CS 7960 Neuromorphic Architectures

Rajeev Balasubramonian
School of Computing, University of Utah

http://www.cs.utah.edu/~rajeev/cs7960

Lecture: Introduction

- Topics: landscape, terminology, motivation
- Logistics: lectures, project, exams, grading, canvas

Hardware Trends

Why the emphasis on accelerators?

This lecture focuses on the motivation for neuromorphic architectures; and the resulting landscape. Let's start with the hardware perspective. For the most part, industry has focused on designing general-purpose processors (like the kind in all our laptops/desktops). For decades, accelerators were not commercially attractive. The most notable exception is a GPU – it is a special-purpose architecture that has been customized for a narrow set of applications (graphics rendering). But in recent years, accelerators have become attractive. There are many reasons for this.

First, general-purpose architectures are not improving appreciably as technology is shrunk. This is because Dennard Scaling has ended. Dennard Scaling allowed voltage to shrink with transistor size; this was necessary to keep power in check as we built better/faster processors. But without Dennard Scaling, we need to find other ways to keep power in check. One is to not turn on all transistors on the chip (Dark Silicon), the other is to turn all transistors on at low clock speeds (Dim Silicon). So if a new processor is not much faster than an old processor, how can it be commercially attractive? By having a suite of accelerators on chip that can speed up (say) encryption, compression, sorting, rendering, dot-products, etc. Since applications will only use a subset of processors/accelerators at a time, such a heterogeneous architecture is also compatible with the Dark Silicon phenomenon.

Second, technology scaling will end altogether. So if performance of general-purpose processors stagnate (they will, because we've already picked all the low-hanging fruit), we'll have to find other ways to add value. Accelerators are the untapped potential.

Hardware Trends

- Why the emphasis on accelerators?
  - Stagnant single- and multi-thread performance with general-purpose cores
    - Dark silicon (emphasis on power-efficient throughput)
    - End of scaling
    - No low-hanging fruit
  - Emergence of machine learning

Accelerators consume silicon area; they require a brand new design cycle; in other words, they are expensive! They are therefore worthwhile only if many customers use them all the time. This wasn't true for many applications in the past. As machine learning emerges as a solution for "everything", it becomes a compelling target for hardware acceleration. For example, in 10 years, most cars will have self-driving capability. Each family will buy cars with tens of chips performing cognitive tasks on every drive – "many customers using acceleration all the time".

Neuromorphic Hardware

- The holy grail: emulating the brain
  - Low power – the brain consumes only 20 W
  - Fault tolerant – the brain loses neurons all the time
  - No programming required – the brain learns by itself
- Significant efforts in Europe, US, China (Human Brain Project, DARPA SyNAPSE)
- Implementations: SpiNNaker, Spikey, TrueNorth

The previous slide made an argument for neuromorphic architectures from a hardware and applications perspective. Here's an alternative argument. As a new Ph.D. student, you may want to consider solving (or making a dent in) a grand challenge problem – "building a computer that is comparable to the human brain". Why not? The brain has amazing computational power, consumes only 20W, programs itself, and even re-programs itself when neurons fail. Clearly, the brain has figured out a very compelling architecture. So we could embark on building a computer that imitates the brain's processes – maybe that'll give us high energy efficiency, high fault tolerance, and high levels of cognition. So, agnostic to recent machine learning advances, it may be exciting to just build a neuromorphic (brain-like) architecture and see what emerges.

