Chapter

Cortical Columns Computing Systems: Microarchitecture Model, Functional Building Blocks, and Design Tools

John Paul Shen and Harideep Nair

Abstract

Reverse-engineering the human brain has been a grand challenge for researchers in machine learning, experimental neuroscience, and computer architecture. Current deep neural networks (DNNs), motivated by the same challenge, have achieved remarkable results in Machine Learning applications. However, despite their original inspiration from the brain, DNNs have largely moved away from biological plausibility, resorting to intensive statistical processing on huge amounts of data. This has led to exponentially increasing demand on hardware compute resources that is quickly becoming economically and technologically unsustainable. Recent neuroscience research has led to a new theory on human intelligence, that suggests Cortical Columns (CCs) as the fundamental processing units in the neocortex that encapsulate intelligence. Each CC has the potential to learn models of complete objects through continuous predict-sense-update loops. This leads to the overarching question: *Can we build Cortical Columns Computing Systems (C3S) that possess brain-like capabilities as well as brain-like efficiency?* This chapter presents ongoing research in the Neuromorphic Computer Architecture Lab (NCAL) at Carnegie Mellon University (CMU) focusing on addressing this question. Our initial findings indicate that designing truly intelligent and extremely energy-efficient C3S-based sensory processing units, using off-the-shelf digital CMOS technology and tools, is quite feasible and very promising, and certainly warrants further research exploration.

Keywords: neuromorphic computing, neocortical columns, spiking temporal neural networks, online continual learning, intelligent sensory processing

1. Introduction

The field of deep learning (DL) has seen extraordinary progress over the last decade and has established itself as the de facto standard technology for sensory processing tasks such as visual object recognition/detection, audio/time-series signal processing, natural language processing, etc. However, this progress has thrived with heavy dependence on increasing hardware resources, and power and energy consumption. These hardware resources such as CPUs, GPUs and specialized accelerators primarily employ Turing computation model and von-Neumann computer architecture. Such computation model and architecture were originally developed for numerical and symbolic processing tasks that were difficult for humans. In contrast, human neocortex is highly efficient in sensory processing and pattern recognition and capable of online continual learning. Current deep learning is attempting to replicate human-like sensory processing capability using conventional Turing-von Neumann computing machines.

It has been observed by Hans Moravec and others that these two computation models and systems are quite different and distinct; this observation is known as the "Moravec's Paradox" [1–3]. The current overarching grand challenge is: Can we design much more energy-efficient computing hardware systems for human-like sensory processing with continuous learning capability by mimicking the architecture, organization, and operation of the human neocortex? This grand challenge is not new and has been around for decades. In 1990, Carver Mead first coined the term "neuromorphic computing" [4] for this grand challenge. But the same notion and aspiration can be traced back to Frank Rosenblatt's "perceptrons" from the late 1950's [5]. This chapter describes our particular approach and strategy in pursuing this grand challenge as part of the current resurgence of interest in neuromorphic computing [6].

Our research builds on the seminal works by James E. Smith on temporal neural networks (TNNs) [7, 8] and is strongly influenced by Jeff Hawkins' recent book "A Thousand Brains: A New Theory of Intelligence" [9]. Unlike convolutional neural networks (CNNs), TNNs are temporal spiking neural networks that embrace a strong adherence to biological plausibility [10]. TNNs encode and process information in temporal form mimicking the brain's neocortical sensory signal processing. We developed a microarchitecture model for implementing highly efficient TNN designs [11, 12] and demonstrated state-of-the-art clustering performance on a wide variety of time-series signals [13].

Hawkins' new theory on intelligence [9], informed by extensive neuroscience research, suggests Cortical Columns (CCs) as the key compute units within the human neocortex. The neocortex gains its intelligence through CC's ability to model sensory information in structured Reference Frames (RFs), and continuously update its models as the sensor interacts with the environment. Each CC is computationally powerful and can learn any specific task. There is great synergy between CCs and TNNs.

Our current research focuses on extending the TNN design and implementation framework to incorporate CC attributes. Unlike artificial neural networks that separate training and inference, CCs store and process information in RFs, support online continuous learning, and can dynamically adapt to sensory input changes. Sensory processing units built based on such CCs, can be truly "intelligent" as per Hawkins' definition, and can enable contextualization and personalization of applications and services for supporting edge AI on diverse mobile and wearable devices.

This chapter presents a framework for designing and implementing *Cortical Columns Computing Systems (C3S)*. This framework consists of three major components: (1) a microarchitecture model for designing and implementing cortical columns and CC-based computing systems using off-the-shelf digital CMOS technology; (2) a suite of specialized functional building blocks to implement application-specific CC-based processing units with significant improvements on Power-Performance-Area (PPA) efficiency; and (3) a PyTorch-based software simulator tool for rapid design space exploration targeting specific applications; and a design synthesis tool for direct CMOS implementation of special-purpose C3S sensory processing units in the form of chiplets, potentially supporting the latest Universal Chiplet Interconnect Express (UCIe) [14] standard. This chapter presents the progress we have made on C3S and our future research directions.

2. Background and motivation

2.1 Unsustainable trend for DL compute

Deep neural networks (DNNs) [15–18] have advanced state-of-the-art in a plethora of applications, particularly those mimicking human sensory processing tasks such as image recognition, object detection, time-series signal (e.g., speech) processing etc. However, due to exponentially growing models performing high-dimensional tensor processing and global gradient backpropagation, their compute requirements are scaling unsustainably. Specifically, a study by OpenAI in 2018 [19] illustrated that DNN training compute is doubling every 3.4 months. Comparing this with hardware driven by Moore's Law doubling every 24 months, the gap between compute demands and hardware supply is increasing at an exponential rate of 8x per year or 500x every 3 years (see **Figure 1**). There is also growing evidence that the current trajectory of deep learning compute scaling is economically, technologically and environmentally unsustainable [20–22]. On a more optimistic note, some other recent studies have claimed that the rate of DNN model explosion has slowed down between 2018 and 2020 [23, 24], and that the trend is expected to plateau and shrink in the future [25].

There are significant ongoing efforts to deal with this DNN compute complexity explosion. Reducing data value precision through quantization and DNN model sizes

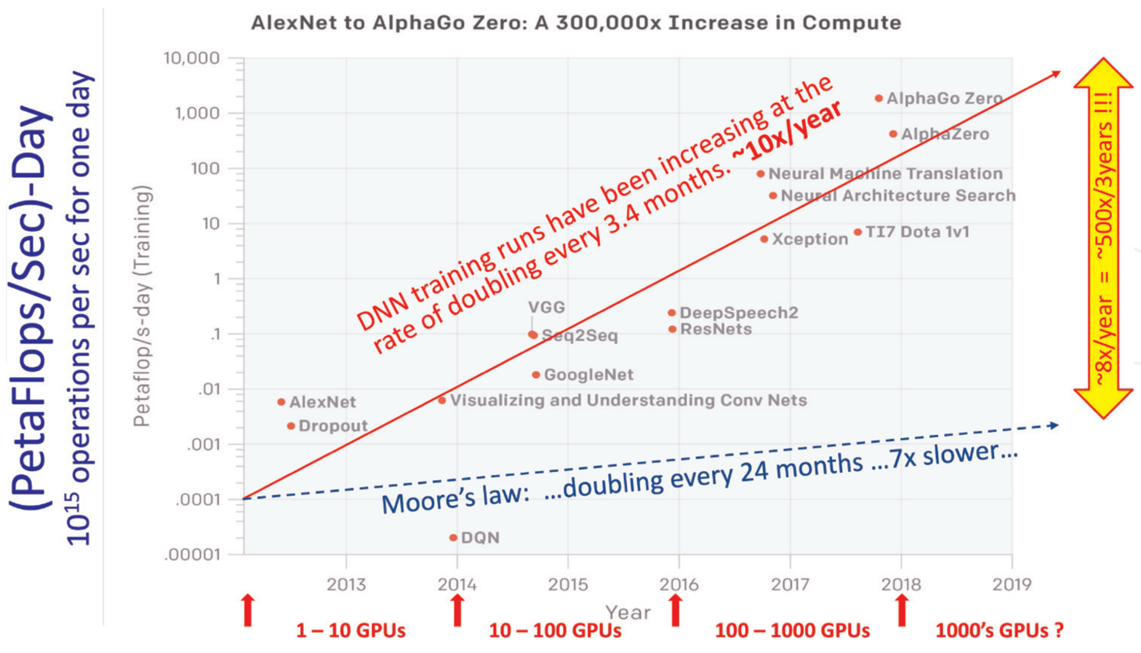


Figure 1.
Figure taken from [19] with annotations added. There is a significant trending gap between DNN computation demand and hardware resource supply provided by Moore's law. This demand-supply gap is increasing at the rate of 8x/year. Thus, to sustain this trend, every year the amount of hardware resources would need to increase by 8× or the training time will need to increase by 8×. A promising candidate solution is to revisit and develop computing paradigms based on the brain's neocortex, which is an existence proof for a highly efficient computing machine for intelligent sensory processing.

through pruning can help mitigate the computation complexity [26–31]. More efficient ways of using the computation hardware infrastructure, including static or dynamic exploitation of value sparsity to avoid unnecessary computation, are also being developed [32–38]. These are all valuable efforts to mitigate the complexity explosion and to help sustain the continuation of the current productive trends. However, there is also the need to concurrently explore other potentially promising alternative paradigms and approaches.

