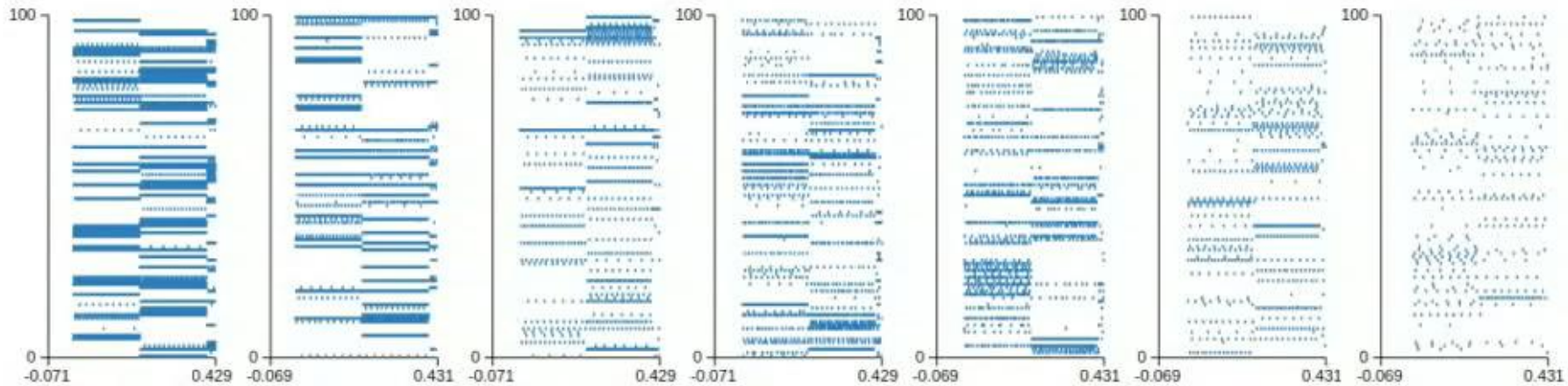
Building applications with next generation neuromorphic hardware

