A Roadmap for Reverse-Architecting the Brain’s Neocortex

J E Smith

FCRC 2019
“There is nothing that is done in the nervous system that we cannot emulate with electronics if we understand the principles of neural information processing.”

Motivation

- The human brain is capable of:
  - Accurate sensory perception
  - High level reasoning and problem solving
  - Driving complex motor activity

- With some very impressive features:
  - Extremely efficient (20 watts)
  - Very flexible – supports a wide variety of cognitive functions
  - Learns dynamically, quickly, and concurrently with operation

- Far exceeds anything conventional machine learning has achieved
  - Will the trajectory of conventional machine learning ever achieve the same capabilities?
  - OR should we seek new approaches based on the way the brain actually works?
Milestone Temporal Neural Network

- Continual, Unsupervised Clustering
  - Learn and identify similar input patterns and map them to concise cluster identifiers (CIds)
  - Training and inference done concurrently and continually

- Emergent
  - All neural operations are local
  - Global behavior emerges

- Hardware implementation
  - Fast
  - Energy efficient
  - Implementable with digital CMOS

- This is a processing core
  - Not a complete system
  - Interfaces with external world will be required
  - For advanced apps this will be challenging

It has a mind of its own!
Outline

- The Biological Neocortex
- Computer Meta-Architecture
- Primitive Abstraction: Biological to Computational
- Column Level Abstraction (“RTL”)
- Mathematical Underpinnings
- Digital CMOS Implementation
- Closing Remarks
The Biological Neocortex
The Neocortex

- Neocortex
  - The “new shell” surrounding the older brain
  - Performs:
    - sensory perception
    - cognition
    - intellectual reasoning
    - generation of high level motor commands

- Thin sheet of neurons
  - 2 to 3 mm thick
  - Area of about 2500 cm²
  - Folds increase area
  - Approx. 100 billion neurons
  - 10K synapses each
Physical Architecture of the Neocortex

- *Physical* architecture probably corresponds to *functional* architecture
- Physical Hierarchy (top down)
  - Lobes
  - Regions
  - Subregions
  - Macro-Columns
  - Micro-Columns
  - Neurons
Physical Architecture Bottom-Up

Micro-Column $O(100)$ neurons

Macro-Column $O(100)$ micro-columns

Regions, Subregions Many Macro-Columns

Copyright JE Smith  (June 2019)
Biological Neurons

*Excitatory pyramidal neuron surrounded by three inhibitory neurons of different types*

tiny dots are **Synapses**
(connection points)

**Dendrites** (Inputs)

**Body**

**Axon** (Output)

from deFelipe 2011
Excitatory Neuron Model

- Basic Spike Response Model (SRM0) -- Kistler, Gerstner, & van Hemmen (1997)

1) A volley of spikes is applied at inputs
2) At each input’s synapse, the spike produces a weighted response function
3) Responses are summed linearly at neuron body
4) An output spike is emitted if/when potential exceeds threshold value ($\theta$)
Meta-Architecture
Engineering highly complex systems requires abstraction

- Conventional computer architecture contains many levels of abstraction

**Architecture and Abstraction**

- Application Software
  - HLL Procedure Hierarchy
    - HLL statements
      - Machine Language statements
        - Processor Core + Memory
          - Functional Block Hierarchy
            - RTL functional blocks
              - Logic gates
                - Physical CMOS

*lowest practical design layer*

fundamental abstraction: *hardware to software*

*lowest practical design layer*

fundamental abstraction: *electrical circuits to logic*
Comprehending neocortical computing will require levels of abstraction

- We (humans) can only comprehend assemblies of a certain limited complexity
  So, we rely on abstraction
- Fortunately, the physical hierarchy seems to match our ability to comprehend
  Each functional block composed of 10 to 100 lower level blocks

**Neuro Architecture Stack**

- Neocortex
- Lobes
- Region Hierarchy
- Macro-Columns
- Feedforward/Micro-Columns
- Model Neurons
- Biological Neurons

**Spatial Thinking**

- fundamental abstraction in here somewhere?

