Non-Turing machines: Stochastic and probabilistic learning circuits

Sandip Tiwari, stiwari@iitk.ac.in, st222@cornell.edu

Exploiting randomness and probabilities as low energy tools for non-Turing usage.

1. Large and small: The problems of scales in semiconductor electronics
2. Non-Turing machines: Stochastic and probabilistic learning circuits
3. Physics-guided AI/ML: Why, how and usage

4. Cultures: Science, engineering, interdisciplinarity and the fallacy of Ockham's razor
5. Semiconductors: Lessons from the past and what it says for semiconductor manufacturing

In the last talk:
Deterministic computing with large number of irreversible Boolean transformations in the midst of fluctuations/noise ends up costing many 1000s of $k_B T$ in energy per bit.

This can not change. It is fundamental to error minimization in a thermal bath.

I ended by relaxing how accurately one wished to compute.

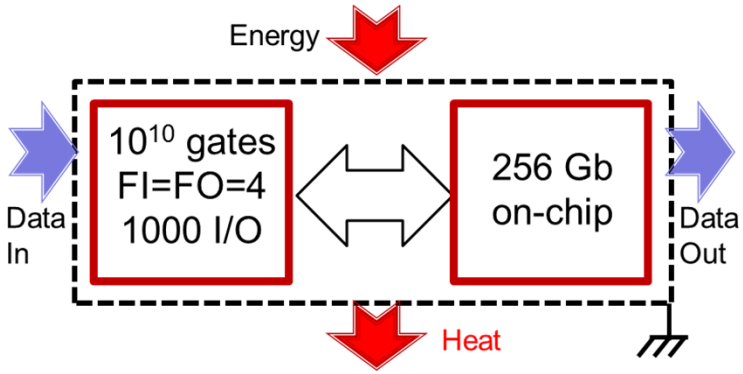
Useful in compression, AI as currently practiced, …., where good enough works.

Today: Randomness as a resource.

https://www.iitk.ac.in/scdt/Sandip_Tiwari_Kanpur_Lectures.html

Energy

From the last talk: Deterministic computing implies $\gtrsim 1000 k_B T$ of energy per bit operation to overpower errors and because energy not recycled.



n circuits, *m* terminals, *t* terminals

Proc. capacity: $N = nm + t$

Phase space: $\Omega = N^N$ logic

$\Omega = M^2$ memory

Information and negentropy: $\mathfrak{I} \approx 1.5 \times 10^{12}$ b

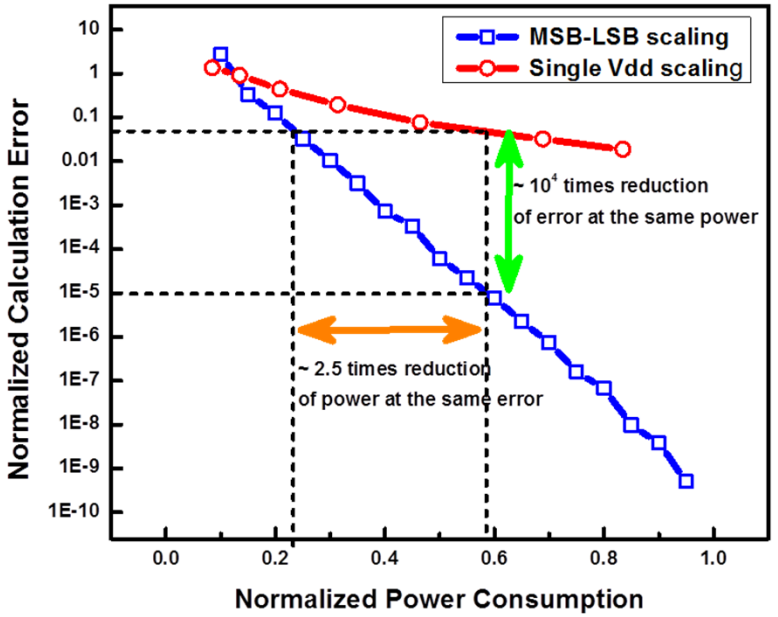
$\mathfrak{N} \approx 2.1 \times 10^{-11}\ J/K$ logic

Memory's contribution smaller

Work capability: $\approx 7.3\ nW.s$

Many orders of magnitude lower!

An adder

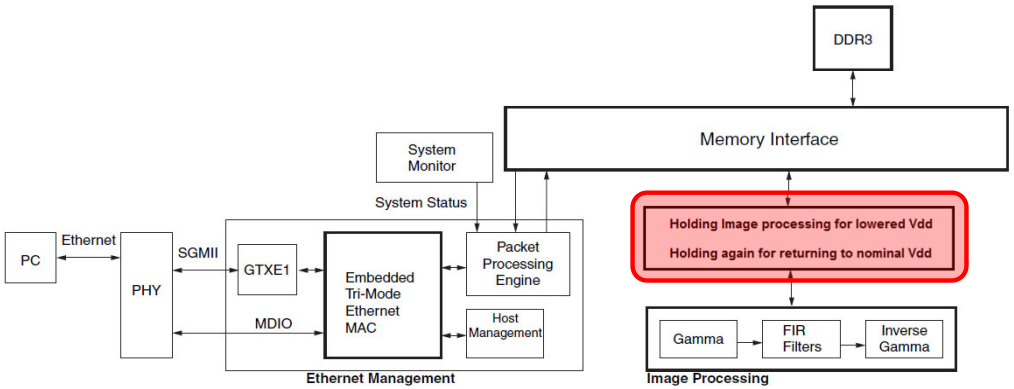


Similar conclusions with use of stochastic approaches to Bayesian networks that can be programmed into deterministic hardware.

Tiwari, Proc. of IEEE Aug. (2015)

Kim & Tiwari (2010)

Inexact DSP with Lower Voltages



x1.0 Energy

x0.66 Energy

x0.52 Energy

Example: Inexact Addition

Floating point sum with *n* *a,b* mantissae

$$A(\equiv a_{n-1}...a_1a_0)+B(\equiv b_{n-1}...b_1b_0) = S(\equiv s_{n-1}...s_1s_0)$$

Consider a sum *S'* within a probability distribution $\mathfrak{p}_\sigma(S^{'} - S)$ centered at *S* of width σ
all expressed normalized – i.e. in units of least precision (*ulp*'s)

If $\sigma \gg 1,$ bits less significant than σ can be ignored and the circuit truncated (i.e. those parts shut off) to make the LSB $\sim \sigma$ *Eliminates wasted power*

$\sigma \sim 1$ is a useful guide mark for error

Consider $FWHM \sim 1$ for

$$\exp\left[-\pi(S-S^{'})^2\right]$$ Gauss

$$\frac{2/\pi}{1+4(S-S^{'})^2}$$ Cauchy-Lorentz

For tails, $|S - S^{'}| \gg 1,$ discreteness can be ignored and what is important is exponential or polynomial tail

Addition

Arithmetic circuit composed of elements: $\alpha = 1, 2, ...M$

For carry-select adders, elements are half adders generating partial sums, and
Kill/Propagate/Generate (KPG) signals.
KPG combine blocks forming carry-propagation tree.
Multiplexers at the output select the partial sums based on carry-propagation tree

For each element α

U_α Energy dissipated per computation using design/voltage scaling

ϵ_α Probability of error

w_α Weight for each element that quantifies mean magnitude of numerical error in the answer caused by incorrectness in the element while all the others are correct

e.g., mux corresponding to bit k of output has the weight 2^k *ulp*s

Addition

Assume errors in each element uncorrelated

When an element $\alpha \in W \subset 1, 2, ..., M$ is wrong,

the error in answer is: $\langle (S - S^{'})^2 \rangle_W = \sum_{\alpha \in W} w_\alpha^2$

with W as the set of erroneous elements with probability: $\left[\prod_{\alpha \in W} \epsilon_\alpha\right] \left[\prod_{\beta \notin W} (1 - \epsilon_\beta)\right]$

The error in the answer would be within constraints of prob distribution $\mathfrak{p}_\sigma\left((S^{'} - S)^2\right)$ so long as

$$\left[\prod_{\alpha \in W} \epsilon_\alpha\right] \left[\prod_{\beta \notin W} (1 - \epsilon_\beta)\right] \leq \frac{1}{g_W} \mathfrak{p}_\sigma \left(\sum_{\alpha \in W} w_\alpha^2\right) \forall W \subset 1, 2, ..., M$$

Degeneracy – different subsets W that lead to same mean squared error $\sum_{\alpha \in W} w_\alpha^2$

Addition

For Gaussian distribution, the constraint on error is satisfied if

$$\epsilon_\alpha \leq \frac{1}{g_W} \exp\left[-\pi w_\alpha^2\right] \quad \forall \alpha$$

For Lorentzian distribution, the constraint on error is satisfied if

$$\epsilon_\alpha \leq \frac{2/\pi}{g_{w_\alpha}(1+4w_\alpha^2)} \quad \forall \alpha$$

