The Rebirth of Neural Networks

Olivier Temam

▶ To cite this version:

Olivier Temam. The Rebirth of Neural Networks. ACM. International Symposium on Computer Architecture, Jun 2010, Saint Malo, France. 2010. inria-00535554

HAL Id: inria-00535554

https://inria.hal.science/inria-00535554

Submitted on 15 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

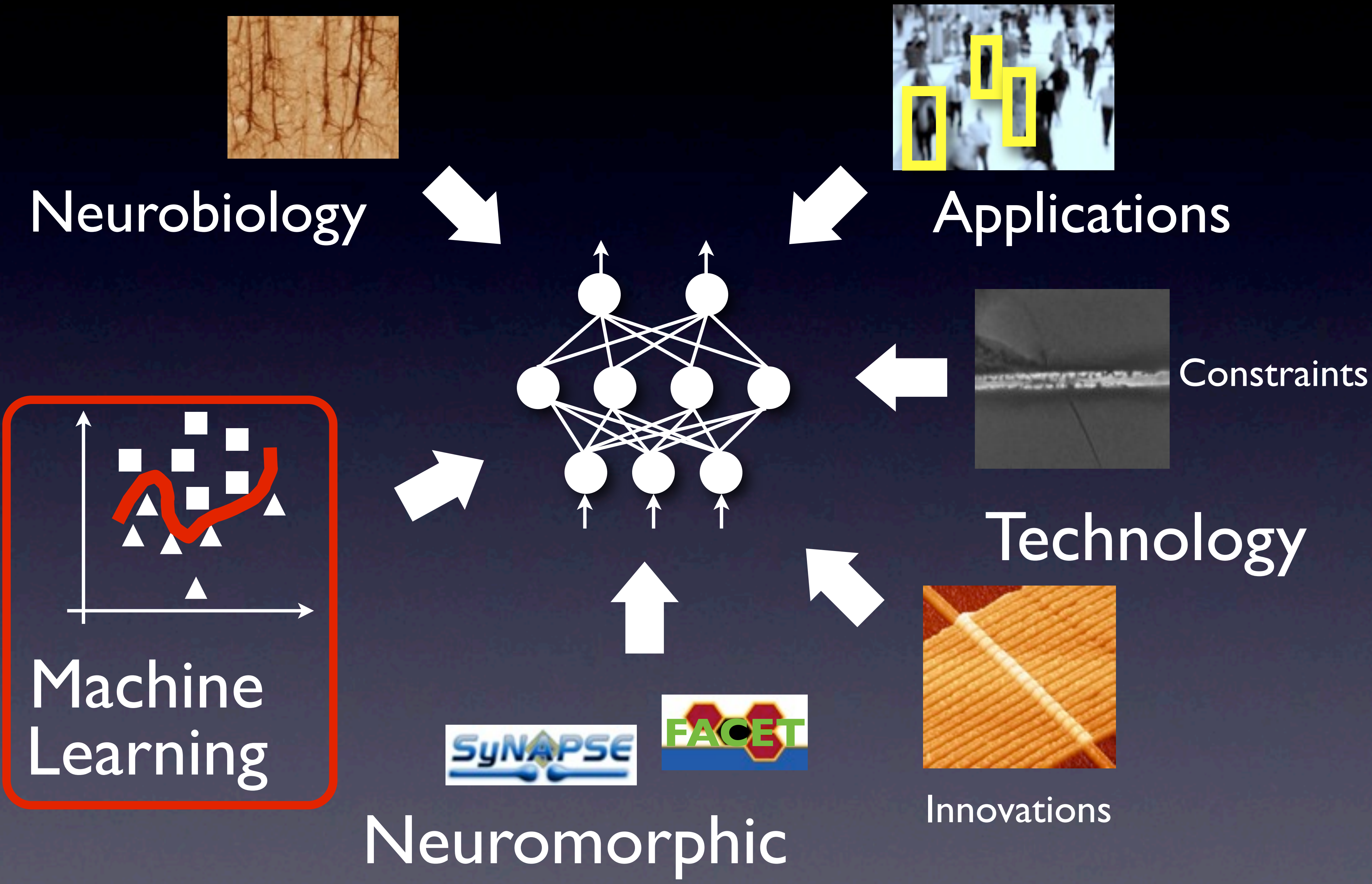
L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The Rebirth of Neural Networks

Olivier Temam
INRIA Saclay

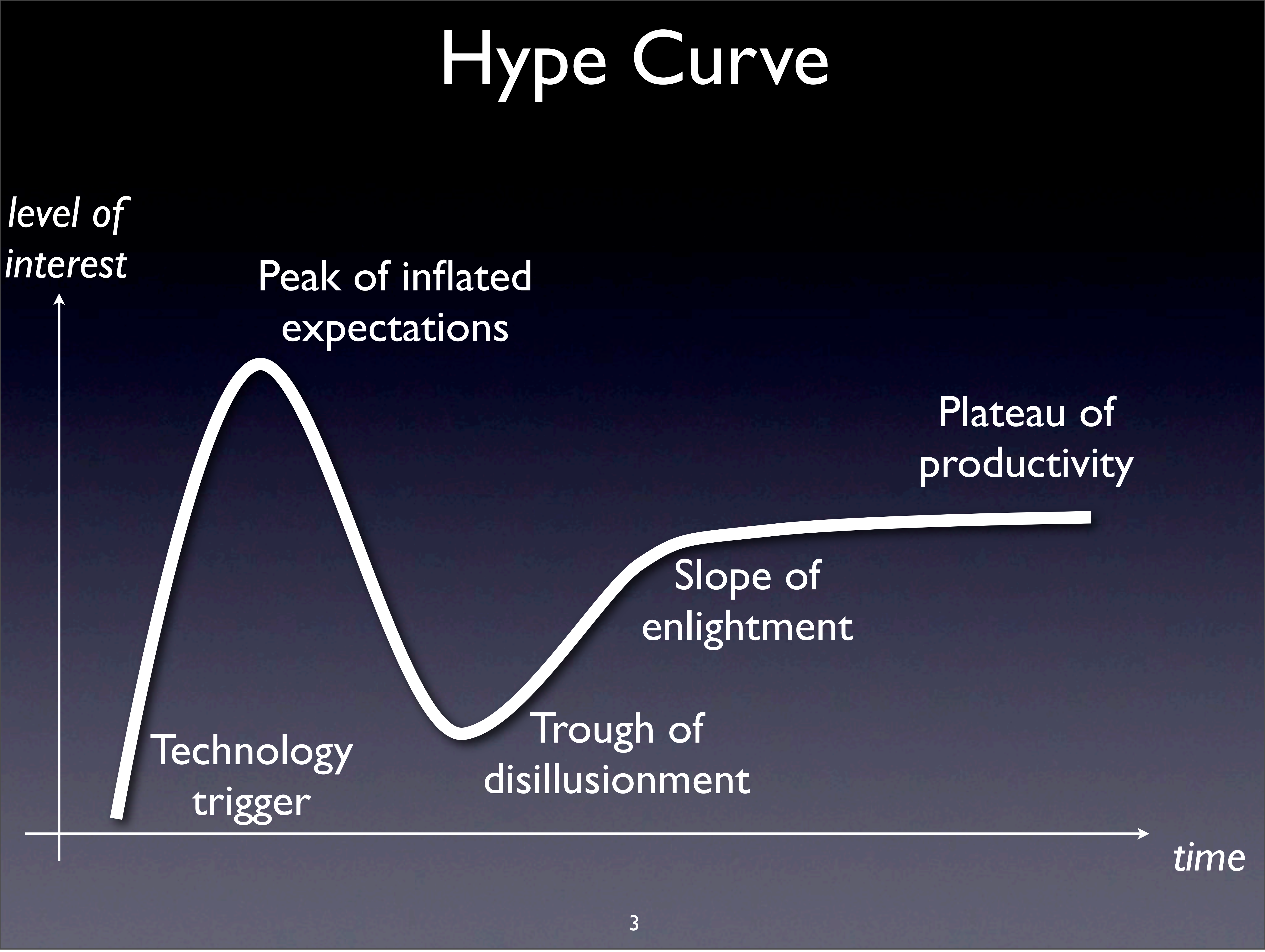
I got requests for a recorded version of the keynote. Rather than a recorded version, I thought that a version with some of the key points of each slide written down would allow to more quickly browse through my slides. These are not my “slides notes”, more of a summary of what I said, which I later wrote down.

Convergence of Trends



with M. Sebag (CNRS)

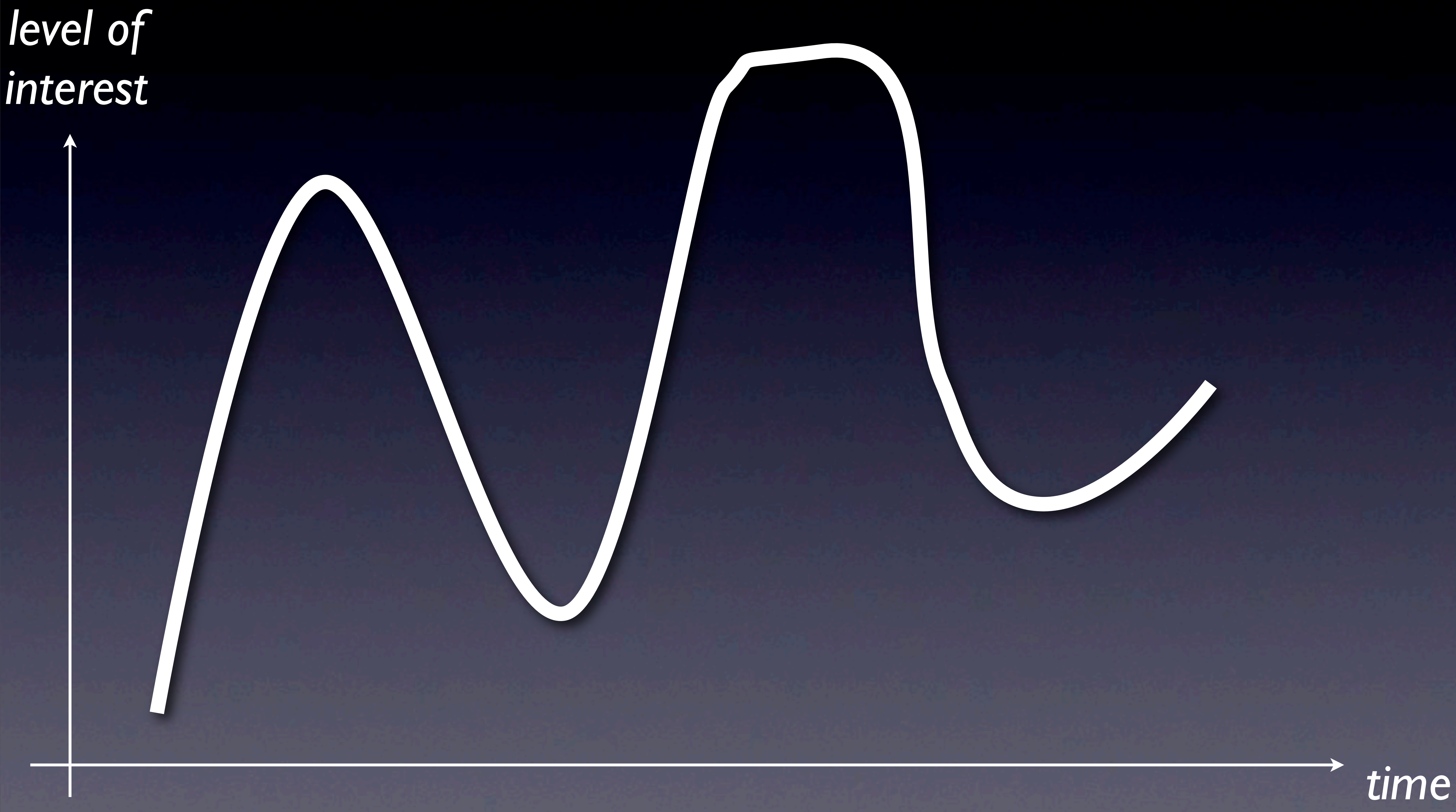
Lots of things happening in several domains of science (machine-learning, neurobiology, physics, neuromorphic and our own domain) could make neural networks increasingly relevant to our community.



Hype Curve

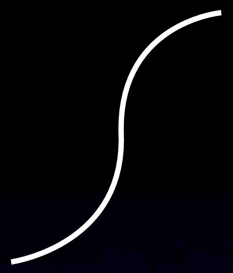
The Gartner hype curve (a similar curve was used during an ISCA keynote at ISCA 2003).

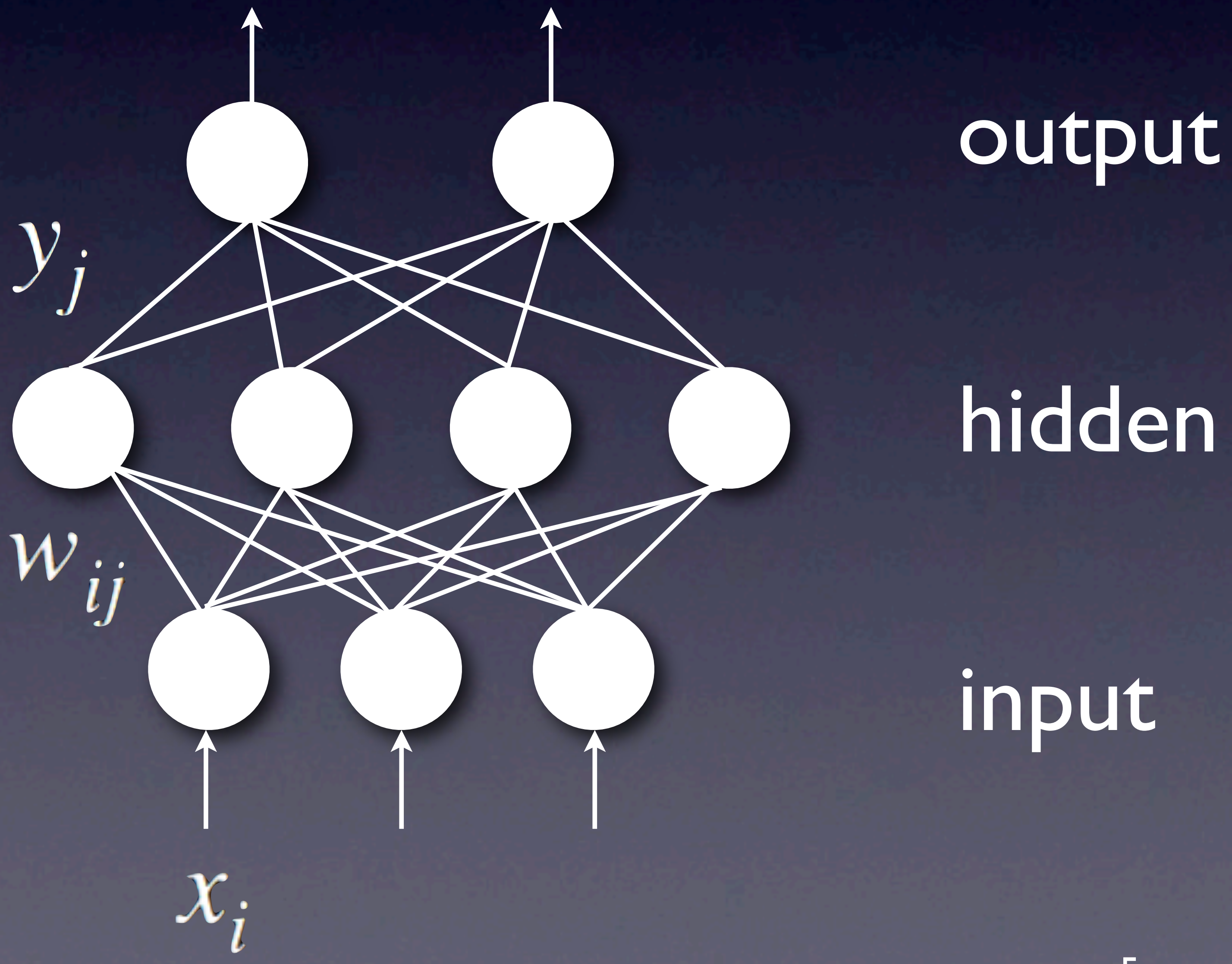
The Hype Curve of Neural Networks

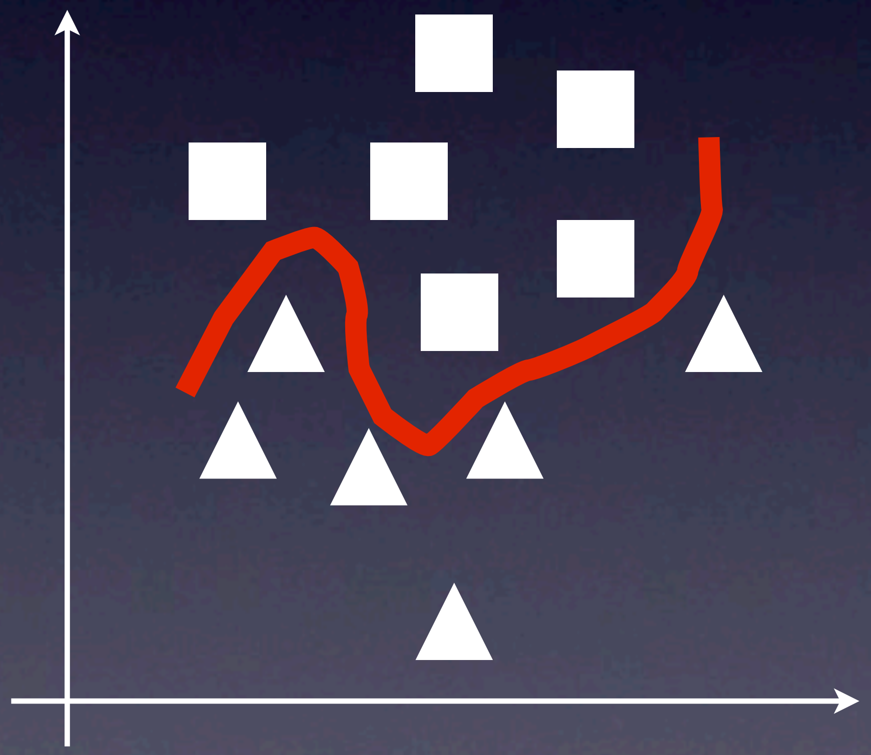


Like any new technology, NNs were hyped when first introduced, but their hype curve is atypical, NNs had a long and complicated history (see later slides for explanations).

