Cognitive Systems Engineering

Course organiser: Prof. Shihua Zhou

Course presenter

Prof Nikola Kasabov

Visiting Professor at Dalian University

Life FIEEE, FRSNZ, FINNS, DVF RAE UK

Founding Director KEDRI

Professor, Auckland University of Technology, NZ

George Moore Chair/Professor, Ulster University, UK

Honorary Professor, University of Auckland NZ , Peking University China

Visiting Professor IICT/Bulgarian Academy of Sciences and Teesside University UK

Doctor Honoris Causa Obuda University Budapest

Director, Knowledge Engineering Consulting Ltd (https://www.knowledgeengineering.ai)

Assistant

Doct Ms Iman AbouHassan

iabouhassan@tu-sofia.bg

abouhassan.iman@gmail.com

Cognitive System Engineering

Cognitive systems (CogSys) are software-hardware systems that have their structure and functionality based on principles of information processing in the human brain. They are part of AI, but AI includes also other systems that manifest cognitive behaviour, such as speech and image recognition, learning and reasoning, but using other methods , such as statistical, empirical, abstract logic , etc.

The course is by research papers.
Every topic will include:

1. Topic/task/problem specification
2. Previously published methods for solving the problem
3. Description of the method and in the paper under discussion
4. Software implementation, experimental results and discoveries
5. Applications
6. Future work to be done for this problem and questions for individual work

Expected results:

1. Students obtain new knowledge and skills in the area of CogSys for AI applications.
2. Students can learn to take a critical approach to the existing methods and systems.
3. Students can get confidence that they can suggest new methods and to publish them in good journals.

Additional materials: https://www.knowledgeengineering.ai/china
ZOOM link for all lectures:
https://us05web.zoom.us/j/4658730662?pwd=eFN0eHRCN3o4K0FaZ0lqQmN1UUgydz09

CogSysEn: Lecture Topics

1. Introduction to the course

Part I : Learning systems

2. Deep learning and deep knowledge representation in the human brain

-Chapter 3 from: N.Kasabov, Time-space, spiking neural networks and brain-inspired artificial intelligence, Springe-Nature, 2029

3. Modelling brain dynamics

- Benuskova, L., Kasabov, N. Modeling brain dynamics using computational neurogenetic approach. Cogn Neurodyn 2, 319–334 (2008). https://doi.org/10.1007/s11571-008-9061-1

4. Evolving learning systems

- N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 31, no. 6, pp. 902-918, Dec. 2001, doi: 10.1109/3477.969494.

- NeuCom software (https://theneucom.com): EFuNN

5. Neuro—fuzzy learning and inference systems: DENFIS

- Kasabov, N. K., & Song, Q. (2002). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE transactions on Fuzzy Systems, 10(2), 144-154.

- DENFIS software in Python.

6. Spatio-temporal learning systems: SNN

- N. Kasabov, K. Dhoble, N. Nuntalid, G. Indiveri, Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. Neural Networks, 41(1995), 188–201 (2013). https://doi.org/10.1016/j.neunet.2012.11.014.

- Software deSNN

7. Reservoir computing and Brain-inspired SNN

- *N. Kasabov, NeuCube: a spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. Neural Netw. 52(2014), 62–76 (2014).*
- *N. Kasabov et al, Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications, Neural Networks, v.78, 1-14, 2016. http://dx.doi.org/10.1016/j.neunet.2015.09.011.*

8. Integrated learning systems:

- *P. Koprinkova-Hristova, D. Penkov, S. Nedelcheva, S. Yordanov and N. Kasabov, "On-line Learning, Classification and Interpretation of Brain Signals using 3D SNN and ESN," 2023 International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, 2023, pp. 1-6, doi: https://doi.org/10.1109/IJCNN54540.2023.10191974,*
- AbouHassan et al, *NeuDen: Integrating evolving Neuromorphic spiking neural networks and Dynamic evolving neuro-fuzzy systems for predictive and explainable learning of multiple time series*

nkasabov@aut.ac.nz

CogSysEn: Lecture Topics

Part II. Associative memories

9. Evolving Associative Memories in bio-neuro systems and in SNN

- *Kasabov, Nikola (2023). STAM-SNN: Spatio-Temporal Associative Memories in Brain-inspired Spiking Neural Networks: Concepts and Perspectives. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.23723208.v1*

