

Hardware based Spatio-Temporal Neural Processing Backend for Imaging Sensors: Towards a Smart Camera

Samiran Ganguly, Yunfei Gu, Mircea R. Stan, and Avik W. Ghosh

Charles L. Brown Dept. of Electrical and Computer Engineering
University of Virginia, Charlottesville, VA 22904

ABSTRACT

In this work we show how we can build a technology platform for cognitive imaging sensors using recent advances in recurrent neural network architectures and training methods inspired from biology. We demonstrate learning and processing tasks specific to imaging sensors, including enhancement of sensitivity and signal-to-noise ratio (SNR) purely through neural filtering beyond the fundamental limits sensor materials, and inferencing and spatio-temporal pattern recognition capabilities of these networks with applications in object detection, motion tracking and prediction. We then show designs of unit hardware cells built using complementary metal-oxide semiconductor (CMOS) and emerging materials technologies for ultra-compact and energy-efficient embedded neural processors for smart cameras.

Keywords: Reservoir Computing, Hierarchical Temporal Memory, Convolutional Neural Networks, Spatio-Temporal Learning, Neural Filtering, Neural Signal Processing, Neural Image Processing

1. INTRODUCTION

Last three decades have seen the progress of Neural Networks from a statisticians playbook to the technology behemoth running the modern information and communications technology (ICT) industry. This has largely been made possible by advances in very large scale integration (VLSI) and transistor scaling providing evermore powerful hardware to run neural network software. This scaling is slowing down and heading towards real physical limits of atomic dimensions. As a result, the Moore's Law is now being reinterpreted as a call for deep multi-functional integration of erstwhile loosely coupled sub-systems: sensing, memory, and logic, in a single computing substrate with a resulting increase in the "user value". In this work, we illustrate how we can leverage these advances to build a technology platform for truly cognitive imaging sensors, inspired from biology.

We first discuss recurrent neural network based architectures designed to process data with features laying in both spatial and temporal dimensions (e.g. video) and associated learning techniques, in particular: Convolutional Neural Network (CNN), Reservoir Computing (RC), and Hierarchical Temporal Memory (HTM), whose hardware implementation and operation are feasible with present day technology. We then demonstrate, using simulations, the use of these networks for learning and processing tasks specific to imaging sensors: a) Enhancing sensitivity and SNR/D* purely through neural filtering that goes beyond the fundamental limits of detection set by the material properties. b) Spatio-Temporal inferencing and pattern recognition capabilities built within these networks that can be used for object detection, motion tracking, and prediction.

We then conclude by illustrating how unit hardware cells can be designed using both conventional CMOS as well as emerging nano-materials based technologies, such as spintronics and memristors, which can enable development of ultra-compact and energy-efficient embedded spatio-temporal neural processors for smart cameras.

Further author information: (Send correspondence to Samiran Ganguly)

Samiran Ganguly: E-mail: sganguly@virginia.edu

Mircea R. Stan: E-mail: mircea@virginia.edu

Avik W. Ghosh: E-mail: ag7rq@virginia.edu

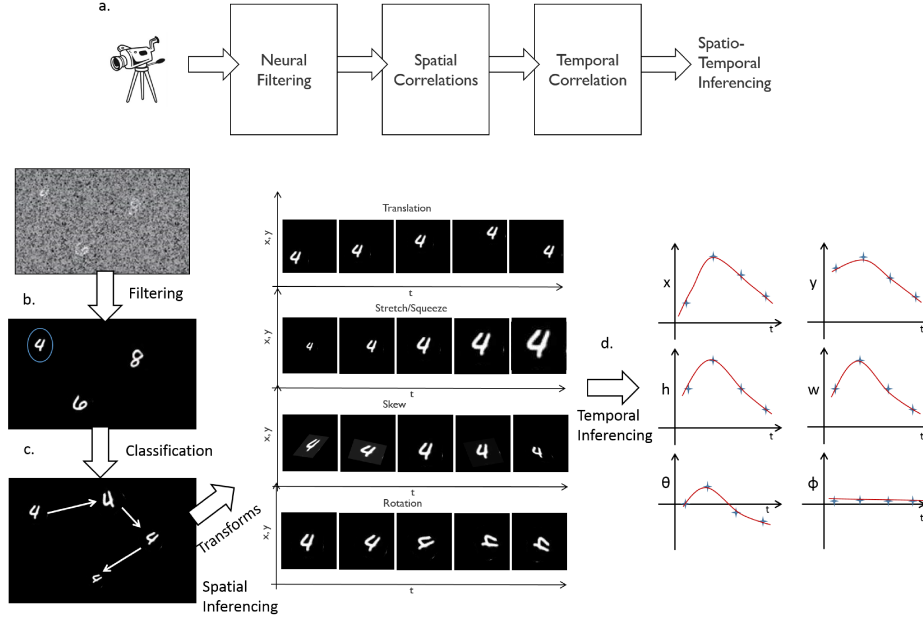


Figure 1. a. Proposed neural filtering and processing architecture and the data flow. b. Neural filtering cleans up noisy images c. Spatial inferencing including classifications and transforms that identify features and extract its position, size, and degree of rotation and skew. d. Temporal inferencing uses the spatial correlations to generate equations of motion of the identified features through real-time learning.

