N.N.N. V.0.1-T “Documentation”

Date Version Released: 01/04/2025
Last Documentation Update: 29/05/2025
Written by Mar13554
N.N.N. team: Mar13554, HamirinTK
Disclaimer: It is not necessary to read this to be able to use the program; it’s just extra, in case you need it.

Information

N.N.N. V.0.1-T is the successor to **V.0.1-t**. The method used is the same, but the key difference is the use of C++ for the model instead. Aiming for quicker training relative to all Python. The “save data” from V.0.1-t is compatible with this version; see the previous document for more information on the “save data” in the “Main_Model.txt” file.

Set up

Installation

1. Unzip either **N.N.N.V.0.1-T.zip** or **N.N.N.V.0.1-T(exe).zip**
2. Check if the following folders and files are present inside the inner folder:
   2.1 **N.N.N.V.0.1-T.zip**
   - Cached_data_files
   - Small_Modules
   - TestData
   - C_pipe.py
   - C++_N.N.N.exe
   - main.py

   2.2 **N.N.N.V.0.1-T(exe).zip**
   - _internal
   - Cached_data_files
   - TestData
   - C++_N.N.N.exe
   - main.exe
3. Insert compatible .json data files[1] into the TestData folder
4. Run the main.exe file. (Or don’t)

[1] You may get some from HamirinTK here

Technical Usage

C++ executable

The C++_N.N.N.exe takes in inputs of needed data and outputs the model in the form of "save data". You need to enter the number of nodes for each layer first; then you can choose the commands. E.g., "2 2 2 2", input layer included.

Here is a table with different "commands" for the program.

Command	**Purpose**	**Input**
p	Set up parameters for one node	layer, node, $n(w) + 1$, W, b[2]
d <int>	Get **the amount** of inputs and outputs (training data)	Amount of inputs and outputs (One value)
i	Get inputs	Iterate through each value in every input
o	Get outputs	Same as above.
t <int>	Train	Epochs (Integer)
b	Break (Exit program)	N/A

Note: "p" is supported when a command is executed manually.

Direct Command

The first question the program gives is
"Would you like to enter the full command to the .exe directly? (Not recommended) Y/N:"
Once it asks you to enter the full command the format should be the same as shown:
"2 2 2\nd 2\ni\n0 0\n1 1\no\n1 1\n0 0\nt 4\nb\n"

Explanation:
"2 2 2" are the amount of nodes for each layer (input layer included).
"d 2", as in, 2 inputs and outputs expected.
"i", set to expect inputs
"0 0 1 1" is split to "0 0" and "1 1" by the program (two inputs).
"o", set to expect outputs
"1 1 0 0" is split to "1 1" and "0 0" by the program (two outputs).
"t 4", as in, train it 4 times (epochs)
"b", once done, exit program.

[2] n(W)+1 as in the number of weights+1; W, b as in each weight and then the bias.

Possible Issues

Issue 1: main.exe crashes when run.

Check all the needed folders and files. Are they placed in the same folder with main.exe? If not, move them. Did you use a cached file that isn't compatible with the model? If so, enter "N" when asked if you want to use the cached file, and it will generate a new one. You can try "catching" the terminal with your cursor by clicking on the top of it for more info of the error.

Issue 2: Pop-up error.

Something went horribly wrong. I can't help you.

Issue 3: Running Forever.

It's completely expected if the number of nodes present is a lot. It's still using numerical differentiation for now, which explains why.

Issue 4: Can't find where the model is saved.

Look for the "Main_model.txt" file; it should be in the same folder as main.exe.

How it works

A not-so-comprehensive section for how the logic works.

1. The program checks for all the needed files after selecting anything that isn't "Y" for the first question.
2. It asks the user for information:
    - To load Main_model.txt (if present)
    - Which data file to use
    - To use the cached version of the data file (if it exists)
    - How many epochs (amount of times) would you like to run it for?
3. It packages this information into a command (see: Page 2)
    - If you want to load from Main_model.txt, then it must first write the model parameters into a "Transfer.bin" file, which will be generated if it doesn't already exist.
    - It will then send the command to C++_N.N.N.exe for it to use.
4. The C++ exe gets it and begins constructing the model locally and then trains it, finally returning model info in Python format:
   [Avg_MAE_loss, Model_info, Model_parameters]
5. Repeat.