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

BETA – Feb. 2018

J E Smith

*Missoula, MT*

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?"***

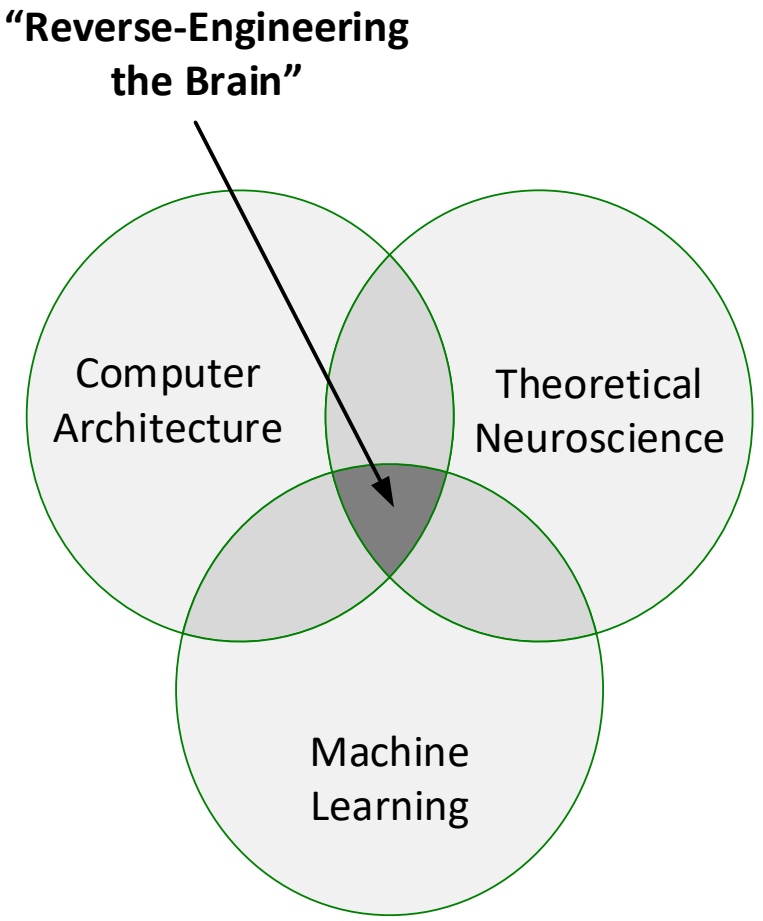
**-- Workshop report: Mary J. Irwin and John P. Shen, "Revitalizing computer architecture research," Computing Research Association (2005).**

# Introduction

- The human brain is capable of:
  - Accurate sensory perception
  - High level reasoning and problem solving
  - A wide variety of other cognitive skills
- With some very impressive features:
  - Highly efficient
  - Very flexible
  - 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 look at new approaches based on the way the brain actually works?
- After the tutorial, it is hoped that attendees will:
  - Better 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.

# Perspective

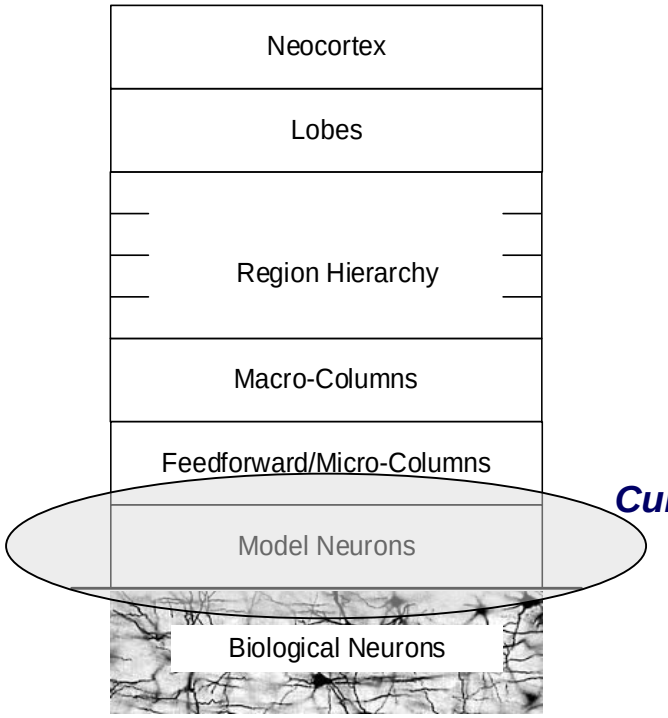
- This research is at the intersection of *computer architecture*, *theoretical neuroscience*, and *machine learning*
  - It is of *major* interest to all three
- Requires some knowledge of all three
  - Most researchers are likely to be stronger in one of them
  - This will shape the perspective and 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*



# What does "Reverse-Engineering the Brain" mean?

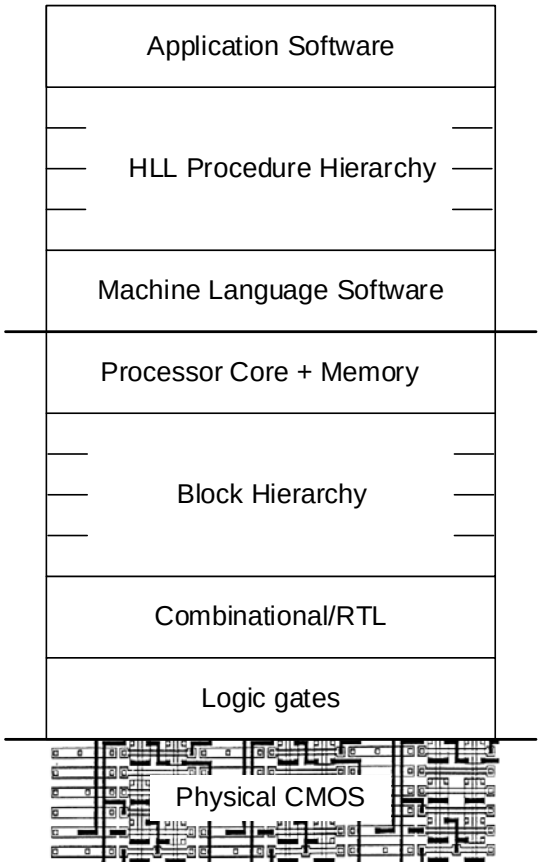
- "*Reverse-abstracting the neocortex*" may be a better way to put it
  - The *neocortex* is the part of the brain that is of interest
  - *Reverse-abstracting* is discovering the *layers of abstraction*, bottom-up

**Neuro Architecture Stack**




***Current Research***

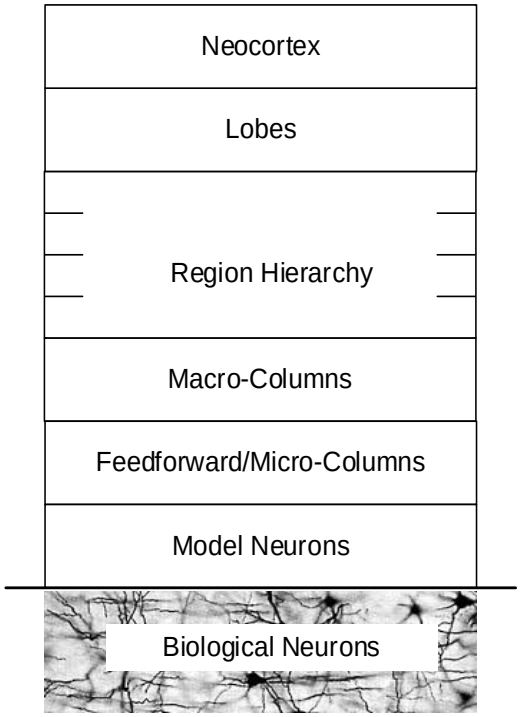
**Silicon Architecture Stack**



# 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 a science perspective:
  - Hypothesize some aspect of the computational paradigm
  - Look for experimental support for the hypothesis
  - Is there experimental support?
  - *The scientific perspective tends to be more downward looking*
- From an architecture perspective:
  - Hypothesize some aspect or feature of the computational paradigm
  - Construct and operate, via simulation, an idealized implementation

    one that exactly fits the hypothesis
  - Does it compute as expected?
  - *The architecture perspective tends to be more upward looking*



# Two Engineering Approaches

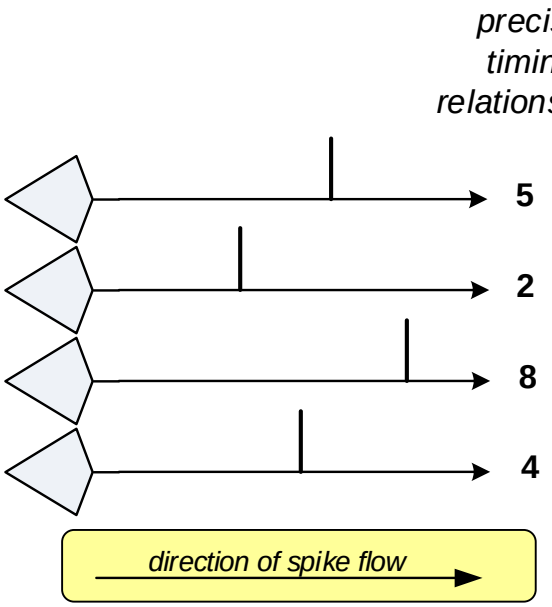
- Approach 1: Feature driven
  - Construct a detailed model that attempts to mimic biology in some way
  - Include any feature that may be significant (subjective)
  - Simulate it… what does it do?
  - Example: Izhikevich and Edelman large scale model (2004-2008)
  - Example: IBM TrueNorth (early "cat's brain" work)
- Approach 2: Hypothesis driven
  - Based on biological plausibility, hypothesize the computing paradigm that evolution may have been shooting for (given the materials and methods at its disposal)
  - Construct an idealized model of the computing paradigm
  - Simulate it… does it work as hypothesized?

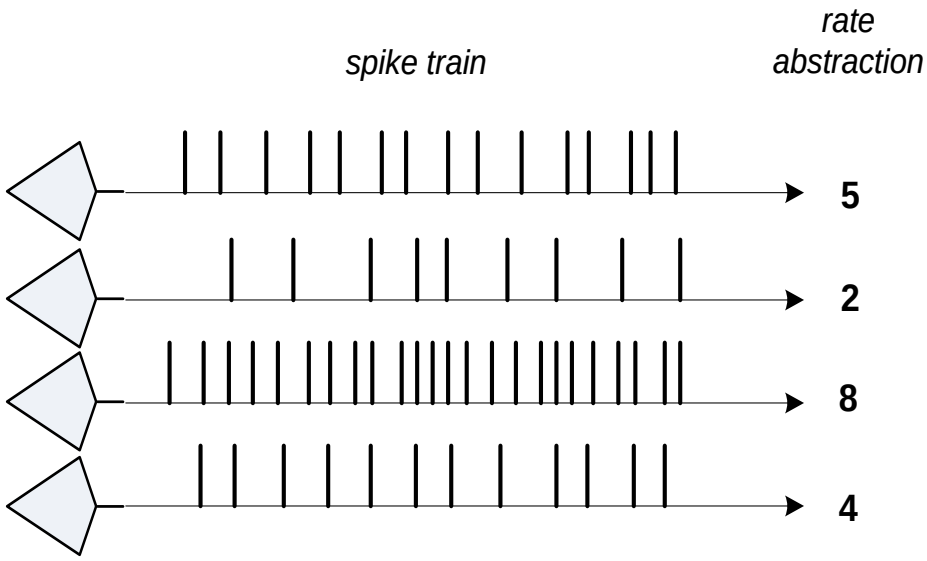
***This tutorial takes the hypothesis driven approach***

# Research Direction

# Spike Communication

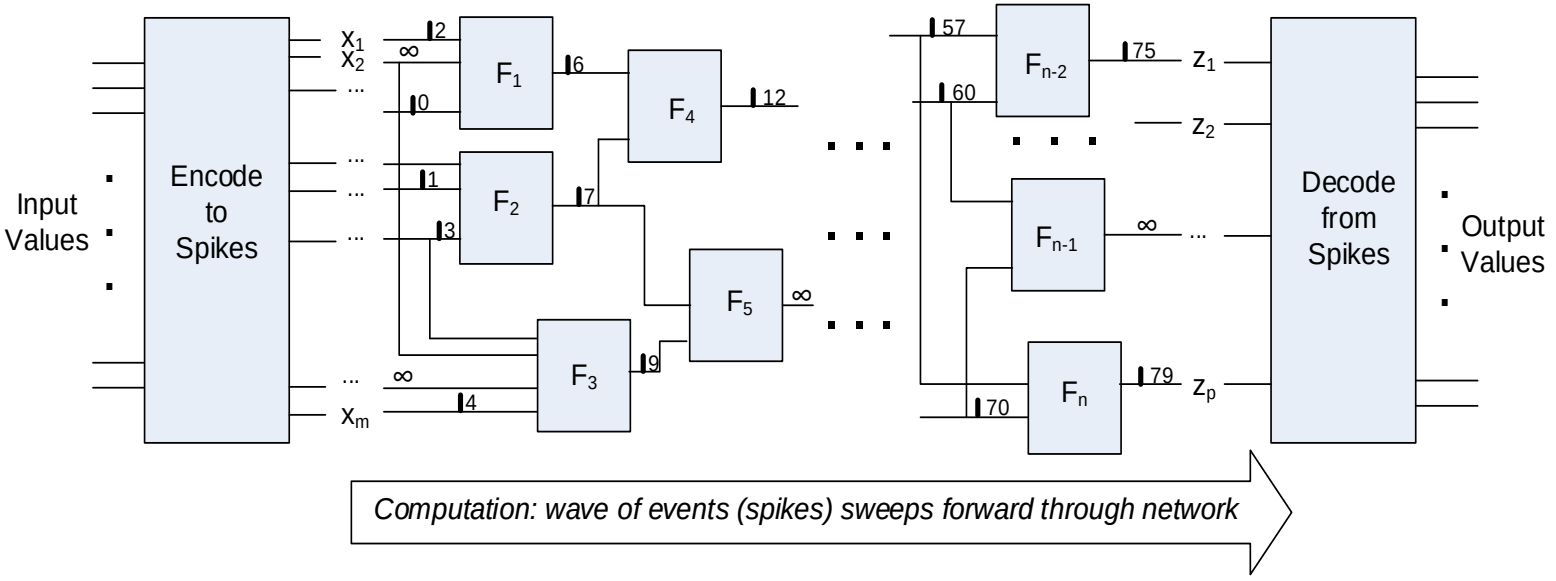
- Consider systems where data is communicated via transient events
  - e.g., voltage spikes
- Data values are encoded as temporal relationships across parallel communication lines
  - In contrast to using (spiking) event rates measured on individual lines
- The temporal communication method has significant experimental support



# Spike Computation

- Temporal Neural Networks
  - Networks of functional blocks whose input and output values are communicated via precise timing relationships
- A critical *first* step
  - At the forefront of current theoretical neuroscience
  - Feedforward flow (Feedback can be studied next)
  - A single wave of spikes passes from inputs to outputs



# Research Focus: *Milestone TNN*

- Most current neuroscientific TNN projects have the same target:
  - *Feedforward clustering networks* (unsupervised training)
- This seems to be a good choice
  - Should be sufficient for demonstrating the capabilities of TNNs
  - And highlight major differences wrt conventional machine learning methods

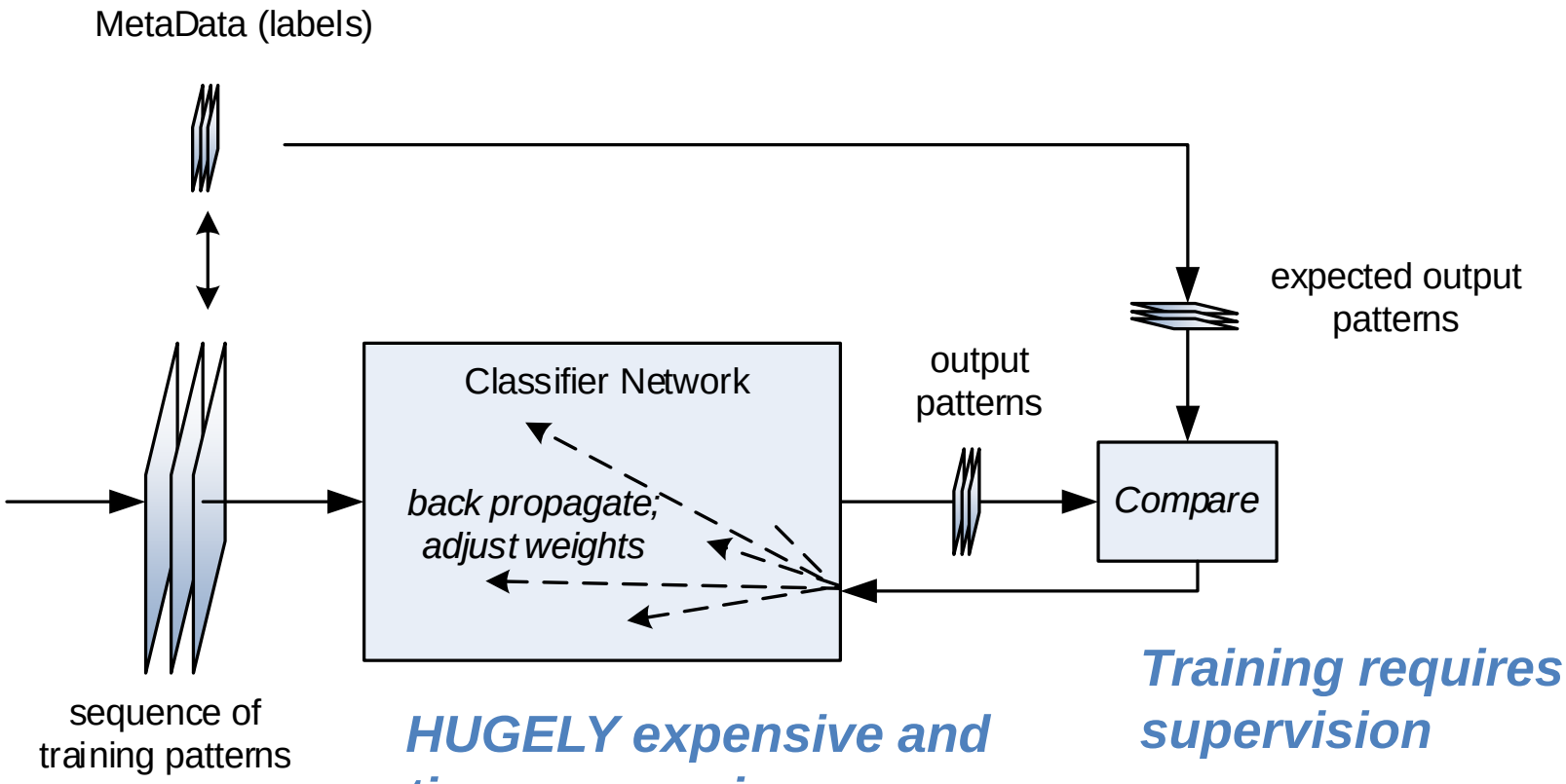
***First, consider the way conventional machine learning works***

# Conventional Machine Learning

- *aka* the *Bright Shiny Object*
- Three phases:
  - Training
  - Evaluation
  - Re-Training

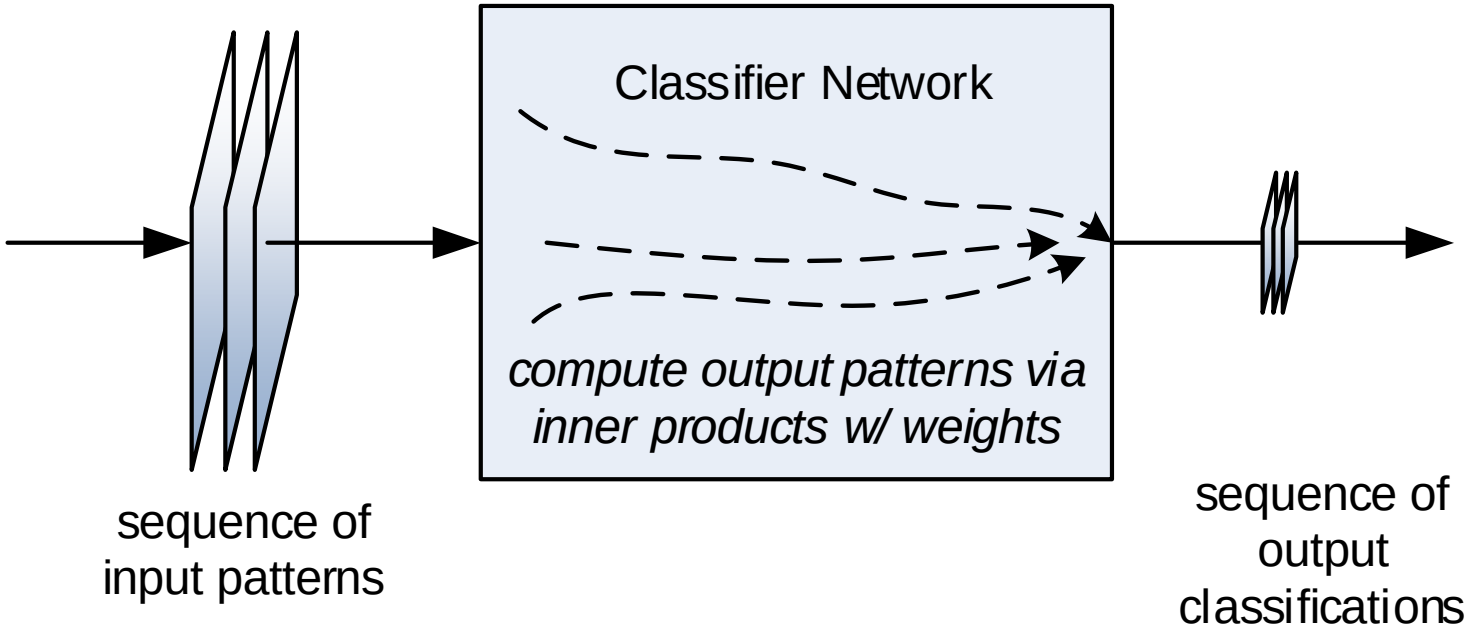
# Conventional Machine Learning: Training

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



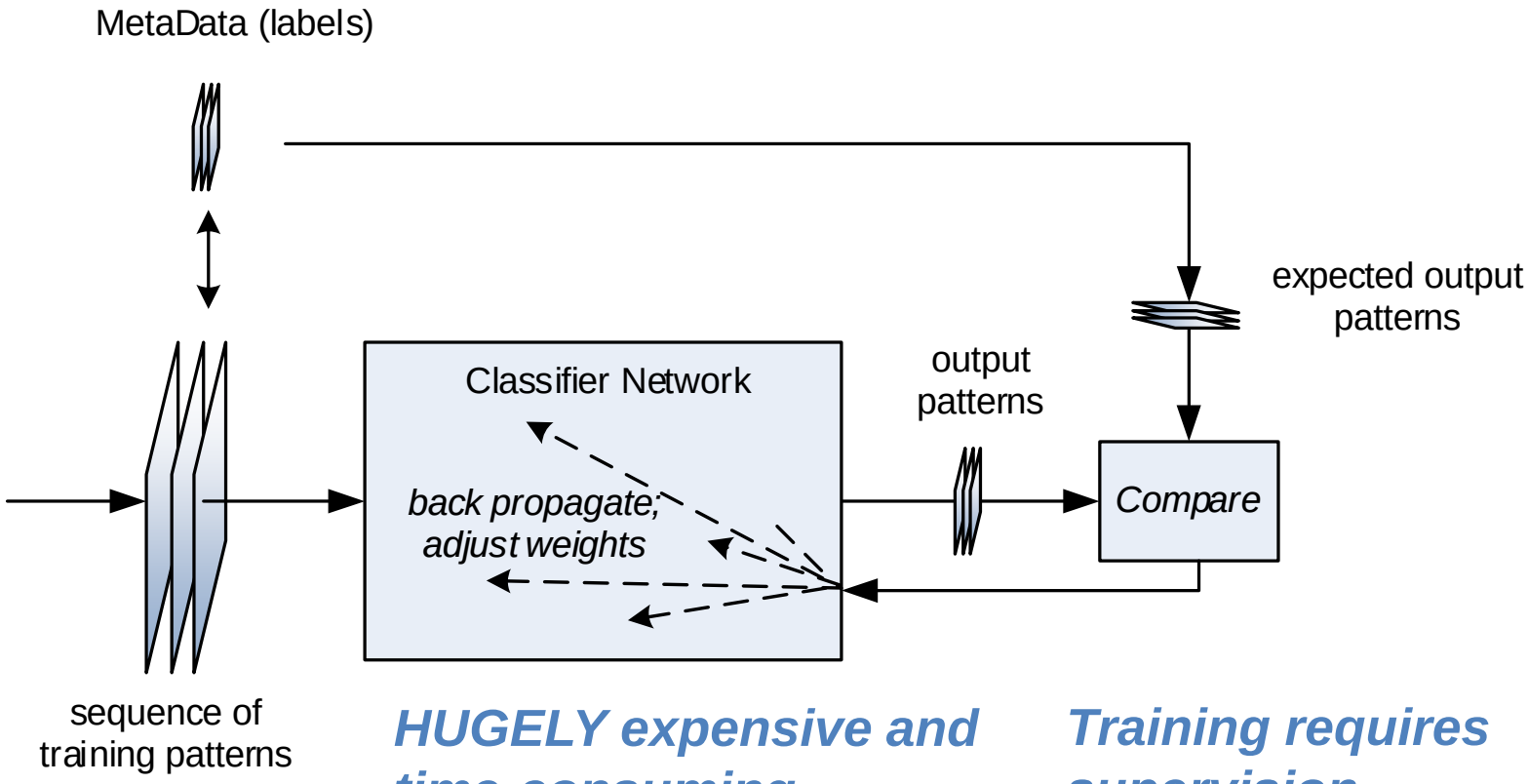
# Conventional Machine Learning: Evaluation (Inference)

- Evaluation process:
  - Apply input patterns, and compute output classes via vector inner products (inputs · 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

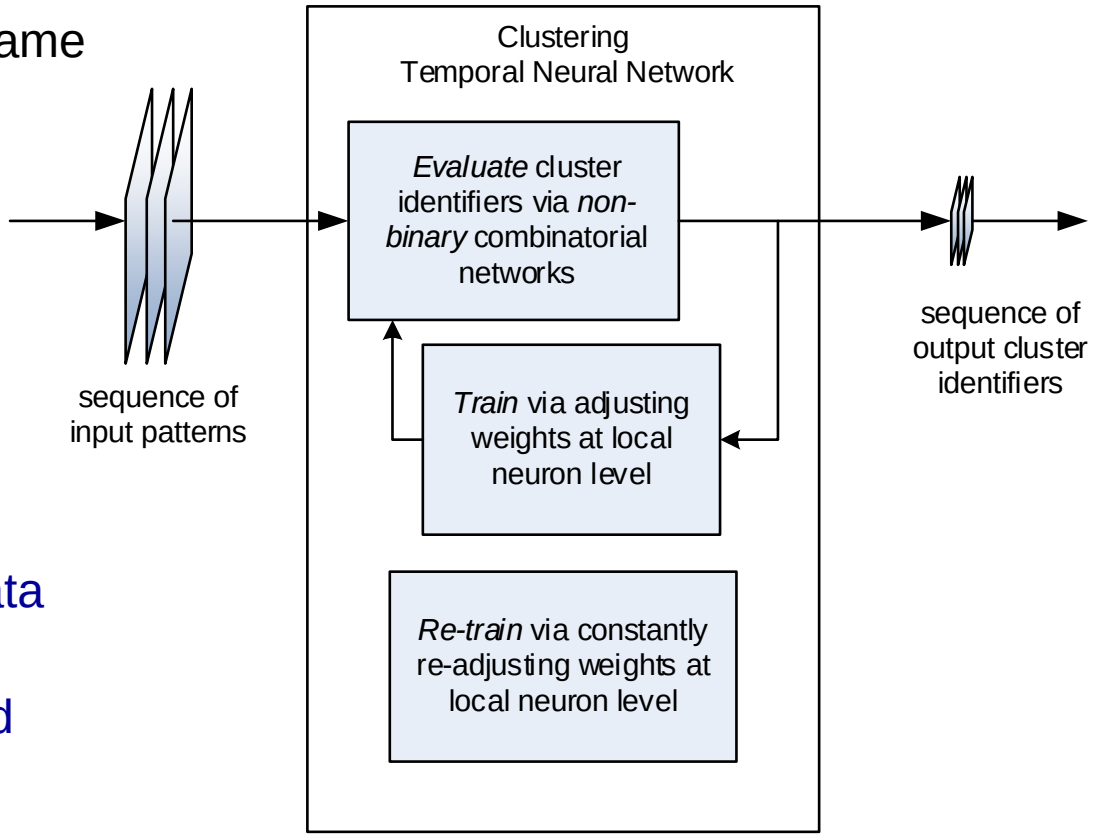


# Milestone Temporal Neural Network

- Performs *unsupervised clustering*
  - Not *supervised classification*
- Informally: A *cluster* is a group of *similar* input patterns
  - Roughly speaking: it's a grouping of input patterns based on their natural similarities
- A *clustering neural network* groups input patterns into clusters
  - Some outlier patterns *may* belong to *no* cluster
- Each cluster has an associated *cluster identifier,* which is a small, concise encoding
  - The mapping of cluster identifiers to clusters is determined by the network internally, without any outside intervention (there is no meta-data)
    - In our case, "similarity" is implied by the implementation
    - (Not an explicit, designer-supplied metric)
    - Network designer may control the maximum number of clusters
- For network outputs to be directly understood by humans, we must map cluster identifiers into known labels (decoding)

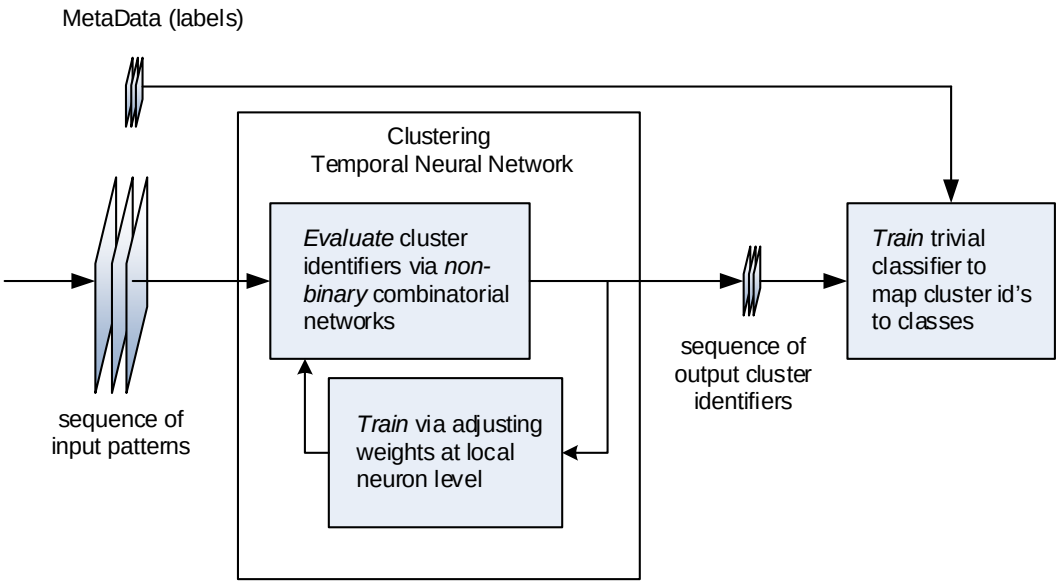
# Milestone Temporal Neural Network

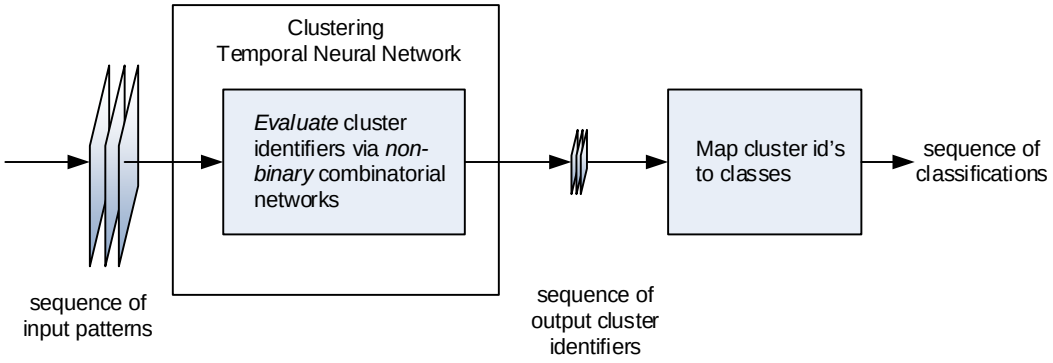
- Everything is done by the same network concurrently
  - Training
  - Evaluation
  - Re-training
- Features
  - Training is *unsupervised*
  - All computation is local
  - All very low resolution data
  - A *direct temporal implementation* may yield extremely good energy efficiency and speed



# Temporal Neural Network as a Classifier

- First: train unsupervised via sequence of input patterns
  - A single application should be sufficient
  - Synaptic weight training is localized and efficient
- Then: establish mapping of cluster id's to classes
  - Trivial classifier, e.g., 1-to-1 mapping
- Evaluate with training disabled



# Milestone Architecture Objectives

- To implement a brain-like function in a brain-like way
  - Takes a constant stream of input patterns and produces an output stream of cluster identifiers, all-the-while making concurrent, local synaptic weight adjustments
  - Automatically adjusts to changes in input patterns by changing clusters
  - Implementation is based on models that communicate and compute via transient temporal events (e.g., implementable via voltage spikes)
  - Simple computational elements (neurons) operate very quickly and energy efficiently

*Achieve brain-like capabilities and efficiencies using silicon-based technology*

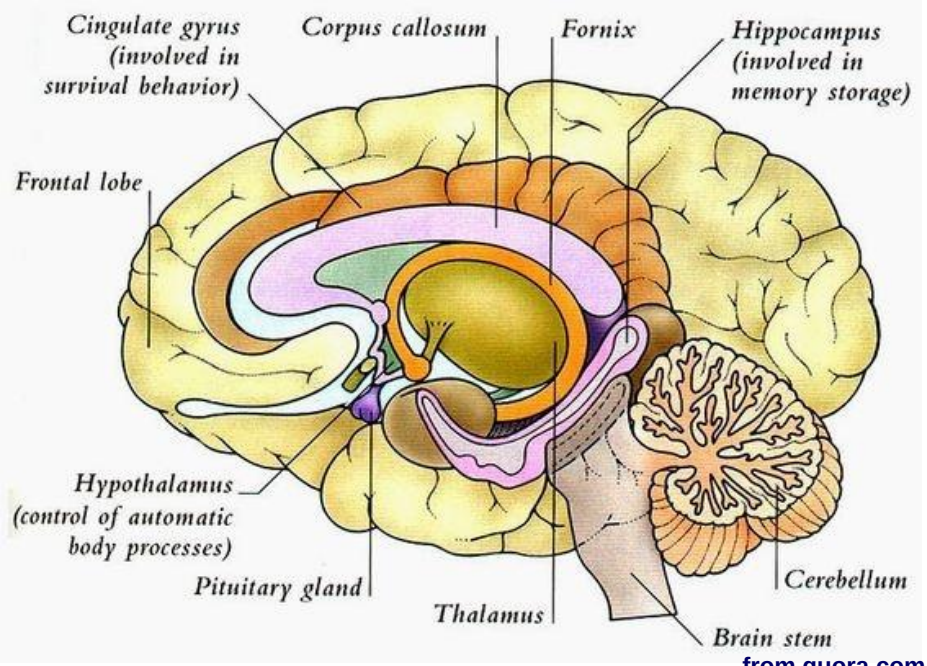
# Biological Background

## *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 motor commands*
- Hippocampus
  - Responsible for memory formation and sense of place
- Cerebellum
  - Responsible for motor control coordination

***Our primary interest is the neocortex***




from quora.com

# Neocortex

- Thin sheet of neurons
  - 2 to 3 mm thick
  - Area of about 2500 $cm^2$
    - Folds increase surface area
  - Approx 100 billion total neurons (human)
- Hierarchical Structure
  - Neurons
  - Micro-Columns
  - Macro-Columns
  - Regions
  - Lobes

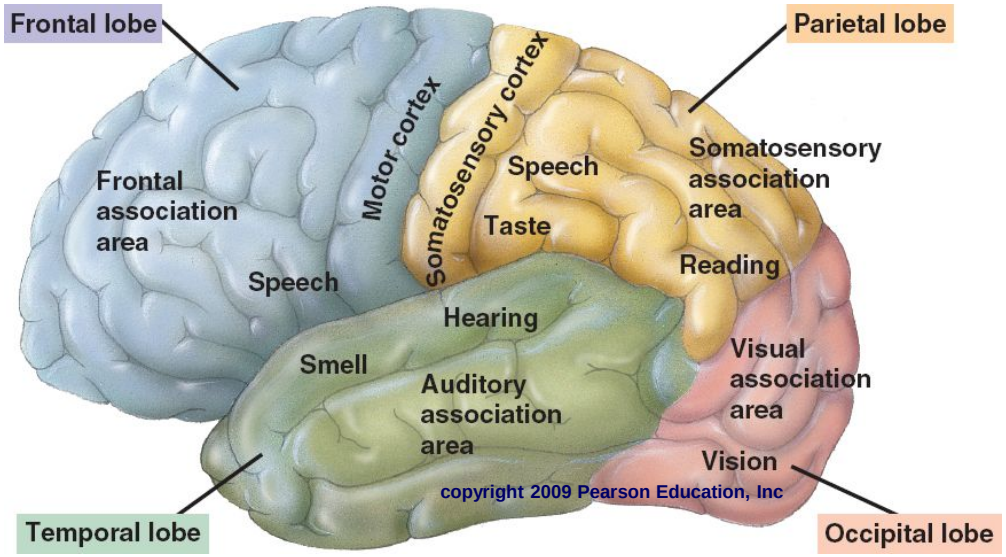



**illustration of regions and lobes**

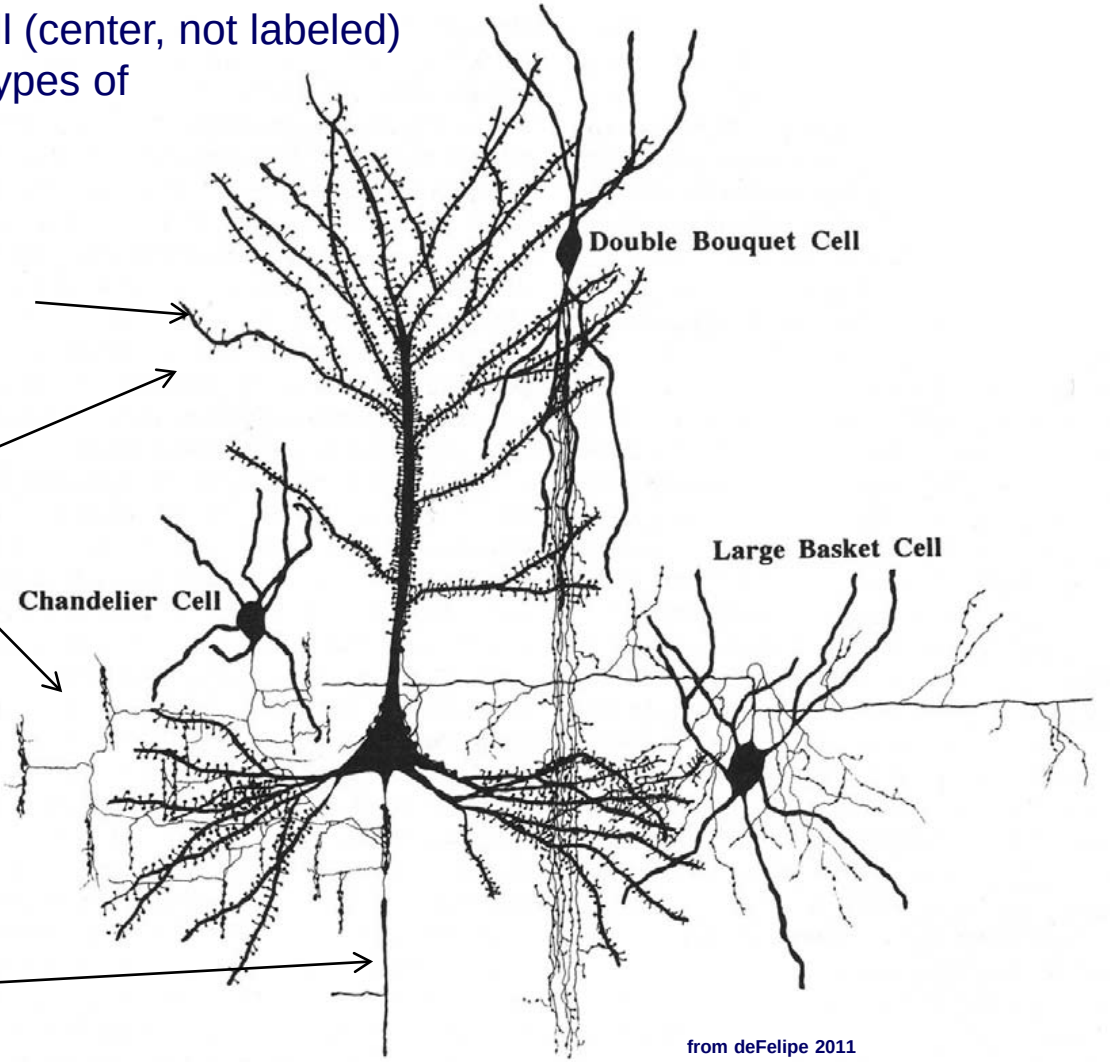
# Biological Neurons

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

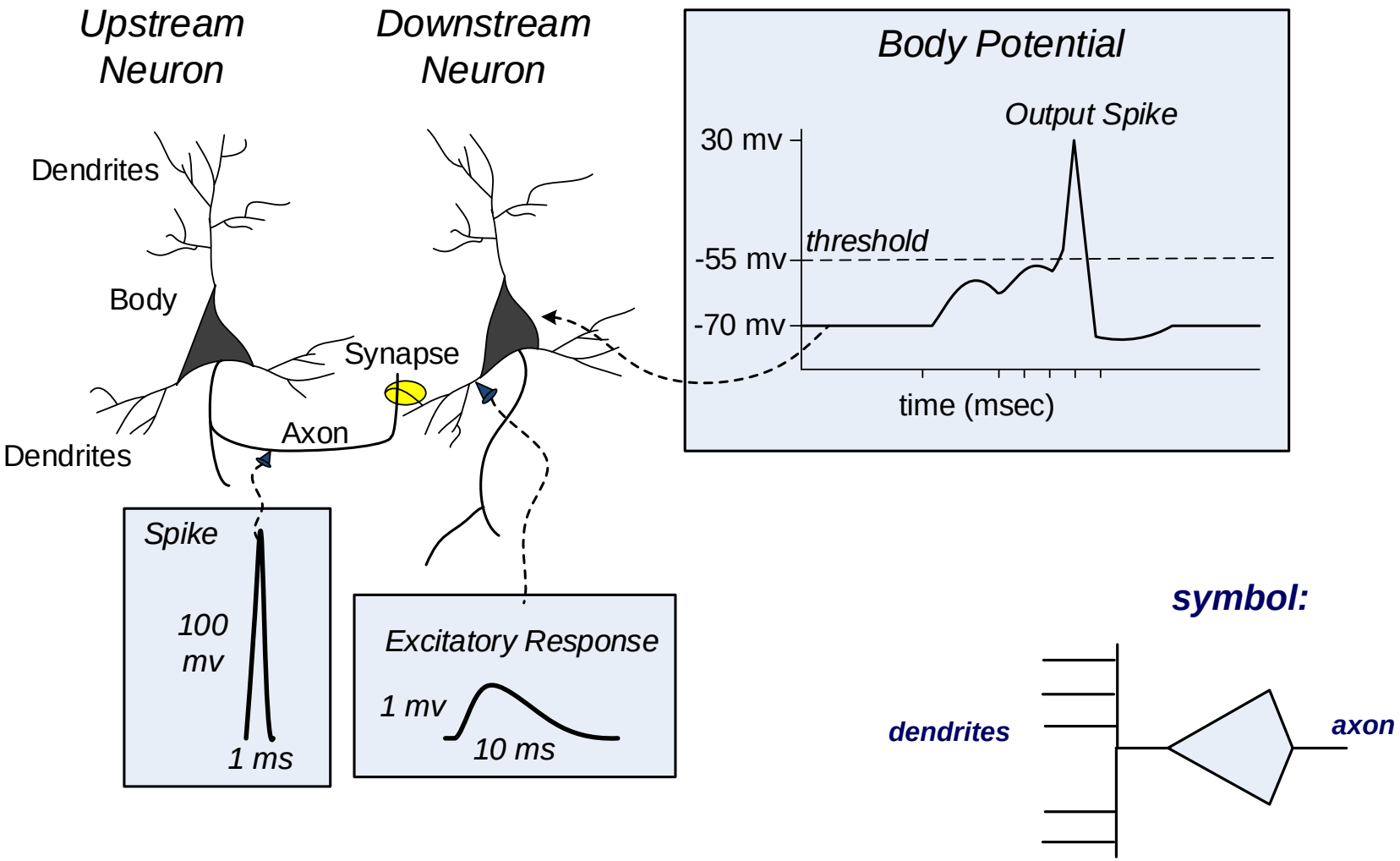
tiny dots are synapses (connection points)

