Building a large-scale brain model with spiking neurons (a recent example)

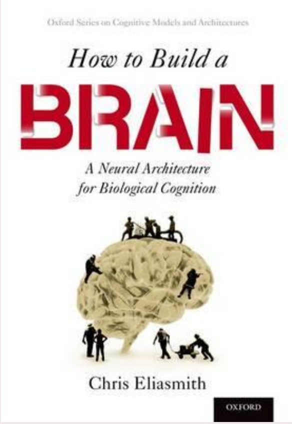
5TH APRIL 2022

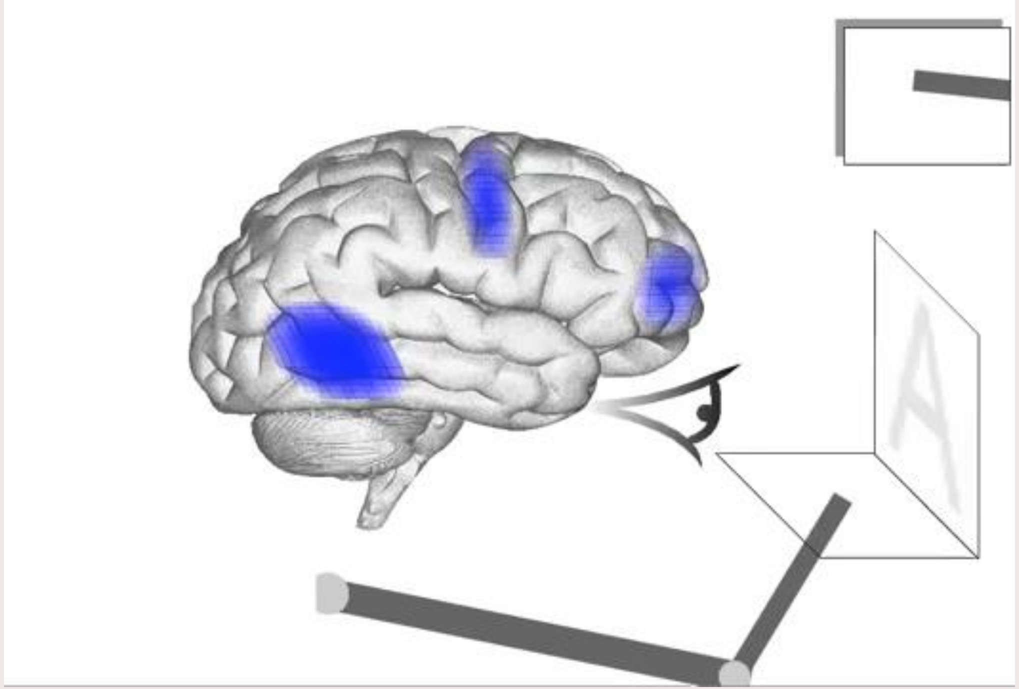
Federico Corradi, Assistant Professor, Neuromorphic Edge Computing Systems Lab.

SPAUN: Semantic Pointer Architecture Unified Network

MULTI-BRAIN AREA SPIKING NEURAL NETWORK SIMULATION

2.5 Million of Leaky-Integrate-and-Fire Neurons with Dynamic Synapses





Eliasmith C, Stewart TC, Choo X, Bekolay T, DeWolf T, Tang Y, Rasmussen D. *A large-scale model of the functioning brain*. **Science**. **2012** Nov 30;338(6111):1202-5.

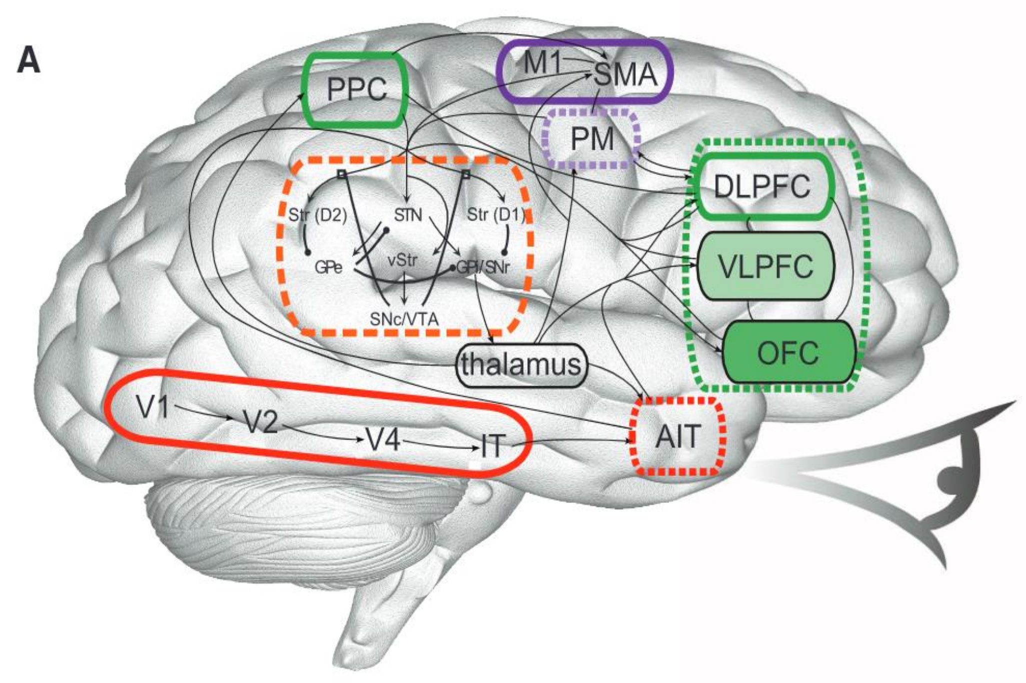
https://www.youtube.com/watch?v=P_WRCyNQ9KY

SPAUN: Semantic Pointer Architecture Unified Network

SPAUN ANATOMICAL DESCRIPTION

Neurons are assigned to specific anatomical areas.

The neuronal ***behavior correlates*** to the kind of behavior of those anatomical areas they represent and to the performance of the task. (e.g., bad recall, check the representation in the network)



SPAUN: Semantic Pointer Architecture Unified Network

SPAUN FUNCTIONAL DESCRIPTION

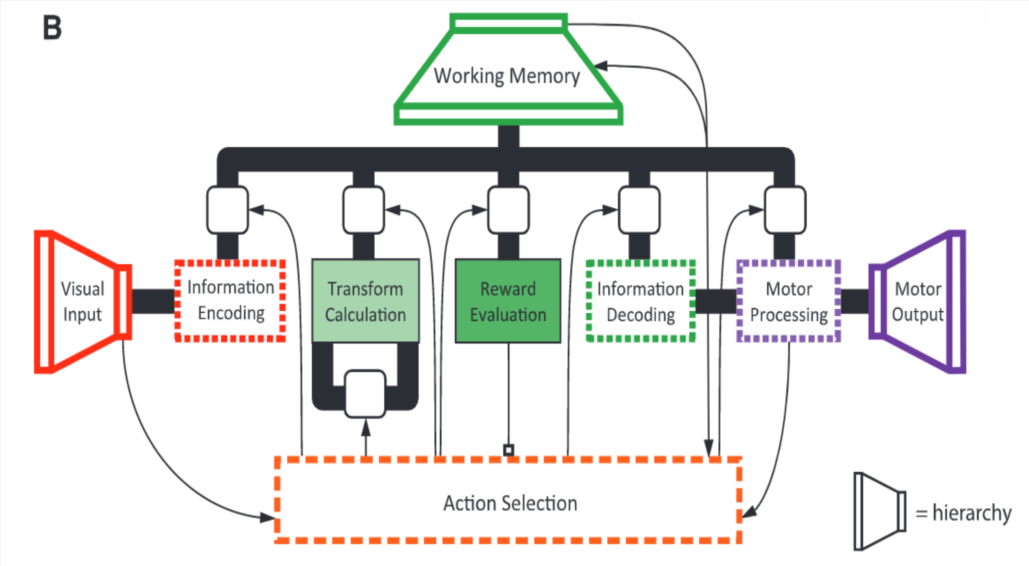
Visual hierarchy *(left)*

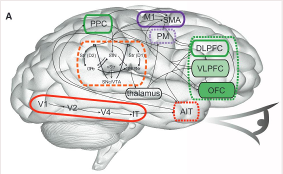
Motor output *(right)*

Working memory *(top)*

Action selection (basal ganglia gating information)

It can be used to ask functional and biological questions, for example, we can simulate changes in activity versus behavior.



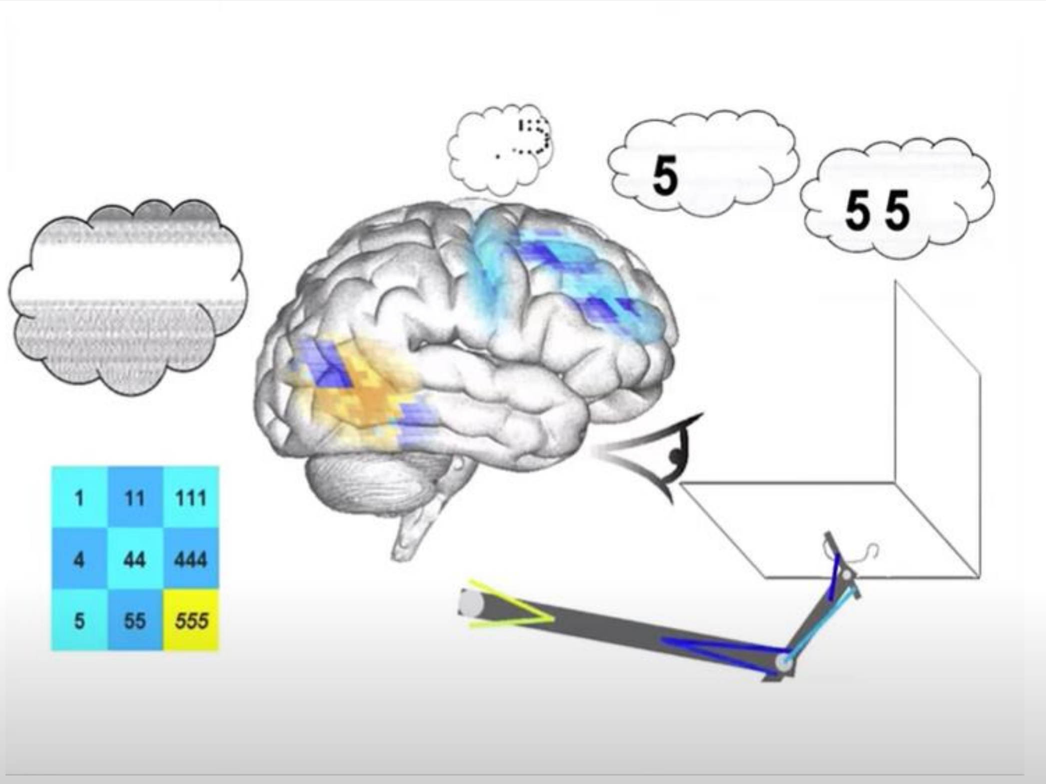


Bigger model will get more sophisticated!!

