Abdal AlKilany and Goodman 2026

Neuromodulation enhances the capability and efficiency of spiking neural networks



Dan Goodman

Imperial College London

Department of Electrical and Electronic Engineering

IMPERIAL

neural-reckoning.org

Outline

1

Philosophy

Why do computational modelling?

2

Background

What is a spiking neural network and how do we train them?

3

Research

Neuromodulation.

Philosophical intro: what can we learn from neural models?

Worth asking this question because everyone sees it differently

I'm interested in understanding "intelligence", not just the brain (multiply realisable)

This opens up many questions that might not obviously interest neuroscientists

But they should!

We can't understand the brain's unique solution to "intelligence" without understanding a bit about the space of possible alternative solutions - not just about fitting or predicting data

My limited approach to neural models:

pure theory, comparison across realisations, no fitting/prediction

Baseline neuron model + Bio-inspired feature = New computational capacity

Biological neuron

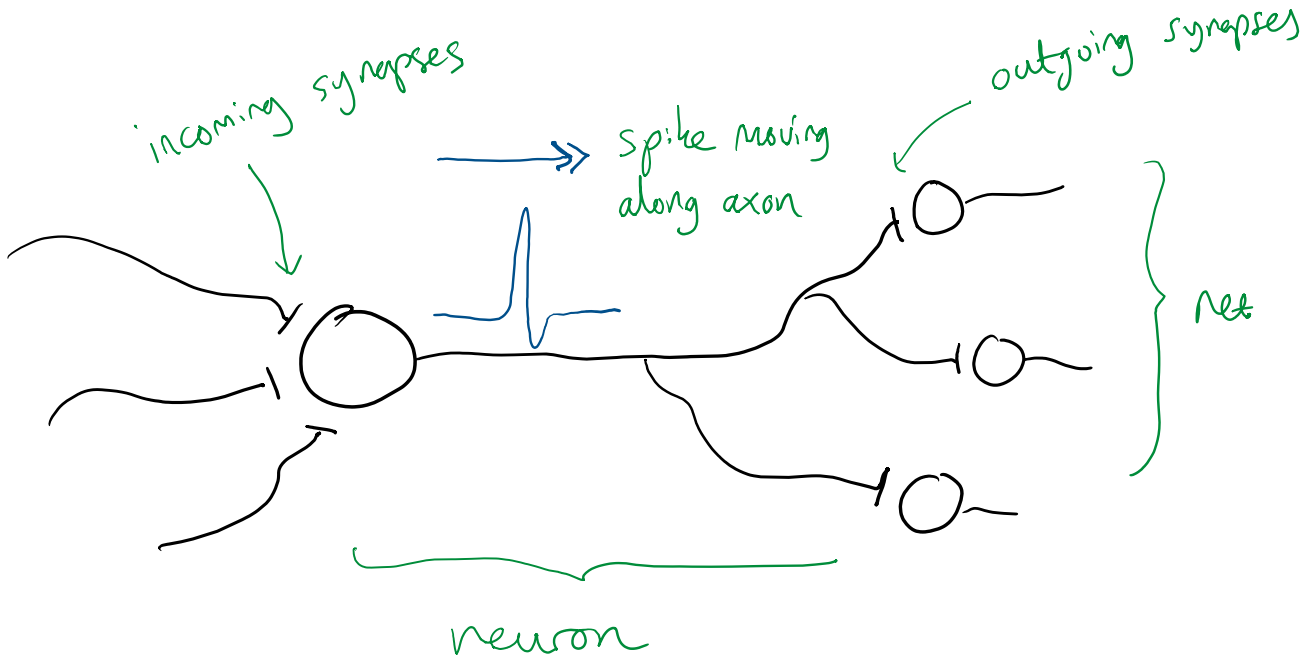
Experimentally validated

Known to exist in brain, mechanism and function not fully understood

Hypothesis for mechanism-function link in brain

Pedagogical intro: What is an SNN and how do we train them?

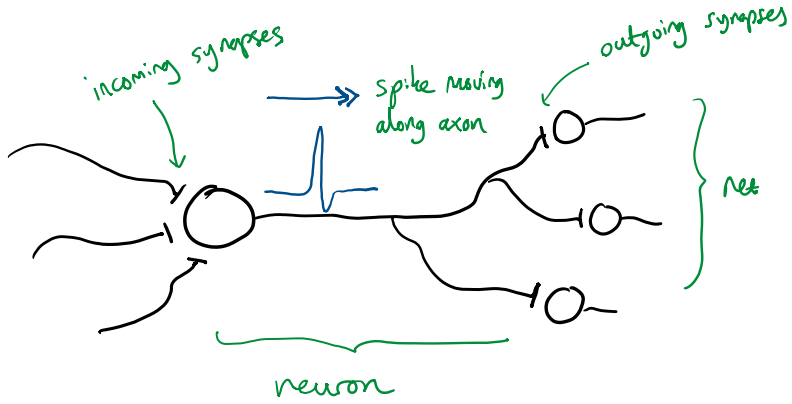
I'll assume everyone knows what a neuron, synapse and spike are.



A "spiking neural network" is a model of this.

As compared to an "artificial neural network" (ANN) or just "neural network": no spikes.

Pedagogical intro: What is an SNN and how do we train them?



Why do many computational neuroscientists use ANNs with no spikes?

Simple to work with, e.g. $\boldsymbol{y} = \sigma(W\boldsymbol{x} + \boldsymbol{b})$ is just a linear operation and sigmoidal function.

Easy to train these with gradient descent because they are differentiable.

Gradient descent on a large network lets you learn incredible things (machine learning, AI).

SNNs are not differentiable, so we can't apply gradient descent in the same way.

But they are how the brain works, so we would like to.

There's a trick to this we use called **surrogate gradient descent** (and other methods).

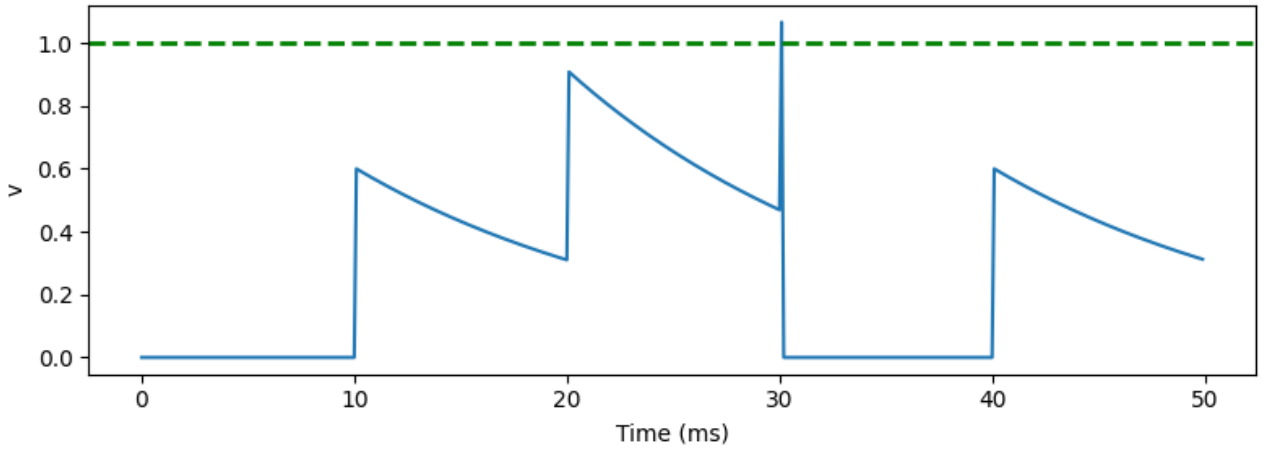
Pedagogical intro: What is an SNN and how do we train them?

