System Verification and Validation Plan for ANN

Tanya Djavaherpour

April 15, 2024

Revision History

Date	Version	Notes
Feb. 15, 2024	1.0	Initial Draft
Mar. 04, 2024	1.1	Modification According to the Feedback 1
Apr. 09, 2024	1.2	Modification According to the Feedback 2
Apr. 10, 2024	1.3	Modification According to Dr. Smith's Feedback
Apr. 15, 2024	1.4	Complete Unit Test

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

symbol	description
ANN	Artificial Neural Network
IM	Instance Model
MG	Module Guide
MIS	Module Interface Specification
N	No
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation
Y	Yes

For complete symbols used within the system, please refer the section 1 in SRS (Djavaherpour, 2024d) document.

This document outlines the Verification and Validation (VnV) plan for the Artificial Neural Network for Image Classification project, as detailed in the SRS (Djavaherpour, 2024d). The purpose of this VnV plan is to ensure that all requirements and objectives outlined in the SRS (Djavaherpour, 2024d) are met with accuracy and efficiency.

The organization of this document starts with the General Information about the ANN in section 2. A verification plan is provided in section 3 and section 4 describes the system tests, including tests for functional and nonfunctional requirements. Test Description is explained in section 5 (will be added).

2 General Information

This Verification and Validation (VnV) Plan outlines the procedures and criteria to be used for ensuring the quality and reliability of the Artificial Neural Network (ANN) developed for Image Classification. Central to this plan is the systematic verification of the software's adherence to its requirements and specifications, as detailed in the Software Requirements Specification (Djavaherpour, 2024d), and the validation of its effectiveness in accurately classifying images within the scope of the CIFAR-10 dataset (Krizhevsky, 2009). This VnV process is crucial to affirm that the ANN system meets both its technical and user-centric goals.

2.1 Summary

The software being validated in this plan is an Artificial Neural Network designed for Image Classification, tailored specifically to work with the CIFAR-10 dataset (Krizhevsky, 2009). It allows users to upload images and efficiently classifies them into predefined categories. It operates within the constraints of available computational resources and is limited to handling images that fall under the CIFAR-10 dataset (Krizhevsky, 2009) categories, ensuring focused and optimized performance in its designated area.

2.2 Objectives

The primary objective of this VnV plan is to build confidence in the correctness and reliability of the Artificial Neural Network for Image Classification.

Our goal is to demonstrate that the system can classify images with a high degree of accuracy. We aim to significantly improve upon the less than 50% accuracy achieved in previous implementation (Djavaherpour, 2022), acknowledging that reaching 100% accuracy is not feasible due to inherent limitations in ANN models and the variability of image data. The focus will be on achieving the highest possible accuracy within these constraints. The system's accuracy will be measured through defined quantitative methods such as the cost function in SRS (Djavaherpour, 2024d).

2.3 Relevant Documentation

The ANN project is supported by several crucial documents. These include a Problem Statement (Djavaherpour, 2024c), which introduces the initial concept, and a Software Requirements Specification (Djavaherpour, 2024d) that outlines the necessary system requirements, accompanied by a Verification and Validation Report (Djavaherpour, 2024e) to ensure the system's compliance and efficacy.

3 Plan

In this section, the VnV plan of ANN is described. It begins with an introduction to the verification and validation team (subsection 3.1) and introduces the members and thei rules. Then is followed by the SRS verification plan (subsection 3.2), design verification plan (subsection 3.3), the VnV verification plan (subsection 3.4), implementation verification plan (subsection 3.5), automated testing and verification tools (subsection 3.6), and Software validation plan (subsection 3.7).

3.1 Verification and Validation Team

The VnV team members and their roles are shown in Table 1.

3.2 SRS Verification Plan

The verification process for the Artificial Neural Network's Software Requirements Specification document will be conducted as follows:

Name	Document	Role	Description
Dr. Spencer Smith	All	Instructor/ Reviewer	Review the documents, design and documentation style.
Tanya Djavaherpour	All	Author	Create and manage all the documents, create the VnV plan, perform the VnV testing, verify the implementation.
Fatemeh Norouziani	All	Domain Expert Reviewer	Review all the documents.
Atiyeh Sayadi	SRS	Secondary Reviewer	Review the SRS document.
Yi-Leng Chen	VnV Plan	Secondary Reviewer	Review the VnV plan.
Cynthia Liu	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

Table 1: Verification and validation team

1. An initial review will be carried out by designated team members, which include Dr. Spencer Smith, Fatemeh Norouziani, Atiyeh Sayadi, and Tanya Djavaherpour. This review will utilize a manual method, guided by an SRS Checklist (Smith, 2022c) developed by Dr. Smith.