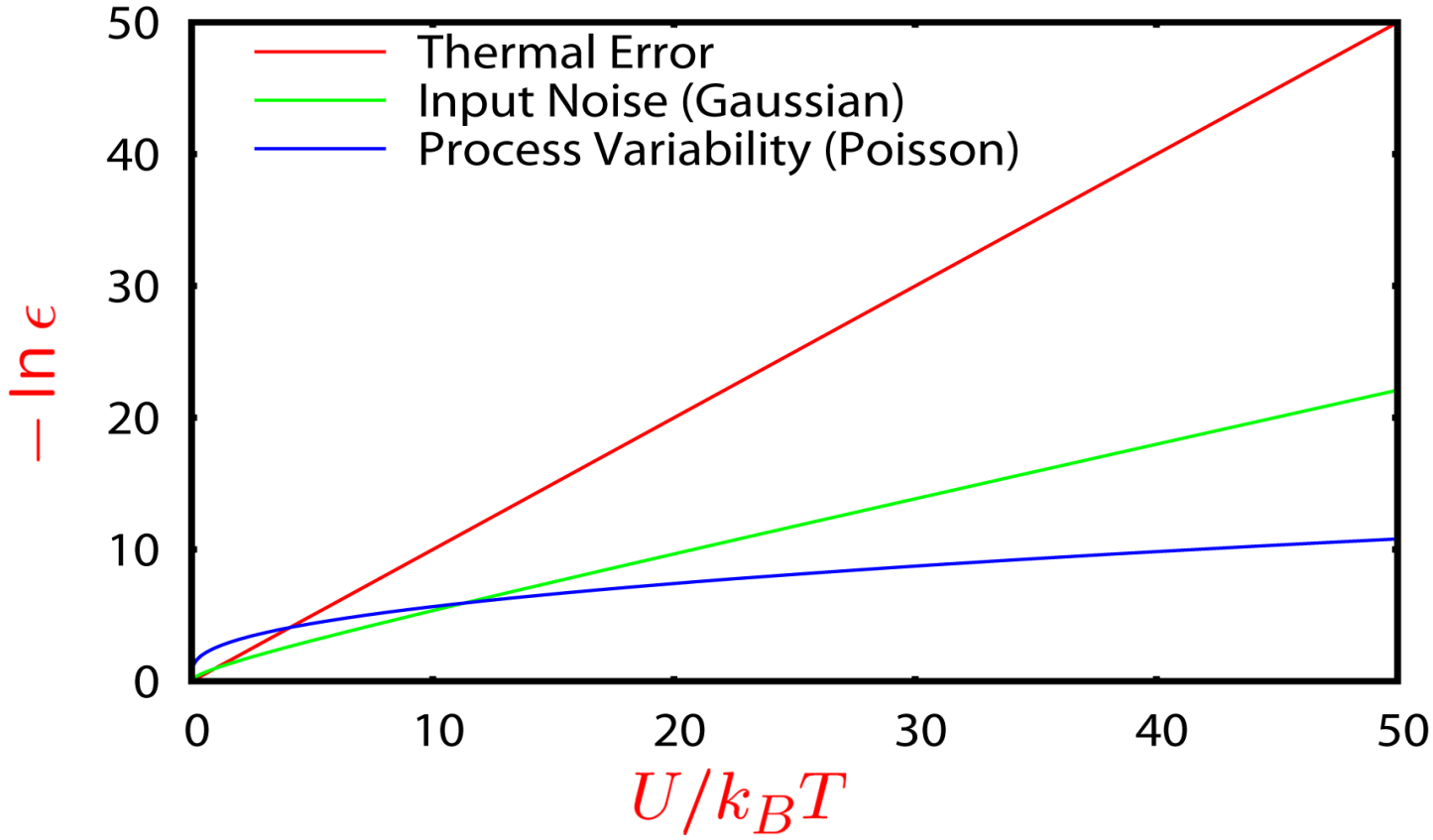
For simple CMOS: $U \propto -\ln \epsilon$

The proportionality constant depends on source of error,

for threshold voltage variations: $\sim CV_{DD}\sigma_{V_T}$

for thermal noise: $\sim k_B T$

Power-Error Trade-Off



By approximate transistor count, half adders are twice as expensive as KPG, Mux and we can estimate energy given the error rate – energy relationship

Adders

Simple case of *carry tree linear chain*: an inefficient ripple-carry adder in carry-select:

One adder, one KPG combine, 1 mux per bit (excluding MSB and LSB end)

Elements at bit level *k* each have a weight 2^k => degeneracy, $g_{w_\alpha} = 3$

Energy dissipated per computation for entire *n*-bit adder :

Gaussian $U \propto -4\sum_{k=0}^{n-1} \ln\left(\frac{1}{3}\exp(-2^{2k}\pi)\right) = 4n\ln 3 + \frac{4\pi}{3}\left(2^{2n}-1\right)$

Lorentzian $U \propto -4\sum_{k=0}^{n-1} \ln\left\{\frac{2/\pi}{3[1+2^2(k+1)]}\right\} \approx 4n\ln\left(\frac{3\pi}{2}\right) + 4n(n+1)\ln 2$

$U \propto 4n\ln\left(\frac{1}{\epsilon}\right)$ (flat error everywhere, i.e., exact arithmetic)

The flat error, exact limit (an estimate of lowest ϵ) can be used to estimate where approximate adders can be more efficient

More Useful Cases in Arithmetic Operations

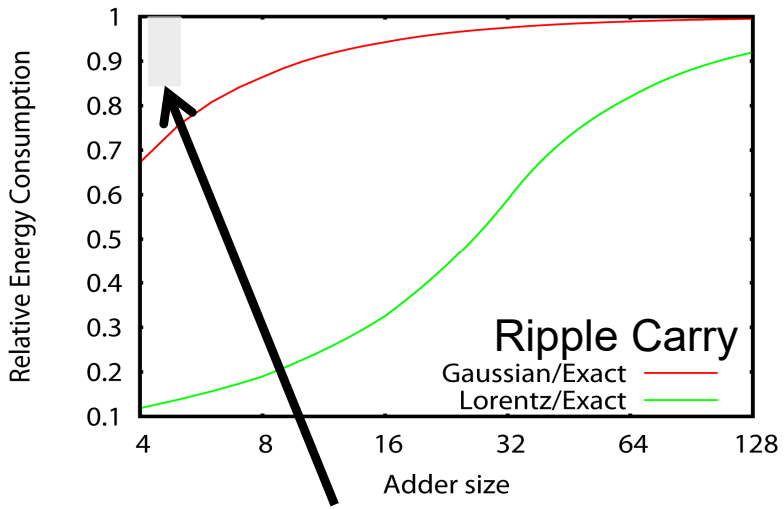
Constraint: no block in adder has an error $< \epsilon_0$,

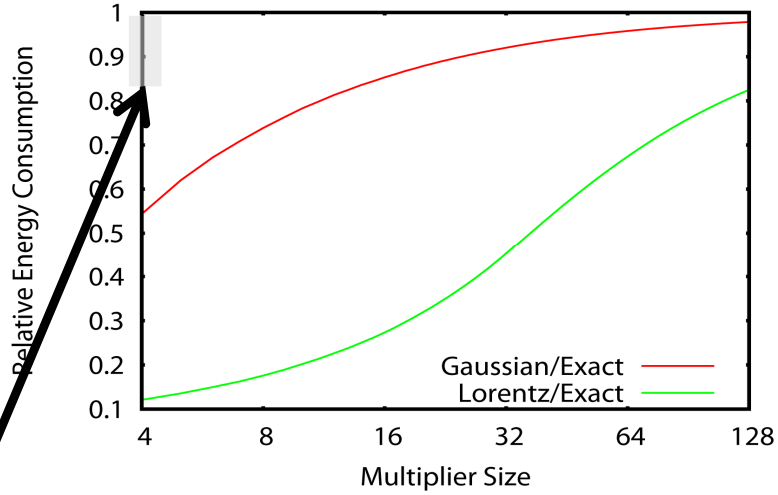
i.e., together with following the distribution $\mathfrak{p}_\sigma(S' - S)$, the answer is random with a miniscule probability $n\epsilon_0$

A more efficient adder: Kogge-Stone

KPG-combine block of weight 2^k is $\log_2 k$

Modified Booth Multiplier





Region of maximum power savings

These were all 25% to 50% energy improvements.

1000s of $k_B T$ per bit is a hard limit.

The history of the information mechanics engine is of an interplay between the algorithm and the platform.

Sometimes it is the algorithms (soft) that lead.

Sometimes it is the platform (hard/physical) that leads.

Both adapt to each other.

Non-determinism (probabilism)---tied to complexity---and their algorithms can extract even higher energy penalty when using the platform that evolved from determinism. As in deep learning implementations of these days.

My view is that the question/issue is much much more subtle, and the wall-plug efficiency argument very simplistic.

Priors (so of learning)
Use of that learning in
Bayesian methods
Neural networks methods (next talk)

Turing

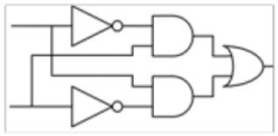
Turing machines are hypothetical abstract devices that yield finite descriptions of algorithms that can handle arbitrarily long inputs.

Specification

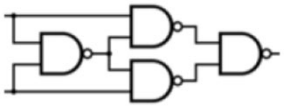
$\{0,1\}^n$	x	$f(x)$
	0000	1
	0001	0
	...	...
	1111	1

Unbounded

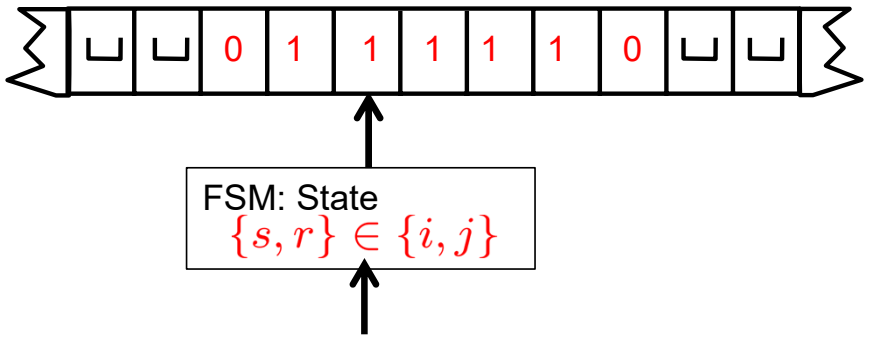
$\{0,1\}^*$	x	$F(x)$
	0	1
	1	0
	00	0
	01	1
	10	1
	...	...



```
t1    = AND(X[0],X[1])
notx0 = NOT(X[0])
t2    = AND(notx0,X[2])
Y[0]  = OR(t1,t2)
```



```
u = NAND(X[0],X[1])
v = NAND(X[0],u)
w = NAND(X[1],u)
Y[0] = NAND(v,w)
```



λ calculus driven by a program

Randomness/ What is not known/ A very specific behavior with hidden features

Photo: C. Bennett

A pottery kiln viewed via its own radiation

Via external radiation

When near thermodynamic equilibrium, external energy necessary to reveal information.

Because it “becomes” hidden

2nd law: Blackbody ⇔ Thermostatics

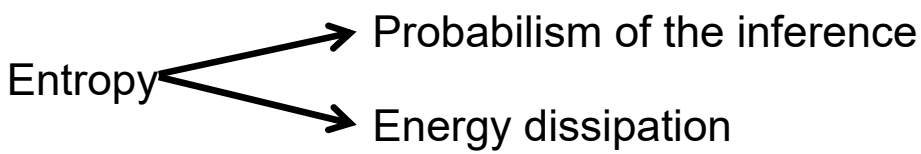
Nondeterminism -- probabilism -- entropy

Classical world and the quantum world are both a play of cause and chance with somewhat different rules.