10. Associative memories for neuroimaging data: EEG and fMRI

- *N K. Kasabov, H Bahrami, M Doborjeh, A Wang, Brain Inspired Spatio-Temporal Associative Memories for Neuroimaging Data: EEG and fMRI, Bioengineering 2023, MDPI 10(12), 1341 https://doi.org/10.3390/bioengineering10121341, www.mdpi.com/journal/bioengineering*

11. Audio-visual associative memories

- *N Kasabov, B Sen Bhattacharya, D Patel, N Aggarwal, T Bankar, IAbouHassan, Cognitive Audio-Visual Associative Memories using Brain-inspired Spiking Neural Networks with Case Studies on Moving Object Recognition .*

12. Predictive associative memories for time series

- *AbouHassan, I; Kasabov, N; Bankar, T; Garg, R; Sen Bhattacharya, B (2023). PAMeT-SNN: Predictive Associative Memory for Multiple Time Series based on Spiking Neural Networks with Case Studies in Economics and Finance. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.24063975.v1, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4665533*

Part III. Software and Hardware Implementation of CogSys.

13. Neuromorphic hardware for CogSys implementations

- *J. Behrenbeck, Z. Tayeb, C. Bhiri, C. Richter, O. Rhodes, N. Kasabov, S. Furber, J. Conrad, Classification and Regression of Spatio-Temporal EMG Signals using NeuCube Spiking Neural Network and its implementation on SpiNNaker Neuromorphic Hardware. J. Neural Eng. (IOP Press, Article reference: JNE-102499) (2018). http://iopscience.iop.org/journal/1741-2552.*

  - *paper for CogSys on Loihi chip*

14. CogSys design and software/hardware implementation

- *NeuCubePy, NEST, PyNN for SpiNNaker, Lava for Loihi, Software for China chips*

15. Quantum computation

- *Ravi , N. Kasabov et al, (2023). From Quantum Computing to Quantum-inspired Computation for Neuromorphic Advancement – A Survey. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.24053250.v1*

16. Revision of the course

nkasabov@aut.ac.nz

Lecture 14. CogSys Design and Software/Hardware Implementations

1. CogSys Design

2. Python based implementations. NeuCubePy

3. PyNN for SpiNNaker

4. NEST

5. Lava for Loihi

6. Questions

1.**CogSys Design**

1. Analysis of the problem and the type of data available
2. Architecture design – usually using SNN
3. Input data encoding
4. Learning in the SNN model:
   - unsupervised,
   - supervised,
   - evolving (life-ling)
   - reinforcement, …….
5. Classification/regression models and calculating the outputs
6. Dynamic parameter optimisation;
7. Model testing
8. Extracting knowledge and visualisation
9. Adaptation on new data in an on-line/ real time mode;
10. Implementation of a SNN model:

- von Neumann vs
- Neuromorphic vs .
- Quantum

Choosing the platform for realisation of a SNN-based CogSys

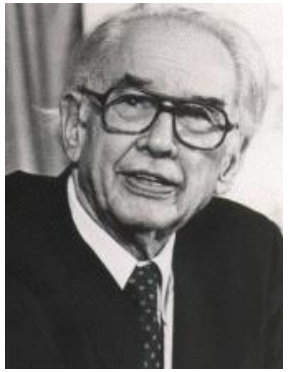
A SNN CogSys can be implemented using:

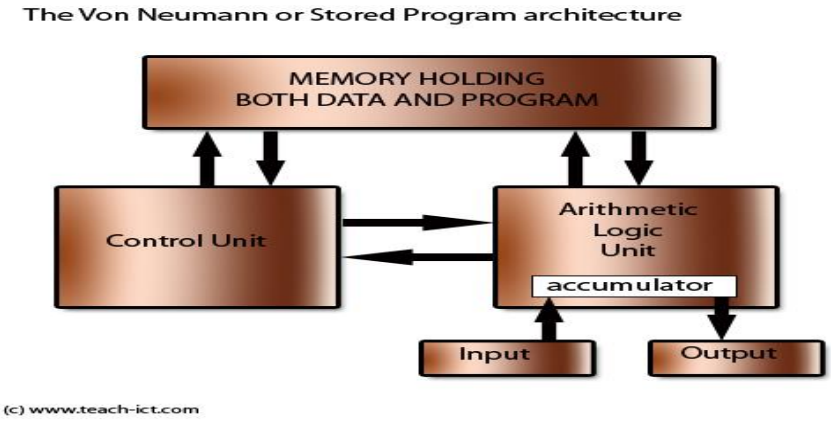
- von Neumann architecture;
- Neuromorphic architecture;
- Quantum computer (not available yet).