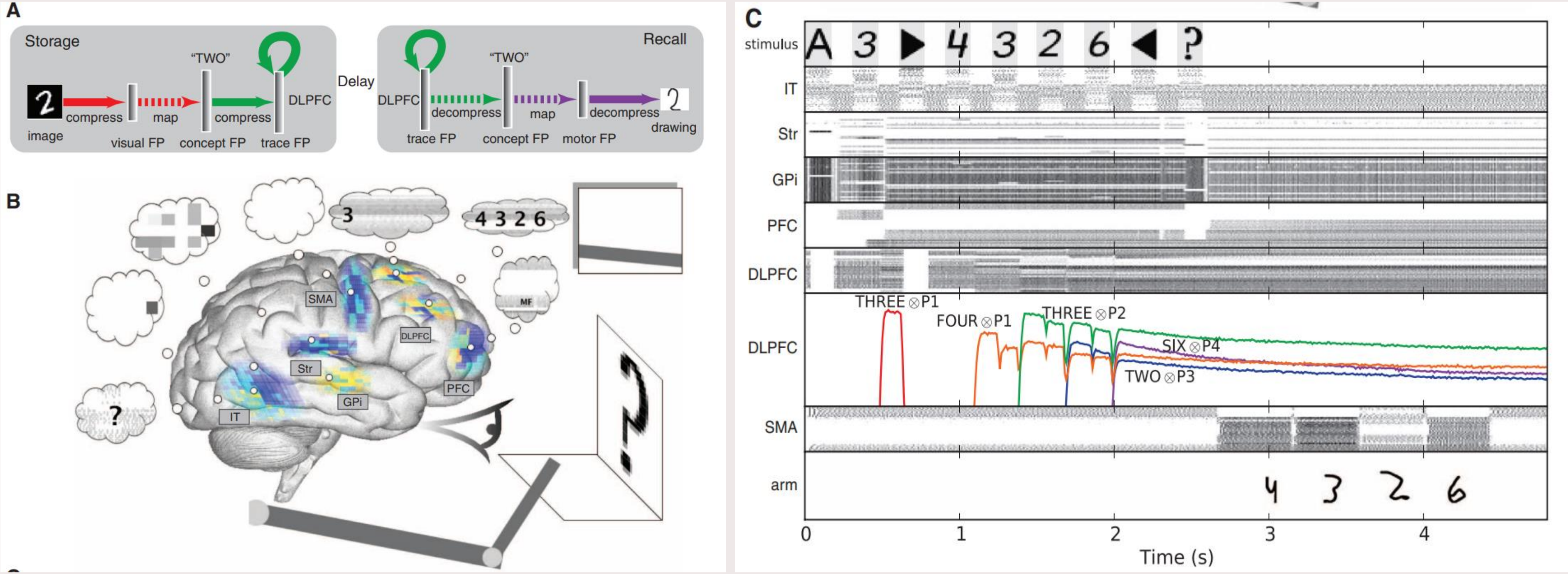
SPAUN: 12 Tasks

1. **Copy Drawing** (MNIST digits)
2. **Digit recognition** (MNIST)
3. **List memory** (reproduce list)
4. **N-arm bandit task** (reinforcement learning)
5. **Counting** (sum of two values)
6. **Simple question answering** (what element is in position x of the list, what position is the number in the list)
7. **Rapid variable creation** (e.g., 0 0 7 4 → 7 4; 0 0 2 4 → 2 4; etc)
8. **Fluid Induction** (Raven Progressive Matrices)
9. **Adaptive arm control** (adapt to varying forces applied on the arm)
10. **Stimulus matching task** (ImageNet retrieval of images in the same category)
11. **Stimulus response task** (given an image classify it accordingly to its classifier)

12..



SPAUN: Task example list memory (reproduce list)



[C. Eliasmith at al., Science 2012]

SPAUN: Semantic Pointer Architecture Unified Network

Why? Compare to experimental data

LINK BEHAVIOR WITH THE MODEL

List of digits and you must repeat them back.

Similar features (people and SPAUN are good at remembering digits at the beginning and the end of the list)

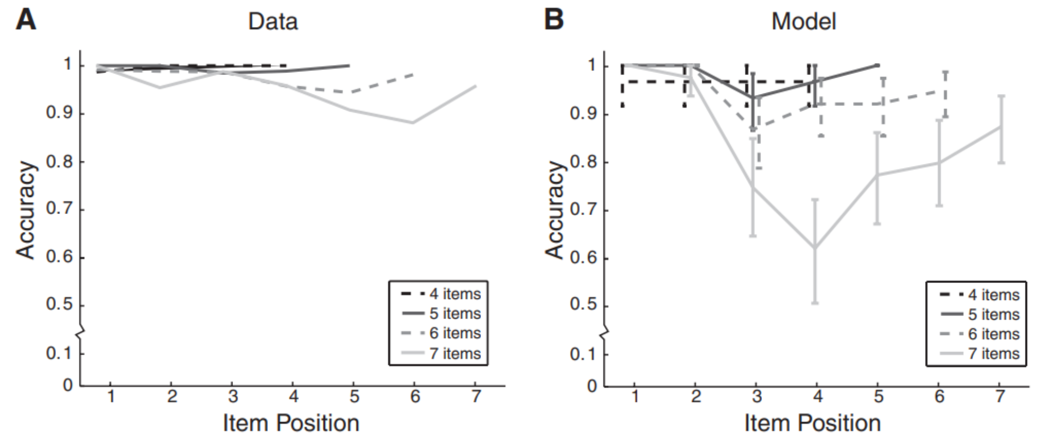


Fig. 4. Population-level behavioral data for the WM task. Accuracy is shown as a function of position and list length for the serial WM task. Error bars are 95% confidence intervals over 40 runs per list length. (**A**) Human data taken from (*18*) (only means were reported). (**B**) Model data showing similar primacy and recency effects.

[C. Eliasmith at al., Science 2012]

SPAUN how does it work?

Uses the Neural Engineering Framework (NEF) for simulating spiking neural networks (LIF models).

Semantics (encoding information efficiently)
Syntax (to build structures and representation to build over those)
Control (how to perform motor control, move the arm)
Learning & Memory (flexibility and supporting behaviors)

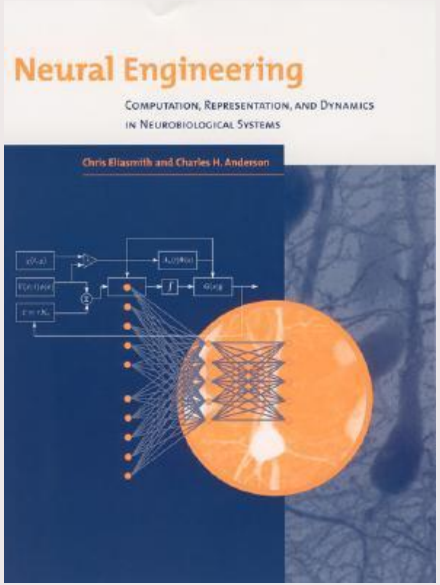
- How do we *use* neurons to do all the tasks together?
- How do you *connect* neurons?
- How do you *program* networks of spiking neurons? (not only functions, but state machines, dynamical systems, motor control, etc.)

SPAUN how does it work? It is based on Neural Engineering Framework (NEF)

SPUAN is built using the **Neural Engineering Framework** (NEF). NEF is a technique used to **construct** and **simulate** spiking neural networks. NEF is based on linear control theory, and it uses a set of primitives.

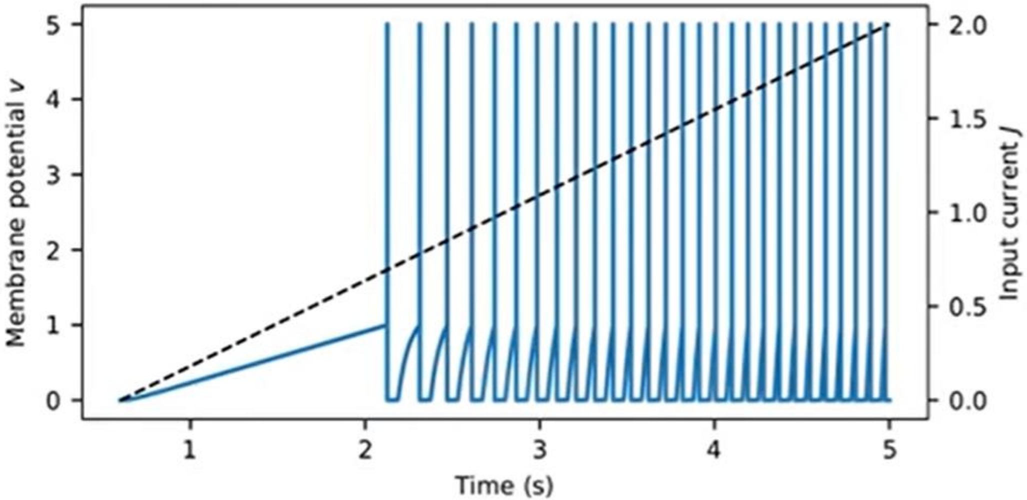
- *Encoding / Decoding*
- *Transformation*
- *Dynamical Systems*
- *Memory & Learning*

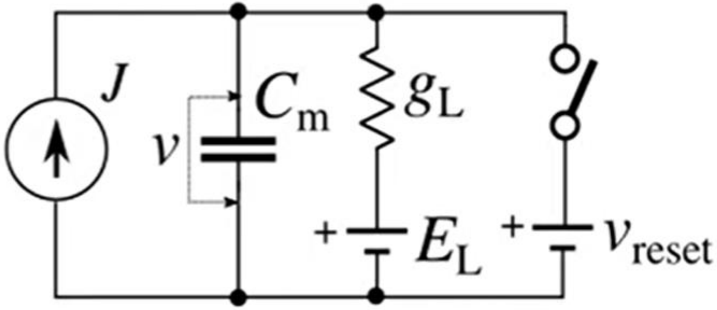
- *www.nengo.ai (free software tool)*



Eliasmith C, Anderson CH. Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT press; 2003.

RECAP: *Leaky-Integrate-and-Fire Neuron Model*





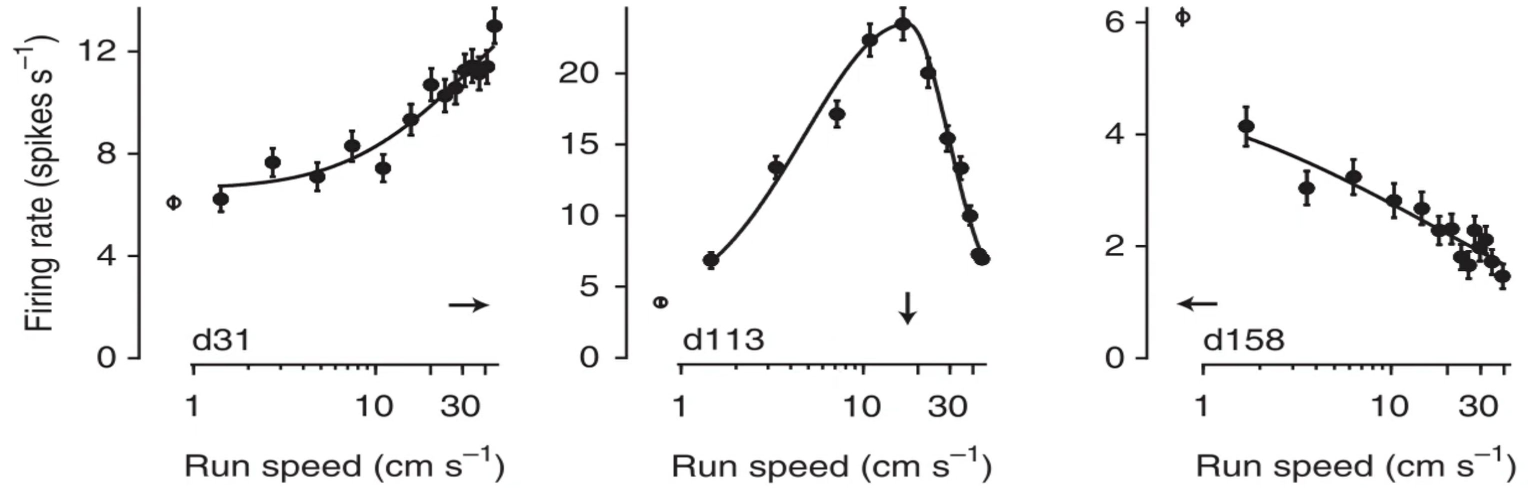
$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} v(t) &= -\frac{1}{\tau_{\mathrm{RC}}}\left(v(t) - J\right), && \text{if } v(t) < 1, \\
v(t) &\leftarrow \delta(t - t_{\mathrm{th}}), && \text{if } t = t_{\mathrm{th}}, \\
v(t) &\leftarrow 0, && \text{if } t > t_{\mathrm{th}} \text{ and } t \geq t_{\mathrm{th}} + \tau_{\mathrm{ref}},
\end{aligned}
$$

Encoding: response curves & tuning curves

Example:

- ***sound intensity at different brain regions***
- ***orientation selectivity (lines at different angles)***
- ***speed (how fast an animal is running)***

How to model all these different tuning curves?