Commercial Hardware for ML

Google TPU (inference and training)

Recent NVIDIA chips (Volta)

Microsoft Brainwave and Catapult

Intel Loihi and Nervana

Cambricon

Graphcore (training)

Cerebras (training)

Groq (inference)

ARM

Western Digital

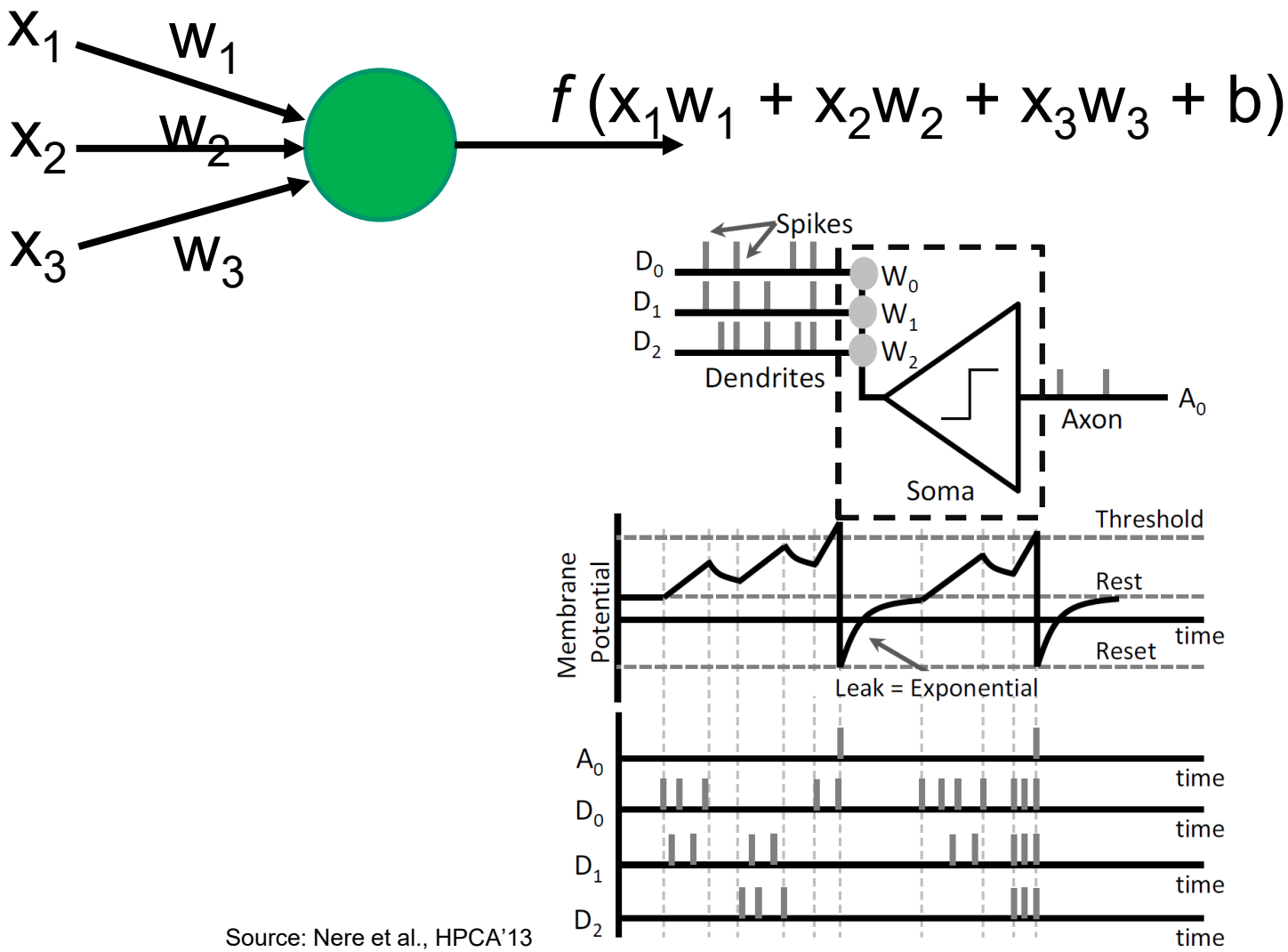
Tesla (FSD)

Taxonomy

- ANN vs. SNN: ML vs. Neuroscience, training via back-prop or via biological processes
- Analog vs. Digital architectures
- Chips for inference vs. training
- Systolic vs. Tensors

We'll discuss all neuromorphic architectures – essentially any architecture that is inspired by neuron behavior. There are several implementation styles (see above). Let's start by considering the differences between the two approaches we've already introduced: the one driven by ML algorithms and the one driven by brain architectures.