Chris Eliasmith

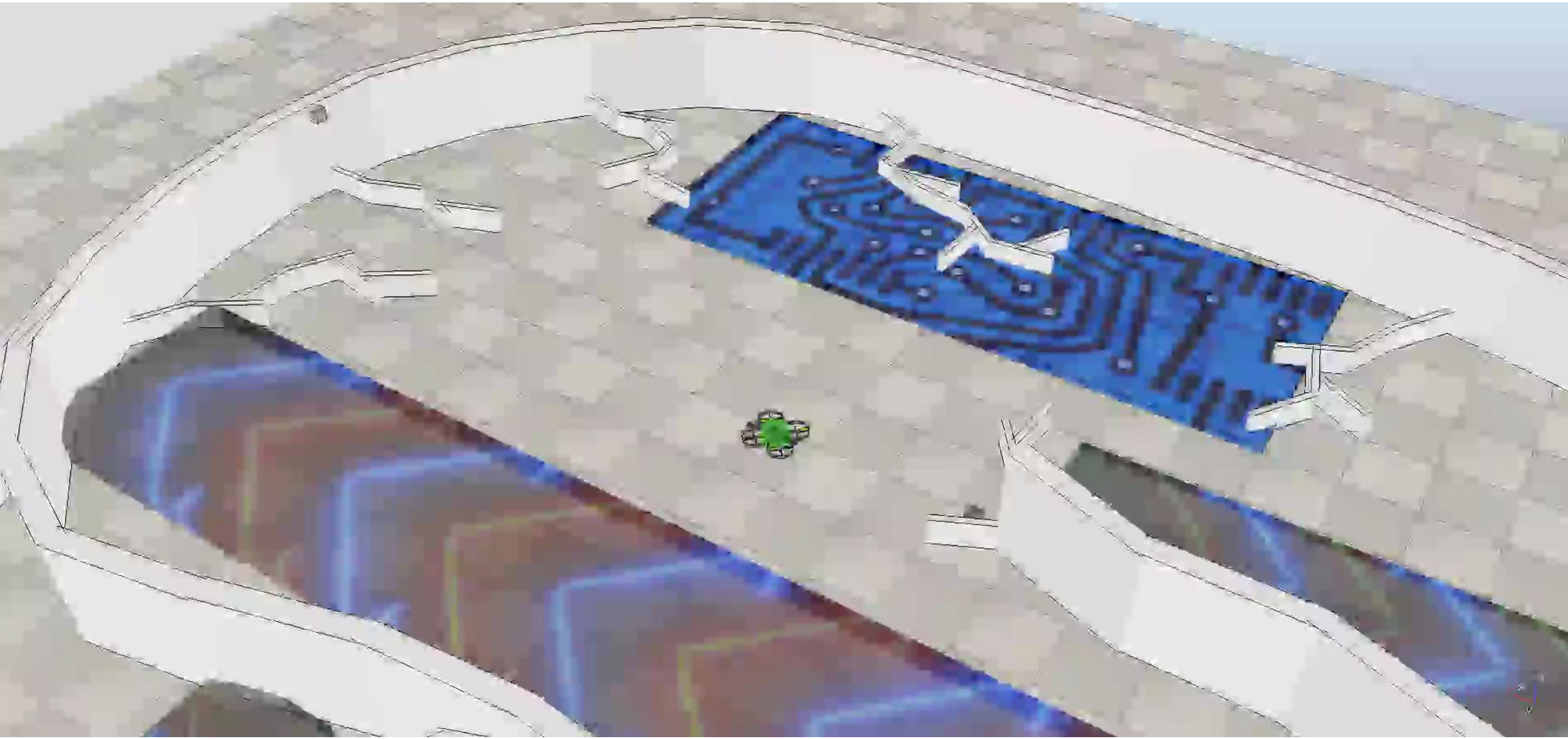
Once upon a time…

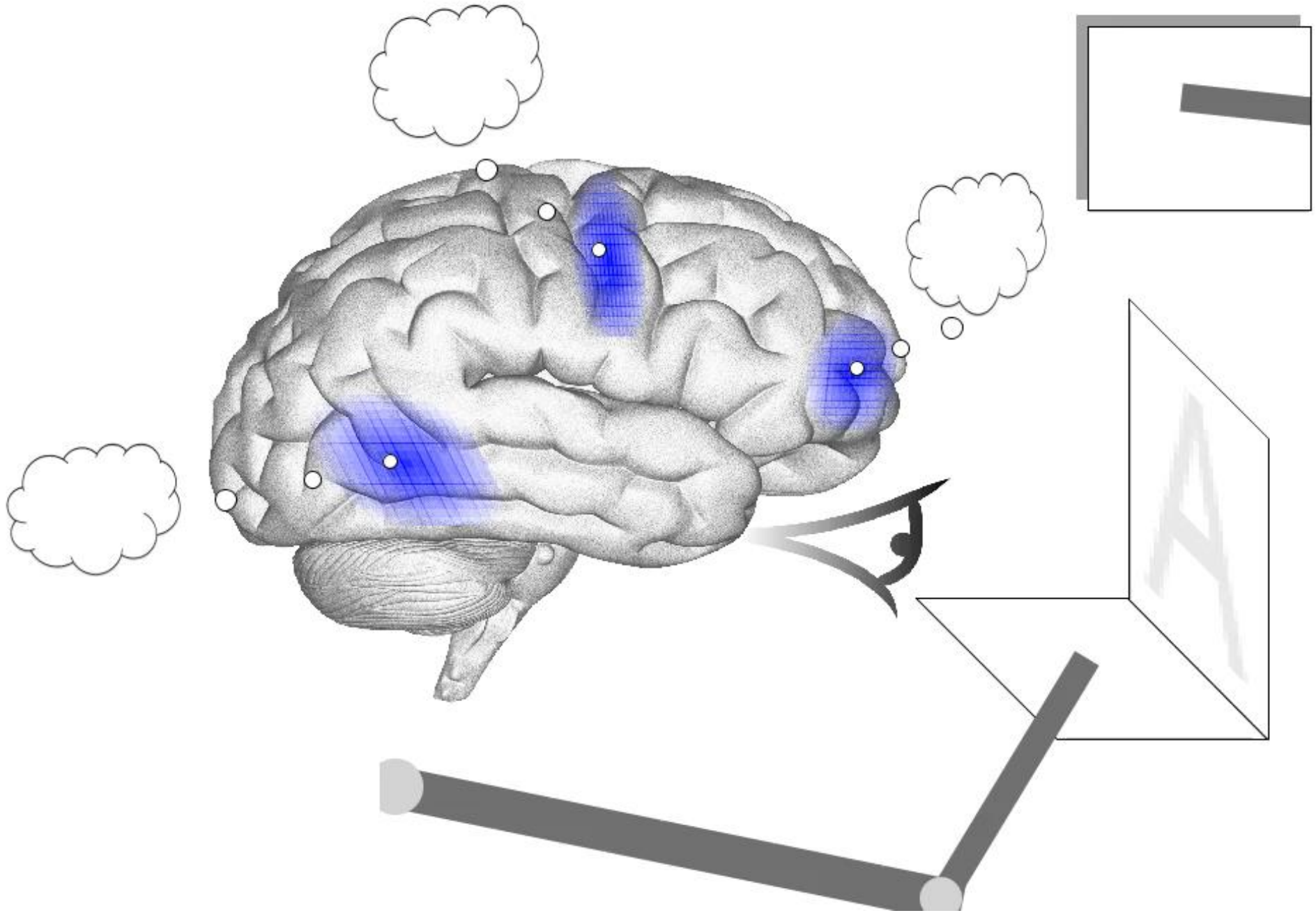

output


100
0
-0.071
0.429
100
0
-0.069
0.431
100
0
-0.071
0.429
100
0
-0.071
0.429
100
0
-0.069
0.431
100
0
-0.069
0.431
100
0
-0.069
0.431


STARFISH
STRAWBERRY
CORAL_REEF
ANEMONE_FISH
SEA_ANEMONE
BRAIN_CORAL
FLATWORM
SEA_SLUG
SEA_URCHIN
SEA_CUCUMBER
HONEYCOMB
CORAL_FUNGUS
STINKHORN
AGARIC