A simple model of a spiking neuron:

Membrane potential v

When no spikes, decays to rest: $\tau \frac{\mathrm{d}v}{\mathrm{d}t} = -v$

When a spike arrives, increases: $v \leftarrow v + w$



When it crosses a threshold $v > 1$ fires a spike and resets $v \leftarrow 0$

Not differentiable!

Threshold operation $S = H(v - 1)$ where H is the non-differentiable Heaviside function

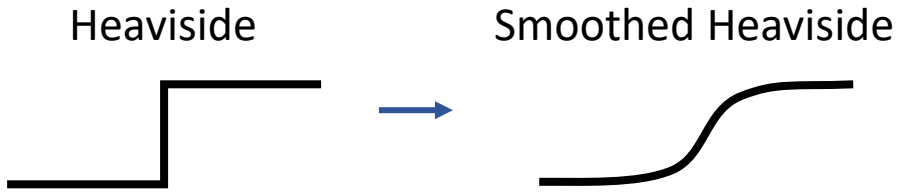
Solution: pretend it is differentiable

Heaviside function $H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$

Replace H with smoothed version in derivative

Weirdly: this works!

(It's an ML technique called "straight-through estimators")



Also easy: 12 lines of PyTorch code

The neuromodulation model

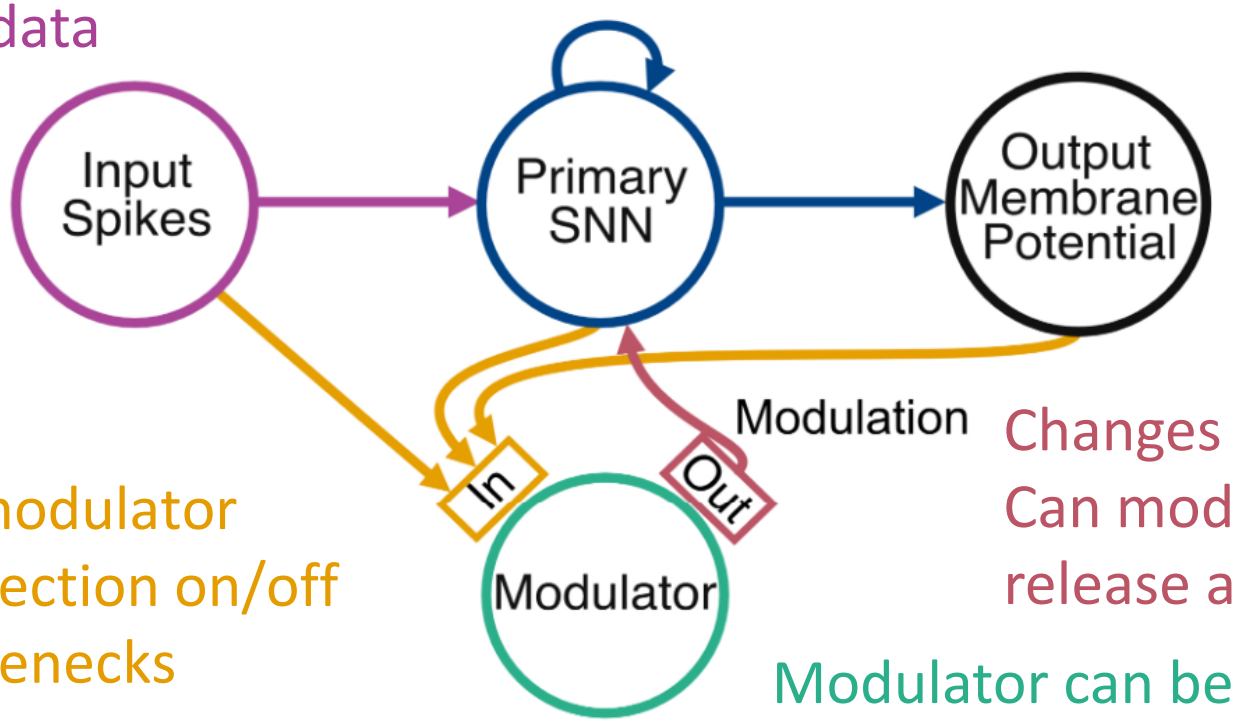
Following the formula on the philosophical intro slide, the approach will be this:

Baseline LIF model + Abstract model of neuromodulation = Lots of nice things (you'll see)

Recurrent SNN trained with surrogate GD

Spike-coded sensory data
Auditory: SHD, SSC
Visual: DVS





→ Linear readout
Classification task

Spikes as input to modulator
Can turn each connection on/off
Can introduce bottlenecks
Can compress

Changes parameters of LIF neurons
Can model neurotransmitter release and diffusion

Modulator can be either:

- MLP (an ANN)
- SNN

The equations slide (feel free to doze if it's not your thing)

Simplest version

Output of the ANN modulator network is x targeting parameter p

Substitution rule: replace $p \leftarrow x$

Addition rule: replace $p \leftarrow p + x$

Output of the SNN modulator network is a spike with weight δp targeting parameter p

SNN addition rule: replace $p \leftarrow p + \delta p$

In every case, one output for each target neuron and parameter, or group these effects

Most general version (won't go into detail)

$$\underset{\mathbf{x}_{ig}}{\text{Released quantity}} \xrightarrow[\text{sparse matrix } S]{\text{Spatial dispersion}} \underset{\mathbf{q}_{jg}}{\text{Received quantity}} \xrightarrow[\text{MLP } f \text{ or identity}]{\text{Neuromodulator interactions}} \underset{\hat{\mathbf{p}}_{jk}}{\text{Target parameter}} \xrightarrow[\text{factor } \lambda]{\text{Temporal diffusion}} \underset{\mathbf{p}_{jk}}{\text{Neuron parameter}}$$

$$i = \text{release site}, \quad j = \text{neuron index}, \quad g = \text{neuromodulator type}, \quad k = \text{parameter index}$$

Same sound. Different response.

