

Brains, artificial neural networks, and some future hardware issues

Ralph Linsker
IBM Research
Yorktown Heights, NY
linsker@us.ibm.com

Outline

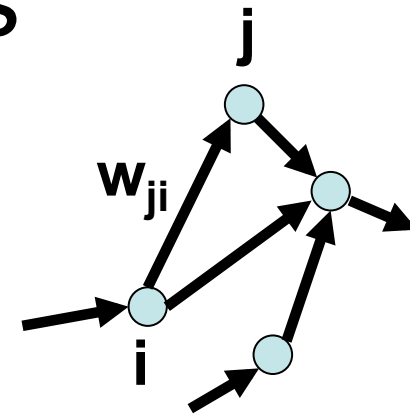
- Why does HW acceleration for artificial neural networks (NNs) matter?
 - Many NN applications, growing in importance
 - Better “deep learning” algorithms are effective for more complex problems using larger NNs; → computationally intensive
- What makes NN acceleration feasible?
 - NN algorithms are highly parallel; NNs use a set of relatively simple core functions; communication & memory tend to dominate
 - Opportunities for analog or A/D hybrid implementations
- What are NNs, and how do they work?
 - What core functions do they require?
 - → Opportunities for future device & circuit designs, & new chip architectures
- Lessons from biological brains
 - What additional functions will likely become important, in more advanced NN architectures?
- Conclusions

Some neural network applications

- **Pattern recognition & classification**
 - Speech, **face**, object, **scene**, handwriting, gestures, radar, geologic data
 - Loan approval, real estate valuation, credit card fraud, email spam
 - Medical: image analysis, EKG, EEG
 - Drugs & explosives detection, chemical analysis
 - **Image and audio search, surveillance**
- **Data processing**
 - “Blind” signal separation; compression; clustering
 - **Classification & analysis of massive datasets (e.g., documents, images, voice)**
 - **Video game AI**
- **Prediction, control, and system identification**
 - **Business analytics, financial apps, data mining**
 - **Analysis and control of large complex networks (e.g., power, transportation)**
 - Vehicle control
 - Chemical and manufacturing process control
 - Time series prediction, modeling
 - Medical diagnosis
 - **Robotics**
- **Complex optimization**
 - **Chemical and drug design**
 - Variable pricing algorithms, scheduling, marketing

Neural networks

- Network of “neurons” and weighted connections
- Conventional (“2nd generation”) “sigmoid” NNs



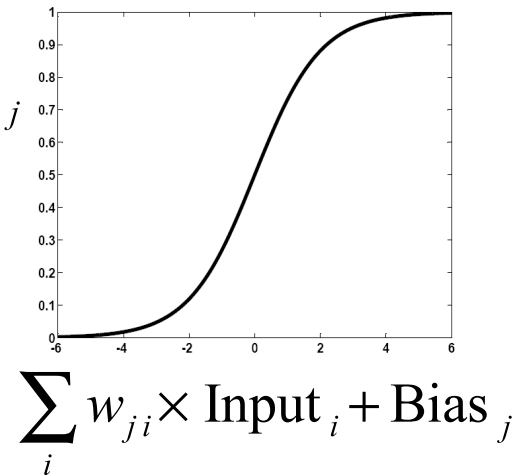
- **Activation:** At each time step, each node’s output is a function of its weighted inputs
- **Learning:** Each weight w is changed by a function of the activities at i and j
 - E.g., “Hebb rule” rewards correlations

$$\Delta w_{ji} \sim \text{Output}_j \times \text{Input}_i$$

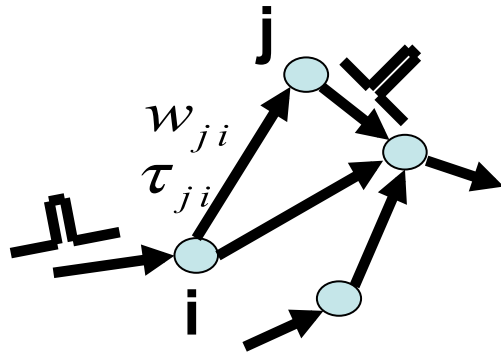
- **Powerful higher-level learning algorithms** exist; they build on Hebb and other basic update rules