Dendrites (Inputs)

Axon (Output)



# Neuron Operation



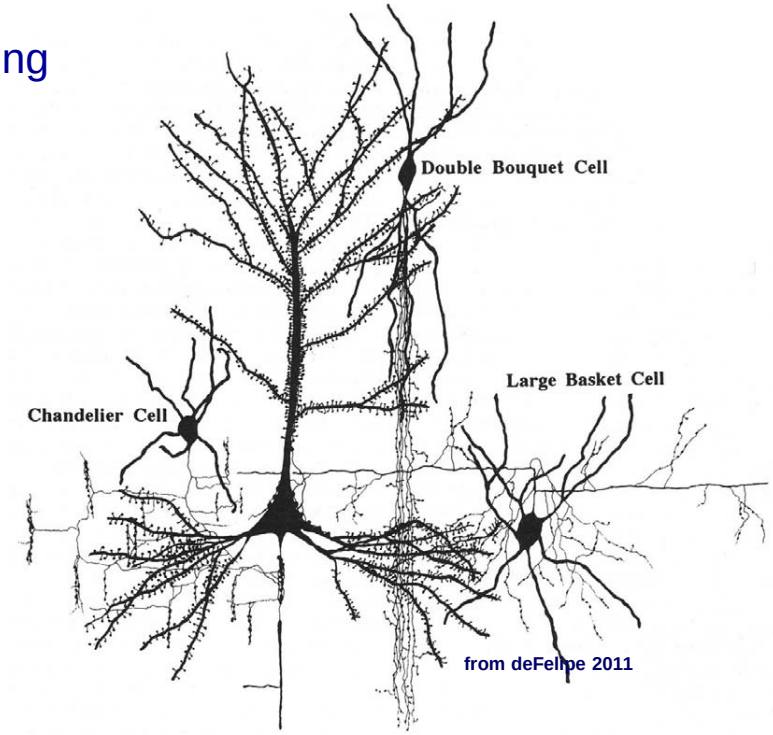
# Excitatory Neurons

- 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
    - And a lot of them are probably redundant for fault tolerance
  - Multiple synapses per connected neuron pair


from Perin et al. 2011

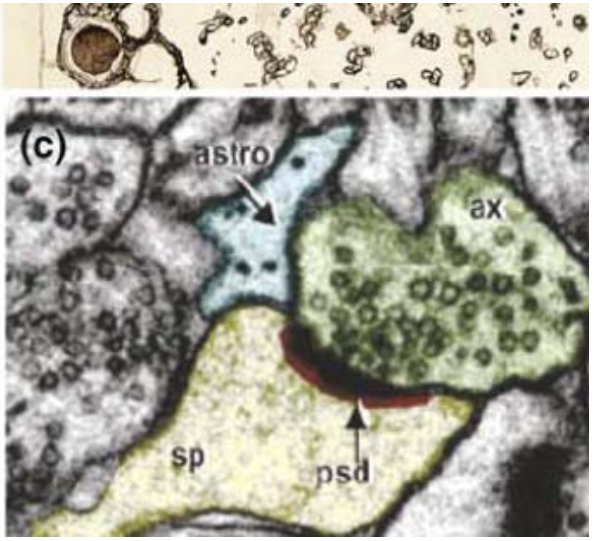
# Inhibitory Neurons

- Inhibitory response functions have the opposite polarity of excitatory neurons
  - *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*

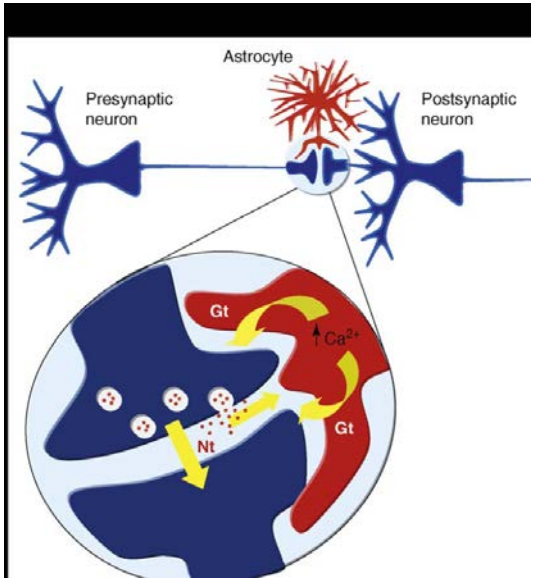


# Glial Cells

- 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)
  - See Perea et al. 2009



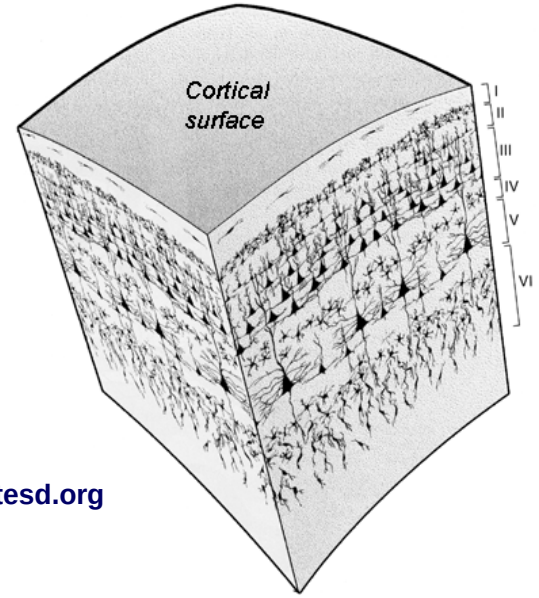

from Witcher et al. 2007



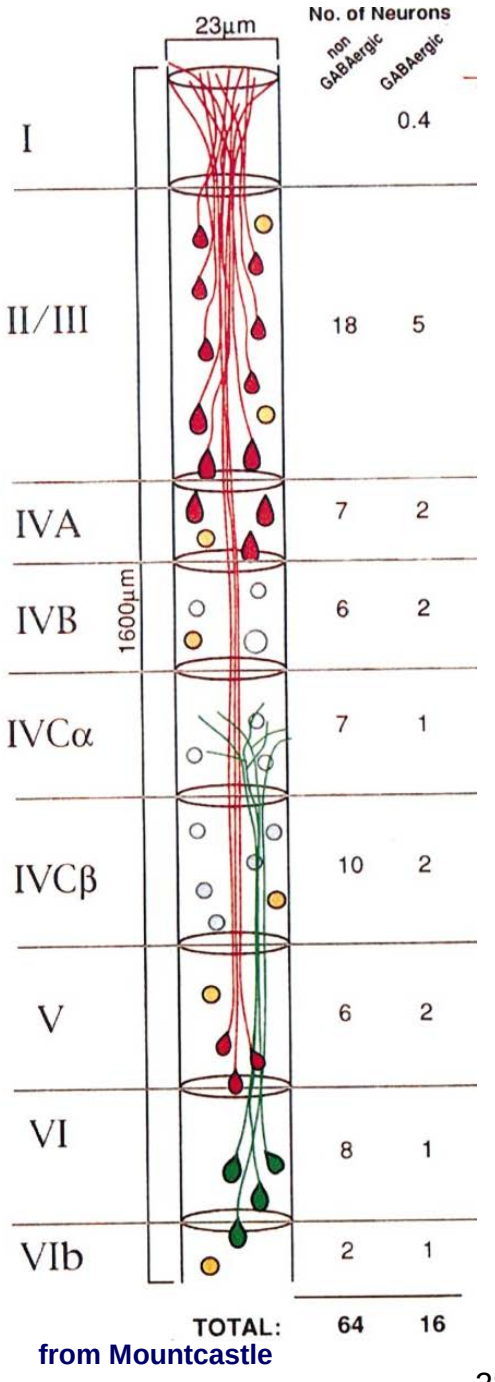

from Perea et al. 2009

# Columnar Structure

- Neurons are structurally organized as *columns* between the top and bottom neocortical surfaces
- Columns are organized into 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



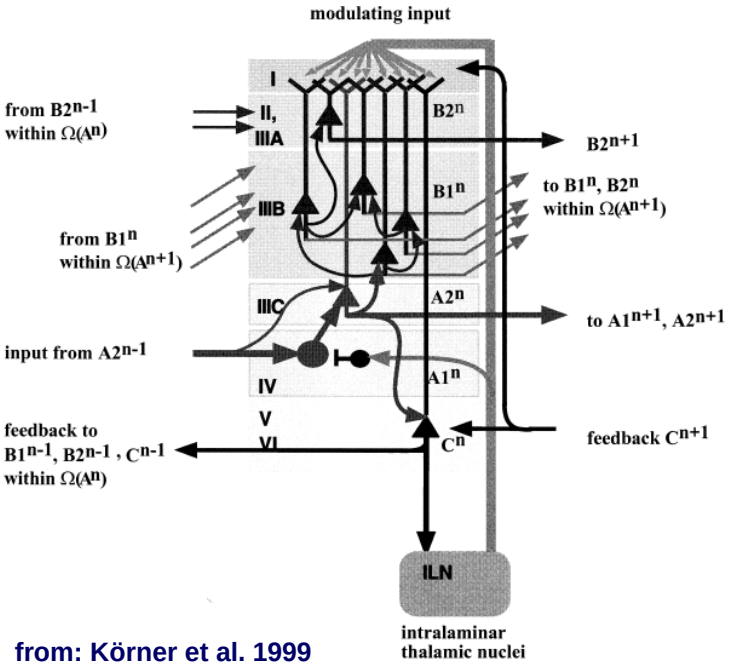

from https://neuwritesd.org



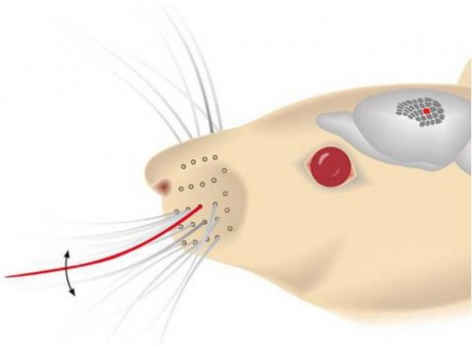

from Mountcastle

# Columnar Structure

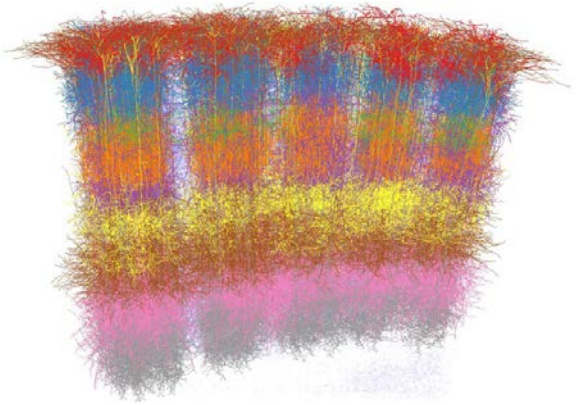
- Macro-Columns
  - O(100 micro-columns)
  - In sensory systems, all micro-columns in a macro-column map to same *receptive field (RF)*
  - An RF may be a patch of retina or skin, or a rat's whisker
- *Some* researchers construct architectures using the biological columnar structure as a guide
  - Hawkins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




from: Körner et al. 1999

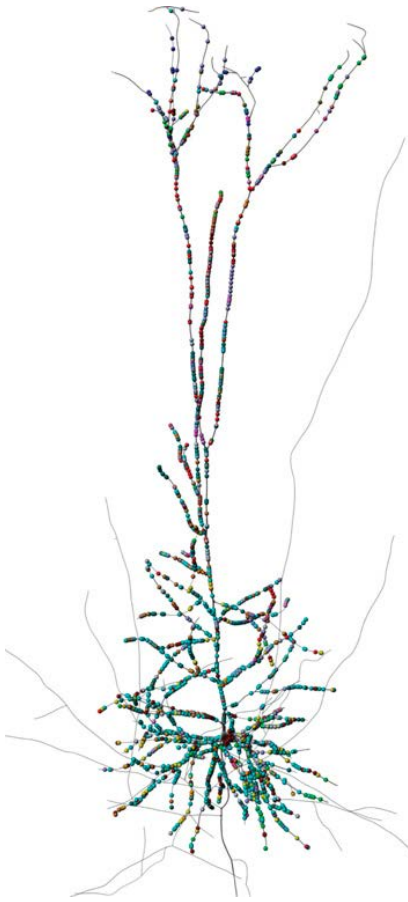


from: http://www.kurzweilai.net



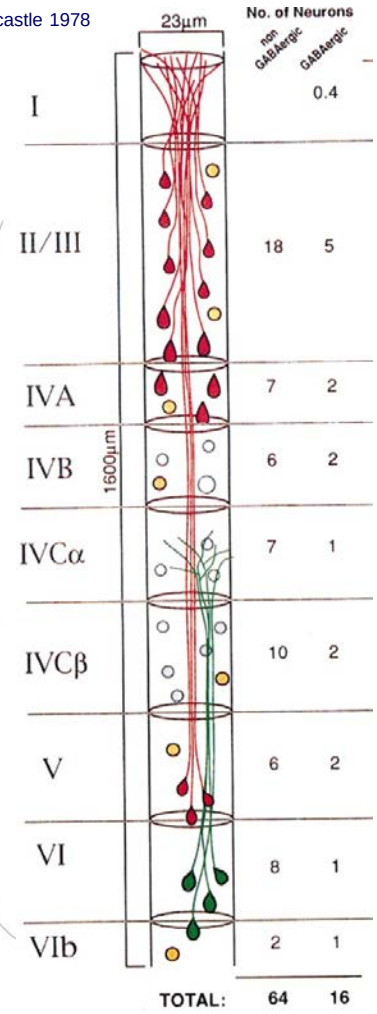
# Hierarchy

from Hill et al. 2012



Neuron

Mountcastle 1978

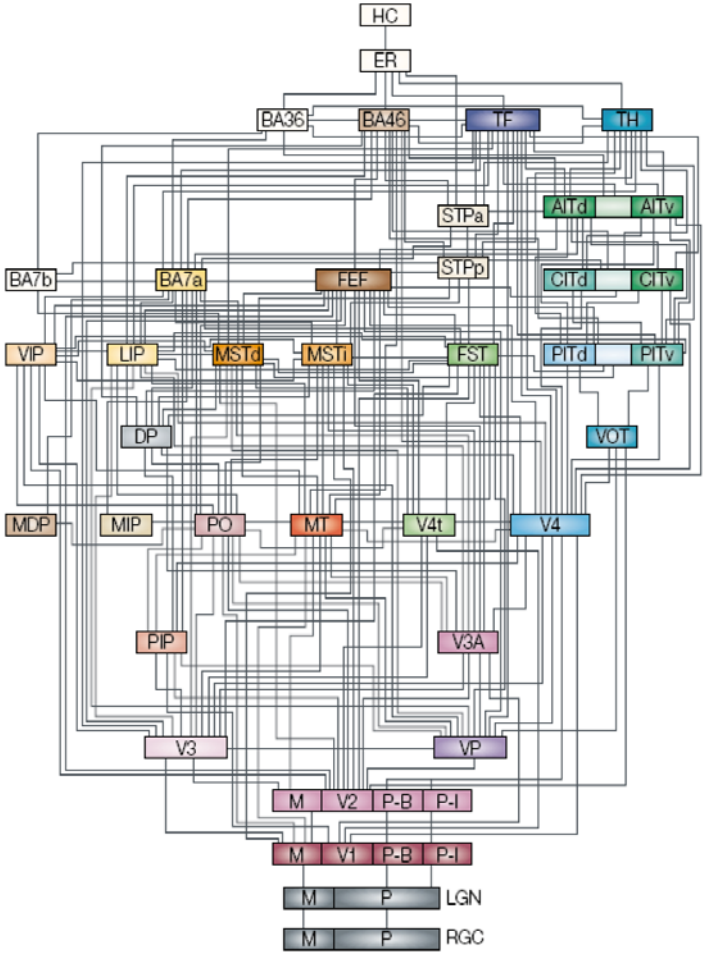



Micro-Column
O(100) neurons

from Ramon y Cajal (wikipedia)

Macro-Column
O(100) micro-columns

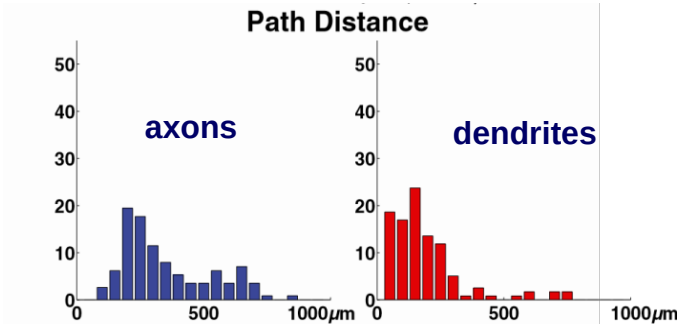
from Felleman and Van Essen (1997)



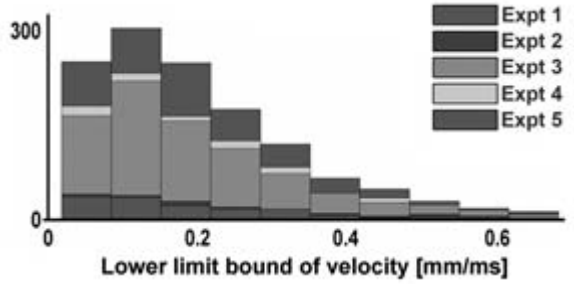

Regions
Many Macro-Columns

# Engineering Considerations

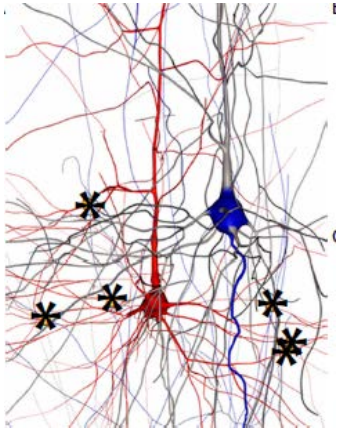
- Neuron latency
  - ***1's of milliseconds per neuron***
- Transmission delay
  - Path lengths order (100s um)
    or more
  - Prop. delay order (100s um per ms)
  - ***1's of milliseconds between two neurons***
- Multiple synapses per neuron pair
  - On the order of 5 -10



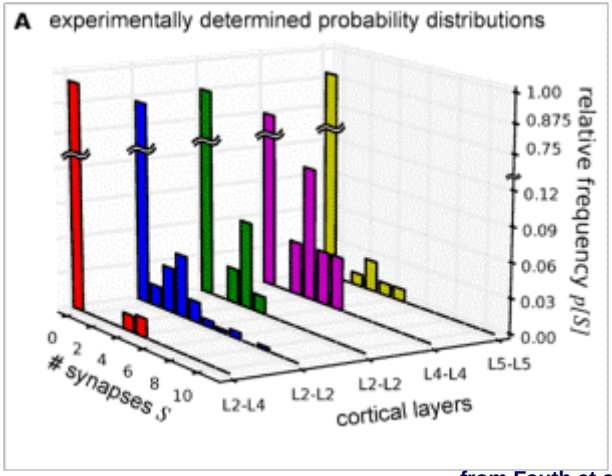

from Hill et al. 2012 – Markram group




from Bakkum et al. 2008



from Hill et al. 2012




from Fauth et al. 2015

# Neuron Doctrine

- The *neuron doctrine* serves as the foundation of 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, we will add several hypotheses, in sequence:
  - Uniformity Hypothesis
  - Temporal Hypothesis
  - Feedforward Hypothesis
  - Reliability Hypothesis
- Disclaimer: these hypotheses *are not* universally accepted and are often implicit

# 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
- Important implication: *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, 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* |
|---|---|---|---|
| | | *Human* | |
| 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 |
| | | *Rat* | |
| 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 |
| | | *Mouse* | |
| 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 |

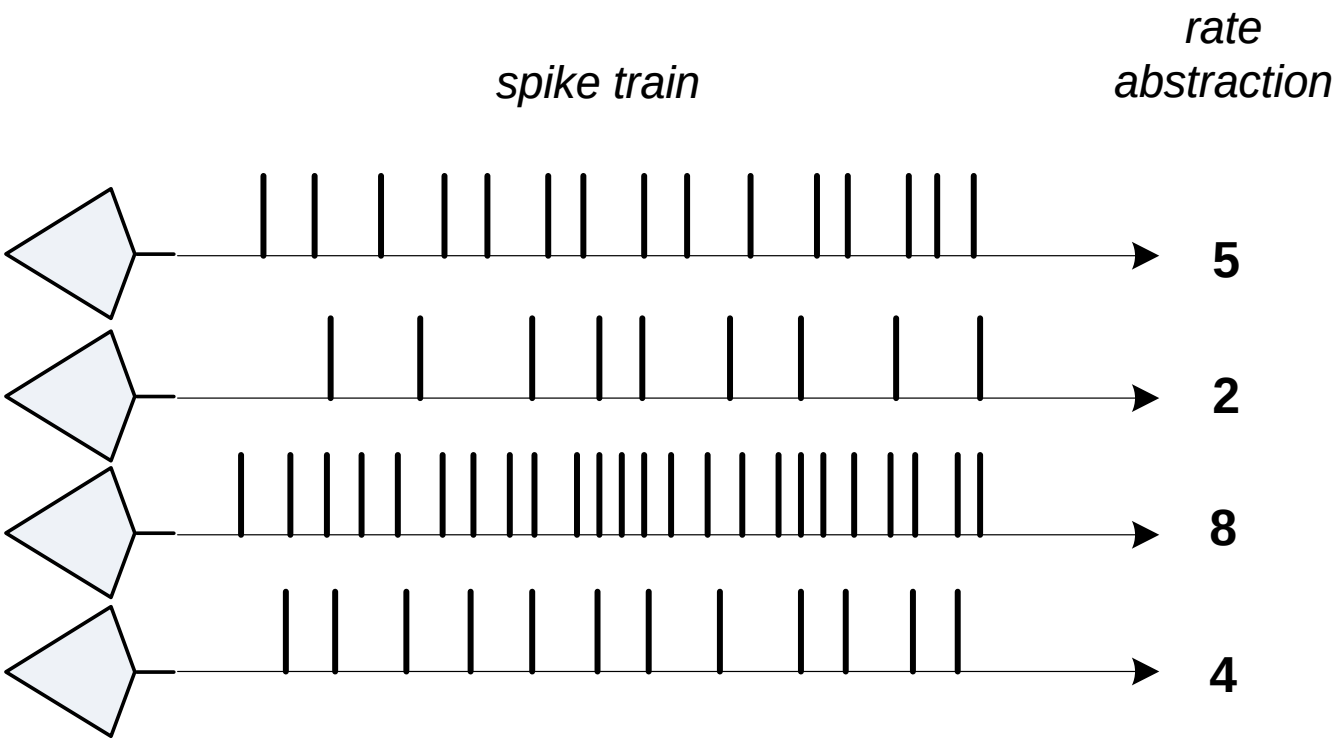
AS, asymmetrical synaptic profiles; SS, symmetrical synaptic profiles.

# Biological Background

## *Neural Information Coding*

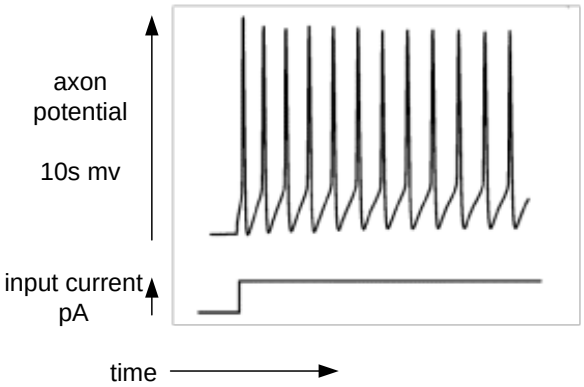
# Rate Coding

- 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

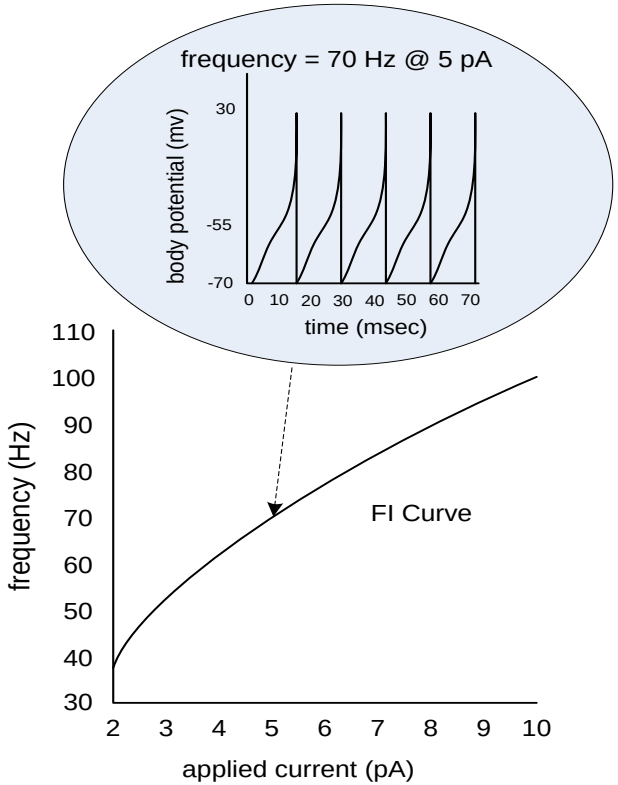


# Rate Coding

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

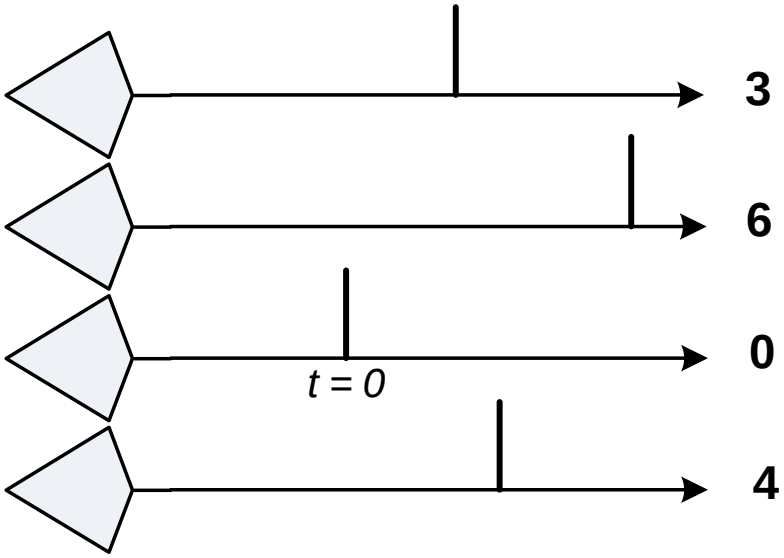



- Such constant current responses are often used to characterize neuron behavior
- Varying current step => Frequency-Current relationship (*FI Curve*)
- *Appears* to support rate coding
  - *The greater the input stimulus, the higher the output spike rate*



# Temporal Coding

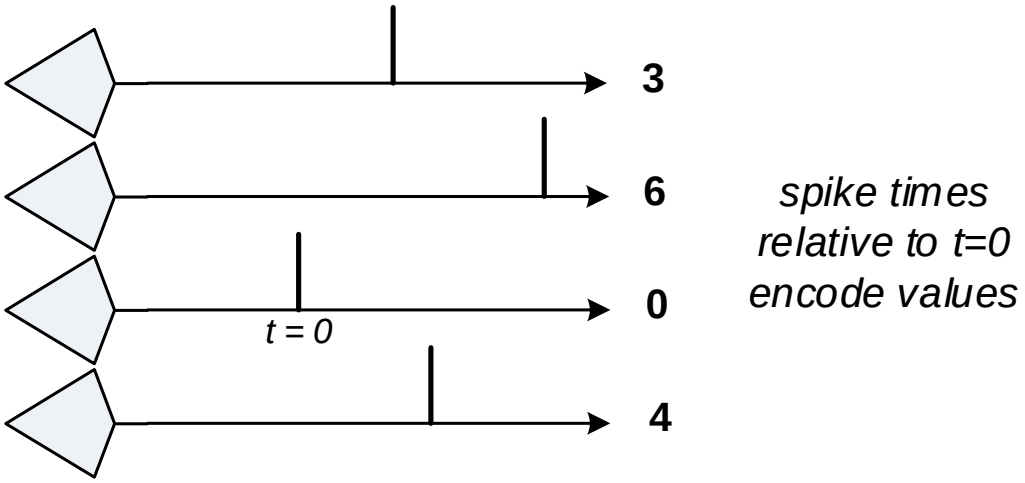
- Use relative timing relationships across multiple parallel spikes to encode information




*spike times relative to t=0 encode values*

# Temporal Hypothesis

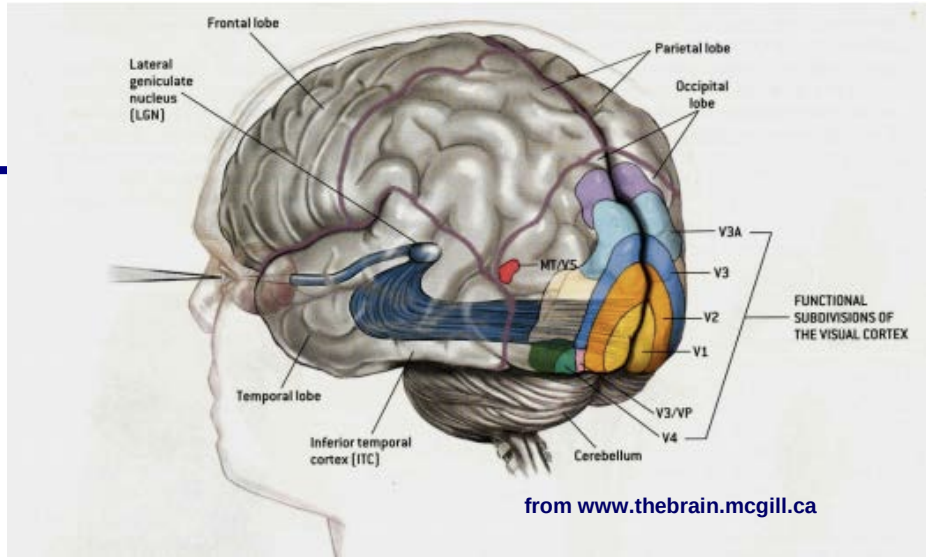
- Hypothesis: *Temporal Coding is used in the neocortex*
- Experimental results (to follow) support this hypothesis
  - Synopsis: temporal coding is much *faster*, much more *efficient*, and much more *plausible* than rate coding.