Neuron parameter modulation

λ = 1.0

REAL-TIME / 1.4 SECONDS

SPOKEN DIGIT · ENGLISH 01 / 05

“six”



0.00 s

LISTENING

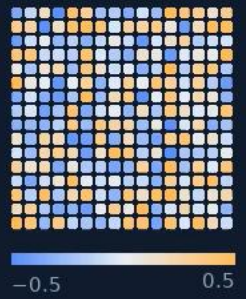
UNMODULATED

Original baseline · fixed neuron parameters

RESET VOLTAGE



256 NEURONS



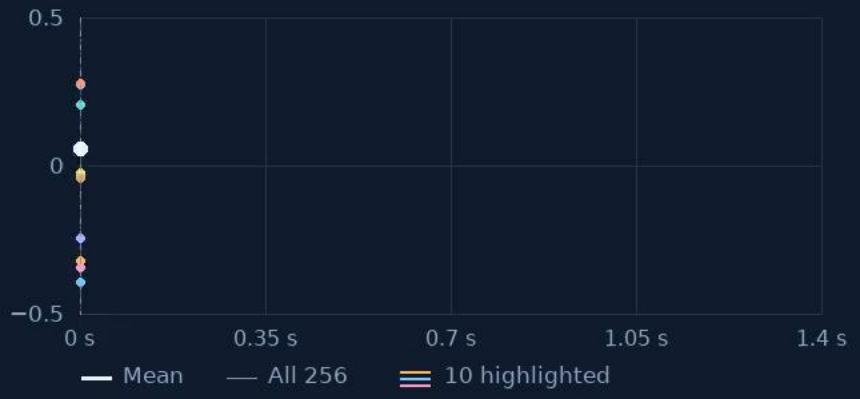
HIDDEN SPIKES

CURRENT PREDICTION **Listening...**

MODULATED

Adaptive neuron parameters · λ = 1.0

RESET VOLTAGE



256 NEURONS



HIDDEN SPIKES

CURRENT PREDICTION **Listening...**

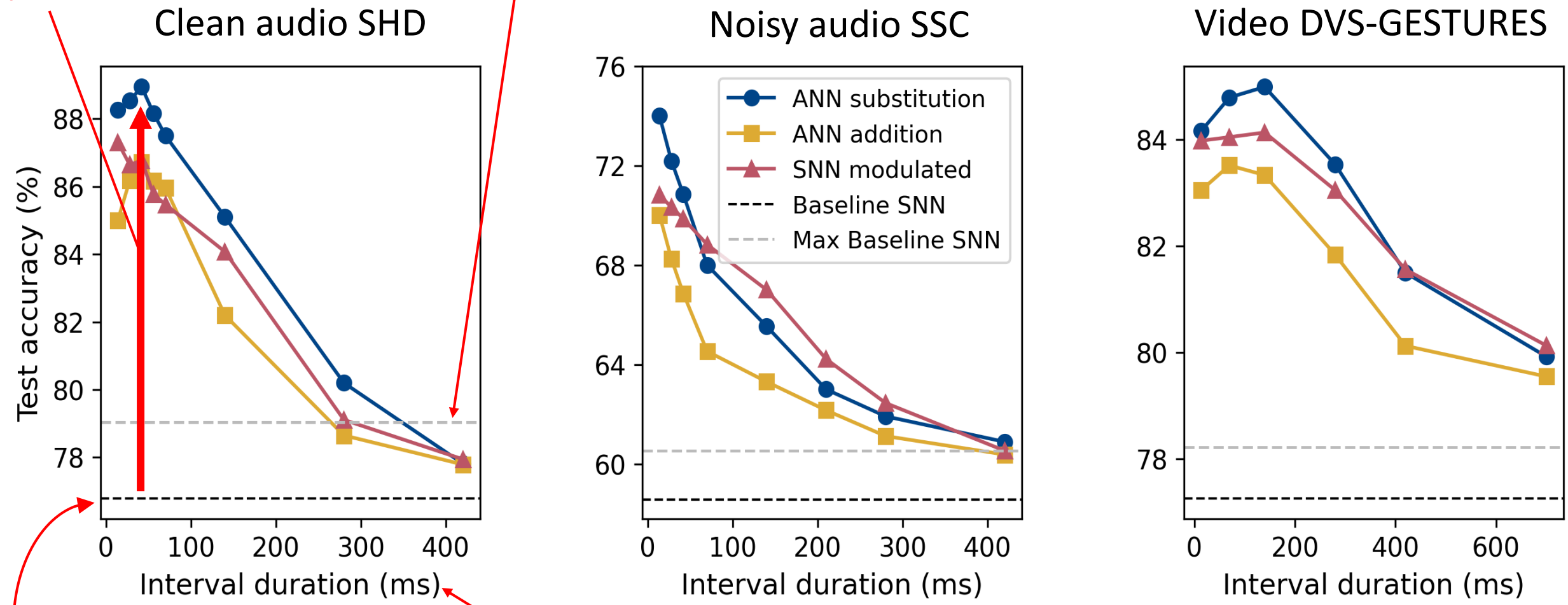
SHD test audio · clean re-encoding · selected comparisons

1× playback · 1 s hold + 1 s fade

It works!

Modulation
Improvement ~12%

Optimal non-modulated SNN performance (large network)



Baseline (unmodulated SNN)