2. COGNITIVE IMAGE PROCESSING ARCHITECTURE: DESIGN AND PROTOCOL

Cognitive image processing, being a broad term can mean different things in different contexts. As examples, for social media networks, the goals are primarily recognizing and mapping faces in an image to users, whereas in a self-driving automotive it would be to recognize street signs and lane markings in variable environmental conditions. Indeed all such “cognitive tasks” proceed from an application goal to most optimized neural network designs to achieve those tasks.

In our present work we define the goal of a smart camera platform to be able to identify and track a user defined feature from the camera’s video feed. We want to formalize the scope of our design to able to perform the following tasks:

- Extraction of clean signal from the noisy images taken by the camera, including various distortions such as haziness, speckles, fading etc.
- Identification of a pre-defined feature in an image frame, and extraction of spatial information of the feature, such as it’s position, size, rotation etc. which can be considered as the canonical variables of motion for the feature.
- Extraction of the generative functions underlying the variables of motion and being able to predict the expected evolution of these variables.

Our goal is not so much as to identify/classify a large set of features in a collection of images, which is the most popular task in neural image processing literature, but rather user-defined feature tracking in real time operation of a camera.

The proposed architecture for this processing task is shown in fig.1a. We assemble three separate blocks which build the logic of processing listed above. This modularity of functionality allows us to select, design, and optimize each of these blocks independent of each other. Since this is an ongoing project, we have chosen to implement these three blocks using the following three types of neural networks, and the final design is subject to optimization as the platform matures.

- Neural Filtering: Reservoir Computing, particularly Echo-State Networks (ESNs).
- Spatial Inferencing: Enhanced Region-based Convolutional Neural Networks (R-CNNs).
- Temporal Inferencing: Hierarchical Temporal Memory (HTM).

The ESN based neural filters extract a signal from the noise by learning to invert the time-dependent distortions in the images (fig.1b) introduced by the sensor and its operational limitations. Enhanced region-based CNNs extract the feature and its variables of motion from samples of the camera video feed, in form of a spatial correlation tuple. This tuple consists of the following quantities: $\{Tag, p, x, y, h, w, \theta, \phi_x, \phi_y\}$. The Tag item identifies the feature, in case we want to develop a multi-feature tracking camera, p represents the probability of feature being in the image, (x, y) represent the most probable position of the feature in the image frame, (h, w) represent the size of the feature in the image frame, θ is the rotation of the features with respect to the initial position, and (ϕ_x, ϕ_y) represents the skew in the feature due to rotation in the 3D and its projection on the 2D image frame. A collection of HTMs then develop equations of motions from a time-sequence of these tuples.

3. SPATIO-TEMPORAL NEURAL NETWORKS

We briefly describe the three types of neural networks we use in this work.

3.1 Reservoir Computing

Reservoir Computing (RC) is a model of computing built to handle time-varying data and learn correlations and patterns in them.^{1,2} RC consists of a collection of loosely coupled neurons with structural recurrence (feedback-loops) on which a time-varying signal $u(t)$ is imposed. The state of the reservoir $x(t)$ as a result at any time is given by a combination of previous states $x(t-1)$ and the input signal $u(t)$. The new state $x(t)$ as a result is an integral of the signal over its previous samples $u(t), u(t-1), u(t-2) \dots$. This gives rise to temporal-memory in the reservoir (also called echo-states, past signal being echoed throughout the network), and allows it to be used as a temporal correlator.³ To prevent runaway positive feedback of signal (energy) driving the reservoir dynamics to chaos, a decay term is incorporated which fades the effect of the previous samples during the integration (dissipation due to friction). The dynamical equation for the reservoir shown in fig.2a is given by:

$$\frac{dx}{dt} = -\eta x + \alpha \tanh(W^{self}x(t) + W^{in}u(t) + W^{fb}y(t)) \quad (1)$$

$$y(t) = W^{out}x(t) \quad (2)$$

The output of the RC is extracted by a weighted sum of all the states of the reservoir. During learning, only the weight W^{out} is adjusted using a simple regression. The training can be performed in both batch as well as online modes.