Speed 0.03x
Time 0.431
-3.5
-3.0
-2.5
-2.0
-1.5
-1.0
-0.5
0.0



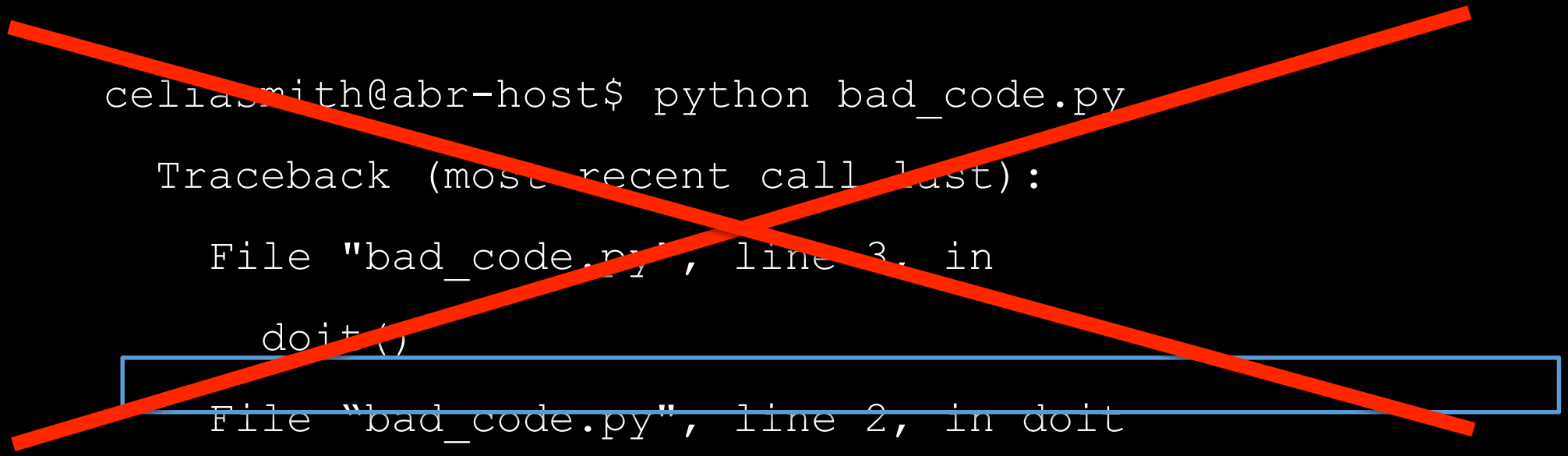

A

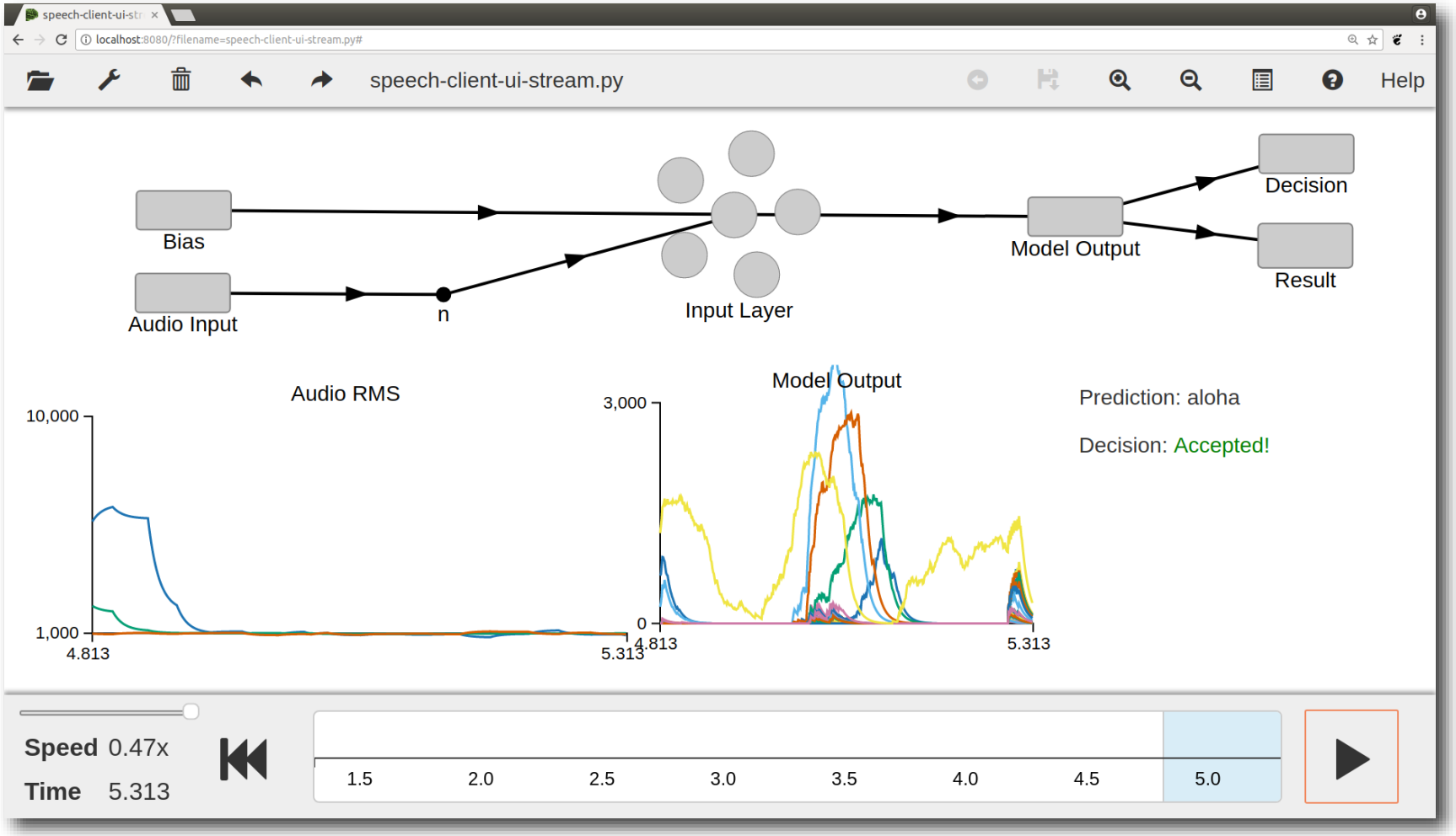
2 Months

= 5,184,000,000 ms

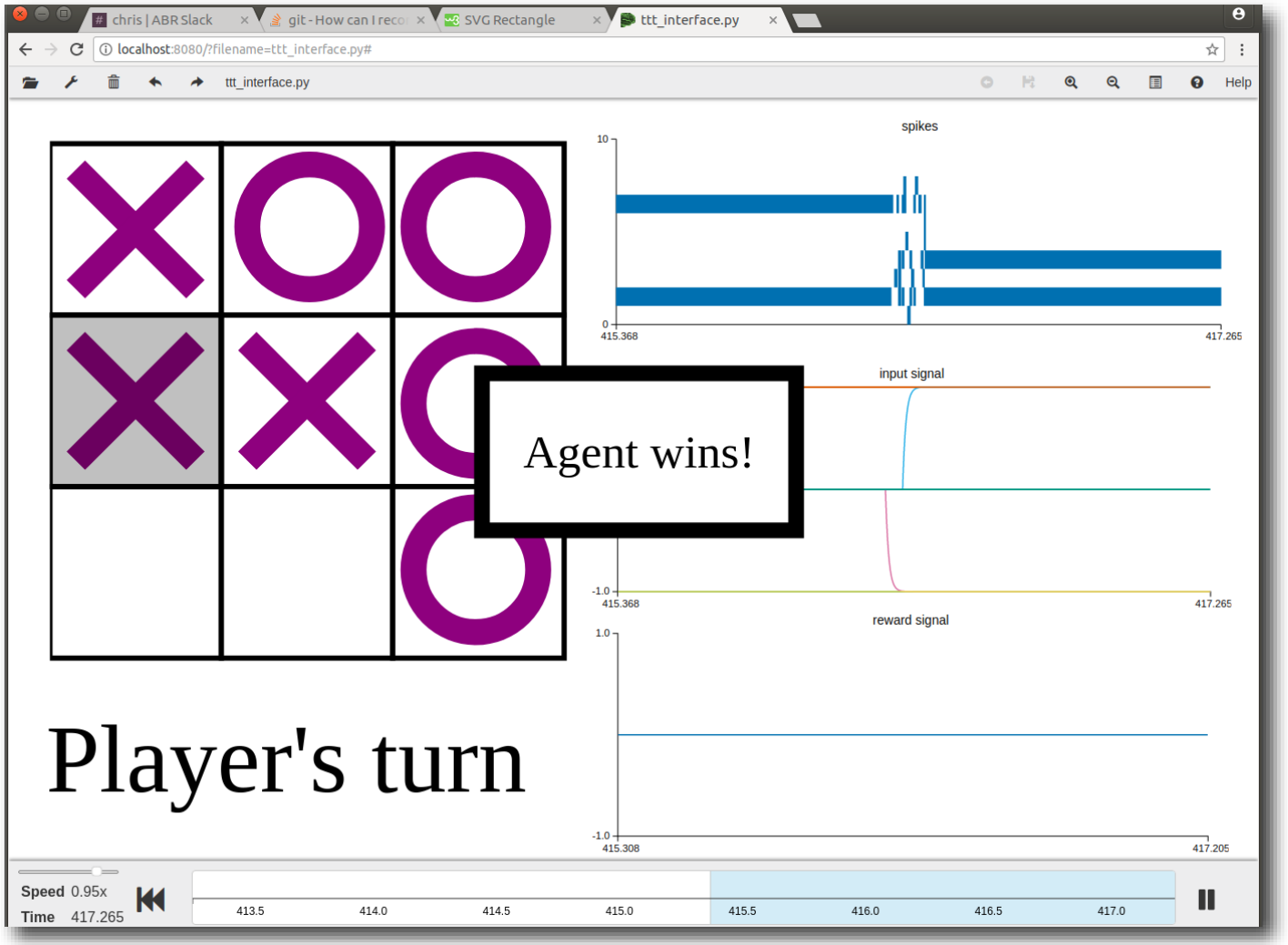
= 2 olympics

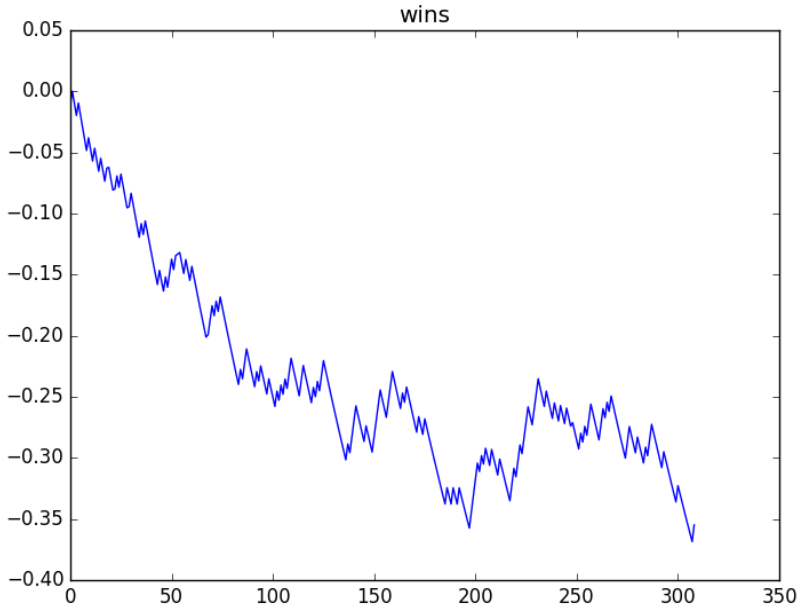
= 1/6 of a year

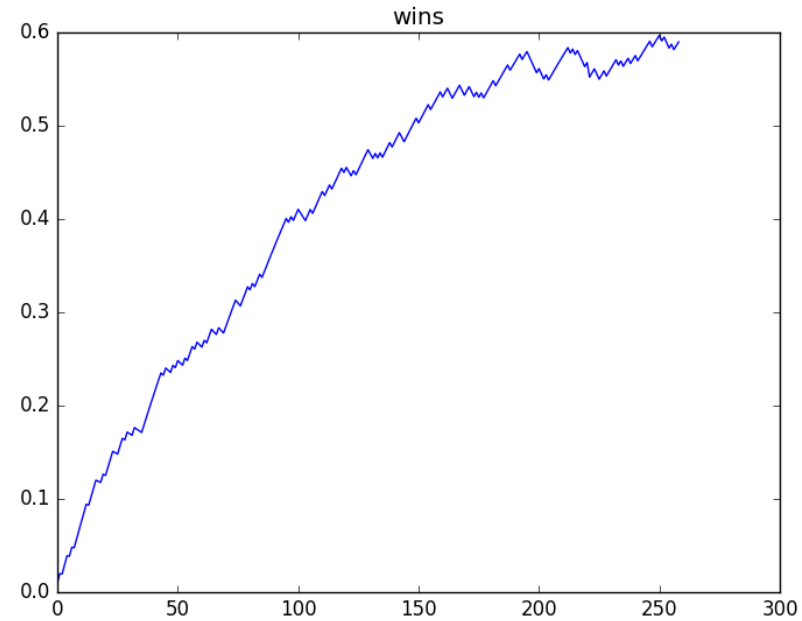

celiasmith@abr-host$ python bad_code.py
Traceback (most recent call last):
File "bad_code.py", line 3, in
doit()
File "bad_code.py", line 2, in doit
a = 6 /0
ZeroDivisionError: integer division or modulo by zero



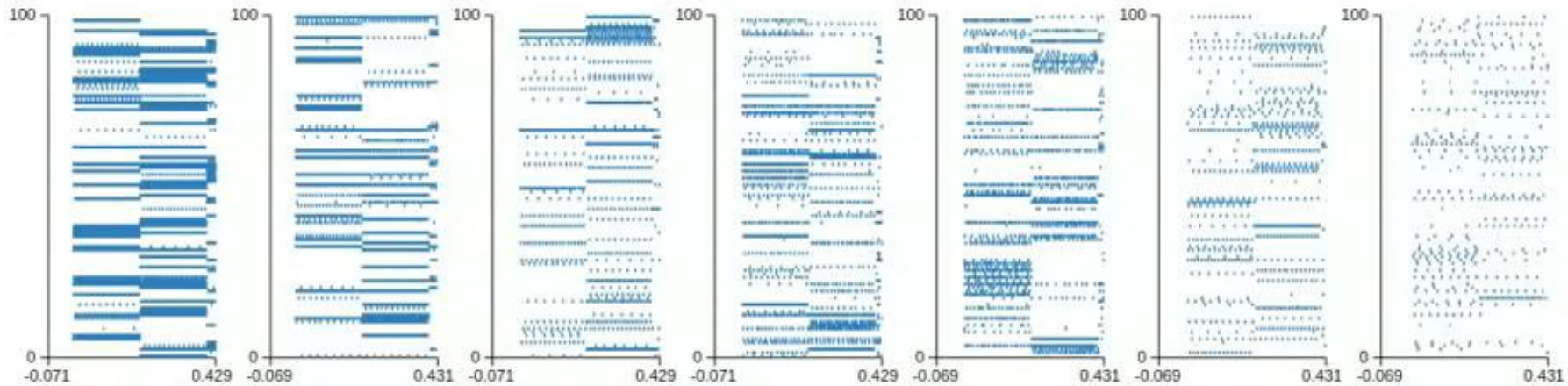
Best Model	Train Accuracy (%)	Test Accuracy (%)	False Positive (%)
Nengo	89.4	87.5	0.0


chris | ABR Slack
git - How can I reco
SVG Rectangle
ttt_interface.py
localhost:8080/?filename=ttt_interface.py#
ttt_interface.py
Help
spikes
10
0
415.368
417.265
input signal
-1.0
415.368
417.265
reward signal
1.0
-1.0
415.308
417.205
Agent wins!
Player's turn
Speed 0.95x
Time 417.265
413.5
414.0
414.5
415.0
415.5
416.0
416.5
417.0


wins
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
−0.40
0
50
100
150
200
250
300
350


wins
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
50
100
150
200
250
300


output


100
0
-0.071
0.429
100
0
-0.069
0.431
100
0
-0.071
0.429
100
0
-0.071
0.429
100
0
-0.069
0.431
100
0
-0.069
0.431
100
0
-0.069
0.431


STARFISH
STRAWBERRY
CORAL_REEF
ANEMONE_FISH
SEA_ANEMONE
BRAIN_CORAL
FLATWORM
SEA_SLUG
SEA_URCHIN
SEA_CUCUMBER
HONEYCOMB
CORAL_FUNGUS
STINKHORN
AGARIC


