rijksuniversiteit groningen

Biologically plausible model of human learning in simple arithmetic tasks

Boris Radovanovic

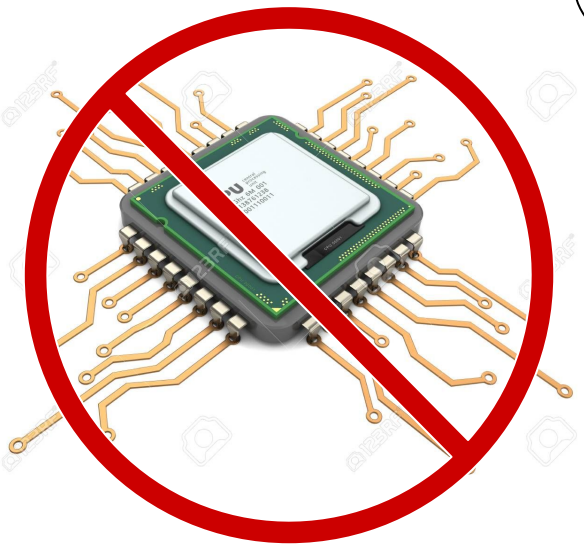
Supervised by Dr. Jelmer Borst

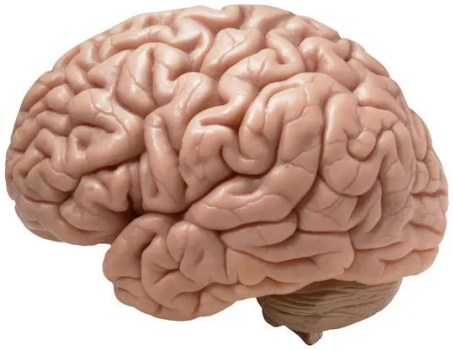
Alphabet arithmetic

A + 2 = C

Alphabet arithmetic

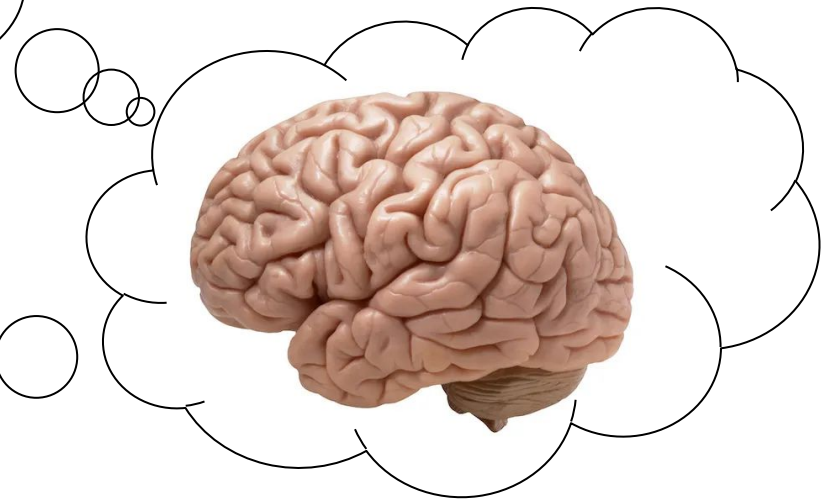
A + 2 = C



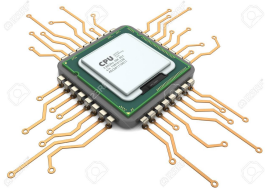


Alphabet arithmetic

A + 2 = C



Nengo



Learning

A + 2 : A → B → C

B + 4 : B → C → D → E → F

A + 2 : A → B → C

C + 3 : C → D → E → F

...

A + 2 : C

Learning

A + 2 : A → B → C

B + 4 : B → C → D → E → F

A + 2 : A → B → C

C + 3 : C → D → E → F

...

A + 2 : C

Counting
Procedural phase

???