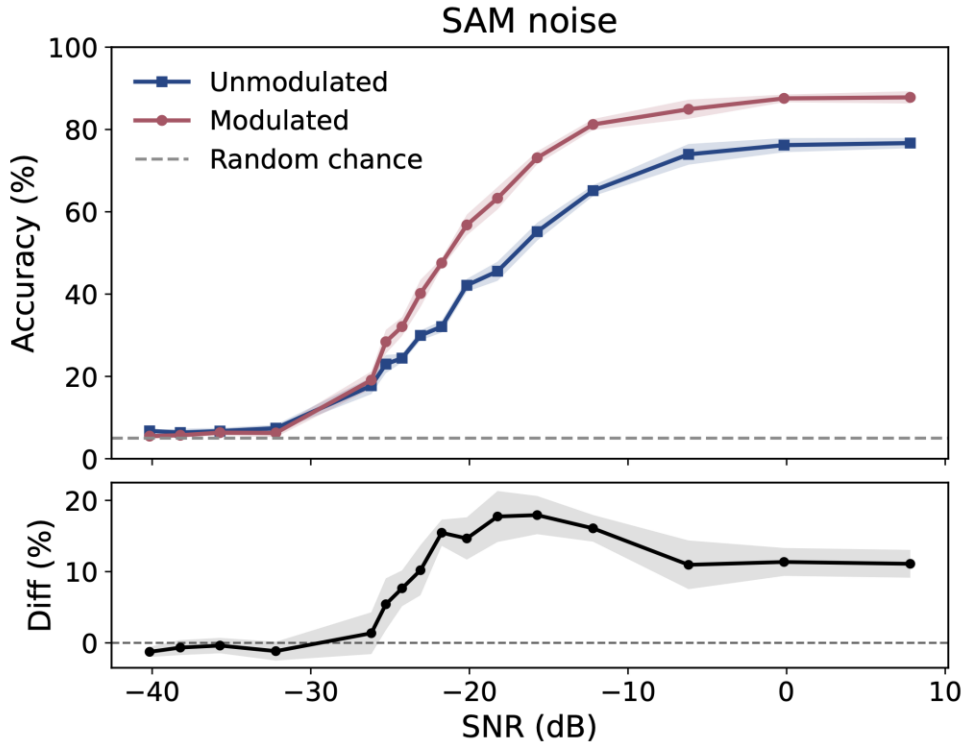
How often the parameters are updated

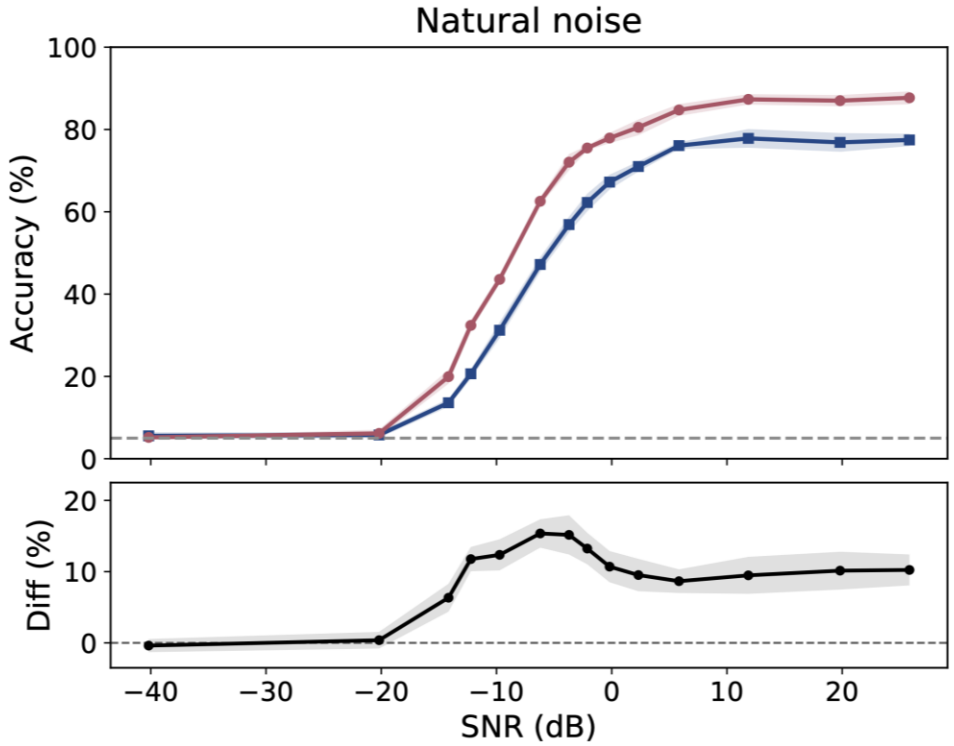
More challenging version: speech in noise

