Module Interface Specification for ANN (Artificial Neural Network)

Tanya Djavehrpour

March 19, 2024

1 Revision History

Date	Version	Notes
March 18	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation Djavaherpour (2024b) at HERE.

Contents

3 Introduction

The following document details the Module Interface Specifications for ANN (Artificial Neural Network). This document specifies how every module is interfacing with every other parts.

Complementary documents include the System Requirement Specifications (SRS) Djavaherpour (2024b) and Module Guide (MG) Djavaherpour (2024a). The full documentation and implementation can be found at Github repository for ANN.

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 ANN.

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 (-∞, ∞)

The specification of ProgName 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, ProgName 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 Djavaherpour (2024a) document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	ANN Control Module Saved ANN Model Module Output Module Input Classifier Module Input Image Module Training Model Module
Software Decision	Input Preparing and Preprocessing Module Data Preparing and Preprocessing Module Training Module Testing Module

Table 1: Module Hierarchy

6 MIS of ANN Control Module

6.1 Module

`main`

6.2 Uses

- Hardware-Hiding Module
- Saved ANN Model Module (7)
- Output Module (8)

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

None.

6.4.3 Assumptions

- The ANN Control Module assumes that the Hardware-Hiding Module, Saved ANN Model Module, and Output Module are implemented according to their specifications. However, it does include error handling to manage unexpected behaviors or failures in these modules.
- The system environment (operating system, hardware) is assumed to be stable. Also, essential libraries and dependencies are presumed to be correctly installed and configured.

6.4.4 Access Routine Semantics

main():

- transition: Initializes the program.

Note: As the ANN Control Module mainly serves as a coordinator between different modules without maintaining its own state or producing output, its primary function is to ensure the correct sequence of operations and interactions between these modules. It relies on the robustness of the called modules' error handling.

6.4.5 Local Functions

None.

7 MIS of Saved ANN Model Module

7.1 Module

`model`

7.2 Uses

- Hardware-Hiding Module
- Training Module (14)

7.3 Syntax

7.3.1 Exported Constants

None.

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
`save_model`	-	`boolean`	`PermissionError`
`load_model`	-	`[weights, biases]`	`FileNotFoundError`

7.4 Semantics

7.4.1 State Variables

- `modelData`: Data structure holding the current ANN model's data. This is an array including weights and biases.

7.4.2 Environment Variables

`modelFile`: A file on the file system where the ANN model data is saved and from where it is loaded.

7.4.3 Assumptions

None.

7.4.4 Access Routine Semantics

`save_model`():

- transition: Writes the current state of the ANN model (weights and biases) to the `modelFile`.
- output: Returns `True` if model is saved successfully.
- exception: Raises `PermissionError` if the module lacks the necessary permissions to write to `modelFile`. It may also raise an IOError if there are issues with the file system, such as insufficient storage space.

`load_model`():

- transition: Reads the model data from `modelFile`.
- output: Returns the trained model based on weights and biases from `modelData`.
- exception: Raises `FileNotFoundError` if `modelFile` does not exist or cannot be accessed. Additionally, an exception may be raised for data corruption or format mismatch, indicating issues with the integrity or compatibility of the stored model data.

7.4.5 Local Functions

None.

8 MIS of Output Module

8.1 Module

`output`

8.2 Uses

- Hardware-Hiding Module
- Input Classifier Module (9)

8.3 Syntax

8.3.1 Exported Constants

None.

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
`set_class`	-	-	-
`set_feedback`	`String`	`boolean`	`PermissionError`

8.4 Semantics

8.4.1 State Variables

None.

8.4.2 Environment Variables

None.

8.4.3 Assumptions

None.

8.4.4 Access Routine Semantics

`get_class`():

- transition: Receives the classification result from the Input Classifier Module (9).
- output: None.
- exception: None.

`set_feedback`():

- transition: Sets user feedback on classification result.
- output: Returns a confirmation message or status after recording the feedback.
- exception: Raises `PermissionError` if the module lacks the necessary permissions to record feedback.

8.4.5 Local Functions

None.