2.2 Right time to revisit neuromorphic computing

All current commercial accelerators for Artificial Intelligence (AI) computation employ the same computing paradigm that emerged more than 70 years ago based on the Turing computation model and the von Neumann stored-program computer architecture. However, these systems were not originally developed for targeting human-type sensory processing workloads that constitute majority of modern AI compute. This underscores the potential need to explore a much more complexity- and energy-efficient computing model and architecture for AI computing. One approach is to revisit biology and examine how to mimic not just the functional behaviors, but also the structural organization of biological neural networks. Such an approach can potentially enable real intelligent computing with significantly less computation complexity and much better energy efficiency. One of the last major efforts taking such "neuromorphic computing" approach was by Carver Mead and his PhD students at CalTech back in the late 1980's utilizing analog VLSI circuits to model neural computation [4, 39, 40]. Both experimental neuroscience research and silicon fabrication technology have made tremendous advancements in the past 40 years, making this a good time to revisit the neuromorphic computing approach [6].

In recent years, several neuromorphic chips have been introduced both from academia and industry, including analog [41], digital [42–48] and mixed-signal [49] implementations. A dominant trend across these approaches is to implement a large number of spiking neurons communicating with each other via event-driven packets that encapsulate spiking information. Such spike-based event-driven computations have been shown to be more energy-efficient than traditional compute. However, they still lack the structural hierarchy and functional abstraction in the form of cortical columns as seen in the neocortex. Digital CMOS implementation of cortical columns and spiking neurons are both discussed as part of our research strategy in the next section.

3. Research approach and strategy

Our research builds on the foundational works of Jeff Hawkins and James E. Smith on reverse-engineering the neocortex from both a neuroscience theorist's perspective and a computer architect's perspective, respectively. These two independent perspectives are uncannily synergistic, and we fully embrace and integrate both of these perspectives in formulating our research approach.

3.1 Hawkins' new theory of neocortical computation

In 2021, Jeff Hawkins published a fascinating book entitled "*A Thousand Brains: A New Theory of Intelligence*" [9]. In this book, accompanied by additional publications from others at Numenta [50–53], Hawkins proposes a new theory of neocortical computation informed by extensive neuroscience research. Hawkins suggests Cortical Columns (CCs) as the key compute units within the human neocortex, which contains about 150,000 of such CCs (see **Figure 2**). The neocortex gains its intelligence through CC's ability to model sensory information in structured Reference Frames (RFs) and continuously update its models as the sensor interacts with the environment.

Each CC is computationally powerful and capable of modeling complete objects. As illustrated in **Figure 2**, complete models of an object exist in multiple CCs across different sensory modalities (e.g., vision, hearing, touch) and across different hierarchy layers at different scales. Multiple CCs, across sensory modalities and layer hierarchies, can communicate and reach consensus via a voting process. This is in contrast to the traditional strict hierarchical view, involving increasing complexity and sophistication with ascending hierarchical layers. Hawkins suggests the increasing level of intelligence demonstrated by higher-functioning mammals is mainly due to the increase in the total number of CCs and not necessarily due to increased complexity or sophistication of the CCs. Furthermore, in contrast to current DNNs that separate training and inference, CCs that store and process information in RFs can support online, concurrent, and continuous learning and can dynamically adapt to sensory input changes.

Recent discussions in the DL community seem to align with the basic ideas of Hawkins' theory. Continually learning structured models of the world via interactions with the environment can help overcome the brittleness and catastrophic forgetting associated with DNNs [54]. With the ability to continually learn and adapt to new

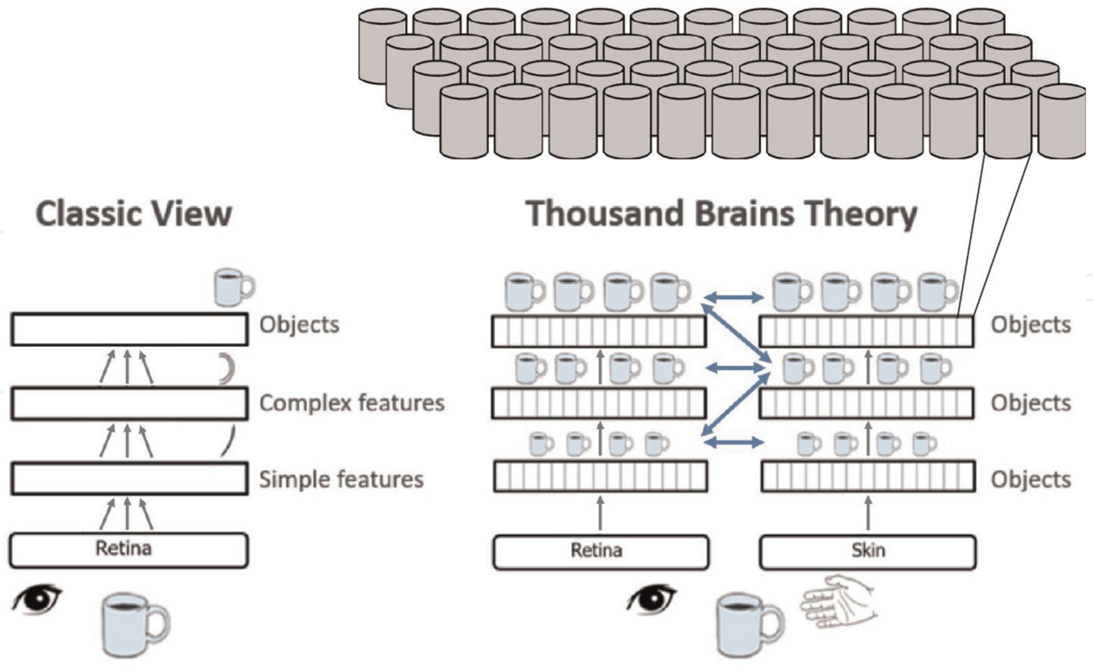


Figure 2.
Figure taken from [51, 53]. Left: Traditional hierarchical view where complexity of features recognized increases up the hierarchy with complete objects detected only at the top. Right: Hawkins' view where neocortex contains about 150 K cortical columns (CCs) and multiple CCs across different sensory modalities and different hierarchy layers can all learn complete models of objects and can communicate to reach a consensus on the output. For example, a coffee cup can be quickly recognized through touch and vision, wherein two sets of CCs with modality-specific models of the coffee cup at different scales all vote together to determine the object.

inputs from the environment, the need to achieve near-perfect accuracy through extensive offline training process can be alleviated. This can potentially reduce the offline training cost and complexity.

3.2 Smith's biologically plausible neural networks

We leverage and build on another significant body of work by James E. Smith on reverse-architecting the brain with the goal of replicating it using off-the-shelf digital CMOS silicon [7, 8, 10, 55]. He proposes a new category of spiking neural networks called temporal neural networks (TNNs) [7] that are architected to mimic the key attributes of biological neocortex (see **Figure 3**). Unlike DNNs performing real-valued tensor-based computations supported by global back propagation for stochastic gradient descent, TNNs employ spiking neurons that encode and process inputs as timings of events or spikes, and can learn using local biologically plausible algorithm called spike timing dependent plasticity (STDP). TNNs also differ from most other spiking neural networks (SNNs) that encode values based on the rate of spikes as opposed to spike timing. **Figure 3** shows this taxonomy. Consequently TNNs, unlike most other ANNs, are more truly "neuromorphic" due to their strong adherence to biological plausibility.

Smith has also developed a TNN-based architecture for the cortical columns and demonstrated the capability for doing unsupervised learning with rapid convergence, and the capability to do online, concurrent, and continuous learning [10, 55]. Formulating the underlying mathematical basis for implementing TNNs, Smith has proposed a new algebra called space-time (temporal) algebra [8]. Based on this new temporal algebra (instead of Boolean algebra), temporal functions can be implemented very efficiently in hardware by re-purposing the current digital logic gates to make use of time as a "free" resource for both encoding and processing of information.