Cause is a relationship
Chance is the unknown or random intermezzo between the knowns.

Chance also appears when we compress (approximate).

Entropy therefore appears in multitudes of ways in nondeterminism.



Randomness as a resource in computing

A sequence of bits is random if the shortest computer program for generating the sequence is at least as long as the sequence itself.

Kolmogorov complexity and G. Chaitin

Pseudorandom numbers are not random.
They just stretch whatever randomness exists in the seed.

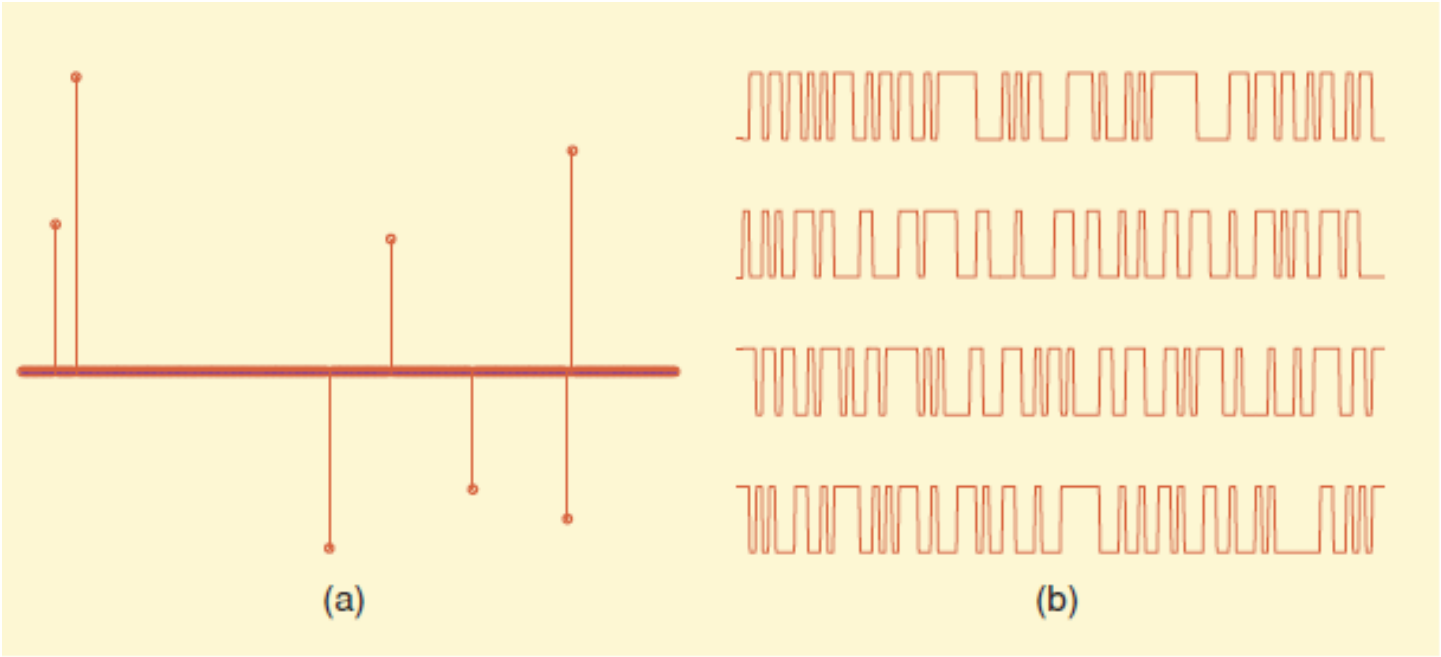
Psuedorandomness has been used in algorithms and in communications and in cryptography and safety as a resource since the dawn of computing.

Nature exploits randomness (fluctuations know how to).

Here, I will work through use of some examples of exploiting randomness:
(a) magnetization in superparamagnetic limit.
(b) thermal fluctuations of electrons wandering through a resistor or a semiconductor junction.

Synchronization by randomness that reveals information because of it.
Randomness as a way to mutual information optimization.
Probabilities by randomness and inferencing from it (Bayes).

Compressive sensing

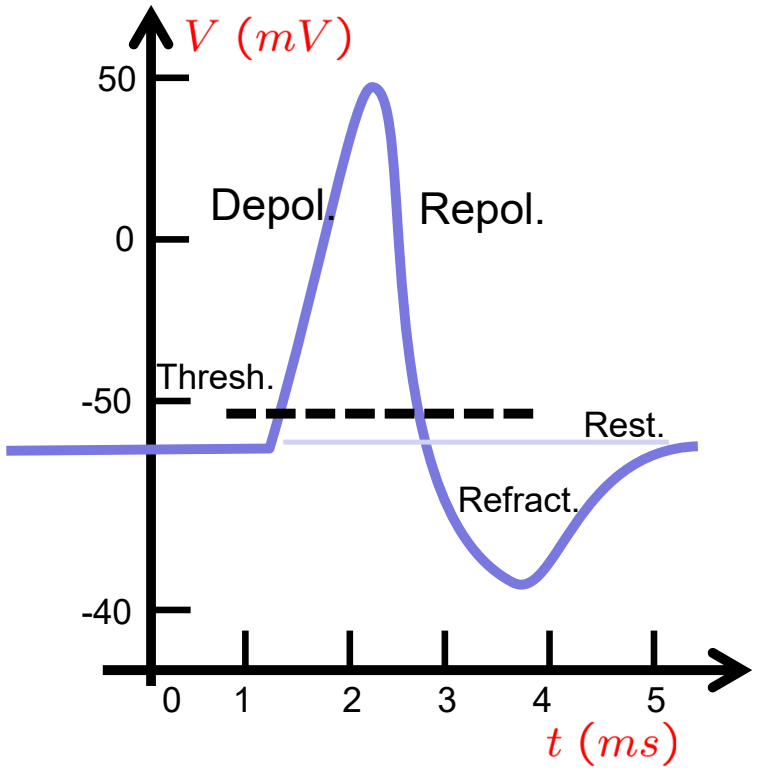


[FIG1] Sampling a sparse vector. (a) An example of a very sparse vector. If we sample this vector directly with no knowledge of which components are active, we will see nothing most of the time. (b) Examples of pseudorandom, incoherent test vectors ϕ_k. With each inner product of a test vector from (b), we pick up a little bit of information about (a).

Romberg, IEEE Signal Processing, March (2008)

Spike action potential

Spikes are based on leaky ion channels (~40 *mV*)



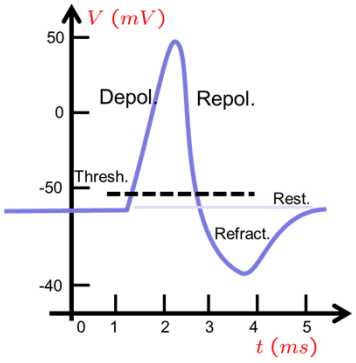
https://en.wikipedia.org/wiki/Action_potential

Spiking action potentials

Thresholding in low signal and noise

Capacitance-based signaling in presence of ion channel conductance and thermal noise.

Sustainable chemical concentrations: few to 100s *mMs*



Tubules

	$[K^+]$	$[Na^+]$
outside	5	140
inside	140	12

Reversal potential

$$V_{Rev} = \frac{RT}{zF} \ln \frac{[K^+]_{out}}{[K^+]_{in}} = \frac{8.314 \times 310}{96845} \ln \frac{5}{140} = -88.7\ mV.$$

Rest potential

$$V_{rest} = \frac{RT}{F} \ln \frac{\sum_i \pi_i [A^+]_{out} + \sum_j \pi_j [B^-]_{in}}{\sum_i \pi_i [A^+]_{in} + \sum_j \pi_j [B^-]_{out}}.$$

Ion sizes: $Na^+ < K^+ (0.138\ nm) < Ca^+$

Ion leakage: $K^+ (1 \times 10^{-6}) > Na^+ (2 \times 10^{-8}) > Cl^- (5 \times 10^{-10}\ cm/s)$

$V_{rest} = -78\ mV$

Energy in spike train

FitzHugh-Nagumo (FN) neuron for potentiation and flipping

$$\begin{aligned} \tau_{\uparrow} d_t u &= u(u-0.5)(1-u) - v \\ \tau_{\downarrow} d_t v &= u - v - \beta + \varepsilon \sin(\omega t) + \zeta_n, \end{aligned}$$

$u(t)$: action potential, $v(t)$: refractory variable, β : bias, ζ_n :noise ($\sim 30\ mV$)

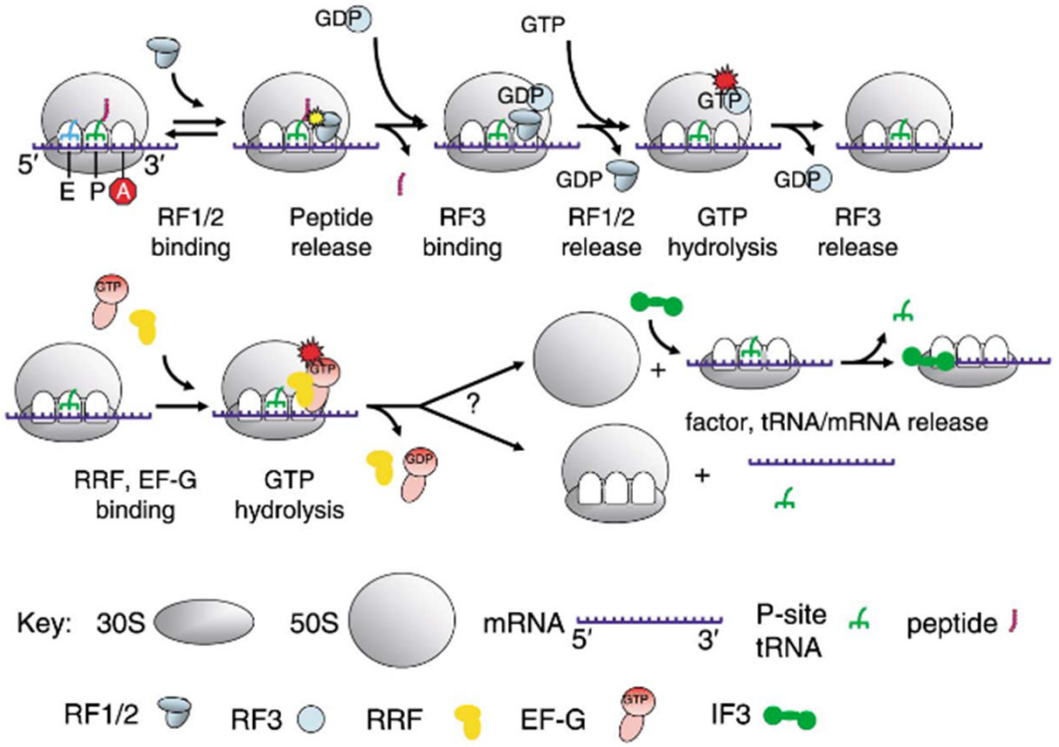
Fisher information for best estimate:

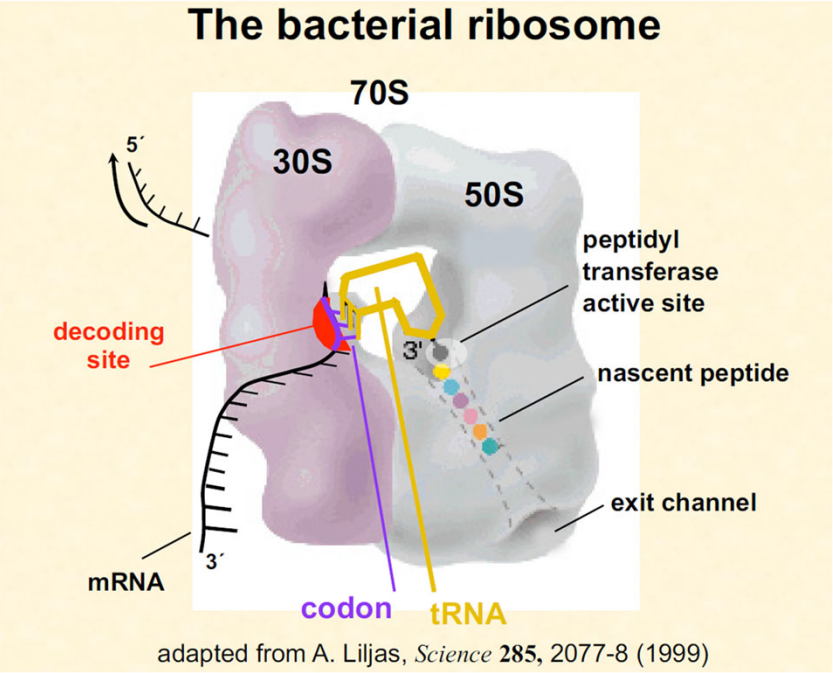
$V_{peak} = 40\ mV$, which is of the order of noise (noise helps!).

Spike energy: $\approx 0.5 \times 110\ mV \times 10^{-3}\ s \times 3\ pA \approx 165\ aJ \approx 40000 k_B T$

The neuron spiking is an energy-centric mutual information stressing process.

Ribosome is not a Turing machine





V. Ramakrishnan, Cell, Vol. 108, 557–572, February 22, 2002

V. Ramakrishnan, Nobel lecture (2009)

Perturbations as technology tools

Dithering, as in lock-in techniques, are a method to improving sensitivity and reducing noise in a bandwidth.

Heterodyning improves sensitivity in receivers.

These are techniques that lower power and energy in detection.

Compressed sensing uses linear projections onto random basis

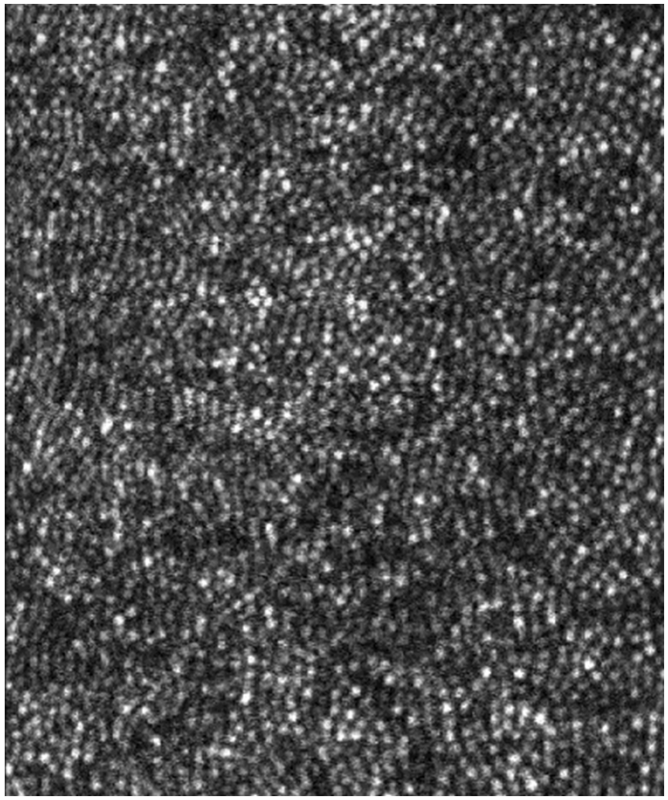
Sparse signals (edges!, changes!).
Reconstruction via nonlinear processing..

Flicker is different (noise) from the “wave” (continuous) method.

But subject to synchronization (in a window) because of nonlinearities.

The eye uses a tuning curve (a window) to capture the visual information.

Flicker usage in human eye.

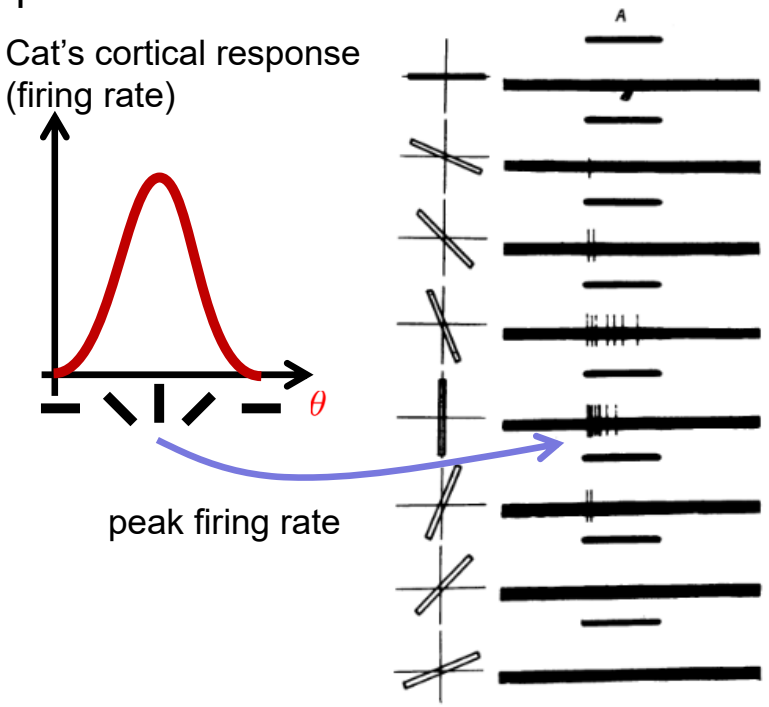


This is basically using of a kernel, or convolution, or Green's function.

Flicker detects edges for the eye's *V1* system just as a suitable convolution will do.

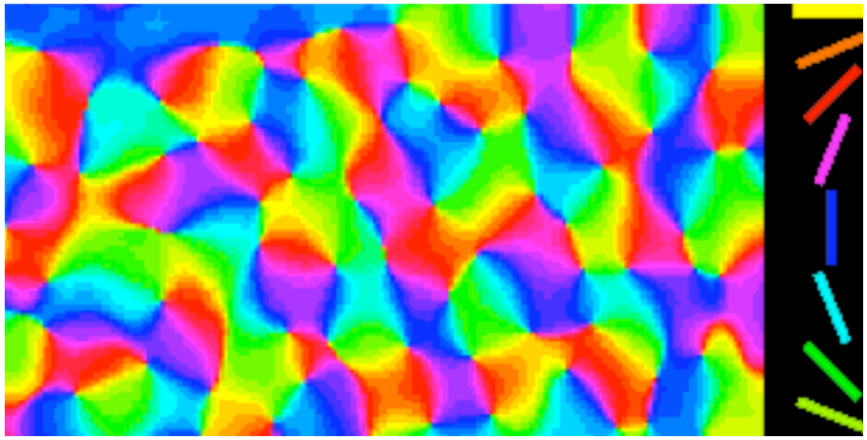
Tuning curve and neuron response

Spike trains coded to the tuning/power spectrum curve response.



Hubel & Wiesel (1959)

Functional map in V1



Butts (2004)

Visual information captured through tuning curve (a power spectrum!)
Flicker helps synchronize, thus becoming useful.

Noise, coupled to signal, can aid synchronization and fidelity.

Fisher information

Uncertainty, randomness, and incompleteness of information

Extremize Fisher information (or Cramér-Rao bound)

$\{x\}$

$x_i = \theta + \varepsilon_i$ $\qquad \langle \varepsilon^2 \rangle \geq \dfrac{1}{I(\mathfrak{p})}$, where $\qquad I(\mathfrak{p}) = \int [\partial_\theta \mathfrak{p}(x_i|\theta)]^2 \mathfrak{p}(x_i|\theta) dx_i$

$\equiv \int [\partial_x \mathfrak{p}(x)]^2 \mathfrak{p}(x) dx$

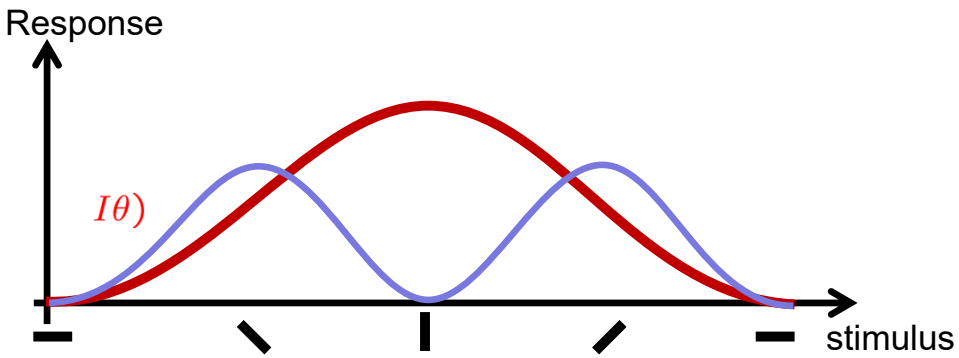
Example:

Best estimate of θ has a mean square error of $1/I$.