It should be noted that the RCs are limited by the choices of network parameters such as the size of the network, decay rate, spectral radius of W^{self} etc. which fix the characteristic window size and the envelope shape of the temporal integration in the reservoir. Therefore, we choose HTMs rather than RCs for temporal inferencing, since we do not want to a priori fix these critical characteristics. We have chosen to use RC for the task of neural filtering, as it has been used to successfully perform an equivalent task in digital communication: channel equalization.⁴ This neural filtering can be applied at both pixel or feature level.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (ConvNets or CNNs) are an application of learning techniques applied to images, and are an extension to traditional image processing techniques. The dataframe is assumed to have a meaningful 2D spatial relation, i.e. pixels close to each other are highly correlated and compose parts of an image.⁵

The central operation of CNNs is the convolution operation (fig.3a), where a “kernel” $U_{j \times k}$ matrix is multiplied to an image $A_{m \times n}$, ($m > j, n > k$) in a sliding manner. Mathematically it is represented as the linear operation (in spatial domain) as:

$$B = U * A \quad (3)$$

It should be noted that the result of the convolution operation B is often trimmed, using a pooling operation, where at any point, a few adjacent pixels are represented using a single pixel data. As an example, a pooling window of 3×3 will reduce a 100×100 image to a 33×33 image.

In traditional image processing,⁶ the U is pre-calculated and designed to perform a specific operation (say edge detection, blurring, contrast enhancement etc.). In CNNs, these kernels are obtained through backpropagation techniques. The central purpose of most CNN applications are to classify features present in the image into target categories. As an example, consider an image consists of hand-written number and letters. The CNN can be trained to extract the number or a letter occurring in the image.

CNNs in general need to be trained in a supervised setting, i.e. during its training it needs to be told the error in classification at its output, so that the CNN can adjust the convolutional kernels. Therefore, training of the targeted feature set for our platform will need to be performed offline. However, once trained on that particular feature set, the rest of the platform can learn, adapt, and predict online and in real-time.

3.3 Hierarchical Temporal Memories

HTMs are a neural network specialized for time-series data/temporal modeling, prediction and anomaly detection. Its design and learning methodology has been developed keeping in mind the design and operation of neocortices in sentient organisms, which perform the task of temporal integration of information and prediction seamlessly and in real time.^{7,8}

In HTM, the neurons or “cells” are organized in a 3D stack as shown in fig.4a. The structural interconnectivity in these cells is designed to be highly recurrent (i.e. with feedback loops) which allows temporal correlations between samples to persist in the network hence the name temporal memory. In HTM a Hebbian like rule is implemented for learning, where the connections between the cells (dendritic connections) is formed or unformed depending upon the degree of correlations between two cells during the operation of the network.⁹ In fact, there is no separate learning phase for the complete network and the dendritic connections are updated online and in real-time during the processing,¹⁰ which makes it suitable for temporal inferencing.

Since the temporal memory size can be dynamically changed in the HTM, unlike RC, it is a more suitable network for the task of temporal correlation.

4. NEURAL FILTERING AND PROCESSING TASKS

4.1 Filtering by Learning

The neural filtering presently implemented is inspired by an equivalent task in digital communication called channel equalization. The source of the information produces a bitstream, which passes through a channel and is distorted in a non-linear fashion. Therefore it is not possible to simply apply a linear filter to extract the pure signal component. Instead, the effect of channel has to be reversed by creating an inverse of the channel characteristics, typically done in frequency domain using principal component analysis based methods.

We use the temporal modeling capabilities of RC to perform this task, where the channel is the detector which introduces distortions and noise due to its physical limitations. The process is illustrated in fig.2b, where the detector (channel) introduces distortion and noise on the original signal $d(t)$ through an unknown function $q(z)$ and as a result the output is given by:

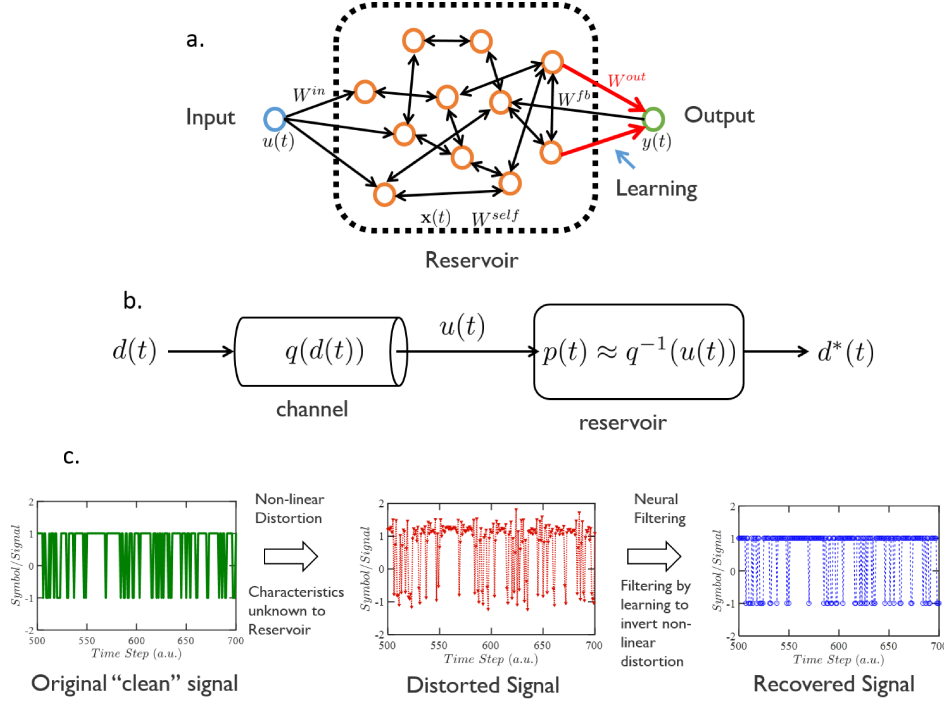


Figure 2. a. Reservoir Computer schematic. A reservoir is composed of loosely and randomly coupled set of neurons whose weights are not changed during the training. The interconnectedness of the reservoir allows a given time sample of the signal to persist in time within the network and get integrated with newer samples. The only weight changed during the training is W^{out} which can be adjusted to learn the correlation of the reservoir for specific tasks. b. Central idea behind neural filtering. The channel distorts the signal $d(t)$ by a non-linear function $q(d)$. The reservoir is trained to invert $q(d)$ to reconstruct the original signal. c. A simulation example showing the recovery of the signal in time.

$$u(t) = q(d(t)) \quad (4)$$

During the training process, the RC is fed this distorted signal $u(t)$ as its input and targeted to reproduced the original $d(t)$ by minimizing the difference from its actual output $d^*(t)$. This in effect turns the reservoir into a inverse modeler:

$$p(z) = q^{-1}(z) \quad (5)$$

$$d^*(t) = p(u(t)) \quad (6)$$

Once the inverse model has been learned, the reservoir can now be used for neural filtering tasks as shown in fig.2c, where an original clean signal can be recovered with high accuracy (normalized root mean square error (NRMSE) $< 10^{-3}$).

4.2 Spatial Inferencing

Spatial inferencing involves extracting the probability of a feature being present in the image along with its variables of motions. However, regular CNNs do not provide any of the variables of motion for the identified feature in the image. For the purpose of tracking an object in an image frame, it is important that the CNN provides the positional and transformation information, i.e. the full tuple described previously, for the temporal inferencing.

To achieve this, we apply a region based CNN (R-CNN) which also extracts the most probable smallest bounding box that contains this feature within the image frame.¹¹ This can be achieved by using sliding windows of various sizes exhaustively over the image frame or through a combination of segmenting the image into various region proposals/regions of interest, and then searching within the regions.^{12,13} In addition, rotation and skew information can also be extracted by geometrically transforming the regions of interests from straight to rotated rectangles with best search incidences to infer the angle of rotation,¹⁴ and using parallelograms instead of rectangles to obtain the skew angles (fig.3c).

Therefore, the enhanced R-CNN allows us to form the spatial correlation tuples $\{\text{Tag}, p, x, y, h, w, \theta, \phi_x, \phi_y\}^T$. Storing the tuple information from the previous image sample in a video allows us to obtain the next sample's tuple faster compared to a purely static image frames because in an time sequenced image samples, physical objects shift only by a finitely small amount. In addition, if the camera is embedded on a mobile platform, sensor data from the platform's motion can be used to compensate for the drift by developing more region proposals.

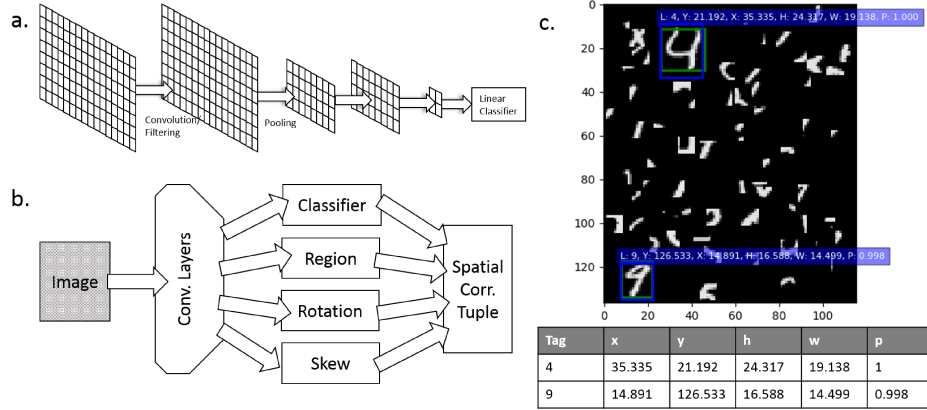


Figure 3. a. Convolutional Neural Networks showing convolution and pooling operations. Feature maps are extracted at each convolutional layer and compactified using pooling. b. An enhanced CNN using extra region, rotation and skew flows that together build the spatial correlation tuples. c. An example of a region based extraction of feature location, size, and probability (see¹⁵ for an implementation). The implemented R-CNN can extract multiple features and their position and sizes with high accuracy, as can be seen from the table. Rotation and skew flows can be implemented in a similar modular fashion.

4.3 Temporal Inferencing

Temporal inferencing involves developing a generative model for the time-sequenced samples of the spatial correlation tuples. The seven variables of motion are separately fed to seven separate HTMs (fig.4a) through a sparse encoding layer and the cortical learning algorithm is applied to the individual data streams. In HTMs, the learning and prediction are interleaved together, which makes it an online and real-time process.

