System Requirements Specification

EasyNN/EasyAI

CS 491, Fall 2021

Team Name: EasyNN

Team Members:

- Jack Nguyen
- Dan Wilczak
- Liam Kehoe
- Nathan Foster

Contents of this Document

Introduction

- System to be Produced
- Applicable Standards

Definition, Acronyms, and Abbreviations

Product Overview

- Assumptions
- Stakeholders
- Event Table
- Use Case Diagram
- Use Case Descriptions

Specific Requirements

- Functional Requirements
- Interface Requirements
- Physical Environment Requirements
- Users and Human Factors Requirements
- Documentation Requirements
- Data Requirements
- Resource Requirements
- Security Requirements
- Quality Assurance Requirements

Supporting Material

Section 1: Introduction

System to be Produced:

- The goal of this product is to create a neural network framework that is easy to use and customize. This will be done in Python and should rely on defaults and reduced features compared to frameworks like TensorFlow to provide an a framework which is much more user friendly.

Applicable Standards

- <You do not have to repeat the standards included in the project plan. Instead, cite any standards that are specific to the system requirements.>

Definitions, Acronyms, and Abbreviations

- A model is a representation of another system.
- A (Artificial) Neural Network (NN or ANN) is a simplified computational model of the human brain, loosely based on the idea of passing information between several neurons to get an output.
- A Neuron is a component in a Neural Network that collects, stores, and sends values.
- A loss function is a numerical measurement of the performance of a model.
- Optimization is the process of either minimizing or maximizing a loss function.Forward Propagation is the process of processing values through a Neural Network, starting from the input nodes and moving to the output nodes.
- Back Propagation is the process of computing components of the gradient iteratively instead of entirely at once, starting from the output nodes and moving to the input nodes.
- A tensor is a multidimensional array with a shape and indexing by references/pointers. Examples of tensors are floats (except for the indexing), vectors, and matrices. These will be denoted as ArrayND e.g. Array1D is a vector and Array2D is a matrix.

Section 2: Product Overview

Assumptions:

- <List all the assumptions the developers are making. For example assumptions about other systems this product will interface with; assumptions about the technological environment in which the product will operate (how much memory, what type of processor, ...); assumptions about availability and capability of COTS, GOTS, or other re-used products, ...>

- The user shall have an upgraded version of Python in order to run EasyNN (currently 3.9.7)
- The user should have a basic understanding of how to program in python to a simple level to be able to:
    - Define variables
    - Call functions
    - Run python scripts
- The user should have something to train the data on

Stakeholders:

- <A stakeholder is anyone who has an interest in the system to be developed. For example, the customer, the various classes of users, applicable regulatory agencies, ... List each category of stakeholder and give a phrase or a sentence to describe their interest or concerns>
- Customer: Professor Cheng. Person to which the product is being delivered.
- Product owner: Daniel Wilczak. Person who has the vision for the product and for whom the product originated.

Event Table:

Event Name	External Stimuli	External Responses	Internal data and state
Creation	Constructor is called	A neural network is created	by default it is set up for good training of [[Whatever the database is called]]
Forward Propagation	Data is given to NN to process	The confidence interval is given	The values for each layer are determined(and stored)
Backward Propagation	The correct answer is given to NN	loss and accuracy are given for the last input/batch	All weights and biases are updated to improve the accuracy of the model