I projects smoothness. A normal $\mathfrak{p}(x)$ of variance σ^2 has $I = 1/\sigma^2$.

If I is small, error is large, so smoothest $\mathfrak{p}(x)$ consistent with additional information is the more likely fit.

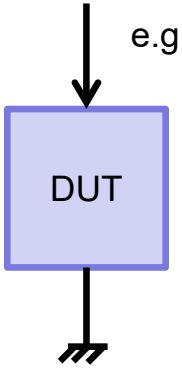
What does the eye--V1 do?
Firing rate to information



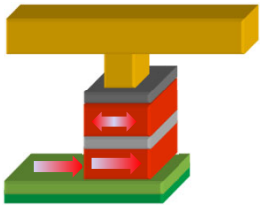
Fisher information: $$I(\theta) = \left\langle \left[\frac{\partial \ln \mathfrak{p}(r|\theta)}{\partial \theta} \right]^2 \right\rangle_r = \frac{1}{\sigma^2}[f'(\theta)]^2$$

Fisher information vanishes at peak and at no firing rate.

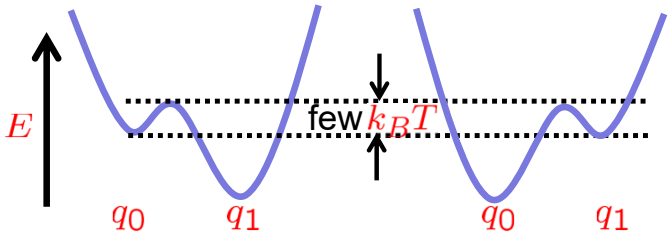
Noise/Fluctuations -> Random

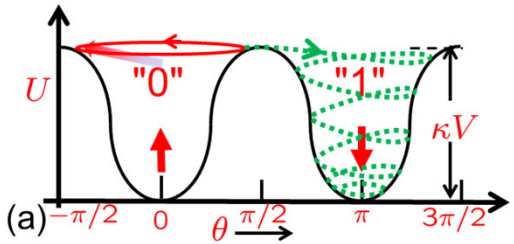


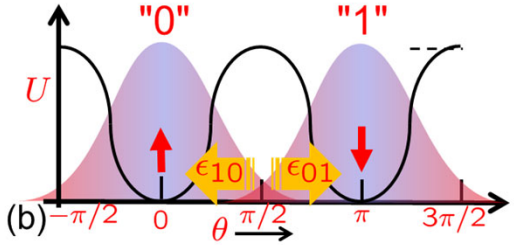
e.g., superparamagnetic junctions

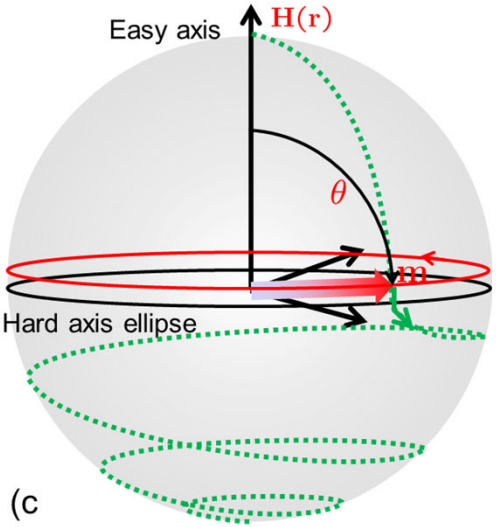


Thermal noise fluctuations

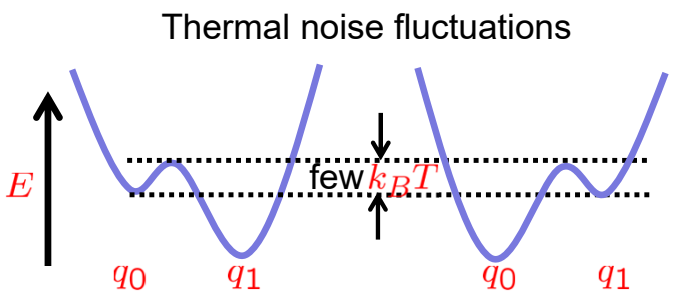


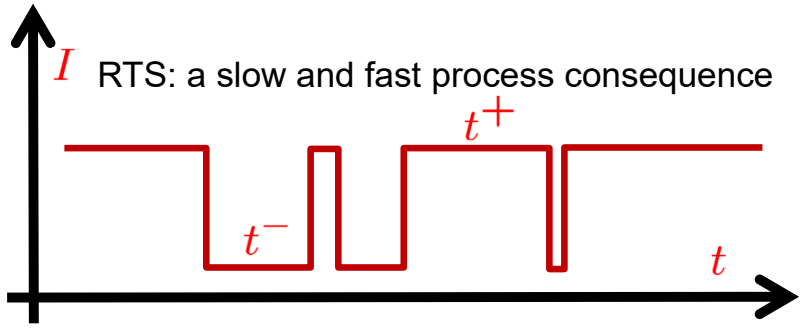






Random telegraph signal from thermal in Superparamagnetism



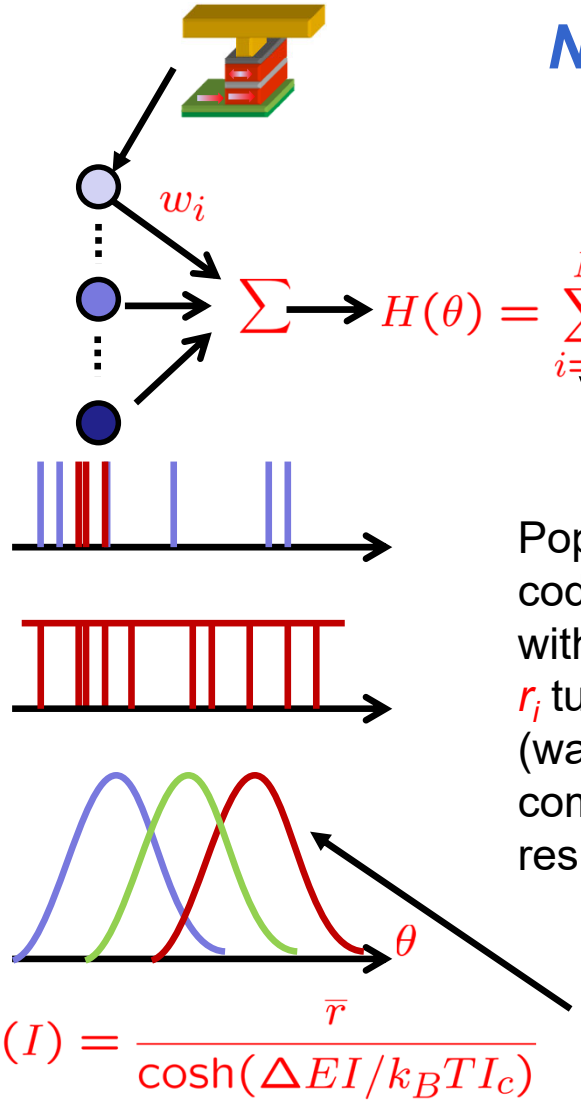


Poisson: $\mathfrak{p}(k,\nu T) = \dfrac{(\nu T)^k}{k!}\exp(-\nu T)$

$$\mathfrak{p}(t^{\pm}) = \frac{1}{\tau^{\pm}}\exp\left(-\frac{t^{\pm}}{\tau^{\pm}}\right)$$

This is flicker! Spike rate related to state transitions.

Noise/Fluctuations -> Random



Population coding with r_i tuned by current (wavefunctions that compose the response)

These are now basis functions obtained from different devices

$$r_{j,out} = \sum_{i}^{N} w_{ij} r_{i,in}$$

$$R = \frac{\sum_{j=1}^{N} I_B r_{j,out}}{\sum_{j=1}^{N} r_{j,out}}$$

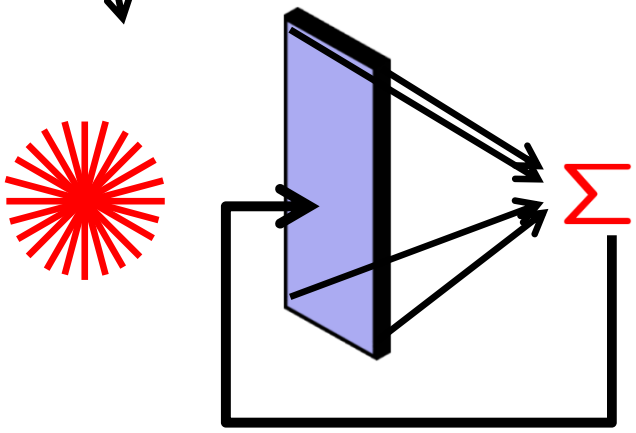
Net output spiking rate representing objective Tune through Fisher.

Feedback and minimize by Fisher measure.