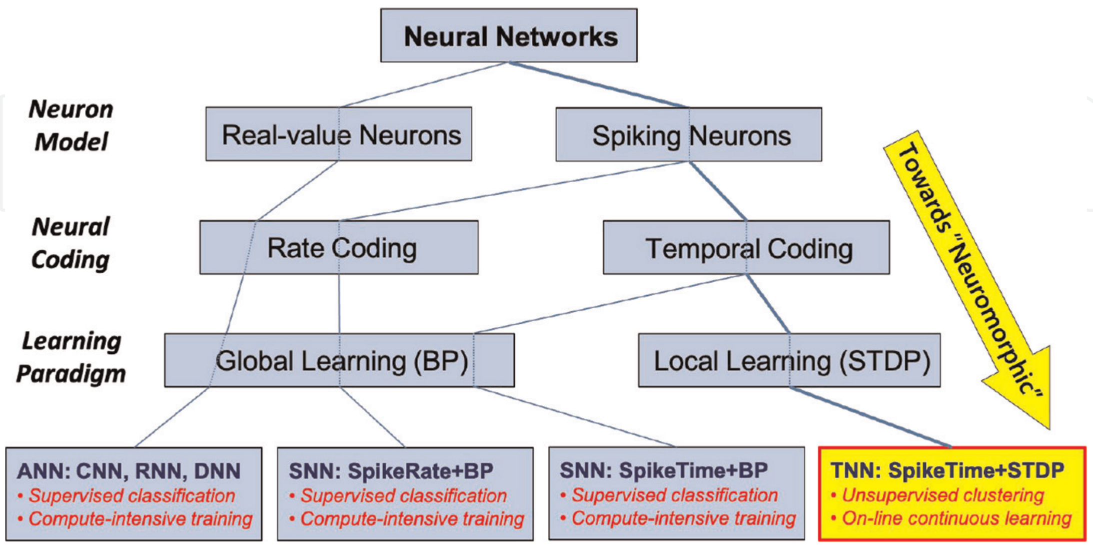
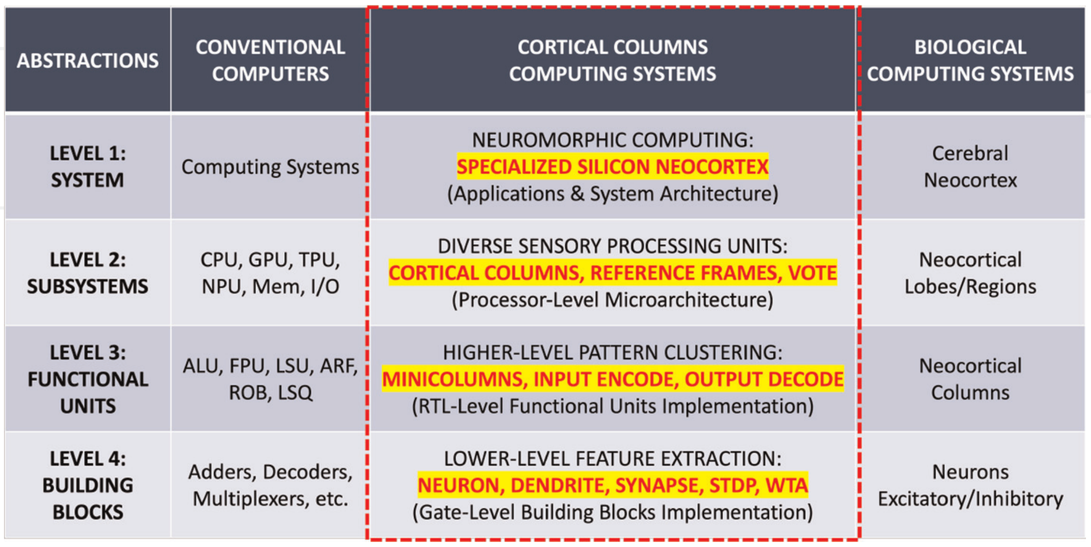


Figure 3.
Neural network taxonomy contrasting neocortex-inspired temporal neural networks (TNNs) with other artificial neural networks (ANNs). In contrast to other ANNs including deep neural networks (DNNs), TNNs incorporate attributes with strong adherence to biological plausibility, including spiking neuron model, temporal coding of inputs, and local simple spike timing dependent plasticity (STDP) learning rules.

3.3 Cortical columns computing systems

Our research builds on these two bodies of prior works to explore the potential of creating brain-inspired and brain-like computing fabric, which we call *Cortical Columns Computing Systems (C3S)*. **Figure 4** compares our C3S approach against conventional computers and biological neocortex. Our goal is to leverage the best attributes of the other two paradigms to create a new computing paradigm. **Figure 4** shows the linkage of comparable levels of abstractions across these three distinct computing system paradigms. Our research spans all four levels of abstractions including sensory processing applications, a processor-level microarchitecture model, RTL-level functional units, and gate-level building blocks. In order to target and impact mass market computing, our research must be able to leverage current digital CMOS technology and design tools. One of our goals is to develop an end-to-end framework for designing and implementing Cortical Columns Computing Systems (C3S). This will require the development of novel design exploration and design synthesis tools. Our target applications include diverse sensory signal processing units capable of energy-efficient and edge-native AI inference and online continuous learning.

As illustrated in **Figure 5**, this framework consists of three key components: (1) a microarchitecture model that can facilitate RTL implementation of Cortical Columns (CCs) and Reference Frames (RFs) employed in C3S designs; (2) a suite of highly optimized functional units and building blocks implemented in System Verilog to support efficient application-specific implementations of C3S designs; and (3) a software tool suite consisting of a PyTorch simulator for rapid design space exploration of C3S designs and a design synthesis flow for mapping PyTorch functional models to corresponding C3S hardware. Specific applications of current interest include visual object recognition, anomaly detection on time-series signals (e.g., ECG), edge-native always-on keyword spotting, and multi-modal human activity recognition (HAR) [56].



ABSTRACTIONS	CONVENTIONAL COMPUTERS	CORTICAL COLUMNS COMPUTING SYSTEMS	BIOLOGICAL COMPUTING SYSTEMS
LEVEL 1: SYSTEM	Computing Systems	NEUROMORPHIC COMPUTING: **SPECIALIZED SILICON NEOCORTEX** (Applications & System Architecture)	Cerebral Neocortex
LEVEL 2: SUBSYSTEMS	CPU, GPU, TPU, NPU, Mem, I/O	DIVERSE SENSORY PROCESSING UNITS: **CORTICAL COLUMNS, REFERENCE FRAMES, VOTE** (Processor-Level Microarchitecture)	Neocortical Lobes/Regions
LEVEL 3: FUNCTIONAL UNITS	ALU, FPU, LSU, ARF, ROB, LSQ	HIGHER-LEVEL PATTERN CLUSTERING: **MINICOLUMNS, INPUT ENCODE, OUTPUT DECODE** (RTL-Level Functional Units Implementation)	Neocortical Columns
LEVEL 4: BUILDING BLOCKS	Adders, Decoders, Multiplexers, etc.	LOWER-LEVEL FEATURE EXTRACTION: **NEURON, DENDRITE, SYNAPSE, STDP, WTA** (Gate-Level Building Blocks Implementation)	Neurons Excitatory/Inhibitory

Figure 4.
Comparison of cortical columns computing systems (C3S) against conventional computers and biological computing systems in terms of levels of computing abstraction. Our research strategy is highlighted in the red box and spans the four abstraction layers: Applications and system architecture, processor-level microarchitecture, RTL-level functional units, and gate-level building blocks.

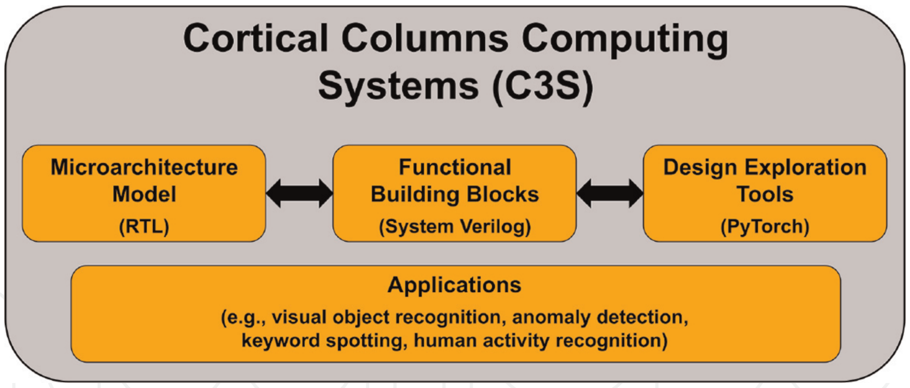

Cortical Columns Computing Systems (C3S)
Microarchitecture Model (RTL)
Functional Building Blocks (System Verilog)
Design Exploration Tools (PyTorch)
Applications (e.g., visual object recognition, anomaly detection, keyword spotting, human activity recognition)

Figure 5.
Proposed framework for implementing C3S designs consists of three key components: Microarchitecture model [11], functional building blocks [12] and design exploration tools. Some applications of interest are visual object recognition, anomaly detection [13], keyword spotting, and human activity recognition [56]. These framework components currently support TNN design and implementation and key relevant publications are cited here. Our ongoing research aims to extend this framework to support more general C3S designs.

4. C3S microarchitecture model

4.1 Mini-columns and TNNs (completed work)

We have developed a microarchitecture model for building highly efficient TNN designs [11]. In this model, values are encoded as timings of events or spikes. Spikes are implemented as logic pulses whose timings are calibrated using unit hardware clock. We refer to this as "direct" implementation as hardware clock itself defines the time unit for temporal processing and spikes directly arrive at precisely timed clock cycles. Here, spike timings are not encoded as binary values propagated in the form of packets. The proposed TNN microarchitecture model consists of the following key modules: (1) a counter-based synapse that performs temporal processing of input spikes to generate specialized responses and unsupervised/supervised STDP learning of synaptic weight; (2) a multi-synapse neuron that accumulates the synaptic responses and fires an output spike when the accumulated response crosses a threshold; and (3) a multi-neuron TNN column (referred henceforth as "mini-column" to distinguish from cortical column) that applies winner-take-all (WTA) inhibition across its neurons. **Figure 6** illustrates the implementation of a mini-column.

A mini-column with p synaptic inputs feeding q neurons via a pxq synaptic crossbar is referred to as a *(pxq) mini-column* in **Figure 6**. The figure illustrates the actual implementation of the major components of a (pxq) mini-column, including the synapses, neuron body, STDP local learning, and WTA lateral inhibition. In our prior work [11], we have taken the RTL designs of various configurations of the mini-column through both the synthesis and physical design tools, and obtained PPA (power, performance, area) results by scaling both p and q. We observed that power and area scale linearly with the total number of synapses (pxq), whereas performance limited by the critical path delay scales logarithmically with the number of synaptic inputs (p). We also derived a set of characteristic equations from the gate-level designs that provide qualitative estimation of gate-level PPA complexity for arbitrary mini-column designs.