Artificial Neural Networks (ANNs)

$y_j = f\left(\sum_i w_{ij} x_i\right)$, $f =$ 

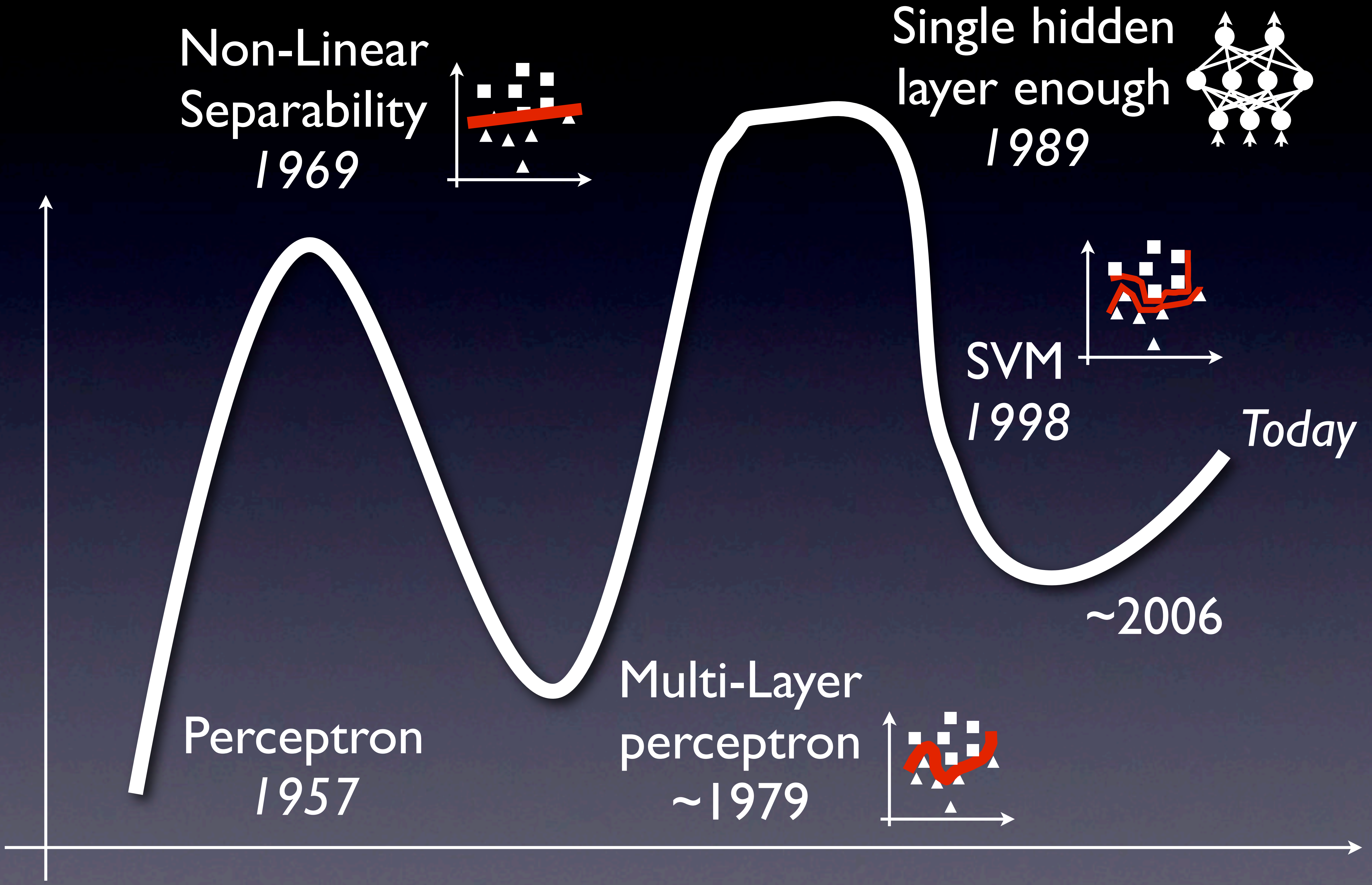




Classifier

First, quick recap on ANNs (will talk about biological NNs later on), especially the Multi-Layer Perceptron (MLP).
Main usage: as a classifier (can map n-characteristic input data to p classes, and learn separation between classes).

The Hype Curve of Neural Networks



Non-Linear Separability *1969*

Single hidden layer enough *1989*

SVM *1998*

Today

~2006

Multi-Layer perceptron ~1979

Perceptron *1957*

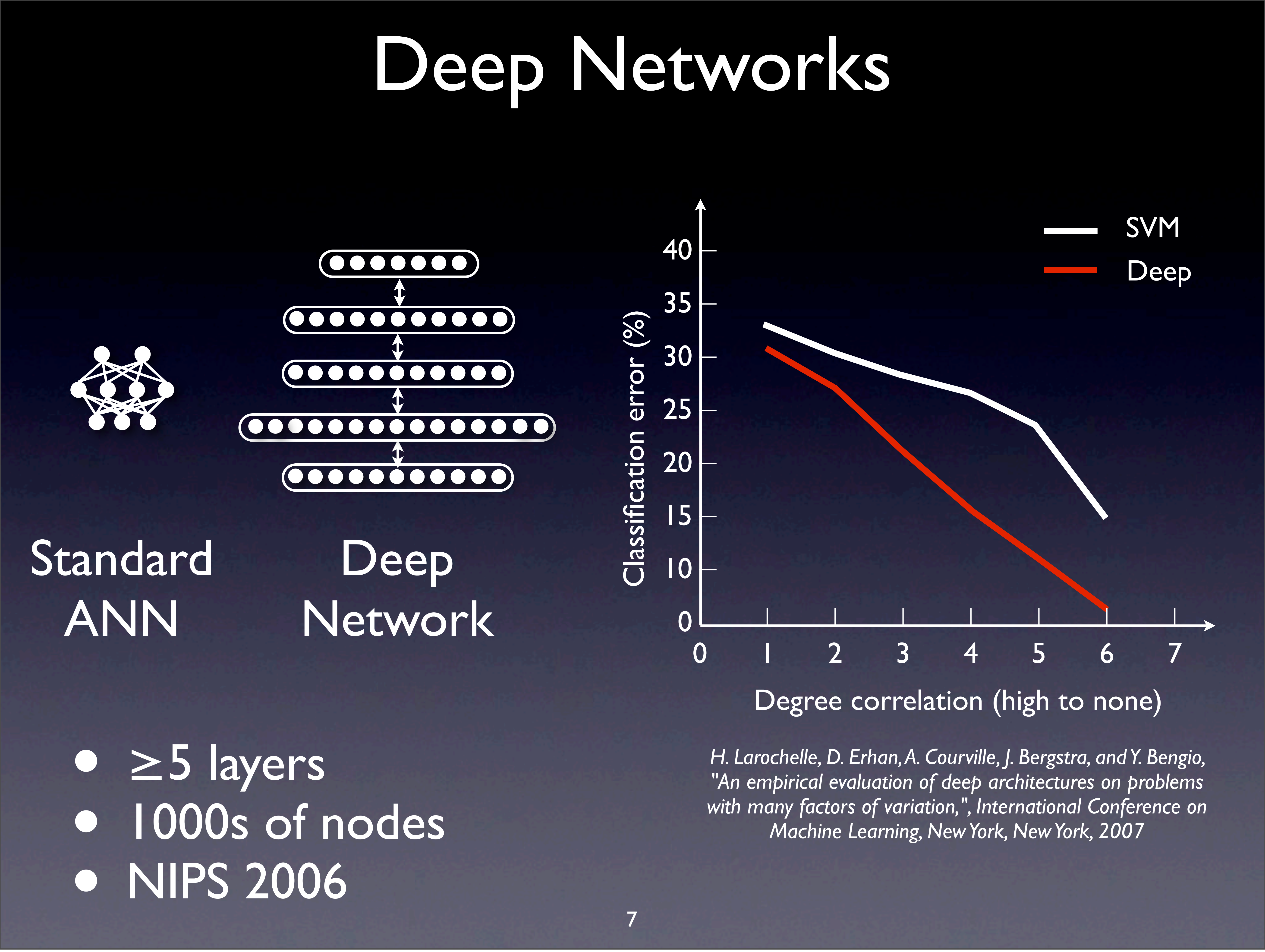
Hype curve with dates. Initial perceptron => excitement. But, with two layers, only linear separability (can only draw lines to separate classes) => disappointment.

Multi-Layer Perceptron: non-linear separability => new excitement.

Cybenko's theorem (MLP can approximate any continuous function with arbitrary accuracy using a <u>single</u> layer) => tricks machine-learning (ML) researchers into thinking a single layer enough.

SVM (Support Vector Machine) => classifier with better theoretical properties than ANNs, and outperforms ANN s=> second disillusionment for ANNs.

Since 2006: things have changed again.



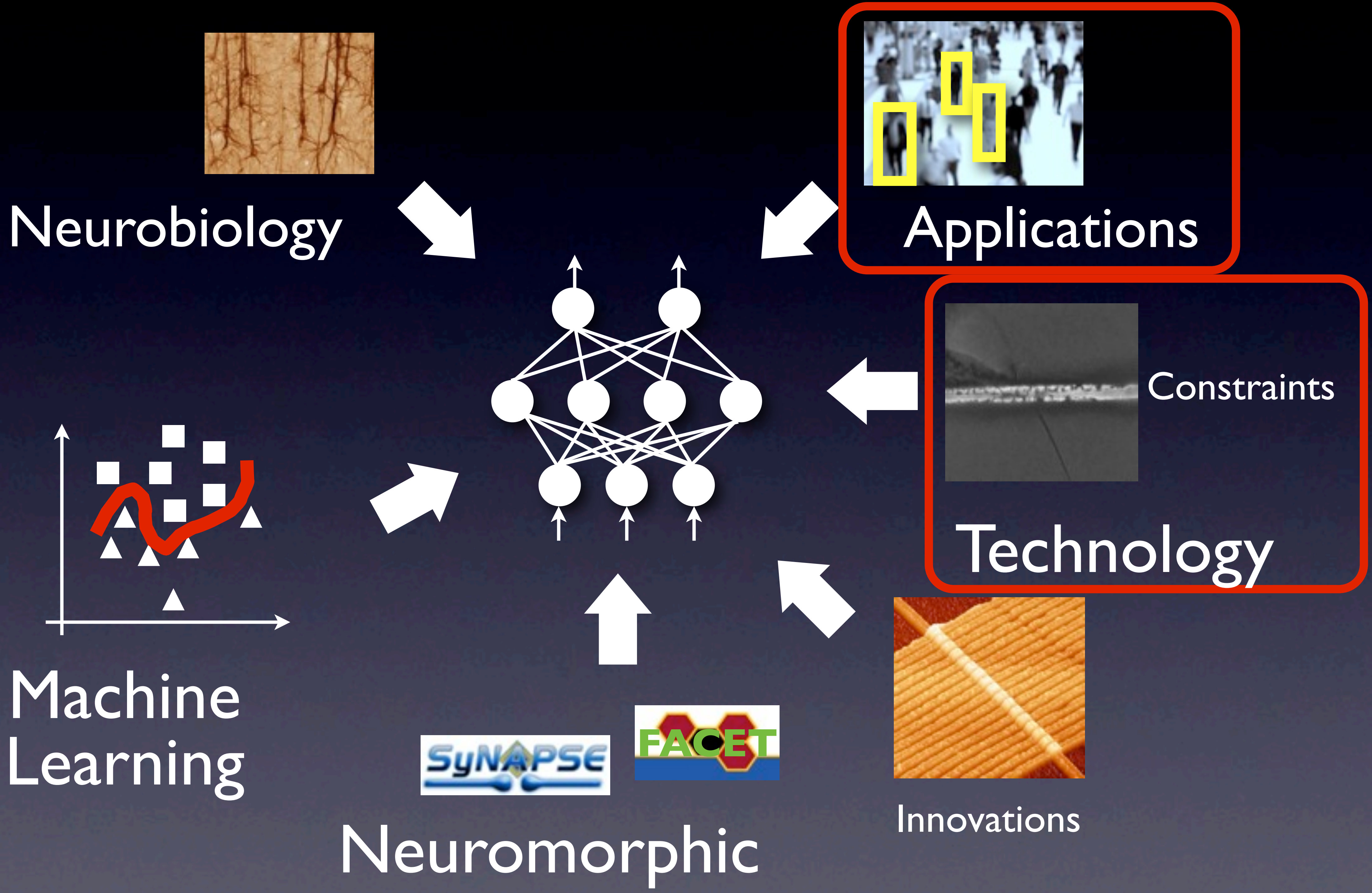
Deep Networks

- ≥5 layers
- 1000s of nodes
- NIPS 2006

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation,", International Conference on Machine Learning, New York, New York, 2007

Deep Networks (NIPS 2006): ML researchers find that by increasing # and size of layers, certain types of ANNs can outperform SVMs on a broad range of tasks.
Experimental evidence provided by Bengio's group (Hugo Larochelle) and many others since then.
ANNs are now state-of-the-art classifiers again.

Convergence of Trends



Neurobiology

Applications

Constraints

Technology

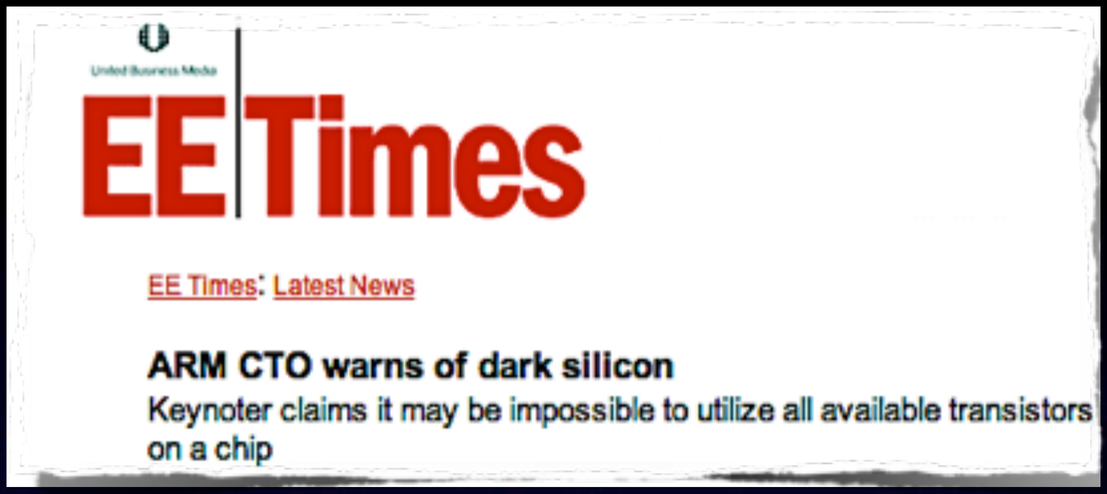
Machine Learning

Neuromorphic

Innovations

In our own domain:
- recent shift in application focus;
- importance of technology constraints, especially power, increasingly defects.