Classification

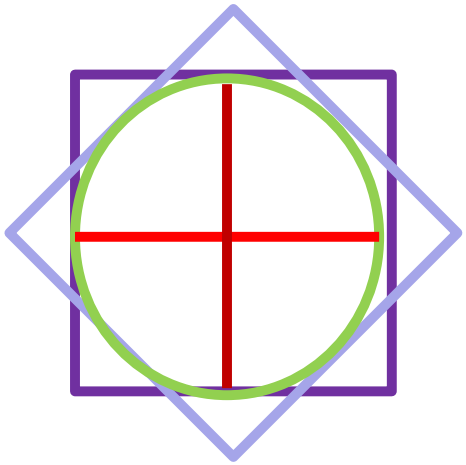
Model simulations

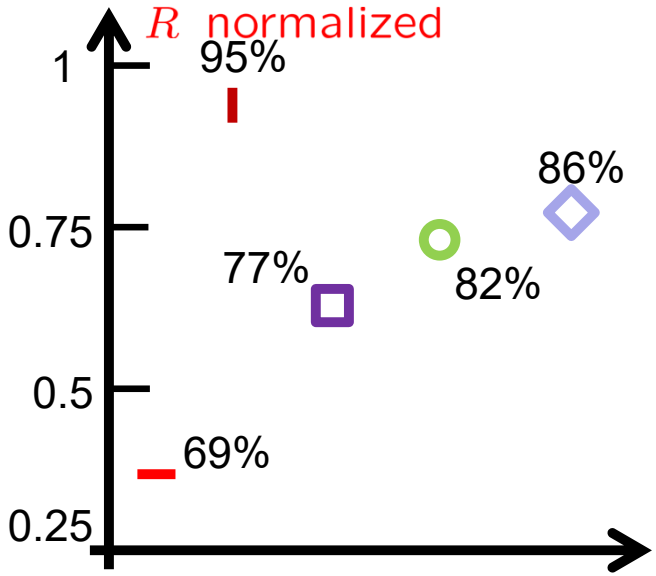
Superparamagnetic array



Train using horizontal, rotated, and vertical lines to minimize Fisher information

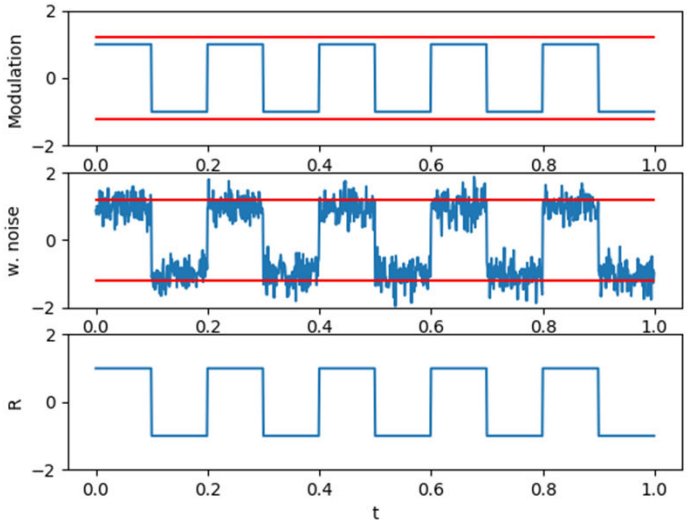
Test



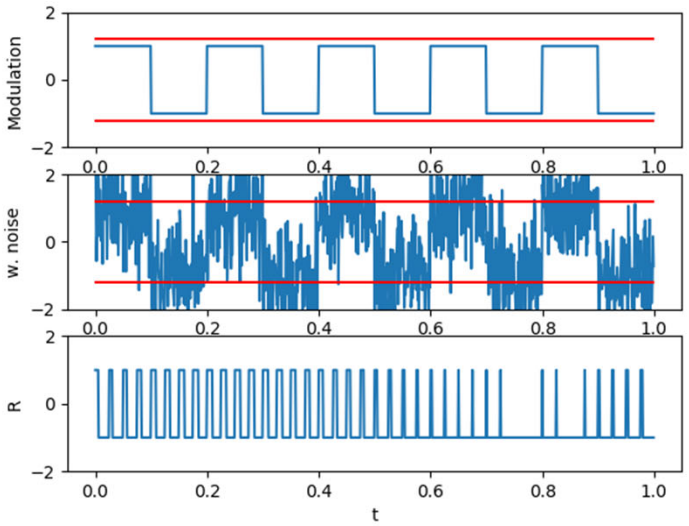


Window of synchronization

Synchronized state



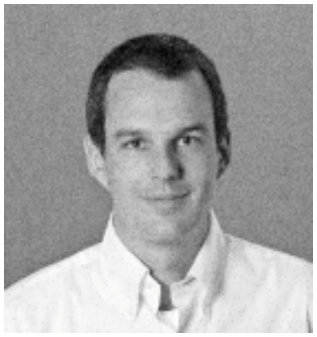
Too much thermal noise



Existence of noise (akin to dithering), linear superposition on modulation makes the synchronization happen.
The synchronization is nonlinear.
Signal fidelity improved in a window.

Noise + eye performing averaging improves the visual

This is like the flickering/noise



Through a python code with Gaussian noise

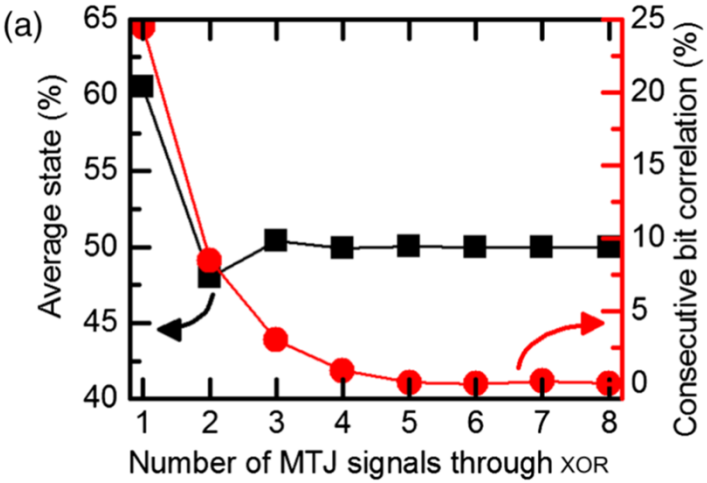
Random number generation

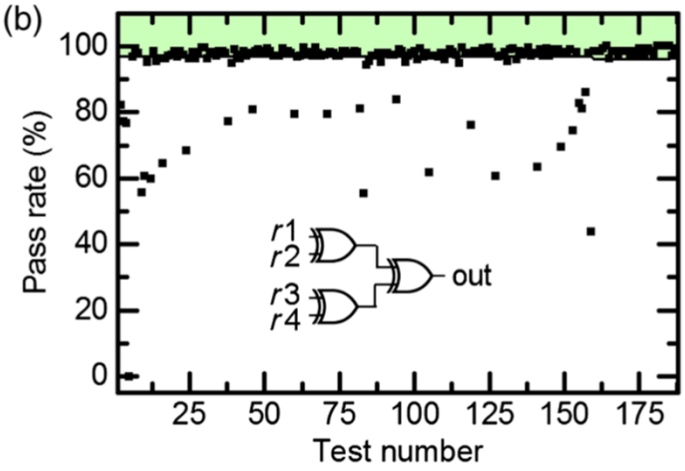
CMOS random generators are 20 pJ/bit.

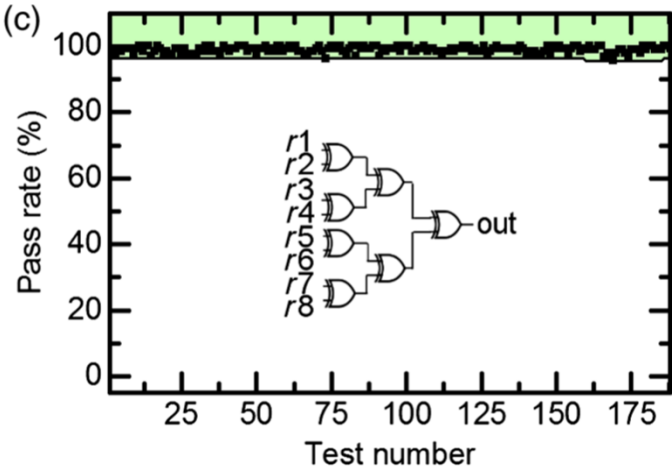
Sample the flicker noise of superparamagnetism; break in chunks, and then XOR it to remove any residual correlations.

x1000 lower energy and are square micrometer.

Whitened stream







Good useful random number generation at an energy 20 fJ per bit

Number of random numbers that can be generated is limited.

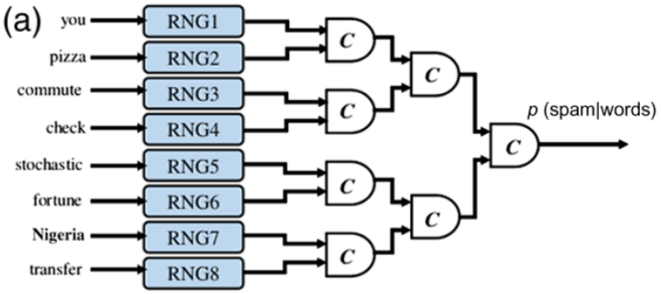
Current circuits do use energy (but not as much as thermal amplification).

Phys. Rev. App. 8, 054045 (2017)

Stochastic Low energy classifying

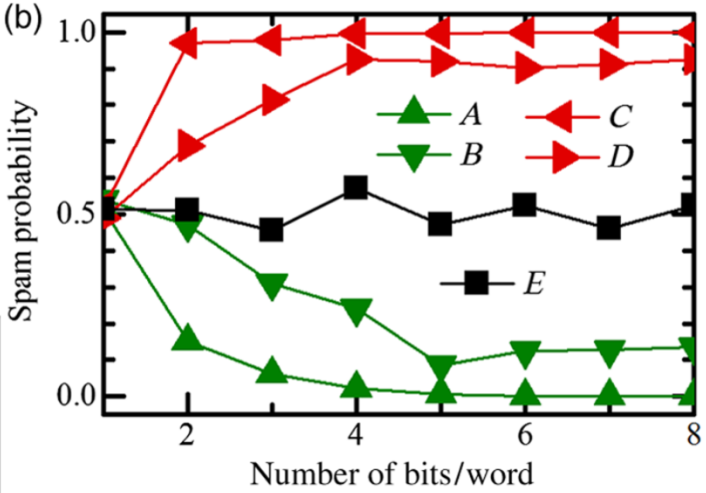
Take a dictionary of known words with their associated occurrence rates in spam and nonspam messages. Associate each word of the dictionary with a probabilistic random binary generator whose probability of drawing a 1 is set to different values depending on the presence (or absence) of the word in the presented sentence. Create multiple binary random generators and use Muller C elements for Bayesian inference.

(a)



A Do **you** want to get **pizza** after the **commute**?
B **You** should **check** these **stochastic** simulations.
C If **you** want to earn a **fortune**, send a **check** to **Nigeria**. We will **transfer** $10,000 to **your** account
D **You** are the 99th visitor. Subscribe and **you** will save a **fortune**.
E **You** won.

