Software Requirements Specification for ProgName: OAR - Optical Alphabet Recognition

Hunter Ceranic

February 19, 2024

Contents

Revision History

Date	Version	Notes
February 5, 2024	1.0	Initial Version
February 19, 2024	1.1	Updated with Feedback from Primary and Secondary Reviewers
February 19, 2024	1.2	Updated with better understanding of bias value
March 17, 2024	1.3	Updated with better understanding of Testing Instance Model

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
px	pixel	picture element

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	unit	description
$\mathbf{x}_{n\times m}$	Pixel	Matrix representing pixels of the input image with n rows and m columns,
$\mathbf{y}_{n\times m}$	Unitless	Matrix representing the probability values of the pixels of the actual label of the input image
$\hat{\mathbf{y}}_{n\times m}$	Unitless	Matrix representing the probability values of the pixels of the predicted label of the input image
$\mathbf{z}_{n\times m}$	Unitless	Matrix representing the combination of pixels from the input image and weights and biaseses, used in the Predicted Label Matrix $\hat{\mathbf{y}}$
$x_{i,j}$	Unitless	An entry in the input matrix $\mathbf{x}$ (at i, j) which represents the level of intensity of the pixel
$\mathbf{w}_{m\times n}$	Unitless	Matrix of weights that is used by the classifier model with m rows and n columns
b	Unitless	Bias value that is used by the classifier model
λ	Unitless	Regularization Parameter that affects flexibilty/variance of the model
α	Unitless	Learning rate of classifier model
σ	Unitless	Represents the Sigmoid Function
L	Unitless	Represents the Log Loss Function

1.3 Abbreviations and Acronyms

symbol	description
2D	Two Dimensional
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
OAR	Optical Alphabet Recognition

1.4 Mathematical Notation

In this project documentation variables which represent matrices are indicated by bolded letters (ie. $\mathbf{x}$), whereas all other variables are represented by regular printed letters.

2 Introduction

Images often contain important information to be used in many different modern applications. One such type of information that can be contained within images are readable characters, and the motivation for the Optical Alphabet Recognition project is to create a system from scratch, that can classify upper-case English letter characters characters accurately. More information about the problem itself can be found within the Problem Statement document. Furthermore, a foundational goal of this project is to be useful for educating people about the fundamentals of image classification.

The following section is an overview of the Software Requirement Specifications for the OAR project, specifically outlining the purpose of the document, the scope of the requirements, characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

The purpose of this SRS document is to establish and clearly communicate the requirements, limitations, definitions, and models, in regards to the OAR project. This document will also serve as basis for reference for the rest of the documents supporting the project.

2.2 Scope of Requirements

The scope for this project is constrained to the recognition of images of singular, hand-written and printed, capital-letter English alphabet characters, in their correct orientations. Character images can be black and white, grayscale or RGB images. However, this does not include hand-written cursive letters. Any pre-processing needed will be handled by supporting libraries, to perform the calculations described in this document. See the assumptions section, 4.2.1, for further details.

2.3 Characteristics of Intended Reader

Readers of this document are intended to have a basic level understanding of 2D image processing techniques (equivalent to a standard undergraduate course on machine learning). However, readers with a prerequisite of at least level one university mathematics knowledge (specifically including matrix linear algebra) will be able to understand and follow most of the concepts in the document, as the project is also intended to be a useful tools for learning about image processing.

2.4 Organization of Document

This SRS document contains an introduction to the problem being addressed by the software, and the overarching goals of the project. It is to be used as reference for readers based on the SRS template by Smith and Lai (2005); Smith et al. (2007). The scope, terminology, definitions and models used in the project are outlined, and more details about the specifics

of the problem and the explored solution are presented. Requirements, potential future changes and traceability information are also defined.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

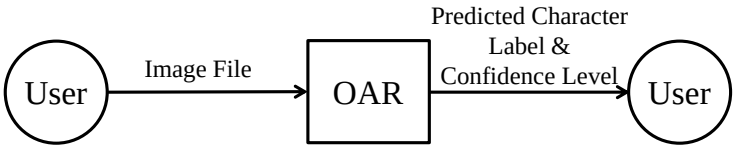

Figure 1: System Context

- User Responsibilities:
  - Provide an image of a character to the program in the proper orientation.
  - Be able to interact with the software program using a keyboard, computer mouse, and monitor.
  - Run the software on a system with the capabilities to sufficently process computer graphics.
- OAR Responsibilities:
  - Provide the ability to load or change image files to be classified.

