

# **Reverse-Engineering the Brain: A Computer Architecture Grand Challenge**

J E Smith

*June 2018*

*Missoula, MT*

[jes.at.ece.wisc.edu@gmail.com](mailto:jes.at.ece.wisc.edu@gmail.com)

# The Grand Challenge

---

***“The most powerful computing machine of all is the human brain. Is it possible to design and implement an architecture that mimics the way the human brain works?”***

**-- Workshop report: Mary J. Irwin and John P. Shen, “Revitalizing computer architecture research,” Computing Research Association (2005).**

# Introduction

---

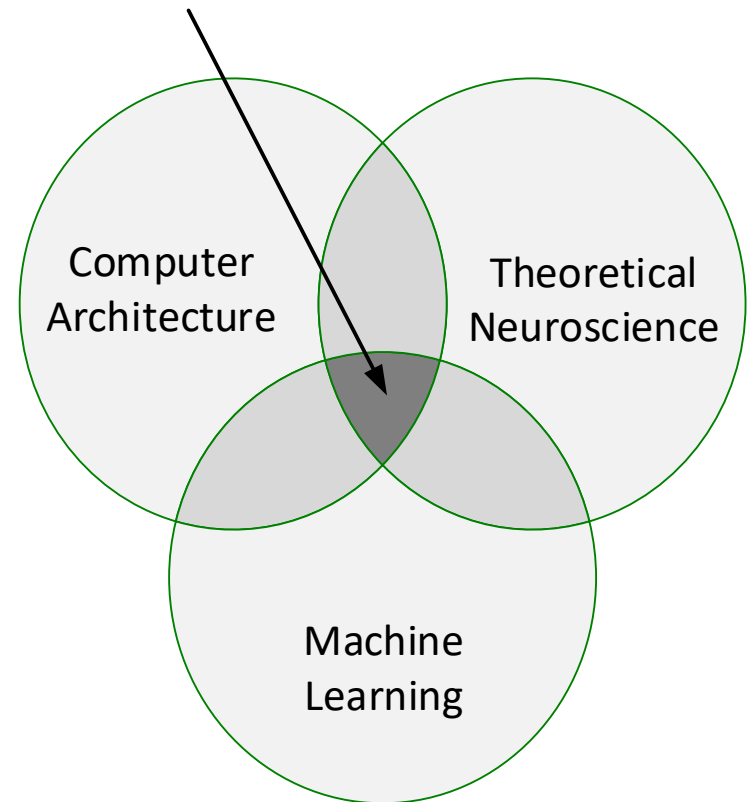
- ❑ The human brain is capable of:
  - Accurate sensory perception
  - High level reasoning and problem solving
  - Driving complex motor activity
- ❑ With some very impressive features:
  - Highly energy efficient
  - Flexible – supports a wide variety of cognitive functions
  - Learns dynamically, quickly, and simultaneously 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?
- ❑ After the tutorial, it is hoped that attendees will:
  - Understand the nature of the problem,
  - View it as a computer architecture research problem,
  - Have a firm foundation for initiating study of the problem,
  - Participate in a serious effort to tackle the grand challenge.

# Computer Architecture Perspective

---

- ❑ This research is at the intersection of *computer architecture, theoretical neuroscience, and machine learning*
  - It is of *major* importance to all three
  - And it requires knowledge of all three
- ❑ Most researchers are likely to be rooted in one of them
  - This will shape the perspective and direction of the research
- ❑ Our perspective (and biases) come from computer architecture
  - *We should tackle the problem from our perspective, using our background and strengths*

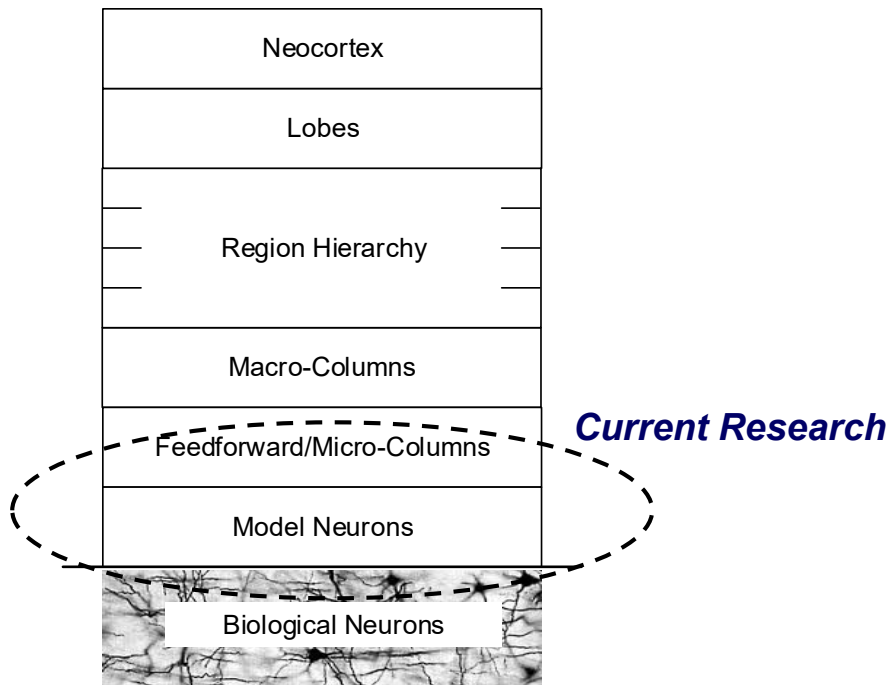
“Reverse-Engineering the Brain”



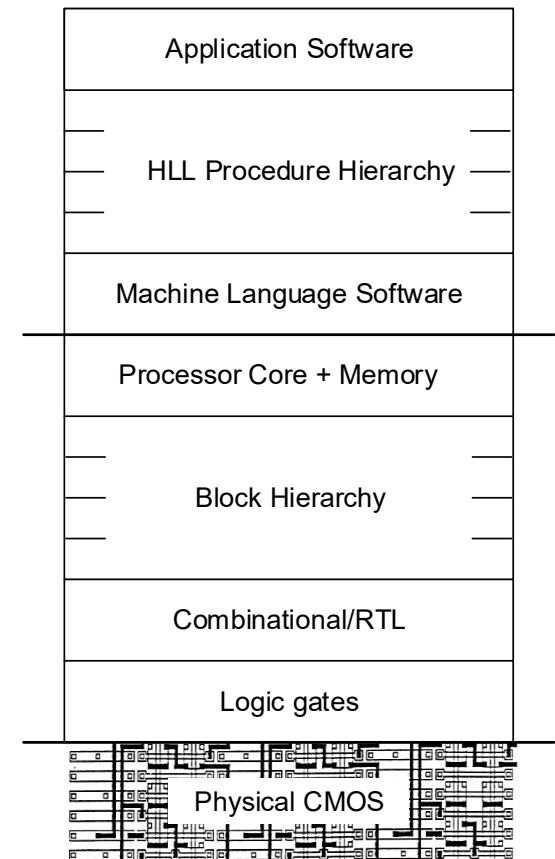
# What does “Reverse-Engineering the Brain” mean?

- “Reverse-abstracting the neocortex” is a better way to put it
  - The *neocortex* is the part of the brain we are interested in
  - *Reverse-abstracting* is discovering the *layers of abstraction*, bottom-up

**Neuro Architecture Stack**



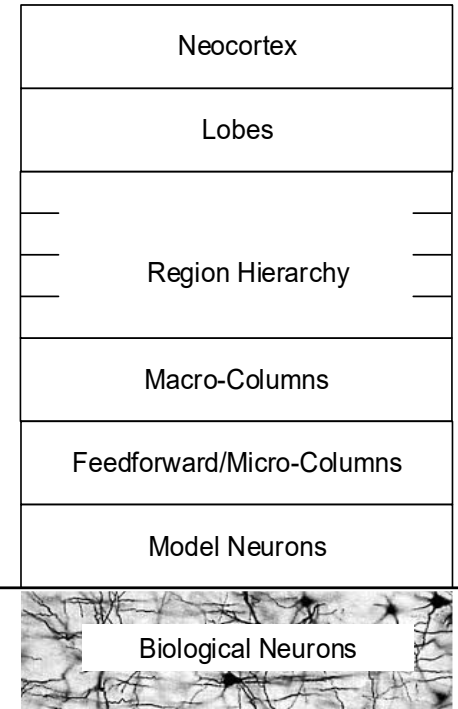
**Silicon Architecture Stack**



# Science v. Architecture

---

- ❑ Consider two implementations of the brain's computing paradigm:
  - Biological implementation that we each possess
  - Human-engineered, silicon-based implementation that we would like to achieve
- ❑ From the neuroscience perspective:
  - *Goal: Discover ever greater and more accurate information regarding biological processes*
- ❑ From a computer architecture perspective:
  - *Goal: Discover a new computational paradigm*
- ❑ Differences in goals may lead to different approaches



# Outline

---

- ❑ Introductory Remarks
- ❑ Research Focus: Milestone TNN
  - Classification and Clustering
  - Neural Network Taxonomy
- ❑ Biological Background
  - Overview
  - Neural Information Coding
  - Sensory Encoding
- ❑ Fundamental Hypotheses
  - Theory: Time as a Resource
- ❑ Low Level Abstractions
  - Excitatory Neuron Models
  - Modeling Inhibition
- ❑ Column Architecture
- ❑ Case Studies
  - Recent TNNs
- ❑ Space-Time Theory and Algebra
- ❑ Implementations
  - Indirect Implementations: Simulation
  - Direct Implementations: Neuromorphic Circuits
- ❑ Concluding Remarks

# **Research Focus: Milestone Network**

# Milestone Temporal Neural Network (TNN)

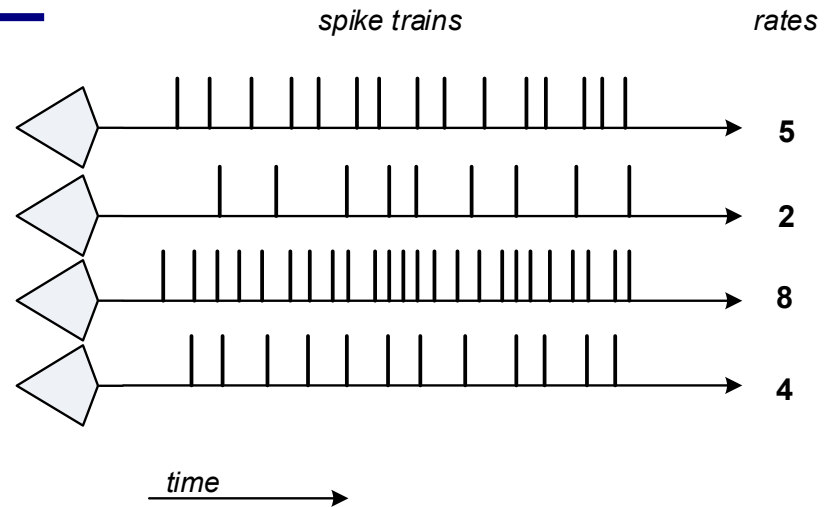
---

- ❑ We are attempting to solve a huge problem
- ❑ Reduce it to “bite-size” pieces
  - This is the first “bite”
- ❑ *Feedforward clustering networks* (unsupervised training)
  - Most current neuroscientific TNN projects have this target
- ❑ Sufficient for:
  - Demonstrating TNN capabilities, and
  - Exposing major differences wrt conventional machine learning methods

***First, some background***

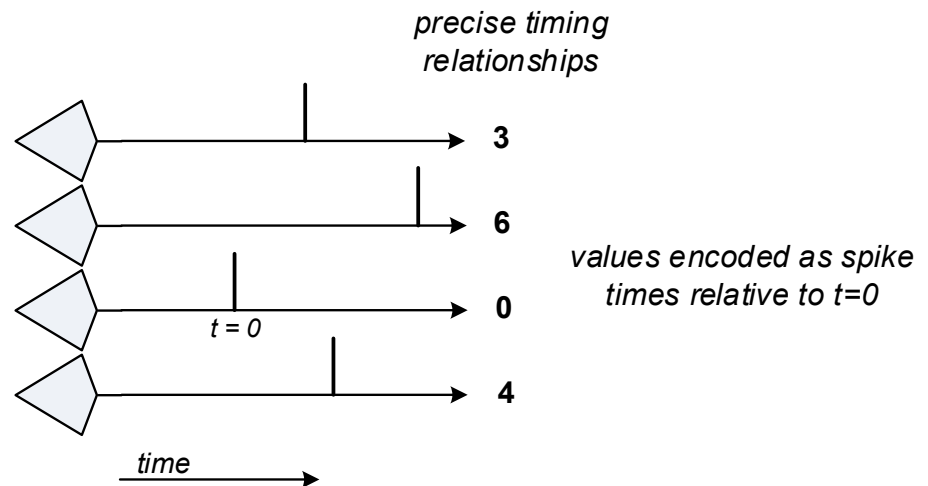
# Spike Communication

- ❑ Consider systems where information is communicated via transient events
  - e.g., voltage spikes
- ❑ Method 1: values are encoded as spike rates measured on individual lines
- ❑ Changing spikes on one line only effects values on that line
  
- ❑ Method 2: values are encoded as temporal relationships across parallel communication lines
- ❑ Changing spikes on one line may effect values on any/all of the others



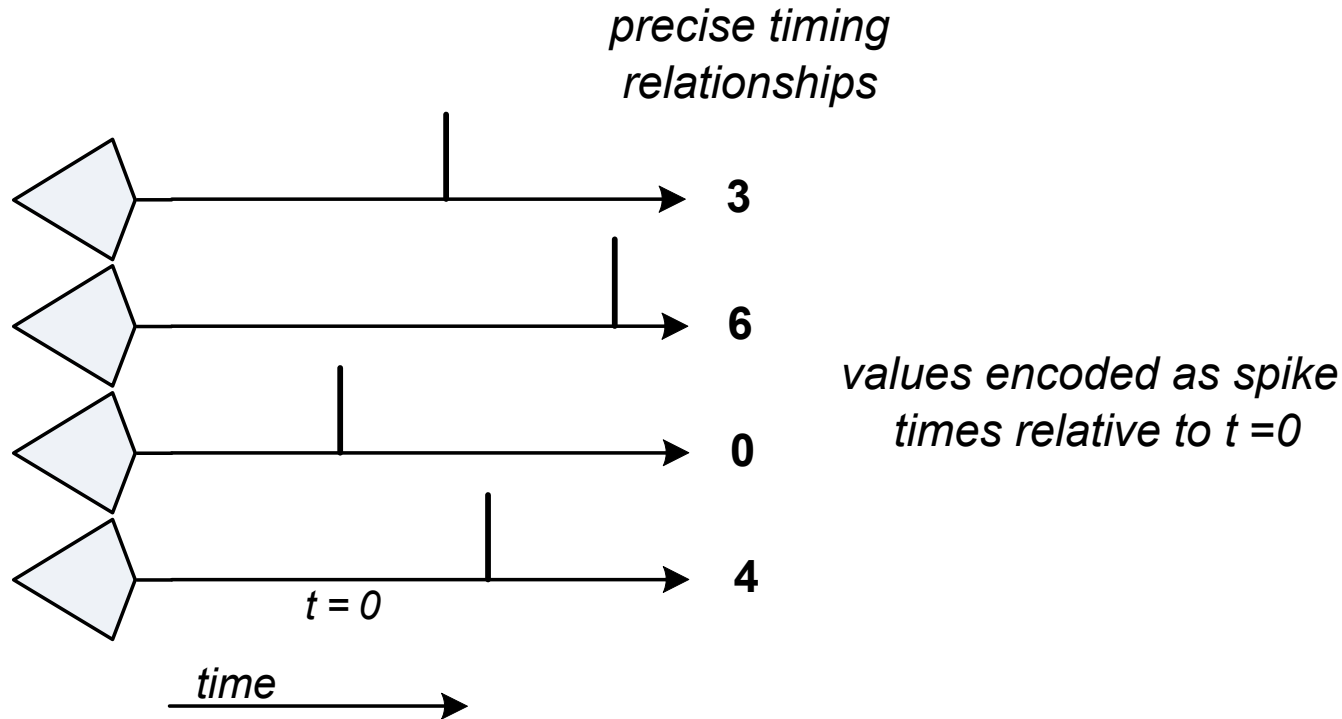
*The temporal communication method has significant, broad experimental support*

- The rate method does not.
- (Much) more later



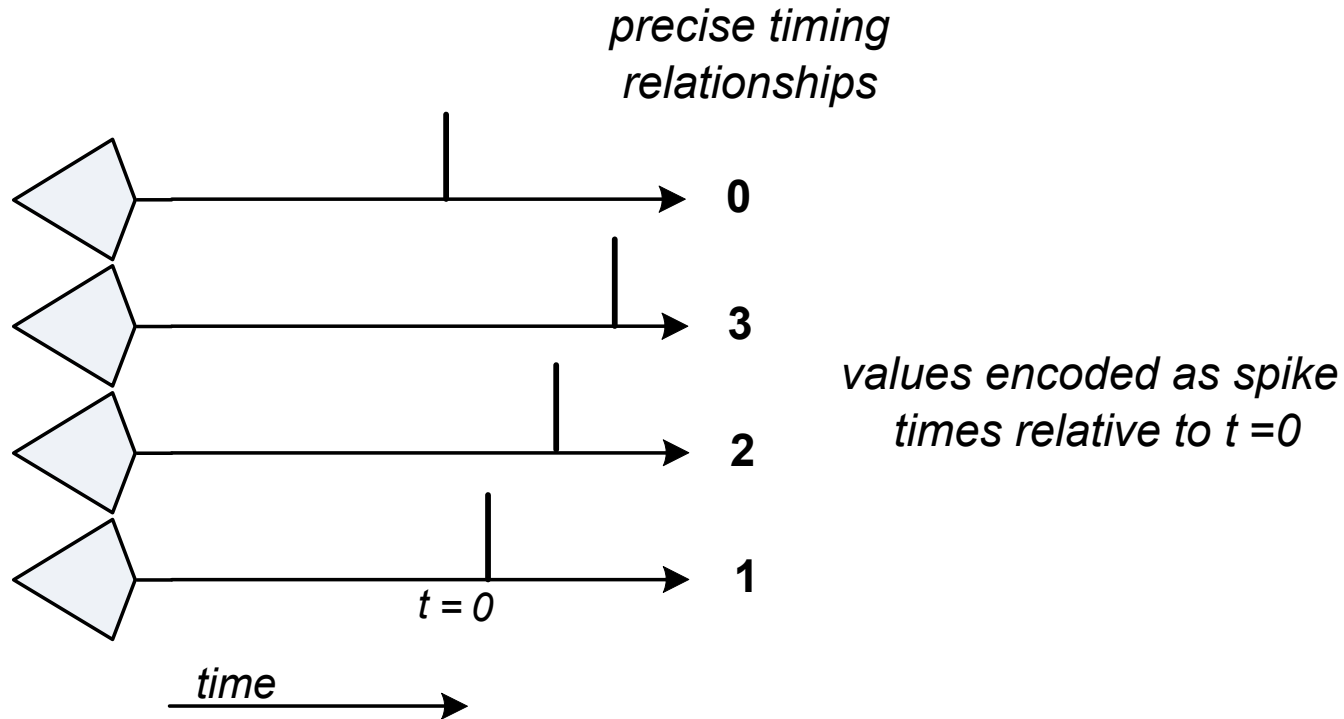
# Temporal Coding

- Use relative timing relationships across multiple parallel lines to encode information
- *Changing the spike time on a given line may affect values on any/all the other lines*



# Temporal Coding

- Use relative timing relationships across multiple parallel spikes to encode information
- *Changing the spike time on a given line may affect values on any/all the other lines*

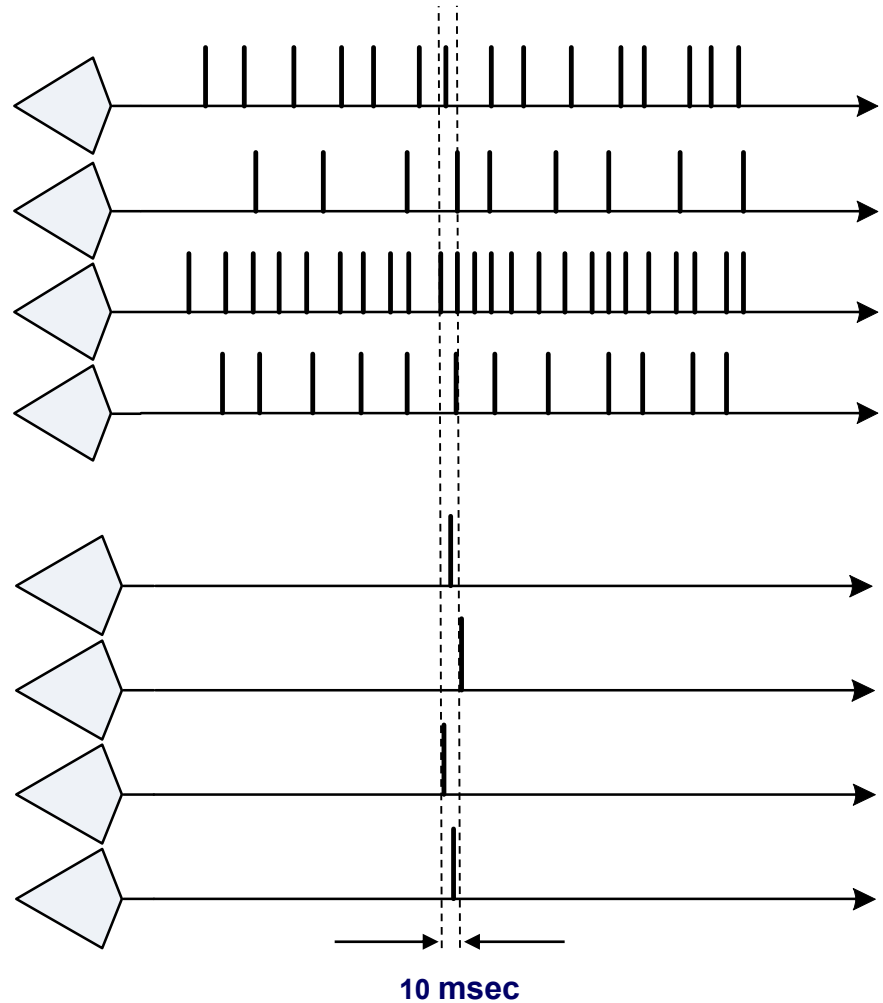


# Plot on Same (Plausible) Time Scale

---

Temporal method is

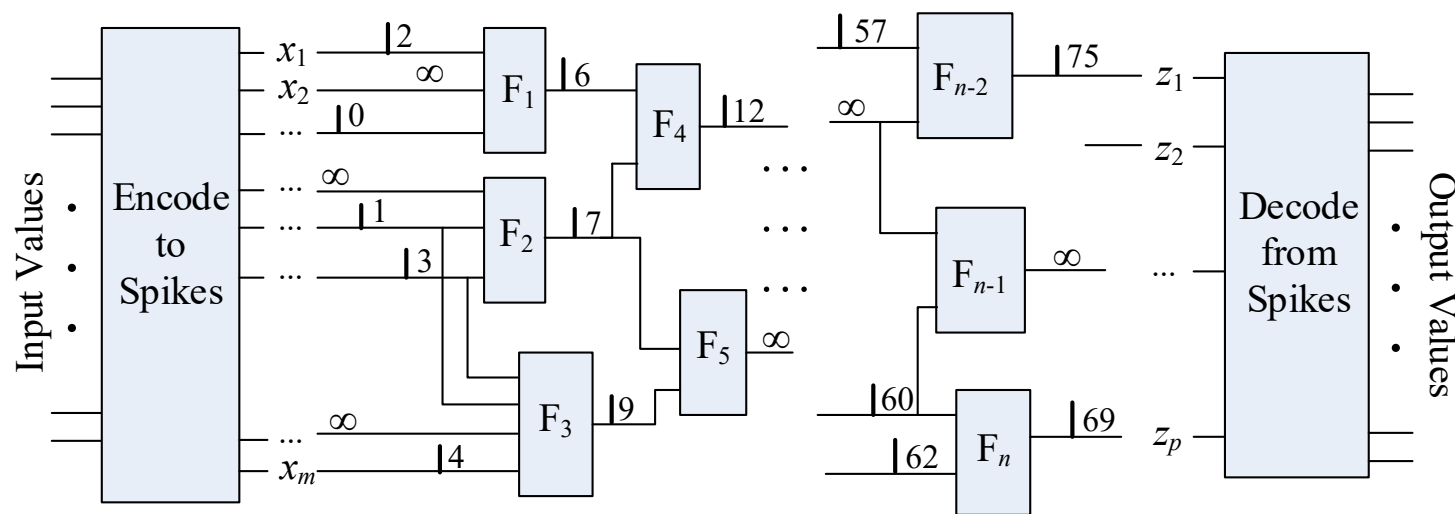
An order of magnitude faster  
An order of magnitude more  
efficient (#spikes)



# Spike Computation

## □ Temporal Neural Networks

- Networks of functional blocks whose input and output values are communicated via precise timing relationships (*expressed as spikes*)
- Feedforward flow (*without loss of computational generality*)
- Computation: a single wave of spikes passes from inputs to outputs



## □ A critical *first* step

- At the forefront of current theoretical neuroscience

# Functional Block Features

1) Total function w/ finite implementation

2) Asynchronous

- Begin computing when the first spike arrives

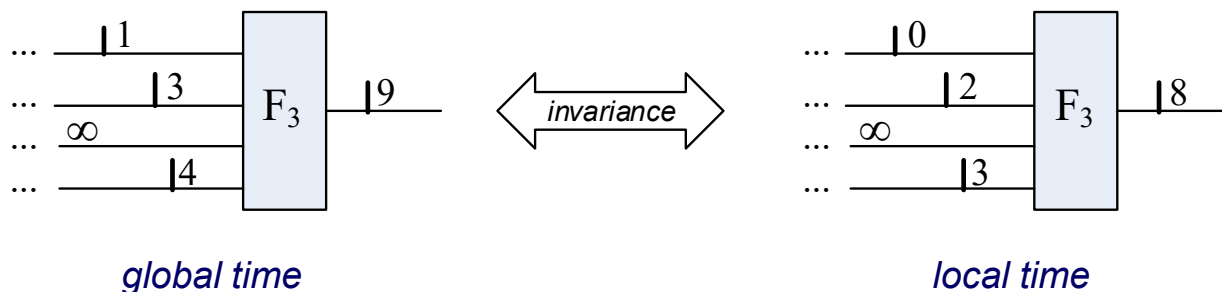
3) Causal

- The output spike time is independent of any later input spike times

4) Invariant

- If all the input spike times are increased by some constant then the output spike time increases by the same constant

⇒ Functional blocks interpret input values according to local time frames



# Compare with Conventional Machine Learning

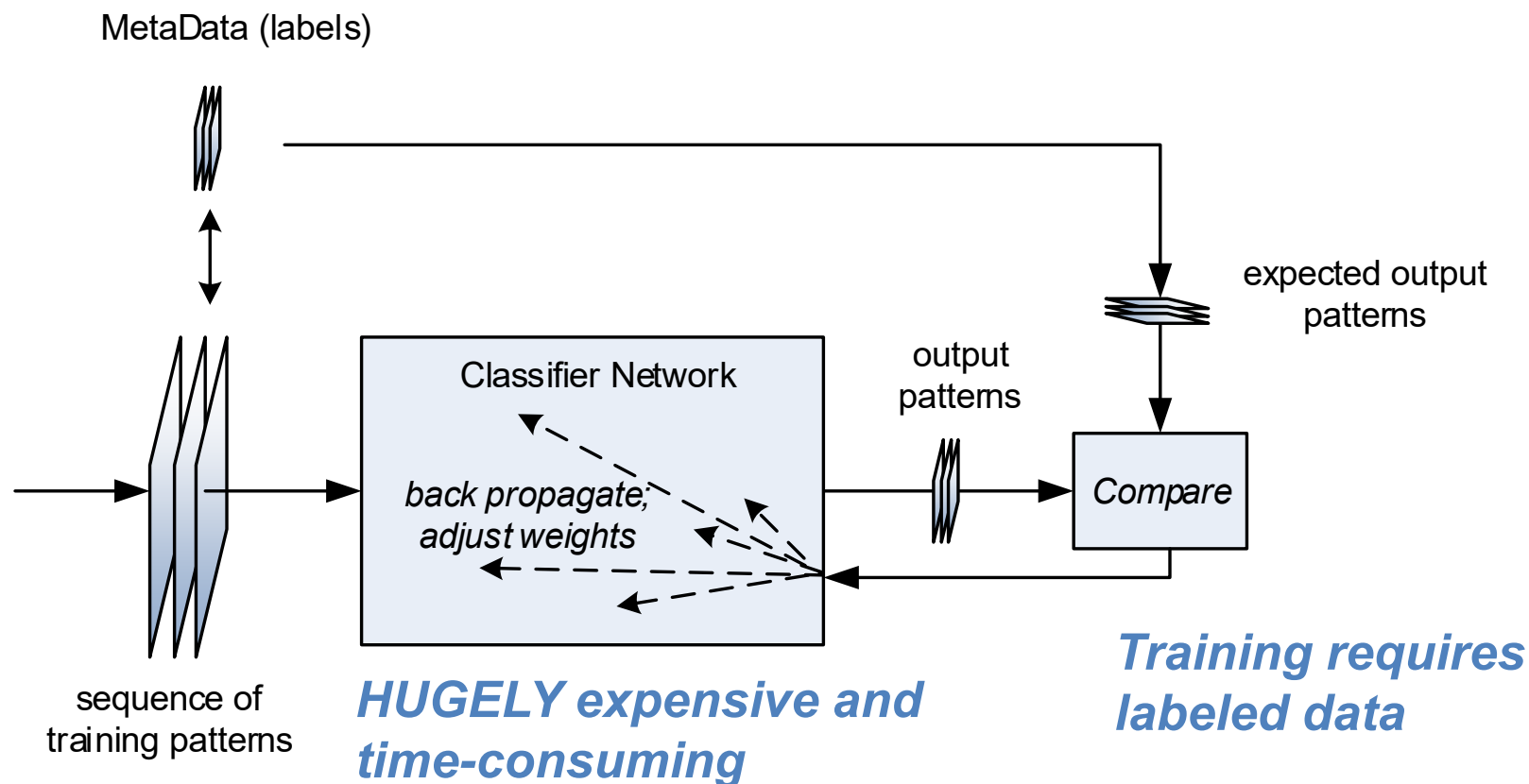
---

aka the *Bright Shiny Object*

- Three major processes:
  - Training (Learning)
  - Evaluation (Inference)
  - Re-Training

# Conventional Machine Learning: Training

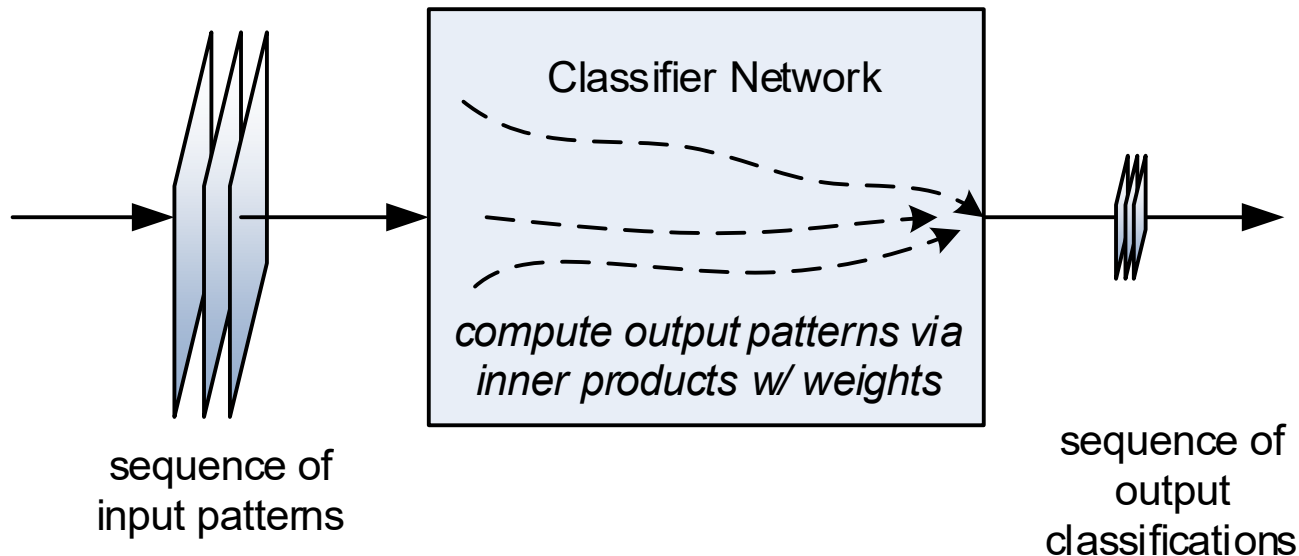
- Training process
  - Apply lots and lots of training inputs, over and over again



# Conventional Machine Learning: Evaluation (Inference)

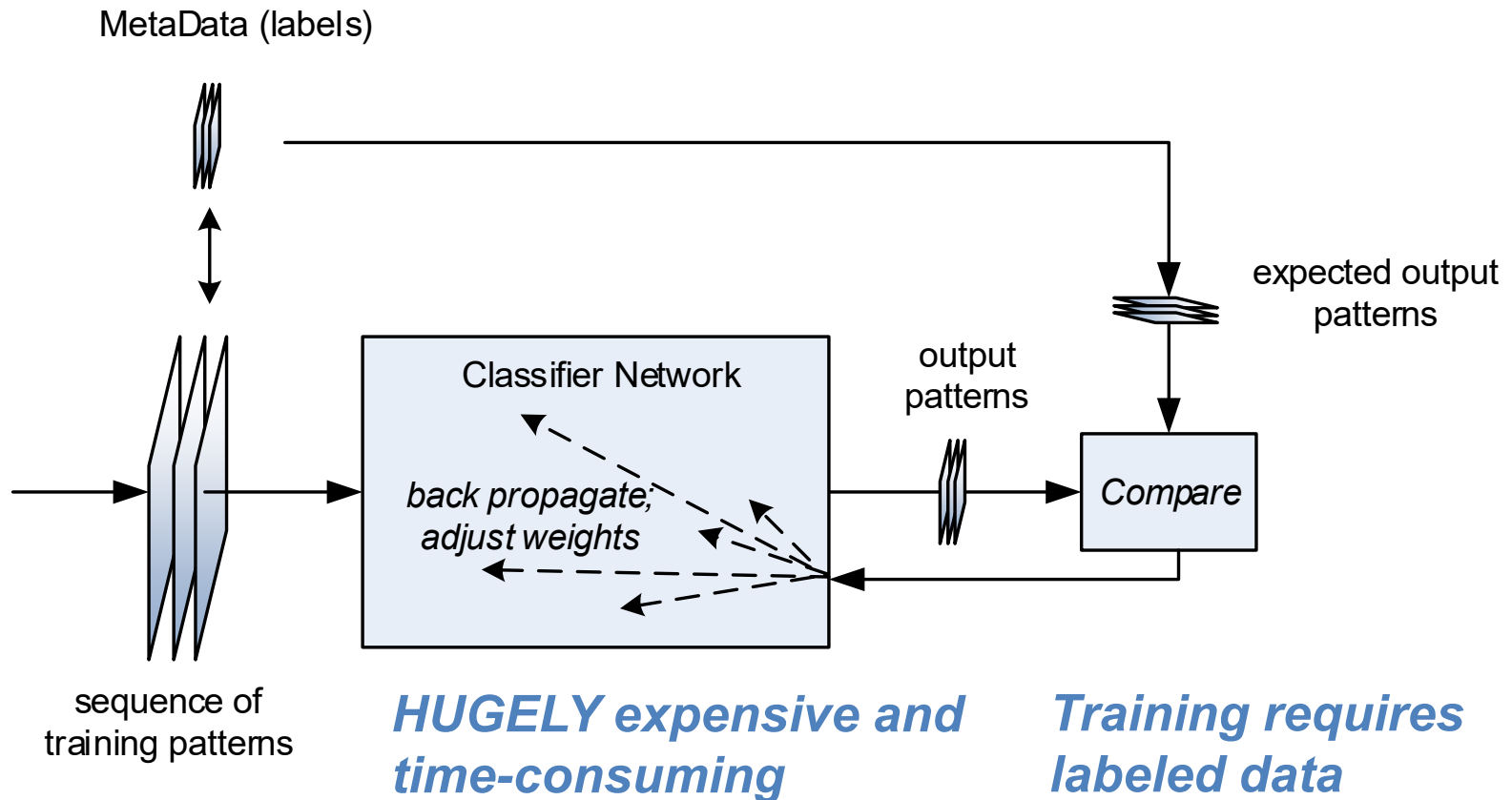
---

- Evaluation process:
  - Apply input patterns, and compute output classes via vector inner products (inputs  $\cdot$  weights).



# Conventional Machine Learning: Re-Training

- If the input patterns change over time, perhaps abruptly, the training process must be re-done from the beginning
  - Apply lots and lots of training inputs, over and over again



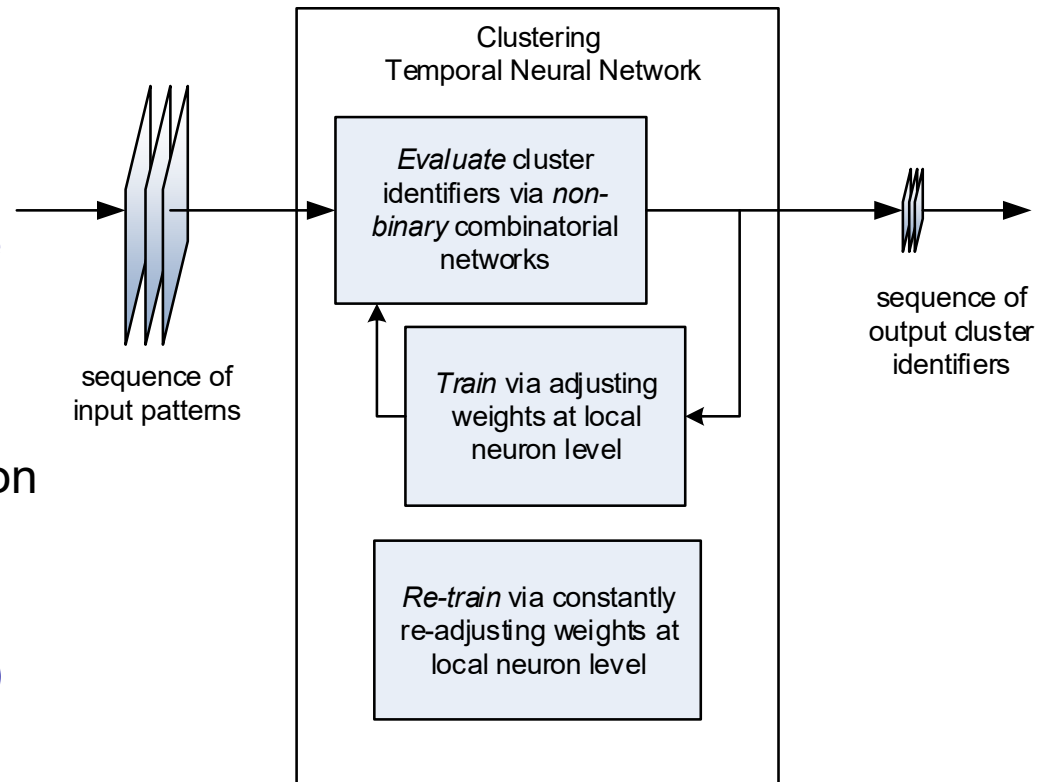
# Milestone Temporal Neural Network

---

- Perform *unsupervised clustering*
  - Not *supervised classification*
- Informally: A *cluster* is a group of *similar* input patterns
  - It's a grouping of input patterns based on *implementation-defined* intrinsic similarities
- A *clustering neural network* maps input patterns into cluster identifiers
  - Note: some outlier patterns *may* belong to *no* cluster

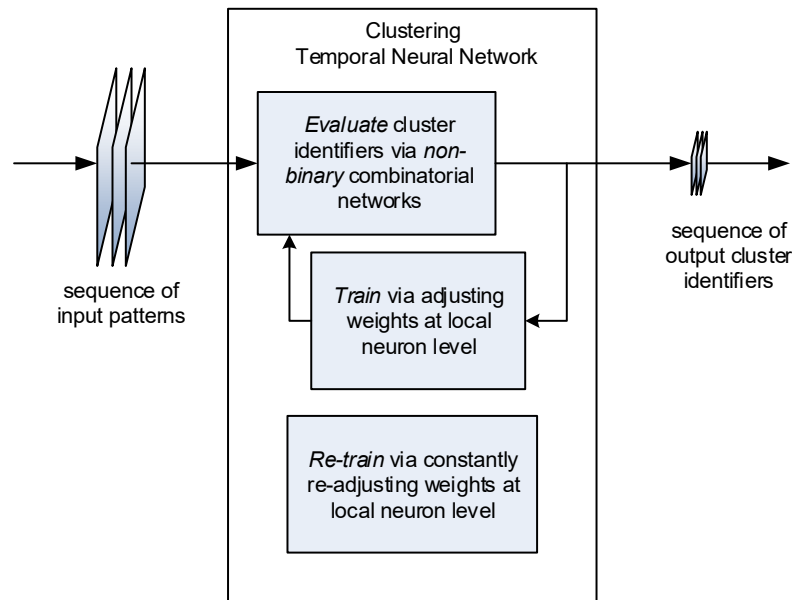
# Milestone Temporal Neural Network

- Each cluster has an associated *cluster identifier*
  - A concise output pattern that identifies a particular cluster
  - If there are  $M$  clusters, then there are  $M$  cluster identifiers.
- The exact mapping of cluster identifiers to clusters is determined by the implementation
  - A direct by-product of unsupervised training  
(*there are no meta-data inputs*)



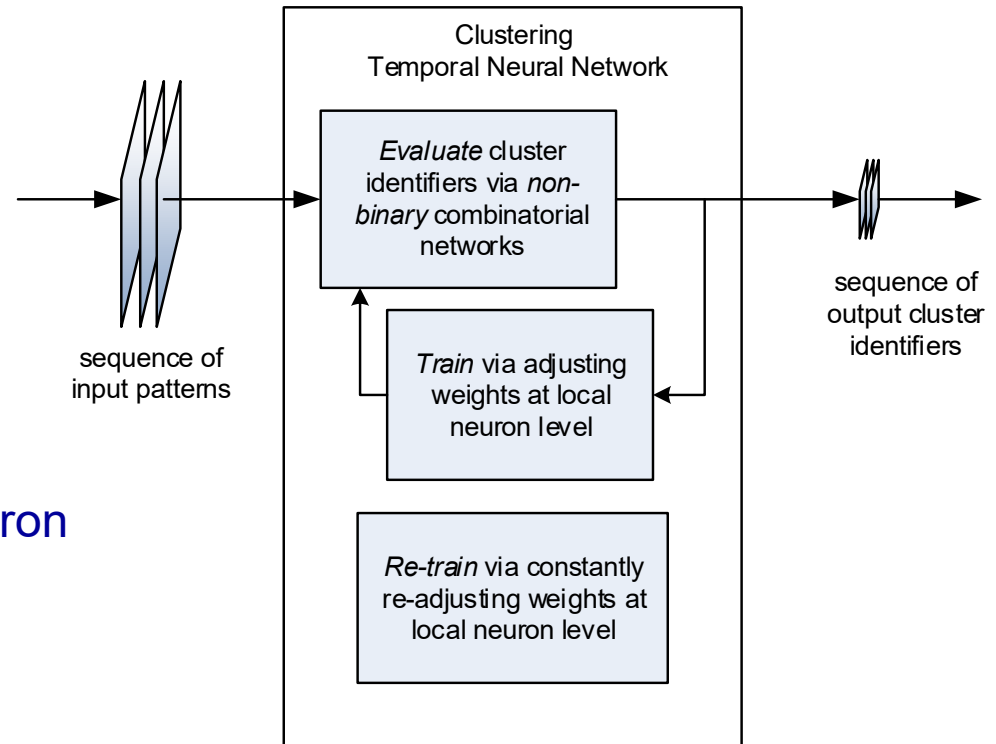
# Milestone Temporal Neural Network

- ❑ For outputs to be directly understood by humans, we must map cluster identifiers into known labels (decoding)
  - Via construction of a trivial classifier at the output
- ❑ The “similarity” metric is implied by the implementation
  - Not an *a priori*, designer-supplied metric
- ❑ Network designer can control the maximum number of clusters



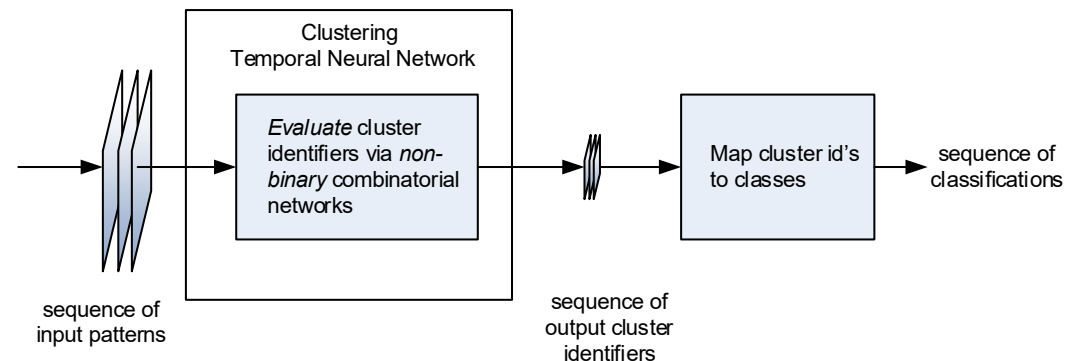
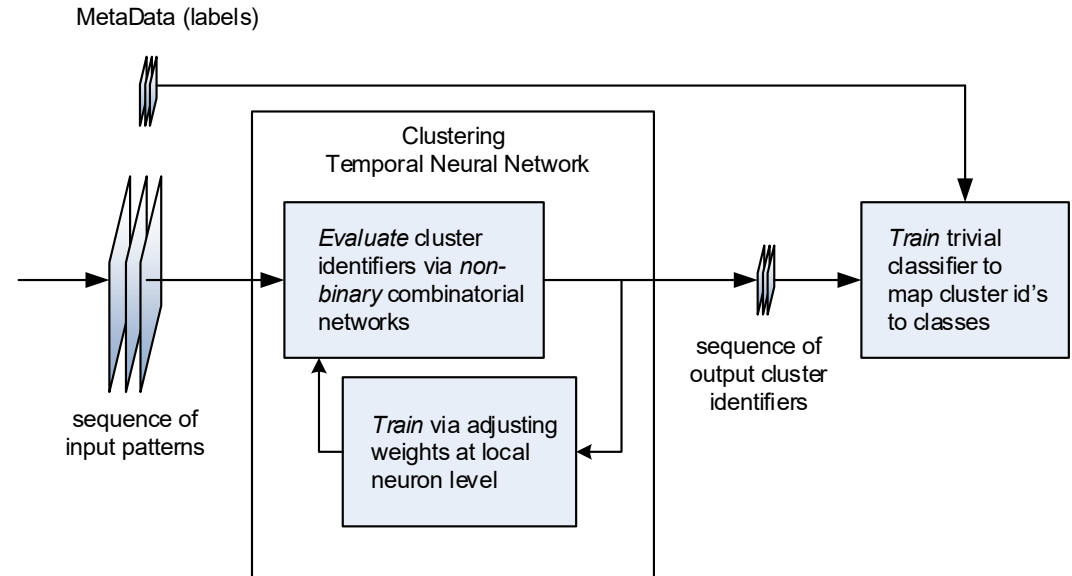
# Summary: Milestone Temporal Neural Network

- ❑ Everything is done by the same network concurrently
  - Training
  - Evaluation
  - Re-training
- ❑ Training
  - Unsupervised
  - Always active
  - Performed locally at each neuron (synapse, actually)
- ❑ Evaluation
  - Concurrent with training
  - Tightly integrated with training



# Temporal Neural Network as a Classifier

- ❑ First: train unsupervised via sequence of input patterns
  - Synaptic weight training is localized and efficient
- ❑ Turn off TNN training
- ❑ Train output decoder
  - Maps cluster id's to classes
  - Trivial classifier, e.g., 1-to-1 mapping



- ❑ Evaluate with training disabled

# Milestone Architecture Objectives

---

- ❑ Function:
  - Encodes information as transient temporal events (e.g., voltage spikes)
  - Groups input patterns into clusters based on similarity
  - Input stream of *patterns* produces output stream of very simple *cluster identifiers*
  - Makes local synaptic weight adjustments concurrent with operation: changes in input patterns cause changes to clustering
- ❑ Implementation Features:
  - Simple computational elements (neurons) that operate very quickly and energy efficiently
  - Implementable with conventional CMOS circuitry

***Demonstrate brain-like capabilities and efficiencies with silicon technology***

# A Good Starting Point: MNIST Benchmark

---

## □ A Goldilocks Benchmark

- Not too large, not too small
- Not too hard, not too easy
- *Just right*

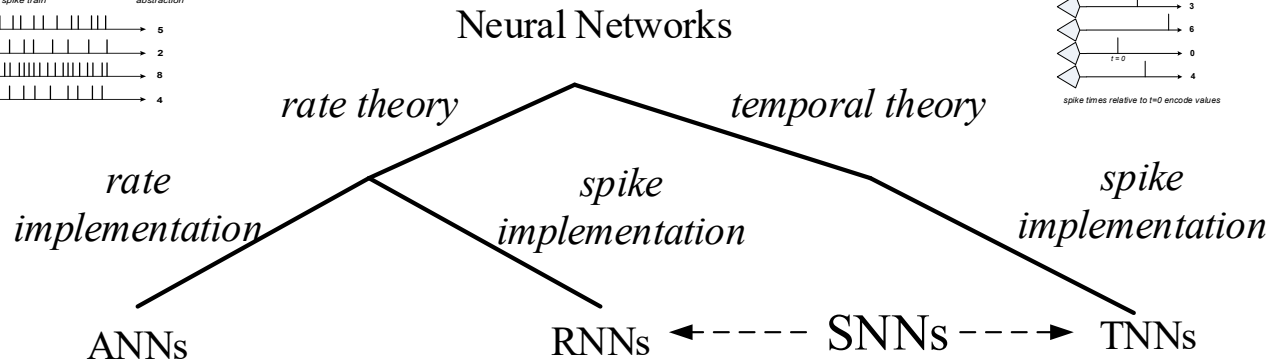
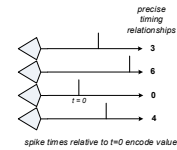
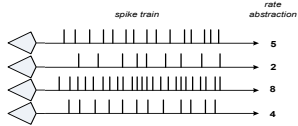
<http://yann.lecun.com/exdb/mnist/>

## □ What it is:

- Tens of thousands of 28 x 28 grayscale images of written numerals 0-9
- *Originally* hand-written zip codes
- Labeled: 60K training patterns; 10K evaluation patterns



# Neural Network Taxonomy



Virtually every machine learning method in use today – deep convolutional nets, RBMs, etc.