**Temporal Processing**

- fundamental abstraction: electrical circuits to primitive computing elements
Start at the bottom of the stack
  • With biological neurons

Reverse-architect to the top
  • A Neuromorphic Architecture implements the computing paradigm(s) used in the neocortex
  • Neuromorphic Circuits are electrical circuits that function in ways similar to neurons and can be used to implement Neuromorphic Architectures.
  • Neuromorphic Architectures do not require Neuromorphic Circuits
Near Term Roadmap

- First, focus on abstraction from biological neurons to computing elements
  - Consider results from experimental neuroscience
  - Consider models from theoretical neuroscience
  - Postulate a set of basic elements

- Next, develop quasi-standard building blocks (10-100 neurons)
  - Analogous to RTL blocks
  - Develop these blocks by constructing and experimenting with Temporal Neural Networks

- First Major Milestone: Deep TNNs
  - Described earlier

- Three layers of abstraction are simultaneously in play:
  - Model neurons
  - Column-level quasi-standard assemblies
  - Macro-Columns
Primitive Abstraction: Biological to Computational
Basic Architectural Elements

Temporal coding

Precise timing relationships

Values encoded as spike times relative to \( t = 0 \)

Excitatory Neurons

Input spikes \( x \)

Output spikes \( y \)

Temporal Neural Networks

Input Values

Encode to Spikes

Temporal Coding

Output Values

Decode from Spikes

Excitatory Neurons

Input Values

Encode to Spikes

Output Values

Decode from Spikes

Bulk Inhibition

STDP

Learning/forgetting params

Input Spike

Output Spike

Input Spike

Output Spike

Initial w(0)

w(x+1)

w(x)
Temporal Coding

- Information is communicated via transient events
  - e.g., voltage spikes
  - Hereafter “spike” is shorthand for “transient temporal event”
- Values are encoded via spike timing relationships across parallel communication lines
  - Based on spike times relative to first \((t = 0)\)
  - Low resolution: 1-in-8, say
  - Example is not a “toy” – values are realistic

Note: in practice, coding is sparser than in this example
The Temporal Resource

The flow of time can be used effectively as a communication and computation resource.

- The flow of time has some ultimate engineering advantages
  - It requires no space
  - It consumes no energy
  - It is free – time flows whether we want it to or not

- Yet, we (humans) try to eliminate the effects of time when constructing computer systems
  - Synchronizing clocks & delay-independent asynchronous circuits
  - This may be the best choice for conventional computing problems and technologies

- How about natural evolution?
  - Tackles completely different set of computing problems
  - With a completely different technology
Compare with Rate Coding

- Plot spikes on same biological time scale
- Both methods convey similar information
- Temporal method is
  - An order of magnitude faster
  - An order of magnitude more efficient (#spikes)

The temporal coding method has significant, broad experimental support
  - The rate method does not.

The diagram shows two time scales, one for rate coding and one for temporal coding, highlighting the efficiency and timing differences.
Temporal Neural Network

- A feedforward network of model neurons
  - Values communicated via temporal codes (implemented as “spikes”)
  - Feedforward flow (without loss of computational generality)
  - Computation: a wave of spikes passes from inputs to outputs
  - At most one spike per line per computation

```
Input Values
Encode to Spikes

x_1 x_2 ... x_m

Decide from Spikes

z_1 z_2 ... z_p

"parrot looking over its right wing"
```
Primary goal: a computing paradigm that learns in an unsupervised, continual, fast, and energy efficient way

- Separates this research from vast majority of “Spiking Neural Network” (SNN) research

Neural Networks

- Rates
- Spikes
- Theory
- Temporal
- Implementation
- Supervised
- Backprop
- Training
- Localized
- Unsupervised

Classic ANNs →
Deep CNNs
- Classification
- Supervised
- Compute-intensive training

“SNNs”
- Classification
- Supervised
- Compute-intensive training
- Accuracy is a struggle
- Energy efficiency in doubt

“SNNs”
- Classification
- Supervised
- Compute-intensive training
- Accuracy is a struggle
- Probable energy efficiency benefits

“SNNs” (TNNs)
- Clustering
- Unsupervised
- Simple, dynamic training
- Accuracy-neutral (?)
- Probable energy efficiency benefits
Excitatory Neuron Model (repeat)

- Basic Spike Response Model (SRM0) -- Kistler, Gerstner, & van Hemmen (1997)

1) A volley of spikes is applied at inputs
2) At each input’s synapse, the spike produces a weighted response function
3) Responses are summed linearly at neuron body
4) An output spike is emitted if/when potential exceeds threshold value ($\theta$)
Bulk Inhibition

Inhibitory neurons act *en masse* over a local volume of neurons
- A “blanket” of inhibition