9 MIS of Input Classifier Module

9.1 Module

`classifier`

9.2 Uses

- Hardware-Hiding Module
- Saved ANN Module (7)
- Input Preparing and Preprocessing Module (12)

9.3 Syntax

9.3.1 Exported Constants

None.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
`load_model`	-	-	-
`get_image_pixels`	-	-	-
`get_class`	-	`String`	-

9.4 Semantics

9.4.1 State Variables

- `modelData`: Data structure holding the current ANN model's data. This is an array including weights and biases.
- `imagePixels`: An array of input image's pixels after preprocessing.

9.4.2 Environment Variables

None.

9.4.3 Assumptions

None.

9.4.4 Access Routine Semantics

`load_model`():

- transition: Receives trained model from the Saved ANN Model Module (7) and saves it in `modelData`.
- output: None.
- exception: None.

`set_image_pixels`():

- transition: Receives the array of input image from Input Preparing and Preprocessing Module (12) and saves in `imagePixels`.
- output: None.
- exception: None.

`get_class`():

- transition: Classifies the input image.
- output: The class of the input image.
- exception: None.

10 MIS of Input Image Module

10.1 Module

`input`

10.2 Uses

- Hardware-Hiding Module

10.3 Syntax

10.3.1 Exported Constants

- `HEIGHT`: A value ($\mathbf{Z}_+$) describing acceptable height of input image (currently 32).
- `WIDTH`: A value ($\mathbf{Z}_+$) describing acceptable width of input image (currently 32).
- `IMAGE_FORMAT`: A list of strings (*str*) of acceptable types of input image (currently PNG and JPEG).

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
set_input	inputImagePath	-	FileNotFoundError, InvalidSize, InvalidFormat
get_image_pixels	-	inputImage	-

10.4 Semantics

10.4.1 State Variables

- inputImage: A 3D of the RGB input image.

10.4.2 Environment Variables

None.

10.4.3 Assumptions

None.

10.4.4 Access Routine Semantics

set_input(inputImagePath):

- transition: Receives the input image from end user and saves its matrix as inputImage.
- output: None.
- exception: Raises FileNotFoundError if inputImagePath does not exist or cannot be accessed. Also, InvalidSize is raised when the size of input image in not compatible with HEIGHT or WIDTH. Additionally, InvalidFormat is thrown if the input image's format is not compatible with IMAGE_FORMAT.

get_image_pixels():

- transition: Getter of the input image's matrix.
- output: 3D matrix of inputImage.
- exception: None.

10.4.5 Local Functions

None.

11 MIS of Training Model Module

11.1 Module

`training_model`

11.2 Uses

- Data Preparing and Preprocessing Module (13)

11.3 Syntax

11.3.1 Exported Constants

- `LAYERS_NUMBER`: A value ($\mathbf{Z}_+$) describing the number of neural network's layers.
- `LAYERS_NEURONS`: An array including each layer's number of neurons.

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
`create_gradients`	-	`gradientsArrays`	-

11.4 Semantics

11.4.1 State Variables

None.

11.4.2 Environment Variables

None.

11.4.3 Assumptions

None.

11.4.4 Access Routine Semantics

`create_gradients`():

- transition: Creates zero arrays for all needed gradients based on the `LAYERS_NUMBER` and `LAYERS_NEURONS`.
- output: All gradients' zero arrays.
- exception: None.

11.4.5 Local Functions

None.

12 Input Preparing and Preprocessing Module

12.1 Module

`input_prep`

12.2 Uses

None.

12.3 Syntax

12.3.1 Exported Constants

None.

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
`input_prep`	-	`1D array`	-

12.4 Semantics

12.4.1 State Variables

- `input`: A 3D matrix incling input image pixels.

12.4.2 Environment Variables

None.

12.4.3 Assumptions

None.

12.4.4 Access Routine Semantics

`set_image_pixels`():

- transition: Prepares and preprocess the input image to change it the way model can use it to predicts the class.

- output: An array including prepared and preprocessed input image.
- exception: None.

12.4.5 Local Functions

- `set_image_pixels`():
  - transition: Receives the input image pixels and saves in `input`.
  - output: None.
  - exception: None.
- `rgb2gray`():
  - transition: Converts RGB data (`input`) into grayscale in order to reduce complexity.
  - output: Grayscaled input.
  - exception: None.
- `prep_pixels`(grayInput):
  - transition: Normalizes grayscaled input to change the range of data between 0 and 1.
  - output: Normalized input.
  - exception: None.
- `flat_data`(normalInput):
  - transition: Data is flatten since input image should be vectorized with the size of 1024. After grayscaling input image is a 2D matrix. this should be an 1D array to be used by implemented model.
  - output: flatten input.
  - exception: None.