Fig.4b shows an example of temporal learning and prediction relevant to our application. A trajectory in $x - y$ plane is generated using a deterministic set of functions. Individual samples of the coordinates $\{x(t), y(t)\}$ are provided as a list to two separate HTMs. It can be seen that the HTMs can learn and model the trajectory successfully, which proves the validity of the approach. Fig.4c shows an anomaly map for the samples, where anomaly is defined as the difference between the actual trajectory and predicted trajectory. There is an initial period of 0 anomaly in the samples, as the HTMs learn and build the model, but after that the HTM runs in a simultaneous predictive-generative mode. For very long periods of operation, the predictive capability of the HTMs improve for smooth trajectories and any “sudden change” in the trajectory will result in large amount of anomalies, which can be reported as an “alarmed event” to the user of the platform.

This approach to model the positional ($\{x(t), y(t)\}$) variables of motion can also be extended to the extensional (size $\{h(t), w(t)\}$), rotational ($\theta(t)$), and skew ($\{\phi_x(t), \phi_y(t)\}$) variables of motion as well. Generative models of the full tuple can then be used by a cyber-physical system (CPS) in the lines of “sensory neuron $\rightarrow$ motor neuron” connection in sentient organisms for applications such object tracking, locomotion etc.

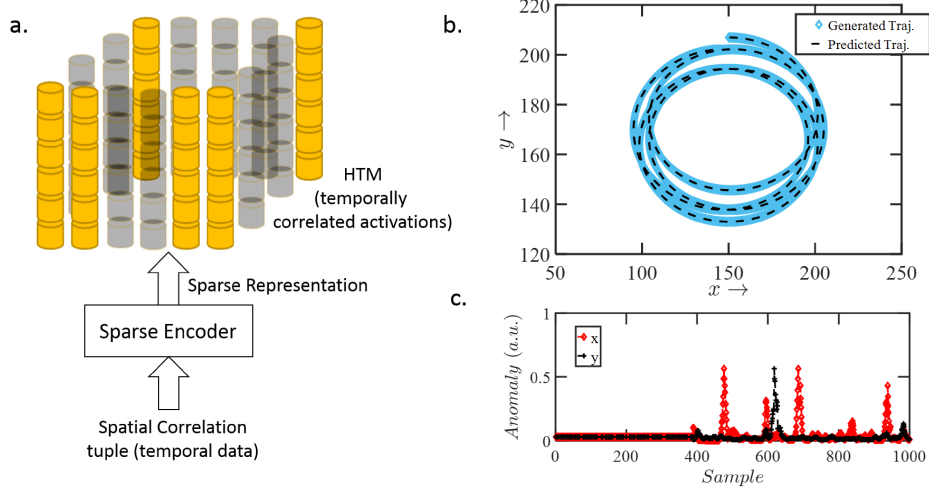


Figure 4. a. HTM with temporally correlated activations. b. An HTM based prediction of time series data. A complex trajectory of motion generated by an underlying model not known to HTM can be learned and predicted simultaneously from the time ordered samples of the trajectory data stream. c. Anomaly between predicted model and actual data.

5. HARDWARE FOR NEURAL NETWORKS

Most state-of-the-art and advertised neural networks are developed and run on Graphical Processing Units (GPUs), primarily due to their ability to perform linear algebra operations such as dot products directly in hardware. This illustrates the mismatch between the von-Neumann architecture with its separation of memory and logic and the needs of neural network designs. In Neural networks, the memory (synaptic weights) is spread throughout the computational fabric, and the basic unit of operations, dot-product and thresholding do not map directly to NAND gate based logic designs.

As a result there is a concerted effort to re-imagine hardware designs that can closely match the neural architectures.¹⁶ Large organizations such as Google, Intel, IBM, Micron etc. are already building prototype Deep Learning accelerators that implement hardware architectures matching neural network designs at close-to-code level. This effort is still broadly at the level of emulation of neural networks using Boolean computing units (CMOS transistors) with better logic primitives (hardware dot-product etc.). There is also a parallel effort in developing custom designed CMOS based “digital neurons” implemented in FPGAs as well as exploring nano-materials such as spintronics and various memristive materials, and physical phenomena governing them to develop better ultra-compact hardware units that encompass some/all physical characteristics necessary to implement a neuron and synaptic weight adjustment during learning directly.

It is largely expected that the uptake of hardware neurons will happen first in the emulation modes, where CMOS based “digital neurons” will implement large neural networks. Eventually these neural fabrics may be enhanced and scaled using the nano-materials based neurons, sparking a new Moore’s Law of electronics.