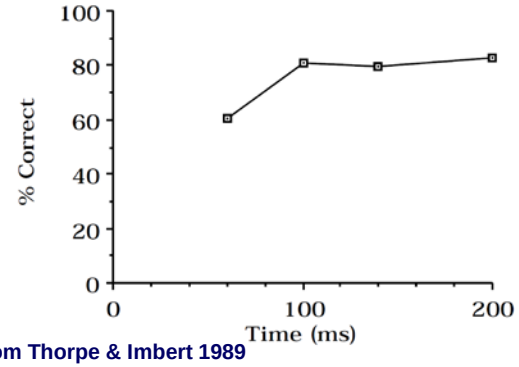


# Key Experiment

- 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

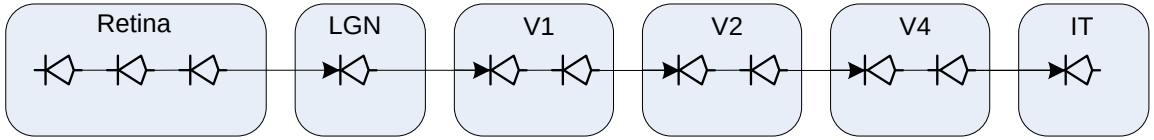


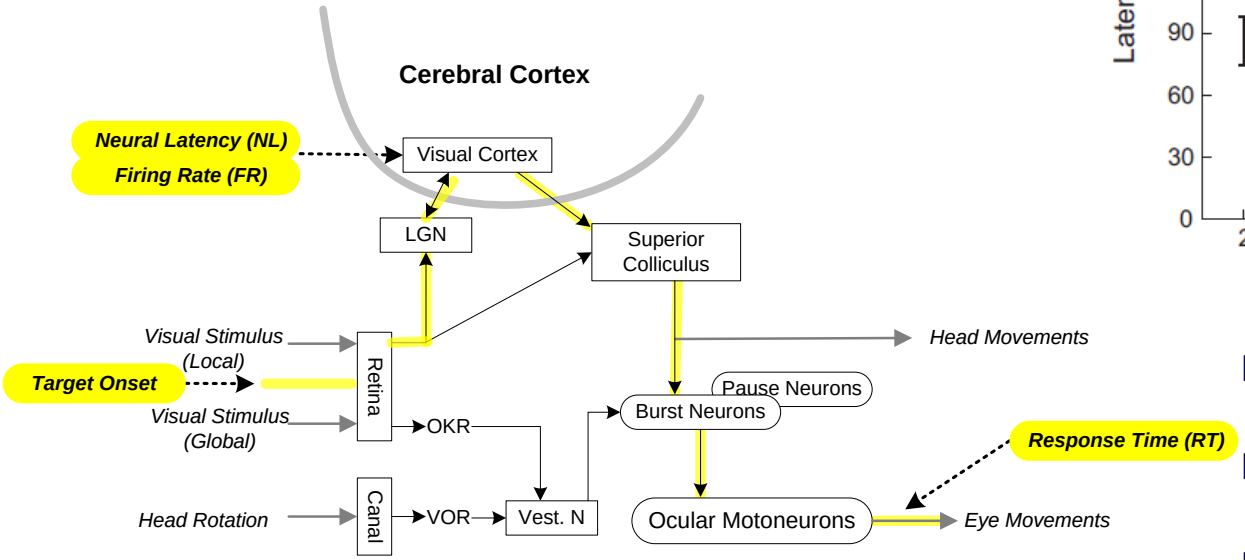

Image pathway from retina to IT, where image is identified

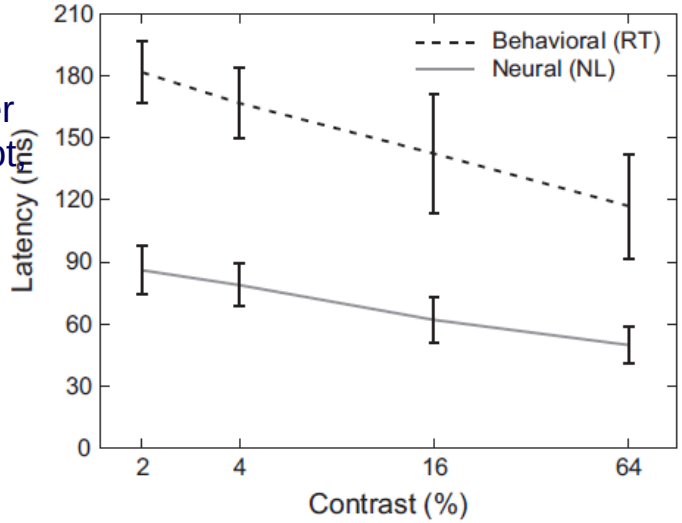
- 100 msec is sufficient in most cases (80%)
- At least 10 neurons in series along feedforward path, each consuming on the order of 10 msec (communication + computation).
- Conclusion: *Only first spike per line* can be communicated this quickly
  - Rates require a minimum of two spikes (probably several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

# 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

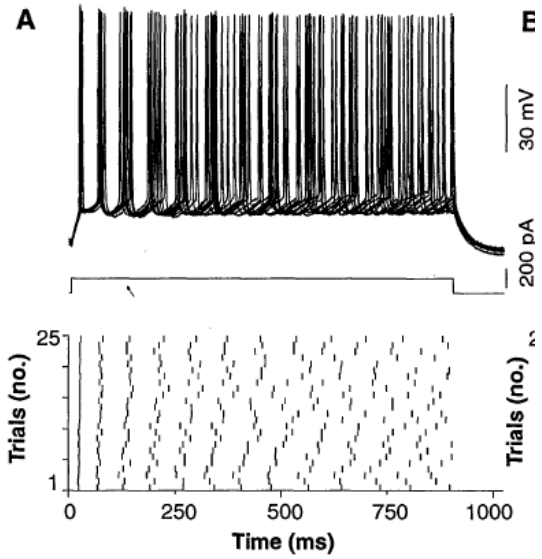
- Any organism that relies on its senses for survival depends on fast cognitive paths
- Experiments strongly suggest sensory input is communicated and processed as a *feedforward wavefront of precisely timed spikes*
  - *This is a very fast and efficient way to communicate information*
- Key question: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

  ***Conjecture: Yes, 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 building block?***

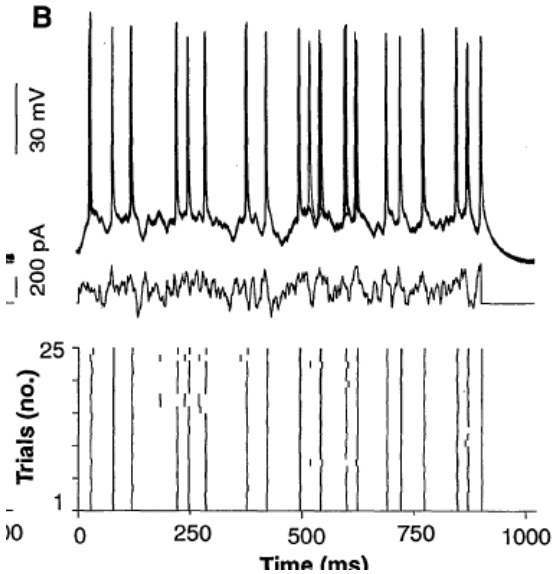
# Reliability of Spike Timing

- Mainen and Sejnowski (1995)
  - Repeatedly inject step current into neuron body
  - Capture and analyze output spike trains
  - Spike trains for first 10 trials and raster plots for first 25 trials




- Jitter => several spikes required to establish a reliable rate

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

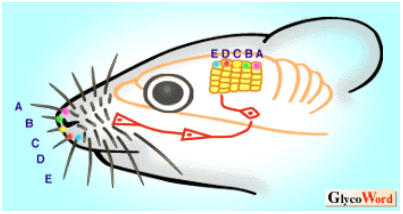



- Individual spikes are very reliable

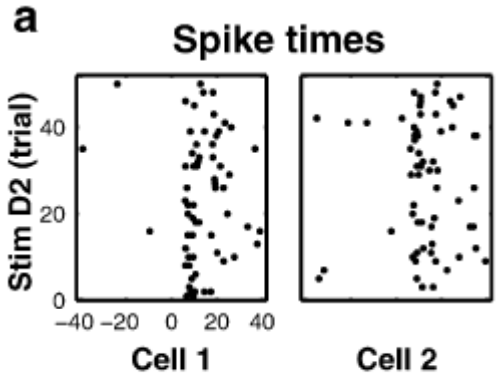
"*stimuli with fluctuations resembling synaptic activity produced spike trains with timing reproducible to less than 1 millisecond*"

# 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

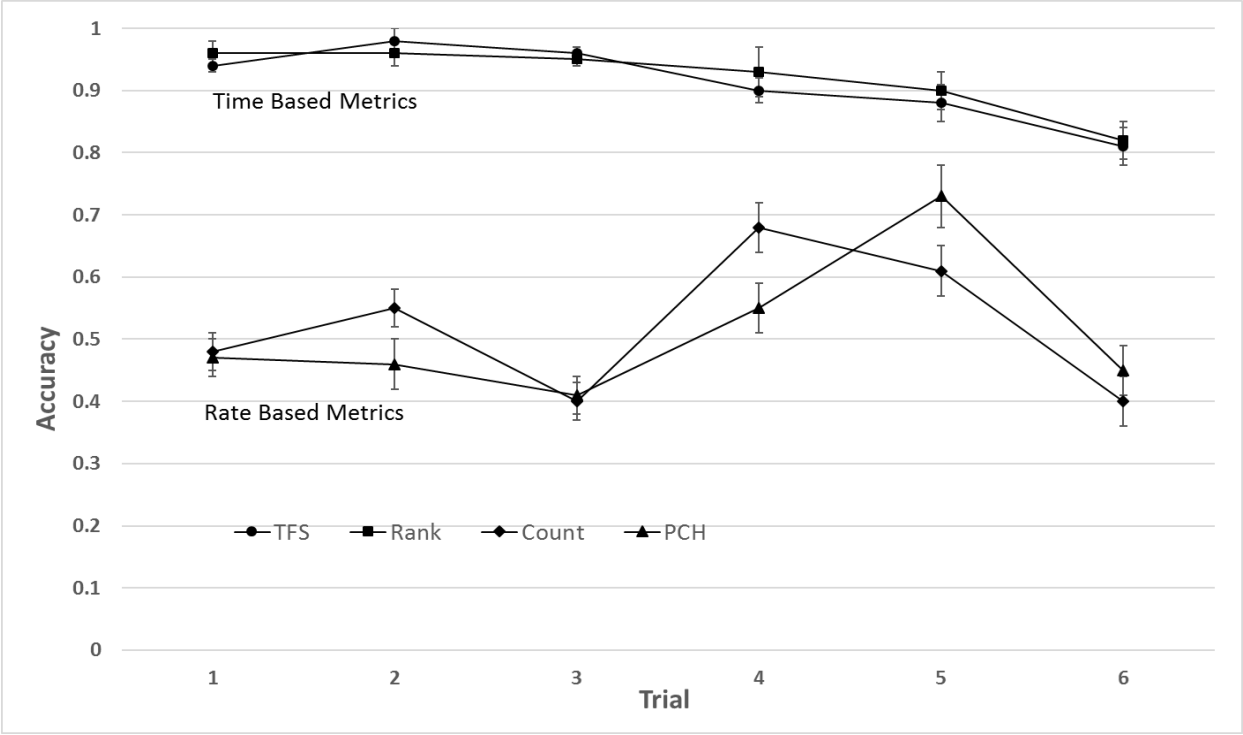



**units are .1 msec**

# Information Content

- Kermany et al. (2010)

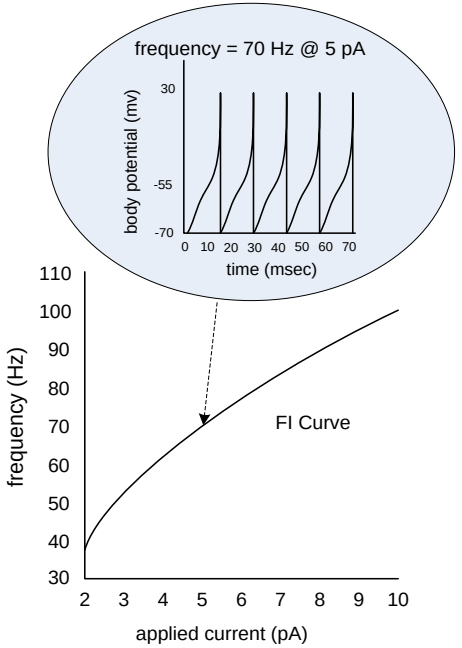
- In vitro culture of neurons
- Multi-grid array
- Five stimulation points
- Collect output spikes at least one neuron removed from stimulation point.
- Collect spike data for 100 msec
- Reduce two ways related to spike timing:
  - *time to first spike (TFS)*
  - *rank order*
- Reduce two ways for rates: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




- Time based reductions contain much more information (leading to higher accuracies)

# Rate Coding, Post Mortem

- Neurons never see constant input current during normal operation
  - See Mainen and Sejnowski
  - Yet, the rate-based hypothesis seems to rest on constant current behavior
- Temporal coding is fully consistent with the FI curve
  - The greater the stimulation, the sooner the first spike




*"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**

# Summary

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

| | Temporal Coding | Rate Coding |
|---|---|---|
| Fast enough? | *Yes* | *No* |
| Information content | *High* | *Low* |
| Energy consumption | *Low* | *High* |

- *Which did evolution choose?*

# Biological Background

## *Encoding Sensory Input*

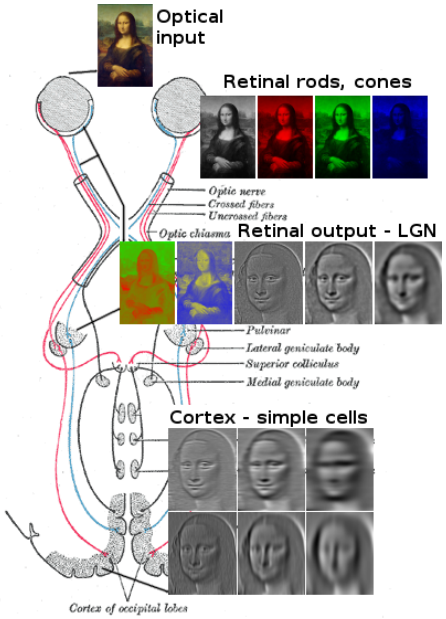
# Sensory Inputs

- In an architecture, raw input values are “Sensory Inputs”
  - Their behavior is analogous to biological senses
  - In an artificial environment, we can also invent our own “senses”

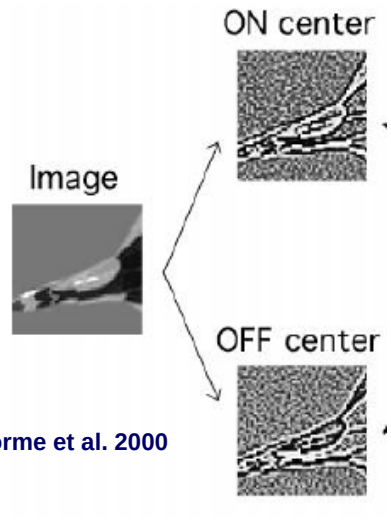
    E.g., Temperature sensors spread out over a grid
- Encoding translates sensory inputs to temporal code
  - Vision: fairly straightforward, often used by theoreticians
  - Auditory: uses frequency transform
  - Olfactory: must learn/operate quickly over a very large feature space
  - Somatosensory: extremely 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

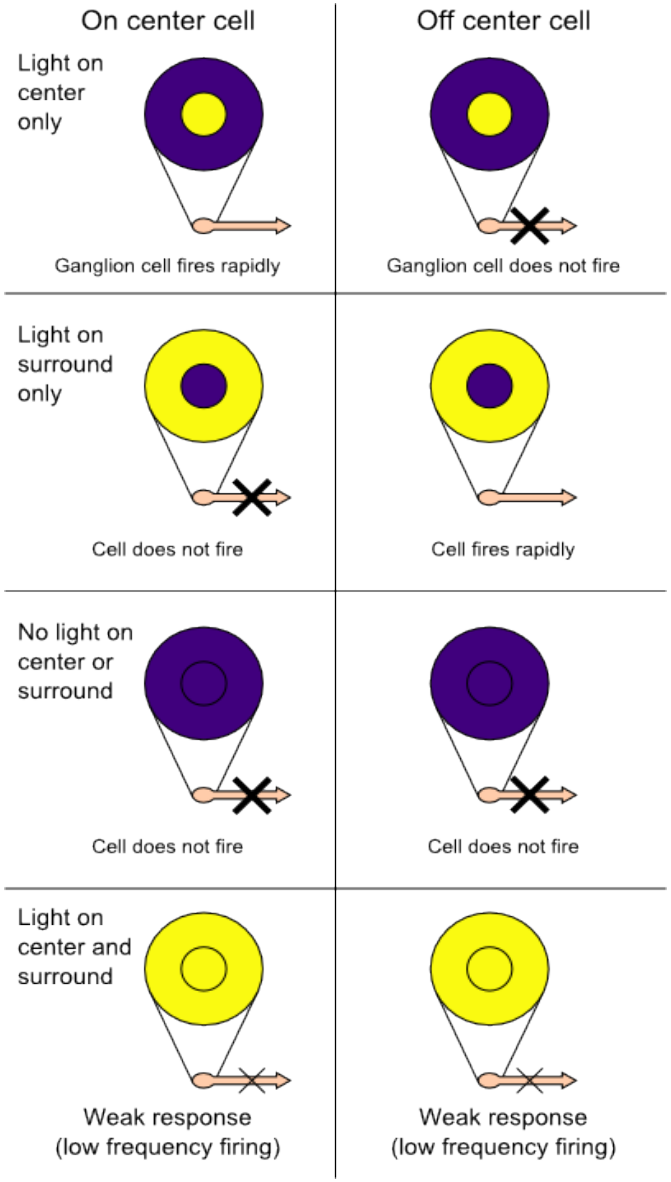
- OnOff retinal ganglion cells
  - Perform edge detection
  - Computer vision people use Gabor filters
- Encode as wave of spikes according to intensity of edges
  - Most intense yield earlier spikes




from Wikipedia



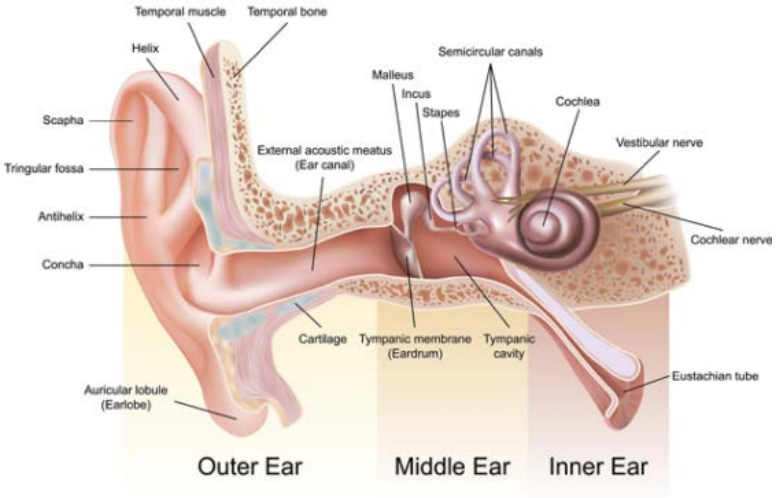

from DeLorme et al. 2000



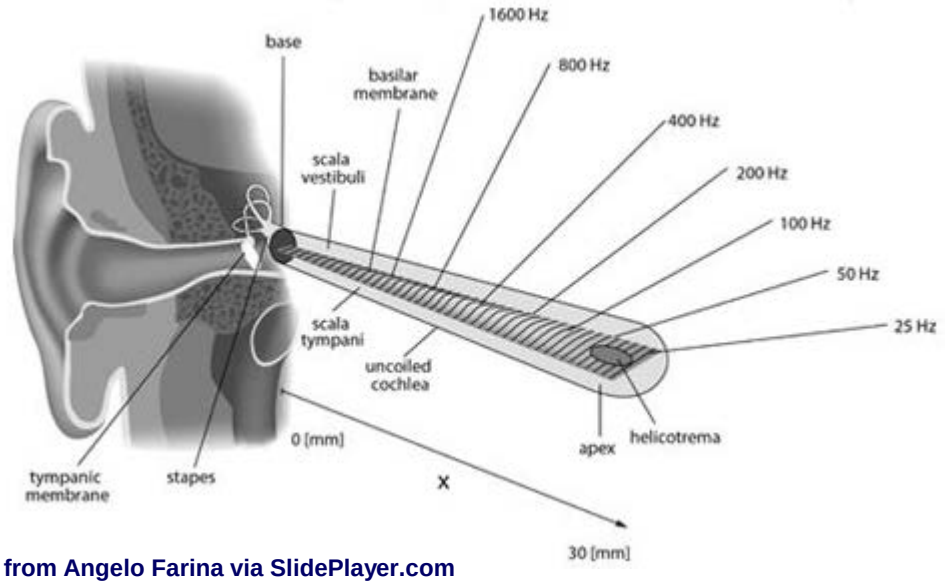

from Wikipedia

# Auditory Input

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

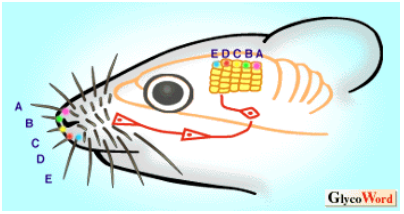



from Lankamedicalaudio.com



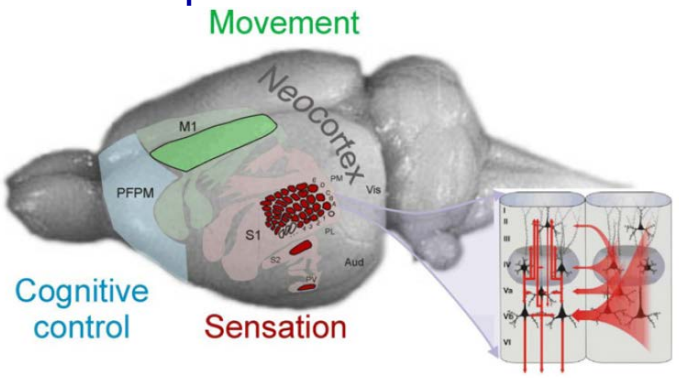

from Angelo Farina via SlidePlayer.com

# Somatosensory Input



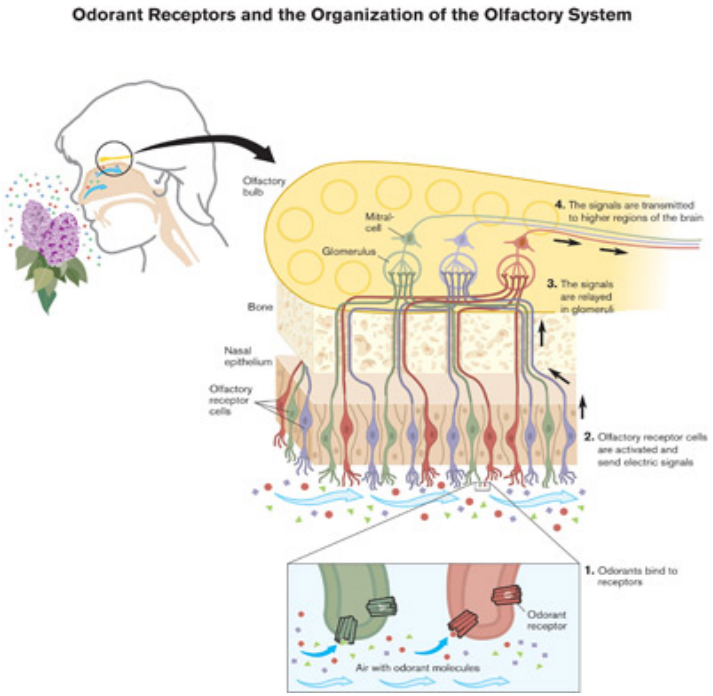

http://www.glycoforum.gr.jp/science/word/proteoglycan/PGA07E.html

- Wide variety of receptors
- Some 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



# Olfactory Input

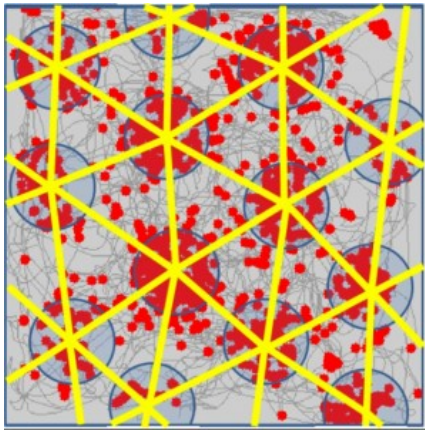
- Olfactory receptor neurons (ORNs)
  - Dendrites connect to tiny hairs containing proteins that bind to odor molecules
  - Stronger odor affinity causes earlier spike
- 10M ORNs in a human
  - Proteins coded with 1000 different genes
- Must learn/operate quickly over a very large feature space
  - To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




from health.howstuffworks.com

# Sense of Place

- The sense of where you are in your current environment – your “place”
- Does not have a physical world interface (unlike all the others)
- Stored as neuron 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

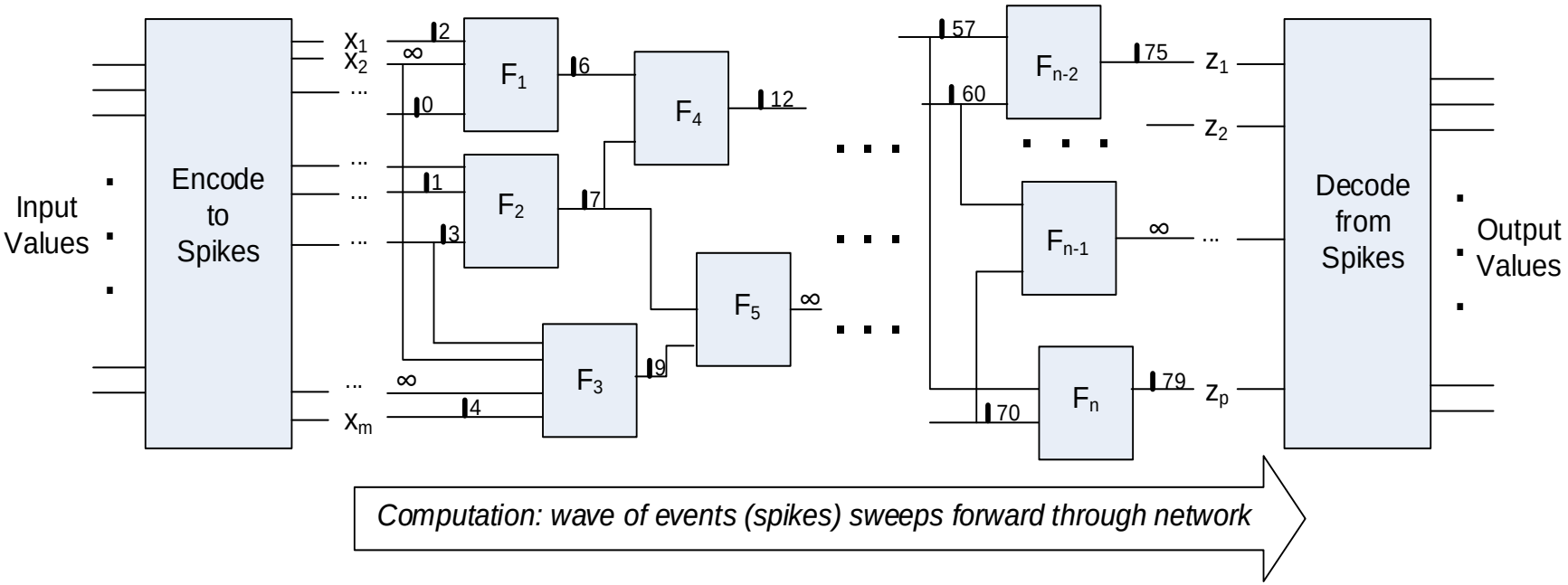


from placeness.com

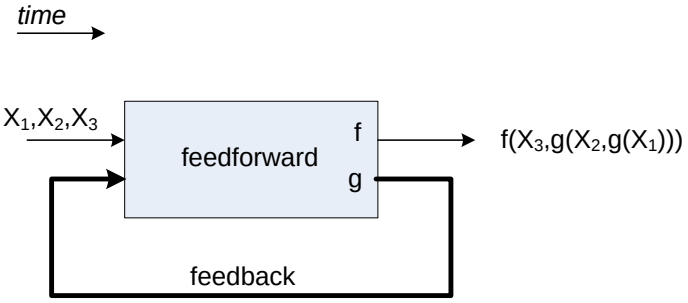
# Major Hypotheses

# TNN Architecture Features

- Feedforward
- Intrinsically fault-tolerant
- Temporally stable

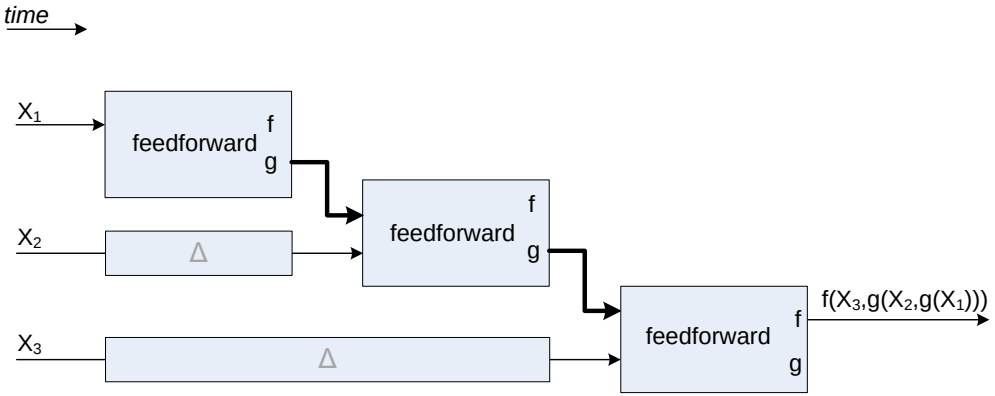


# Feedforward Hypothesis




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

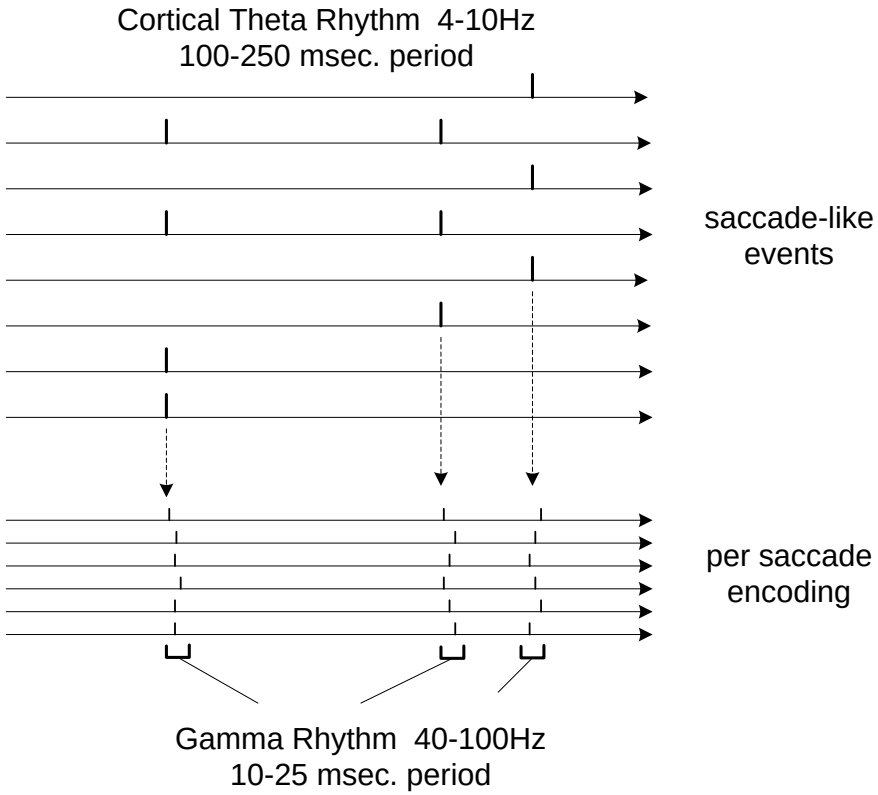
*If* a feedback computation satisfies the discrete-ness and finite-ness constraints of Turing's thesis
*Then* there is a feedforward functional equivalent




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

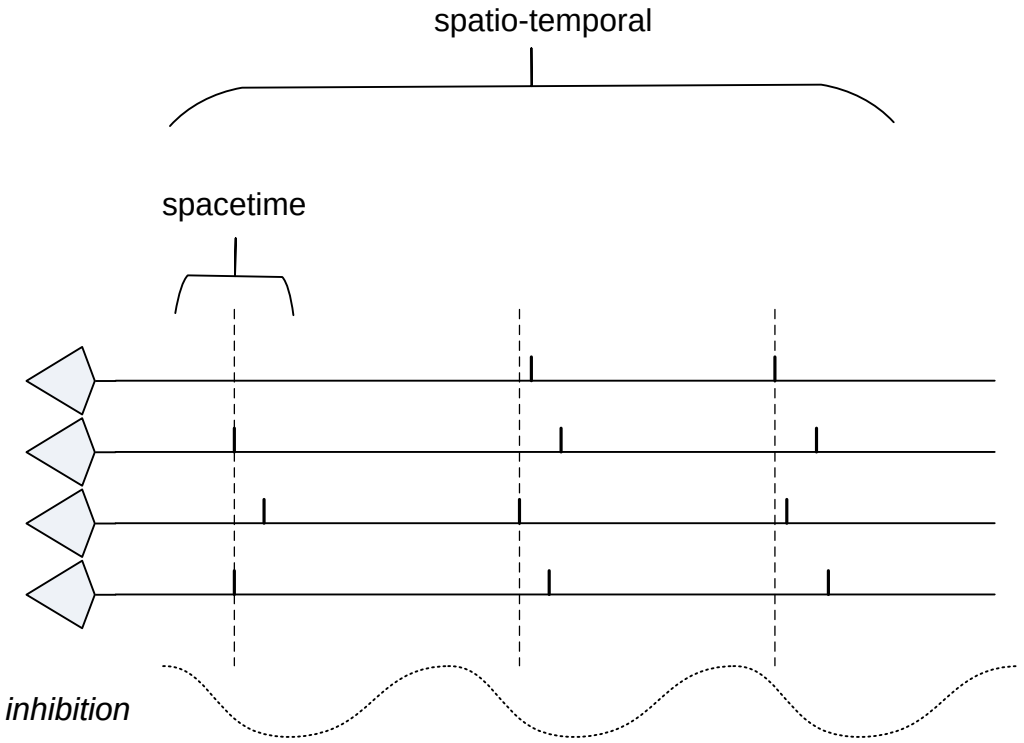
# Aside: Synchronizing Rhythms

- Theta rhythm
  - Roughly 4-10 Hz
  - Saccades at roughly same rate
  - Perhaps a “refresh” of sorts?
- Gamma rhythm
  - Roughly 40-100 Hz
  - 10-25 msec coding window
- Paths may have per neuron synchronization
  - Like a pipeline w/ a single gate per stage
  - Neuron feeding itself is easily handled



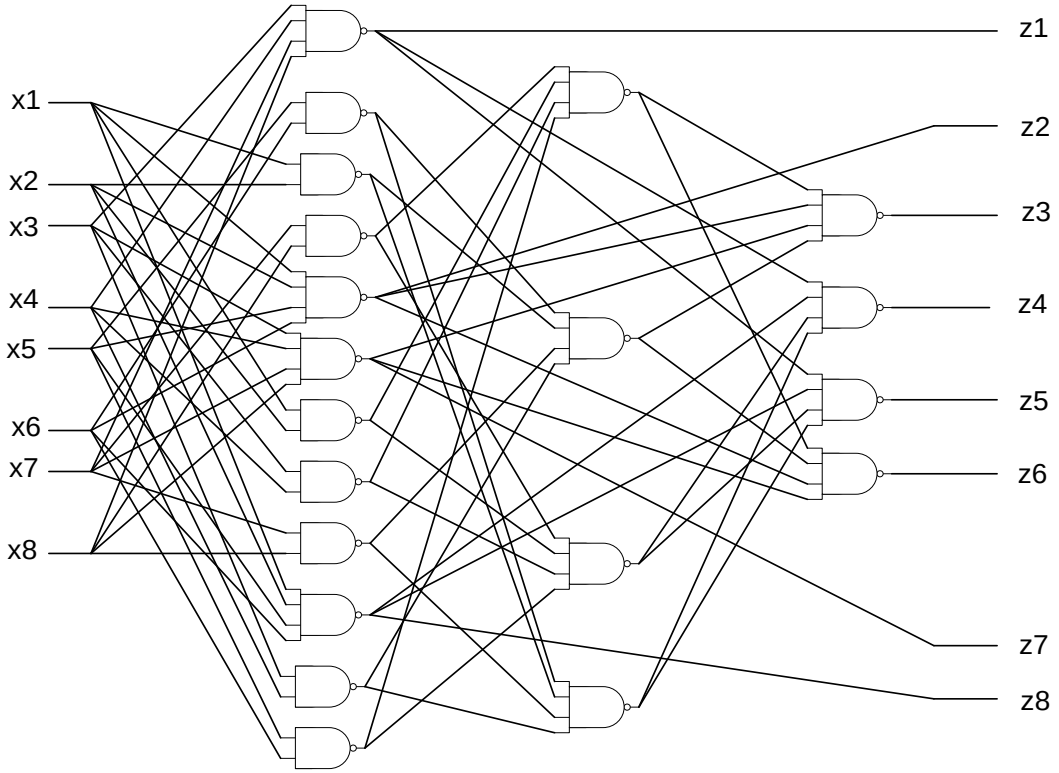
# “Space-time” v. “Spatio-temporal”

- “Space-time” rolls off the tongue much more easily
- We are actually modeling characteristics of physical spacetime
- Reserve “spatio-temporal” for a different level of abstraction



# Fault Tolerance

- What does this logic network do?




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

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

# Temporal Stability

- Required, in the absence of clocked storage elements
- Based on temporal uniformity
  - Uniformity among compound paths
  - Over at least an entire excitatory column

    More likely at least a macro-column
- 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
- *It is assumed that the feedforward paradigm can be studied independently of mechanisms that provide temporal uniformity*

# 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*

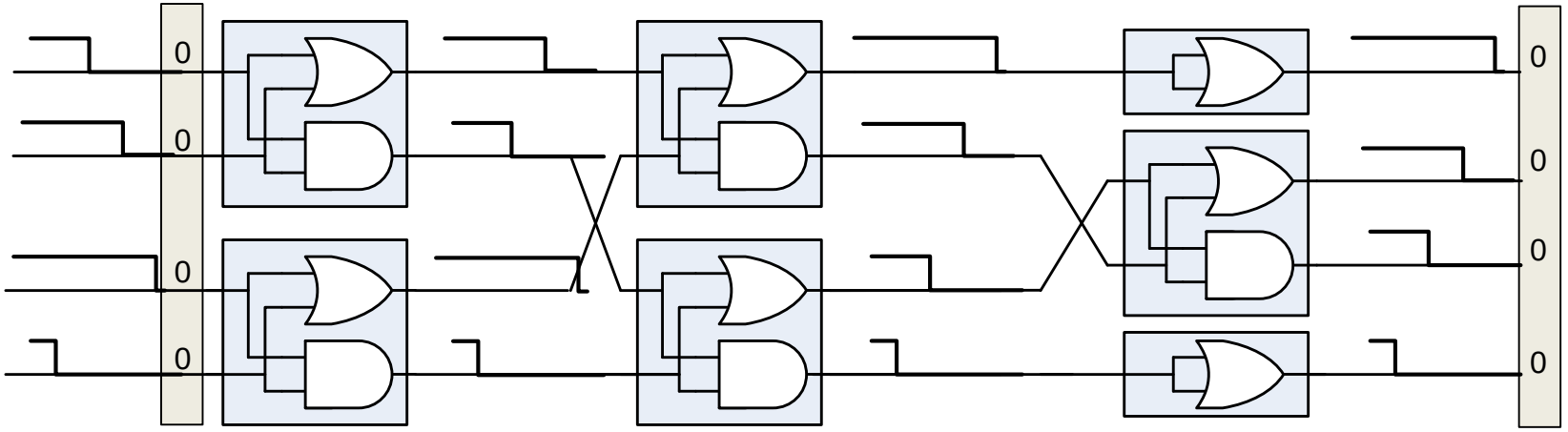
# Summary of Hypotheses

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

# Theory: Time as a Resource

# Step 1: Synchronous Binary Logic

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

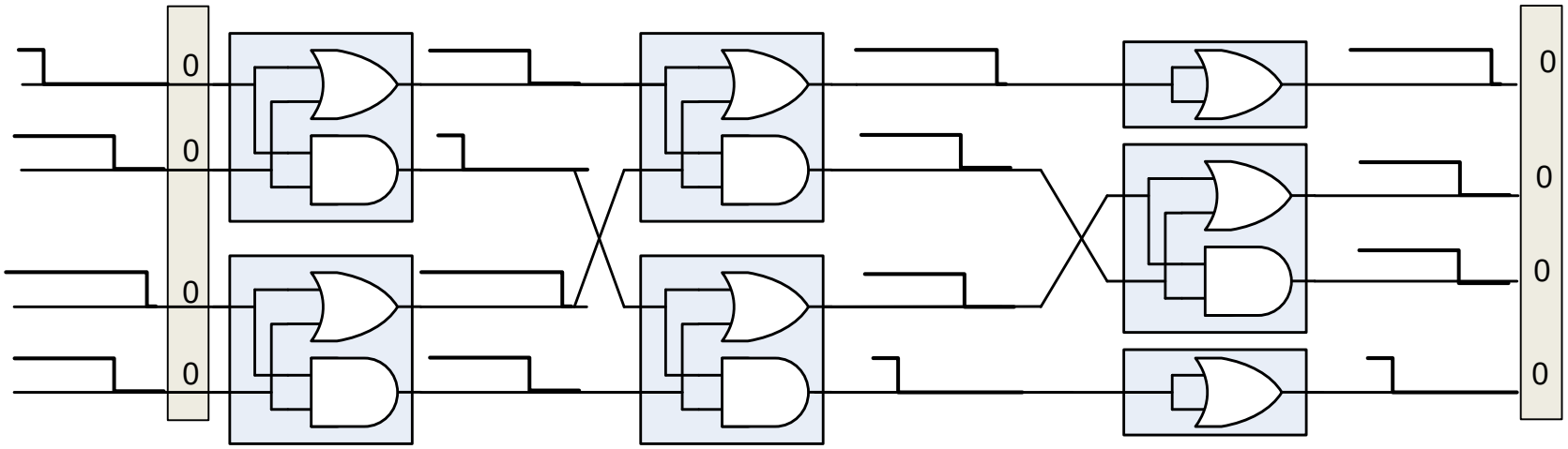



1) Apply input signals
2) Input signals stabilize
3) Wait for worst-case delay (3+ gate delays in this instance)
4) Output signals stabilize at result: <0,0,0,0> → <0,0,0,0>

# Synchronous Binary Logic, contd.

- Again, with different input timing

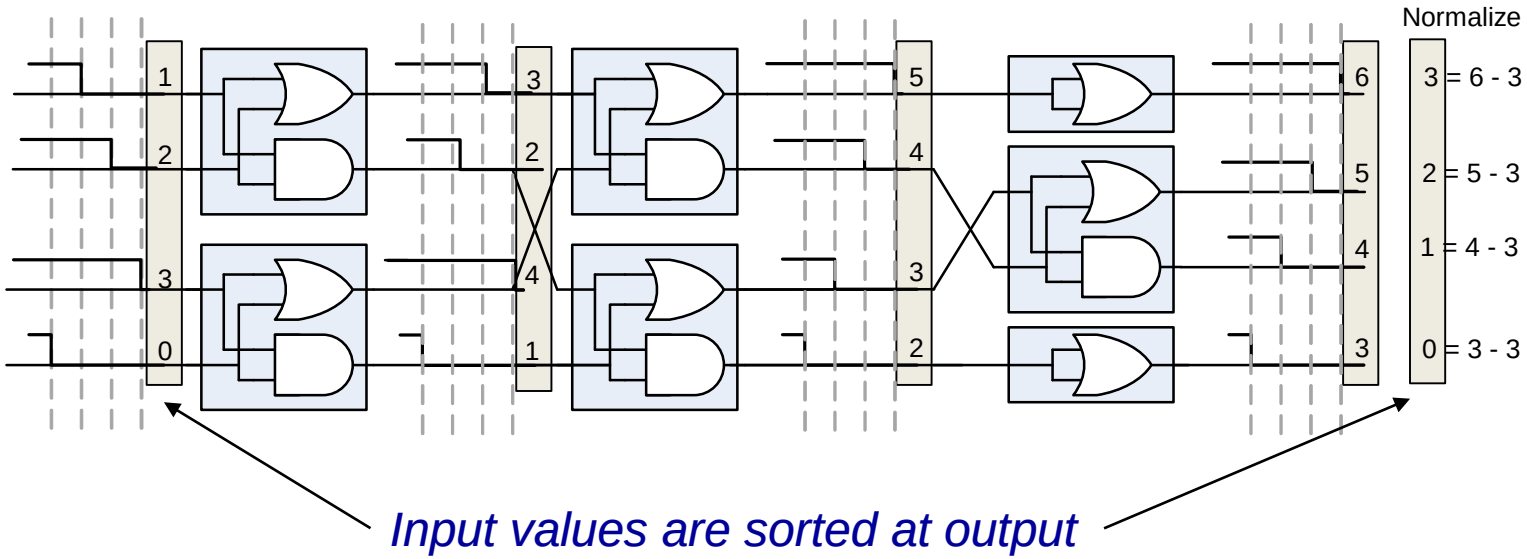
  *Regardless of exact input timing, we always get the same, "correct" answer*




