Architecture Write-up: Discovering Novel Memory Cell Architectures for Recurrent Neural Networks Using Evolutionary Algorithms

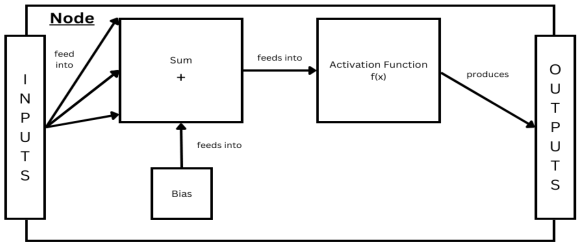
Jared Murphy
Graduate Student
Department of Software Engineering
Rochester Institute of Technology
jmm2188@rit.edu

Travis Desell
Associate Professor
Department of Software Engineering
Rochester Institute of Technology
tjdvse@rit.edu

The overall thesis of this work is that the search for optimal RNN architecture should be automated when possible. Not only is automation the most cost effective method, but given the enormous complexity of modern day recurrent neural networks, likely no human being is capable of handling the task of discovering the optimal build. Just as backpropagation efficiently and automatically adjusts the weights of the network, so too can evolutionary algorithms adjust the architecture, i.e. choose what node types, edge types, layering and connections are used in network design.

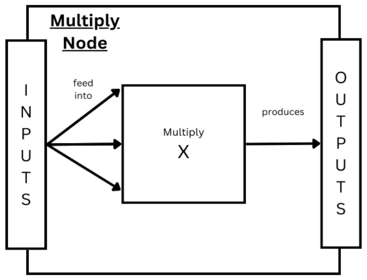
Ororbia, ElSaid and Desell saw promising results when the evolutionary algorithm EXAMM was allowed to use complex memory cells in combination with simple nodes, i.e. nodes that employed a sum with tanh activation function, as building blocks for evolving RNN's. It stands to reason that comparable and interesting results could be achieved if complex memory cells were broken down into their constituent parts and given to EXAMM for evolving RNN's. The diagrams below help to explain how this could be achieved.

Node



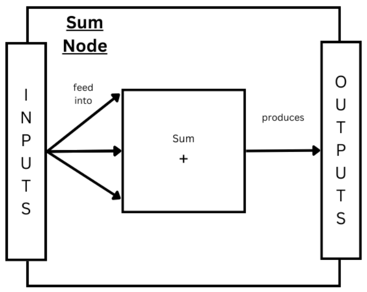
A Node is defined as an object that contains a value called bias. This bias is trainable through back-propagation. A node takes several numerical inputs and then sums them along with the bias. It then passes the total through a differentiable activation function f(x), and produces a single value that can be propagated forward to one or more other entities. This is one of the core building blocks of all neural networks.

Multiply Node



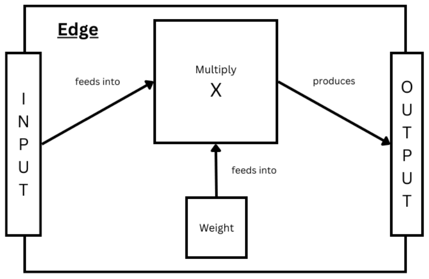
A multiply node performs the simple task of multiplying several inputs and produces their product as an output to be passed to one or more entities. This node type is one of building blocks for memory cells. It is where elements of genetic programming begin to be incorporated into the process.

Sum Node



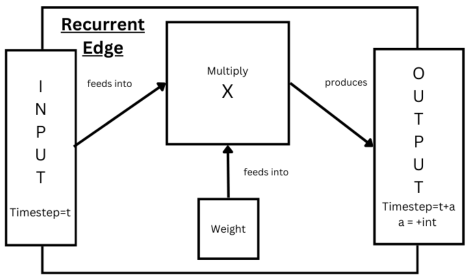
This node takes inputs and produces their sum as an output to one or more other entities. This node is also an element of genetic programing, and is one of the building blocks for complex memory cells.