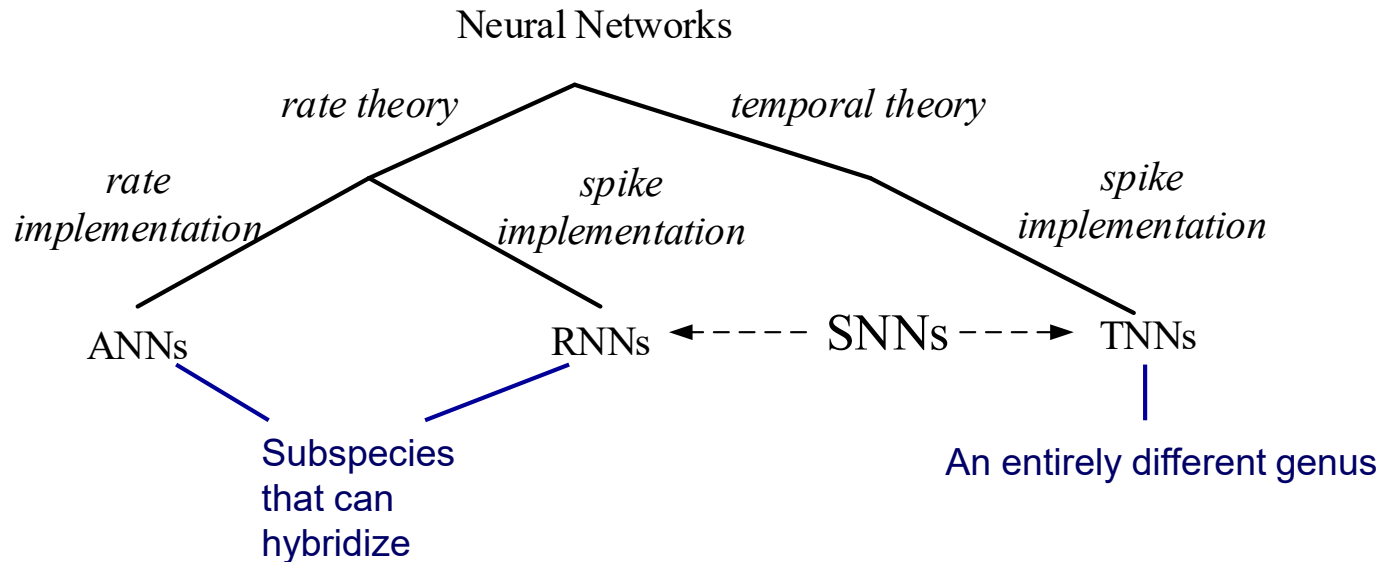
Eliasmith et al. 2012 (Spaun)  
 Recent ETH papers  
 Brader et al. 2007  
 Diehl and Cook, 2015  
 O'Connor et al. 2015  
 Beyeler et al. 2013 (UC Irvine, GPU)  
 Querlioz et al., 2013 (memristors)

Maass 1999  
 Thorpe and Imbert 1989  
 Masquelier and Thorpe 2007  
 Kheradpisheh, Ganjtabesh, Masquelier 2016  
 Natschläger and Ruf 1998 (RBF neurons)  
 Probst, Maass, Markram, Gewaltig 2012 (Liquid State Machines)

Distinguish *computing model* from *SNN implementation*

# Neural Network Taxonomy

---



*RNNs and TNNs are two very different models, both with SNN implementations, and they should not be conflated*

# Example: Conflating SNNs

---

[Yann LeCun](#)

[August 7, 2014](#) ▪

My comments on the IBM TrueNorth neural net chip. IBM has an article in Science about their TrueNorth neural net chip....

...

Now, what wrong with TrueNorth? My main criticism is that TrueNorth implements networks of integrate-and-fire spiking neurons. This type of neural net that has never been shown to yield accuracy anywhere close to state of the art on any task of interest (like, say recognizing objects from the ImageNet dataset). Spiking neurons have binary outputs (like neurons in the brain). The advantage of spiking neurons is that you don't need multipliers (since the neuron states are binary). But to get good results on a task like ImageNet you need about 8 bit of precision on the neuron states. To get this kind of precision with spiking neurons requires to wait multiple cycles so the spikes "average out". This slows down the overall computation.

...

TrueNorth is a platform upon which a wide variety spiking neuron models (both RNNs and TNNs) can be studied.

*An implementation platform and a computing model should not be conflated.*

This criticism assumes the RNN (rate) model and correctly points out a major flaw.

HOWEVER, this criticism does not apply to TNNs

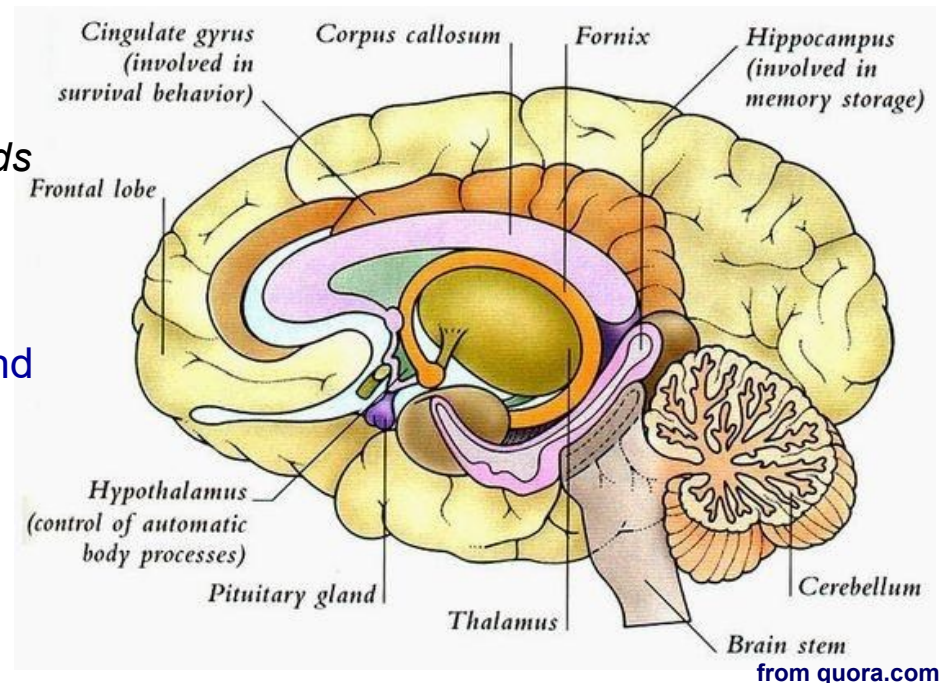
*The two SNN-based models should not be conflated*

# **Biological Background**

## ***Overview***

# The Brain

- ❑ Inner, “older” parts of the brain are responsible for automatic processes, basic survival processes, emotions
  - Not of interest for this line of research
- ❑ Neocortex
  - The “new shell” that surrounds the older brain
  - Performs the high level skills we are most interested in
    - sensory perception*
    - cognition*
    - intellectual reasoning*
    - generating high level motor commands*
  - *Our primary interest is the neocortex*
- ❑ Hippocampus
  - Responsible for memory formation and sense of place
- ❑ Cerebellum
  - Responsible for motor control coordination



# Neocortex

---

- Thin sheet of neurons
  - 2 to 3 mm thick
  - Area of about 2500 cm<sup>2</sup>
    - Folds increase surface area
  - Approx 100 billion total neurons (human)
- Hierarchical Physical Structure
  - Lobes
  - Regions
  - Subregions, etc.
  - Macro-Columns
  - Micro-Columns
  - Neurons

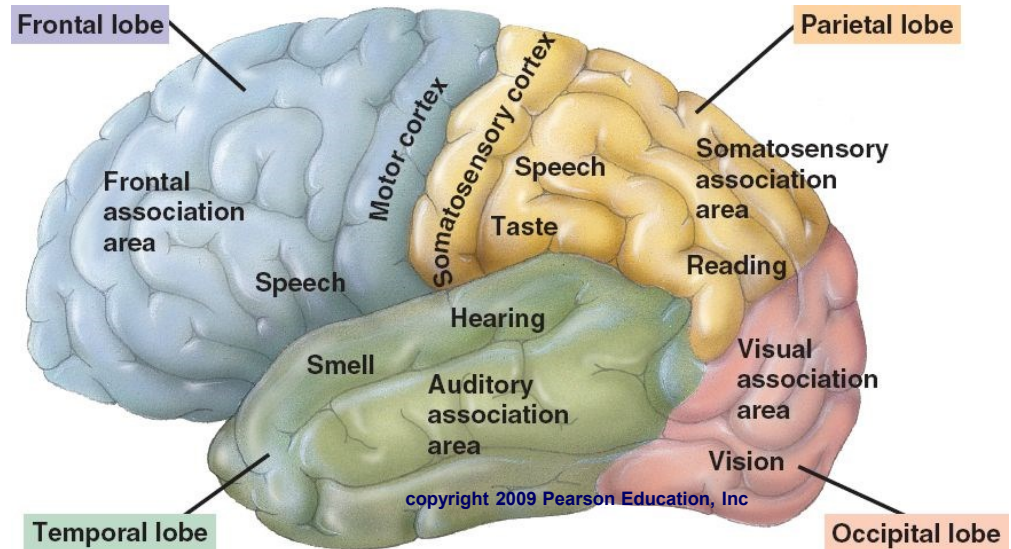


illustration of major regions and lobes

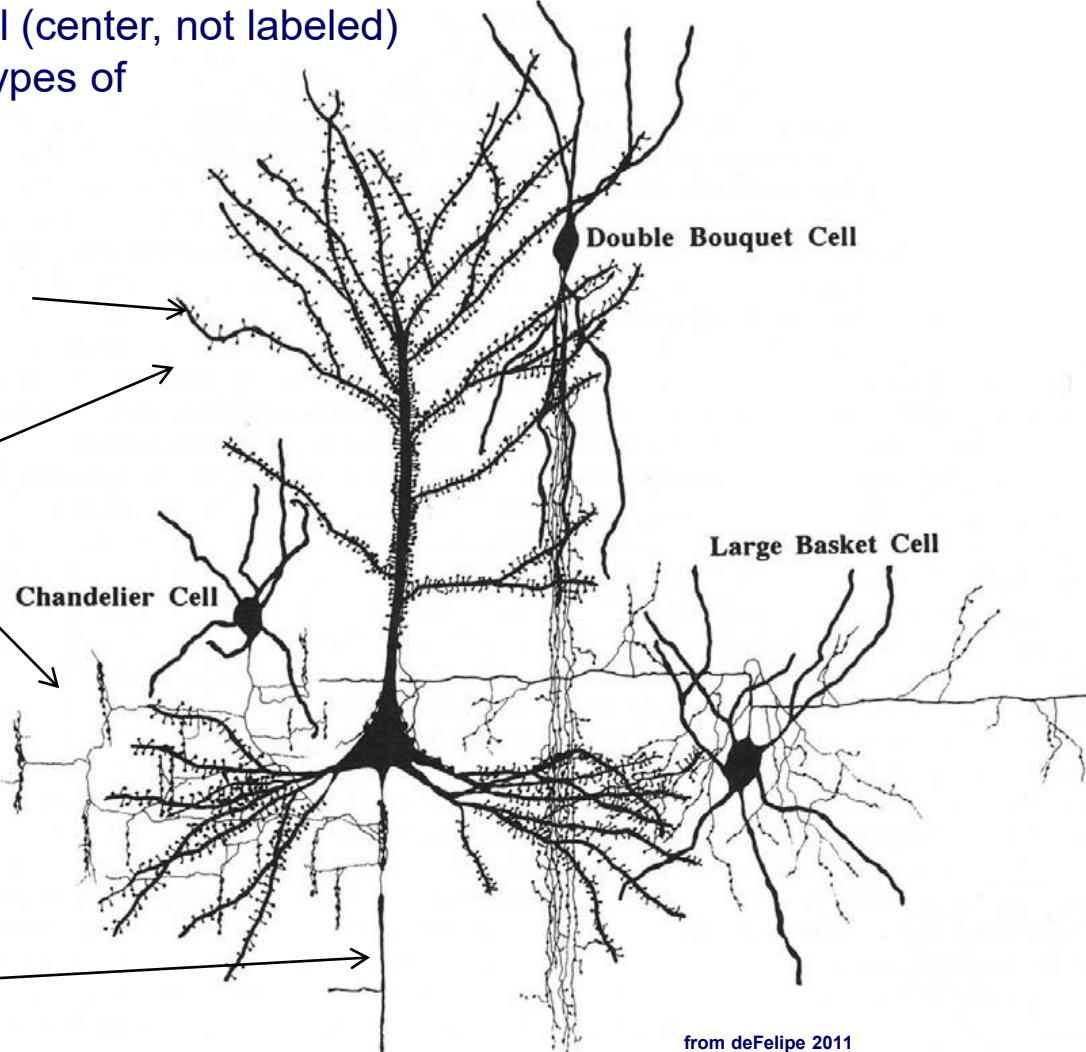
# Biological Neurons

Excitatory pyramid cell (center, not labeled)  
surrounded by three types of  
Inhibitory cells

tiny dots are synapses  
(connection points)

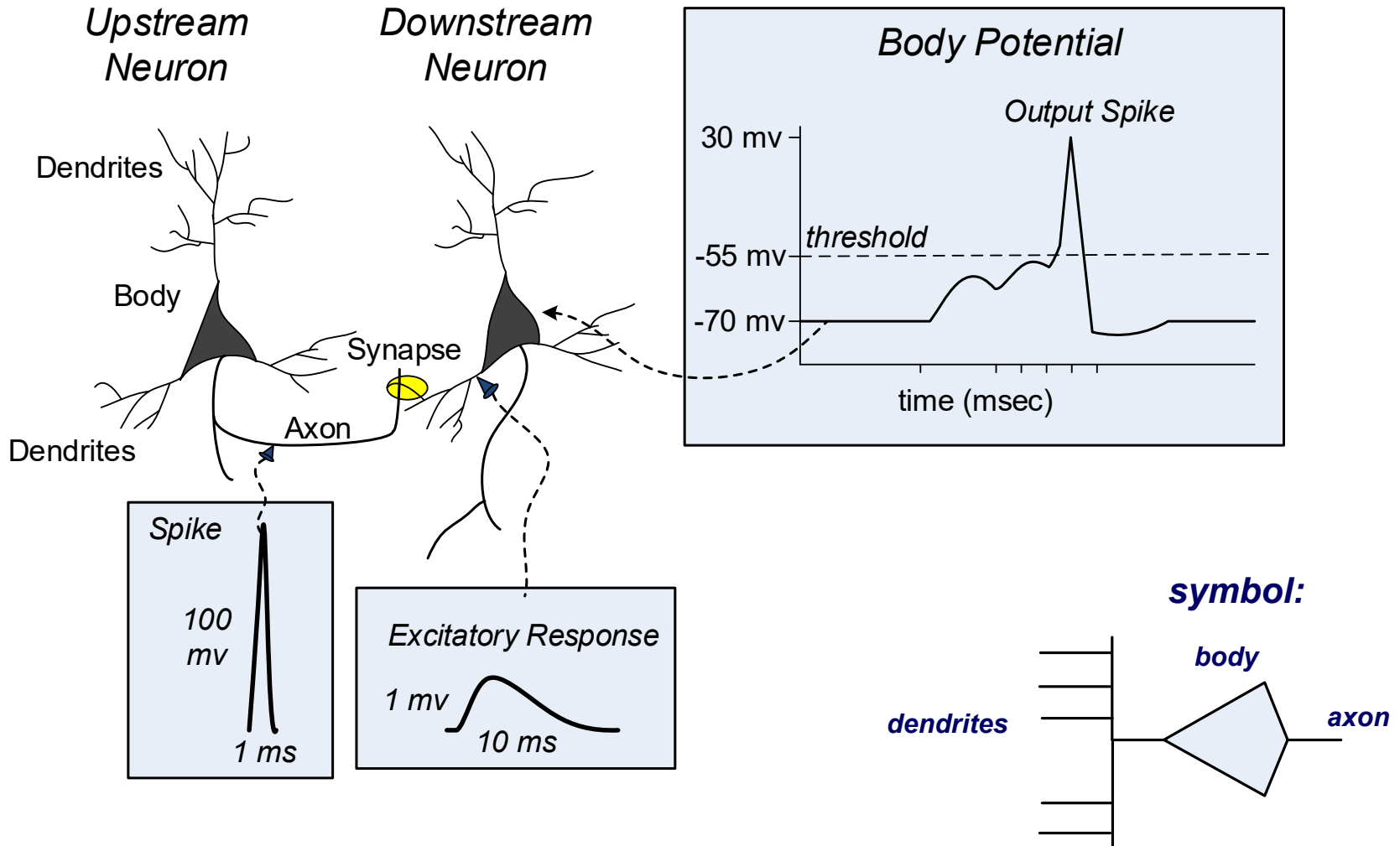
Dendrites (Inputs)

Axon (Output)



from deFelipe 2011

# Neuron Operation



# Excitatory Neurons

---

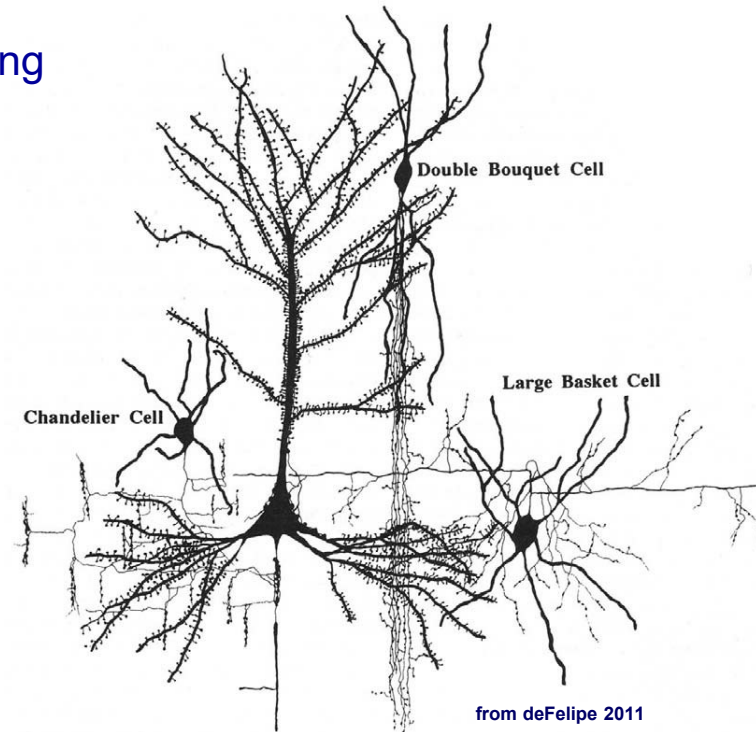
- ❑ About 80% of neurons are excitatory
- ❑ Most are *pyramidal* cells
  - Axon reaches both near and far
  - Capable of “precise” temporal operation
- ❑ Approximately 10K synapses per neuron
  - Active connected pairs are far fewer than physical connections suggest
    - 90% or more are “silent” at any given time
  - Multiple synapses per connected neuron pair



from Perin et al. 2011

# Inhibitory Neurons

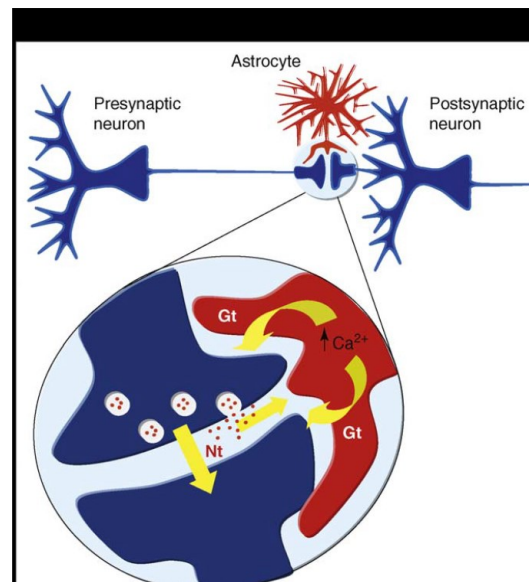
- ❑ Inhibitory response functions have the opposite polarity of excitatory response functions
  - *Reduce* membrane potential and inhibit spike firing
- ❑ Also called “interneurons”
  - Many types – most with exotic names
  - Some are fast-spiking (electrical synapses)
  - Others have slow response functions
- ❑ Less precise than excitatory neurons
- ❑ Act *en masse* on a local volume of neurons
  - A “blanket” of inhibition
- ❑ Perform a number of tasks related to
  - Maintaining stability
  - Synchronization (via oscillations)
  - *Information filtering* – emphasized in this tutorial



Refs: [Markram et al. 2004] [Karnani et al. 2014]

# Glial Cells

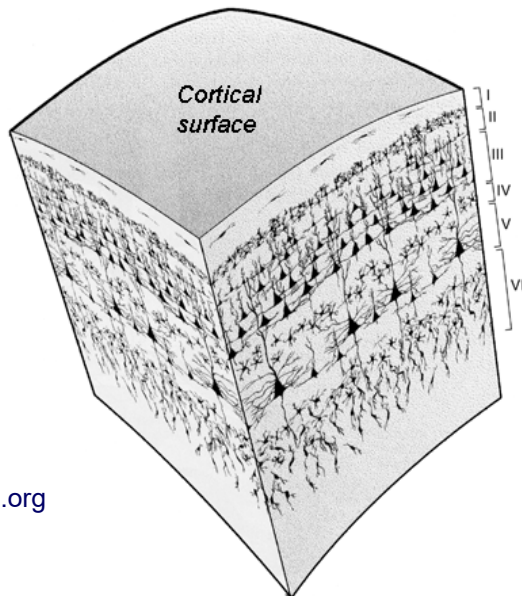
- ❑ Outnumber neurons 4:1 in neocortex
- ❑ *Astrocytes* may provide a chemical stabilization mechanism
  - Maintain ion balance, for example, in the neurons' ambient surroundings
- ❑ BUT could also influence computation in a more direct way
  - May play a role in synaptic plasticity (“tripartite” synapses)



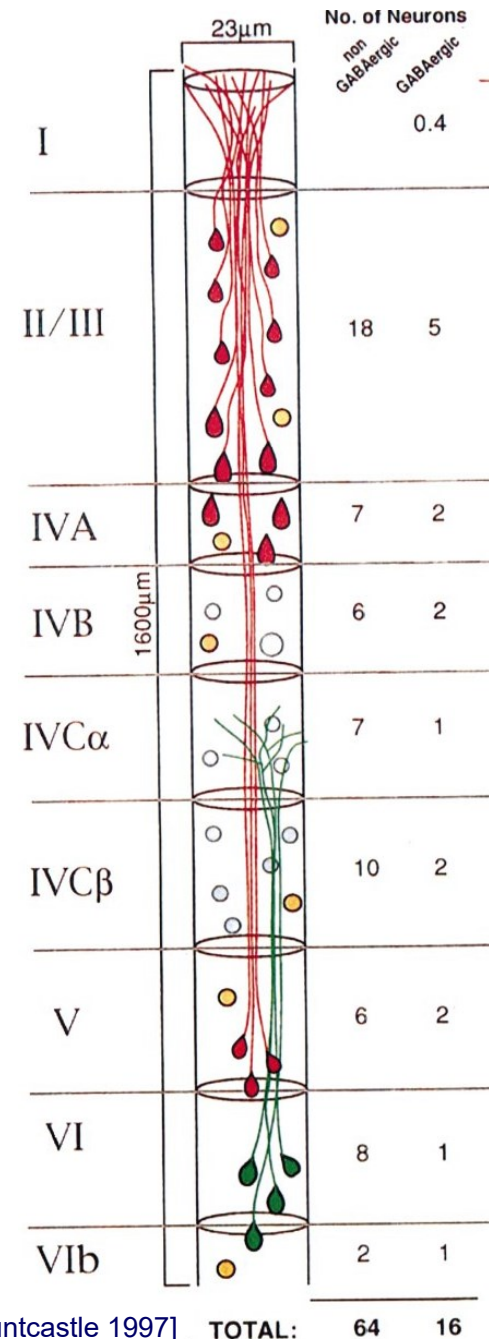
[Perea et al. 2009]

# Columnar Structure

- ❑ Neurons are structurally organized as *columns* between the top and bottom neocortical surfaces
- ❑ Columns are composed of layers
  - Nominally 6, but it depends a lot on who is counting
- ❑ Micro-Columns (Mini-Columns)
  - O(100 neurons) -- 80 excitatory; 20 inhibitory
  - 64 excitatory and 16 inhibitory in the famous drawing at right



from <https://neuwritesd.org>

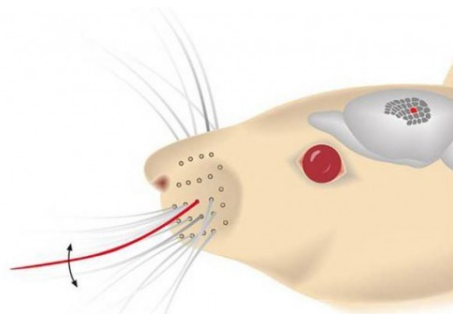


[Mountcastle 1997] TOTAL: 64 16

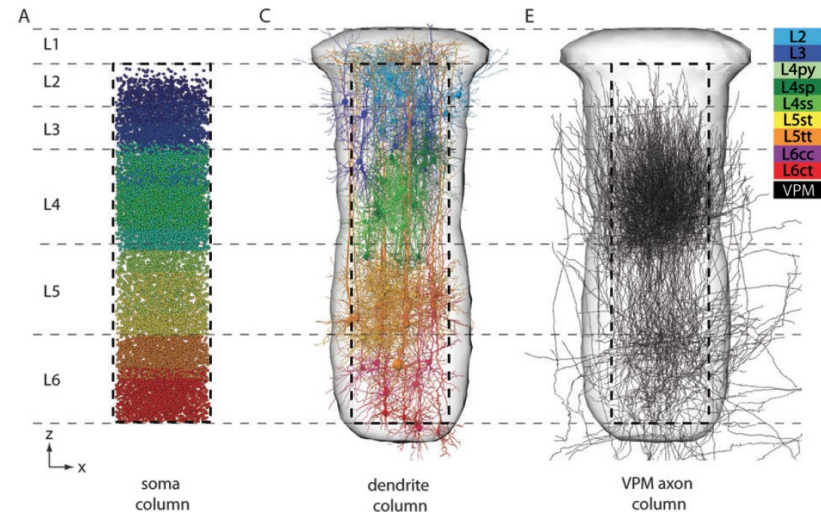
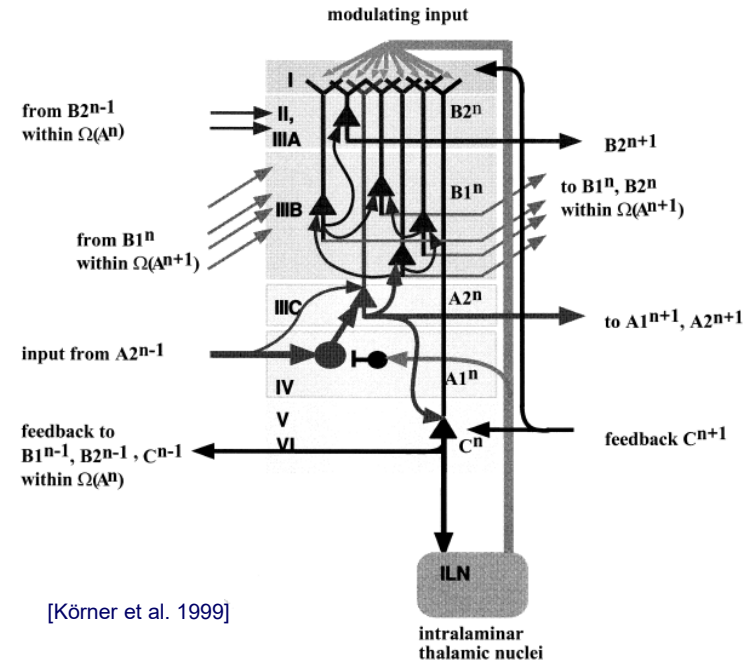
# Columnar Structure

- Macro-Columns
  - $O(100s)$  micro-columns per macro-column
  - In sensory systems, all micro-columns in a macro-column map to same *receptive field (RF)*
  - An RF may be a patch of retina or skin, or a rat's whisker
- Some theorists construct architectures using the biological columnar structure as a guide
  - Hawkins (On Intelligence) takes this approach
  - *This approach is not covered in this tutorial*

Refs: [Hawkins and Blakeslee 2007] [Rinkus 2010]



from: <http://www.kurzweilai.net>



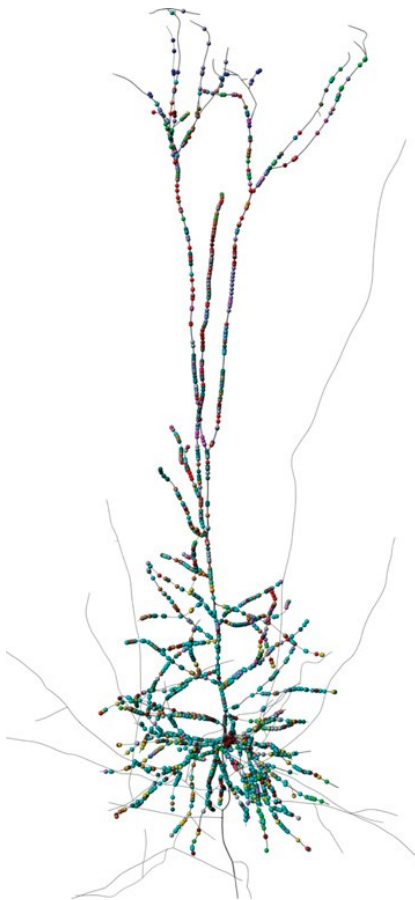
[Oberlander et al. 2012]

# Summary

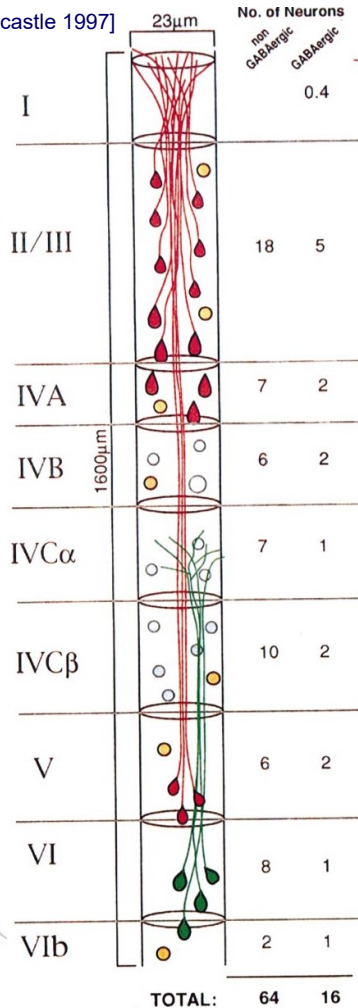
from Ramon y Cajal  
(via wikipedia)

[Hill et al. 2012]

[Mountcastle 1997]



Neuron

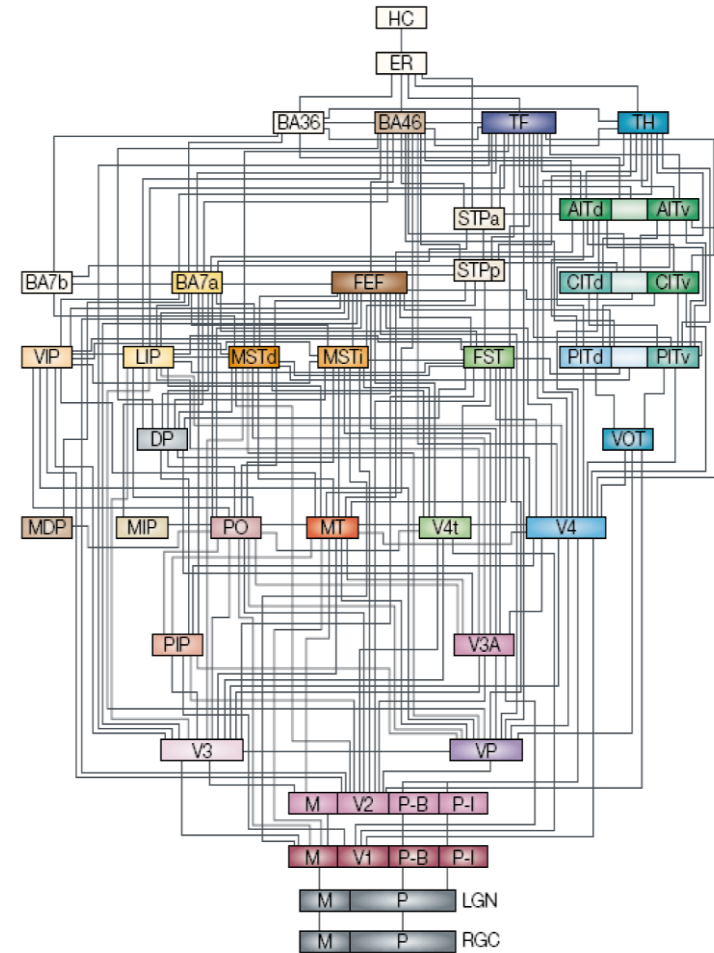


Micro-Column  
*O(100)* neurons



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

[Felleman and Van Essen 1991]



# Uniformity Hypothesis

- Hypothesis: The structure and operation of the neocortex is uniform
  - Across species
  - Across neocortical regions within a species
- “Uniform” is a subjective term
  - It depends on the user’s perspective
- From our perspective: *All cognitive processing follows the same paradigm(s) as sensory processing*
  - Why important? Most *in vivo* experimental data is for sensory processing,
  - Across a number of species
- So, by accepting uniformity, there is lots of useful experimental data regarding the basic paradigm(s) of cognitive processing

Table 5. Number of synapses per neuron in the human, rat and mouse cerebral cortex. Values were obtained by dividing the density of synapses by the density of neurons. Uncharacterized synapses were included in the asymmetrical and symmetrical types, according to the frequency of both types of synapses in each layer.

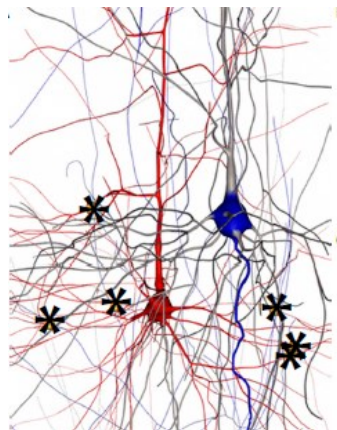
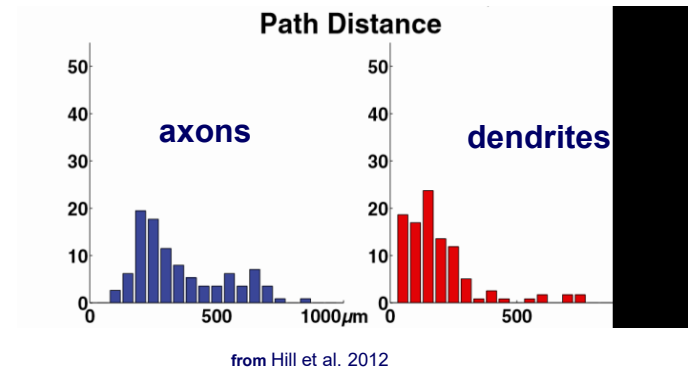
Layer	No. AS per neuron	No. SS per neuron	No. all synapses per neuron
<i>Human</i>			
I	83883	19188	103071
II	14886	2007	16894
IIIa	32573	4436	37009
IIIb	50266	6951	57217
IV	14379	1806	16186
V	26806	3561	30367
VI	25482	2811	28293
I–VI	26096	3711	29807
<i>Rat</i>			
I	286409	76766	363175
II–III	15992	1831	17823
IV	10493	969	11462
Va	19484	2428	21912
Vb	22588	2734	25322
VI	12545	756	13301
I–VI	16127	1891	18018
<i>Mouse</i>			
I	143438	26593	170031
II–III	15647	2086	17733
IV	14952	4524	19476
V	23068	4444	27512
VI	15561	3093	18654
I–VI	17595	3538	21133

AS, asymmetrical synaptic profiles; SS, symmetrical synaptic profiles.