Recall
Associative phase

Procedure for letter (L) + number (N)

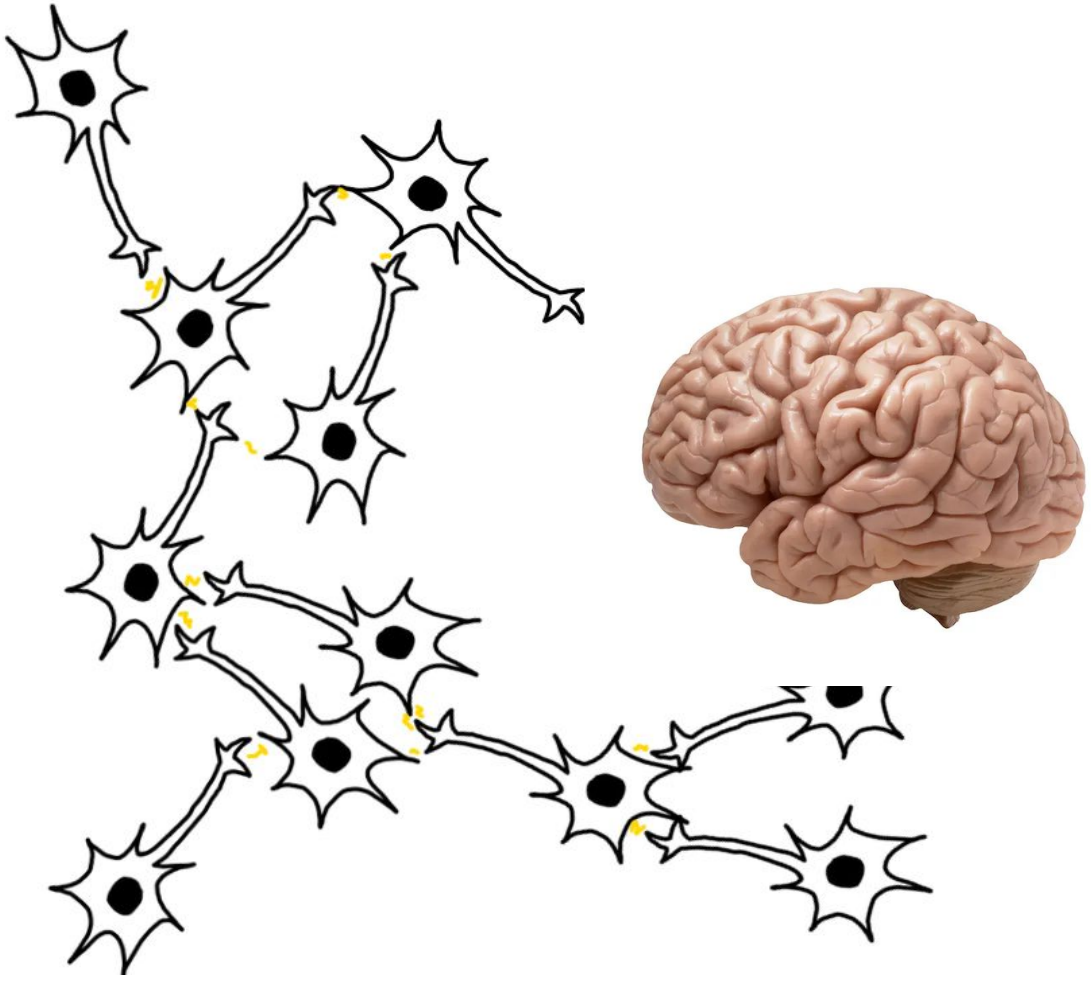
```
result ← recall L + N = ?

if recall was not successful
    result ← L
    count ← 0
    until count = N
        increment result
        increment count
    end
end

learn L + N = result
```

How do we actually do this with neurons?