NEF: response curves & tuning curves

$$a = f(x) = G(J_i(x))$$

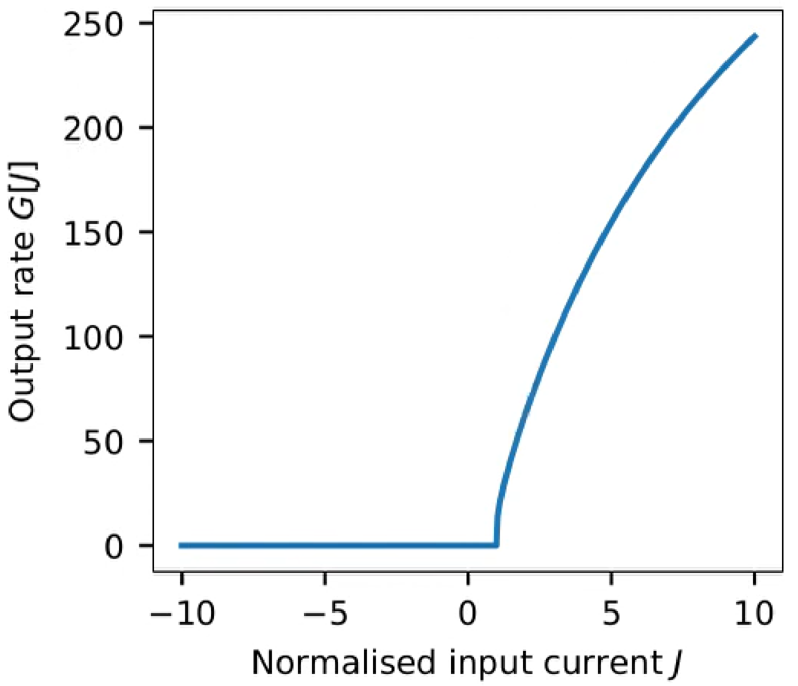
Introduce a gain α_i and a bias J_i^{bias}:

$$J_i(x) = \alpha_i x + J_i^{\text{bias}}$$
$$a_i(x) = G(\alpha_i x + J_i^{\text{bias}})$$

α_i controls the slope

J_i^{bias} shifts curve left and right

Mean-rate model

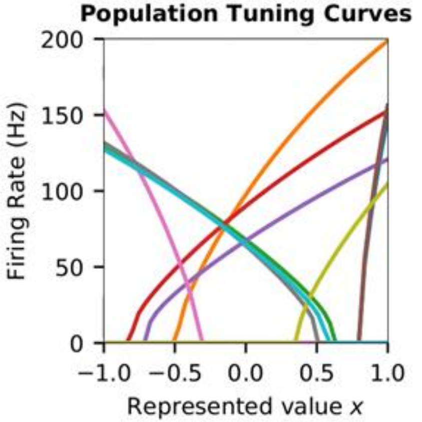


*A neuron represents values via **nonlinear encoding**.*

NEF: response curves & tuning curves

$$\mathbf{a}_i = G\left[\alpha_i \langle \mathbf{x}, \mathbf{e}_i \rangle + J_i^{\text{bias}}\right],$$ Encoding

$$\hat{\mathbf{x}} = \mathbf{D}\mathbf{a}$$ Decoding



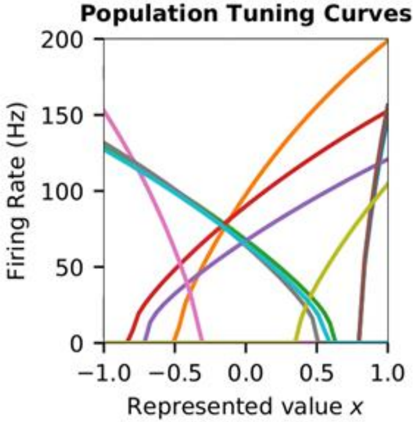
$$\arg\min_{\mathbf{D}} E = \frac{1}{|\mathbb{X}|}\int_{\mathbb{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|\, d\mathbf{x} = \frac{1}{|\mathbb{X}|}\int_{\mathbb{X}} \|\mathbf{x} - \mathbf{D}\mathbf{a}(\mathbf{x})\|\, d\mathbf{x}$$

By using least-square minimization we can find the optimal linear decoders.

NEF: response curves & tuning curves

$$\mathbf{a}_i = G\left[\alpha_i \langle \mathbf{x}, \mathbf{e}_i \rangle + J_i^{\text{bias}}\right],$$ Encoding

$$\hat{\mathbf{x}} = \mathbf{D}\mathbf{a}$$ Decoding



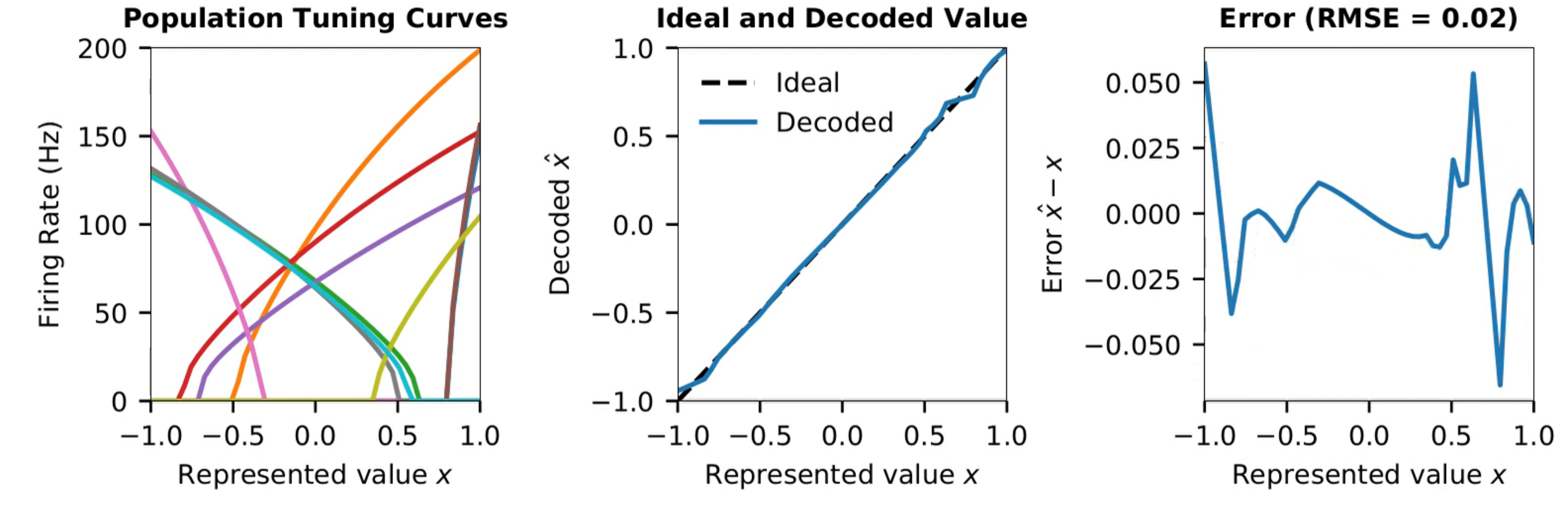
In python np.linalg.lstsq

$$\arg\min_{\mathbf{D}} E = \frac{1}{N}\sum_{i=0}^{N} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}(\mathbf{x}_i)\|$$

If we can't do that analytically we can use the samples that are available.

NEF: response curves & tuning curves

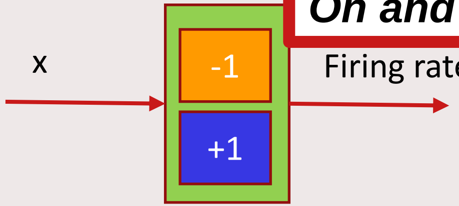
NB: It works well with mean-rates (at steady states)!

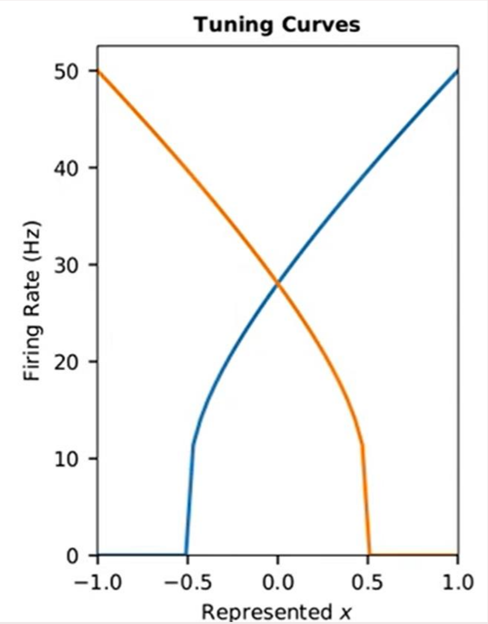


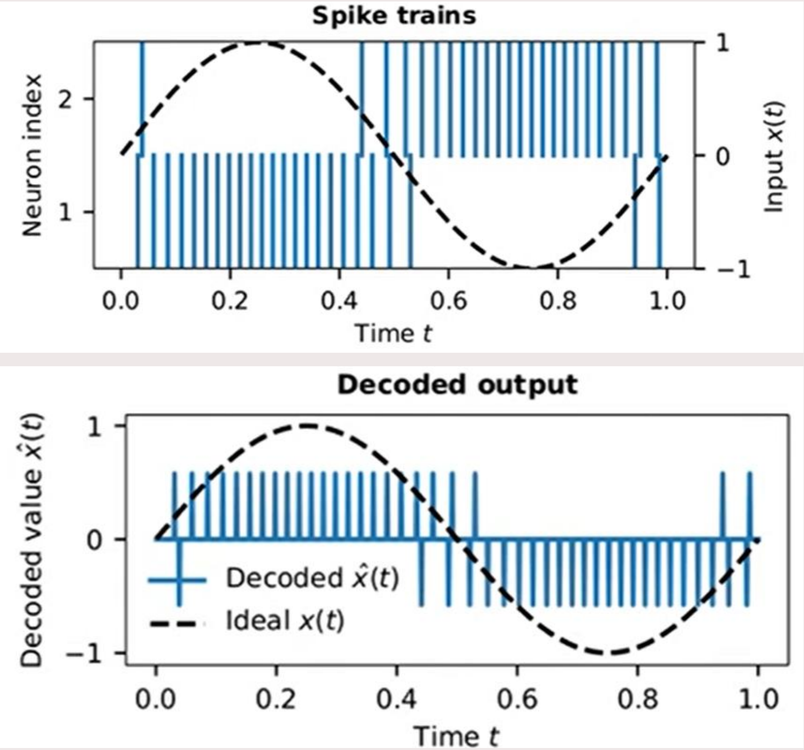
$\mathbf{A}$ $\mathbf{A}^T\mathbf{D}^T$ $\mathbf{A}^T\mathbf{D}^T - \mathbf{X}^T$

Temporal Encoding with a population of *LIF neurons*

On and Off neurons



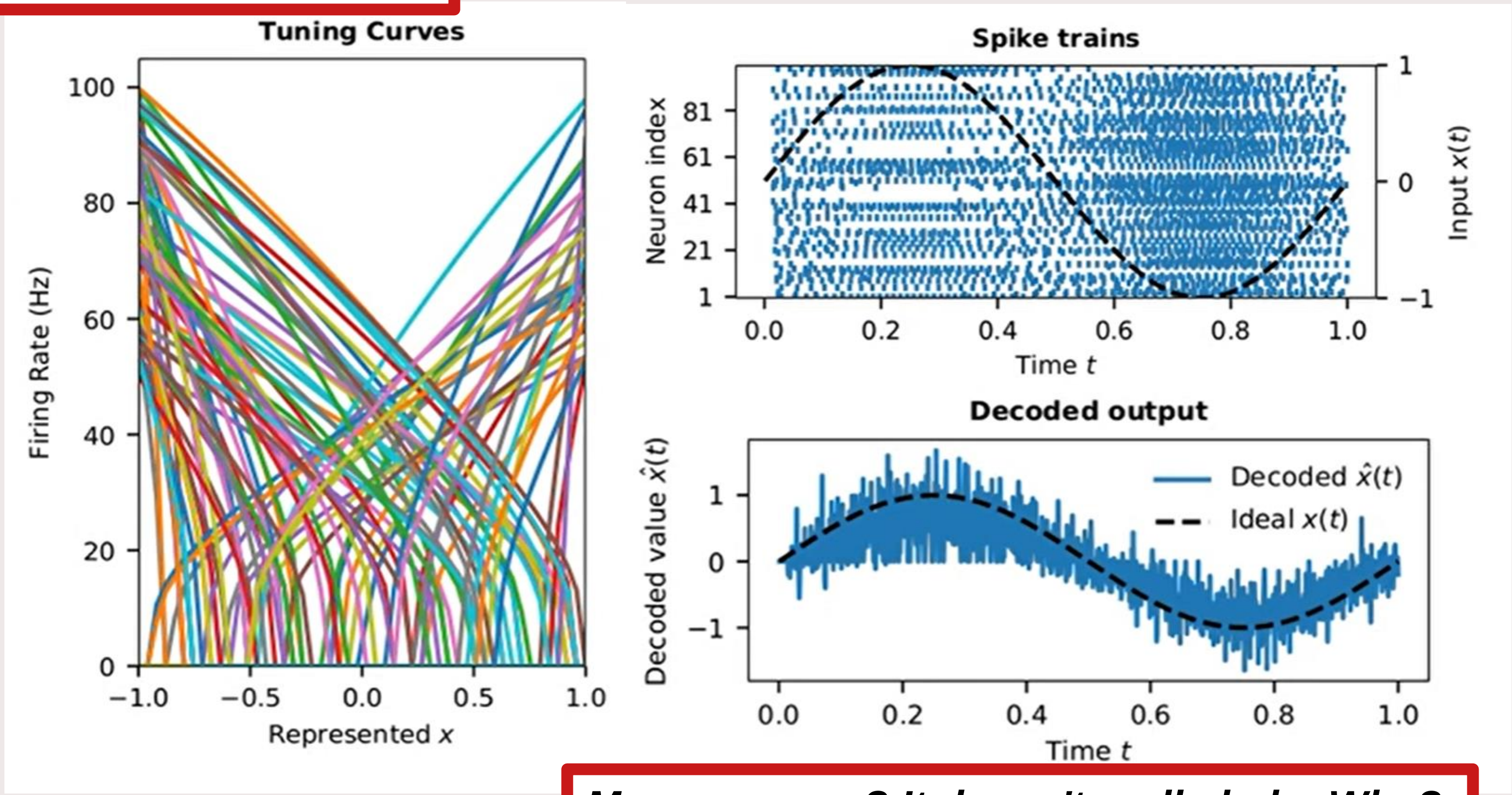




Temporal varying signals are more complicated!