(b)



(c)

Method	Messages correctly classified
No whitening	60%
XOR4	100%
XOR8	100%
Blum	100%

Muller Cs; hysteresis FF

Phys. Rev. App. 8, 054045 (2017)

Data ≠ Information

Olshausen (2008)

Data ≠ Information

Olshausen (2008)

1 piece of data (black-fill) reveals the information.
Context was important. Algorithmic.
Information and data are not equivalent

Mooney faces (Data ≠ Information and Priors matter)

Bayesian Architectures

With observations x_0 , hidden variables x_h to be inferred, and contextual variables x_1 , the probabilistic relationship between them is:

$$\mathfrak{p}(x_0, x_1|x_h) = \mathfrak{p}(x_0|x_1, x_h)\mathfrak{p}(x_1|x_h)$$

prior

$$\mathfrak{p}(x_1|x_0, x_h)\mathfrak{p}(x_0|x_h) = \mathfrak{p}(x_0, x_1|x_h)$$

$$\therefore \quad \mathfrak{p}(x_1|x_0, x_h) = \mathfrak{p}(x_0|x_1, x_h)\mathfrak{p}(x_1|x_h)/\mathfrak{p}(x_0|x_h)$$

independent of what is hidden
/to be inferred
(a normalization)

This relationship form allows one to, e.g., maximize $\mathfrak{p}(x_1|x_0, x_h)$ by *a posteriori* estimation of x_1

This can be done at several hierarchical levels to arrive at inferences, such as matching patterns – a large class of difficult computational problems

Bayesian Circuits

Bayesian operations:

Multiplication: And gate $\mathfrak{p}(Output) = \mathfrak{p}(Input_1) \times \mathfrak{p}(Input_2)$

Addition:

ModOr gate $\mathfrak{p}(Output) = \underbrace{\mathfrak{p}(Input_1) + \mathfrak{p}(Input_2)}_{\text{pOr}} - \underbrace{\mathfrak{p}(Input_1) \times \mathfrak{p}(Input_2)}_{\text{pAnd}}$

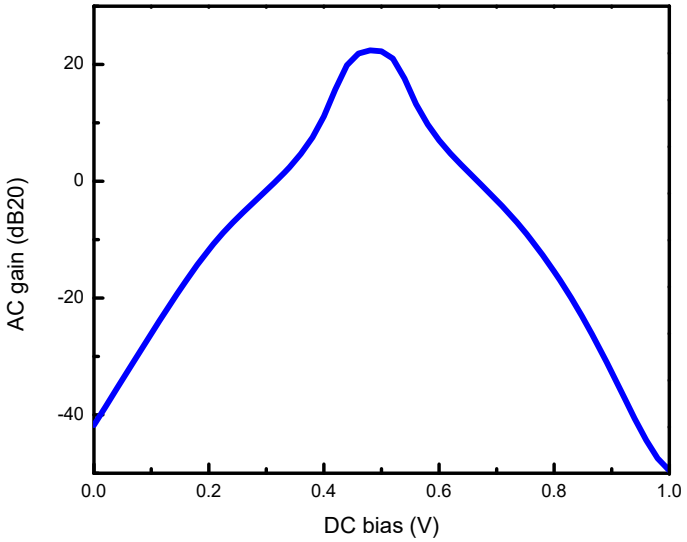
Or eliminate in circuit design the "1" probability for both inputs

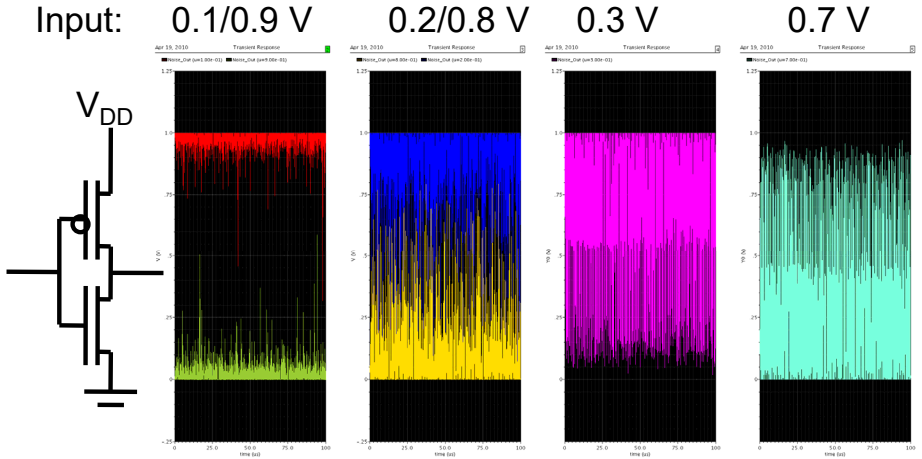
All probabilistic operations can now be mapped

Thermal probabilistic gates

Tiwari, NanoArch(2009)

IBM's 45 nm 12SOI

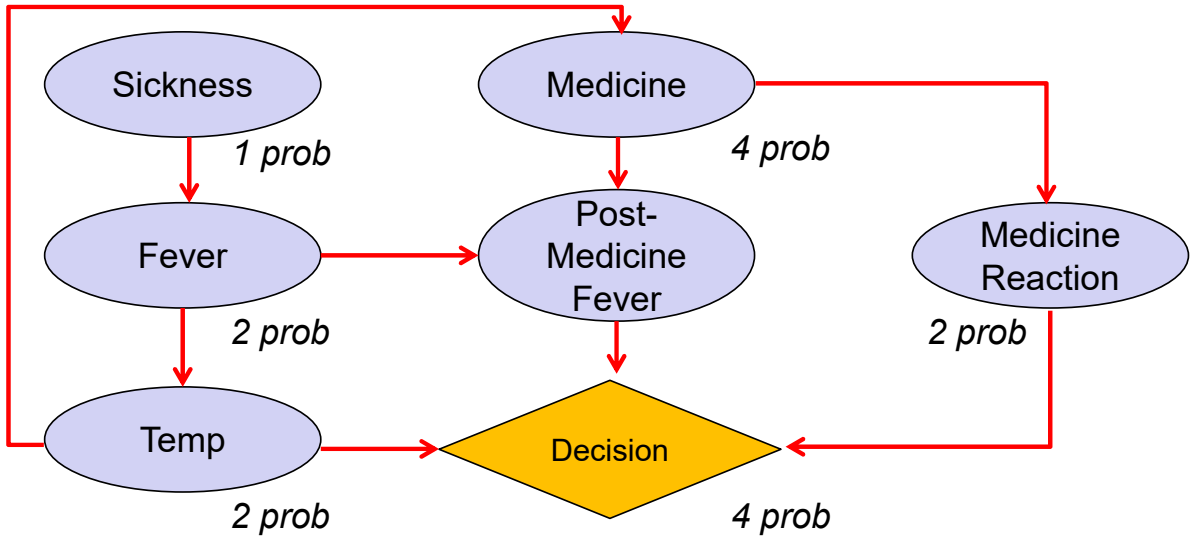


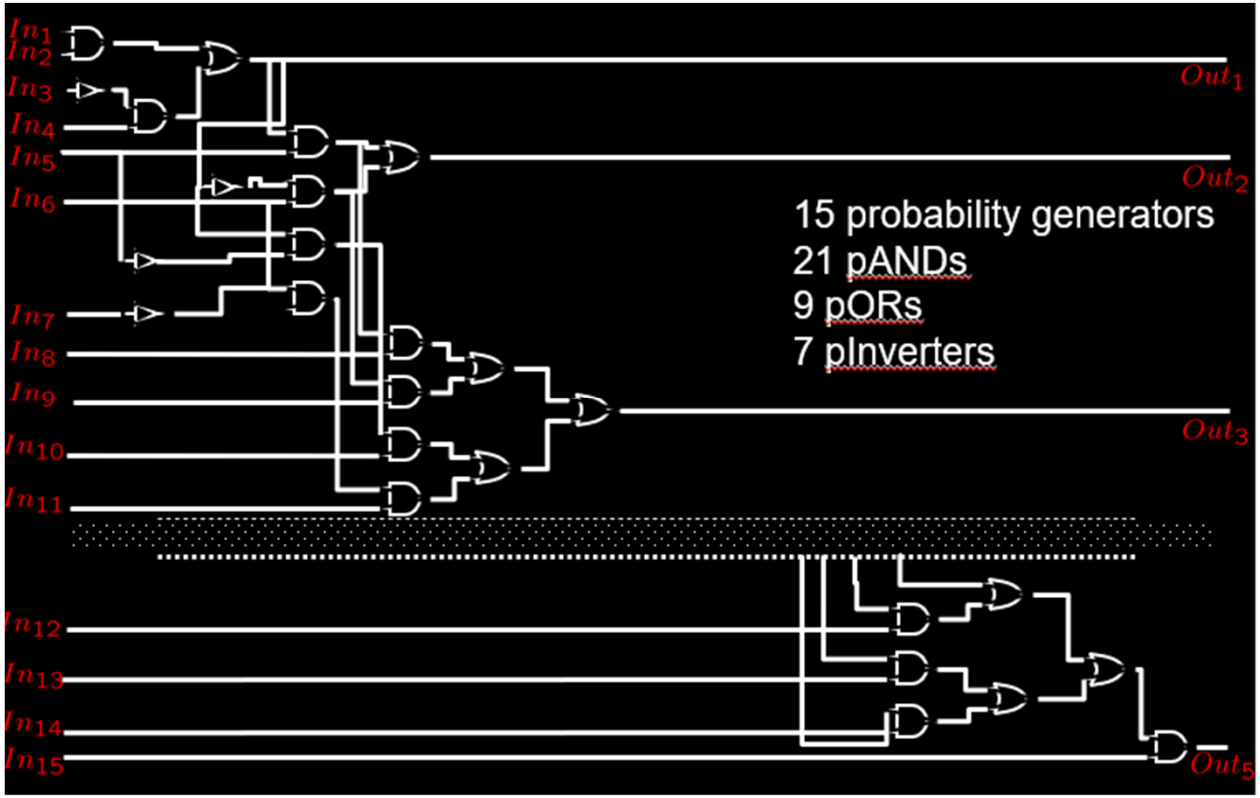


Glitches (Amplitude Errors) and jitters (Timing Errors) are generated when input noise is propagating through digital circuits.

Kim & Tiwari (2010)

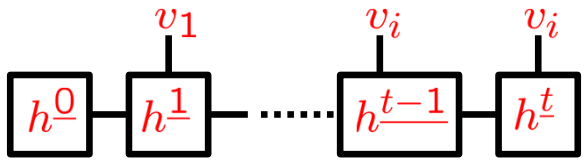
A simple example





It works, and it works with less precision even if some of the inner elements are pulled out. Is robust.