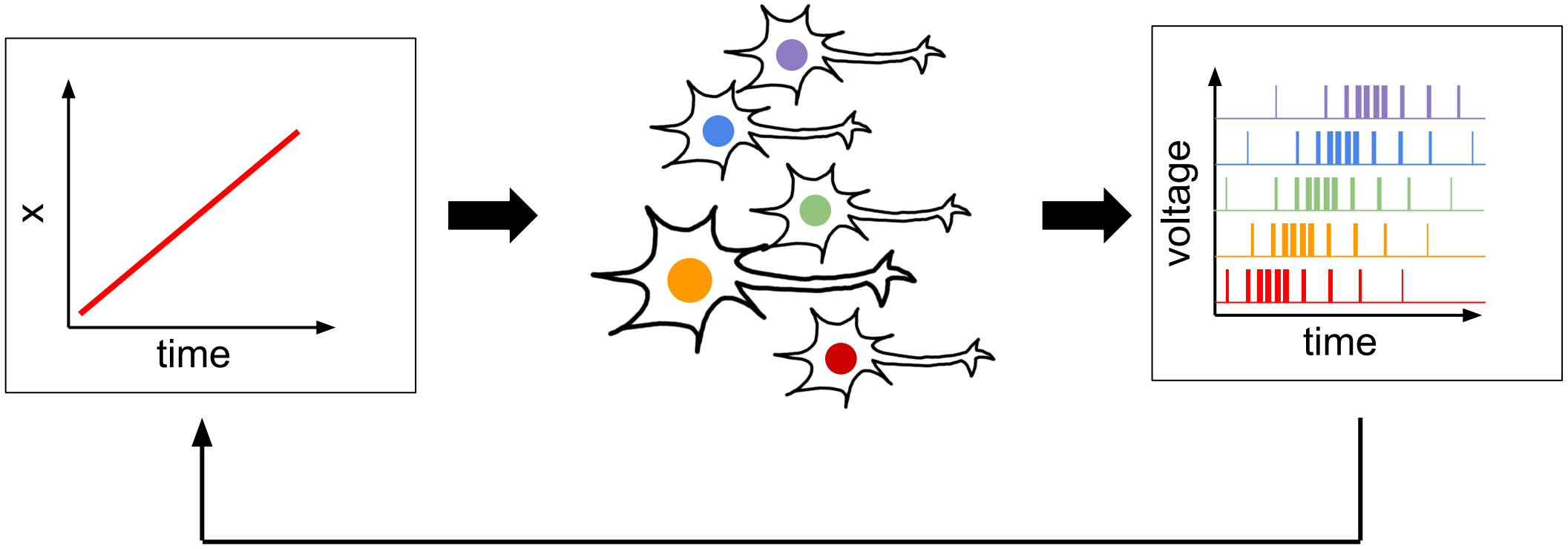
result ← **recall** L + N = ?

if recall was *not* successful
    result ← L
    count ← 0
    until count = N
        increment result
        increment count
    end
end

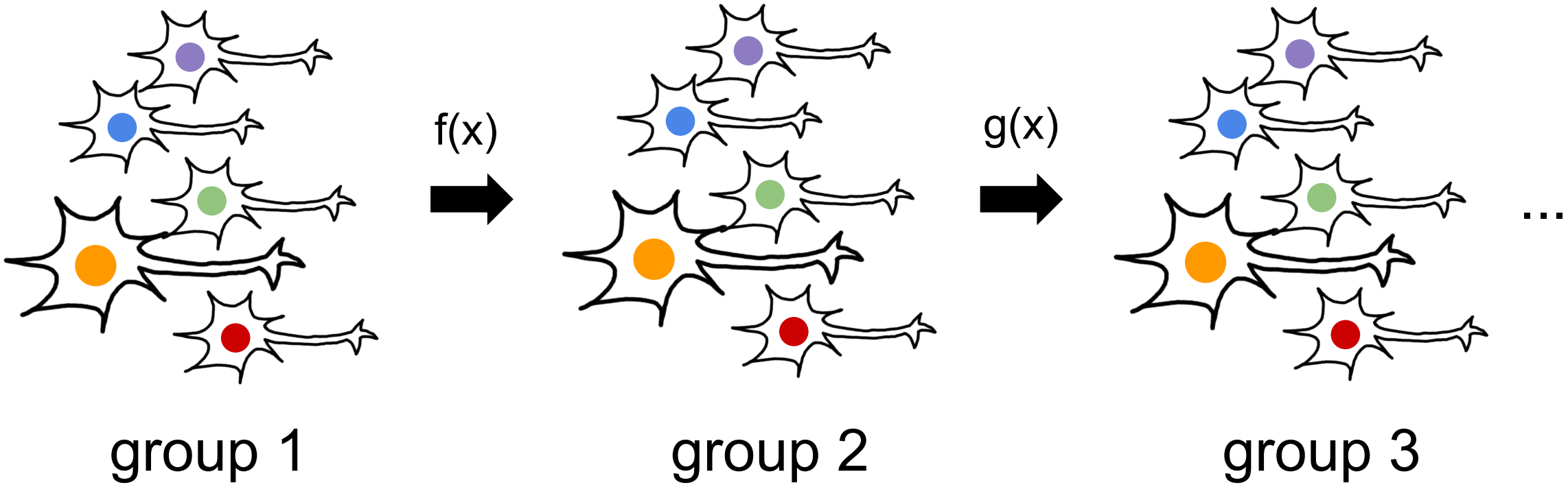
learn L + N = result



Spiking leaky integrate-and-fire (LIF) neurons



Spiking leaky integrate-and-fire (LIF) neurons



Biologically plausible learning rules

Voja

makes neurons selective on inputs over time.

