Seminar on Stochastic Computational Deep Belief Network

Mohammed Muddasser

Chair for Hardware Oriented Computer Science

20/07/2020

Agenda

1. Deep Learning
2. Stochastic Computation (SC)
3. Deep Belief Networks (DBN)
4. DBN Working Principle and Hardware implementations:
    - Unsupervised phase
    - Supervised phase
5. Evaluation
6. Conclusion

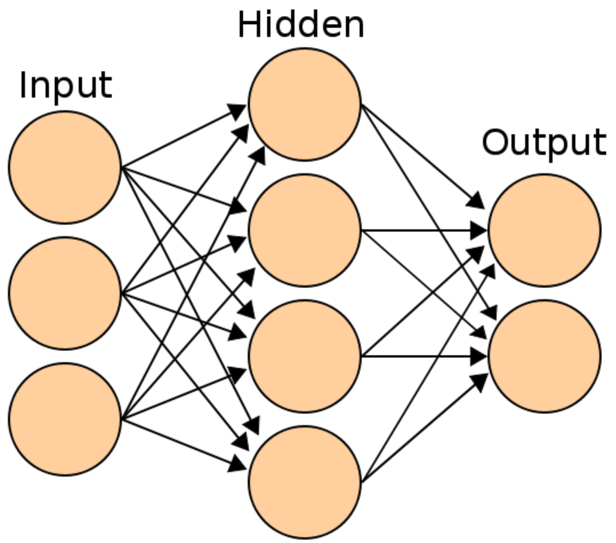
Deep Learning

Class of machine learning algorithms.

Neural Networks (NN). What are they?

Deep Learning and Deep Neural Networks (DNNs).

- Detection and Pattern Recognition problems.
- Types of NN's – Artificial neural network (ANN), Deep Belief Network (DBN), Recurrent Neural Networks (RNN), General Adversarial Network (GAN) etc.
- Efficient NN design and architecture search.



Simple Neural Network

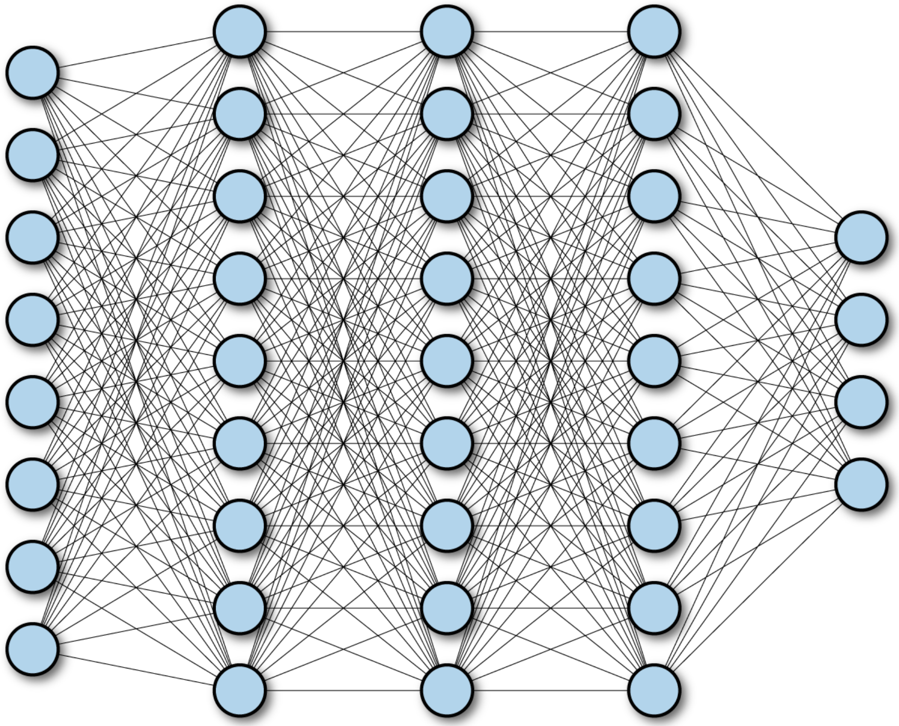
Source: Wikimedia common http://bit.ly/2xP70qL

Deep Learning

A Fully connected Deep Neural Network.

DNNs in general involve:

- Training the network.
- Inference – To infer result from trained network.



Fully Connected DNN

Source: O'Reilly, https://bit.ly/2ZmP997

Deep Learning

Hardware realization of a DNN typically involves:

- Design for inference.
- Fixed circuit and no online learning capabilities.

Yet, they can be computationally expensive:

- Large chip area.
- High latency.
- High power consumption.

Always a trade-off between these parameters.

Seminar discussion about, an energy-efficient Stochastic Computational Deep Belief Network (SC-DBN) with online-learning.

Stochastic Computation

Bernoulli bit stream $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_\mathrm{n})$, such that:

$$\mathrm{A} = \mathrm{P}(\mathrm{A}_\mathrm{i} = 1) = \frac{a}{n}$$

where, n - total length and a - number of 1's.

For example $\mathrm{A} = \frac{6}{8}$, represented by sequence $(1,0,1,1,1,0,1,1)$.

- Unipolar notation: $\mathrm{A} = \frac{a}{n}$, with range [0, 1]
- Bipolar Notation: $\mathrm{A} = \frac{2a - n}{n}$, with range [-1, 1]
- Extended SC

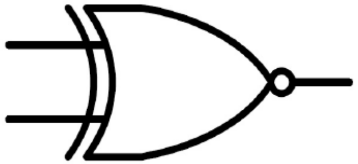
The complex binary arithmetic computations reduce to simple SC operations.

Stochastic Computation

Bipolar SC Multiplier using simple X-NOR operation :