- The computer architecture of John von Neumann separates data and programmes (kept in the memory unit) from the computation (ALU); uses *bits*. First machine ABC by Atanassov and Berry.

- A Neuromorphic architecture integrates the data, the programme and the computation in a SNN structure, similar to how the brain works; uses *spikes* (bits at times).

- A quantum computer uses *q-bits* (bits in a superposition) .





N. Sengupta et al, (2018), From von Neumann architecture and Atanasoff's ABC to Neuromorphic Computation and Kasabov's NeuCube: Principles and Implementations, Chapter 1 in: Advances in Computational intelligence, Jotzov et al (eds) Springer 2018.

Neuromorphic hardware systems

SpiNNaker (*Furber, S., To Build a Brain, IEEE Spectrum, vol.49, Number 8, 39-41, 2012).*

Silicon retina (the DVS) and silicon cochlea (ETH, Zurich, Toby Delbruck)) for input data encoding

Loihi chip of Intel

High speed and low power consumption!

nkasabov@aut.ac.nz

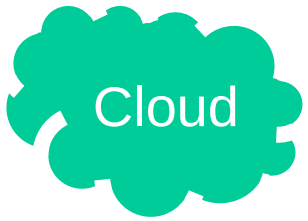
3. Python based implementations. NeuCubePy

NeuCube development environment for SNN system design

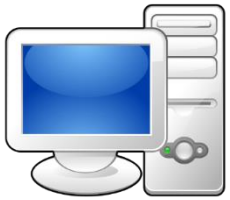
nkasabov@aut.ac.nz www.kedri.aut.ac.nz/neucube/

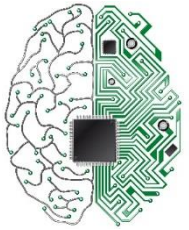
NeuCube Implementations

Software versions:



Hardware-specific versions:





Future development: NeuCube chips for AI applications

NeuCubePy

https://github.com/KEDRI-AUT/NeuCube-Py

Sources:

neucube/reservoir.py
neucube/topology.py
neucube.egg-info/PKG-INFO
neucube.egg-info/SOURCES.txt
neucube.egg-info/dependency_links.txt
neucube.egg-info/requires.txt
neucube.egginfo/top_level.txt
neucube/encoder/__init__.py
neucube/encoder/encoder.py
neucube/sampler/__init__.py
neucube/sampler/sampler.py
neucube/training/__init__.py
neucube/training/nrdp.py
neucube/training/stdp.py
neucube/utils/__init__.py
neucube/utils/utils.py
neucube/validation/__init__.py
neucube/validation/pipeline.py

File Edit Selection View Go Run Terminal Help

NeuCubePy

EXPLORER

SEARCH

NEUCUBEPY

- build
- dist
- example_data
- neucube
- neucube.egg-info
- LICENSE.md
- neucube.egg-info.zip
- readme.md
- requirements.txt
- setup.py
- wrist_movement_demo.ipynb

OUTLINE

TIMELINE

wrist_movement_demo.ipynb ● | Release Notes: 1.90.0

wrist_movement_demo.ipynb > ...

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...

Python undefined.undefined.undefined

```
from neucube import Reservoir
from neucube.encoder import Delta
from neucube.validation import Pipeline
from neucube.sampler import SpikeCount
import torch
```

[5] Python

```
import numpy as np
import pandas as pd

filenameslist = ['sam'+str(idx)+'_eeg.csv' for idx in range(1,61)]

dfs = []
for filename in filenameslist:
    dfs.append(pd.read_csv('./example_data/wrist_movement_eeg/'+filename, header=None))

fulldf = pd.concat(dfs)

labels = pd.read_csv('./example_data/wrist_movement_eeg/tar_class_labels.csv', header=None)
y = labels.values.flatten()

feat_names = pd.read_csv('./example_data/wrist_movement_eeg/feature_names_eeg.txt', header=None).values.flatten()
brain_coordinates = pd.read_csv('./example_data/wrist_movement_eeg/brain_coordinates.csv', header=None).values
eeg_mapping = pd.read_csv('./example_data/wrist_movement_eeg/eeg_mapping.csv', header=None).values
```

[6] Python

```
X = torch.tensor(fulldf.values.reshape(60,128,14))
encoder = Delta(threshold=0.8)
X = encoder.encode_dataset(X)
y = labels.values.flatten()
```