Toward Heterogeneous Systems



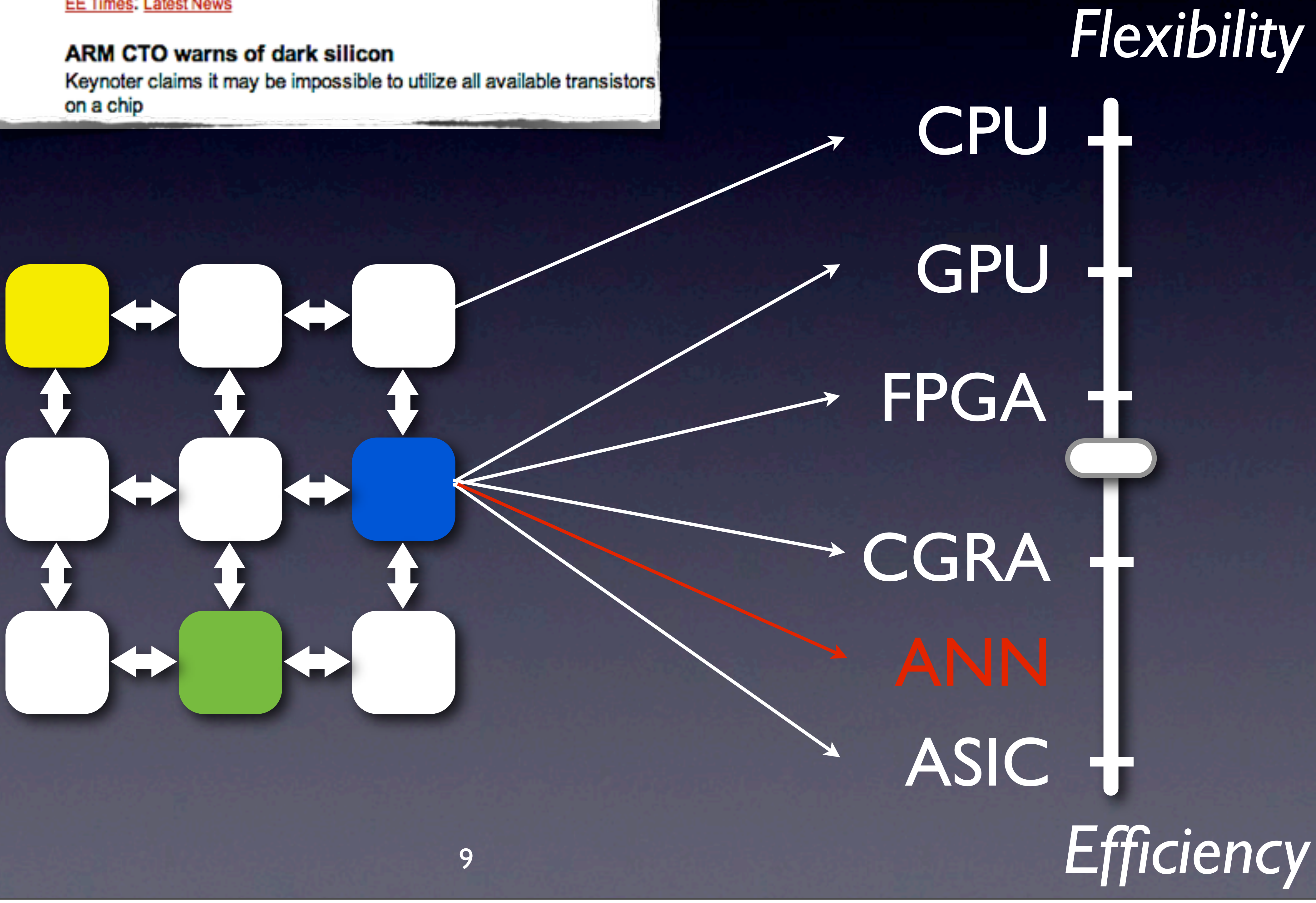
United Business Media

EE|Times

EE Times: Latest News

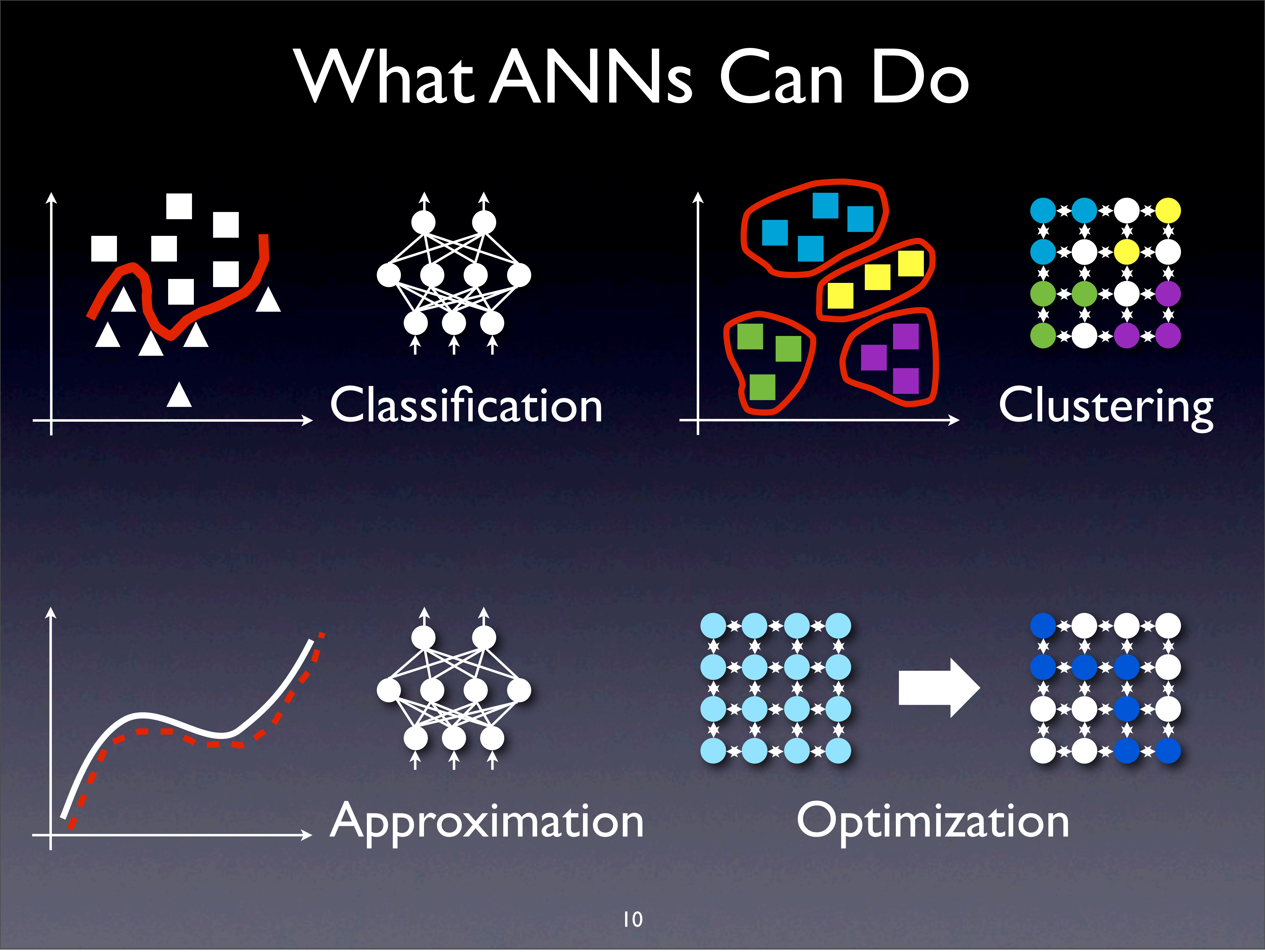
ARM CTO warns of dark silicon

Keynoter claims it may be impossible to utilize all available transistors on a chip



Recent warnings about power (e.g., “Dark Silicon” as coined by ARM). As a result, possible evolution towards heterogeneous systems: program decomposed into “sequence of algorithms”, with each algorithm mapped onto one or a few accelerators at any given time: a fraction of transistors used at any given time (circumvents “Dark Silicon” issue). But which accelerators ?

All design points valid. ASIC best “power” design point but not flexible. FPGA flexible but significantly less power efficient. Alternative: “multi-purpose” ASICs which target several generic (possibly fine-grain algorithms). ANN is a candidate multi-purpose ASICs. Naturally, interrogations about its application scope.



What ANNs Can Do

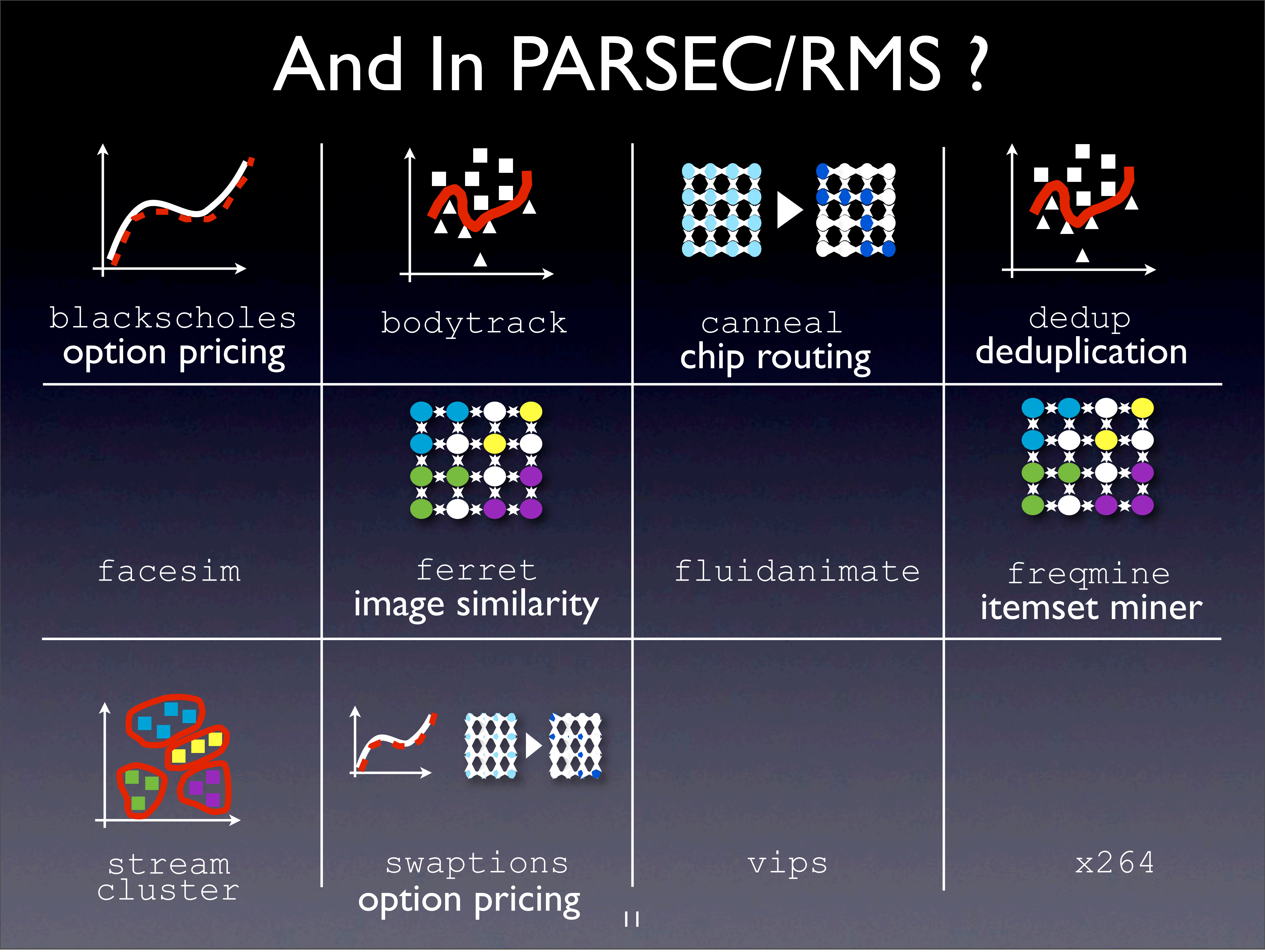
Classification

Clustering

Approximation

Optimization

The four major types of algorithms which ANNs are good at.

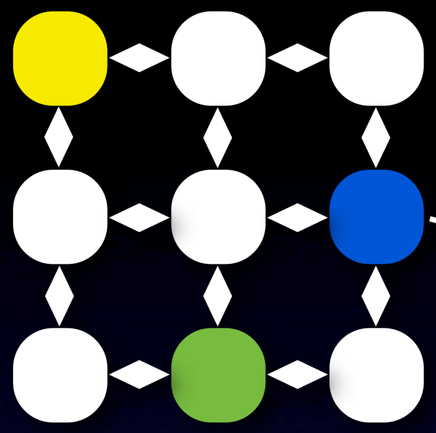


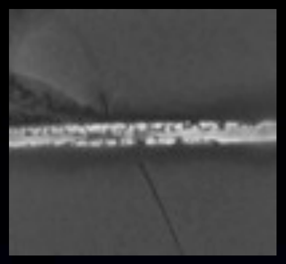
And In PARSEC/RMS ?

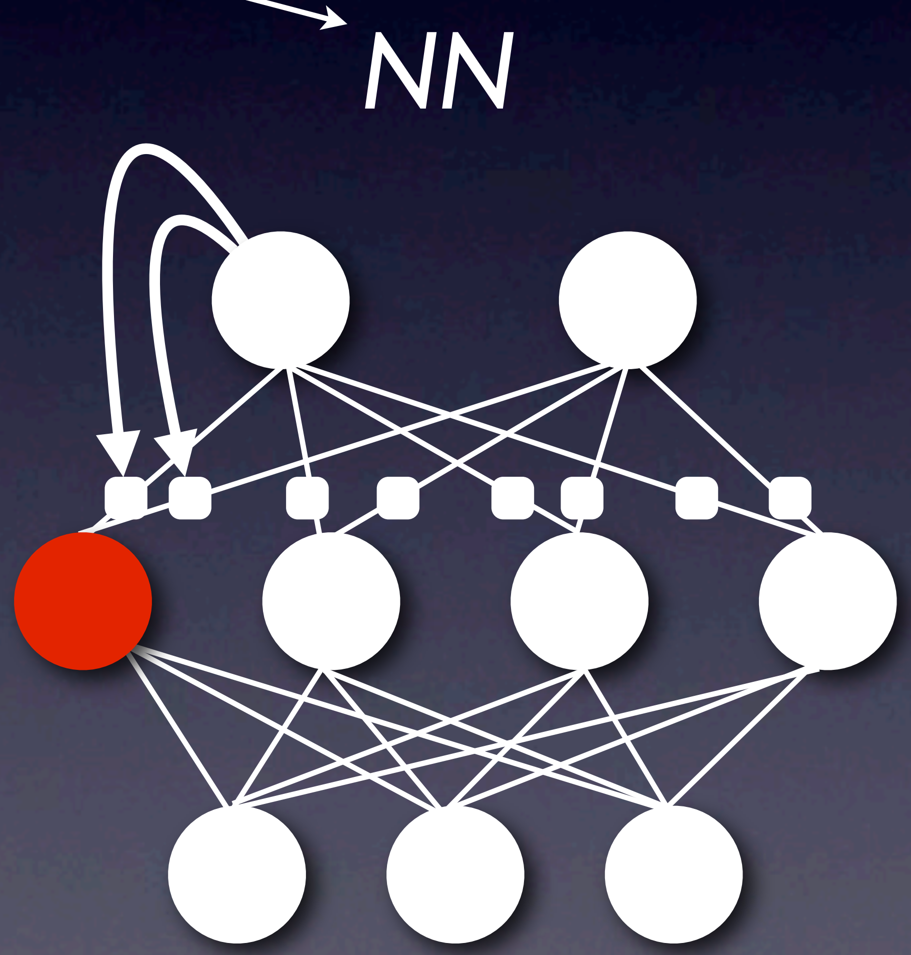
blackscholes option pricing	bodytrack	canneal chip routing	dedup deduplication
facesim	ferret image similarity	fluidanimate	freqmine itemset miner
stream cluster	swaptions option pricing	vips	x264

Intel attracted attention on RMS (Recognition, Mining, Synthesis) as emerging high-performance applications in 2005. Many of these applications (especially RM) rely on statistical and machine-learning algorithms. For quite a few of these algorithms, competitive implementations based on ANNs exist (clearly for 6 out of 12 PARSEC benchmarks; a bit less clear-cut for dedup and freqmine). However, no compelling reason to use ANNs for implementing most of these tasks.

Defects-Tolerant Accelerators ?



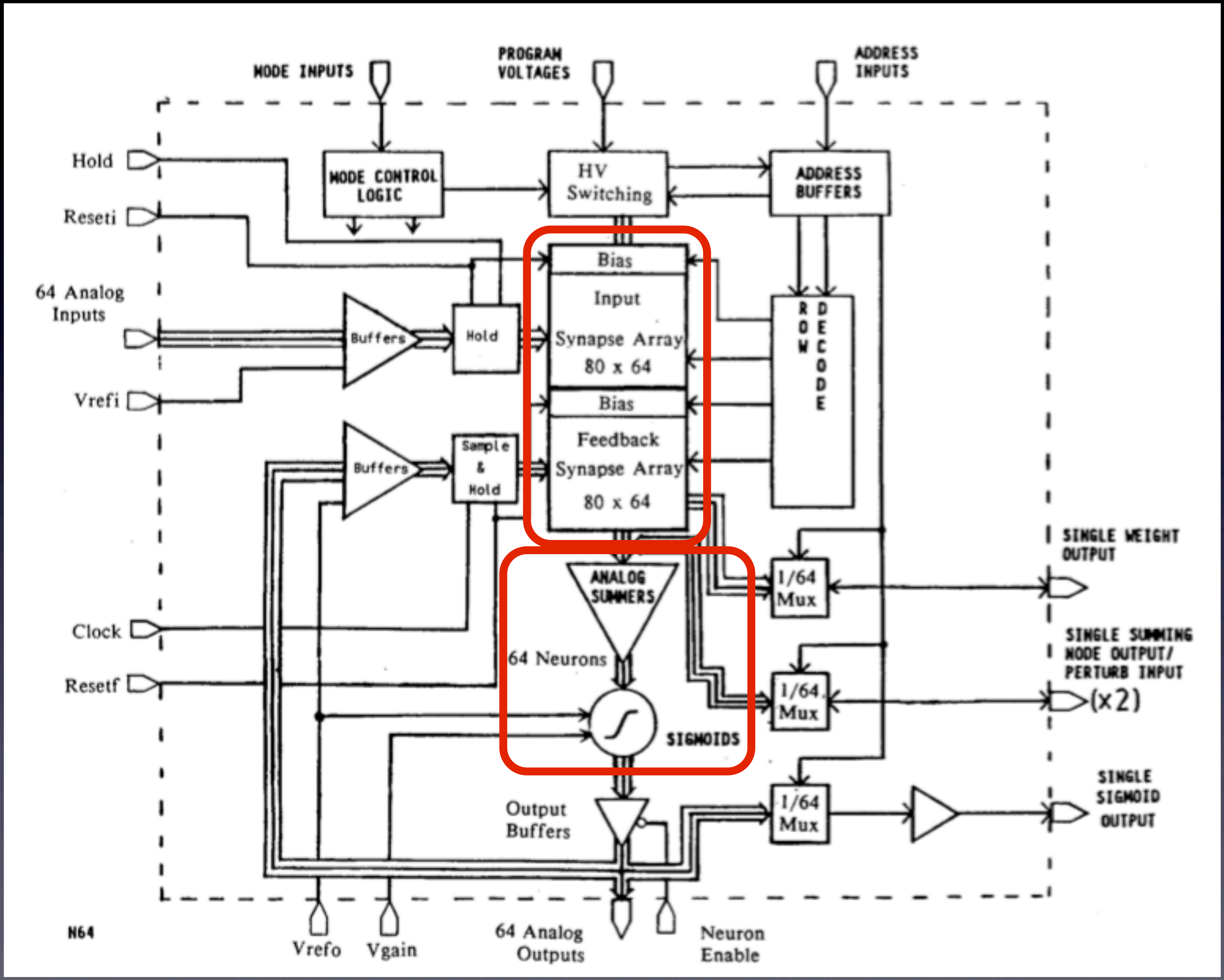




- No need to identify/ disable: just learn
- Demise of hardware NNs:
  - SVM/algorithmic;
  - application scope
  - “killer micro”

A more compelling reason: seek defects-tolerant implementations of these tasks.
Defect tolerance is a strong point of ANNs: no need to identify/disable faulty parts, training algorithm naturally/automatically silences out faulty synapses/neurons by decreasing synaptic weights if erratic (uncorrelated) values.
Now, hardware NNs not new, a lot of research end of 1980s, beginning 1990s; died off because (among others): (1) dominance of SVM for a while, (2) application scope limited in era of Perfect Club benchmarks, could be changing, (3) killer micro, just like for massively parallel machines: software ANNs run on GPP competitive with hardware ANNs after a few generations; no longer true with lack of clock scaling.
Do we just have to reuse past hardware ANNs designs ?