S1: $\frac{4}{8}$: 01101010

S2: $\frac{6}{8}$: 10111011

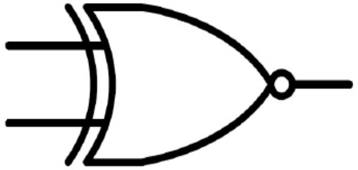


S3: $\frac{3}{8}$: 00101010

Correct value

S1: $\frac{4}{8}$: 01011100

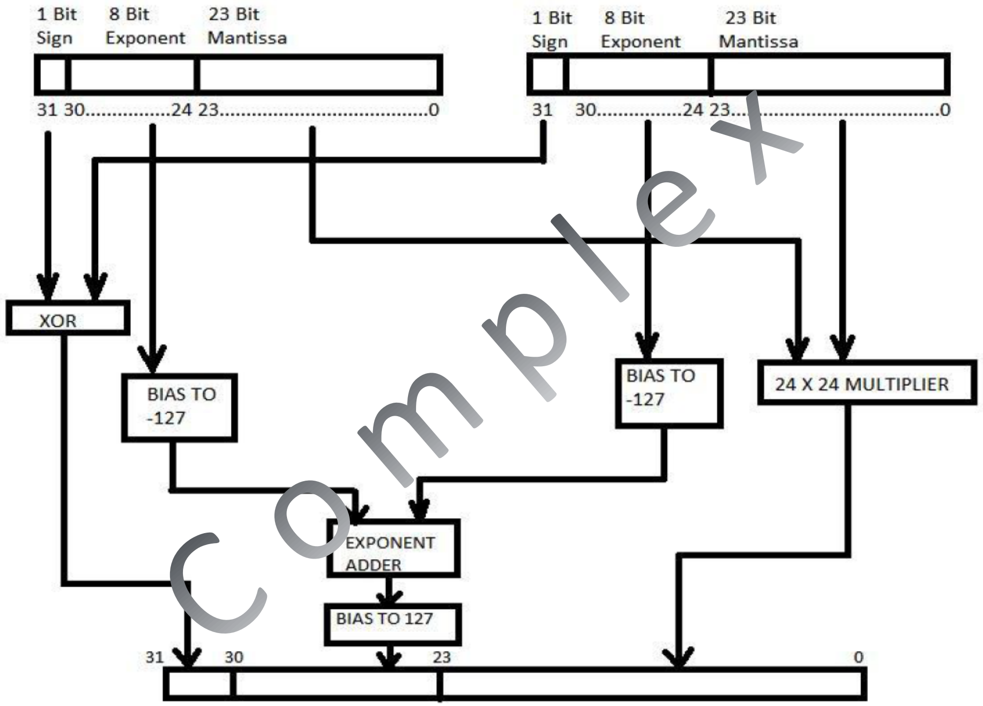
S2: $\frac{6}{8}$: 11101011



S3: $\frac{2}{8}$: 01001000

Incorrect but approximate value

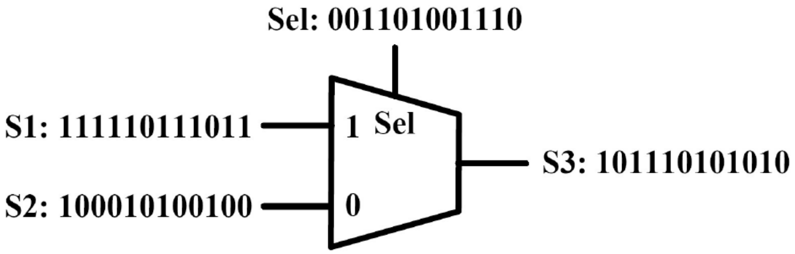
Binary floating point multiplier:



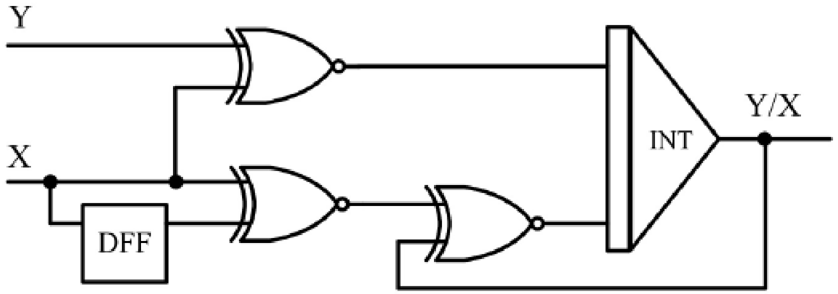
Single precision floating point multiplier [2].

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018. [2] - Sharma, Bhavesh & Mitra, Aritra & Bakshi, Amit & didwania, Nilesh & mishra, ruchika. (2015). Comparison Of Single Precision Floating Point Multiplier Using Different Multiplier Algorithms. Int. Journm. of Elec., Elect. & Data Comm. 2320-2084. 3. 106-109.

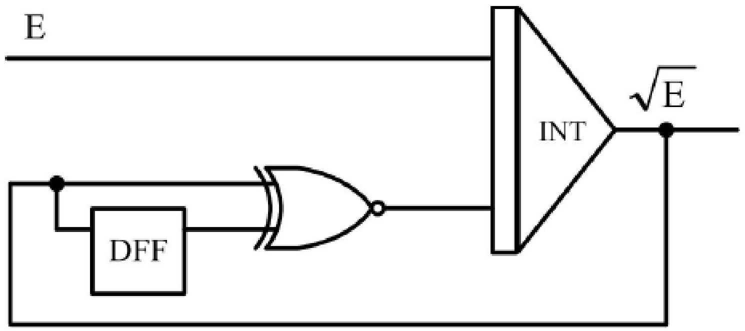
Stochastic Computation



SC Adder circuit using a MUX. [1]



SC divider circuit. [1]



SC square root circuit. [1]

DFF – D Flip flop
INT - Integrator

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

Stochastic Computation

Advantages of using SC:

- Reduced chip area.
- Implementation using conventional CMOS logic.

Limitations of SC:

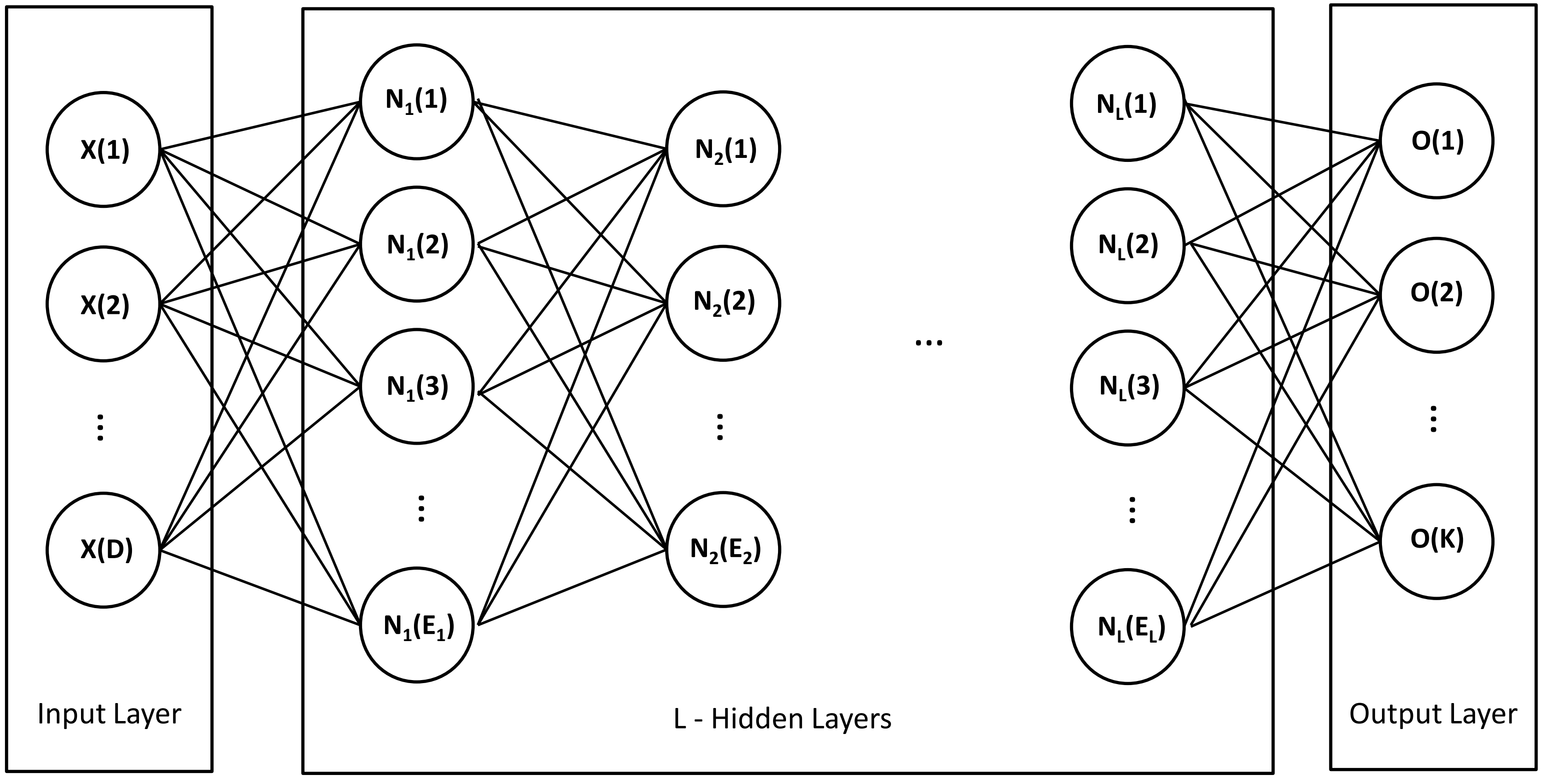
- **Correlation -** similarity in numbers.
- High latency for large sequences.
- Stochastic nature reduces accuracy.

Probabilistic nature of the deep neural network makes them suitable to be implemented in SC.

Deep Belief Networks

DBN has an input layer, multiple hidden layers and an output layer.

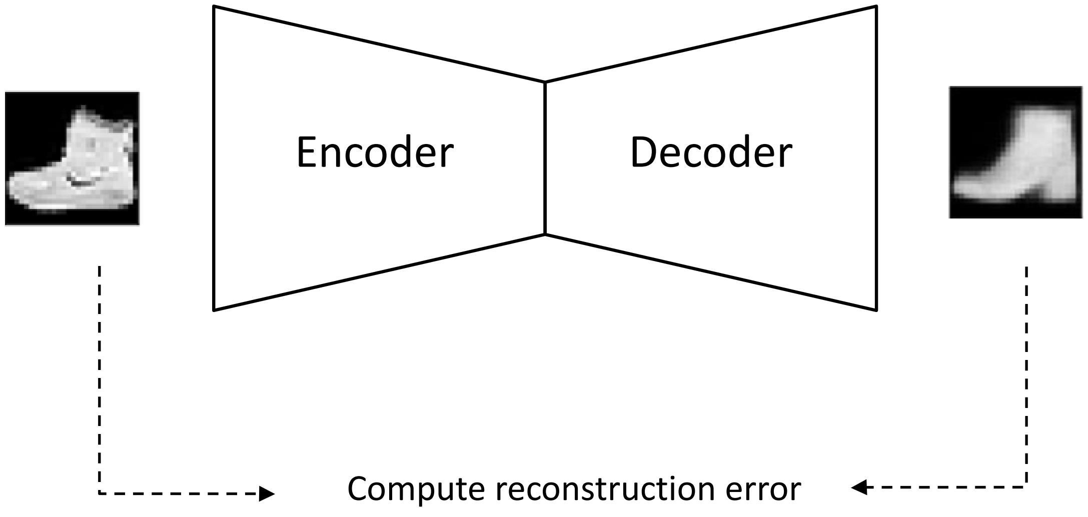
- A type of DNN
- Fully connected
- Not exactly feed forward network



Deep Belief Network

DBN Training:

- Unsupervised learning phase
- Supervised learning phase



Autoencoder unsupervised learning

Source: Wikimedia https://bit.ly/2OI4n2E

DBN Unsupervised Training

Fast – Greedy Algorithm:

Encoder-decoder pair trained as a Restricted Boltzmann Machine (RBM).

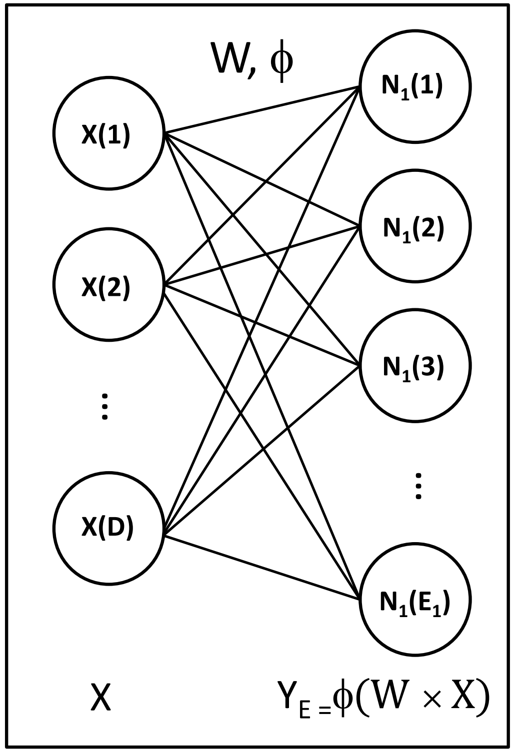
RBM training step:

- Encode: X --> Y_E
- Decode: Y_D <-- Y_E
- Encode: Y_D --> Y_{E2}
- Compute and minimise reconstruction error.

where, X - input vector encoded to Y_E. Y_D - decoder output, encoded to Y_{E2}.

'ϕ' - activation function.