Speed 0.03x
Time 0.431
-3.5
-3.0
-2.5
-2.0
-1.5
-1.0
-0.5
0.0

Adventure™

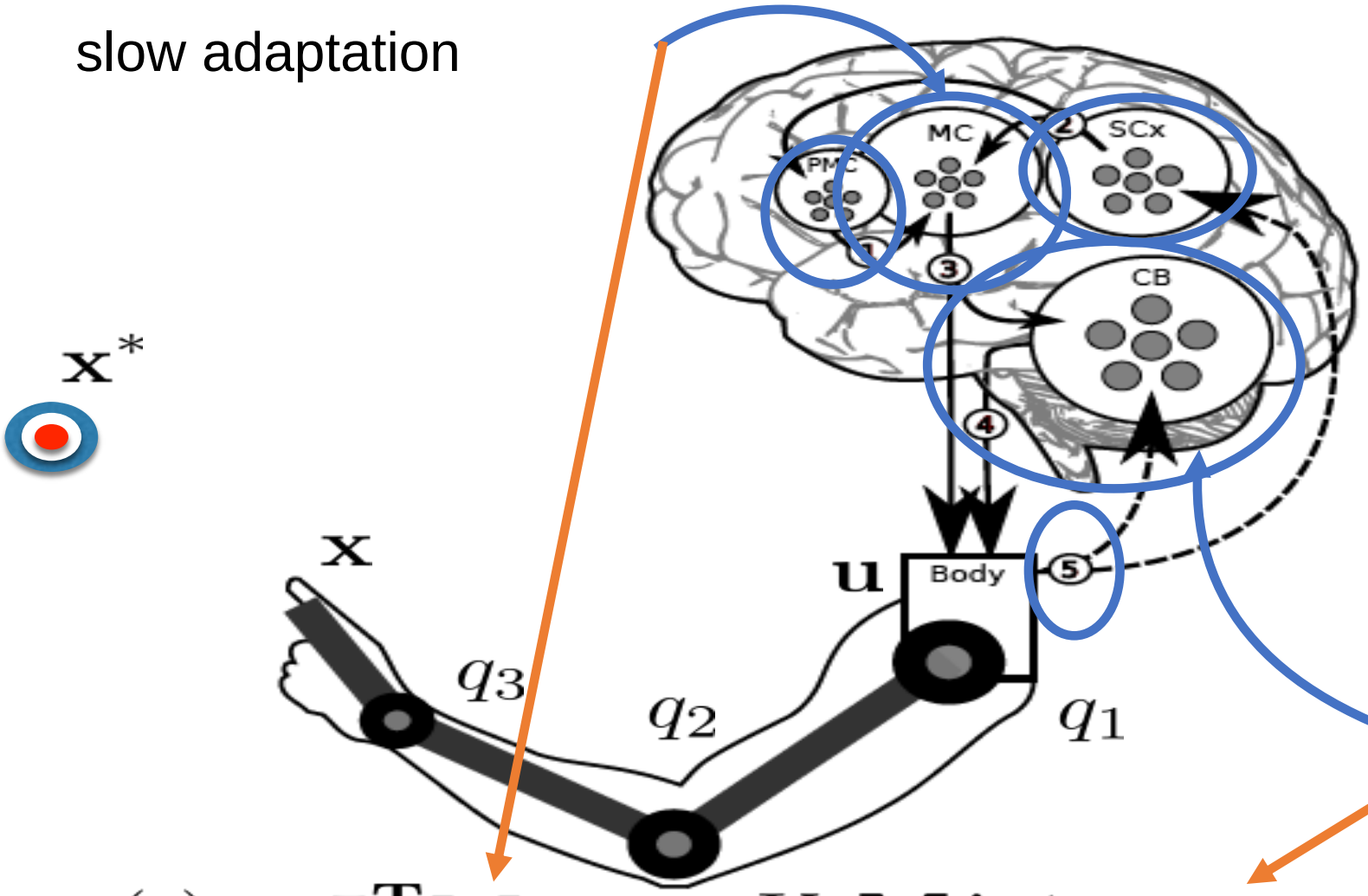
Motor Control

Variable nonlinear dynamics (fast adaptation)

Variable kinematics (slow adaptation)

REACH: Adaptive Motor Control

slow adaptation

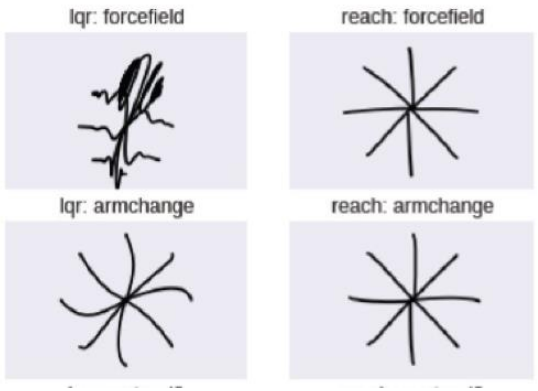


1. $\mathbf{x}^*$
2. $\mathbf{u_x} = K_p(\mathbf{x} - \mathbf{x}^*)$
3. $\mathbf{J^T M_x u_x}$
4. $-K_v\mathbf{M\dot{q}} + \mathbf{u}_{adapt}$
5. $\mathbf{x}$

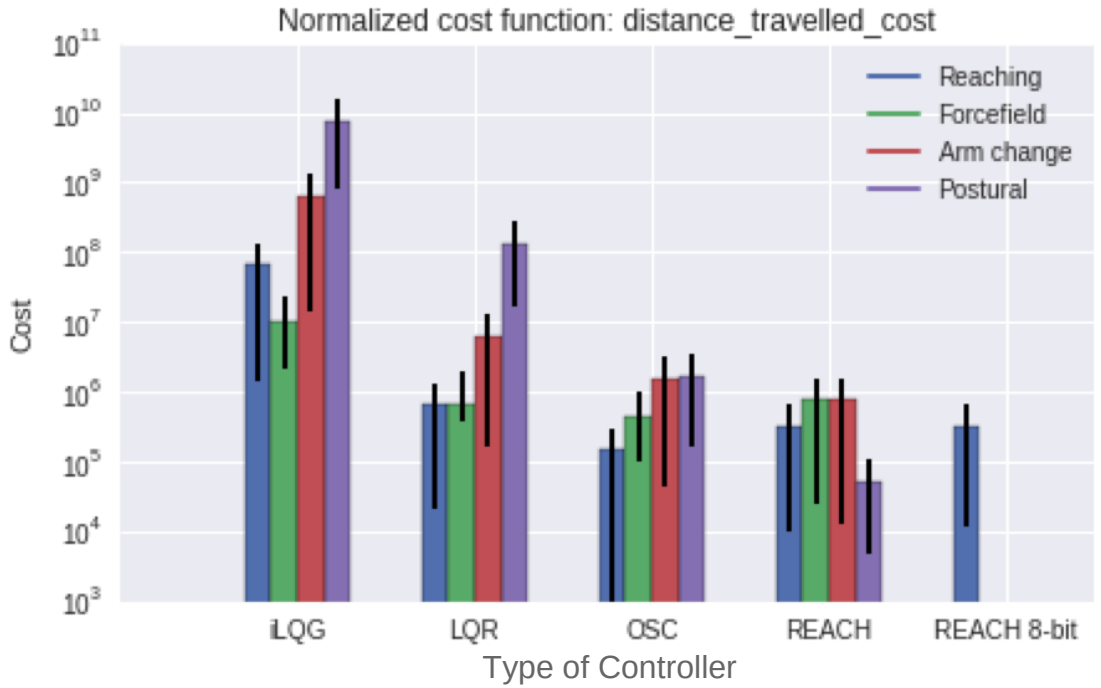
rapid adaptation

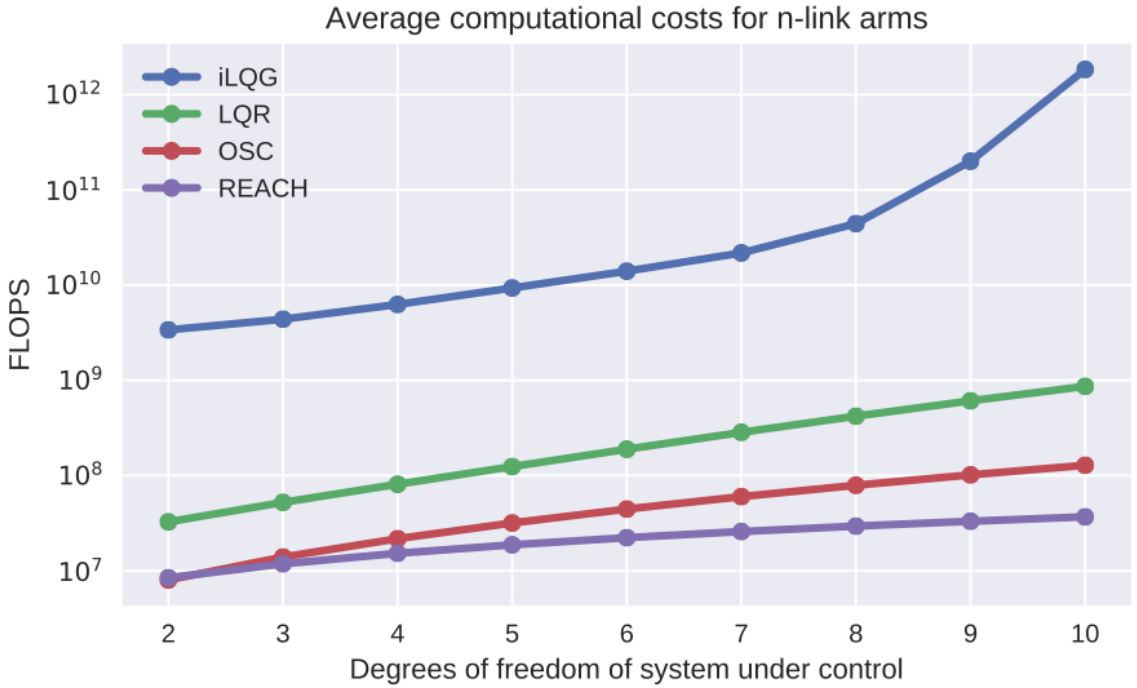
$$\mathbf{u}(t) = \mathbf{J^T M_x u_x} - K_v\mathbf{M\dot{q}} + \mathbf{u}_{adapt}$$