A few inhibitory neurons control many excitatory neurons
- Up to 30 synapses per target excitatory neuron (avg. = 15)
- Some connections directly to excitatory body and axon

Model as parameterized Winner-Take-All (WTA) inhibition

Note: this mechanism is probably built into a soft synchronization method based on inhibitory oscillations
Spike Timing Dependent Plasticity – where the magic is

- Each synapse updates weight based on current weight and local spike time relationships
- Implemented as a small finite state machine
- Many methods under study
- Decision tree + update functions:

If \( x(s) \leq z(s) \) then
\[
\begin{align*}
    w(s+1) &= w(s) + B(\mu_1) \\
    w(s+1) &= w(s) - B(\mu_2)
\end{align*}
\]

Else
\[
\begin{align*}
    w(s+1) &= w(s) - B(\mu_3) \\
    w(s+1) &= w(s) + B(\mu_4)
\end{align*}
\]
Spike Timing Dependent Plasticity (STDP) establishes weights in a way that decodes the most frequent input patterns:

- Relies on bimodal synaptic weight distribution (0 or $W_{\text{max}}$)
- Timing of output spikes depends on response function
  
  Step no-leak in this example

- In general decodes clusters rather than individual patterns

0: spike @ $t = 0$
- : = no spike

$$\theta = 9$$
In the neocortex, computation is inextricably combined with obfuscating infrastructure.

In the computer architecture “lab”, we can consider the computing paradigm absent all the complications.
A Pantheon of Neuroscience Architects

- Theoretical neuroscientists have been developing brain-based computing paradigms for over two decades
  - Lots of good ideas have been put forward
  - Computer architects don’t start from scratch

Simon Thorpe
  - Damien Querlioz
  - Rudy Guyonneau
  - Rufin VanRullen
  - Timothée Masquelier
  - Wolfgang Maass
  - Henry Markram
  - Wulfram Gerstner
  - Sander Bohte
  - Wolfgang Singer
  - Pascal Fries

- temporal coding,
- STDP,
- TNN architectures
- TNN (SNN) theory
- STDP
- Neuron Models, STDP
- TNN architecture
- Inhibitory oscillation;
- soft synchronization
Column Level Abstraction: “RTL”
Column Level Abstraction

- Combine primitives into higher level computing assemblies
  - Analogous to Register Transfer Level (RTL) in digital logic
  - Design will probably be done at this level
Computational Column (CC)

- Basic TNN building block
- Learns and maps inputs having similar features to the same Cluster Id
- Input lines may be interpreted as features
  - The presence of a spike indicates the presence of the feature
  - The timing of a spike indicates the relative strength of the feature
- A ClId is a 1-hot temporal coding
  - The better the cluster “match”, the earlier the spike
  - ClIds become features for the next network Layer
TNN Roadmap Waypoints
Waypoint 0: Input Encoding

- Leverage biology
- Example: OnOff retinal ganglion cells
  - Perform edge detection
- Encode spikes according to contrast between center and surround
  - Most intense contrast yields earlier spikes
- However, binarize primary input to simplify initial experiments
  - Separates Layer 1 temporal computation from temporal communication

Leveraging biology, retinal ganglion cells such as OnOff cells can be used for edge detection. Spikes are encoded according to the contrast between the center and surround of the receptive field. The most intense contrast yields earlier spikes, allowing for a simpler initial experiment by binarizing the primary input. This separation helps in isolating the temporal computation from the temporal communication within Layer 1.
Waypoint 1: Dense-to-Sparse CC

- Unsupervised clustering
  - Example 6x6 RFs from MNIST – OnOff encoded, *binarized*
STDP Works.
Waypoints 2 & 3: Sparse-to-Sparse CCs

- The goal is a “cookie cutter” CC
  - To allow construction of arbitrarily wide, arbitrarily deep TNNs
  - No one has been successful to date – Wide-open research area
Temporal coding
- efficient coding based on temporal relationships

Excitatory Neurons
- consistent with the rules of Newtonian time

Inhibition Blocks
- consistent with the rules of Newtonian time

STDP
- localized, unsupervised learning

Dendritic Computation
- largely unexplored

Large idea space practically un-touched

Compound Synapses
- biologically correct; largely unexplored

Large idea space practically un-touched

Neuron Body Potential
\[ \sum D W \]

Dendrite

Input Values
- Encode to Spikes

Output Values
- Decode from Spikes

Temporal Neural Networks
- Computation proceeds as a wave of spikes passes from inputs to outputs

Large interconnect space

F1
F4
Fn
F2
F3

Input Values
- Encode to Spikes

Output Values
- Decode from Spikes

Dendritic Computation
- largely unexplored

Large idea space practically un-touched

Compound Synapses
- biologically correct; largely unexplored

Large idea space practically un-touched

Neuron Body Potential
\[ \sum D W \]
Spikes are not the only way to encode values as the times of transient temporal events.