[7] Python

⊗0 ⚠0 0

Ln 3, Col 29 Spaces: 4 Spaces: 4 CRLF Cell 5 of 5

14°C ENG 8:32 pm 7/06/2024

File Edit Selection View Go Run Terminal Help

NeuCubePy

EXPLORER

SEARCH

NEUCUBEPY

- build
- dist
- example_data
- neucube
- neucube.egg-info
- LICENSE.md
- neucube.egg-info.zip
- readme.md
- requirements.txt
- setup.py
- wrist_movement_demo.ipynb

OUTLINE

TIMELINE

wrist_movement_demo.ipynb ● | Release Notes: 1.90.0

wrist_movement_demo.ipynb > ...

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

Python undefined.undefined.undefined

```
from sklearn.metrics import accuracy_score as accuracy
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from tqdm import tqdm

kf = KFold(n_splits=5, shuffle=True, random_state=123)
y_total, pred_total = [],[]

for train_index, test_index in tqdm(kf.split(X)):
  X_train, X_test = X[train_index], X[test_index]
  y_train, y_test = y[train_index], y[test_index]

  res = Reservoir(inputs=14)
  sam =  SpikeCount()
  clf = LogisticRegression(solver='liblinear')
  pipe = Pipeline(res, sam, clf)

  pipe.fit(X_train, y_train)
  pred = pipe.predict(X_test)

  y_total.extend(y_test)
  pred_total.extend(pred)

print(accuracy(y_total, pred_total))
print(confusion_matrix(y_total, pred_total))
```

[8] Python

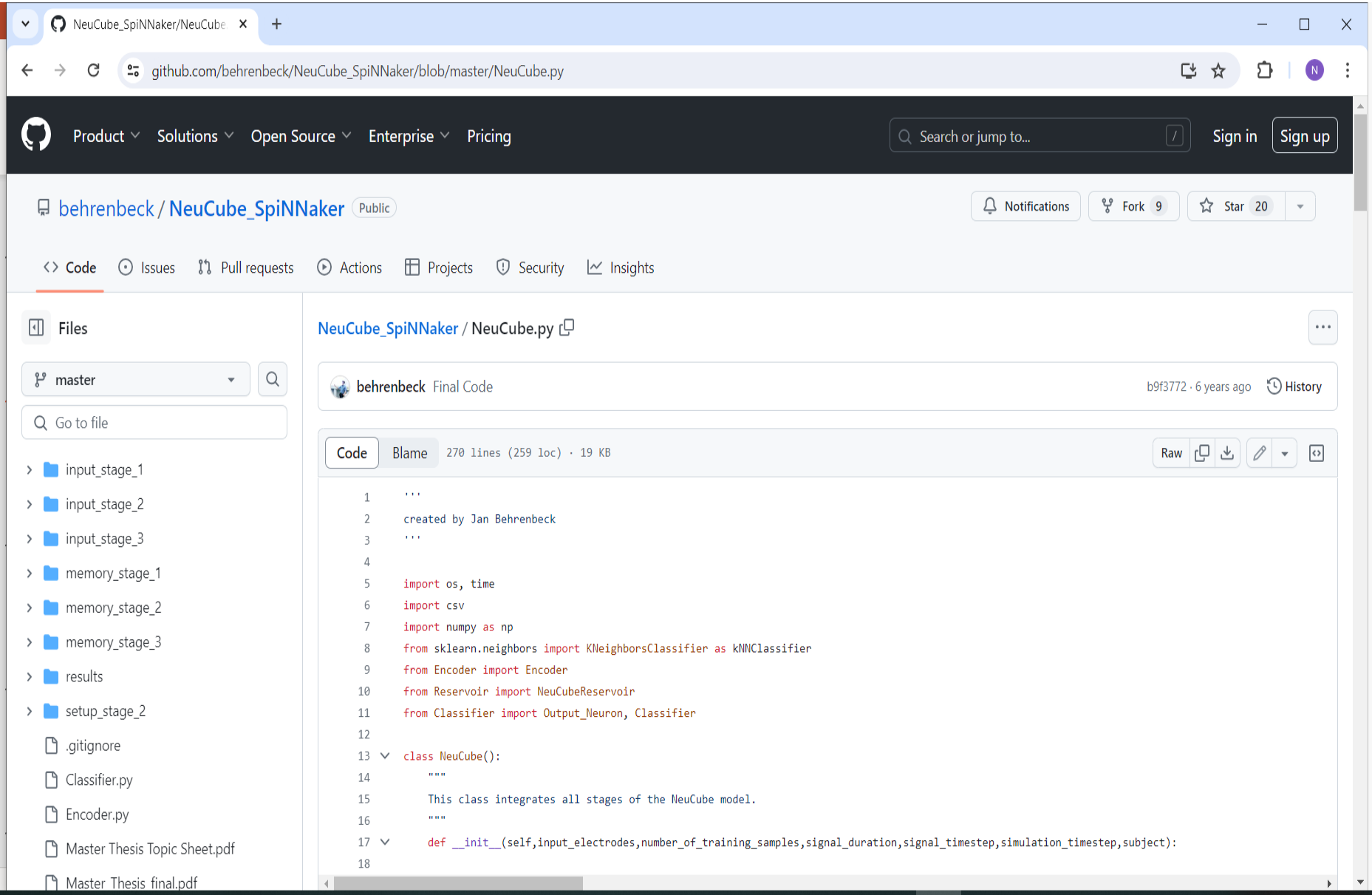
```
5it [00:27,  5.43s/it]
0.7666666666666667
[[17  3  0]
 [ 7 11  2]
 [ 0  2 18]]
```