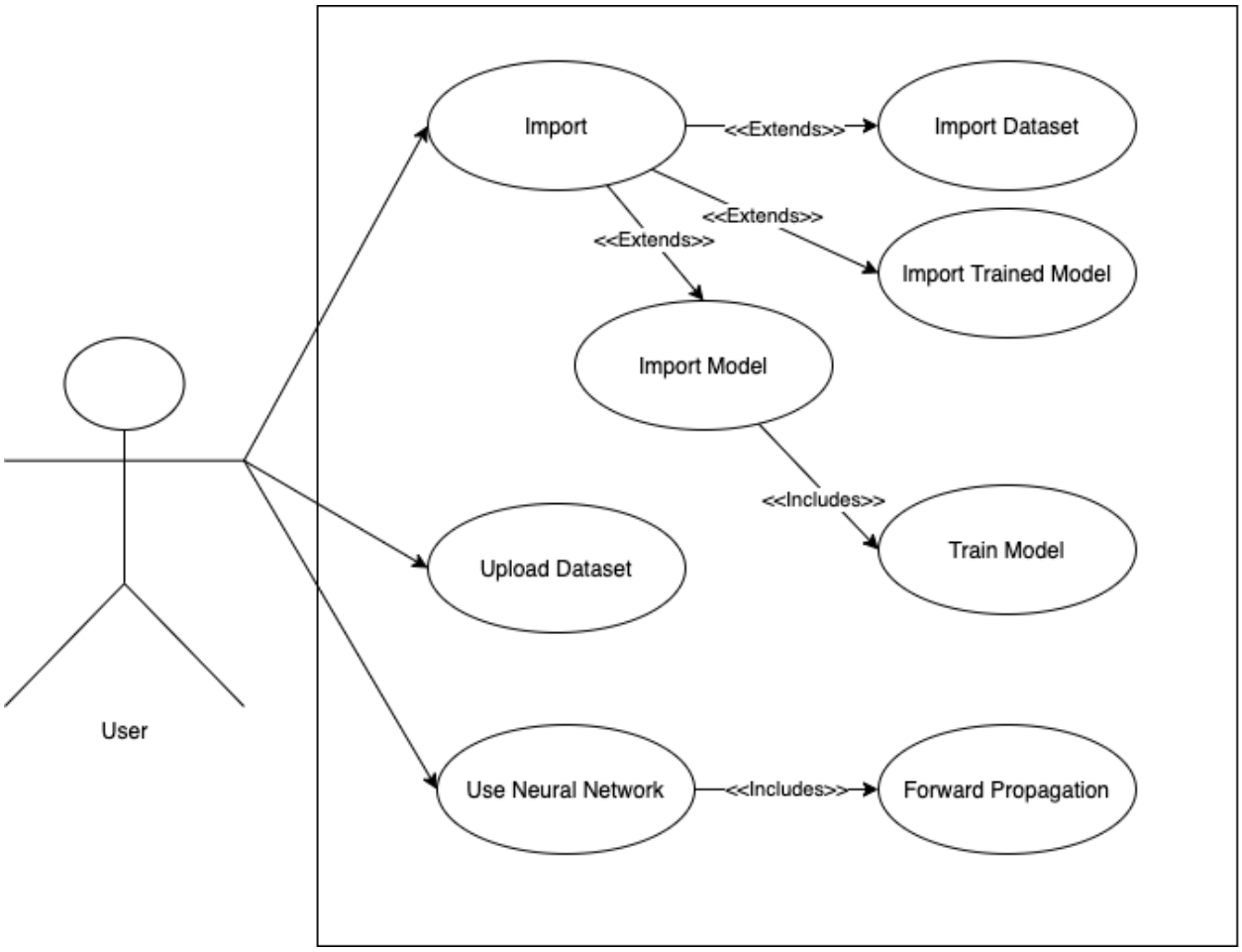

Figure 1: Use case diagram for the EasyNN package.

Use Case Descriptions:

1. Import
    a. The user imports a dataset(s) and/or a trained or untrained model(s).
2. Import Dataset
    a. The user imports a dataset(s).
3. Import Trained Model
    a. The user imports a trained model(s).
4. Import Model
    a. The user imports an untrained model(s).
5. Upload Dataset
    a. The user uploads their own dataset to be used.
6. Use Neural Network
    a. The user uses the Neural Network.
7. Forward Propagation
    a. The part of the Neural Network that determines what the output is given a set of inputs.

Section 3: Specific Requirements

<Use the following template for each requirement. >

No: <unique requirement number>
Statement: <the "shall" statement of the requirement>
Source: <source of the requirement>
Dependency: <list each other requirements on which satisfaction of this requirement depends. (Maybe "None")>
Conflicts: <list each other requirements with which this requirement conflicts. (Maybe "None")>
Supporting Materials: <list any supporting diagrams, lists, memos, etc.>
Evaluation Method: <How can you tell if the completed system satisfies this requirement? >
Revision History: <who, when, what>

3.1 Functional Requirements

- < Describe the fundamental actions that the system must perform. Functional requirements can be partitioned into subfunctions or subprocesses. Note: the System design partition does not have to correspond with the functional requirements partition. Functional requirements include:
    - validity checks on the inputs,
    - the exact sequence of operations,
    - responses to abnormal situations
    - relationship of outputs to inputs
        - input/output sequences, formulas for input to output conversion, etc.
    - ...>

3.1.1
Description: The Machine Learning system shall find the local minima of a function, provided a built-in optimization algorithm.
Evaluation: The Machine learning system should be tried on test cases from https://en.wikipedia.org/wiki/Test_functions_for_optimization.

3.1.2
Description: The Optimizer shall use given values and derivatives to train on.
Evaluation: The Machine Learning system should pass the values and derivatives into the optimizer.

3.1.3
Description: The EasyNN system shall save network models in a database when asked to.
Evaluation: The EasyNN system should save the topology, machine learning parameters, and activation functions to a database.

3.1.4
Description: The EasyNN system shall have all neural network models as listed in 4.1.
Evaluation: Implementations of each model should be included.

3.1.5
Description: The EasyNN system shall have all optimization algorithms as listed in 4.2.
Evaluation: Implementations should follow formulas found in the literature.

3.1.6
Description: The Neural Network shall use backpropagation (through time) to compute derivatives during training. Recurrent models require a modified form of back propagation.
Evaluation: Implementations of back propagation should follow formulas found in the literature.

3.2 Interface Requirements

- < Describe the interactions of the system with other entities. Interface requirements include a precise description of the protocol for each interface:
    - what data items are input
    - what data items are output
    - what are the data type, the format, and the possible range of values for each data item? (i.e. what is the "domain" of this data item?)
    - how accurate must each data item be?
    - how often will each data item be received or sent?
    - timing issues (synchronous/asynchronous)>
    - how many will be received or sent in a particular period?
    - how accurate must the data be?
    - ...>

3.2.1
Description: The Neural Network shall only accept ArrayND of floats as input.
Evaluation: Check if the Neural Network casts inputs to float tensors to ensure correct types and shapes.