Temporal Encoding with a population of *LIF neurons*

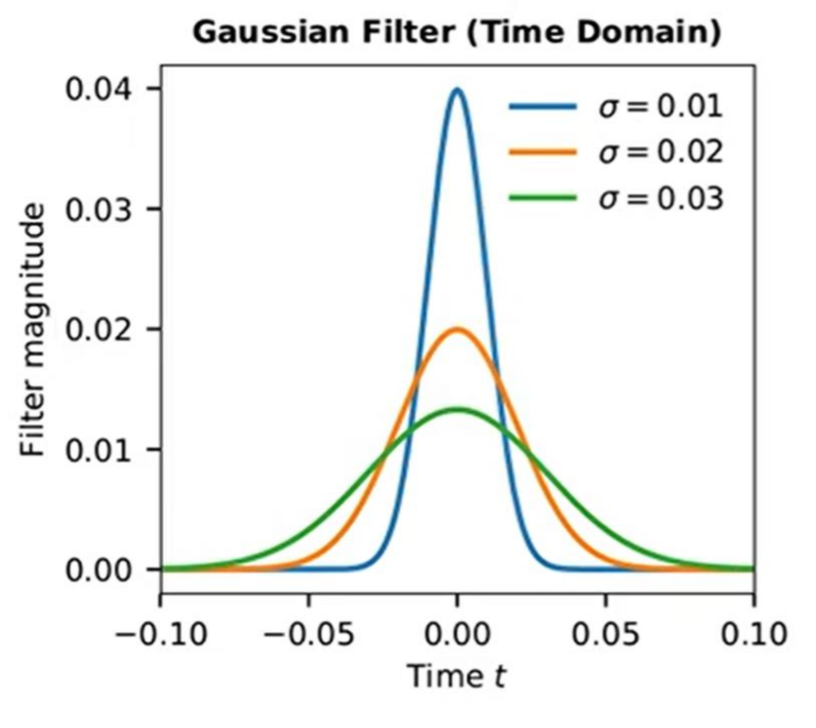
What about more neurons?



More neurons? It doesn't really help. Why?

Filtering with convolutions

Apply kernels to the spike trains, to be able to approximate time-varying input stimuli more precisely.



Gaussian Filter

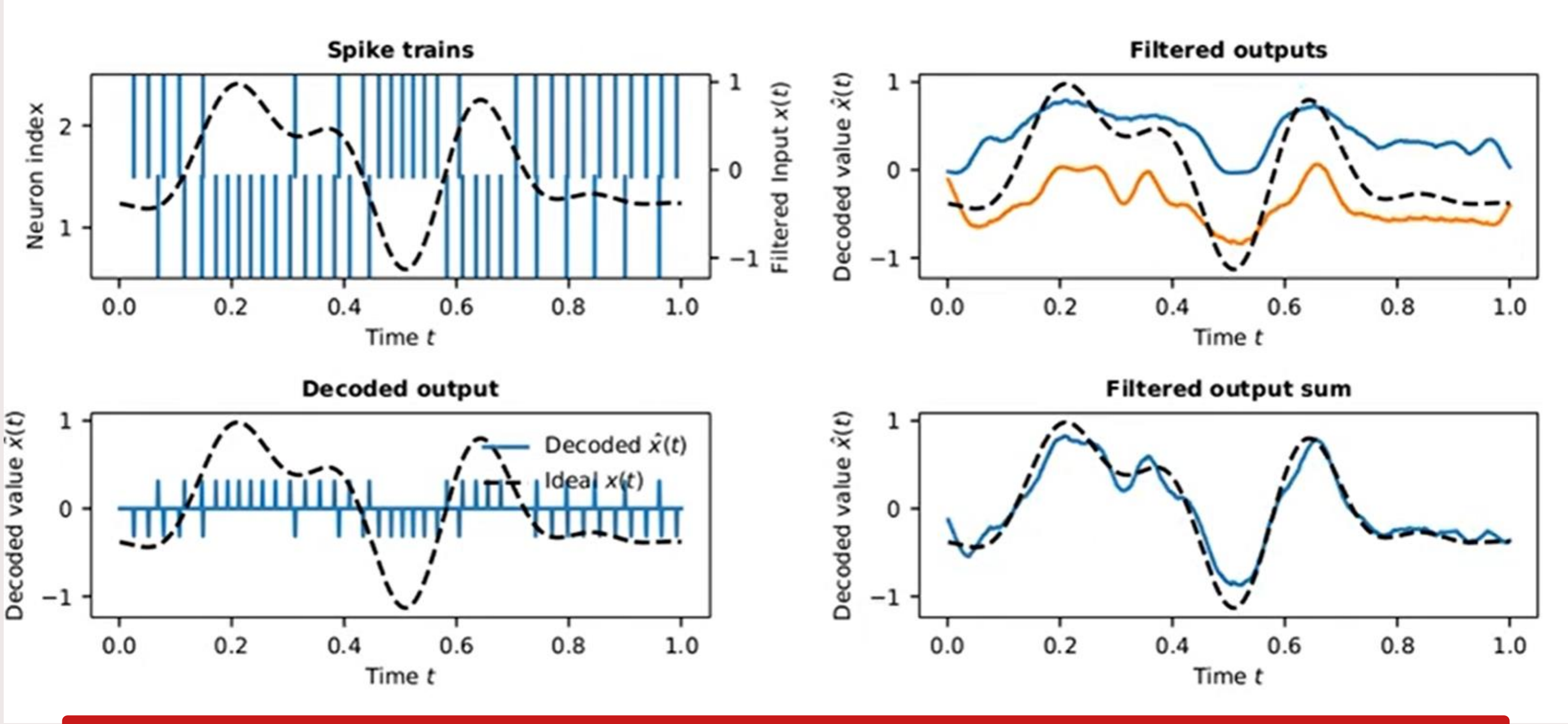
$$h(t) = c\exp\left(\frac{-t^2}{\sigma^2}\right)$$

where c chosen s.t. $\int_{-\infty}^{\infty} h(t)\,\mathrm{d}t = 1$

Convolution

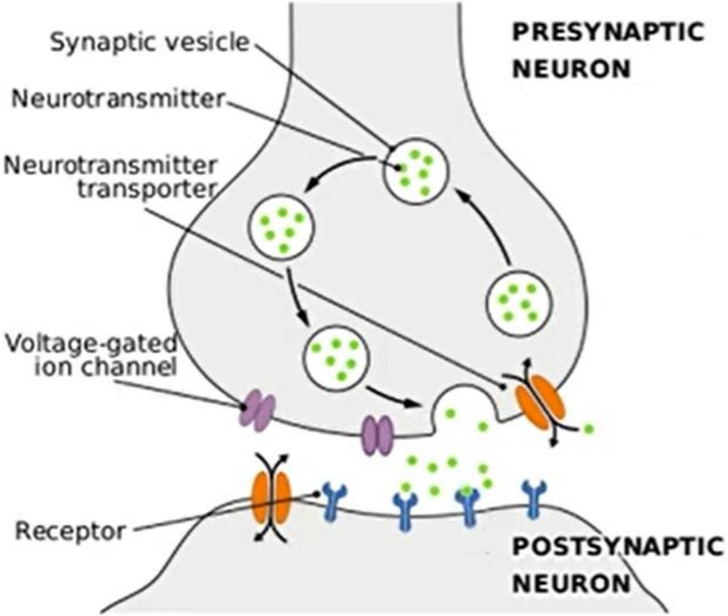
$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,\mathrm{d}\tau$$

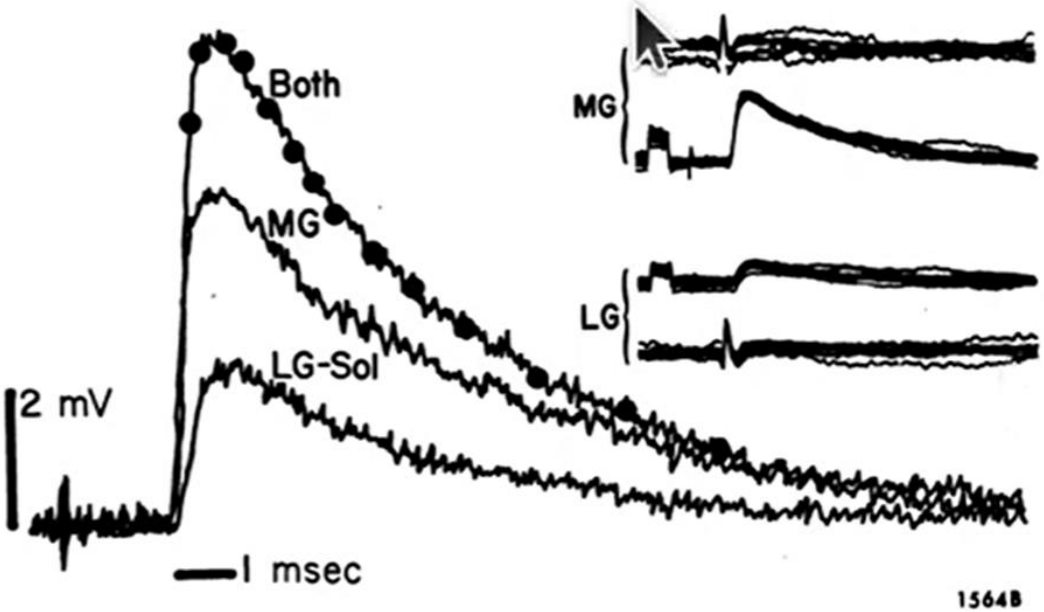
Filtering with convolutions



Pros: precision of encoding
Cons: Gaussian kernels have no real biological foundation (non-causal!)