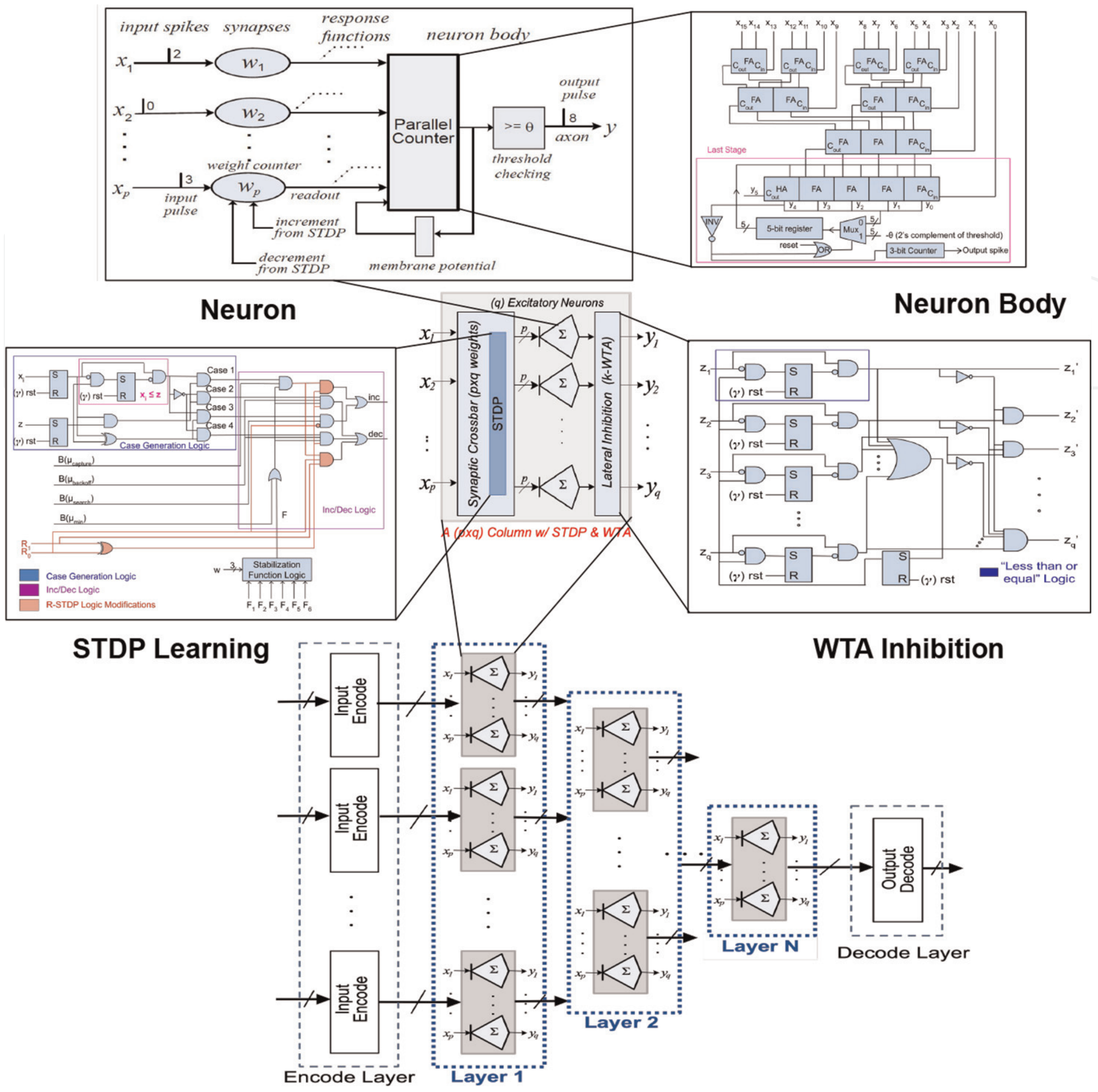


Figure 6.
Microarchitecture model for implementing multi-column and multi-layer TNNs [11]. This figure highlights details of a generic TNN column (or mini-column). Each mini-column consists of p inputs feeding a stack of q neurons via a pxq synaptic crossbar. Each cross point in the pxq crossbar stores a weight value that is updated based on spike timing dependent plasticity (STDP) updating rules. Each neuron performs the weighted sum of its inputs. Each mini-column is also supported by winner-take-all (WTA) inhibition across its neuron outputs that selects the winner neuron. Gate-level implementations of these mini-column components are illustrated.

Multiple mini-columns can be grouped and organized into a hierarchy of layers to form multi-layer temporal neural networks (TNNs), as shown at the bottom of **Figure 6**. Multiple multi-neuron mini-columns are stacked to form a single layer and multiple multi-column layers are cascaded to form a multi-layer TNN. A TNN is typically bookended by input-encode and output-decode layers, and is effectively a feed-forward network. The mini-column is the fundamental building block for TNNs and can learn to distinguish distinct input patterns. Hence, a single mini-column can be viewed as a fully functioning TNN. **Table 1** in Section 6 demonstrates the efficacy of single mini-column designs in performing unsupervised clustering with minimal PPA complexity.

UCR Mini-column Design (pxq)	UCR Benchmark Name	Synapse Count	Power (μW)	Comp. Time (ns)	Area (mm^2)
65x2	SonyAIBORobotSurface2	130	0.78	13.86	0.001
96x2	ECG200	192	1.12	14.49	0.001
152x2	Wafer	304	1.76	15.96	0.002
343x2	ToeSegmentation2	686	3.95	17.32	0.005
637x2	Lightning2	1274	7.33	18.75	0.011
470x5	Beef	2350	14.62	22.08	0.018
270x25	WordSynonyms	6750	39.00	17.51	0.054

All mini-columns are specifically trained for their corresponding benchmarks and achieve competitive performance relative to state-of-the-art. Even the largest mini-column design only consumes 39 μ W.

Table 1.
Design space exploration for UCR time-series clustering: Using TNN7 macros, PPA [12] for seven sample TNN prototype designs (mini-columns) across diverse synapse counts and application benchmarks from [13].

4.2 Cortical columns with RFs (ongoing work)

The current phase of our research aims to leverage the high synergy between cortical columns and TNN mini-columns. We plan to extend the TNN microarchitecture model to incorporate CCs consisting of RFs, making it generic enough to accomplish integration across diverse sensory modalities (see **Figure 7**).

In contrast to TNNs that employ feed-forward processing, CCs store structured information about inputs in Ref. Frames (RFs), and process and update the stored information through feedback connections. This continuous feedback loop of predict-sense-update effectively introduces an additional dimension of "memory" that remembers past observations and patterns. This "sequential" behavior is missing in feed-forward TNNs. Hence, TNN mini-columns combined with feedback mechanism implementing the predict-sense-update loop can be used to build cortical columns. As shown in **Figure 7**, multiple CCs targeting different sensory modalities can interact with each other and seek consensus on the output via voting within and across sensory modalities. Each CC, irrespective of its sensory modality, broadly implements two components: 1) *a Reference Frame* that maintains a "map" of the sensory information, and 2) an *Agent* that achieves goal-oriented behavior based on information from the Reference Frame and the input signals. Agent comprises of two TNN-type mini-columns performing unsupervised clustering and supervised classification. Reference Frame involves three functionalities which can be mapped to three types of mini-columns: *Where* Column, *What* Column and *Output* Column. In the context of visual object recognition, the three types of mini-columns together create models of objects by tracking locations of features on the object. The *Output* Column is envisioned to be very similar to the TNN mini-column discussed previously and determines the object identity. The *Where* and *What* Columns take the outputs from the *Output* Column as feedback information. The Where Column generates location of sensor on the object based on this feedback and the latest movement information from the agent. The *What* Column predicts object features based on the result from the *Where* Column and updates its model based on the actual sensory input and the feedback from the *Output* Column. In order to apply this model to other time-series applications, a key