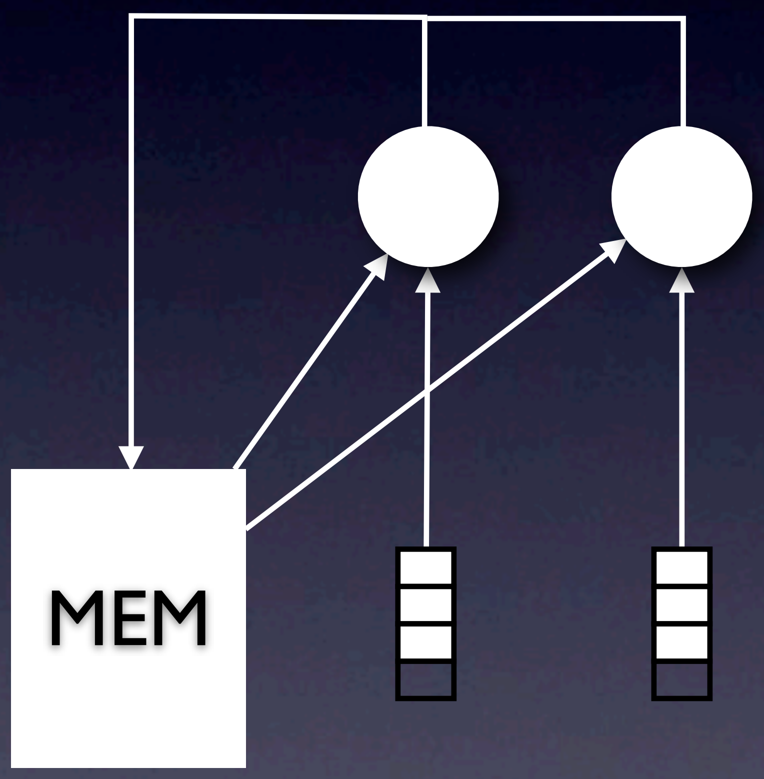
Example: Intel ETANN



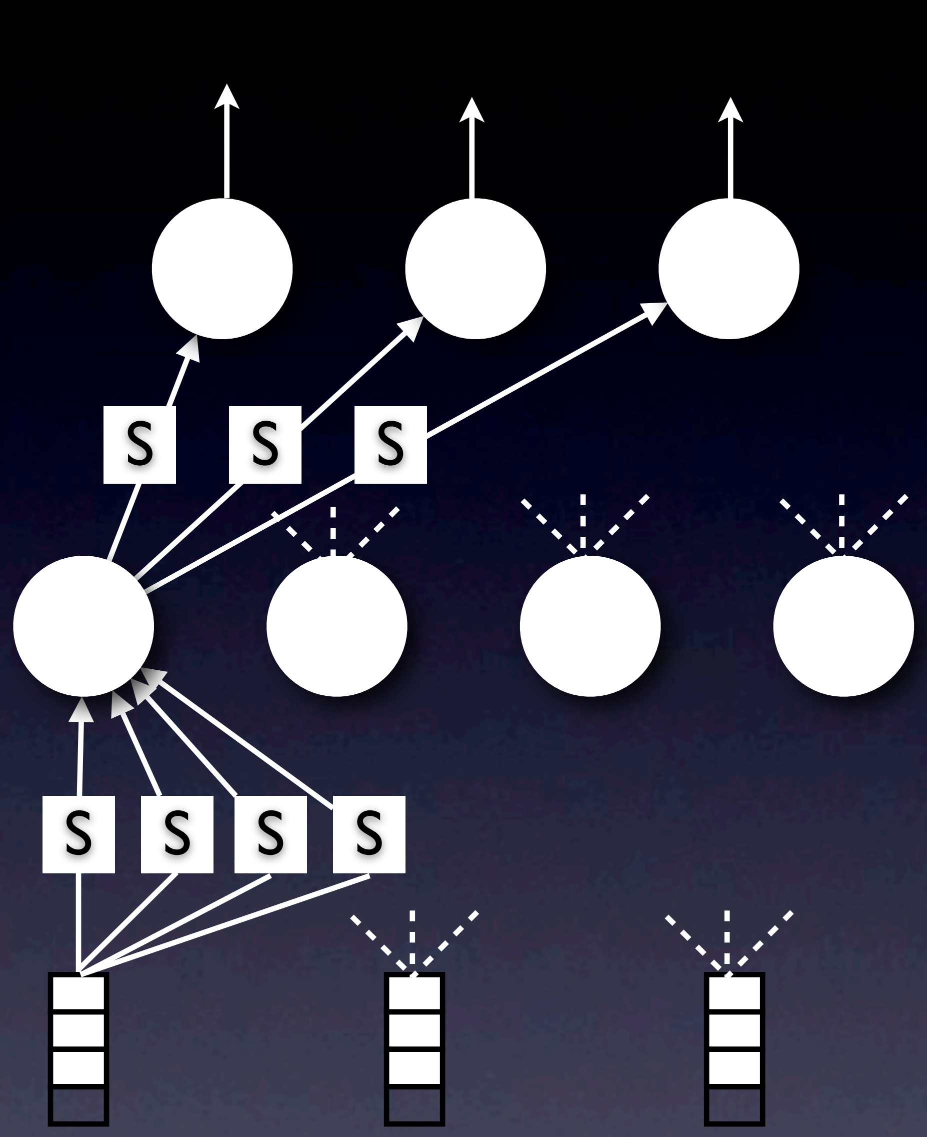
M. Holler, S.Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 "floating gate" synapses," Artificial neural networks, Piscataway, NJ, USA: IEEE Press, 1990

An example design from Intel, this one was analog; both analog and digital designs were proposed at the time.
Fairly typical of what existed at the time: bank of physical neurons, memory bank to store synapses and output of intermediate neurons.

Defects-Tolerant ANNs



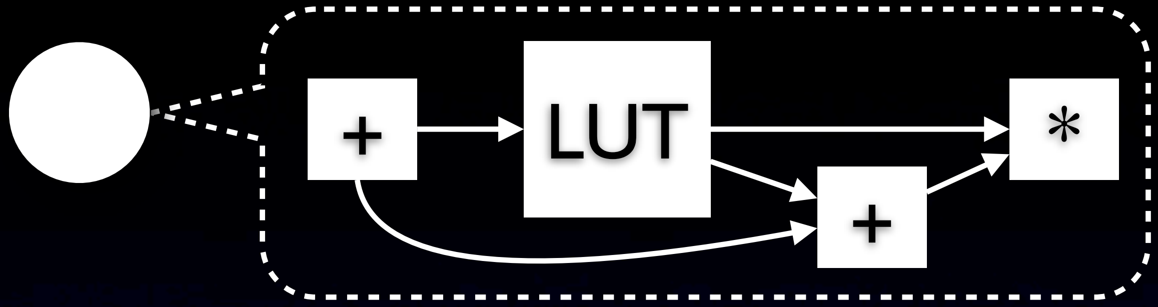
Spatially folded



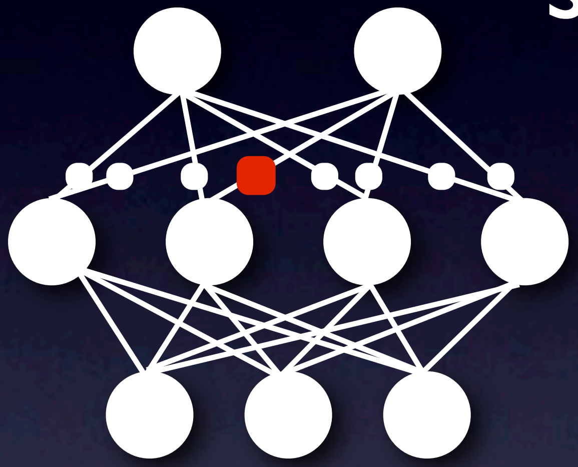
Spatially unfolded network

Most designs were “spatially folded” (see left) because (1) not many transistors at the time, (2) defects tolerance was not a primary motivation at the time.
But this design is not very defects tolerant: a single defect in the memory decoder could wreck the accelerator; a defect in a neuron would either result in loss of significant share of network or significantly degrade performance of network.
Now, more transistors allow to “spatially unfold” designs, closer to conceptual view of ANNs. Multiple benefits: better defects tolerance (amplified by spatial distribution of synaptic storage and by degree of expansion), possibility to spatially distribute storage (synapses) close to computations which is key for reducing power, combinational execution is possible (no need to pipeline), etc.

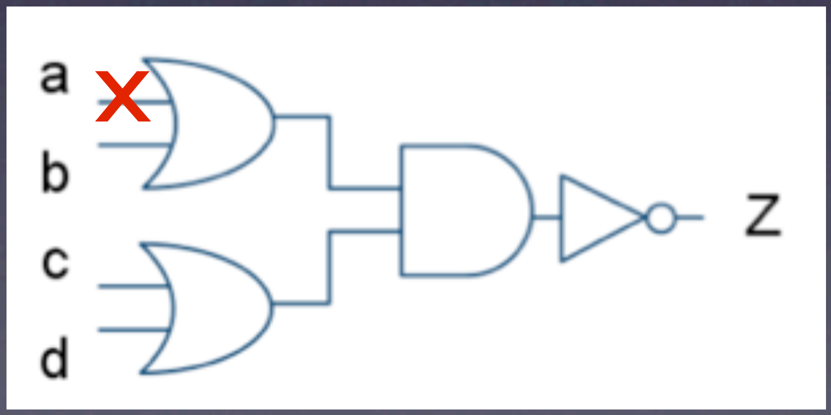
Defect Modeling



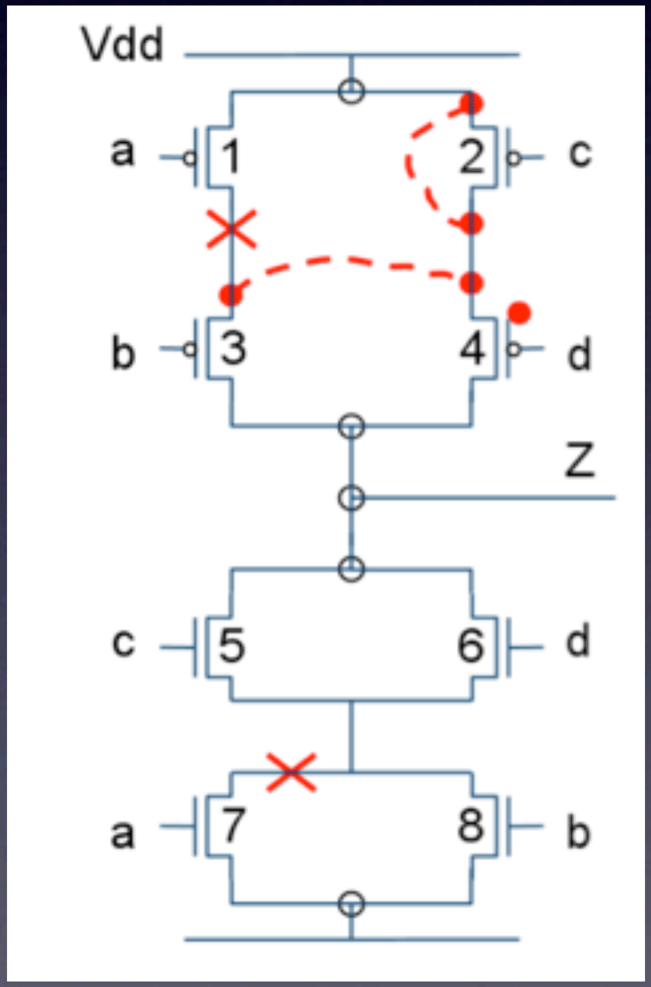
sigmoid(sum(input * weight))



Stuck-At synapse



Stuck-At gate input



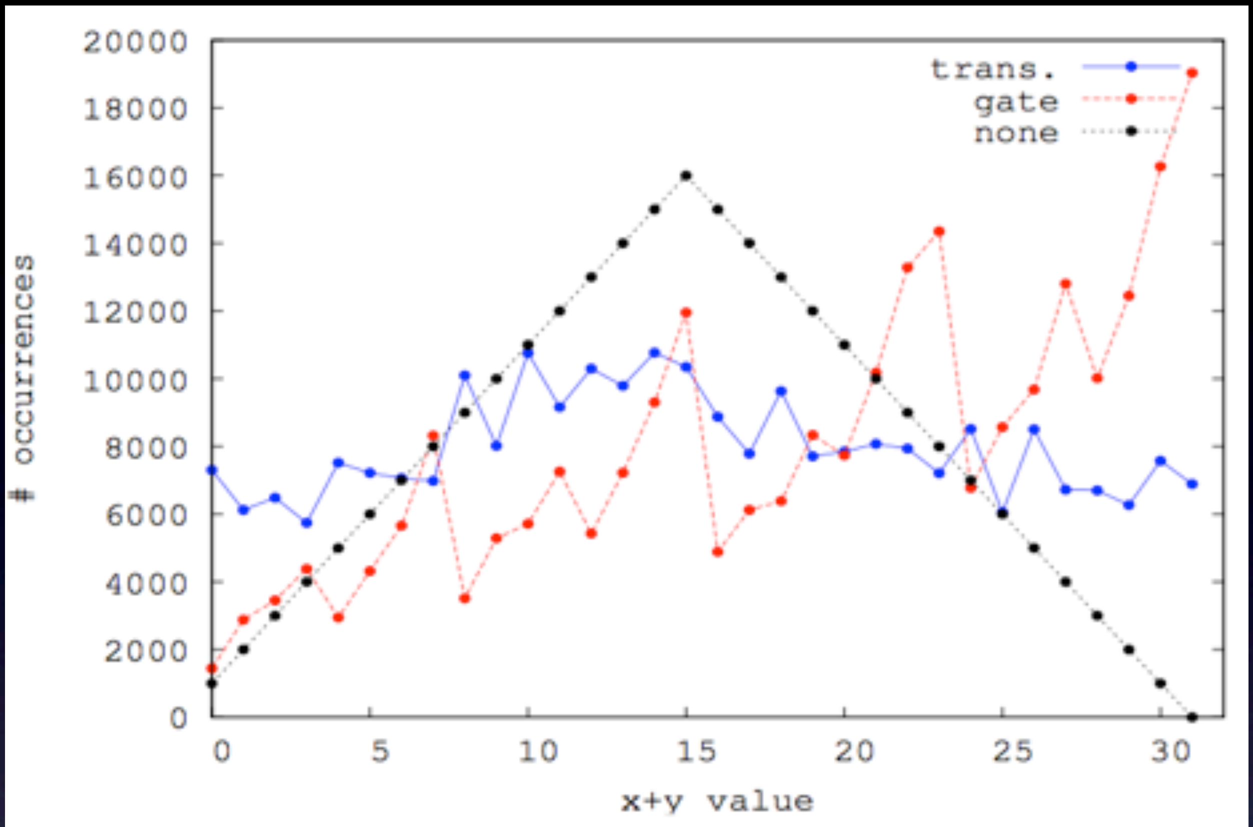
Shorts & Opens

→ Logic gate ?

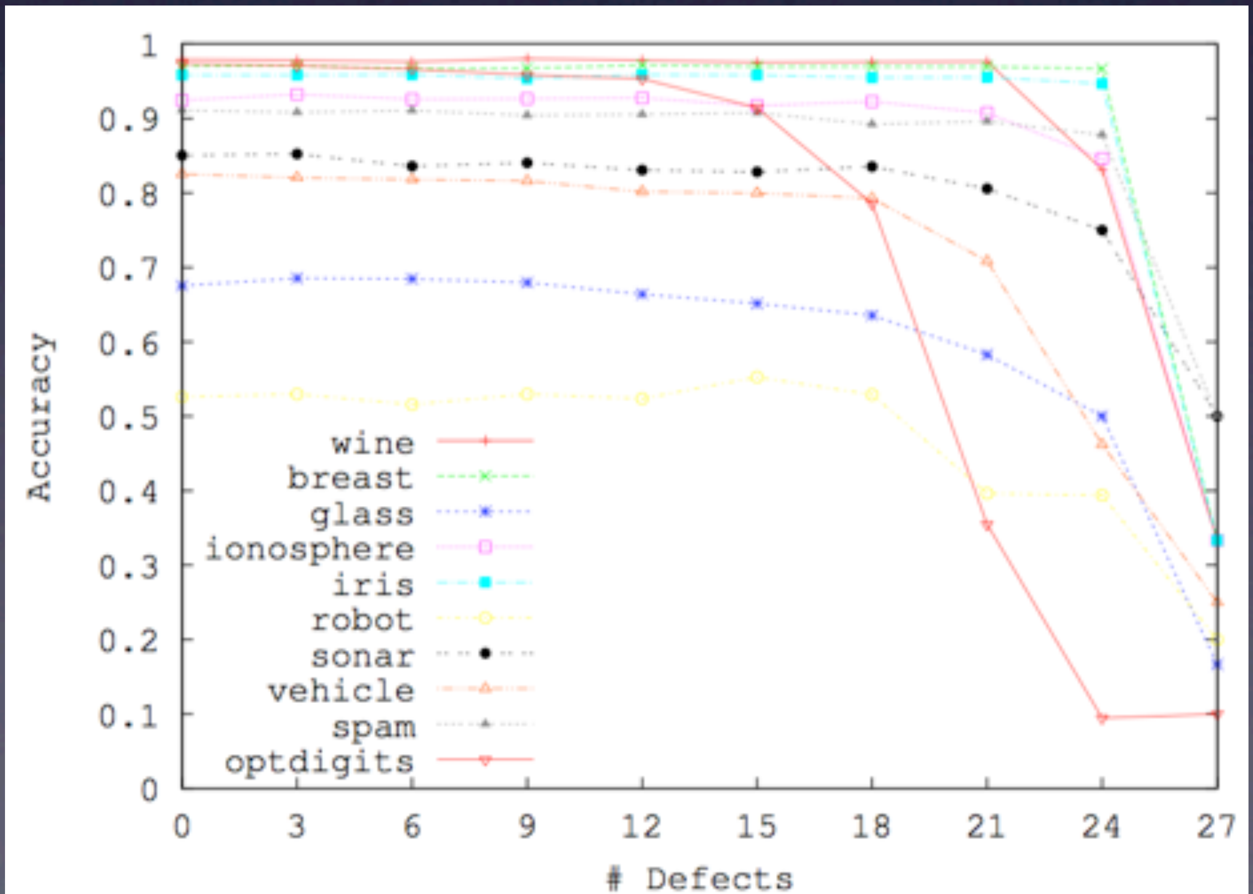
Defects tolerance of ANNs still "conceptual", what does it exactly mean at the hardware level ?
Need to accurately model defects because values produced by "faulty" neurons or synapses could influence behavior of training algorithm. Defects tolerance can be properly assessed, and hardware ANN can be properly designed only if we know "correct" values output by faulty neurons/synapses (oxymoron). Moreover, in digital implementations, logic operators (add, mult) account for significant share of area. In many papers, even recent ones, ANN defect often = stuck-at synapse; not realistic from hardware standpoint. Need to properly emulate behavior of faulty operators. Stuck-at gate model OK for test purposes in micro-architectures, but not sufficient for obtaining "correct" values of faulty neurons.
Inject defects at transistor level and reconstruct corresponding flawed gate.