PES

learns a function between neuron groups based on an error signal.

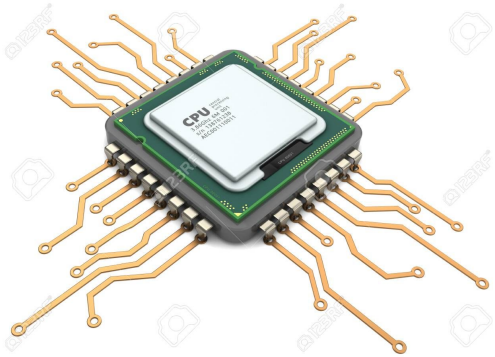
So....

```
result ← recall L + N = ?

if recall was not successful
    result ← L
    count ← 0
    until count = N
        increment result
        increment count
    end
end

learn L + N = result
```

looks simple, but is *really* hard

Models already exist, but are too brittle.

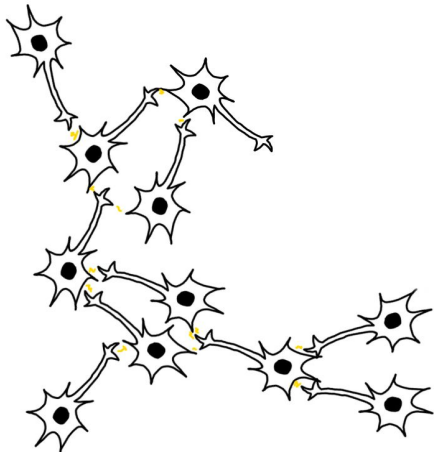
Counter intuitive behavior

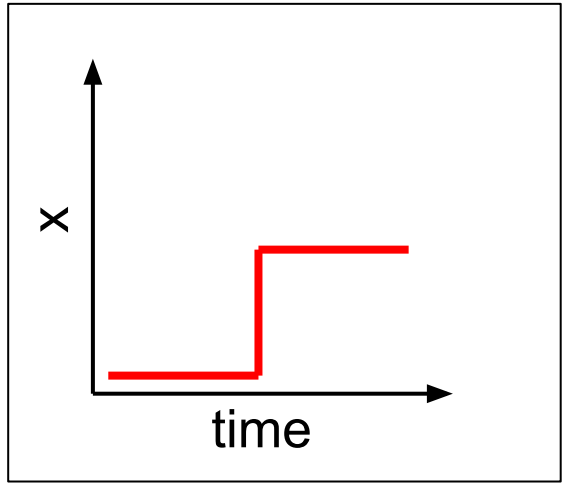


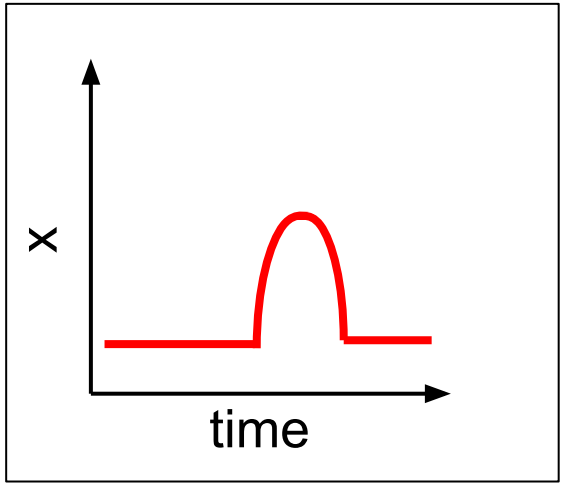
var x ← 0

x ← 1

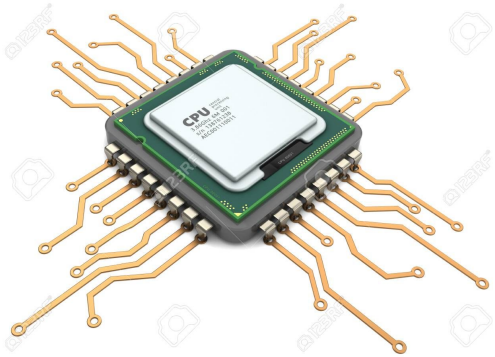
...