Edge



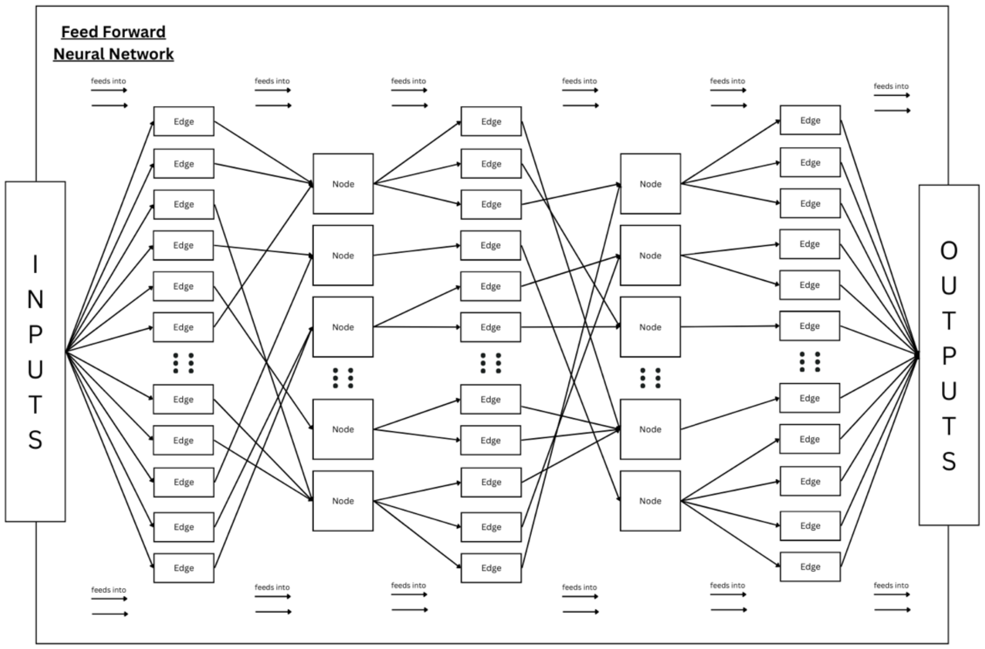
An edge takes a single input, multiplies it by a single value called weight, and produces a single output to be passed to a single entity. Weight is trainable using backpropagation.

Recurrent Edge



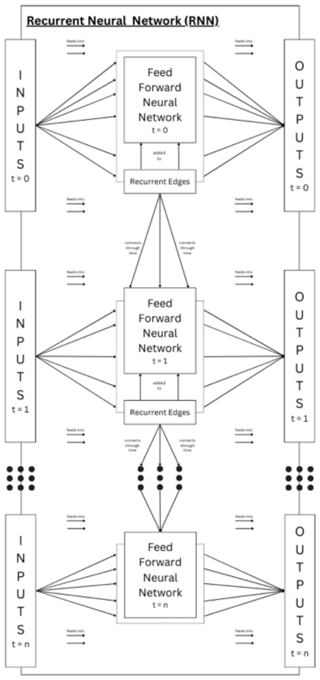
The Recurrent Edge object is the same as Edge except for one key difference: A Recurrent Edge produces an output that occurs at a future time step. This temporal element is what allows for the shift between Feed Forward Neural Networks and Recurrent Neural Networks.

Feed Forward Neural Network



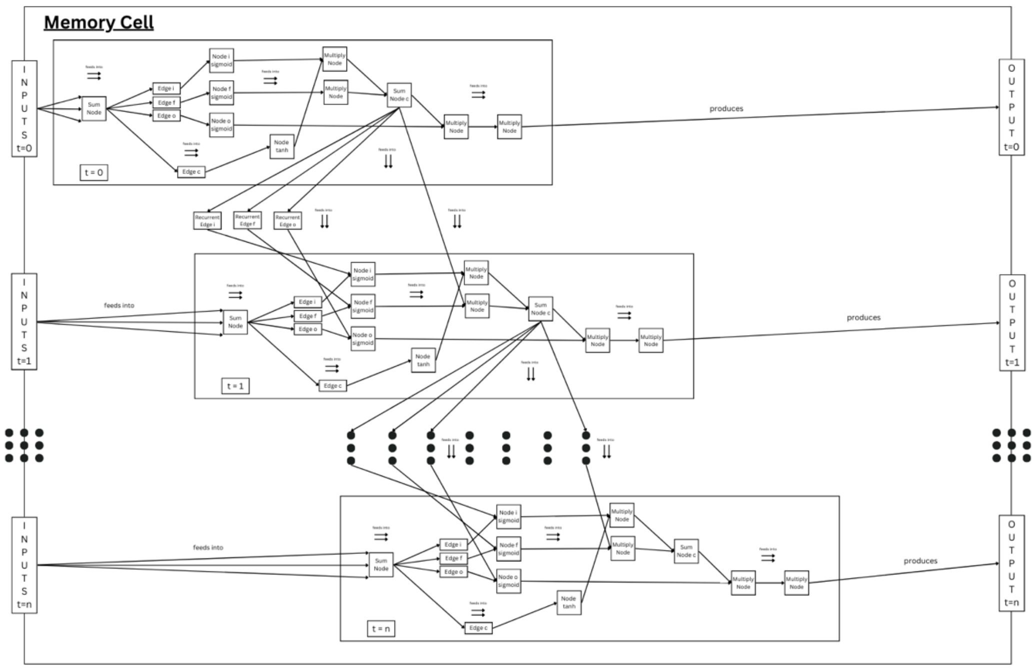
A Feed Forward Neural Network (NN) is a collection of Nodes and Edges organized in "layers". An input layer is fed through several hidden layers of Edge's and Node's to produce an output layer. It is more common for edges to be represented as the arrows connecting nodes, however for this project it is important to change the frame of reference to clearly see that a NN is nothing more than a collection of Nodes and Edges.