- Detect and warn about invalid or corrupt image files, upon input.
- Display the confidence in the result of classified characters.
- Provide a user interface which is able to take user input at anytime, with minimal down time where the program is unresponsive.

3.2 User Characteristics

The end user of the OAR software is anyone who can use character recognition software or wants to learn about it. This may include the intended readers of this document (as described in Section 2.3), however none of those reader requirements are neccessary to use the software.

3.3 System Constraints

The only real world design constraint for this project is that all input images must have the character in their proper legible orientation.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

OAR is intended to address the simple problem of converting data stored in images into usable string data through the use of optical character recognition. Specifically OAR focuses on classifying images of individual characters according to their corresponding English language alphabet characters. It should be able to do this and communicate to users the confidence level in how correctly it classified the image. Furthermore, this project is also intended to demystify the underlying theories behind optical character recognition for students and newer programmers, so it can be used as a learning tool.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- Classification: A method of assigning pre-existing labels to processed images.
- Confidence Level: The probability that the label assigned to an image was correctly identified.

- Image: A representation of visual information that can be represented by a 2D matrix where each entry represents a pixel with an intensity value.
- Intensity: The value of a pixel representing it's grayscale colour, on a scale from 0 (which represents black) to 255 (which represents white).
- Label: A value which associates the result of image classification with a real world category, such as English alphabet letters.
- Normalization: Altering the aspect ratio and scale of images to some arbitrarily set standard, Karandish (2022).

4.1.2 Physical System Description

The physical system of OAR, as shown in Figure 2, includes the following elements:

PS1: The image that is being input.

Figure 2: A sample input image depicting the letter A

To maintain consistency within the software the images will be normalized to some set of n×m pixels such that the trained model can classify any given input image. The physical representation of what this resizing will look like is seen in Figure 3.

Figure 3: A sample input image depicting the letter A, after being processed

4.1.3 Goal Statements

Given an input image, the goal statements are:

GS1: Calculate and display the predicted corresponding character label for the unknown character in the image.

GS2: Calculate and display the confidence level that the predicted label is correct.

4.2 Solution Characteristics Specification

This section provides the assumptions, theoretical models, instance models, general definitions, and data constraints. The information in this section is intended to reduce ambiguity about the project and to present the problem in clear mathethematical or logical terms.

The instance models that govern the OAR porject are presented in Subsection 4.2.5. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The image will already be normalized and pre-processed, as required by TM2, prior to any classification calculations.

A2: The dataset used to train the software model as described by IM1, will only contain labels of capital letter characters.

A3: The dataset used for training the software model as described by IM1, will contain a balanced distribution of every label that can be identified by the program.

A4: All input images will display one single, properly oriented character, to be utilized by IM2.

4.2.2 Theoretical Models

This section focuses on the general equations and laws that OAR is based on.

Number	TM1
Label	**Sigmoid Function**
Equation	$\sigma(z) = \frac{1}{1+e^{-z}}$
Description	The sigmoid function takes input z and returns a value between 0 and 1. This function will be used to calculate the confidence level of the predicted value, to satisfy GS2.
Notes	The value of z can be a scalar or matrix.
Source	Turin (2020)
Ref. By	GD1

Number	TM2
Label	**Log Loss Function**
Equation	$L_{(y,\hat{y})} = -\frac{1}{n} \cdot \sum_{i=1}^{n} (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$
Description	y represents the actual label value $\hat{y}$ represents the predicted label value n is the total number of input values i is the current step in the summation $$L_{\hat{y}} = \begin{cases} -\log(\hat{y}) & \text{, if } y = 1 \\ -\log(1 - \hat{y}) & \text{, if } y = 0 \end{cases}$$ This function utilizes the properties of the natural logarithm to determine how similar the actual value y is to the predicted value $\hat{y}$.
Notes	The values of y and $\hat{y}$ can be scalar or matrices, but they must be the of the same size (see A1).
Source	Turin (2020)
Ref. By	A1, GD1, GD2, GD3, IM1, IM2

