Harnessing Noise for Neuromorphic Control

Marcel van Gerven

Professor of Artificial Cognitive Systems

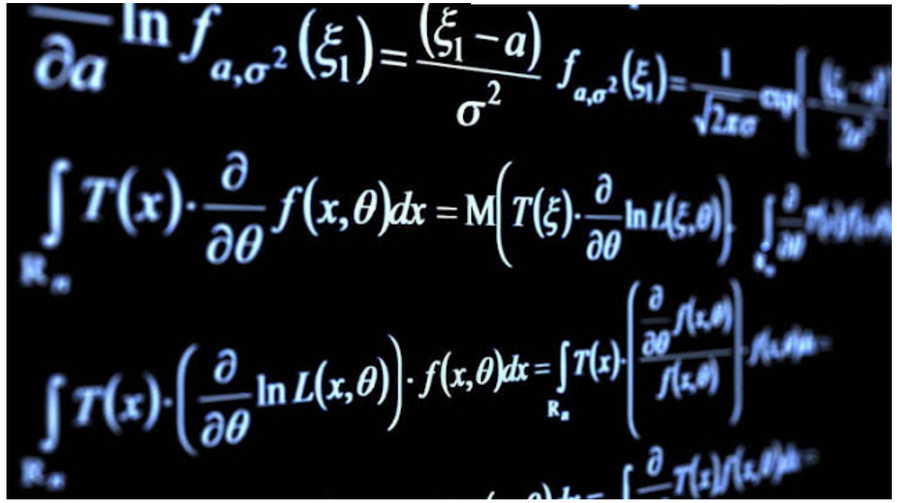
Department of Machine Learning and Neural Computing

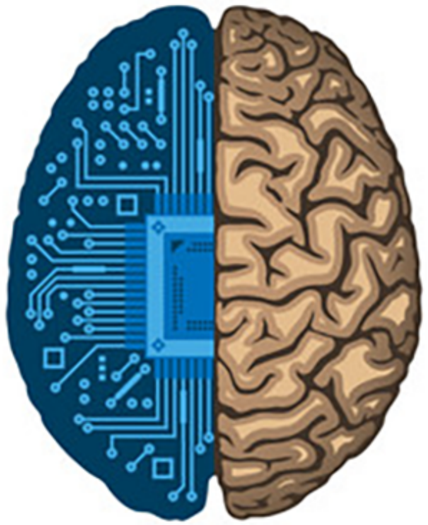
Donders Institute for Brain, Cognition and Behaviour, Radboud University

ELLIS Fellow, Director of ICAI lab for Semiconductor Manufacturing

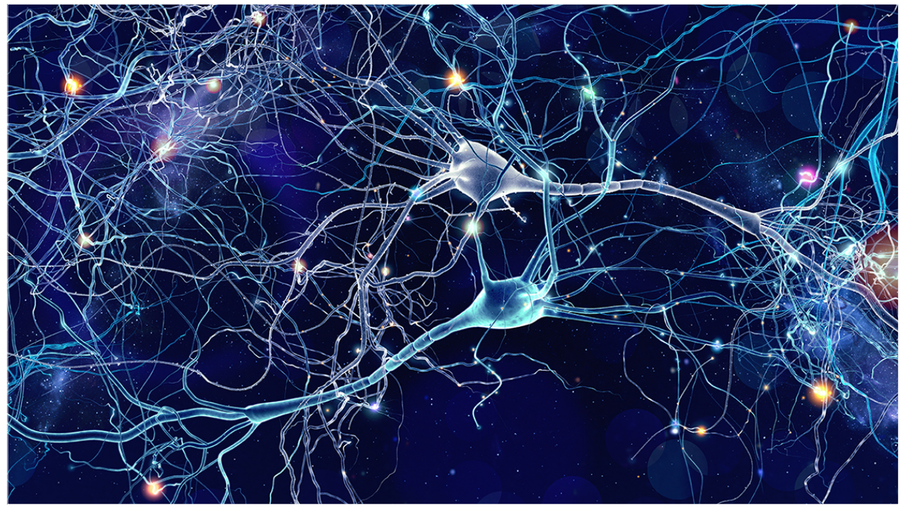
ARTIFICIAL COGNITIVE SYSTEMS LAB

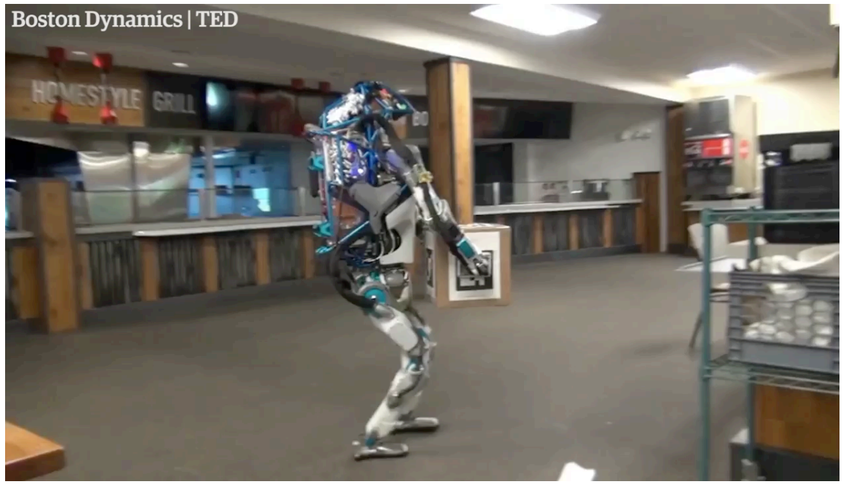
Machine learning





Neural computing



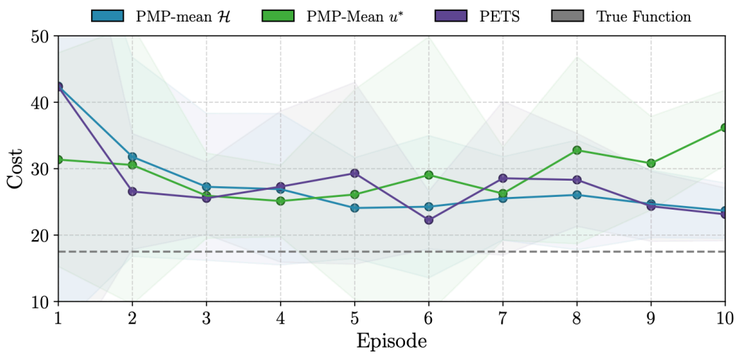


bridging the gap

Mathematics
Physics
Chemistry
Biology
Neuroscience
Cognition

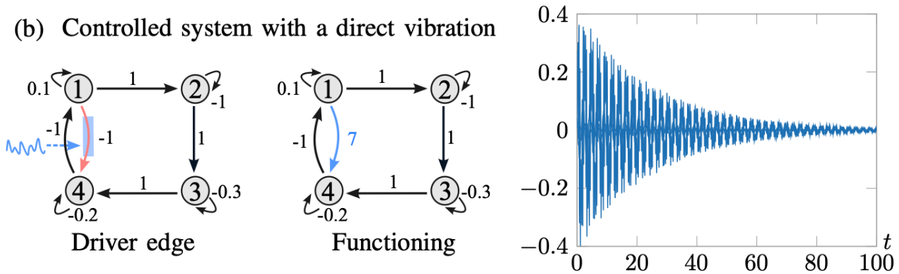
THEORETICAL RESEARCH

Probabilistic Pontryagin maximum principle



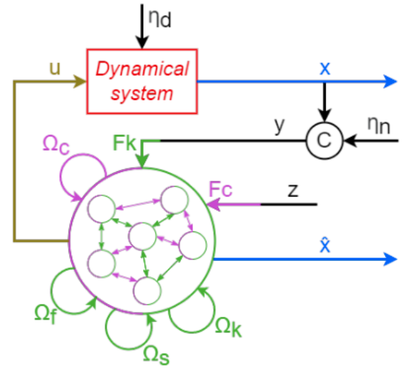
Leeftink, D., A˘ Gatay Yıldız, C. ¸, Ridderbusch, S., Hinne, M., & van Gerven, M. (2005). Probabilistic Pontryagin's Maximum Principle for Continuous-Time Model-Based Reinforcement Learning. Arxiv.

Vibrational control of complex networks



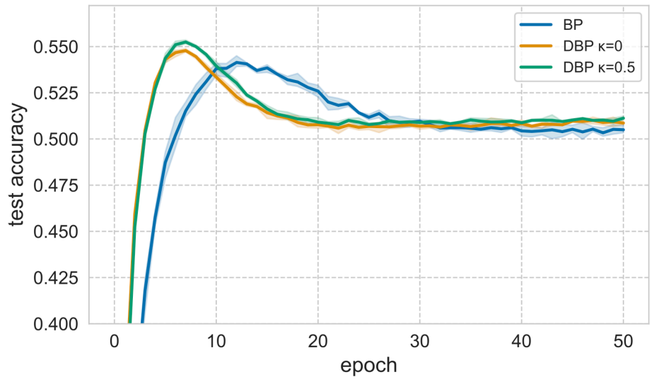
Y. Qin, F. Pasqualetti, D. Bassett, and M. van Gerven, Vibrational Control of Complex Networks, IEEE Transactions on Control of Network Systems, submitted, 2024

Spiking control



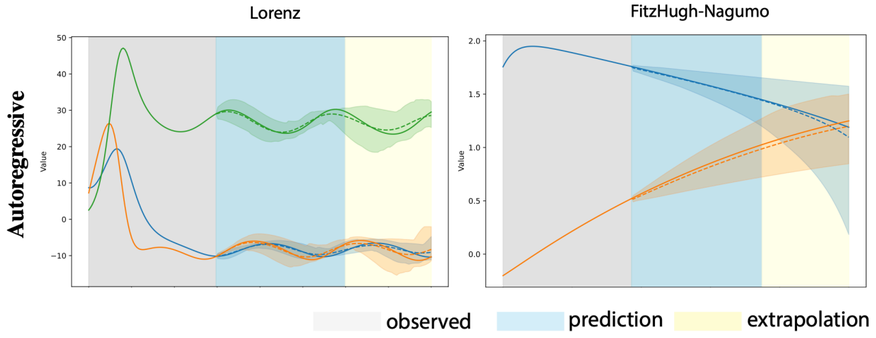
Slijkhuis, F. S., Keemink, S. W., & Lanillos, P. (2022). Estimating and controlling dynamical systems through coordinated spikes. Arxiv.

Efficient deep learning



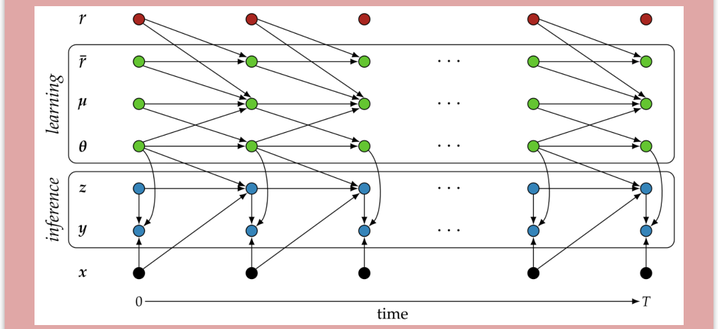
Dalm, S., Offergeld, J., Ahmad, N., & van Gerven, M. (2024). Efficient Deep Learning with Decorrelated Backpropagation. ArXiv Preprint ArXiv:2405.02385 [Cs.LG]. http://arxiv.org/abs/2405.02385

Autoregressive flow matching



El-Gazzar, A., & van Gerven, M. (2025). Probabilistic Forecasting via Autoregressive Flow Matching. ArXiv:2503.10375 (Cs).

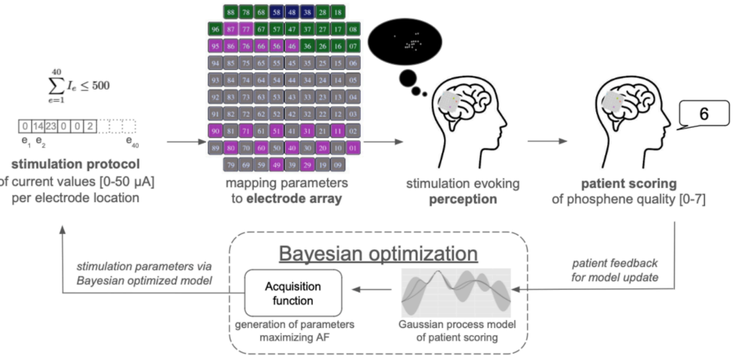
Ornstein-Uhlenbeck Adaptation



García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

APPLIED RESEARCH

Neuroscience

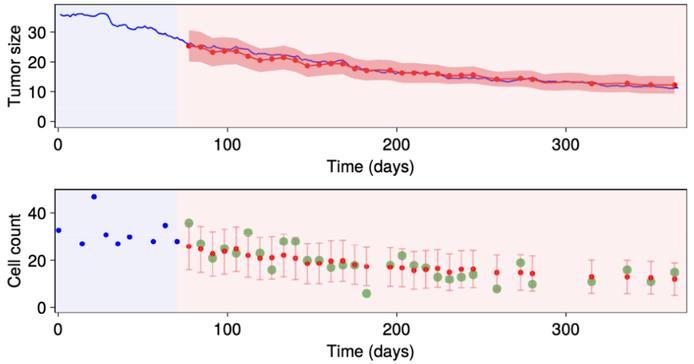


Küçüko˘ Glu, B., Soo, L., Leeftink, D., Grani, F., Soto Sanchez, C., Güçlü, U., van Gerven, M., & Fernandez, E. (2025). Bayesian optimization of cortical neuroprosthetic vision using perceptual feedback. BioRxiv,

Semiconductor manufacturing

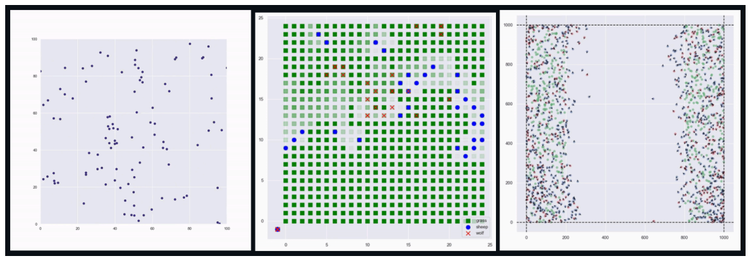
Under embargo

Healthcare



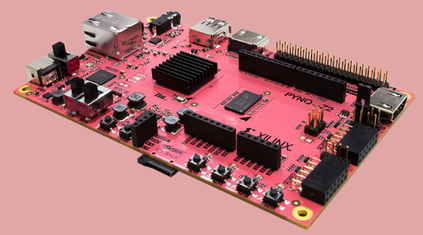
Submission in progress

Agent-based modeling



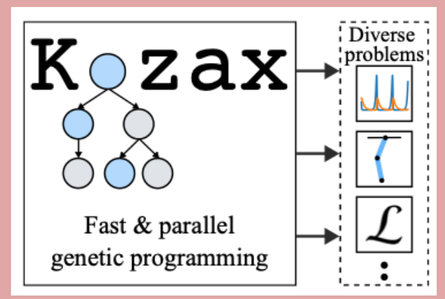
Siddharth Chaturvedi and Ahmed EL-Gazzar and Marcel van Gerven. ABMAX: An agent-based modeling framework in Jax. https://github.com/i-m-iron-man/abmax