Recurrent Neural Network



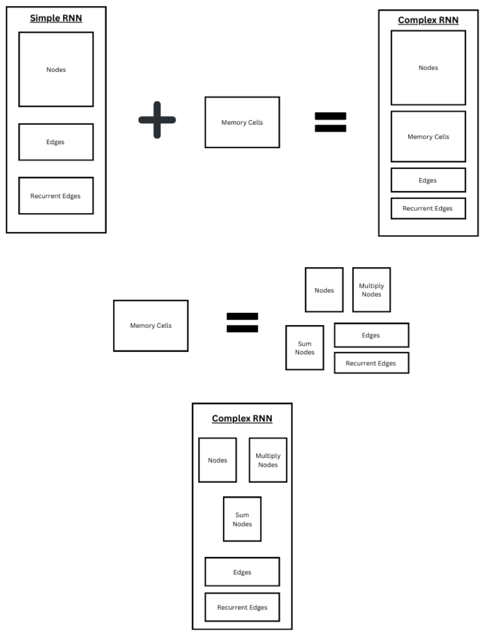
A Recurrent Neural Network (RNN) is an object that connects several NN's through time using Recurrent Edges. This allows for Time Series Data to be used for training.

Memory Cell



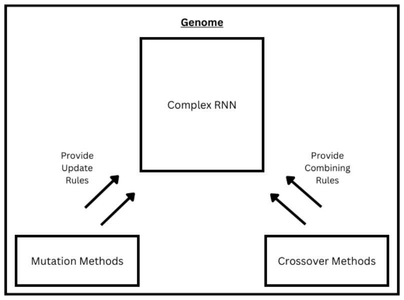
A memory cell is a collection of Edges, Recurrent Edges, Nodes, Sum Nodes and Multiply Nodes. It can be thought of a RNN in and of itself with the addition of the Sum Node and Multiply Node. Please zoom in to the photograph for more detail.

Complex RNN



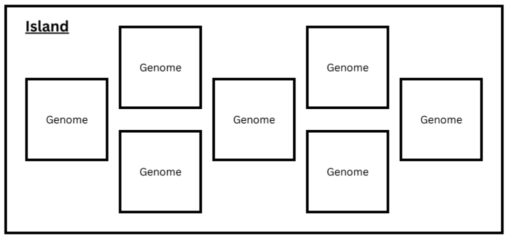
In this project, a Simple RNN will be defined as a network of Nodes, Edges and Recurrent Edges, whereas a Complex RNN will be defined as a Simple RNN with the addition of Memory Cells. Because Memory Cells were shown in the previous diagram to be a collection of Nodes, Multiply Nodes, Sum Nodes, Edges and Recurrent Edges, we can equally define a Complex RNN to be a collection of those constituent parts.

Genome



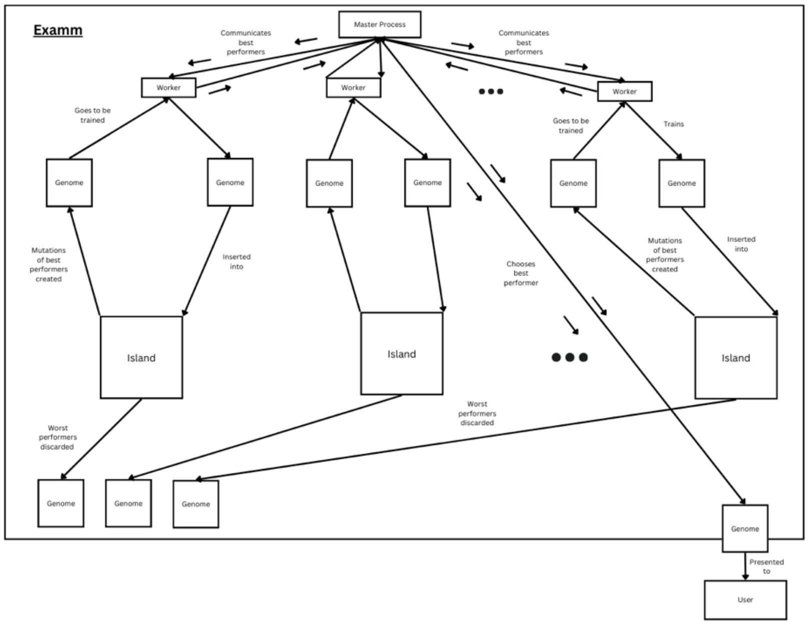
A Genome is an object that provides added functionality to a Complex RNN. A Genome contains methods for mutating the network, as well as methods combining it with others. This added functionality is what allows for the evolutionary process to unfold.

Island



Islands are data structures that delineate populations of Genomes which are used to breed with one another. Genomes on a specific island are combined and mutated in round robin fashion. When new genomes are produced and trained, the worst performers are discarded.

EXAMM



The EXAMM algorithm contains a master process which manages several worker processes. It is the job of each worker to train and evaluate each new Genome with respect to their specific Island's population. The workers manage the evolutionary process for each island and communicate with the master process to decide which architectures were the best performers. The end state of the algorithm is for the user to be presented with the Genome that contains the Complex RNN with the optimal architecture for the data it was trained on.