Suan Sunandha Science and Technology Journal

WNN : File Format for Neural Network Interchange

Anirudh Giri
Department of Computer Science Engineering, SRMIST Kattankulathur
9/4B, Bharathwajar Street, East Tambaram, Chennai 600059, India
Corresponding author e-mail: anirudhges@gmail.com

Received: *16 June 2020* / **Revised:** *1 November 2020* / **Accepted:** *27 November 2020*

Abstract
A programming language agnostic, neural network library agnostic, standardized file format used to save an already trained neural network model would facilitate the process of sharing or releasing said neural network model online in a simplistic fashion. A standard file format for saving neural network models would include metadata about the neural network such as the number of inputs, hidden layers, nodes per hidden layer, outputs and the activation function used along with the weights and bias values. Such a file can be parsed to reconstruct a neural network model in any programming language or library and would remove a neural network model's dependency on the library it was created on.

Keywords: Artificial neural networks, File format, Neural network model sharing, Neural network model saving

1. Introduction
Due to the recent spike in interest resulting from the developments in the fields of Artificial Intelligence and Machine Learning, there has been no shortage of neural network libraries such as TensorFlow by Google, PyTorch by Facebook, Keras, DeepLearning4J, OpenNN, FANN, and much more (Erickson, Korfiatis, Akkus, Kline, & Philbrick, 2017).

The ability to save a trained neural network model on disk for reuse is provided by all major neural libraries but each library uses its own approach to save a neural network model and they are incompatible with each other. This hinders advancements in the machine-learning field as it imposes restrictions on how much a neural network model can be shared between users. This is analogous to designing a webpage that will only load as intended on one web browser and will fail to load on all other browsers.

1.1 Neural networks at their core
A neural network, at its core, is described as a universal function approximator (Hornik, Stinchcombe, & White, 1989; Poggio & Girosi, 1990). It can be represented as a collection of floating-point numbers, along with some metadata. A neural network is defined by its constituent weight and bias values, and anything else that has to do with a neural network (such as the ability to feed forward the input values and obtain an output) is up to the implementation.

Thus, a file that contains the weights and bias values, along with some important header information such as the number of inputs, number of outputs, number of hidden layers, number of nodes per hidden layer and the employed activation function (sigmoid, ReLU, tanh, etc.) can be used to reconstruct a neural network in the language or neural network library of the user's choice.

1.2 History and background information
The idea of artificial neural networks was first conceived by Warren McCulloch and Walter Pitts (McCulloch & Pitts, 1943) who created a computational model for artificial neural networks based on algorithms called threshold logic. Breakthroughs in the field of computer science that made use of artificial neural networks include the creation of the perceptron in 1958 by Frank Rosenblatt (Rosenblatt, 1958) and the formulation of the backpropagation algorithm in 1986 (Rumelhart, Hinton, & Williams, 1986).

1.3 Notations and terminology
Superscript – denotes the layer of the neural network

Subscript – denotes the specific node in a layer (starting with 0 at the top)

w – denotes the weight between two nodes

b – denotes the bias value of the node

2. The Proposed Format
WNN (**W**eights of **N**eural **N**etwork) is the proposed file format whose intended use is to save neural network models in a programming language and neural network library agnostic way.

Suan Sunandha Science and Technology Journal

A WNN file can be downloaded from a neural network repository or obtained through a CDN and can be parsed* to recreate a functioning neural network model without the need to train the neural network again.

The wnn file will consist of three parts –

- Header Information
- Weight Values
- Bias Values

Header Information

The file must begin by listing the following information in order, separated by a line break –

- Number of Inputs
- Number of Hidden Layers
- Number of Nodes in each Hidden Layer
- Number of Outputs
- Employed activation function for each layer

The number of nodes in each hidden layer should be listed in the same line, separated by spaces. If all hidden layers have the same number of nodes, listing that number once should suffice.

The employed activation function should be represented by a single-digit integer, ranging from 0 to 6, according to the given table –

Value	Activation Function
0	Sigmoid
1	Tanh
2	Arctan
3	Softmax
4	Softplus
5	ReLU
6	Leaky ReLU
7	ELU

In case the activation function being employed is Leaky ReLU($y = \max(\boldsymbol{\alpha x}, \mathbf{x})$) or the ELU$\left(y = \begin{cases} \boldsymbol{\alpha}(\mathbf{e^x - 1}), & x < 0 \\ x, & x \geq 0 \end{cases}\right)$, the $\boldsymbol{\alpha}$ value must be listed with the activation function index separated by a whitespace**.**

The Weight Values

Following the header information, separated by a line feed, will be the value of the weights of the connections between each node.

Let w_{ij}^{L} be a weight value of the connection between nodes of index i and j in the layers L and L+1 respectively (Clarkson, 1996).
Let m be the number of nodes in the preceding layer and n be the number of nodes in the succeeding layer.