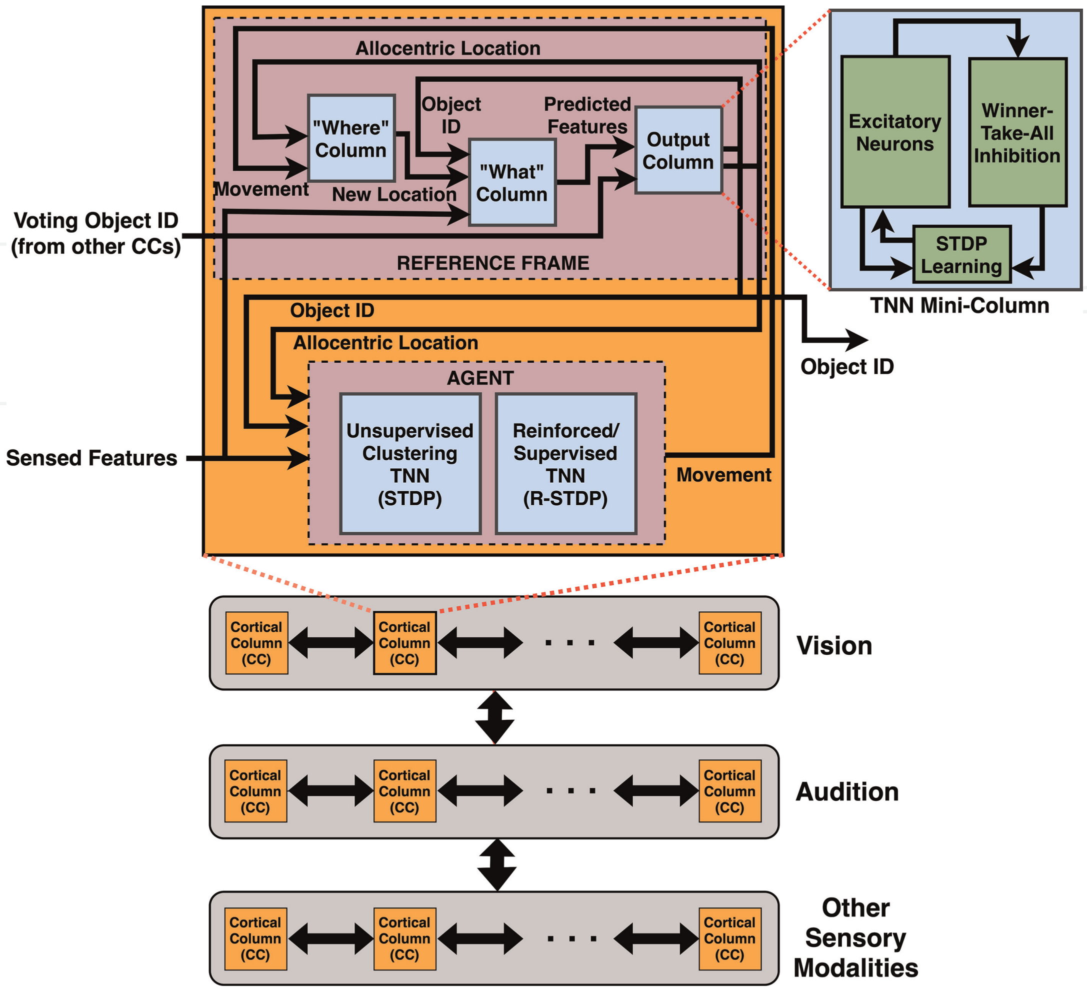


Figure 7.
Cortical columns computing system (C3S) architecture consisting of multiple CCs targeting multiple sensory modalities interacting with each other to form a consensus on the output via voting. Each CC broadly consists of five TNN-style mini-columns: Where, What and Output mini-columns that together implement the Reference Frame (RF), and unsupervised and supervised mini-columns comprising the agent. For visual object recognition, the respective functionalities of the three RF mini-columns are: derive locations of sensor on the object, map features to locations, and derive the object ID based on the feature map. In contrast to feedforward TNNs, each CC learns through feedback from output and possesses a form of "memory" in the learning process.

component that needs to be investigated is pre-processing of the diverse sensory signals to extract abstract "location-feature" pairs from raw sensory signals. Developing a detailed parameterized design template of a generic Cortical Column (CC) containing the five mini-column types is largely ongoing research.

5. C3S functional building blocks

5.1 Custom macro cells for mini-column designs (completed work)

From the TNN microarchitecture model, we identified the key fundamental building blocks and implemented them as a suite of custom standard cell macros in 7 nm CMOS, called *TNN7* [12]. Using these custom macros, we can further optimize the TNN designs and enhance their scalability. These building blocks and the

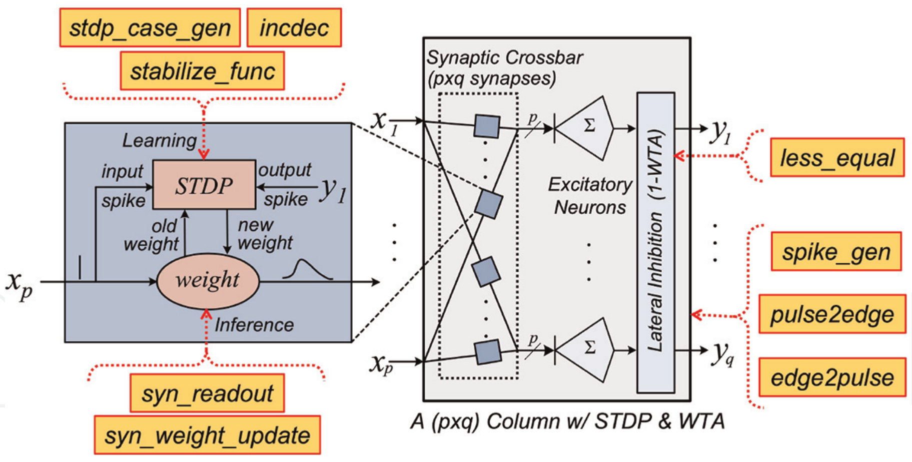


Figure 8.
Functional building blocks of a mini-column (q neurons with p synapses each) and associated proposed TNN7 macros (highlighted in yellow) [12]. Two macros are developed for synaptic feedforward processing, three for synaptic STDP learning, one for WTA inhibition and three for generic utility functions such as spike encoding.

corresponding proposed nine macros are detailed in **Figure 8**. Two macros are implemented for synaptic temporal processing, three for synaptic STDP learning, one for WTA inhibition, and three for general-purpose utility functions such as spike encoding. These macros have been optimized such that they incur minimal numbers of gates and transistors to achieve their corresponding functionalities. After implementing the custom macro-based TNN designs in SystemVerilog, we observed significant improvements in all PPA metrics. Specifically, we achieved 14, 16, 28, and 45% improvements in power, performance, area, and energy-delay product (EDP) respectively, relative to our original designs that simply use off-the-shelf standard cells. Post-layout designs are also considerably simplified using *TNN7* (see **Figures 9a** and **b**). Further, instantiating the *TNN7* macros during logic synthesis reduces the netlist generation runtime considerably (by more than 3x typically).

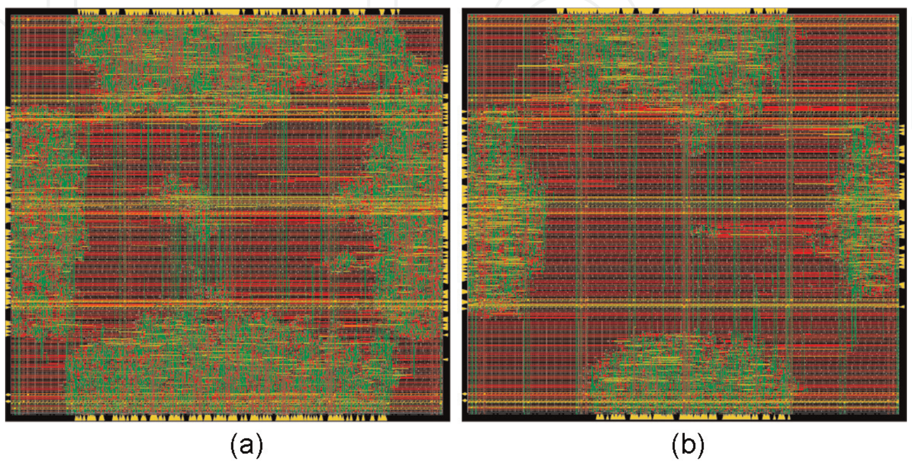


Figure 9.
Layout comparison for a sample mini-column [12]: Original design using off-the-shelf standard cell library vs. custom design using enhanced cell library with TNN7 macros. TNN7 design incurs much lesser wiring and logic complexity, and delivers 14, 16, 28, and 45% improvements in power, performance, area, and energy-delay product respectively. Note: TNN7 macros are developed using standard toolchain and design flow.

5.2 Custom macro cells for C3S designs (ongoing work)

In our follow-on work, we plan to examine decomposition of the extended C3S microarchitecture model into its fundamental building blocks, and implement them as custom macro cells. These new macro cells can be added to our earlier *TNN7* library to create a new extended library with custom macros for supporting C3S designs. We envision distinct specialized macros for "storage", "predict", "update" and "voting" functionalities that are generic enough to build *Where*, *What* and *Output* mini-columns at arbitrary scales and modalities. A key goal here is to have a rich set of macros that support diverse sensory modalities to enable implementation of potentially new types of mini-columns in the future. Currently, content-addressable memories (CAMs) seem to be a promising candidate as building blocks for implementing mini-columns for Reference Frames. In order to effectively predict relevant information based on stored knowledge in RFs, we believe these CAMs would need to support fuzzy matching; not strict exact matching as done conventionally. Implementing fuzziness in an RF can help recover similar features that can aid rapid learning of new patterns and features. This is largely ongoing and future work.

6. C3S design tools

6.1 TNN simulation tool and example applications (completed work)

Along with the hardware framework consisting of microarchitecture model and *TNN7* custom macros, we also developed a software tool, called *TNNSim*, that follows the above TNN microarchitecture model and allows rapid application-specific design exploration (see **Figure 10**). This tool is based on PyTorch [57], which is a popular deep learning framework widely used in academia. Using PyTorch enables easy integration of standard DL benchmarks and fast parallel processing using its native computational libraries. The *TNNSim* toolflow and its major libraries of functions are illustrated in **Figure 10**. Combining design exploration on *TNNSim* with *TNN7* implementation, we demonstrate TNN designs for the following two classes of applications. **Tables 1** and **2** summarize ten TNN design points for these applications.