Neuromorphic computing



https://github.com/umutcanaltin/fpgai_compiler/blob/main/paper/assets/hls.pdf

Genetic programming



de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

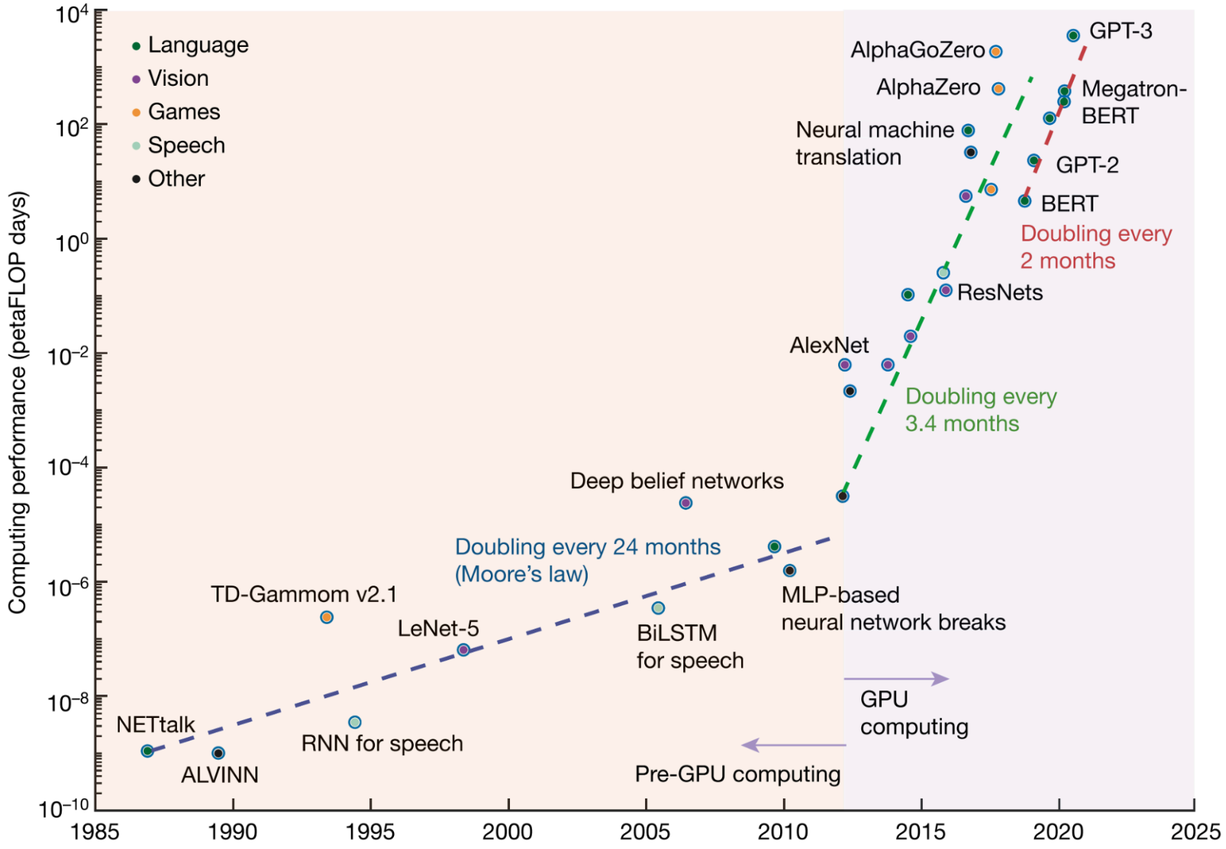
OUTLINE

- NEUROMORPHIC COMPUTING
- NOISE-BASED LEARNING
- GENETIC PROGRAMMING
- CONCLUSIONS

OUTLINE

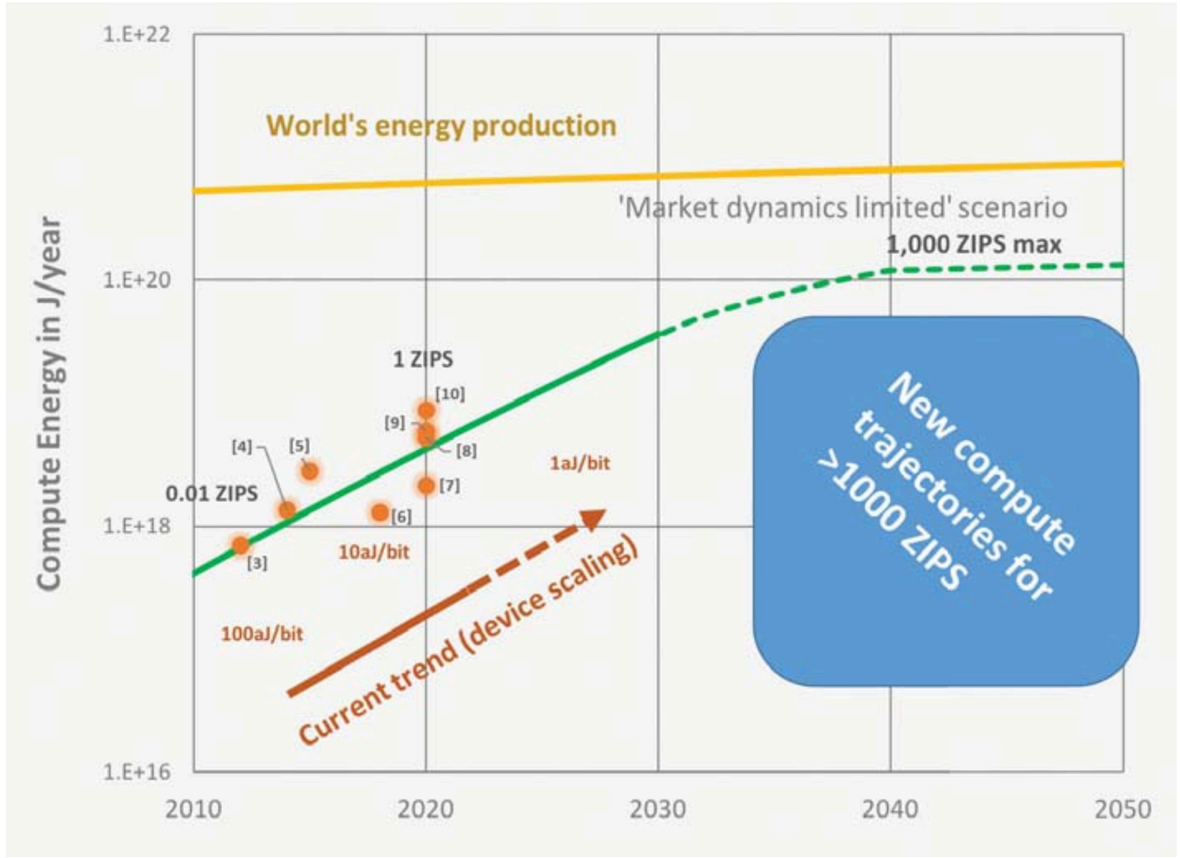
- NEUROMORPHIC COMPUTING
- NOISE-BASED LEARNING
- GENETIC PROGRAMMING
- CONCLUSIONS

THE COST OF COMPUTE



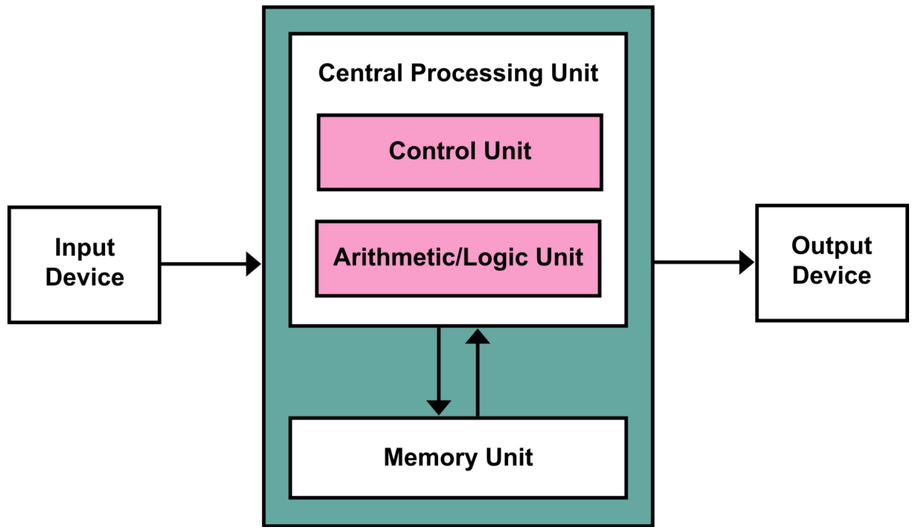
Mehonic, A., & Kenyon, A. J. (2022). Brain-inspired computing needs a master plan. Nature.

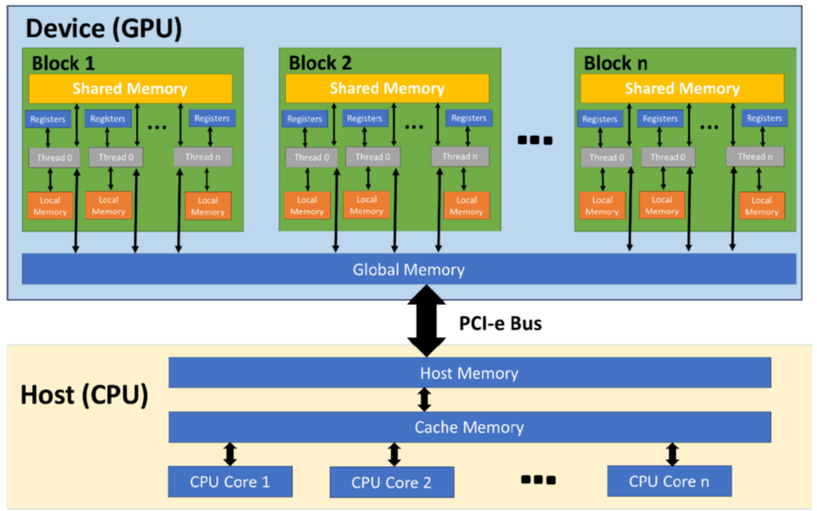
1 ZIP = 10^{15} MIPS

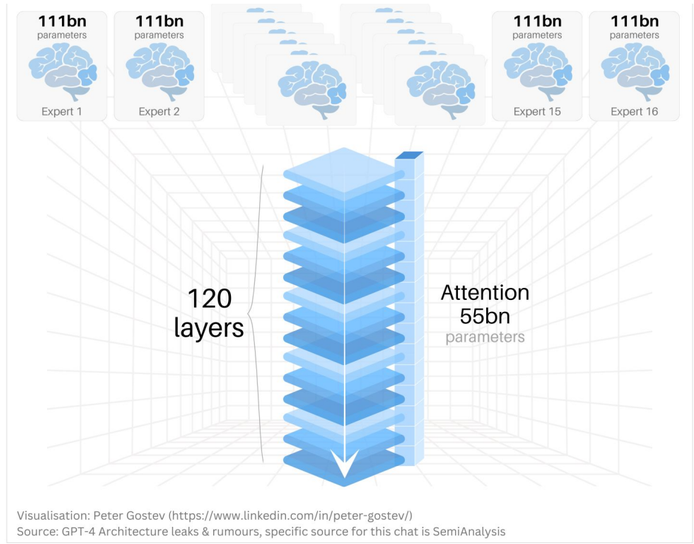


https://www.src.org/about/decadal-plan/

CONVENTIONAL COMPUTING



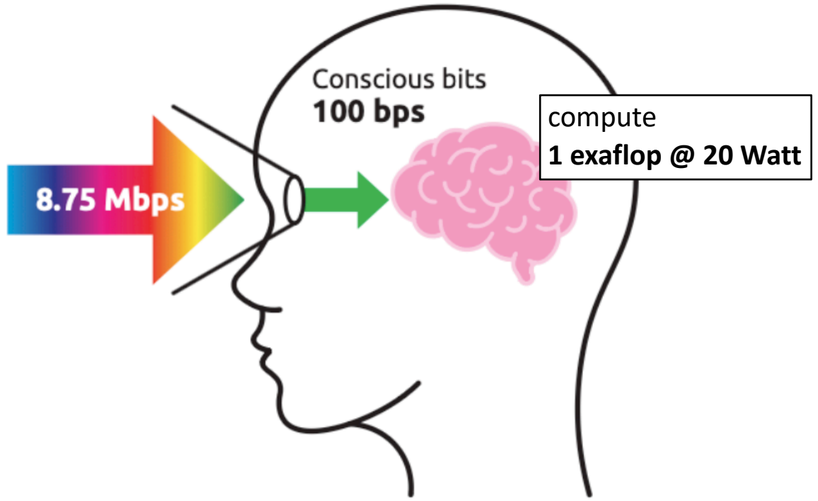




- Conventional computing is based on the von Neumann architecture
- Separation of memory and compute creates a **von Neumann bottleneck**
- Training and inference of deep neural networks as stored programs that run *on the hardware*

WHAT IS NEUROMORPHIC COMPUTING?

Neuromorphic computing is a **brain-inspired unconventional computing paradigm** which aims to mimic neural information processing in biological systems



- Orders of magnitude more efficient than conventional computing **(event-driven, compressed sensing, low power)**
- Robust compute in noisy physical devices **(analog computing)**
- Assumes that there is no distinction between hardware and software **(in-materia computing)**
- Computation that is local in space and time **(in-memory computing)**

Mead, C. (1990). Neuromorphic Electronic Systems. Proceedings of the IEEE, 78(10), 1629–1636.

WHY NEUROMORPHIC COMPUTING?

SCIENTIFIC ARGUMENT

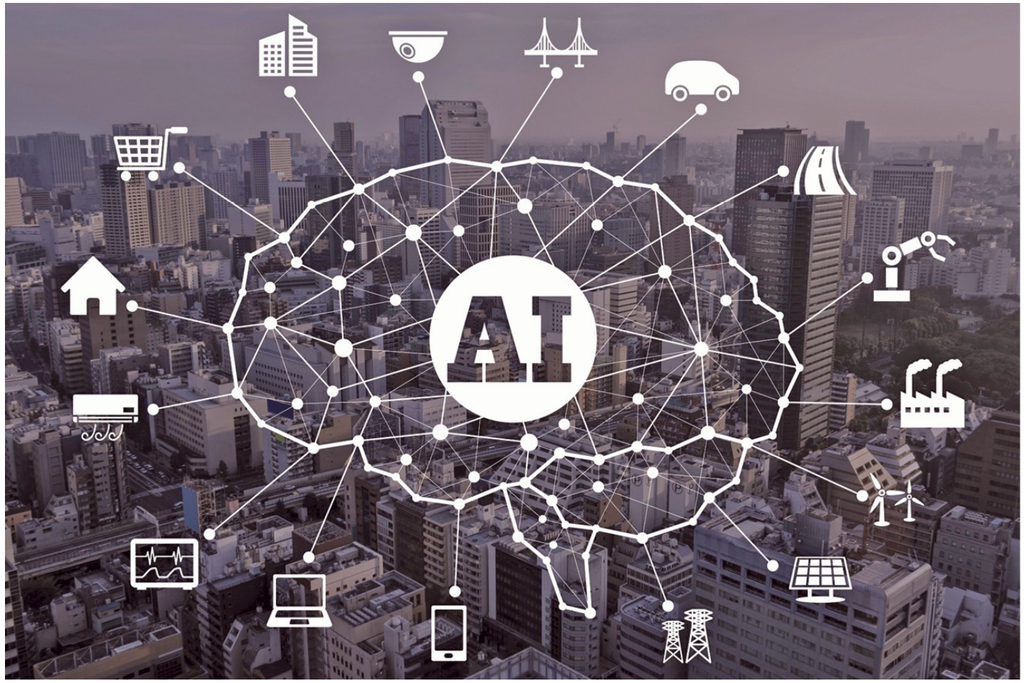
“What I cannot create I do not understand.”

ENGINEERING ARGUMENT

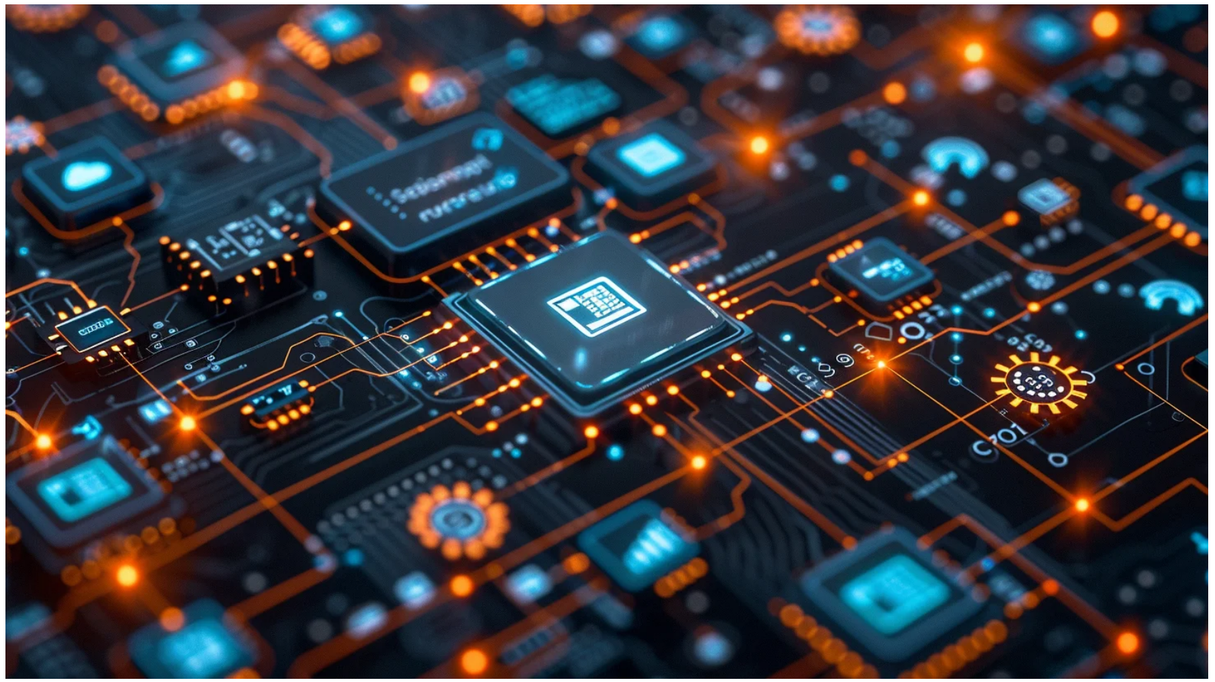
“There’s plenty of room at the bottom.”