Ln 3, Col 29 Spaces: 4 Spaces: 4 CRLF Cell 5 of 5

14°C ENG 8:35 pm 7/06/2024

4. PyNN for SPiNNaker

https://github.com/behrenbeck/NeuCube_SpiNNaker



Classification and Regression of Spatio-Temporal Signals using NeuCube and its implementation on SpiNNaker Neuromorphic Hardware

Jan Behrenbeck, Zied Tayeb, Cyrine Bhiri, Christoph Richter, Oliver Rhodes, Nikola Kasaboov, Josafath Ramos, Steve Furber, Gordon Cheng and Joerg Conradt

nkasabov@aut.ac.nz

4. NEST

Koprinkova N3-BG 2023.pdf

File | C:/nkasabov/R&D/BG/N3-BG/Koprinkova%20N3-BG%202023.pdf

Draw | 55 of 55

development

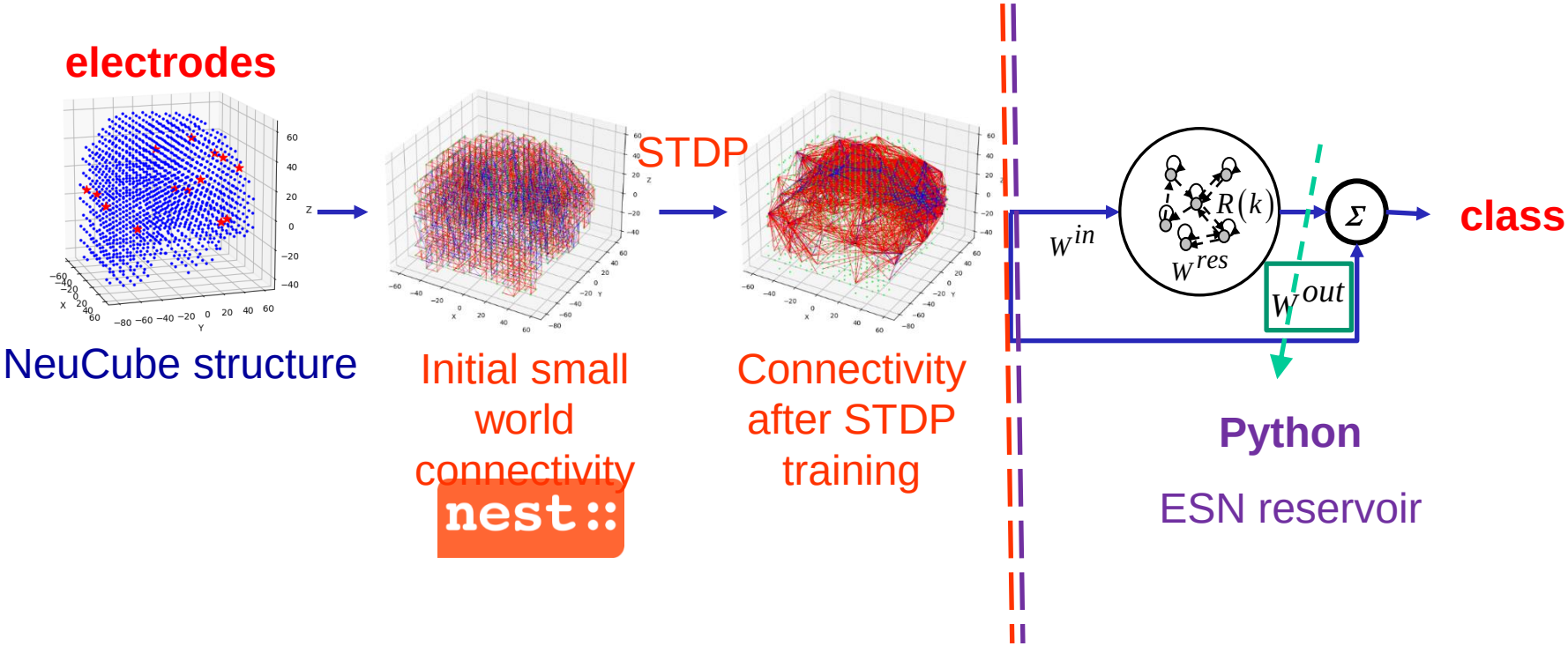
NEST simulator

A tool for biologically plausible AI development

1. NEST provides over 50 neuron models many of which have been published. Choose from simple integrate-and-fire neurons with current or conductance based synapses, over the Izhikevich or AdEx models, to Hodgkin-Huxley models.
2. NEST provides over 10 synapse models, including short-term plasticity (Tsodyks & Markram) and different variants of spike-timing dependent plasticity (STDP).
3. NEST provides many examples that help you getting started with your own simulation project.
4. NEST offers convenient and efficient commands to define and connect large networks, ranging from algorithmically determined connections to data-driven connectivity.
5. NEST lets you inspect and modify the state of each neuron and each connection at any time during a simulation.