Output_{*j*}

Activation rule

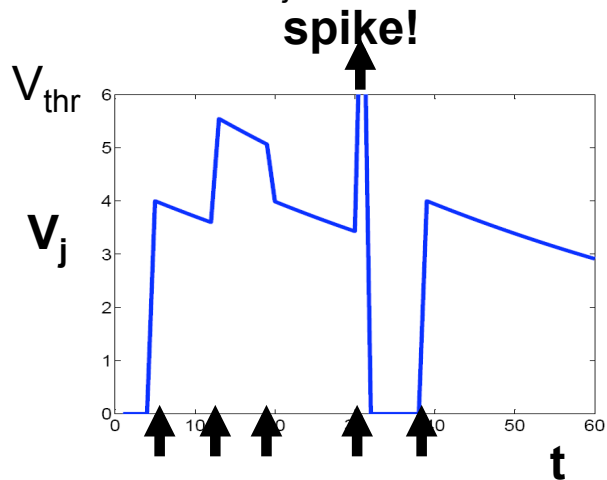


(1) “Spiking” (“3rd gen.”) NN

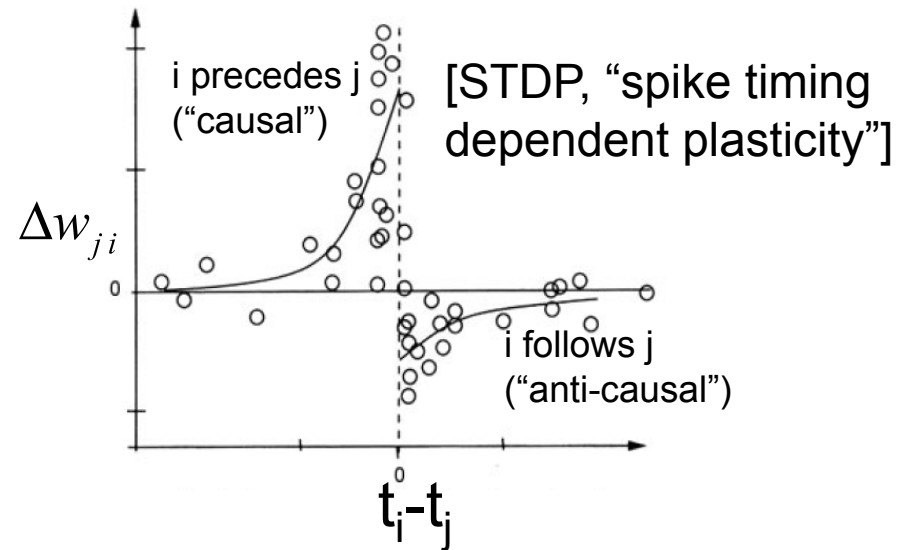


(2) Activation:

- “Voltage” V_j changes by w_{ji} when spike arrives from i ;
- When $V_j > V_{thr}$, node j emits a spike, and V_j resets to 0;
- Otherwise, V_j decays toward 0.



(3) Learning: w_{ji} changes as function of time interval between spike arrival from i and spike emission by j



(4) Challenge: Higher-level learning rules are not yet mature; thus *spiking* NNs are not yet commercially significant.

But: **Power and bandwidth advantages** over 2nd-gen NNs – in HW and in brains!