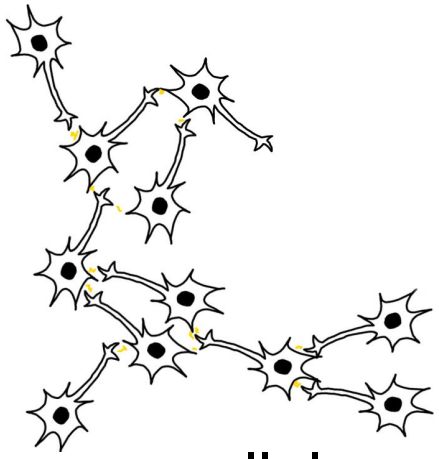




Everything always runs in parallel



synchronous

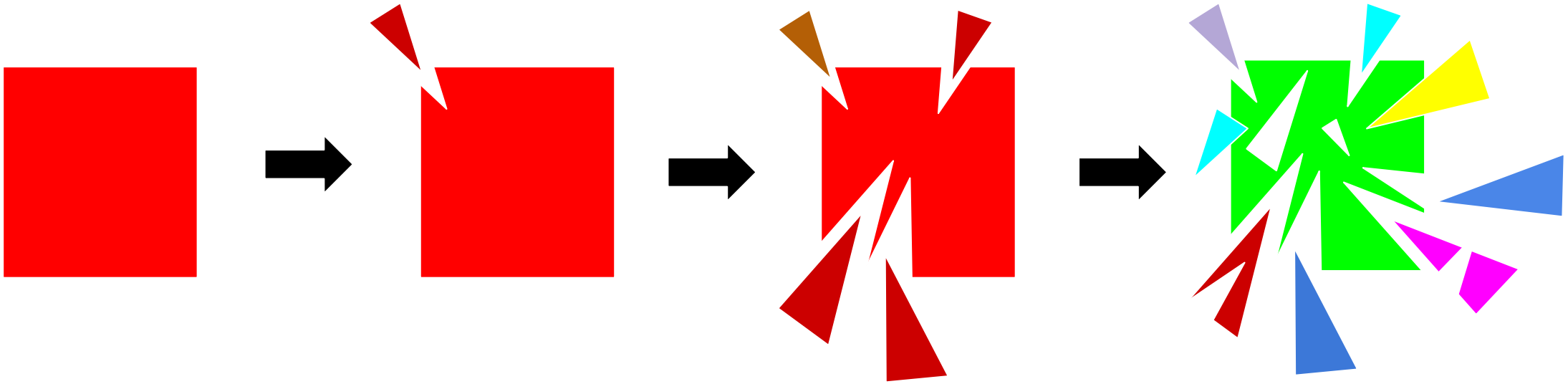


parallel

```
break invariants
    do stuff
restore invariants
```

```
while maintaining invariants
    do stuff
end
```

Noise accumulates over time



Hyperparameters are everywhere

0.3 · dot(state, Run) + **1.2** · load_result - inc_result →
assoc_result = **2.5** · result_mem
total_assoc = **2.5** · total_mem
result_gate = Close
total_gate = Close
state_gate = Close
count_gate = Close
load_comp_a = **0.75** · One
load_comp_b = **0.75** · One
inc_comp_a = gen_inc_assoc
inc_comp_b = count_res
gen_inc_assoc = **2.5** · result_mem

Aubin et al. (2016)