6. NEST is fast and memory efficient. It makes best use of your multi-core computer and compute clusters with minimal user intervention.
7. NEST runs on a wide range of UNIX-like systems, from MacBooks to BlueGene supercomputers.
8. NEST has minimal dependencies. All it really needs is a C++ compiler. Everything else is optional.
9. NEST developers are using agile continuous integration-based workflows in order to maintain high code quality standards for correct and reproducible simulations.
10. NEST has one of the largest and most experienced developer communities of all neural simulators. NEST was first released in 1994 under the name SYNOD and has been extended and improved ever since.
11. NEST is open source software and is licensed under the GNU General Public License v2 or later.

Result | ENG | 6:59 pm | 16/06/2024

Application of using NEST and Python for a BMI



P. Koprinkova-Hristova, D. Penkov, S. Nedelcheva, S. Yordanov and N. Kasabov, "On-line Learning, Classification and Interpretation of Brain Signals using 3D SNN and ESN," *2023 International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, Australia, 2023, pp. 1-6, doi: https://doi.org/10.1109/IJCNN54540.2023.10191974, IEEE Xplore Full-Text PDF.

6. LAVA for Loihi

M.Davies et al, EEE Micro Published by the IEEE Computer Society, January/February 2018 0272-1732/18/$33.00 ©2018

S.Dey, A.Dimirov, Mapping and Validating a Point Neuron Model on Intel's Neuromorphic Hardware Loihi, Front.Neurosci, 2022