NEUROMORPHIC APPLICATIONS

Adaptive embedded edge devices



Heterogeneous computing for efficient AI

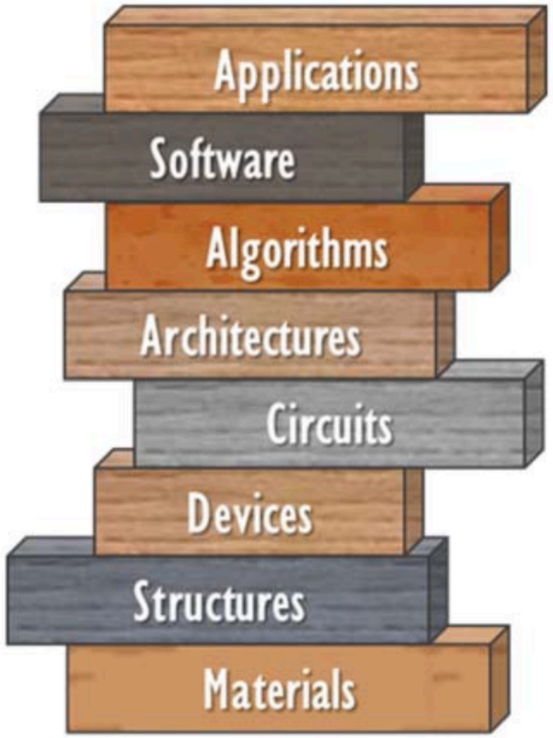


- ➡ Less energy consumption (green AI)
- ➡ Less heat generation
- ➡ Higher throughput
- ➡ Independent of the cloud (edge AI)

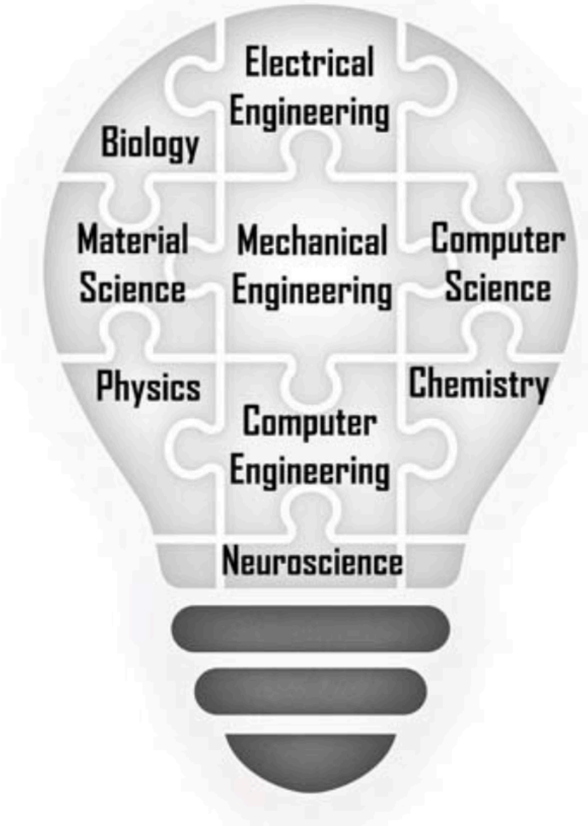
NEUROMORPHIC COMPUTING ROADMAP

Holistic Optimal Solutions

Driven by Hardware/Software Co-Optimization

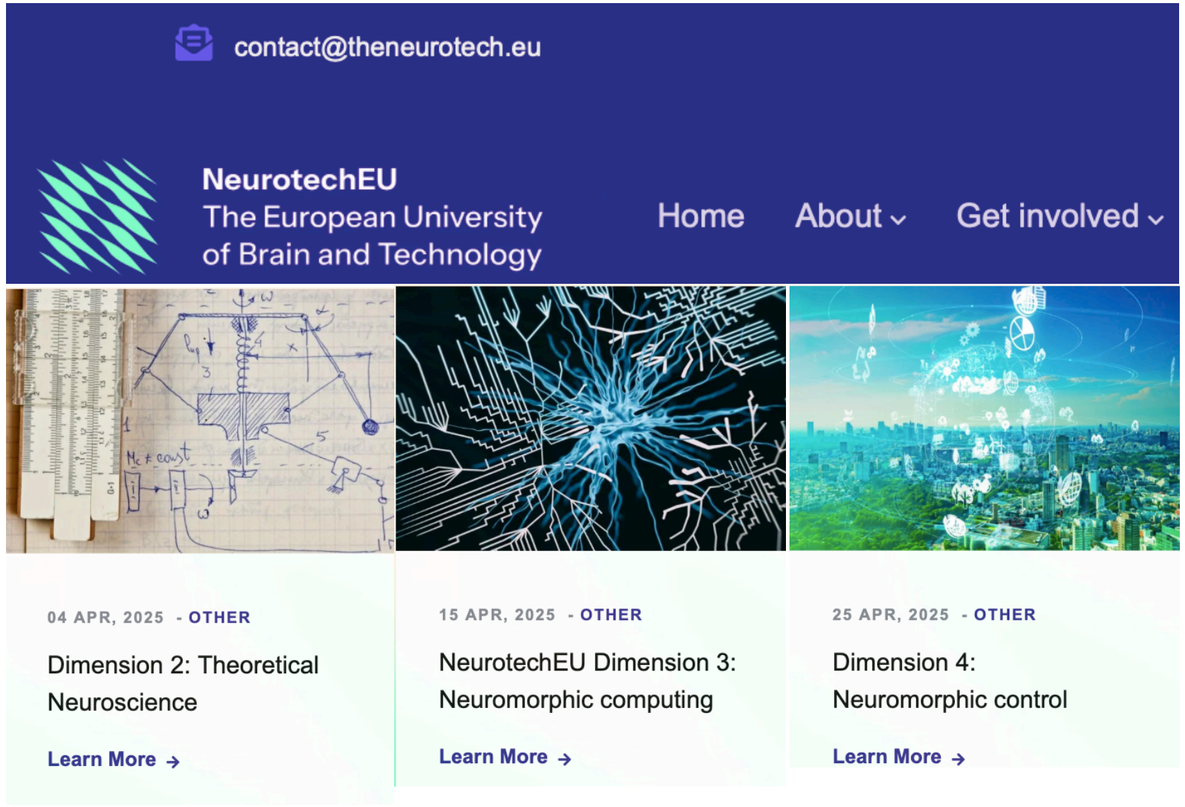


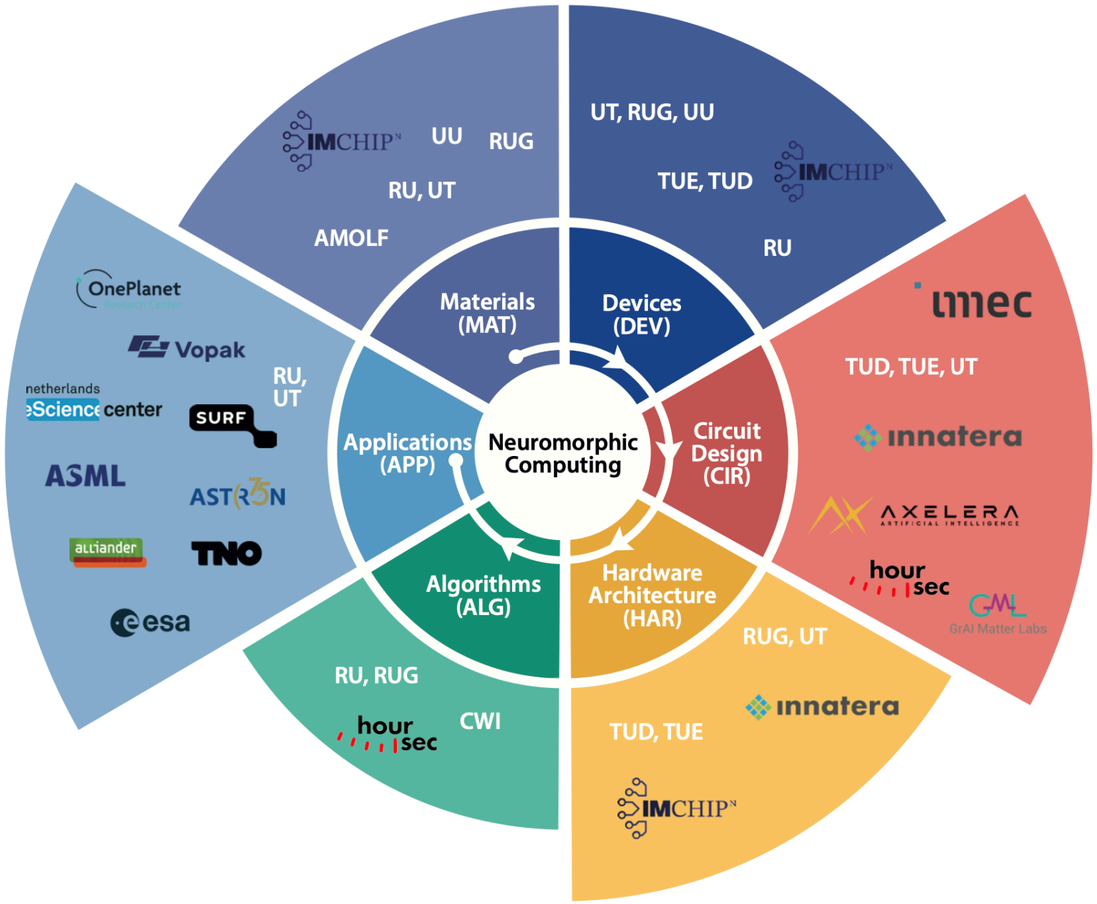
Interlocked Multidisciplinary Research



https://www.src.org/about/decadal-plan/

NEUROMORPHIC COMPUTING IN EUROPE AND THE NETHERLANDS





White paper highlights potential of brain-inspired computing

28 November 2024 • Research news item

NEUROMORPHIC COMPUTING @ RADBOUD UNIVERSITY & DONDERS INSTITUTE

Radboud Universiteit

Neuromorphic computing

The most efficient computer we know is our brain. The researchers at Radboud University who work on neuromorphic computing translate this complex network of synapses to an energy-efficient computing infrastructure. Nijmegen has a unique position with expertise from both the Donders Institute for Brain, Cognition and Behavior and the Institute for Molecules and Materials. These combined forces result in fundamentally new hardware designs and software approaches, with significant energy savings in comparison with conventional computing.

Coordinators

Prof. A.A. Khajetoorians (Alex) →

professor (Scanning Probe Microscopy)

Alexander Khajetoorians focuses on the electronic and magnetic properties of materials at the scale of a single atom. Using the Netherlands highest resolution microscope, he aims to understand and utilise the undiscovered properties of materials for new computing paradigms.

Prof. M.A.J. van Gerven (Marcel) →

professor (Artificial Intelligence), Principal Investigator (Donders Centre for Cognition), Principal Investigator (Donders Institute for Brain, Cognition and Behaviour)

Marcel van Gerven focuses on the development of artificial intelligence (AI) that is inspired by the way in which the human brain works. This provides the basis for smarter and more efficient brain-based technology.

Dr J.H. Mentink (Johan)

assistant professor (Ultrafast Spectroscopy of Correlated Materials)

Johan Mentink is an expert in the field of magnetism. He researches new magnetic effects so that future technologies will be able to store and process information much more quickly and economically while requiring considerably less processing capacity.

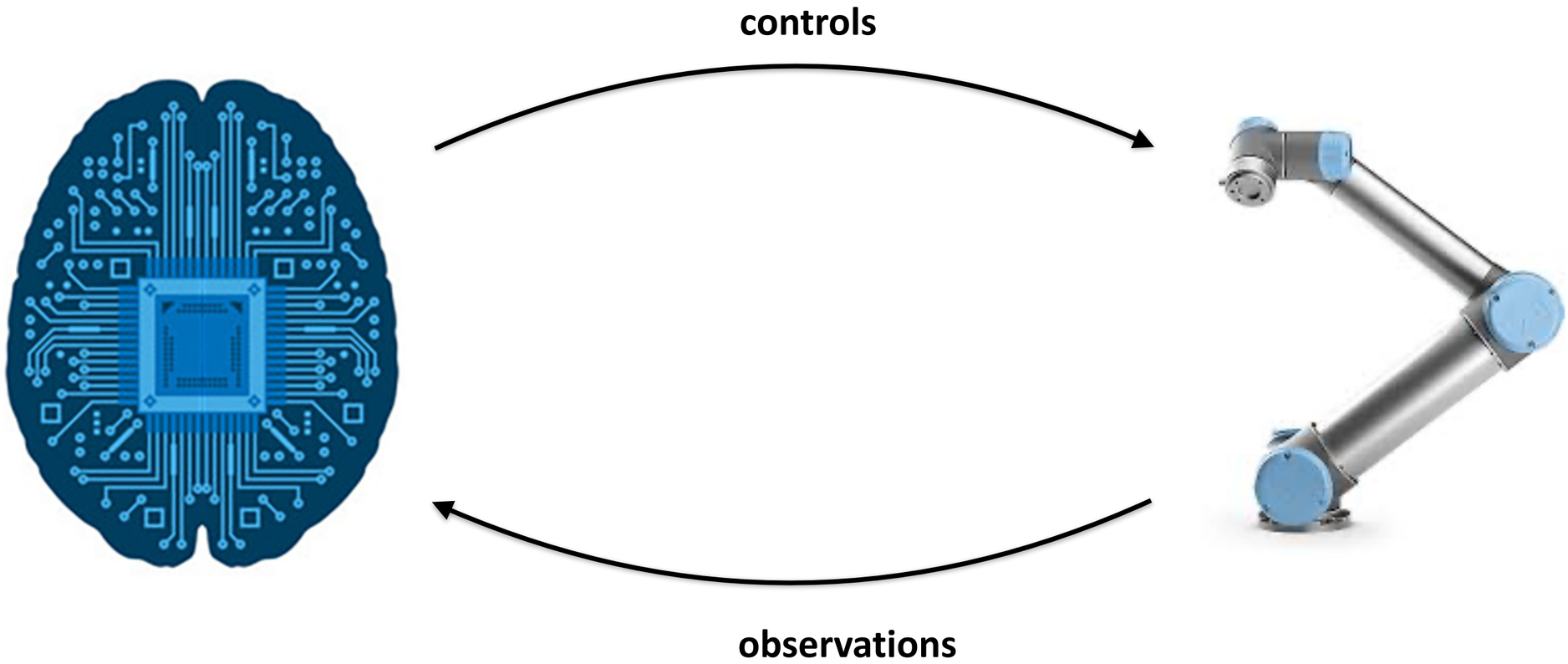
Donders Institute for Brain, Cognition and Behaviour

Donders Challenge: Neuromorphic Computing

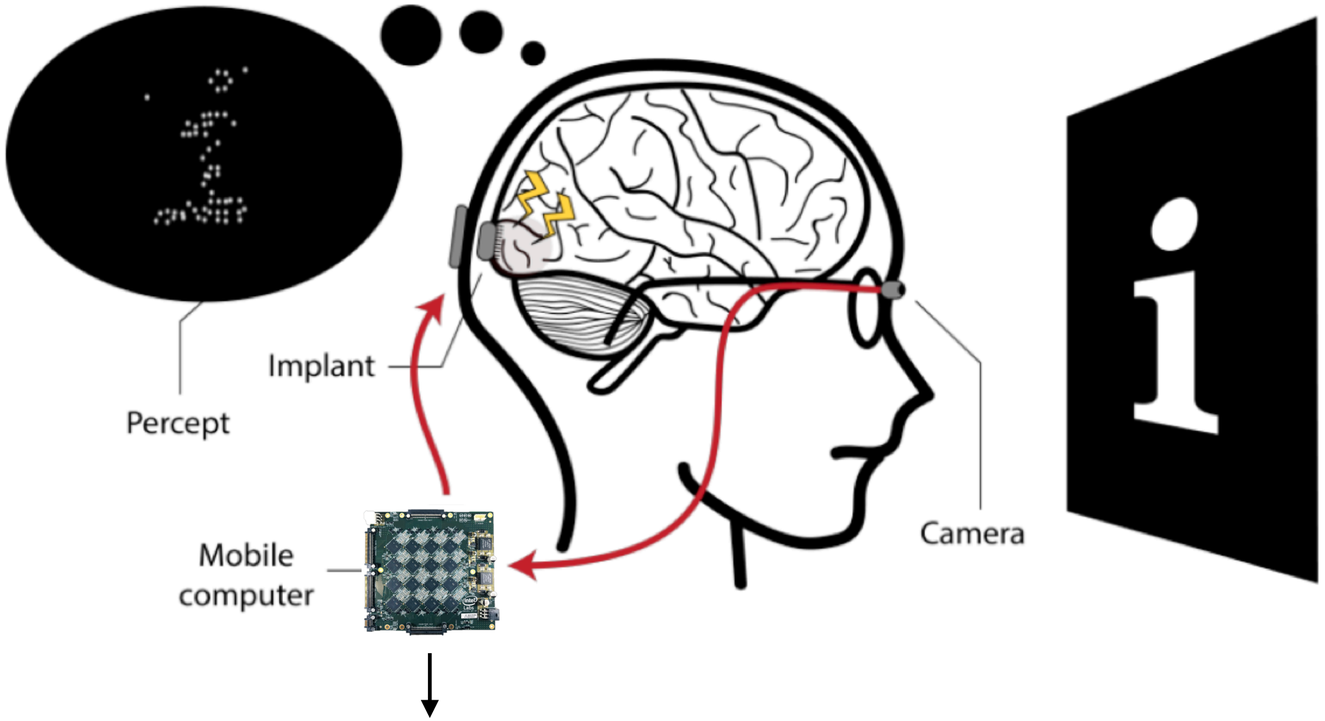
Project member(s)

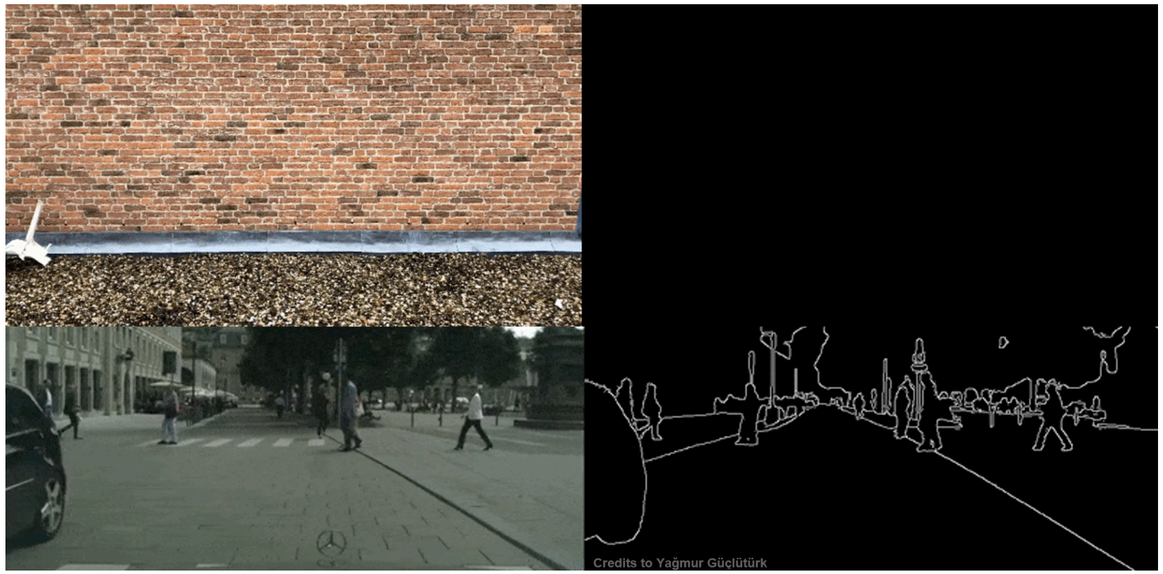
Prof. M.A.J. van Gerven (Marcel) , Dr M. Shahsavari (Mahyar) , Dr S.W. Keemink (Sander) , Dr N. Ahmad (Nasir) , Dr M.Y. Ben Zion (Matan Yah) , Dr T. Kachman (Tal) , Dr Y. Qin (Yuzhen) , U.C. Altin (Umut) , Prof. A.C.C. Coolen (Ton) , Prof. H.J. Kappen (Bert) , Prof. P.H.E. Tiesinga (Paul) , Dr F. Zeldenrust (Fleur) , Dr A. Ingrosso (Alessandro) , Prof. J.H.P. Kwisthout (Johan) , Dr S. Thill (Serge) , Prof. F.P. Battaglia (Francesco) , Dr Y. Qin (Yuzhen)