- 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

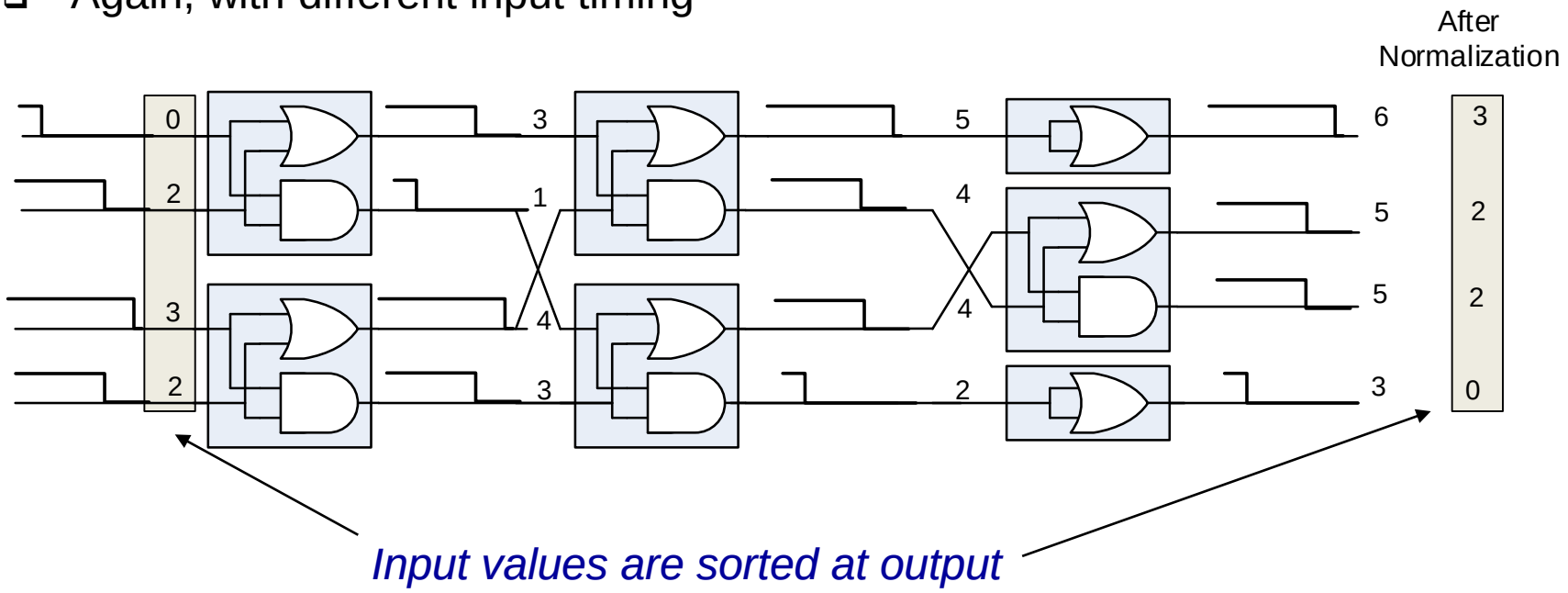
- Sherwood and Madhavan (2015)
- Process information encoded as relative edge times




- Time is divided into discrete intervals
- 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




*Input values are sorted at output*

- Same circuit, same signals, same everything as with synchronous method,
  *except* network implements a much more powerful function
  integer sort vs. <0,0,0,0> → <0,0,0,0>

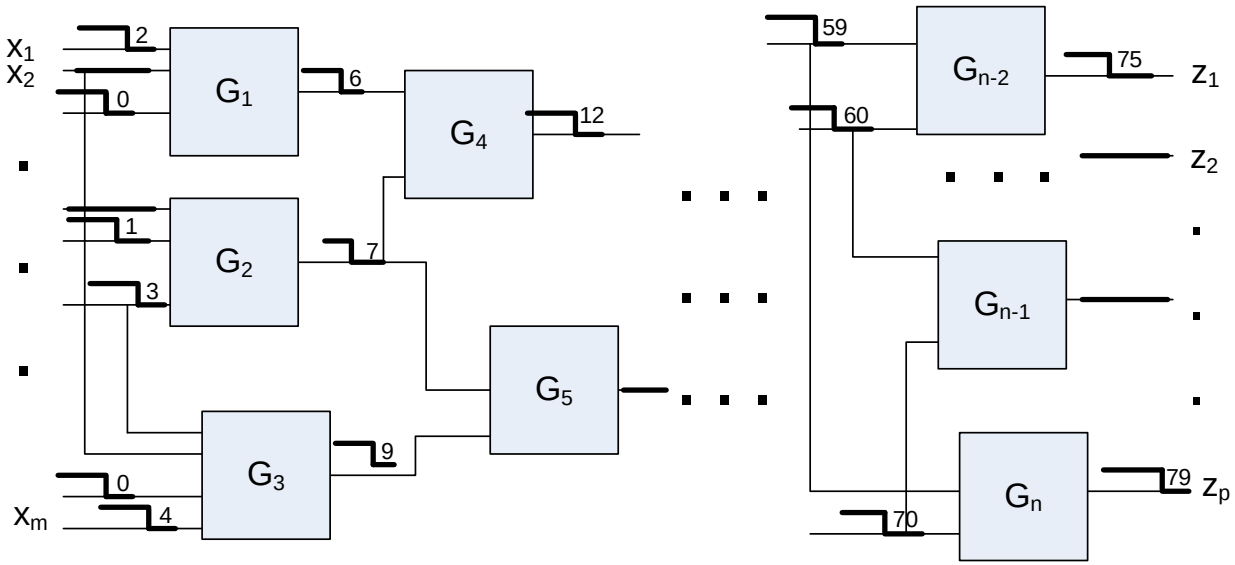
*same circuit sorts numbers of any size*

# The Temporal Resource

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

- The *flow of time* has huge 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 avoid the effects of time when we engineer computer systems
  - Assume worst case delays (w/ synchronizing clocks) & delay-independent asynchronous circuits
  - *May be best choice for conventional computing problems and technology*
- 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
- Aside: *emergent behavior*
  - Each block operates on its own, using only local information, no physical global synchronization (other than the uniform flow of time)
  - Yet globally produced results *emerge* at the output

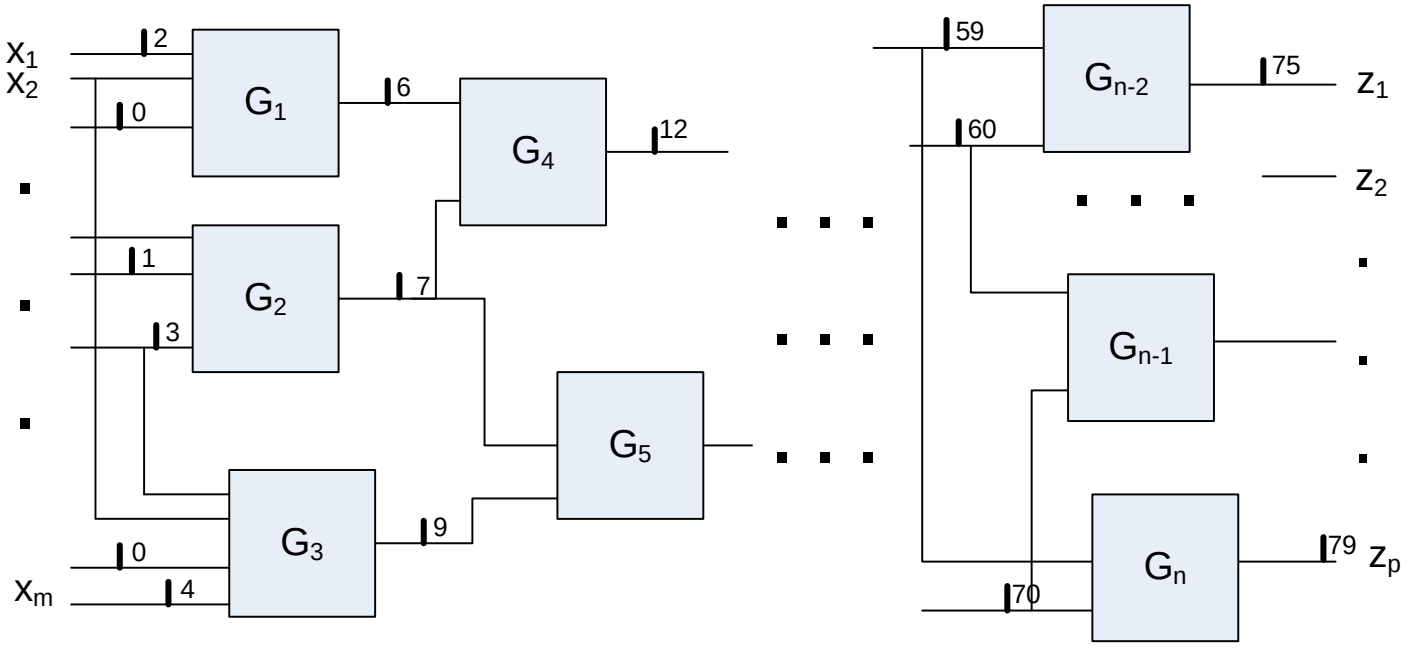
# Step 4: Generalize Transient Events

Nothing special about 1 → 0 edges…

Could use 0 → 1 edges

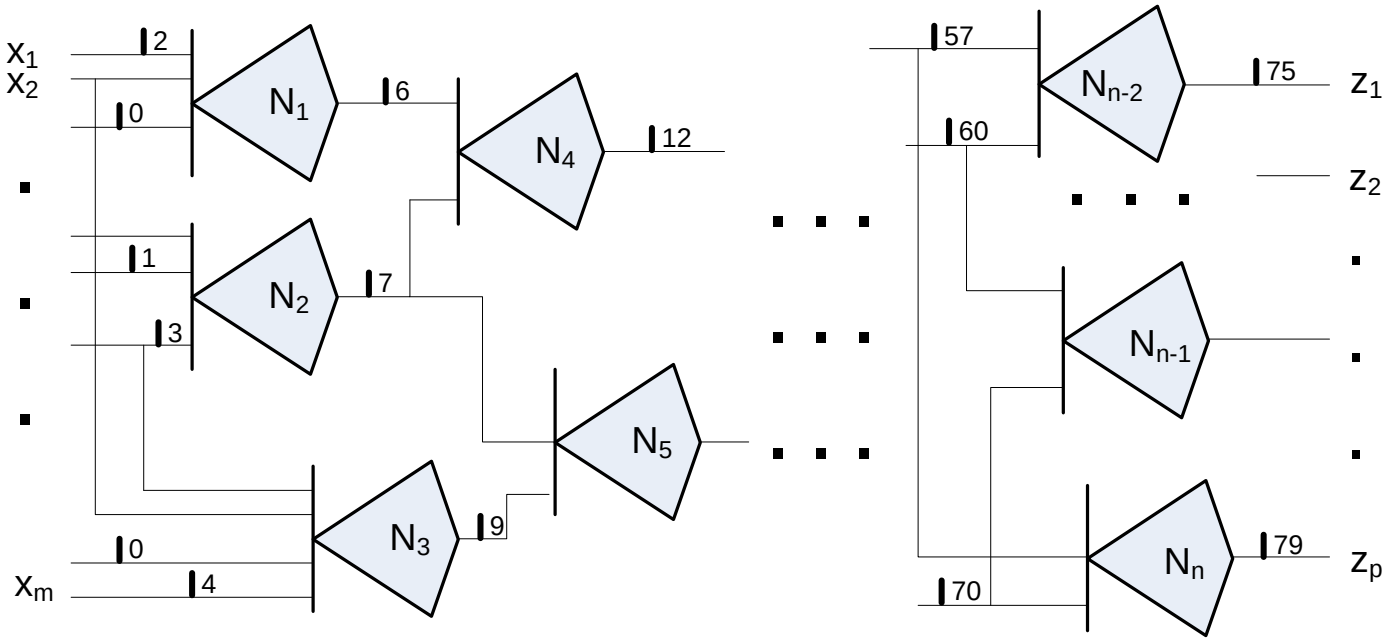
OR

pulses (spikes):



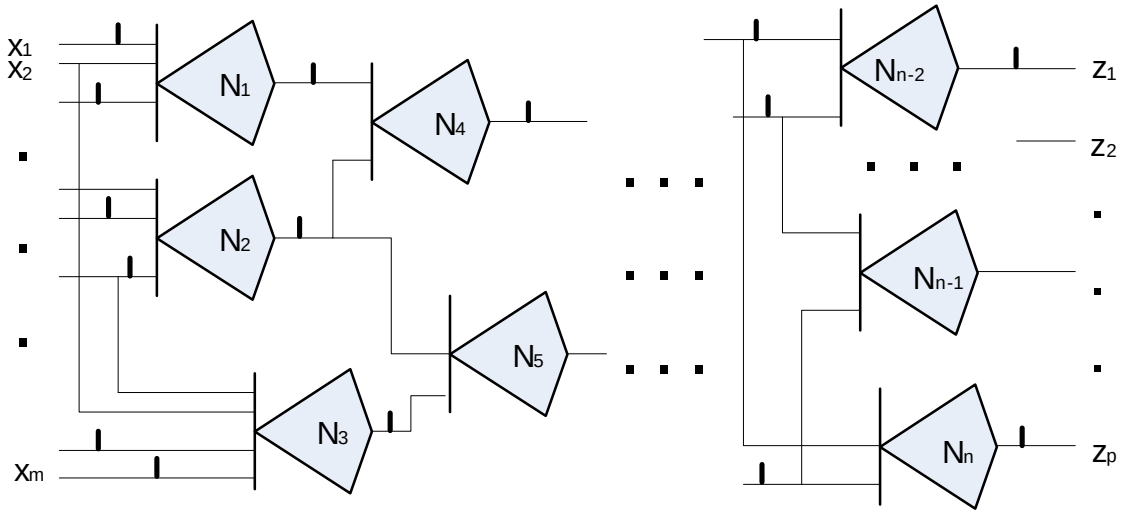
# Step 5: Specialize Functional Blocks

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

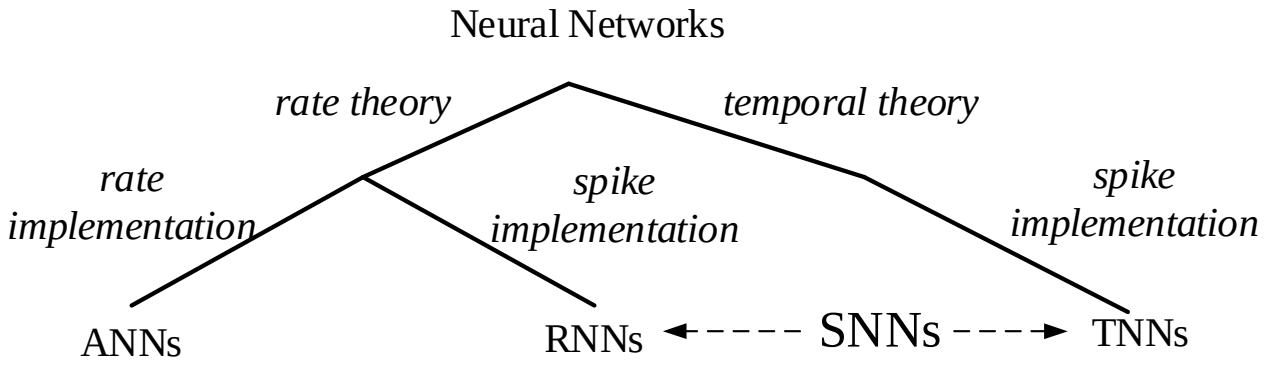


# Temporal Neural Networks (TNNs)

- 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



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

# Example: Conflating SNNs

- Yann LeCun's negative comments on IBM TrueNorth and SNNs

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

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

It will take significant research and development to achieve high accuracies.

BUT accuracy alone is not the goal – compare unsupervised clustering w/ supervised classification

Clearly assumes the RNN (rate) model and correctly points out a major flaw.
HOWEVER, this comment does not apply to TNNs

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

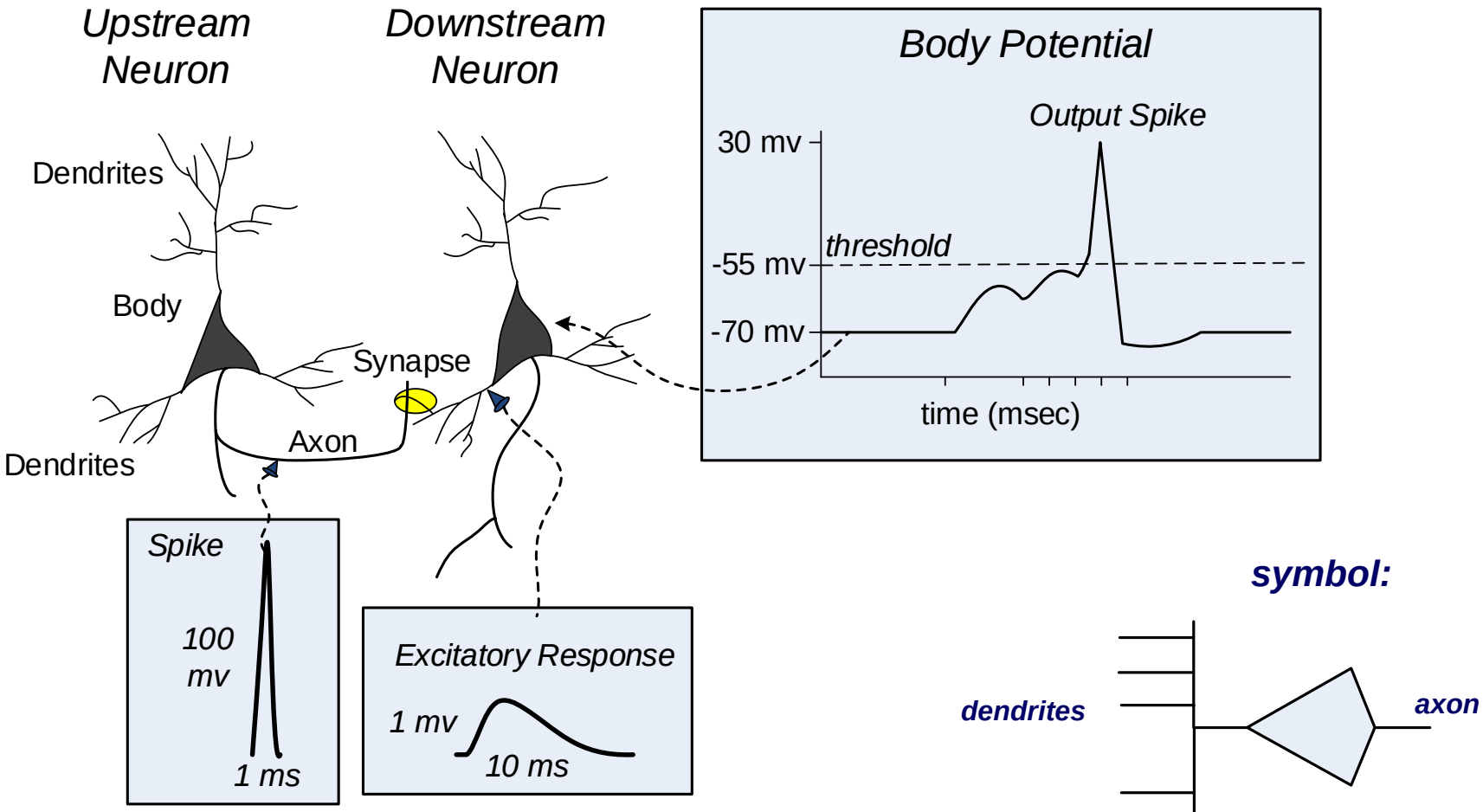
**Yann LeCun**

August 7, 2014

My comments on the IBM TrueNorth neural net chip.
IBM has an article in Science about their TrueNorth neural net chip. The chip has 4096 processors, each of which has 256 integrate-and-fire spiking neurons (with binary output) each with 256 inputs. Each processor can compute all 256 neurons 1000 times per second. Neuron states are binary (spikes) and synaptic weights are 9-bit signed integers with a 4-bit time delay. The overall peak performance is 4096x256x256x1000 = 266 GSops (billion synaptic operations per second). The power consumption is supposed to be around 100mW.
…
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.
…

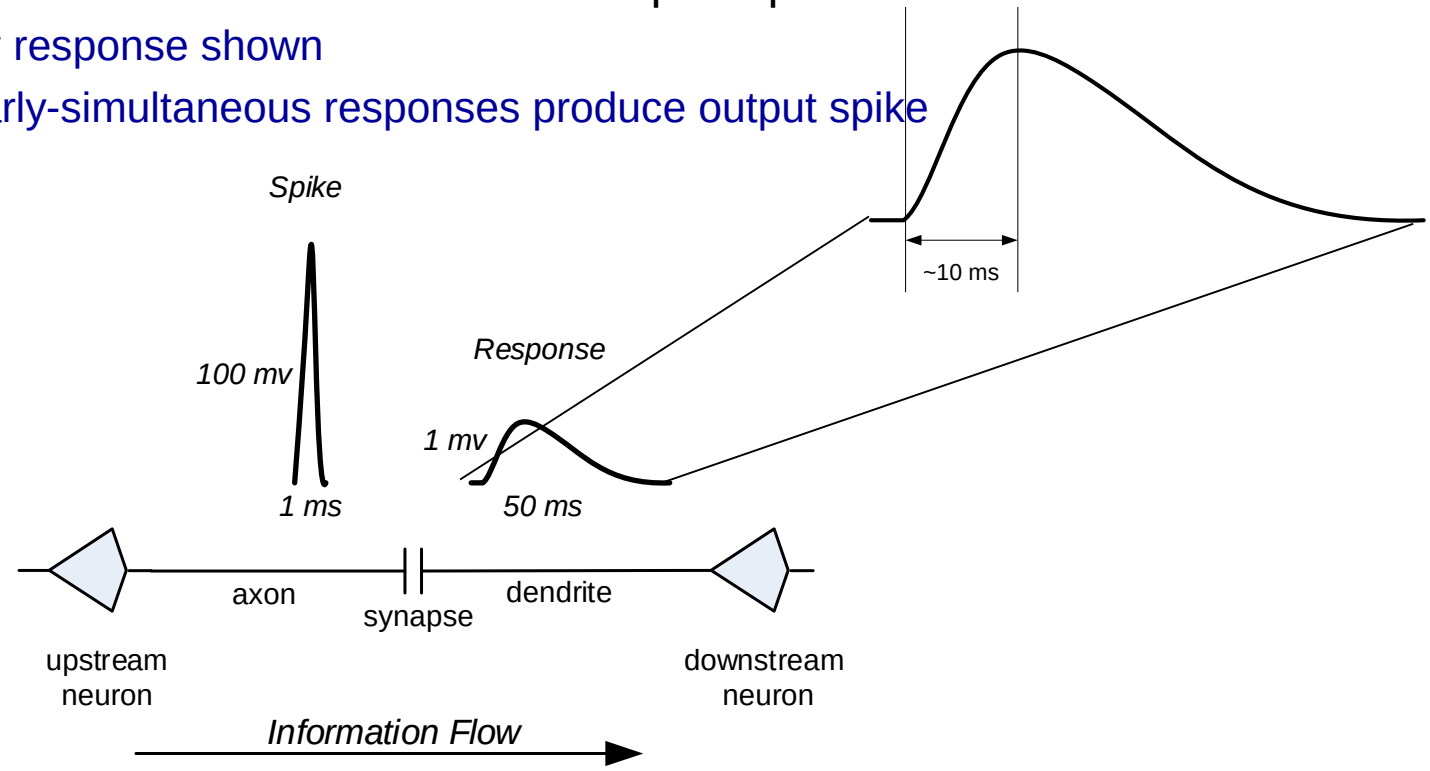
# Excitatory Neuron Models

# Neuron Operation (Re-Visited)



# Spikes and Responses

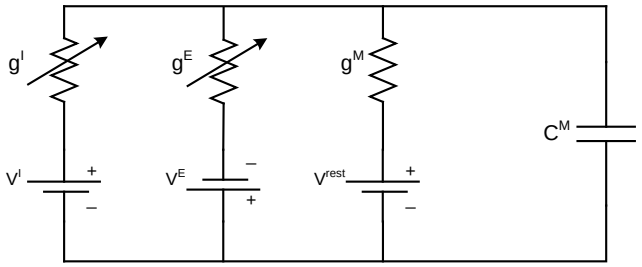
- *Spike* followed by *Response*
  - Both sides of electrical-chemical-electrical translation at synaptic gap
- Spike is very narrow
  - Result of ions being actively moved along the axon
- Response is much wider with much lower peak potential
  - Excitatory response shown
  - Many nearly-simultaneous responses produce output spike



# Neuron Models

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

$$C_m \frac{dV}{dt} = -g_L(V - V_L) - \bar{g}_{Na} m^3 h (V - V_{Na}) - \bar{g}_K n^4 (V - V_K)$$
$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V) m$$
$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V) h$$
$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V) n$$




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

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

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

# LIF Discretized Solution

- Targeted at efficient digital implementation

  *shorthand:*
  $r_M = dt/\tau_M \quad g^M = dt/C_M$
  $k_M = 1 - dt/\tau_M$
  $k_E = 1 - dt/\tau_E$
  $k_I = 1 - dt/\tau_I$

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

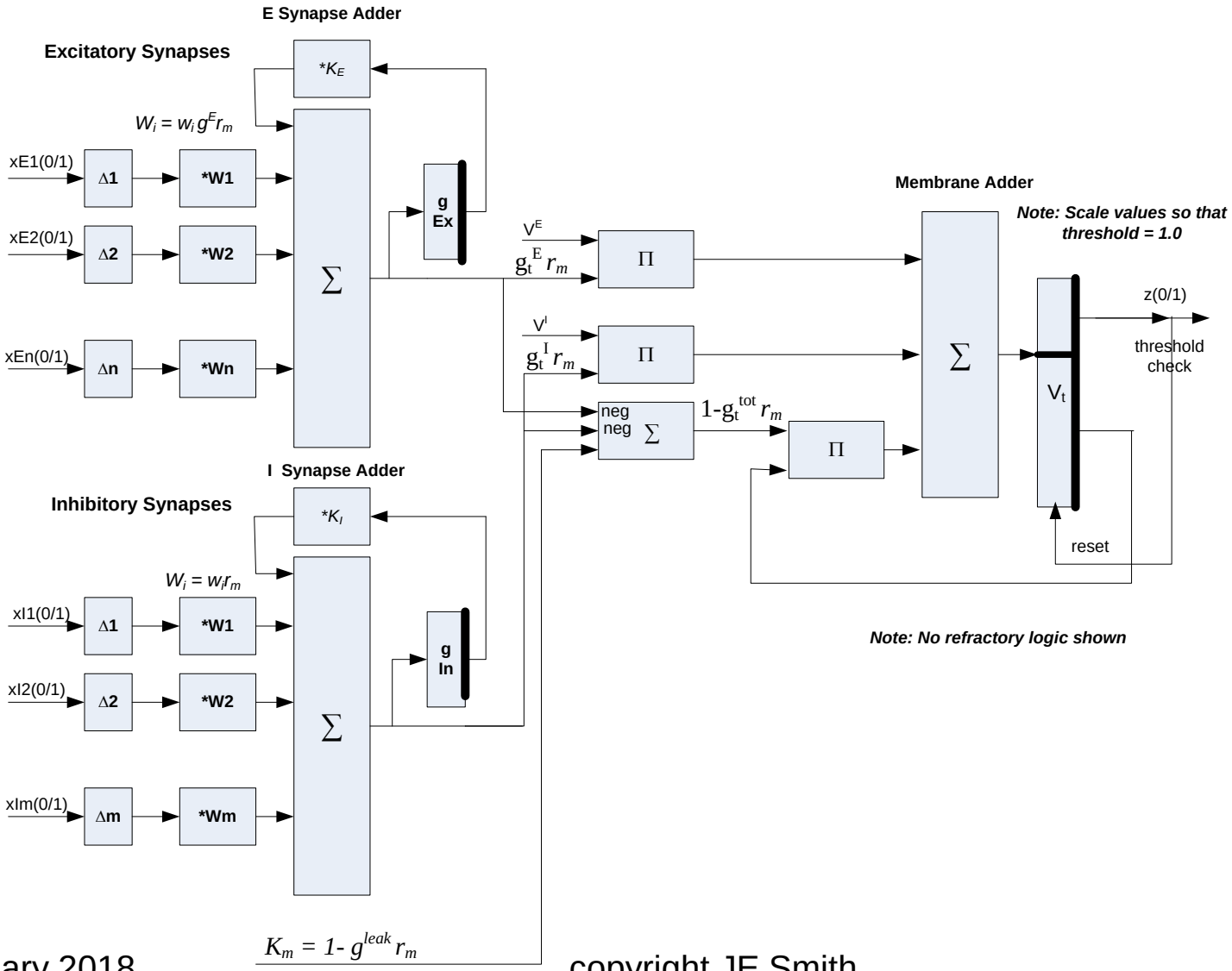
  $$r_m\, g_t^E = g_{t-1}^E\ (k_E) + \sum g^E\, r_m\, w_{ij}$$
  $$r_m\, g_t^I = g_{t-1}^I\ (k_I) + \sum g^I\, r_m\, w_{ij}$$

  *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 $V_{rest} = 0$
  - Scale voltages so that $V_{th} = 1$

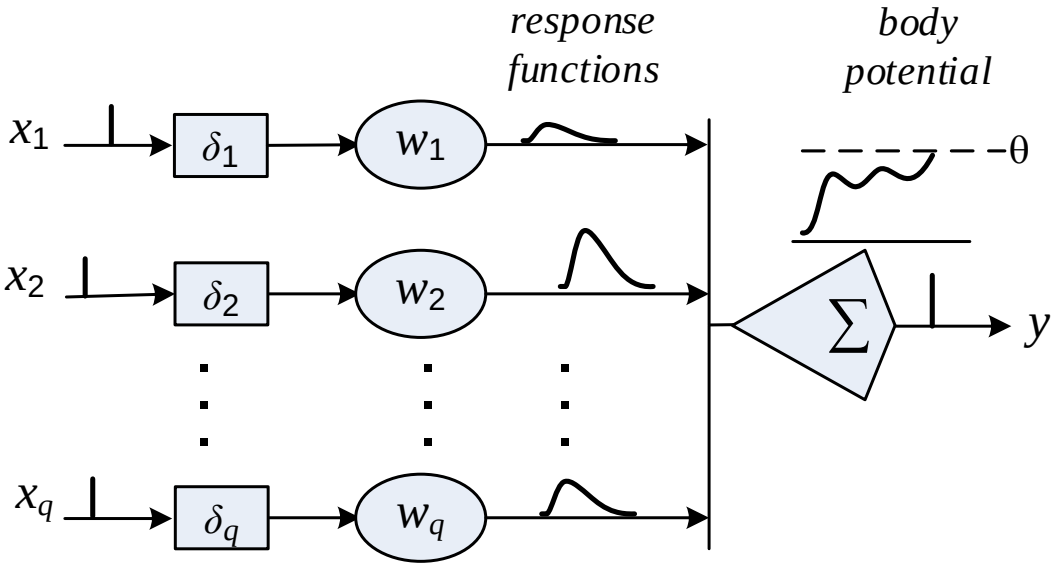
    Simplifies threshold comparison (check only MSB)
- Assume connections via loss-less cables with distance-related delay

# LIF Digital Implementation

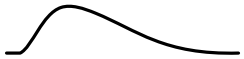


# Excitatory Neuron Model: SRM0

- Basic Spike Response Model (SRM0) -- Kistler, Gerstner, & van Hemmen
- Spike at input may be delayed ($\delta_i$)
- At the input's synapse, a response function is produced
  - Synaptic weight ($w_i$) determines amplitude: larger weight => 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 from strong synapse**



**response from weaker synapse**

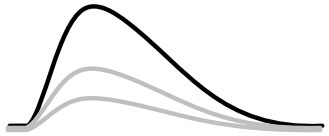
# Response Functions

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

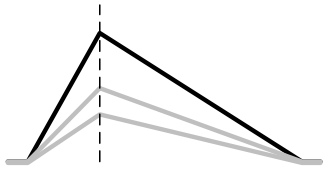
*Biexponential* – most realistic

*Non-leaky* often used

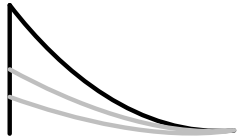
*Compound synapse* has skewed peak amplitude



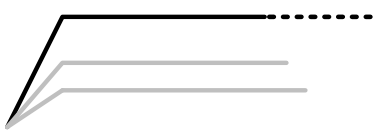
Biexponential



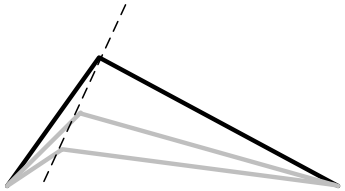
Piecewise linear biexponential approximation



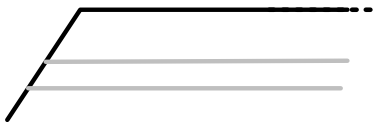
Original LIF
(Stein 1965)



Non-leaky

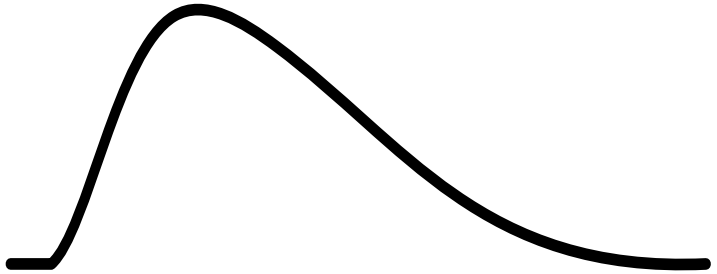


Compound synapse

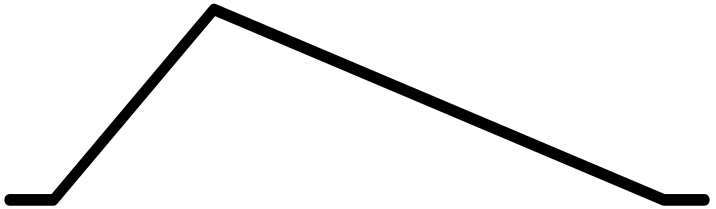


Compound synapse
Non-leaky

# Which Is the Approximation?

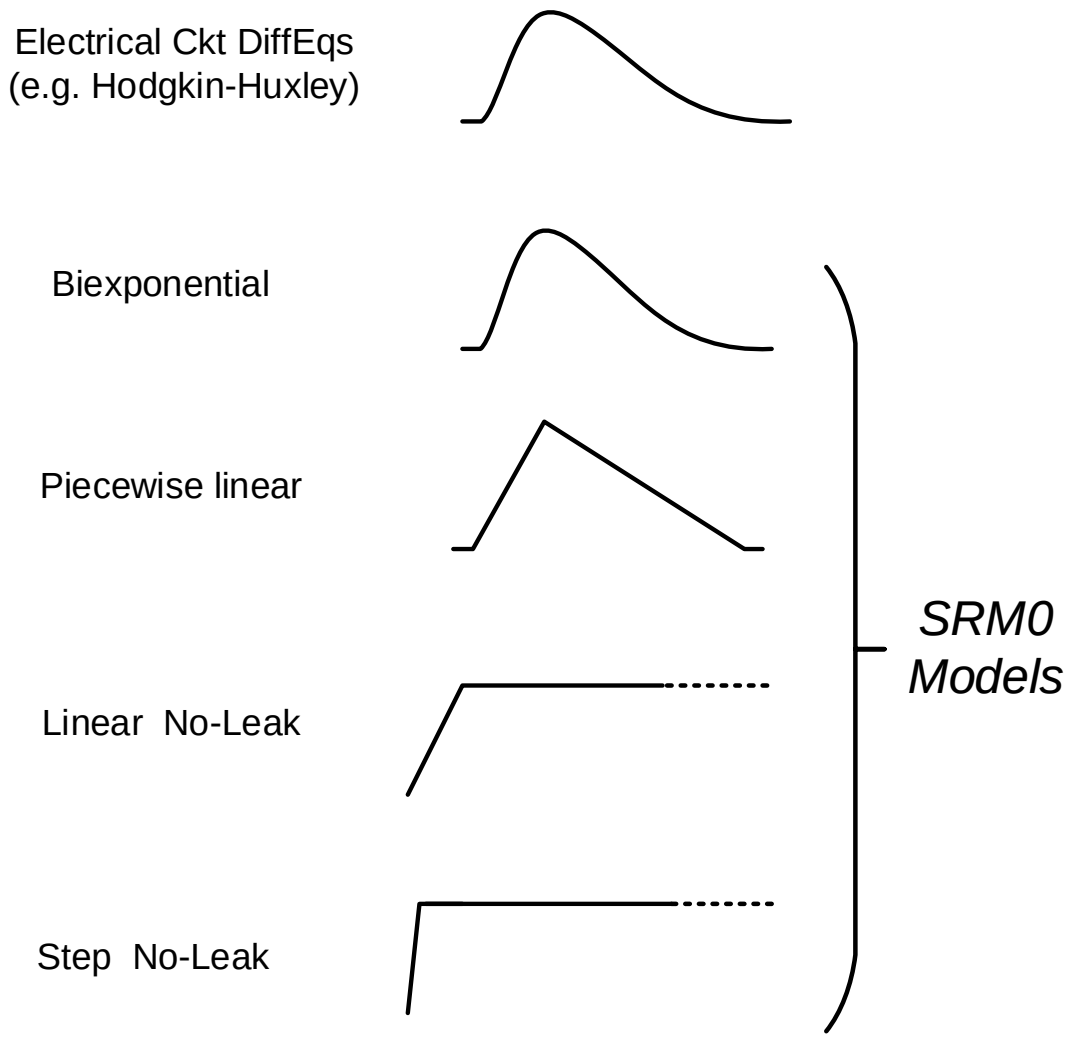


a) Biexponential spike response



b) Piecewise linear spike response

# A Hierarchy of Spiking Neuron Models



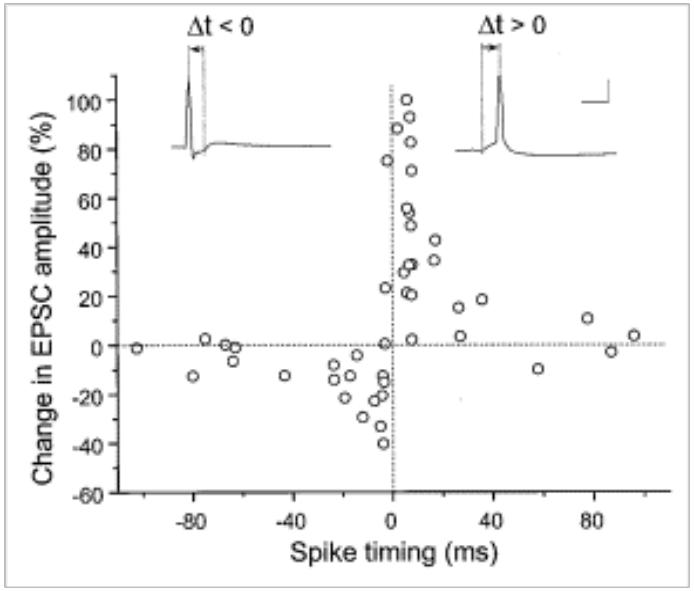
# Nomenclature: LIF Neuron Models

- 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

# 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 basis

- *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 input and output spikes) – $\Delta t$ may be pos. or neg.




from Bi and Poo 1998

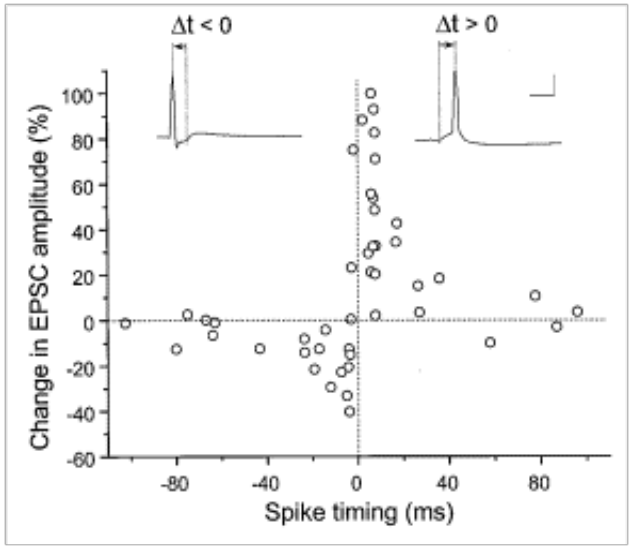
- Also see Markram et al. (1997) and Morrison et al. (2008)

# STDP

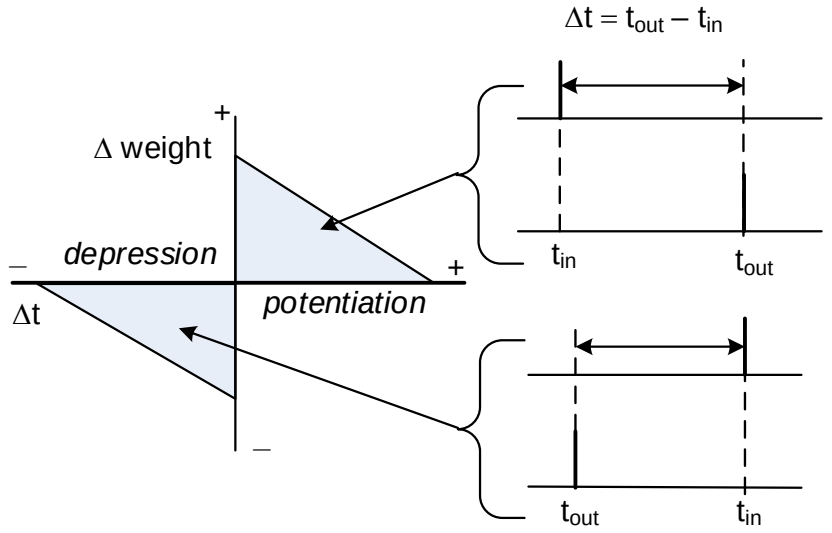
- *Spike Timing Dependent Plasticity* :
  - Compare relative input and output spike times

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

    *Input spike time ≥ 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



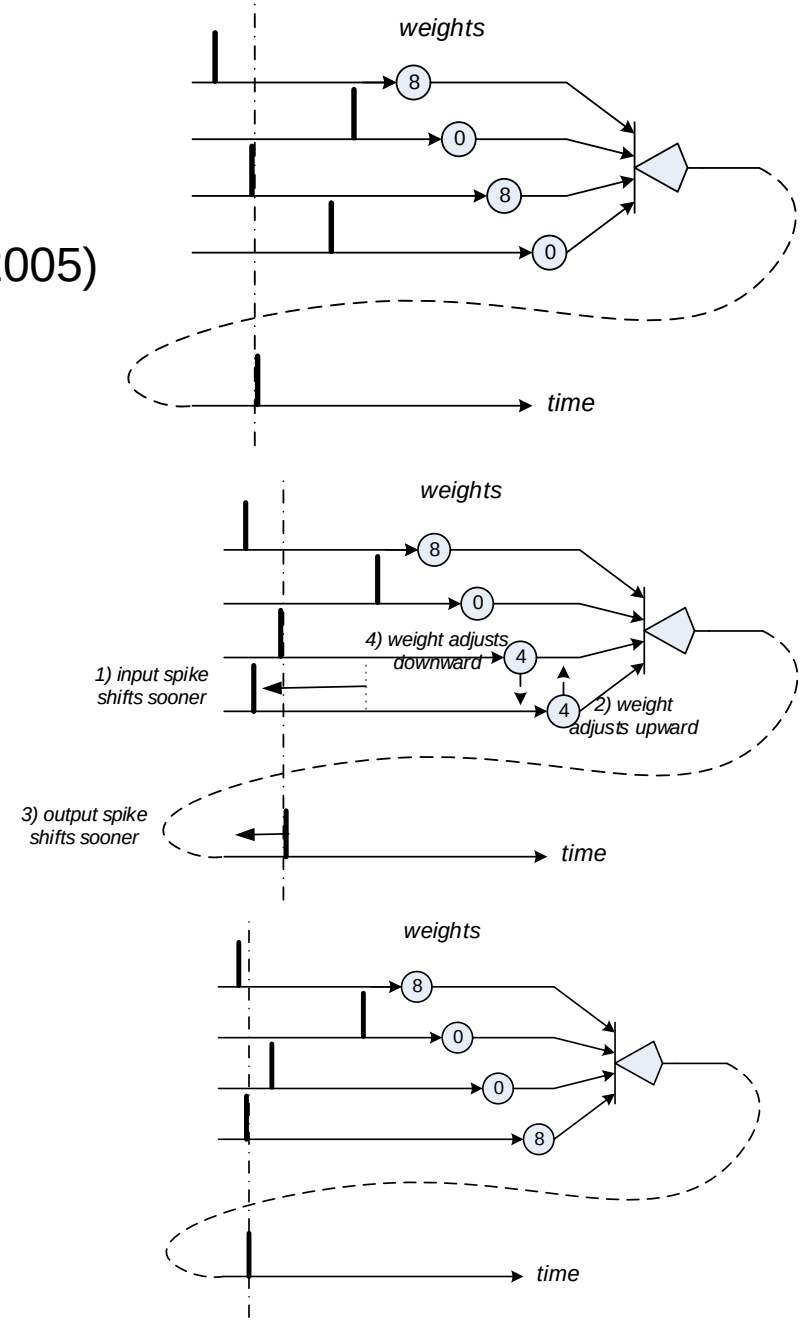

- Training is *unsupervised*, *localized*, and *emergent*
  - This is a major difference wrt conventional error back-propagation

# STDP In Action

from Guyonneau, VanRullen, and Thorpe (2005)

*Paraphrased*

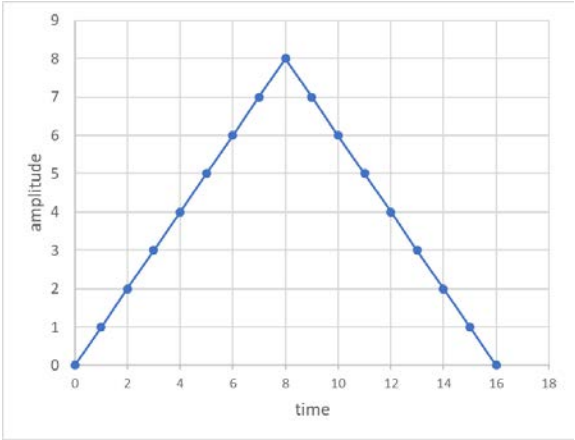
- For a given input pattern, input spikes elicit an output spike, triggering the STDP rule.
- Synapses carrying input spikes just preceding the output spike are potentiated, while later ones are weakened.
- The next time the input pattern is applied, the firing threshold will be reached sooner, which 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.