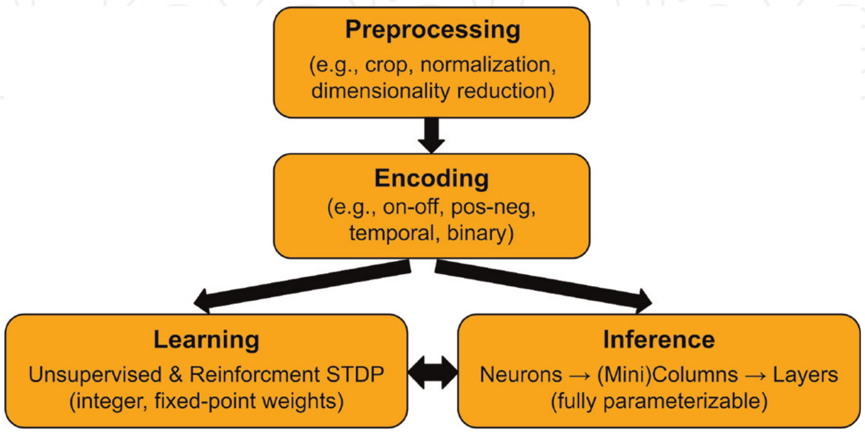

Preprocessing
(e.g., crop, normalization, dimensionality reduction)
Encoding
(e.g., on-off, pos-neg, temporal, binary)
Learning
Unsupervised & Reinforcment STDP
(integer, fixed-point weights)
Inference
Neurons → (Mini)Columns → Layers
(fully parameterizable)

Figure 10.
TNNSim flow consisting of libraries implementing the four major functionalities: Preprocessing, encoding, inference and continual learning. Each functionality is implemented as a separate class and can be instantiated in a modular fashion to design arbitrary TNNs. TNNSim has enabled online learning demonstration and application performance exploration in our prior works [11–13].

6.1.1 UCR time-series clustering benchmark suite

Using *TNNSim*, we designed 36 single-(mini) column TNNs of various sizes to perform unsupervised clustering on 36 UCR time-series datasets [58]. These single-column TNNs can achieve state-of-the-art performance surpassing most of the contemporary comparable algorithms [13]. **Table 1** provides synapse count and PPA for 7 of the 36 benchmark-specific designs to demonstrate the efficacy of single mini-column designs across various configurations and various benchmark applications. The seven benchmarks and their input signal types are briefly described: (1) *SonyAIBORobotSurface2* - accelerometer signals to detect two types of walking surfaces; (2) *ECG200* - ECG signals to detect normal heartbeat vs. myocardial infarction; (3) *Wafer* - fabrication process control sensor signals to detect normal vs. abnormal silicon wafers; (4) *ToeSegmentation2* - motion sensors to detect normal vs. abnormal walking; (5) *Lightning2* - power-density series derived from optical and RF sensor spectogram to detect lightning; (6) *Beef* - food spectograph to detect varying levels of adulteration; and (7) *WordSynonyms* - 1D series from word outlines to detect 25 different words.

The smallest mini-column (130 synapses) for *SonyAIBORobotSurface2* and the largest mini-column (6750 synapses) for *WordSynonyms* perform very efficient unsupervised clustering within only 1 μW and 40 μW power, respectively [12]. PPA metrics scale with synapse count as expected. Note that mini-column for *Beef* has the highest input synapse count p and therefore incurs the largest computation time, as delay depends on p. These benchmarks demonstrate even relatively small single mini-column designs can perform practically useful clustering with minimal hardware complexity and power consumption.

6.1.2 MNIST handwritten digit recognition benchmark

Here, we demonstrate much larger multi-column multi-layer TNN designs with each layer composed of multiple TNN mini-columns, trained to recognize 10 classes of handwritten digits from 0 through 9 [59]. Three multi-layer TNNs are presented here. They include 2-layer, 3-layer and 4-layer designs containing 389 K, 1.3 M and 3 M total synaptic counts respectively, as shown in **Table 2**. With increasing number of layers, these three TNNs can achieve 93, 97 and 99% accuracy on the MNIST dataset, while consuming only about 2, 8 and 18 mW of power, respectively [12]. This demonstrates the huge energy-efficiency potential of TNNs. Using *TNNSim*, we also illustrate online incremental learning capability wherein a TNN design learns a new

MNIST TNN Design	Error Rate	Synapse Count	Power (mW)	Comp. Time (ns)	Area (mm^2)
2-Layer	7%	389 K	2.25	41.38	3.09
3-Layer	3%	1310 K	7.57	66.16	10.42
4-Layer	1%	3096 K	17.89	91.58	24.63

*2-layer, 3-layer and 4-layer TNN designs achieve progressively decreasing error rates (7%, 3% and 1%) while consuming increasing power. The 4-layer design incurs just 18 mW power in delivering state-of-the-art accuracy of 99%. Note the synapse counts of these multi-column multi-layer TNN designs are significantly higher than that of the single mini-column TNN designs in **Table 1**.*

Table 2.
Design space exploration for MNIST: Using TNN7 macros, PPA for three multi-layer TNN prototype designs from [10].

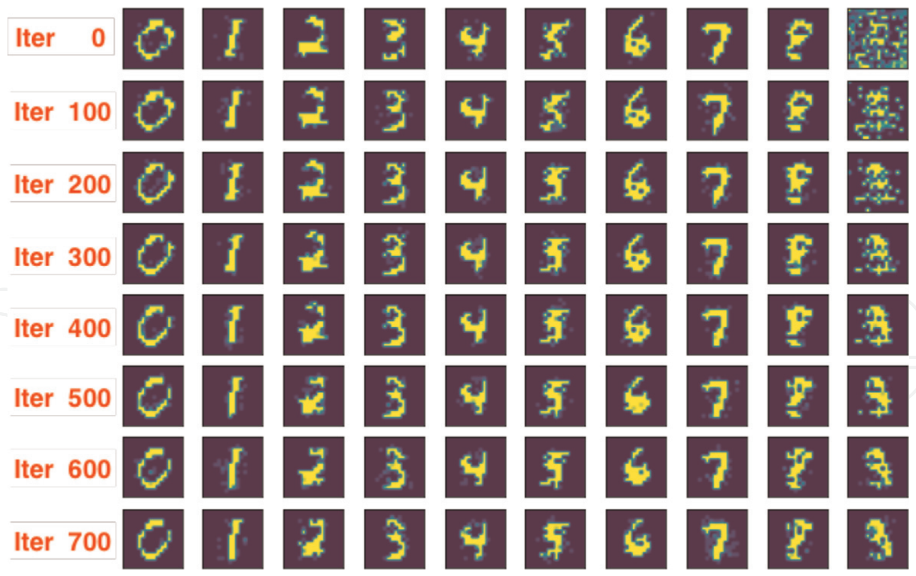


Figure 11.
Online incremental learning: A single TNN mini-column trained only on the digits 0–8 learns a new previously unseen digit '9' within 500 samples (that consist about 50 samples of digit '9') in an unsupervised manner [11]. This demonstrates the online learning capability of TNNs, which can quickly learn new classes from relatively few examples without forgetting previously learned information.

previously unseen pattern using unsupervised STDP. **Figure 11** shows the converged weights of a single mini-column that is first trained using only digits 0–8 (iteration 0). When introducing the previously unseen digit '9', within around 500 new examples (iteration 500) that contain about 50 examples of '9', it learns the new digit '9' without forgetting previously learned digits in an unsupervised fashion [11]. This demonstrates the ability for this mini-column to perform online continual learning.

6.2 Proposed C3S simulation & synthesis tools (ongoing work)

Building on the previously described three components, the overarching goal of our current research is to develop an end-to-end framework that can automatically translate application-specific C3S models in software to highly customized hardware designs. It can be utilized as an architecture and design synthesis framework for implementing C3S processing units. We envision this as a complete design framework spanning applications, architecture, microarchitecture, and custom macro suite, to generate application-specific C3S processing units for diverse sensory processing applications. As shown in **Figure 12**, the framework consists of two main components.

C3S-Sim (Application and Architectural Design Exploration): This is the software simulator framework, consisting of (1) a PyTorch simulator (extended from *TNNSim* to incorporate C3S functional modeling), and (2) a cycle-accurate architectural simulator in C++ to mimic hardware and derive accurate latency performance information for C3S designs.

C3S-Syn (Micro-architectural Design and Implementation): This is the hardware implementation framework that takes in the PyTorch C3S functional models and generates C3S hardware designs. It is envisioned to include (1) PyVerilog conversion for automated RTL generation from PyTorch, (2) automated RTL-to-GDSII flow that leverages C3S-specialized custom macro cells and generates application-specific post-layout netlist and PPA results for a specific C3S design.