Hardware ANN Robustness

4-bit adder,
20 defects



ANN,
defect
tolerance



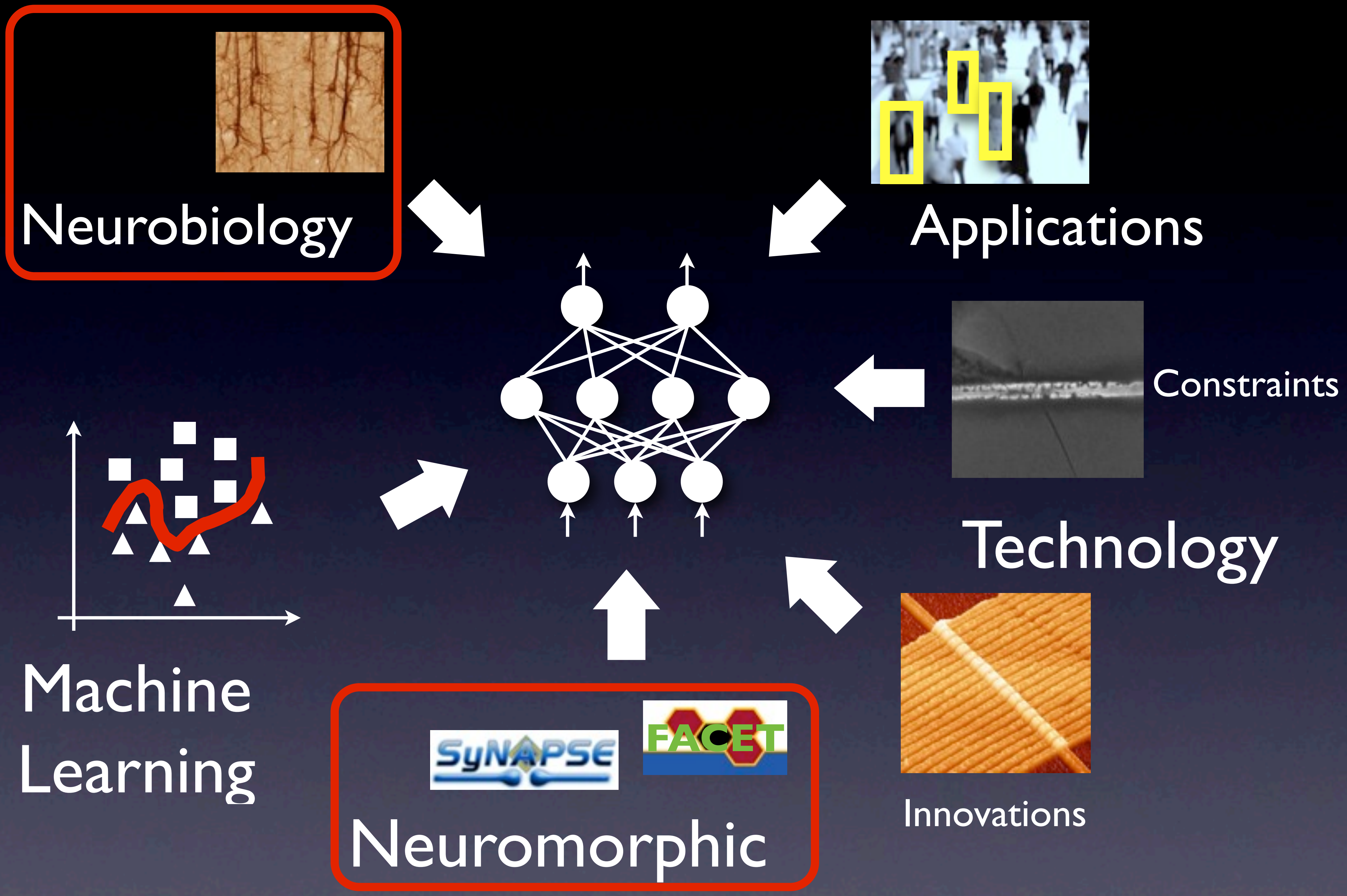
UCI
Machine Learning Repository

53100:	Iris
42404:	Adult
36191:	Wine
31222:	Breast Cancer
24315:	Abalone
24191:	Poker Hand
17796:	Yeast
17366:	Forest Fires
16933:	Internet Advert
15807:	SPECT Heart

90 inputs,
10 outputs
90% UCI

Illustration of the different behavior of a faulty operator (4-bit adder) if defects injected at gate or transistor level.
Method used to inject defects in hardware ANN (Verilog implementation). Small digital ANN but can already tackle about 90% tasks of UC Irvine Machine-Learning Repository based on # attributes/classes. Defects tolerance of hardware ANN observed on a sample of 10 UCI cases.

Convergence of Trends



Neurobiology

Applications

Constraints

Technology

Machine Learning

Innovations

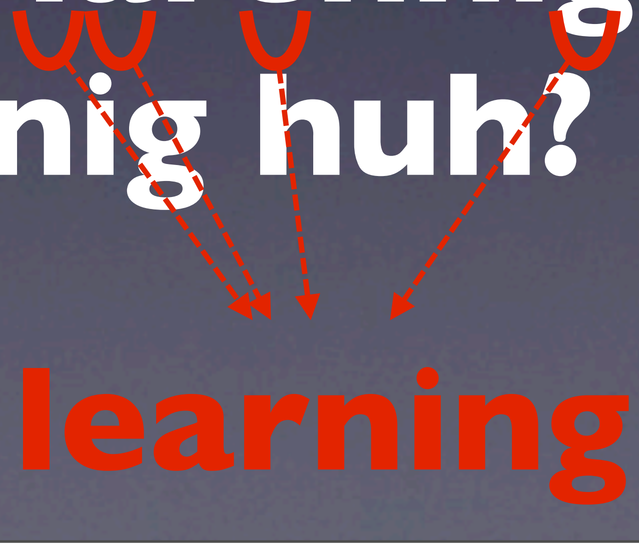
Neuromorphic

with A. Hashmi, A. Nere, M. Lipasti (Univ. Wisconsin)
H. Berry (INRIA)

Both application scope and defects tolerance capabilities of ARTIFICIAL Neural Networks already significant. But how can they be further expanded ?
Biological neural networks suggest both capabilities (application scope and defects tolerance) can be significantly expanded.

Beyond ANNs: Biological NNs

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be at the rghit pclae. And we spnet hlaf our lfie larennig how to splel wrods. Amzanig huh?



learning

Illustrating the defects tolerance (even to noisy input) and application capabilities of biological neural networks.
Intuitively, principle is to map "features" (e.g., presence of letters, at certain positions) to high-level concepts, such as words.

How Can Computers Do the Same ?

FACETS

FET

Integrate & Fire
250,000 neurons
wafer
~ 10^4 acceleration

Blue Brain Project

IBM/EPFL

Molecular level
10,000s neurons
1000s cores
~ real-time

SyNAPSE

DARPA

Integrate & Fire (?)
10^8 neurons
new silicon devices

SpiNNaker
a universal Spiking Neural Network architecture

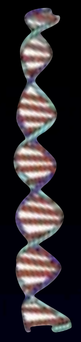
Integrate & Fire
10^9 neurons
65,000 chips
~ real-time

- Faithfully emulate neurons/synapses
    - Right abstraction level ?
- Achieve critical mass
- Find “algorithm”

10^{11} neurons
10^{15} synapses
30-400Hz

Lots of researchers hard at work trying to understand how biological neural networks work.
Four example projects (2 in US, 2 in Europe), roughly same approach:
- Emulate elementary components; disagreement on appropriate “abstraction level” (e.g., molecular vs. integrate and fire neuron model);
- Implement a large quantity of neurons with the expectation that bio-like behavior will start to emerge; hardware used varies a lot;
- Accept that assembling a large number of neurons is not sufficient and understand what, in structure of networks and connectivity (among others), yields bio-like behavior.

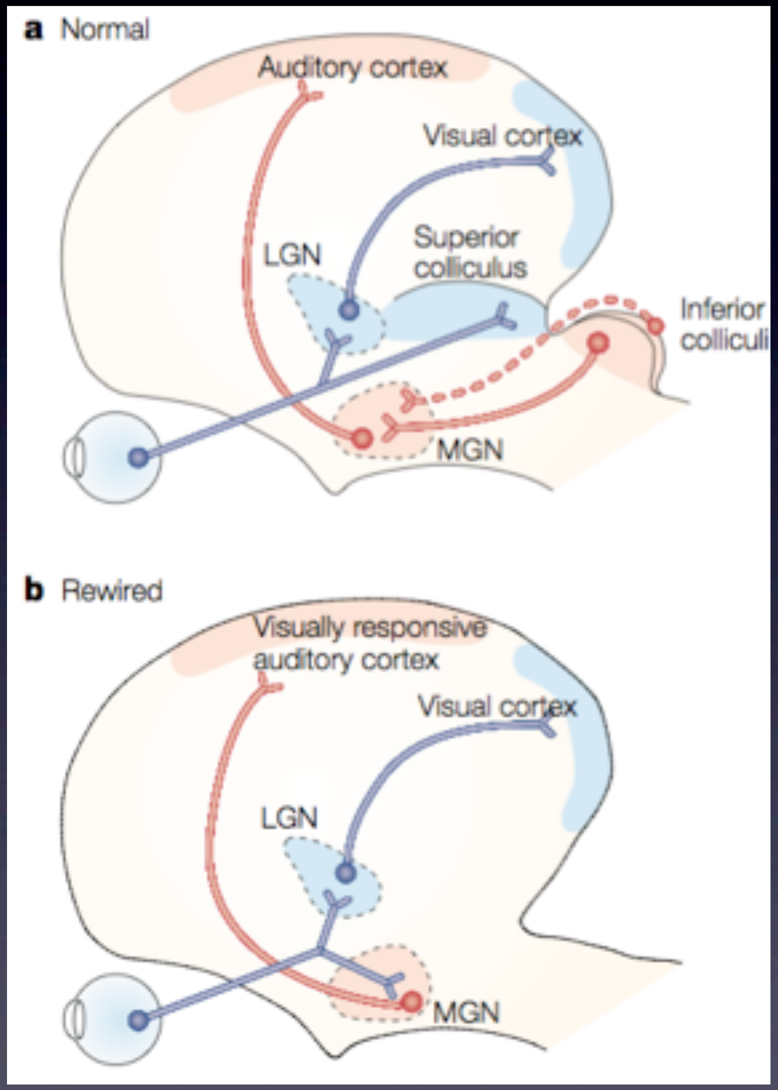
“Algorithm” ?



25000 genes

10^{11} neurons

- Likely existence of “generic algorithm”
- Neuroscientists starting to <u>reverse-engineer</u>
- Algorithm: automatic abstraction of data



Plasticity

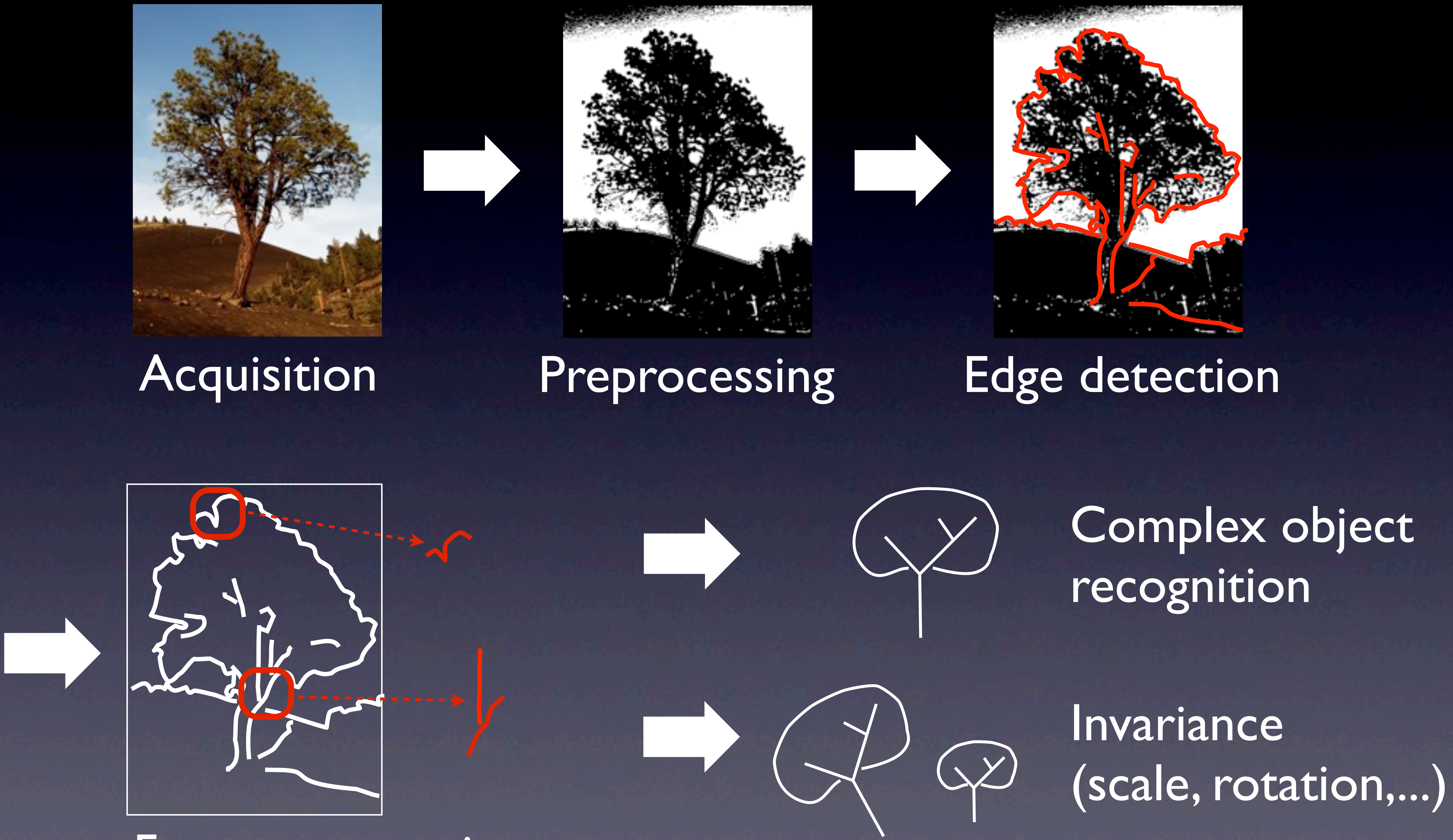
M. Sur and C.A. Leamey, "Development and plasticity of cortical areas and networks," Nature Reviews Neuroscience, vol. 2, 2001, pp. 251-62.

Probable existence of a “generic” algorithm born out of the network structure. Some biological evidences of the existence of that algorithm, as mentioned by biologists:

- # genes vs. # neurons;
- physiological inspection confirms cortical columns identical almost everywhere in cortex, including within “specialized” areas;
- plasticity, demonstrated by multiple experiments; here one case of “rewiring” the auditory/visual cortex of ferrets leads to auditory cortex performing visual-like processing, i.e., plasticity even in “specialized” areas;

Neuroscientists starting to reverse engineer “algorithm”. In a nutshell, it consists in automatically abstracting raw data into increasingly complex notions.