Sinusoidally amplitude modulated (SAM) noise

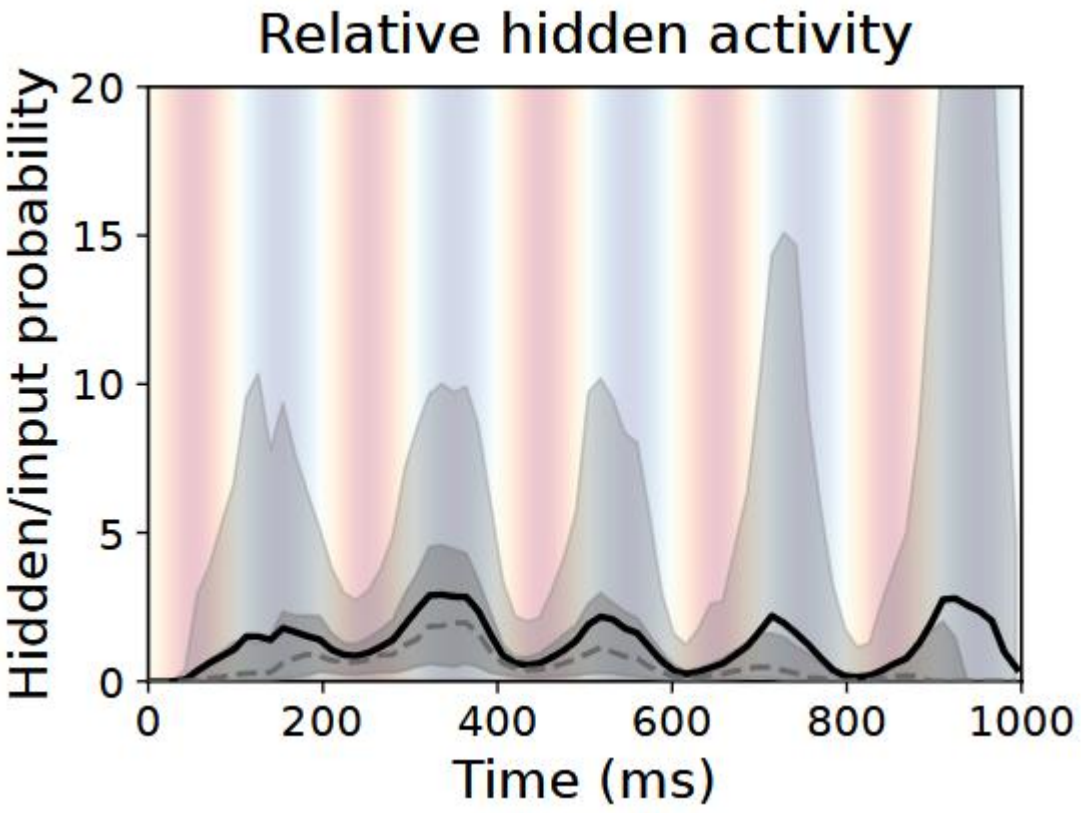


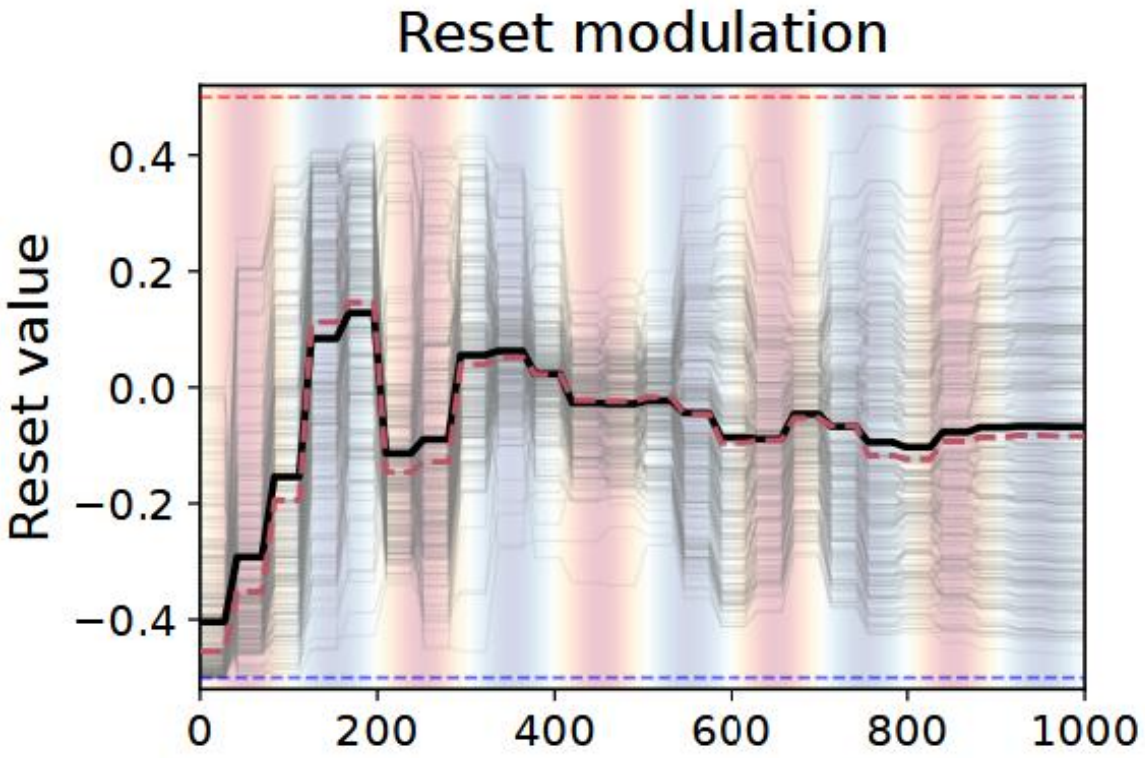