Example demo

- Learning to recognize & generate handwritten digits [work by G Hinton et al.]
- A “2nd-gen.” four-layer NN with a new “higher-level” learning rule
 - <http://tinyurl.com/mrb3qa>
- Network of ~4000 neurons & 1.7M conn’s trained on a set of 50K images, shown a few hundred times.
- Movies (showing *activation, not learning*):
 - <http://tinyurl.com/klyqcj>

- For examples of correctly recognized handwritten digits that the neural network had never seen before (from G Hinton), see:
 - <http://www.cs.toronto.edu/~hinton/talks/gentle.ppt#61>

Spiking-NN HW chip sizing

- Use an “address event representation” (AER)
- Notation: “n” = “neuron”; “ts” = “simulation time step” (~1 msec for brain modeling)
- **Parameters**
 - b bits sent for each transmitted spike
 - f ~ average prob’y that a neuron fires during a given time step (large fluct’ns!)
 - K ~ typical fan-in or fan-out of each neuron
 - c bits stored / connection (for weights, time delays, addresses, other information)
 - p_1 ops/n/ts ; plus p_2 ops / received spike ; plus p_3 ops / spike-pair during learning
 - L = width (in ts) of “learning window” that defines a received & a generated spike as forming a “pair”
- **Communication bandwidth** (bits/n/ts) ~ **b K f**
 - Assume or estimate the fraction of spikes that travel off-chip; apply appropriate constraints to both on- and off-chip BW
- **Computational load** (ops/n/ts) ~ $p_1 + p_2 K f + p_3 K f^2 L$
 - Typically undemanding; favors mapping many “neurons” to each HW processing unit
- **Memory** (bits/n) ~ **c K** + {other registers as needed}
- Typically (in current or near-term designs): **Memory area will constrain N, the # of neurons/chip (or $N \cdot K$, the # of connections/chip)**
- **Then BW (on- and/or off-chip) & power will constrain the NN’s performance (in ts/sec)**
- **Optimize degree of parallelism** (i.e., # of simulated neurons per HW processing unit), I/O handling, N, etc., to balance computation, communication, and memory demands, improve performance, etc.

Core NN operations for “sigmoid” NNs

- **Multiply/adds** (sum over conn's, of weights * input activities; sum over input presentations, of input * output activities)
 - Old idea for analog NN activation: crosspoint network; weights at intersections; weight=conductance; input activity=voltage; Kirchoff sum of currents
 - Nominally “zero” conductances must be kept extremely small, if all-to-all crosspoint connectivity!
 - More compact crosspoint geometry for nanoscale connections among microscale “neurons”: “CMOL” (Likharev et al.); see <http://tinyurl.com/m2ywaw>
- **Nonlinear (“sigmoid”) function** at each neuron
- [Random generator if want to stochastically binarize output activities]
- **Send continuous-valued (or binarized) activity along each conn'n at each time step**

Core NN operations for spiking NNs

- Activation:
 - Spike arrival \rightarrow weighted change in neuronal “voltage” V
 - Time decay of V
 - Spike emission & V reset when V exceeds threshold
- Learning:
 - Identify spike-pairs; change weight according to ordering & inter-spike interval
- Ideas for using nanodevice connections (e.g., phase change memory elements) with controlled input signal sequences for changing conductance (e.g., Ovshinsky)
- Send spike signals or digital packets along conn’s, only when needed. Implement specified time delays.

A look at biological brains – for inspiration and challenges

- **The brain’s “computational style”:** How is information inputted, stored, transformed, communicated, organized, and integrated?
 - Very differently from present-day computers (serial or parallel), in almost all respects!
- **Some (very rough!) numbers:**
 - Cerebral cortex (human): $\sim 2E10$ neurons; $\sim 1E14$ synapses; thickness ~ 1.5 - 4.5 mm; surface area ~ 0.25 m²; $\sim 2.5E4$ neurons/mm³ ($\sim 1E5$ in visual cortex).
 - In a 50-micron (on a side) “minicolumn” – a type of “functional unit”: ~ 40 neurons; ~ 100 - 200 in visual cortex.
 - $\sim 1E9$ synapses/mm³ (across mammals)
 - Spike generation rate by a neuron: zero to hundreds/sec
- The brain’s complex and multi-scale dynamics, and its architecture, may provide important hints for designing much more powerful NNs!