NEUROMORPHIC CONTROL



MOTIVATING EXAMPLE: NEUROTECHNOLOGY



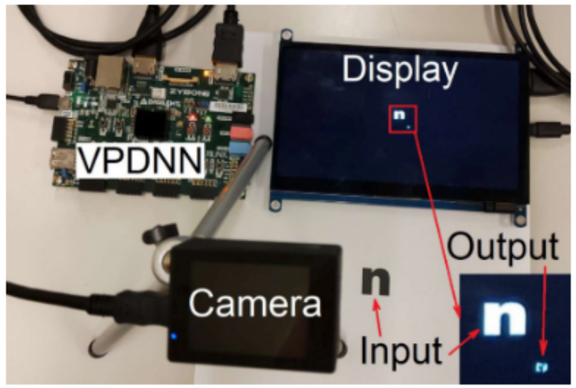


de Ruyter van Steveninck et al. J. Vision. 2020; 2024); van der Grinten et al. *eLife* 13:e85812

neuromorphic chips for optimal neural control

Advantages:

- Portable
- Battery-efficient
- High-throughput
- Privacy-preserving
- Theoretical alignment

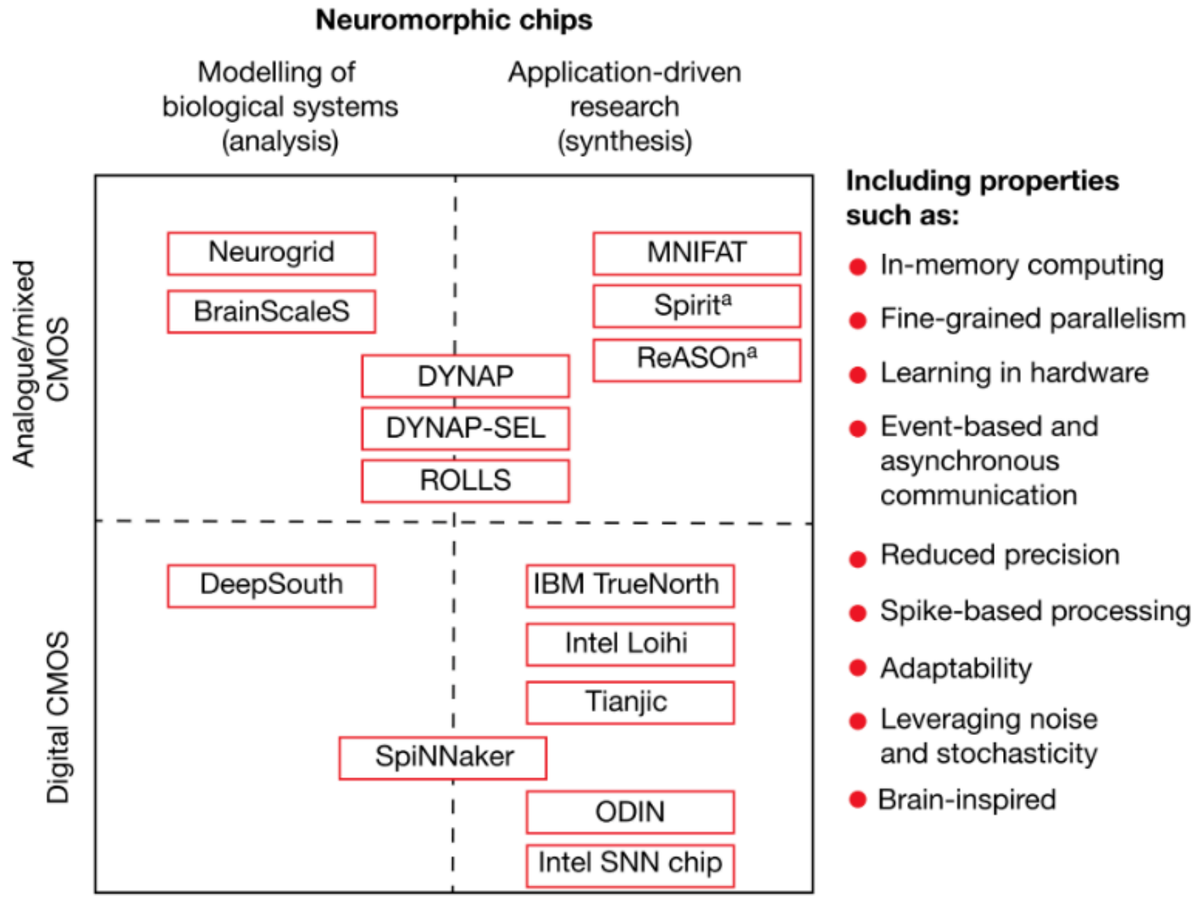


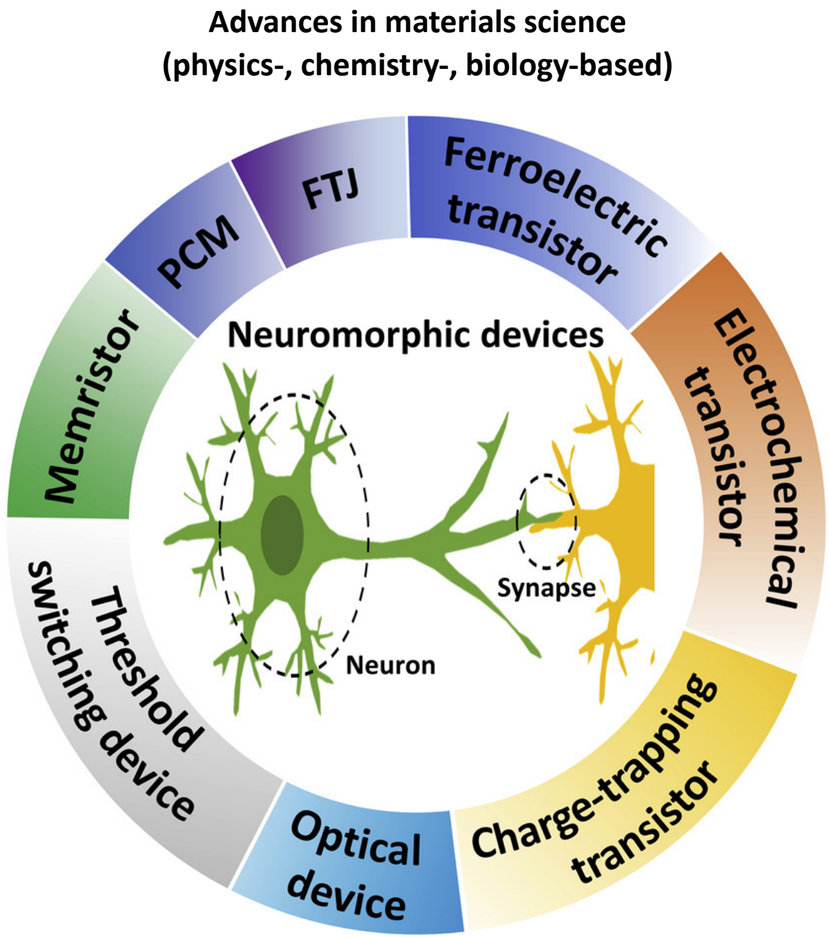
Mohamed, H., Raducanu, B., Kiselev, I., Wang, Z., Kucukoglu, B., Rueckauer, B., van Gerven, M., Mora Lopez, C., & Liu, S.-C. (2023). A 128-channel real-time VPDNN stimulation system for a visual cortical neuroprosthesis. The 2023 IEEE Biomedical Circuits and Systems Conference (BioCAS).

Similar arguments hold for:

- Healthcare
- High-tech industry
- Semiconductor manufacturing
- Autonomous systems
- Automotive
- Space
- …

NEUROMORPHIC SUBSTRATES

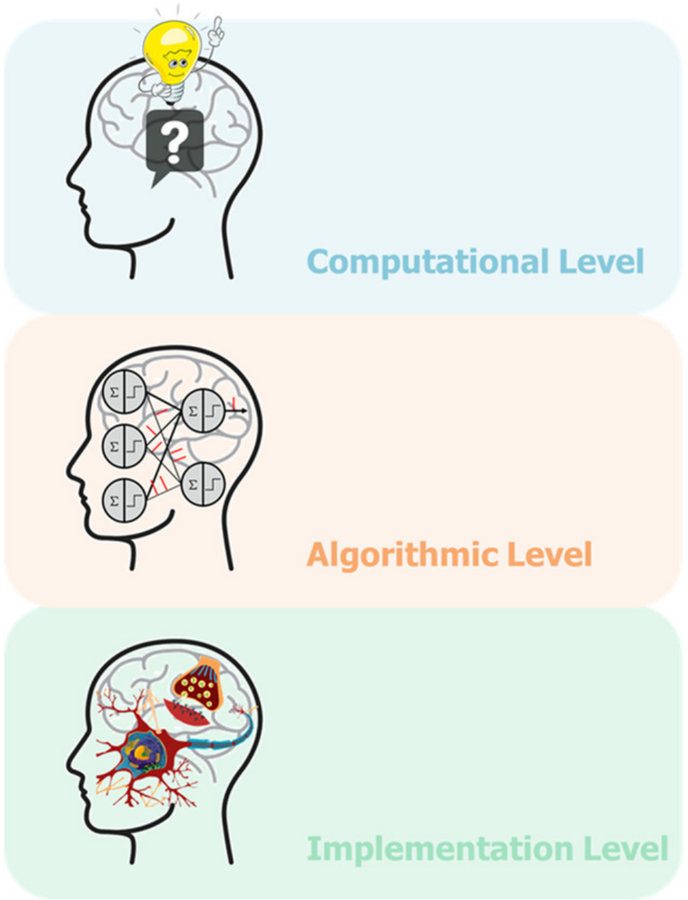




Mehonic, A., & Kenyon, A. J. (2022). Brain-inspired computing needs a master plan. In Nature (Vol. 604, Issue 7905, pp. 255–260).
Kim, M. K., Park, Y., Kim, I. J., & Lee, J. S. (2020). Emerging Materials for Neuromorphic Devices and Systems. IScience, 23(12), 101846.

MARR'S THREE LEVELS OF ANALYSIS

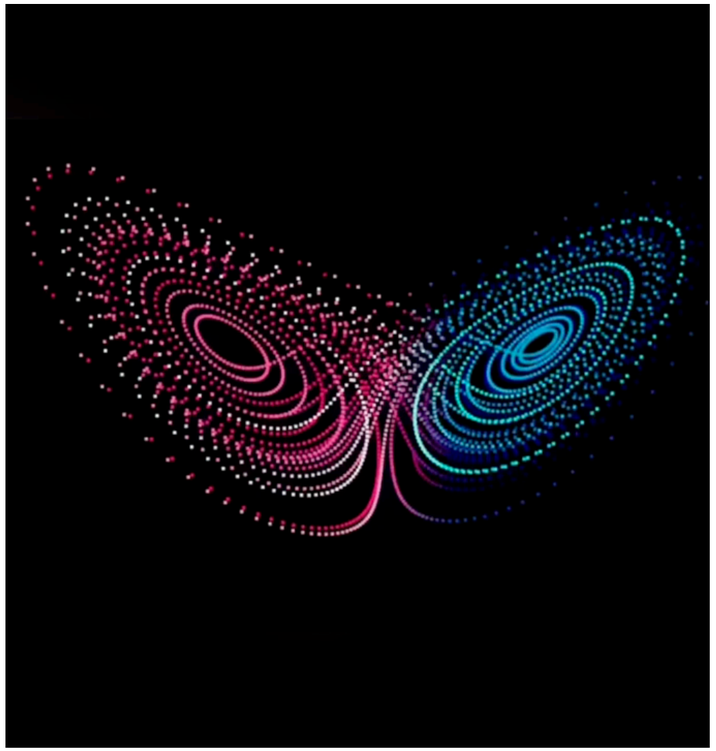
But what are the **algorithms** that should run **in** neuromorphic substrates to mimic natural intelligence?



Guo, Y., Zou, X., Hu, Y., Yang, Y., Wang, X., He, Y., Kong, R., Guo, Y., Li, G., Zhang, W., Wu, S., & Li, H. (2021). A Marr's Three-Level Analytical Framework for Neuromorphic Electronic Systems. Advanced Intelligent Systems, 3(11), 2100054. https://doi.org/10.1002/AISY.202100054

DYNAMICAL SYSTEMS THEORY

Dynamical systems theory provides the ideal common language for the development of neuromorphic computing



A dynamical system is a rule for time evolution on a state space, which can be described in terms of (a system of) differential equations

DYNAMICAL SYSTEMS THEORY

We will assume that the equations of motion of a neuromorphic system can described in terms of a stochastic differential equation (SDE) given by

$$\mathrm{d}x = f(x)\,\mathrm{d}t + g(x)\,\mathrm{d}W \quad \text{given the initial state } x(0) = x_0$$

state vector at time t

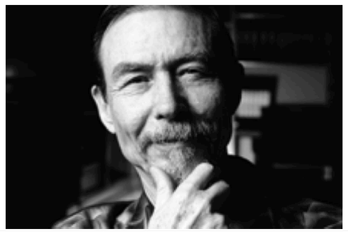
drift function

diffusion function

Brownian noise

The processes of drift and diffusion are the stuff of which all information processing devices—both neural and semiconductor—are made.

—Carver Mead (1989)

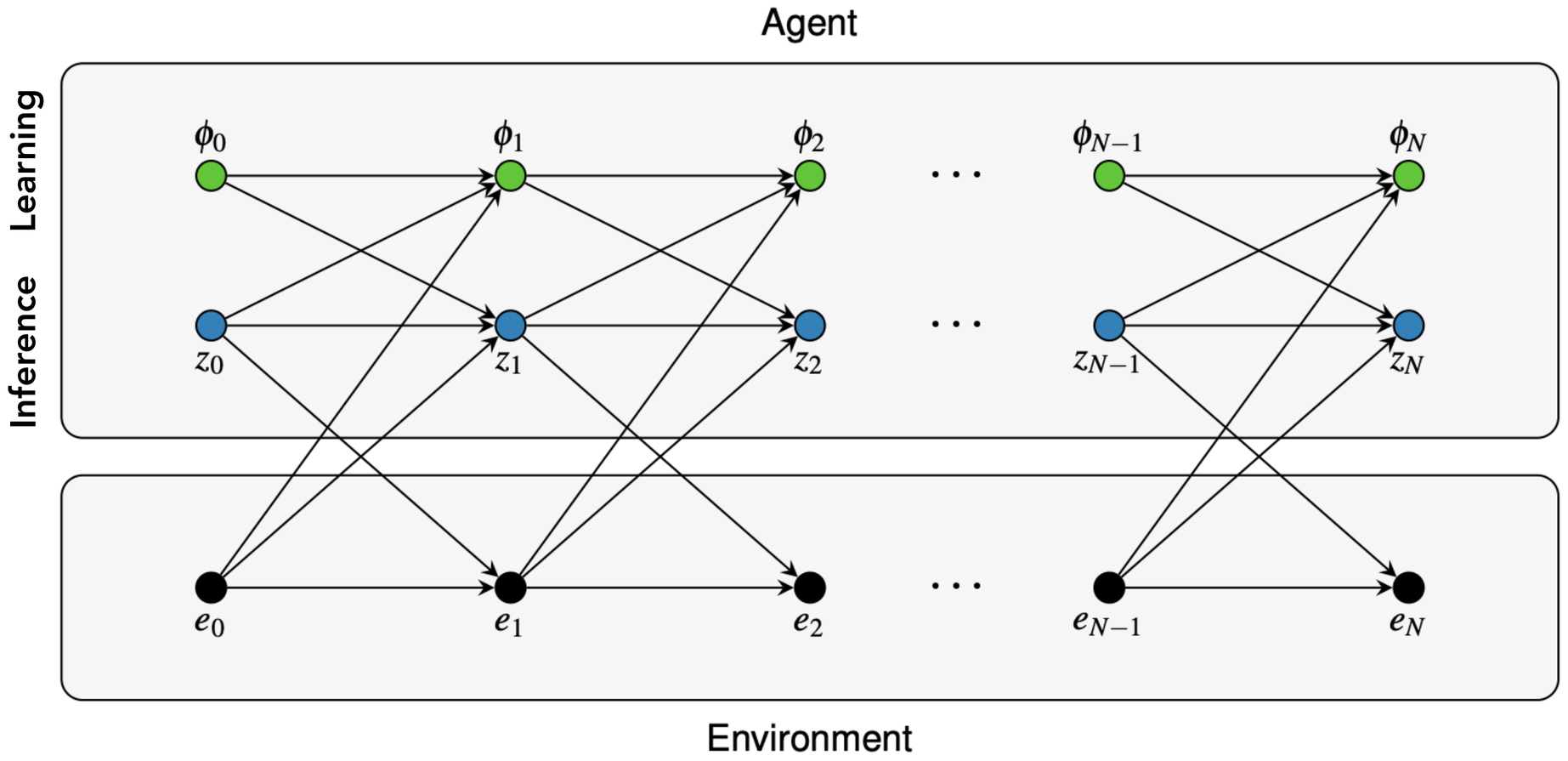


OUTLINE

- NEUROMORPHIC COMPUTING
- NOISE-BASED LEARNING
- GENETIC PROGRAMMING
- CONCLUSIONS

LEARNING FROM EXPERIENCE

How to set up the equations of motion such as to induce adaptation in an agent?



Very different from learning in conventional AI systems using backpropagation through time!