Example with Visual Cortex



Acquisition

Preprocessing

Edge detection

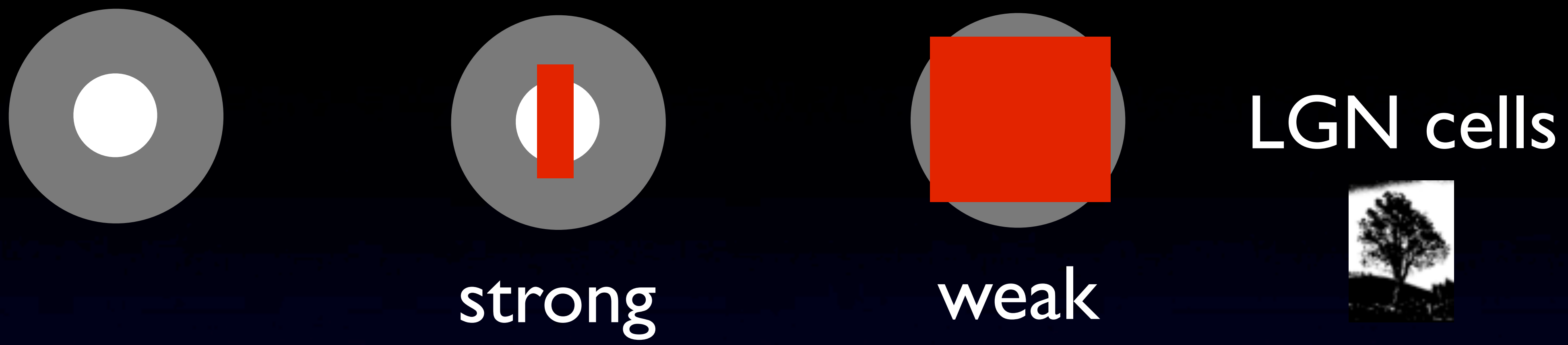
Feature extraction

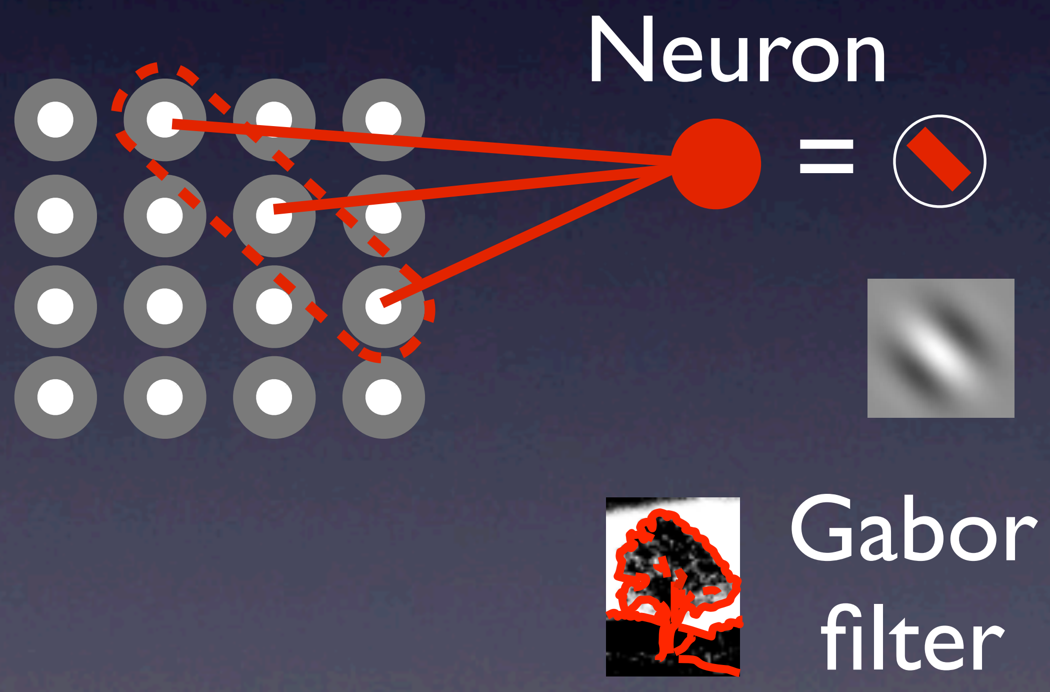
Complex object recognition

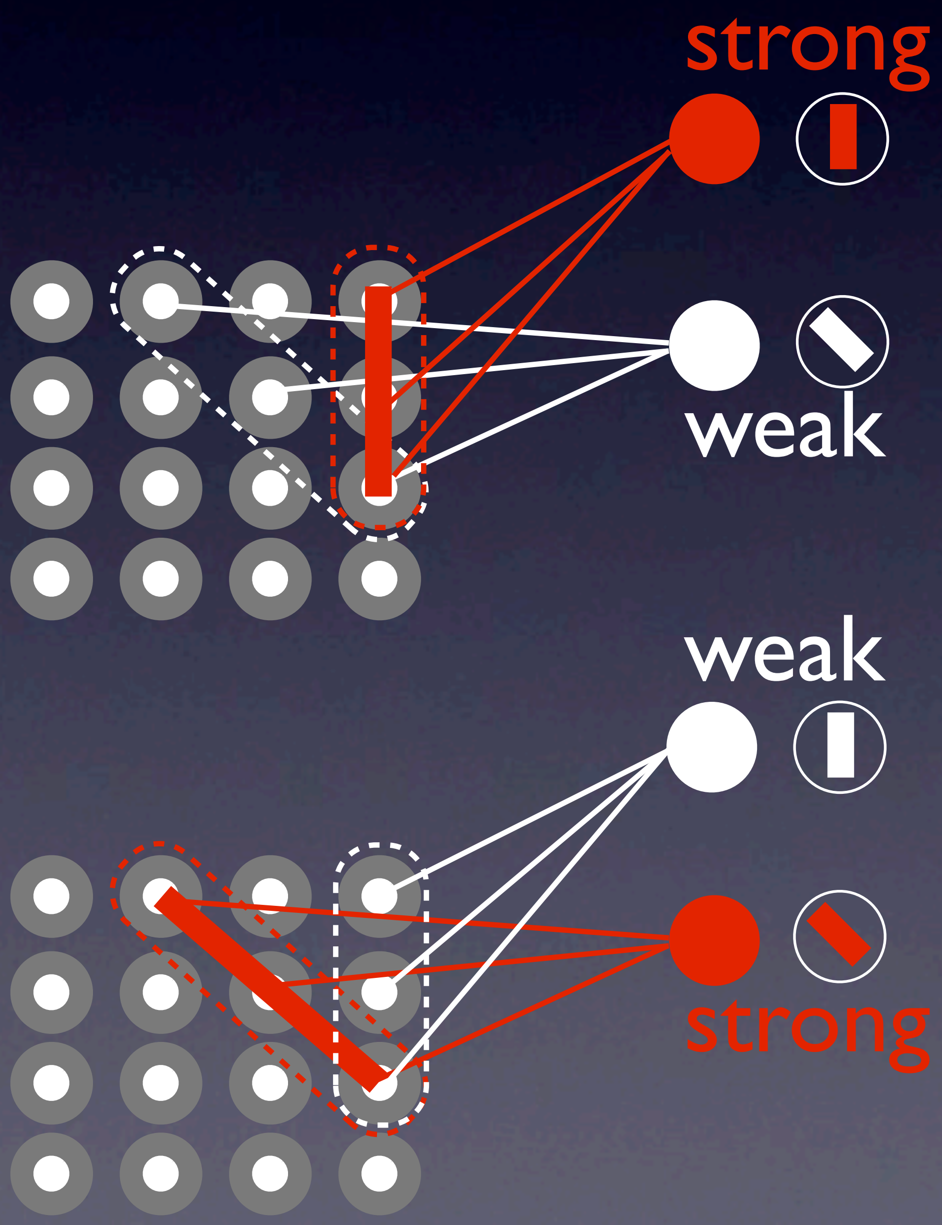
Invariance (scale, rotation,...)

Example with the visual cortex. Typical phases of the visual cortex. Biological implementation ?

Neuron-Level Model



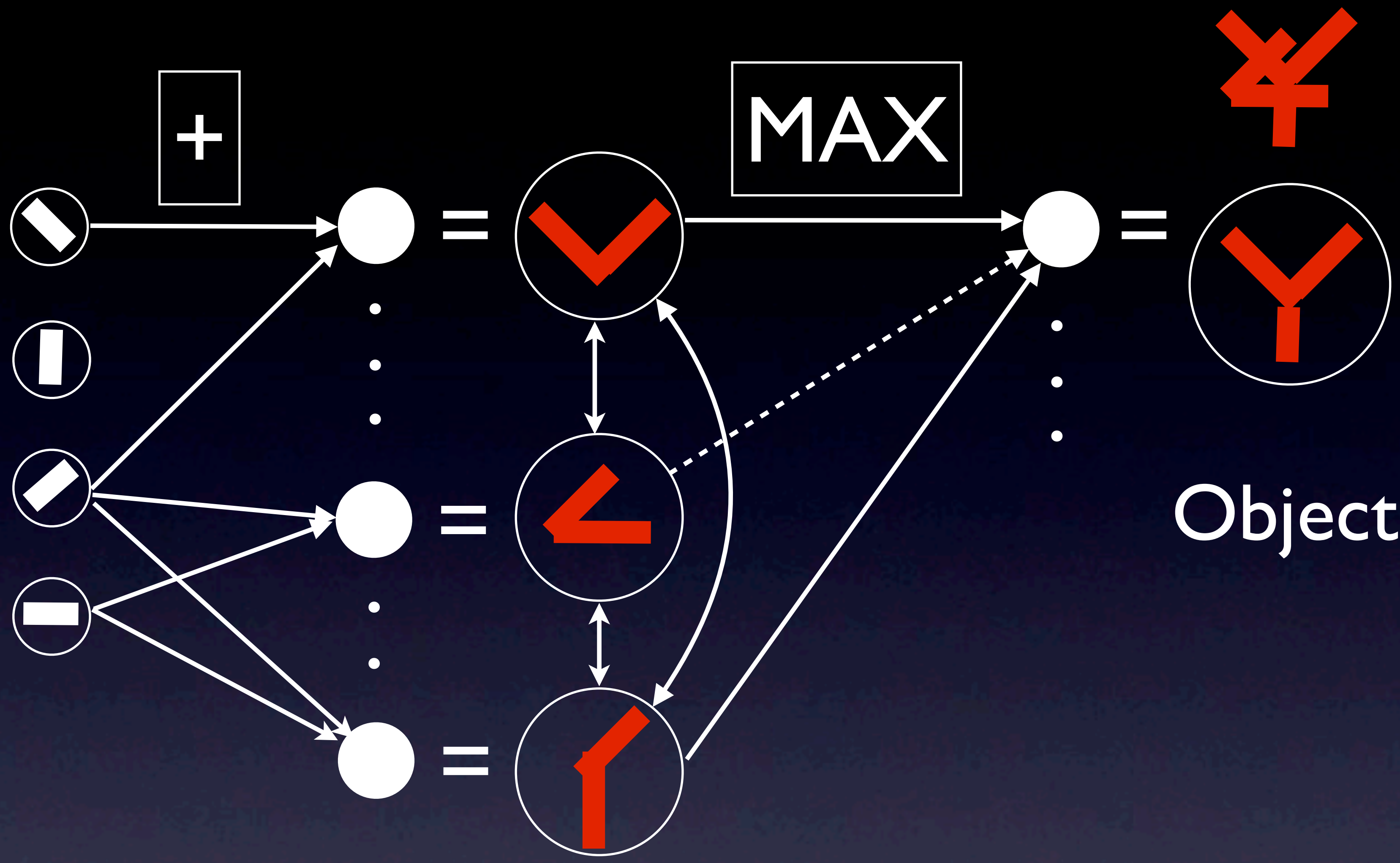




- Neurons sampling LGNs

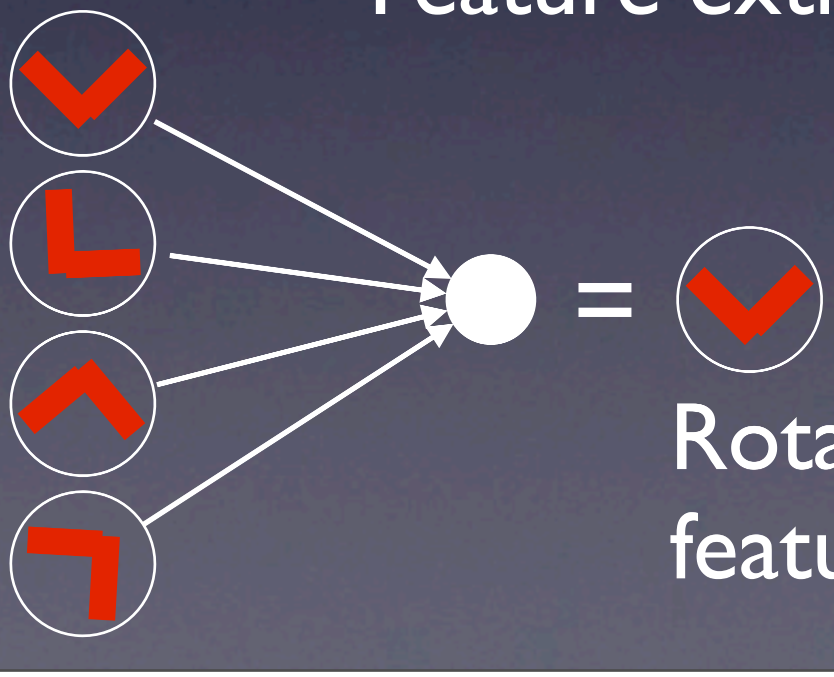
Each level has a “semantic”. LGN (visual sensory) cells: detect an illuminated pixel surrounded by darkness (on-off LGN).
A higher-level neuron sampling LGNs can, for instance, have the semantic of an illuminated diagonal segment surrounded by darkness.

Neuron-Level Model (contd)

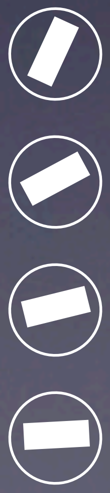


Object recognition

Feature extraction



Rotation-invariant feature

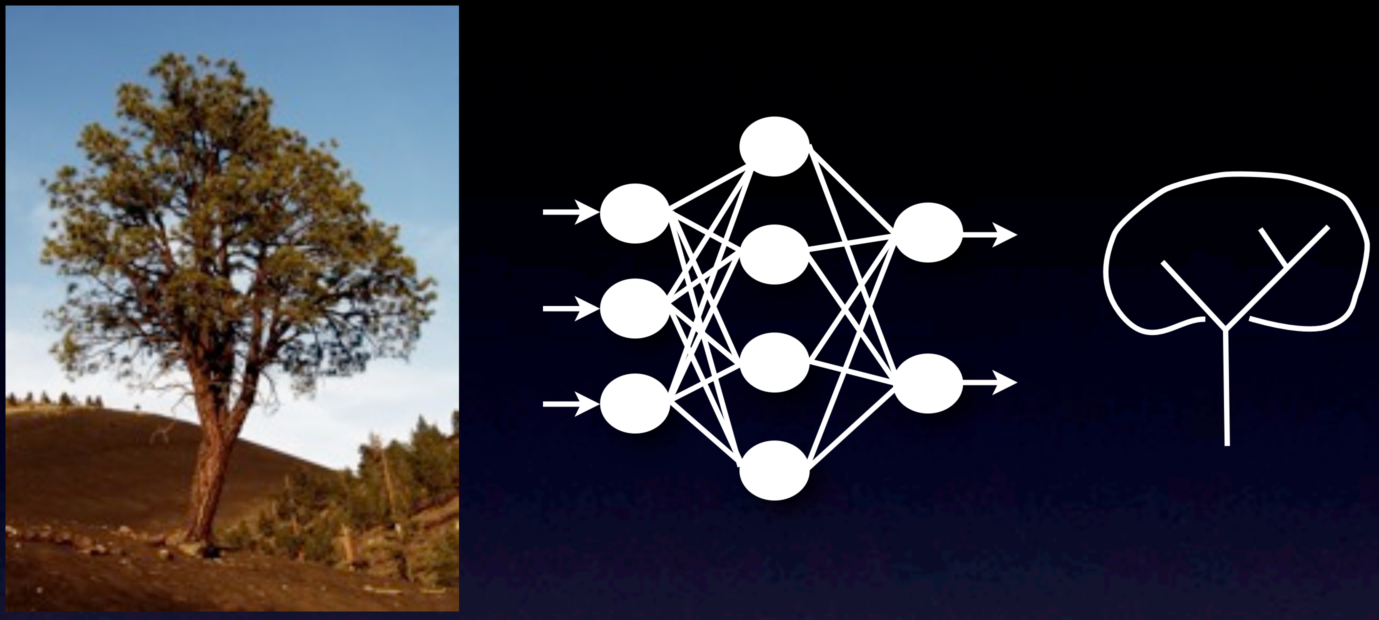


More neurons
=
more accuracy

Same principles when moving up the hierarchy: at next level, neurons sampling “segment-level” neurons have semantic of tiny shapes which are combinations of segments (see red shapes at 2nd-level, left of segment-level neurons). At next level, more complex shapes start to emerge, e.g., “Y”-shape evoking elementary tree shape.

Structure is more complex than just “additions”: each neuron can poll down 20 or more neurons, so “blob”-like shapes with no clear semantic can easily emerge if all neurons, including weak ones, are left to contribute (consider adding all three 2nd-level shapes, see above “Y”, in 3rd level). At every level: competition among neurons (lateral inhibitory connections), strong neurons silence weak ones ~ MAX operation. Results in shapes with crisp semantic.