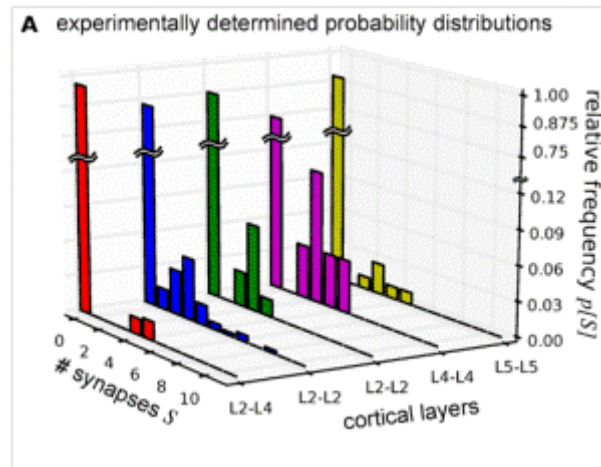
Refs: [Mountcastle 1997] [DeFelipe et al. 2002] [Carlo and Stevens 2013]

# Some Engineering Considerations

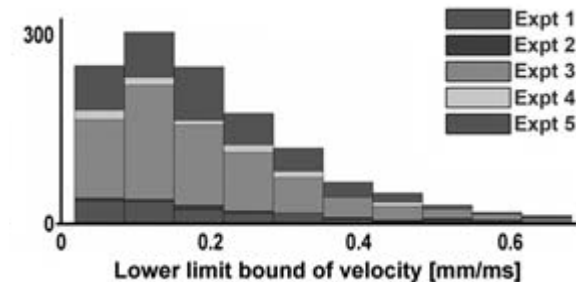
- ❑ Neuron latency
  - *1's of milliseconds per neuron*
- ❑ Transmission delay
  - Path lengths order (100s um)
  - Prop. delay order (100s um per ms)
  - *1's of milliseconds between two neurons*
- ❑ Multiple synapses per neuron pair
  - On the order of 5 -10



from Hill et al. 2012



from Fauth et al. 2015



from Bakkum et al. 2008

# Low Resolution Data

---

- ❑ Biological materials and “hardware” structure support only low resolution data
  - 3-4 bits
- ❑ But that is OK
  - A lot of information can be captured by low resolution values
  - Consider 4-bit grayscale:



- ❑ Low resolution makes temporal encoding practical
  - Because temporal encoding is a unary encoding
  - Temporal encoding exploits time as an implementation resource
- ❑ Ultimately, low resolution signaling leads to low energy consumption

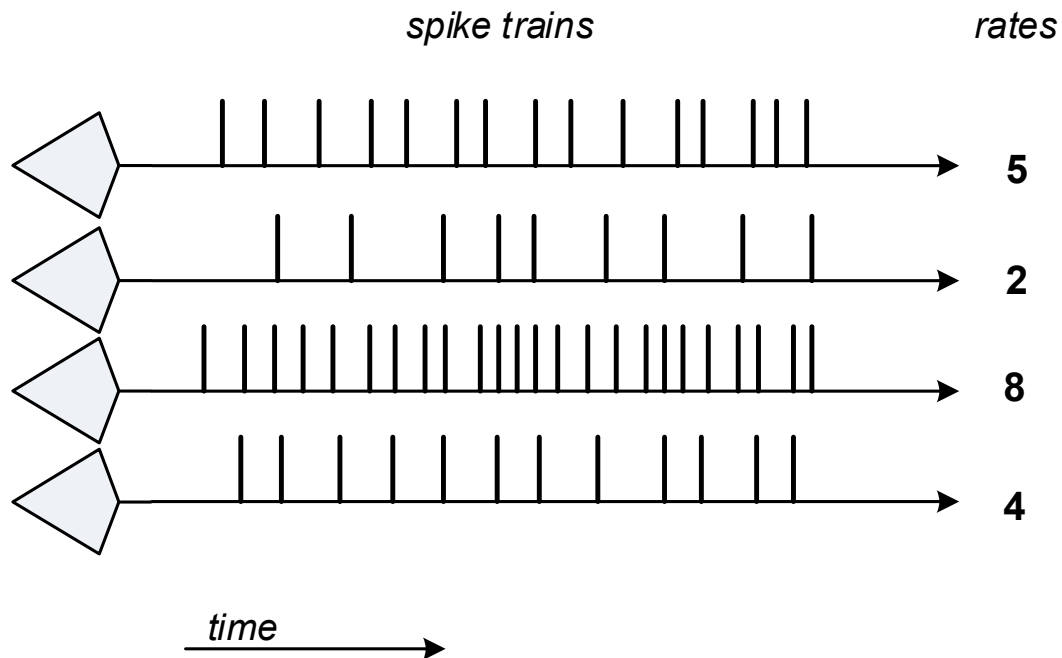
# **Biological Background**

## ***Neural Information Coding***

# Rate Coding

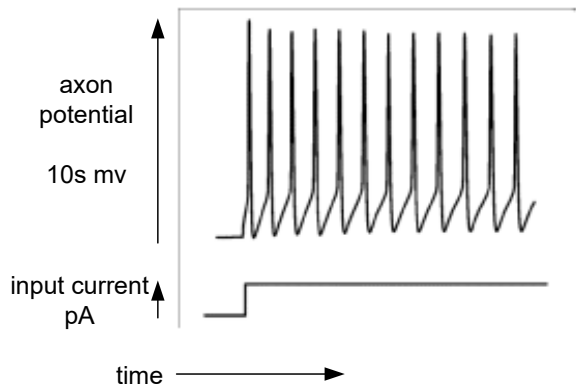
---

- Rate Coding: The name just about says it all
  - The first thing a typical computer engineer would think of doing
  - Consistent with neuron doctrine: spikes communicate information

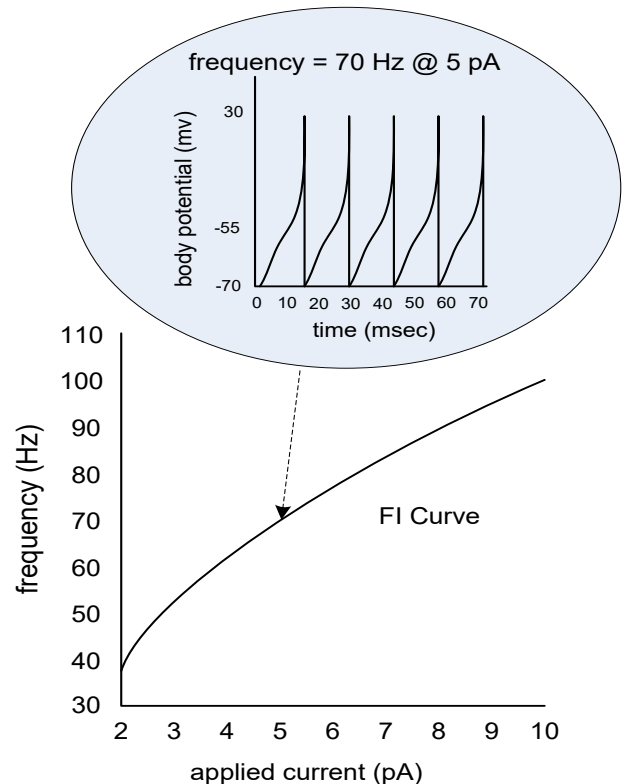


# Rate Coding

- Early experimenters applied constant current to neuron body and observed:



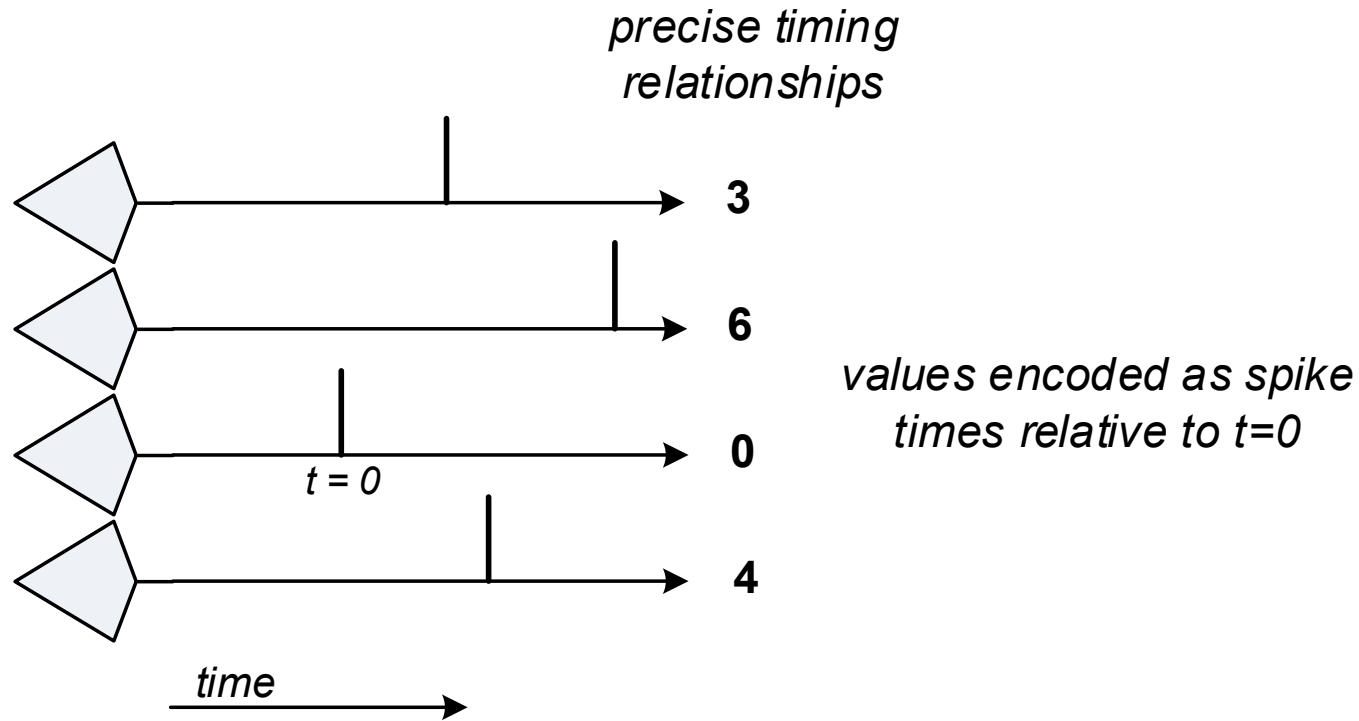
- Such constant current responses are often used to characterize neuron behavior
  - *Vary step amplitude*  $\Rightarrow$  *Frequency-Current relationship (FI Curve)*
- *Appears to support rate coding*
  - *The greater the input stimulus, the higher the output spike rate*



# Temporal Coding

---

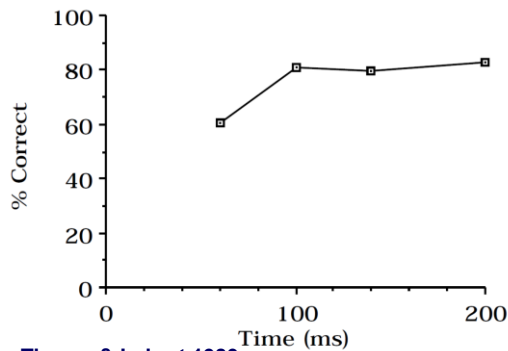
- Use relative timing relationships across multiple parallel spikes to encode information



# Key Experiment

## □ Thorpe and Imbert (1989)

- Show subject image #1 for tens of msecs (varied)
- Immediately show subject image #2
- Ask subject to identify image #1
- Measure correct responses



from Thorpe & Imbert 1989

- 100 msec is sufficient in most cases (80%)
- At least 10 neurons in series along feedforward path, each consuming ~10 msec (communication + computation).
- Conclusion: *Only first spike per line* can be communicated this quickly
  - Rates require a minimum of two spikes (probably several more)
  - Temporal coding must be present

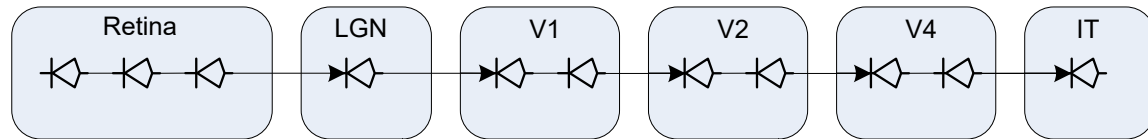
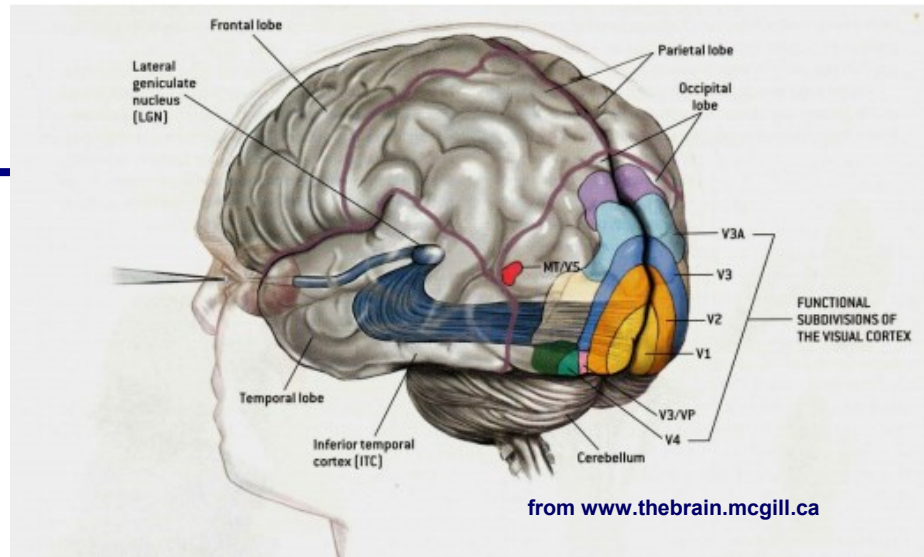
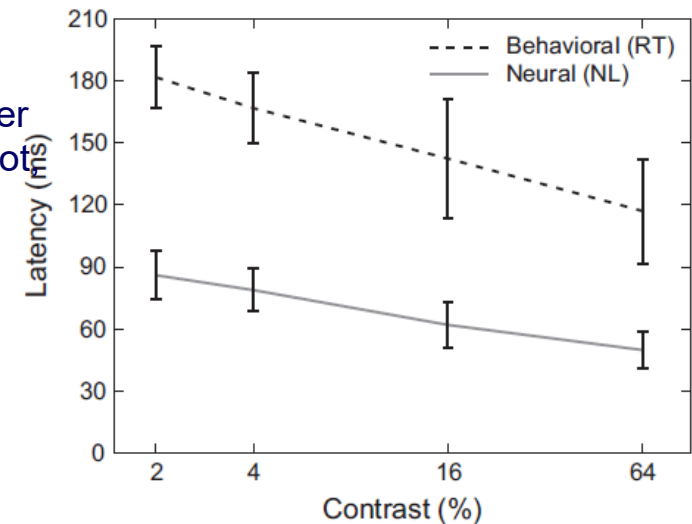
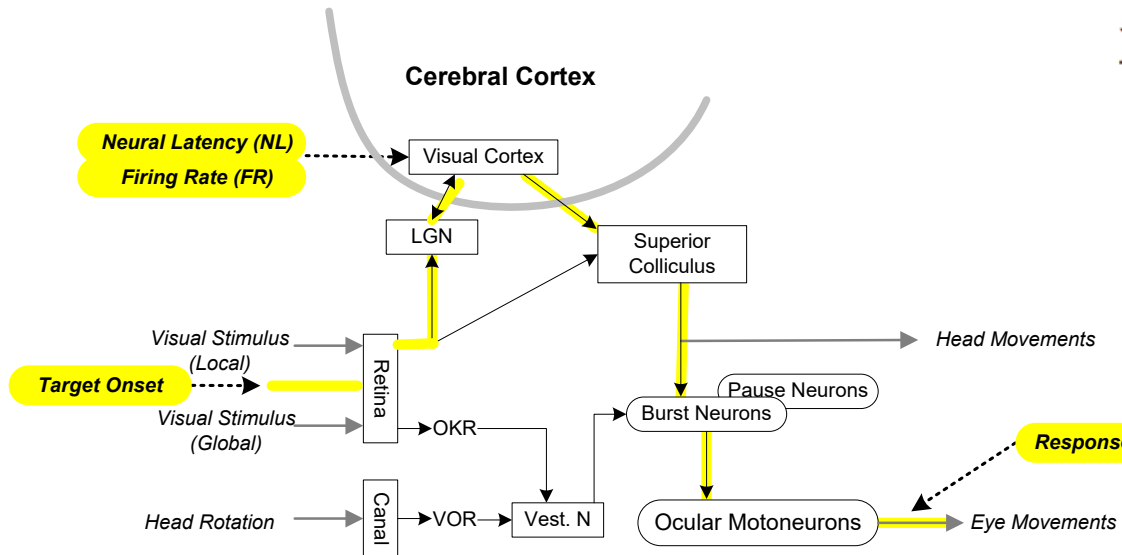


Image pathway from retina to IT, where image is identified

# Input/Processing/Output Pathway

□ Lee et al. (2010)

- Trained monkey fixes its gaze on a spot in the center of a screen
- A “dot” of variable contrast appears at a point away from the center
- If the monkey’s eyes immediately saccade to the new contrast spot
- Monkey is rewarded and a data point is collected



NL – time from target onset to spikes at V1

RT – time from target onset to saccade

FR – spike firing rate at V1

- Path length similar to Thorpe and Imbert ⇒ there is time only for first spikes
- Higher contrast spot ⇒ spikes occur sooner for stronger stimulus
- Statistically significant correlation between RT and NL
- No statistically significant correlation between FR and RT.

# It's a Matter of Survival

---

- Any organism that relies on its senses for survival depends on fast cognitive paths
- Experiments strongly suggest sensory input is communicated and processed as a *feedforward wavefront of precisely timed spikes*
  - *This is a very fast and efficient way to communicate information*
- Key question: Is this temporal communication paradigm used throughout the entire neocortex?

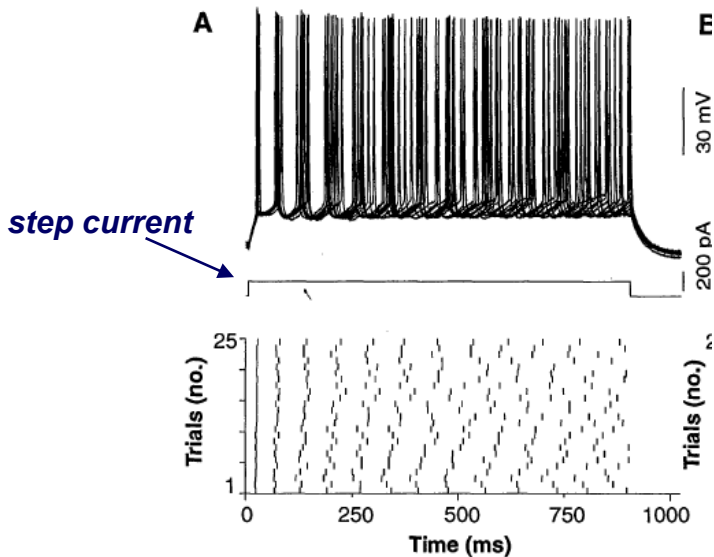
***Conjecture: Yes, especially considering uniformity hypothesis***

***Because it's a matter of survival, this is probably the fastest, most efficient cognitive method that evolution could devise. So wouldn't evolution use it everywhere as a basic technology?***

# Temporal Stability

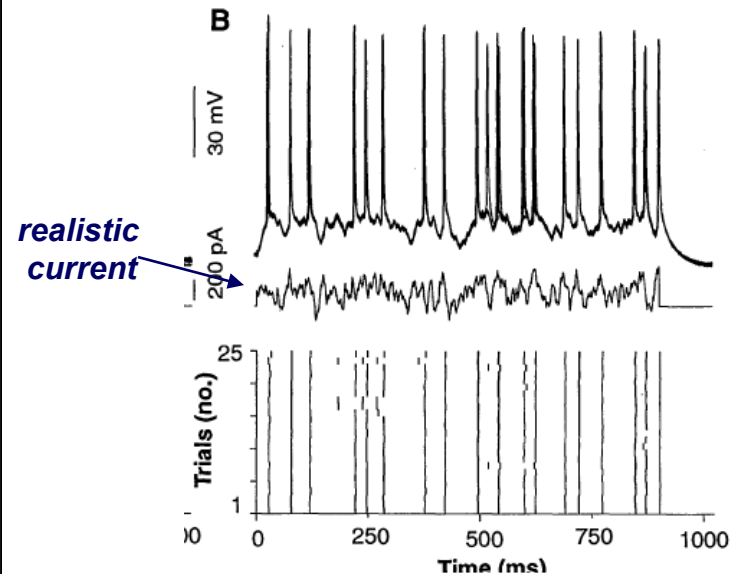
Mainen and Sejnowski (1995)

- Repeatedly inject step current into neuron body
- Capture and analyze output spike trains
- Spike trains (10 trials); Raster plots (25 trials)



- Jitter  $\Rightarrow$  several spikes required to establish a reliable rate

- Repeatedly inject *realistic* pseudorandom current into neuron body
- Capture and analyze output spike train

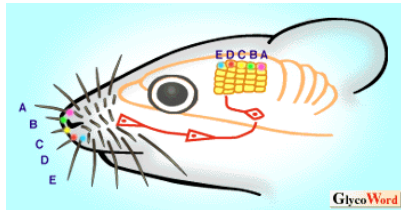


- Spike times are very stable

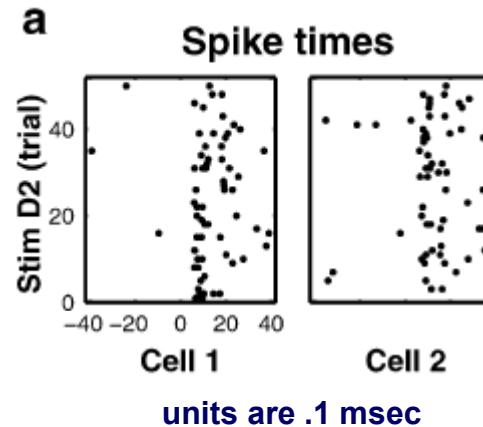
*“stimuli with fluctuations resembling synaptic activity produced spike trains with timing reproducible to less than 1 millisecond”*

# Time Resolution

- Petersen, Panzeri, and Diamond (2001)
  - Somatosensory cortex of rat (barrel columns)
  - Map a specific whisker to two cells in the receiving column
  - Figure shows latency to first spike in response to whisker stimulation



<http://www.glycoforum.gr.jp/science/word/proteoglycan/PGA07E.html>

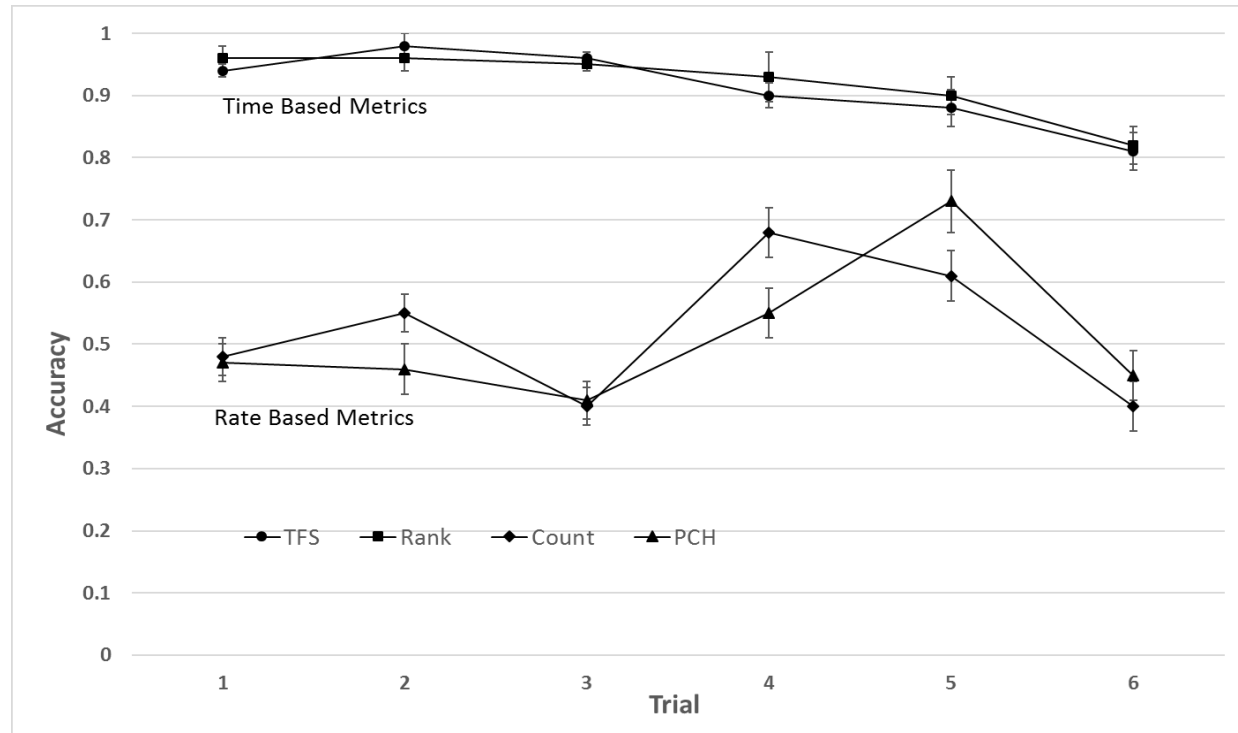


***“82%–85% of the total information was contained in the timing of individual spikes: first spike time was particularly crucial. Spike patterns within neurons accounted for the remaining 15%–18%.”***

# Information Content

## □ Kermany et al. (2010)

- In vitro culture of neurons
- Multi-grid array
- Five stimulation points
- Collect output spikes at least one neuron removed from stimulation point.
- Collect spike data for 100 msec
- Which was the stimulation point?
- Analyze data two ways for spike timing:
  - time to first spike (TFS)*
  - rank order*
- Analyze two ways for rates:
  - total spikes*
  - pop. count histogram (PCH)*
- Use a conventional classifier to determine the accuracies



- Time based reductions have higher accuracies (implying higher information content)

# Summary

---

- If you were selecting a technology and had these two choices, which would you choose?

	Temporal Coding	Rate Coding
Fast enough?	<i>Yes</i>	<i>No</i>
Information content	<i>High</i>	<i>Low</i>
Energy consumption	<i>Low</i>	<i>High</i>

*“When rephrased in a more meaningful way, the rate-based view appears as an ad hoc methodological postulate, one that is practical but with virtually no empirical or theoretical support.”* -- Brette 2015

# **Biological Background**

## ***Encoding Sensory Input***

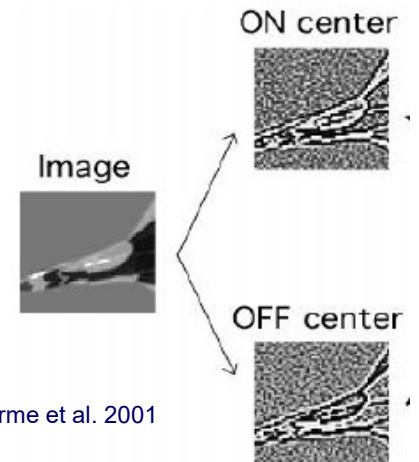
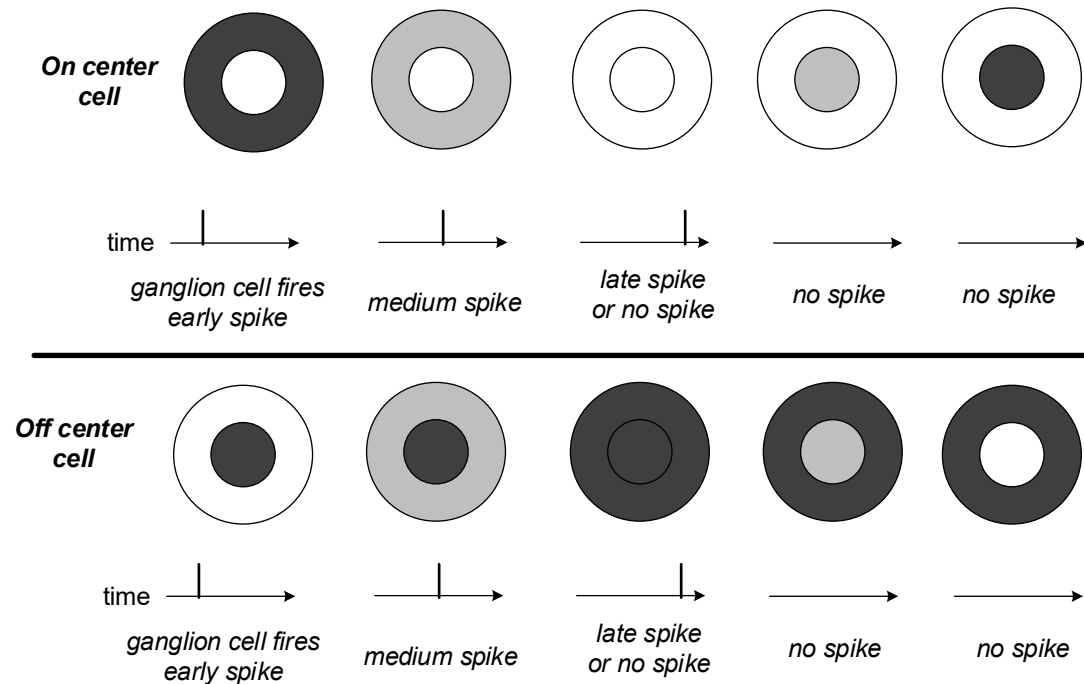
# Sensory Inputs

---

- ❑ Raw input values are “Sensory Inputs”
  - They correspond to physical inputs from the biological senses
  - In an artificial environment, we can also invent our own “senses”
    - E.g., Temperature sensors spread out over a grid
- ❑ Encoding translates sensory inputs to temporal code
  - **Vision**: fairly straightforward, often used by theoreticians
  - **Auditory**: uses frequency transform
  - **Olfactory**: must learn/operate quickly over a very large feature space
  - **Somatosensory**: extremely straightforward, has proven valuable to experimentalists
  - **Place**: an extremely interesting sense that is internally produced
  - **Generic**: a grid of temperature sensors, for example

# Visual Input

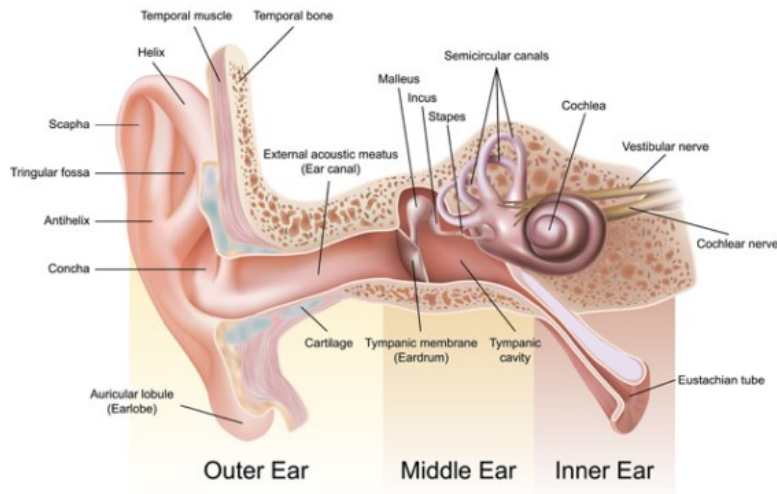
- ❑ OnOff retinal ganglion cells
  - Perform edge detection
- ❑ Encode spikes according to contrast between center and surround
  - Most intense contrast yields earlier spikes



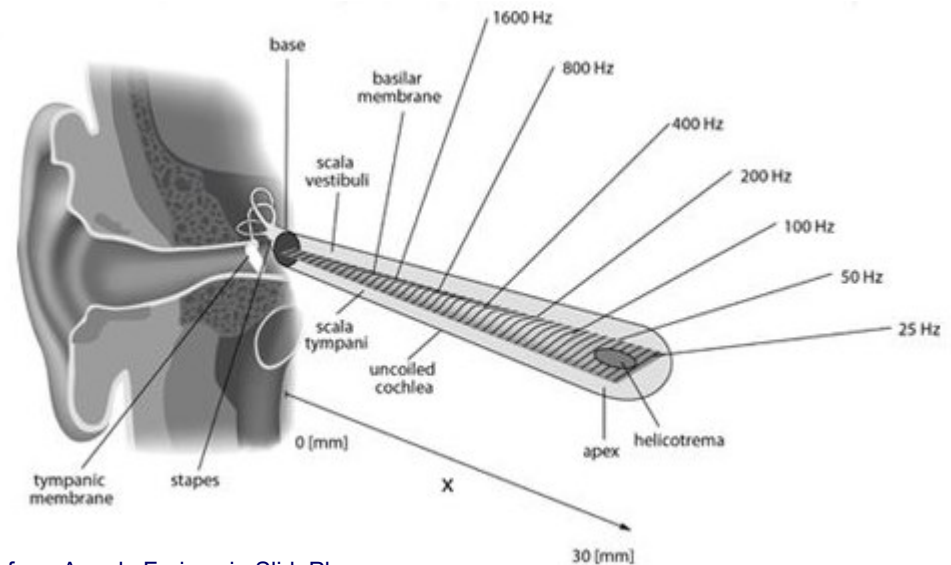
from DeLorme et al. 2001

# Auditory Input

- Tiny hairs (cilia) at different locations in cochlea are sensitive to different frequencies
  - Hair cells perform frequency transform
  - Most intense frequencies generate earlier spikes



from Lankamedicalaudio.com

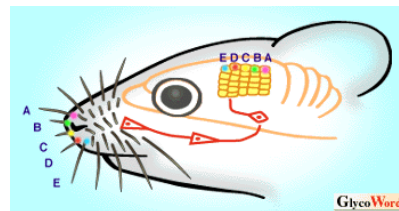


from Angelo Farina via SlidePlayer.com

# Somatosensory Input

---

- ❑ Wide variety of receptor types
  - Many are mechanical
- ❑ Rat whiskers are a prime example
  - Highly developed, finely-tuned sensor in nocturnal rodents
  - Generate spikes that correlate with degree of “whisking”
  - The greater/faster the deflection, the sooner the spike
- ❑ Widely used by experimental researchers
  - Each whisker is a receptive field which drives a particular macro-column or “barrel column”
  - This relationship permits well-controlled *in vivo* experiments

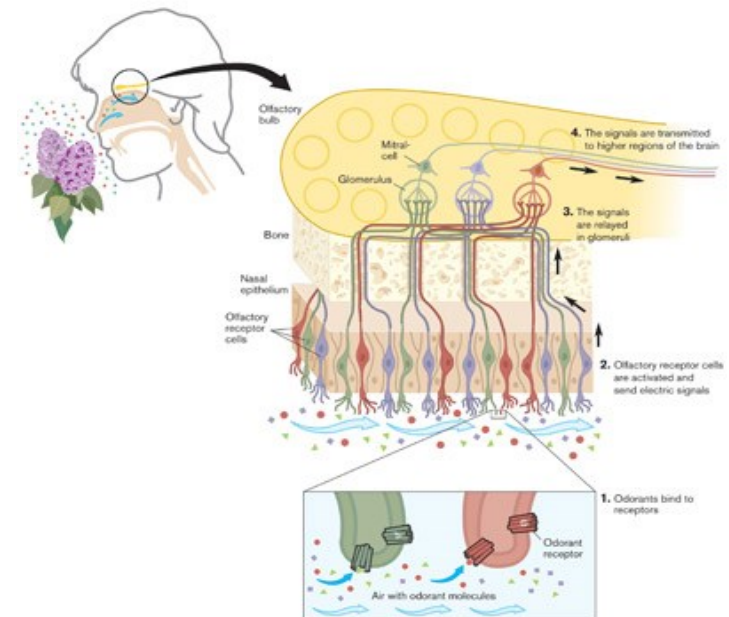


<http://www.glycoforum.gr.jp/science/word/proteoglycan/PGA07E.html>

# Olfactory Input

- ❑ Olfactory receptor neurons (ORNs)
  - Dendrites connect to tiny hairs containing proteins that bind to odor molecules
  - Stronger odor affinity causes earlier spike
- ❑ 10M ORNs in a human
  - Proteins coded with 1000 different genes
- ❑ Must learn/operate quickly over a very large feature space
  - To deal with the unpredictability of the olfactory world
- ❑ Low bandwidth
  - In contrast: Vision is high bandwidth
  - (Imagine reading a book where different odors encode letters)
- ❑ See Laurent (1999)

Odorant Receptors and the Organization of the Olfactory System

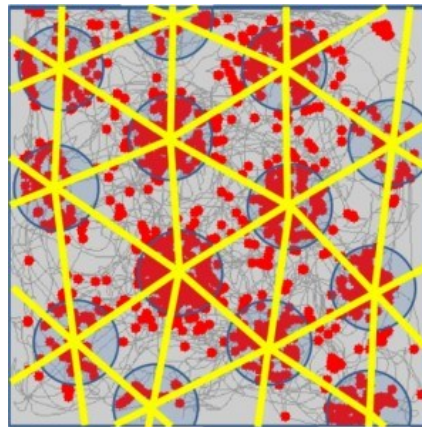


from [health.howstuffworks.com](http://health.howstuffworks.com)

# Sense of Place

---

- ❑ The innate sense of where you are in space
  - your “place”
- ❑ Does not have a physical world interface (unlike the other senses)
- ❑ Stored as macro-column grid in entorhinal cortex
  - Hexagonal/triangular
  - Located between neocortex and hippocampus
  - Neurons fire as subject moves in space
- ❑ Aside: hippocampus is responsible for spatial memories & navigation
  - A lot of theoreticians are interested in the Hippocampus



from placeness.com

***BREAK***

# Outline

---

- ❑ Introductory Remarks
- ❑ Research Focus: Milestone TNN
  - Classification and Clustering
  - Neural Network Taxonomy
- ❑ Biological Background
  - Overview
  - Neural Information Coding
  - Sensory Encoding
- ❑ Fundamental Hypotheses
  - [Theory: Time as a Resource](#)
- ❑ Low Level Abstractions
  - [Excitatory Neuron Models](#)
  - [Modeling Inhibition](#)
- ❑ Column Architecture
- ❑ Case Studies
  - Recent TNNs
- ❑ Space-Time Theory and Algebra
- ❑ Implementations
  - Indirect Implementations: Simulation
  - Direct Implementations: Neuromorphic Circuits
- ❑ Concluding Remarks

# **Fundamental Hypotheses**

# Neuron Doctrine

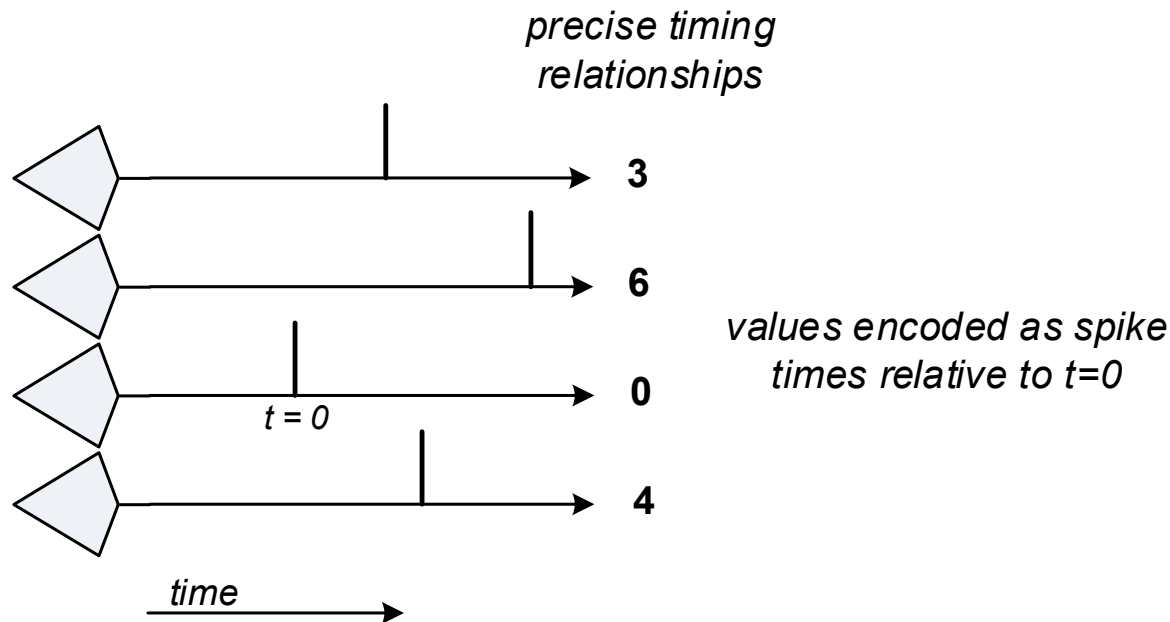
---

- ❑ The *neuron doctrine* serves as the foundation for our enterprise
- ❑ A set of accepted principles (or laws) developed over a span of 100+ years
  - Neurons are the atomic units of the brain's structure and function
  - Neurons consist of a body, dendrites, and axons
  - Information is received at dendrites connected to the neuron body and information is transmitted to other neurons via the axon (law of polarization)

*(and others)*
- ❑ Dale's law: *A given neuron drives only one synapse type: either all excitatory or all inhibitory*
- ❑ To this foundation, add several hypotheses:
  - Uniformity Hypothesis (*given earlier*)
  - Temporal Hypothesis
  - Feedforward Hypothesis
  - Reliability Hypothesis
- ❑ *Caveat*: these hypotheses *are not* universally accepted and are often implicit

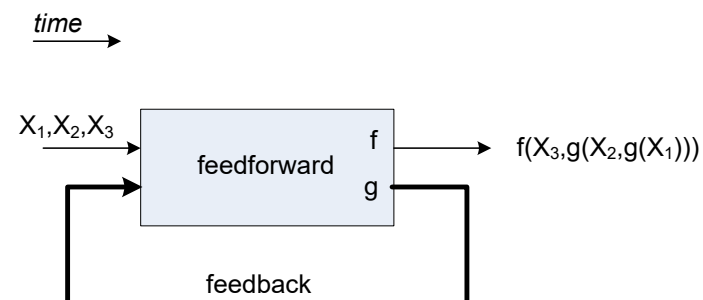
# Temporal Hypothesis

- Hypothesis: *Temporal Coding is used in the neocortex*
- Experimental results (given earlier) support this hypothesis
  - Synopsis: temporal coding is much *faster*, much more *efficient*, and much more *plausible* than rate coding.

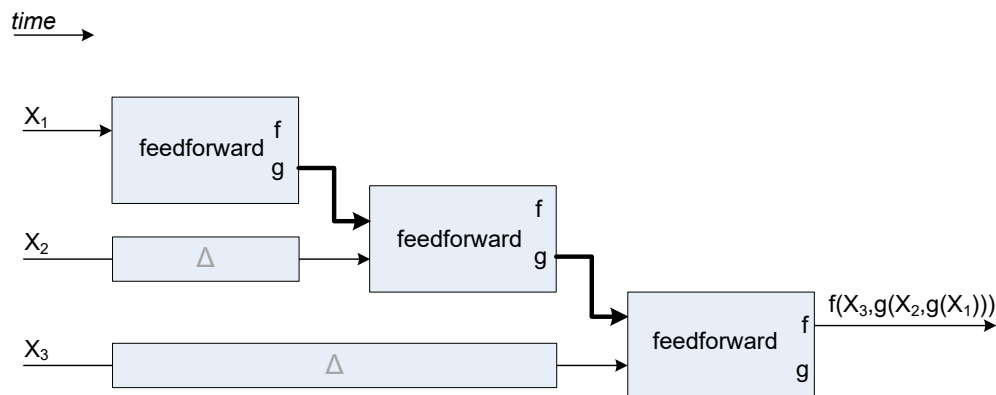


# Feedforward Hypothesis

- Consider any computing network with feedback
  - Can be recurrent ANN, RBM, HTM... almost anything, really



*If* a feedback computation satisfies the discrete-ness and finite-ness constraints of Turing's thesis  
*Then* there is an equivalent feedforward computation

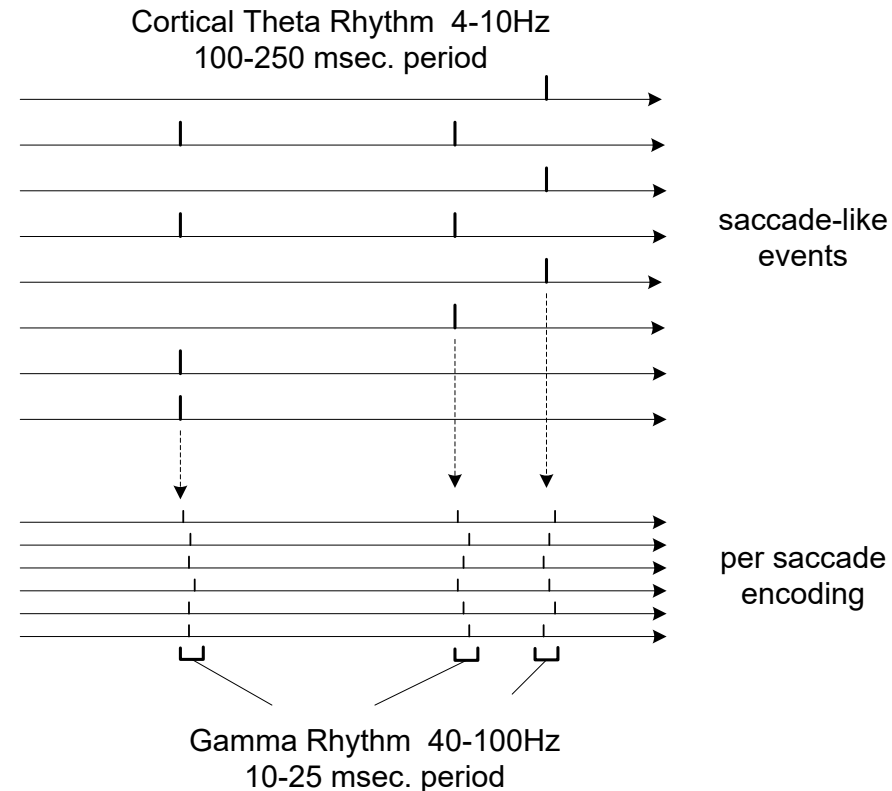


- Feedforward nets have *same* intrinsic computing power as Feedback nets
  - Feedback nets may be *much much* more efficient, but they can't perform a theoretically larger set of computations (in the Turing sense)
- Therefore, we study feedforward networks without losing any computational generality

# Aside: Synchronizing Rhythms

- ❑ If we add feedback, how might synchronization be performed?
- ❑ There are no clocked storage elements
- ❑ Inhibitory oscillations
  - Gamma rhythm: roughly 40-100 Hz
  - 10-25 msec coding window
- ❑ This is per-neuron synchronization
  - Like a pipeline w/ a single gate per stage
  - Neuron feeding itself is easily handled

*This is speculative*

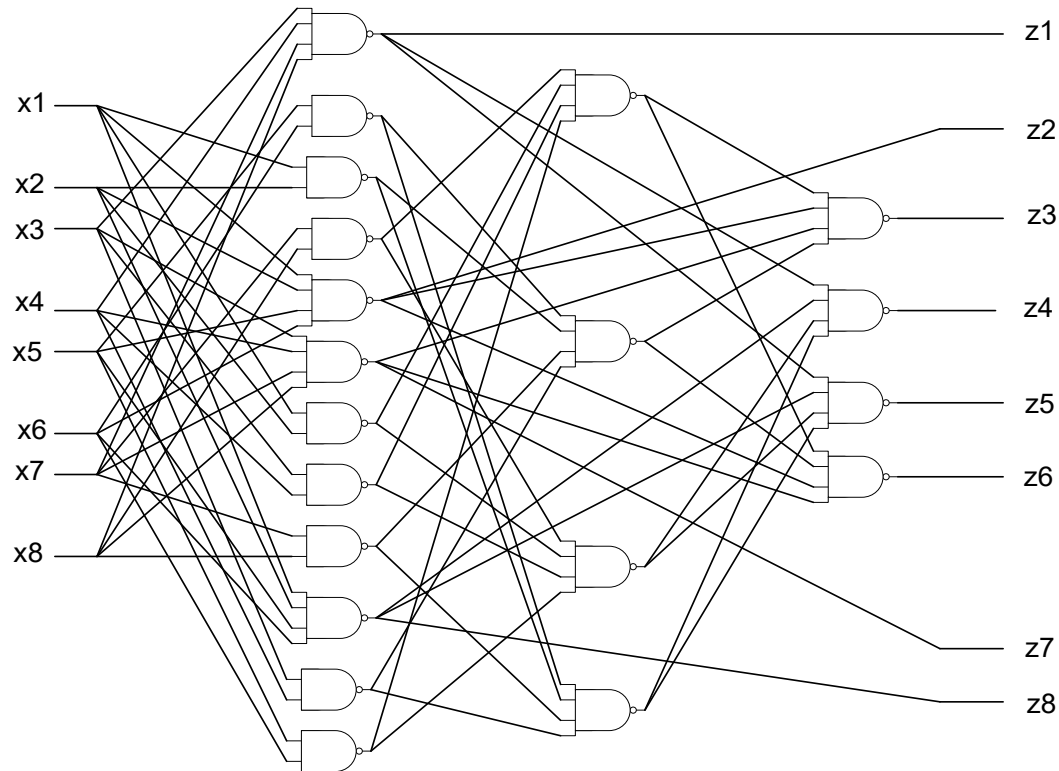


Refs: [Singer 2010] [Fries et al. 2007]

# Fault Tolerance

---

- What does this logic network do?