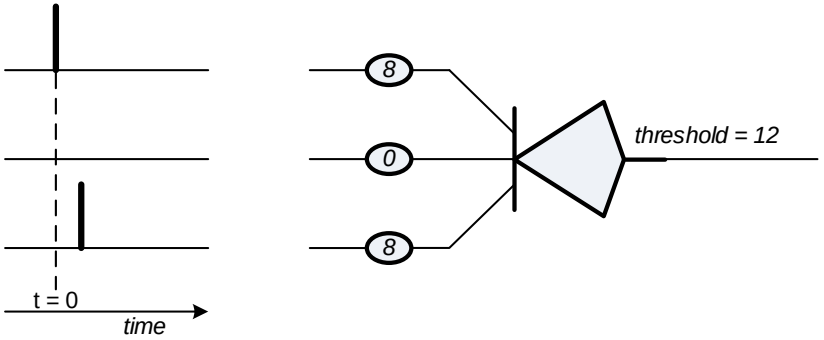
# Example

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

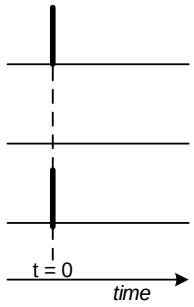
***Response function for Weight = 8***

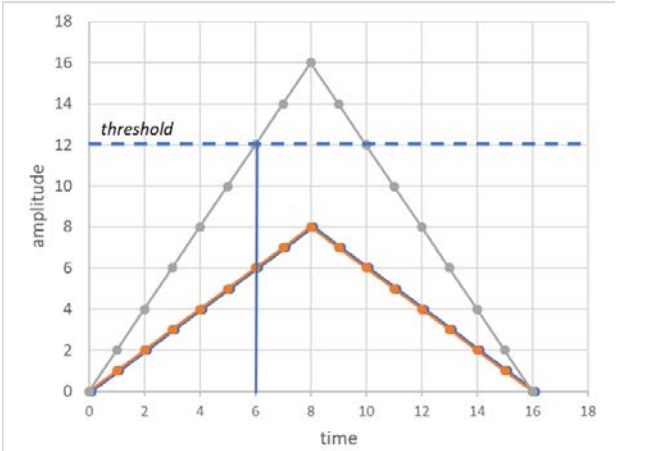



*typical training pattern*

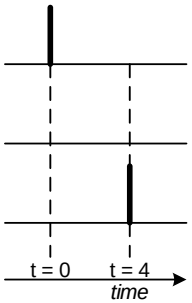


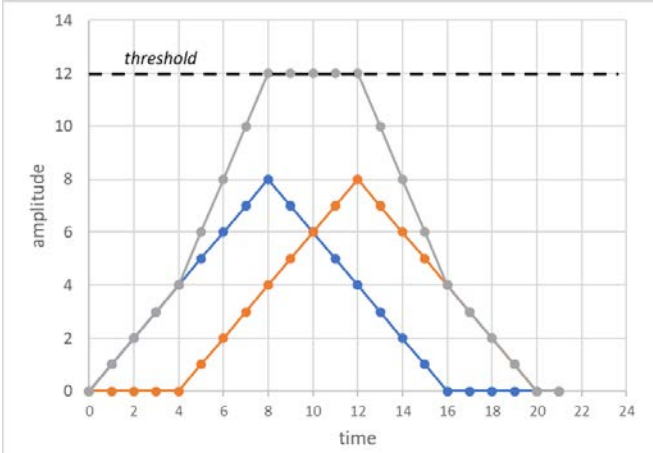
**two spikes match**
**ideal alignment**
***output spike time = 6***



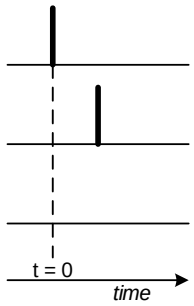



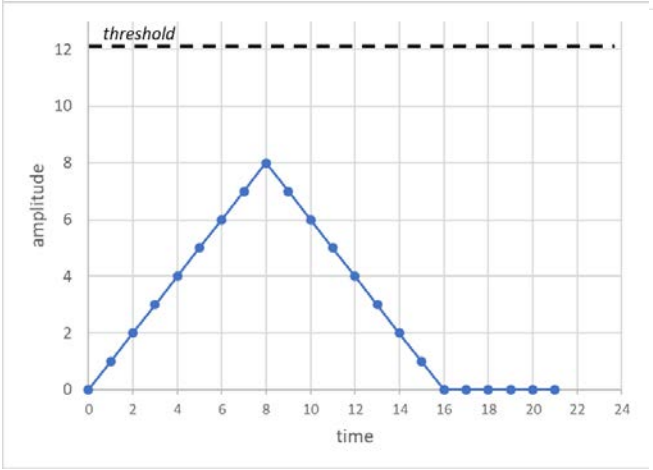

**two spikes match**
**some miss-alignment**
***output spike time = 8***



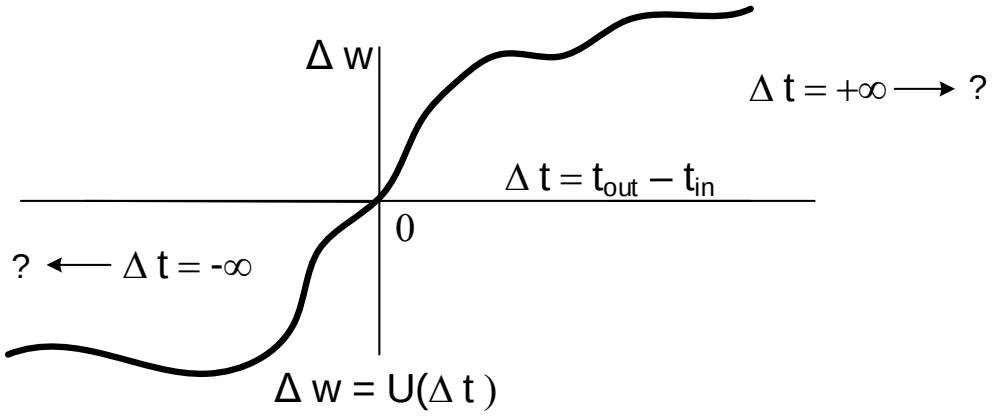

**only single spike matches**
***no output spike***



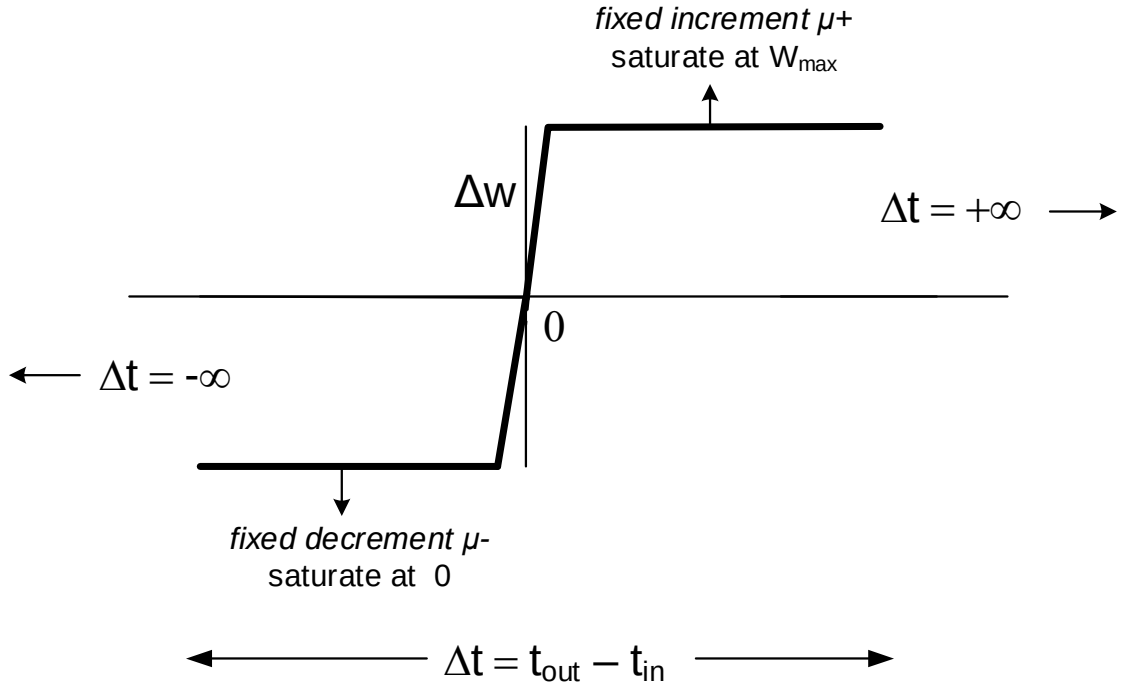
# Excitatory Update Functions

- At each synapse:
  - In response to difference between input and output spike times: $\Delta t = t_{out} - t_{in}$
  - 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



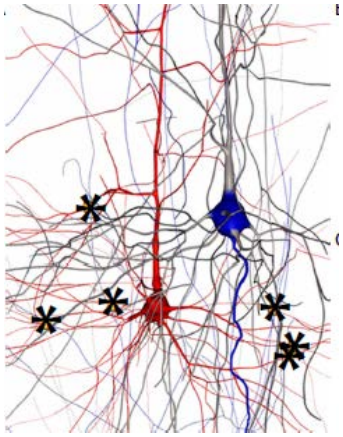
# Model I Currently Use

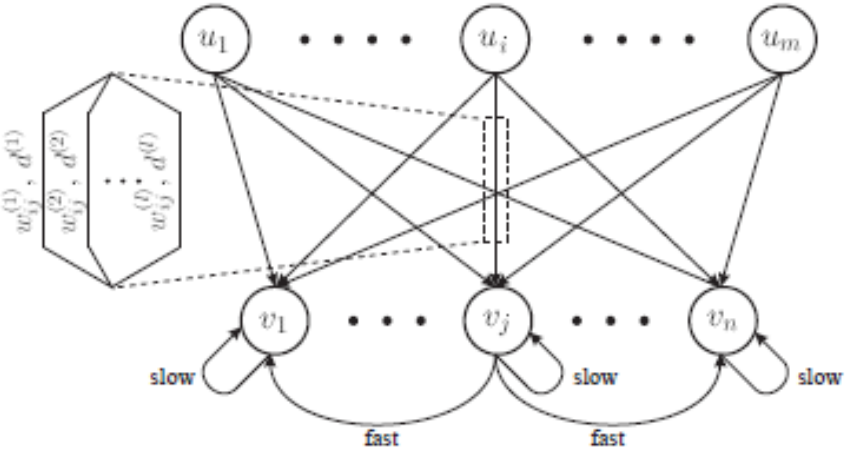
- Model parameters: Increment and decrement constants (μ+ and μ-)
  - U is independent of current weight
  - Saturate at 0 and $W_{max}$
  - |μ-| > |μ+|  (typically)
- Will account for one-spike cases later
  - Very important cases to be discussed at a higher design level

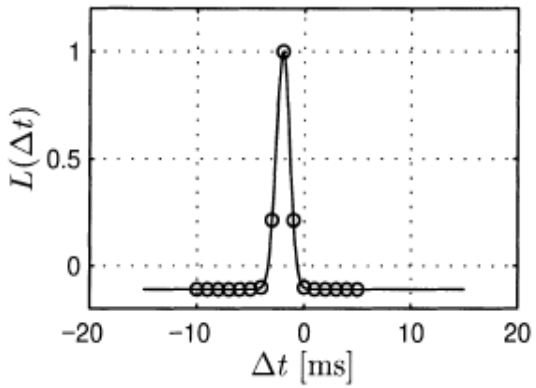


# RBF Neurons

- 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 like classical Radial Basis Functions (RBF)
- Natschläger and Ruf (1998) based on Hopfield (1995)
  - Also see Bohte et al. (2002)

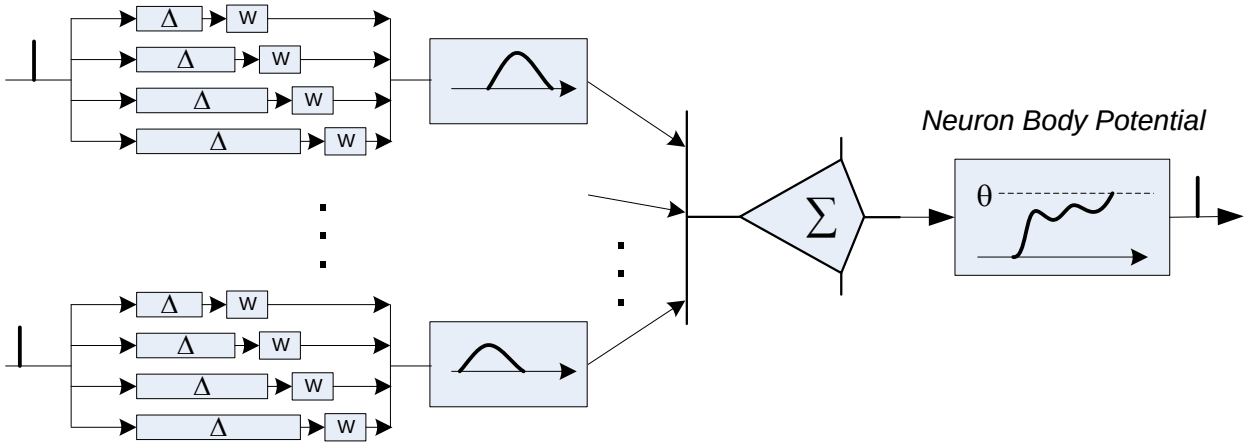





**STDP function**

# Compound Synapses

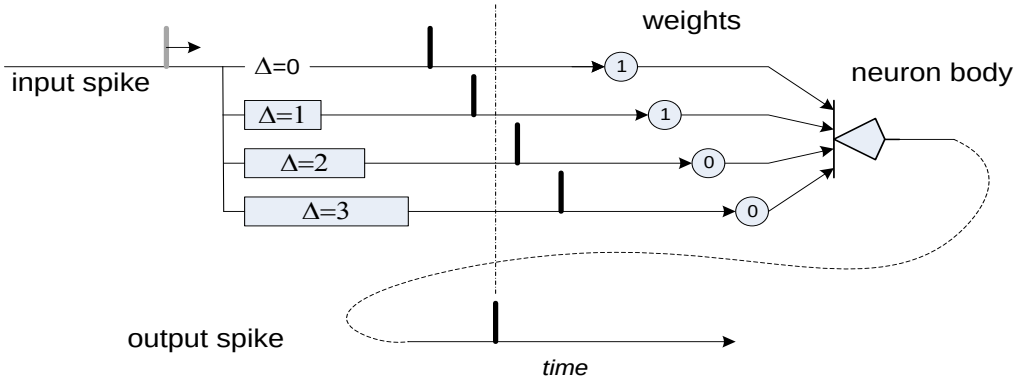



- Uses basic SRM0 model, *except*
- Multiple paths from one neuron to another
  - *Axon drives multiple dendrites belonging to the same downstream neuron*
- Each path has its own delay and synaptic weight
- STDP training yields computationally powerful response functions

# 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

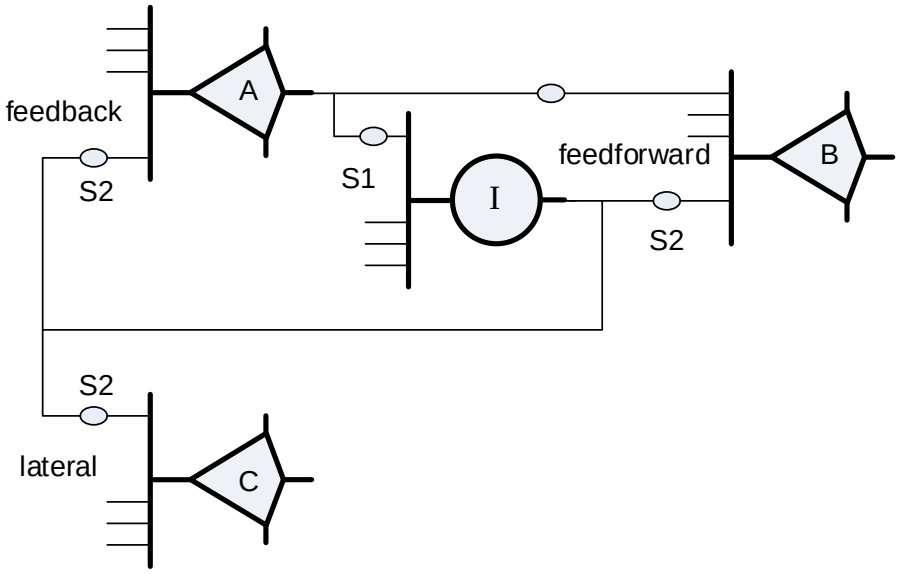
- When applied to a compound synapse – multiple delay paths – the weights reflect average relative input spike times.



# Modeling Inhibition

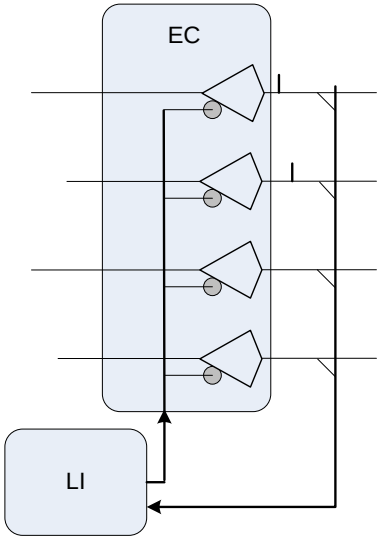
# Inhibition

- Inhibitory neurons:
  - Sharpen/tune and reduce spike "clutter"
    Only first spike(s) are important
  - Provide localized moderating/throttling
    Saves energy
- Only computational applications of inhibition are considered here
- Three basic types of feedback
  - lateral
  - feedforward
  - feedback

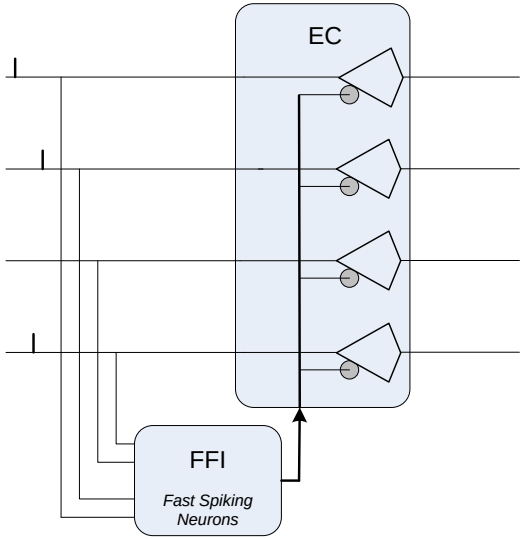


# Modeling Inhibition

- Model in bulk
  - w/ time-based parameters
- Our primary interest is feedforward nets
  - So consider only lateral and feedforward inhibition



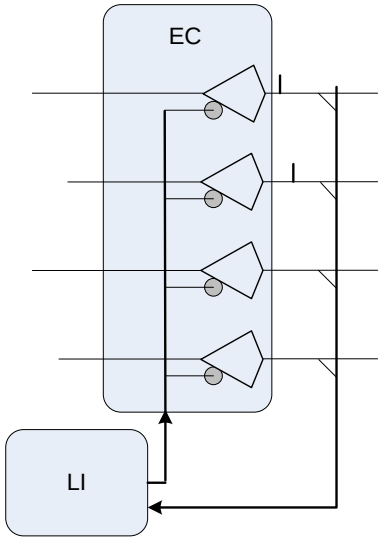

**lateral inhibition**




**feedforward inhibition**

# WTA Inhibition

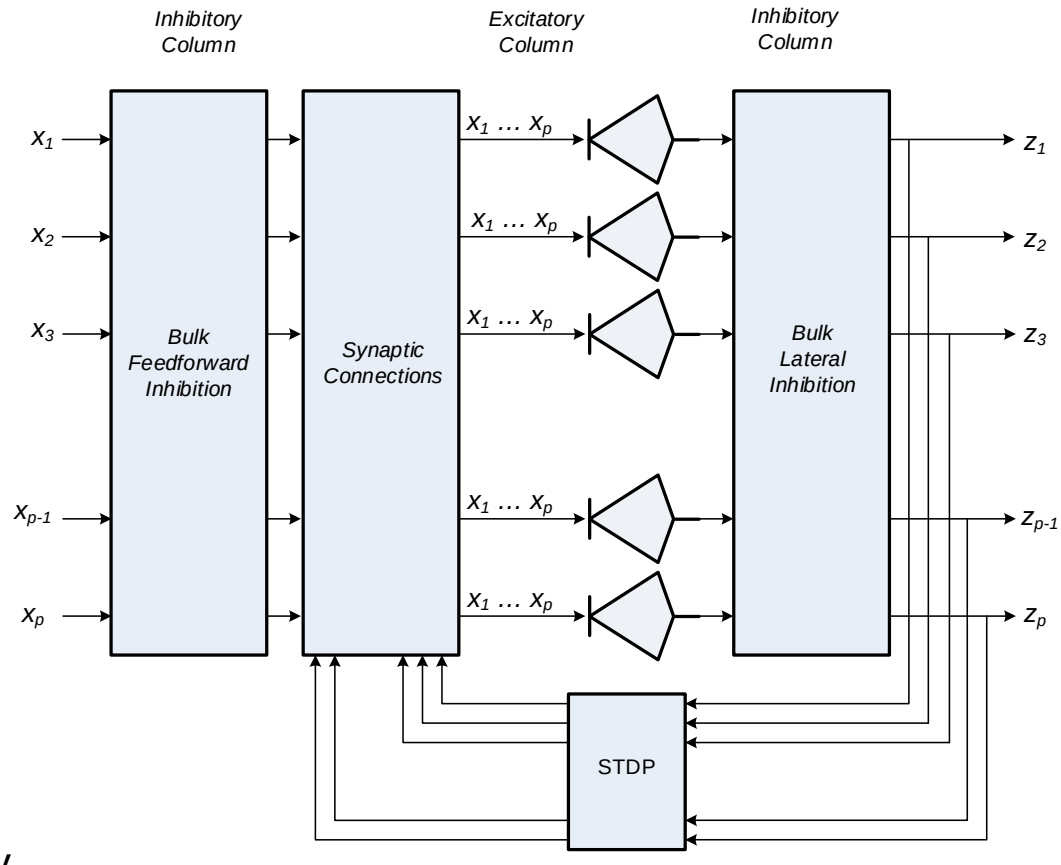
- Winner-Take-All
- Implement parameterized *t*-*k*-WTA
  - All spikes occurring at time > *t* (relative to first spike) are inhibited
  - Of the remaining spikes, at most *k* of them are kept (others are inhibited)
- *Tie cases are very important*
  - After *t* inhibition is applied, more than *k* spikes may remain
  - Say that $k_t$ of them occur at exactly time *t* and these span the *k* boundary.
  - Options:
    1) Inhibit all $k_t$
    2) Inhibit a random selection of $k_t$ so that exactly *k* remain
    3) Inhibit a systematic selection of the $k_t$
- In prototype TNN, I am currently using 0t-1k-WTA
  - Keep only k=1 of the spikes occurring at relative time t = 0
  - To more clearly reveal the basic processing mechanisms that are at work



# General Computing Architecture

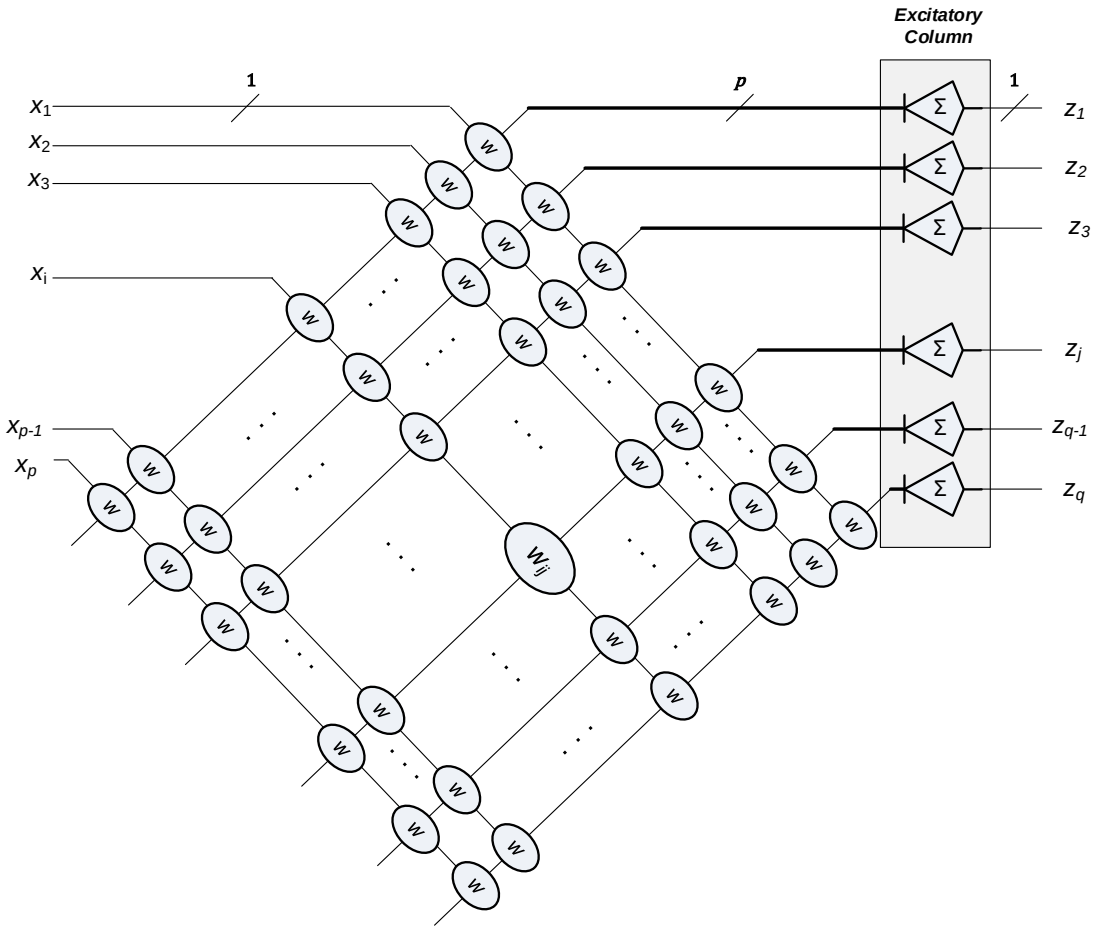
# Computational Column

- *Computational Column (CC)* is basic unit of computation
  - Excitation creates output spikes
  - Bulk inhibition only 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
  - Most do not include feedforward inhibition
- Each neuron in an excitatory column is associated with a specific "feature" in machine learning parlance
  - In effect, the column is a set of feature maps
- Although the inhibitory column may look like "max-pooling", it is not.



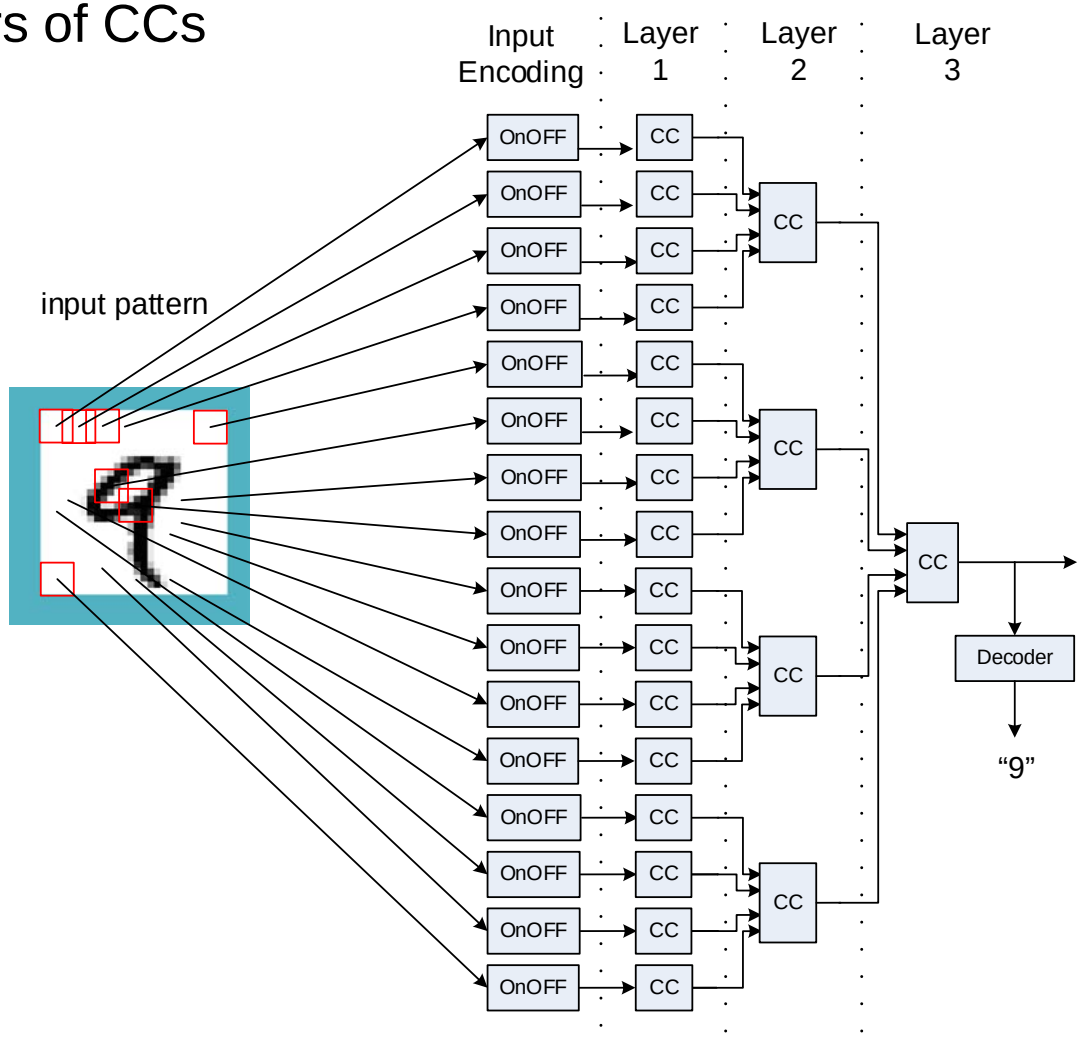
# Synaptic Connections

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



# Typical Network Architecture

- Multiple layers of CCs



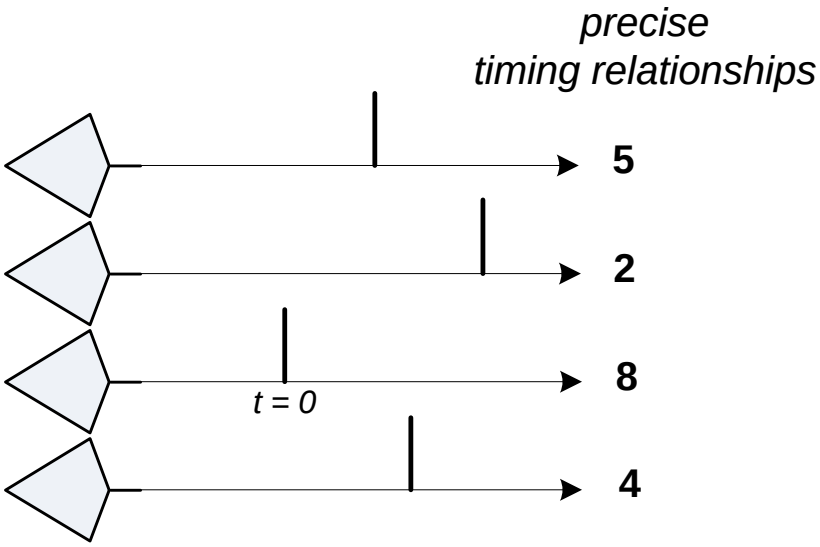
# The Role of Time

- *Training and Evaluation are local*
- Yet, operation is coordinated in some fashion
- The coordinating mechanism is the uniform passage of time

**<Placeholder Slide>**

# FSMI

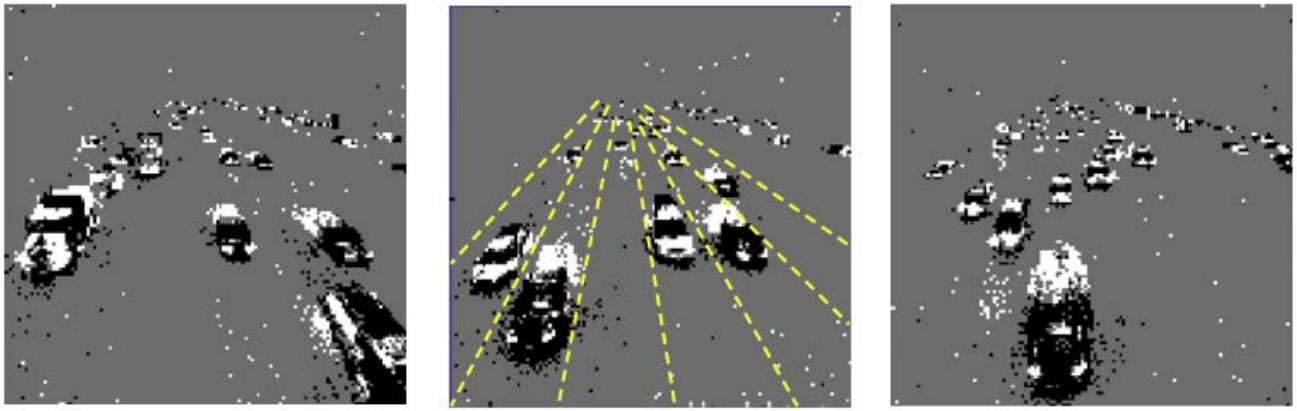
- *F*irst *S*pike is *M*ost *I*mportant
- A natural, pervasive, and guiding principle of temporal models
  - *Neurons*: the best pattern match yields earliest output spike
  - *Volleys*: encode multiple features; the first spike indicates the strongest feature
- Winner-take-all inhibition depends on FSMI



# Case Studies

# 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)

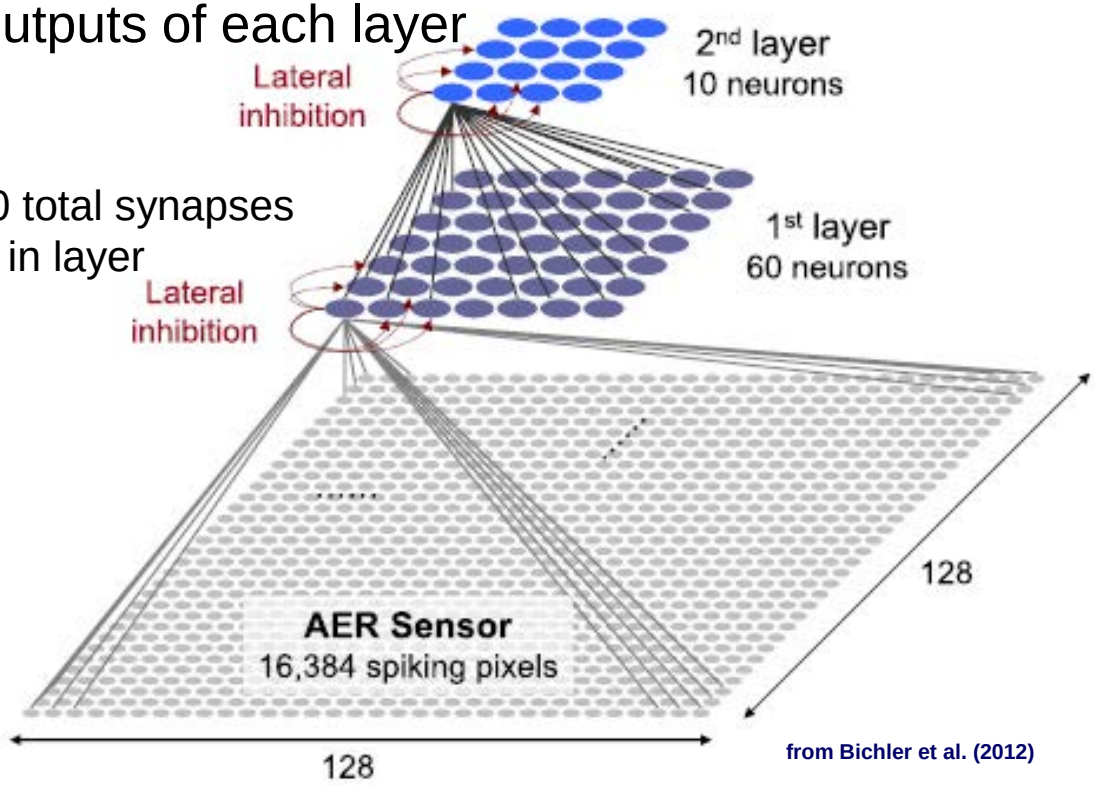
- Feedforward, unsupervised learning w/ STDP
- Silicon retina with Address-Event Representation (AER)
  - Serial link transmitting <pixel, spike time> pairs
  - AER employs a basic sparse vector representation
- Encoder: emits a spike when pixel intensity changes by at least some positive or negative amount
  - 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
2×128×128 60 + 60×10 = 1,966,680 total synapses
Lateral inhibition across all neurons in layer