The weights of the same layer should be listed as a group, separated by spaces, and each group of weights per layer should be separated by line feeds as follows –

$$w_{00}^{0}\ w_{01}^{0}\ w_{02}^{0}\ \dots\ w_{0(n-1)}^{0}\ w_{10}^{0}\ w_{11}^{0}\ w_{12}^{0}\ \dots\ w_{(m-1)(n-1)}^{0}$$
$$w_{00}^{1}\ w_{01}^{1}\ w_{02}^{1}\ \dots\ w_{0(n-1)}^{1}\ w_{10}^{1}\ w_{11}^{1}\ w_{12}^{1}\ \dots\ w_{(m-1)(n-1)}^{1}$$
$$\vdots$$
$$w_{00}^{L}\ w_{01}^{L}\ w_{02}^{L}\ \dots\ w_{0(n-1)}^{L}\ w_{10}^{L}\ w_{11}^{L}\ w_{12}^{L}\ \dots\ w_{(m-1)(n-1)}^{L}$$

The Bias Values

Let b_i^L be the bias of the i[th] node in the L[th] layer of the neural network. Let m be the number of nodes in the corresponding layer. The biases of each node in a layer should be listed in a line, separated by spaces, and each layer should be separated by a line feed as follows –

$$b_0^0\ b_1^0\ b_2^0\ \dots\ b_{(m-1)}^0$$
$$b_0^1\ b_1^1\ b_2^1\ \dots\ b_{(m-1)}^1$$
$$\vdots$$
$$b_0^L\ b_1^L\ b_2^L\ \dots\ b_{(m-1)}^L$$

**Sample WNN file and parser can be found at https://www.github.com/anirudhgiri/WNN-File-Parser*

Suan Sunandha Science and Technology Journal

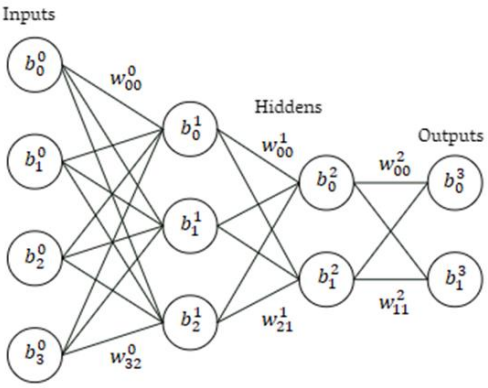

Figure 1. A sample neural network with the correct notations for reference.

3. Examples and Experimentation

Taking a neural network model with 2 inputs, 2 hidden layers with 2 nodes in the first hidden layer and 1 node in the second hidden layer and 2 outputs where all layers use the ReLU activation function (except the output layer, which uses the softmax function) and where all the weights and biases are set to 0, the resulting WNN file was generated as follows -

```
2
2
2 1
2
5 5 5 3
0 0 0 0
0 0
0 0
0 0
0 0
0
```

Another wnn file for a deep neural network model to simulate the XOR function was generated**.** The WNN file was as follows –

```
2
1
2
1
0
13.83 13.83 15.31 15.31
-11.52 11.52
-19.35 -6.78
-5.13
```

The model consisted of 2 inputs, 1 hidden layer, 2 nodes in the hidden layer and 1 output where all layers used the sigmoid activation function**.**

4. Results and Discussions

The parser was successfully able to retrieve enough information from both files to be able to rebuild the artificial/deep neural network model which their respective wnn file was describing**.**

```
Number of Inputs : 2
Number of Hidden Layers : 2
Number of Nodes in layer 1 : 2
Number of Nodes in layer 2 :
Number of Nodes in layer 3 : 1
Number of Outputs : 2
Activation Function Used In Layer 0 : ReLU
Activation Function Used In Layer 1 : ReLU
Activation Function Used In Layer 2 : ReLU
Activation Function Used In Layer 3 : Softmax
Weight values :
0 0 0 0
0 0
0 0
Bias Values:
0 0
0 0
0
```

Figure 2. The output of the parser program when it was served with the example wnn file listed in section 4.

```
Number of Inputs : 2
Number of Hidden Layers : 1
Number of Nodes Per Hidden Layer : 2
Number of Outputs : 1
Activation Function Used (For all layers) : Sigmoid
Weight values :
13.83 13.83 15.31 15.31
-11.52 11.52
Bias Values:
-19.35 -6.78
-5.13
```

Figure 3. The output of the parser program when it was served with the example WNN file listed in section 4.

5. Scope for the Future

A standard file format, if used by all major neural network libraries to save models and load them with their own parsers, would be greatly beneficial to the field of artificial intelligence and machine learning**.** It could give rise to a centralised repository for neural network models hosted on the cloud where machine learning engineers can

Suan Sunandha Science and Technology Journal

collaborate to create the most efficient models for all sorts of use cases**.**

Neural network models with high prediction accuracies could be searched for online through a catalogue instead of training them from scratch every time they are needed**.** It would be similar to using open source libraries from GitHub instead of writing the code from scratch every time certain functionalities need to be added**.**

Software similar to package managers like npm and pip could be created to access and download neural network models through the command line on demand**.**

6. Conclusion

This paper has described the need for a standardized file format for neural network models, the advantages of having the format, and the potential applications that can be created to use the format to facilitate the sharing of neural network models over the internet**.**

This paper has also outlined the specifications of such a file format**.** While the proposed format is not perfect, has drawbacks **(**such as its inefficiency in storing the topology of networks where two adjacent layers are not complete graphs**)** and is not universal as it does not cover some exotic neural network types, WNN is a start and can be used as a foothold to build a truly universal and maximally efficient format**.**

7. References

Clarkson, T. G. (1996). Introduction to neural networks. *Neural Network World*, *6*(2), 123-130. doi:10.1201/9781482277180-13

Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., & Philbrick, K. (2017). Toolkits and libraries for deep learning. *Journal of Digital Imaging*, *30*(4), 400-405. doi:10.1007/s10278-017-9965-6

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359-366. doi:10.1016/0893-6080(89)90020-8

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*, 115-133. doi:10.1007/BF02478259

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481-1497. doi:10.1109/5.58326

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408. doi:10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536. doi:10.1038/323533a0