not need to have any pre-defined state variables or access routines.

14.4.4 Access Routine Semantics

N/A

14.4.5 Local Functions

N/A

References

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.

15 Appendix

[Extra information if required —SS]