**answer: half-adder w/ inverted outputs**

**fault-tolerant interwoven logic (quadded)**

Ref: [Tryon 1962]

# Temporal Stability

---

- ❑ Relatively little variation (skew) in delays and latencies within regions of closely communicating neurons
- ❑ Probably required, in the absence of clocked storage elements
  - Over at least an entire excitatory column
    - More likely at least a macro-column
- ❑ Global skew may higher than local skews
  - Skew within a column or macro-column may be less than between widely separated columns
- ❑ There may be a web of locally interacting mechanisms that tune delays and latencies to provide temporal uniformity
  - Oscillatory inhibition may be an associated cause and/or effect
- ❑ These mechanisms probably add a lot of *apparent* complexity

***Everything on this slide is speculative***

# Reliability Hypothesis

---

- ❑ The computing paradigm used in the neocortex is *independent* of supporting fault-tolerant and temporal stability mechanisms.
  - If reliability and temporal stability are perfect – then the paradigm probably works better than with less reliable networks.
- ❑ We will assume:
  - *All circuits are completely reliable*
  - *Temporal stability guarantees at least 3-bits of timing resolution*

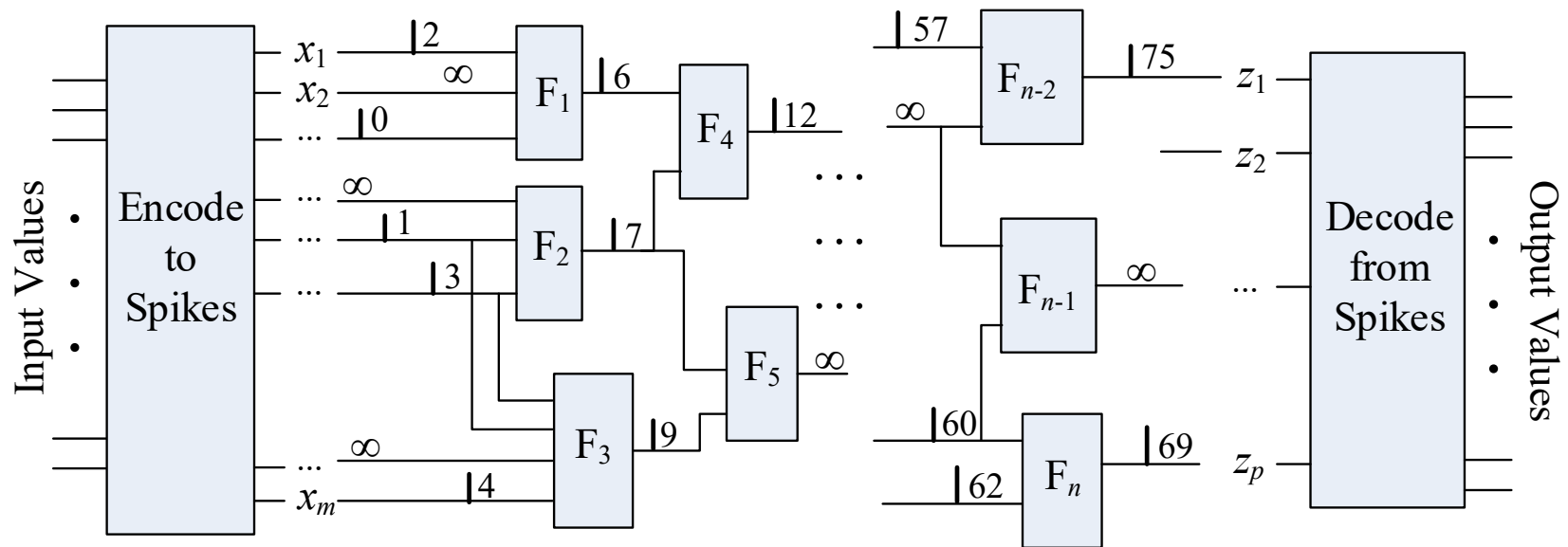
# Summary of Hypotheses

---

- ❑ Neuron Doctrine
- ❑ Uniformity Hypothesis
- ❑ Temporal Hypothesis
- ❑ Feedforward Hypothesis
- ❑ Reliability Hypothesis

# Summary: TNN Architecture Features

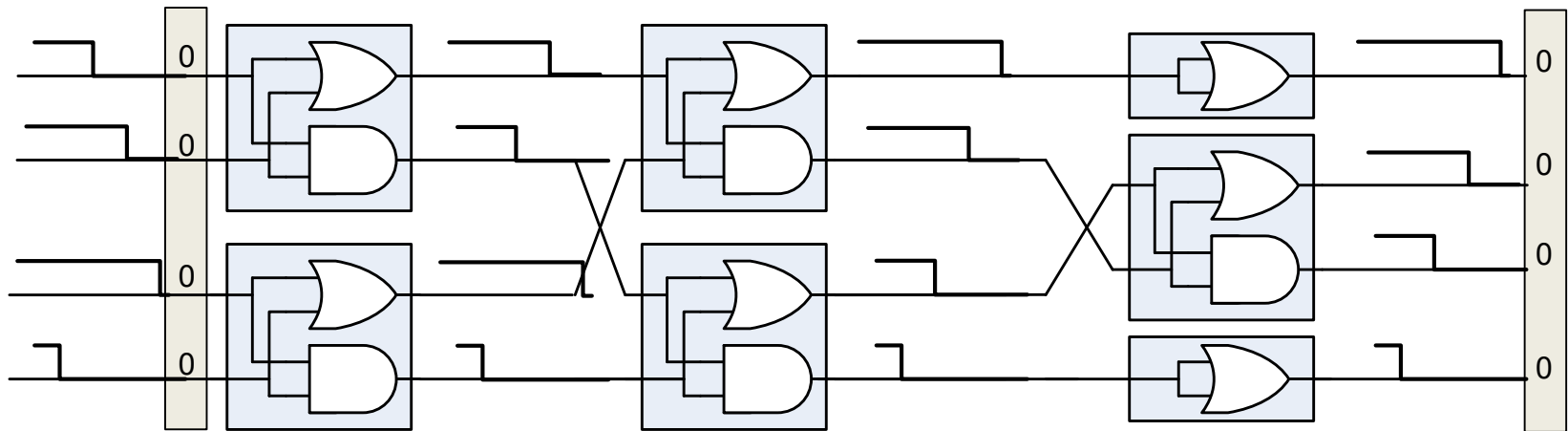
- ❑ Feedforward
- ❑ Intrinsically fault-tolerant
- ❑ Temporally stable



# **Theory: Time as a Resource**

# Step 1: Synchronous Binary Logic

- The way deal with time when constructing digital systems:  
(Assume unit delay gates)

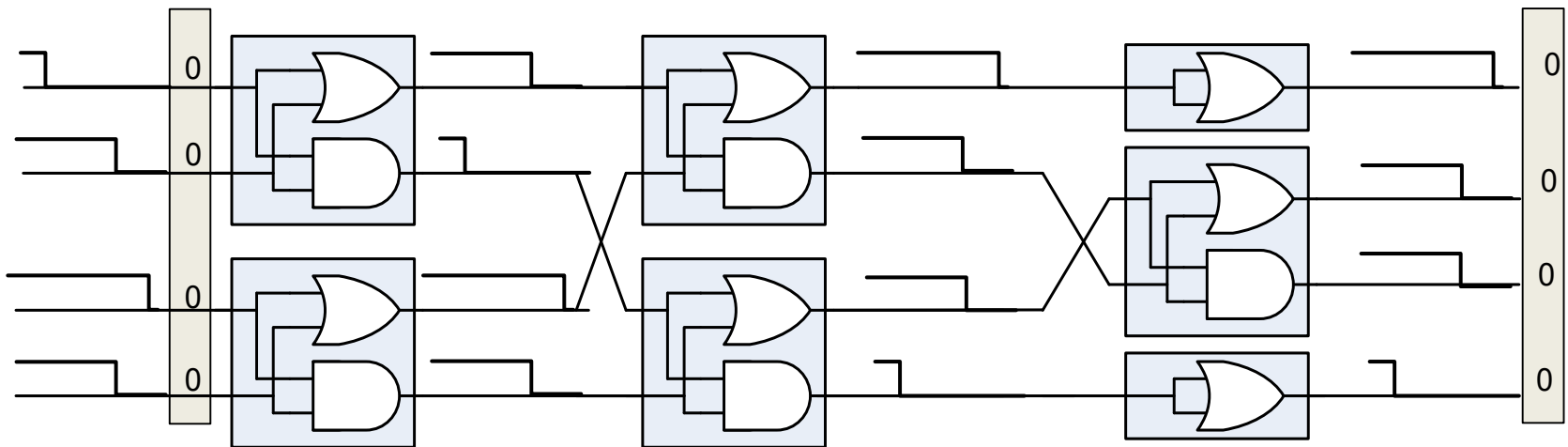


- 1) Input signals change
  - 2) Wait for output signals to stabilize
- $\langle 0,0,0,0 \rangle \rightarrow \langle 0,0,0,0 \rangle$

# Synchronous Binary Logic, contd.

- Again, with different input timing

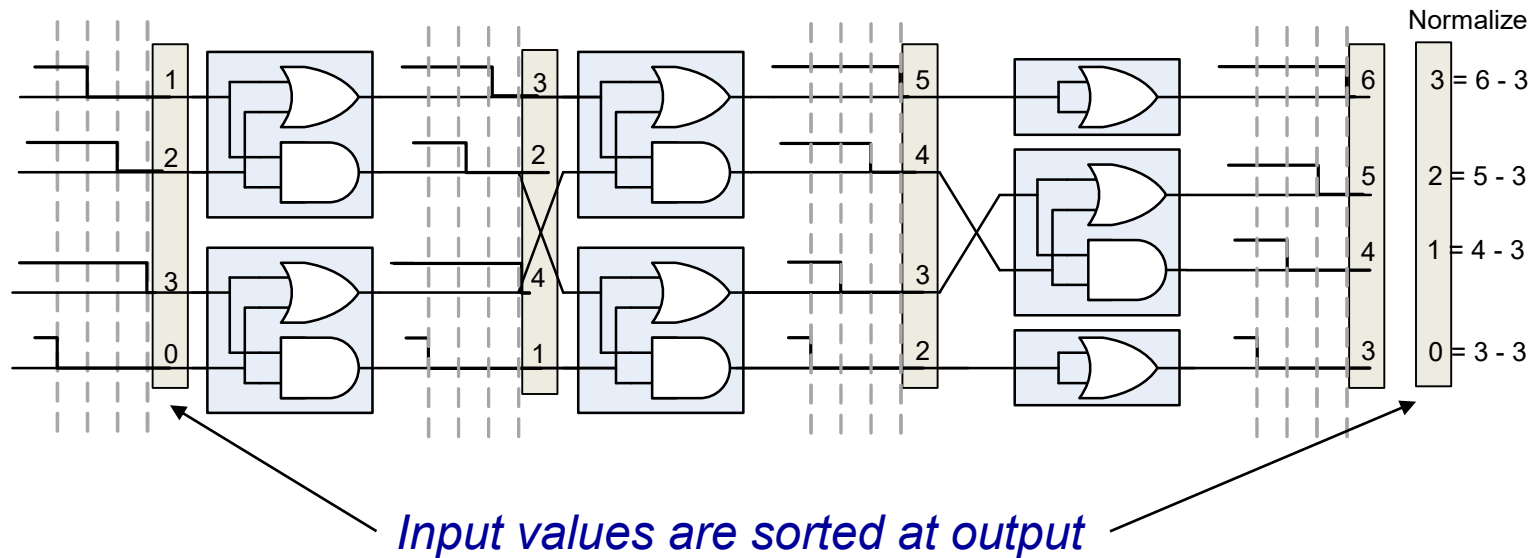
*We always get the same, "correct" answer*



- Synchronized logic removes the effects of physical time
  - Results *do not* depend on the actual delays along network paths
  - Ditto for delay independent asynchronous logic (e.g., handshaking)

## Step 2: Race Logic

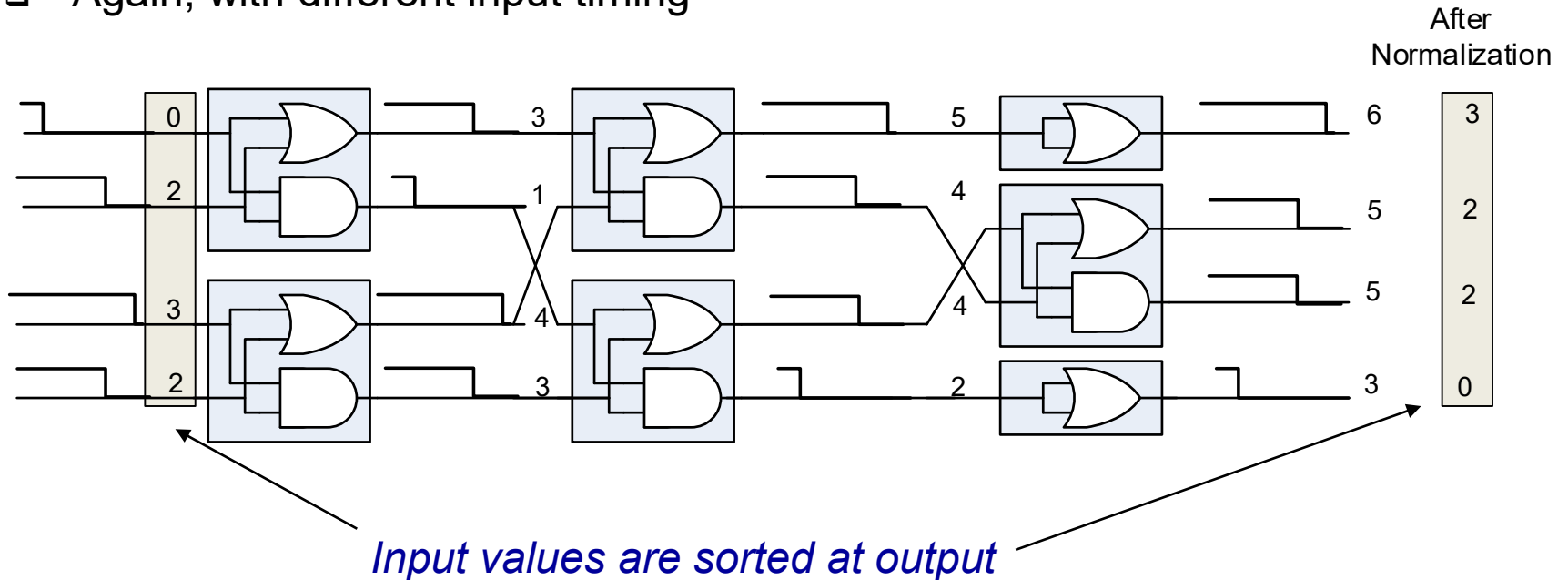
- ❑ Sherwood and Madhavan (2015)
- ❑ Process information encoded as relative edge times



- ❑ Time is divided into discrete intervals (unit time)
- ❑ Input edge times encode input information (**relative to first edge**)
- ❑ Output edge times encode the result (**relative to first edge**)

# Race Logic, contd.

- Again, with different input timing



- Same circuit, same signals, same everything as with synchronous method, *except* network implements a much more powerful function  
integer sort vs.  $\langle 0,0,0,0 \rangle \rightarrow \langle 0,0,0,0 \rangle$

*same circuit sorts numbers of any size*

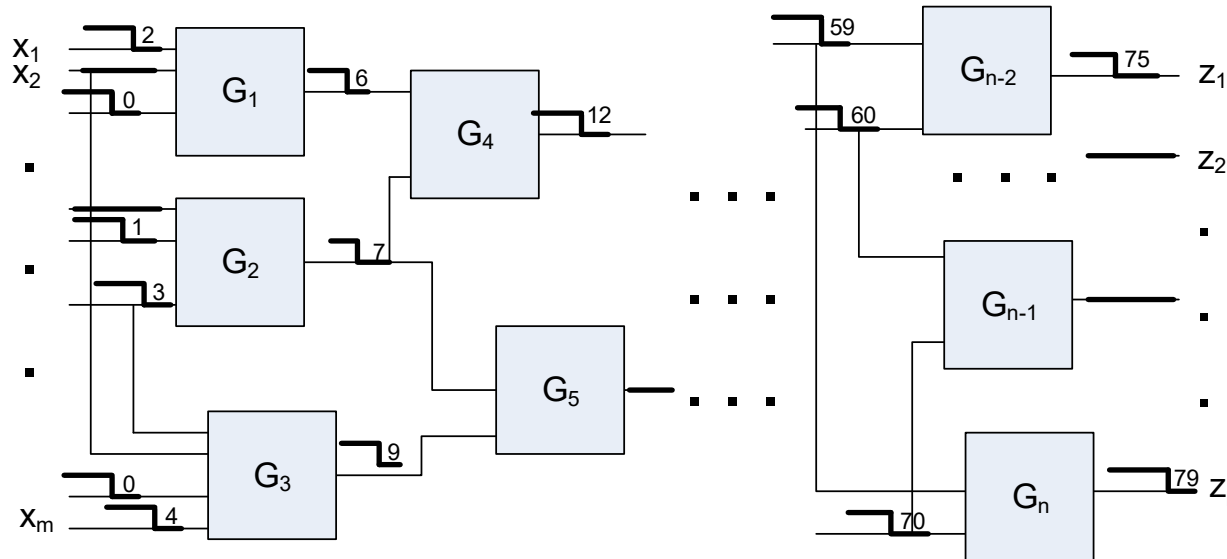
# The Temporal Resource

---

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

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

# Step 3: Generalize Race Logic



□ Each functional block is:

- **Causal:** time of output edge is only affected by earlier-occurring input edges.
- **Invariant:** shifting all input edge times by a uniform amount causes the output edge to time shift by the same amount

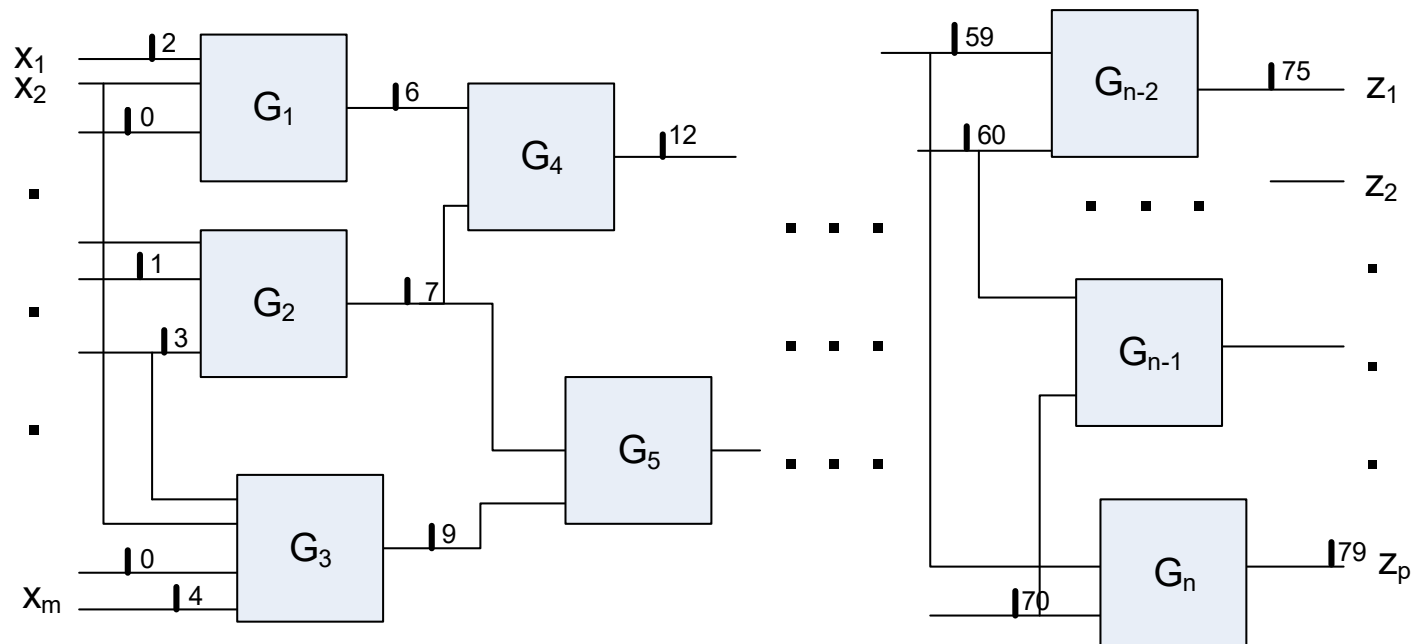
# Step 4: Generalize Transient Events

Nothing special about  $1 \rightarrow 0$  edges...

Could use  $0 \rightarrow 1$  edges

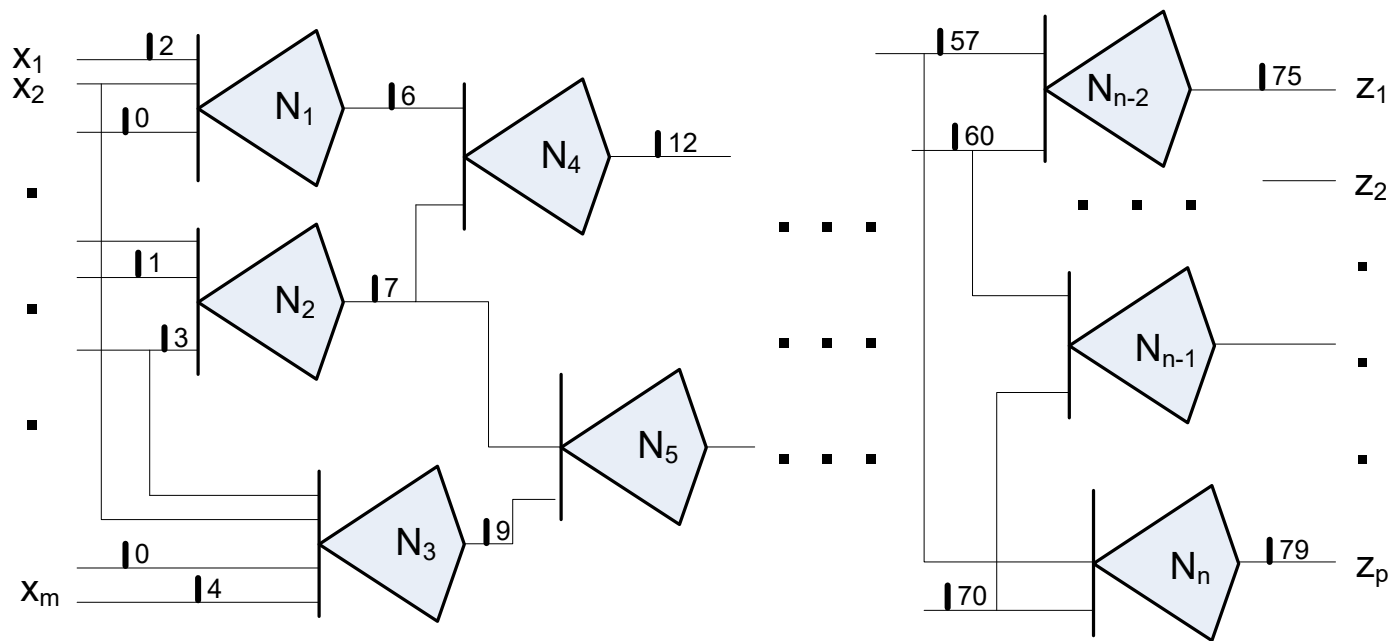
OR

pulses (spikes):



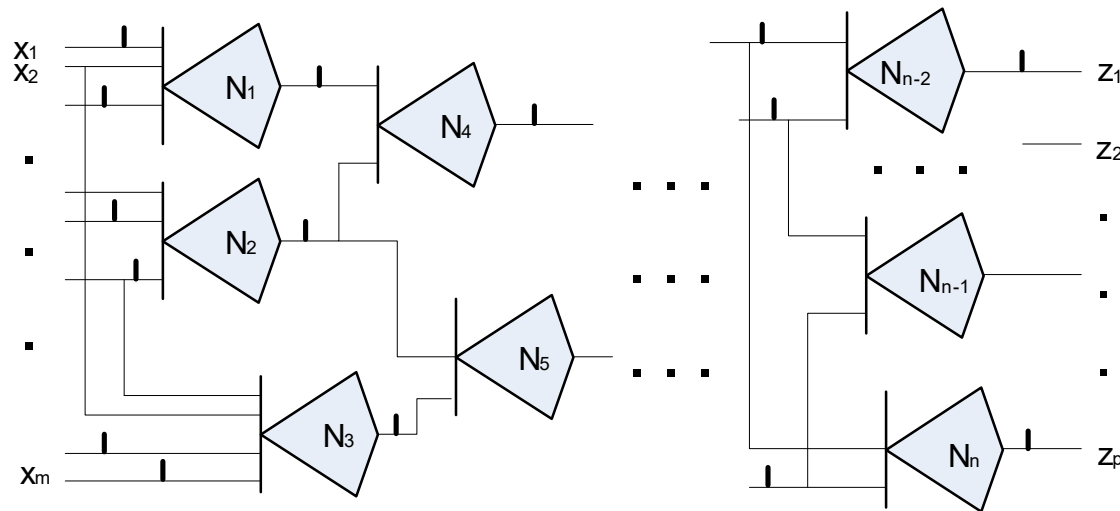
# Step 5: Specialize Functional Blocks

- Function blocks can be *spiking neurons*
- This is a Temporal Neural Network (TNN)
  - As envisaged by an important group of theoretical neuroscientists



# Temporal Neural Networks (TNNs)

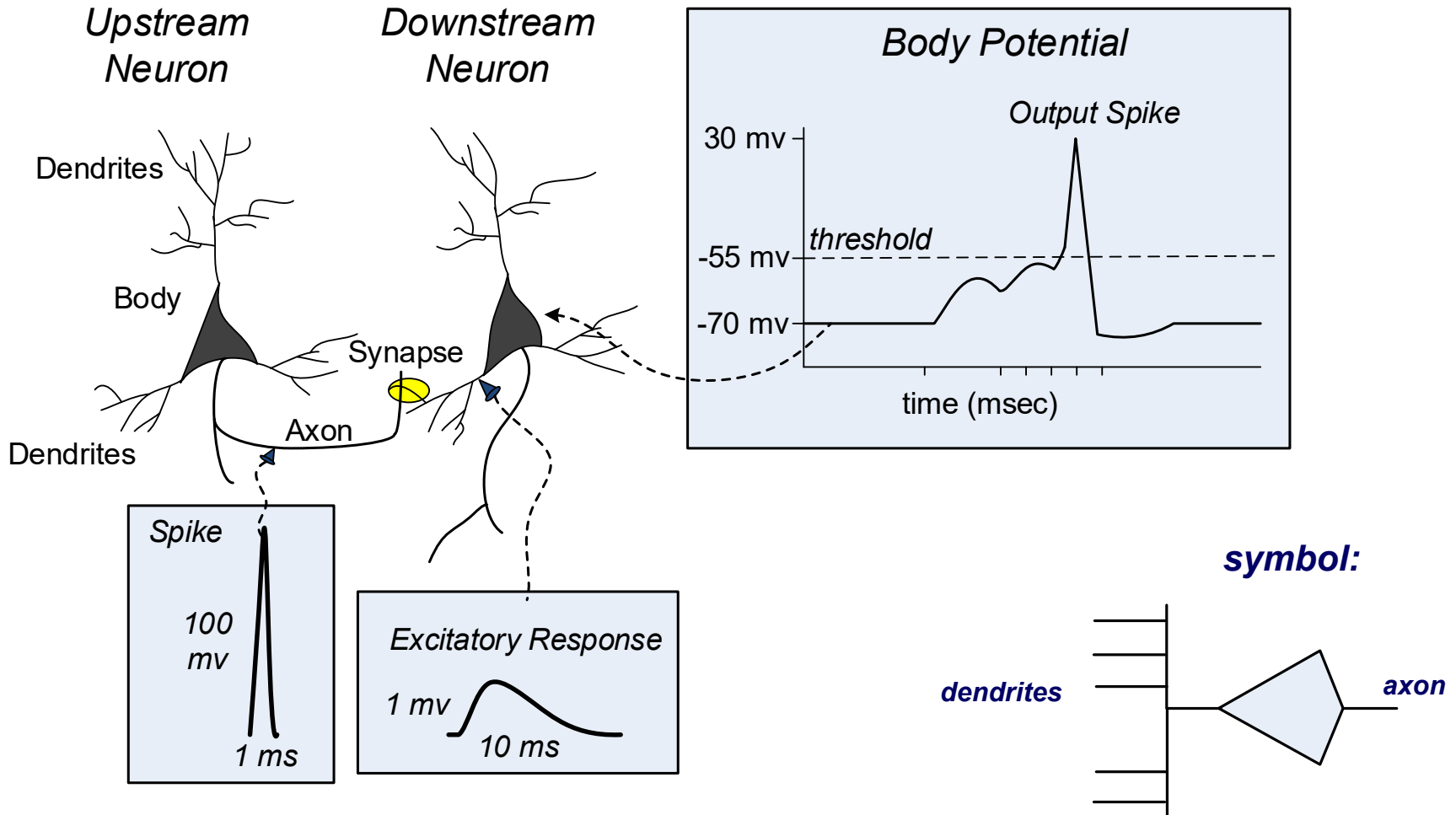
- ❑ Input spike times encode information (temporal coding)
- ❑ Input initiates a wave of spikes that sweeps forward through the network
  - At most one spike per line per computation
- ❑ Output spike times encode result information



# **Low Level Abstractions**

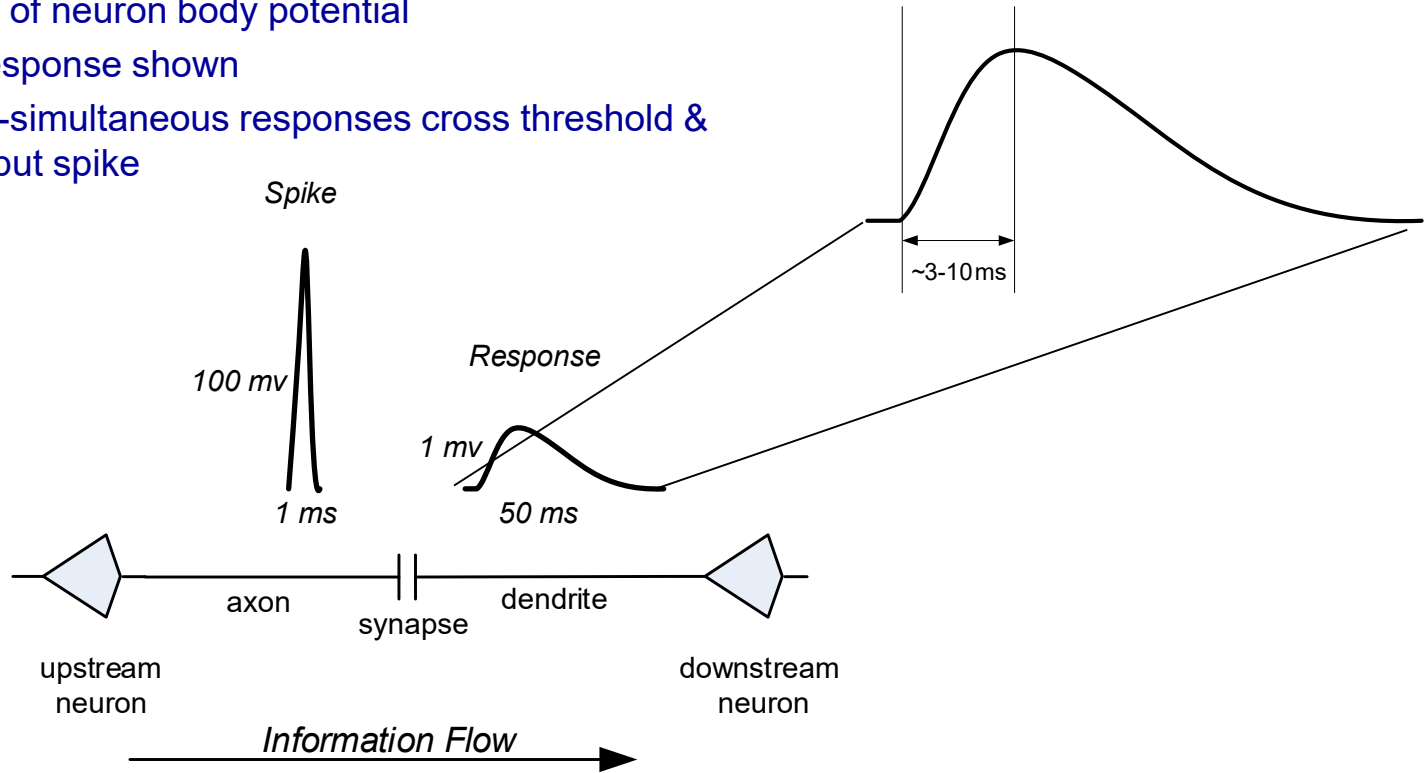
## ***Excitatory Neuron Models***

# Neuron Operation (Re-Visited)



# Spikes and Responses

- Spike followed by Response
  - Two sides of electrical-chemical-electrical translation at synaptic gap
- Spike is very narrow
  - Result of ions being actively moved along the axon
- Response is much wider with much lower peak potential
  - Rise and fall of neuron body potential
  - Excitatory response shown
  - Many nearly-simultaneous responses cross threshold & produce output spike



# Neuron Models

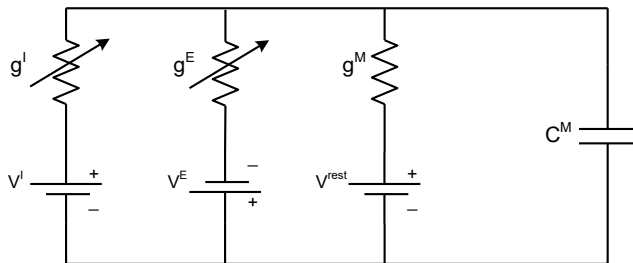
- Hodgkin-Huxley (1952)
  - Nobel Prize
  - Complex coupled differential equations
  - Gold standard for biological accuracy
  - But we are not interested biological accuracy! (only plausibility)
  - We want *computational capability*
- Leaky Integrate and Fire (LIF) simplification

$$C_m \frac{dV}{dt} = -g_L(V - V_L) - \bar{g}_{Na} m^3 h (V - V_{Na}) - \bar{g}_K n^4 (V - V_K)$$

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n$$



$$\tau_M dV_t/dt = -(V_t - V^{rest}) - ((g_t^E(V_t - V^E) - g_t^I(V_t - V^I) + I_b) / g^M)$$

$$\tau_E dg_t^E/dt = -g_t^E ; g_t^E = g_{t-1}^E + \sum g^E w_{ij}$$

$$\tau_I dg_t^I/dt = -g_t^I ; g_t^I = g_{t-1}^I + \sum g^I w_{ij}$$

# LIF Discretized Solution

---

- Targeted at efficient digital implementation

*shorthand:*

$$r_M = dt/\tau_M \quad g^M = dt/C_M$$

$$k_M = 1 - dt/\tau_M$$

$$k_E = 1 - dt/\tau_E$$

$$k_I = 1 - dt/\tau_I$$

$$V_t = V_{t-1} (k_m - r_m g_t^E - r_m g_t^I) + r_m g_t^E V^E + r_m g_t^I V^I$$

$$r_m g_t^E = g_{t-1}^E (k_E) + \sum g^E r_m w_{ij}$$

$$r_m g_t^I = g_{t-1}^I (k_I) + \sum g^I r_m w_{ij}$$

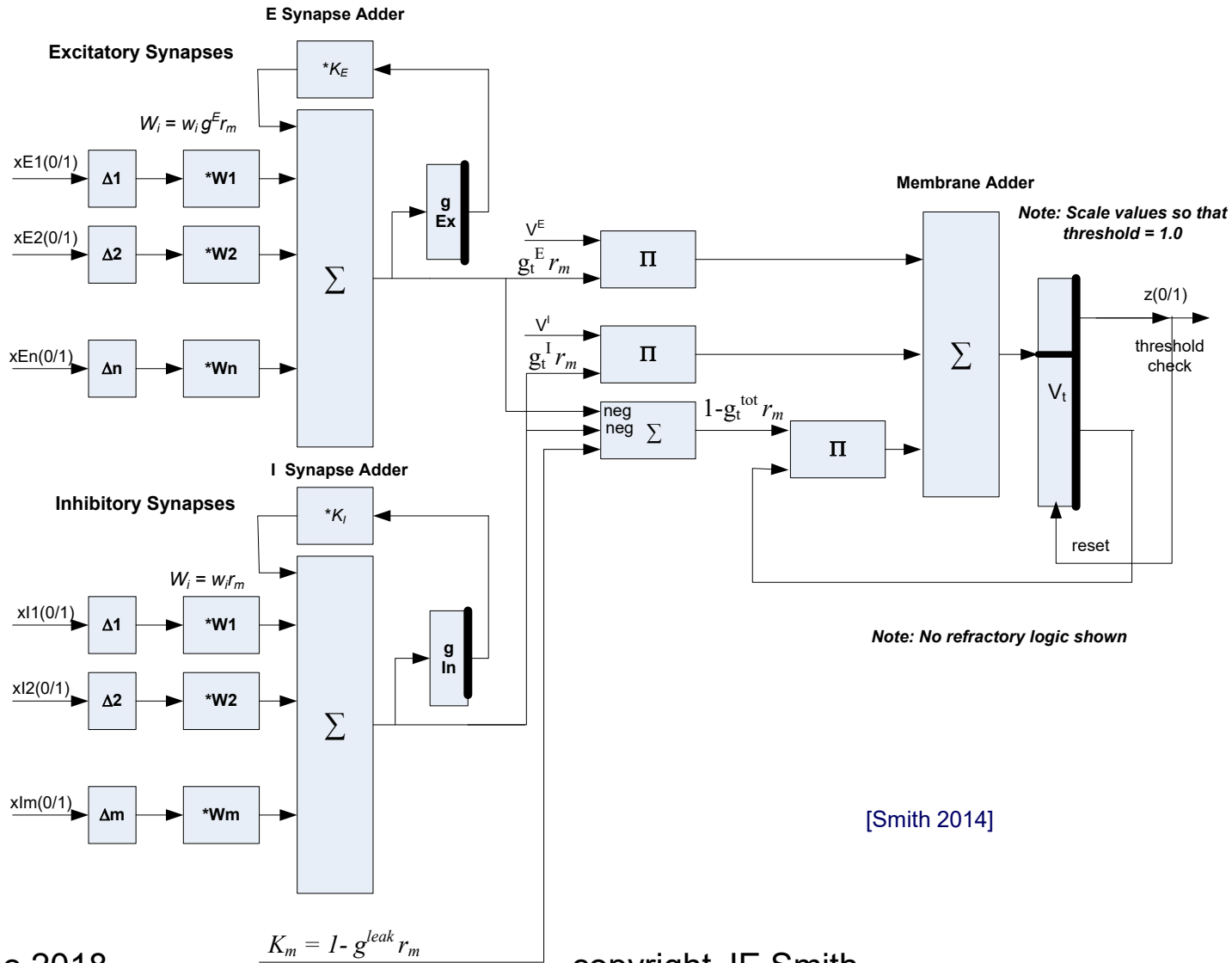
*If  $V_t \geq \theta$  then fire and set  $V_t = 0$*

- Discretized version is linear

- Conductances can be scaled to  $g^M = 1$
- Shift voltages so that  $V_{\text{rest}} = 0$
- Scale voltages so that  $V_{\text{th}} = 1$

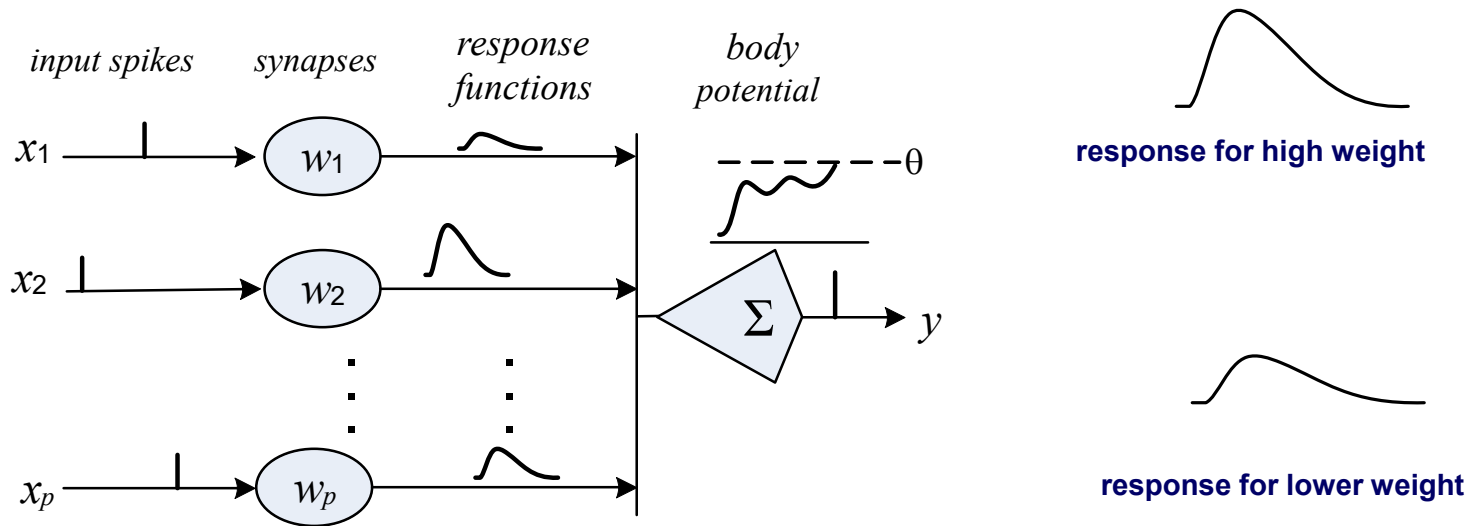
Simplifies threshold comparison (check only MSB)

# LIF Digital Implementation



# Excitatory Neuron Model

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



- A volley of spikes is applied at inputs
- At each input's synapse, the spike produces a response function
  - Synaptic weight ( $w_i$ ) determines amplitude: larger weight  $\Rightarrow$  higher amplitude
- Responses are summed linearly at neuron body
  - Models the neuron's membrane (body) potential
- Fire output spike if/when potential exceeds threshold value ( $\theta$ )

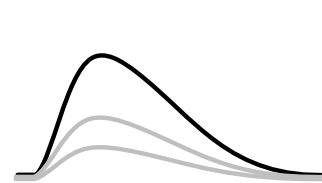
# Response Functions

- A wide variety of response functions are used
  - Researchers' choice
  - Synaptic weight determines amplitude

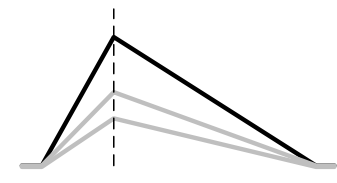
*Biexponential* – most realistic

*Non-leaky* often used

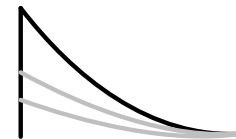
*Compound synapse* has skewed peak amplitude



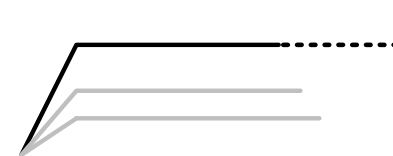
Biexponential



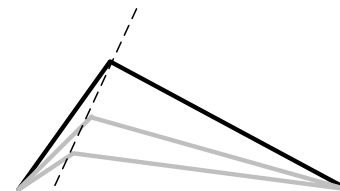
Piecewise linear  
biexponential approximation



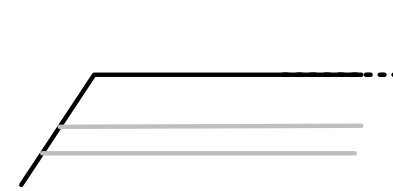
Original LIF  
(Stein 1965)



Non-leaky



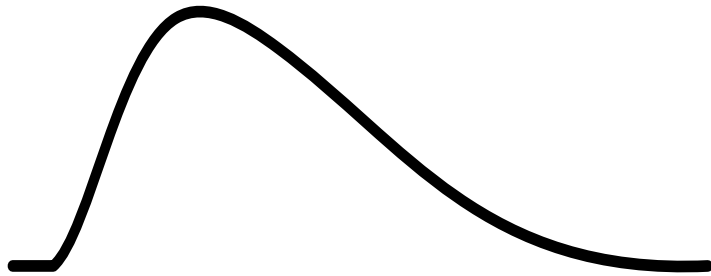
Compound synapse



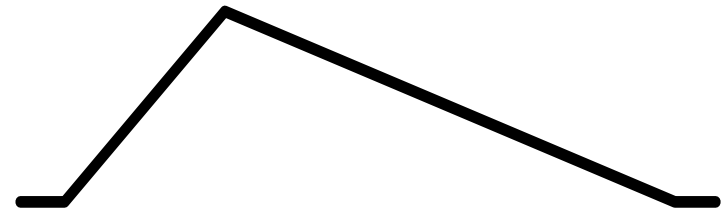
Compound synapse  
Non-leaky

# Which Is the Approximation?

---



a) Biexponential spike response



b) Piecewise linear spike response

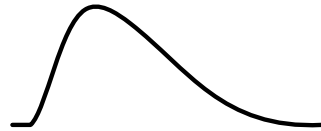
# A Hierarchy of Spiking Neuron Models

---

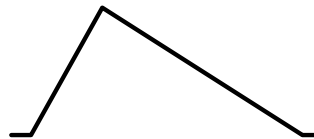
Electrical Ckt DiffEqs  
(e.g. Hodgkin-Huxley)



Biexponential



Piecewise linear



Linear No-Leak



Step No-Leak



*SRM0  
Models*

# Nomenclature: LIF Neuron Models

---

- ❑ Leaky Integrate-and-Fire
- ❑ Often mentioned in the literature
- ❑ A generic term for a broad class of neuron models
  - Many neuron models *leak, integrate, and fire*
- ❑ Most SRM0 models are LIF
- ❑ When an LIF model is not SRM0, it is due to the leakage model

# Terminology

---

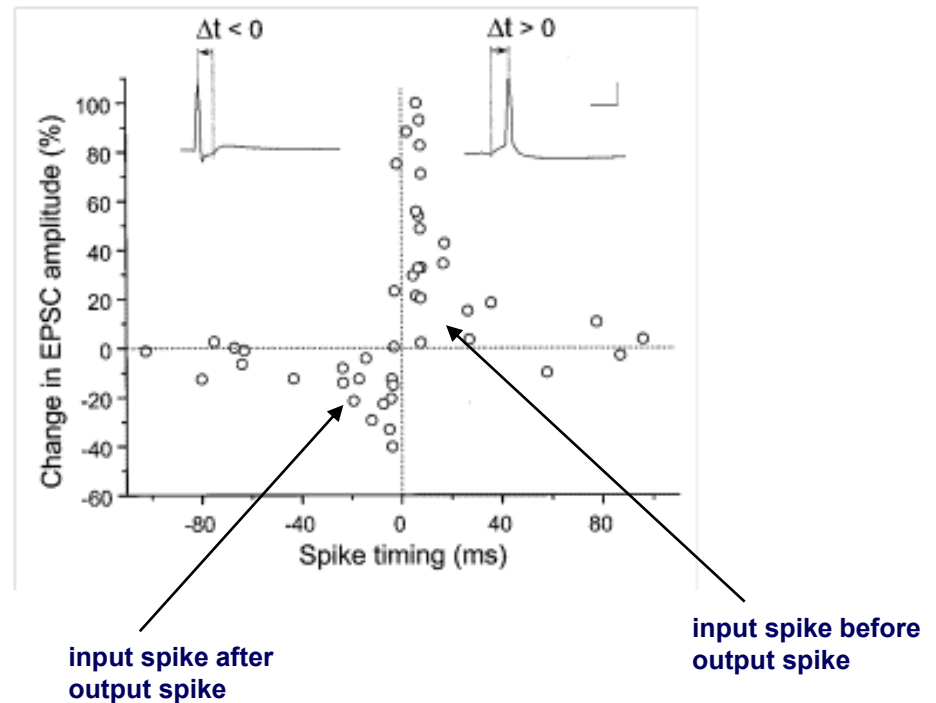
- Physical connectivity
  - *A bundle is composed of multiple lines*
- Information connectivity
  - *A volley consists of multiple spikes*

# Neuron Model: Synaptic Plasticity

- ❑ Synaptic weights determine a neuron's overall function
- ❑ Hence, training synaptic weights is an essential part of the paradigm
- ❑ Bi and Poo (1998) provide experimental support

- *In vitro* rat hippocampal cells
- Probe pairs of synaptically connected excitatory neurons
- Repetitively stimulate upstream neuron at low frequency
- Measure downstream neuron response (amplitude and spike time)
- Plot change in response amplitude vs.  $\Delta t$  (time difference between output and input spikes).  $\Delta t$  may be pos. or neg.