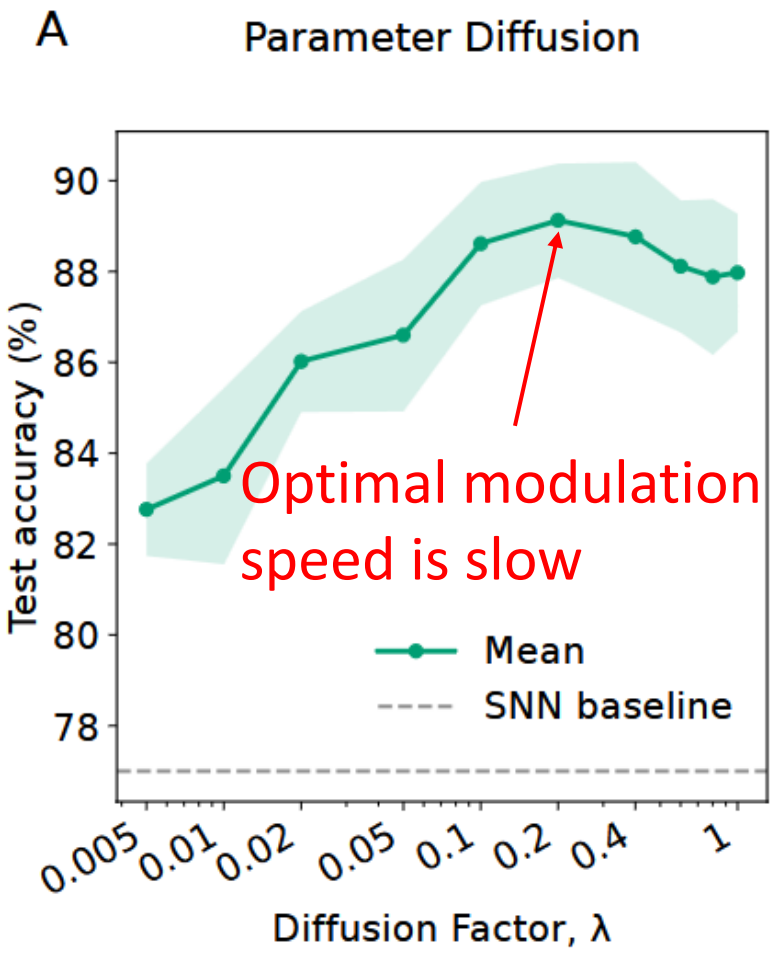


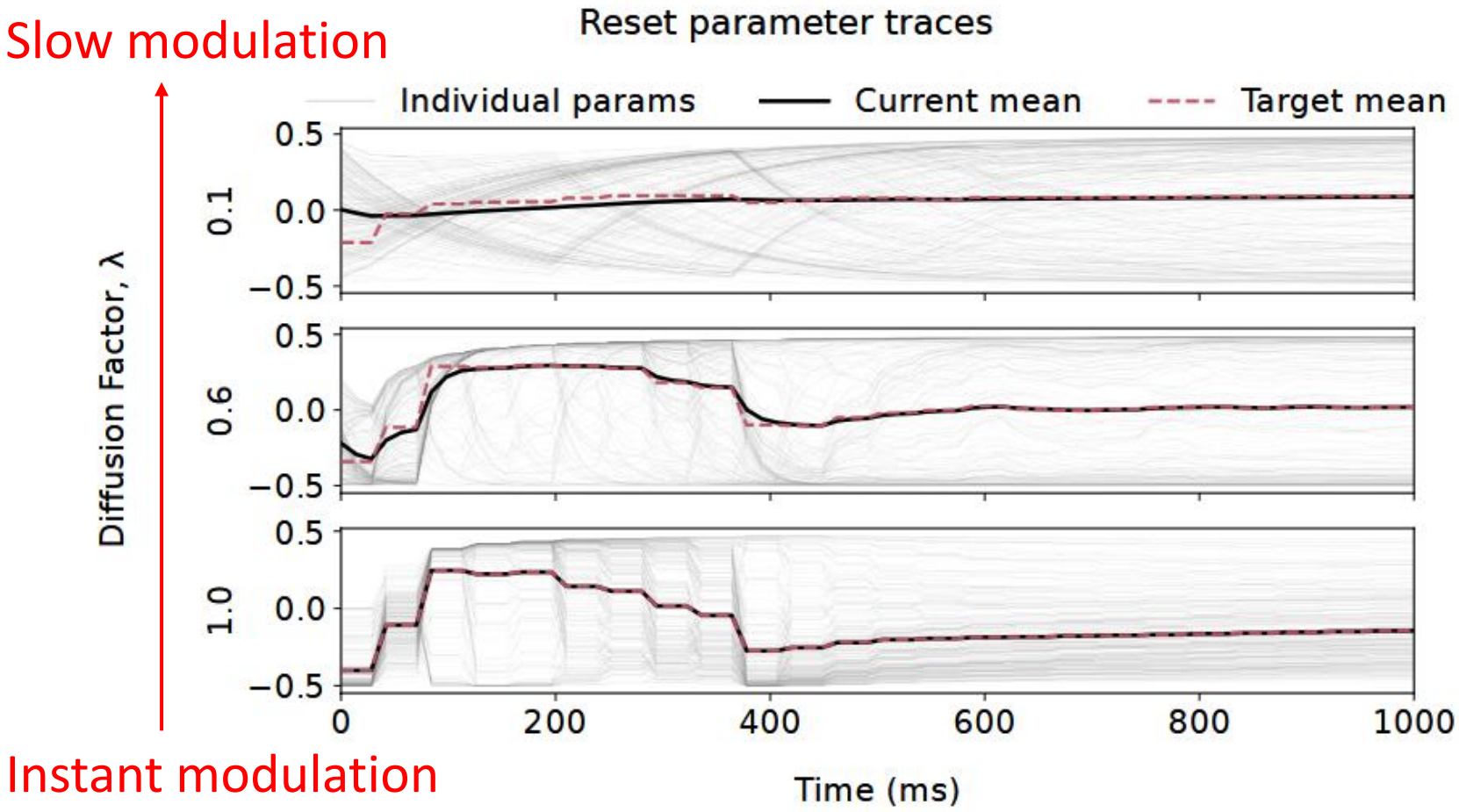
Learns dynamic gain control strategy (“listening in the dips”)