5.1 Field Programmable Gate Arrays (FPGAs) and Digital Neurons

FPGAs are the platform of choice for prototyping new architecture designs. They are composed of large arrays of programmable Boolean gates and interconnection networks which can be used to build basic building blocks from bottom up such as ADDER, MUX, register, LFSR, Encoder and Decoders, Content Addressable Memories etc. It is possible to build complex logic designs in FPGAs by combining these blocks programmatically using well developed workflow based on Hardware Description Languages (HDLs) (Verilog, SystemC, SystemVerilog) which describe the dataflow through the logic blocks (also called register-transfer level design).

A possible example of a digital neuron and synapse that can be designed and implemented on FPGAs is shown in 5a. The synaptic weights are stored in a memory buffer and used to create the argument ($z = wx + b$) for neural activation using the PROD+SUM block, which can be implemented using separate MULTs (such as

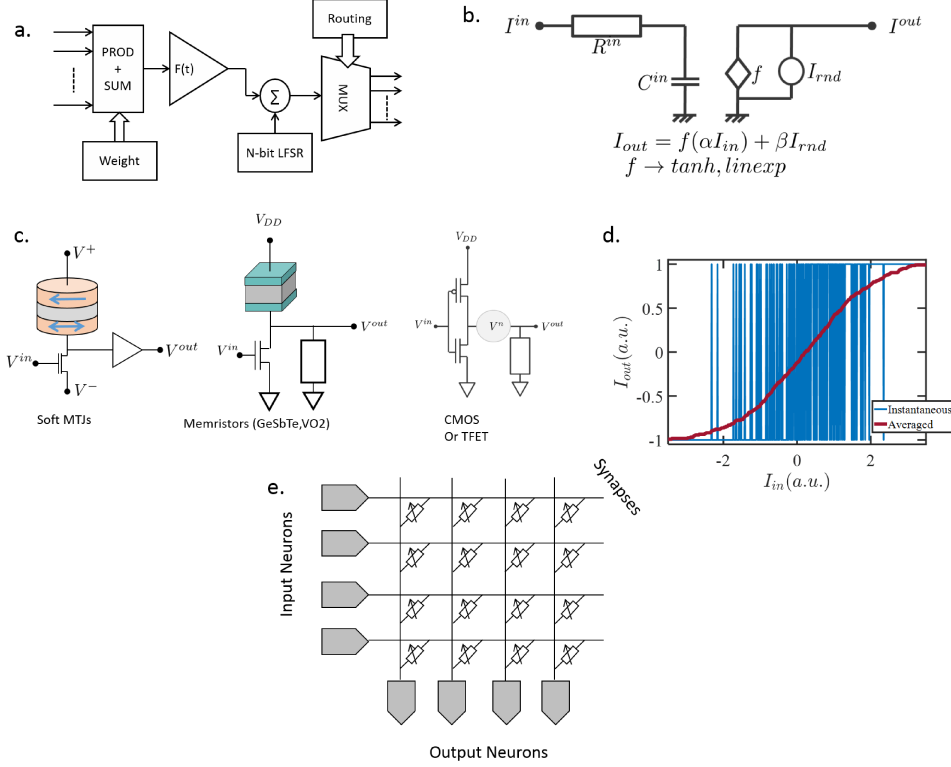


Figure 5. a. An example digital hardware neural model that can be implemented in FPGAs or ASICs. b. General hardware neural model. c. The neural model can also be obtained directly by ultra-compact hardware units built using emerging nano-materials that show stochastic switching: soft-magnets, memristors, and subthreshold TFETs. d. An example switching characteristics of the neural model implemented using the soft-magnet MRAM, which encodes the neural output in the average response (red averaged current) which has a logistic like activation. e. Representational layout of hardware neural networks using a memristive cross-bar array as synaptic connections, while the neurons sit at the periphery. The individual interconnections (synaptic weights) can be programmed during training phase by programming the controllable resistance of the memristor connecting the row and the column between the input and output neurons. Deep networks can be built by cascading/stacking this design repeatedly.

Wallace and Dadda trees) and ADDERs (Ripple Carry, Koggestone etc.) or using Fused Multiply-Add (FMA) cores. The thresholding function $f(wx + b)$ can be implemented either using low order power law approximations of $f(z)$ function, or using a lookup table based method, in which the numerical value of the argument z is mapped to the memory address where the result of the computation is stored (also called hashing). Linear feedback shift registers (LFSRs) can be used to generate random numbers if stochasticity is necessary for the computation, and the fan-out routing logic between the various neurons is implemented using a MUX.

From the above design it is clear that the performance of these digital neurons can be improved significantly if the synaptic weighing and addition, thresholding, and routing can be implemented using compact hardware units whose physics matches these requirements, since implementing these blocks in FPGAs or ASICs using CMOS based Boolean gates still require a lot of silicon area and associated energy costs. Emerging nano-materials based technology such as spintronics^{17,18} and memristors^{16,19,20} are of particular interest for such applications.