2. Reviewers can give feedback and revision suggestions to the author by creating issues on GitHub.

3. It is the responsibility of the author, Tanya Djavaherpour, to respond to and resolve these issues, incorporating feedback from both primary and secondary reviewers, as well as addressing any recommendations provided by the instructor, Dr. Spencer Smith.

3.3 Design Verification Plan

The verification of the design documentation, including the Module Guide (MG) and Module Interface Specification (MIS), will be conducted via a static analysis approach, namely document inspection. As shown in Table 1

this process will be led by the domain/primary expert, Fatemeh Norouziani, and supported by the secondary reviewer, Cynthia Liu. Additionally, Dr. Spencer Smith, the class instructor, will also conduct a review of these documents. Reviewers are encouraged to provide their feedback directly to the author by creating issues in the project's GitHub repository. It is the responsibility of the author to address and resolve these issues, taking into account all the suggestions made. The review process will be facilitated by utilizing the MG (Smith, 2022a) and MIS (Smith, 2022b) Checklists, which have been formulated by Dr. Spencer Smith.

3.4 Verification and Validation Plan Verification Plan

Following the structure outlined in Table 1, the development and preliminary verification of the Verification and Validation plan will be undertaken by Author, Tanya Djavaherpour. Subsequent to this phase, domain expert, Fatemeh Norouziani, along with Yi-Leng Chen as a secondary reviewer, will review it. They give feedback and suggestions via GitHub issues. Once done, Instructor will do final review of the VnV plan. The whole review process will be aligned with the VnV Checklist (Smith, 2022d) that Dr. Smith has prepared. One specefic template has been defined for this purpose on the GitHub repository. It is the author's responsibility to check the submitted issues regularly and make necessary modifications.

3.5 Implementation Verification Plan

The implementation of the software will be verified using several techniques involving manual and automated techniques as outlined below:

- **Static Verification Techniques:** Code Walkthroughs will be the primary technique used for static verification. These sessions involve the development team, consisting of the author, Tanya Djavaherpour, and the domain expert, Fatemeh Norouziani, reviewing the code together. Initially, th author, Djavaherpour, thoroughly reviews the code independently, pinpointing possible issues. The final version of the code will be shared with the domain expert prior to the meeting, during which important test cases will be manually examined. In our static verification approach, we use code walkthroughs, guided by domain expert Fatemeh Norouziani. In the walkthrough meetings, we collaboratively discuss these points, focusing on logical inconsistencies and adherence to standards. The discussions and resolutions

are then documented systematically. Following this, the author addresses the highlighted issues, ensuring all modifications are well-documented. If substantial issues are detected, we schedule an additional walkthrough to verify the resolutions. This methodical process is crucial for ensuring the software's adherence to the project's quality standards. Also, this process allows the domain expert to present findings and raise questions, facilitating a comprehensive review.

• **Dynamic Testing:** Evaluation of the code will include both unit and system testing, focusing on the functional and nonfunctional requirements outlined in the SRS (Djavaherpour, 2024d) document. The tools used for these evaluations will be detailed in 3.6. Furthermore, the test cases used in system and unit testing will be stated in sections 4 and 5 (will be added), respectively. This approach ensures a thorough validation of the software across various levels of testing.

3.6 Automated Testing and Verification Tools

In this image classification project, Pylint (Python Code Quality Authority, 2023) and a regular testing process with a dedicated testing dataset of CIFAR-10 (Krizhevsky, 2009) are employed for automated testing and verification. Pylint (Python Code Quality Authority, 2023) is thorough in identifying coding issues, including syntax errors, logical bugs, and signs of problematic coding patterns, like complexity or redundancy. It upholds the PEP 8 style guide, promoting best practices in code formatting and naming. Pylint (Python Code Quality Authority, 2023) also supports type checking for annotated projects, identifies unused code, and flags potential string format errors. Security-wise, it alerts on usage of risky functions such as "exec" or "eval" and checks for documentation quality. Its flexibility in configuration allows customization according to project needs, playing a key role in sustaining the quality of our Python code. The combination of Pylint (Python Code Quality Authority, 2023) for code quality analysis and performance validation through targeted testing provides a comprehensive approach to maintaining robustness and effectiveness in the image classification system.