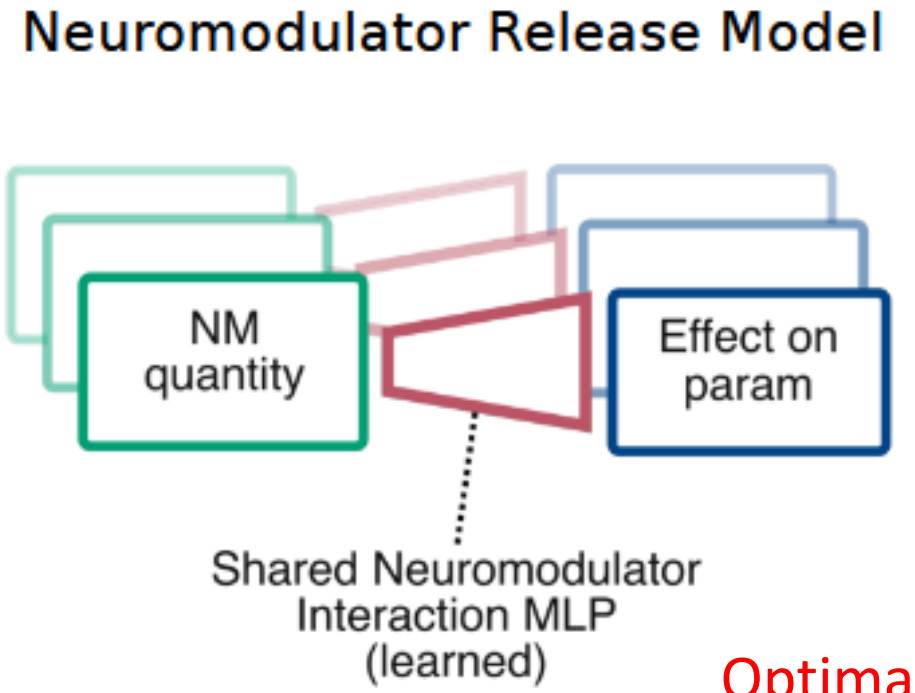


Temporal diffusion can help performance

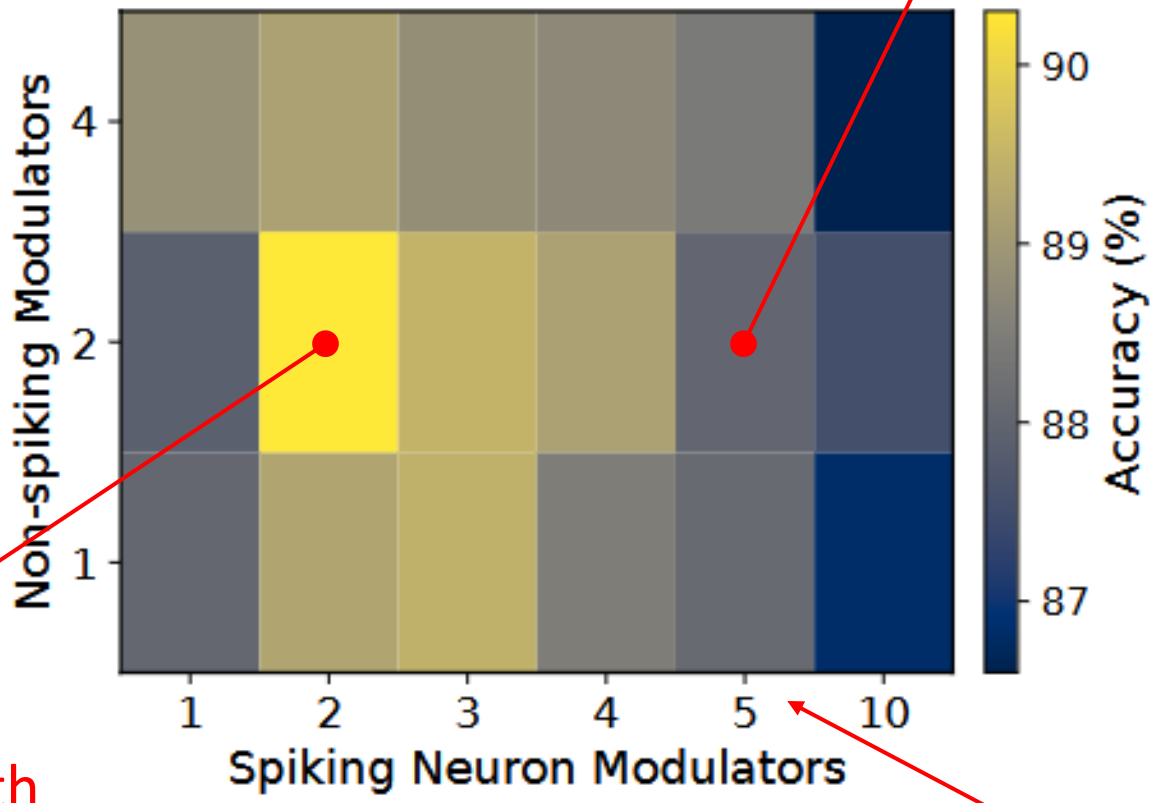




A small number of neuromodulator types is best



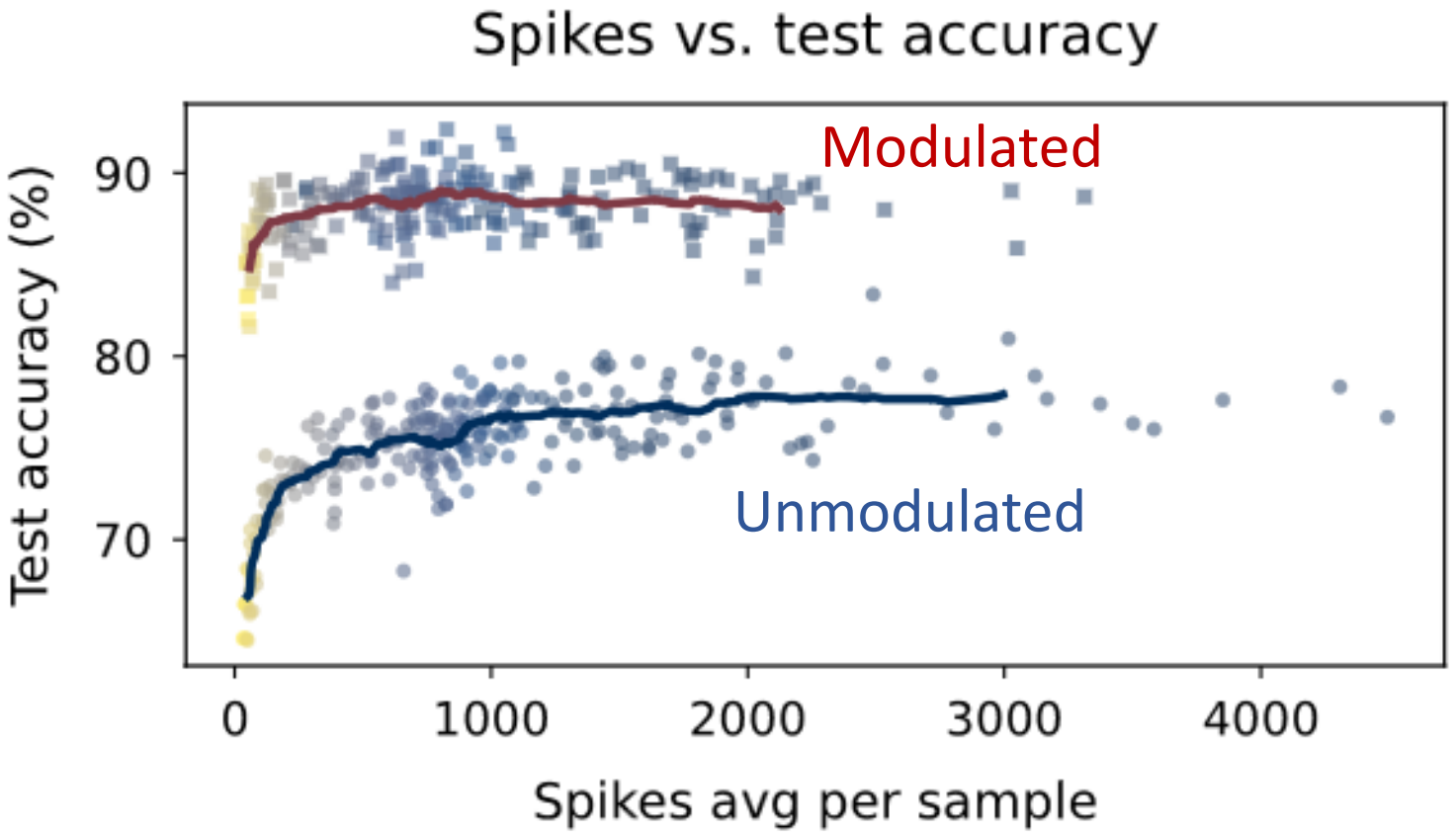
Previous results

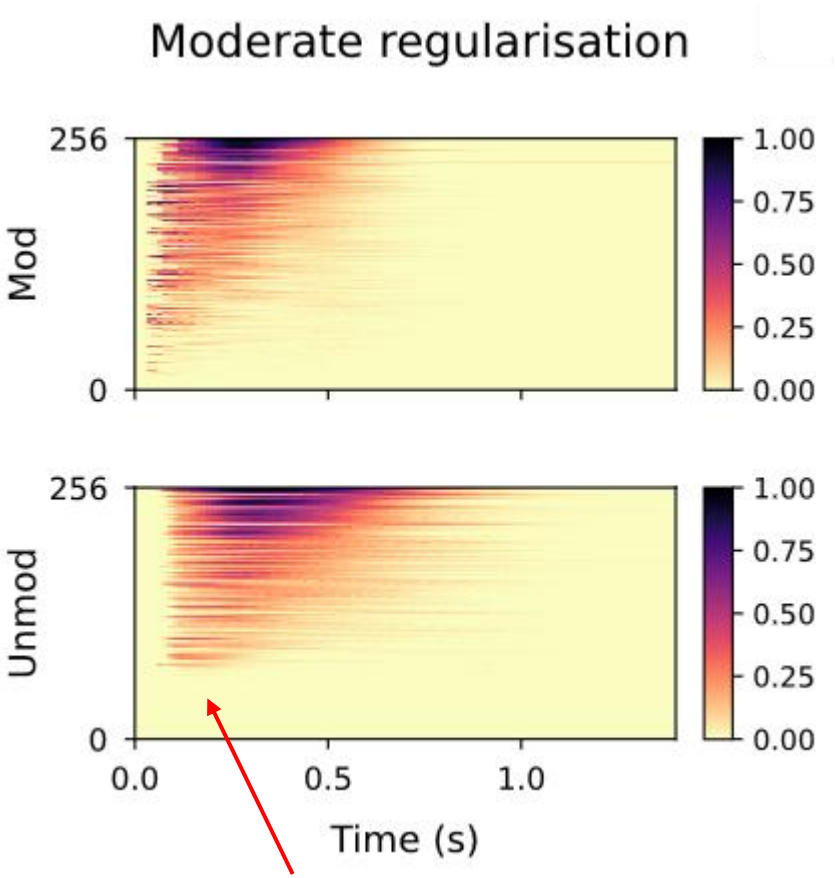


Optimal performance with fewer modulator types

Number of spiking neuron parameters = 5

Can use many fewer spikes (lower energy cost)





Wasted neurons (never active)

AlKilany and Goodman 2026

Conclusions

- Can add neuromodulation to SNNs
- Easy to implement and train (GitHub code with tutorials)
- Improves performance, parameter efficient
- Allows strong spike regularisations (useful for neuromorphic)
- Generate hypotheses for biological brains