Number	TM3
Label	**Chain Rule**
Equation	$\frac{d}{dx}[f(u)] = \frac{d}{du}[f(u)]\frac{du}{dx}$
Description	This formula is used to derive the composition of two differentiable functions.
Source	Turin (2020)
Ref. By	GD2, GD3

4.2.3 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Sigmoid Function with Input Image and Weights**
SI Units	Unitless
Equation	$\hat{\mathbf{y}} = \sigma(z) = \frac{1}{1+(e^{-\mathbf{z}})}$, where $z = \mathbf{w}^T \cdot \mathbf{x} + b$
Description	This is a refinement of the Sigmoid function (see TM1) to represent how it will be utilized as the basis of the model. $\mathbf{x}$ is the input image matrix. $\mathbf{w}$ is the matrix of weights b is the matrix of biases Note that the predicted value $\hat{\mathbf{y}}$ from TM2 will be determined from the sigmoid function.
Source	Turin (2020)
Ref. By	GD2, GD3, IM1, IM2

Number	GD2
Label	**Gradient of Log Loss Function with respect to w**
SI Units	Unitless
Equation	$\nabla_{\mathbf{w}}(\mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \cdot \log(1 - \hat{\mathbf{y}}_i)) = \mathbf{x}_n \cdot (\mathbf{y}_n - \hat{\mathbf{y}}_n)$
Description	Here we derive the Log Loss Function with respect to $\mathbf{w}$ using chain rule from TM3, with $\hat{\mathbf{y}}$ defined using the Sigmoid Function (see GD1), so that we can minimize the Log Loss Function to train the model (see IM1).
Source	Turin (2020); Sharma (2022a)
Ref. By	IM1

Number	GD3
Label	**Gradient of Log Loss Function with respect to b**
SI Units	Unitless
Equation	$\nabla_b(\mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \cdot \log(1 - \hat{\mathbf{y}}_i)) = \mathbf{y}_n - \hat{\mathbf{y}}_n$
Description	Here we derive the Log Loss Function with respect to b using chain rule from TM3, with $\hat{\mathbf{y}}$ defined using the Sigmoid Function (see GD1), so that we can minimize the Log Loss Function to train the model (see IM1).
Source	Turin (2020); Sharma (2022a)
Ref. By	IM1

Detailed Derivation of Gradients

There are three steps to derive the Gradients of the Log Loss Function, using TM3. First the outer-most derivative:

$$[-\mathbf{y} \cdot \log(\hat{\mathbf{y}})]' = \frac{-\mathbf{y}}{\hat{\mathbf{y}}},$$

and,

$$[-(1 - \mathbf{y}) \cdot \log(1 - \hat{\mathbf{y}})]' = \frac{(1 - \mathbf{y})}{(1 - \hat{\mathbf{y}})},$$

which combines to,

$$\frac{-\mathbf{y}}{\hat{\mathbf{y}}} + \frac{(1 - \mathbf{y})}{(1 - \hat{\mathbf{y}})} = \frac{(\hat{\mathbf{y}} - \mathbf{y})}{(\hat{\mathbf{y}} \cdot (1 - \hat{\mathbf{y}}))}$$

Then we take the derivative of $\hat{\mathbf{y}} = \sigma((z))$ using the following identities:

$$1.\ 1 - \sigma(z) = 1 - \frac{e^z}{(1+e^z)} = \frac{1}{(1+e^z)} = \sigma(-z)$$

$$2.\frac{d}{dz} \cdot \sigma(z) = \frac{d}{dz} \cdot (1 + e^z)^{-1} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})} \cdot \frac{1}{(1+e^{-z})} = \sigma(-z) \cdot \sigma(z) = (1 - \sigma(z)) \cdot \sigma(z)$$