3.7 Software Validation Plan

Considering the extensive amount of experimental data required for a thorough software validation, this process falls outside the practical scope of our

ANN project. The constraints of time make it infeasible to collect and analyze the necessary data to validate system behavior comprehensively. However, we will ensure that the system undergoes rigorous verification to maintain high standards of quality and performance within these constraints.

4 System Test Description

In this section, we outline the test strategies for validating the ANN system's functional and nonfunctional requirements. The tests are designed to ensure compliance with the specifications detailed in the SRS, encompassing all aspects of system performance and quality.

4.1 Tests for Functional Requirements

The functional requirements are described in the SRS (Djavaherpour, 2024d) section 5.1. Implemented ANN will detect invalid inputs and, although the accuracy will not be 100%, classes of uploaded image will be determined. There are five functional requirements for this system. Testing R1 and R2 will be explained in the input verification section. R3 will be explained in the output verification test.

4.1.1 Input Verification

The inputs will be tested to satisfy R1 and R2 from ANN SRS (Djavaherpour, 2024d). Specifically, this test will ensure values of the inputs align with the input constraints. Table 2 displays the inputs and outputs of test cases for the input constraints tests.

Input Verification Test

1. T1: Valid Inputs

   Control: Automated Test

   Initial State: Pending Input

	Input (an imgae)		Output	
ID	*type*	*size*	valid?	Error Message
TC-ANN-1	PNG	mxn	Y	NONE
TC-ANN-2	JPG	mxn	Y	NONE
TC-ANN-3	GIF	mxn	N	Invalid Input Type
TC-ANN-4	PNG	(m+1)x(n)	N	Invalid Input Size
TC-ANN-5	JPG	(m+1)x(n)	N	Invalid Input Size
TC-ANN-6	GIF	(m+1)x(n)	N	Invalid Input Type
TC-ANN-7			N	Empty

Table 2: TC-ANN- Test Cases for Input Verification

Input: Set of input values for area of particular object given in the Table 2.

Output: Either give an appropriate error message for TC-ANN-1-3 to TC-ANN-1-7, or the class of the image object identified by the ANN as an output defined in the Table 2. It must be mentioned that if the type is not valid, the exception will be raised and the size will not be checked (like TC-ANN-6).

Test Case Derivation: Justified by the ANN's training to accurately classify valid input images according to the SRS (Djavaherpour, 2024d) specifications.

How test will be performed: An automated script will input valid and invalid images to the ANN. This automated test works with Pytest (Pytest Testing Framework, 2023).

4.1.2 Output Verification Test

To satisfy R3 from the SRS (Djavaherpour, 2024d), any input image should be properly classified and related to the correct output label from the set of 10 classes of CIFAR-10 (Krizhevsky, 2009).

1. T2: Classifier Test

Control: Automated Test

Initial State: Loading Trained ANN

Input: One image with one object

Output: Class of the image object

Test Case Derivation: The test is designed to evaluate the ANN's precision in classifying an individual image, reflecting its real-world application for singular image analysis.

How test will be performed: This test will be conducted using an automated script that inputs a single, randomly selected test image with one object into the trained ANN. The output will be the classified label provided by the ANN, which will then be compared to the actual label of the image to verify the accuracy of the classification. This test will be run 10 times and in each time the random image will be chosen from one folders. That said, after this pocess the software's perforamnec will be tested for all of the 10 classes of CIFAR-10 dataset (Krizhevsky, 2009), including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The purpose of this test is to make sure that system can print all of the classes' name properly.

4.2 Tests for Nonfunctional Requirements

NonFunctional requirements for ANN are given in SRS (Djavaherpour, 2024d) section 5.2.

4.2.1 Nonfunctional: Accuracy

Accuracy

1. T3: Accuracy Test

   Control: Automated Test

   Initial State: Loading Trained ANN

   Input: Set of test images form the CIFAR-10 dataset (Krizhevsky, 2009)

   Output: System accuracy based on prediction of class of the image object

How test will be performed: The test will be executed using an automated script that feeds a batch of test images from the CIFAR-10 dataset (Krizhevsky, 2009) into the trained ANN. The script will then compare the ANN's predicted class for each image against the known label, calculating the overall accuracy of the system. This process will quantify the model's performance in terms of correct classifications, providing a clear measure of its effectiveness in fulfilling the specified requirements. During this test, we just have the accuracy percentage to find how good is the trained network's performance. In fact, accuracy is the percentage of correct predictions.