W - weight matrix



DBN Unsupervised Training

1. Encode: X --> Y_E :

$$Y_E = \phi(\, W \times X \,) = \phi(\, \textstyle\sum_{i=1}^{D} x_i w_{ij})$$ *[1]*

2. Decode: Y_D <-- Y_E :

$$Y_D = \phi(\, W^T \times Y_E \,)$$ *[1]*

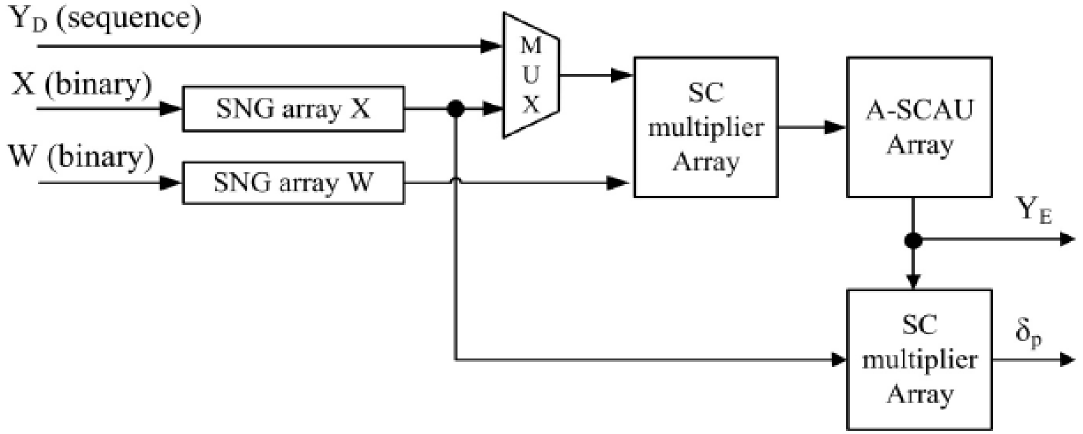
3. Encode: Y_D --> Y_{E2} :

$$Y_{E2} = \phi(\, W \times Y_D \,)$$ *[1]*

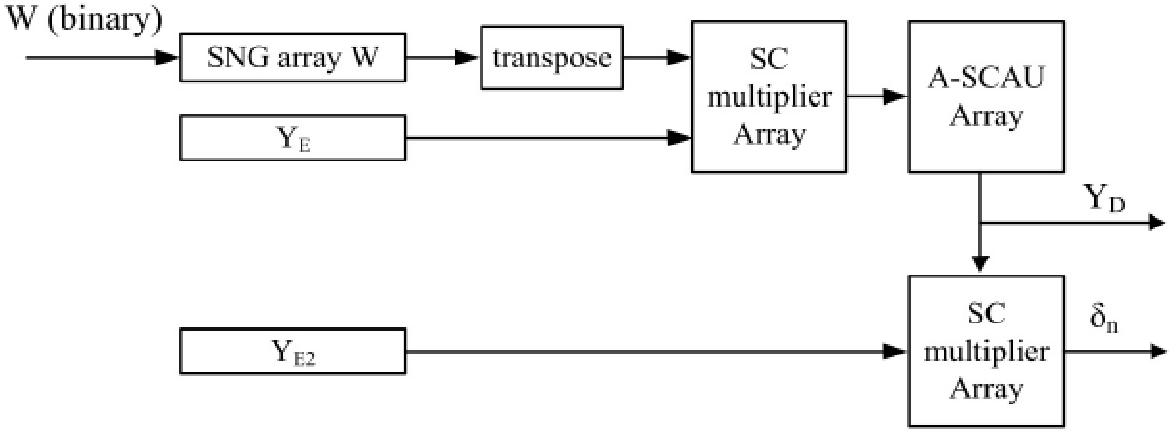
SNG – Stochastic number generator

MUX – Multiplexer

A-SCAU – Approximate SC Activation Unit



Encoder for fast greedy algorithm. [1]



Decoder for fast greedy algorithm. [1]

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

DBN Unsupervised Training

- RBM training simplifies to **One time Gibb's sampling** (OTGS):

$$W(t) = \mu W(t - 1) + (\delta_P - \delta_N) \qquad [1]$$

$$\delta_P = X^T \times Y_E \qquad [1]$$

$$\delta_N = Y_D{}^T \times Y_{E2} \qquad [1]$$

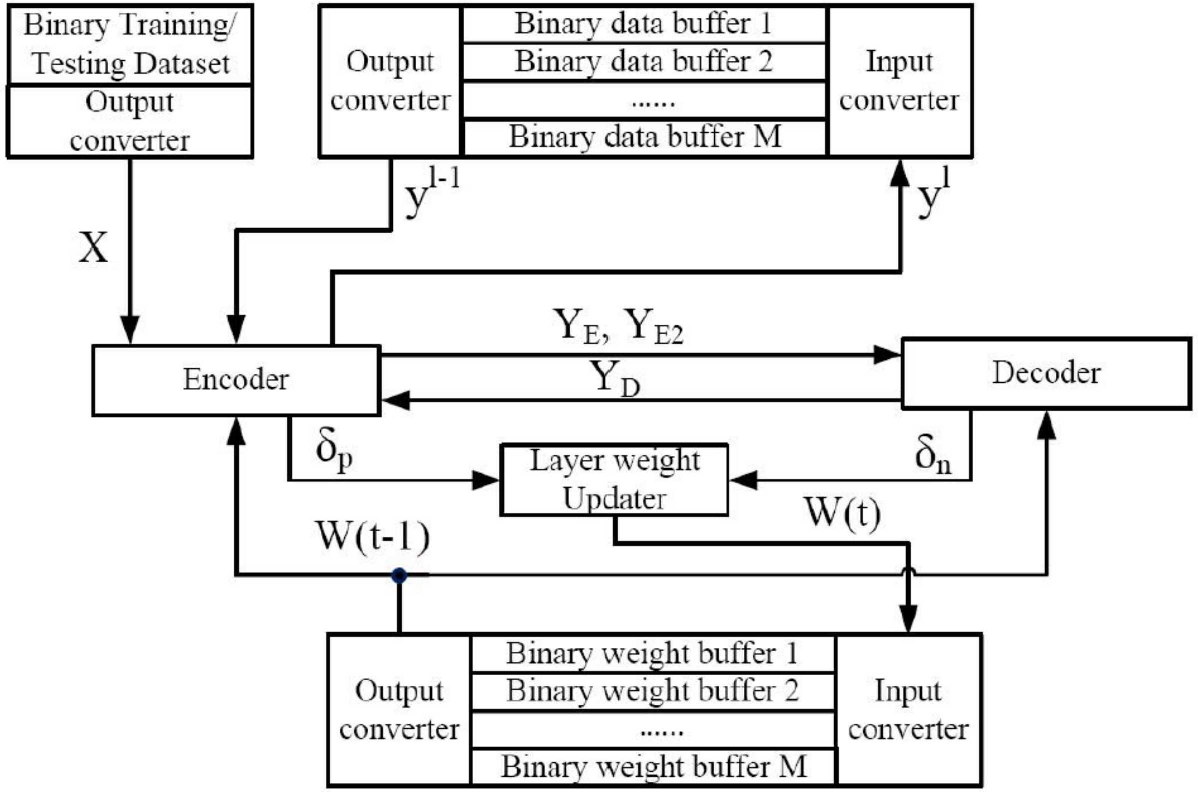
t' time step, μ - learning rate, δ_P – **positive difference**, δ_N - **negative difference**

- Iterate over all samples in multiple epochs.