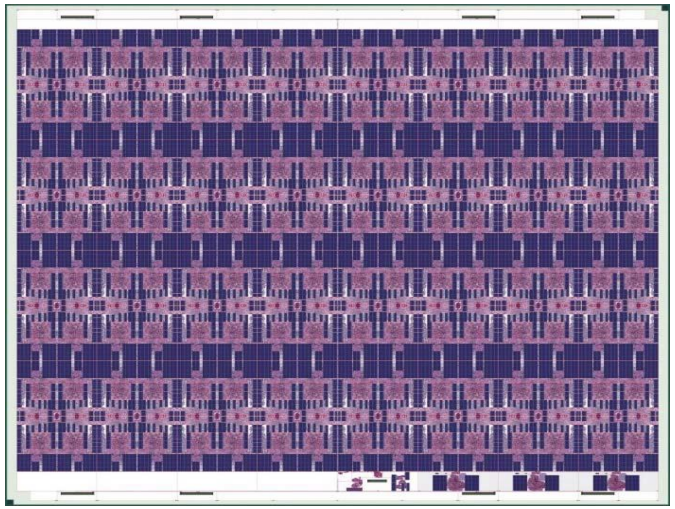
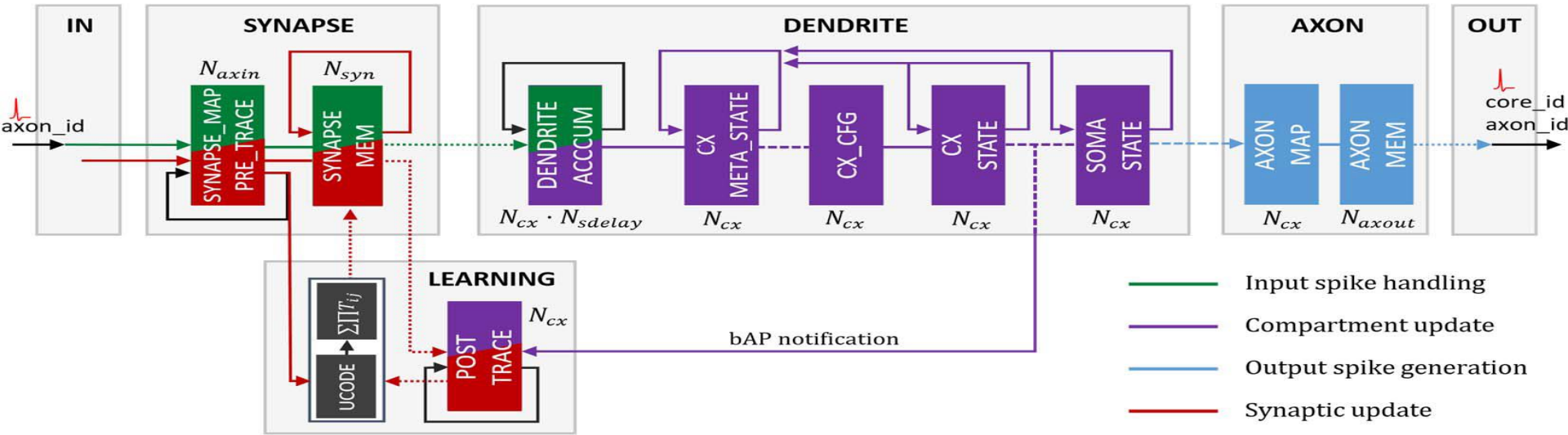


Table Loihi pre-silicon performance and energy measurements.

Measured parameter	**Value at 0.75 V**
Cross-sectional spike bandwidth per tile	3.44 Gspike/s
Within-tile spike energy	1.7 pJ
Within-tile spike latency	2.1 ns
Energy per tile hop (E-W / N-S)	3.0 pJ / 4.0 pJ
Latency per tile hop (E-W / N-S)	4.1 ns / 6.5 ns
Energy per synaptic spike op (min)	23.6 pJ
Time per synaptic spike op (max)	3.5 ns
Energy per synaptic update (pairwise STDP)	120 pJ
Time per synaptic update (pairwise STDP)	6.1 ns
Energy per neuron update (active / inactive)	81 pJ / 52 pJ
Time per neuron update (active / inactive)	8.4 ns / 5.3 ns
Mesh-wide barrier sync time (1-32 tiles)	113-465 ns



LAVA implementation language for Loihi chips

Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook, By: MIKE DAVIES , ANDREAS WILD , GARRICK ORCHARD , YULIA SANDAMIRSKAYA , GABRIEL A. FONSECA GUERRA , PRASAD JOSHI, PHILIPP PLANK , AND SUMEDH R. RISBUD, Proceedings of IEEE, *Vol. 109, No. 5, May 2021,* https://doi.org/10.1109/JPROC.2021.3067593

(from lava-nc.org)