Complex neuronal and network dynamics

- Intra-neuronal dynamics – affect spiking behavior on time scale of <1 to 100s of msec
- Multiple **time scales** for synaptic changes (**slow and fast learning**)
- Multiple **spiking modes** (e.g., bursting vs. individual spikes)
- Multiple **network modes** (e.g., wake & sleep)

- Many types of **network oscillations** (from few Hz to >100 Hz)
- **Synchronization or phase-locking** transiently “associates” widely-separated parts of network:
 - A way to “bind” activities evoked by different features of an object, so that we **perceive an object as a coherent whole**
 - Also can “bind” activity across processing areas, enabling **selective focussed attention** (heightened responsiveness)

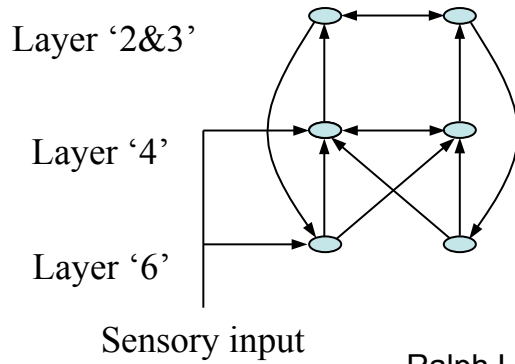
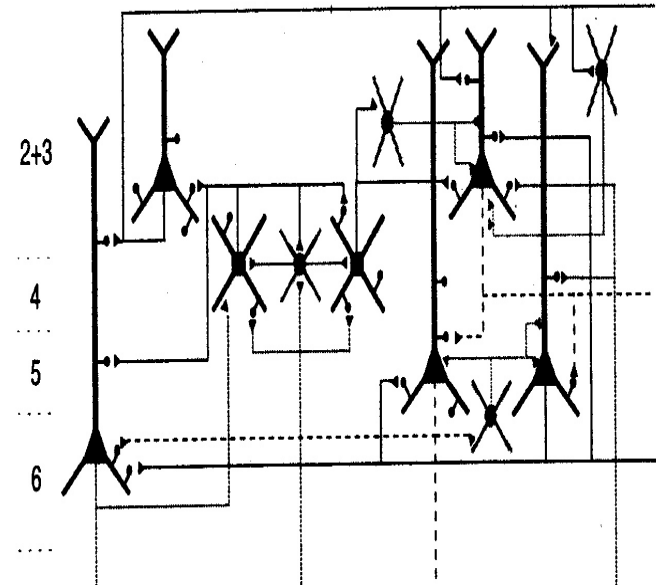
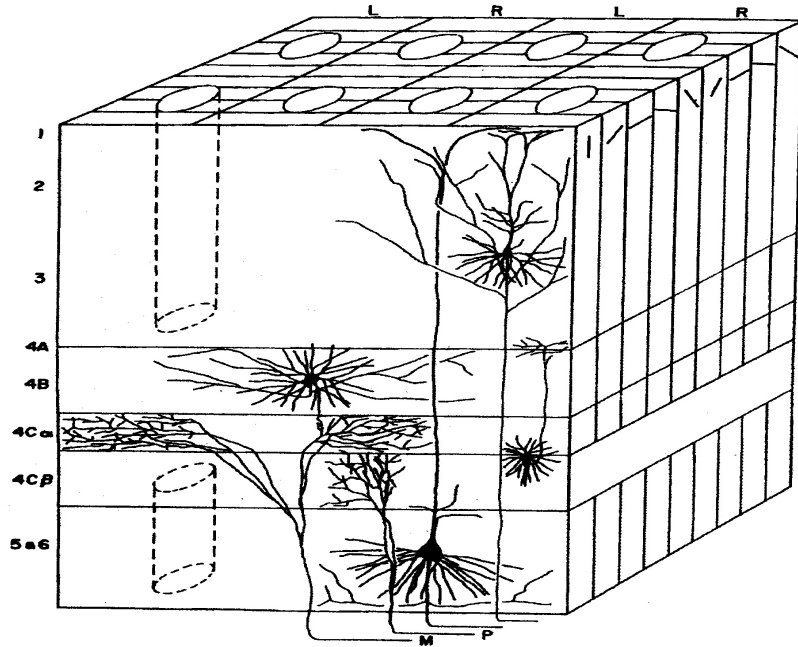
- **Recurrent activation** of circuits in “working” (short-term) memory and in hippocampus (learning temporal sequences)
- Re-entrant loops (e.g., Edelman) that integrate proc’g across brain regions (external sensory, internal sensory, motor, decision-making). Also perhaps responsible for continuously updated sense-of-self (Damasio)?

