# An Algebraic Generalization for Graph and Tensor-Based Neural Networks

CIBCB 2017 – Manchester, U.K.

Ethan C. Jackson[1], James A. Hughes[1], Mark Daley[1], and Michael Winter[2]

August 24, 2017

The University of Western Ontario — London, Canada[1]
Brock University — St. Catharines, Canada[2]

# Motivation

# Neural Network Representations

No current standard language or mathematical framework for neural networks — no standard representation.

# Neural Network Representations

No current standard language or mathematical framework for neural networks — no standard representation.

In computational neuroscience:

- CSA — Connection Set Algebra [5]
- NeuroML — XML-based model description [6]
- Custom representations

# Neural Network Representations

No current standard language or mathematical framework for neural networks — no standard representation.

In computational neuroscience:

- CSA — Connection Set Algebra [5]
- NeuroML — XML-based model description [6]
- Custom representations

In computer science:

- Tensor-based — TensorFlow[1], Theano[12], Keras [2]
- Graph-based — NEAT, HyperNEAT [7]
- Custom representations

# Evolutionary Computation

NEAT

- Architecture and parameters evolved simultaneously
- Connections are encoded point-to-point

Tensor Frameworks

- Architectures typically hand-designed
- High level interfaces increasingly available
- Evolutionary algorithms beginning to be used [4]

# Connectomics

Connectomics is concerned with studying brain function in relation to network organization and dynamics. See work by O. Sporns [10], [11].

- Self-similarity, fractal-like patterns
- At multiple levels of organization

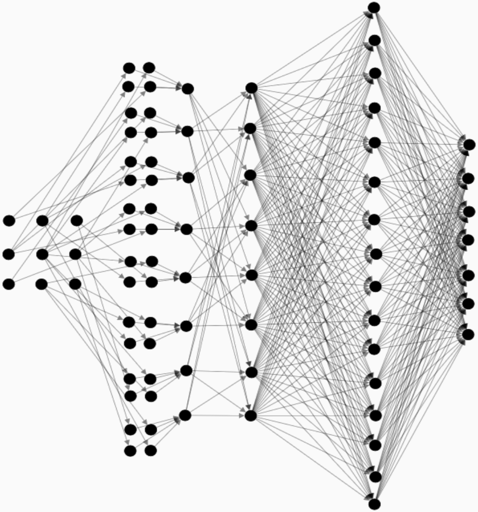
The human genome does not encode each neural connection, point-to-point — it does not encode an adjacency list.

# Goals

Find a common mathematical framework for existing tools that provides:

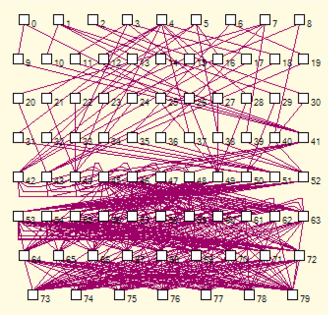
- A universal network description language
- Modular, generative operations
- Symbolic reasoning
- Model portability
- Self-similar or fractal-like representation

# Example — Feed-Forward Network Architecture



A feed-forward neural network architecture.

# Example — Graph and Tensor



Example network as a HyperNEAT genome.