Additional Refs: [Markram et al. 1997] [Morrison et al. 2008]



from Bi and Poo 1998

# STDP

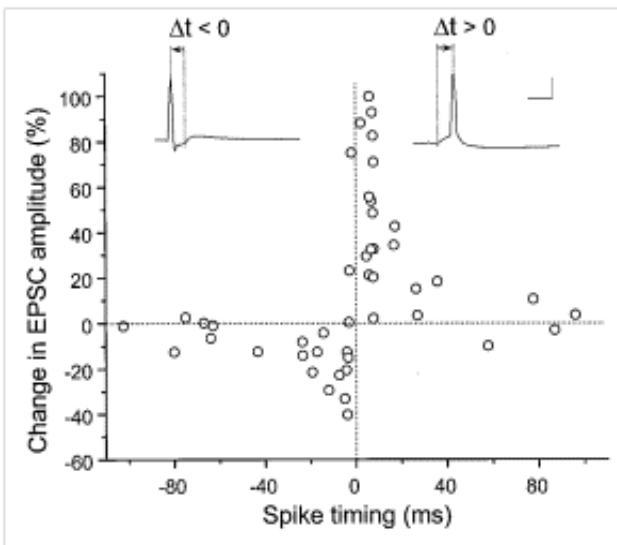
## □ Spike Timing Dependent Plasticity :

- Compare relative input and output spike times

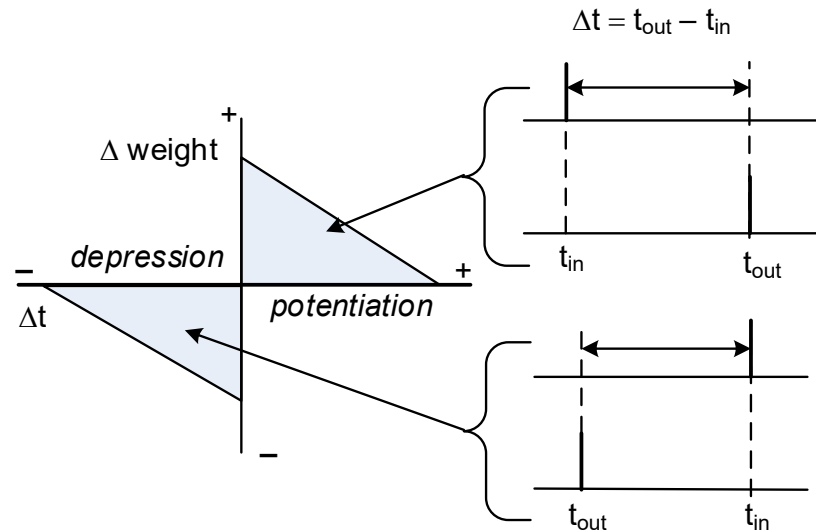
*Input spike time < output spike time: associated synapse is potentiated*

*Input spike time  $\geq$  output spike time: associated input synapse is depressed.*

- Synaptic weights stabilize in a way that captures a neuron's observed input patterns



from Bi and Poo 1998



## □ Training is *unsupervised*, *localized*, and *emergent*

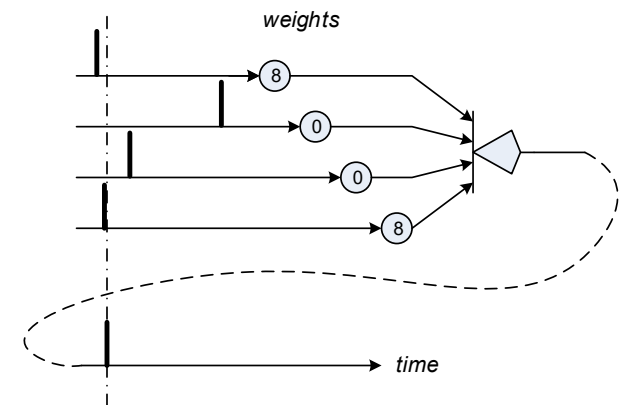
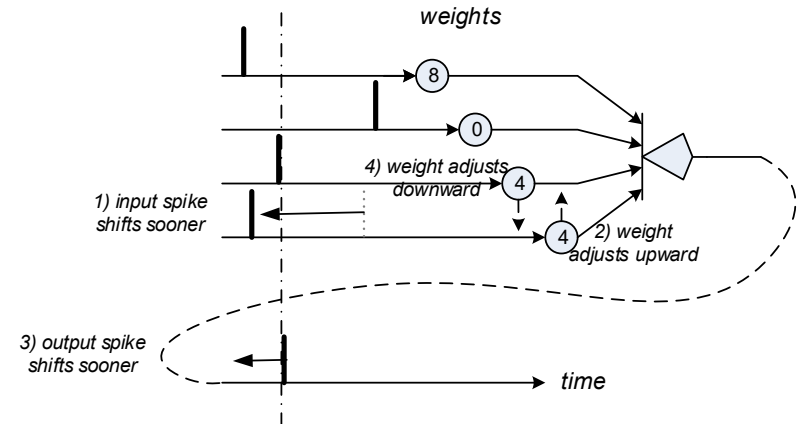
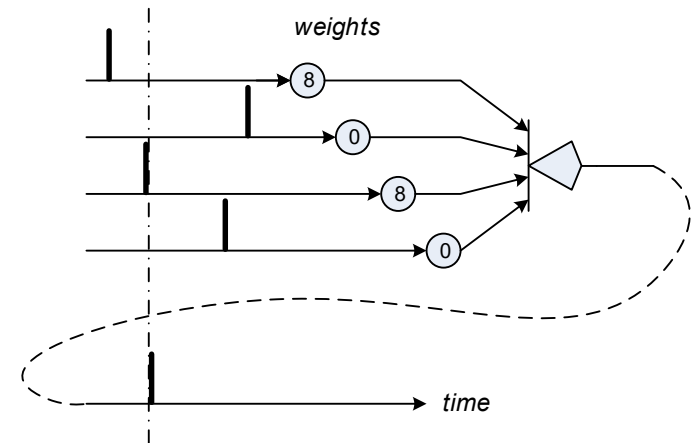
- This is a major difference wrt conventional error back-propagation

# STDP In Action

from Guyonneau, VanRullen, and Thorpe (2005)

## *Paraphrased*

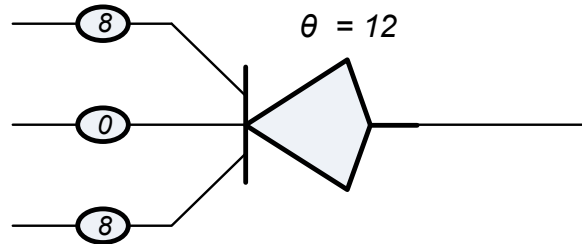
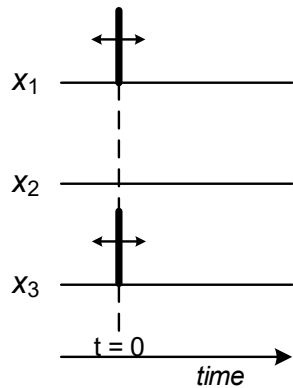
- For a given input pattern, input spikes elicit an output spike, triggering the STDP rule.
- Synapses carrying input spikes just preceding the output spike are potentiated, while later ones are weakened.
- The next time the input pattern is applied, the firing threshold will be reached sooner, which implies the output spike will occur sooner.
- Consequently, the STDP process, while depressing some synapses it had previously potentiated, will now reinforce different synapses carrying even earlier spikes.
- By iteration, when the same input spike pattern is repeated, the output spike time will stabilize at a minimal value while the earlier synapses become fully potentiated and later ones fully depressed.



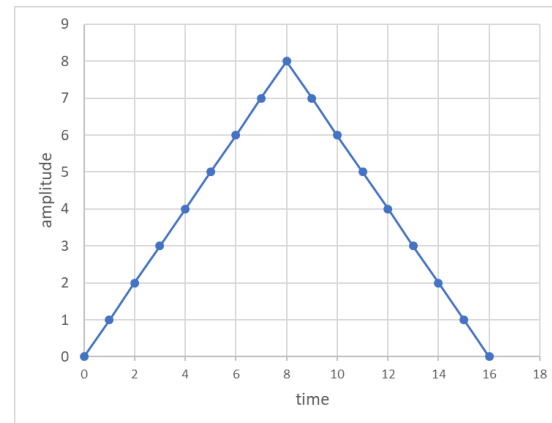
# Example

- ❑ Three input neuron
- ❑ Train for a single pattern
- ❑ Apply evaluation inputs

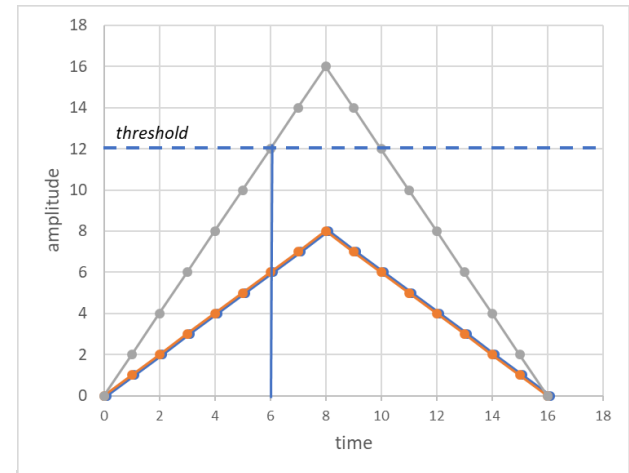
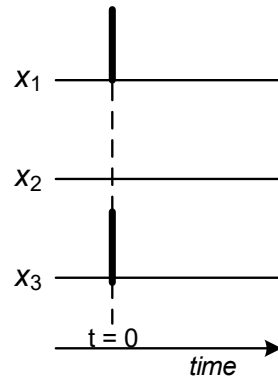
*typical training pattern*



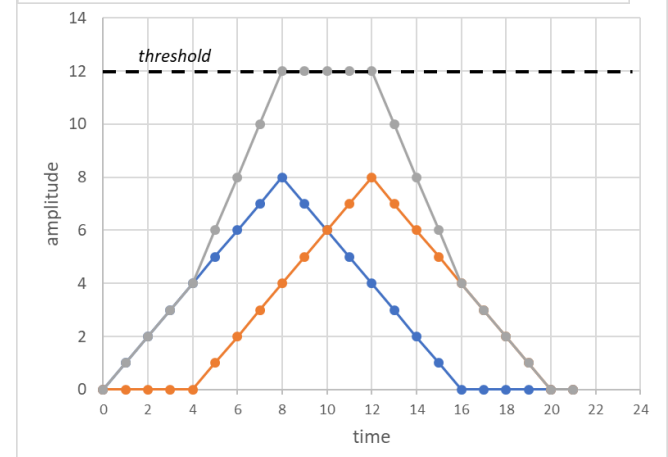
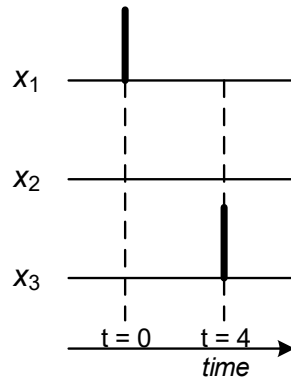
**Response function  
for  
Weight = 8**



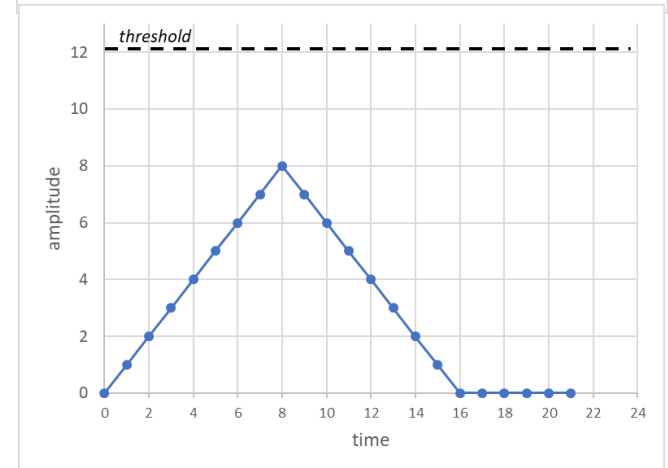
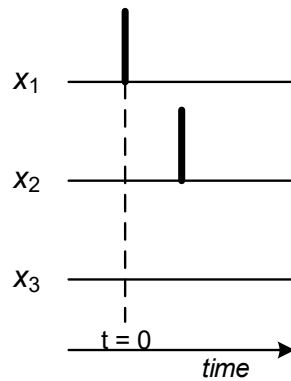
**input matches training pattern**  
**ideal alignment**  
*output spike time = 6*



**input matches training pattern**  
**some miss-alignment**  
*output spike time = 8*

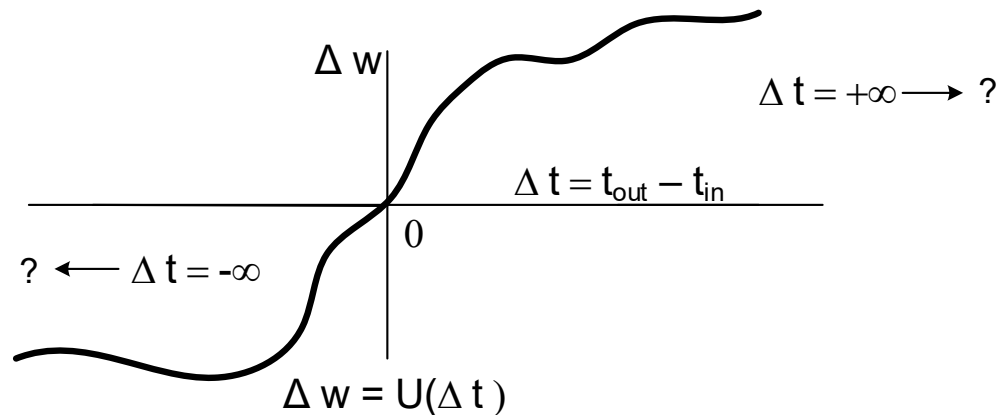


**input miss-match**  
**no output spike**



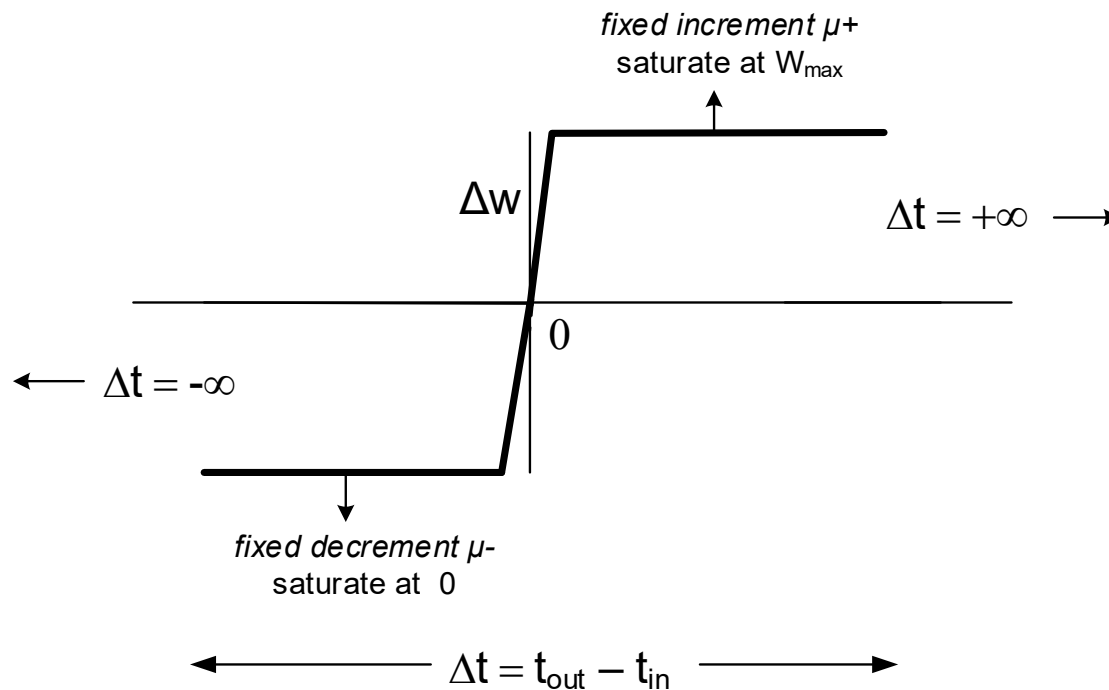
# Excitatory Update Functions

- At each synapse:
  - In response to difference between input and output spike times:  $\Delta t = t_{\text{out}} - t_{\text{in}}$
  - Current synaptic weight,  $w$ , is updated by adding  $\Delta w$
- $\Delta w = U(\Delta t)$ 
  - Update function,  $U$ , determined by:
    - model-dependent input parameters
    - current weight



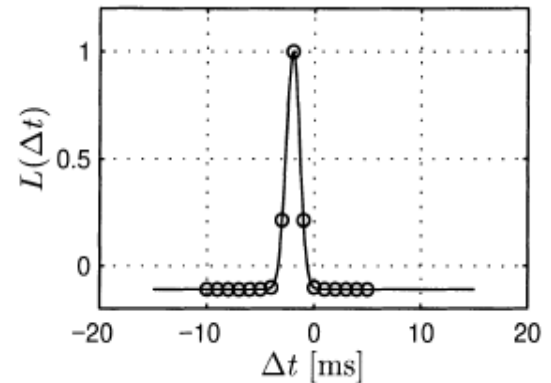
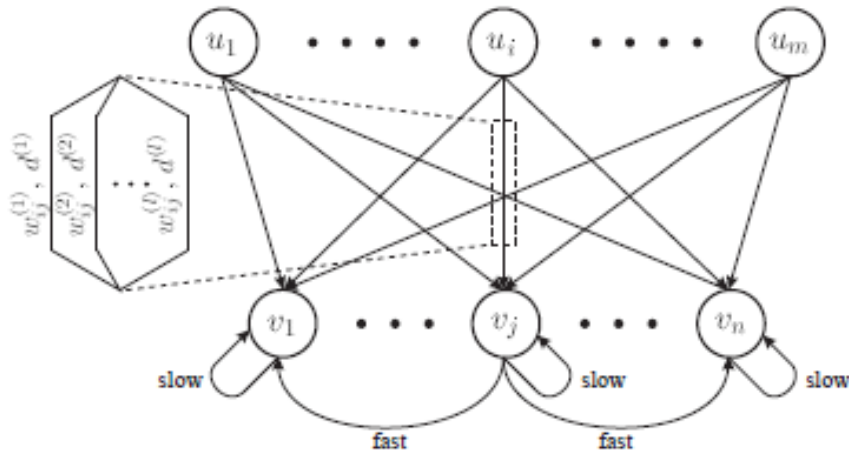
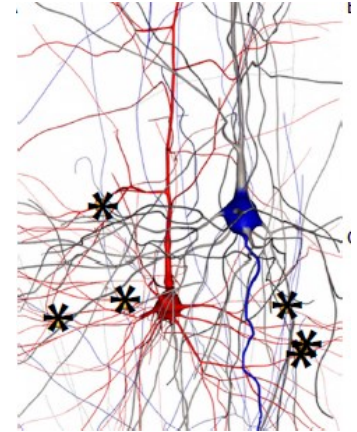
# A Simple STDP Model

- Model parameters: *increment* and *decrement* constants ( $\mu+$  and  $\mu-$ )
  - U is independent of current weight
  - Saturate at 0 and  $W_{\max}$
- Also must account for single spike cases
  - Very important part of overall STDP function



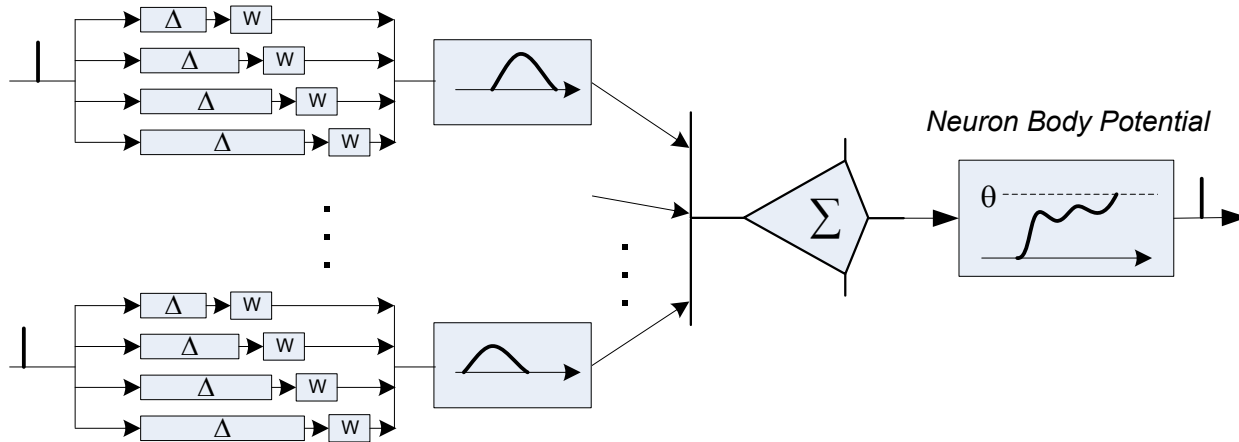
# RBF Neurons

- Observe that multiple synapses connect same two neurons
  - Treat the collection as tapped delay line
  - STDP training to isolate a single delay via a non-zero weight
- Act *sort of* like classical Radial Basis Functions (RBF)
- Natschläger and Ruf (1998) based on Hopfield (1995)
  - Also see Bohte et al. (2002)



**STDP function**

# Compound Synapses



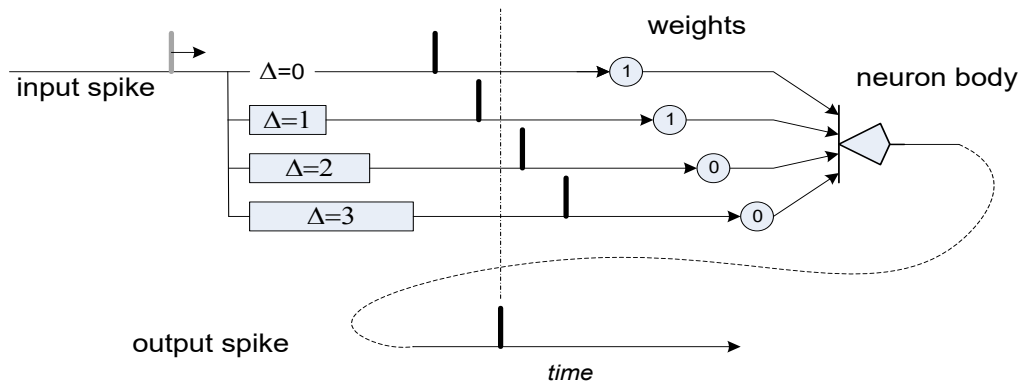
- ❑ Uses basic SRM0 model, *except*
- ❑ Multiple paths from one neuron to another
  - *Axon drives multiple dendrites belonging to the same downstream neuron*
- ❑ Each path has its own delay and synaptic weight
- ❑ STDP training yields computationally powerful response functions

# Training Compound Synapses

## □ STDP:

- Synapses associated with later arriving spikes are depressed
- Synapses associated with earlier arriving spikes are potentiated
- On a given line, this causes the weight to approach an “average” point with respect to all the input spike times

- When applied to a compound synapse – multiple delay paths – the weights reflect average relative input spike times.

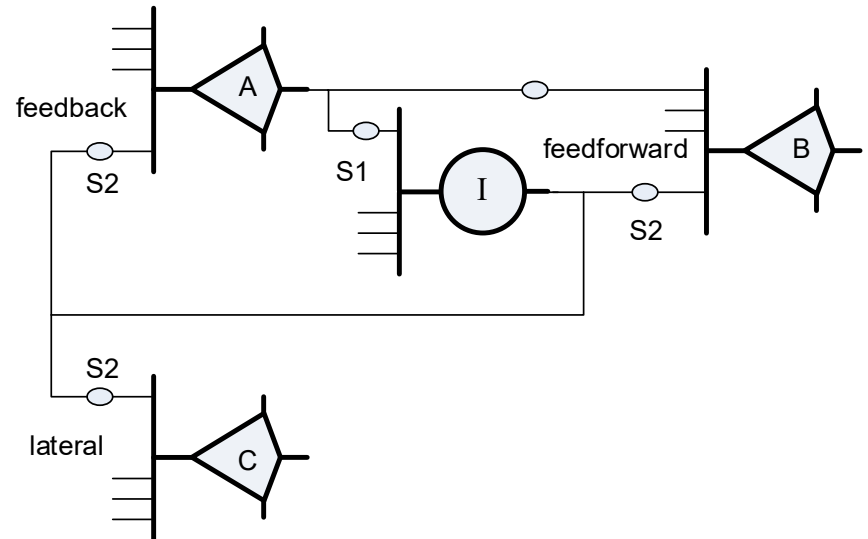


# **Low Level Abstractions**

## ***Modeling Inhibition***

# Inhibition

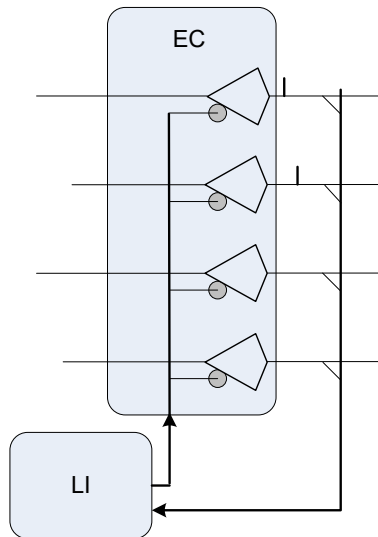
- Inhibitory neurons:
  - Sharpen/tune and reduce spike “clutter”  
Only first spike(s) are important
  - Provide localized moderating/throttling  
Saves energy
  - *Provide information filtering*
- Three types of inhibitory paths
  - feedforward ( $A \rightarrow I \rightarrow B$ )
  - lateral ( $A \rightarrow I \rightarrow C$ )
  - feedback ( $A \rightarrow I \rightarrow A$ )



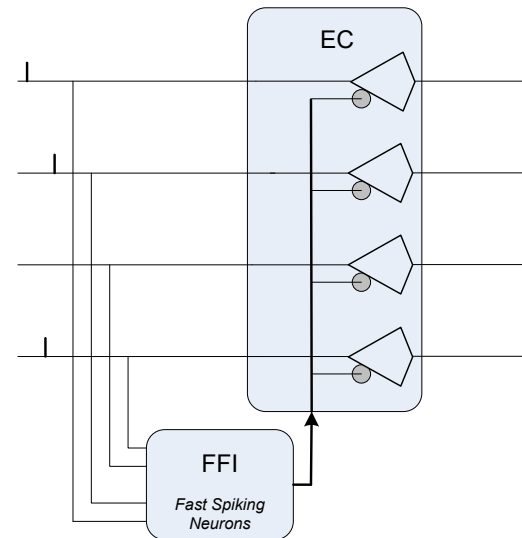
# Modeling Inhibition

- ❑ Model in bulk
  - w/ time-based parameters
- ❑ Our primary interest is feedforward nets
  - Consider only *lateral* and *feedforward* inhibition

EC: Excitatory Column



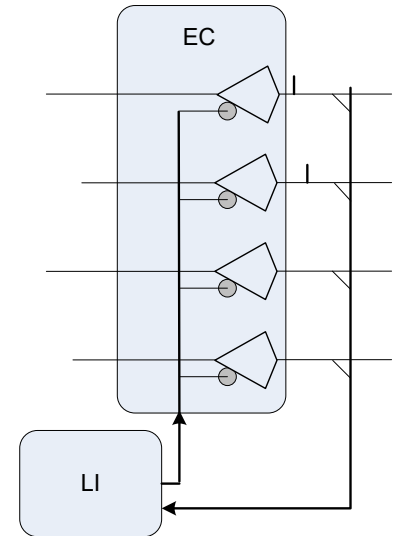
**lateral inhibition**



**feedforward inhibition**

# WTA Inhibition

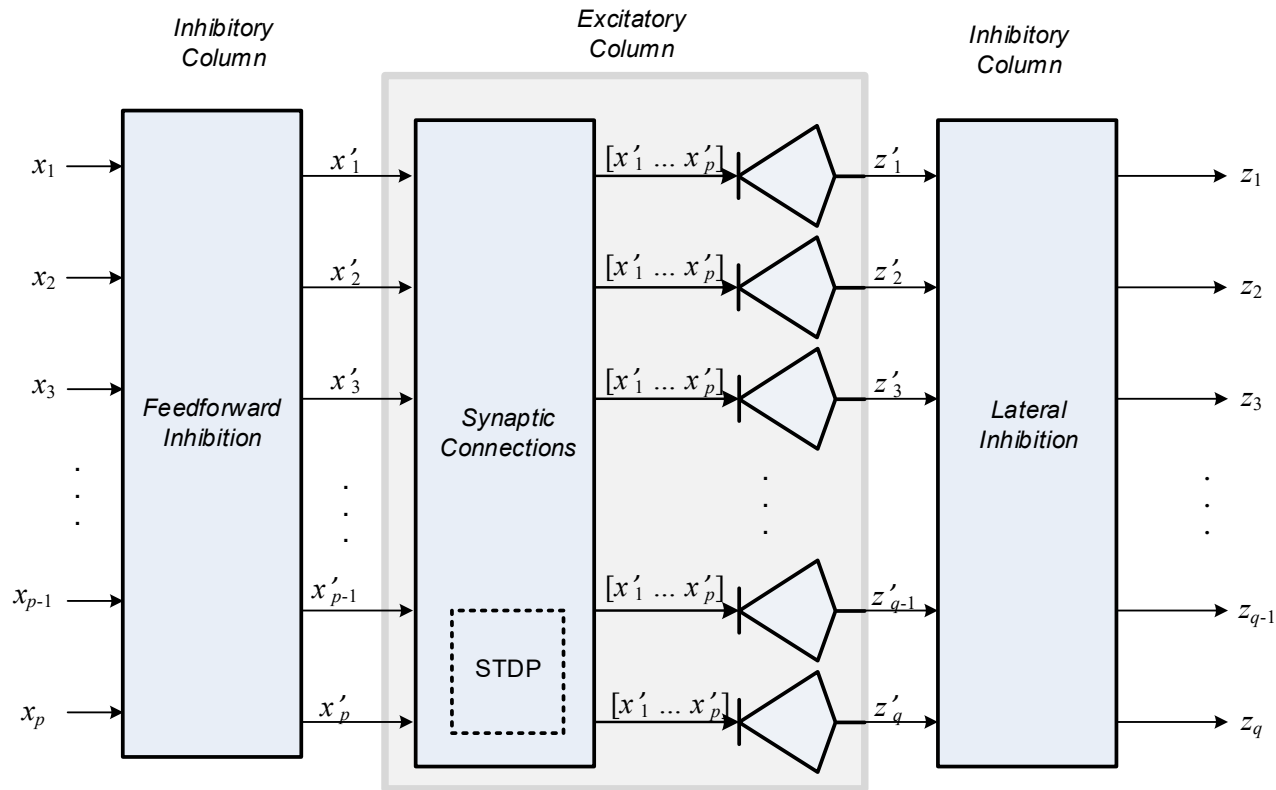
- ❑ Winner-Take-All
  - The first spike in a volley is the “winner”, and only the winning spike is allowed to pass through
  - A frequently-used form of inhibition
- ❑ Implement parameterized  $t$ - $k$ -WTA
  - Only spike occurring within time  $t$  of the first spike are retained
  - Of the remaining spikes, at most  $k$  of them are kept (others are inhibited)
- ❑ *Tie cases are very important*
  - After  $t$  inhibition is applied, more than  $k$  spikes may remain
  - Some options:
    - 1) Inhibit all  $k$  (*feedforward inhibition only*)
    - 2) Inhibit all but a pseudo-random selection of  $k$
    - 3) Inhibit all but a systematic selection of  $k$



# **Column Architecture**

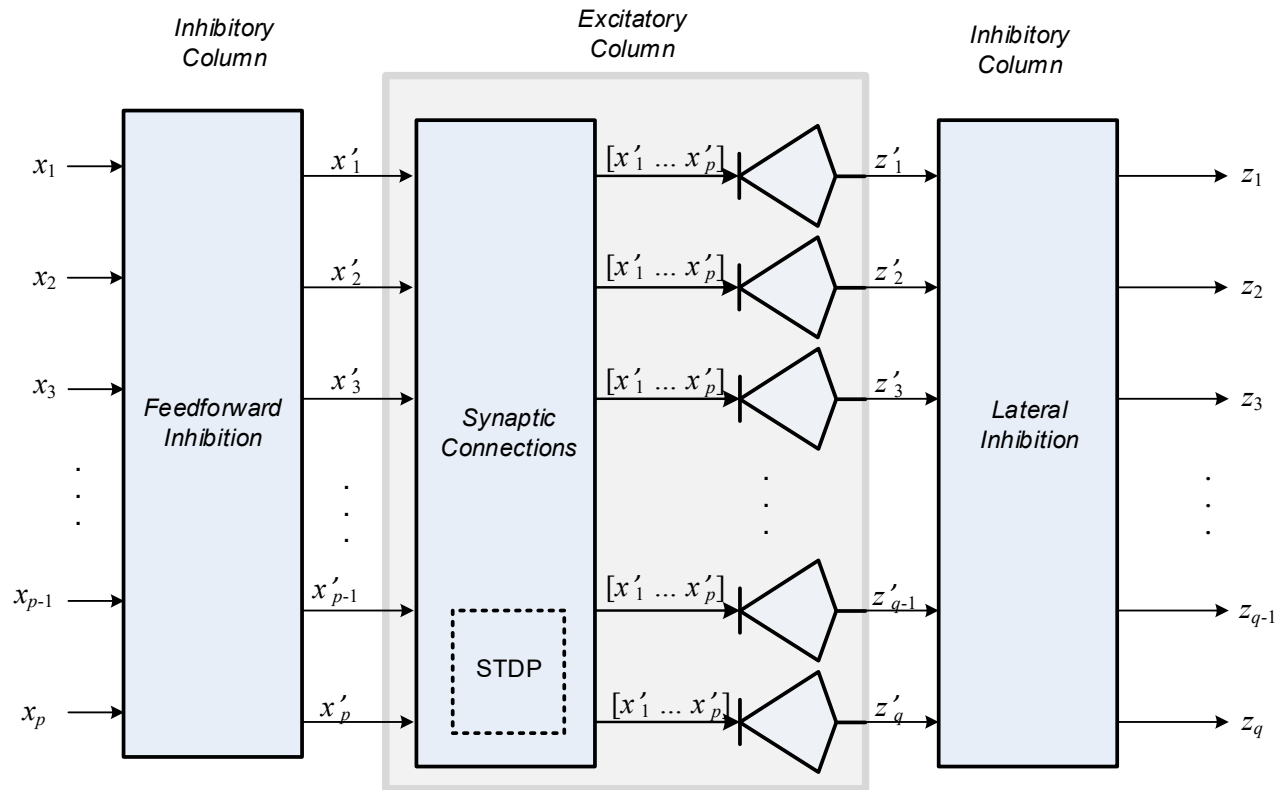
# Computational Column

- A *Computational Column (CC)* is an architecture layer composed of:
  - Parallel excitatory neurons that generate spikes
  - Bulk inhibition that removes spikes
- Essentially all proposed TNNs are composed of CCs that fit this general template
  - Including simple systems consisting of a single CC or excitatory neuron



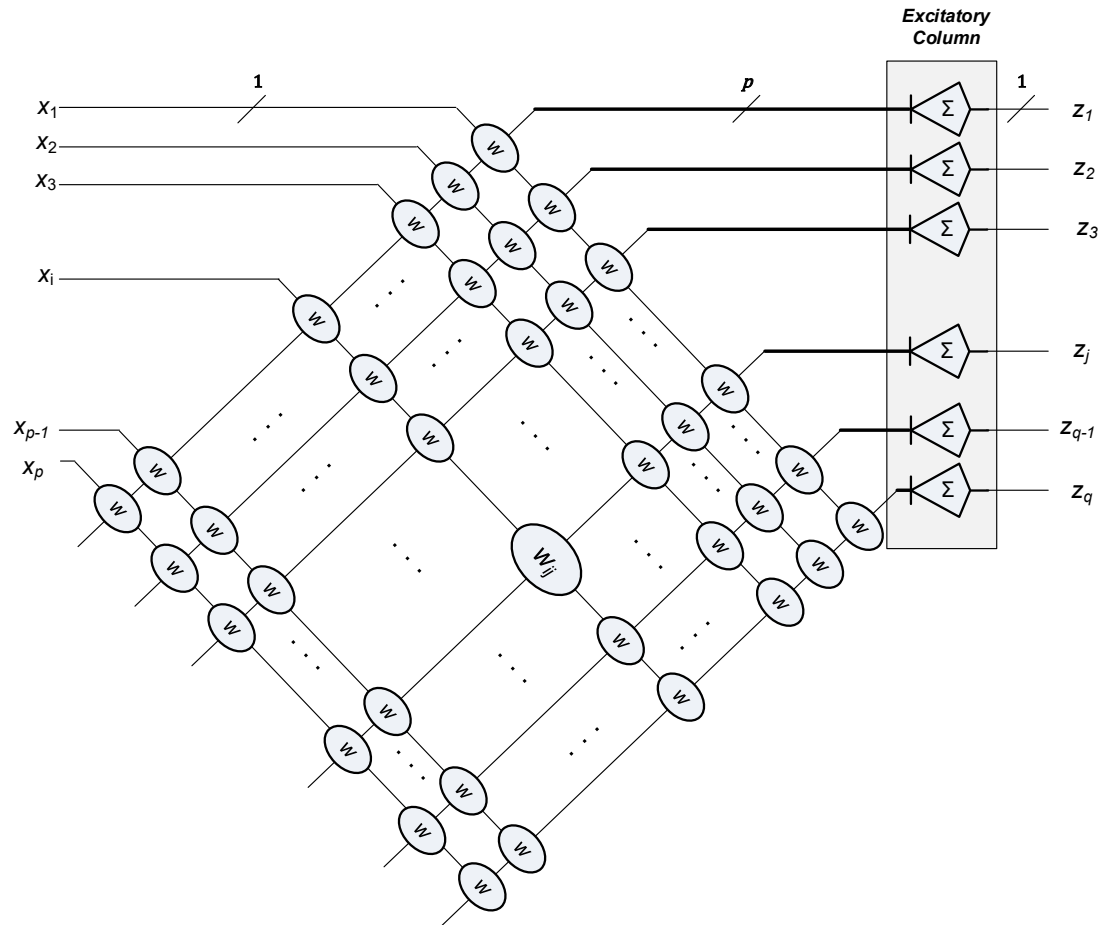
# Computational Column

- *Machine learning interpretation:* Each output is associated with a specific “feature”
  - In effect, the column is a set of feature maps
  - It produces a feature vector (volley) as an output
  - Inhibition performs a pooling-like function



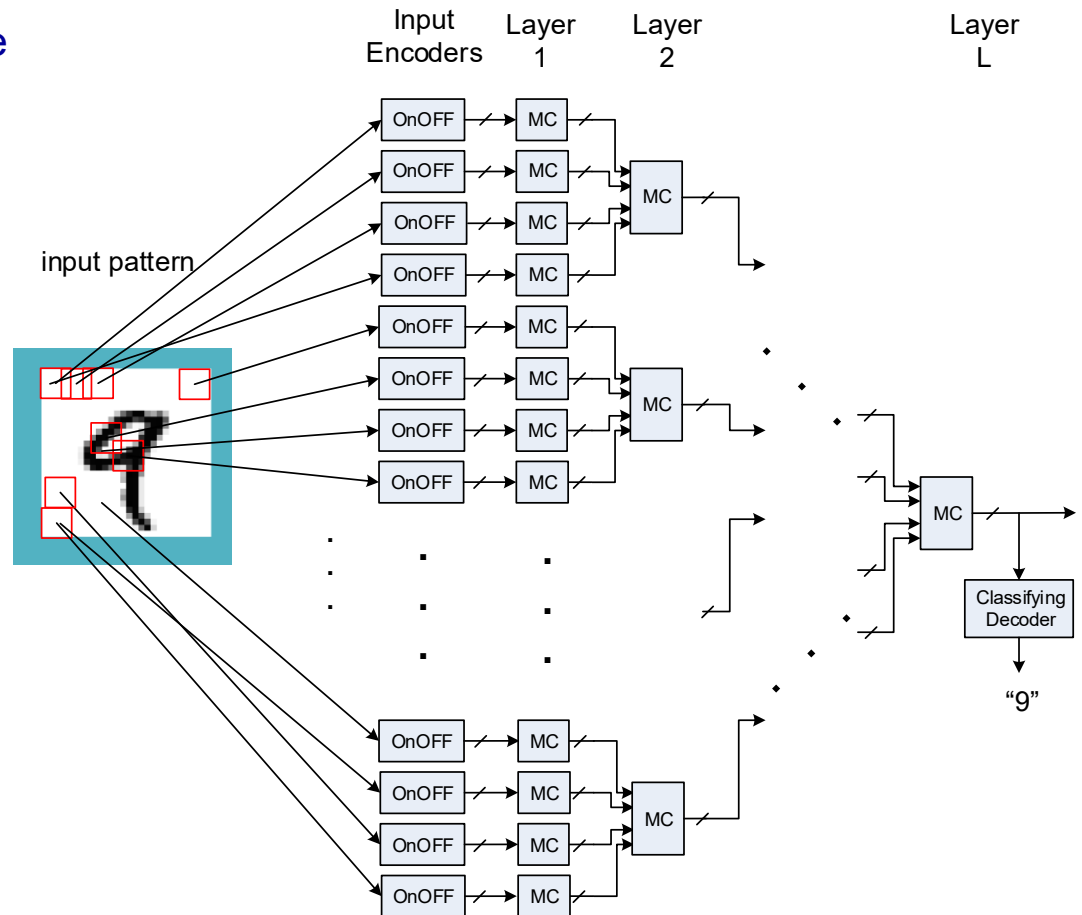
# Synaptic Connections

- ❑ Model for *dendritic* input structure
- ❑ Synaptic weights are associated with cross points
  - All weighted inputs  $x_1 \dots x_p$  are fed to every excitatory neuron
- ❑ May be fully connected or partial
  - *No-connect* is equivalent to zero weight



# Typical Network Architecture

- Input encoding
  - OnOff performs simple edge detection
  - Similar to biological method
- Multiple layers of CCs
  - Hierarchical
  - Wide variety of configurations and interconnection nets
- Output decoding
  - E.g., 1-to-1 classifier



***BREAK***

# Outline

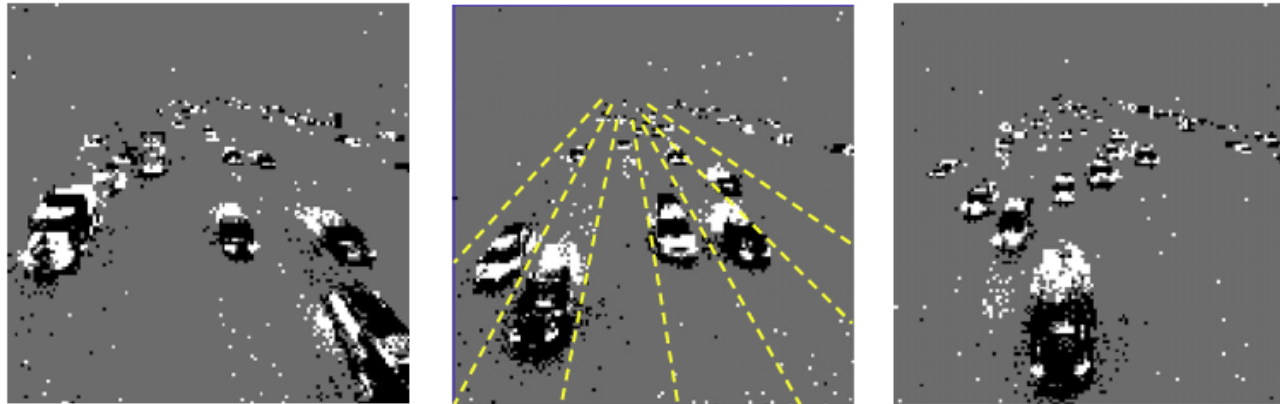
---

- ❑ Introductory Remarks
- ❑ Research Focus: Milestone TNN
  - Classification and Clustering
  - Neural Network Taxonomy
- ❑ Biological Background
  - Overview
  - Neural Information Coding
  - Sensory Encoding
- ❑ Fundamental Hypotheses
  - Theory: Time as a Resource
- ❑ Low Level Abstractions
  - Excitatory Neuron Models
  - Modeling Inhibition
- ❑ Column Architecture
- ❑ Case Studies
  - [Recent TNNs](#)
- ❑ Space-Time Theory and Algebra
- ❑ Implementations
  - [Indirect Implementations: Simulation](#)
  - [Direct Implementations: Neuromorphic Circuits](#)
- ❑ Concluding Remarks

**Case Studies**  
**(All directed at Milestone TNN)**

# Case Study 1: Identifying Moving Cars

- Bichler et al. (2012)
  - Count cars over 6 traffic lanes; 210 freeway in Pasadena, CA



from Bichler et al. (2012)

- Feed-forward, multilayer, unsupervised learning w/ STDP
- Input stream: silicon retina with Address-Event Representation (AER)
  - Serial link transmitting <pixel, spike time> pairs
  - A basic sparse vector representation
- Encoder: emits a spike when pixel intensity changes by at least some positive or negative delta
  - Two synapses per pixel: ON(+) and OFF(-)