ORNSTEIN-UHLENBECK ADAPTATION

We consider **Ornstein-Uhlenbeck adaptation** (OUA) which exploits **noise as a mechanism for learning**

García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

ORNSTEIN-UHLENBECK ADAPTATION

Define learning variables

$$\phi = (\theta, \mu, \hat{v})$$

with **parameters** $\theta=(\theta_1,\ldots,\theta_N)$, **means** $\mu=(\mu_1,\ldots,\mu_N)$ and **value estimate** $\hat{v}(t,x)$

Define

$$\delta = v(t,x) - \hat{v}(t,x)$$

as the **reward prediction error** with $v(t,x)$ the true value

García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

ORNSTEIN-UHLENBECK ADAPTATION

OUA induces learning via the following dynamics:

$$\mathrm{d}\theta_i = \lambda(\mu_i - \theta_i)\mathrm{d}t + \sigma\mathrm{d}W \quad \text{for } i=1,\ldots,N$$

$$\mathrm{d}\mu_i = \eta\delta(\theta_i - \mu_i)\mathrm{d}t \quad \text{for } i=1,\ldots,N$$

$$\mathrm{d}\hat{v} = \rho\delta\mathrm{d}t$$

- Parameters θ of our model evolve according to an Ornstein-Uhlenbeck process
- Mean vector μ follows its own dynamics modulated by reward prediction error δ
- Value (estimate) simplified to the instantaneous reward (estimate)

García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

OUA CONTROL OF A STOCHASTIC DOUBLE INTEGRATOR (SDI)

Environment (SDI):

$$\mathrm{d}p = v\mathrm{d}t$$
$$\mathrm{d}v = (-\gamma v + u)\,\mathrm{d}t + \epsilon \mathrm{d}W$$

with control $u = \tanh(Cz)$

Agent (CTRNN):

$$\mathrm{d}z = \tau^{-1} \circ (-z + As(z + b) + y)\,\mathrm{d}t$$

with observations $y = Be$ where $e = \mathrm{vec}(p, v)$

Reward function:

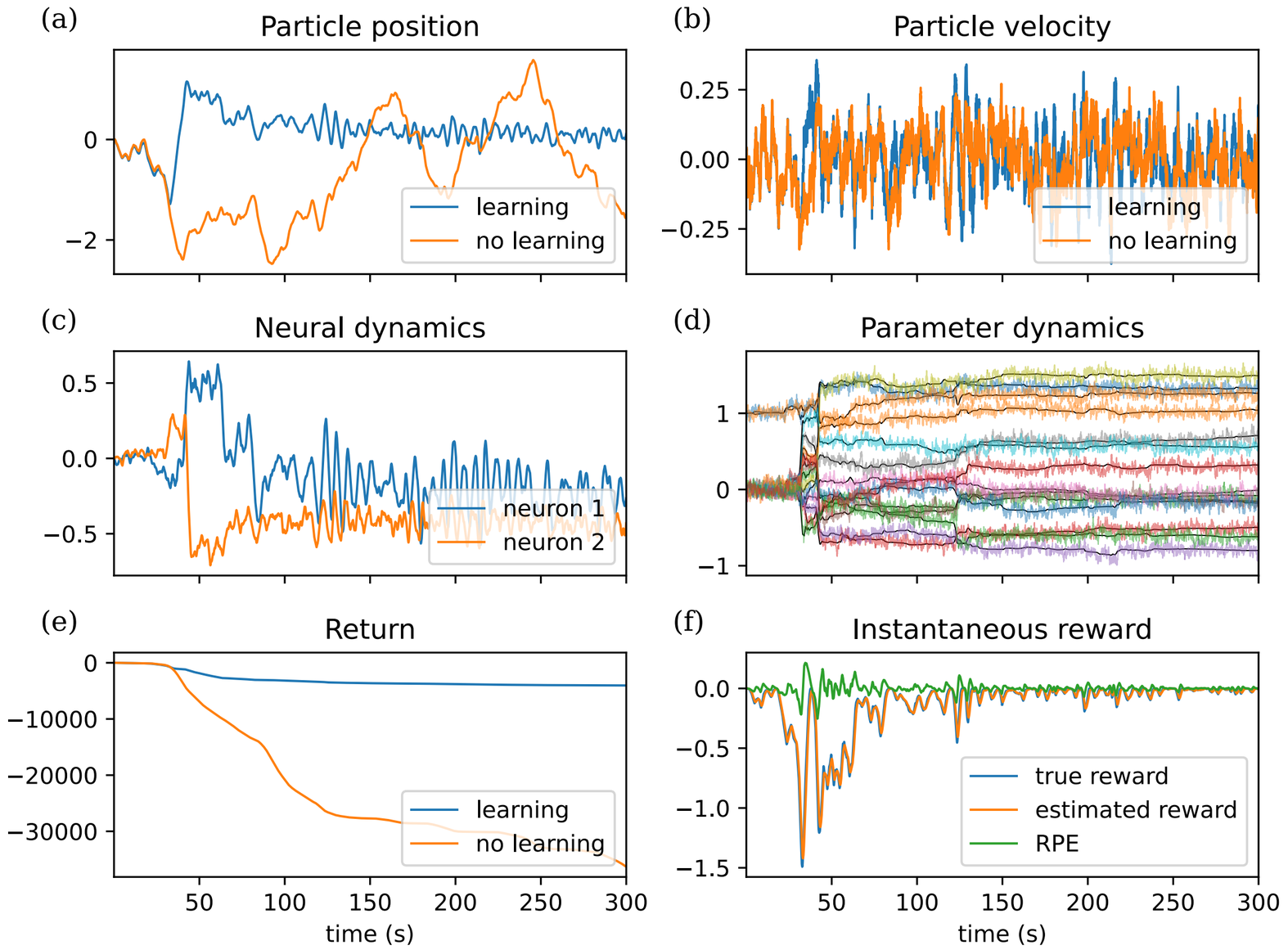
$$r(t) = -0.9p(t)^2 - 0.1u(t)^2$$

Parameters:

$$\theta = (\tau, b, A, B, C)$$

García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

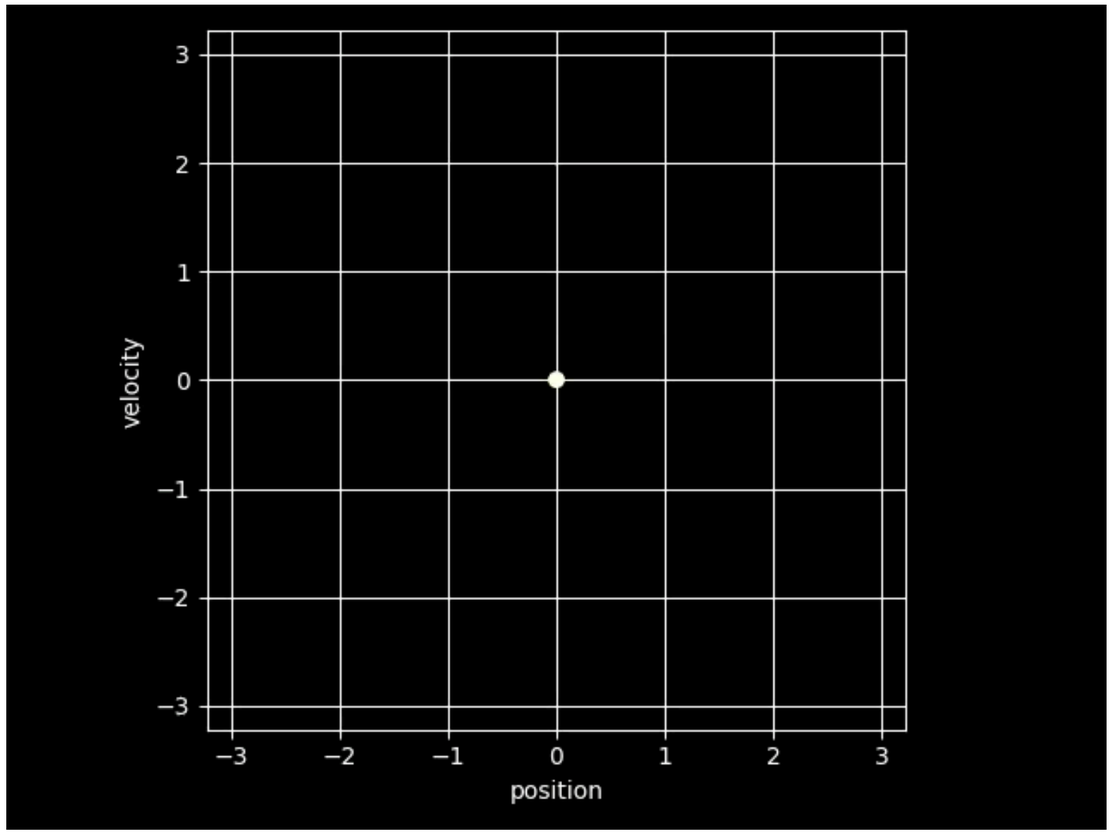
OUA CONTROL OF A STOCHASTIC PARTICLE



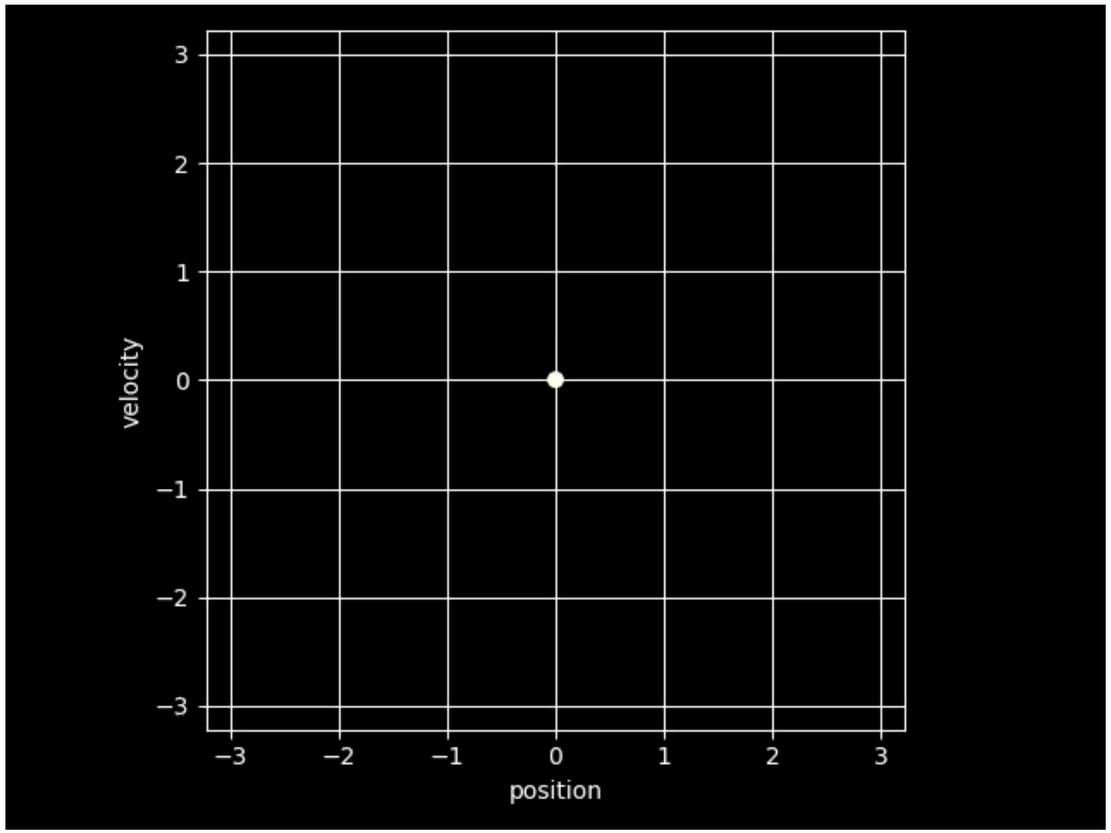
García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

OUA CONTROL OF A STOCHASTIC PARTICLE

Particle dynamics **without** learning



Particle dynamics **with** learning



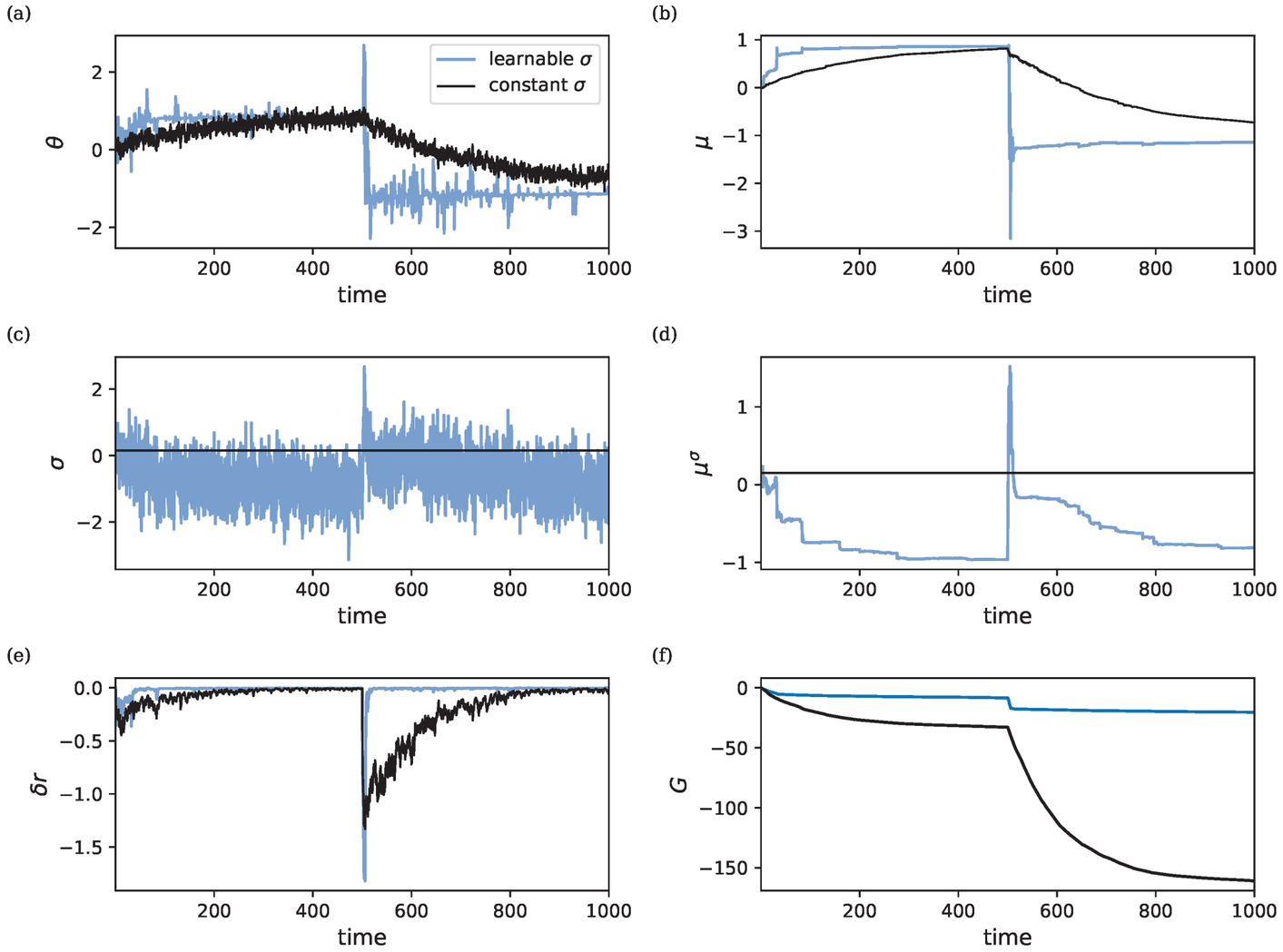
García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

SOLVING THE EXPLORATION-EXPLOITATION DILEMMA

OUA learning of the noise variance

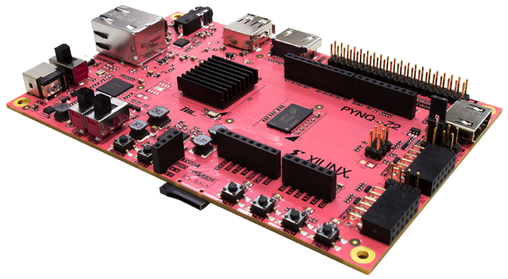
$$\mathrm{d}\sigma = \lambda^{\sigma}(\mu^{\sigma} - \sigma)\,\mathrm{d}t + \rho\,\mathrm{d}W$$
$$\mathrm{d}\mu^{\sigma} = \eta^{\sigma}\delta(\sigma - \mu^{\sigma})\,\mathrm{d}t$$

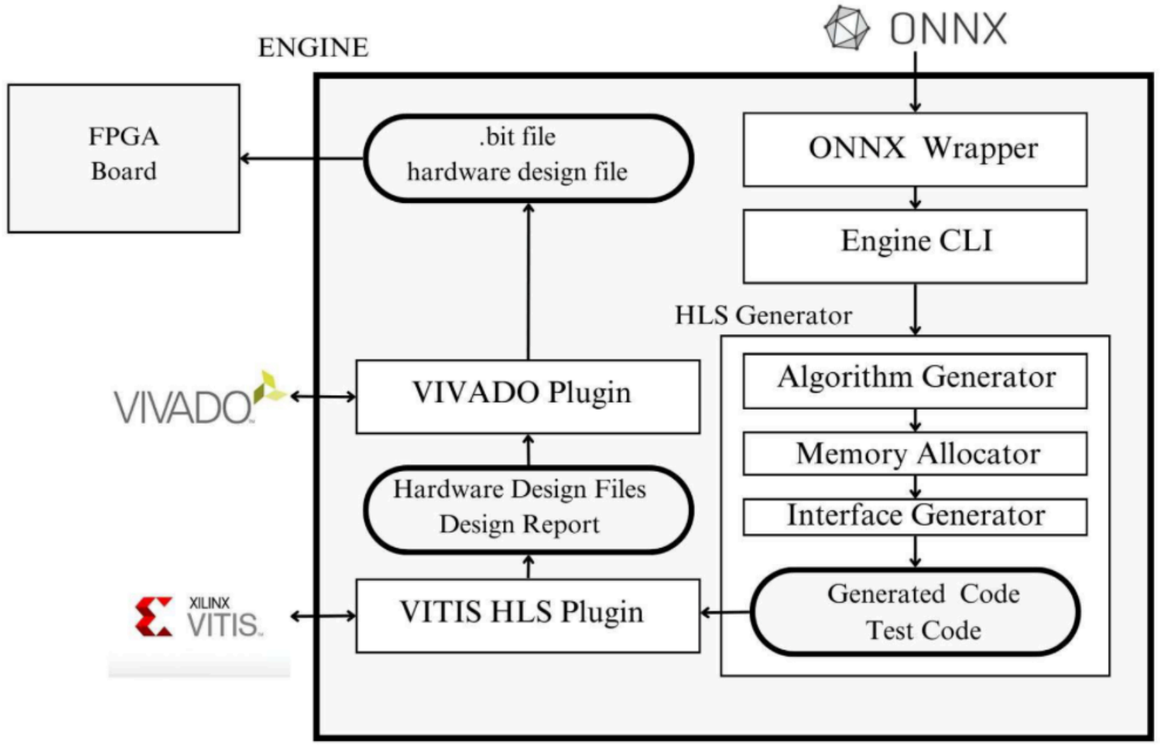
$$\mathrm{d}\theta_i = \lambda(\mu_i - \theta_i)\mathrm{d}t + \sigma\mathrm{d}W \quad \text{for } i=1,\ldots,N$$
$$\mathrm{d}\mu_i = \eta\delta(\theta_i - \mu_i)\mathrm{d}t \quad \text{for } i=1,\ldots,N$$
$$\mathrm{d}\hat{v} = \rho\delta\mathrm{d}t$$

