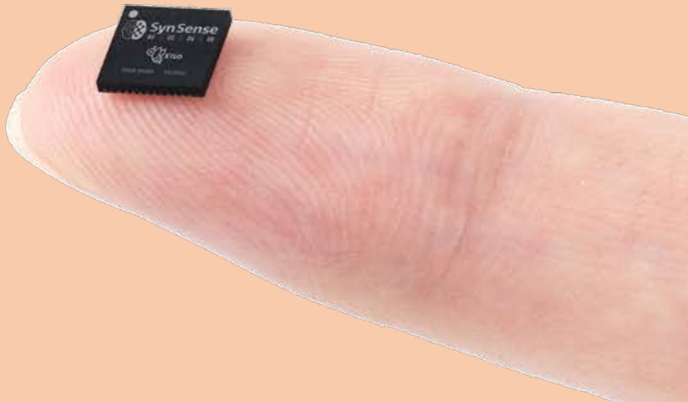
Training and deploying SNN applications with Rockpool and Xylo



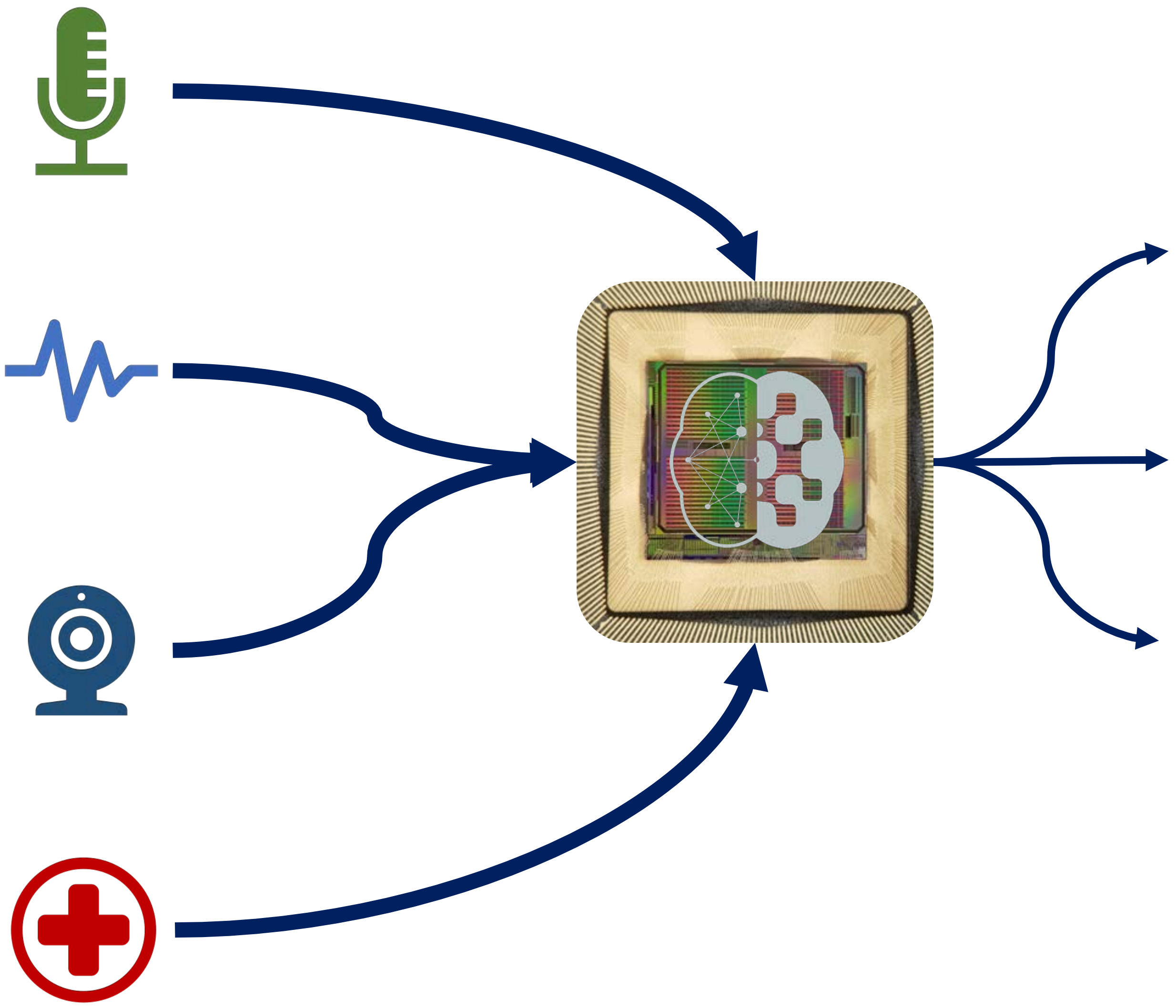
OpenNeuromorphic | April 26th 2023

All code: https://github.com/synsense/OpenNeuromorphic_26042023

SynSense

Hardware / IP / Applications
Ultra-low-power compute
Sensory processing
At the edge