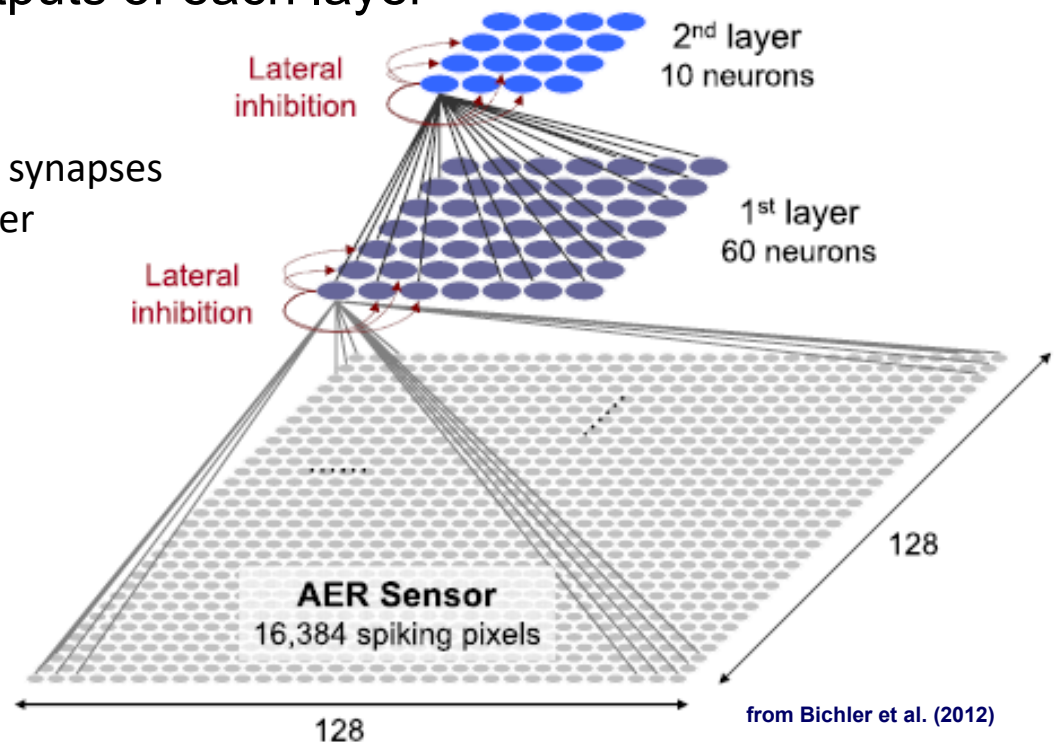
# System Architecture

- Hierarchical structure of excitatory neurons
  - Layer 1: 60 neurons each taking all 16K pixels (x2) as input  
Essentially 60 feature maps over the entire image
  - Layer 2: 10 neurons + simple classification (not shown)
- WTA lateral inhibition at outputs of each layer

70 total neurons

$2 \times 128 \times 128 \times 60 + 60 \times 10 = 1,966,680$  total synapses

Lateral inhibition across all neurons in layer

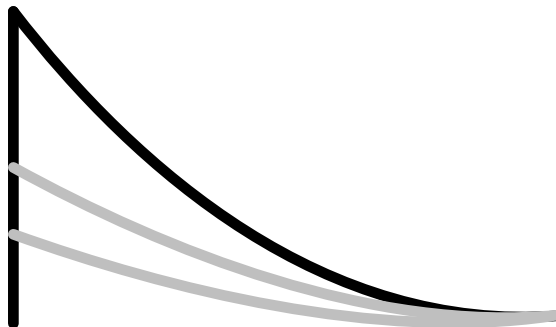


# Excitatory Neuron Model

---

- ❑ SRM0 neuron
- ❑ Response Function
  - Stein LIF (1965)
  - Step up, exponential decay down
  - Amenable to event-driven simulation

Response functions need only be summed in response to an input spike



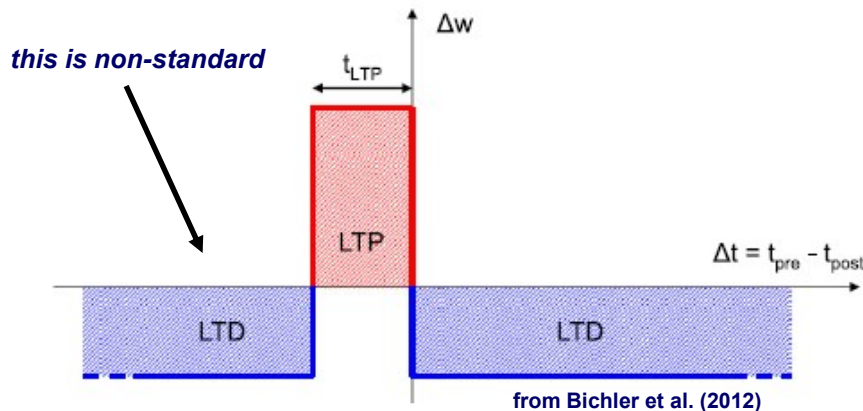
# Excitatory Neuron Model

- Training
  - *Unsupervised*
  - STDP learning rule
  - Does not fit standard model

**Step 1: determine *mode*: long term potentiation (LTP) or long term depression (LTD)**

**Step 2: determine the change of weight for the selected mode**

simple additive update, either increment by  $\alpha_+$  or decrement by  $\alpha_-$



# STDP

## □ General STDP functions

$$\Delta w_+ = \alpha_+ \cdot \exp\left(-\beta_+ \cdot \frac{w - w_{\min}}{w_{\max} - w_{\min}}\right)$$

$$\Delta w_- = \alpha_- \cdot \exp\left(-\beta_- \cdot \frac{w_{\max} - w}{w_{\max} - w_{\min}}\right)$$

## □ In this research, $\beta$ s are 0

$$\Delta w_+ = \alpha_+$$

$$\Delta w_- = \alpha_-$$

## □ Synaptic parameters

- $w_{\min}$ ,  $w_{\max}$ , and  $w_{\text{init}}$  are pseudo-random

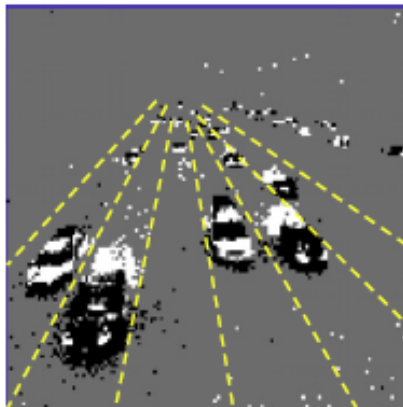
Parameter	Mean	Std. dev.	Description
$w_{\min}$	1	0.2	Minimum weight (normalized).
$w_{\max}$	1000	200	Maximum weight.
$w_{\text{init}}$	800	160	Initial weight.
$\alpha_+$	100	20	Weight increment.
$\alpha_-$	50	10	Weight decrement.
$\beta_+$	0	0	Increment damping factor.
$\beta_-$	0	0	Decrement damping factor.

from Bichler et al. (2012)

# Inhibition

---

- Bulk WTA Lateral Inhibition
  - When a neuron spikes, it inhibits all others from spiking for time  $T_{\text{inhibit}}$
  - Divides processing into softly synchronized “chunks” aka “volleys”
  - (This may be a good model for synchronizing inhibitory oscillations)
- Note -- *there is no feedback in the processing path*
  - Although object is moving, network actually processes static “snapshots”
  - Objects’ leading and trailing edges trigger different On and Off spikes



- During training: Inhibition causes pseudo-randomly initialized neurons to converge to different final synaptic states
  - Consequently, each neuron identifies a different prominent input feature

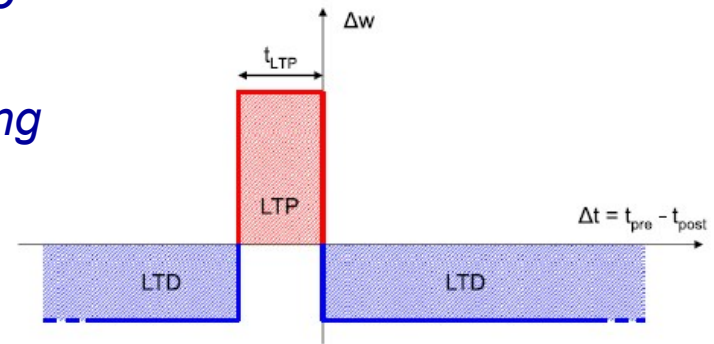
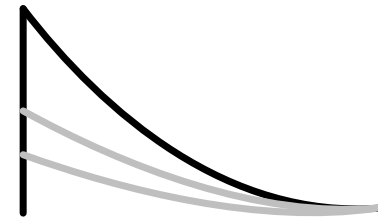
# Training Process

---

- ❑ To properly train multiple neurons w/ the same inputs, there must be some differences in connectivity and/or synaptic parameters
  - Otherwise all the neurons will train to identical synaptic weights
  - Case study architecture: initialize synapses to pseudo-random values
    - Coupled w/ WTA inhibition, this causes different neurons to converge to synaptic weights
  - Another option: use pseudo-random connectivity so that each neuron sees a different set of input spikes
- ❑ Two training methods
  - Global – train multiple layers at once
  - Layer-by-layer – train 1<sup>st</sup> layer, then turn off lateral inhibition and STDP, and train 2<sup>nd</sup> layer, etc.
  - *Layer-by-layer works better*
- ❑ Unsupervised Training Regime: Apply full AER input data set 8 times
  - Training time: 10 minutes per layer on a PC

# Neuron Parameters

- Parameters determined by genetic algorithm
  - Using meta-data (labels)
  - So... selecting neuron parameters is *de facto* supervised
  - *Using meta data for neuron parameter training is likely to be a feature of most TNN models*



from Bichler et al. (2012)

Parameter	Global learning		Layer-by-layer learning	
	1st layer	2nd layer	1st layer	2nd layer
$I_{thres}$	500 000	1500	1 060 000	2240
$T_{LTP}$	12 ms	300 ms	14.7 ms	46.5 ms
$T_{refrac}$	300 ms	250 ms	517 ms	470 ms
$T_{inhibit}$	50 ms	100 ms	10.2 ms	182 ms
$\tau_{leak}$	450 ms	300 ms	187 ms	477 ms
Recog. rate	47%–100%/lane		98% overall	

# Performance (Accuracy)

- ❑ Output spike patterns self-select due to pseudo-random initial state and unsupervised training
- ❑ To determine an accuracy measure:
  - Produce hand-labeled “correct” spike trains
  - Using these, the most accurate neuron for each labeled spike train is identified
  - This neuron’s accuracy determines the network’s accuracy
  - Layer-by-layer big network: 207 cars, 4 missed, 9 false positive

Parameter	Global learning		Layer-by-layer learning	
	1st layer	2nd layer	1st layer	2nd layer
$I_{\text{thres}}$	500 000	1500	1 060 000	2240
$T_{\text{LTP}}$	12 ms	300 ms	14.7 ms	46.5 ms
$T_{\text{refrac}}$	300 ms	250 ms	517 ms	470 ms
$T_{\text{inhibit}}$	50 ms	100 ms	10.2 ms	182 ms
$\tau_{\text{leak}}$	450 ms	300 ms	187 ms	477 ms
Recog. rate	47%–100%/lane		<b>98% overall</b>	

# Advanced System Architecture

- ❑ To reduce cost significantly
  - Layer 1 divided into squares (groups) – in a receptive-field-like manner
  - Layer 2: 10 neurons + simple classification (not shown)

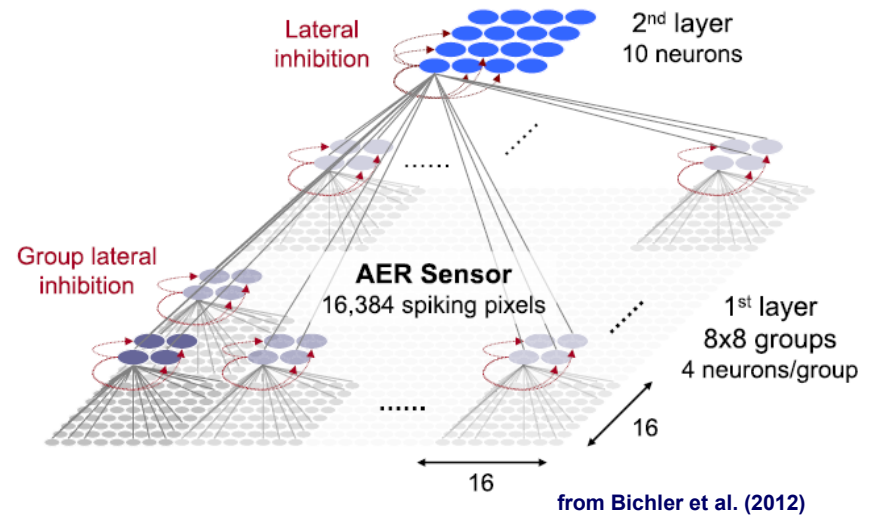
8 by 8 squares of  $16 \times 16$  pixels each

With 4 neurons per square group

$8 \times 8 \times 4 + 10 = 266$  total neurons

$16 \times 16 \times 2 \times 8 \times 8 \times 4 + 8 \times 8 \times 10 = 131,712$  synapses

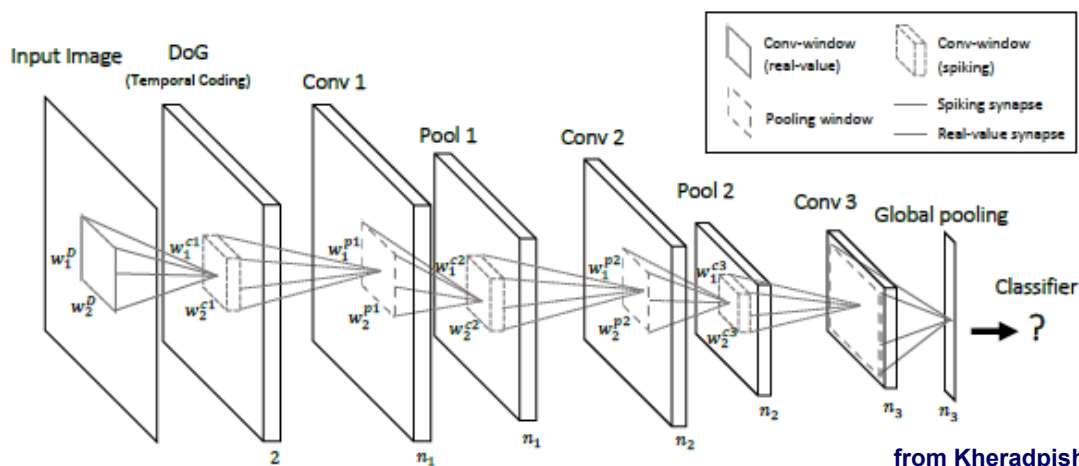
Lateral inhibition is localized to the group



Parameter	1st layer	2nd layer
$I_{thres}$	60 000	2000
$T_{LTP}$	6 ms	70 ms
$T_{refrac}$	700 ms	300 ms
$T_{inhibit}$	100 ms	450 ms
$\tau_{leak}$	100 ms	250 ms
Recog. rate	>95% overall	

# Case Study 2: Clustering/Classification

- Kheradpisheh, Ganjtabesh, Thorpe, and Masquelier (2016)
- Classifier w/ largely unsupervised training
  - This summary is focused primarily on MNIST implementation



*“Although the architecture of DCNNs is somehow inspired by the primate’s visual system (a hierarchy of computational layers with gradually increasing receptive fields), they totally neglect the actual neural processing and learning mechanisms in the cortex.”* DCNN = Deep Convolutional Neural Network

# System Architecture

---

- ❑ Uses DCNN terminology
  - “Convolutional layers” and “pooling”
  - Architecture is a hybrid of sorts between SNNs and DCNNs
- ❑ Hierarchical structure
  - 5 x 5 convolution windows – total number of windows not given
  - Layer 1: 30 neuronal maps – i.e., a *column* consisting of 30 excitatory neurons
    - All instances of same feature have same weights (borrowed from DCNNs)
  - Max-pooling -- i.e., a WTA-inhibition layer
    - Across same feature in multiple columns (borrowed from DCNNs)
  - Layer 2: 100 neuronal maps
    - MNIST is reduced down to 100 clusters (a lot of clusters)
    - (note: *Increasing numbers of maps w/ increasing layers is not a good thing*)
  - Global pooling prior to classification uses max body potential, not spike time
    - (note: *Non-causal*)

# Encoding

---

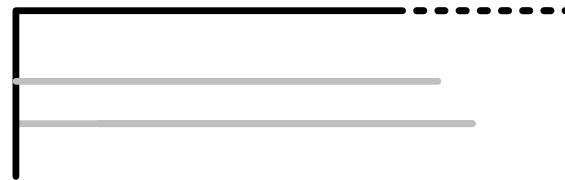
- ❑ Encoding via Difference of Gaussian (DoG) filters to detect contrasts (edges)
  - On Center and Off Center maps
  - One sensitive to positive contrast and one sensitive to negative contrast
  - Stronger contrast means earlier spike
  - At most one of On and Off Center maps generates a spike
- ❑ Example of DoG filtering from Wikipedia:



# Excitatory Neuron Model

---

- ❑ SRM0 neuron
- ❑ Response Function
  - Non-leaky
  - Step up, then flat



- ❑ Training
  - *Unsupervised*
  - Additive STDP learning rule with gradual saturation
  - Only *sign* of time difference taken into account
  - Lower weights get larger increments (or decrements)

$$\begin{cases} \Delta w_{ij} = a^+ w_{ij} (1 - w_{ij}), & \text{if } t_j - t_i \leq 0, \\ \Delta w_{ij} = a^- w_{ij} (1 - w_{ij}), & \text{if } t_j - t_i > 0, \end{cases}$$

# Overall STDP Strategy

---

- ❑ In convolutional layer, multiple maps
  - Each map appears multiple times w/ different inputs
- ❑ Intra-Map competition
  - WTA among all replicas of same map
  - Only the first is updated via STDP; weights copied to the others
- ❑ Local Inter-Map competition
  - WTA among different maps in local neighborhood
  - Only the winner updates
- ❑ Initial weights are pseudo-random (.8 w/ std. dev. .2)
  - So neurons converge to different features

# Classification Layer

---

- ❑ Last neuron layer uses global pooling
  - Use highest final membrane potential
  - *Not spike times*
- ❑ Feed into conventional trained classifier (Support Vector Machine)
  - Note: for MNIST 100 neuronal maps in last layer
- ❑ Question: how much of the work is actually done in the TNN?
  - Edge detecting front-end does some of the work
  - SVM over 100 maps in last layer does some of the work
  - In between is the TNN
- ❑ Weight sharing is implausible
  - Not just an optimization feature...
  - Required for max-pooling method
- ❑ *This research exposes an inevitable tension between biological plausibility and the urge to compete w/ conventional machine learning approaches*

# MNIST Performance

- ❑ Includes comparison with other neural networks
- ❑ For a TNN, the performance is extremely good
  - For MNIST, it is probably state-of-the-art for TNNs
- ❑ Low activity
  - Only 600 total spikes per image (*low energy*)
  - At most 30 time steps per image (*low computational latency*)
- ❑ Summary: fast, efficient, unsupervised training

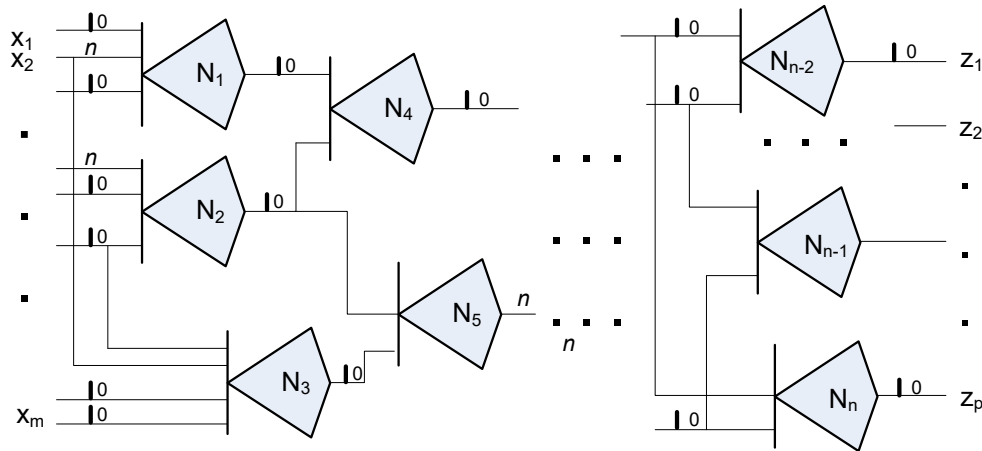
Table 2: Recognition accuracies of the proposed SDNN and some other SNNs over the MNIST dataset.

Architecture	Neural coding	Learning-type	Learning-rule	Accuracy (%)
Dendritic neurons [48]	Rate-based	Supervised	Morphology learning	90.3
Convolutional SNN [37]	Spike-based	Supervised	Tempotron rule	91.3
Two layer network [36]	Spike-based	Unsupervised	STDP	93.5
Spiking RBM [49]	Rate-based	Supervised	Contrastive divergence	94.1
Two layer network [38]	Spike-based	Unsupervised	STDP	95.0
Convolutional SNN [50]	Rate-based	Supervised	Back-propagation	<b>99.1</b>
Proposed SDNN	Spike-based	Unsupervised	STDP	98.4

**Note:** references point to a variety of SNNs

# Bonus Case Study: Esser et al. 2016 (IBM TrueNorth)

- Recent convolutional network supported by IBM TrueNorth
  - TrueNorth is a *platform*, not a computing model
- Uses 0/1 Perceptrons: {0/1 inputs, range of weights, threshold, 0/1 output}



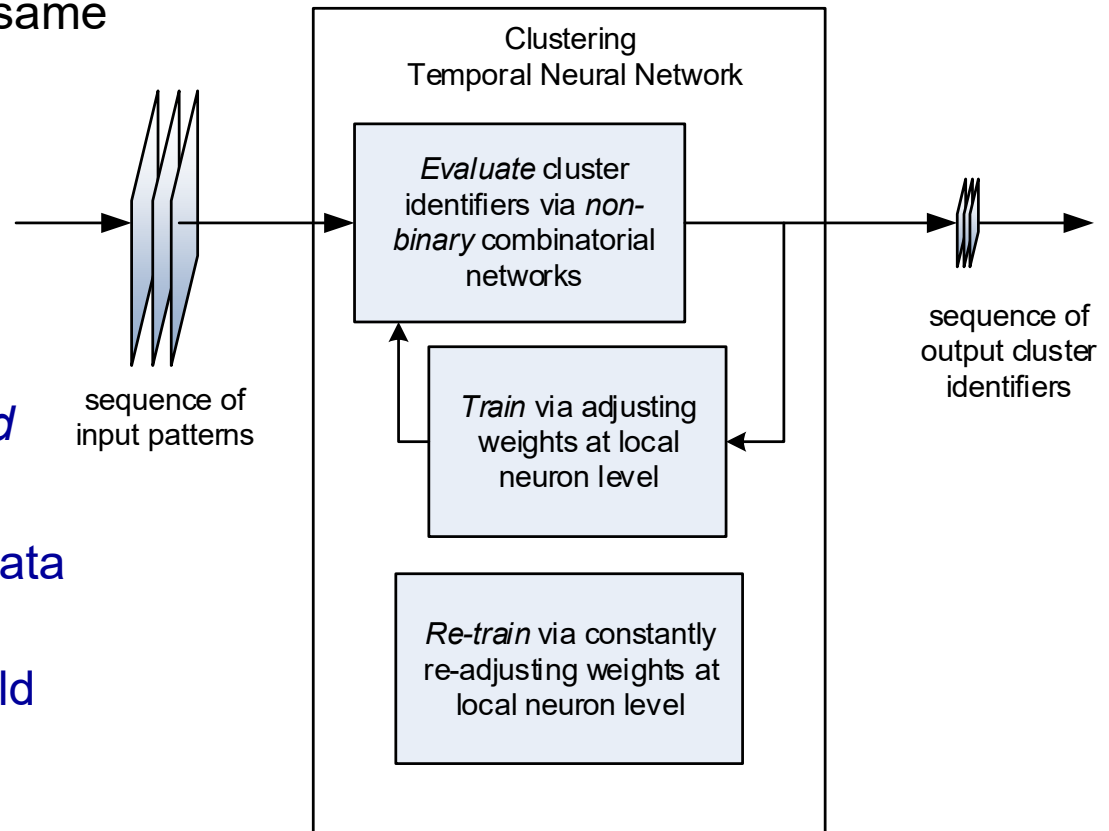
- A special case TNN, where all temporal events happen at time  $t = 0$

# **Prototype TNN Architecture**

*currently being developed by the speaker*

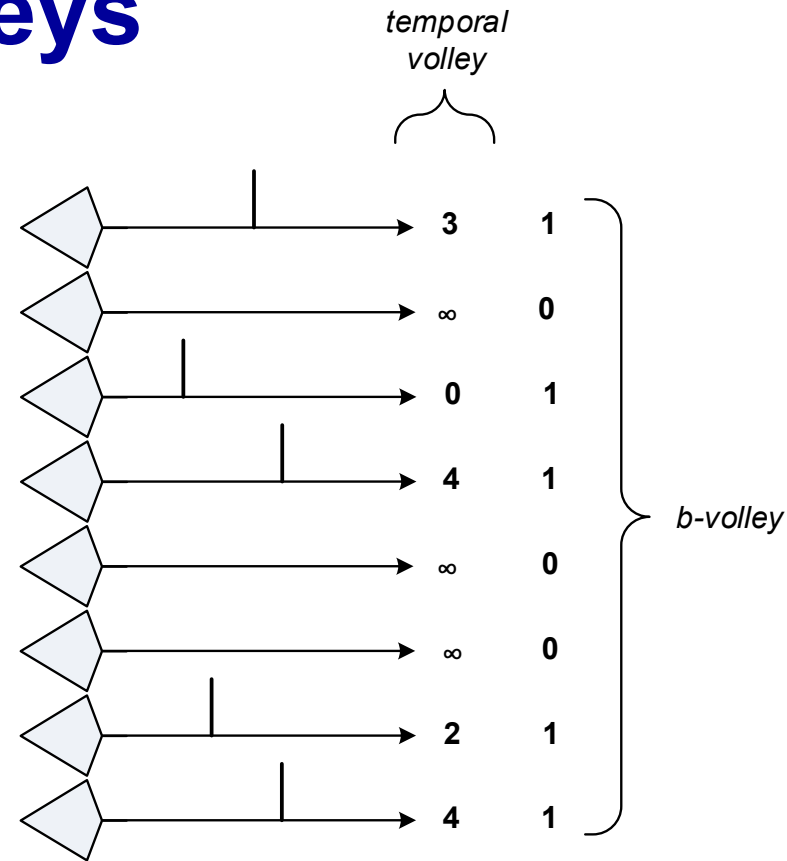
# Review: Milestone Temporal Neural Network

- Everything is done by the same network concurrently
  - Training
  - Evaluation
  - Re-training
- Features
  - Training is *unsupervised*
  - All computation is local
  - All very low resolution data
  - A *direct temporal implementation* may yield extremely good energy efficiency and speed



# Interpreting Spike Volleys

- Each line in a bundle is associated with a *feature*
  - A spike on the line indicates the *presence* of the feature
  - The timing of the spike indicates the *relative strength* of the feature
- *Currently*, the focus is:
  - *Primarily* on the spatial -- a combination of features is a *spatial pattern*
  - *Secondarily* on the temporal – e.g., sharper edges imply “stronger” features
- To reason about the problem, it is sometimes convenient to reduce a volley to purely spatial form: a binary vector or *b-volley*



# Abstraction Hierarchy

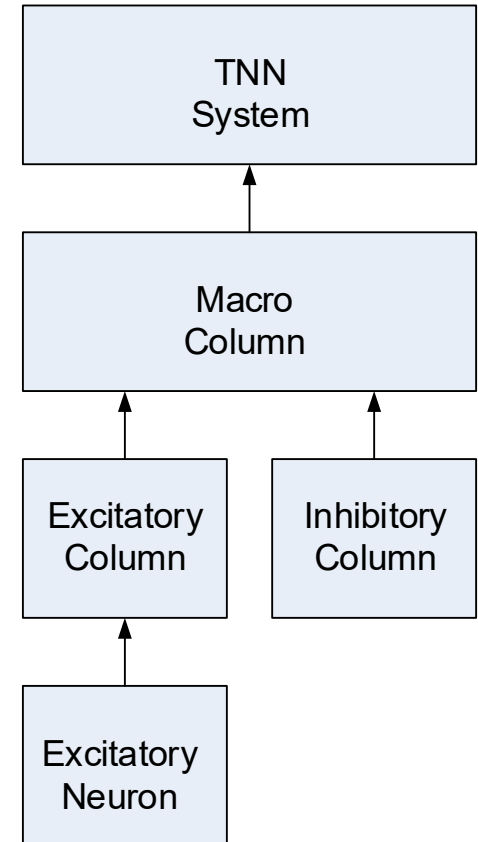
---

*Clustering TNN System* constructed as a layered hierarchy, composed of:

*Macro-Columns*, composed of:

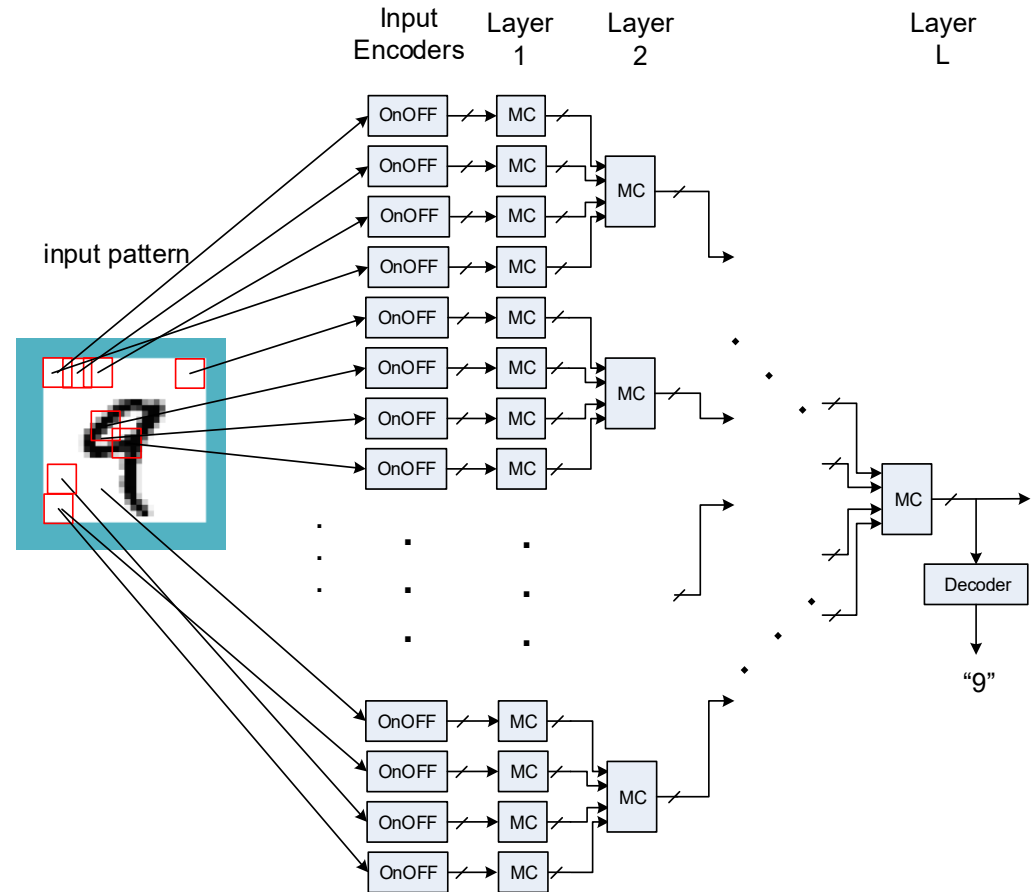
*Inhibitory Columns* and  
*Excitatory Columns (ECs)* composed of  
*Excitatory Neurons*

*Currently studying systems at Macro-Column level*

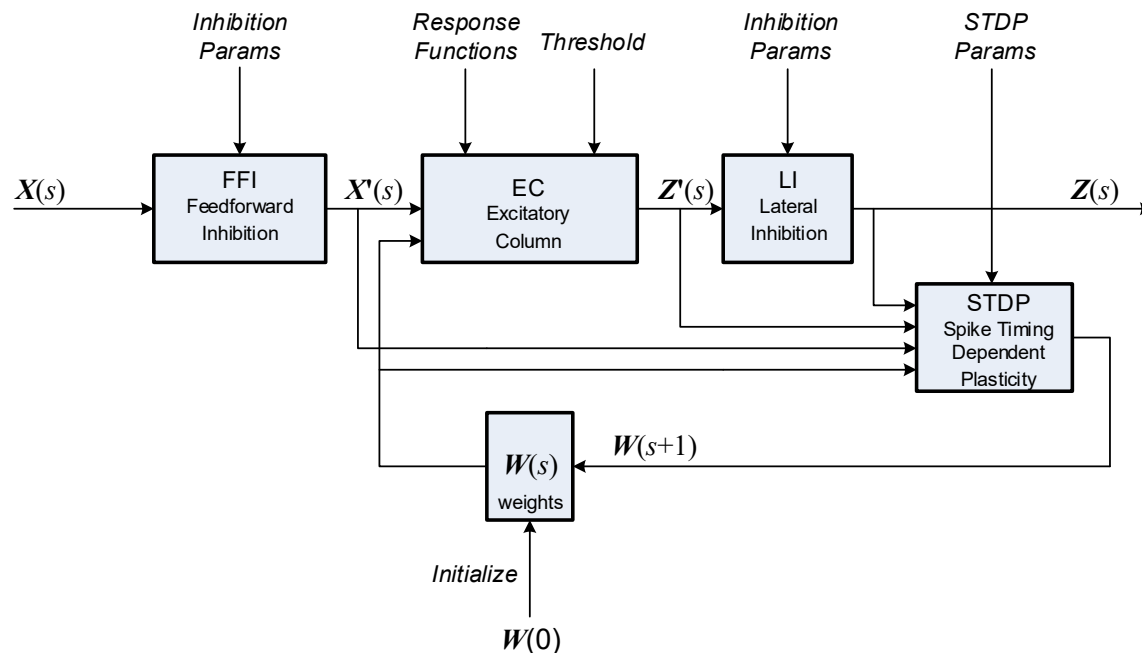


# Prototype TNN

- A layered hierarchy of Macro-Columns (MCs)
- Each data path is a *bundle of lines*
  - 16-64 lines per bundle
  - sparse spikes
- Encoding at input converts input patterns to spikes
- *Unsupervised, concurrent* training



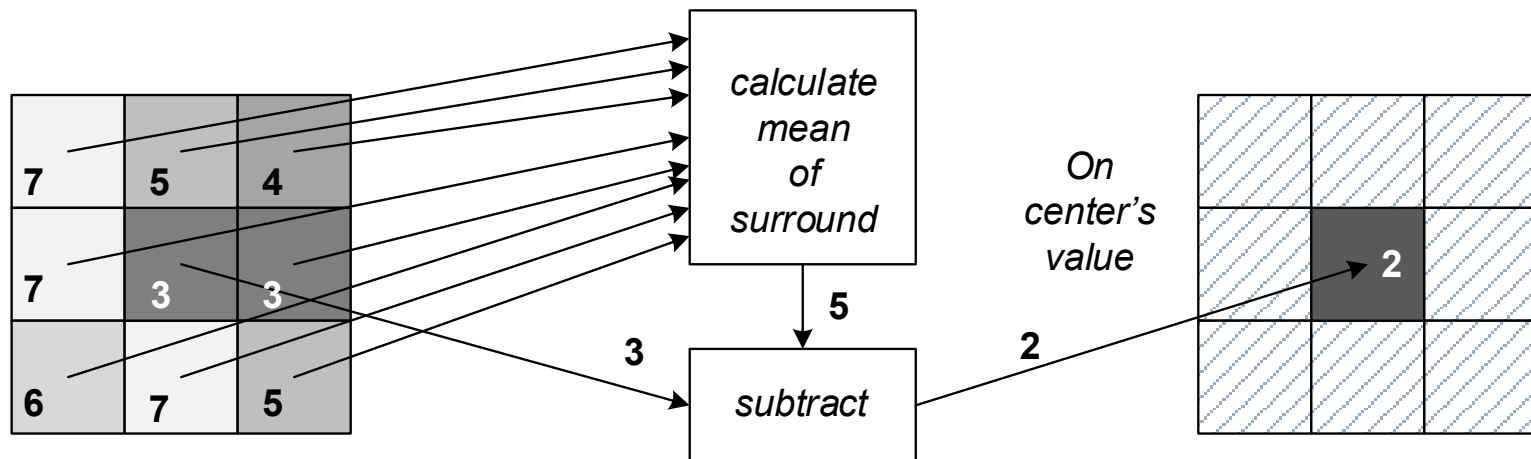
# Functional Block Diagram



- Drawn as a finite-state-machine
  - Weights,  $w(i)$ , are the only state
  - STDP modifies weights which feed back into computation
  - Everything else is *forward flow*

# Input Translation (OnOff Encoding)

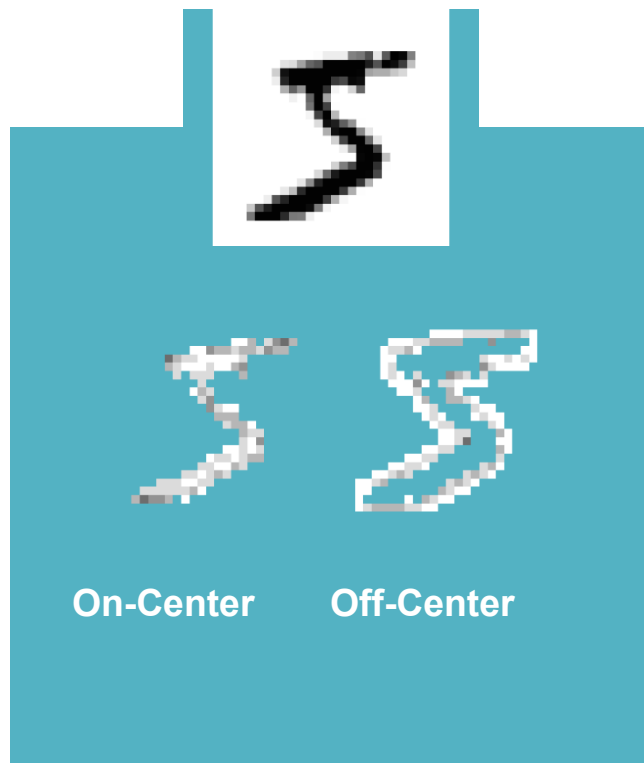
- ❑ Done on per pixel basis
  - ❑ Compute mean of “surround”
    - Currently using arithmetic mean
  - ❑ Compare with “center” – subtract
  - ❑ Determine center’s output spike time
- ❑ Example is an On-Center cell
    - Convention: a lower number is more intense input & earlier spike
  - ❑ Same inputs for an Off-Center cell yields no spike
  - ❑ On-Center and Off-Center for same pixel *never* simultaneously spike



# Input Translation (OnOff Encoding)

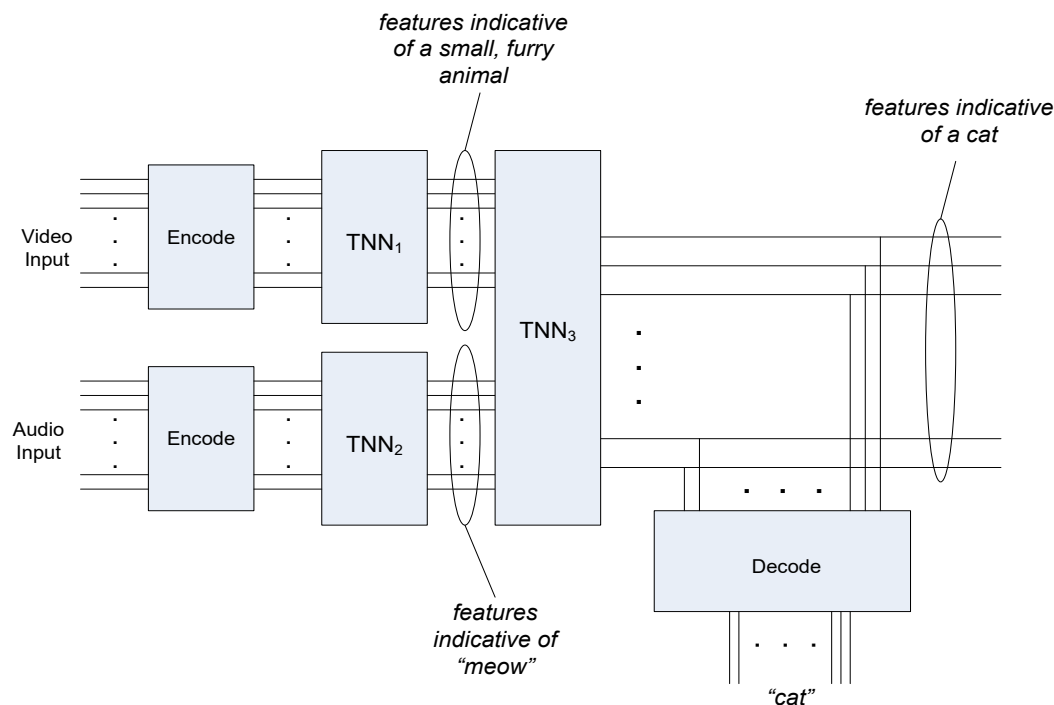
---

- ❑ MNIST examples
- ❑ Darker pixels correspond to earlier spikes



# Output Decoding

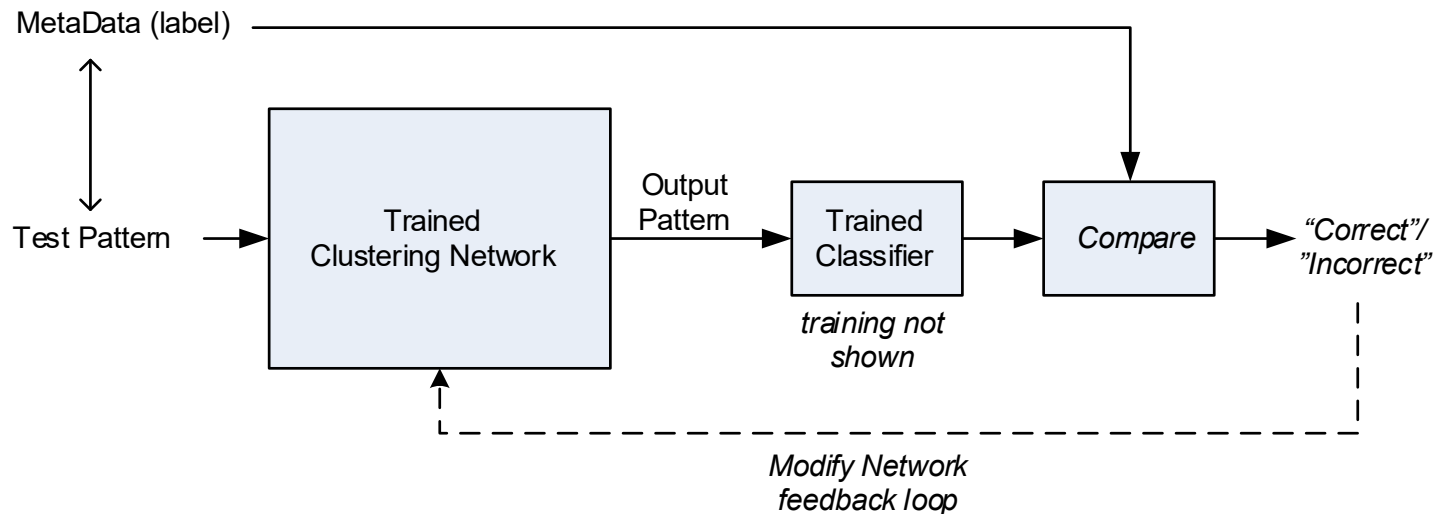
- If the outputs feed another TNN block, then no decoding is needed
- If we want to understand the outputs then we need to decode cluster Ids
  - Unsupervised training (and connectivity pattern) chooses the cluster Ids



- Construct a small, simple output classifier (decoder)

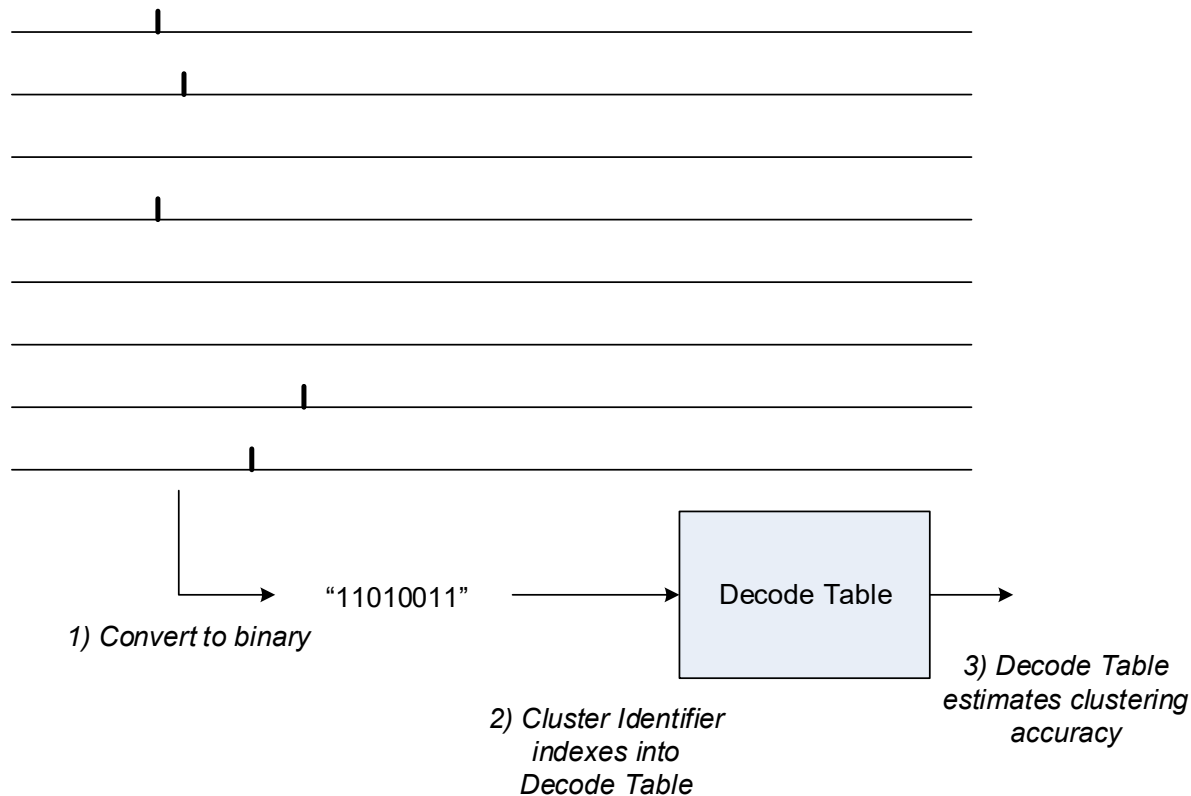
# Unsupervised Training -- Measuring “Accuracy”

- ❑ A challenge because mapping from cluster identifiers to class information is opaque
- ❑ Add trained classifier at output to make it transparent
  - For external human interpretation
  - Classifier should be simple (trivial)
  - Classification may be lossy
- ❑ Classifier results may feed back into network design process
- ❑ If Clustering Network is part of a larger system, outputs can be fed directly to downstream network(s)



# Volley Analyzer

- ❑ First, convert to purely spatial approximation: binary cluster identifier
- ❑ Note: some temporal information may be discarded



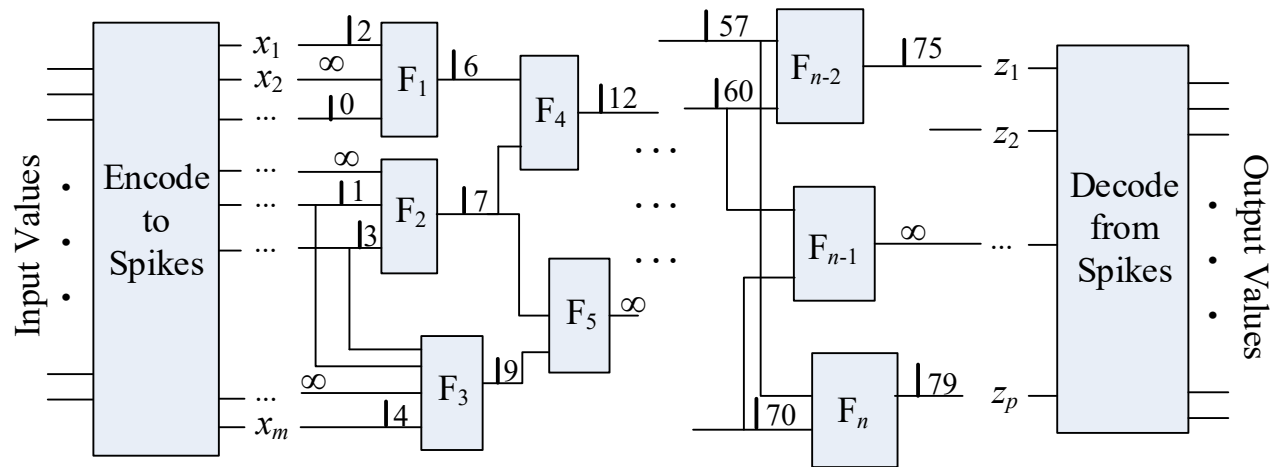
# Decode Table

- ❑ Use labels to associate inputs with cluster identifiers
  - Assign “accuracy”
- ❑ Example

cluster identifier	label counts				total	maximum	similarity
	A	B	C	D			group
000011	11	34	58	0	103	58	C
000110	49	116	192	329	686	329	D
001100	41	71	426	11	549	426	C
001001	4	2	35	0	41	35	C
001010	0	6	160	7	173	160	C
011000	15	397	36	2	450	397	B
010100	0	24	2	0	26	24	B
010010	56	0	6	2	64	56	A
010001	110	0	95	45	250	110	A
110000	0	82	65	5	152	82	B
100100	1	0	1	0	2	1	C
101000	6	3	6	0	15	6	C
100001	101	58	18	62	239	101	A
					2750	1785	
					accuracy =	0.65	

# **Space-Time Theory and Algebra**

# Space-Time Computing Network



A *Spacetime Computing System* is a feedforward composition of functions

$$F_i: \langle x_1..x_q \rangle \rightarrow z; \quad x_{1..q}, z \in \{0, 1, \dots, \infty\}$$

All  $F_i$  satisfy:

*computability:*  $F_i$  implements a computable total function.