4.2.2 Nonfunctional: Usability

Usability

1. T4: Usability Test

   Control: Manual with group of Computer Science students at McMaster

   Initial State: None

   Input: None

   Output: None

   How test will be performed: A diverse group of users will be asked to install and interact with the software, performing specified tasks. Following this, they will complete a survey focusing on aspects of usability such as ease of use, intuitiveness, and overall satisfaction. The survey results will be analyzed to identify areas for improvement in the software's usability. The questions are given in Table 3.

4.2.3 Nonfunctional: Maintainability

Maintainability

1. T5: Code Walkthrough Maintainability Test

   Control: Code Walkthrough

No.	Question	Answer
1.	Which operating system are you using?	
2.	Was system running smoothly on your computer? (Y or N)	
3.	How would you rate the response time of the software for various operations? (On a scale of 1-10)	
4.	How does the software handle errors or unexpected user actions?	
5.	During using the software, were received errors clear to recover them, if you received any?	
6.	Were you able to find all the functionalities easily?	
7.	Was there sufficient help or documentation available?	
8.	What, if anything, surprised you about the experience?	
9.	What specific feature or aspect of the software did you find least effective or most challenging?	
10.	What improvements or additional features would you suggest for enhancing the software's functionality or user experience?	
11.	How likely are you to recommend this software to others? (On a scale of 1-10)	

Table 3: Usability Test Survey

Initial State: None

Input: None

Output: Walkthrough Meeting aimed at identifying areas for improvement in system maintainability and documentation quality.

How test will be performed: In the code walkthrough meeting, team members (referenced in Table 1) will review the software's code and documentation, focusing on readability, modularity, and documentation clarity. Key findings and action points for enhancing maintainability will be documented for future reference and implementation.

2. T6: Automatic Maintainability Test

Control: Automatic

Initial State: None

Input: None

Output: Improve code quality with Pylint (Python Code Quality Authority, 2023).

How test will be performed: Pylint (Python Code Quality Authority, 2023) library can check the quality of code against different coding styles such as unused libraries and undefined variable. Which leads to easier maintenance.

4.2.4 Nonfunctional: Portability

Portability

1. T7: Portability Test

Control: Manual

Initial State: None

Input: None

Output: Successful running system over all platforms with different operating environments, including Windows and mcOS.

How test will be performed: The Author (Tanya Djavaherpour) will try to install and run whole software on different operating systems. Also, need to ensure regression testing that means all the given test cases pass on all different operating system.

4.3 Traceability Between Test Cases and Requirements

The traceability between test cases and requirements is indicated in Table 4

5 Unit Test Description

The modules of the ANN software are available in MG (Djavaherpour, 2024a) and MIS (Djavaherpour, 2024b) documents.

	R1	R2	R3	NFR1	NFR2	NFR3	NFR4	NFR5	NFR6	NRF7
4.1.1	X	X								
4.1.2			X							
4.2.1				X					X	
4.2.2					X			X		X
4.2.3						X				
4.2.4							X			

Table 4: Tracebility between test cases and requirements

5.1 Unit Testing Scope

Unit testing is performed on modules with highest priority, including **ANN Control Module** and **Data Preparing and Preprocessing Module**. The reason of defining Data Preparing and Preprocessing Module as a module with high priority is the impact and the role of data processing in every machine learining project. Additionally, Control Module includes the main function of running the software, which makes it an important module. Control Module uses Output Module and Saved ANN Module directly, and all other modules indirectly.

5.2 Tests for Functional Requirements

R1 and R3 are tested with T1 and T2 respectively in 4.1 using Pytets (Pytest Testing Framework, 2023). Functional tests not automated by PyTest (Pytest Testing Framework, 2023) are described in this section.

5.2.1 ANN Control Module (M2)

ANN Control Module is the main executable script for training a model or classifying an image.

1. train-test-id1

   Type: Automatic

   Initial State: N/A

Input: 1 to train the model

Output: Pass if the model is trained correctly, and Fail if it is not.

Test Case Derivation: The expected value in the Output field is justified by the following reasoning: test_train_model:

- The main function should instantiate a Model object and call its save_model method when the user selects option '1' to train the model.
- Therefore, if the main function executes successfully, it indicates that the model is trained or loaded correctly, and the save_model method is called, resulting in a Pass.