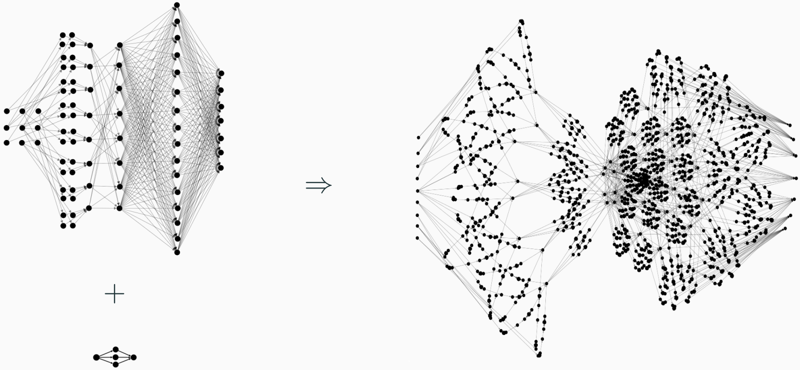
$$
\begin{aligned}
T_1 &= R_1 \cdot \iota_{F_1 \to F_1 \oplus F_2} + R_2 \cdot \kappa_{F_2 \to F_1 \oplus F_2} \\
T_2 &= \pi_{F_1 \oplus F_2 \to F_1} \cdot R_3 \cdot \iota_{A_1 \to A_1 \oplus B_1} \\
&\quad + \rho_{F_1 \oplus F_2 \to F_2} \cdot R_5 \cdot \kappa_{B_1 \to A_1 \oplus B_1} \\
T_3 &= \pi_{A_1 \oplus B_1 \to A_1} \cdot R_4 \cdot \iota_{A_2 \to A_2 \oplus B_2} \\
&\quad + \rho_{A_1 \oplus B_1 \to B_1} \cdot R_6 \cdot \kappa_{B_2 \to A_2 \oplus B_2} \\
T_4 &= \pi_{A_2 \oplus B_2 \to A_2} \cdot R_7 + \rho_{A_2 \oplus B_2 \to B_2} \cdot R_8 \\
T_5 &= R_9
\end{aligned}
$$

$$eval(R_1 \ldots R_9) : In \to Out :=$$

$$T_1 \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5$$

Example network as a Tensor expression.

# Example — Network, Module, and Substitution



The substitution of network connections by a repeated module.

# Example — Matrix

$$
\begin{array}{cc}
 & \begin{array}{ccccccccc} In & F_1 & F_2 & A_1 & A_2 & B_1 & B_2 & C & Out \end{array} \\
\begin{array}{c} In \\ F_1 \\ F_2 \\ A_1 \\ A_2 \\ B_1 \\ B_2 \\ C \\ Out \end{array} &
\begin{bmatrix}
 & R_1 & R_2 & & & & & & \\
 & & & R_3 & & & & & \\
 & & & & & R_5 & & & \\
 & & & & R_4 & & & & \\
 & & & & & & & R_7 & \\
 & & & & & & R_6 & & \\
 & & & & & & & R_8 & \\
 & & & & & & & & R_9 \\
 & & & & & & & & 
\end{bmatrix}
\end{array}
$$

A relational matrix describing a high-level neural network architecture.

# Background

# Connection Set Algebra

CSA is a mathematical framework for compactly describing neural architectures.

- Masks
- Value sets
- Elementary connection sets
- ‘Pseudo’-algebraic operations
- Functions for establishing connectivity

# Connection Set Algebra

CSA is a mathematical framework for compactly describing neural architectures.

- Masks — Boolean relations
- Value sets — Real-valued vectors
- Elementary connection sets — Elementary relations
- ‘Pseudo’-algebraic operations — No formal properties
- Functions for establishing connectivity — Incomplete set

# Relations

Classical or Boolean-valued relations

$$R \subseteq A \times B \qquad R : A \times B \to \{0,1\}$$

Often interpreted as Boolean-valued matrices, e.g.:

$$\begin{array}{c c} & \begin{array}{cccc} B_1 & B_2 & \dots & B_n \end{array} \\ \begin{array}{c} A_1 \\ A_2 \\ \dots \\ A_n \end{array} & \begin{bmatrix} 1 & 0 & & 0 \\ 1 & 1 & & 1 \\ & & \ddots & \\ 1 & 0 & & 1 \end{bmatrix} \end{array}$$

# Relation Algebraic Operations

A relation algebra is an extended, residuated Boolean algebra.

Elementary relation algebraic operations:

- Union*
- Intersection*
- Composition*
- Complement*
- Difference*
- Converse

*Corresponds to set-theoretic operation.

Converse reverses order of all pairs.

# Semiring Matrices

A semiring is a very general algebraic structure. Its elements can be used to model connectivity between neurons.

$R = (\mathbb{R}, +, \cdot, 0, 1)$ with the standard addition and multiplication operations forms a semiring.

A matrix over the semiring $R$:

$$\begin{bmatrix} 1.2 & 8.2 & & 0 \\ 0.1 & 9.9 & & 1.2 \\ & & \ddots & \\ 3.4 & 0.4 & & 1 \end{bmatrix}$$

# A Unified Approach

A relation algebra is embedded in the matrix algebra over a large class of semirings that includes the field $\mathbb{R}$ and functions over $\mathbb{R}$.