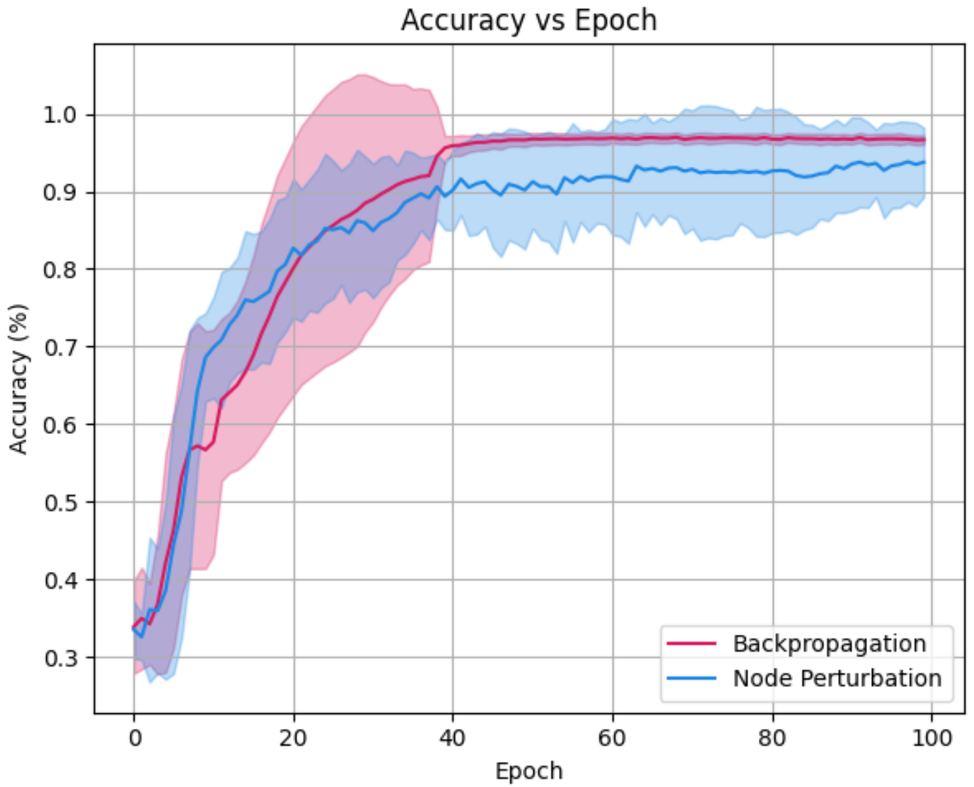


García Fernández, J., Ahmad, N., & van Gerven, M. (2024). Ornstein-Uhlenbeck adaptation as a mechanism for learning in brains and machines. Entropy, 26(12), 1125. https://doi.org/10.3390/e26121125

TOWARDS ADAPTIVE INTELLIGENT CHIPS

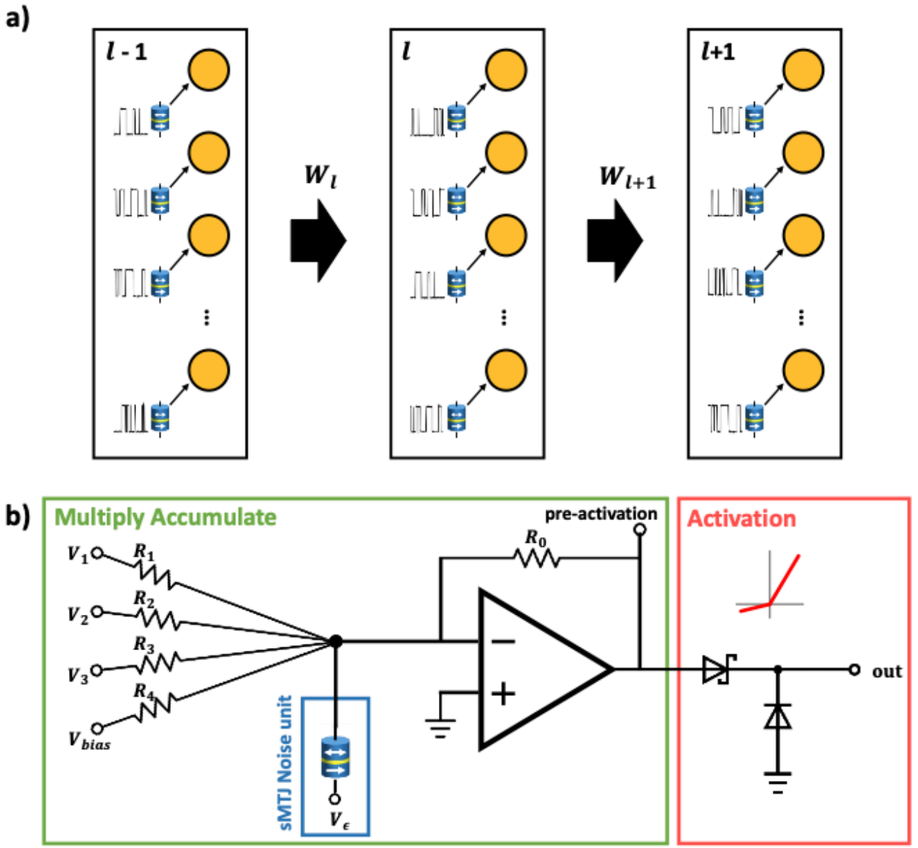


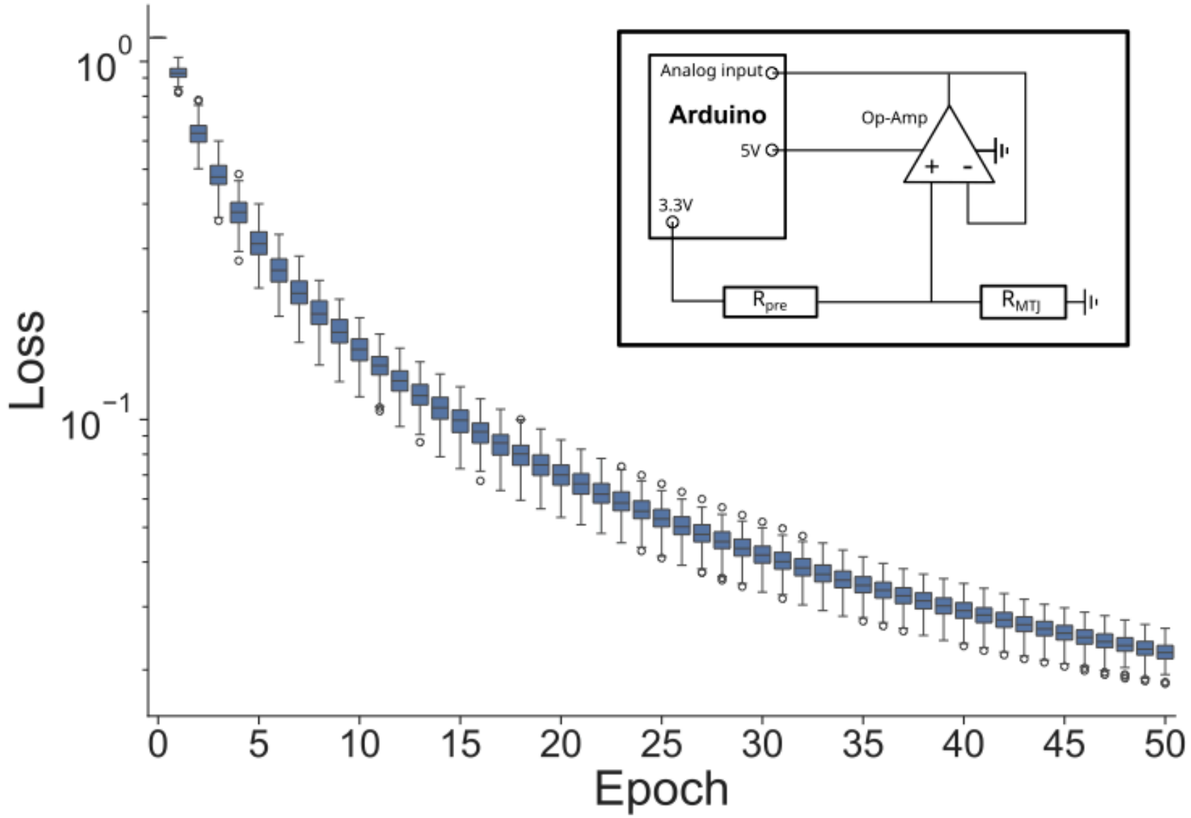




https://github.com/umutcanaltin/fpgai_compiler/blob/main/paper/assets/hls.pdf

TOWARDS PHYSICAL LEARNING MACHINES





Koenders, K., Schnitzpan, L., Kammerbauer, F., Shu, S., Jakob, G., Kläui, M., Mentink, J. H., Ahmad, N., & van Gerven, M. (2024).
Noise-based Local Learning using Stochastic Magnetic Tunnel Junctions.Physical Review Applied (accepted for publication)

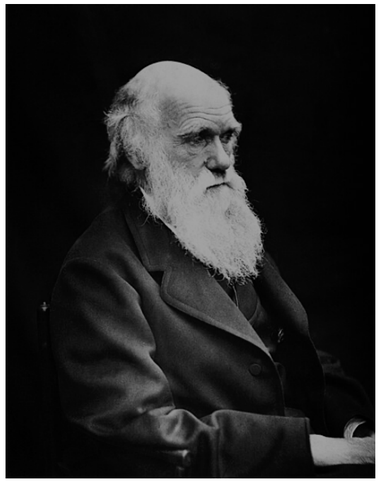
OUTLINE

- NEUROMORPHIC COMPUTING
- NOISE-BASED LEARNING
- GENETIC PROGRAMMING
- CONCLUSIONS

EVOLUTION

In nature, adaptation takes place at different timescales:

- **Ontogenetic** timescale: adaptation during the lifetime of an individual (i.e. learning)
- **Phylogenetic** timescale: adaptation across generations of individuals (i.e. evolution)



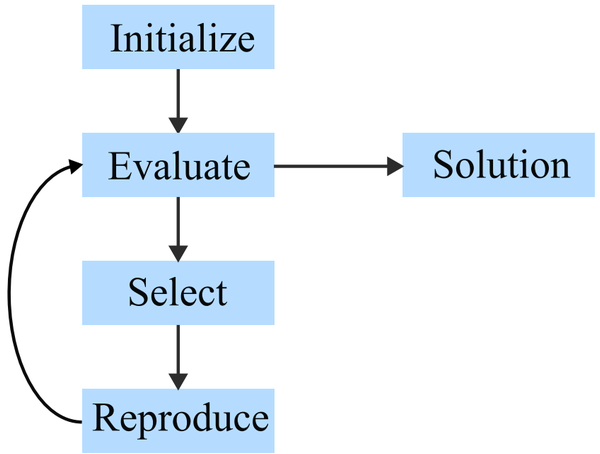
"Nothing in Biology Makes Sense Except in the Light of Evolution" (Dobzhansky, 1973)

EVOLUTIONARY COMPUTATION

Evolutionary computation is a family of algorithms for **black-box optimization** (a.k.a. gradient-free optimization) inspired by biological evolution

Example of a generic evolutionary algorithm:

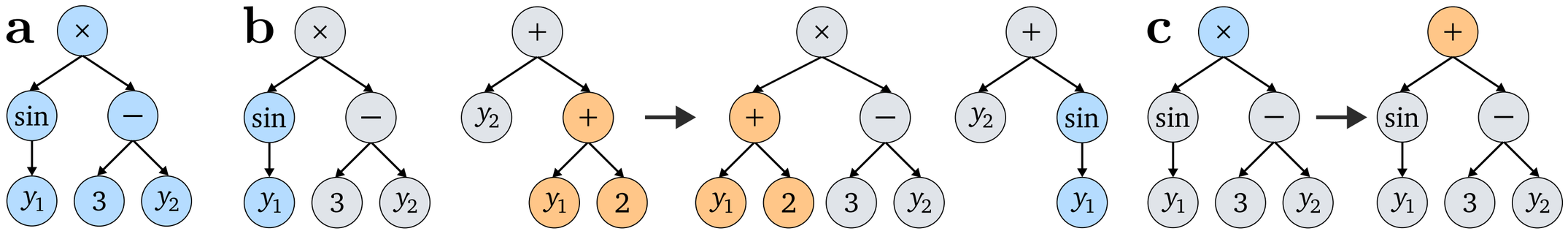
1. Randomly generate the initial population of individuals
2. Evaluate the fitness of each individual in the population.
3. Check, if the goal is reached and the algorithm can be terminated.
4. Select individuals as parents, preferably of higher fitness.
5. Produce offspring with optional crossover (mimicking reproduction).
6. Apply mutation operations on the offspring.
7. Select individuals preferably of lower fitness for replacement with new individuals (mimicking natural selection).
8. Return to 2



Many different variants: Genetic algorithm, evolution strategies, evolutionary programming, differential evolution, neuroevolution, gene expression programming, grammatical evolution, genetic programming, Cartesian genetic programming, …

GENETIC PROGRAMMING

Genetic programming (GP) is an evolutionary algorithm that evolves populations of **programs** represented as computational graphs



- First proposed by Turing in Computing Machinery and Intelligence
- Explored by students of John Holland (pioneer in genetic algorithms)
- Developed further by John Koza including four books on the topic

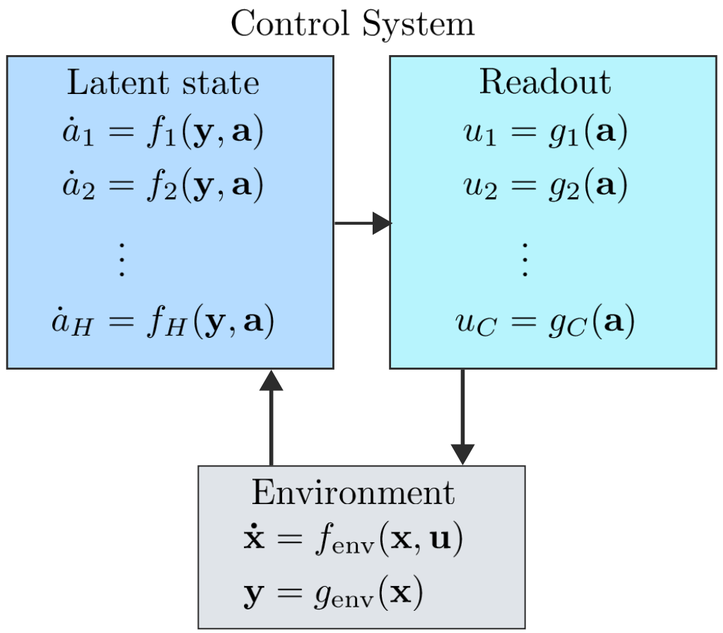
GENETIC PROGRAMMING

Can we use genetic programming to evolve intelligent agents?

Define agent dynamics by $\mathrm{d}a = f(a, y)\mathrm{d}t$ with observations y and control output $u = g(a)$

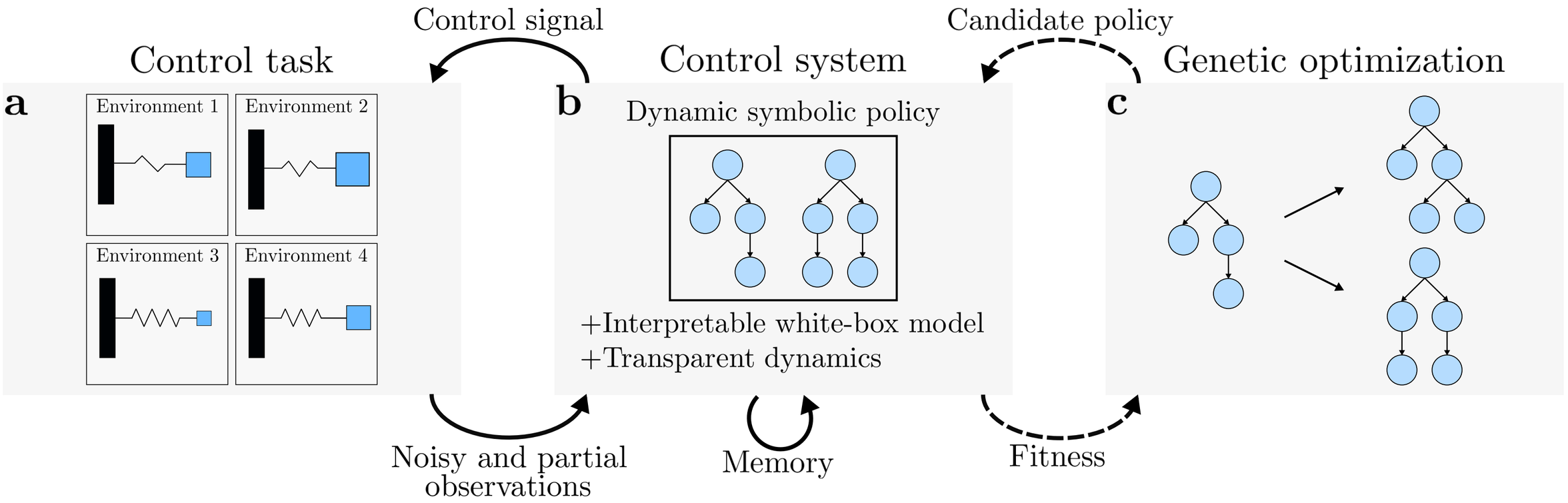
this is a program!

this is a program!