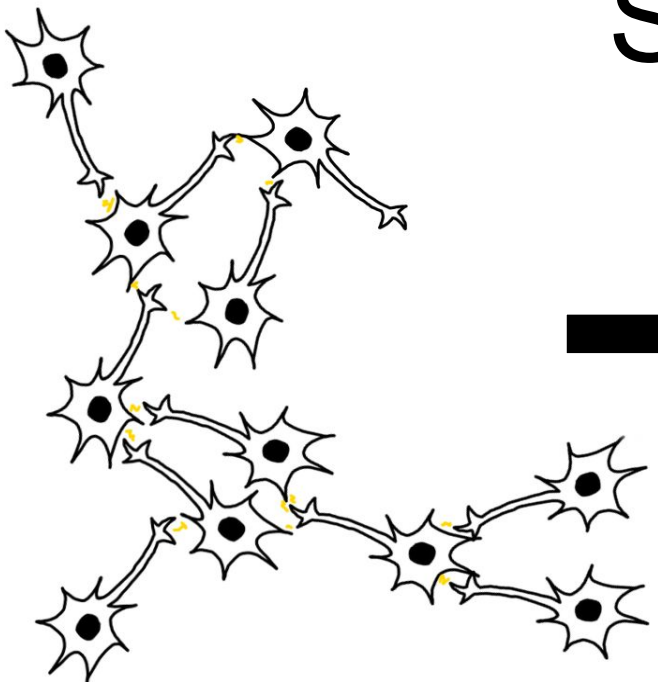
5 hyperparameters in **1** rule

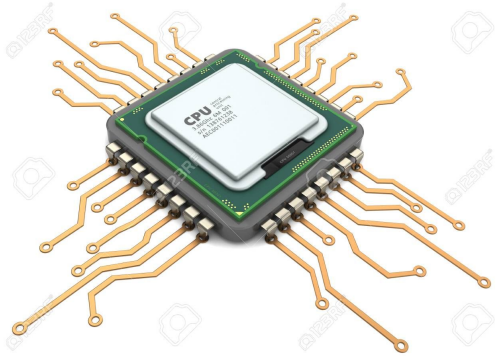
Probably around 40 in the whole model

Changing one parameter breaks **everything**

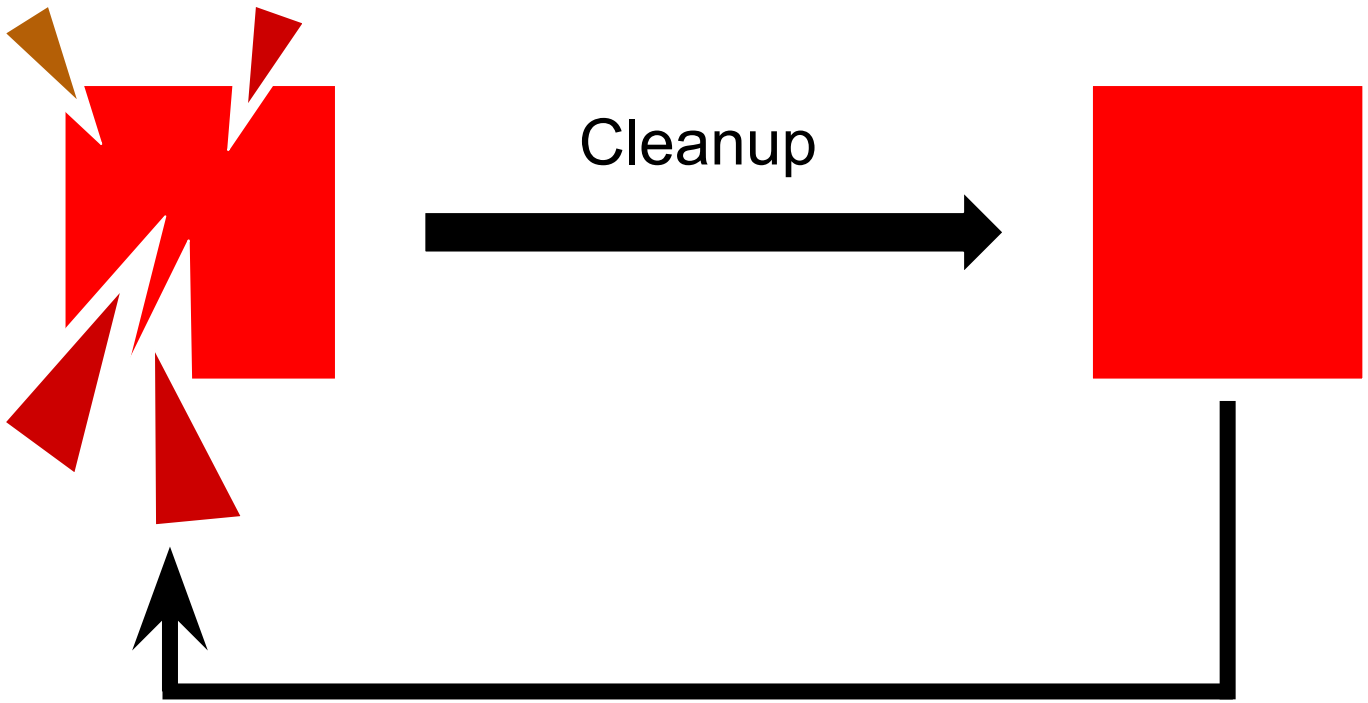
Enough complaining..