- Requires minimal extension – a *flattening* operation
- Connectivity and values can be modelled using a single object
- A basic descriptive framework can be implemented as an extension of linear algebra
- Algebraic properties, other operations ‘for free’

# Algebraic Framework and Extended Operations

# Core Algebraic Framework

The core algebraic framework consists of matrices over the field $\mathbb{R}$ within which relations are the matrices with $\{0,1\}$-valued entries.

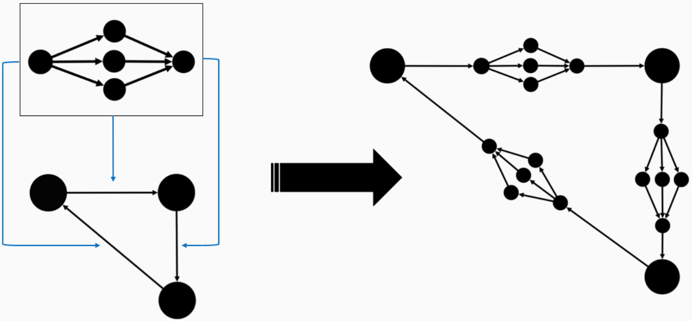
We use special matrices called **relational sums** to compose modular network architectures.

$$\begin{array}{cc} & N \\ N & \begin{bmatrix} \mathrm{R} \end{bmatrix} \end{array} \quad \Rightarrow \quad \begin{array}{cc} & \begin{array}{cc} N & B \end{array} \\ \begin{array}{c} N \\ A \end{array} & \begin{bmatrix} \mathrm{R} & 0 \\ 0 & 0 \end{bmatrix} \end{array}$$

Application of operation *inject-top-left* to a matrix R. Left: a matrix $N \to N$. Right: a matrix denoted by $N \oplus A \to N \oplus B$.

# Extended Operations — Substitution

Using relational sums, we implemented extended operations to build modular networks.



Substitution of connections by a module. The resulting network could in turn be used as another substitution module.

# Extended Operations — Substitution

Because relations are embedded in the matrix algebra, we can conveniently use matrices and relational reasoning to define operations.

$$\begin{array}{cc}
 & \begin{array}{cccccccc} N_1 & N_2 & N_3 & C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\
\begin{array}{c} N_1 \\ N_2 \\ N_3 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array} &
\begin{bmatrix}
 & \cancel{1} & & \underline{1} & & & & \\
 & & 1 & & & & & \\
1 & & & & & & & \\
 & & & & 0.2 & 0.3 & 0.6 & \\
 & & & & & & & \\
 & & & & & & & 0.2 \\
 & & & & & & & 0.9 \\
 & \underline{1} & & & & & & 0.1
\end{bmatrix}
\end{array}$$

- Substitution of one connection by a module
- Can prove that connectedness is preserved

# Extended Operations — Substitution

Algebraic definition and reasoning about matrices using relational methods.

## Definition

Let $N$ and $C$ be finite sets, $Net : N \rightarrow N$ and $S : C \rightarrow C$ be relations, and $(N_x, N_y) \in Net$. The substitution operation is defined by:

$$\begin{aligned} & subst(Net, S, (N_x, N_y)) := \\ & \quad iTL(Net - \{(N_x, N_y)\}, C, C) \cup \\ & \quad iBR(S, N, N) \cup \{(N_x, C_1), (C_n, N_y)\} \end{aligned}$$

## Lemma

*Let $N$ and $C$ be finite sets, $Net : N \rightarrow N$ and $S : C \rightarrow C$ be relations, and $(N_x, N_y)$ be an element in $Net$. If $(C_1, C_n) \in S^+$ then $(N_x, N_y) \in NetS^+$, the transitive closure of $NetS$, where $NetS = subst(Net, S, (N_x, N_y))$.*

# Summary

In our paper we show that:

- Neural network architecture can be described and manipulated using existing and custom relational operations
- Operations are generic with respect to the set used to describe neural connectivity
- Properties about operations can be proven using relation algebraic methods