ANN vs. SNN



Source: Nere et al., HPCA'13

ANN vs. SNN

ANN

Most machine learning algorithms are based on a “perceptron” or “artificial neuron”.
The neuron has no “state”. It receives synchronous inputs, it performs its math, and produces an output. In the next cycle, it receives new inputs, and it starts all over again with no memory of what it did in the previous cycle. (Of course, the weights in the neuron represent the “memory” of the training samples.)
What is the math? Let’s say that the neuron has 3 inputs x1, x2, x3. The inputs are multiplied by weights (determined by training) w1, w2, w3. These are added up to produce the value z = x1*w1 + x2*w2 + x3*w3. We are essentially measuring the “strength” of the weighted inputs to decide if this neuron should fire or not. That is, is the weighted sum greater than a threshold or not? One option is to produce an output of 1 if yes, or an output of 0 if no. This is referred to as an activation function, producing a = f(z). (Note that the threshold or bias b is included in z.) Since the above is a non-continuous activation function, many use a sigmoid function instead. Sigmoid caps the value of a, so people are now instead using ReLU as an activation function. Training is performed with back-propagation with gradient descent. You start with a number of labeled training samples, i.e., a list of inputs and expected outputs. Starting with random weights, you compute your outputs for your inputs, and then measure the distance between your outputs and the expected (labeled) outputs. This distance (or cost function C) should be minimized. Computing the value of dC/dw1 tells us if increasing w1 will increase or decrease C. You accordingly tweak w1 so C is reduced. Similarly, do this for w2, w3, and b.

ANN vs. SNN

SNN

Spiking neurons are designed to resemble the chemical reactions in our brains. A neuron has a certain "potential" that represents the inputs it has previously received. When the neuron receives more inputs, its potential rises or falls, depending on the relative importance of those inputs. If the neuron does not receive any inputs, its potential starts to leak away. This is because the potential is determined by ions that are received from the inputs or that leak out of the neuron. When the potential reaches a threshold, the neuron fires and its potential comes down to a resting potential. Note that all inputs/outputs are in the form of binary spikes. Two neurons are connected through a synapse that either amplifies or de-amplifies the spike as it enters the second neuron (this is essentially the synaptic weight that determines the importance of that input).

Spiking neurons are attractive because communication between neurons is in the form of binary spikes, which are much more efficient than moving 8-bit or 16-bit numbers (as is done in ANNs). They also result in efficient computations because the neuron computation does not involve multiplications (you just increment the potential by the synaptic weight) or complex activation functions. Because a neuron has state, it is potentially a more powerful construct, i.e., it may be useful in applications that have a notion of time, e.g., video analysis, language analysis. Also, because a neuron's firing depends on the relative timing of input spikes, there is the potential for the inputs to carry a large amount of information in a few bits. All of these properties are perhaps critical in helping the brain achieve its high energy efficiency.

Training is performed with a technique called STDP (more on this later). In a nutshell, it strengthens an input if that input led to the neuron spiking. This is a form of unsupervised learning.