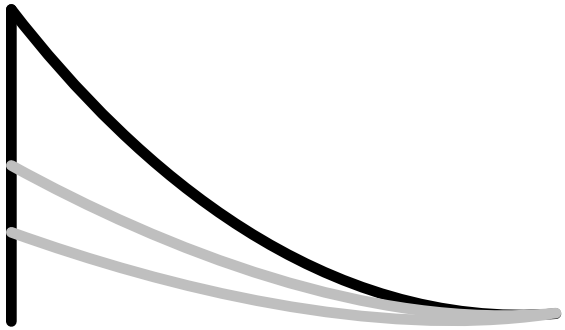



from Bichler et al. (2012)

# Excitatory Neuron Model

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

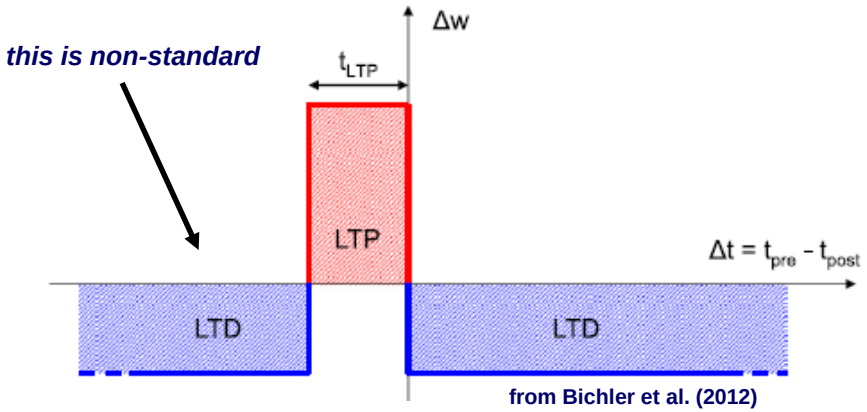
    Summed response functions need only be evaluated in response to an input spike



# 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)**




from Bichler et al. (2012)

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

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

# STDP

- STDP functions
  - Not nearly as complicated as it looks because βs are 0

$$\Delta w_+ = \alpha_+ \cdot \exp\left(-\beta_+ \cdot \frac{w - w_{\min}}{w_{\max} - w_{\min}}\right). \quad (1)$$

In the LTD case, the equation is quite similar:

$$\Delta w_- = \alpha_- \cdot \exp\left(-\beta_- \cdot \frac{w_{\max} - w}{w_{\max} - w_{\min}}\right) \quad (2)$$

where $\alpha_+ > 0, \beta_+ \geq 0, \alpha_- < 0$ and $\beta_- \geq 0$ are four parameters. $w$ is the weight of the synapse and is allowed to change between $w_{\min}$ $(>0)$ and $w_{\max}$. Depending on the two $\beta$ parameters, one can have either an additive $(\beta = 0)$ or a pseudo-multiplicative weight update rule, which can model different possible hardware (or software) implementations (Querlioz, Dollfus, Bichler, & Gamrat, 2011) without compromising the working principle of the proposed scheme.

| 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. |

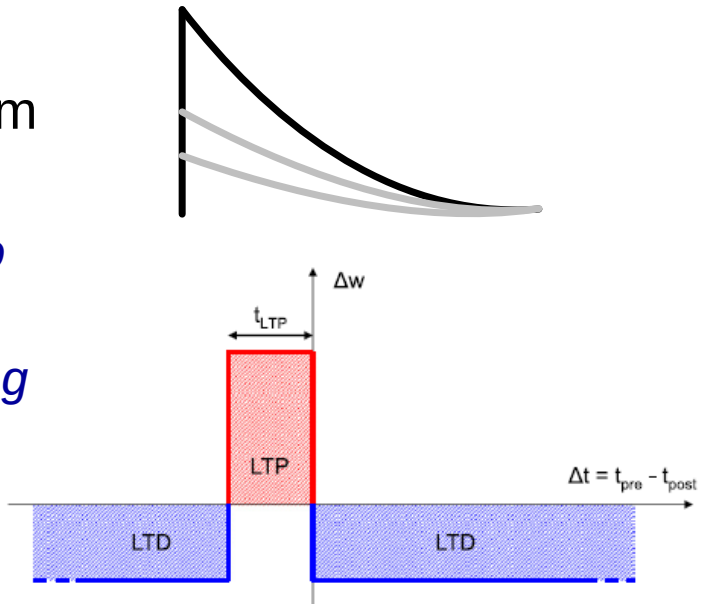
from Bichler et al. (2012)

# Inhibition

- Bulk WTA Lateral Inhibition
- When a neuron spikes, it inhibits all others from spiking for time $T_{inhibit}$
  - Divides processing into softly synchronized "chunks" aka volleys
  - This may be a good model for synchronizing inhibitory oscillations
  - Synchronization is needed because input is taken from moving objects
- However -- *there is no feedback in the processing path*
  - Although object is moving, network actually processes static "snapshots"
- During training: Inhibition causes pseudo-randomly initialized neurons to converge to different final synaptic states
  - Consequently, each neuron identifies a different prominent input feature

# 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 | |

# Training Process

- To properly train multiple neurons w/ the same RF, there must be some differences in connectivity and/or synaptic parameters
  - Otherwise all the neurons will train to identical synaptic weights
  - One option: use pseudo-random connectivity so that each neuron sees a different set of input spikes
  - Case study architecture: initialize synapses to pseudo-random values

    Coupled w/ WTA inhibition, this causes different neurons to converge to synaptic weights
- Two training methods
  - Global – train multiple layers at once
  - Layer-by-layer – train 1st layer, then turn off lateral inhibition and STDP, and train 2nd 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

# Synaptic Parameters

- Synaptic parameters
  - $w_{min}$, $w_{max}$, and $w_{init}$ are pseudo-random
  - Note: neurons have different $w_{max}$ values
  - 4 to 5 bits of weight resolution

| Parameter | Mean | Std. dev. | Description |
|---|---|---|---|
| $w_{min}$ | 1 | 0.2 | Minimum weight (normalized). |
| $w_{max}$ | 1000 | 200 | Maximum weight. |
| $w_{init}$ | 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. |

# 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

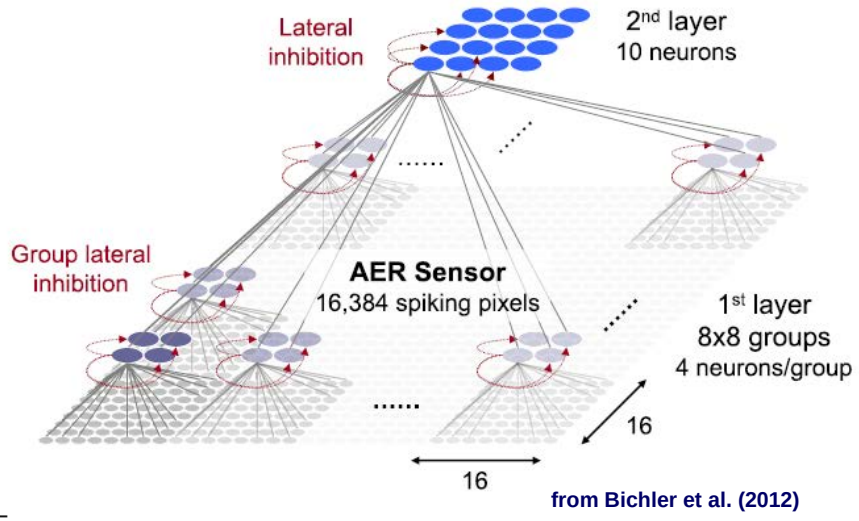
| Parameter | Global learning | | Layer-by-layer learning | |
|---|---|---|---|---|
| | 1st layer | 2nd layer | 1st layer | 2nd layer |
| $I_{thres}$ | 500 000 | 1500 | 1 060 000 | 2240 |
| $T_{LTP}$ | 12 ms | 300 ms | 14.7 ms | 46.5 ms |
| $T_{refrac}$ | 300 ms | 250 ms | 517 ms | 470 ms |
| $T_{inhibit}$ | 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 | |

- In Essence: This accuracy measure employs a simple classifier, with supervised training

# Advanced System Architecture

- To reduce cost
  - 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 × 16 pixels each
With 4 neurons per square group
8×8×4 + 10 = 266 total neurons
16×16×2×8×8×4 + 8×8×10 = 131,712 synapse
Lateral inhibition is localized to the group

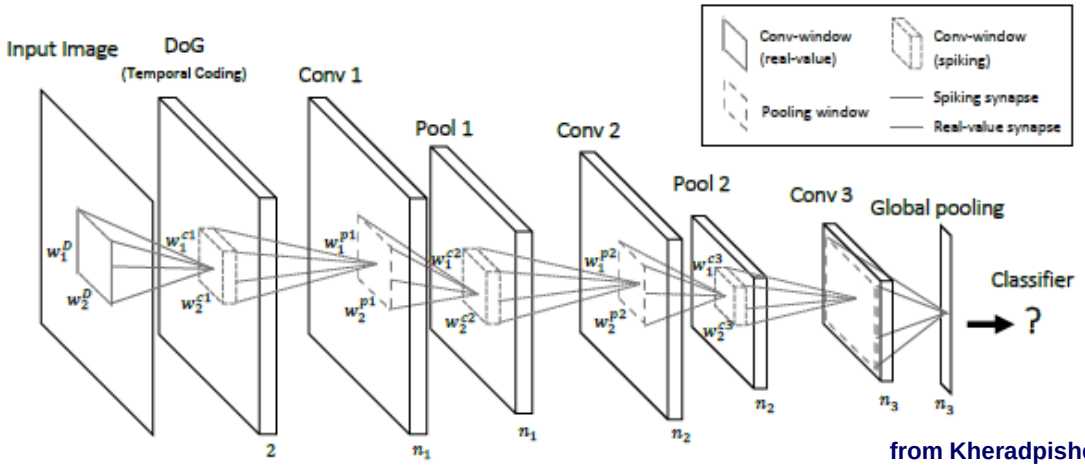



from Bichler et al. (2012)

| 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 their MNIST implementation




from Kheradpisheh et al. (2016)

*"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)

    (Increasing numbers of maps w/ increasing layers is not a good thing)
  - Global pooling prior to classification uses max body potential, not spike time

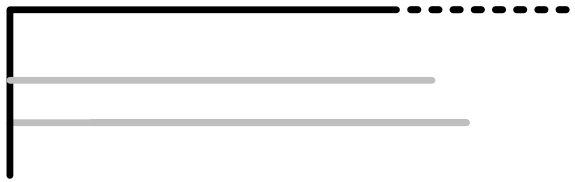
    Non-causal

# 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
  - Only *sign* of time difference taken into account
  - Lower weights get larger increments (or decrements)

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

# 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
- *There is 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
  - At most 30 time steps per image
- Summary: fast, efficient, unsupervised training

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

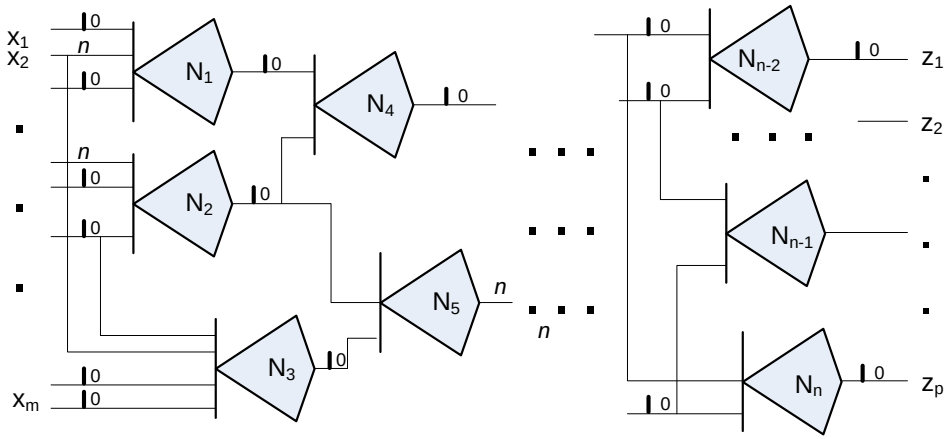
| Architecture | Neural coding | Learning-type | Learning-rule | Accuracy (%) |
|---|---|---|---|---|
| Dendritic neurons [48] | Rate-based | Supervised | Morphology learning | 90.3 |
| Convolutional SNN [37] | Spike-based | Supervised | Tempotron rule | 91.3 |
| Two layer network [36] | Spike-based | 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 | **99.1** |
| Proposed SDNN | Spike-based | Unsupervised | STDP | 98.4 |

# MNIST Efficiency

- Placeholder

# 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*}




**like binary network if:**
**n => binary 1**
**0 => binary 0**

- Is a special case TNN, where everything happens at the same time
- The time resource is not used

# Prototype TNN Architecture

***currently being developed by the author***

# Architecture Objectives: Functions and Features

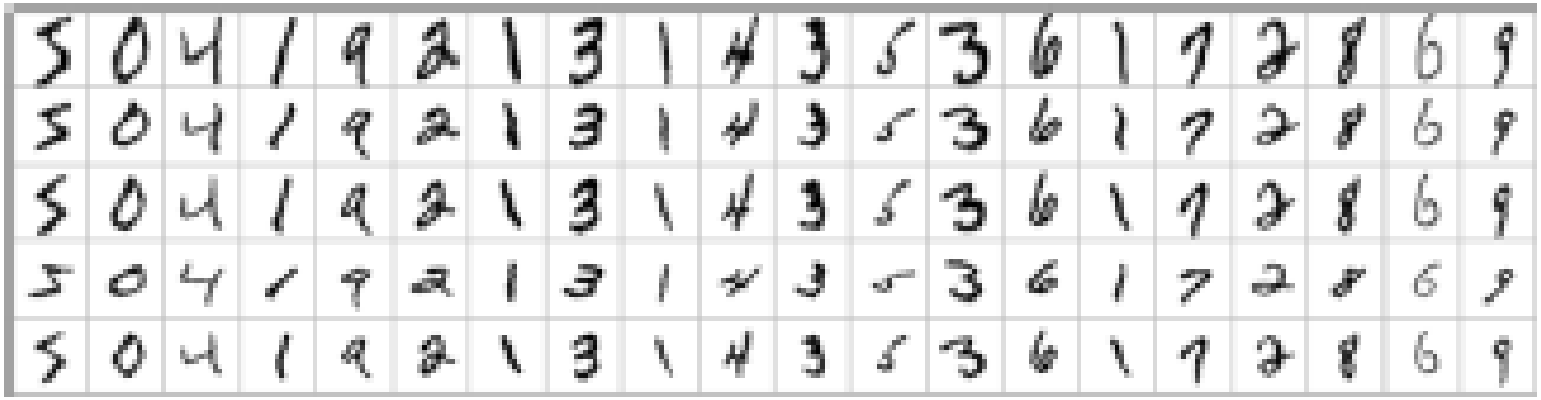
- TNNs: spiking neural networks, based on precise spike timing
- Basic function: clustering
  - Coalesce similar input patterns into *clusters*

    I.e. *Similar input patterns map to similar output patterns*

    (with many fewer output patterns than input patterns)
- Features (goals)
  - Uniform processing of input from a variety of sensory or logical sources
  - Unsupervised training
  - Training performed concurrently with evaluation
    - Training is as efficient as evaluation
    - Both are linear with respect to input size with small constants
  - Low precision implementation
  - Scalable
- Important simplifying assumption:
  - Use single-spike cluster IDs

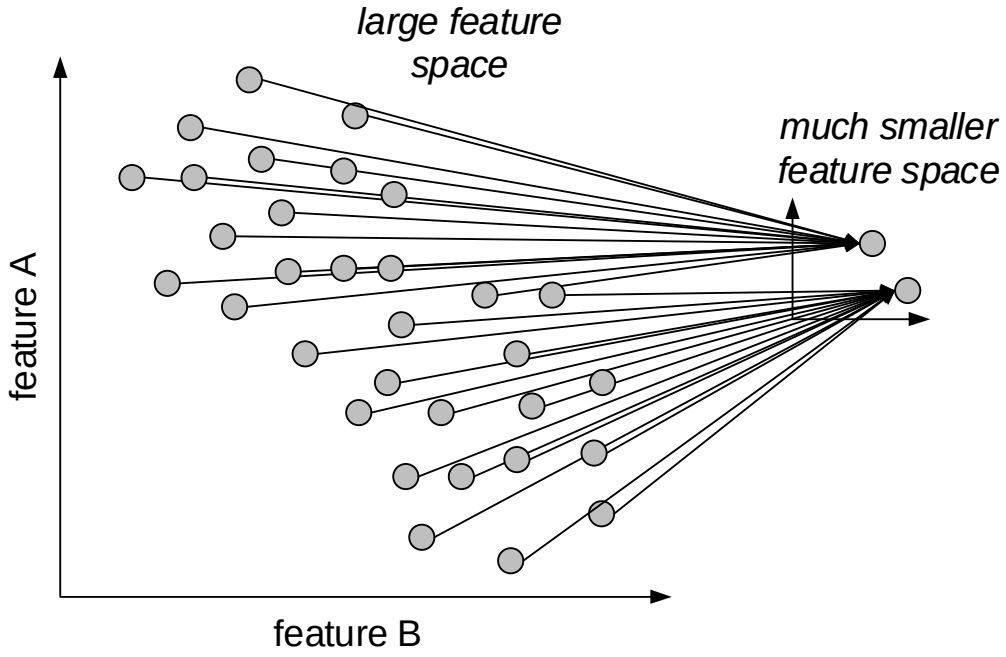
# Benchmark: MNIST

- A Goldilocks Benchmark
  - Not too large, not too small
  - Not too hard, not too easy
  - *Just right*
- What it is:
  - Tens of thousands of 28 x 28 grayscale images of written numerals 0-9
  - *Originally* hand-written zip codes
  - Pixelized w/ interpolation from B&W images
  - Labeled: 60K training patterns; 10K evaluation patterns



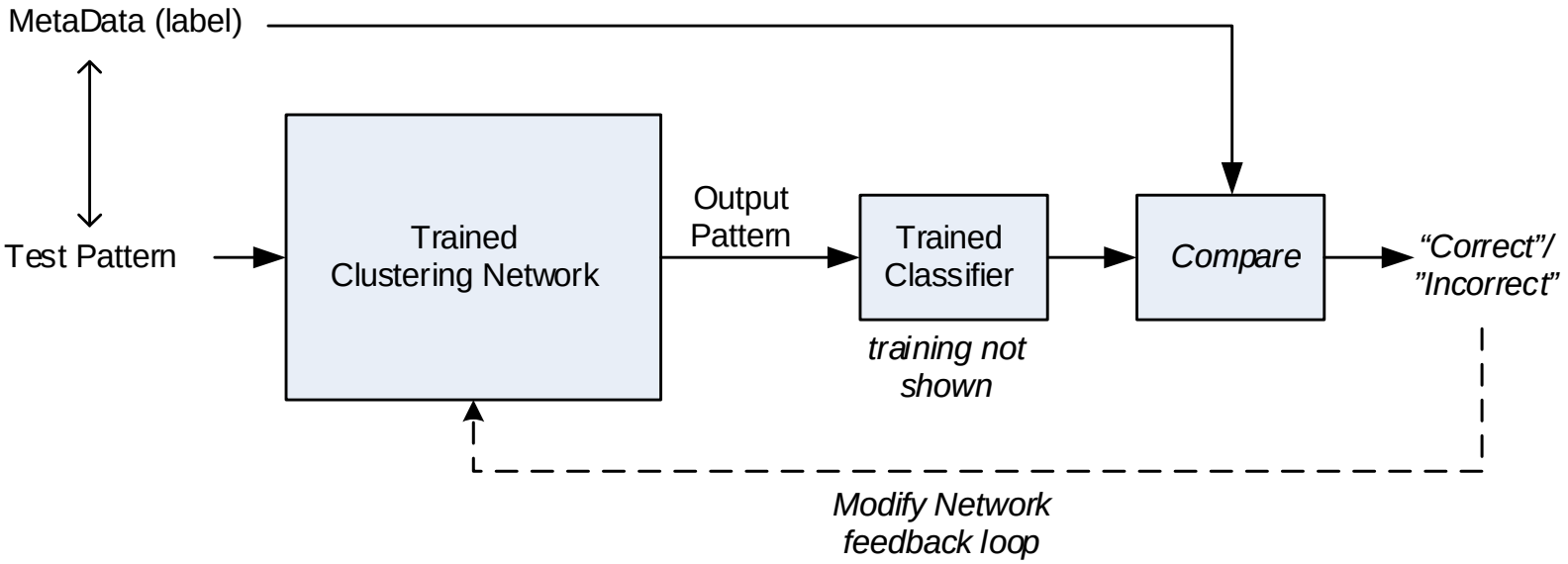
# Clustering

- Unsupervised training
  - Unlabeled input patterns
- Similarity is determined by the data, itself
  - Output feature space has fewer dimensions (*not illustrated in this simplified 2-d drawing!*)
  - There may be more or fewer output “clusters” than input “classes”



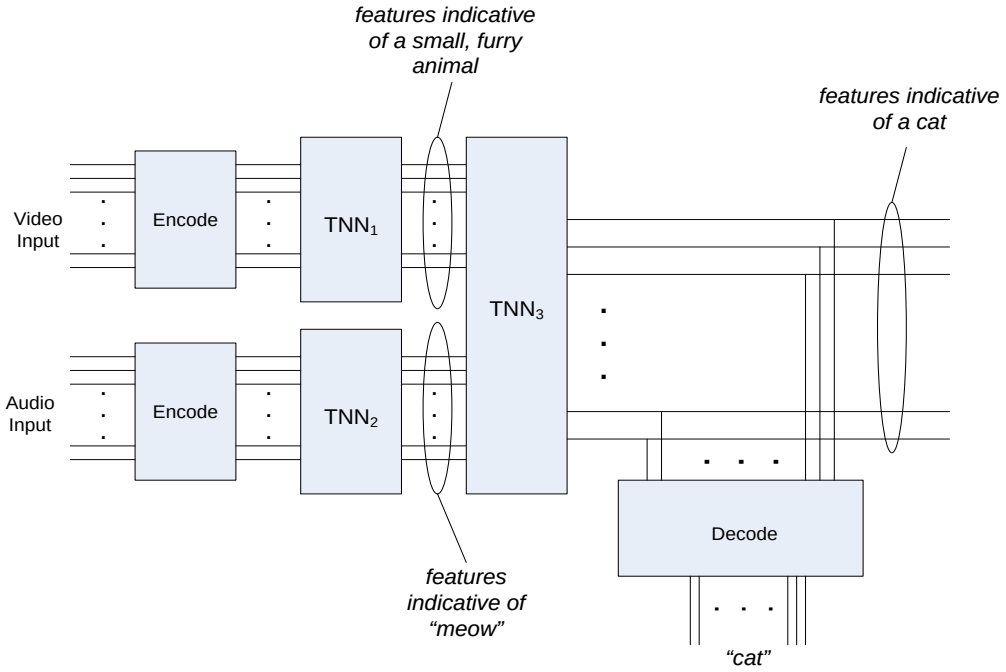
# Unsupervised Training -- Measuring "Accuracy"

- A challenge
  - Mapping from output pattern to cluster identifier is opaque
- Add trained classifier to output to make it more transparent
  - Only 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)



# Output Decoding

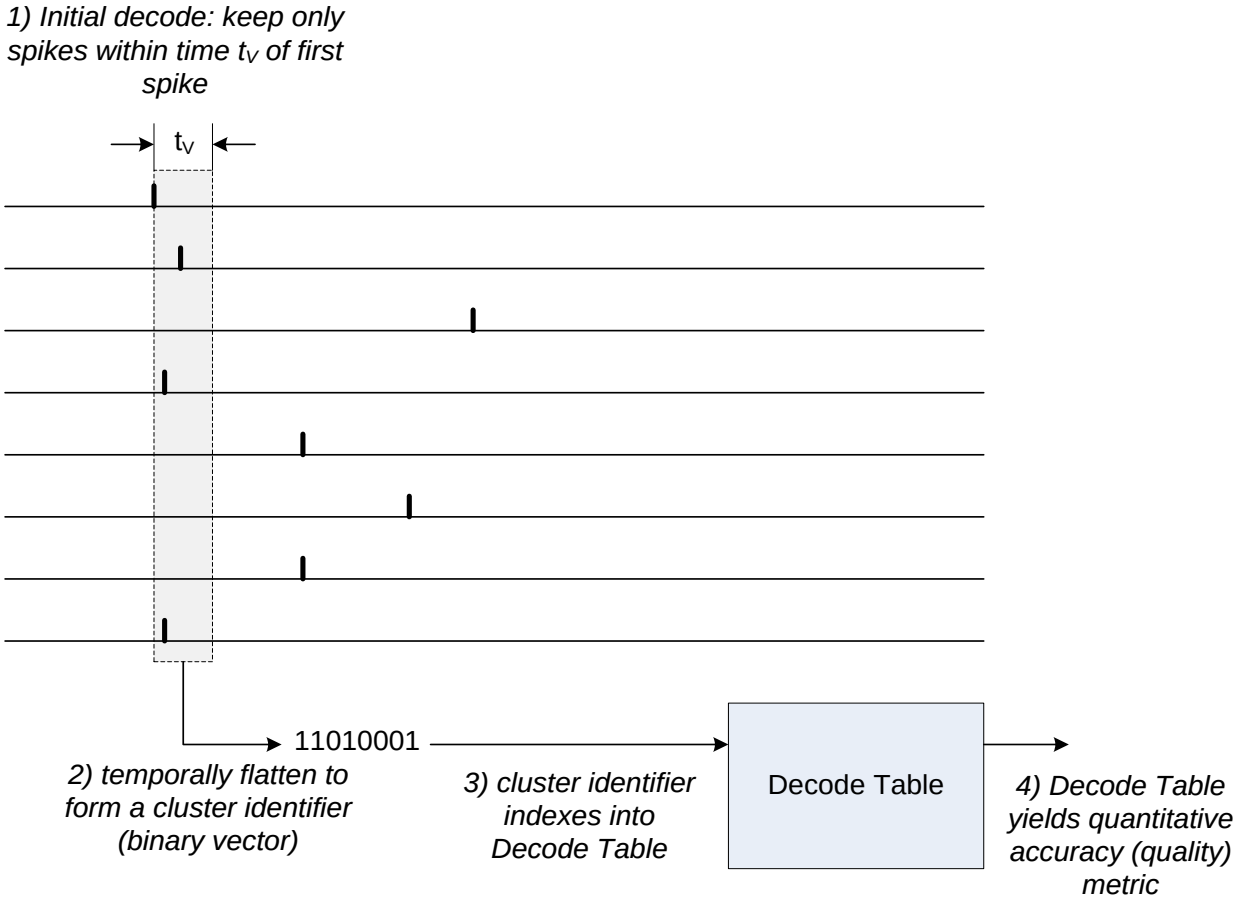
- If the outputs feed another TNN block, then no decoding is needed
- If we, as humans, want to understand what the outputs are "saying" then we need to decode them
  - Unsupervised training (and connectivity pattern) chooses the cluster identifiers




- Construct a small, simple output classifier to understand what is going on
  - Use a Volley Analyzer for this purpose

# Volley Analyzer

- First, decode by flattening into a binary cluster identifier
- Note: some temporal information is thrown away



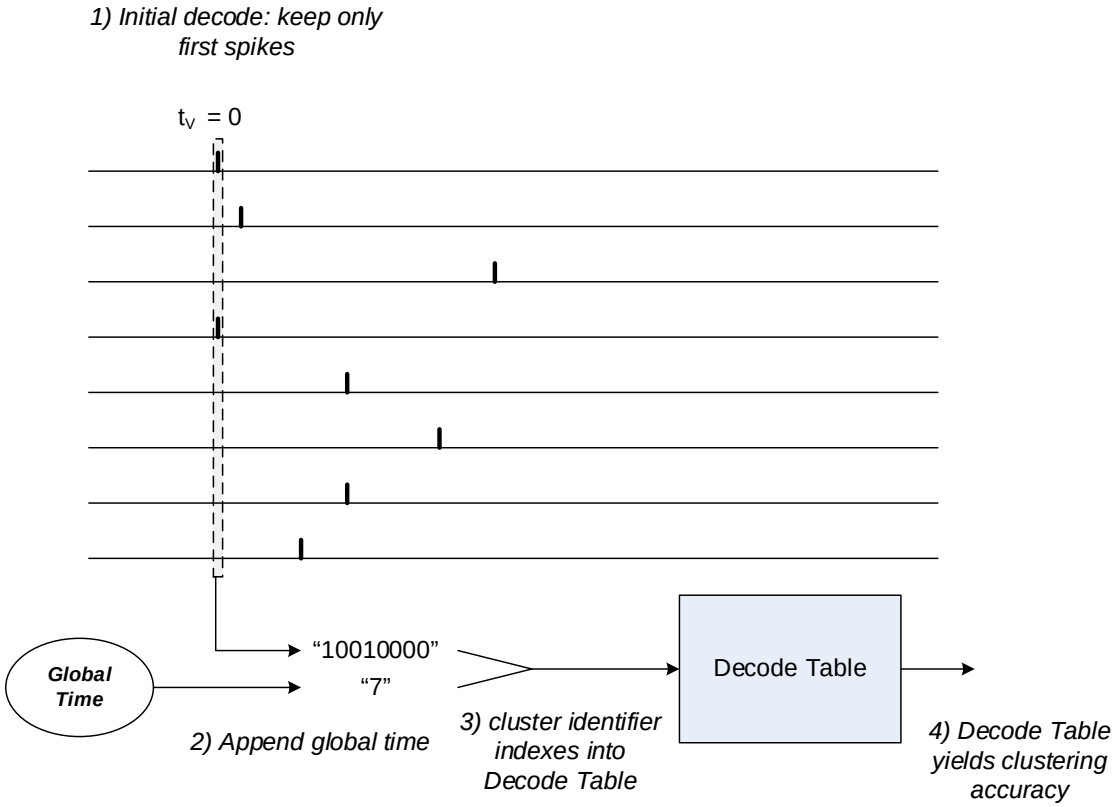
# Volley Analysis

- Example

| | | label counts | | | | | similarity |
|---|---|---|---|---|---|---|---|
| cluster identifier | A | B | C | D | total | maximum | 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 | |

# Volley Analyzer: Special Case

- If spikes are 1-WTA-filtered – i.e., $t_V = 0$; only literal *first* spikes
- There is no flattening needed, and spike time is appended to cluster identifier
- Then, volley analyzer becomes strict (not a slightly pessimistic approximation)



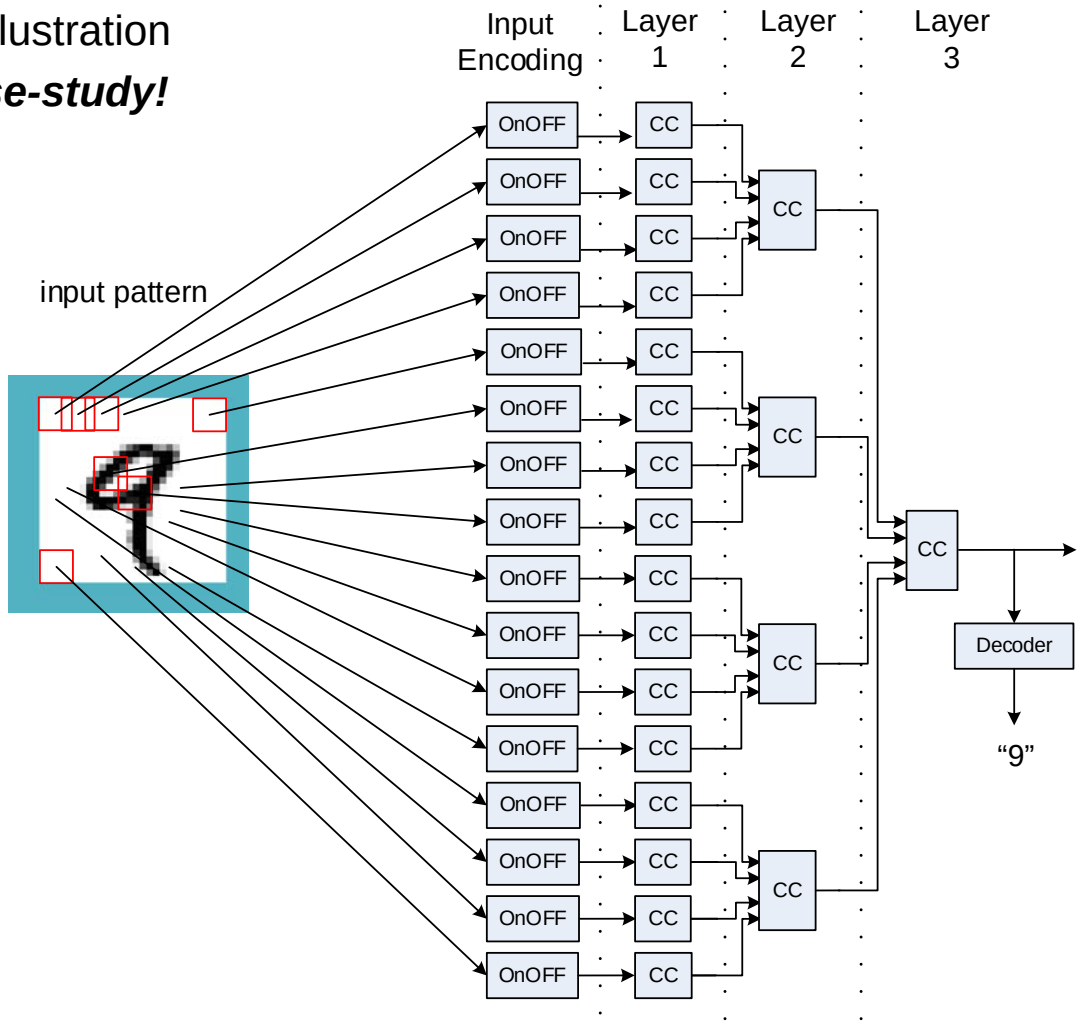
# Volley Analysis: Special Case

- Example

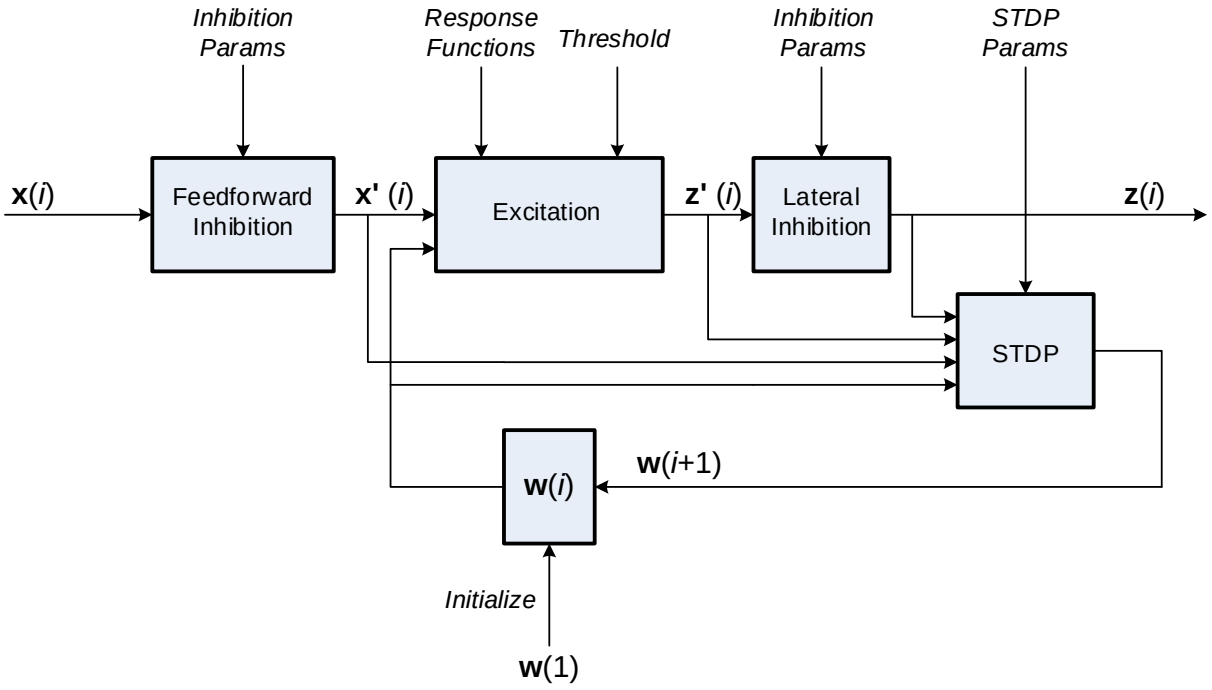
| cluster identifier | | label counts | | | | | | similarity |
|---|---|---|---|---|---|---|---|---|
| spikes | time | A | B | C | D | total | maximum | group |
| 000100 | 7 | 11 | 34 | 58 | 0 | 103 | 58 | C |
| 010000 | 7 | 49 | 116 | 192 | 329 | 686 | 329 | D |
| 000001 | 7 | 41 | 71 | 426 | 11 | 549 | 426 | C |
| 001000 | 7 | 4 | 2 | 35 | 0 | 41 | 35 | C |
| 100000 | 7 | 0 | 6 | 160 | 7 | 173 | 160 | C |
| 000001 | 8 | 15 | 397 | 36 | 2 | 450 | 397 | B |
| 001000 | 8 | 0 | 24 | 2 | 0 | 26 | 24 | B |
| 000100 | 8 | 56 | 0 | 6 | 2 | 64 | 56 | A |
| 000100 | 9 | 110 | 0 | 95 | 45 | 250 | 110 | A |
| | | | | | | 2342 | 1595 | |
| | | | | | | accuracy = | 0.68 | |

# Prototype Network Architecture

- Dimensions shown are only for illustration
- ***Consider this to be a large case-study!***
- Use WTA-1 encoding throughout (except Layer 1)
  - A simplification that yields insight
  - Can later be extended
  - Sparse
- *Unsupervised*, *concurrent* training



# Computational Column Block Diagram

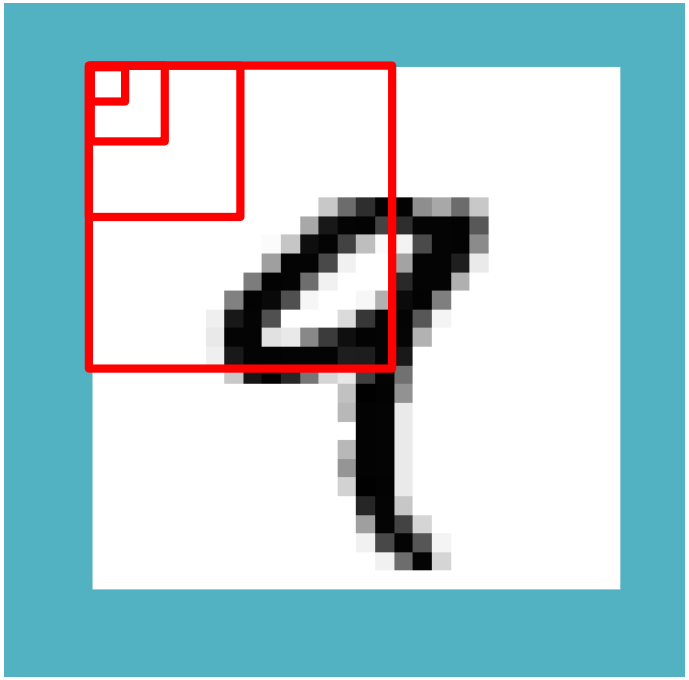



- Drawn as a finite-state-machine
  - Weights, w(*i*), are the only state
  - STDP modifies weights which feed back into computation
  - Everything else is *forward flow*

# Use an RF Tiling Structure

- Simple example:

  Layer 1: 2x2 RFs
  Layer 2: 4x4 RFs
  Layer 3: 8x8 RFs
  Layer 4: 16x16 RFs
  Layer 5: 28 x 18 RF