Edges work, too.
- Signal via 1 → 0 transitions

Efficiencies remain intact

Edges + race logic yields direct off-the-shelf CMOS implementation

An alternative to neuromorphic circuits

*Race logic: Madhavan, Sherwood, Strukov, UC-Santa Barbara

see 2018 ISCA paper
Mathematical Underpinnings
Contrasting Mathematical Approaches

- Neuroscience approach
  - Real arithmetic – differential equations
  - Supports unbounded computational resolution
  - Discretization done implicitly through conversion to floating point

- Computer Architecture approach
  - Simple mathematics (Boolean algebra)
  - Inherently discrete

- A Computer Architecture approach to modeling neural operation
  - The devices being modeled are naturally very low resolution (1-in-8)
  - Use discrete math and small integers to implement temporal functions

low resolution, unary computation
A Space-Time Computing Network is a feedforward composition of functions, $F_i$, where:

1) Each $F_i$ has a **finite state implementation**
2) Each $F_i$ is **causal**
   - The output spike time is independent of later input spike times
   - No spontaneous output spikes
3) Each $F_i$ is **invariant**
   - If all the input spikes are delayed by some constant amount then the output spike is delayed by the same constant amount
A Space-Time Computing Network is a feedforward composition of functions, \( F_i \), where:

1) Each \( F_i \) has a \textit{finite state implementation}

2) Each \( F_i \) is \textit{causal}
   - The output spike time is independent of later input spike times
     - No spontaneous output spikes

3) Each \( F_i \) is \textit{invariant}
   - If all the input spikes are delayed by some constant amount then the output
     spike is delayed by the same constant amount

\textit{TNNs are an important special case}
(Newtonian) Space-Time Algebra

Bounded Distributive Lattice

- 0, 1, 2,…, $\infty$
- Interpretation: points in time
- not complemented

Primitive Operators

- **inc**: $b = a + 1$
- **min**: if $a < b$ then $c = a$
  else $c = b$
- **lt**: if $a < b$ then $c = a$
  else $c = \infty$
Theorem: Any feedforward composition of s-t functions is an s-t function

⇒ Build networks by composing s-t primitives

• Example:

note: shorthand for $n$ increments in series: $a \overline{n} b = a + n$
## Elementary Functions

- Table of all two-input s-t functions
  - All implementable with the three primitives

<table>
<thead>
<tr>
<th>function</th>
<th>name</th>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>if ( a &lt; b ) then ( a ); else ( b )</td>
<td>min</td>
<td>( \wedge )</td>
</tr>
<tr>
<td>if ( a \leq b ) then ( a ); else ( \infty )</td>
<td>less or equal</td>
<td>( \preceq )</td>
</tr>
<tr>
<td>if ( a \neq b ) then ( a ); else ( \infty )</td>
<td>not equal</td>
<td>( \neq )</td>
</tr>
<tr>
<td>if ( a &lt; b ) then ( a )</td>
<td>exclusive min</td>
<td>( \wedge \wedge )</td>
</tr>
<tr>
<td>else if ( b &lt; a ) then ( b ); else ( \infty )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if ( a &lt; b ) then ( a ); else ( \infty )</td>
<td>less than</td>
<td>( &lt; )</td>
</tr>
<tr>
<td>if ( a \geq b ) then ( a ); else ( b )</td>
<td>max</td>
<td>( \vee )</td>
</tr>
<tr>
<td>if ( a &gt; b ) then ( a )</td>
<td>exclusive max</td>
<td>( \wedge \vee )</td>
</tr>
<tr>
<td>else if ( b &gt; a ) then ( b ); else ( \infty )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if ( a \geq b ) then ( a ); else ( \infty )</td>
<td>greater or equal</td>
<td>( \succeq )</td>
</tr>
<tr>
<td>if ( a = b ) then ( a ); else ( \infty )</td>
<td>equal</td>
<td>( \equiv )</td>
</tr>
<tr>
<td>if ( a &gt; b ) then ( a ); else ( \infty )</td>
<td>greater than</td>
<td>( &gt; )</td>
</tr>
</tbody>
</table>
TNN Primitives Implemented as ST Functions