- Lava is an open-source software framework for developing neuro-inspired applications and mapping them to neuromorphic hardware. Lava provides developers with the tools and abstractions to develop applications that fully exploit the principles of neural computation. Constrained in this way, like the brain, Lava applications allow neuromorphic platforms to intelligently process, learn from, and respond to real-world data with great gains in energy efficiency and speed compared to conventional computer architectures.
- The vision behind Lava is an open, community-developed code base that unites the full range of approaches pursued by the neuromorphic computing community. It provides a modular, composable, and extensible structure for researchers to integrate their best ideas into a growing algorithms library, while introducing new abstractions that allow others to build on those ideas without having to reinvent them.
- For this purpose, Lava allows developers to define versatile *processes* such as individual neurons, neural networks, conventionally coded programs, interfaces to peripheral devices, and bridges to other software frameworks. Lava allows collections of these processes to be encapsulated into modules and aggregated to form complex neuromorphic applications. Communication between Lava processes uses event-based message passing, where messages can range from binary spikes to kilobyte-sized packets.
- The behavior of Lava processes is defined by one or more *implementation models*, where different models may be specified for different execution platforms (“backends”), different degrees of precision, and for high-level algorithmic modeling purposes. For example, an excitatory/inhibitory neural network process may have different implementation models for an analog neuromorphic chip compared to a digital neuromorphic chip, but the two models could share a common “E/I” process definition with each model’s implementations determined by common input parameters.
- Lava is platform-agnostic so that applications can be prototyped on conventional CPUs/GPUs and deployed to heterogeneous system architectures spanning both conventional processors as well as a range of neuromorphic chips such as Intel’s Loihi. To compile and execute processes for different backends, Lava builds on a low-level interface called *Magma* with a powerful compiler and runtime library. Over time, the Lava developer community may enhance Magma to target additional neuromorphic platforms beyond its initial support for Intel’s Loihi chips.

Questions, exercises, assignments and project work

1. What are the main software systems for neuromorphic computers ?
2. What are the benefits of CoSysEn on neuromorphic platforms?

Course References

1. N.Kasabov, Time-Space, Spiking Neural Networks and Brain-Inspired AI, Springer 2019 (course book).
2. N. Kasabov Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, MIT Press, 1996 (additional reading)
3. N.Kasabov, Evolving connectionist systems, Springer 2003 and 2007 (additional reading)
4. Kasabov, N. (ed) (2014) The Springer Handbook of Bio- and Neuroinformatics, Springer. (additional reading)
5. NeuCube: http://www.kedri.aut.ac.nz/neucube/
6. NeuCom: https://theneucom.com
7. KEDRI R&D Systems available from: http://www.kedri.aut.ac.nz
8. N. Kasabov, et al, Design methodology and selected applications of evolving spatio- temporal data machines in the NeuCube neuromorphic framework, Neural Networks, v.78, 1-14, 2016. http://dx.doi.org/10.1016/j.neunet.2015.09.011.
9. Furber, S., To Build a Brain, IEEE Spectrum, vol.49, Number 8, 39-41, 2012.
10. Benuskova, L., N.Kasabov (2007) Computational Neurogenetic Modelling, Springer, New York
11. Indiveri, G. et al, Neuromorphic silicon neuron circuits, Frontiers in Neuroscience, 5, 2011.
12. Kasabov, N. (2014) NeuCube: A Spiking Neural Network Architecture for Mapping, Learning and Understanding of Spatio-Temporal Brain Data, Neural Networks, 52, 62-76.
13. Kasabov (2010) To spike or not to spike: A probabilistic spiking neural model, Neural Networks, v.23,1, 16-19
14. Merolla, P.A., J.V. Arhur, R. Alvarez-Icaza, A.S.Cassidy, J.Sawada, F.Akopyan et al, “A million spiking neuron integrated circuit with a scalable communication networks and interface”, Science, vol.345, no.6197, pp. 668-673, Aug. 2014.
15. Wysoski, S., L.Benuskova, N.Kasabov (2007) Evolving Spiking Neural Networks for Audio-Visual Information Processing, Neural Networks, vol 23, issue 7, pp 819-835.
16. Kasabov, Nikola; Tan, Yongyao Tan; Doborjeh, Maryam; Tu, Enmei; Yang, Jie (2023): Transfer Learning of Fuzzy Spatio-Temporal Rules in the NeuCube Brain-Inspired Spiking Neural Network: A Case Study on EEG Spatio-temporal Data. TechRxiv. Preprint. https://techrxiv.org), https://doi.org/10.36227/techrxiv.21781103.v1, licence CC BY 4.0)
17. Nikola K. Kasabov, Iman AbouHassan, Vinayak G.M. Jagtap, Parag Kulkarni, Spiking neural networks for predictive and explainable modelling of multimodal streaming data on the Case Study of Financial Time Series Data and on-line news, SREP, Nature, pre-print on the Research Square, DOI: https://doi.org/10.21203/rs.3.rs-2262084/v1, licence CC BY 4.0,

- https://orcid.org/0000-0003-4433-7521
- https://knowledgeengineering.ai
- http://scholar.google.com/citations?hl=en&user=YTa9Dz4AAAAJ&view_op=list_works
- https://www.scopus.com/authid/detail.uri?authorId=35585005300