5.2 Emerging Technologies: Spintronics and Memristors

Spintronics and memristors are emerging technologies that do not use electronic charge as the state variable of computation, but rather use direction of magnetization of a thin magnetic layer (expressed as a vector $\hat{m}$), and internal chemical configuration (expressed by chemical potential μ) respectively. These variables effect the electrical readout of a device built using them, typically by controlling the resistance of the structure. These state

variables can be manipulated by passing a current through them enabling a write mechanism. Most prominent use case for these materials at present is in non-volatile memory, where spintronics in particular has made a niche for itself as a possible successor of NAND flash. We describe how these materials can be used to build hardware units for neurons. We will focus on spintronics, but the central principles can be transferred to memristive systems as well.

A general hardware neuron model is shown in fig.5b. The input side has a natural delay line built from a resistor and a capacitor, which stores the state of the cell. The charge on the capacitor Q_{in} determines whether the device will switch ($Q_{in} \geq Q_c$) or not ($Q_{in} < Q_c$). The output transfer function $f(\alpha \frac{dQ_{in}}{dt})$ is a non-linear one, typically $\tanh(z)$ but can be others like rectified linear units (ReLU) etc. There is also an additive white Gaussian noise (AWGN) current source in parallel to the transfer function and generates the output noise for the neuron. This noise is particularly critical in implementing the stochasticity that is needed in modern neural learning and optimization algorithms. The characteristic switching function is given by:

$$I_{out} = f(\alpha \frac{dQ_{in}}{dt}) + \beta I_{rnd} \quad (7)$$

This generalized behavior is demonstrated by a variety of materials (fig.5c). Magnetic Tunnel Junction (see²¹ for a comprehensive review) built using soft-magnets or super-paramagnets are a particularly attractive option for implementing stochastic neurons, even though deterministic neurons using hard-magnets have also been proposed, e.g. see.²² The degree of hardness of magnets is characterized by the height of its internal energy barrier separating the two minimum energy states. This energy barrier is a function of its material parameters (saturation magnetization M_s , uniaxial anisotropy field strength (H_k), and volume Ω) and is given by:

$$U = \frac{M_s H_k \Omega}{2} \quad (8)$$

This determines the state retention time (degree of hardness) that can be captured by:

$$\tau = \tau_0 \exp(\frac{U}{kT}) \quad (9)$$

where attempt frequency τ_0 is typically $0.1 - 1 \text{ ns}$. Therefore for $U \leq 1kT$, the retention time of the magnet is a few ns or less. An input current I_{in} , converted to a spin transfer torque by passing through a polarizing layer, if strong enough can align the magnet to a preferable direction. Therefore, if we sweep the I_{in} , the instantaneous response of the device is given by:

$$I_{out} \propto m_z^{out} = \text{sgn}(\tanh(\kappa m_z^{in}(I_{in}) + \text{rnd}(-1, +1))) \quad (10)$$

However, the long time average of the output I_{out} follows a $\tanh(z)$ characteristic, and therefore matches the requirement of a binary stochastic neuron, which is the general neuron model used in most deep learning algorithms. The metallic leads of the device can automatically add the currents, whereas the synaptic weights can be implemented by controlled conductors, since $I = GV$. A hybrid crossbar array of programmable memristors can compactly implement these programmable synaptic connections at the junctions and the neurons at the periphery (fig.5e).

These networks can then be used in a variety of neural network designs, including combinatorial optimization solvers, Boltzmann Machines (BM) and Deep Belief Networks (DBNs), and Reservoir Computers (RCs), some of the central types of neural networks we have discussed here. Please see the refs.²³⁻²⁶ for more detailed exposition and applications.

6. SUMMARY

In this work, we presented an architecture for a smart camera platform that will eventually employ a combination of best-in-class neural networks to perform its principle tasks: a) neural filtering, b) spatio-temporal inferencing. The platform has been designed with an application of user defined feature tracking and prediction in mind. The platform is generic enough to be applied to a wide variety of established and emerging fields including defense, automated vehicles, and autonomous sensor networks.

We described the three classes of neural networks used in the work and demonstrated learning and prediction tasks individually from these networks. Using Reservoir Computing, we demonstrated signal recovery by inverse modeling of a noisy detector (fig.2). Using enhanced CNNs we demonstrated extraction of a feature, along with its variables of motion within an image frame (fig.3). Using HTM, we demonstrated learning and prediction of the trajectory of a feature in the image frame (fig.4).

We further pointed out possible hardware implementation of neurons for energy-efficient processing that can be embedded close to the image sensor. We discussed how digital neurons can be implemented on present day FPGAs and how it is possible to leverage the physics of emerging nano-materials to enhance conventional CMOS based computing in directly implementing neural networks in large scale circuits.

ACKNOWLEDGMENTS

This work was supported in part by the NSF I/UCRC on Multi-functional Integrated System Technology (MIST) Center IIP-1439644, IIP-1738752 and IIP-1439680. We would like to thank Dr. Kerem Yunus Camsari for useful discussions on soft-magnet based stochastic neurons.