Solution?



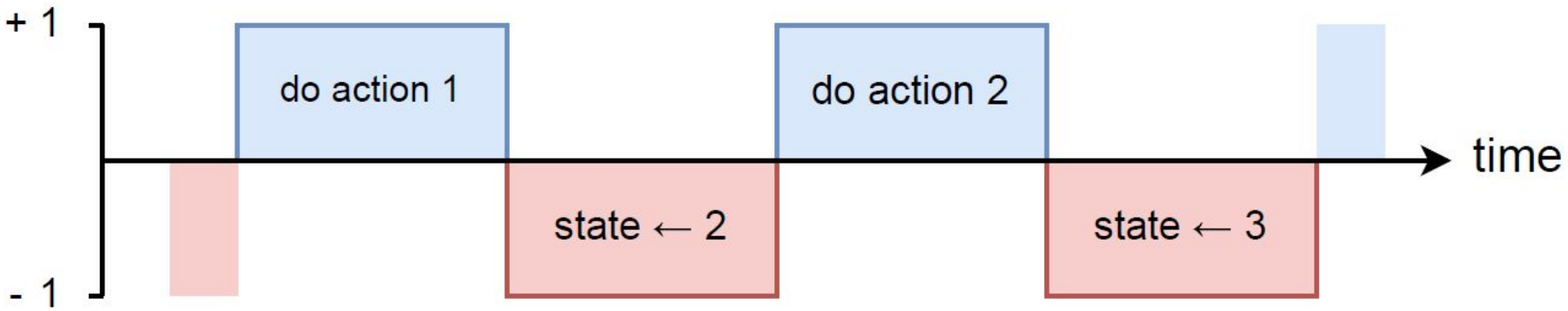


Solution 1: Cleanup memory



Retains information permanently!