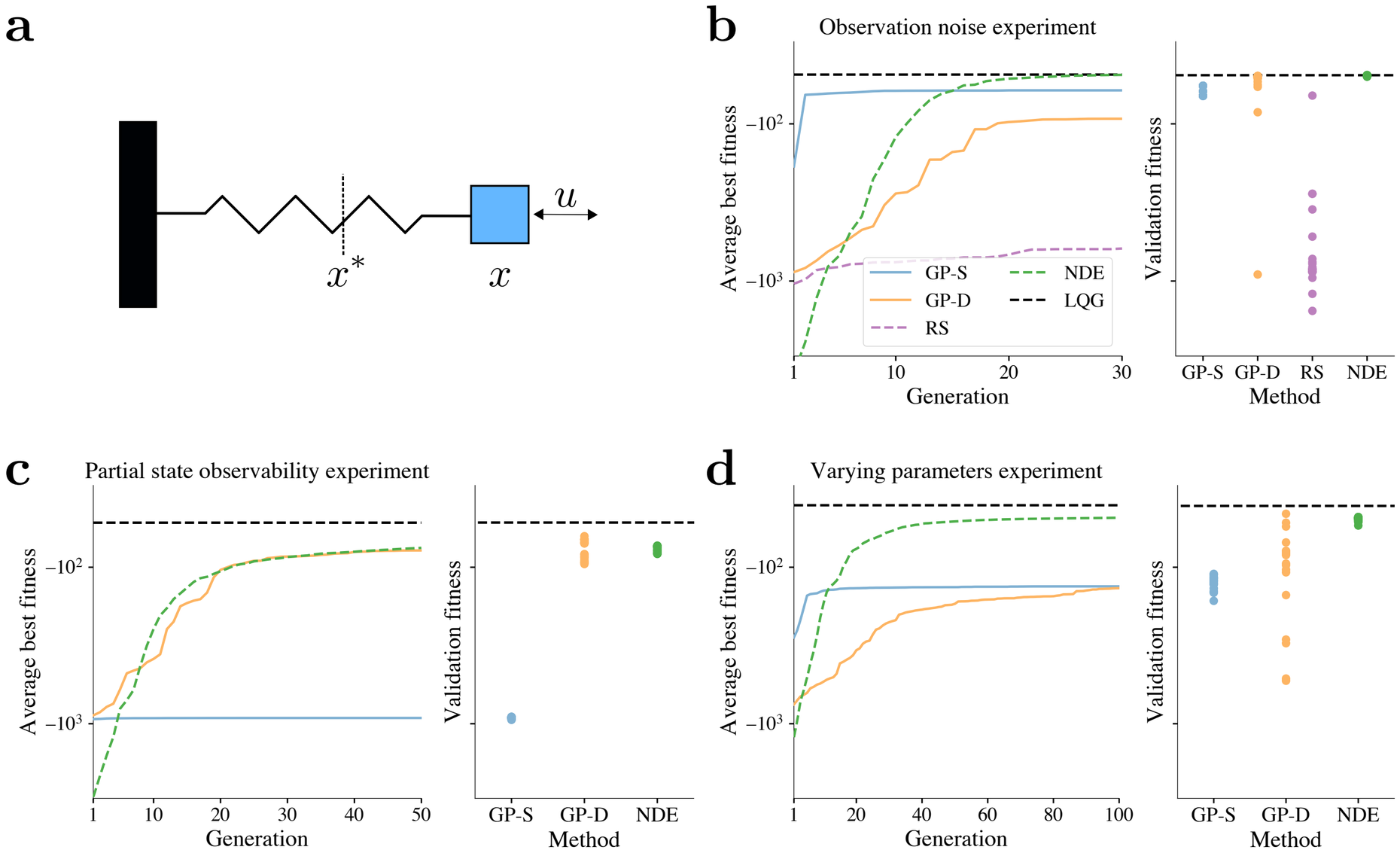
We learn symbolic policies from scratch!

GENETIC PROGRAMMING



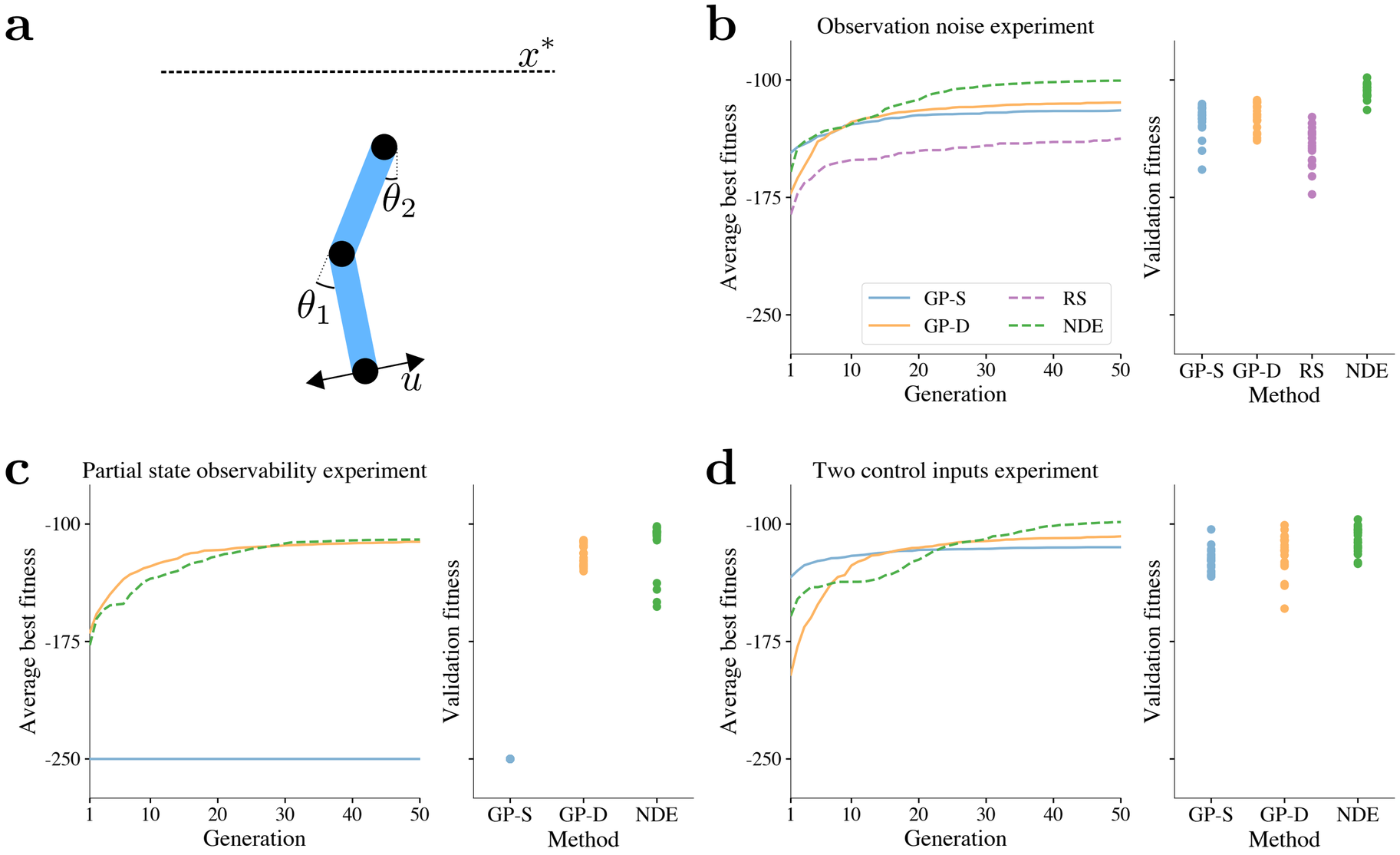
de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

CONTROLLING A STOCHASTIC HARMONIC OSCILLATOR



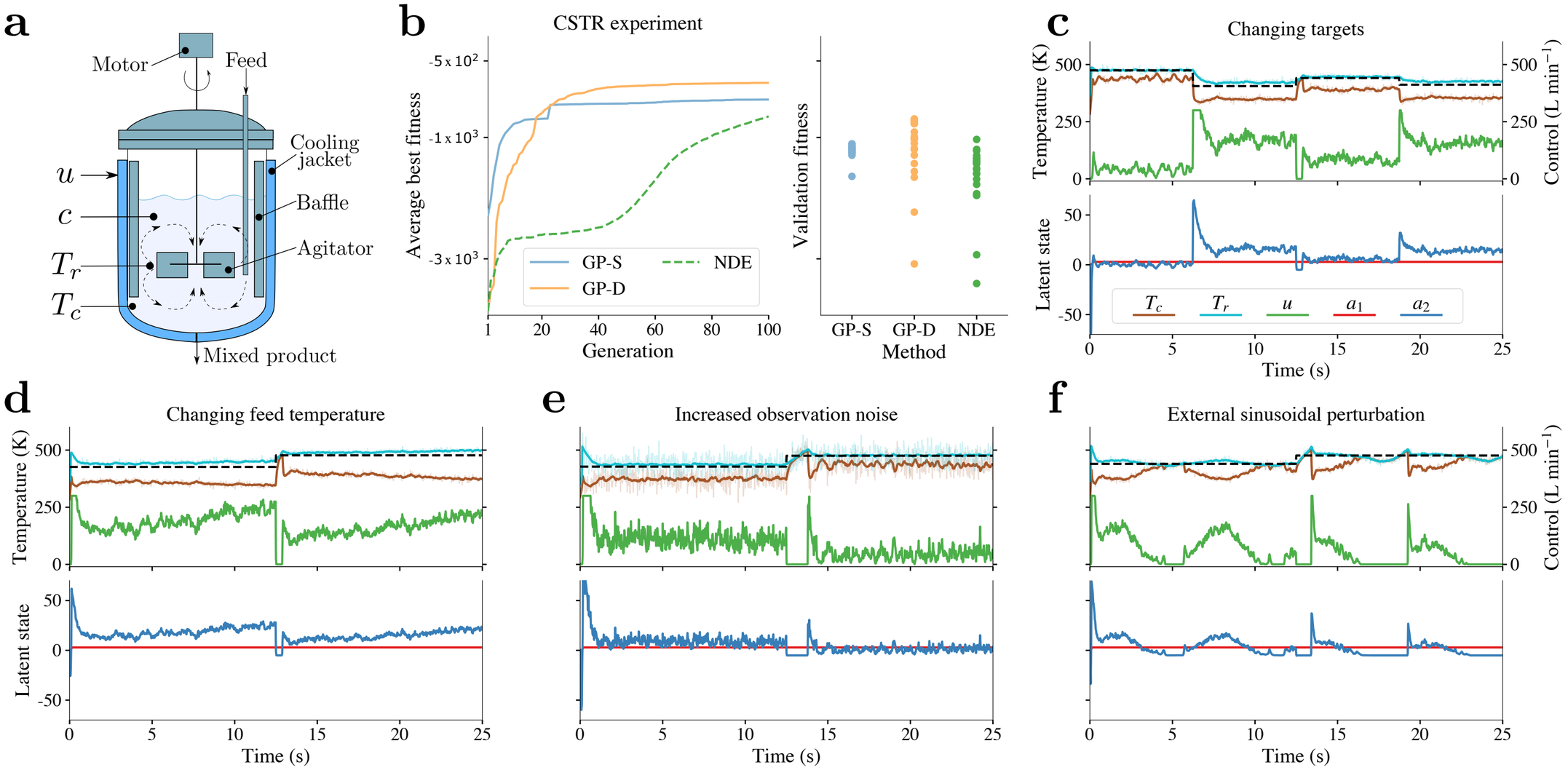
de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

CONTROLLING THE ACROBOT



de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

CONTROLLING A CONTINUOUS STIRRED TANK REACTOR



de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

PERFORMANCE COMPARISON

		RS		NDE		GP-S		GP-D	
Environment	**Experiment**	Fitness	Size	Fitness	Size	Fitness	Size	Fitness	Size
SHO	Observation noise	-56.11	15	**-48.85**	215	-57.28	5	-49.66	19
SHO	Partial state observability	-		-72.76	195	-903.44	5	**-63.35**	21
SHO	Varying parameters	-		-47.76	215	-110.25	13	**-45.56**	25
Acrobot	Observation noise	-123.64	6	**-98.45**	231	-115.31	8	-112.82	20
Acrobot	Partial state observability	-		**-101.47**	191	-250.0	1	-110.23	12
Acrobot	Two control inputs	-		**-97.02**	272	-103.43	12	-100.69	18
CSTR	Standard	-		-1017.72	215	-1059.31	9	**-845.33**	27

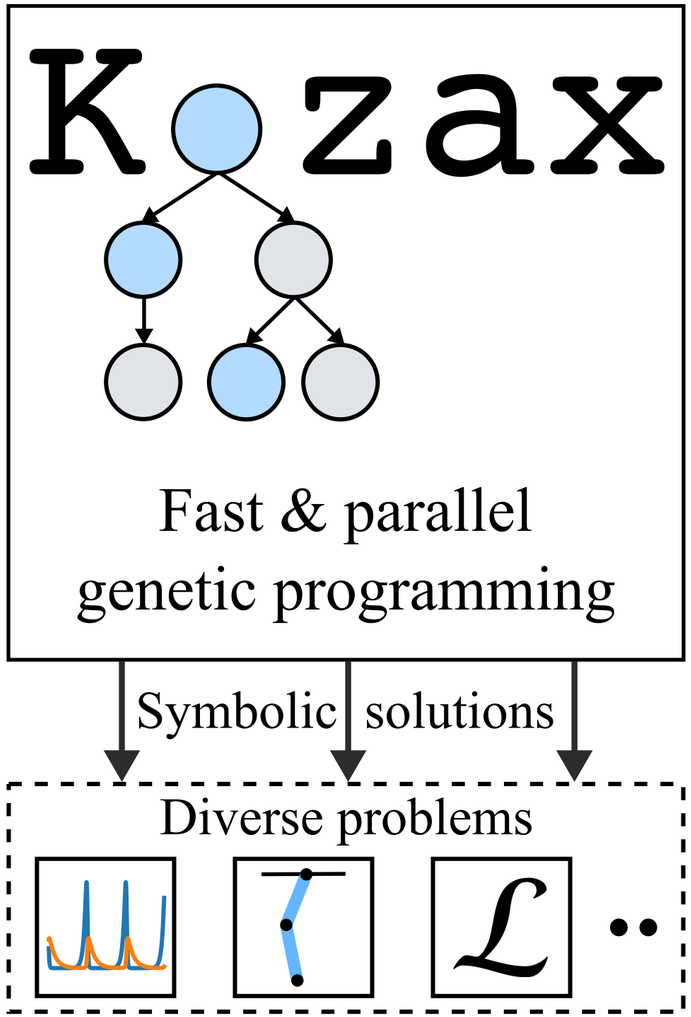
de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

LEARNT SYMBOLIC POLICIES

Environment	Experiment	Static policy	Dynamic policy
SHO	Observation noise	$u_1 = -0.61y_2 + x^*$	$u_1 = -2a_1 + 2.60a_2 + x^*$ $\dot{a}_1 = y_2$ $\dot{a}_2 = -u_1 + x^*$
SHO	Partial state observability	$u_1 = 0.75x^* - 0.11$	$u_1 = 0.45(a_1 + x^*)$ $\dot{a}_1 = 2a_2 - u_1 + x^*$ $\dot{a}_2 = -a_2 - 0.99u_1 + y_1$
SHO	Varying parameters	$u_1 = -1.10y_1 - 0.73y_2 + 1.83x^* + 0.17$	$u_1 = -a_1^3 - a_2 + x^*$ $\dot{a}_1 = 27.76(-a_1 + y_1 + y_2 - x^*)$ $\dot{a}_2 = 0.32a_1$
Acrobot	Observation noise	$u_1 = -y_3 + 1.29\sin(y_4)$	$u_1 = 2a_2 - \cos(2a_1)$ $\dot{a}_1 = 2y_1 - 2y_2$ $\dot{a}_2 = 5.47\sin(\sin(y_1))$
Acrobot	Partial state observability	No successful policies	$u_1 = 1.87a_1 + \cos(a_2)$ $\dot{a}_1 = 2.06y_1$ $\dot{a}_2 = y_1 - 2.68$
Acrobot	Two control inputs	$u_1 = 0.64y_4 + \sin(y_4) - 0.04$ $u_2 = \sin(0.37y_3)$	$u_1 = 2.71\cos(a_2 - 0.57)$ $u_2 = -a_2 - 1.48$ $\dot{a}_1 = -8.66a_1 + y_4$ (inactive) $\dot{a}_2 = -3.44y_2$ $\dot{a}_3 = y_3$ (inactive)
CSTR	Standard	$u_1 = T_c^4(T_r^*)^{-3} - T_r$	$u_1 = a_1^2a_2 + 2a_1\log(T_r^*)$ $\dot{a}_1 = (2.92 - a_1)(u_1 + 2.98)$ $\dot{a}_2 = (u_1 - a_2 + 2.37)(a_2 - T_r^* + T_r)$

de Vries, S., Keemink, S., & van Gerven, M. (2024). Discovering Dynamic Symbolic Policies with Genetic Programming. http://arxiv.org/abs/2406.02765