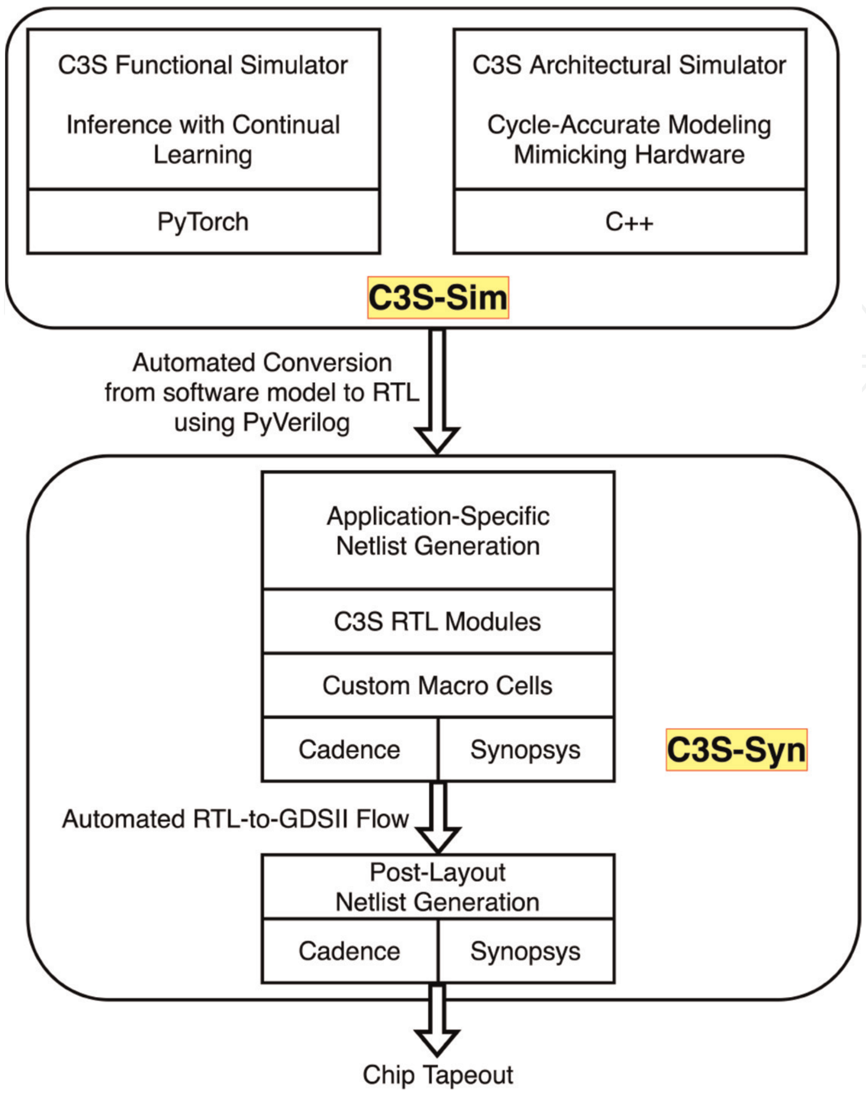

C3S Functional Simulator
Inference with Continual Learning
PyTorch
C3S Architectural Simulator
Cycle-Accurate Modeling Mimicking Hardware
C++
C3S-Sim
Automated Conversion from software model to RTL using PyVerilog
Application-Specific Netlist Generation
C3S RTL Modules
Custom Macro Cells
Cadence
Synopsys
C3S-Syn
Automated RTL-to-GDSII Flow
Post-Layout Netlist Generation
Cadence
Synopsys
Chip Tapeout

Figure 12.
Envisioned end-to-end cortical columns computing system (C3S) design framework consisting of C3S-Sim for application exploration and C3S-Syn for microarchitectural implementation. C3S-Sim consists of a PyTorch tool (extension of TNNSim) to design application-specific C3S functional models and a cycle-accurate architectural simulator for hardware performance estimation. C3S-Syn incorporates the extended microarchitecture model and functional building blocks for C3S implementation, with an automated design flow to translate PyTorch functional models to application-specific hardware designs.

6.3 Targeted applications for C3S designs (future work)

Our targeted applications for C3S designs are influenced by the emergence of several dominant industry trends: (1) *Increasing AI Compute Demand*: The computation required for supporting Deep Learning (DL) is increasing exponentially at the staggering rate of doubling every 3.4 months [19]. Many believe that this trend is not technologically nor economically sustainable [20–22]. (2) *Migration Towards Edge AI*: There is strong efficiency, security, and privacy motivations for migrating AI computation from the centralized cloud infrastructure to distributed personal edge devices. (3) *Emerging Ubiquity of Wearables*: Mobile devices, e.g. smartphones, have replaced PCs as the technology, innovation, and profitability driver for mass market computing, due to their more stringent form factor and energy efficiency requirements. As wearable devices become more ubiquitous, we believe they will in turn become the new technology and innovation driver for mass market computing [60]. (4) *Interests* *in Neuromorphic Chips*: Orders of magnitude improvement on energy efficiency is necessary to support autonomous intelligent wearable devices. We believe to achieve such level of improvement will require adopting the neuromorphic approach in creating new computing devices that mimic the organization and operation of biological neural networks of the neocortex.

Hence, our C3S research focuses on targeting the design and implementation of *Neuromorphic Intelligent Sensory Processing (NISP) Chiplets* with truly brain-like sensory processing capability as well as brain-like energy efficiency. This new genre of C3S based sensory processing units exhibit the following attributes: (1) **Neuromorphic**: based on reverse architecting the neocortex; (2) **Intelligent**: capable of edge-native autonomous and continuous online learning; (3) **Sensory Processing**: support near-sensor on-device processing of diverse modalities of sensory signals; (4) **Chiplets**: target chiplet implementations (potentially UCIe-compliant [14]) in standard digital CMOS, for ease of integration into diverse edge AI platforms on mobile, wearable, and IoT devices.

7. Summary and conclusion

Figure 13 summarizes our overall research vision and provides a broader context for our work. Current deep learning approach adopts the path on the left, employing high dimensional tensor processing and hardware accelerators with massive array of MAC units. Our approach takes the path on the right, leveraging spike timing processing and hardware fabric inspired by cortical columns. Our initial focus was on feed-forward mini-columns based on TNN principles. Our current focus broadens to

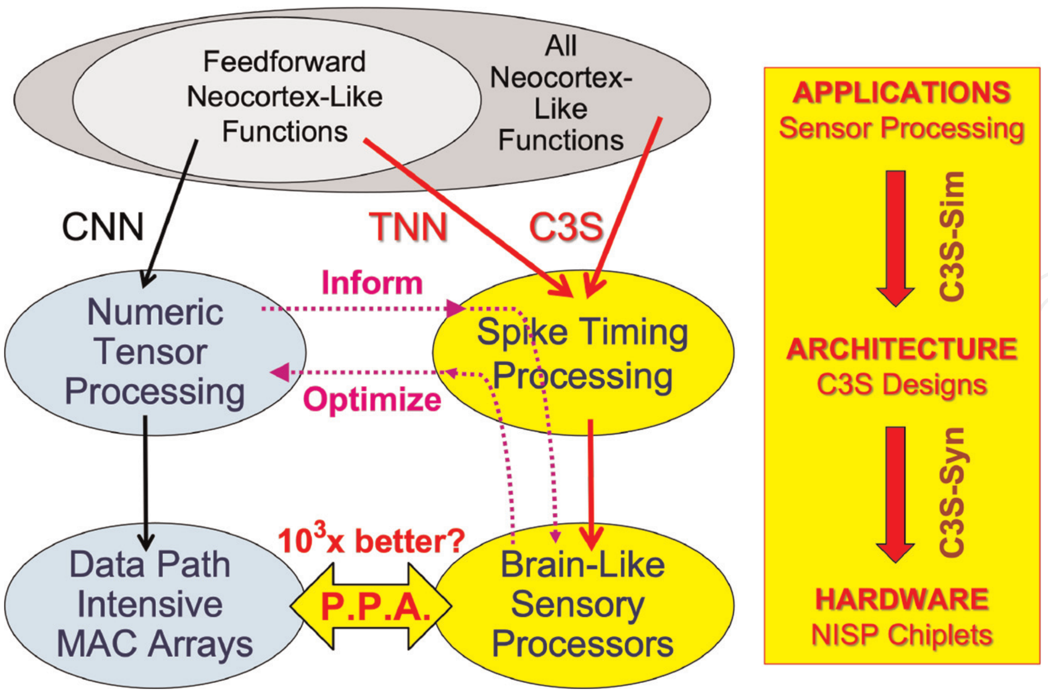


Figure 13.
Summary of envisioned research in a broader context. Left path corresponds to the current deep learning approach that mainly employs numeric tensor processing deployed on hardware accelerators consisting of large MAC arrays. Right path illustrates our research approach that employs spike timing processing in the design and implementation of special-purpose processing units inspired by cortical columns, or Neuromorphic Intelligent Sensory Processing (NISP) units. Interesting cross-over ideas between both paths should be explored.

all cortical columns with feedback in the form of reference frames. We are developing a framework and software tool suite for designing and implementing application-specific, highly energy-efficient sensory processing units, or Neuromorphic Intelligent Sensory Processing (NISP) units, with continuous online on-device learning capability. Our goal is to achieve potentially up to three orders of magnitude improvements on power and energy efficiency, relative to current deep learning accelerators. We also anticipate there could be interesting cross-over ideas between the current deep learning path and our new C3S path that should be explored.

This chapter presents a neuromorphic computer architecture and design approach that focuses on implementing neocortical computing fabric using digital off-the-shelf CMOS technology. This work builds on the foundational works of Jeff Hawkins' "*A Thousand Brains Theory*" and James E. Smith's *biologically plausible neural networks*. This research effort in NCAL at CMU aims to build Cortical Columns Computing Systems (C3S) that exhibit brain-like capabilities and brain-like efficiency. We hope our work serves as one step towards the holy grail of building a silicon neocortex. We hope to generate broad interest through this chapter that can lead to a vibrant research community pursuing this line of research. We believe there are tremendous opportunities for novel innovations that can have significant industry impact.

Acknowledgements

Our research effort has been inspired and guided by the foundational work of **James E. Smith** (Emeritus Professor, University of Wisconsin, and Adjunct Professor, Carnegie Mellon University). We are indebted to Jim's work and guidance. Two other PhD students in NCAL (Neuromorphic Computer Architecture Lab), **Shreyas Chaudhari** and **Prabhu Vellaisamy**, along with many MS students in CMU's 18-743 "*Neuromorphic Computer Architecture and Processor Design*" course, have made and are still making significant contributions to this research effort.