Thanks to



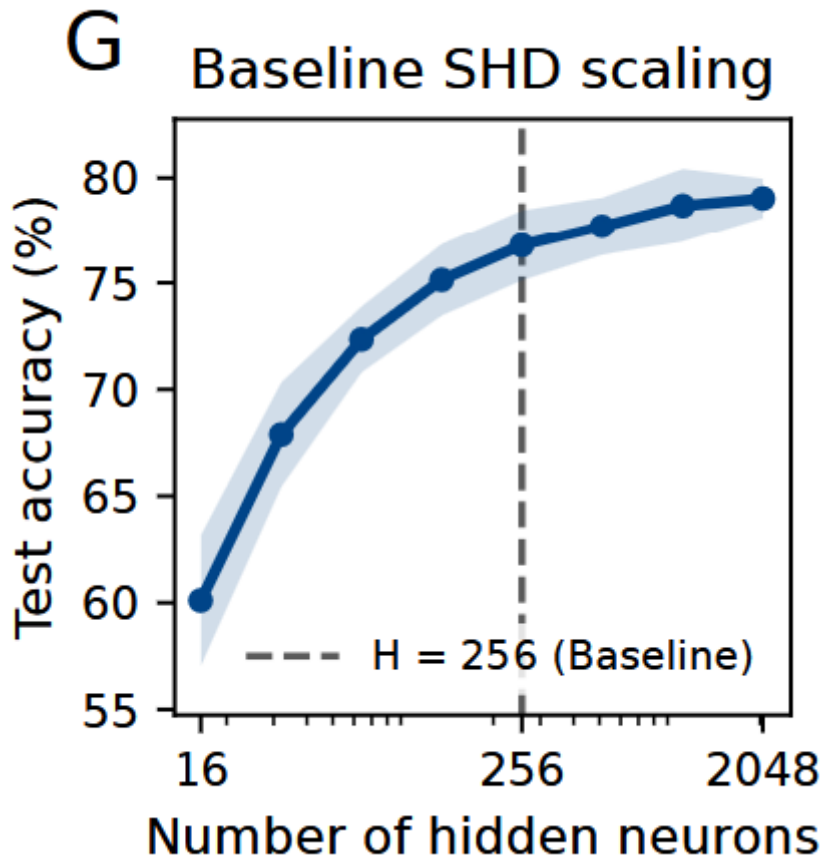
Abdal AlKilany

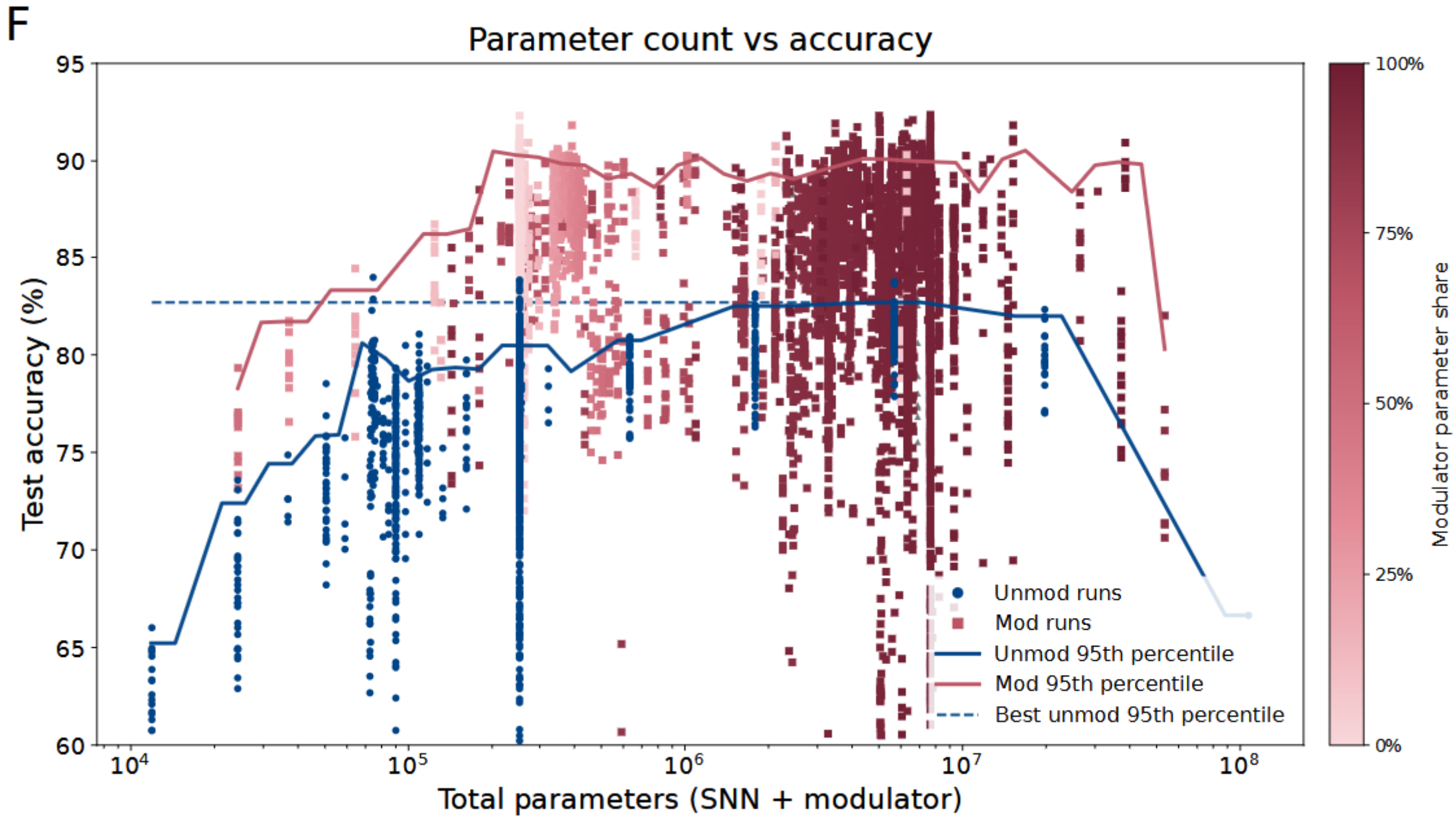
And thanks to you!

❤️

Want to learn more about spiking neural networks? Try my free online course: https://neuro4ml.github.io/

Backup slides


G
Baseline SHD scaling
Test accuracy (%)
80
75
70
65
60
55
H = 256 (Baseline)
16
256
2048
Number of hidden neurons


F
Parameter count vs accuracy
Test accuracy (%)
95
90
85
80
75
70
65
60
10^4
10^5
10^6
10^7
10^8
Total parameters (SNN + modulator)
Unmod runs
Mod runs
Unmod 95th percentile
Mod 95th percentile
Best unmod 95th percentile
Modulator parameter share
100%
75%
50%
25%
0%