KOZAX: A JAX FRAMEWORK FOR GENETIC PROGRAMMING

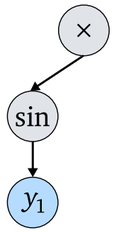


https://github.com/sdevries0/Kozax/tree/main

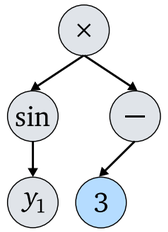
de Vries, S., Keemink, S. W., & van Gerven, M. A. J. (2025). Kozax: Flexible and Scalable Genetic Programming in JAX. GECCO (accepted for publication)

KOZAX INTERNALS

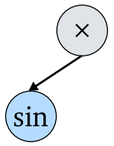
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	—	—	—	—
3	—	—	—	—
4	—	—	—	—
5	—	—	—	—
6	—	—	—	—
7	4(×)	—	—	—



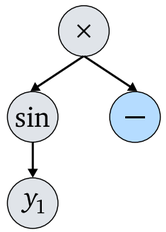
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	—	—	—	—
3	—	—	—	—
4	—	—	—	—
5	7(y_1)	—	—	—
6	5(sin)	5	—	—
7	4(×)	6	—	—



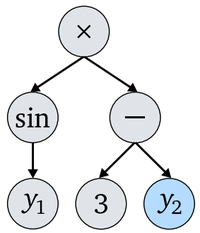
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	—	—	—	—
3	1(constant)	—	—	3
4	3(—)	3	—	—
5	7(y_1)	—	—	—
6	5(sin)	5	—	—
7	4(×)	6	4	—



Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	—	—	—	—
3	—	—	—	—
4	—	—	—	—
5	—	—	—	—
6	5(sin)	—	—	—
7	4(×)	6	—	—



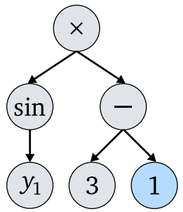
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	—	—	—	—
3	—	—	—	—
4	3(—)	—	—	—
5	7(y_1)	—	—	—
6	5(sin)	5	—	—
7	4(×)	6	4	—



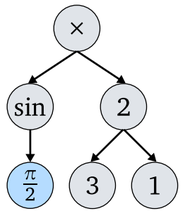
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	8(y_2)	—	—	—
3	1(constant)	—	—	3
4	3(—)	3	2	—
5	7(y_1)	—	—	—
6	5(sin)	5	—	—
7	4(×)	6	4	—

- Computational graphs mapped to matrices
- Allows for efficient vmapping and jit-compilation

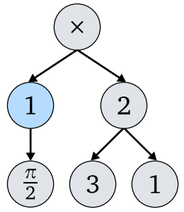
KOZAX INTERNALS



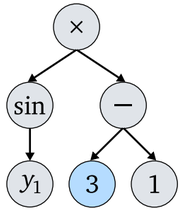
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	0
5	$7(y_1)$	−	−	0
6	5(sin)	5	−	0
7	4(×)	6	4	0



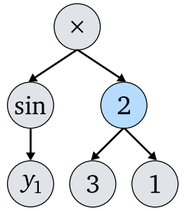
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	2
5	$7(y_1)$	−	−	$\pi/2$
6	5(sin)	5	−	0
7	4(×)	6	4	0



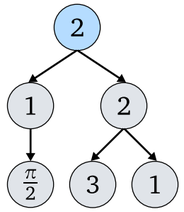
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	2
5	$7(y_1)$	−	−	$\pi/2$
6	5(sin)	5	−	1
7	4(×)	6	4	0



Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	0
5	$7(y_1)$	−	−	0
6	5(sin)	5	−	0
7	4(×)	6	4	0



Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	2
5	$7(y_1)$	−	−	0
6	5(sin)	5	−	0
7	4(×)	6	4	0

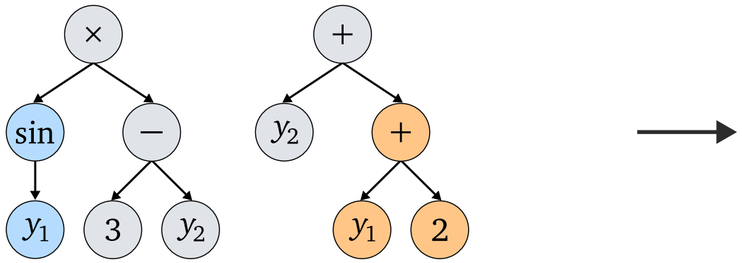


Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	−	−	−	0
1	−	−	−	0
2	$8(y_2)$	−	−	1
3	1 (constant)	−	−	3
4	3(−)	3	2	2
5	$7(y_1)$	−	−	$\pi/2$
6	5(sin)	5	−	1
7	4(×)	6	4	2

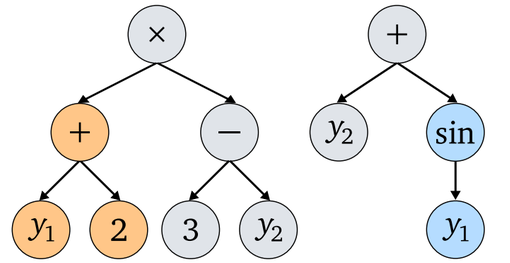
Processing of a matrix in Kozax:

- The matrix is iteratively solved with the input values: y1 = π/2 and y2 = 1
- A blue row in the matrix corresponds to a blue node in the tree, where the computed value of the node is stored in the last column
- The final value of the tree is obtained by taking the stored value in the last row

a



Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	$8(y_2)$	—	—	—
3	1 (constant)	—	—	3
4	$3(-)$	3	2	—
5	$7(y_1)$	—	—	—
6	5(sin)	5	—	—
7	$4(\times)$	6	4	—

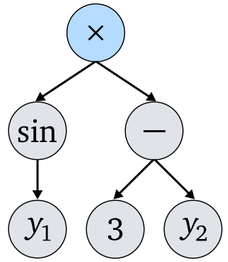


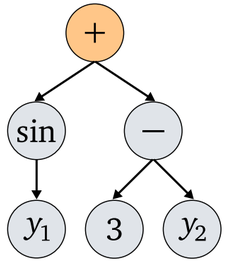
Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	$8(y_2)$	—	—	—
2	1 (constant)	—	—	3
3	$3(-)$	2	1	—
4	1 (constant)	—	—	2
5	$7(y_1)$	—	—	—
6	$2(+)$	5	4	—
7	$4(\times)$	6	3	—

Crossover applied to a pair of trees, producing two new trees:

- A random node is selected in both trees and the corresponding subtrees are swapped, indicated by the blue and orange subtrees.
- The matrix shows the representation of the left tree before and after crossover, where the blue and orange rows correspond to the removed and added subtrees respectively.
- The green cells show the nodes that remain in the tree, but of which the position or child indices have been changed accordingly.

KOZAX INTERNALS





Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	8 (y_2)	—	—	—
3	1 (constant)	—	—	3
4	3 (−)	3	2	—
5	7 (y_1)	—	—	—
6	5 (sin)	5	—	—
7	4 (×)	6	4	—

Idx	Function idx	Child 1 idx	Child 2 idx	Node value
0	—	—	—	—
1	—	—	—	—
2	8 (y_2)	—	—	—
3	1 (constant)	—	—	3
4	3 (−)	3	2	—
5	7 (y_1)	—	—	—
6	5 (sin)	5	—	—
7	2 (+)	6	4	—

Mutation is applied to a tree to evolve a new tree:

- In this example, the root node changes from a multiplication to an addition
- The matrix representation is shown before and after mutation in blue and orange respectively.

FRAMEWORK COMPARISON: FEATURES

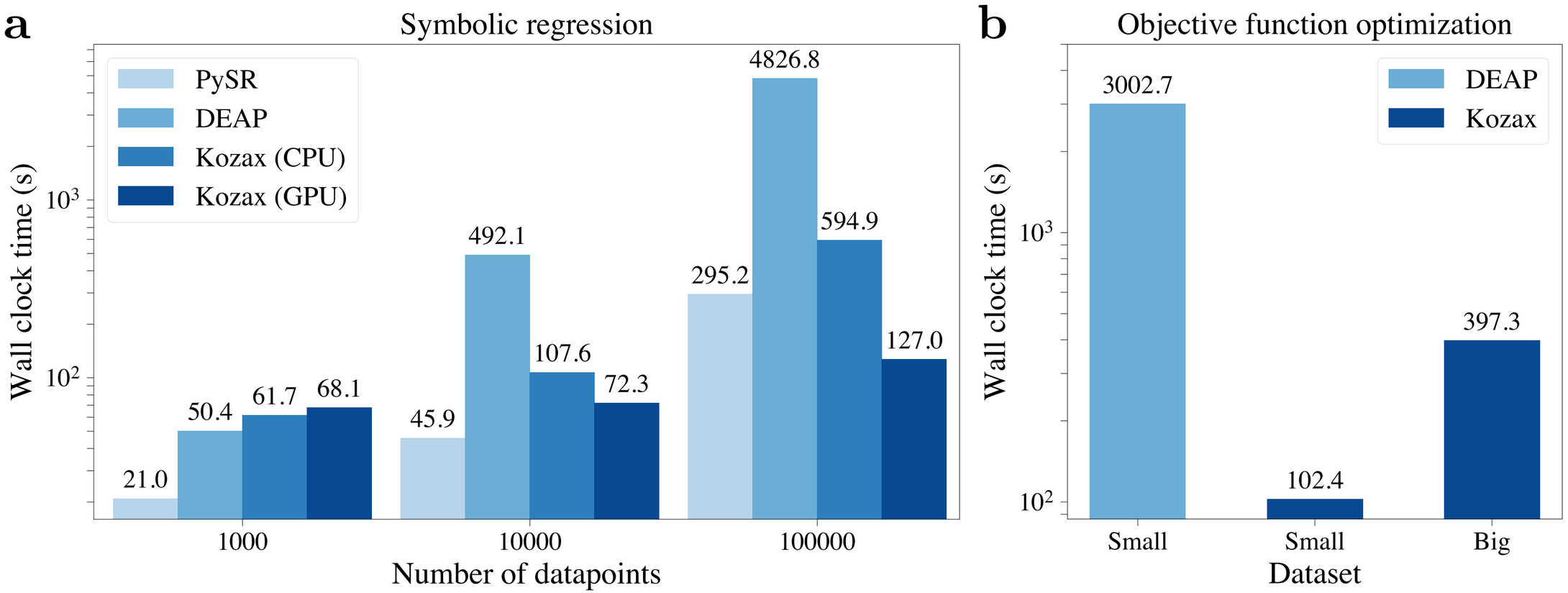
Feature	**PySR**	**DEAP**	**Kozax**
Custom operators	✓	✓	✓
Custom fitness function	✓	✓	✓
Pareto front	✓	✓	✓
Symbolic constraints	✓	–	–
Simplification	✓	–	–
Multi-objective optimization	–	✓	–
JIT compilation	✓	–	✓
Constant optimization	✓	–	✓
Flexible tree definition	–	✓	✓
Different tree classes	–	–	✓
Runs on GPU	–	–	✓

FRAMEWORK COMPARISON: PERFORMANCE

We can solve a wide range of different problems including scientific discovery, circuit design and optimal control

Experiment	PySR		DEAP		Kozax	
	Fitness	Size	Fitness	Size	Fitness	Size
Kepler's third law	**2.00** ± 0.10	5.2 ± 0.6	9.69 ± 9.19	83.1 ± 26.0	5.15 ± 5.98	7.8 ± 2.2
Newton's law	**0.15** ± 0.21	10.2 ± 3.3	1.12 ± 0.21	50.0 ± 19.1	**0.14** ± 0.19	9.6 ± 1.3
Bode's law	**0.07** ± 0.00	7.0 ± 0.0	0.28 ± 0.11	27.6 ± 9.7	**0.07** ± 0.00	7.0 ± 0.0
Fully observable LV	**0.00** ± 0.00	14.0 ± 0.0	0.47 ± 0.20	26.1 ± 4.4	0.01 ± 0.01	16.2 ± 1.4
Partially observable LV	–	–	–	–	**0.24** ± 0.30	16.8 ± 2.0
Acrobot	–	–	**0.32** ± 0.00	15.0 ± 2.3	**0.33** ± 0.01	9.3 ± 1.8
Loss function (small)	–	–	−0.72 ± 0.02	3.4 ± 2.7	**-0.85** ± 0.15	8.9 ± 4.3
Loss function (big)	–	–	–	–	**-0.98** ± 0.00	10.3 ± 1.6

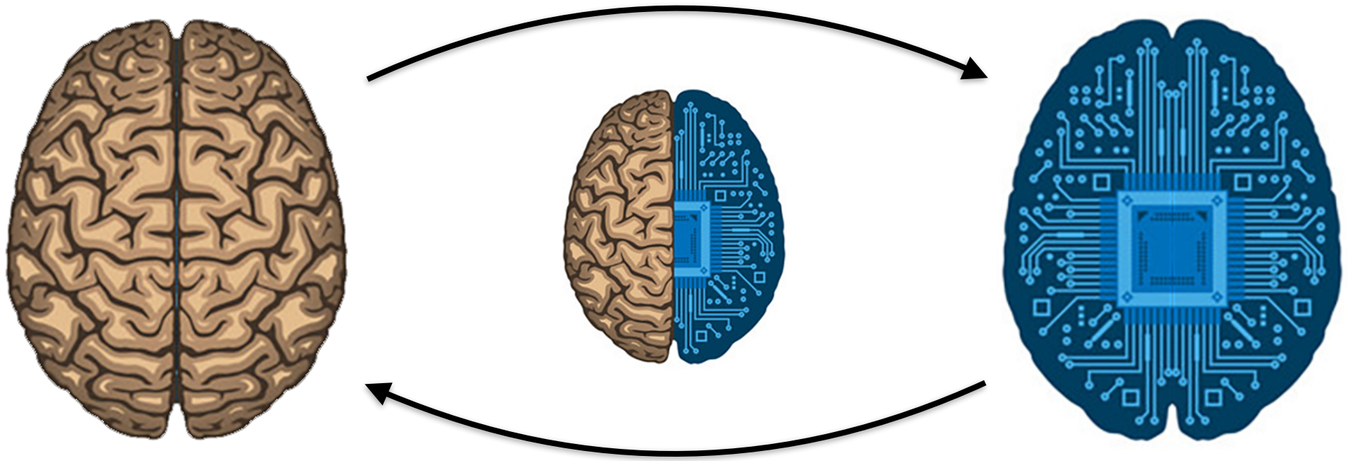
FRAMEWORK COMPARISON: SPEED



OUTLINE

- NEUROMORPHIC COMPUTING
- NOISE-BASED LEARNING
- GENETIC PROGRAMMING
- CONCLUSIONS

CONCLUSIONS

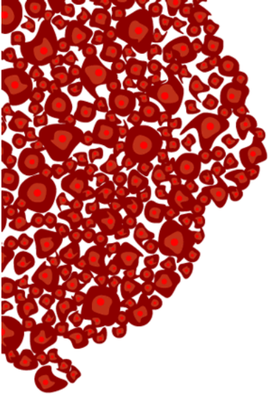


Neuromorphic computing is a beautiful approach for the future of compute

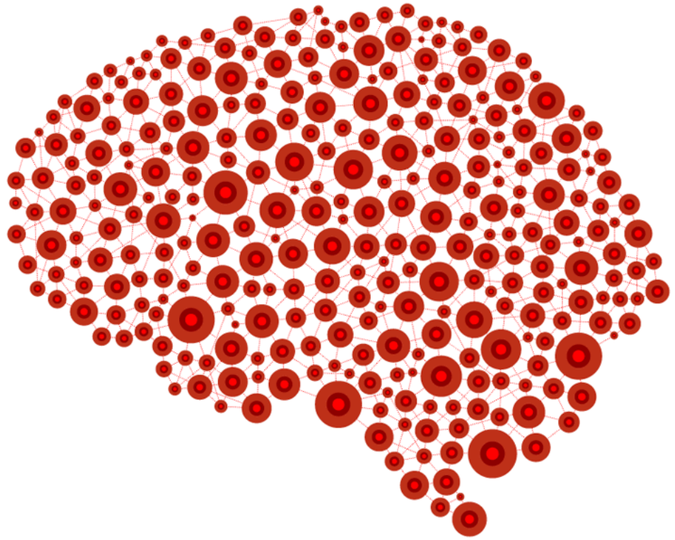
Huge opportunity for the European scientific and strategic roadmap

Breakthroughs drive advances in science, industry and society:

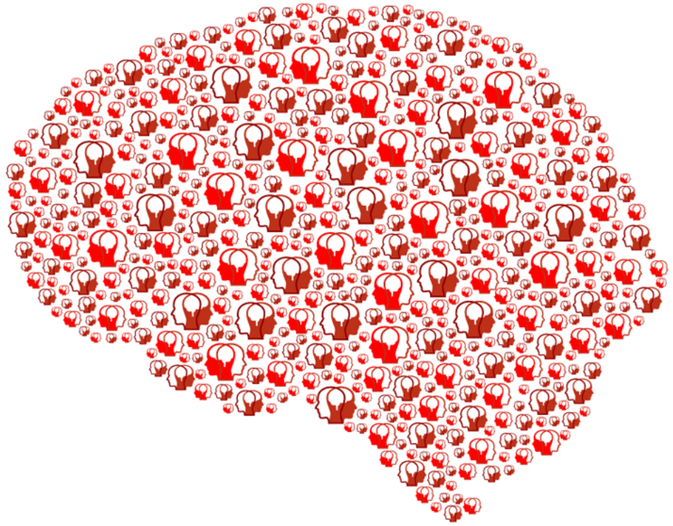
- Self-learning neuromorphic system-on-chip for sustainable AI
- Neuromorphic edge devices for optimal control
- Brain-inspired autonomous systems

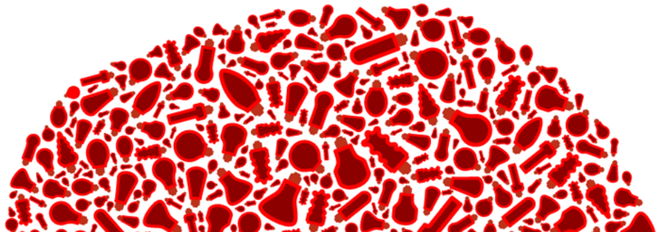


DONDERS INSTITUTE



THANK YOU



Radboud University Radboudumc