ANN vs. SNN

- Perceptron (ANN): well-understood, high accuracy
- Spiking neural network: infant technology (relatively), potential for low-energy computations and low-energy communication, potential for high information content
- Potential convergence: brain-inspired ideas being injected into machine learning (RNNs, few bits per value), and machine learning techniques (back-prop) being used to train brain models

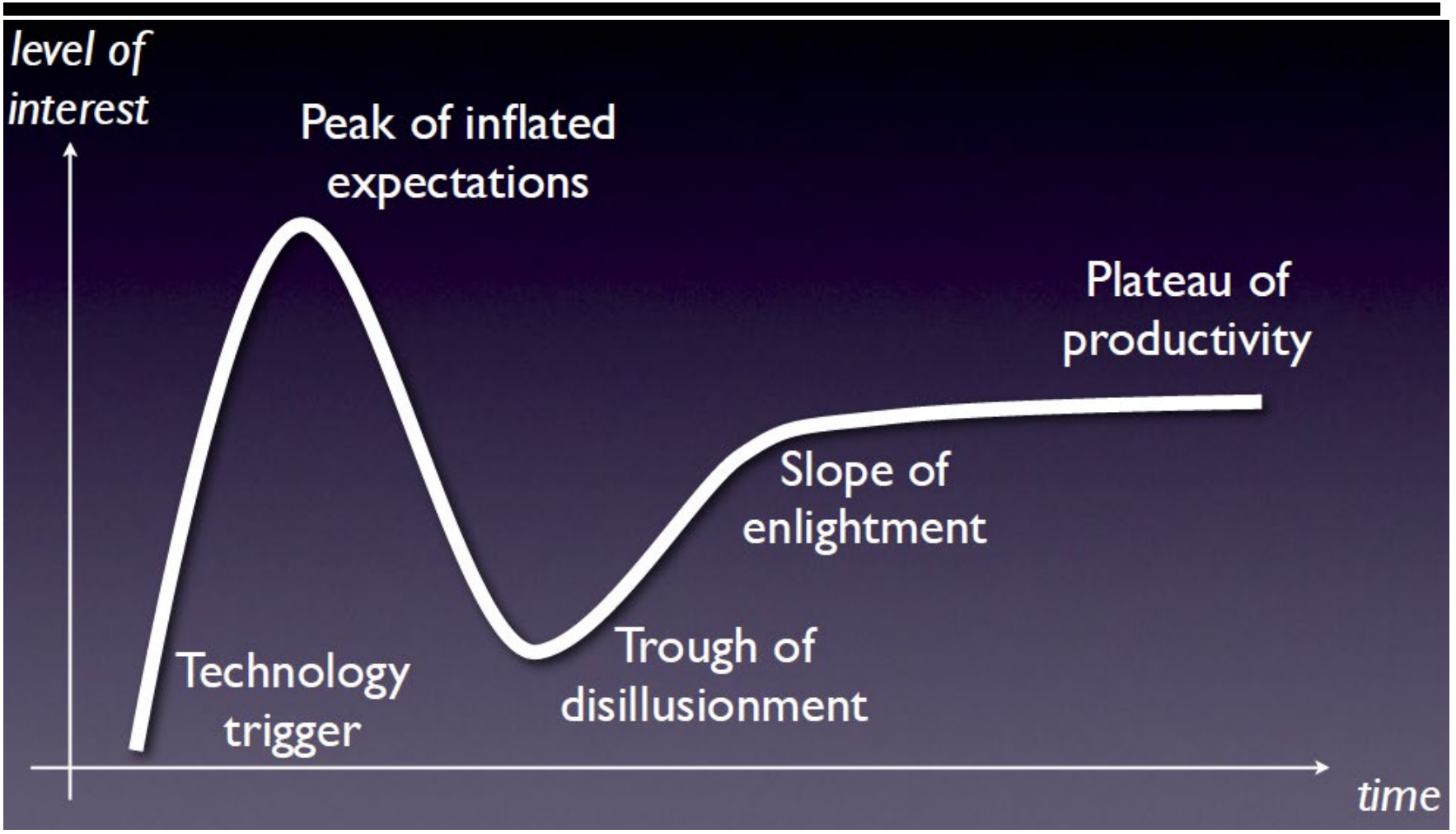
Today, SNNs don't achieve very high accuracies on cognitive applications because the theory is not as well understood. We see a cross-fertilization of ideas between the two fields. For example, recent papers have done back-propagation on SNNs, and conversely, some ANNs use feedback loops to create recurrent neural networks that do retain "state".