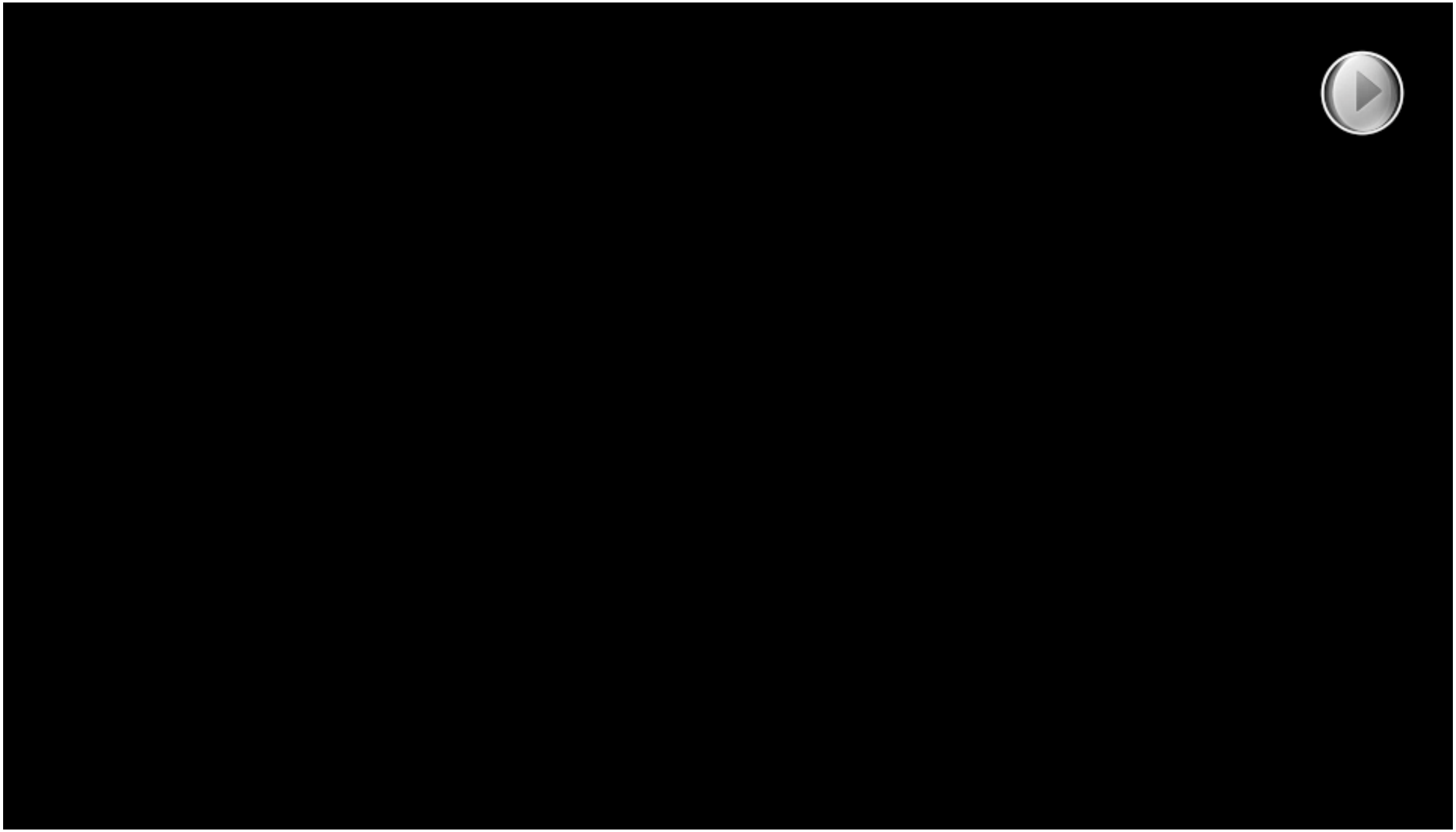
REACH



- **Lower control cost and more efficient than state-of-the-art**
  - REACH is a neural architecture with specific sites for adaptation
  - Improves on state-of-the-art controllers