Which gives $\mathbf{y}'_n = \mathbf{y}_n \cdot (1 - \mathbf{y}_n)$
Then finally we take the last derivative in the chain, which is where the gradient derivations differentiate:

$$\text{For GD2: } \frac{d}{d\mathbf{w}} \cdot (b + \mathbf{x}_1 \cdot \mathbf{w}_1 + ... + \mathbf{x}_n \cdot \mathbf{w}_n) = \mathbf{x}_i$$
$$\text{For GD3: } \frac{d}{db} \cdot (b + \mathbf{x}_1 \cdot \mathbf{w}_1 + ... + \mathbf{x}_n \cdot \mathbf{w}_n) = 1$$

Then each derivative in the chain is multiplied together for $\nabla_{\mathbf{w}}$ and ∇_b respectively, resulting in the final formulas seen in GD2 and GD3.

4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given.

Number	DD1
Label	**Learning Rate**
Symbol	α
SI Units	Unitless
Equation	Not Applicable
Description	Learning Rate is the value that is used to help the model minimize the Log Loss function effectively at each step. While there is no concrete equation to determine the learning rate it is suggested that fine-tuning is done by increasing or decreasing the value by a factor of 10.
Sources	Sharma (2022b)
Ref. By	IM1

Number	DD2
Label	**Regularization Parameter**
Symbol	λ
SI Units	Unitless
Equation	Not Applicable
Description	The regularization parameter is a value that is used to help the model minimize the Log Loss function to reduce overfitting. While there is no concrete equation to determine the regularization parameter it is suggested that fine-tuning is done by increasing or decreasing the value by a factor of 10.
Sources	Sharma (2022b)
Ref. By	IM1

4.2.5 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals GS1 and GS2 are solved by IM2. IM1 is used to train the model so that IM2 can be used reliably and efficiently.

Number	IM1
Label	**Model Training via Gradient Descent**
Input	$\mathbf{x}_n$, $\mathbf{y}_n$, $\hat{\mathbf{y}}_n = \frac{1}{1+(e^{(\mathbf{w}^T \cdot \mathbf{x}+b)})}$, α from DD1, λ from DD2, N which is the number of training images The input is constrained so that A2 and A3 can be satisfied.
Output	$\mathbf{w}, b$
Description	Using gradient descent we minimize the Log Loss Function to optimize the weights and biases for better predictions This is executed in the following order, over multiple epochs (t) to further optimize the values: • 1. For each set of training images, calculate the following: – 1.1. $d\mathbf{w}^{(t)} = \mathbf{x}_n \cdot (\mathbf{y}_n - \hat{\mathbf{y}}_n) - \frac{\lambda}{N} \cdot \mathbf{w}^{(t)^2}$ – 1.2. $db^{(t)} = \mathbf{y}_n - \hat{\mathbf{y}}_n$ – 1.3. $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha \cdot d\mathbf{w}^{(t)}$ – 1.4. $b^{(t+1)} = b^{(t)} + \alpha \cdot db^{(t)}$ • 2. Calculate the Log Loss and test the results using the updated weights.
Sources	Turin (2020); Sharma (2022a)
Ref. By	A2, A3, IM2, R3

Number	IM2
Label	**Label Classification**
Input	$\mathbf{x}$, $\mathbf{y}_n$, $\mathbf{w}$, b
Output	$\hat{\mathbf{y}}$, and the string representation of the corresponding predicted label
Description	Using the optimized weights and biases from IM1 and combining that with an unknown input $\mathbf{x}$ in the Sigmoid function (see GD1), $\hat{\mathbf{y}}$ is calculated. Then comparing across the Log Likelihood Function values (see TM2) of the 26 possible labels, the most likely label can be predicted. In addition the confidence level in the prediction can be calculated as the output of the Log Likelihood function for the related predicted label.
Sources	Turin (2020); Sharma (2022a)
Ref. By	R4, R5

4.2.6 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
$x_{i,j}$	-	$\min x_{i,j} \leq x_{i,j} \leq \max x_{i,j}$	128	-
$\mathbf{x}^1_{n\times m}$	-	$\min n, m \leq n, m \leq \max n, m$	1024×1024	-

[1] The constraints on the input image matrix are strictly on it's size $n \times m$ not on the matrix itself.