Analog vs. Digital

- A single analog device can perform multiple multi-bit operations
- But, analog has challenges, e.g. noise/precision
- Cognitive applications amenable to analog because of their noise tolerance

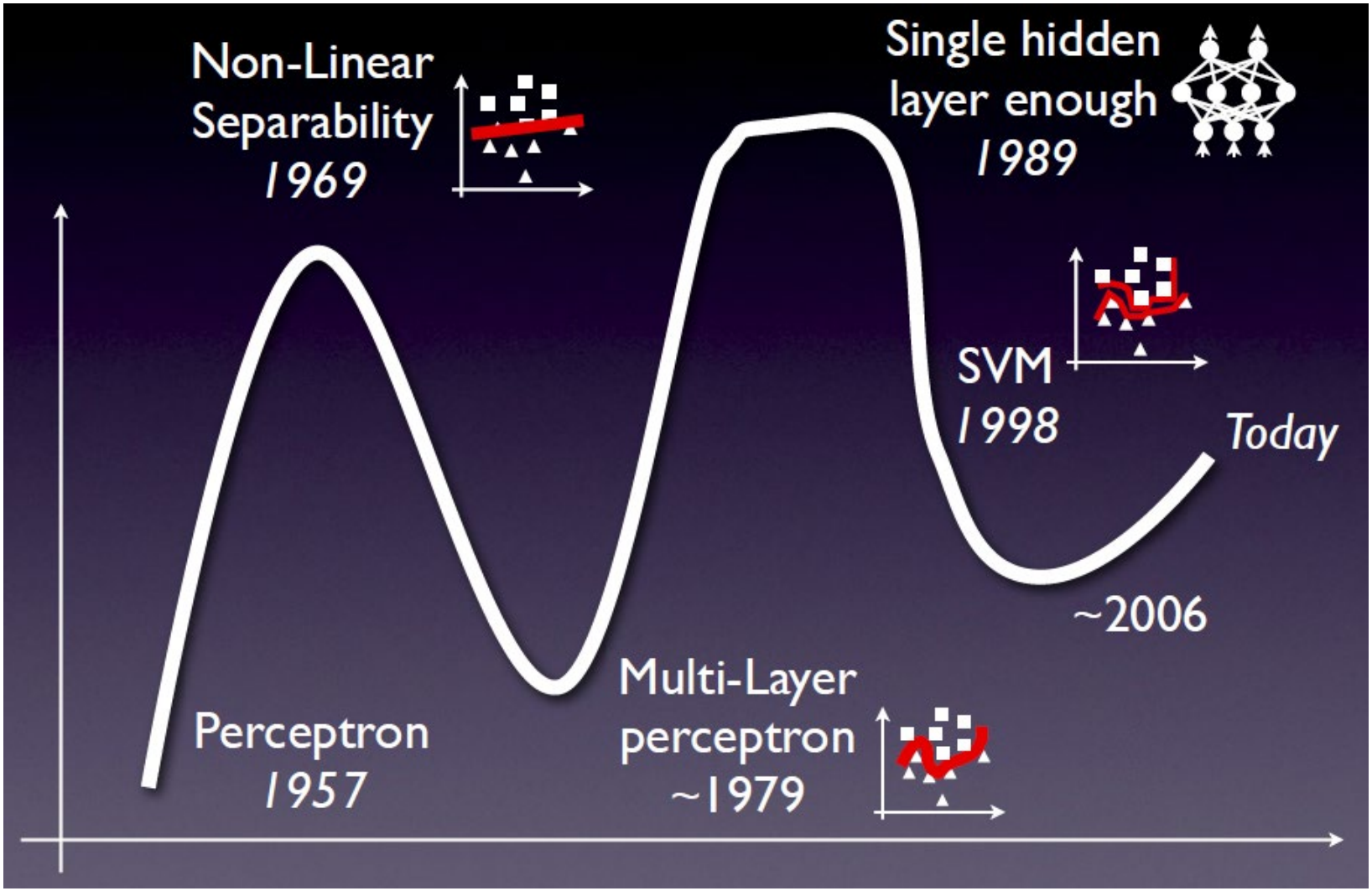
Just as the use of ANN vs. SNN is a distinguishing feature of a neuromorphic architecture, the use of analog vs. digital circuits is another important distinguishing feature. Digital circuits use standard CMOS transistors and gates for most of their computations, and exclusively deal with 0s and 1s (in other words, the computers that we are all familiar with). Analog circuits use physical phenomena to carry information, e.g., the current in a wire or the charge in a capacitor can represent a rational number. You can perform addition by merging the currents in two wires. Multiplication can be represented by the current that emerges when a voltage is applied to a conductor. Thus, an analog circuit can offer low-power computation over several bits worth of information. But they are a poor choice for "regular" computers. As temperature changes, currents change – this type of instability is unacceptable for financial transactions, but may be tolerated by neural networks.