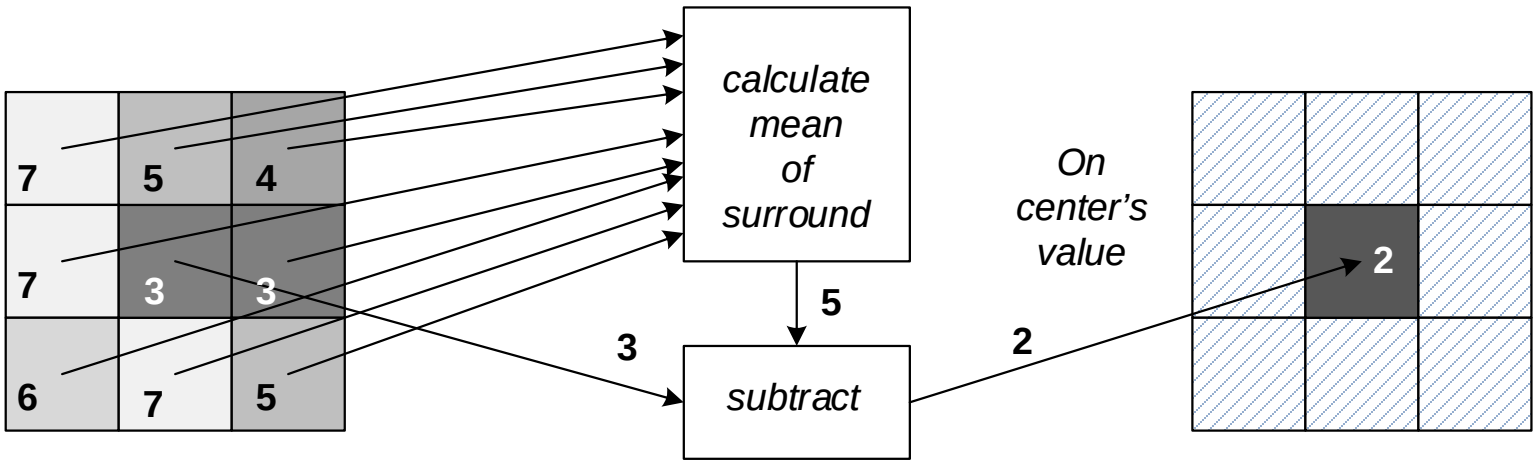


28x28 image

# Input Translation (OnOff Encoding)

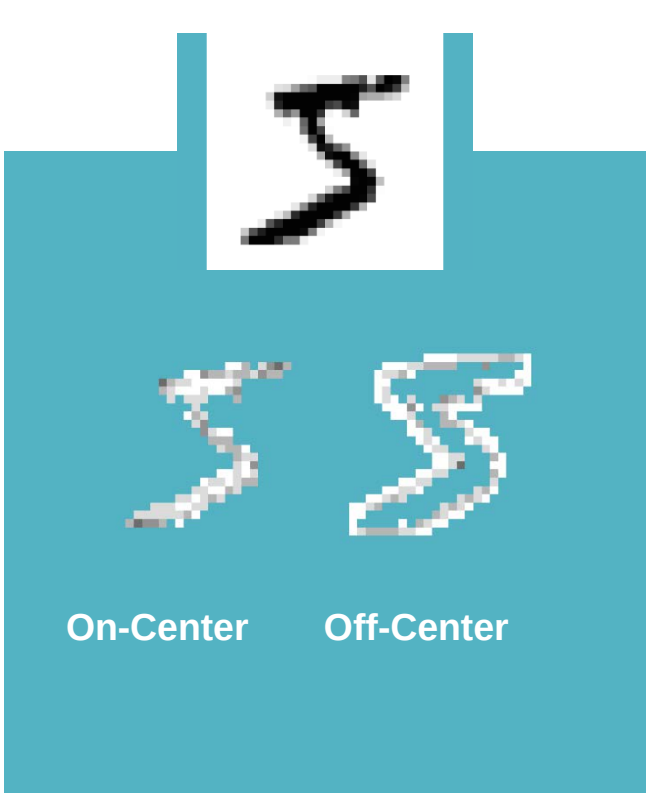
- Done on per pixel basis
- Compute mean of "surround"
  - Currently using arithmetic mean
- Compare with "center" – subtract
- Determine center's output spike time

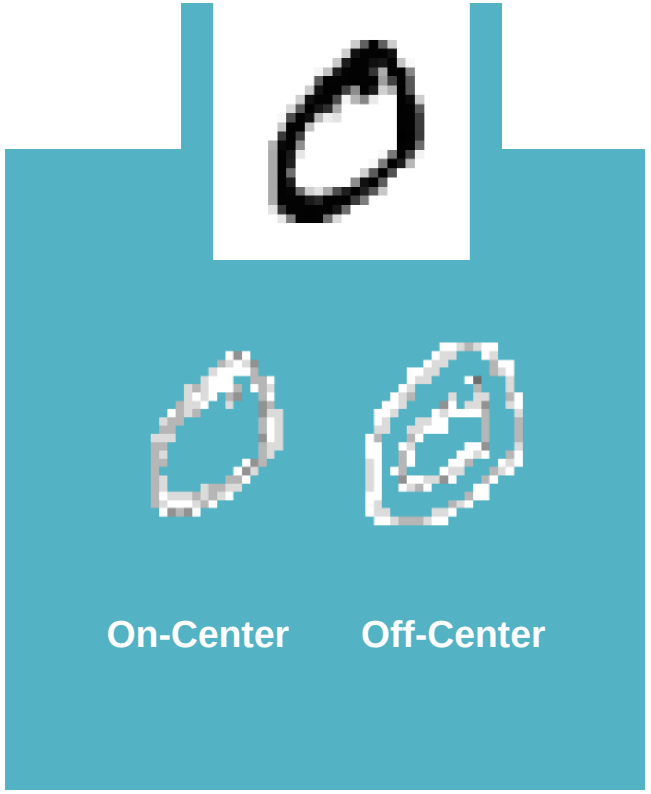
- Example is an On-Center cell
  - Convention: a lower number is more intense input & earlier spike
- Same inputs for an Off-Center cell yields no spike
- On-Center and Off-Center for same pixel *never* simultaneously spike



# Input Translation (OnOff Encoding)

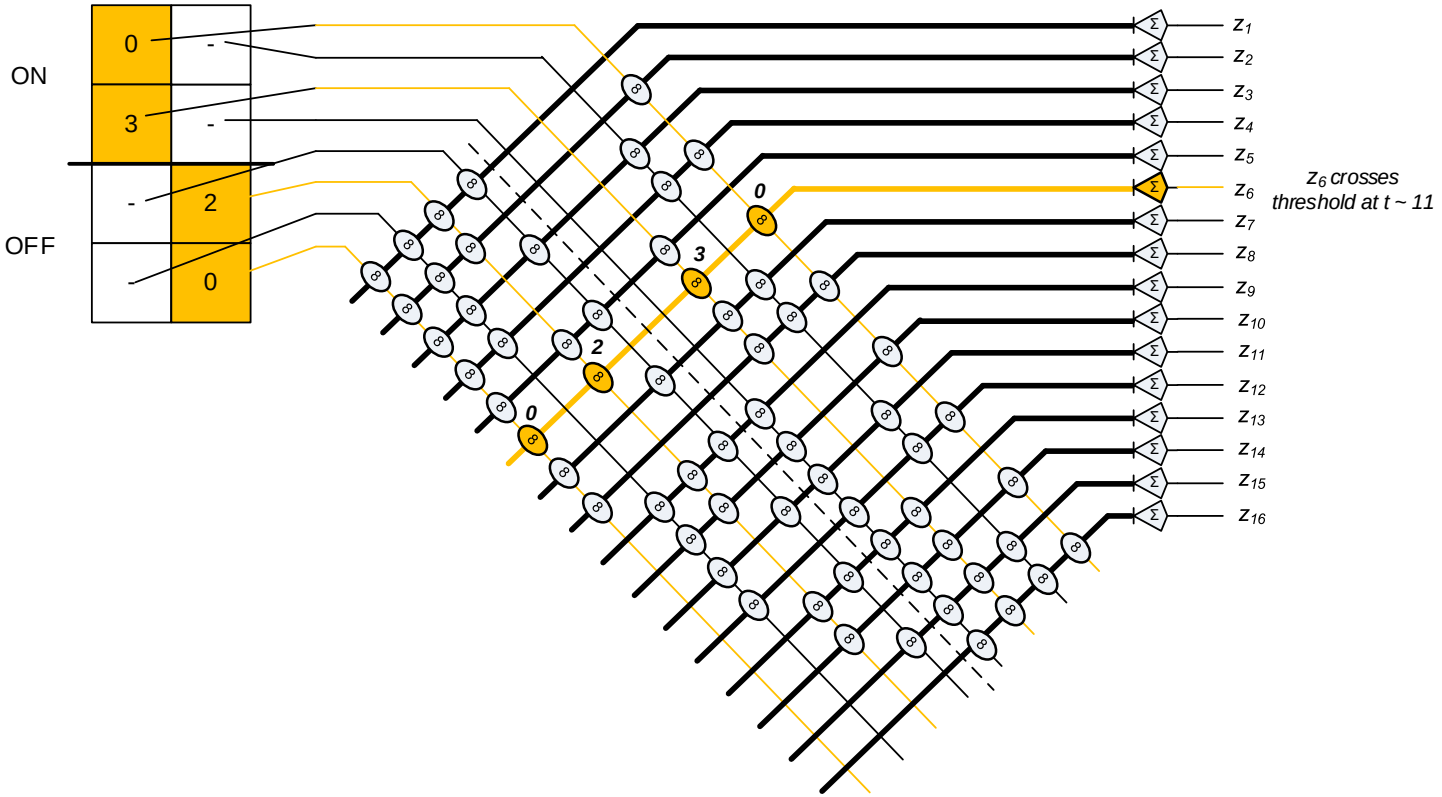
- MNIST examples



# Layer 1 Implementation

- Convert grayscale to *OnOff*
- Perform complete decode
  - Four pixels yield $2^4 = 16$ different spiking patterns (ignoring times)
  - Lateral inhibition and STDP paths not shown



# STDP: the Anthropomorphized Synapse

- Synapses want to do their part for the greater society when establishing their own personal weights.
- It is better for the greater society to capture a set of dominant clusters

So, what each synapse "wants" to do:

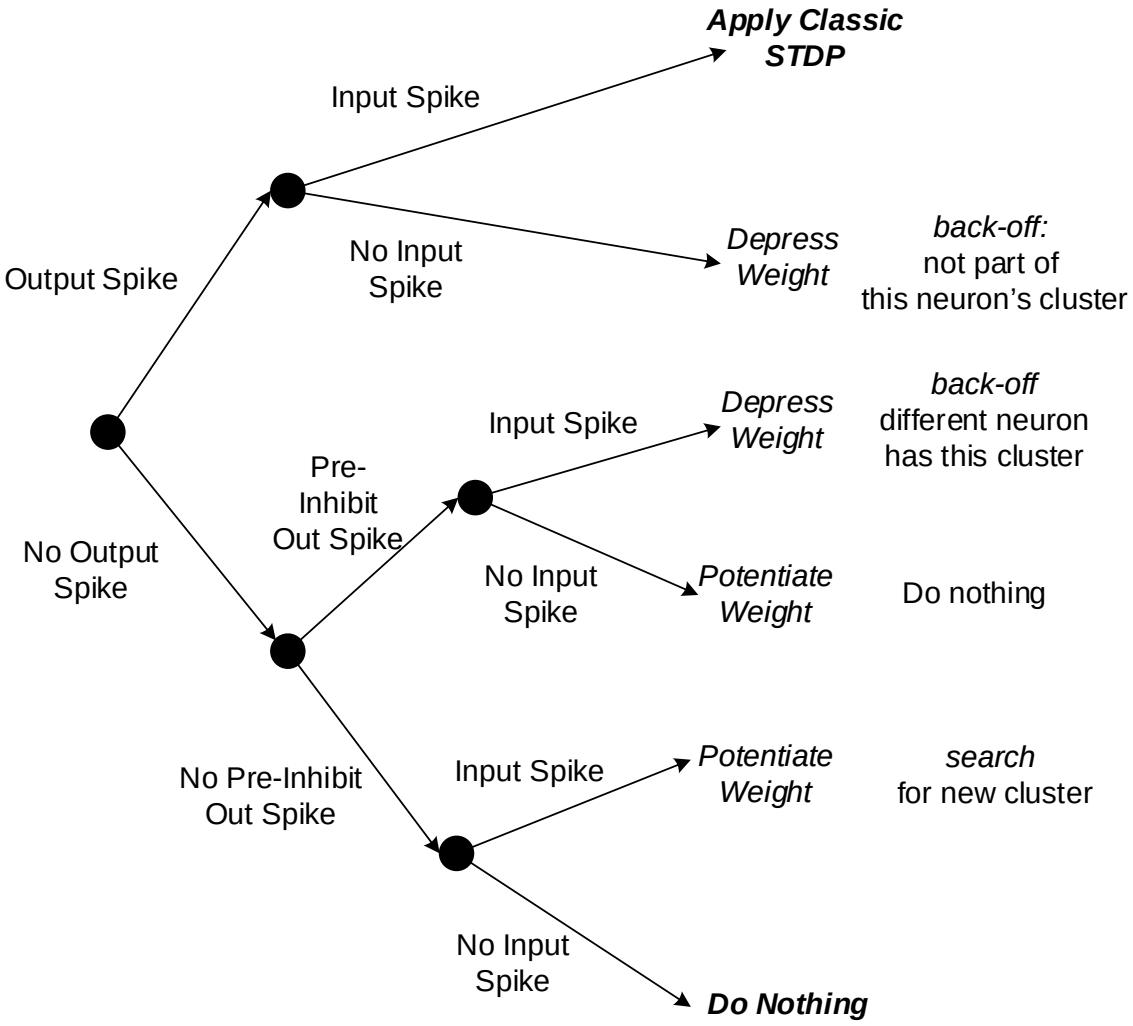
- If its neuron is not already associated with a cluster:
  - The synapse wants to become part of a new cluster
- If its neuron is already associated with a cluster:

  1) And If the synapse is associated with that cluster, it wants to remain so (and even strengthen)

  2) Else if the synapse is *not* associated with the cluster, it wants to remain so (and even weaken)
- If its neuron's current cluster eventually becomes less dominant:
  - The synapse wants to start looking for a new, more dominant cluster

# Decision Tree

- Performed at each synapse
  - AFTER inhibition
- Only two cases commonly discussed in literature (in bold)
- *Search* case may be result of astrocytes (glial cells)
- The ability to differentiate pre-inhibit output spikes is not supported by experiment (nor contradicted)
- Perhaps modulate back-off w/ neuron potential in absence of inhibition (more potential, more back-off)



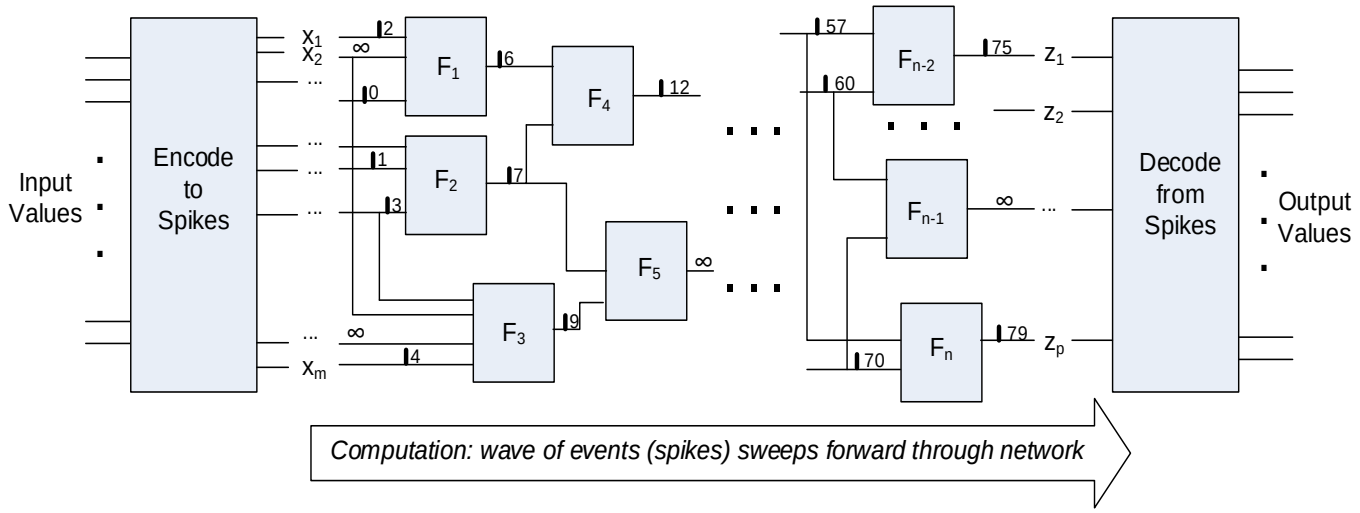
# Current TNN Model (3rd Generation)

- Excitatory neurons with piecewise, non-leaky response functions (*recent change*)
  - Uses space-time model of computation
- Weights and times at 3 bits of resolution (*changed about a year ago*)
- Simple synapse model (*recent change*)
- WTA-1 inhibition (*recent change*)
- Anthropomorphic STDP (*new*)
  - Includes pseudo-randomness
- OnOff encoding (*new*)
  - Previously used PosNeg

Almost all of the above changes involve *simplification* over previous generation

# Space-Time Theory and Algebra

# Space-Time Computing Network




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

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

All $F_i$ satisfy:

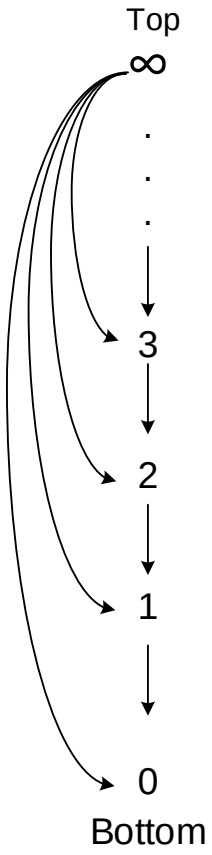
*computability:* $F_i$ implements a computable total function.

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

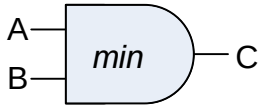
*invariance:* $F_i(x_1 + 1 .. x_q + 1) = F_i(x_1..x_q) + 1$

*TNNs are Space-Time Computing Networks*

# Space-Time Algebra

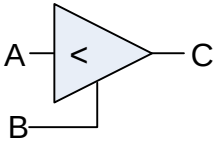



- Bounded Distributive Lattice
  - 0, 1, 2,…, ∞
  - Ordering relation: “>”
  - Interpretation: points in time
- Primitives:



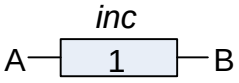

if A < B then C = A
else C = B

*“atomic excitation”*




if A < B then C = A
else C = ∞

*“atomic inhibition”*

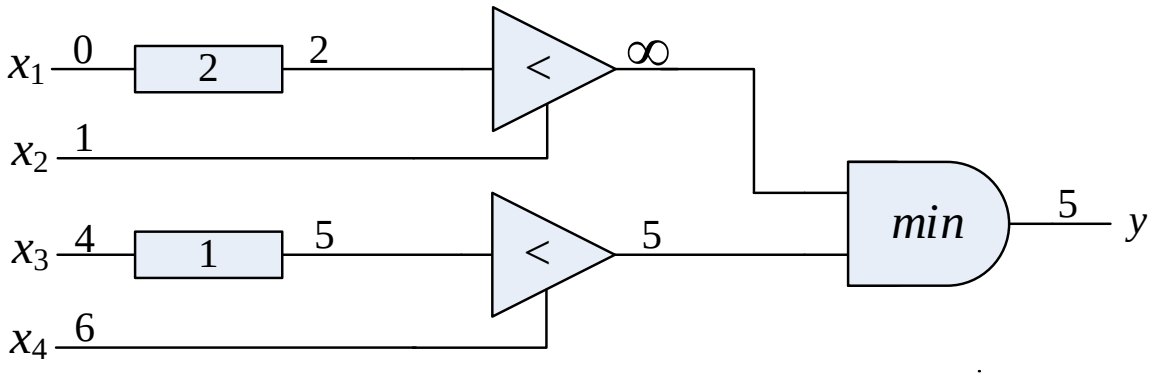



B = A + 1

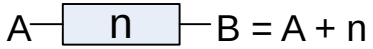
*“atomic delay”*

# Space-Time Algebra, contd.

- *Theorem*: Any feedforward composition of *s-t* functions is an *s-t* function
  - Build networks by composing primitives
  - Example:

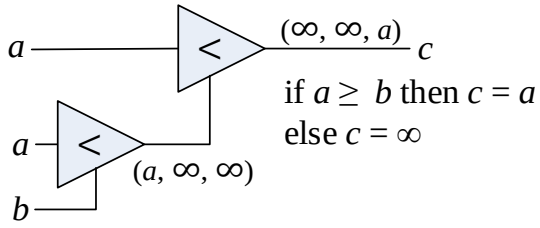



note: shorthand for n increments in series:

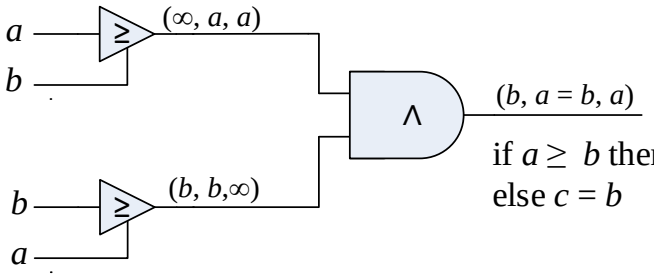


# Implementing *max* function

- *Theorem*: *max* can be implemented with *min* and *lt* only.
  - Construction:

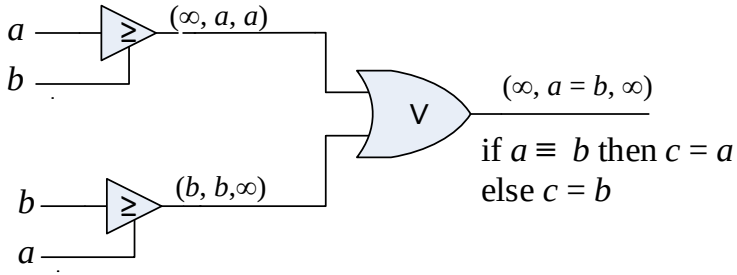



**$c = a \geq b$**

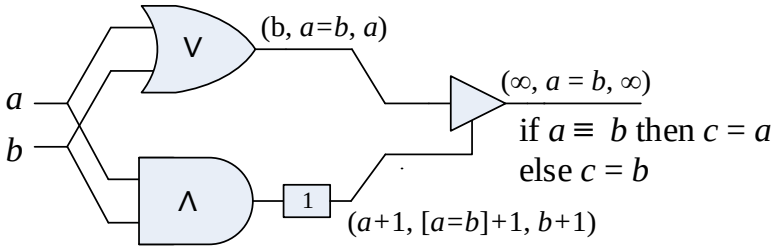



**$c = a \vee b$**

ack: Cristóbal Camarero




**$c = a \equiv b$**




**$c = a \equiv b$**

***cases:* ( $a < b$ , $a = b$, $a > b$ )**

# Elementary Functions

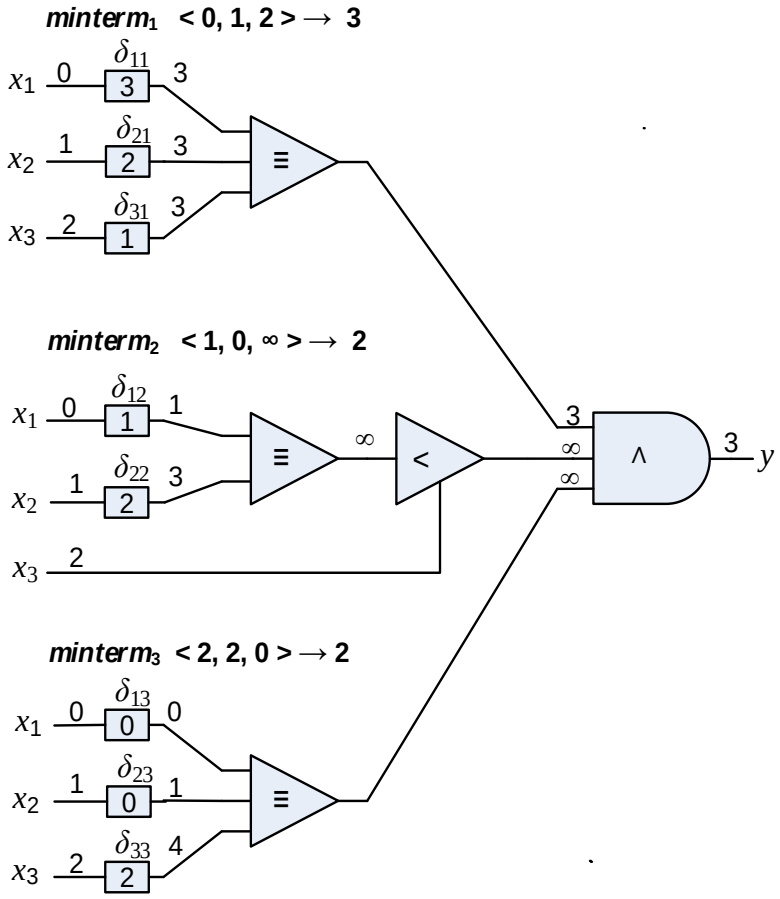
- Increment and identity are the only unary functions
- Exhaustive table of two-input functions
  - Most of similar complexity

| a < b | a = b | a > b | function | name | symbol |
|---|---|---|---|---|---|
| a | a=b | a | **a** | **identity** | **a** |
| a | a=b | b | **if a ≤ b then a; else b** | **min** | ∧ |
| a | a=b | - | **if a ≤ b then a; else ∞** | **less or equal** | ≤ |
| a | - | a | **if a ≠ b then a; else ∞** | **not equal** | ≠ |
| a | - | b | **if a < b then a; else if a > b then b; else ∞** | **exclusive min** | x∧ |
| a | - | - | **if a < b then a; else ∞** | **less than** | < |
| b | a=b | a | **if a > b then a; else b** | **max** | ∨ |
| b | a=b | b | **b** | **identity** | **b** |
| b | a=b | - | if a ≤ b then b; else ∞ | | |
| b | - | a | **if a > b then a; else if a < b then b; else ∞** | **exclusive max** | x∨ |
| b | - | b | if a ≠ b then b; else ∞ | | |
| b | - | - | if a < b then b; else ∞ | | |
| - | a=b | a | **if a≥ b then a; else ∞** | **greater or equal** | ≥ |
| - | a=b | b | if a≥ b then b; else ∞ | | |
| - | a=b | - | **if a = b then a; else ∞** | **equal** | ≡ |
| - | - | a | **if a > b then a; else ∞** | **greater than** | > |
| - | - | b | if a > b then b; else ∞ | | |
| - | - | - | ∞ | **constant ∞** | ∞ |

# Completeness

- *Theorem*: *min*, *inc*, and *lt* are functionally complete for the set of *s-t* functions
  - *max* is also very convenient (can be implemented w/ *min* and *lt*)
- *Proof:* by construction
  - *Example:*

| $x_1$ | $x_2$ | $x_3$ | y |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 0 | ∞ | 2 |
| 2 | 2 | 0 | 2 |

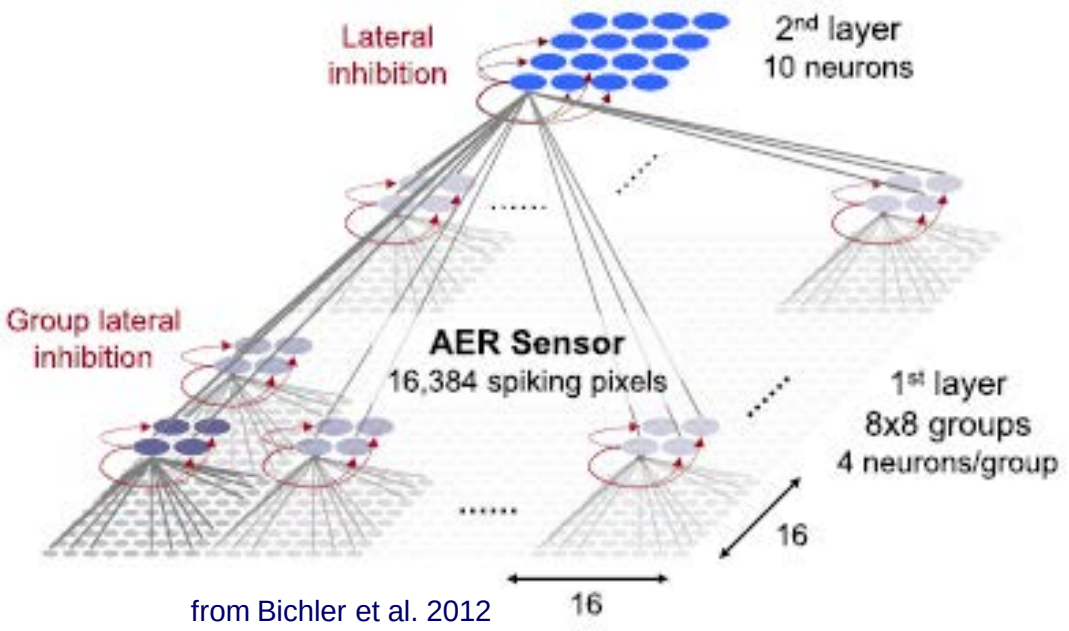



- *Corollary*: All TNN architectures can be implemented using only *min*, *inc*, and *lt*

# Implement TNNs With S-T Primitives

- Build set of functions sufficient for TNN architectures
  - Theorem says we can do it… so let's do it

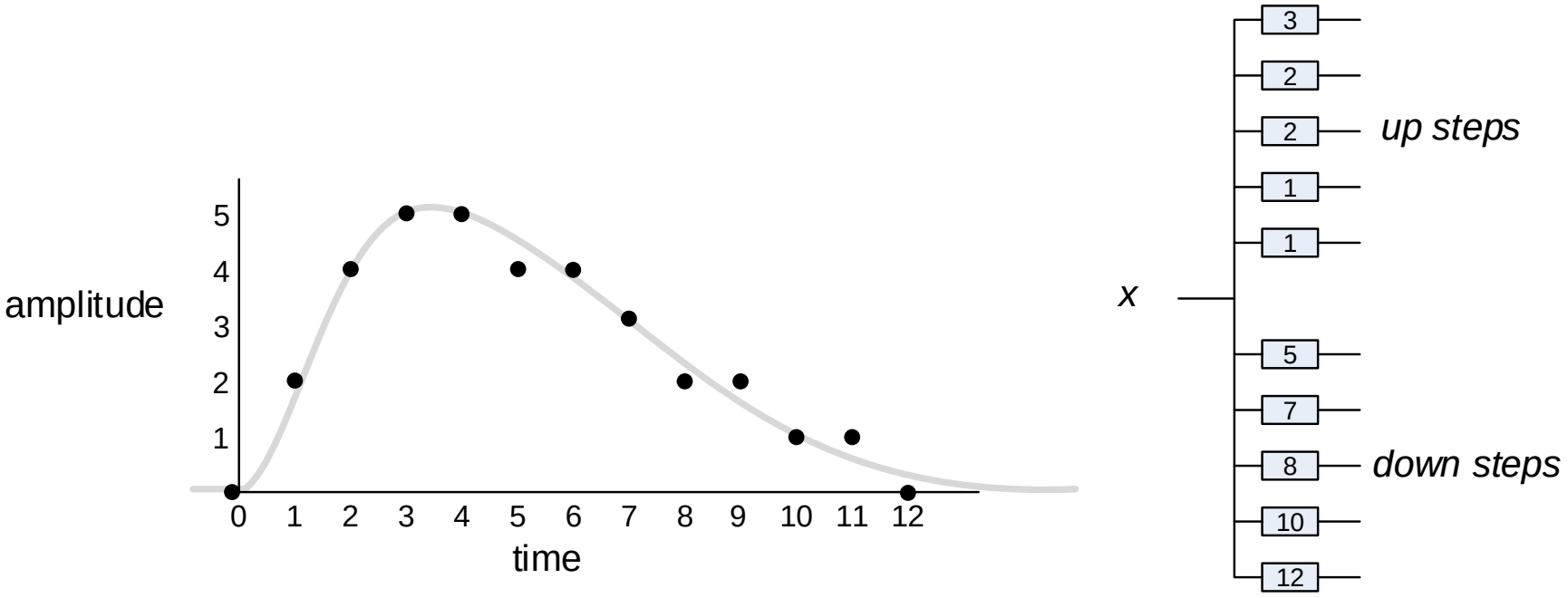
1) SRM0 neurons
2) Synaptic weights
3) WTA (lateral) inhibition




from Bichler et al. 2012

# SRM0 Neurons

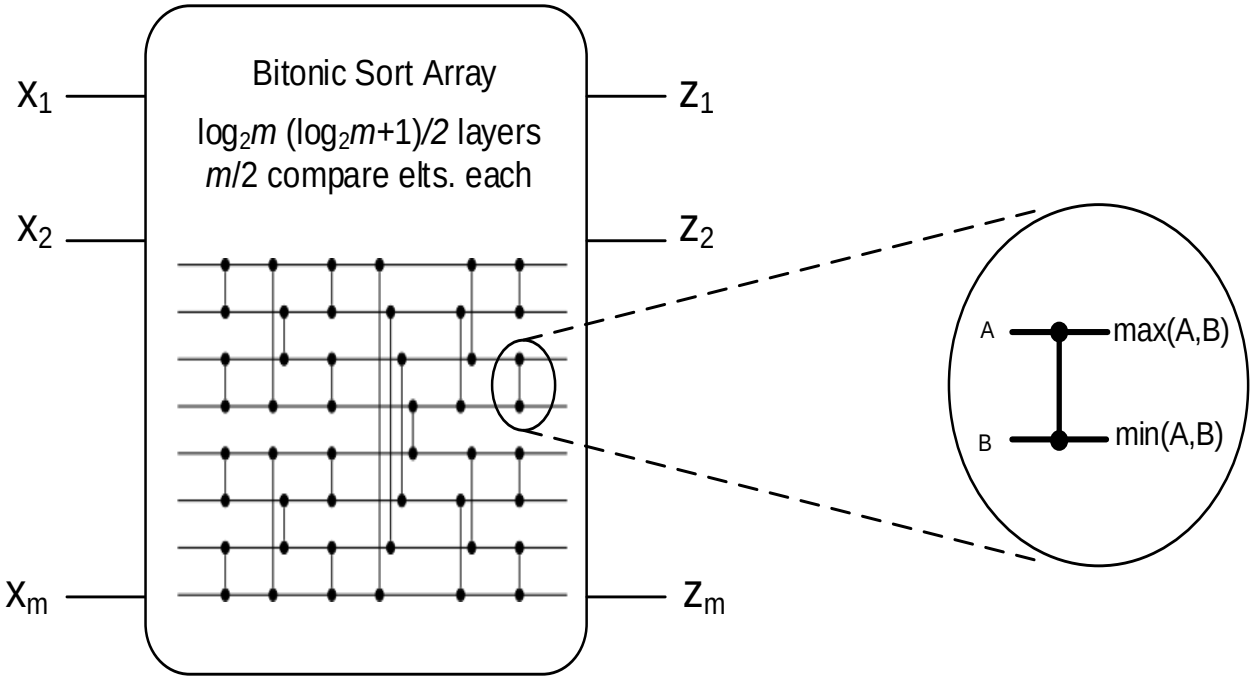
- Start with SRM0 neurons
  - A response function can be specified as up and down steps




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

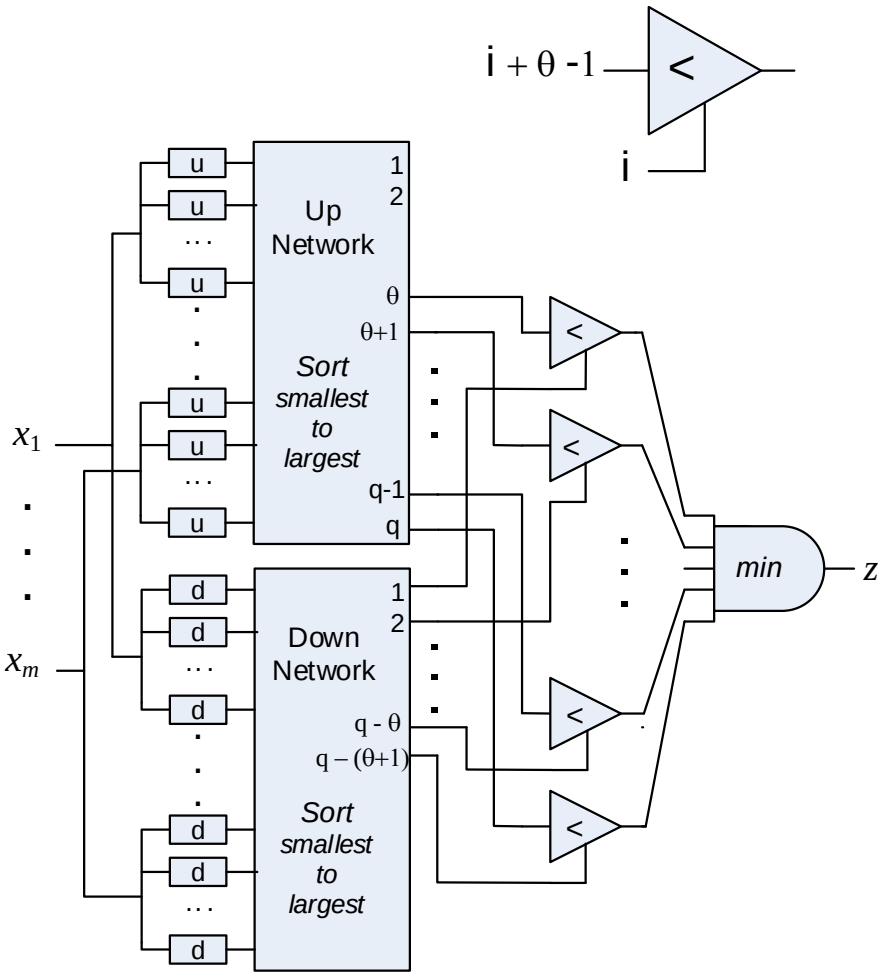
# Bitonic Sort (Batcher 1968)

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



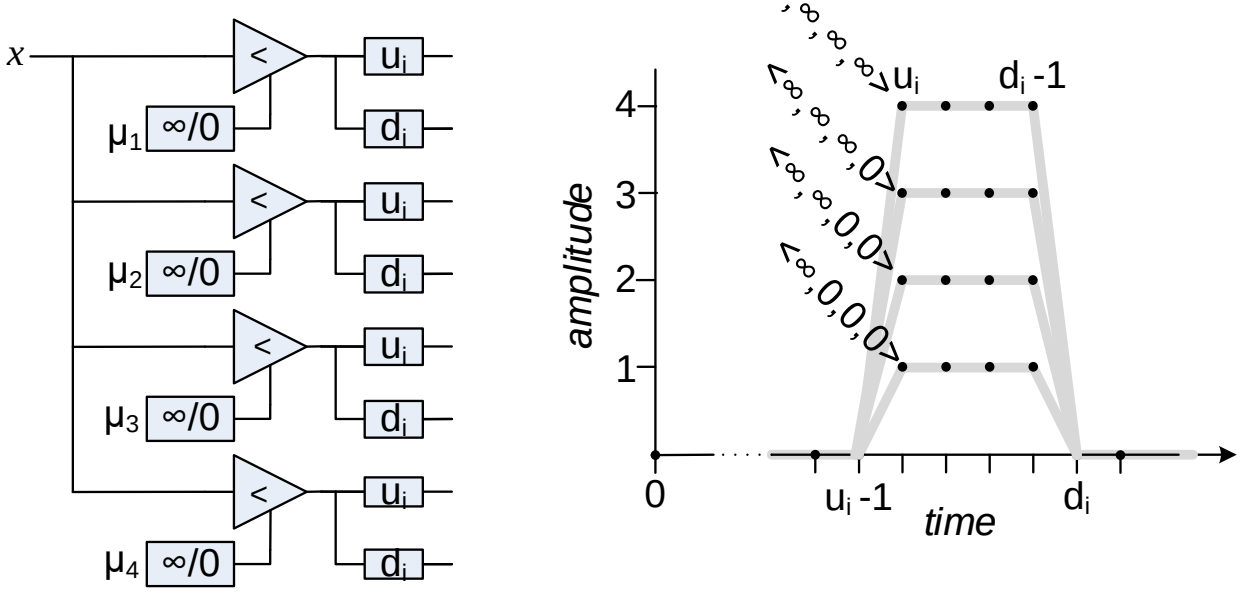
# Complete SRM0 neuron

- Up Network: sort times of all *up* steps
- Down Network: sort times of all *down* steps
- *lt*/*min* tree determines the first time *no. of up steps* > *no. of down steps* + θ
  - This is the first time the threshold is crossed
- Supports *all possible* finite response functions
  - I.e., all response functions that can be physically implemented