How test will be performed: Run the main function with input '1' to simulate training the model and verify that the model is instantiated and the save_model method is called.

2. classify-test-id2

   Type: Automatic

   Initial State: N/A

   Input: 2 to classify the image.

   Output: Pass if the model classifies input correctly, and Fail if it is not.

   Test Case Derivation:

   - The main function should instantiate an Output object, call its set_class_name method, and then call its save_feedback method when the user selects option '2' to classify an image and provide feedback.
   - Therefore, if the main function executes successfully, it indicates that the image is classified correctly, and the set_class_name and save_feedback methods are called, resulting in a Pass.

   How test will be performed: Run the main function with input '2' to simulate classifying an image and providing feedback, and verify that the Output object is instantiated, its methods set_class_name and save_feedback are called.

5.2.2 Data Preparing and Preprocessing Module(M9)

Data Preparing and Preprocessing Module is for loading and processing data.

1. data-prep-test-id3

   Type: Automatic

   Initial State: N/A

   Input: Random RGB images and labels for training and testing.

   Output: Pass if the dataset is processed correctly and the preprocessing steps are applied as expected, Fail otherwise.

   Test Case Derivation: The expected value in the Output field is justified by the following reasoning:

   - The test_get_dataset test case verifies the behavior of the get_dataset method of the Data class, which processes the dataset by applying various preprocessing steps.
   - The method is patched to mock the preprocessing steps, such as RGB to grayscale conversion, normalization, flattening, and shuffling.
   - After calling the method, the test checks if the images in the dataset are flattened correctly and if the preprocessing steps are called as expected.
   - Therefore, if the method behaves as expected and all assertions pass, the test case is considered to Pass.

   How test will be performed:

   - The setUp method sets up the test environment by creating an instance of the Data class and generating random RGB images and labels for training and testing.
   - The method under test, get_dataset, is called with the mocked preprocessing steps.
   - The assertions are made to verify that the images in the dataset are flattened correctly, and the preprocessing steps are called as expected.

- The test is considered a Pass if all assertions pass, indicating that the dataset is processed correctly, and a Fail otherwise.

5.3 Tests for Nonfunctional Requirements

Unit testing the non-functional requirements is beyond the scope.

5.4 Traceability Between Test Cases and Modules

A traceability between test cases and modules is shown in Table 5.

	M2	M3	M4	M5	M6	M7	M8	M9	M10
T1					X				
T2		X	X	X					
train-test-id1	X	X	X				X	X	
classify-test-id2	X	X		X		X			X
data-prep-test-id3						X	X		

Table 5: Tracebility between test cases and module

References

Tanya Djavaherpour. Cifar-10 classification using ANN. https://github.com/tanya-jp/CIFAR-Classification/blob/main/CIFAR_ANN.ipynb, 2022.

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

Tanya Djavaherpour. Module interface specification. https://github.com/tanya-jp/ANN-CAS741/blob/main/docs/Design/SoftDetailedDes/MIS.pdf, 2024b.

Tanya Djavaherpour. Problem statement and goals. https://github.com/tanya-jp/ANN-CAS741/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf, 2024c.

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

Tanya Djavaherpour. System verification and validation report. https://github.com/tanya-jp/ANN-CAS741/blob/main/docs/VnVReport/VnVReport.pdf, 2024e.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report, 2009.

Pytest Testing Framework. Pytest. https://docs.pytest.org/en/8.0.x/, March 2023. Python testing framework.

Python Code Quality Authority. Pylint. https://pypi.org/project/pylint/, December 2023. Python static code analysis tool.

W. Spencer Smith. MG checklist. https://github.com/smiths/capTemplate/commits/9251702fdcb9800c59f6ed3d11d91e2bd62fca6d/docs/Checklists/MG-Checklist.pdf, 2022a.

W. Spencer Smith. MIS checklist. https://github.com/smiths/capTemplate/commits/9251702fdcb9800c59f6ed3d11d91e2bd62fca6d/docs/Checklists/MIS-Checklist.pdf, 2022b.

W. Spencer Smith. SRS checklist. https://github.com/smiths/capTemplate/commits/9251702fdcb9800c59f6ed3d11d91e2bd62fca6d/docs/Checklists/SRS-Checklist.pdf, 2022c.

W. Spencer Smith. VnV checklist. https://github.com/smiths/capTemplate/blob/9251702fdcb9800c59f6ed3d11d91e2bd62fca6d/docs/Checklists/VnV-Checklist.pdf, 2022d.