3.2.2
Description: The Neural Network shall return a float tensor as output.
Evaluation: Check software for operations used and if they result in tensors.

3.2.3
Description: The non-Convolutional Neural Networks shall return a float vector as output.
Evaluation: Check software for operations used and if they result in vectors.

3.2.4
Description: The Convolutional Neural Networks shall return a 3D tensor as output.
Evaluation: Check software for operations used and if they result in 3D tensors.

3.2.5
Description: The Optimizer shall set their hyperparameters through initialization.
Evaluation: The Optimizer initializers should accept hyperparameters as parameters.

3.2.6
Description: The Optimizer shall take the current iteration, values, and derivatives as input for updating the values.
Evaluation: Check software for Optimizer and Machine Learning usage.

3.2.7
Description: The Neural Network shall accept a topology for initialization based on the type of neural network.
Evaluation: Check Neural Network software and documentation on how to define its topology.

3.3 Physical Environment Requirements

- < Describe the environment in which the system must operate. Physical environment requirements include:
    - type of equipment/environment on which the system must run
    - location of the equipment
    - environmental considerations: temperature, humidity, ...
    - ...>

- Someone fill in the physical server information.

3.4 User and Human Factors Requirements

- <Describe the users and their constraints:
    - What different types of users must the system support?
    - What is the skill level of each type of user? What type of training and documentation must be provided for each user?
    - Do any users require special accommodations (large font size, ...)
    - Must the system detect and prevent misuse? If so, what types of potential misuse must the system detect and prevent?
    - ...>

- The EasyNN system shall save the model's data to the database.

3.5 Documentation Requirements

- <Describe what documentation is required:
    - on-line, printed, or both?
    - what is the assumed skill level of the audience of each component of documentation?
    - ...>

- The code shall follow PEP8's formatting standards.

- Classes, attributes, methods, and functions shall be documented using doc-strings and type-hints.

- The systems shall use duck typing instead of enforcing type hints.

- The built-in help() function shall print documentation for classes and functions.

- The stdlib typing.get_type_hints() function shall reveal the attributes of classes and signatures of functions.

- The GitHub repository shall include documentation in the wiki.

- In-line comments shall be used where determined necessary during code review.

- Source code documentation shall assume the reader is proficient with python.

- GitHub documentation shall assume the reader is proficient with python.

- Programmers not on the EasyNN development team should be able to construct a model with the following topologies by following the documentation

3.6 Data Requirements

- <Describe any data calculations: what formula will be used? to what degree of precision must the calculations be made? >
- <Describe any retained data requirements: exactly what must be retained?
- ...>

3.6.1
Description: The Machine Learning system shall store the current iteration, and Optimizer, and a tensor for values and derivatives.

3.6.2
Description: The Neural Network shall store the Machine Learning system.

3.6.3
Description: The Neural Network shall store a collection of Neurons.

3.6.4
Description: The Neurons shall store weights, biases, and their derivatives as references/pointers to values in the Machine Learning system.

3.6.5
Description: The Neural Network shall use its respective feedforward and backpropagation formulas as found in the literature.

3.6.6
Description: The EasyNN system shall use NumPy's linear algebra operations to evaluate large matrix/vector calculations e.g. norm, dot product, and matrix multiplication.

3.7 Resource Requirements

- <Describe the system resources:
  - skilled personnel required to build, use, and maintain the system?
  - physical space, power, heating, air conditioning, ...?
  - schedule?
  - funding?
  - hardware/software/tools?
  - ...>

3.8 Security Requirements

- <Describe any security requirements:
  - must access to the system or information be controlled?
  - must one user's data be isolated from others?
  - how will user programs be isolated from other programs and the operating system?
  - how often will the system be backed up?
  - must the backup copies be stored at a different location?
  - should precautions be taken against fire, water damage, theft, ...?
  - what are the recovery requirements?
  - ...>

3.9 Quality Assurance Requirements

- <Describe quality attributes:
  - What are the requirements for reliability, availability, maintainability, security, portability ...?
  - How must these quality attributes be demonstrated?
  - Must the system detect and isolate faults? If so, what types of faults?
  - Is there a prescribed mean time between failures?
  - Is there a prescribed time the system must be available?
  - Is there a maximum time allowed for restarting the system after a failure?
  - What are the requirements for resource usage and response times?
  - ...>

Section 4: Supporting Material

4.1. List of desired models to be supported:

4.1.1. Non-dense feed-forward
4.1.2. Dense feed-forward
4.1.3. Non-dense recurrent
4.1.4. Dense recurrent
4.1.5. Markov chain
4.1.6. Convolutional
4.1.7. Auto-encoder

4.2. List of optimization algorithms to be supported:

4.2.1. Gradient descent
4.2.2. Momentum-based gradient descent
4.2.3. Stochastic gradient descent
4.2.4. Perturbed gradient descent
4.2.5. Adagrad
4.2.6. Adadelta
4.2.7. RMSprop