A bio-realistic synaptic filter

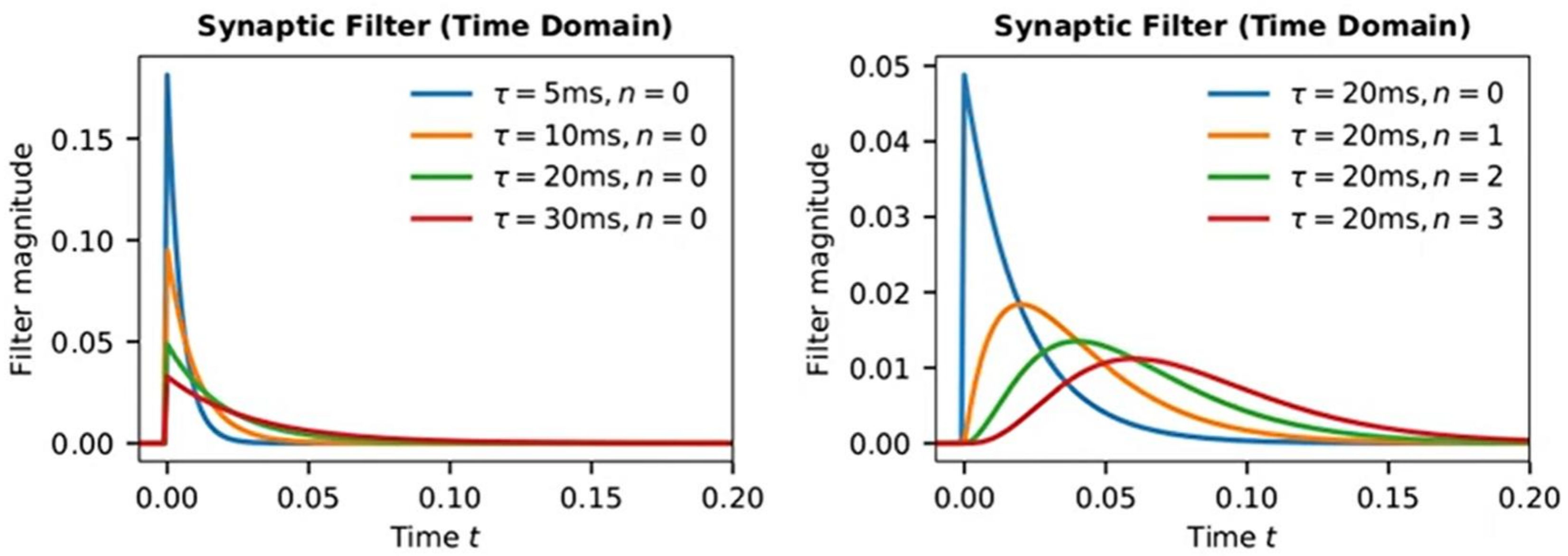




Postsynaptic filtering can be used to model different neurotransmitters, and it is causal!

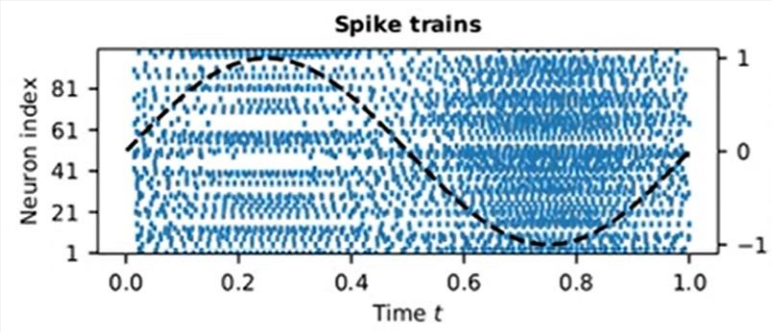
TU/e

A bio-realistic synaptic filter

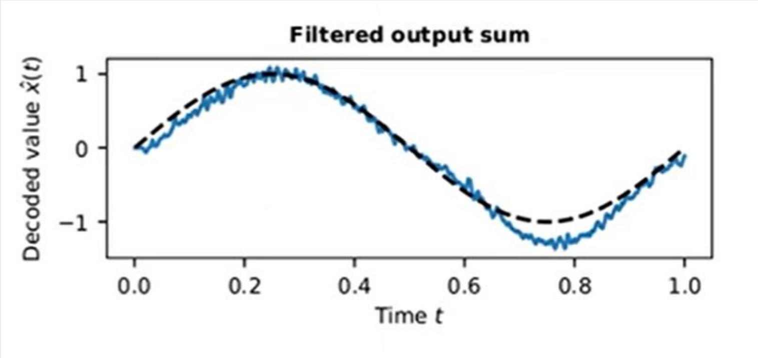


$$h(t) = \begin{cases} c^{-1} t^n \exp^{-t/\tau} & \text{if } t \geq 0\,, \\ 0 & \text{otherwise}\,, \end{cases} \qquad \text{where } c = \int_0^\infty t^n \exp^{-t/\tau}\, \mathrm{d}t\,.$$

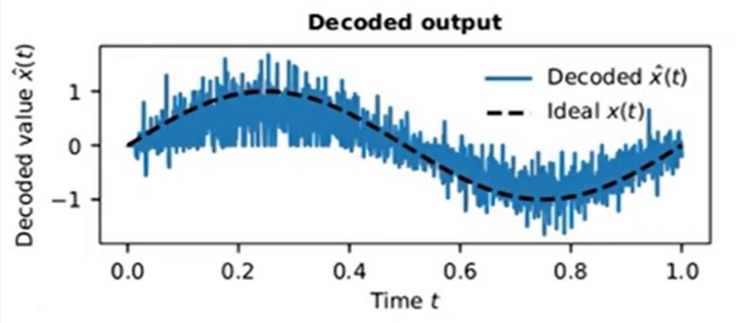
Encoding Temporal Varying Stimuli with LIF neurons



With postsynaptic filter!
tau = 5ms, n=1



Without postsynaptic filter



Encoding Temporal Varying Stimuli with LIF neurons

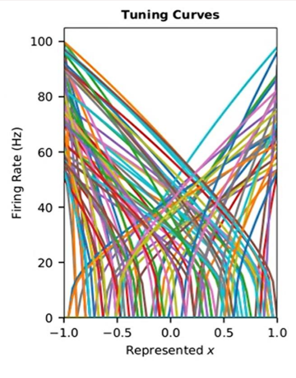
Summary

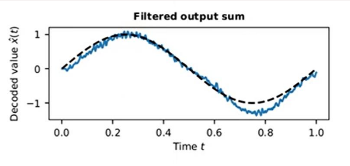
Encoding of a stimulus x(t) by a pool of neurons Ai

$$x(t) \xrightarrow{\text{input}} A_i \xrightarrow{\text{output}} a_i(x(t))$$

$$a_i(x(t)) = G_i[\alpha_i e_i x(t) + J_i^{bias}]$$

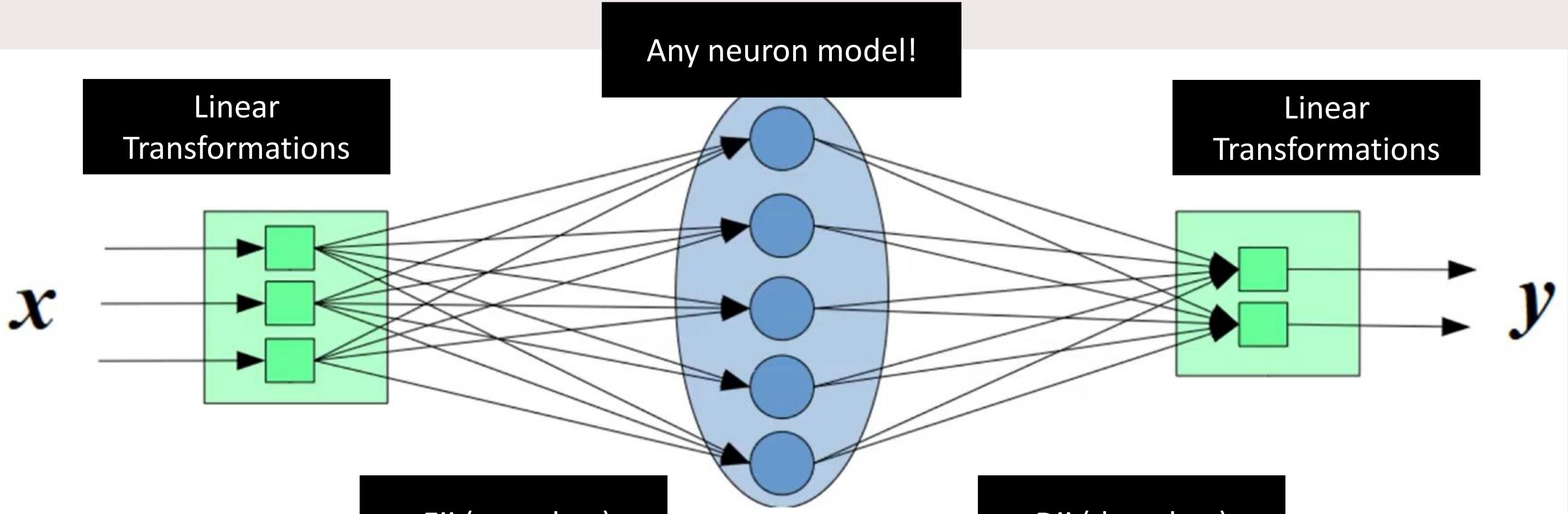
$$\hat{x}(t) = \sum_i a_i(x(t)) d_i^x, \quad d_i^x = \arg\min_x \langle (x - \hat{x})^2 \rangle$$





With postsynaptic filter!

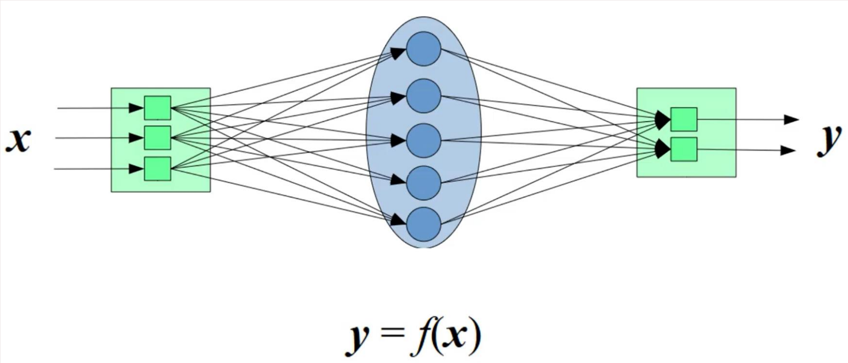
Transformation: function approximations with LIF neurons

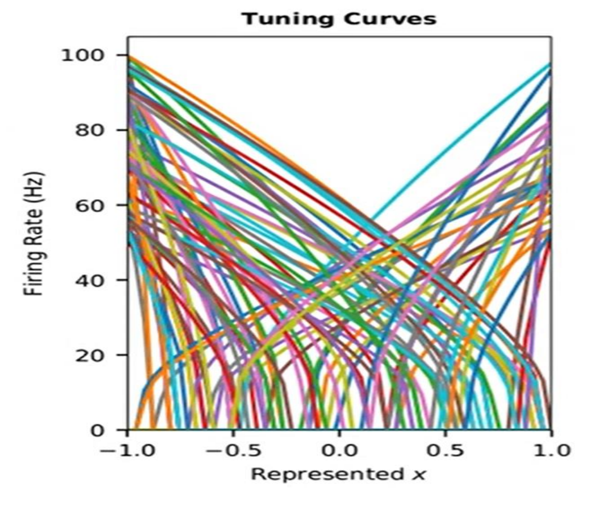


$$y = f(x)$$

Find **Eij** and **Dij** to produce the **desired f(x)**
Use any methods (back-prob, or randomly generate E and solve for D using linear regression – works for spiking)

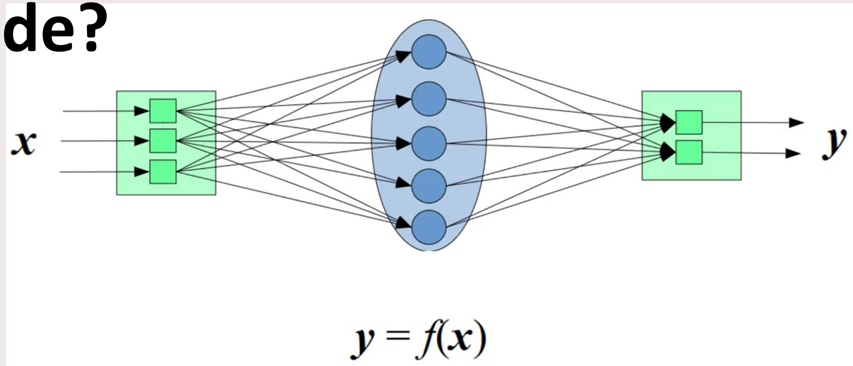
Encoding with NEF

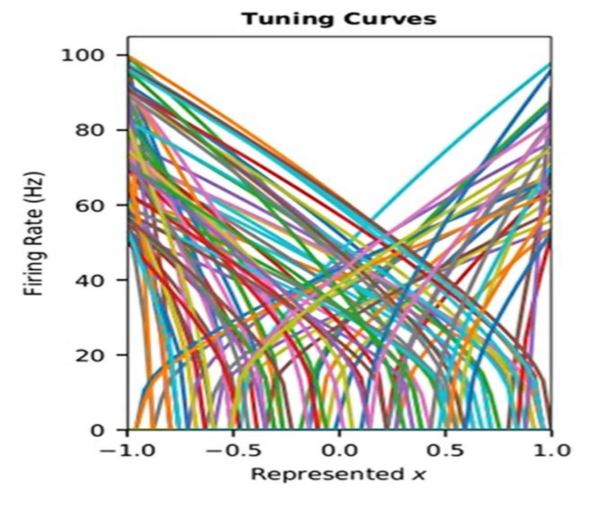




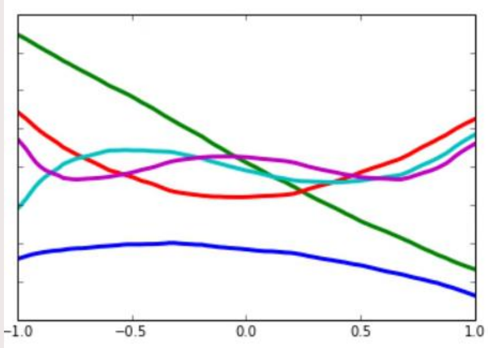
LIF neurons (mean-rate models)
Each neuron has its unique tuning curve
Given enough neurons we can approximate any functions

What set of functions can we encode?