Full Image Recognition, Only Neurons

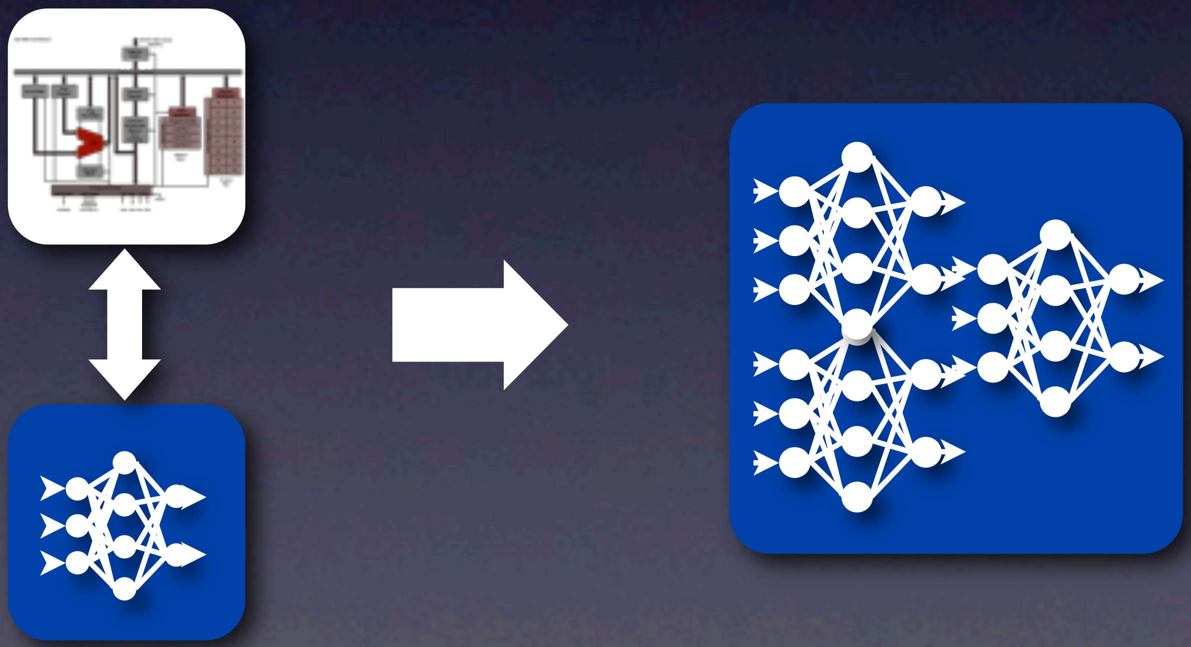


PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

Using Poggio's HMAX model

~ ranked 7/10

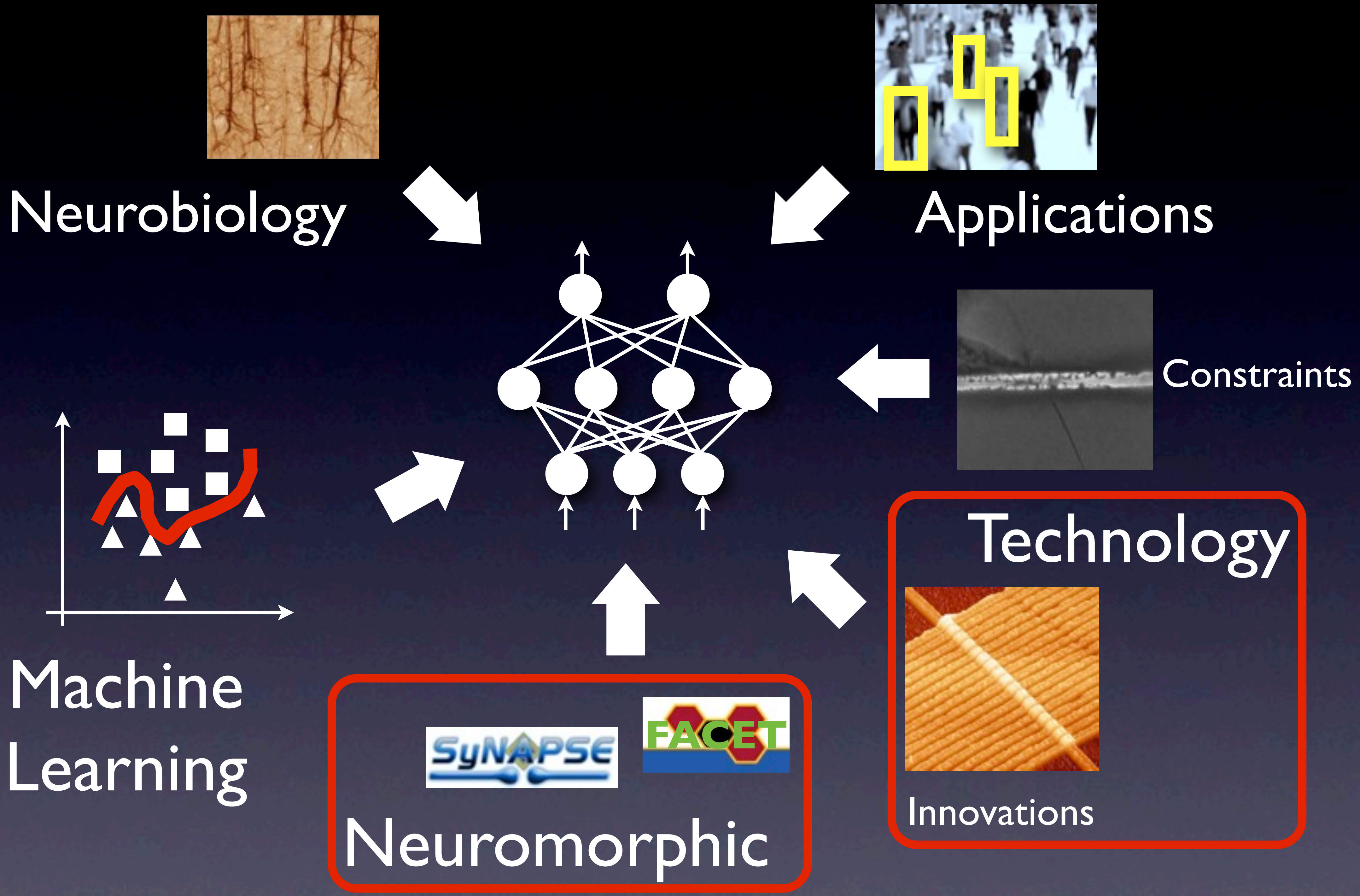
BerlinFIRSTNikon
CASIA_LinSVM
CASIA_NeuralNet
CASIA_NonLinSVM
ECPLIAMA
FIRST_SC1C
FIRST_SCST
INRIASaclay_CMA
INRIASaclay_MEVO
LEAR_flat
LEAR_shotgun
SurreyUvA_SRKDA
TKK_ALL_SFBS
TKK_MAXVAL
UvA_FullSFS
UvA_Soft5ColorSift
UvA_TreeSFS
XRCE



Neurobiology (realism) ⟷ Architecture (robustness)

Example: Poggio's HMAX model; such a bio-inspired approach can compare to state-of-the-art image recognition algorithms (tested in PASCAL challenge).
Poggio's model assumes very "regular" neurons organization and wiring: not "biologically realistic" enough, and does not fit hardware purposes (algorithm breaks if neuron/synapse breaks, which defeats hardware defects-tolerance purposes): achieve model capability using statistical connections ? Needs of neurobiology meet needs of architecture.
Back to hardware: now possible to implement FULL image recognition application using ONLY neurons. Therefore, power benefits and defects tolerance not only apply to "core algorithm" (mapped to hardware NN, rest of application on traditional core), but extend to full image application entirely mapped on hardware NN accelerator, provided hardware NN can accommodate large enough network.

Convergence of Trends



Neurobiology

Applications

Constraints

Machine Learning

Technology

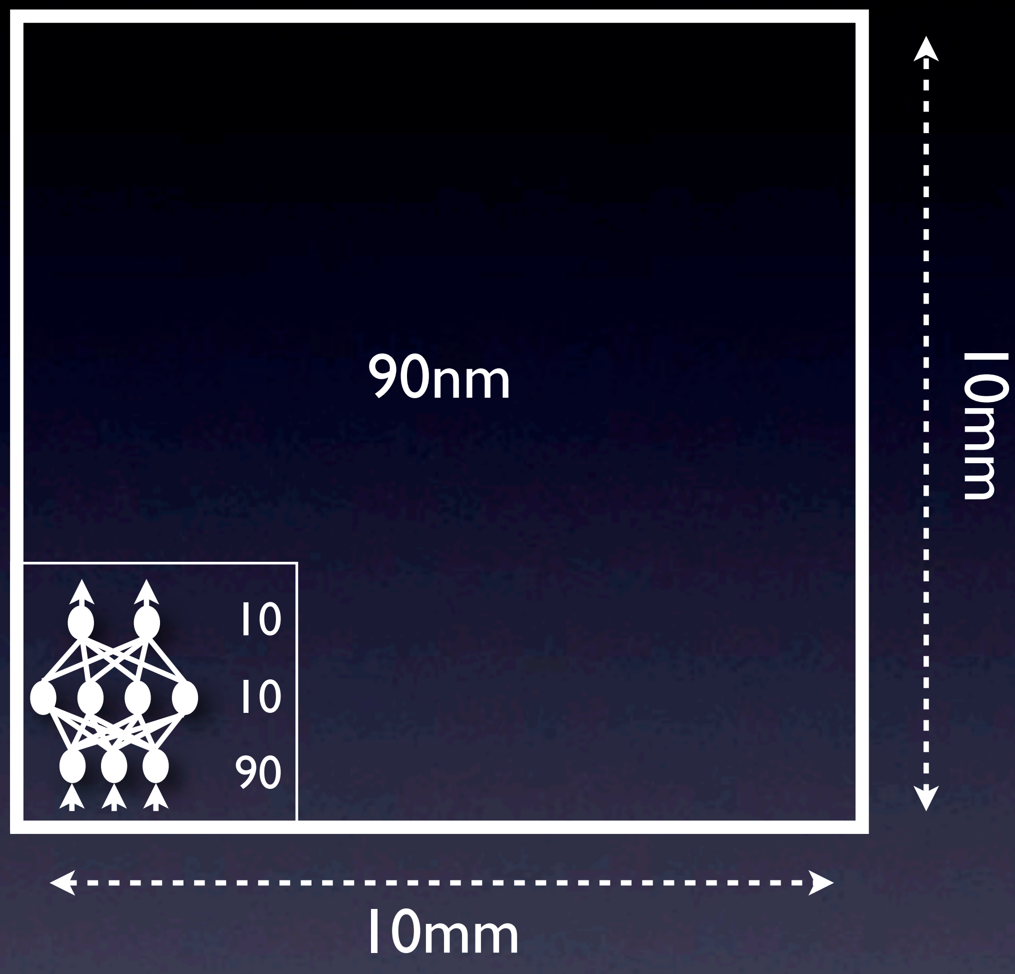
Innovations

Neuromorphic

with R. Heliot, A. Joubert (CEA)
S. Saïghi, J. Tomas (IMS), J. Grollier (CNRS/Thales)

Large networks are thus needed. How can we implement them ?

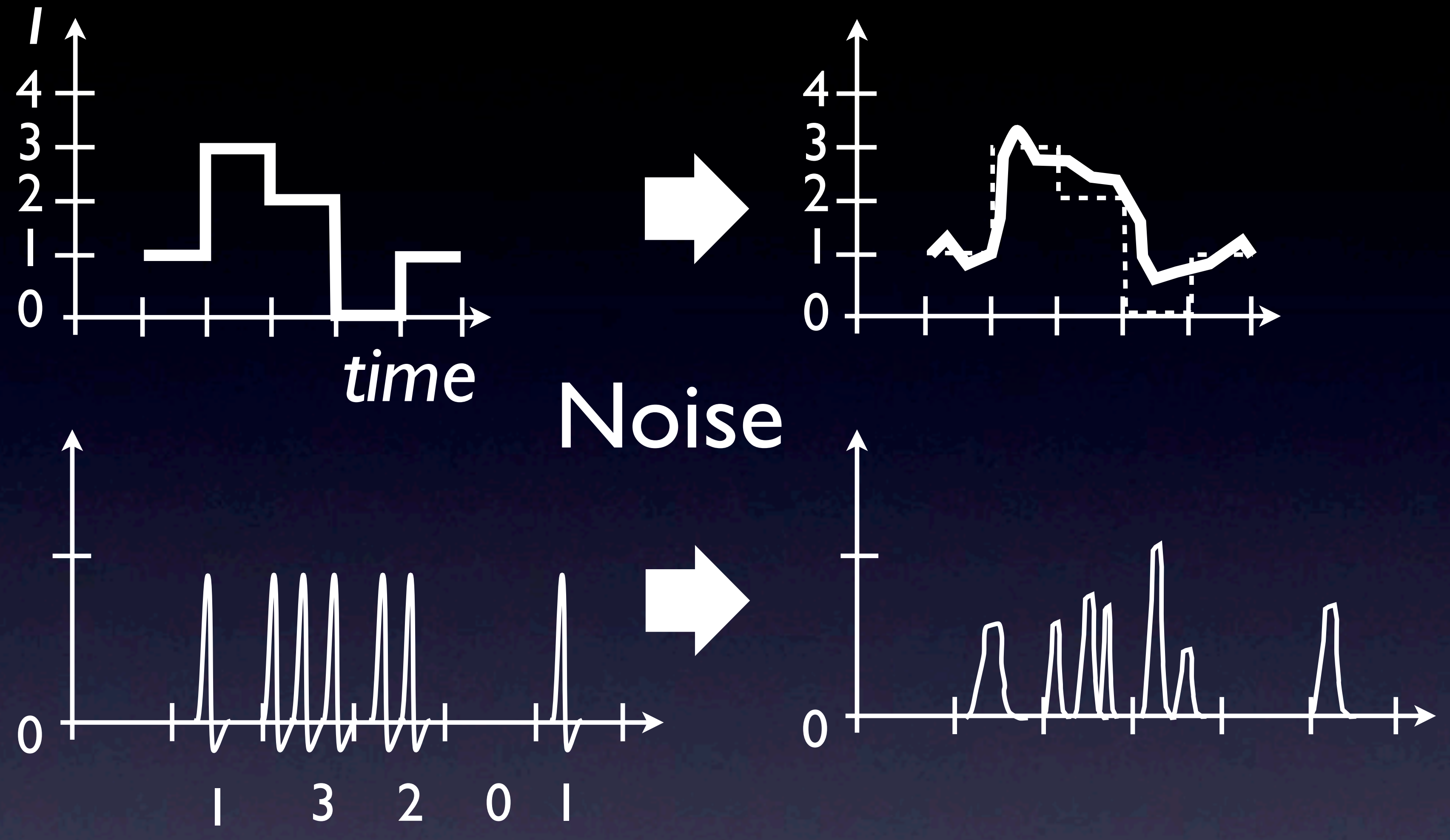
Size

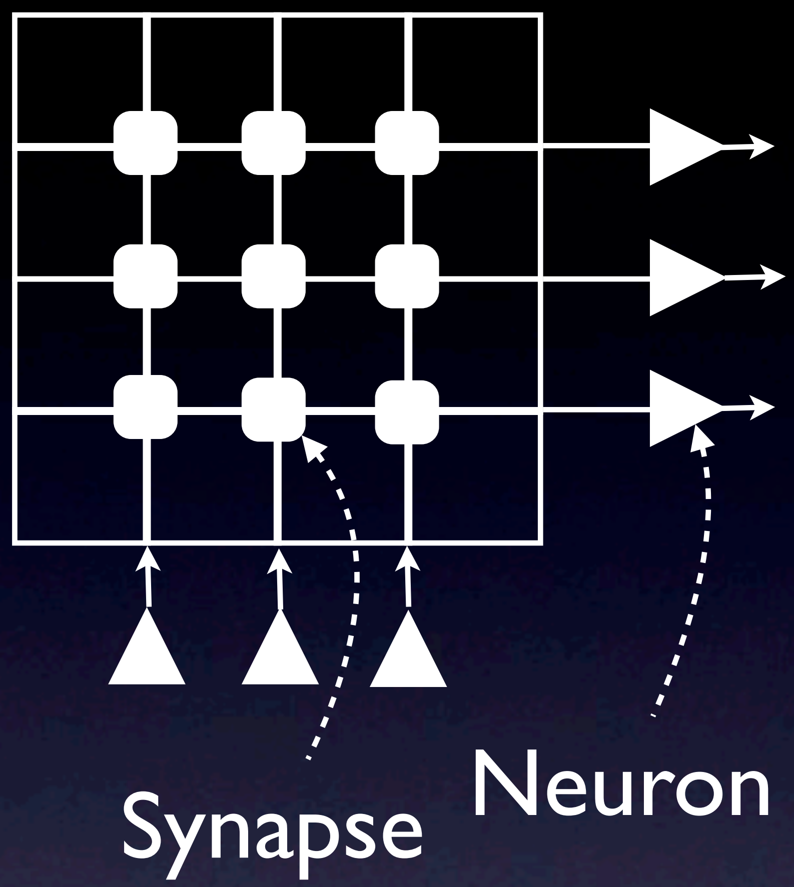


- 90 inputs, 10 hidden, 10 outputs

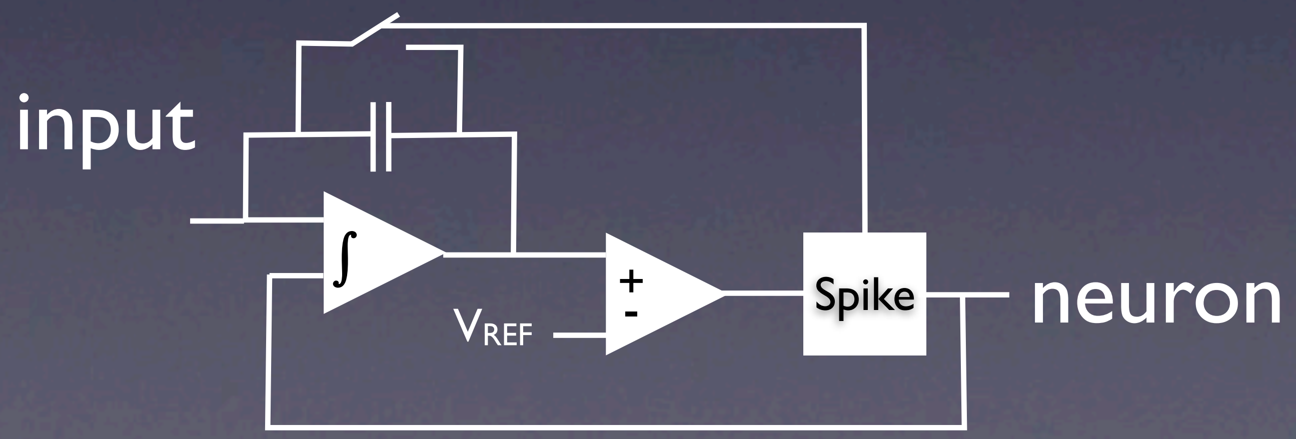
Digital CMOS has assets (same technology as current processors, readily compatible), allows implementation of useful hardware ANNs accelerators, but not dense enough for the largest Deep Networks or bio-inspired networks.