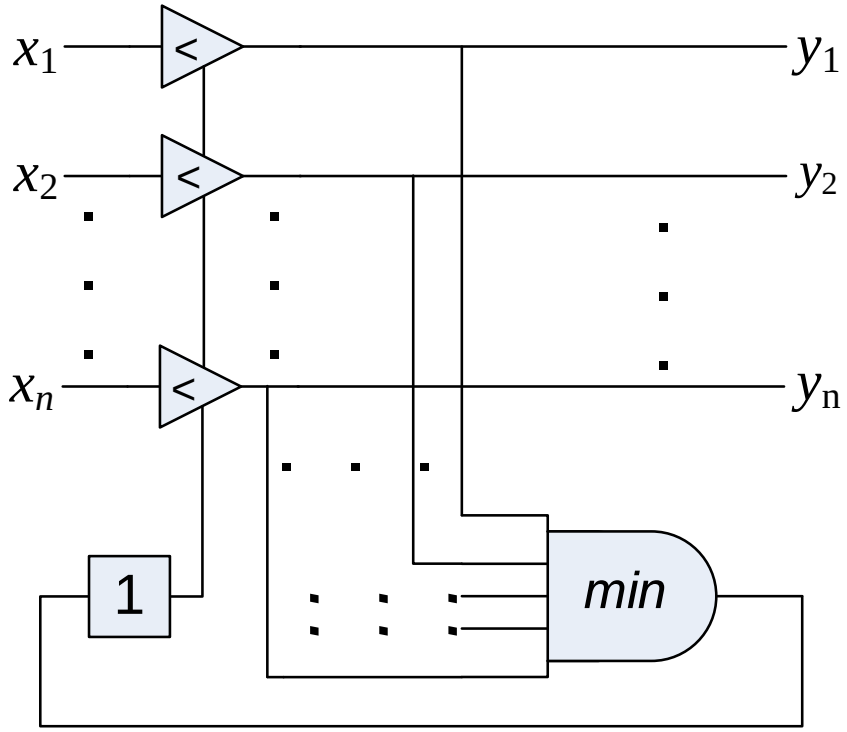
# Synaptic Weights

- Synaptic weights determine the response function
  - Synaptic weights map to ∞/0 "micro-weights" that control network composed of *on/off lt* 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

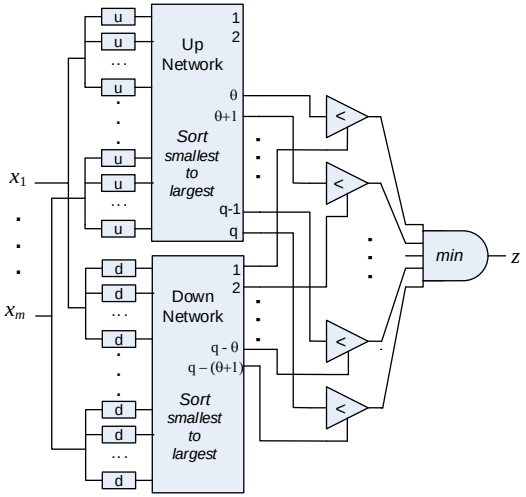


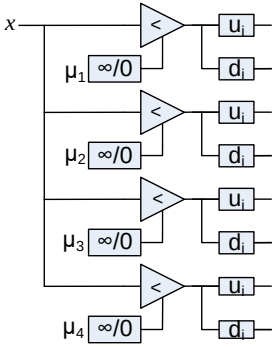

*Aside: this is an example of physical feedback, but no functional feedback*

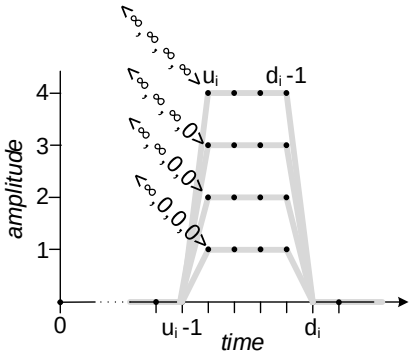
# TNN Summary

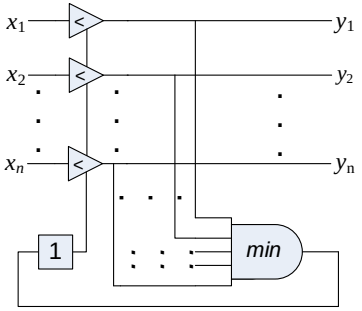
- We have implemented all the important functions

1) SRM0 neurons
2) Synaptic weights
3) WTA (lateral) inhibition



# Engineered Implementations

# Neural Networks: Platforms vs. Paradigms

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

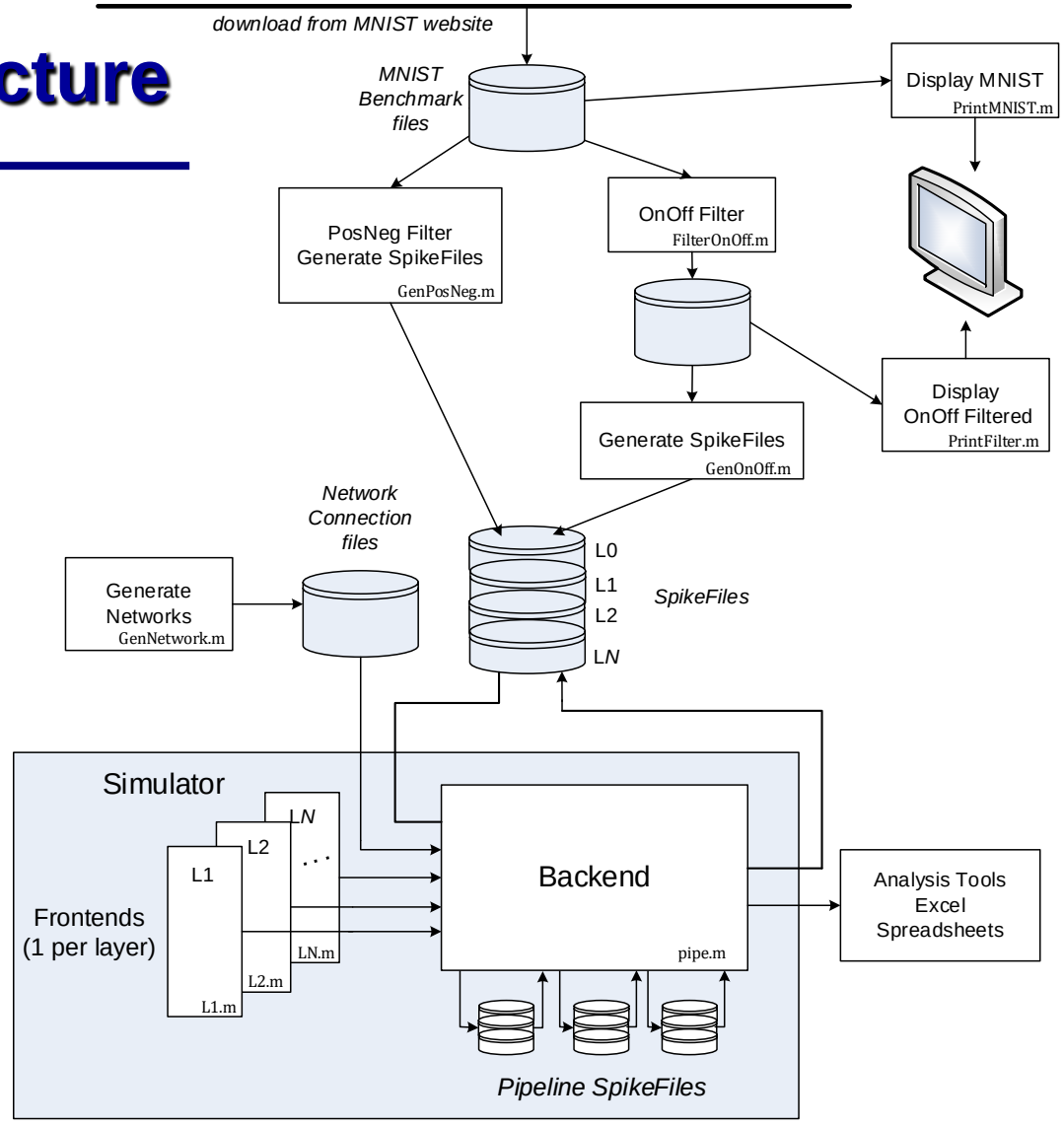
# Platforms

- General purpose indirect
  - Example: simulator written in Matlab running on a desktop
- Special purpose indirect
  - FACET
  - TrueNorth
  - SpiNNaker
  - Intel Loihi
- Special purpose direct
  - Analog neuromorphic, ala Carver Mead
  - Generalized race logic

# Simulation (Indirect Implementation)

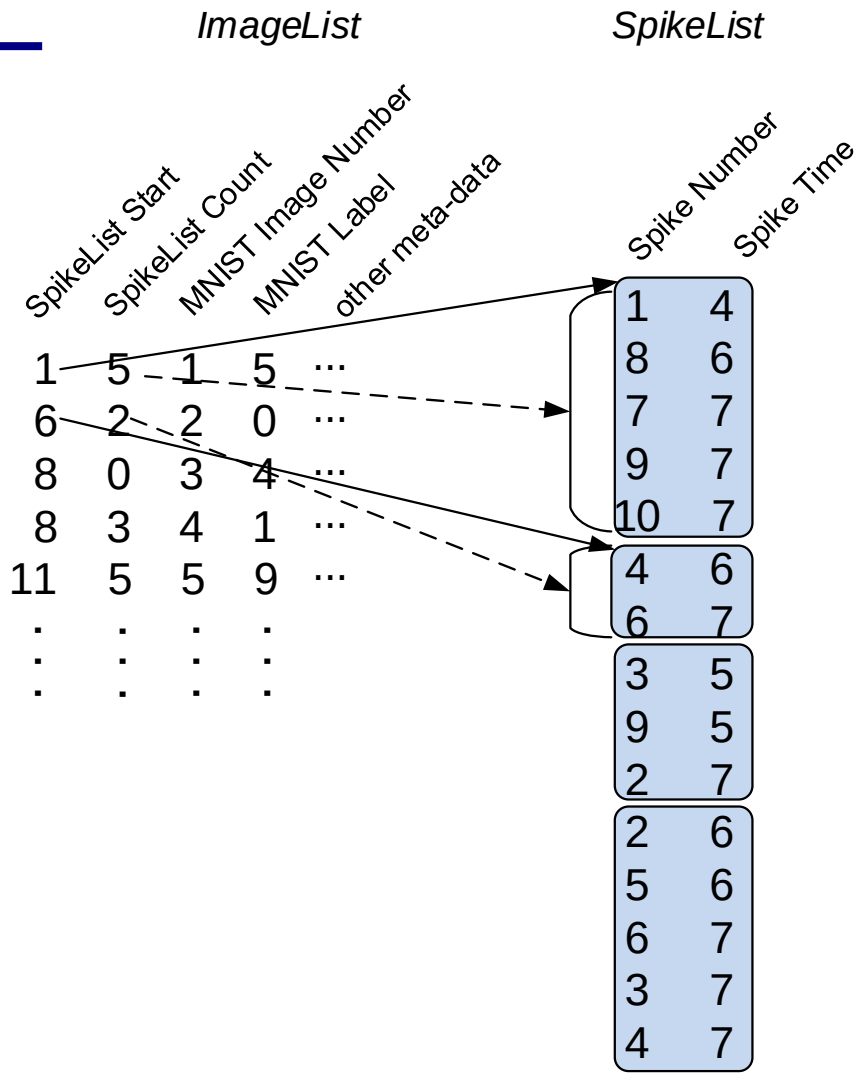
# Simulation Infrastructure

- 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 in *ad hoc* ways using Excel as a base



# SpikeFile Format

- Two data arrays
  - ImageList has one entry per image
    - pointer into SpikeList
    - label and other meta-data
  - SpikeList contains all spikes in a sparse vector format
- Use Matlab save/load to store and communicate these structures via secondary storage



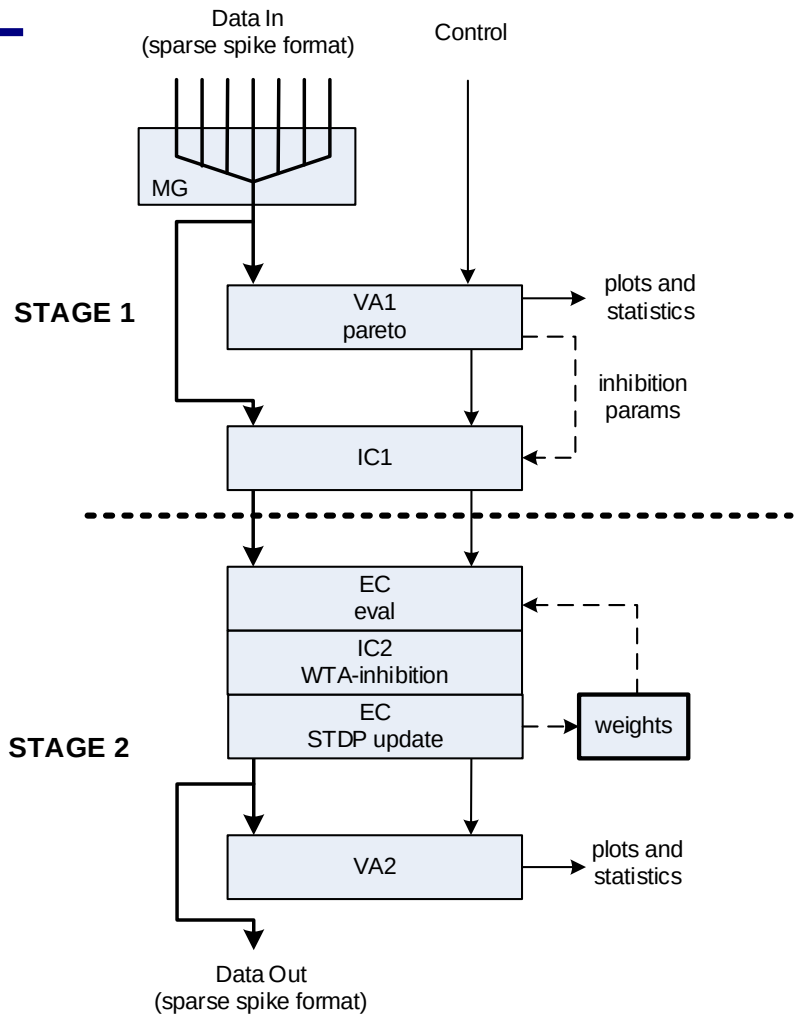
# Simulator Backend Pipeline

- Two main stages
  - 2nd larger than first
  - Files saved in between
  - Decomposes pipeline development
- Weights are the only maintained state
  - Updated dynamically via STDP
- Several secondary stages
  - MG: merge input streams
  - VA: volley analyzer; computes pareto curves for inhibition
  - IC: inhibitory column
  - EC: excitatory column

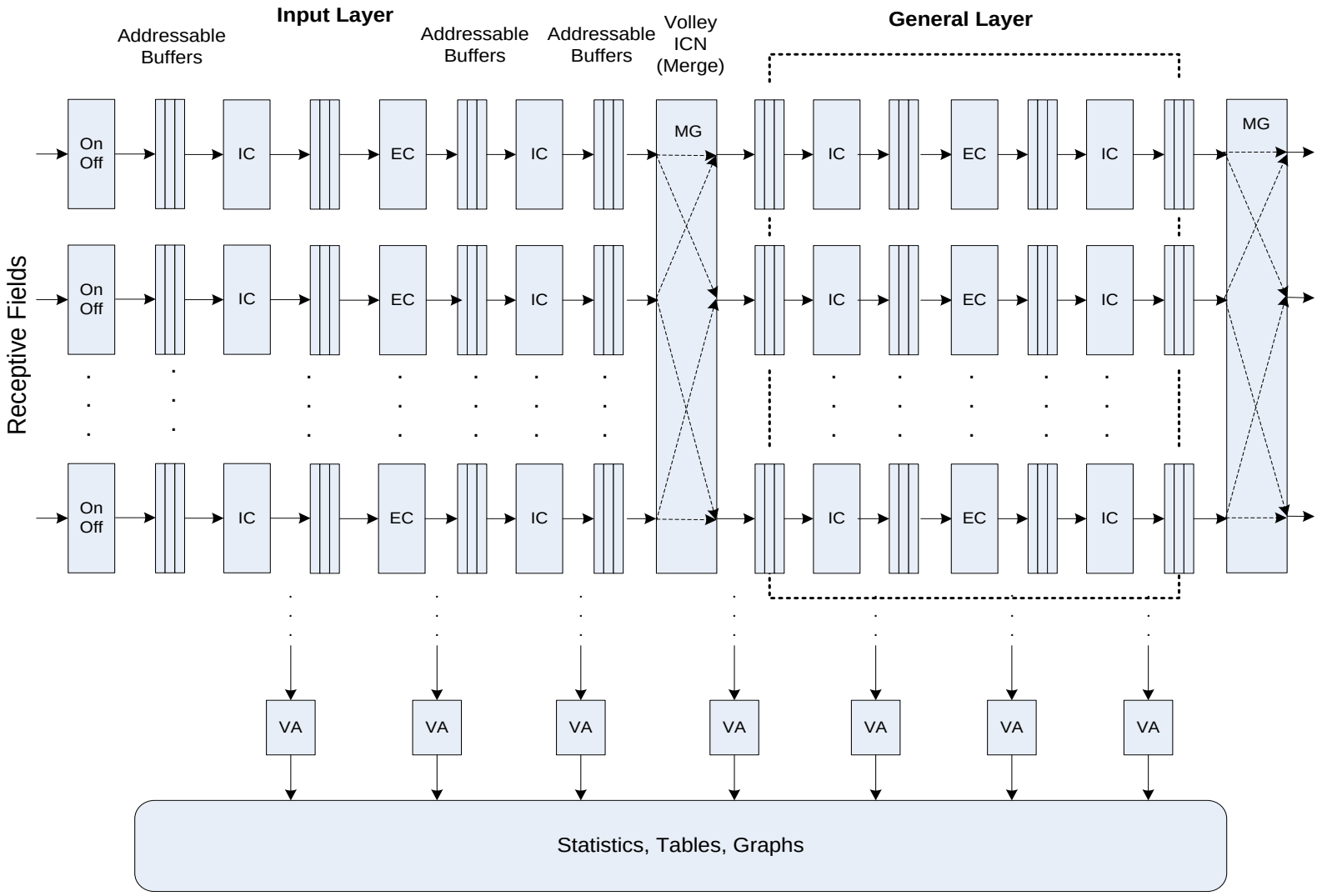
    evaluates new inputs

    applies WTA inhibition

    updates weights according to STDP

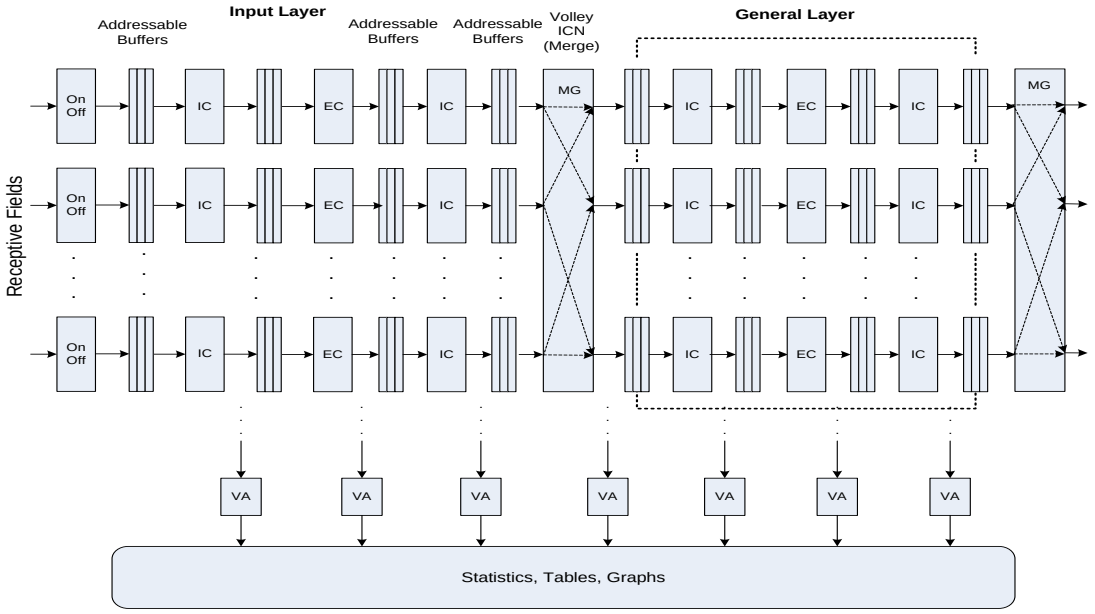


# Simulator Architecture



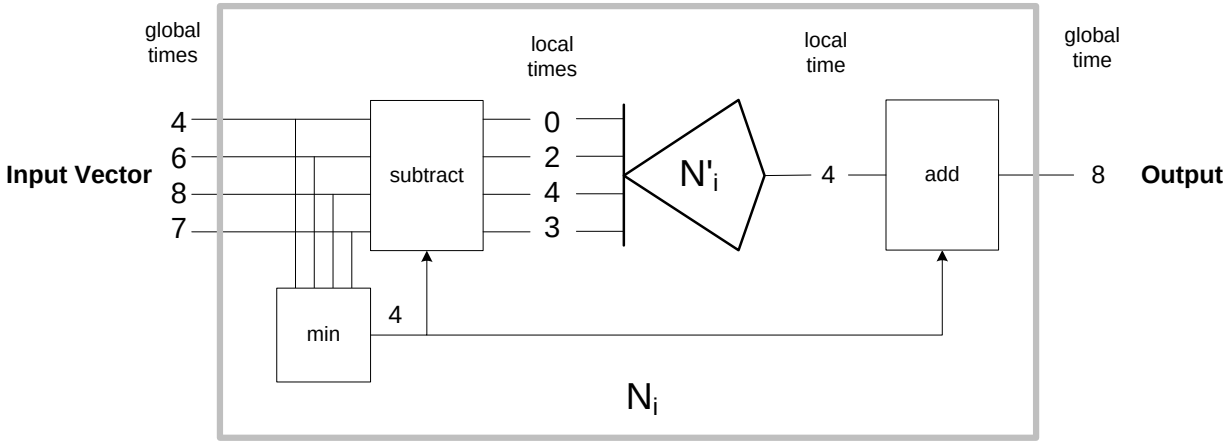
# Simulator Architecture

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



# Global Time Wrapper

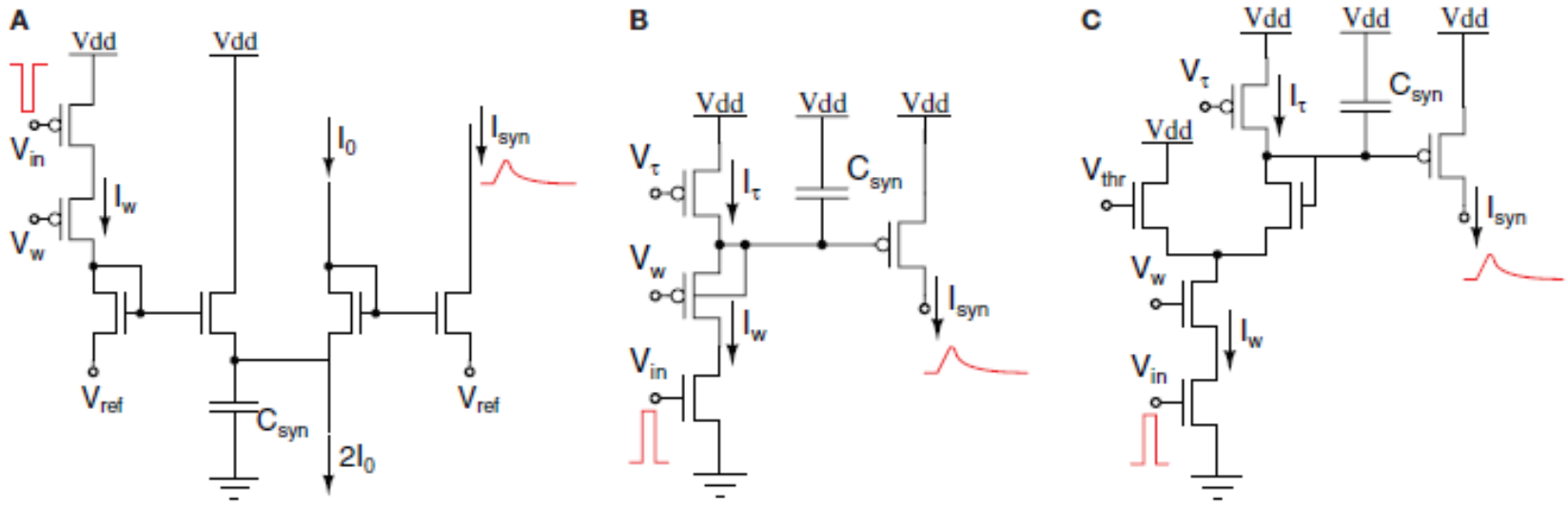
- Overall, maintain global time
- Neurons are invariant
  - Convert global to local times by subtracting normalizing value from all global times
  - Perform computation using a narrow (normalized) range of values
  - Add normalizing value back in
- When performed via simulation, GTW conversion requires computation
  - And it's a lot of computation per neuron
  - The biological neocortex gets this computation for free due to passage of physical time



# Neuromorphic (Direct) Implementations

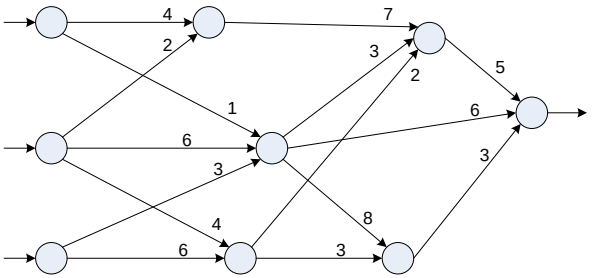
# Neuromorphic Systems

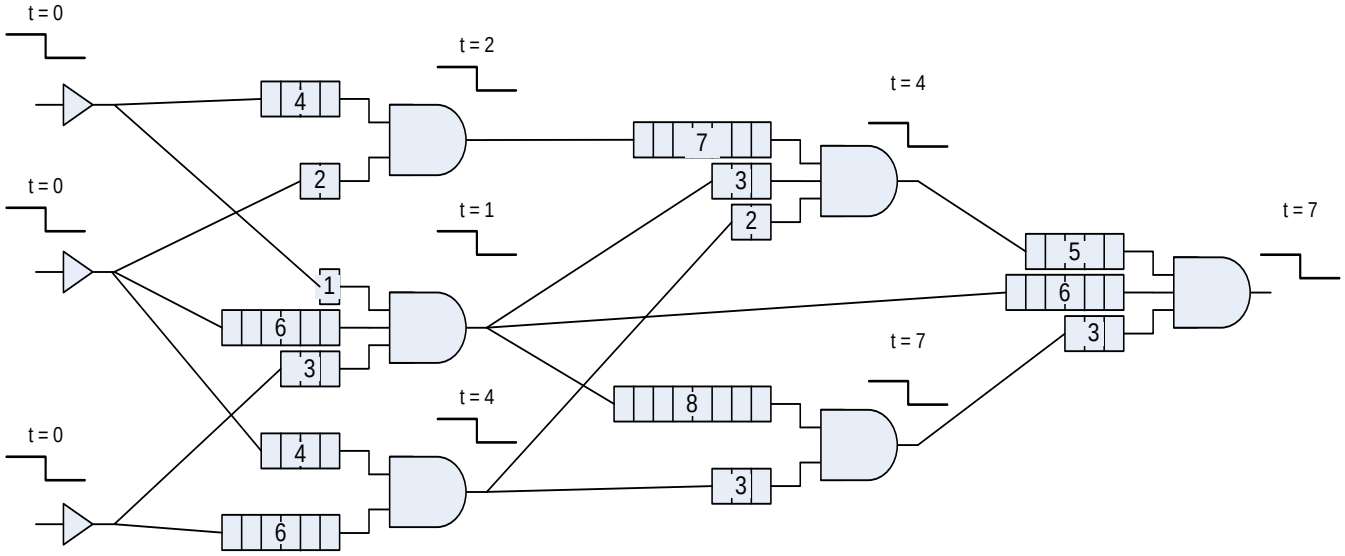
- The idea:
  - Literally build hardware that communicates via spikes in time
  - As an attempt to capture efficiency benefits of spiking neurons
- This is a *direct* implementation
  - Extend to implementations that communicate via transient *events* in time
  - *Edges* are also events in time… Generalized Race Logic



# Race Logic: Shortest Path

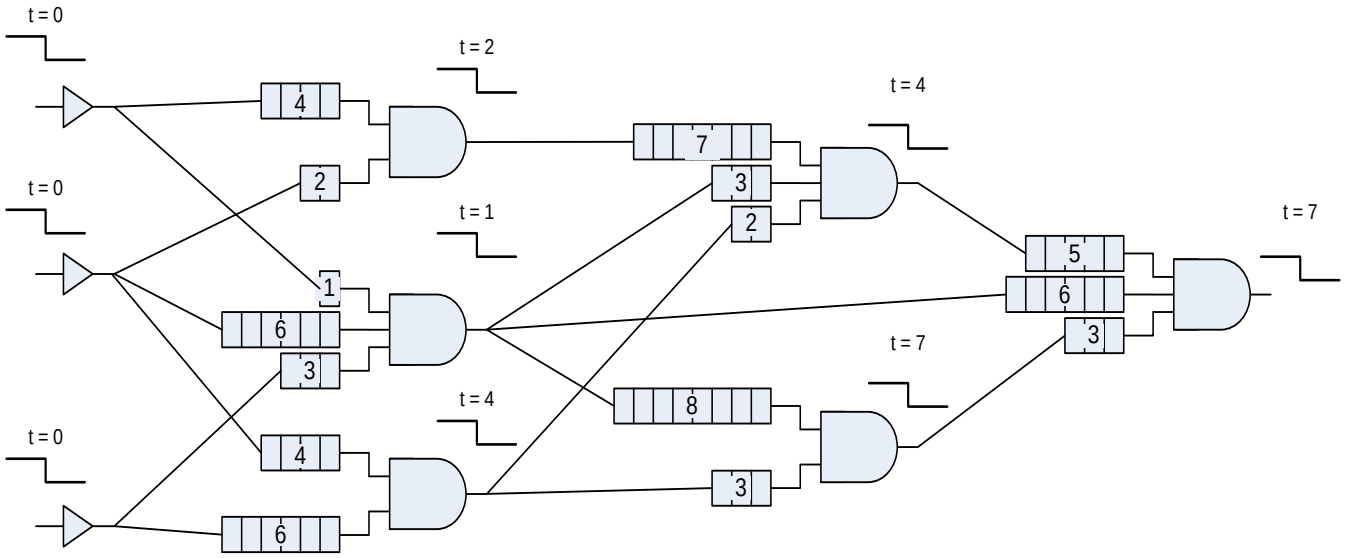
- By Madhavan et al. (Tim Sherwood group)
- Shortest path computing network
  - *Weights* implemented as shift registers
  - *Min* functions implemented as AND gates



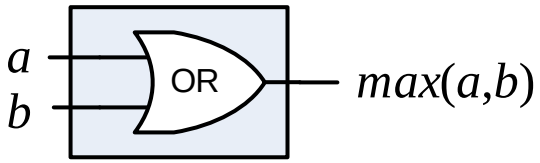
# Network Operation

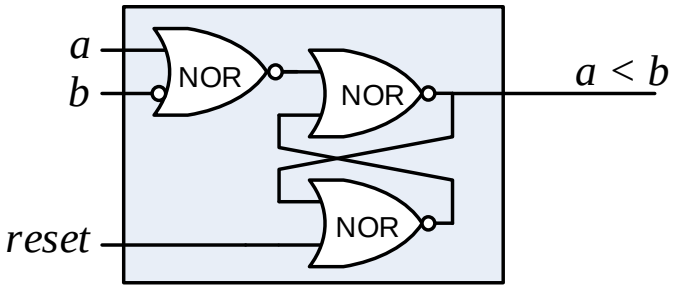
- Initialize all signals to 1
- At time *t*=0, transition 1→ 0 on all primary inputs
- Transitions sweep forward in network
  - Transition times are values
- The time it takes to compute the short path *is* the short path

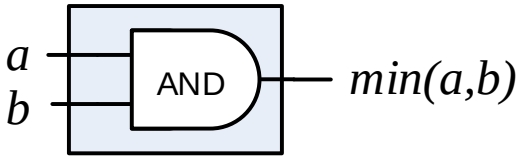


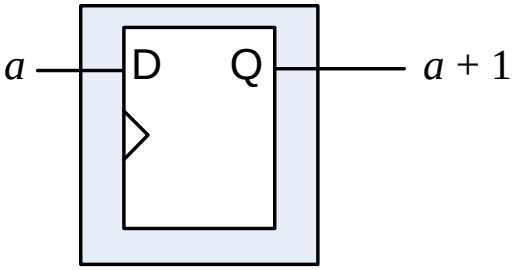
# Generalized Race Logic

- Primitives implemented w/ digital circuits
  - Signal via 1 → 0 transitions



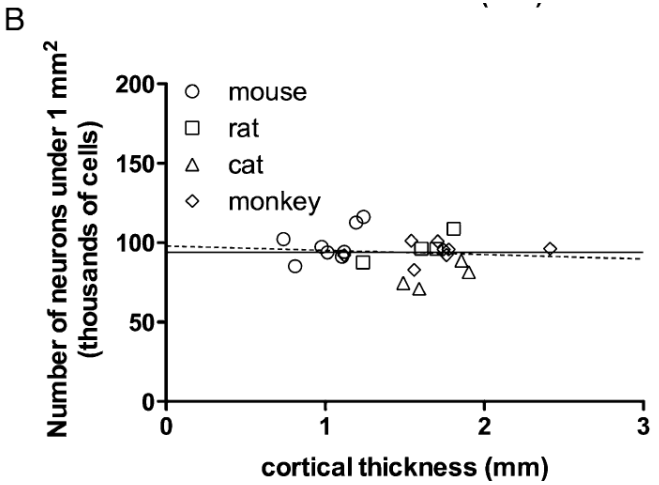




- Exercise: Build an SRM0 neuron using generalized race logic

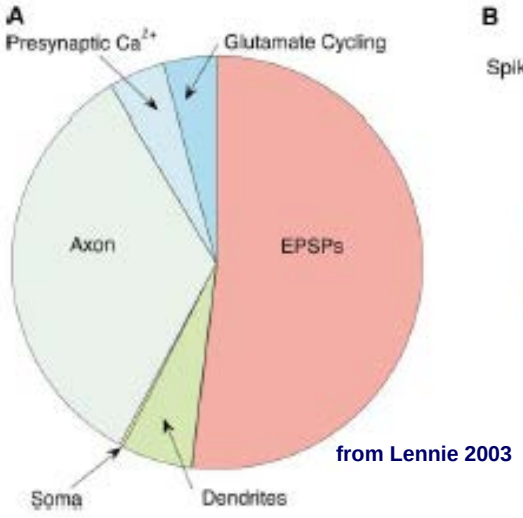
*Implication: Construct architectures in the neuroscience domain*
*Implement directly with off-the-shelf CMOS*

# Compare with Silicon

- Both are “surface” technologies
- Components per $mm^2$
  - $10^5$ neurons ($10^8$ active synapses)
  - $10^7$ transistors
- Watts per $mm^2$
  - $10^{-4}$ for neurons
  - 1 for transistors
- Cycle time
  - $10^{-2}$ sec for neurons
  - $10^{-9}$ sec for transistors

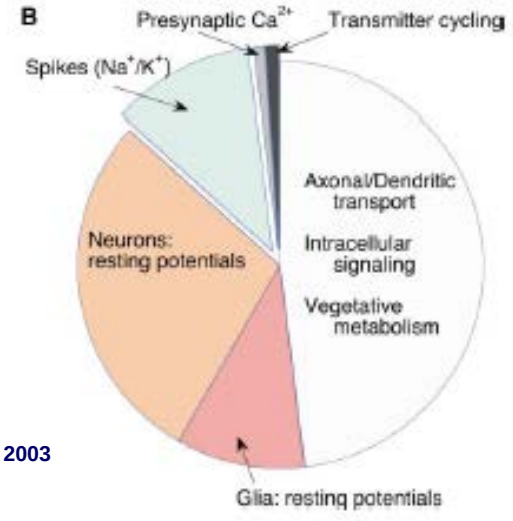



from Carlo and Stevens 2013




from Lennie 2003

**A: energy expenditure of a single spike**

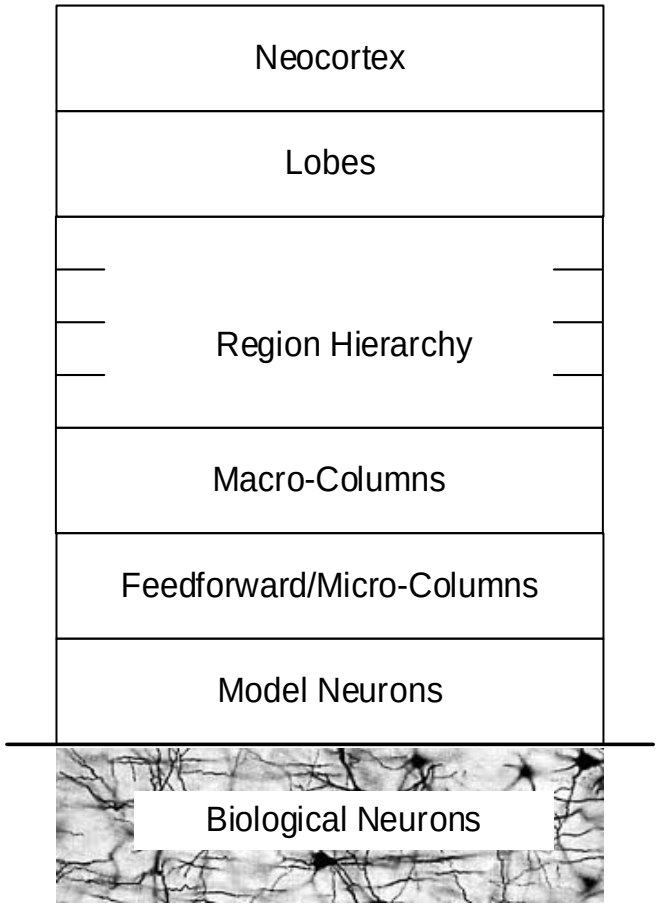



**B: Total energy expenditure**

# Concluding Remarks

# Research Directions

- Huge research space
  - Response functions
  - Computation in dendrites
  - STDP update functions
  - Connectivity patterns
  - Sensory interfaces
  - Hippocampus and memory
- Study *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 in function implementations
  - *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 for *s-t* functions only
  - There is no inversion, complementation, or negation
- We strive for algorithmic, top-down design
  - TNNs are based on *implied functions*
  - Bottom-up, trial-and-error design methodology

**Think outside the box!**

# Acknowledgements


- *Impetus and encouragement*: Raquel, Mikko Lipasti, Mark Hill, Margaret Martonosi
- *Technical*: Cristobal Camarero, Mario Nemirovsky, Mikko Lipasti, Tim Sherwood, Advait Madhavan, Shlomo Weiss, Ido Guy, Ravi Nair, Abhishek Bhattacharjee

# Bibliography


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.

Batcher, K. E., "Sorting networks and their applications," *Proceedings of the Spring Joint Computer Conference,* pp. 307-314, 1968.

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, 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, S. M., H. La Poutré, 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, S. M., J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing* 48, no. 1 pp.17-37, 2002.

Brette, R., "Philosophy of the spike: rate-based vs. spike-based theories of the brain," *Frontiers in Systems Neuroscience* 9, 2015.

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.

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, 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, 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.

# Bibliography


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

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

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

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

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

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.

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.

# Bibliography


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, 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, 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.

LeCun, Y., Y. Bengio, and G. Hinton, “Deep learning,” *Nature* 521, no. 7553, pp. 436-444, 2015.

Lippmann, Richard, “An introduction to computing with neural nets” , *ASSP Magazine,* pp. 4-22, 1987.

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, W., “On the computational complexity of networks of spiking neurons” (extended abstract), *Advances in Neural Information Processing Systems*, vol. 7, p. 183, 1995.

Maass, W., “Networks of spiking neurons: the third generation of neural network models,” *Neural Networks* 10.9, pp.1659-1671, 1997.

Mainen, Z. F., and T. J. Sejnowski, "Reliability of spike timing in neocortical neurons,” *Science* 268, no. 5216 , pp. 1503-1506, 1995.

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.

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

# Bibliography


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

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

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

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

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.

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, 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.

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

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

# Bibliography


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.

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

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

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

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

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

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

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