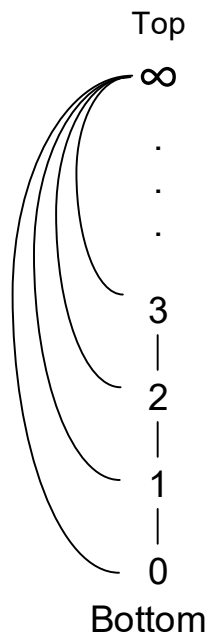
*causality:* For all  $x_j > z$ ,  $F_i(x_1, \dots, x_j, \dots, x_q) = F_i(x_1, \dots, \infty, \dots, x_q)$ , and if  $z \neq \infty$ , then  $z \geq x_{\min}$ .

*invariance:*  $F_i(x_1 + 1 .. x_q + 1) = F_i(x_1..x_q) + 1$

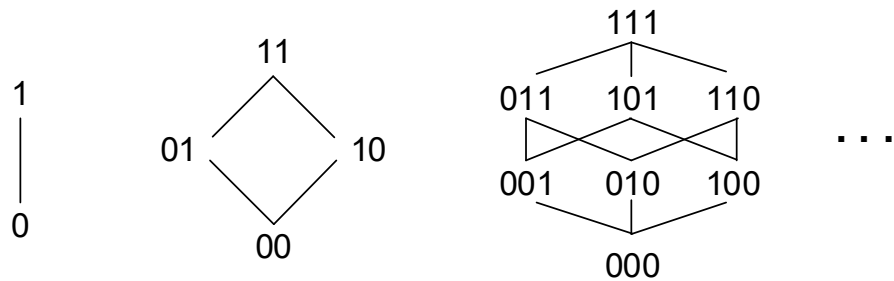
*TNNs are Space-Time Computing Networks*

# Space-Time Algebra

- S-T Algebra
- Bounded Distributive Lattice
  - $0, 1, 2, \dots, \infty$
  - Ordering relation: “<”
  - Interpretation: points in time

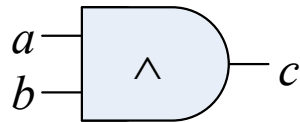


- Boolean Algebra
- Complemented Distributive Lattice
  - $0, 1$  (2-valued case)
  - Ordering relation: “<”
  - Interpretation: logical values



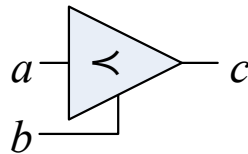
# Primitive S-T Operations

---



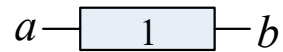
*“atomic excitation”*

*min: if  $a < b$  then  $c = a$   
else  $c = b$*



*“atomic inhibition”*

*lt: if  $a < b$  then  $c = a$   
else  $c = \infty$*

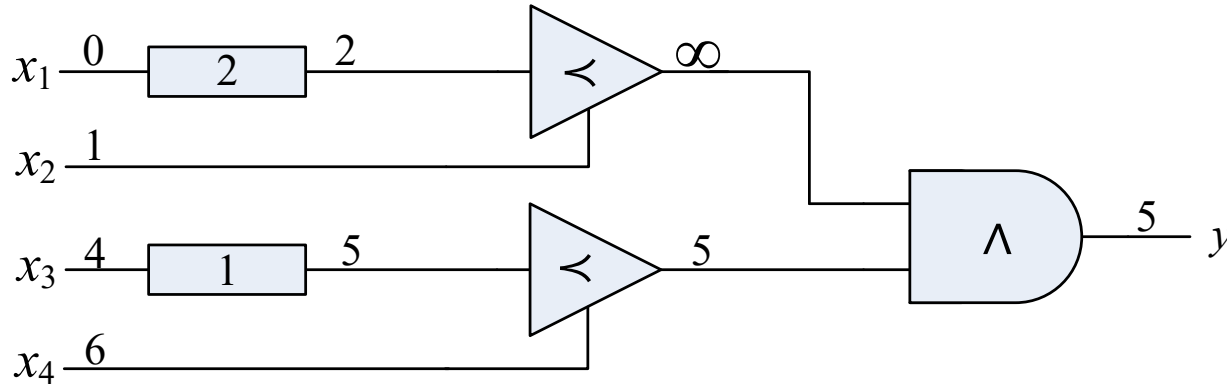


*“atomic delay”*

*inc:  $b = a + 1$*

# Space-Time Networks

- *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:  $a - \boxed{n} - b = a + n$

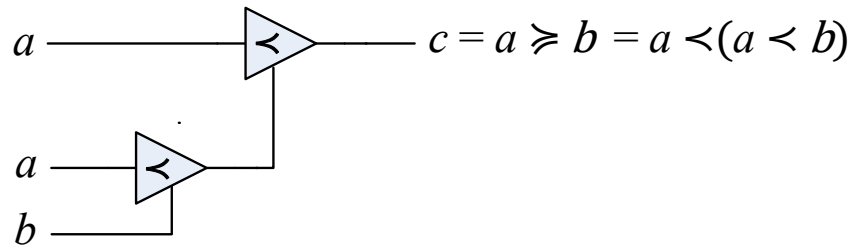
# Elementary Functions

- *Increment* and *identity* are the only one-input functions
- Table of all two-input *s-t* functions
  - Many are non-commutative

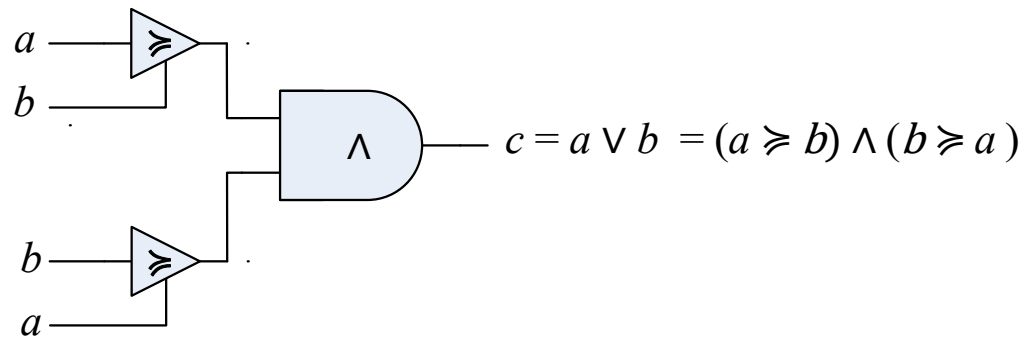
function	name	symbol
if $a < b$ then $a$ ; else $b$	min	$\wedge$
if $a > b$ then $a$ ; else $b$	max	$\vee$
if $a < b$ then $a$ ; else $\infty$	less than	$\prec$
if $a \leq b$ then $a$ ; else $\infty$	less or equal	$\preceq$
if $a > b$ then $a$ ; else $\infty$	greater than	$\succ$
if $a \geq b$ then $a$ ; else $\infty$	greater or equal	$\succeq$
if $a = b$ then $a$ ; else $\infty$	equal	$\equiv$
if $a \neq b$ then $a$ ; else $\infty$	not equal	$\neq$
if $a < b$ then $a$ else if $b < a$ then $b$ ; else $\infty$	exclusive min	$x\wedge$
if $a > b$ then $a$ else if $b > a$ then $b$ ; else $\infty$	exclusive max	$x\vee$

# Implementing Some Elementary Functions

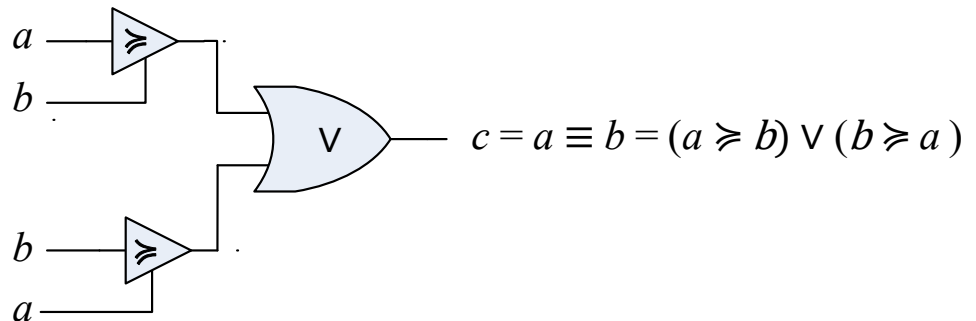
*greater or equal,  
using lt*



*max using ge and min*



*equals using ge and  
max*

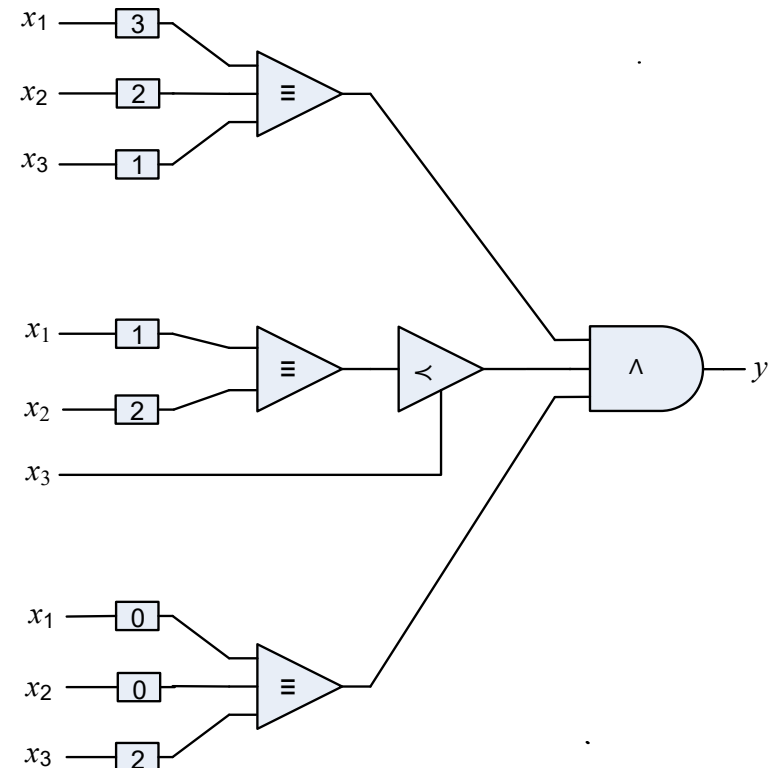


ack: Cristóbal Camarero

# Completeness

- *Theorem: min, inc, and It are functionally complete for the set of s-t functions*
  - *equals can be implemented w/ min and It*
- *Proof: by construction via normalized function table*
  - *Example:*

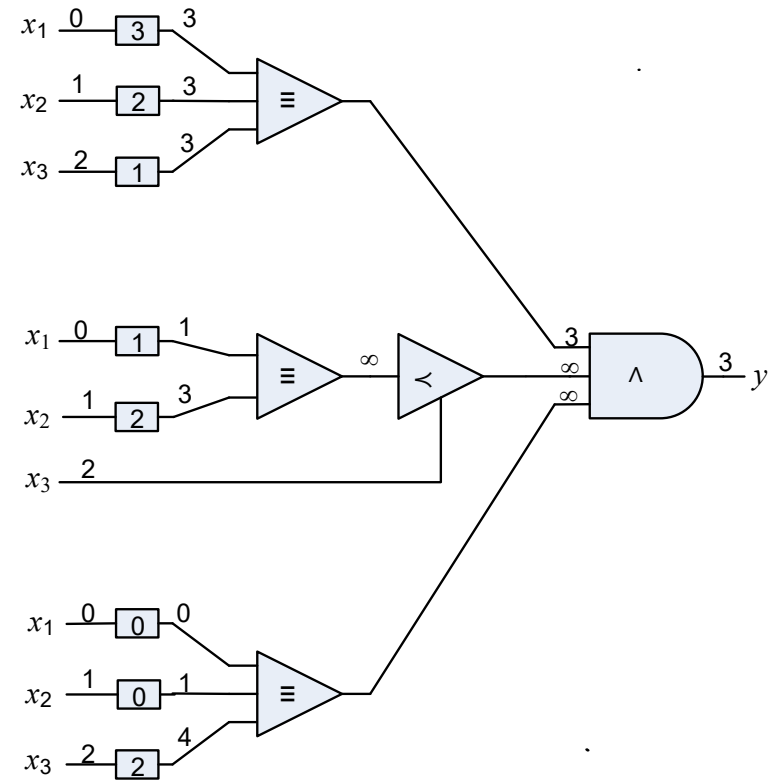
$x_1$	$x_2$	$x_3$	$y$
0	1	2	3
1	0	$\infty$	2
2	2	0	2



# Example

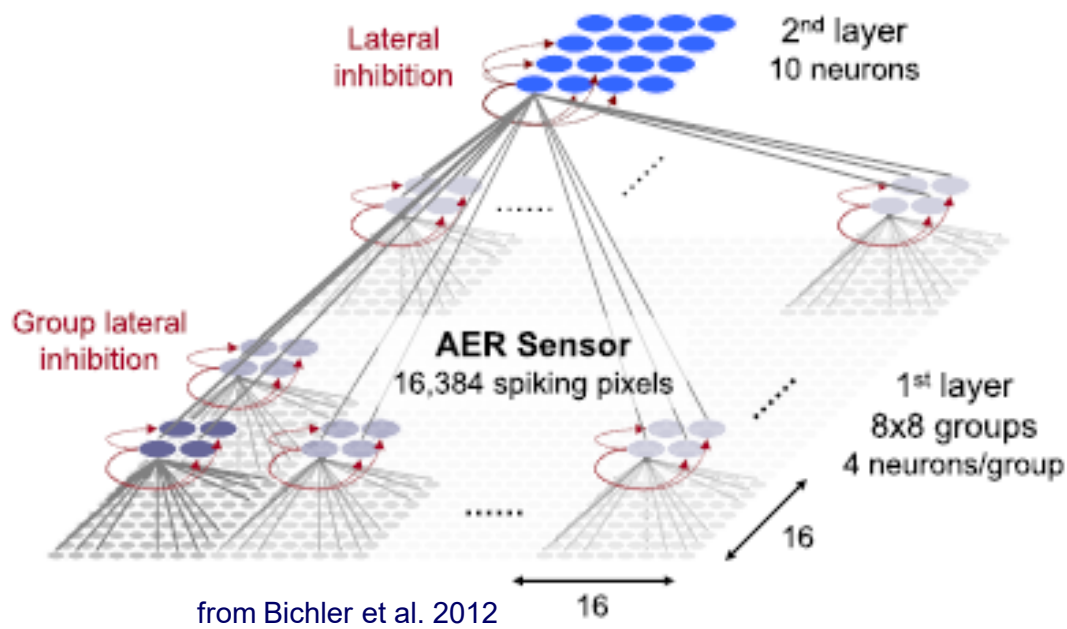
Input  $x_1 = 0, x_2 = 1, x_3 = 2$

$x_1$	$x_2$	$x_3$	$y$
0	1	2	3
1	0	$\infty$	2
2	2	0	2



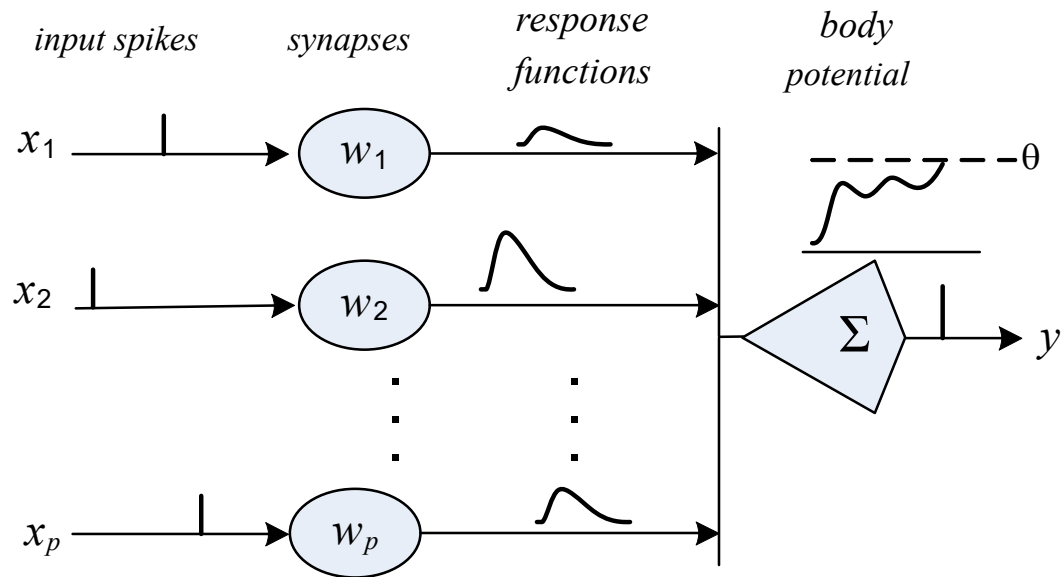
# Implement TNNs With S-T Primitives

- *Corollary*: All TNN architectures can be implemented using only *min*, *inc*, and *It*
  - Theorem/Corollary say we can do it... so let's construct basic TNN elements
  - Focus on excitatory neurons and lateral inhibition



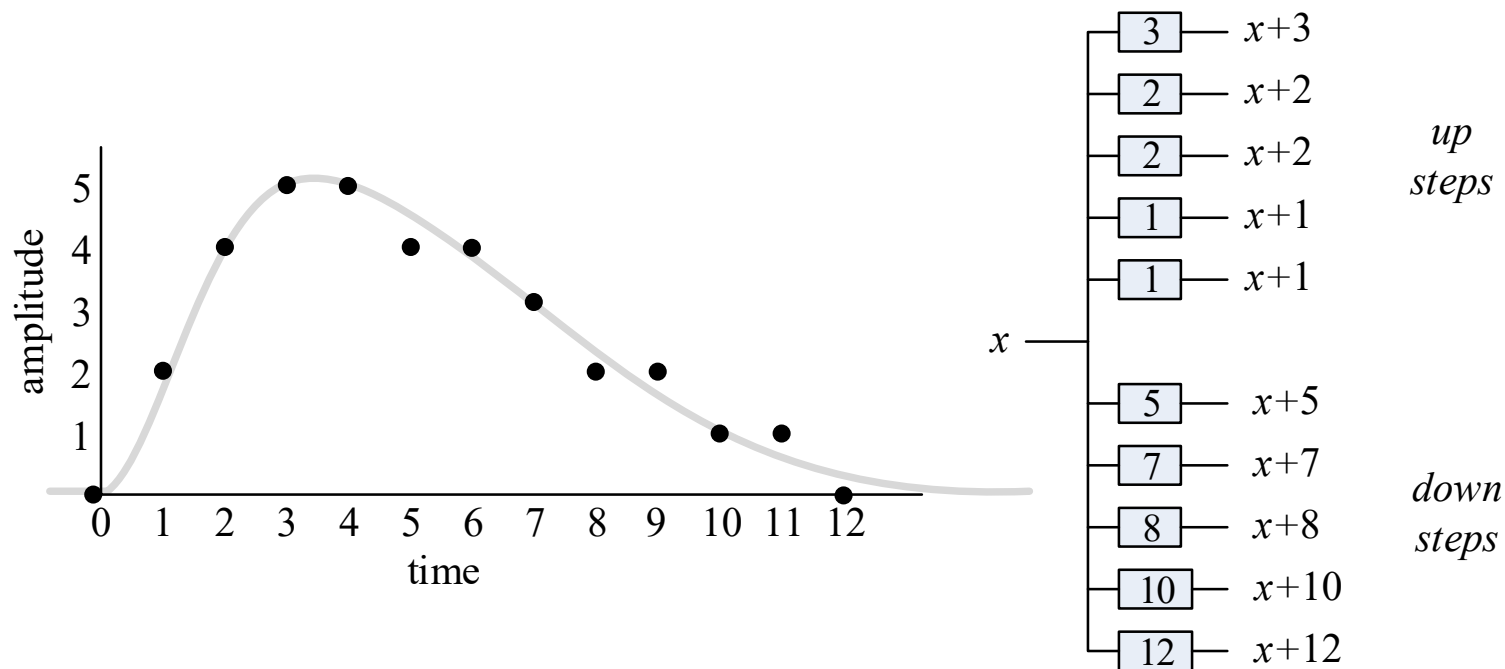
# Quick Review: SRM0 Neuron Model

---



# SRM0 Neurons

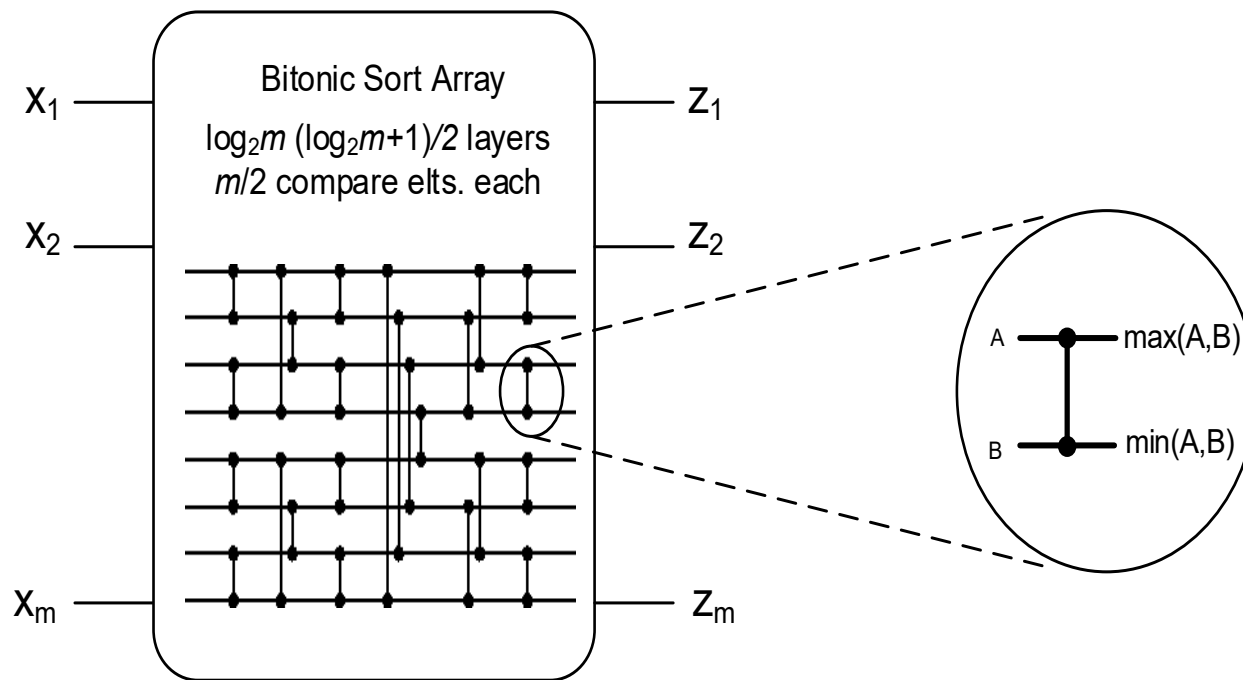
- Start with SRM0 neurons
  - A response function can be specified as up and down steps



*This not a toy example – it is realistically low resolution*

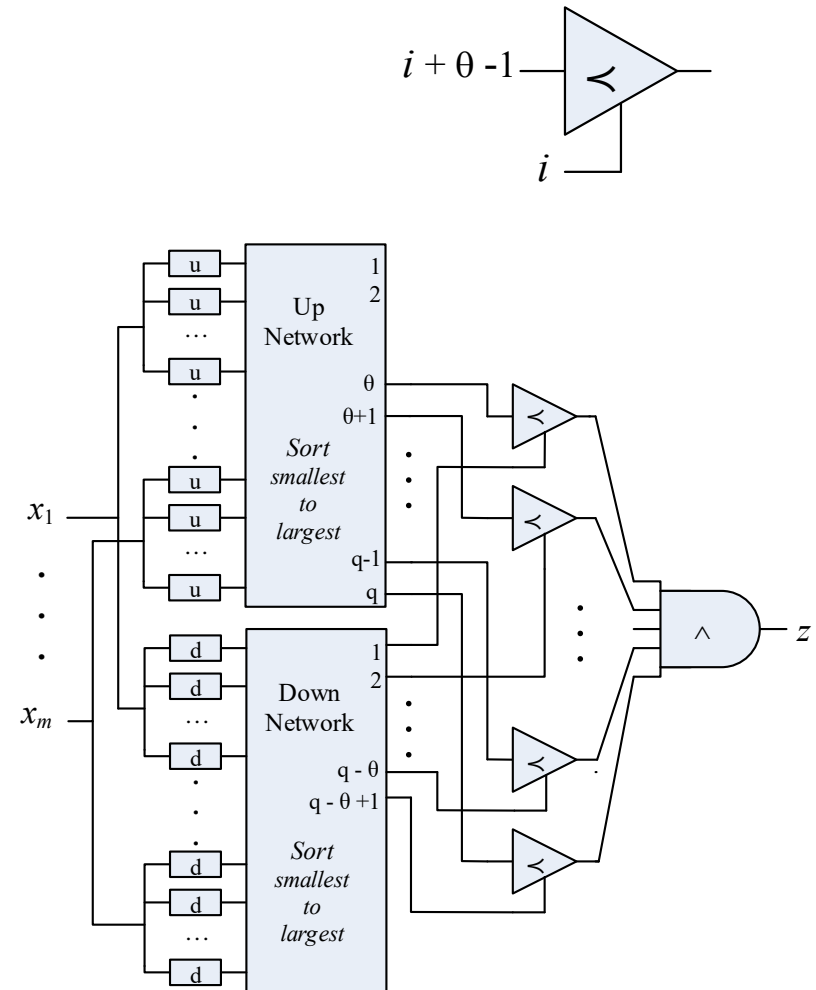
# Bitonic Sort (Batcher 1968)

- ❑ *Sort* is a space-time function
- ❑ A key component to the SRM0 construction
- ❑ Bitonic sorting network (Batcher) is a composition of max/min elements



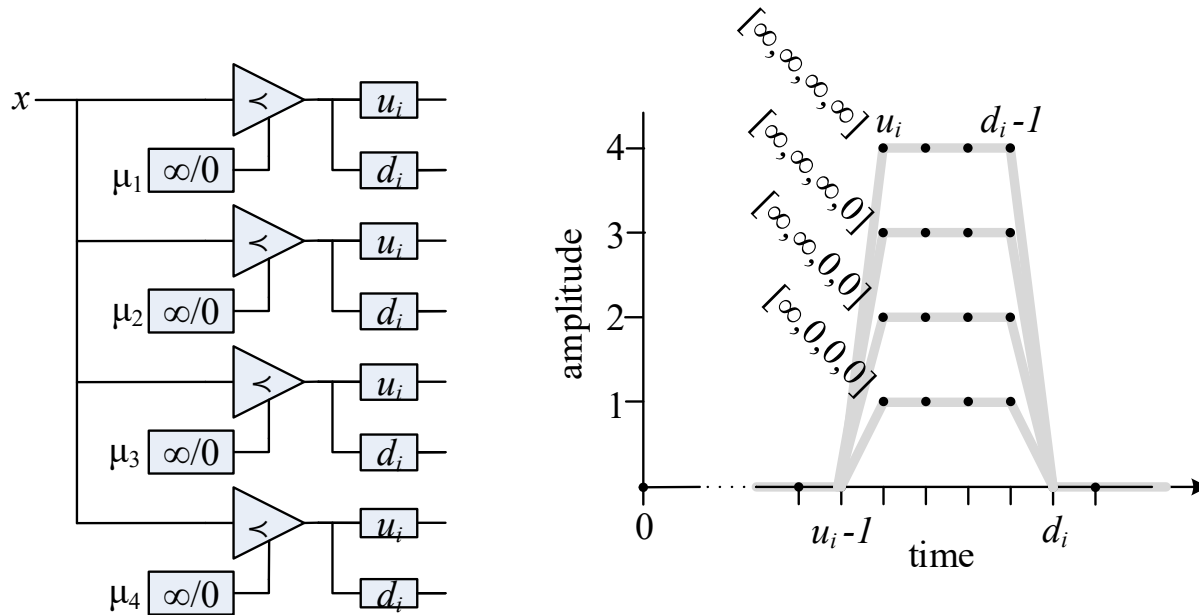
# Complete SRM0 neuron

- Up Network: sort times of all *up* steps
- Down Network: sort times of all *down* steps
- $<$  & *min* tree determines the first time *no. of up steps*  $\geq$  *no. of down steps* +  $\theta$ 
  - This is the first time the threshold is reached
- Supports *all possible* finite response functions
  - I.e., all response functions that can be physically implemented
  - Includes inhibitory (negative) response functions



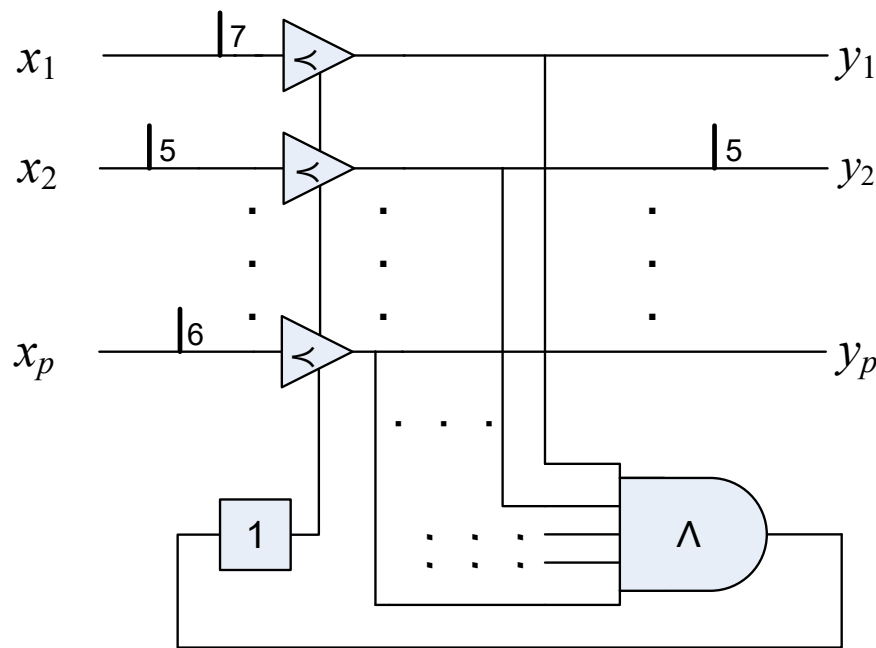
# Synaptic Weights

- Synaptic weights determine the response function
  - Synaptic weights map to  $\infty/0$  “micro-weights” that control network composed of *on/off It* gates and fanned-out up/down *increment* gates
  - Hence, micro-weight switch settings determine *s-t* response function



# WTA Inhibition

- ❑ Winner take all inhibition
  - Only the first spike in a volley is allowed to pass
  - Inhibitory neurons modeled *en masse*
  - A form of lateral inhibition



*Aside: this is an example of physical feedback, but no functional feedback*



# **Implementations**

# Paradigms, Platforms, Implementations

---

- ❑ Paradigm
  - A mathematical/algorithmic computing model
- ❑ Platform
  - Hardware and software infrastructure upon which a paradigm is implemented
- ❑ Implementations
  - *Direct*: transient events in the paradigm are implemented directly in the hardware (e.g. voltage pulses or edges)
  - *Indirect*: transient event timing is simulated; e.g., times are maintained and operated on as binary encoded values

# Platform Types

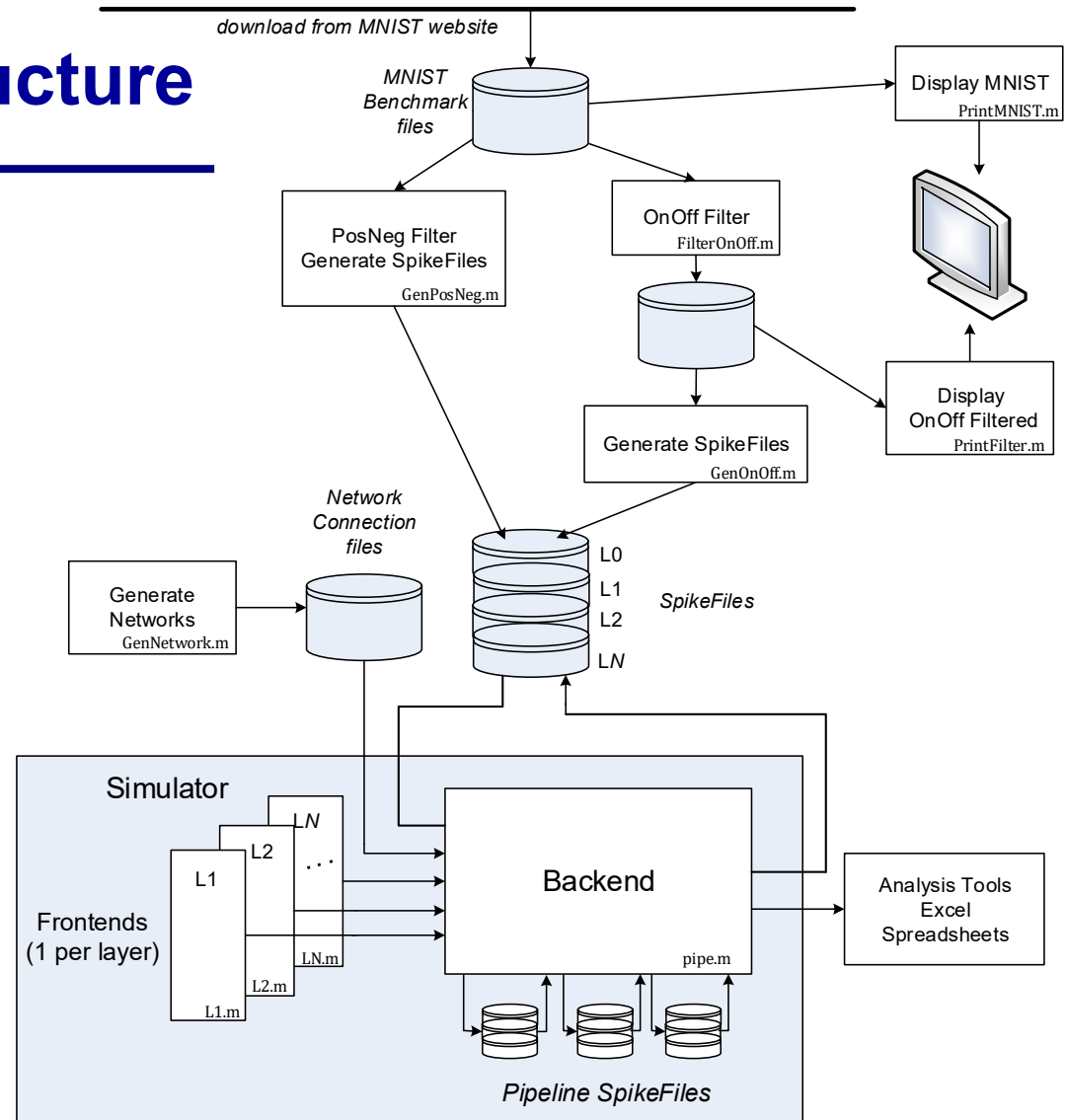
---

- ❑ General purpose, indirect
  - Example: simulator written in Matlab running on a desktop
- ❑ Special purpose, indirect
  - FACET [Schemmel et al. 2010]
  - IBM TrueNorth [Merolla et al. 2011]
  - SpiNNaker [Furber et al. 2014]
  - Intel Loihi [Davies et al. 2018]
- ❑ Special purpose, direct
  - Analog neuromorphic, ala Carver Mead [Mead 1990]
  - Generalized race logic (to follow)

# **My Simulator (Indirect Implementation)**

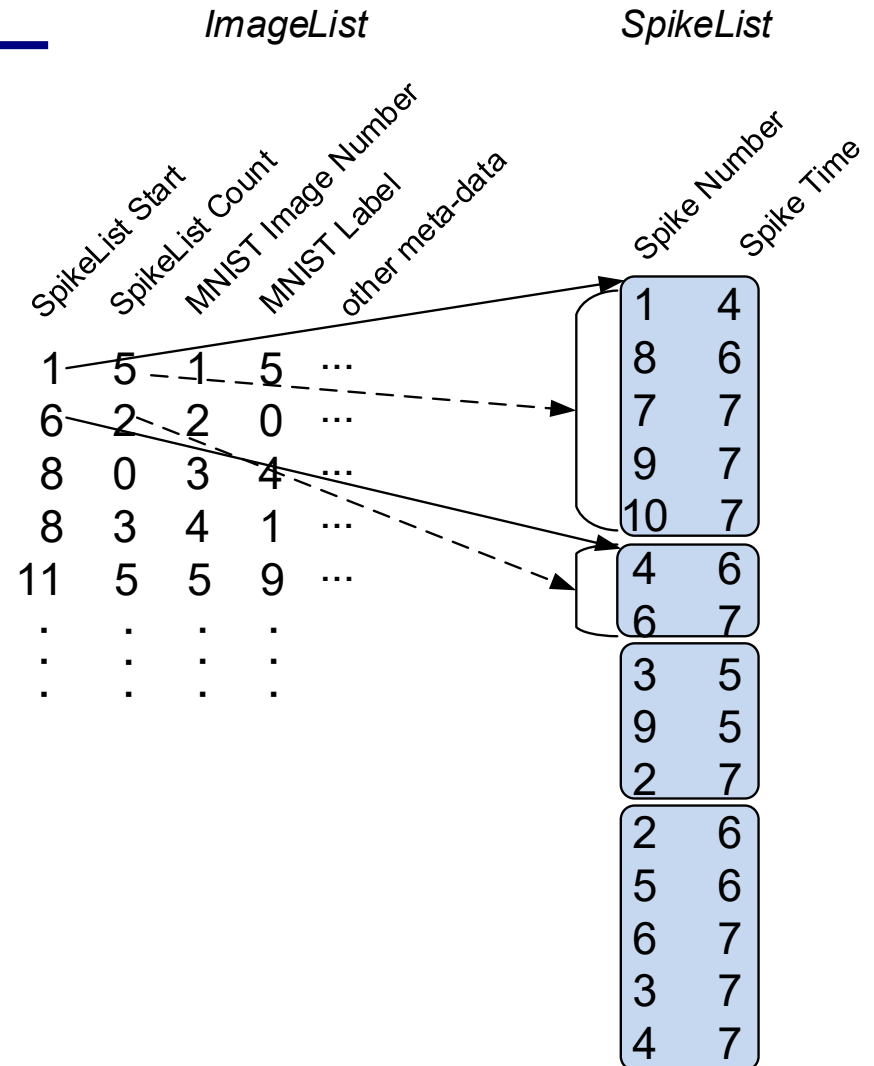
# Simulation Infrastructure

- ❑ Written in Matlab (*in a block-structured FORTRAN style*)
- ❑ Convert input data to standard SpikeFile form and save
- ❑ Thereafter, all spike information is passed around in SpikeFile format
- ❑ Simulator consists of:
  - Multiple frontends (1 per layer)
  - Common backend
  - Intermediate data passed through save/load SpikeFiles
- ❑ Network connection files are generated separately
- ❑ Analysis and engineering “note taking” is done *ad hoc* using Excel as a base



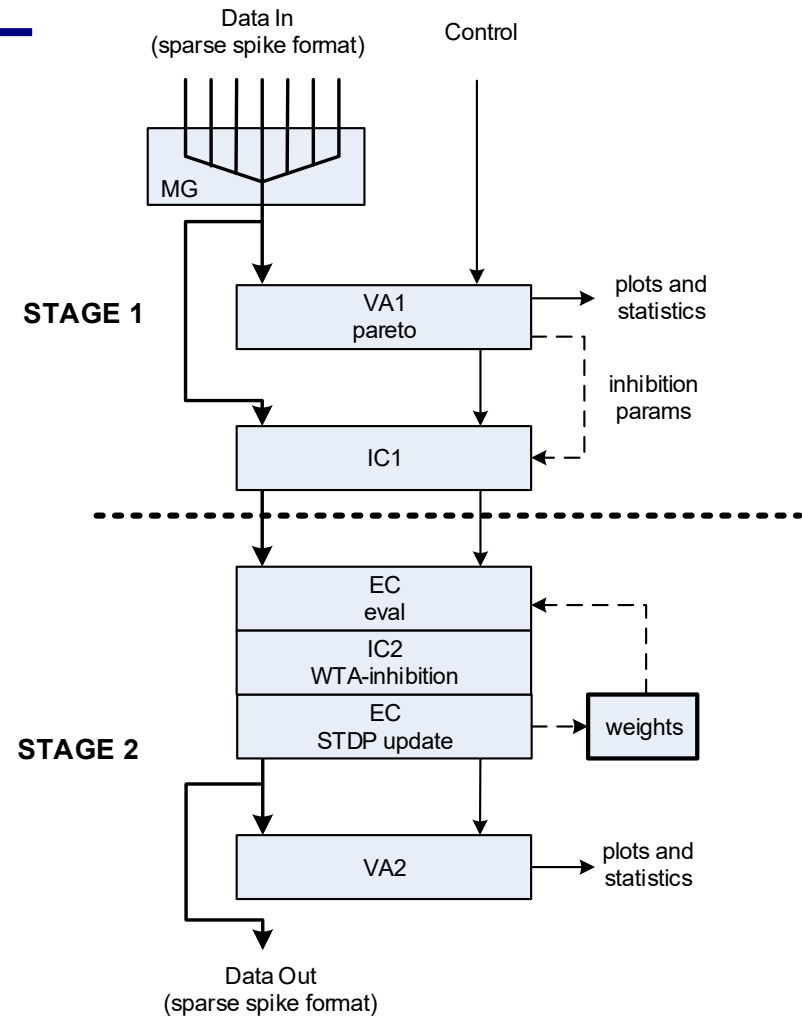
# SpikeFile Format

- ❑ Two data arrays
  - ImageList has one entry per image
    - pointer into SpikeList
    - spike count
    - label and other meta-data
  - SpikeList contains all spikes in a sparse vector format
- ❑ Matlab save/load stores and communicates these structures via secondary storage