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

SC-DBN Fast Greedy Algorithm

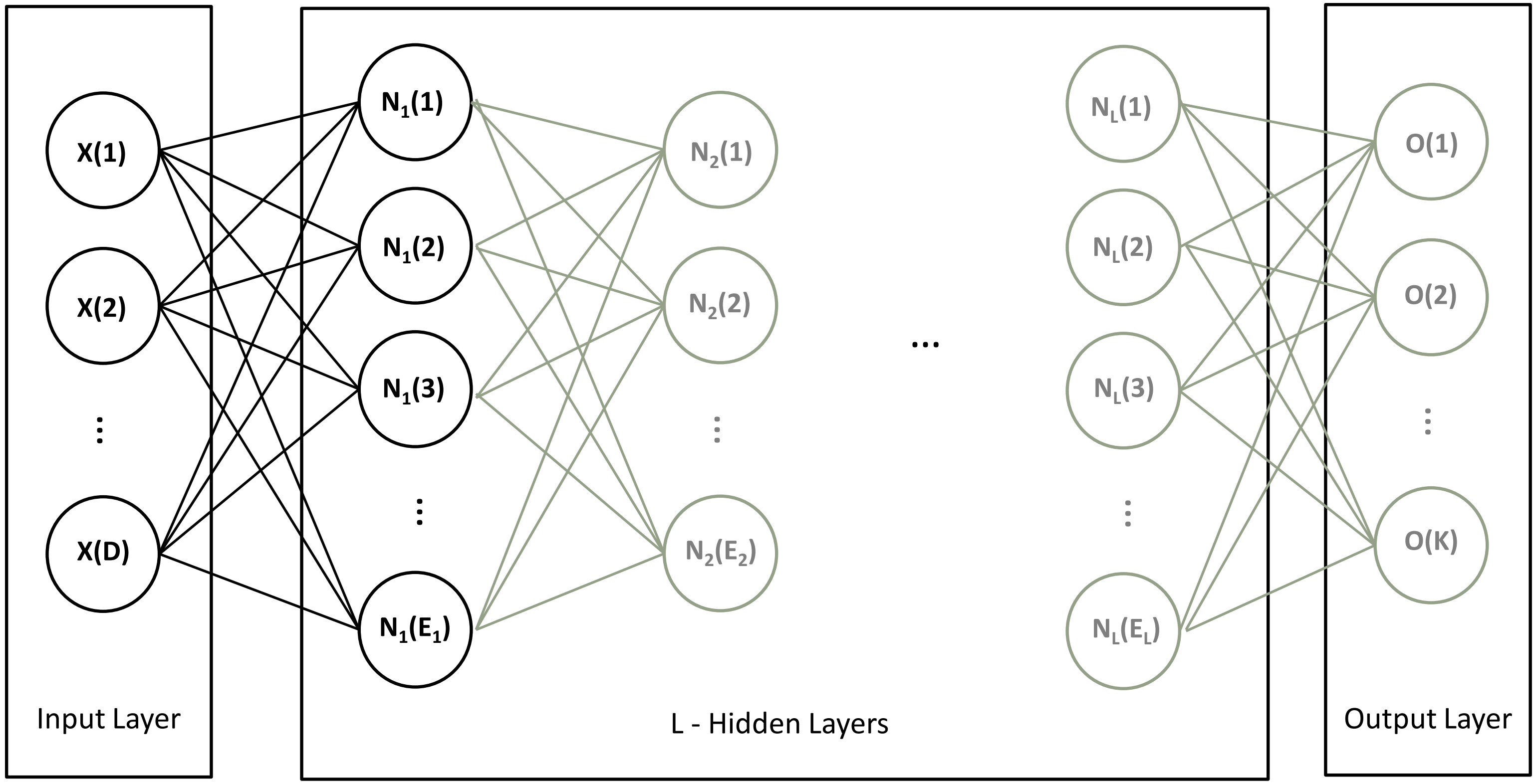
- A single hardware blocked shared by all neurons in the same layer.
- Once an RBM pair is trained, stack the next layer to form a new RBM.



SC-DBN block for fast greedy algorithm. [1]

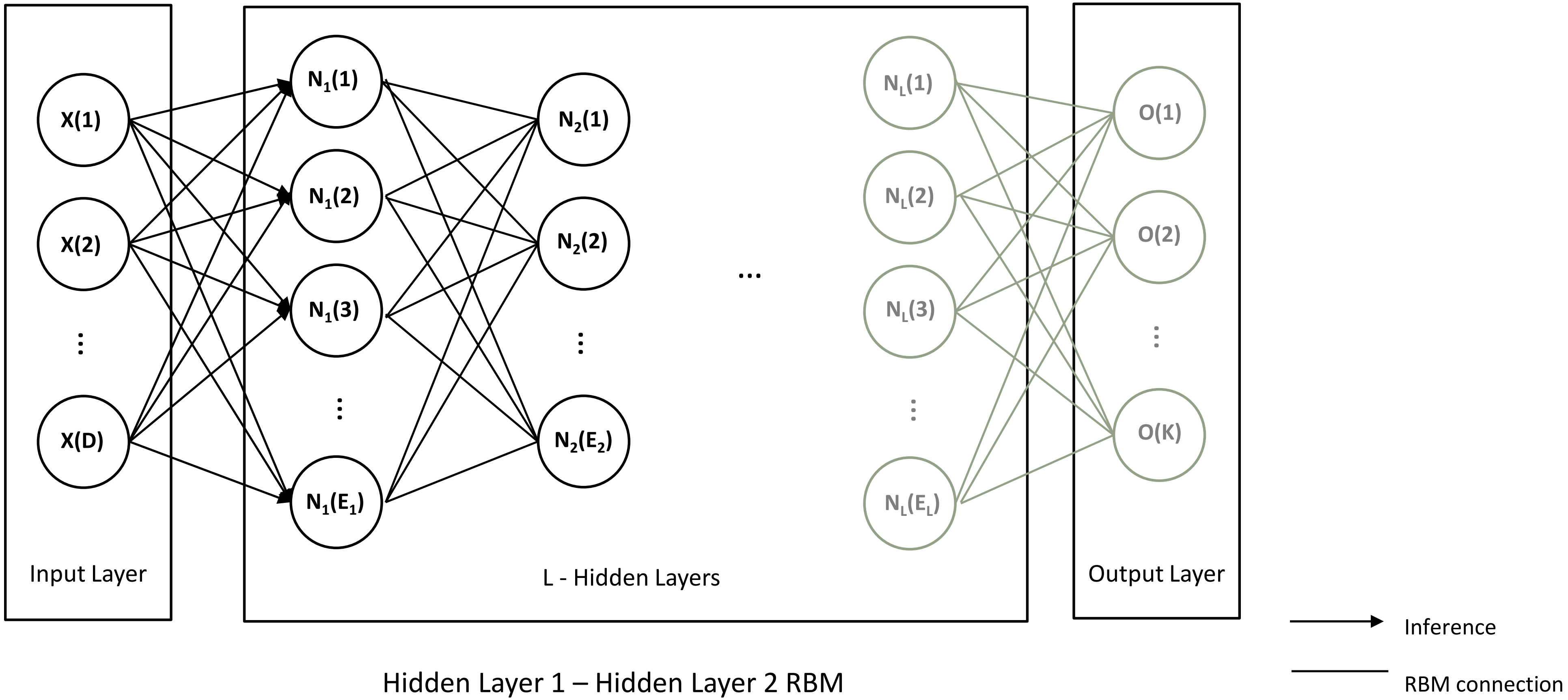
[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