Gartner Hype Curve



Every grad student should be aware of this standard hype curve. The trough of disillusionment, while not helpful to job-seekers, is only a sign of "interest". Most of the really impactful work happens in the slope of enlightment.

NN Hype Curve



Older Commercial Examples

Analog

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Intel ETANN	Feedforward, ML	No	6×6 bits	64	10280	2 GCPS
Synaptics silicon retina	Neuromorphic	No	N.A.	48×48	Resistive net	N.A.

Digital

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Slice architectures						
Micro devices MD-1220[a]	Feedforward, ML	No	1×16 bits	8	8	1.9 MCPS
NeuraLogix NLX-420 [a]	Feedforward, ML	No	1–16 bits	16	Off chip	300 CPS
Philips Lneuro-1	Feedforward, ML	No	1–16 bits	16 PE	64	26 MCPS
Philips Lneuro-2.3	N.A.	No	16–32 bits	12 PE	N.A.	720 MCPS
SIMD						
Inova N64000[a]	GP, SIMD, Int	Program	1–16 bits	64 PE	256k	870 MCPS 220 MCUPS
Hecht-Nielson HNC 100-NAP[b]	GP,SIMD,FP	Program	32 bits	4 PE	512k Off chip	250 MCPS 64 MCUPS
Hitachi WSI	Wafer, SIMD	Hopfield	9×8 bits	576	32k	138 MCPS
Hitachi WSI	Wafer, SIMD	BP	9×8 bits	144	N.A.	300 MCUPS
Neuricam NC3001 TOTEM	Feedforward, ML, SIMD	No	32 bits	1–32	32k	1 GCPS
Neuricam NC3003 TOTEM	Feedforward, ML, SIMD	No	32 bits	1–32	64k	750 MCPS
RC Module NM6403	Feedforward, ML	Program	$1–64 \times 1–64$ bits	1–64	1–64	1200 MCPS

Ref: “Artificial Neural Networks: A Review of Commercial Hardware”, Dias et al., Elsevier, 2004.

Early Bust

Commercial NN chips fell out of favor in the early 2000s

- SVMs overtook ANNs
- General-purpose processors were getting faster every year and quickly overtaking ASICs
- Limited market for machine learning

Note that none of the above is applicable today

According to Olivier Temam, the many neural network chips (see previous slide) faded away because neural networks ceased to be fashionable. Support vector machines took over, general-purpose processors and GPUs were quickly outpacing ASICs, and machine learning was not being used everywhere. Today, NNs > SVM, GPPs and GPUs will stagnate in performance, and ML is hot!