Singular Value Decomposition

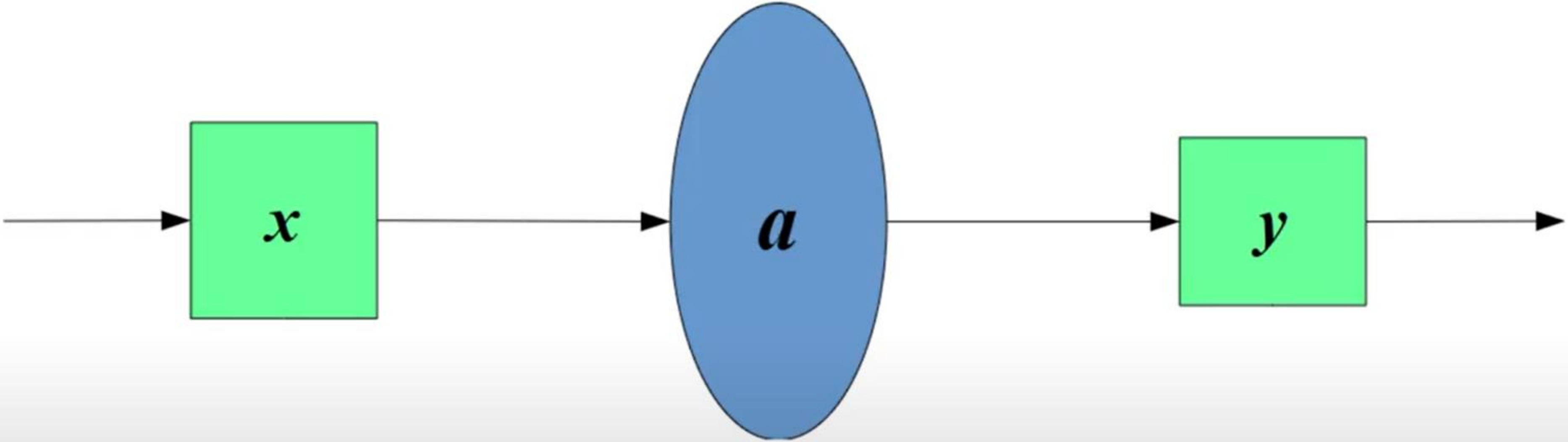


Example functions y=c (blue), y=-x (green), y=x^2 (red), y=x^3 (purple)

One layer of neurons is used to approximate **smooth functions** (low-degree polynomial)

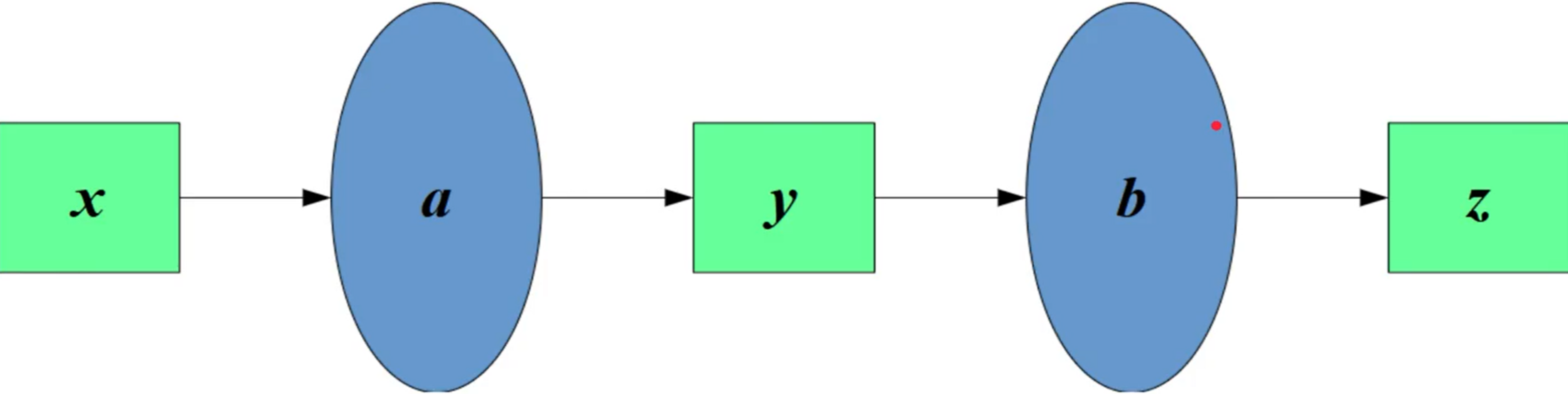
Neural Engineering Framework – NEF

What does it do? Function approximation



Neural Engineering Framework – NEF

What does it do? We can build larger systems made of function approximators

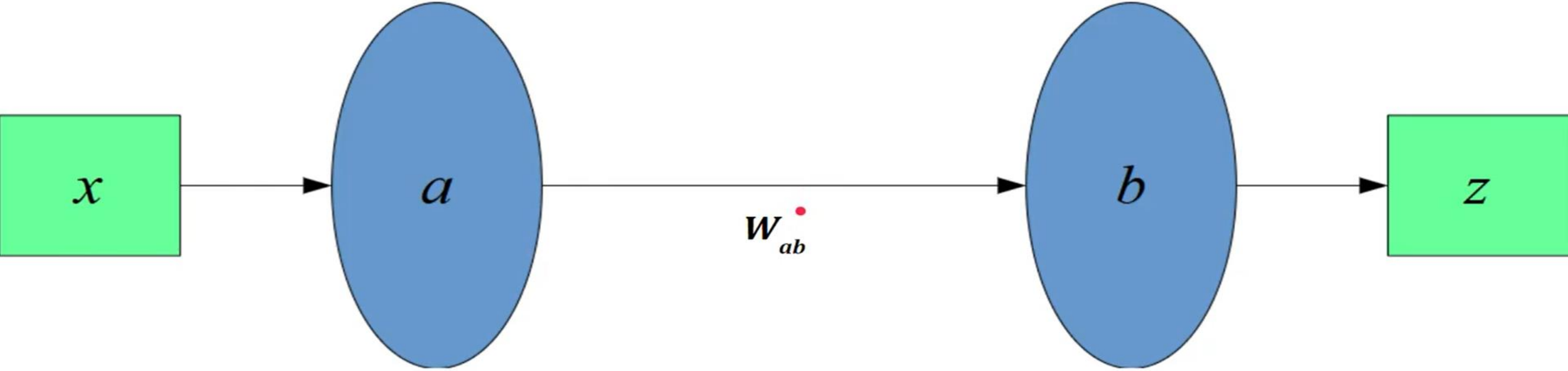


$$y = f(x) \qquad z = g(y)$$

$$z = g(f(x))$$

Neural Engineering Framework – NEF

What does it do? We can build larger systems made of function approximators

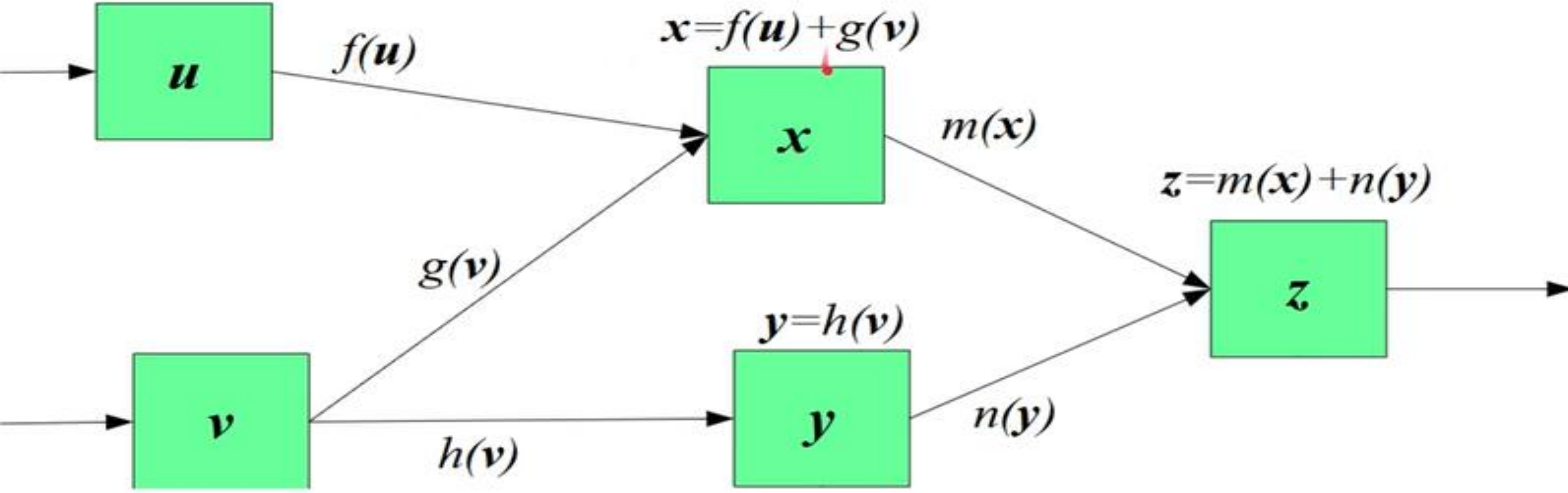


$$W_{ab} = E_b D_a$$

We can generate a set of connections weights by generating two neural networks and then merge them together.

Neural Engineering Framework – NEF

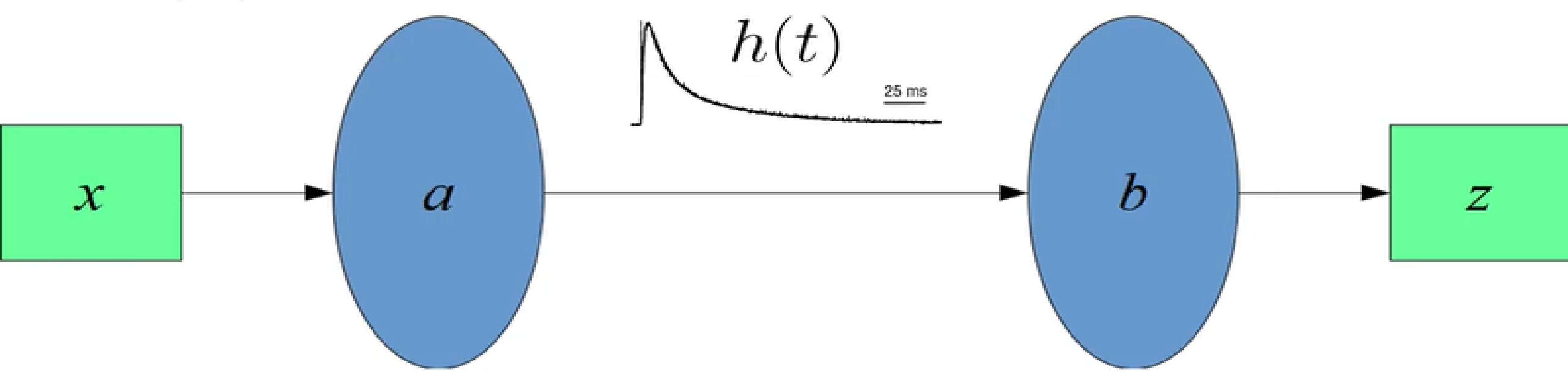
Programming neural network with functions approximators



We can also extent this simple principle in building up larger and more complex systems.