DBN Unsupervised Training



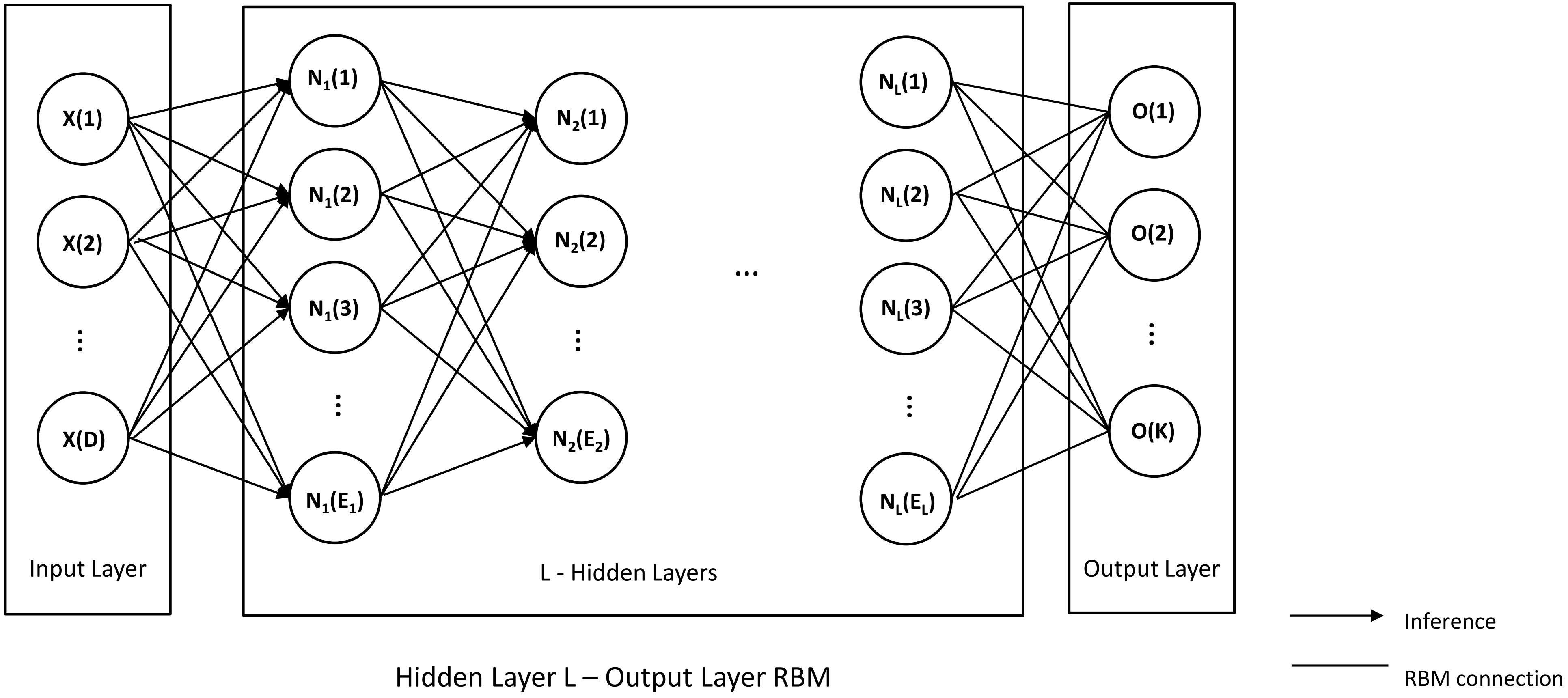
Input Layer – Hidden Layer 1 RBM

DBN Unsupervised Training



Hidden Layer 1 – Hidden Layer 2 RBM

DBN Unsupervised Training

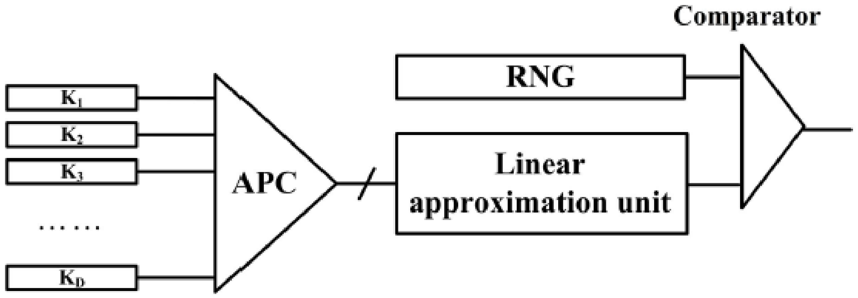


Hidden Layer L – Output Layer RBM

A-SCAU

SC-DBN uses one of the following activation functions ϕ:

- Sigmoid function: $\phi(x) = ½(\tanh(\frac{x}{2}) + 1)$ *[1]*
- Rectified Linear Unit (ReLU): $\phi(x) = \min(1, \max(0, x))$ *[1]*
- Pure line function: $\phi(x) = \min(1, \max(-1, x))$ *[1]*



A-SCAU block diagram. [1]

- The APC performs the summation and is **invariant to sequence correlation**.
- Linear Approximation Unit (LAU) implements approximate activation functions as per equation:

$$\phi(x) = \min(1, \max(p, \frac{x}{r} + s))$$

p, r and s are parameters of the LAU.