Early Bust

Commercial NN chips fell out of favor in the early 2000s

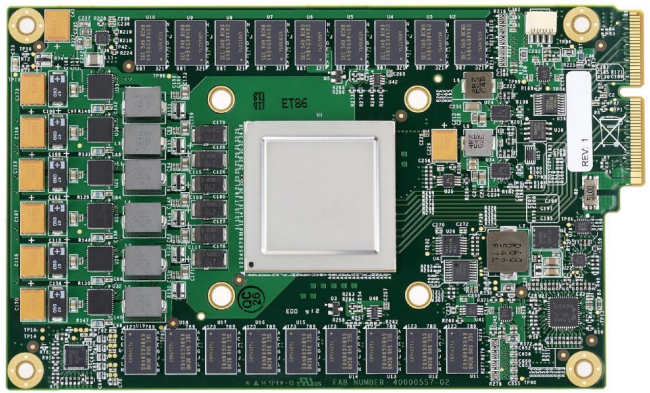
- SVMs overtook ANNs
- General-purpose processors were getting faster every year and quickly overtaking ASICs
- Limited market for machine learning

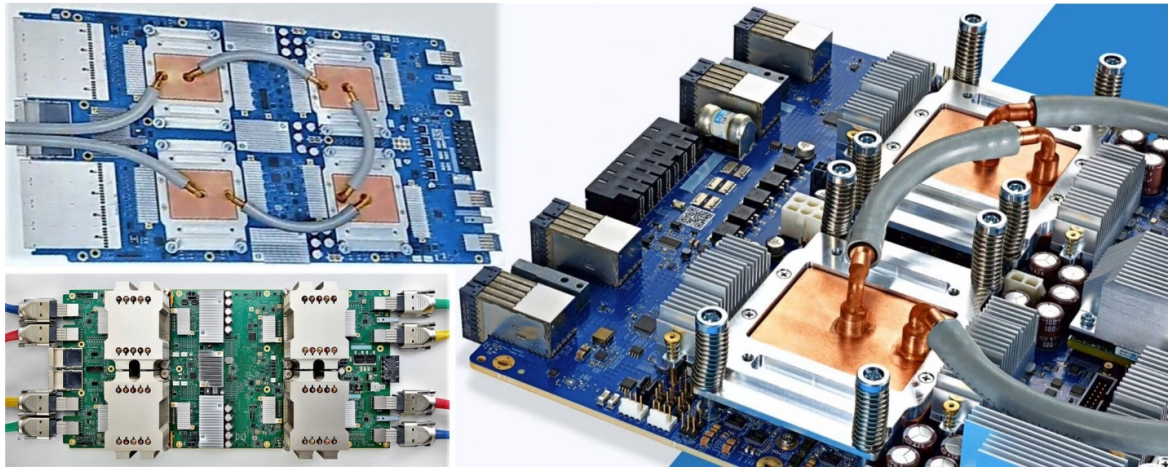
Note that none of the above is applicable today

In summary, ML accelerators are very important today because of the simultaneous excitement in both accelerators (from a hardware perspective) and ML (from a software perspective). There are many implementation options. Newer chips will benefit from emerging technologies and they will likely not suffer from the same problems that plagued chips in earlier decades. Neuroscience advances may also reveal new approaches for energy-efficient cognition.

Google TPU

- Version 1: 15-month effort, basic design, only for inference, 92 TOPs peak, 15x faster than GPU, 40 W 28nm 300 mm^2 chip
- Version 2: designed for training, a pod is a collection of v2 chips connected with a torus topology
- Version 3: 8x higher throughput, liquid cooled





Ref: Google

References

- "Neuromorphic Computing: The Machine of a New Soul", The Economist, August 2013
- "Artificial Neural Networks: A Review of Commercial Hardware", Dias et al., Elsevier, 2004.
- "The Rebirth of Neural Networks", Olivier Temam, Keynote at ISCA'10
- Chapter 1 of Nielsen's book: http://neuralnetworksanddeeplearning.com/chap1.html