# Next Steps

*"Models will become more like programs ... will be grown automatically."* [3]

— François Chollet, author of Keras

# Modular Neuroevolution

$$
\begin{array}{c|c}
 & \begin{matrix} In & F_1 & F_2 & A_1 & A_2 & B_1 & B_2 & C & Out \end{matrix} \\
\begin{matrix} In \\ F_1 \\ F_2 \\ A_1 \\ A_2 \\ B_1 \\ B_2 \\ C \\ Out \end{matrix} &
\begin{bmatrix}
 & R_1 & R_2 & & & & & & \\
 & & & R_3 & & & & & \\
 & & & & & R_5 & & & \\
 & & & & R_4 & & & & \\
 & & & & & & & R_7 & \\
 & & & & & & R_6 & & \\
 & & & & & & & R_8 & \\
 & & & & & & & & R_9 \\
 & & & & & & & &
\end{bmatrix}
\end{array}
$$

$$\Downarrow$$

$$R : N \to N \qquad N = In \oplus F_1 \oplus F_2 \oplus A_1 \oplus A_2 \oplus B_1 \oplus B_2 \oplus C \oplus Out$$

# Modular Neuroevolution

A relational representation neural networks could have several advantages over other approaches.

- Multiple levels of organization
- Algebraic manipulation
- Conventional ANN modules
- Relational and functional modules

# Modular Neuroevolution

- We are currently working on a neuroevolution framework based on Cartesian genetic programming (CGP) and existing work on CGP-based ANNs.

# Modular Neuroevolution

- We are currently working on a neuroevolution framework based on Cartesian genetic programming (CGP) and existing work on CGP-based ANNs.
- The algebraic framework introduced in this work will be the basis for the genetic representation and operators.

# Modular Neuroevolution

- We are currently working on a neuroevolution framework based on Cartesian genetic programming (CGP) and existing work on CGP-based ANNs.
- The algebraic framework introduced in this work will be the basis for the genetic representation and operators.
- Evolved networks will be a mix of *de novo* evolved modules and existing modules in the form of ANN layers, relational, and functional programs.

# Modular Neuroevolution

- We are currently working on a neuroevolution framework based on Cartesian genetic programming (CGP) and existing work on CGP-based ANNs.
- The algebraic framework introduced in this work will be the basis for the genetic representation and operators.
- Evolved networks will be a mix of *de novo* evolved modules and existing modules in the form of ANN layers, relational, and functional programs.
- The representation will be based on a mapping between algebraic expressions and a recursive, modular adjacency structure.

**Thank you!**

# References i


Abadi, M., et. al.: TensorFlow: Large-scale machine learning on heterogeneous systems, (2015).

Chollet, F. et. al.: Keras. github.com/fchollet/keras/ (2015).

Chollet, F.: The Future of Deep Learning. https://blog.keras.io/the-future-of-deep-learning.html (2017).

Davison, J.: DEvol - Automated deep neural network design with genetic programming. https://github.com/joeddav/devol (2017).

- Djurfeldt, M.: The Connection-set Algebra—A Novel Formalism for the Representation of Connectivity Structure in Neuronal Network Models. Neuroinformatics 10(3), 1539-2791 (2012).
- Gleeson, P. and Crook, S. and Cannon, R. C. and Hines, M. L. and Billings, G. O. et al.: NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail. PLoS Computational Biology 6(6) (2010).
- HyperNEAT: eplex.cs.ucf.edu/hyperNEATpage/
- Jones, E., Oliphant, E., Peterson, P., et al.: SciPy: Open Source Scientific Tools for Python, scipy.org/ (2001 – ).

- Killingbeck, D., Santos Teixeira, M., and Winter, M.: Relations among Matrices over a Semiring. In Kahl, W., Oliveira, J.N., Winter, M. (Eds.): RAMiCS 15. LNCS 9348, 96-113 (2015).
- Sporns, O.: Discovering the Human Connectome. MIT Press (2012).
- Sporns, O.: Small-world connectivity, motif composition, and complexity of fractal neuronal connections. BioSystems 85 5564 (2006).
- Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arxiv.org/abs/1605.02688 (2016).

Van der Walt, S., Colbert, S.C., and Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011)