13 Data Preparing and Preprocessing Module

13.1 Module

`data`

13.2 Uses

None.

13.3 Syntax

13.3.1 Exported Constants

None.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
`get_train_dataset`	-	[trainImages, trainImagesLabels]	`UnableToLoad`
`get_test_dataset`	-	[testImages, testImagesLabels]	`UnableToLoad`

13.4 Semantics

13.4.1 State Variables

- `train_data`: Data structure holding train data images and their labels.
- `test_data`: Data structure holding test data images and their labels.
- `train_label`: An array incling traing data labels.
- `test_label`: An array incling test data labels.

13.4.2 Environment Variables

None.

13.4.3 Assumptions

None.

13.4.4 Access Routine Semantics

`get_train_dataset`():

- transition: Returns the prepared and processed train data.
- output: `train_data`.
- exception: None.

`get_test_dataset`():

- transition: Returns the prepared and processed test data.
- output: `test_data`.
- exception: None.

13.4.5 Local Functions

- `load_data`():
  - transition: Downloads and extracts dataset. After extracting data, this function saves `train_label` and `test_label`.
  - output: Unprocessed train and test data, and their labels. These images are in RGB.
  - exception: This function raises `UnableToDownload`, when there is a problem with downloading or extracting data.
- `rgb2gray`(unprocessedTrainData, unprocessedTestData):
  - transition: Converts RGB data into grayscale in order to reduce complexity.
  - output: Grayscaled train and test data.
  - exception: None.
- `prep_pixels`(grayTrainData, grayTestData):
  - transition: Normalizes grayscaled images to change the range of data between 0 and 1.
  - output: Normalized train and test data.
  - exception: None.
- `flat_data`(normalTrainData, normalTestData):
  - transition: Data is flatten since imagws should be vectorized with the size of 1024. After grayscaling each image is a 2D matrix. These images should be an 1D array to be used in training and testing process by implemented model.
  - output: flatten train and test data.
  - exception: None.
- `shuffle_data`(flatTrainData, flatTestData):
  - transition: Attaches traind data and test data to their labels (`train_label` and `test_label`). Also, shuffle the attached data to improve the performance.
  - output: `train_data` and `test_data`.
  - exception: None.

14 MIS of Training Module

14.1 Module

`train`

14.2 Uses

- Data Preparing and Preprocessing Module (13)

14.3 Syntax

14.3.1 Exported Constants

- `BATCH_SIZE`: The partition size of the dataset for each step of learning, typically a power of two.
- `LEARNING_RATE`: Speed at which the model learns; controls adjustments to the weights.
- `EPOCHS`: Total number of training cycles through the entire dataset.

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
`set_input`	`inputImagePath`	-	`FileNotFoundError`, `InvalidSize`, `InvalidFormat`
`get_image_pixels`	-	`inputImage`	-

14.4 Semantics

14.4.1 State Variables

- `inputImage`: A 3D of the RGB input image.

14.4.2 Environment Variables

None.

14.4.3 Assumptions

None.

14.4.4 Access Routine Semantics

`set_input`(inputImagePath):

- transition: Receives the input image from end user and saves its matrix as `inputImage`.
- output: None.

- exception: Raises `FileNotFoundError` if `inputImagePath` does not exist or cannot be accessed. Also, `InvalidSize` is raised when the size of input image in not compatible with `HEIGHT` or `WIDTH`. Additionally, `InvalidFormat` is thrown if the input image's format is not compatible with `IMAGE_FORMAT`.

`get_image_pixels`():

- transition: Getter of the input image's matrix.
- output: 3D matrix of `inputImage`.
- exception: None.

14.4.5 Local Functions

None.

References

Tanya Djavaherpour. Module guide. https://github.com/tanya-jp/ANN-CAS741/blob/main/docs/Design/SoftArchitecture/MG.pdf, 2024a.

Tanya Djavaherpour. System requirements specification. https://github.com/tanya-jp/ANN-CAS741/blob/main/docs/SRS/SRS.pdf, 2024b.

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]

16 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)