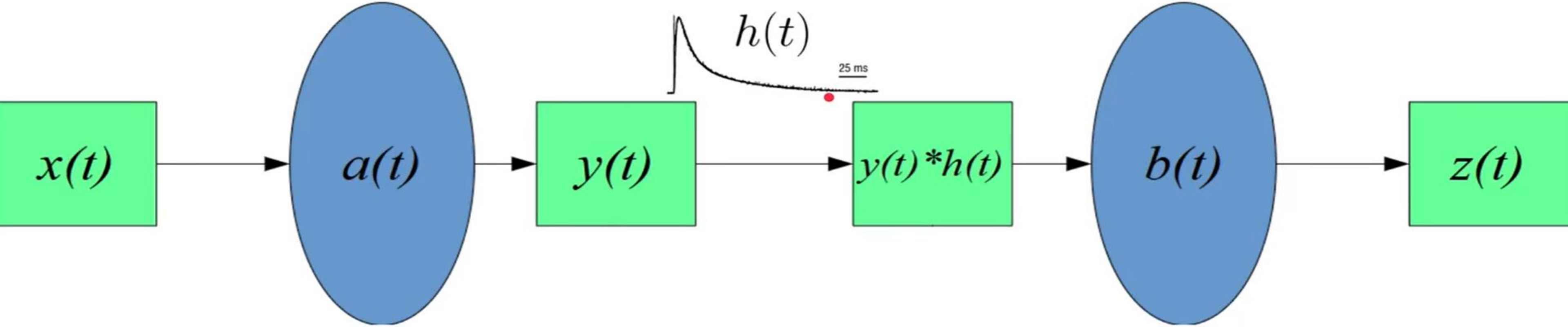
TU/e

Adding more bio realism: excitatory postsynaptic potential (EPSP)



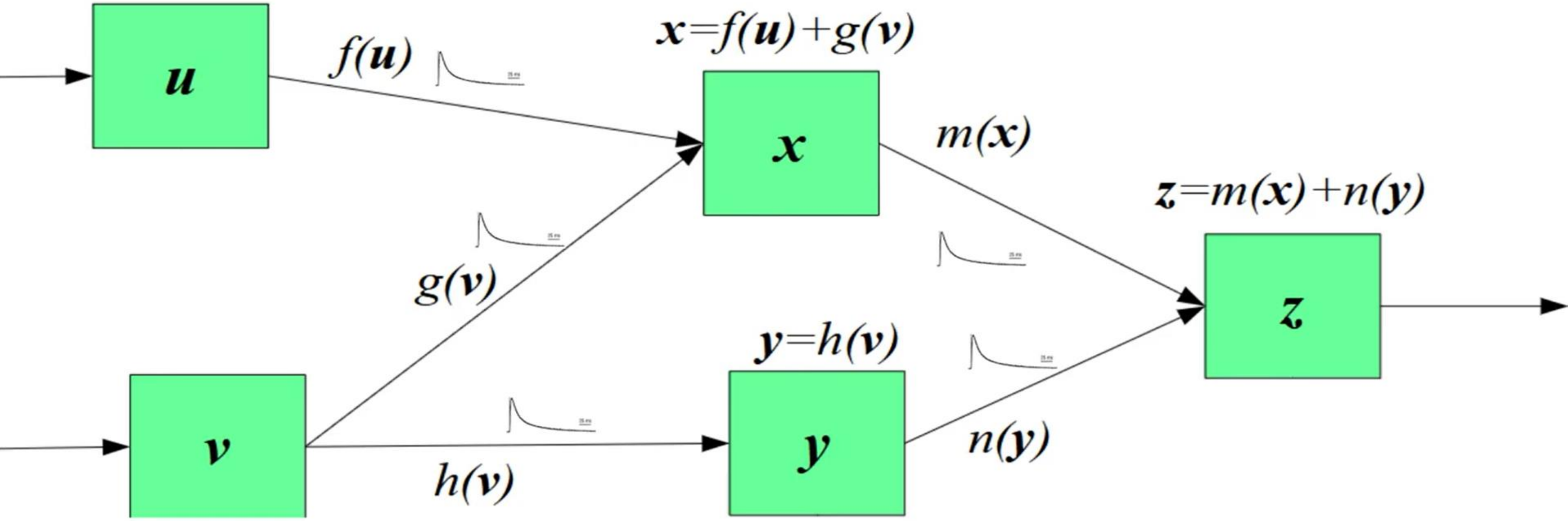
Convolving the spike train with h(t) (**excitatory postsynaptic potential**).
Synapses act to filter (*smooth*) the data value.

Adding more bio realism: excitatory postsynaptic potential (EPSP)



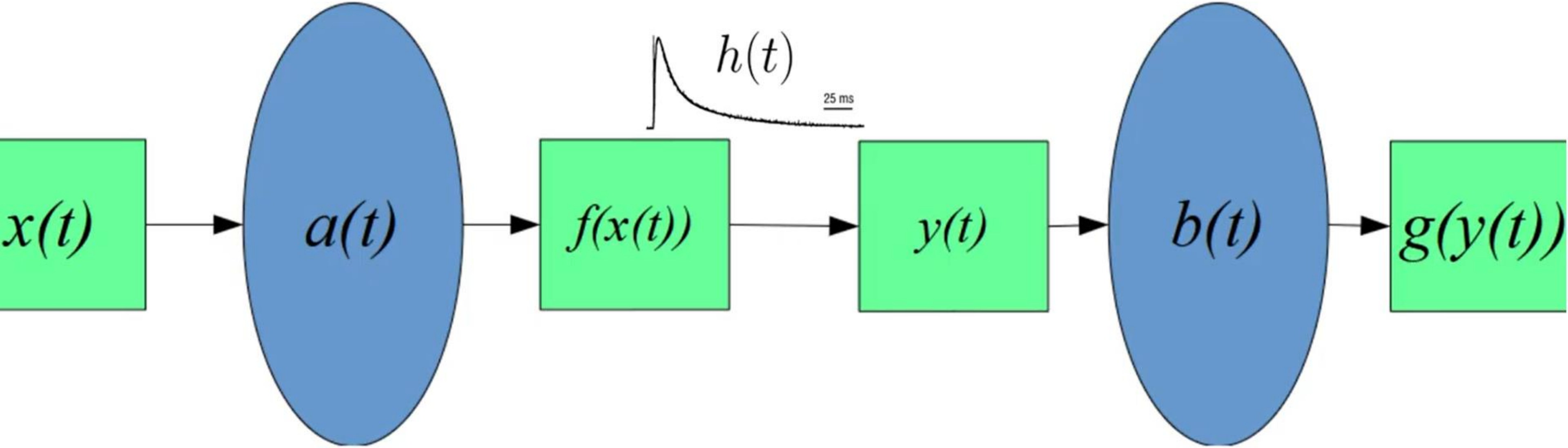
Even if synapses act on the spiking activity (a), it is mathematically equivalent to think as acting on the decoded value y, before passing it to the next group of neurons. Synapses act to filter (smooth) the data over time.

Adding more bio realism



Different neuron transmitters can have different properties, temporal, different filter operations.

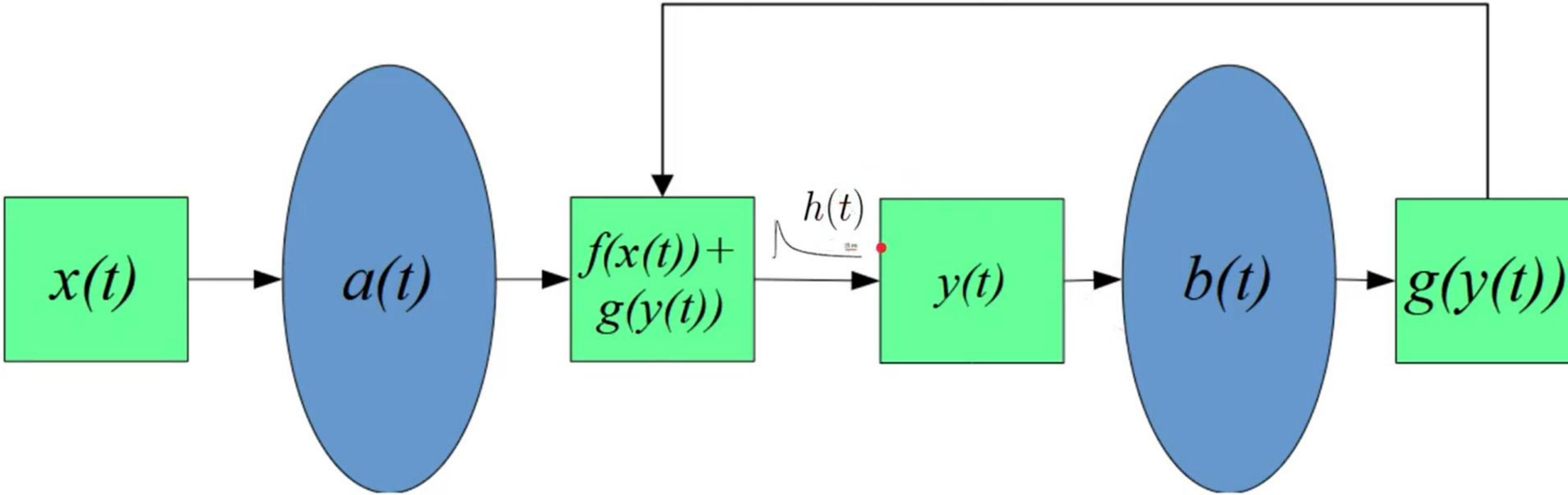
Recurrent Networks… next slide.



$$y(t) = h(t) * f(x(t))$$

TU/e

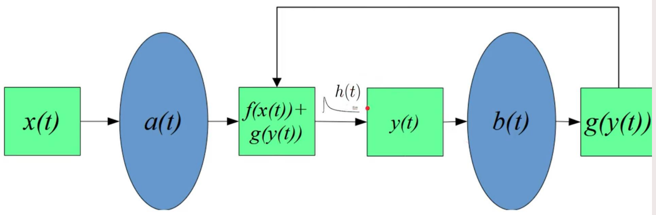
Recurrent Networks



$$y(t) = h(t) * (f(x(t)) + g(y(t)))$$

TU/e

Recurrent Networks



$$y(t) = h(t) * (f(x(t)) + g(y(t)))$$

Get rid of convolution with a **Laplace Transformation** and turns convolution into a multiplication. Note that **st** represents the time constant of the postsynaptic filter!

$$y(t) = h(t) * (f(x(t)) + g(y(t)))$$

$$Y = \frac{1}{1+s\tau}[G(s)+F(s)]$$

$$sY = \frac{G(s)-Y}{\tau} + \frac{F(s)}{\tau}$$

$$\frac{dy}{dt} = \frac{g(y)-y}{\tau} + \frac{f(x)}{\tau}$$

This will tell us how y will change when we set up our networks to approximate g(y) and f(x)

This means that we can approximate differential equations!