Brain architecture – complexity with order

- **“Vertical” org’n** of six-layer cortex: the **“local cortical circuit” (LCC)**
 - A perhaps-ubiquitous functional “module”? [See fig.]
 - Conjectured core functions: **prediction, fill-in of missing/noisy data, feature discovery**
 - Relation to **Bayesian inference**? I.e., does it learn a model that best captures the statistical relationships among an ensemble of inputs?
 - Many recent papers on Bayesian inference in NNs
 - A special case of Bayesian inference: Kalman prediction & control in a NN (R Linsker, Neural Networks, 2008 – also gives many Bayesian refs.)
- **“Horizontal” organization**: Structure within **2-d functional “maps”** (e.g., representing position, motion, & edge-orientation of parts of a visual scene)
- **Interactions**
 - Among maps of a given system (e.g., vision): cabling, re-entry (feed-forward and feedback), synchrony. [See fig.]
 - Among maps of different systems (vision, hearing, motor output, decision-making, etc.)
 - Between cerebral cortex and other brain regions (hippocampus, amygdala, thalamus, etc.)

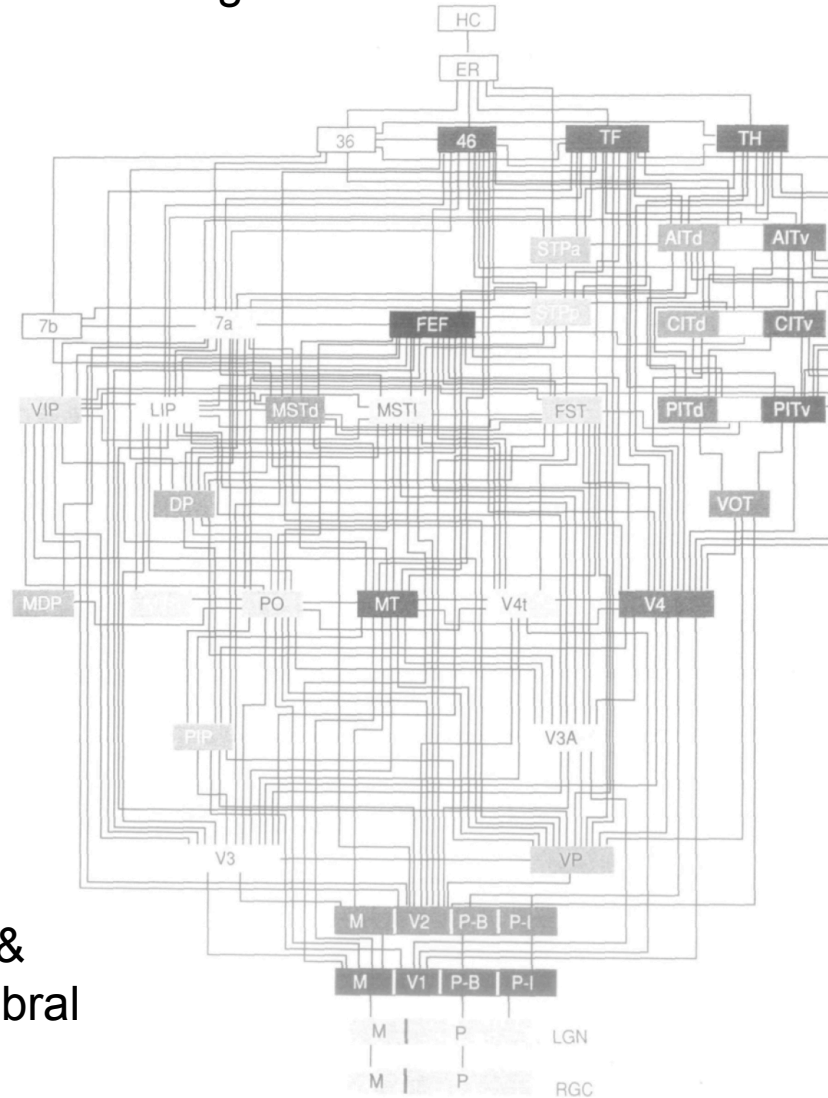
Anatomy of “local cortical circuit” (LCC)