$$\phi(X \times W) = \phi(\sum_{i=1}^{D} x_i w_{ij})$$

Computation at each neuron. [1]

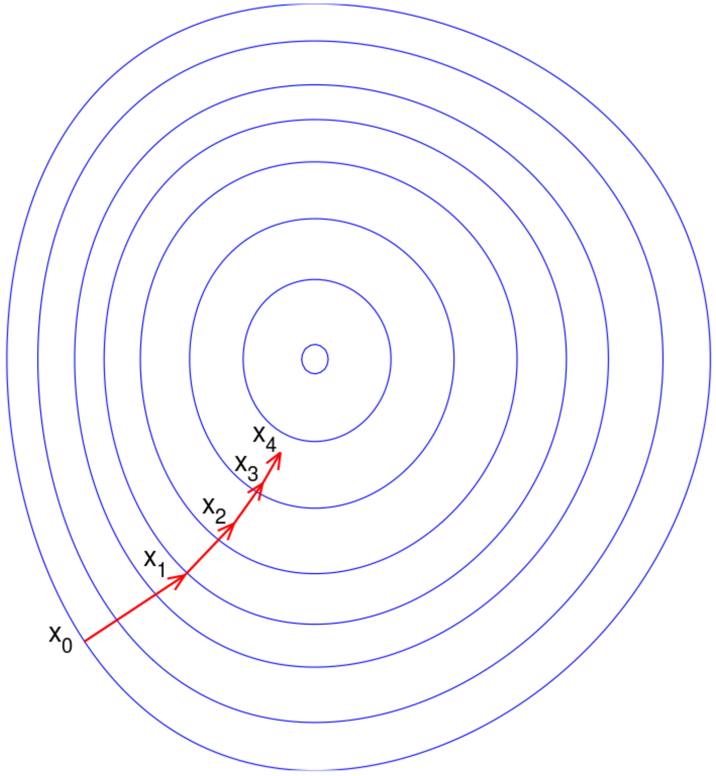
- Overall **only 3 RNG's** are used and single unsupervised block is shared.

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

DBN Supervised Fine-Tuning

Gradient and back propagation:

- To learn the network parameters θ.
- Gradient w.r.t all the network parameters and step.
- Gradient of activation functions are simple.



Source: Wikimedia common https://bit.ly/38UaWZ8

ADAM

- Iterative algorithm for fast convergence and prevents oscillations.
- Here, g_t is the gradient. m_t *and* v_t are first and second order moments.
- β_1 and β_2 are exponential decay rates. α is the learning rate and ε is to prevent division by zero.
- All equations can be implemented in SC.

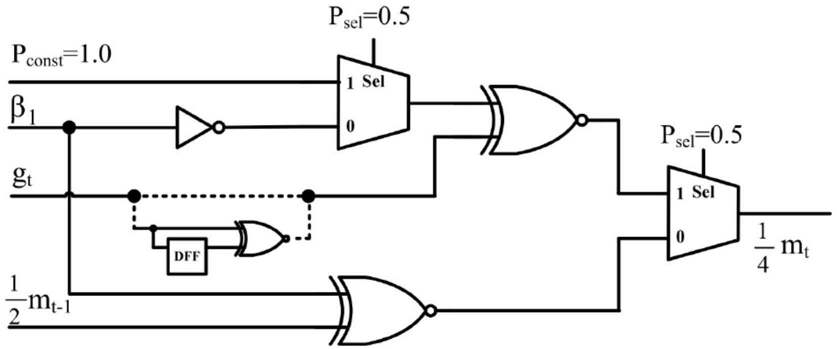
$$\begin{cases} t = t + 1, \\ g_t = \nabla_\theta f_t(\theta_{t-1}), \\ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \\ v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2, \\ \hat{m}_t = m_t / (1 - \beta_1^t), \\ \hat{v}_t = v_t / (1 - \beta_2^t), \\ \theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon). \end{cases}$$

ADAM pseudo code [1]

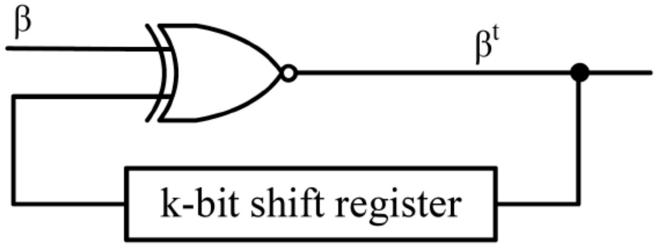
The typical values are α = 0:001, β_1 = 0:9, β_2 = 0:999 and $\varepsilon = 10^{-8}$

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

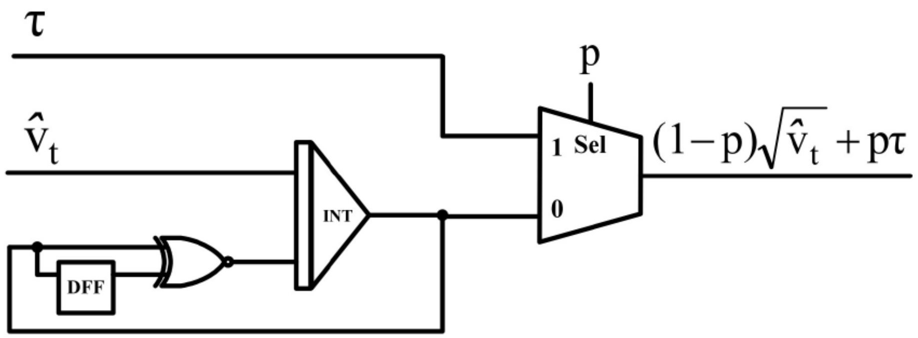
ADAM



ADAM the first (m_t) and the second (v_t) order moment calculator [1]



Circuit computing $\beta_1^{\,t}$ *and* $\beta_2^{\,t}$ [1]



Circuit computes the last step of ADAM - $\alpha.\widehat{m}_t / \sqrt{v_t^2} + \varepsilon$ [1].
Here, $\alpha = (1\text{-p})$ and $\varepsilon = \text{p}\tau$

