Project Title: System Verification and Validation Plan for ANN

Tanya Djavaherpour

February 19, 2024

Revision History

Date	Version	Notes
Feb. 15, 2024	1.0	Initial Draft

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test
ANN	Artificial Neural Network
IM	Instance Model
SRS	Software Requirements Specification
VnV	Verification and Validation

For complete symbols used within the system, please refer the section 1 in SRS document.

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

2 General Information

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, 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.

2.3 Relevant Documentation

The ANN project is supported by several crucial documents. These include a Problem Statement, which introduces the initial concept, and a Software Requirements Specification that outlines the necessary system requirements, accompanied by a Verification and Validation Report 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:

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 developed by Dr. Smith.

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

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

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 and MIS 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 that Dr. Smith has prepared. 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. 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. 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 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, 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 key for enhancing code quality by identifying coding errors, enforcing standards, and encouraging best practices. Alongside, a robust testing process, involving thorough evaluation on a separate testing dataset, ensures the accuracy and reliability of the image classification model. 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

Validation is the process of comparing the outputs of models to experimental values. Since the CIFAR-10 dataset Krizhevsky (2009) does not come with a predefined validation set and there is not enough experimental data, the validation of the implemented ANN cannot be measured in a traditional sense. Instead, as mentioned previously in 3.6, the network will be tested to find the accuracy using testing data.

It's essential to understand that this approach, while deviating from conventional validation methods due to dataset constraints, still provides a valid assessment of the model's performance. This testing process offers a reliable evaluation of the model's ability to generalize and perform accurately on unseen data.

4 System Test Description

4.1 Tests for Functional Requirements

The functional requirements are described in the SRS 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, R4 and R5 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. 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

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

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

1. T1: Valid Inputes

   Control: Automated Test

   Initial State: Pending Input

   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.

   Test Case Derivation: Justified by the ANN's training to accurately classify valid input images according to the SRS specifications.

How test will be performed: An automated script will input valid and invalid images to the ANN, comparing the output classifications with expected results to verify accuracy.

4.1.2 Output Verification Test

To satisfy R3, R4 and R5 from the SRS, 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. T1: Accuracy 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.

4.2 Tests for Nonfunctional Requirements

NonFunctional requirements for ANN are given in SRS section 5.2.

4.2.1 Nonfunctional: Accuracy

Accuracy

1. T1: 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.

4.2.2 Nonfunctional: Usability

Usability

1. T1: Usability Test

Control: Manual with group of people

Initial State: None

Input: None

Output: A survey to know about the user perspective towards the system

How test will be performed: The user group will be asked to install the software on their system and give input on their own. Then user need to fill out the scale of the question given in the survey 0000000000000000

4.2.3 Nonfunctional: Maintainability

Maintainability

1. T1: Code Walkthrough Maintainability Test

Control: Code Walkthrough

Initial State: None

Input: None

Output: Walkthrough Meeting helps to improve the maintainability of the system and all documentations.

How test will be performed: During code walkthrough meeting all the details related to the software lifespan, coupling of the software architecture, documentation of the software will discussed among team members listed in 1.

2. T2: Automatic Maintainability Test

Control: Automatic

Initial State: None

Input: None

Output: Using Pylint Python Code Quality Authority (2023) helps to improve the code quality

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

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access

program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

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

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

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?