If you want this

$$\frac{dy}{dt} = a(y) + b(x)$$

Then find weights that do this

$$g(y) = \tau a(y) + y$$

$$f(x) = \tau b(x)$$

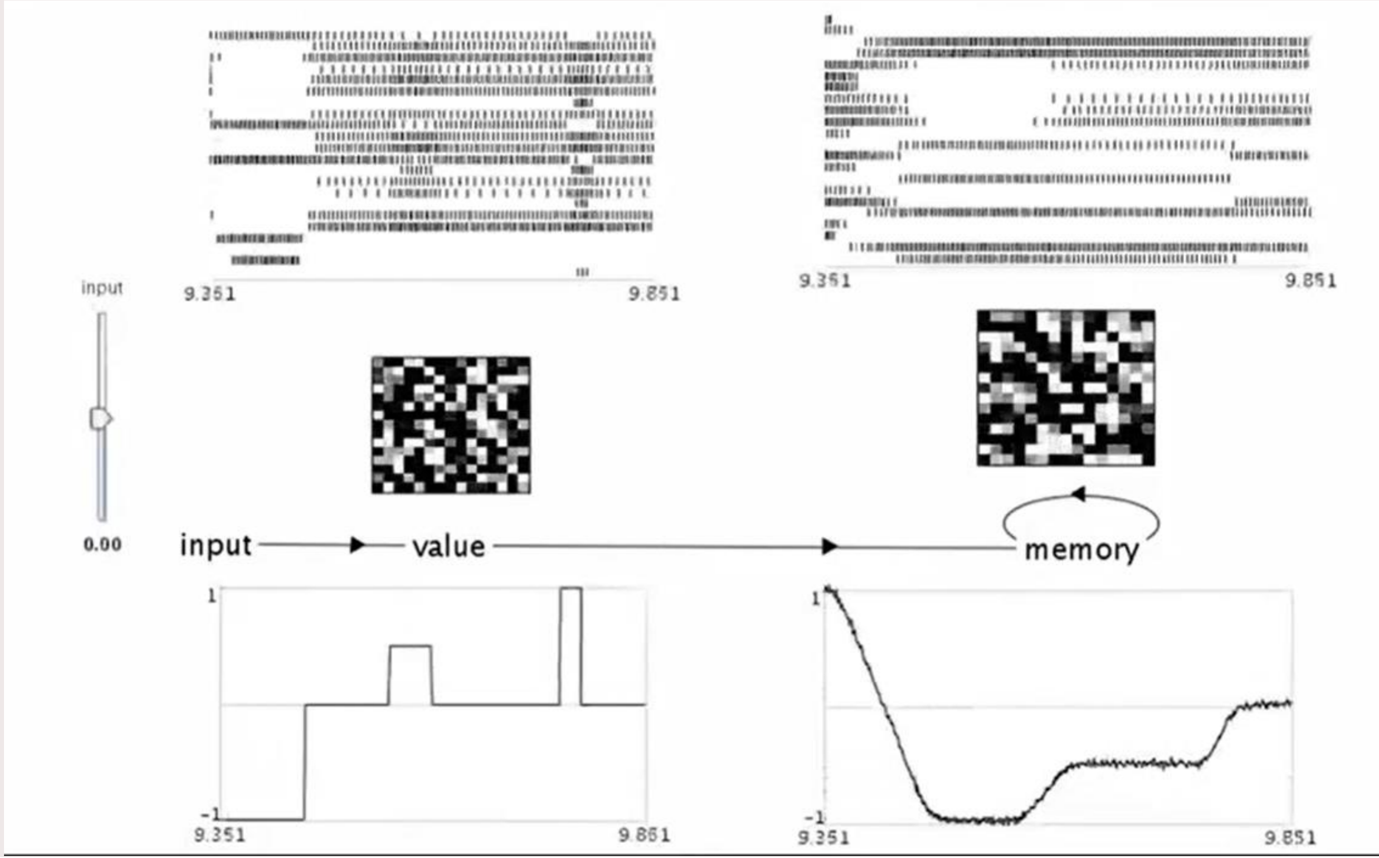
TU/e

Recurrent Networks

Integrator

$$\frac{dy}{dt} = x$$

$$g(y) = y$$
$$f(x) = \tau x$$



TU/e

Recurrent Networks

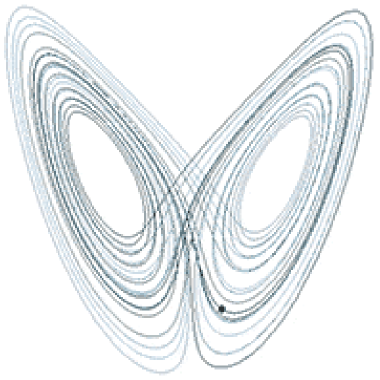
Dynamical System: Lorenz Chaotic Attractor

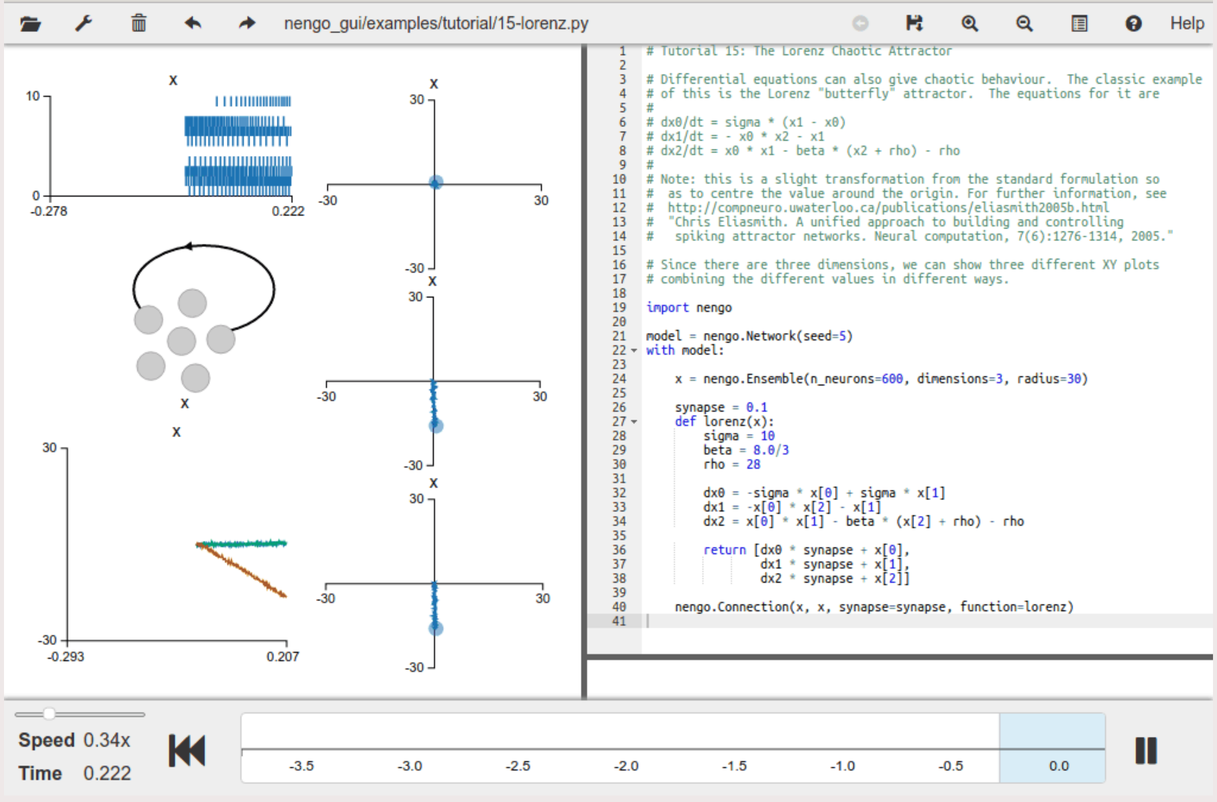
Novel techniques for building up SNN!!
NEF is a constructive approach.

$$\frac{dx}{dt} = \sigma(y - x),$$

$$\frac{dy}{dt} = x(\rho - z) - y,$$

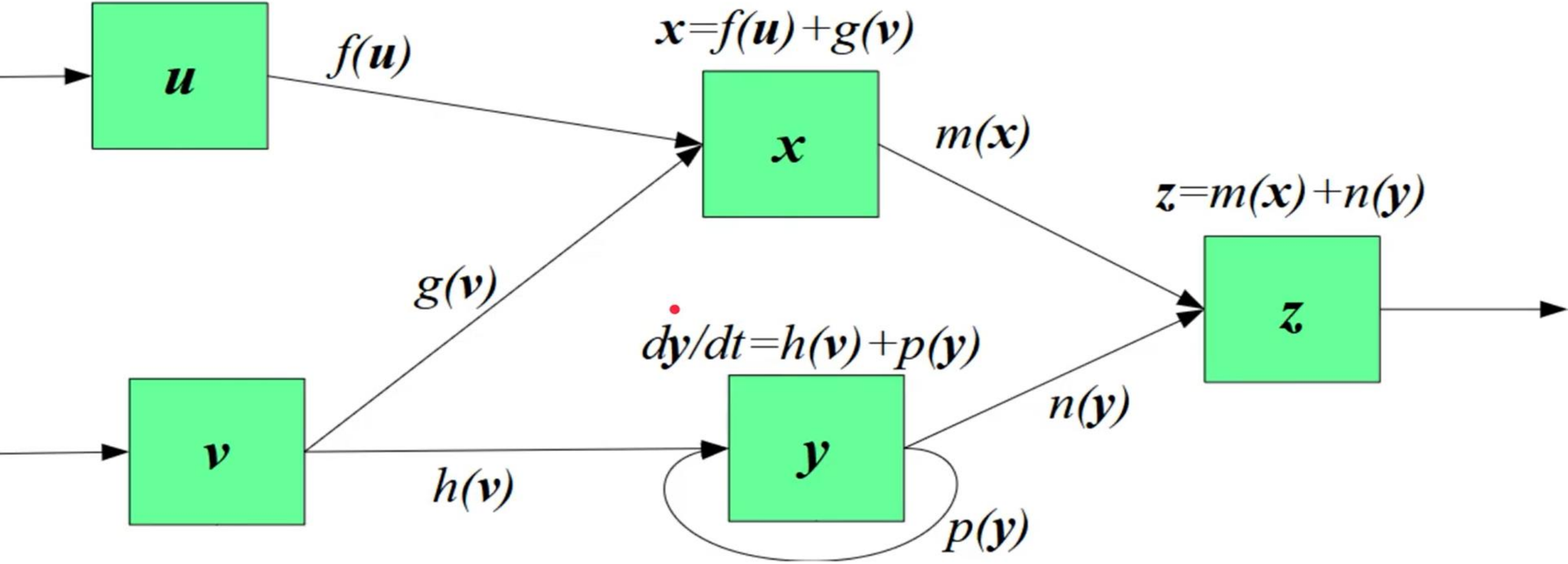
$$\frac{dz}{dt} = xy - \beta z.$$





Neural Engineering Framework

Novel techniques for building up SNN!!
NEF is a constructive approach.



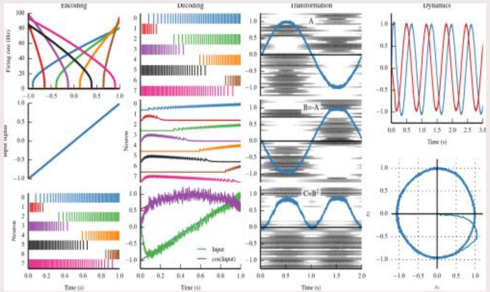
We can now train each network separate
and build up based on those!

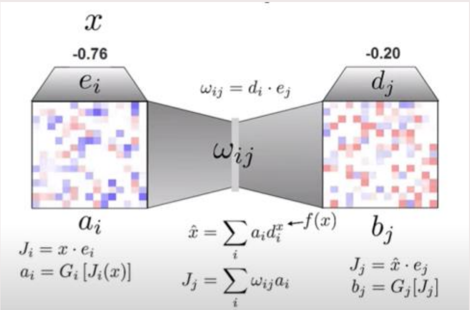
Neural Engineering Framework

Thanks to Chris Eliasmith and Terrence C Stewart!

Summary

- ***General approach to build NN***
  - ***Recurrent, feed forward***
  - ***Express the model in terms of vectors, functions, differential equations***
  - ***Choose neuron model and the level of biomimicry***
  - ***Generate the model on paper***
  - ***Evaluate performance (compare to biological data)***
- ***Function should be smooth***
  - ***Otherwise, the model will end up implementing a smoothed version of it***
- ***Programming with differential equation is hard***
  - ***No "FOR" loops and difficult to do "IF" statements***



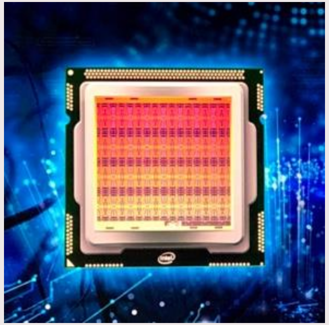


TU/e

Nengo.ai (Software)

Summary

- ***Pro:***
  - ***Can be used to build large scale models***
  - ***SPA enable abstractions and problem-solving skills***
  - ***Grounded on mathematics***
- ***Cons:***
  - ***Mainly based on mean-rate models***
  - ***Still limited performance compared to deep nets***



Intel Loihi

- Deep Learning derived models
- Neuromorphic Hardware (Loihi, FPGA, etc)
- Online learning
- Cognitive modelling with Semantic Pointer Architecture
- Startup (Applied Brain Research)
- Summer School in Waterloo (two weeks)

Semantic pointers?

Semantic pointers are:

- State-space representation for spiking neurons (encoding, e.g., the highest level of the visual hierarchy)
- Generated by compressor operators (vision, motor, vector symbolic architecture)
- Efficient for manipulation (because compressed, simple representations)
- Useful for large-scale, dynamics, or discrete continuous structured, anti semantic representations

Brain Anatomy and Functions

BRAIN ANATOMY & FUNCTIONS

Specific brain areas are responsible for particular functions. Here is a very high-level overview.