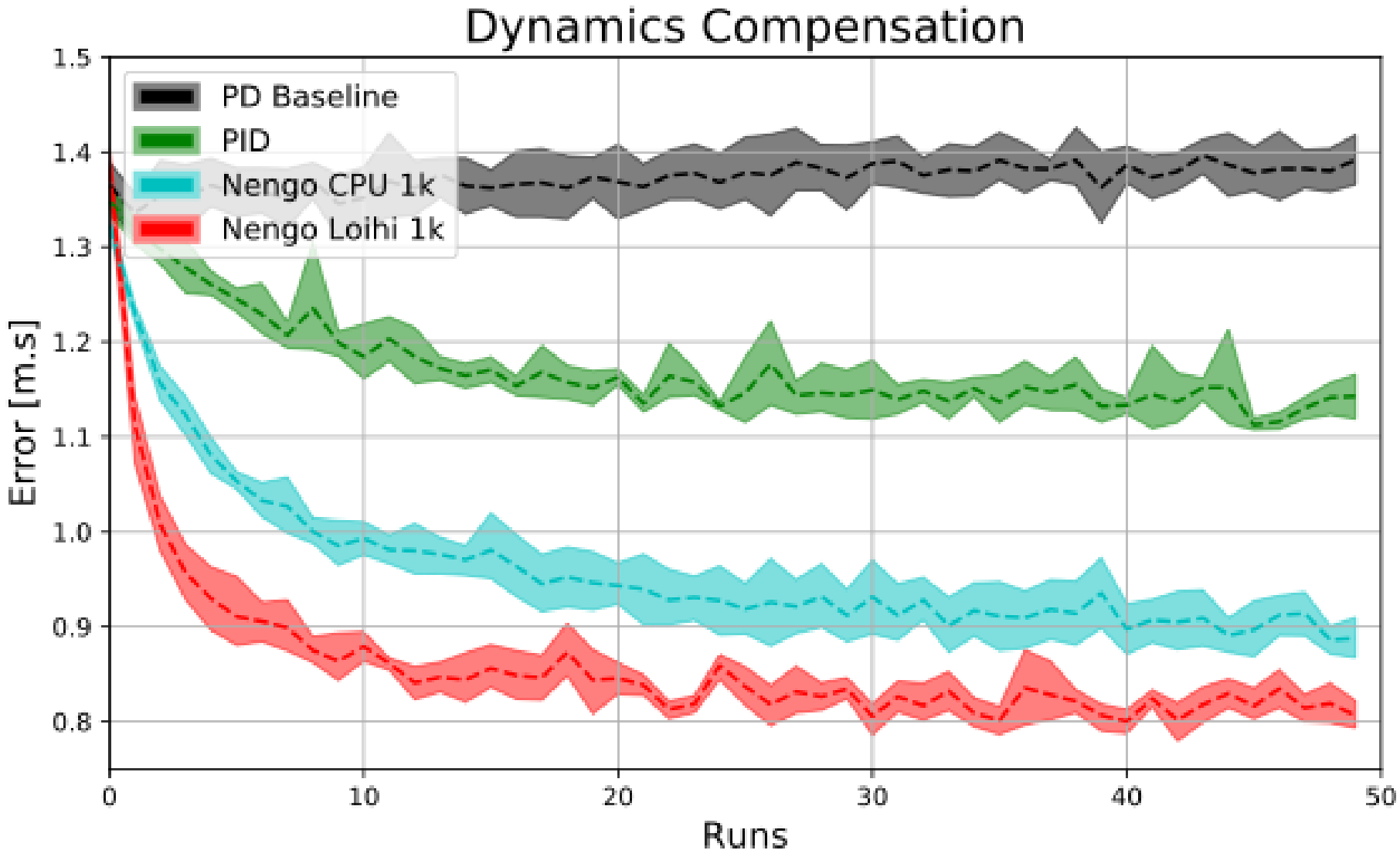


Challenges

- Build the adaptive controller in *spiking hardware* (Loihi)
- Demonstrate it is better than other solutions (e.g. PID)
- Demonstrate an *industrially relevant* result
- Demonstrate that the result *depends on* the adaptation running on hardware

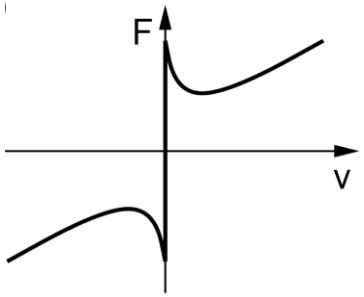
Results

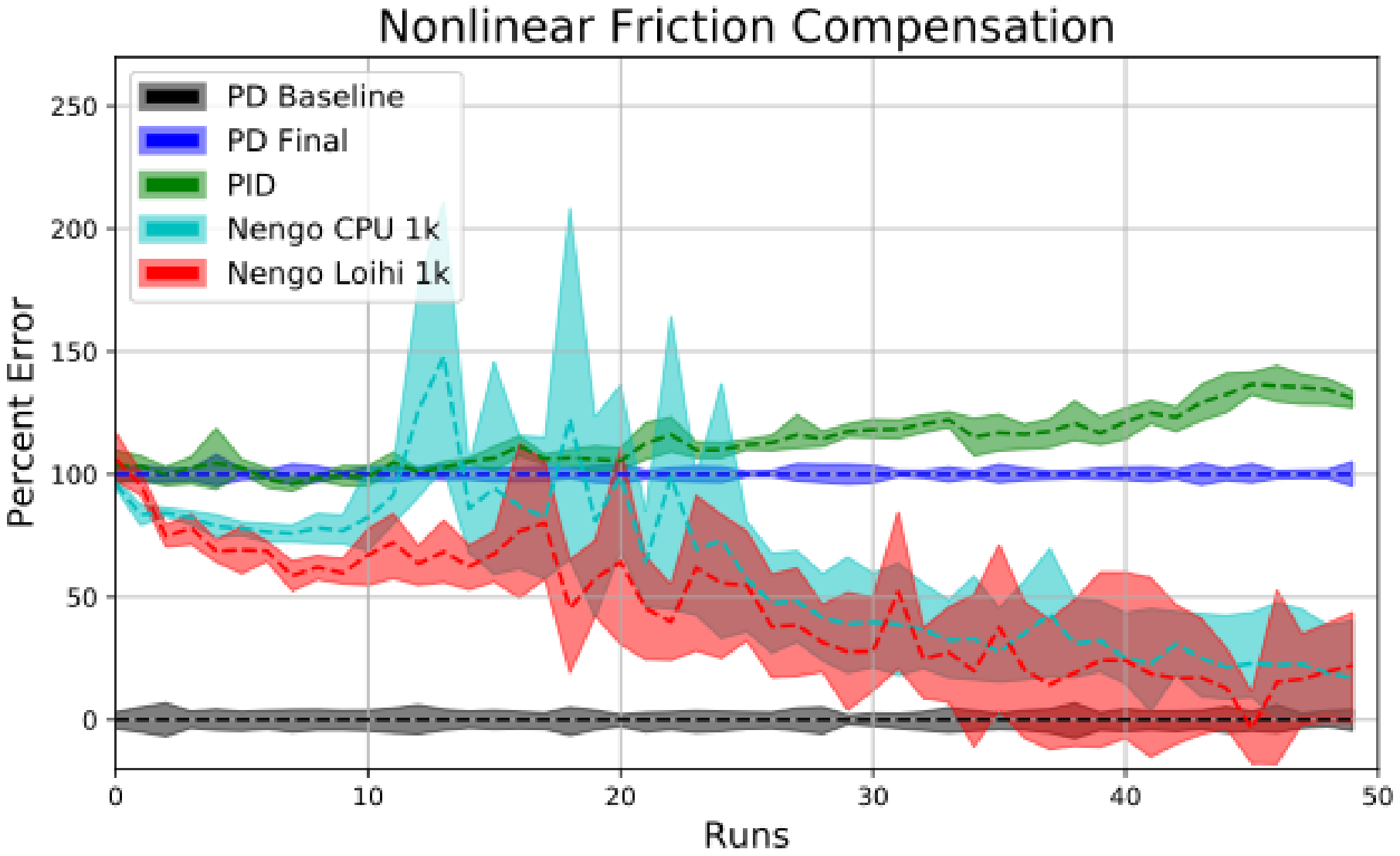
- One reach
- 5 trials
- 50 runs
- 1k neurons
- Save weights