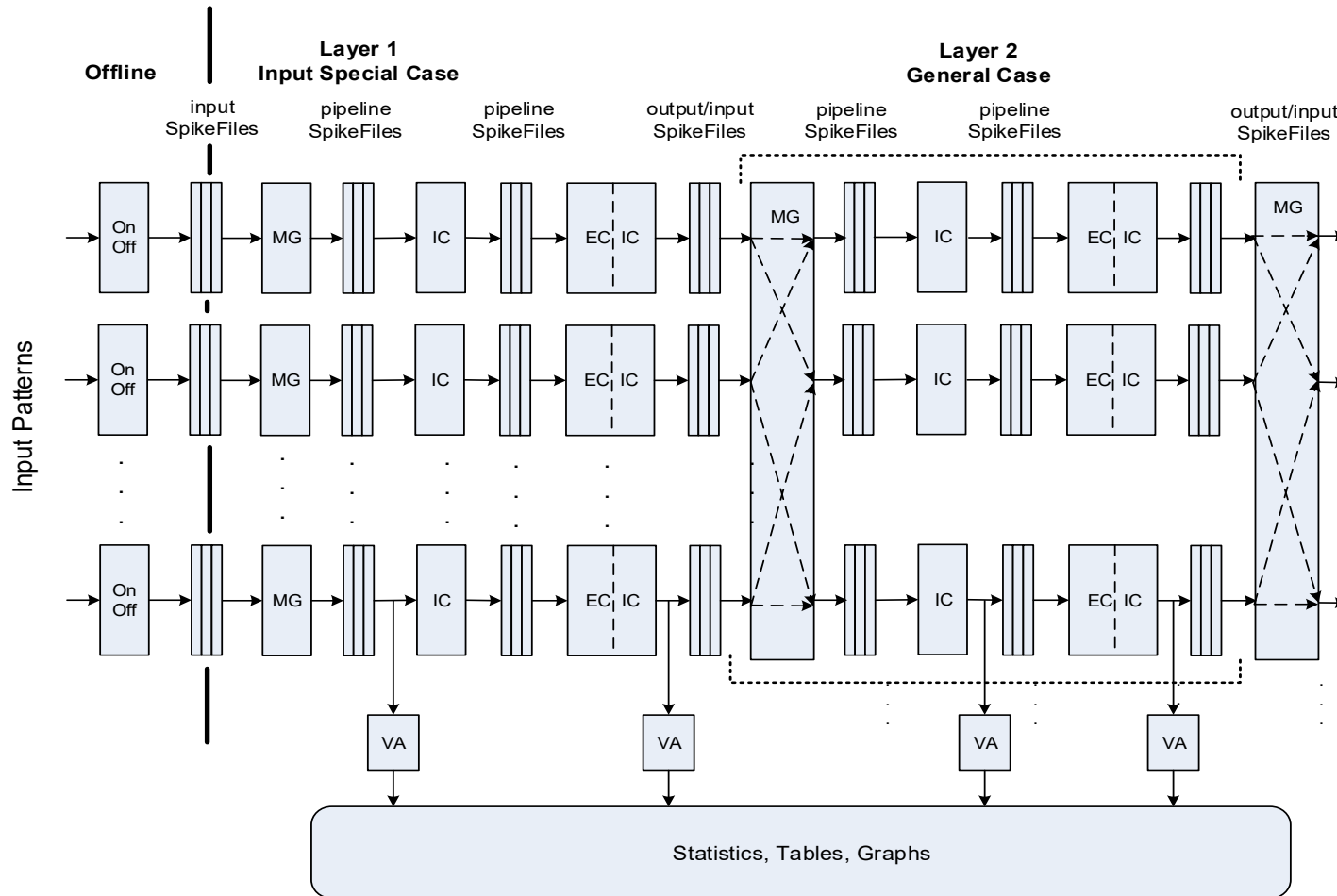


# Simulator Backend Pipeline

- Two main stages
  - 2<sup>nd</sup> larger than first
  - Files saved in between
  - Decomposes pipeline development
- Weights are the only maintained state
  - Updated dynamically via STDP
- Several sub-stages
  - MG: merge input streams
  - VA: volley analyzer; computes pareto curves for inhibition
  - IC: inhibitory column
  - EC: excitatory column
    - evaluates new inputs
    - applies WTA inhibition
    - updates weights according to STDP

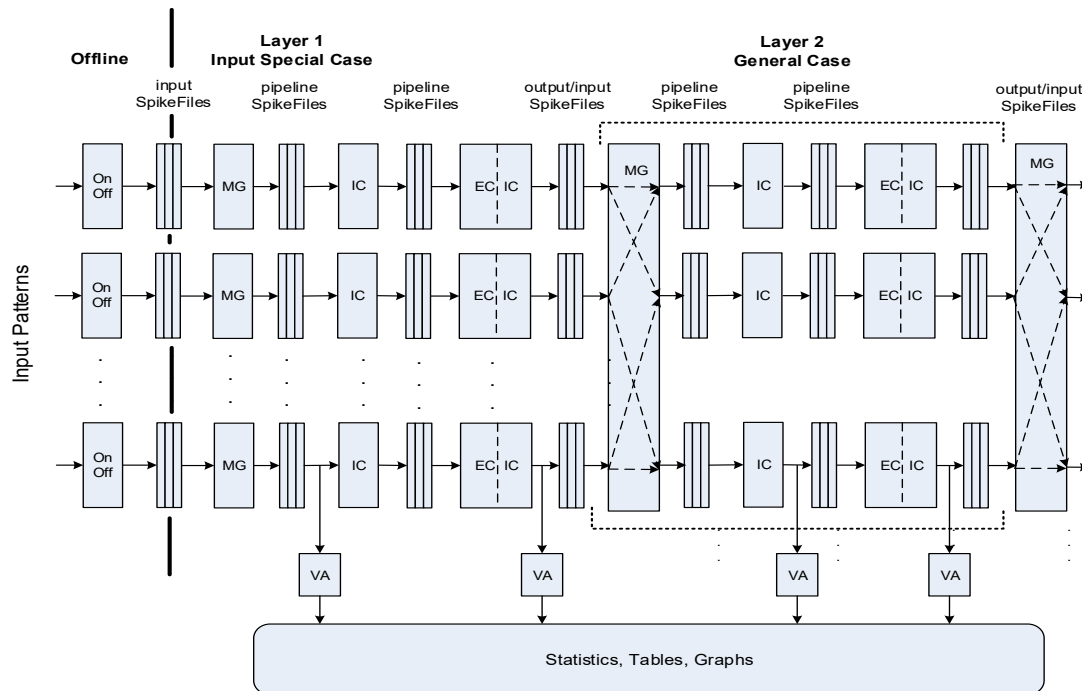


# Simulator Architecture



# Simulator Architecture

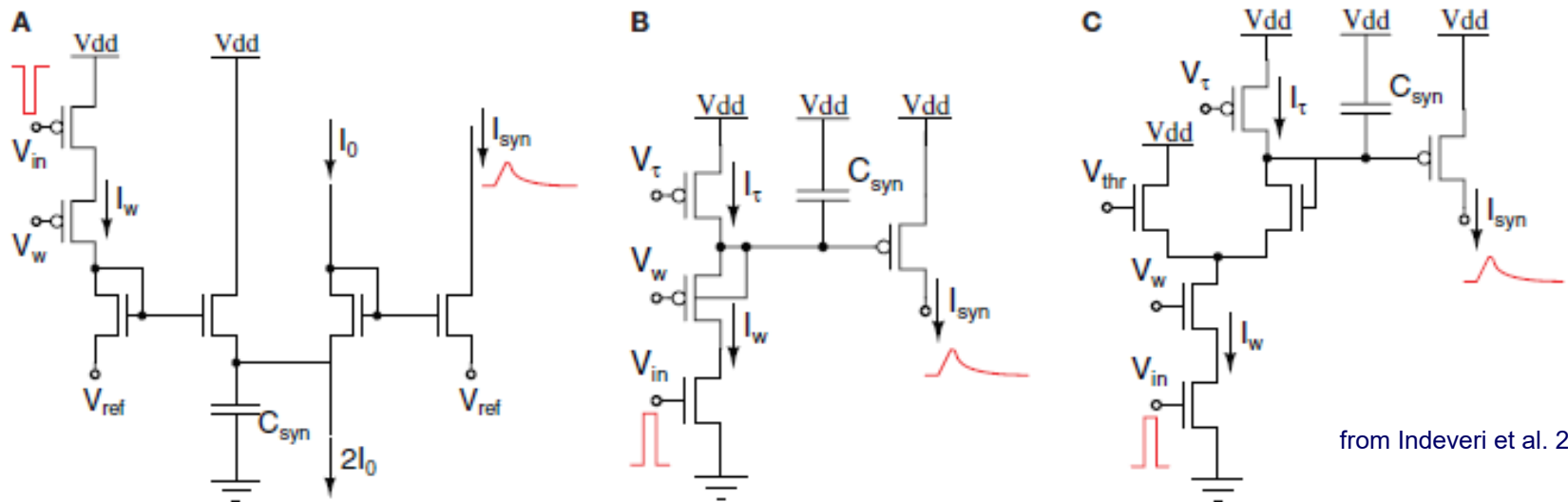
- ❑ Forward flow in simulator mimics forward flow in network
- ❑ OnOff encoding done once on MNIST images, then saved
- ❑ Addressable buffers separate stages
  - Connectivity established via buffer addresses
  - Buffers are actually files, so stages can be simulated in different simulator runs



# **Direct Implementations**

# Classic Neuromorphic Systems

- The idea:
  - Literally build hardware that communicates via spikes in time
  - An attempt to capture efficiency benefits of spiking neurons

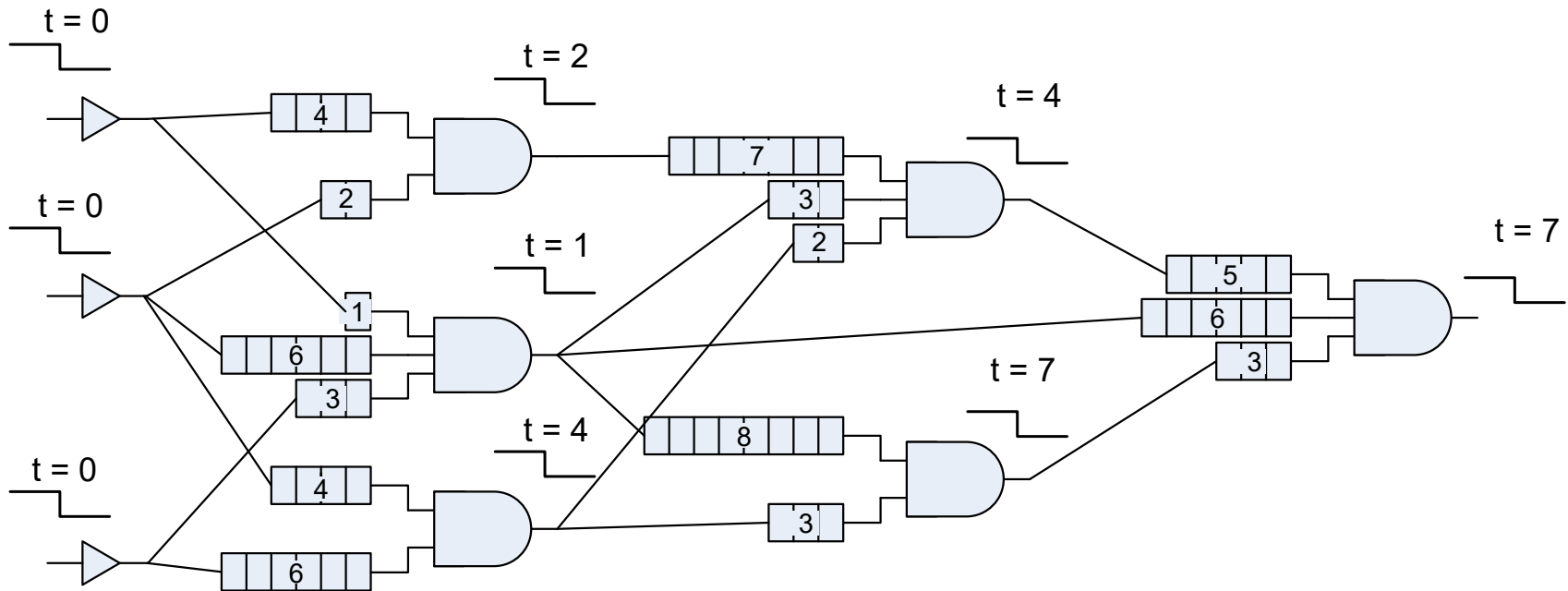
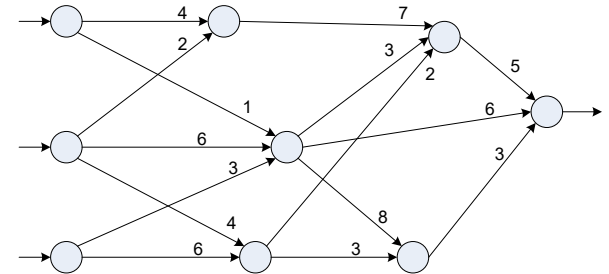


from Indeveri et al. 2011

- This is a *direct* implementation
  - Extend to other implementations that communicate via *events* in time
  - *Edges* are also events in time...  $\Rightarrow$  Generalized Race Logic

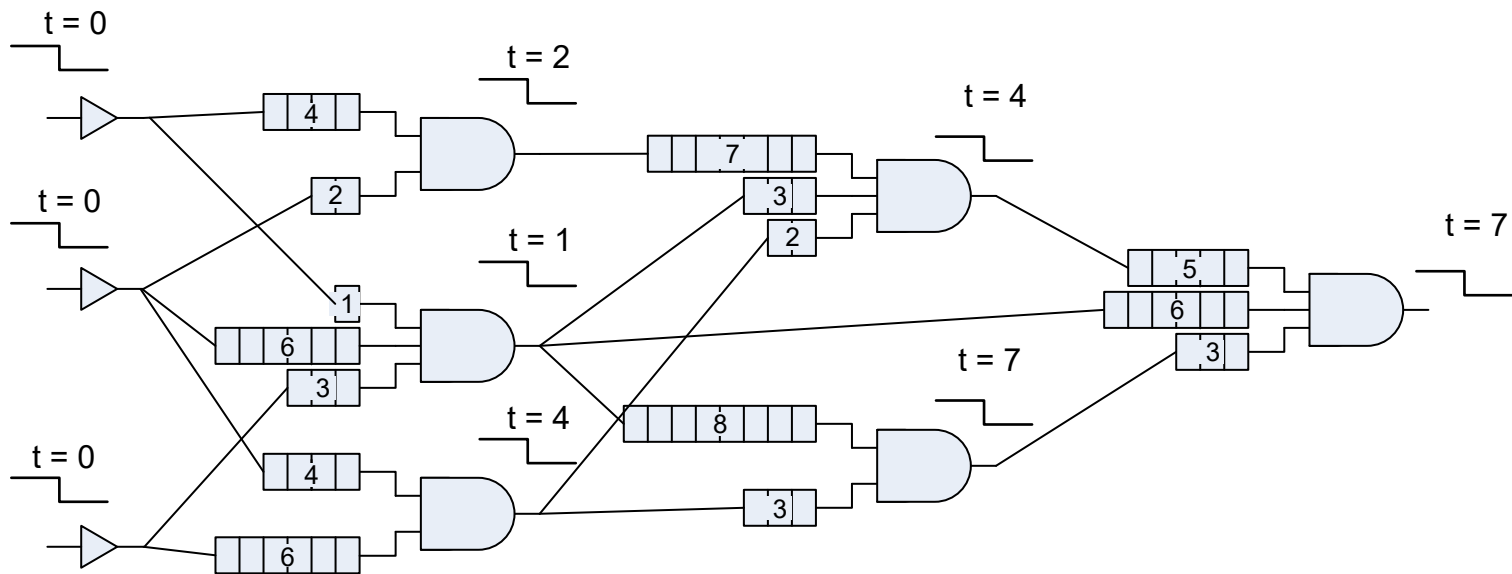
# Race Logic: Shortest Path

- By Madhavan et al. (2015)
- Shortest path computing network
  - *Weights* implemented as shift registers
  - *Min* functions implemented as AND gates



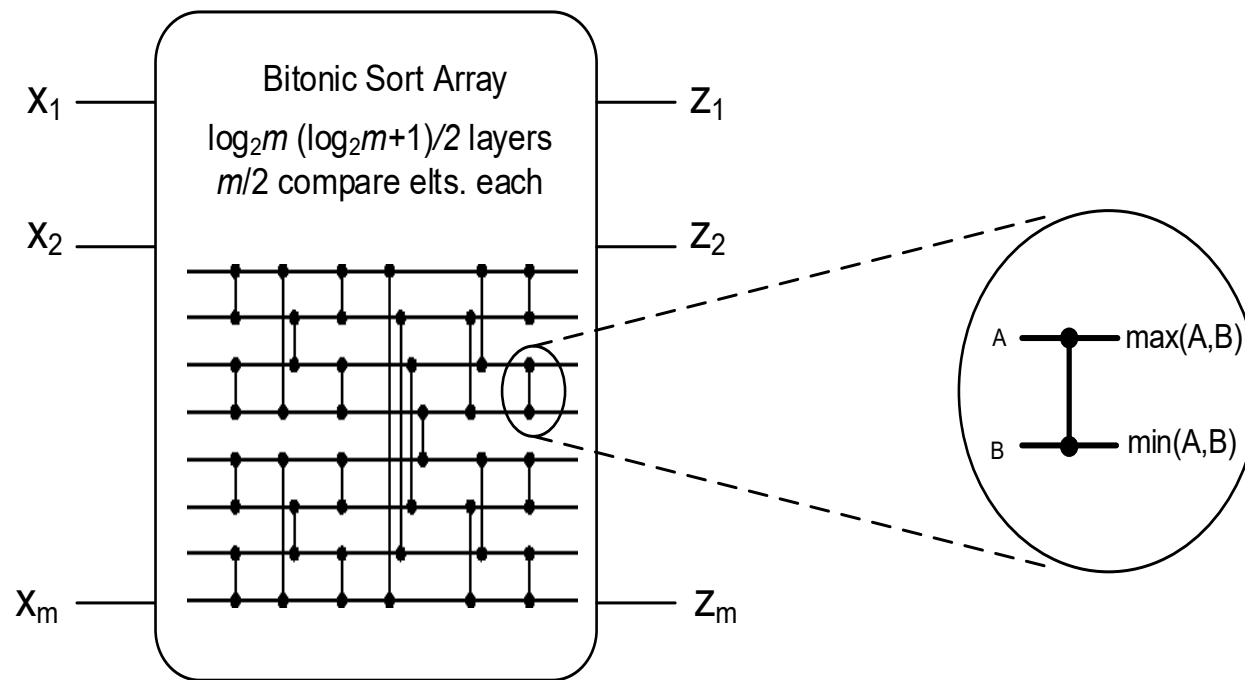
# Network Operation

- ❑ Initialize all signals to 1
- ❑ At time  $t=0$ , transition  $1 \rightarrow 0$  on all primary inputs
- ❑ Transitions sweep forward in network
  - Transition times are values
- ❑ The time it takes to compute the short path is the short path



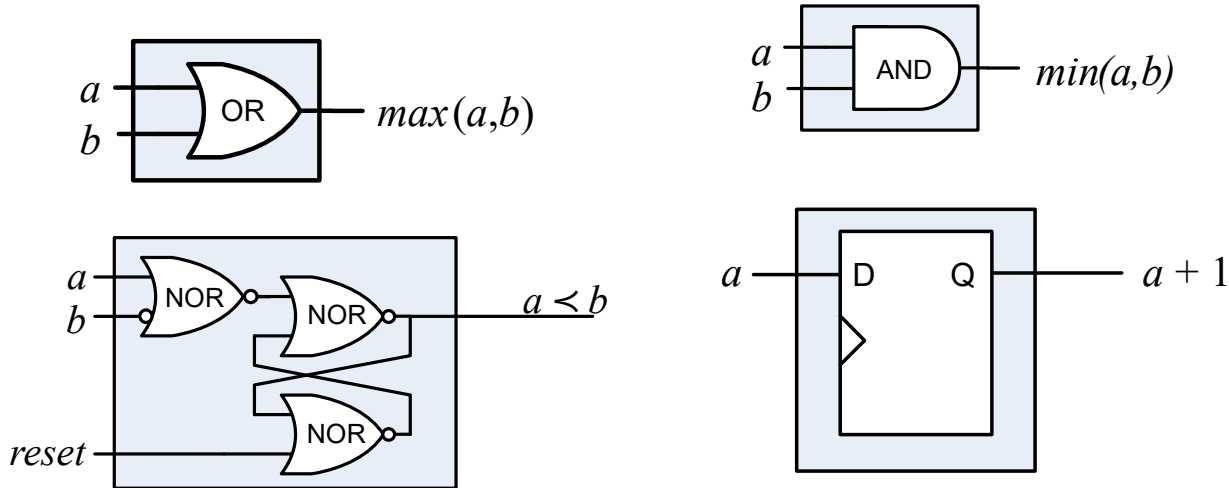
# Bitonic Sort (Batcher 1968)

- ❑ *Sort* is a space-time function
- ❑ A key component to the SRM0 construction
- ❑ Bitonic sorting network (Batcher) is a composition of max/min elements



# Generalized Race Logic

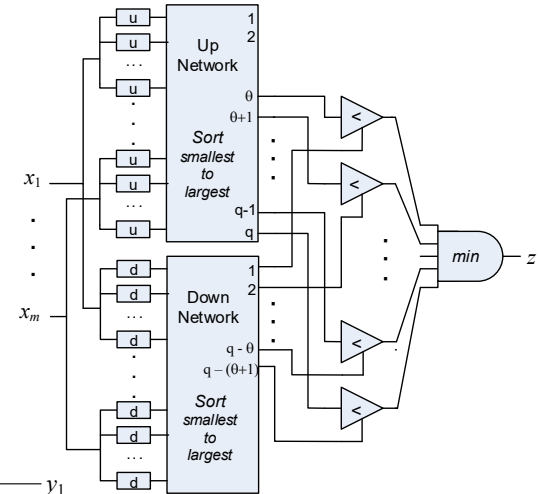
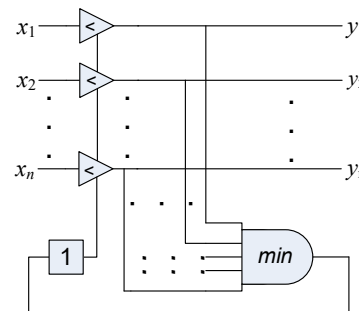
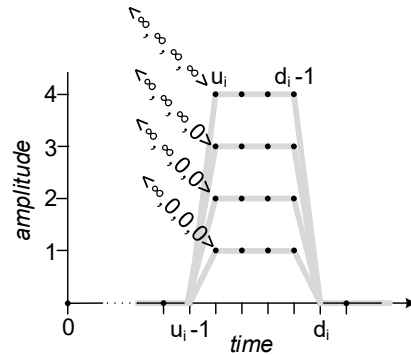
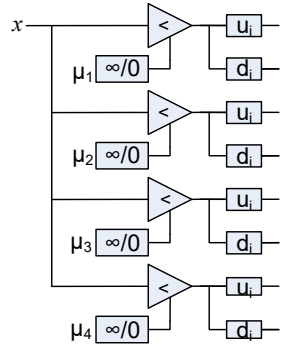
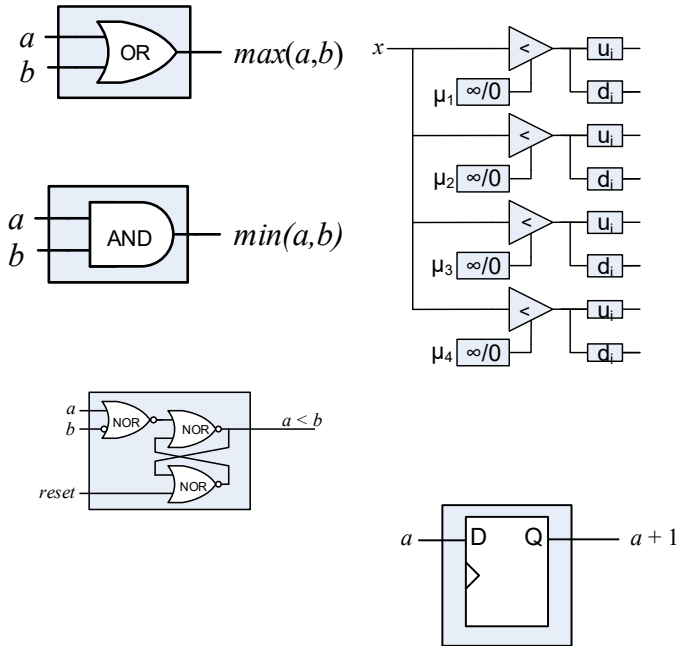
- $S$ - $T$  primitives implemented directly with conventional digital circuits
  - Signal via  $1 \rightarrow 0$  transitions



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

# Exercise

- Build an SRM0 neuron using generalized race logic



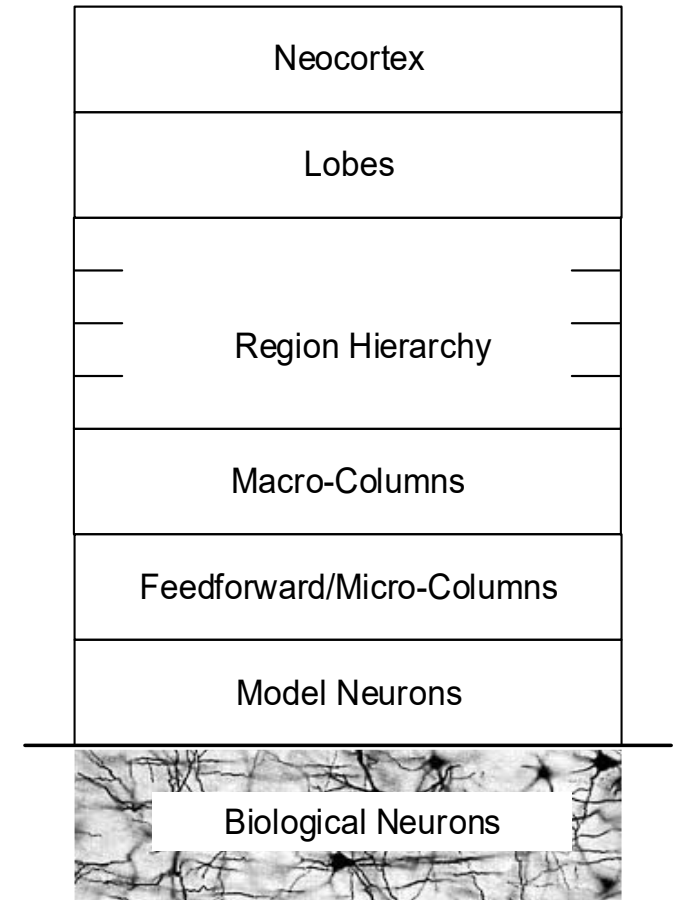
*Implication: Construct architectures in the neuroscience domain  
 Implement directly with off-the-shelf CMOS*

# **Concluding Remarks**

# Research Directions

---

- Huge research space
  - Response functions
  - Computation in dendrites
  - STDP update functions
  - Network connectivity patterns
  - Sensory interfaces
  - Hippocampus and memory
  - Motor control
- *s-t* algebraic methods for implementing functions
  - Look beyond TNNs
  - What design methodology works? (natural, intuitive)
- Study direct implementation with GRL
  - Study performance/energy tradeoffs
  - What is the best way to implement delays?



# The Box: The way we (humans) perform computation

---

- ❑ We try to eliminate temporal effects when implementing functions
  - *s-t* uses the uniform flow of time as a key resource
- ❑ We use *add* and *mult* as primitives for almost all mathematical models
  - Neither *add* nor *mult* is an *s-t* function
- ❑ We prefer high resolution (precision) data representations
  - *s-t* practical only for very low-res direct implementations
- ❑ We strive for complete functional completeness
  - *s-t* primitives complete *only* for *s-t* functions
  - There is no inversion, complementation, or negation
- ❑ We strive for algorithmic, top-down design
  - TNNs are based on *implied functions*
  - Bottom-up, empirically-driven design methodology

**Think outside the box!**

# Acknowledgements

---

Raquel Smith, Mikko Lipasti, Mark Hill, Margaret Martonosi, Cristobal Camarero, Mario Nemirovsky, Mikko Lipasti, Tim Sherwood, Advait Madhavan, Shlomo Weiss, Ido Guy, Ravi Nair, Joel Emer, Quinn Jacobson, Abhishek Bhattacharjee

# Bibliography

---

- Ananthanarayanan et al. 2009:** R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with  $10^9$  neurons,  $10^{13}$  synapses." *High Performance Computing Networking, Storage and Analysis*, 2009.
- Arthur et al. 2012:** Arthur, J.V., Merolla, P.A., Akopyan, F., Alvarez, R., Cassidy, A., Chandra, S., Esser, S.K., Imam, N., Risk, W., Rubin, D.B. and Manohar, R., "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," *2012 International Joint Conference on Neural Networks*, pp. 1-8, 2012.
- Bakkum et al. 2008:** Bakkum, Douglas J., Zenas C. Chao, and Steve M. Potter, "Long-term activity-dependent plasticity of action potential propagation delay and amplitude in cortical networks", *PLOS one* 3.5 (2008): e2088.
- Batcher 1968:** Batcher, K. E., "Sorting networks and their applications," *Proceedings of the Spring Joint Computer Conference*, pp. 307-314, 1968.
- Beyeler et al. 2013:** Beyeler, Michael, Nikil D. Dutt, and Jeffrey L. Krichmar. "Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule." *Neural Networks* 48 (2013): 109-124.
- Bi and Poo 1998:** Bi, G.-Q., and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *The Journal of Neuroscience* 18, no. 24, pp. 10464-10472, 1998.
- Bichler et al. 2012:** Bichler, O., D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks* 32, pp. 339-348, 2012.
- Bohte et al. 2002a:** Bohte, S. M., H. La Poutre, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Transactions on Neural Networks*, 13, no. 2, pp. 426-435, 2002.
- Bohte et al. 2002b:** Bohte, S. M., J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing* 48, no. 1 pp.17-37, 2002.
- Brader et al. 2007:** Brader, Joseph M., Walter Senn, and Stefano Fusi. "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics." *Neural computation* 19, no. 11 (2007): 2881-2912.
- Brette 2015:** Brette, R., "Philosophy of the spike: rate-based vs. spike-based theories of the brain," *Frontiers in Systems Neuroscience* 9, 2015.

# Bibliography

---

**Carlo and Stevens 2013:** Carlo, C. Nikoosh, and Charles F. Stevens. "Structural uniformity of neocortex, revisited." *Proceedings of the National Academy of Sciences* 110, no. 4 (2013): 1488-1493.

**Davies et al. 2018:** Davies, Mike, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou et al. "Loihi: A neuromorphic manycore processor with on-chip learning." *IEEE Micro* 38, no. 1 (2018): 82-99.

**DeFelipe et al. 2002:** DeFelipe, Javier, Lidia Alonso-Nanclares, and Jon I. Arellano. "Microstructure of the neocortex: comparative aspects." *Journal of neurocytology* 31, no. 3-5 (2002): 299-316.

**DeFelipe 2011:** DeFelipe, Javier. "The evolution of the brain, the human nature of cortical circuits, and intellectual creativity." *Frontiers in neuroanatomy* 5 (2011).

**Delorme et al. 1999:** Delorme, A., J. Gautrais, R. van Rullen, and S. Thorpe, "SpikeNet: A simulator for modeling large networks of integrate and fire neurons," *Neurocomputing*, 26-27, pp. 989-996, 1999.

**DeLorme et al. 2001:** Delorme, Arnaud, Laurent Perrinet, and Simon J. Thorpe, Networks of Integrate-and-Fire Neurons using Rank Order Coding B: Spike Timing Dependant Plasticity and Emergence of Orientation Selectivity *Neurocomputing* 38 (2001): 539-545.

**Diehl et al. 2015:** Diehl, P. U., D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," *2015 International Joint Conference on Neural Networks*, pp. 1-8, 2015.

**Deiss et al. 1999:** Deiss, S. R., R. J. Douglas, and A. M. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," *Pulsed Neural Networks* , pp. 157-178, 1999.

**Eliasmith et al. 2012:** Eliasmith, C., T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, C. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science* 338, no. 6111, pp.1202-1205, 2012.

**Fauth et al. 2015:** Fauth, Michael, Florentin Wörgötter, and Christian Tetzlaff. "The Formation of Multi-synaptic Connections by the Interaction of Synaptic and Structural Plasticity and Their Functional Consequences." *PLoS computational biology* 11, no. 1 (2015).

**Felleman and Van Essen 1991:** Felleman, Daniel J., and David C. Van Essen. "Distributed hierarchical processing in the primate cerebral cortex." *Cerebral cortex* 1, no. 1 (1991): 1-47.

# Bibliography

---

**Fries et al. 2007:** Fries, P., D. Nikolić, and W. Singer, "The gamma cycle," *Trends in Neurosciences* 30, no. 7 pp. 309-316, 2007.

**Furber et al. 2014:** Furber, Steve B., Francesco Galluppi, Steve Temple, and Luis A. Plana. "The spinnaker project." *Proceedings of the IEEE* 102, no. 5 (2014): 652-665.

**Fusi and Mattia 1998:** Fusi, S., and M. Mattia, "Collective behavior of networks with linear (VLSI) integrate-and-fire neurons," *Neural Computation* 11.3, pp. 633-652, 1998.

**Gerstner et al. 1996:** Gerstner, W., R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature* 383, no. 6595, pp. 76-78, 1996.

**Gütig and Haim 2006:** Gütig, R., and Haim S., "The tempotron: a neuron that learns spike timing-based decisions," *Nature Neuroscience* 9, no. 3, pp. 420-428, 2006.

**Gütig et al. 2013:** Gütig, R., Tim G., H. Sompolinsky, and M. Meister, "Computing complex visual features with retinal spike times," *PloS One* 8, no. 1, e53063, 2013.

**Guyonneau et al. 2004:** Guyonneau, R., R. VanRullen, and S. J. Thorpe, "Temporal codes and sparse representations: A key to understanding rapid processing in the visual system," *Journal of Physiology-Paris* 98, pp. 487-497, 2004.

**Guyonneau et al. 2005:** Guyonneau, R., R. VanRullen, and S. J. Thorpe, "Neurons tune to the earliest spikes through STDP," *Neural Computation* 17, no. 4, pp. 859-879, 2005.

**Hawkins and Blakeslee 2007:** Hawkins, J., and S. Blakeslee. *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan, 2007.

**Hill et al. 2012:** Hill, Sean L., et al., Statistical connectivity provides a sufficient foundation for specific functional connectivity in neocortical neural microcircuits, *Proceedings of the National Academy of Sciences* (2012).

**Hodgkin and Huxley 1952:** Hodgkin, Alan L., and Andrew F. Huxley, A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve -- *The Journal of physiology* 117.4 (1952): 500.

# Bibliography

---

**Hopfield 1995:** Hopfield, J. J., "Pattern recognition computation using action potential timing for stimulus representation," *Nature* 376, pp. 33-36, 1995.

**Indiveri et al. 2011:** Indiveri, G., Linares-Barranco, B., Hamilton, T.J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.C., Dudek, P., Häfliger, P., Renaud, S. and Schemmel, J., "Neuromorphic silicon neuron circuits", *Frontiers in Neuroscience* 5, 2011.

**Irwin and Shen 2005:** Irwin, M. J., and J. P. Shen, "Revitalizing computer architecture research," *Computing Research Association*, 2005.

**Izhikevich and Edelman 2008:** Izhikevich, Eugene M., and Gerald M. Edelman Large-scale model of mammalian thalamocortical systems, *Proceedings of the national academy of sciences* 105.9 (2008): 3593-3598. Also: Supporting Information

**Izhikevich et al. 2004:** Izhikevich, Eugene M., Joseph A. Gally, and Gerald M. Edelman, "Spike-timing dynamics of neuronal groups", *Cerebral Cortex* 14.8 (2004): 933-944.

**Karnani et al. 2014:** Karnani, M. M., M. Agetsuma, and R. Yuste, "A blanket of inhibition: functional inferences from dense inhibitory connectivity," *Current opinion in Neurobiology* 26, pp. 96-102, 2014.

**Kermany et al. 2010:** Kermany, Einat, et al., Tradeoffs and Constraints on Neural Representation in Networks of Cortical Neurons –*The Journal of Neuroscience* 30.28 (2010): 9588-9596.

**Kheradpisheh et al. 2016a:** Kheradpisheh, S. R., M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing* 205, pp. 382-392, 2016.

**Kheradpisheh et al. 2016b:** Kheradpisheh, S. R., M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep neural networks for object recognition," *arXiv preprint arXiv:1611.01421*, 2016.

**Kistler et al. 1997:** Kistler, W. M., W. Gerstner, and J. L. van Hemmen, "Reduction of the Hodgkin-Huxley equations to a single-variable threshold model", *Neural Computation* 9.5, pp. 1015-1045, 1997.

**Körner et al. 1999:** Körner, Edgar, M-O. Gewaltig, Ursula Körner, Andreas Richter, and Tobias Rodemann. "A model of computation in neocortical architecture." *Neural Networks* 12, no. 7 (1999): 989-1005.

# Bibliography

---

- Laurent 1999:** Laurent, Gilles. "A systems perspective on early olfactory coding." *Science* 286, no. 5440 (1999): 723-728.
- LeCun et al. 2015:** LeCun, Y., Y. Bengio, and G. Hinton, "Deep learning," *Nature* 521, no. 7553, pp. 436-444, 2015.
- Lee et al. 2010:** Lee, Jungah, HyungGoo R. Kim, and Choongkil Lee. "Trial-to-trial variability of spike response of V1 and saccadic response time." *Journal of neurophysiology* 104, no. 5 (2010): 2556-2572.
- Lennie 2003:** Lennie, Peter. "The cost of cortical computation." *Current biology* 13, no. 6 (2003): 493-497.
- Lippmann 1987:** Lippmann, Richard, "An introduction to computing with neural nets" , *ASSP Magazine*, pp. 4-22, 1987.
- Madhavan et al. 2015:** Madhavan, A., T. Sherwood, and D. Strukov, "Race logic: abusing hardware race conditions to perform useful computation," *IEEE Micro* 35, no. 3, pp. 48-57, 2015.
- Maass 1995:** Maass, W., "On the computational complexity of networks of spiking neurons" (extended abstract), *Advances in Neural Information Processing Systems*, vol. 7, p. 183, 1995.
- Maass 1997:** Maass, W., "Networks of spiking neurons: the third generation of neural network models," *Neural Networks* 10.9, pp.1659-1671, 1997.
- Maass 1999:** Maass, W. Computing with Spiking Neurons, In *Pulsed Neural Networks*, W. Maass and C. M. Bishop, editors, pages 55-85. MIT Press (Cambridge), 1999.
- Mainen and Sejnowski 1995:** Mainen, Z. F., and T. J. Sejnowski, "Reliability of spike timing in neocortical neurons," *Science* 268, no. 5216 , pp. 1503-1506, 1995.
- Markram et al. 1997:** Markram, H., J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs," *Science* 275, no. 5297, pp. 213-215, 1997.
- Markram et al. 2004:** Markram, Henry, Maria Toledo-Rodriguez, Yun Wang, Anirudh Gupta, Gilad Silberberg, and Caizhi Wu. "Interneurons of the neocortical inhibitory system." *Nature Reviews Neuroscience* 5, no. 10 (2004): 793-807.

# Bibliography

---

**Masquelier and Thorpe 2010:** Masquelier, T., and S. J. Thorpe, "Learning to recognize objects using waves of spikes and Spike Timing-Dependent Plasticity," *2010 International Joint Conference on Neural Networks*, pp. 1-8, 2010.

**Masquelier and Thorpe 2007:** Masquelier, T., and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput Biol* 3, no. 2, e31, 2007.

**McKEnnoch et al. 2006:** McKEnnoch, S., D. Liu, and L. G. Bushnell, "Fast modifications of the spikeprop algorithm," *International Joint Conference on Neural Networks*, pp. 3970-3977, 2006.

**Mead 1990:** Mead, Carver. "Neuromorphic electronic systems." *Proceedings of the IEEE* 78, no. 10 (1990): 1629-1636.

**Merolla et al. 2011:** Merolla, Paul, John Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, and Dharmendra S. Modha. "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm." In *2011 IEEE custom integrated circuits conference (CICC)*, pp. 1-4. IEEE, 2011.

**Morrison et al. 2008:** Morrison, A., M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological Cybernetics* 98, no. 6, pp. 459-478, 2008.

**Mountcastle 1997:** Mountcastle, V., "The Columnar organization of the neocortex," *Brain* 120, no. 4, pp. 701-722, 1997.

**NAE:** NAE Grand Challenges for Engineering, "Reverse Engineer the Brain", <http://www.engineeringchallenges.org/challenges/9109.aspx>

**Natschläger and Ruf 1998:** Natschläger, T., and B. Ruf, "Spatial and temporal pattern analysis via spiking neurons," *Network: Computation in Neural Systems* 9, no. 3, pp. 319-332, 1998.

**Natschläger and Ruf 1999:** Natschläger,., and B. Ruf, "Pattern analysis with spiking neurons using delay coding," *Neurocomputing* 26, pp. 463-469, 1999.

**Oberlander, et al. 2012:** Oberlander, Marcel, et al. "Cell Type-Specific Three-Dimensional Structure of Thalamocortical Circuits in a Column of Rat Vibrissal Cortex." *Cerebral Cortex* 22.10 (2012): 2375-2391.

**O'Connor et al. 2013:** O'Connor, P., D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience* 7, 2013.

# Bibliography

---

**Perea et al. 2009:** Perea, Gertrudis, Marta Navarrete, and Alfonso Araque. "Tripartite synapses: astrocytes process and control synaptic information." *Trends in neurosciences* 32, no. 8 (2009): 421-431.

**Perin et al. 2011:** Perin, Rodrigo, Thomas K. Berger, and Henry Markram. "A synaptic organizing principle for cortical neuronal groups " *Proceedings of the National Academy of Sciences* 108, no. 13 (2011): 5419-5424. – 2011.

**Petersen et al. 2001:** Petersen, Rasmus S., Stefano Panzeri, and Mathew E. Diamond, Population Coding of Stimulus Location in Rat Somatosensory Cortex -- *Neuron* 32.3 (2001): 503-514.

**Pfeil et al. 2012:** Pfeil, T., T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier, "Is a 4-bit synaptic weight resolution enough?—constraints on enabling spike-timing dependent plasticity in neuromorphic hardware," *Frontiers in Neuroscience* 6, 2012.

**Probst et al. 2012:** Probst, Dimitri, et al. , "Liquid Computing in a Simplified Model of Cortical Layer IV: Learning to Balance a Ball", *Artificial Neural Networks and Machine Learning–ICANN 2012* (2012): 209-216.

**Querlioz et al. 2013:** Querlioz, D., O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Transactions on Nanotechnology* 12, no. 3, pp. 288-295, 2013.

**Rauch et al. 2003:** Rauch, A., G. La Camera, H. R. Luscher, W. Senn, and S. Fusi, "Neocortical pyramidal cells respond as integrate-and-fire neurons to in vivo-like input currents," *Journal of Neurophysiology* 90, no. 3, pp. 1598-1612, 2003.

**Rinkus, 2010:** Rinkus, Gerard J. "A cortical sparse distributed coding model linking mini-and macrocolumn-scale functionality." *Frontiers in Neuroanatomy* 4 (2010): 17.

**Salakhutdinov et al. 2007:** Salakhutdinov, R., A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," *24th International Conference on Machine Learning*, pp. 791-798, 2007.

**Schemmel et al. 2010:** Schemmel, Johannes, D. Bruderle, A. Grubl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. "A wafer-scale neuromorphic hardware system for large-scale neural modeling." In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 1947-1950. IEEE, 2010.

**Schrauwen and Van Campenhout 2004:** Schrauwen, B., and J. Van Campenhout, "Extending spikeprop," *IEEE International Joint Conference on Neural Networks*, pp. 471-475, 2004.

# Bibliography

---

**Singer 2010:** Singer, W., "Neocortical rhythms," *Dynamic Coordination in the Brain: From Neurons to Mind*, eds. Von der Malsburg, Christoph, William A. Phillips, and Wolf Singer. MIT Press, 2010.

**Smith 2014:** Smith, James E. "Efficient digital neurons for large scale cortical architectures." In *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 229-240. IEEE Press, 2014.

**Stein 1965:** Stein, R. B. "A theoretical analysis of neuronal variability," *Biophysical Journal* 5.2, pp. 173-194, 1965.

**Thorpe and Imbert 1989:** Thorpe, S. J., and M. Imbert, "Biological constraints on connectionist modelling," *Connectionism in Perspective*, pp. 63-92, 1989.

**Thorpe et al. 2001:** Thorpe, S., A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks* 14.6-7, pp. 715-725, 2001.

**Tryon 1962:** J. C. Tryon, "Quadded logic" in *Redundancy Techniques for Computing Systems*, Washington, DC:Spartan, pp. 205-228, 1962.

**Tuckwell 1979:** Tuckwell, H. C. "Synaptic transmission in a model for stochastic neural activity," *Journal of Theoretical Biology* 77.1, pp. 65-81, 1979.

**Weidenbacher and Neumann 2008:** Weidenbacher, U., and H. Neumann, "Unsupervised learning of head pose through spike-timing dependent plasticity," in *Perception in Multimodal Dialogue Systems, ser*, Lecture Notes in Computer Science. Springer Berlin 1 Heidelberg, vol. 5078/2008, pp. 123-131, 2008.

**VanRullen et al. 2005:** VanRullen, R., R. Guyonneau, and S. J. Thorpe, "Spike Times Make Sense", *Trends in neurosciences* 28.1, pp. 1-4, 2005.

**Wysoski et al. 2010:** Wysoski, S. G., L. Benuskova, and N. Kasabov, "Evolving spiking neural networks for audiovisual information processing," *Neural Networks* 23, pp. 819-835, 2010.

**Yuste 2015:** Yuste, R., "From the neuron doctrine to neural networks," *Nature Reviews Neuroscience* 16, no. 8, pp. 487-497, 2015.

**Zhao et al. 2015:** Zhao, B., R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward categorization on AER motion events using cortex-like features in a spiking neural network," *IEEE Transactions on Neural Networks and Learning Systems* 26, no. 9, pp. 1963-1978, 2015.