REFERENCES

- [1] Jaeger, H., “Technical Report GMD Report 148,” *German National Research Center for Information Technology* (2001).
- [2] Maass, W., Natschlger, T., and Markram, H., “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural computation* **14**(11), 2531–2560 (2002).
- [3] Jaeger, H. and Haas, H., “Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication,” *Science (New York, N. Y.)* **304**, 78–80 (Apr. 2004).
- [4] Boccato, L., Lopes, A., Attux, R., and Von Zuben, F. J., “An echo state network architecture based on Volterra filtering and PCA with application to the channel equalization problem,” in [*Neural Networks (IJCNN), The 2011 International Joint Conference on*], 580–587, IEEE (2011).
- [5] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” in [*Advances in neural information processing systems*], 1097–1105 (2012).
- [6] Jain, A. K., [*Fundamentals of Digital Image Processing*], Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1989).
- [7] Hawkins, J. and Blakeslee, S., [*On Intelligence: How a New Understanding of the Brain Will Lead to the Creation of Truly Intelligent Machines*], Macmillan (Apr. 2007). Google-Books-ID: Qg2dmntfxmQC.
- [8] Hawkins, J., Ahmad, S., and Cui, Y., “A Theory of How Columns in the Neocortex Enable Learning the Structure of the World,” *Frontiers in Neural Circuits* **11** (2017).
- [9] Hawkins, J. and Ahmad, S., “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,” *Frontiers in Neural Circuits* **10** (2016).
- [10] Cui, Y., Ahmad, S., and Hawkins, J., “Continuous Online Sequence Learning with an Unsupervised Neural Network Model,” *Neural Computation* **28**, 2474–2504 (Sept. 2016).
- [11] Gidaris, S. and Komodakis, N., “Object Detection via a Multi-region and Semantic Segmentation-Aware CNN Model,” in [*Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*], 1134–1142, IEEE Computer Society (2015).
- [12] Ren, S., He, K., Girshick, R., and Sun, J., “Faster r-cnn: Towards real-time object detection with region proposal networks,” in [*Advances in neural information processing systems*], 91–99 (2015).

- [13] He, K., Gkioxari, G., Dollr, P., and Girshick, R., “Mask r-cnn,” in [*Computer Vision (ICCV), 2017 IEEE International Conference on*], 2980–2988, IEEE (2017).
- [14] Liu, Z., Hu, J., Weng, L., and Yang, Y., “Rotated region based CNN for ship detection,” in [*2017 IEEE International Conference on Image Processing. Piscataway, NJ: IEEE*], (2017).
- [15] Liang, K., “tf-Faster-RCNN: TensorFlow implementation of Faster R-CNN,” (Mar. 2018). <https://github.com/kevinliang/tf-Faster-RCNN/>.
- [16] Misra, J. and Saha, I., “Artificial neural networks in hardware: A survey of two decades of progress,” *Neurocomputing* **74**(1-3), 239–255 (2010).
- [17] Debashis, P., Faria, R., Camsari, K. Y., Appenzeller, J., Datta, S., and Chen, Z., “Experimental demonstration of nanomagnet networks as hardware for Ising computing,” in [*Electron Devices Meeting (IEDM), 2016 IEEE International*], 34–3, IEEE (2016).
- [18] Camsari, K. Y., Salahuddin, S., and Datta, S., “Implementing p-bits with Embedded MTJ,” *IEEE Electron Device Letters* **38**(12), 1767–1770 (2017).
- [19] Du, C., Cai, F., Zidan, M. A., Ma, W., Lee, S. H., and Lu, W. D., “Reservoir computing using dynamic memristors for temporal information processing,” *Nature Communications* **8**, 2204 (Dec. 2017).
- [20] Kulkarni, M. S. and Teuscher, C., “Memristor-based reservoir computing,” in [*Nanoscale Architectures (NANOARCH), 2012 IEEE/ACM International Symposium on*], 226–232, IEEE (2012).
- [21] Kent, A. D. and Worledge, D. C., “A new spin on magnetic memories,” *Nature Nanotechnology* (Mar. 2015).
- [22] Quang Diep, V., Sutton, B., Behin-Aein, B., and Datta, S., “Spin switches for compact implementation of neuron and synapse,” *Applied Physics Letters* **104**, 222405 (June 2014).
- [23] Ganguly, S., Camsari, K. Y., and Datta, S., “Evaluating Spintronic Devices Using the Modular Approach,” *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* **2**, 51–60 (Dec. 2016).
- [24] Camsari, K. Y., Faria, R., Sutton, B. M., and Datta, S., “Stochastic p-Bits for Invertible Logic,” *Physical Review X* **7**, 031014 (July 2017).
- [25] Sutton, B., Camsari, K. Y., Behin-Aein, B., and Datta, S., “Intrinsic optimization using stochastic nanomagnets,” *Scientific Reports* **7**, 44370 (Mar. 2017).
- [26] Ganguly, S., Camsari, K. Y., and Ghosh, A. W., “Reservoir Computing using Stochastic p-Bits,” *arXiv:1709.10211 [cond-mat]* (Sept. 2017). arXiv: 1709.10211.