Analog Spiking Neuron





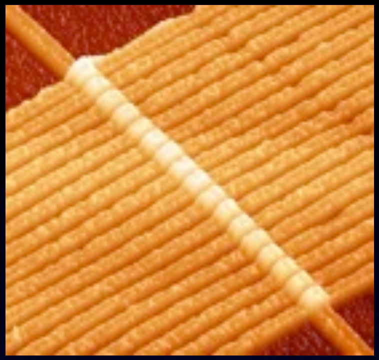
~ 14 transistors



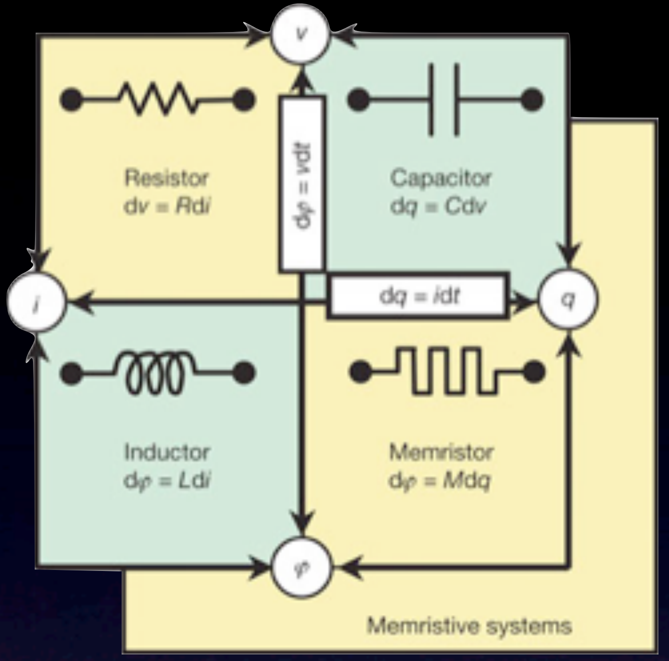
- Dense analog implementation
- Spikes more resilient to noise

Analog neurons: much denser implementations of operators (add, mult, activation function). Simple grid-like design possible (triangles are neurons, lower-layer bottom, upper-layer right; white squares are synapses). Most area now used by synapses (storage and multiplication). Analog spiking neuron: may be more resilient to noise (e.g., thermal noise) than current-coding implementations.

Synapses ? Memristors



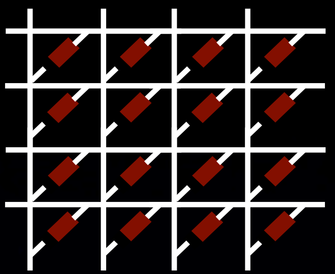
3nm x 3nm, 1ns

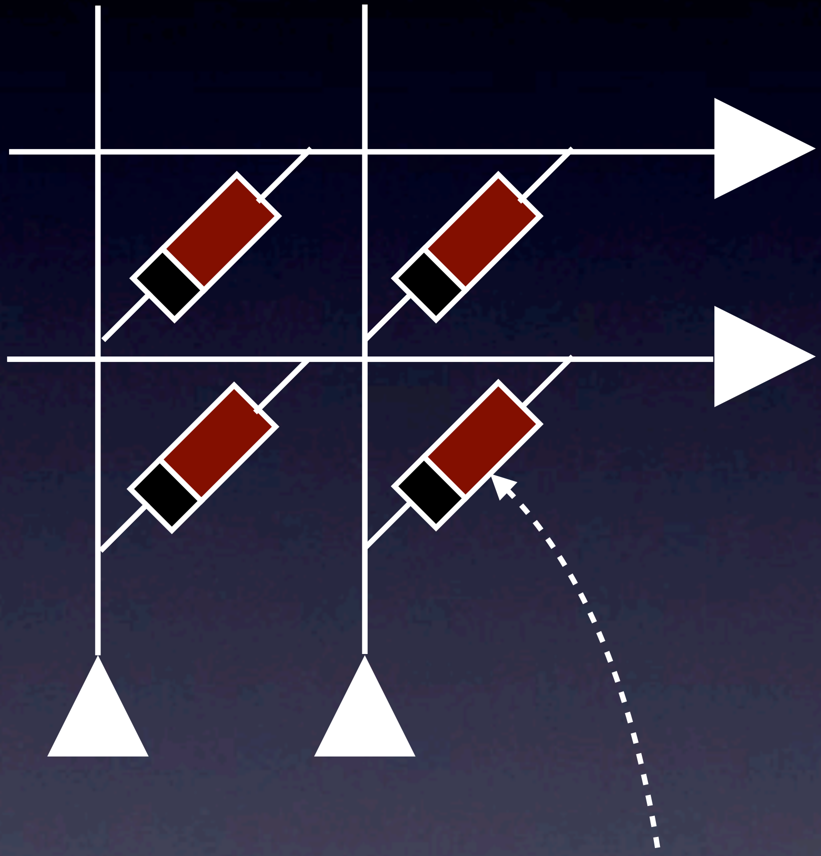


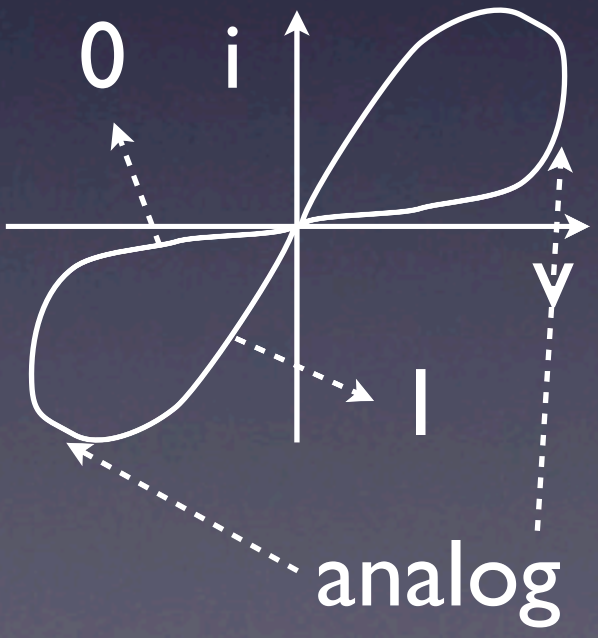
J.J.Yang/HP Labs

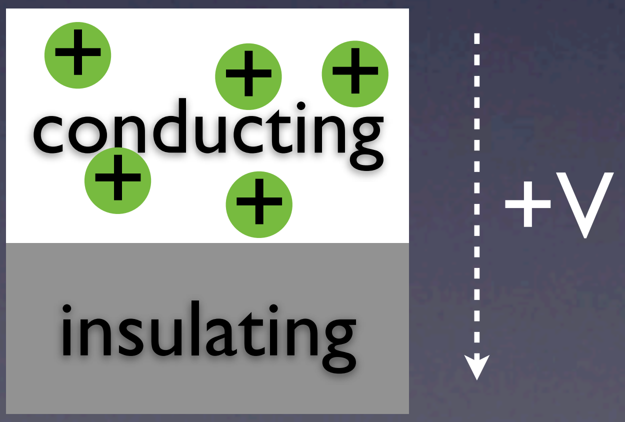
V = MxI, M = dΦ/dq

M: memristance







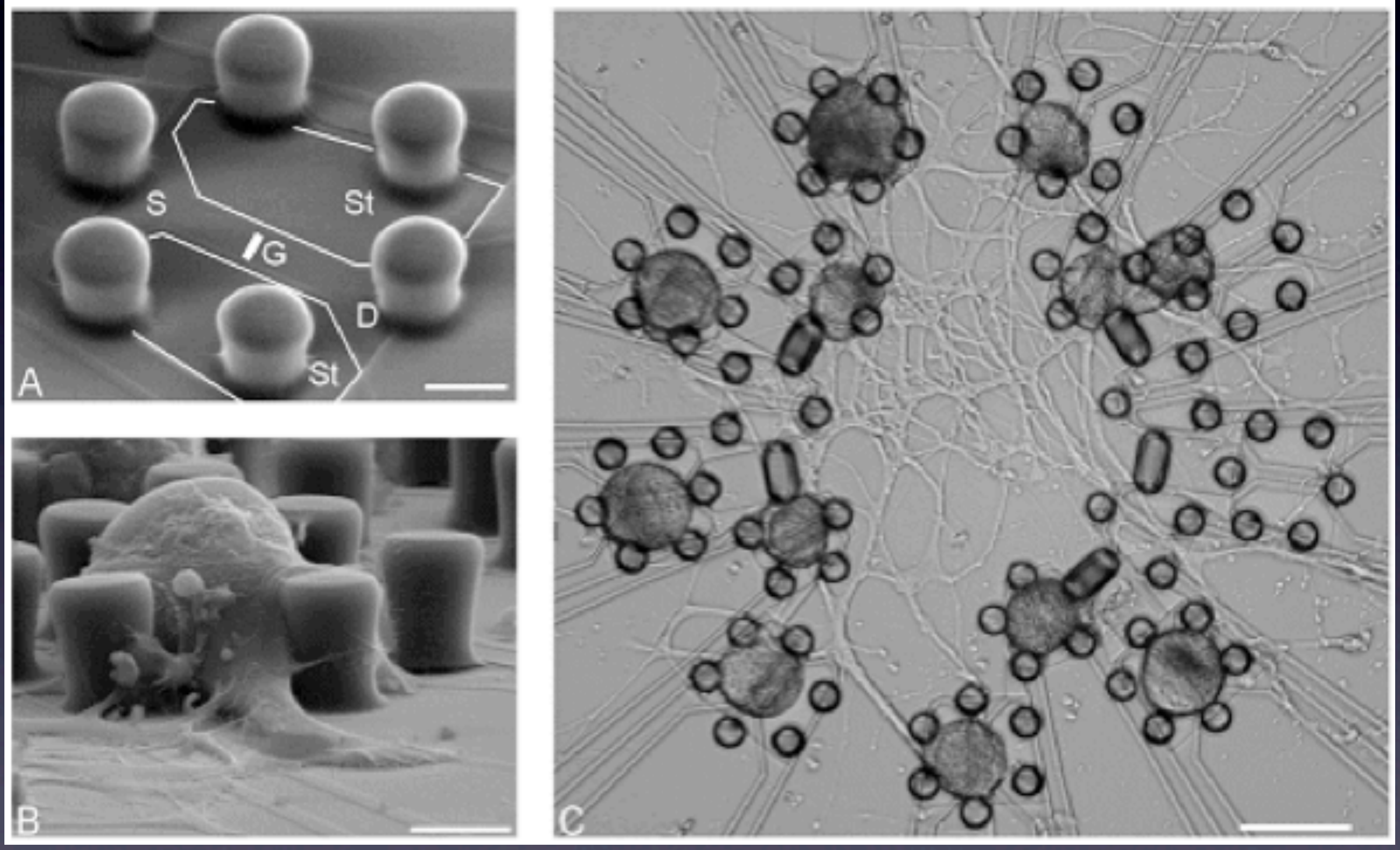


Synapse implementation can be made much more dense using memristors; recent device implementation by HP Labs (2008). Component almost ideally suited to hardware implementation of synapses (can be used as a switch, but also as analog storage; memristance is used to code synaptic weight).

Recent Ferroelectric Tunnel Junctions (2009) can allow to implement memristors with high endurance and low write power.

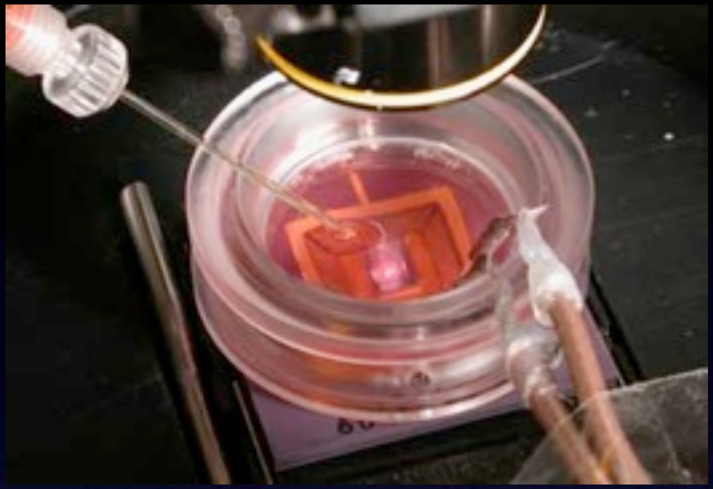
Analog neurons + memristors = very dense implementations of neural networks, opening up hardware NNs with broad application scope (see previous slides).

Beyond Silicon

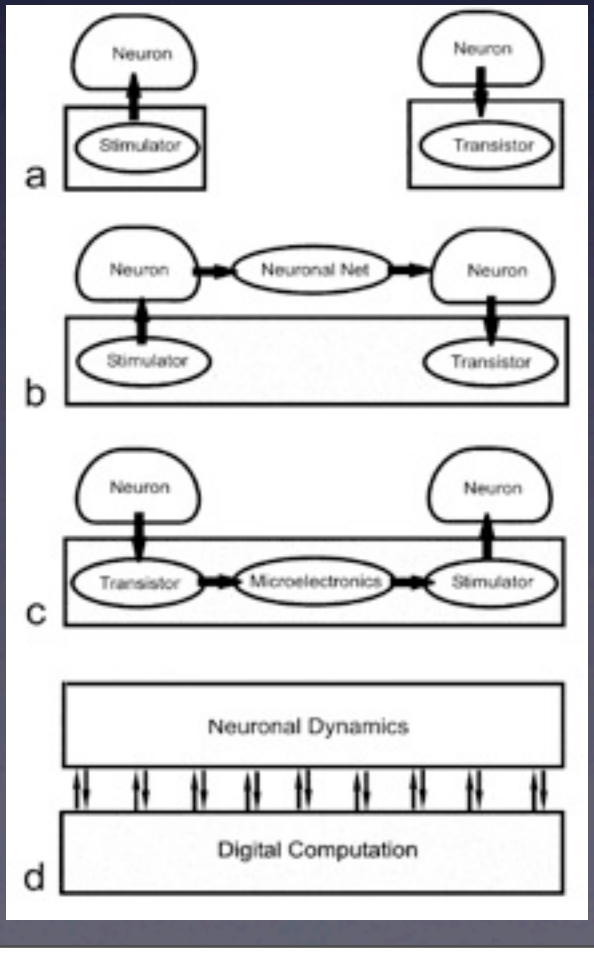


Individual transistor-neuron links

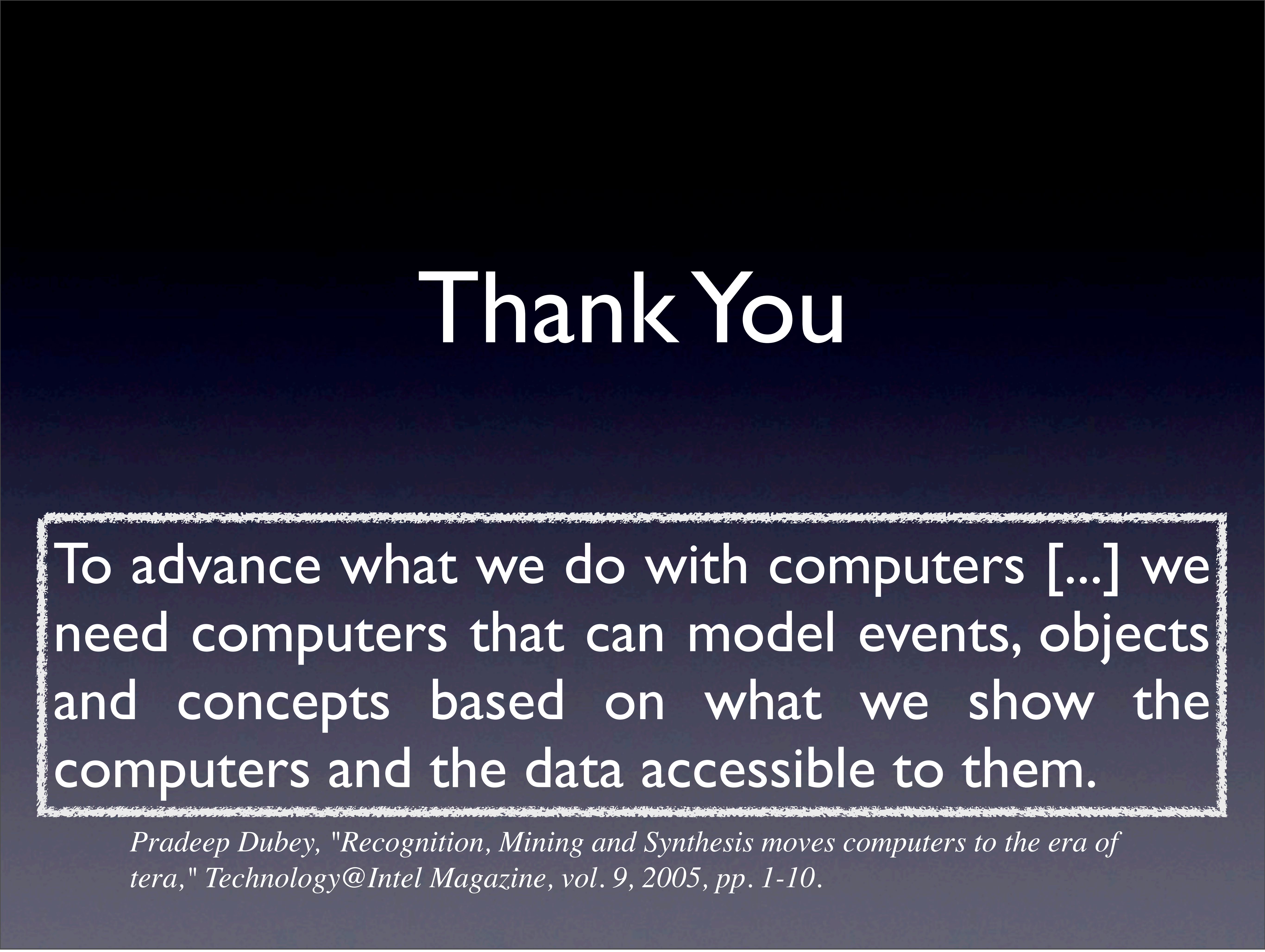
P. Fromherz



Infineon NeuroChip



And if not enough: why not directly use biological neural networks ? Not only allows large networks, but pretty cheap to implement too.
May not be so preposterous: Fromherz successfully created information loop between individual biological neurons (connected through biological synapses) and silicon transistor/stimulator pairs. Other groups have since then achieved similar capabilities. Infineon teamed with Fromherz to implement prototype NeuroChip for connecting whole layer of biological neurons with transistors.



Thank You

> To advance what we do with computers [...] we need computers that can model events, objects and concepts based on what we show the computers and the data accessible to them.

Pradeep Dubey, "Recognition, Mining and Synthesis moves computers to the era of tera," Technology@Intel Magazine, vol. 9, 2005, pp. 1-10.

Quote from Pradeep Dubey, author of RMS article. Disclaimer: I am not suggesting Pradeep mentioned neural networks in his article, he did not. Nonetheless, the quote refers to something pretty much in the spirit of “automatic abstraction of raw data into complex notions”...