[Top two figures from: GM Shepherd, *The Synaptic Organization of the Brain*, 4th edn., pp. 476-477 (1998)]



Schematic of LCC
 [One “minicolumn” ~ 100 neurons
 ~ cylinder of 50-micron diameter;
 ~100M similar modules?]

Interactions among functional areas of the visual system



[From: DJ Felleman &
DC Van Essen, Cerebral
Cortex 1: 30 (1991)]

Figure 4. Hierarchy of visual areas. This hierarchy shows 32 visual cortical areas, shaded according to the same scheme as in Figure 2, 2 subcortical visual stages (the retinal ganglion cell layer and the LGN), plus several nonvisual areas (area 7b of somatosensory cortex, perirhinal area 36, the ER, and the hippocampal complex). These areas are connected by 187 linkages, most of which have been demonstrated to be reciprocal pathways.

Conclusions: Future of hardware NNs

- Is the goal to “mimic the brain”?
 - No – we don’t know how! And the notion isn’t well-defined.
- Rather, it is to develop HW that (we hope) will efficiently run the best NN algorithms of ~5+ years from now. My bet is that these will use:
 - **Multistage hierarchical networks** of simple “neurons” with feedforward, feedback, & lateral connections;
 - **Additional control structures** to direct processing flows (as in modern robotics);
 - Communication of information via (a) neuronal state $s_i(t)$, (b) spike-time coding, and/or (c) higher-level representations (e.g., summary outputs of a multi-neuron ensemble);
 - Learning algorithms related to **Bayesian inference** & using local (esp. Hebb-type) rules;
 - Additional learning alg’ms that facilitate “**fast**” (one- or several-shot) learning; and
 - Algorithms that exploit more advanced aspects of **brain-like temporal dynamics & architecture.** →

Conclusions [2]

- Special-purpose NN HW vs. NN SW on general-purpose (super)computers?
 - What figures of merit?
 - Cost, speed
 - Compactness and low-power: for portable app's
 - Ease of algorithm development / modifications
 - General-purpose computer + NN HW accelerator
 - Note: GPGPUs (“general-purpose” graphics processing units) may become an important player in NN simulation!
- At device and circuit level: Opportunities for implementing core NN operations more efficiently using novel device properties and “weight” modification techniques.
- Some “NN HW” may have broader (non-NN) applications
 - E.g., to other AER applications →

Conclusions [3]

- Our understanding of the brain's "computational style" is evolving
 - Leading to **new NN algorithms**, and new ideas as to **organizing principles** for biological brains and NNs
 - An old & fruitful example – the "infomax" principle (R Linsker, IEEE Computer, March 1988). Important but unexplained experimental vision results (from 1960) → a NN model → an optimization principle → {application to other neuroscience problems; new NN learning rules; and a new method for "blind separation" of signal mixtures into their independent components}
 - Another example: STDP learning rules.
- It is overly simplistic to try to compare the brain's "computational power" to a computer's raw speed (as in: "*human brain ~ 1E14 connections * 1E3 ops/conn/sec ~ 100 petaFLOPS*")
 - If we don't understand the brain's organizing principles well enough, the raw performance won't yield a "brain-like" computer.
 - The power lies in inferring correct organizing principles, and discovering the brain's "algorithms" – or, better, inventing **new algorithms that capture the essential features of brain "computation" and are well-suited to the HW that will run them.**
 - For a given set of tasks & algorithms (& their implementation), we can *then* assess how much computing power is needed to execute them, and what level of brain functioning they correspond to!