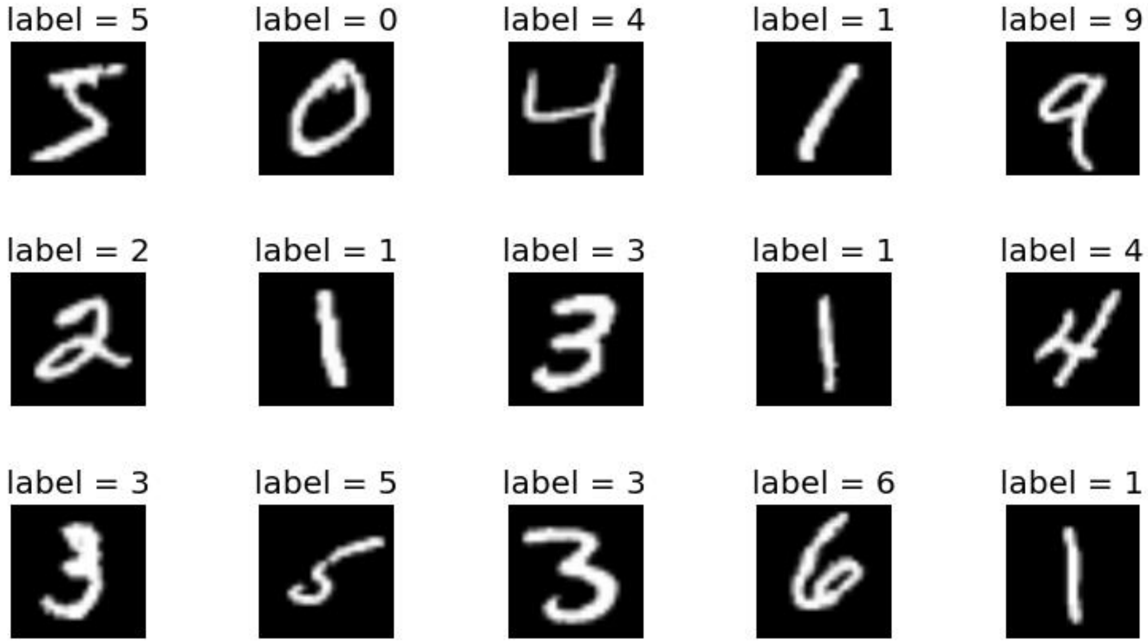
[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

Evaluation

The SC-DBN to classify MNIST dataset consisting of 28x28 pixel images of handwritten digits from 0 to 9.

SC-DBN input layer of 784 neurons, 2 hidden layers of 100 and 200 neurons and an output layer of 10 neurons.

- A 256 length, 16x parallelization and full length of 4096.



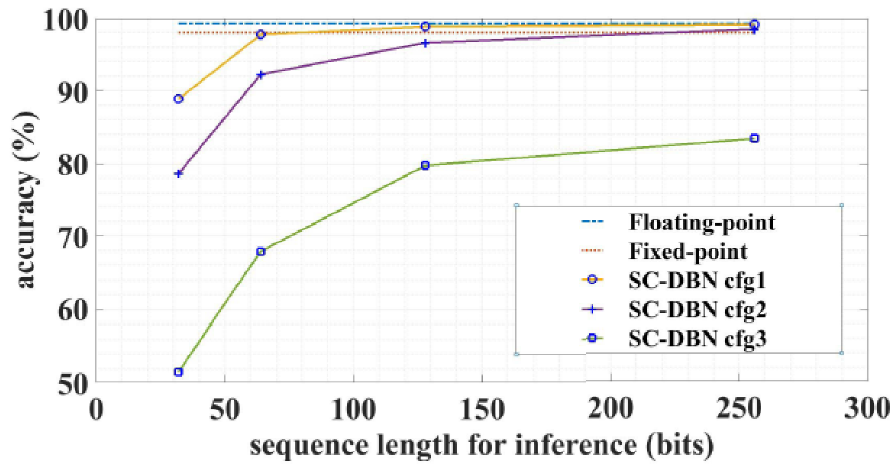
MNIST dataset

Source: towardsdatascience https://bit.ly/2BdXDX0

Evaluation

Network	sequence length (bit)	accuracy (%)
SC-DBN (16× parallelization)	32	89.90
	64	97.78
	128	**98.90**
	256	**99.15**
8-bit fixed point	–	98.10
32-bit floating-point	–	99.27
Integral stochastic NN [19]	64	97.73
Hybrid SC-binary NN [30]	128	99.01
SC DNN [18]	1024	97.59
FPGA-RBM [29]	1024	94.28
FPGA-DBN [17]	4096	94.10

Accuracy for various pretrained implementations of DNNs [1]



Accuracy and sequence length relationship for various DNNs [1]

Evaluation

Performance metrics for inference models:

ASIC implementation with ST 28nm technology library and Synopsys Design Suite.

	SC-DBN	8-bit fixed-point circuit	32-bit floating-point circuit (pipelined, non-pipelined)
Area (μm^2)	23345	86875	(437767, 357548)
Power (mW)	1.12	4.01	(24.86, 18.32)
Frequency (MHz)	134.7	167.3	(159.7, 90.2)
Cycle (/sample)	128/256	296	(412, 4)
Latency (μs/sample)	0.94/1.90	1.77	(2.580, $\boxed{0.044}$)
Energy (nJ/sample)	1.05/2.12	7.10	(64.14, $\boxed{0.81}$)

21 times smaller

SC-DBN has 1.3 times the energy

Inference Hardware performance SC-DBN vs 8 bit fixed and 32 bit FP implementations [1]

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

Evaluation

Performance metrics with online learning:

- Reduced overall chip area,
- Reduced latency per epoch but over all latency higher compared to 8-bit implementation.
- Large reduction in power consumption.

	SC-DBN	**8-bit fixed-point circuit**	**32-bit floating-point circuit**
Back-propagation circuit area (μm^2)	33150	116413	656651
Learning control unit area (μm^2)	3525	1785	1829
Total area (μm^2)	60019	205072	1096247
Latency per epoch (μs)	7.60	$\boxed{6.92}$	9.43
Total latency (200 epochs) (ms)	1.52	$\boxed{1.38}$	1.88
Energy per sample (μJ)	4.37	13.11	117.40

Online Learning DBN hardware performance [1]

ADAM metrics:

- Fast convergence with 31 epochs.
- Overall latency reduced.

ADAM area (μm^2)	27181
Total area (μm^2)	87200
Total latency (31 epochs) (μs)	529.1
Energy per sample (μJ)	1.10

Performance changes with ADAM[1]

[1] - Yidong Liu , Yanzhi Wang, Fabrizio Lombardi and Jie Han. An energy-efficient online-learning stochastic computational deep belief network. IEEE Journal on emerging and selected topics in circuits and systems, 8(3):454{465, 2018.

Conclusion

- SC simplifies arithmetic implementations.
- DBN has its benefits compared to other models.
- Shared hardware blocks and hence reduced chip area.
- Unsupervised learning with fast greedy algorithm results in simple OTGS computation.
- A dynamic reconfigurable A-SCAU free of SC correlation.
- ADAM results in fast convergence and reduces long latencies.
- Results in an energy efficient, reconfigurable DBN with online learning capability and good accuracy.

The End

Questions