Results

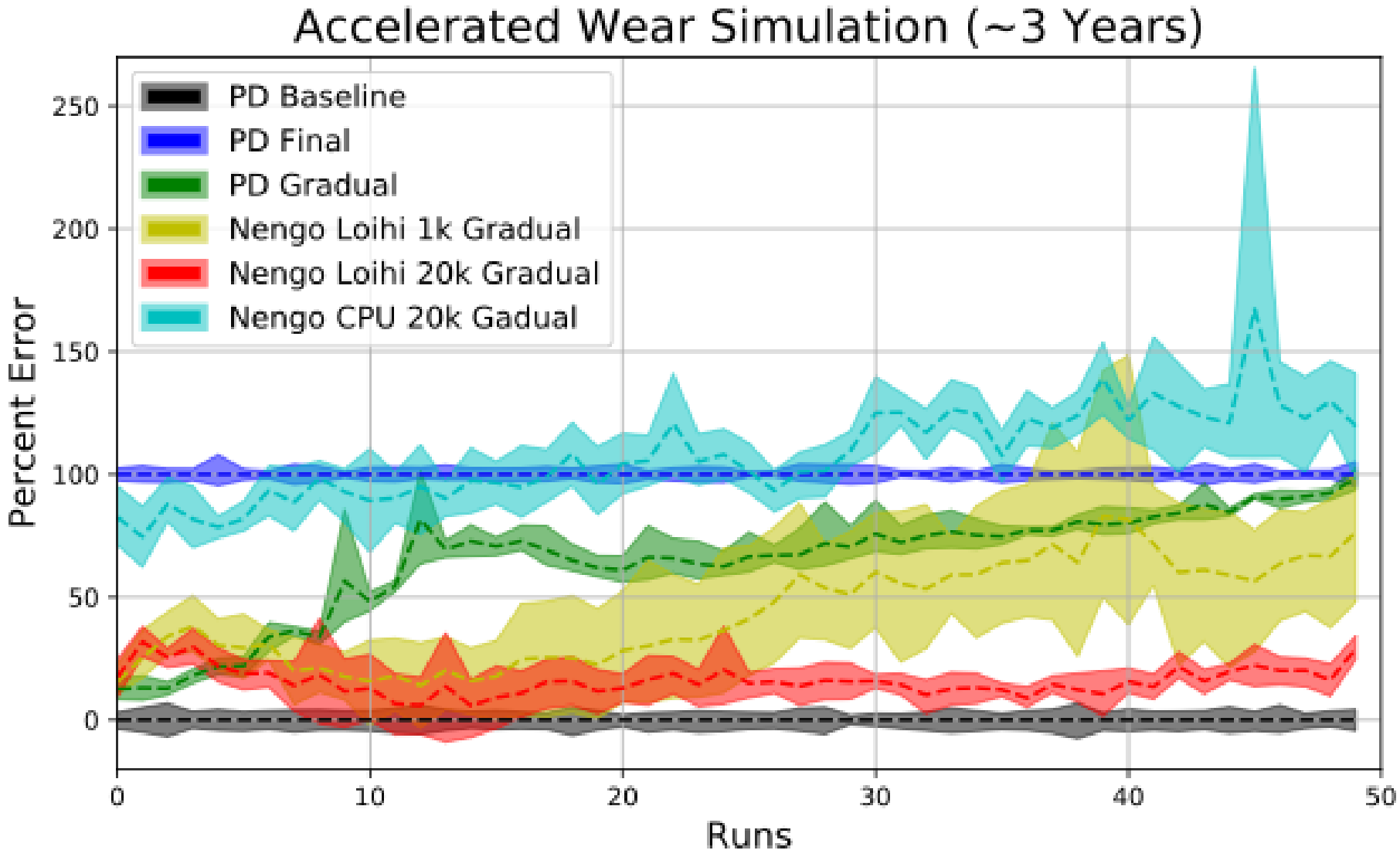
- Five reaches
- Nonlinear friction (3 yrs)
- 5 trials
- 50 runs
- 1k neurons
- Save weights





Results

- Same as prev
- Gradual friction increase



Icing…

- Got power measurements yesterday
- 20K neurons - CPU vs Loihi improvement:
  - 36x
- Speech demo - CPU vs Loihi improvement:
  - 47x

Addressing challenges for AI

- **A way to embrace the strengths and address the challenges facing contemporary AI**
  - We use what works in machine learning, and fix the things that don't

Strengths of ML	Challenges for ML	Our Solution
Feedforward classification	Dynamics	Optimal Dynamics (computed, learned)
Broadly applicable	Integration of methods	Nengo
Massively scalable	Power consumption	Spiking neurons, custom hardware

Scaling

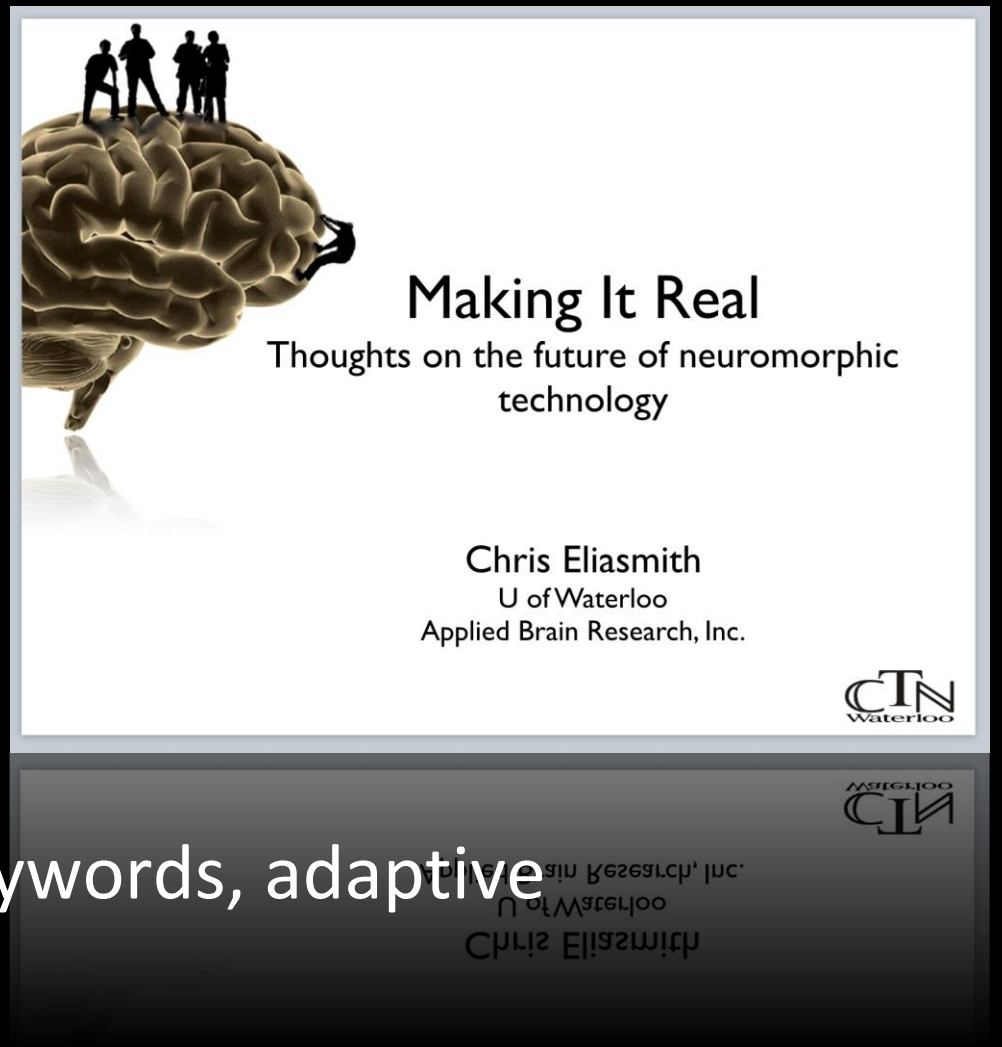
Complexity - 20K, 84K neurons

Scaling

Generality - DL, RL, adaptive control, audition

Scaling

Applications - Industrially relevant (keywords, adaptive control)



The race is on…

…the adventure continues…