(sort is a space-time function)

Response function generator

SRM0 Neuron

WTA Inhibition
The Box: The way we (humans) think about computation

- We try to eliminate temporal effects when implementing functions
  - TNNs uses the uniform flow of time as a key resource
- We use \texttt{add} and \texttt{mult} as primitives for almost all mathematical models
  - Neither \texttt{add} nor \texttt{mult} (except add of a constant) is an \texttt{s-t function}
- We prefer high resolution (precision) data representations
  - \textit{Unary computing} practical only for very low-res direct implementations
- We strive for complete functional completeness
  - \texttt{s-t} primitives complete \textit{only} for \texttt{s-t functions}
  - There is no inversion, complementation, or negation
Digital CMOS Implementation
Race Logic*

- *Spikes* are not the only way to encode values as the times of transient temporal events.
- *Edges* work, too.
  - Signal via $1 \rightarrow 0$ transitions.
- Efficiencies remain intact.
- Combined with race logic yields direct off-the-shelf CMOS implementation.

\[\begin{align*}
\text{values} & \\
0 & \text{Spikes} \\
7 & \\
\infty & \\
3 & \\
0 & \text{Edges} \\
7 & \\
\infty & \\
3 & \\
\end{align*}\]

\[\text{time}\]

*Race logic: Madhavan, Sherwood, Strukov, UC-Santa Barbara*

\[\text{see 2018 ISCA paper}\]
Generalized Race Logic

- S-T primitives implemented directly with conventional digital circuits
  - Signal via 1 $\rightarrow$ 0 transitions

⇒ We can implement SRM0 neurons and WTA inhibition with off-the-shelf CMOS
  ⇒ Very fast and efficient TNNs
TNN Primitives Implemented with CMOS Gates

- Signal via edges w/ off-the-shelf CMOS
  - minimize static power
  - lots of wires
  - signaling and functional operation very sparse

- A direct implementation
  - An alternative to analog spiking neuromorphic circuits

\[
\begin{align*}
x_1 & < \Delta = 3 \\
x_2 & < \\
x_m & \leq 0 \\
x_1 & < \Delta = 2 \\
x_2 & < \\
x_m & \leq 0 \\
x_1 & < \Delta = 1 \\
x_2 & < \\
x_m & \leq 0 \\
x_1 & < \Delta = 0 \\
x_2 & < \\
x_m & \leq 0 \\
\end{align*}
\]
Put It All Together: 1st Major Milestone

- TNN with unsupervised, continual learning via STDP
- Describable w/ a temporal algebra
  - Supports low resolution, discrete computation
- Hardware implementation
  - Implementable with digital CMOS
  - Fast
  - Energy efficient

Temporal Neural Network

- Inference
  - non-binary combinatorial network
- Training
  - concurrent, local adjustment of synaptic weights

sequence of input patterns

sequence of output cluster identifiers (CIds)

similar input patterns map to same CId
Closing Remarks
The Barrier to Entry is Low

- The TNN literature is relatively small
  - TNN development is not very far along
  - So there isn’t a lot of stuff to learn
- Low computational requirements
  - A high-end desktop computer running parallel threads is adequate
- It is possible to be up to speed in a few months (at most)
  - Writing a simulator is a good way to start
Are We at a Tipping Point?

- Experimental neuroscience spans more than 100 years
  - The published literature is vast and continues to grow at a fast rate

- What if all experimental neuroscience research were to cease tomorrow?
  - Is enough already known to allow reverse-architecting the neocortex?

- This would a tipping point for computer architecture research
  - *No more experimental data is needed*
  - We may already be there, or are fast approaching

- At the tipping point:
  - Sufficient first-order effects are known
  - It’s only a matter of combining them in a coherent and effective way
Bibliography


Temporal Coding


Excitatory Neurons


STDP


TNNs


Oscillatory Behavior (Network Synchronization)

Acknowledgements

Raquel Smith

Mario Nemirovsky, Cristobal Camarero, Ravi Nair, Joel Emer, Abhishek Bhattacharjee

Mikko Lipasti, Mark Hill, Margaret Martonosi, Michael Morgan

John Shen, Harideep Nair, Amy Zhang

Tim Sherwood, George Tzimpragos, Advait Madhavan

Shlomo Weiss, Ido Guy