Table 2: Specification Parameter Values

Var	Value
min $x_{i,j}$	0
max $x_{i,j}$	255
min $n \times m$	20 px $\times$ 20px
max $n \times m$	4096 px $\times$ 4096px

4.2.7 Properties of a Correct Solution

A correct solution must come in the form of one of the 26 possible labels, {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}, as well as an associated confidence level, as a percentage, that the label assigned is correct.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: Accept an input image in the following formats:

- PNG (Portable Net Graphics)
- JPEG (Joint Photographic Experts Group)
- BMP (Bitmap)

R2: The input image shall be pre-processed to normalize data to be used in calculations.

R3: OAR shall provide a trained model that can accurately classify images using 26 different labels (see IM1).

R4: Calculate and display the classified label for the input image (see IM2).

R5: Calculate and display the confidence level that the assigned label is correct (see IM2).

5.2 Nonfunctional Requirements

NFR1: **Accuracy** The accuracy of the computed label confidence should be consistent enough that the results could be considered comparable to other existing solutions for this problem. As such, the accuracy of the model on the training data set should be output in the form of a confusion matrix. In addition the confidence level output has to be calculated in a similar way to existing solutions so that the output can be verified and compared to other algorithms.

NFR2: **Usability** The user should be able to intuitively use the software and understand the output that is displayed. To accomplish this the software should be user-friendly, simple, and require little setup. A user survey may be conducted to verify the software's percieved usability.

NFR3: **Maintainability** The code should be clear and understandable to make further development by other programmers possible with little hassle, as well as to help educate newer programmers as to how image classification works. Adding further features, or understanding how the software presently works should not be a difficult task.

NFR4: **Portability** The software should be cross-platform (Windows, Linux, MacOS) with little to no setup required. This can be in the the form of an executable python-based application.

5.3 Rationale

The rationale for A1 to limit the scope of the project to the problem of classification, rather then including other problems such as normalization, given that GS1 and GS2 are the goals of the software which return an output to the user.

6 Likely Changes

The changes listed below are likely to be implemented as the software is developed further.

LC1: Expand the number of classification labels to include lower-case English letter alphabet, numerical, and punctuation characters.

LC2: Allow the user to specify whether the image is being input from a camera or from a pre-existing file on their system.

7 Unlikely Changes

LC3: Expand the functionality of the program to classify whole English words and sentences instead of just individual characters.

LC4: Provide the choice of different image classification algorithms.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

	A1	A2	A3	A4
TM1				
TM2				
TM3				
GD1				
GD2				
GD3				
DD1				
DD2			X	
IM1	X	X	X	X
IM2	X			X
LC1		X	X	X
LC2				X
LC3		X	X	X
LC4	X			X

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items

9 Values of Auxiliary Constants

There are no auxiliary constants defined so this section is not applicable for this project.

	TM1	TM2	TM3	GD1	GD2	GD3	DD1	DD2	IM1	IM2
TM1				X	X	X			X	X
TM2					X	X			X	X
TM3					X	X			X	
GD1									X	X
GD2									X	
GD3									X	
DD1									X	
DD2									X	
IM1										
IM2										X

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM2	R1	R2	R3	R4	R5	NFR1	NFR2	NFR3	NFR4
IM1											
IM2											
R1											
R2											
R3	X										
R4		X									
R5		X									
NFR1		X									
NFR2		X	X								
NFR3											
NFR4											

Table 5: Traceability Matrix Showing the Connections Between Requirements and Instance Models

References

Forough Karandish. The comprehensive guide to optical character recognition (ocr). https://moov.ai/en/blog/optical-character-recognition-ocr, 2022.

Heena Sharma. Logistic regression-python implementation from scratch without using sklearn. https://heena-sharma.medium.com/logistic-regression-python-implementation-from-scratch-without-using-sklearn-d3fca7d3dae7,

2022a.

Heena Sharma. Regularization in machine learning. https://heena-sharma.medium.com/regularization-in-machine-learning-e7445c3166cd, 2022b.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

Arseny Turin. Logistic regression from scratch. https://towardsdatascience.com/logistic-regression-from-scratch-69db4f587e17, 2020.