Hierarchical Markov Bayesian tracking

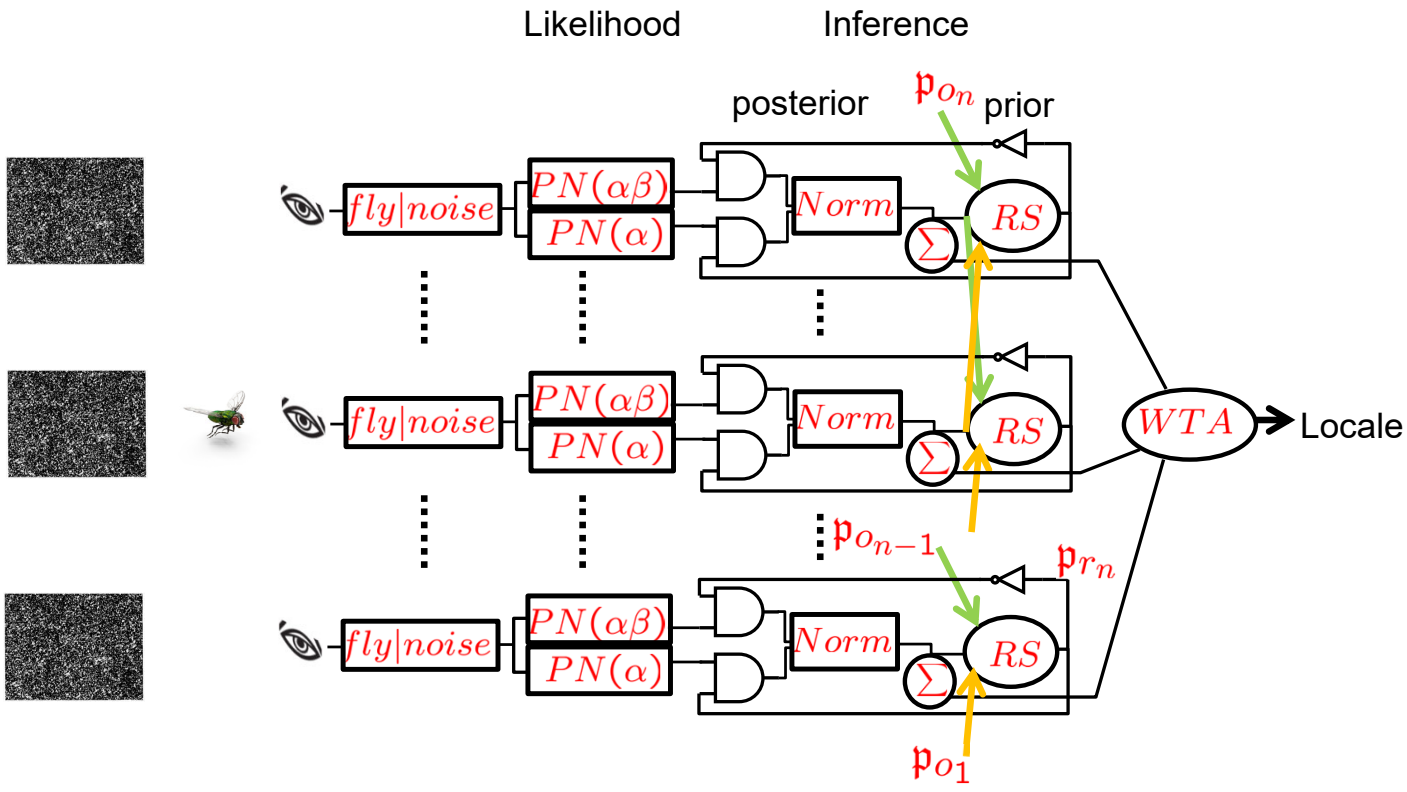


$$\mathfrak{p}(f_l^t|s^{1:t}) = \frac{\mathfrak{p}(s_l^t|f_l^t)\mathfrak{p}(f_l^t|s^{1:t-1})}{\mathfrak{p}(s_l^t|f_l^t)\mathfrak{p}(f_l^t|s^{1:t-1}) + \mathfrak{p}(s_l^t|\overline{f}_l^t)\mathfrak{p}(\overline{f}_l^t|s^{1:t-1})}$$

$$\mathfrak{p}(s_i|f_i) = \alpha\mathfrak{p}(s_i|\overline{f}_i) = \alpha\beta\mathfrak{p}(\overline{s}_i|f_i) = 1-\alpha\beta\mathfrak{p}(\overline{s}_i|\overline{f}_i) = 1-\alpha\beta$$

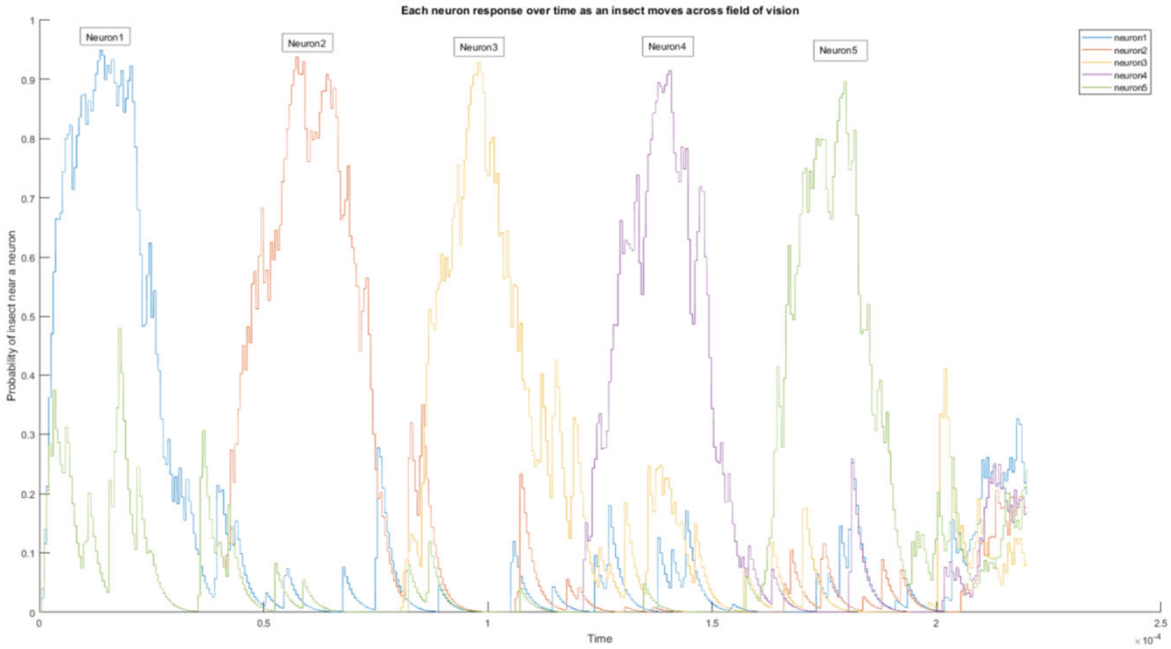
HMM Bayesian: A Poisson neuron toy

Poisson neuron



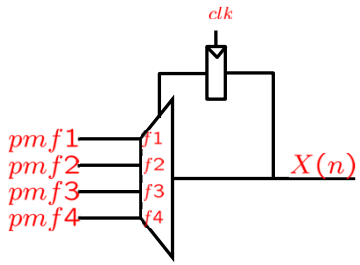
After C. S. Thakur et al, Frontiers in Neuroscience (2016)

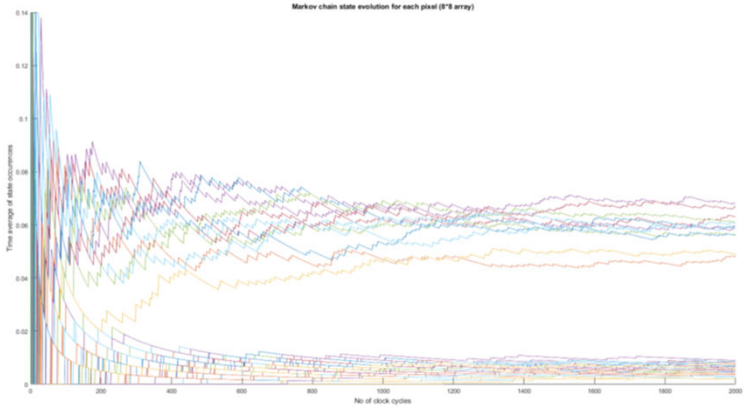
As the fly passes by (Poissonian)



Non-Turing mapping: Image recovery

Contraction mapping on state space with iterated function (low area)

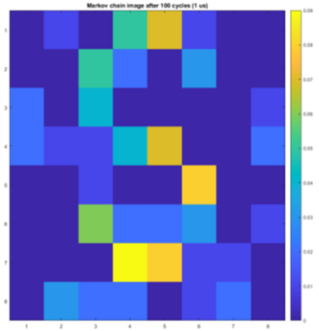




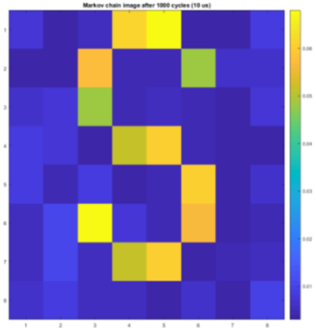
Iteration in Markov chain

Clocked tuning

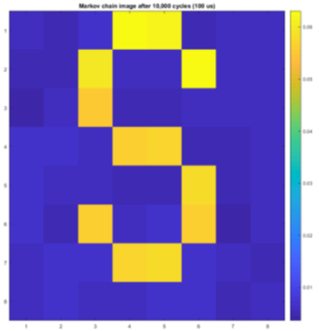
10^2 iterations



10^3 iterations



10^4 iterations



There are places where one can relax the error-free constraint.

Approaches of LSB relaxation are of limited utility.

This will be true for the AI/ML world too.

Using randomness and probabilities at the edge of the computational world---human-centric interfaces of all types---can be useful.

Stochastic, e.g., is low energy, small area, tolerant to error, and progressively precise.

Bayesian provides robustness.

(but the center of technology world will not change. The edge world will)

Next talk: a physics-centric view of the AI/ML world and its utility as a new tool for research and understanding.

I thank students over the years, insightful colleagues and my teachers who have influenced this thinking and collaborated in the pursuits.