Author details

John Paul Shen* and Harideep Nair
Carnegie Mellon University, Pittsburgh, USA

*Address all correspondence to: jpshen@cmu.edu

References

[1] Moravec H. Mind Children: The Future of Robot and Human Intelligence. Cambridge, MA, USA: Harvard University Press; 1988

[2] Minsky M. Society of Mind. New York, NY, USA: Simon and Schuster; 1988

[3] Brooks RA. Intelligence without representation. Artificial Intelligence. 1991;**47**(1–3):139-159

[4] Mead C. Neuromorphic electronic systems. Proceedings of the IEEE. 1990; **78**(10):1629-1636

[5] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review. 1958;**65**(6): 386

[6] Schuman C, D, Potok TE, Patton RM, Birdwell DJ, Dean ME, Rose GS, et al. A survey of neuromorphic computing and neural networks in hardware. arXiv preprint arXiv:1705.06963. 2017

[7] Smith JE. Space-time computing with temporal neural networks. Synthesis Lectures on Computer Architecture. 2017;**12**(2):i-215

[8] Smith J. Space-time algebra: A model for neocortical computation. In: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). Los Angeles, CA, USA: IEEE; 2018. pp. 289-300

[9] Hawkins J. A Thousand Brains: A New Theory of Intelligence. London, United Kingdom: Hachette UK; 2021

[10] Smith JE. A temporal neural network architecture for online learning. arXiv preprint arXiv:2011.13844. 2020

[11] Nair H, Shen JP, Smith JE. A microarchitecture implementation framework for online learning with temporal neural networks. In: 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). Tampa, FL, USA: IEEE; 2021. pp. 266-271

[12] Nair H, Vellaisamy P, Bhasuthkar S, Shen JP. Tnn7: A custom macro suite for implementing highly optimized designs of neuromorphic tnns. In: 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). Pafos, Cyprus: IEEE; 2022. pp. 152-157

[13] Chaudhari S, Nair H, Moura JMF, Shen JP. Unsupervised clustering of time series signals using neuromorphic energy-efficient temporal neural networks. In: ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Toronto, Canada: IEEE; 2021. pp. 7873-7877

[14] Sharma DD, Pasdast G, Qian Z, Aygun K. Universal chiplet interconnect express (ucie): An open industry standard for innovations with chiplets at package level. IEEE Transactions on Components, Packaging and Manufacturing Technology. 2022;**12**(9): 1423-1431

[15] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;**521**(7553): 436-444

[16] Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, et al. A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys (CSUR). 2018;**51**(5):1-36

[17] Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, et al. A

state-of-the-art survey on deep learning theory and architectures. Electronics. 2019;**8**(3):292

[18] Dong S, Wang P, Abbas K. A survey on deep learning and its applications. Computer Science Review. 2021;**40**: 100379

[19] OpenAI. Ai and compute. 2018. Available from: https://openai.com/blog/ai-and-compute/.

[20] Thompson NC, Greenewald K, Lee K, Manso GF. The computational limits of deep learning. arXiv preprint arXiv:2007.05558. 2020

[21] Lohn A, Musser M. Ai and Compute: How Much longer Can Computing Power Drive Artificial Intelligence Progress. Washington, D.C., USA: Center for Security and Emerging Technology (CSET); 2022

[22] Numenta Inc. Ai Is Harming our Planet: Addressing ai's Staggering Energy Cost. CA, USA: Numenta Inc.; 2022

[23] Alex Lyzhov. "ai and compute" trend isn't predictive of what is happening (blog post). 2021. Available from: https://www.alignmentforum.org.

[24] Sevilla J, Heim L, Ho A, Besiroglu T, Hobbhahn M, Villalobos P. Compute trends across three eras of machine learning. arXiv preprint arXiv: 2202.05924. 2022

[25] Patterson D, Gonzalez J, Hölzle U, Le Q, Liang C, Munguia L-M, et al. The carbon footprint of machine learning training will plateau, then shrink. Computer. 2022;**55**(7):18-28

[26] Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149. 2015

[27] Li F, Zhang B, Liu B. Ternary weight networks. arXiv preprint arXiv: 1605.04711. 2016

[28] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research. 2017;**18**(1):6869-6898

[29] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. Venice, Italy: ICCV; 2017. pp. 1389-1397

[30] Wu H, Judd P, Zhang X, Isaev M, Micikevicius P. Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv preprint arXiv:2004.09602. 2020

[31] Sun X, Wang N, Chen C-Y, Ni J, Agrawal A, Cui X, et al. Vijayalakshmi Viji Srinivasan, and Kailash Gopalakrishnan. Ultra-low precision 4-bit training of deep neural networks. Advances in Neural Information Processing Systems. 2020;**33**:1796-1807

[32] Wen W, Chunpeng W, Wang Y, Chen Y, Li H. Learning structured sparsity in deep neural networks. Advances in Neural Information Processing Systems. 2016;**29**:1-9

[33] Gale T, Elsen E, Hooker S. The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574. 2019

[34] Hoefler T, Alistarh D, Ben-Nun T, Dryden N, Peste A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural

networks. Journal of Machine Learning Research. 2021;**22**(241):1-124

[35] Zhang S, Zidong D, Zhang L, Lan H, Liu S, Li L, et al. Cambricon-x: An accelerator for sparse neural networks. In: In 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Taipei, Taiwan: IEEE; 2016. pp. 1-12

[36] Parashar A, Rhu M, Mukkara A, Puglielli A, Venkatesan R, Khailany B, et al. Scnn: An accelerator for compressed-sparse convolutional neural networks. ACM SIGARCH computer architecture news. 2017;**45**(2): 27-40

[37] Gondimalla A, Chesnut N, Thottethodi M, Vijaykumar TN. Sparten: A sparse tensor accelerator for convolutional neural networks. In: Proceedings of the 52nd Annual IEEE/ ACM International Symposium on Microarchitecture. 2019. pp. 151-165

[38] Yang D, Ghasemazar A, Ren X, Golub M, Lemieux G, Lis M. Procrustes: A dataflow and accelerator for sparse deep neural network training. In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2020. pp. 711-724

[39] Mead C, Ismail M. Analog VLSI Implementation of Neural Systems. Vol. 80. New York, NY, USA: Springer, Science & Business Media; 1989

[40] Douglas R, Mahowald M, Mead C. Neuromorphic analogue vlsi. Annual Review of Neuroscience. 1995;**18**: 255-281

[41] Schemmel J, Fieres J, Meier K. Wafer-scale integration of analog neural networks. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). Hong Kong: IEEE; 2008. pp. 431-438

[42] Furber SB, Lester DR, Plana LA, Garside JD, Painkras E, Temple S, et al. Overview of the spinnaker system architecture. IEEE Transactions on Computers. 2012;**62**(12):2454-2467

[43] Frenkel C, Lefebvre M, Legat J-D, Bol D. A 0.086-mm ^{2} 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. IEEE Transactions on Biomedical Circuits and Systems. 2018; **13**(1):145-158

[44] Frenkel C, Legat J-D, Bol D. Morphic: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. IEEE Transactions on Biomedical Circuits and Systems. 2019;**13**(5):999-1010

[45] Stuijt J, Sifalakis M, Yousefzadeh A, Corradi F. μ brain: An event-driven and fully synthesizable architecture for spiking neural networks. Frontiers in Neuroscience. 2021;**15**:538

[46] Merolla PA, Arthur JV, Alvarez-Icaza R, Cassidy AS, Sawada J, Akopyan F, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science. 2014;**345**(6197): 668-673

[47] Davies M, Srinivasa N, Lin T-H, Chinya G, Cao Y, Choday SH, et al. Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro. 2018;**38**(1):82-99

[48] BrainChip Holdings Ltd. Akida neuromorphic system-on-chip. Available from: https://brainchip.com/akida-neura l-processor-soc/.

[49] Benjamin BV, Gao P, McQuinn E, Choudhary S, Chandrasekaran AR, Bussat J-M, et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. Proceedings of the IEEE. 2014;**102**(5):699-716

[50] Hawkins J, Ahmad S, Cui Y. A theory of how columns in the neocortex enable learning the structure of the world. Frontiers in Neural Circuits. 2017; **11**:81

[51] Hawkins J, Lewis M, Klukas M, Purdy S, Ahmad S. A framework for intelligence and cortical function based on grid cells in the neocortex. Frontiers in Neural Circuits. 2019;**12**:121

[52] Lewis M, Purdy S, Ahmad S, Hawkins J. Locations in the neocortex: A theory of sensorimotor object recognition using cortical grid cells. Frontiers in Neural Circuits. 2019;**13**:22

[53] Kjell Jørgen Hole and Subutai Ahmad. A thousand brains: Toward biologically constrained ai. SN Applied Sciences. 2021;**3**(8):1-14

[54] Heaven D. Deep trouble for deep learning. Nature. 2019;**574**(7777): 163-166

[55] Smith JE. A macrocolumn architecture implemented with temporal (spiking) neurons. arXiv preprint arXiv: 2207.05081. 2022

[56] Nair H, Tan C, Zeng M, Mengshoel OJ, Shen JP. Attrinet: Learning mid-level features for human activity recognition with deep belief networks. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019. pp. 510-517

[57] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems. 2019; **32**:8026-8037

[58] Dau HA, Bagnall A, Kamgar K, Yeh C-CM, Zhu Y, Gharghabi S, et al. The ucr time series archive. IEEE/CAA Journal of Automatica Sinica. 2019;**6**(6): 1293-1305

[59] Yann LeCun, Corinna Cortes, Christopher Burges JC. The mnist database of handwritten digits. Availalbe from: http://yann.lecun.com/exdb/ mnist/.

[60] John Dian F, Vahidnia R, Rahmati A. Wearables and the internet of things (iot), applications, opportunities, and challenges: A survey. IEEE Access. 2020; **8**:69200-69211