System Requirements Specification

EasyNN

CS 491, Fall 2021

Jack Nguyen

Dan Wilczak

Liam Kehoe

Nathan Foster

Version/Author	Date
1.0 / Wilczak/Nguyen/Kehoe/Foster	9/28/2021
2.0 / Wilczak/Nguyen/Kehoe/Foster	10/26/2021
3.0 / Wilczak/Nguyen/Kehoe/Foster	11/30/2021

Table Of Contents

Table Of Contents **1**

Section 1: Introduction **2**

1.1 System to be Produced 2

1.2 Definitions, Acronyms, and Abbreviations 2

Section 2: Product Overview **2**

2.1 Assumptions 2

2.2 Stakeholders 3

2.3 Event Table 3

2.4 Use Case Descriptions 4

Section 3: Specific Requirements **5**

3.1 Functional Requirements 5

3.2 Interface Requirements 5

3.3 Physical Environment Requirements 6

3.4 User and Human Factors Requirements 6

3.5 Documentation Requirements 6

3.6 Data Requirements 7

3.7 Resource Requirements 7

Section 4: Supporting Material **8**

4.1 List of desired models to be supported 8

4.2 List of optimization algorithms to be supported 8

Section 1: Introduction

1.1 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 a framework which is much more user friendly.

1.2 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

2.1 Assumptions

2.1.1 The user shall have an upgraded version of Python in order to run EasyNN (currently 3.9.7)

2.1.2 The user shall 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

2.1.3 The user shall have datasets to train the data on

2.2 Stakeholders

2.2.1 Customer: Professor Cheng. Person to which the product is being delivered.

2.2.2 Product owner: Daniel Wilczak. Person who has the vision for the product and for whom the product originated.

2.3 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 the database
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

Table 1: Event Table

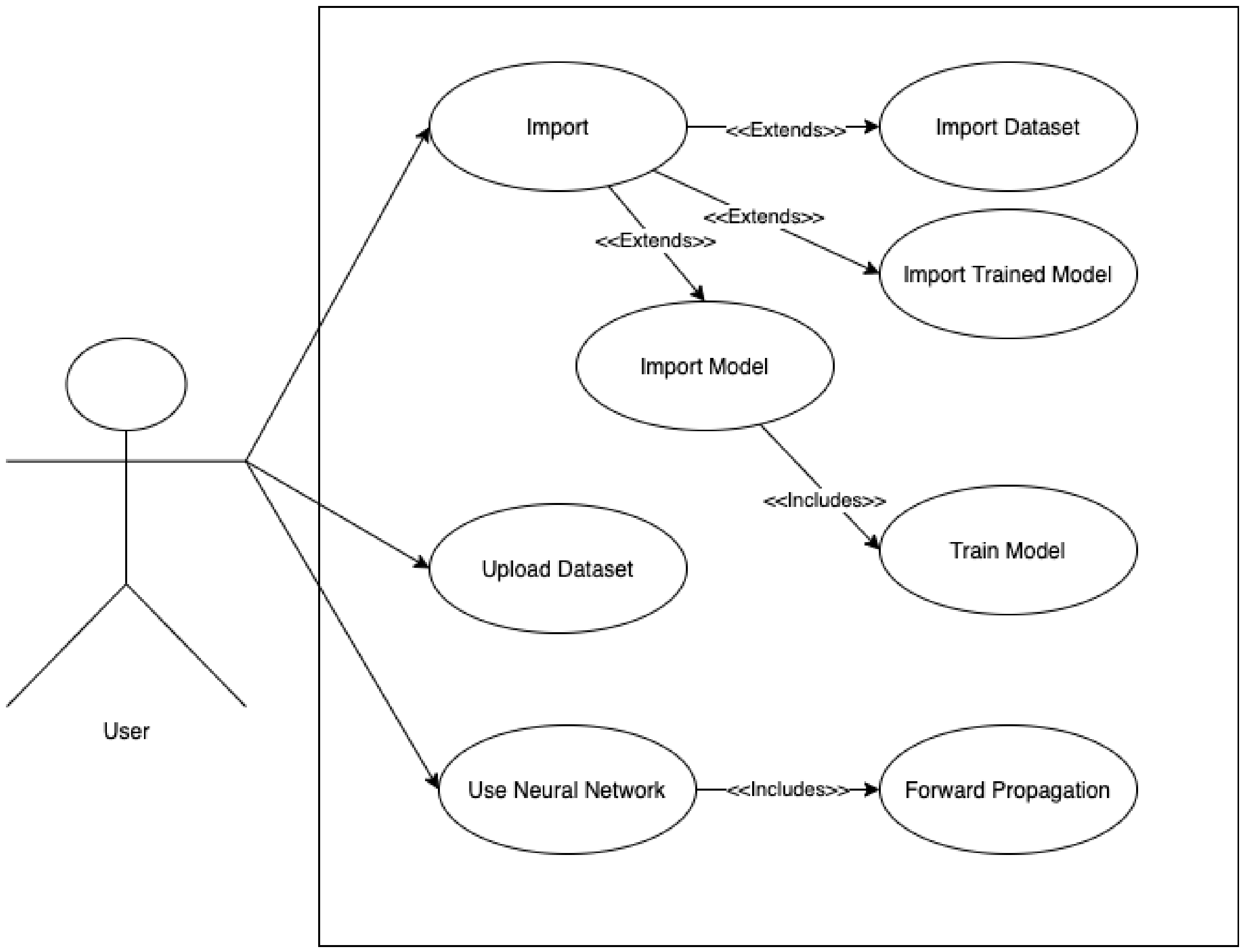

Figure 1: Use case diagram for the EasyNN package.

2.4 Use Case Descriptions

2.4.1 Import
a. The user imports a dataset(s) and/or a trained or untrained model(s).
2.4.2 Import Dataset
b. The user imports a dataset(s).
2.4.3 Import Trained Model
c. The user imports a trained model(s).
2.4.4 Import Model
d. The user imports an untrained model(s).
2.4.5 Upload Dataset
e. The user uploads their own dataset to be used.
2.4.6 Use Neural Network
f. The user uses the Neural Network.
2.4.7 Forward Propagation
g. The part of the Neural Network that determines what the output is given a set of inputs.

Section 3: Specific Requirements

3.1 Functional Requirements

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 shall 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

3.2.1
Description: The Neural Network shall only accept ArrayND of floats as input.
Evaluation: Check if Neural Network casts inputs to float tensors and if they result in 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

No physical environment requirements other than the user's physical PC.

3.4 User and Human Factors Requirements

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

3.5 Documentation Requirements

3.5.1 The code shall follow PEP8's formatting standards.

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

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

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

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

3.5.6 The GitHub repository shall include documentation in the wiki.

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

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

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

3.5.10 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

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

3.6.2 The Neural Network shall store the Machine Learning system.

3.6.3 The Neural Network shall store a collection of Neurons.

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

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

3.6.6 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

No resource requirements as no skilled personnel, special environment, nor money is needed as the system is easy to use, on the user's personal PC, and is free.

Section 4: Supporting Material

4.1 List of models 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 supported

4.1.8. Gradient descent
4.1.9. Momentum-based gradient descent
4.1.10. Stochastic gradient descent
4.1.11. Perturbed gradient descent
4.1.12. Adagrad
4.1.13. Adadelta
4.1.14. RMSprop