Solution 2: Clock

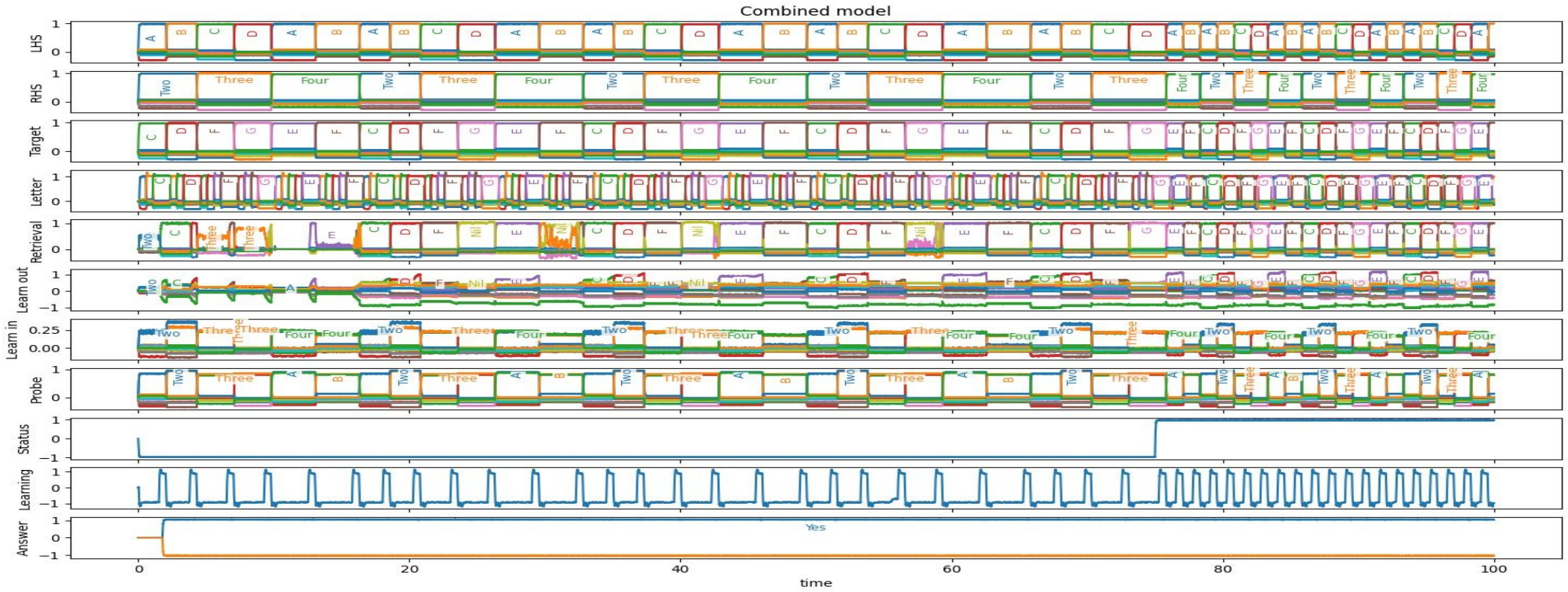


Perform actions on high pulse.
Change states on low pulse.

Eliminated most problems...

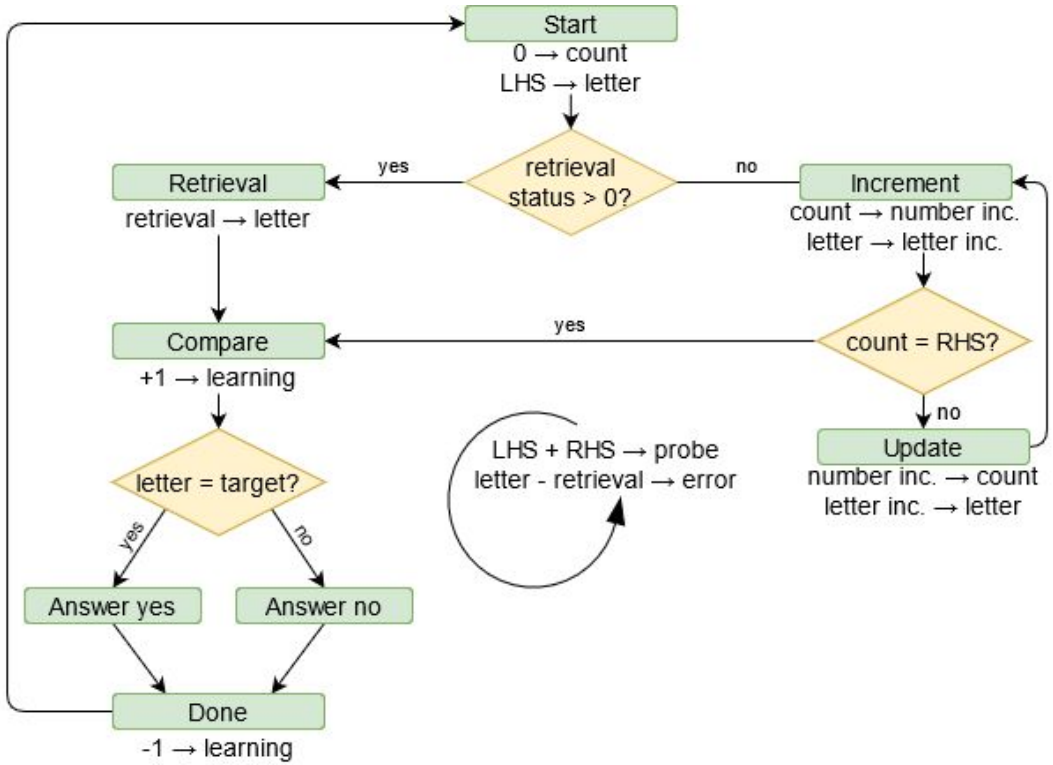
But model still didn’t work.

After a month of tinkering and rewriting, 2 days ago, the model suddenly worked!



¯_(ツ)_/¯

Model is much simpler and more robust



Code matches diagram almost 1-to-1

Thanks for listening!