Neuromorphic Smart Sensors



- Highly informative output / low bandwidth output
- Smart condition detection
- Smart wake-up
- Continuous monitoring
- Low latency → <200 ms
- Low power → <10 mW

Hardware families

Vision processing with high speed, low power

DynapCNN

Scalable CNN cores

Speck

Integrated vision sensing

HDK

HDK

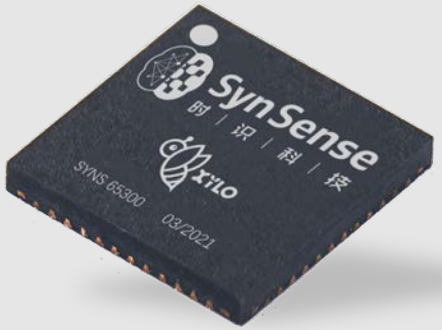
Smart visual wake-up
Object trakcing
Presence detection

Real-time motion estimation
Behaviour detection
Gesture interaction

Natural signal processing

Xylo

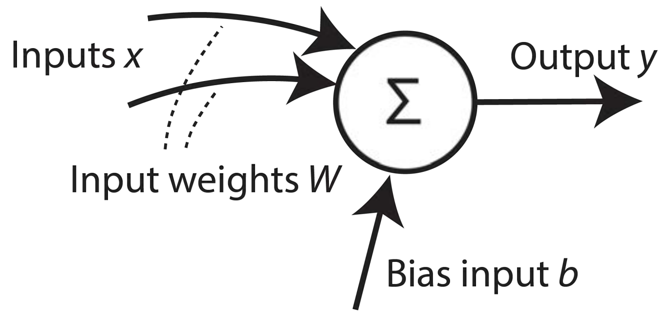
Ultra-low-power



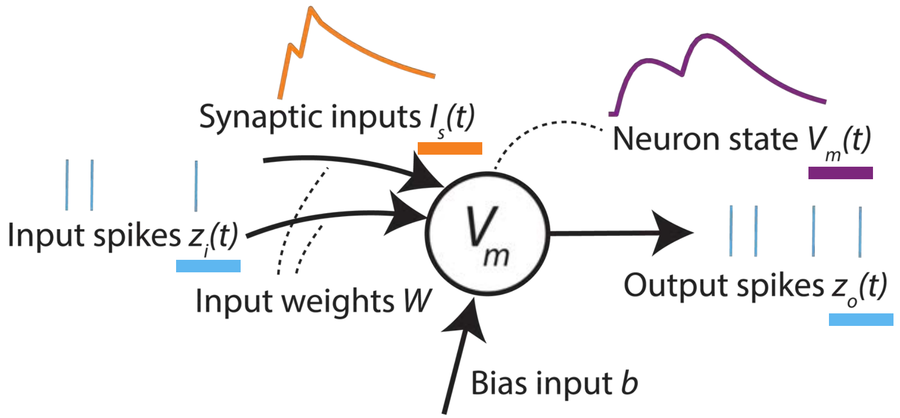
HDK

Audio processing
Bio-signal processing
IMU processing
Condition monitoring

Temporal computation with SNNs



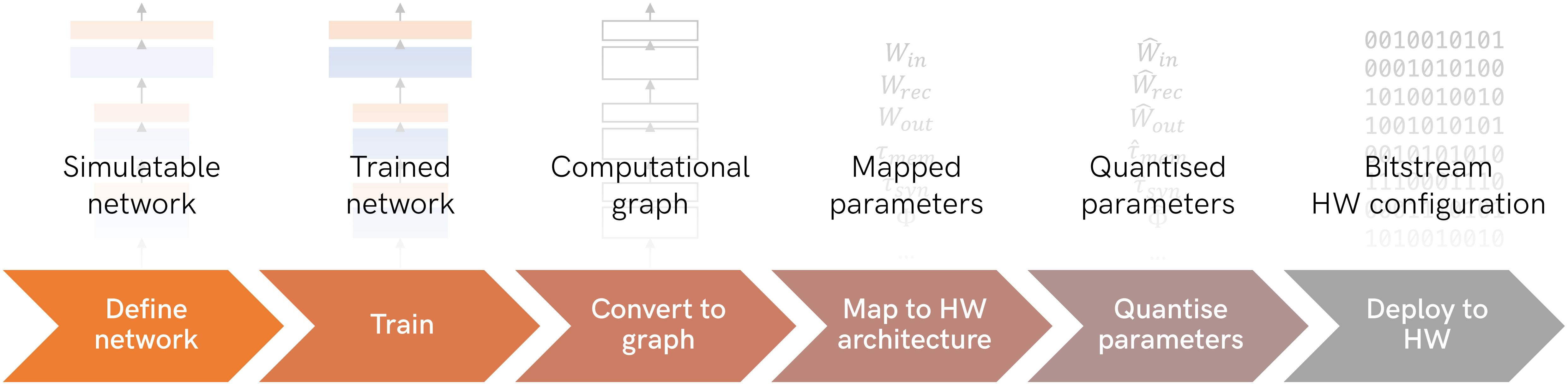
$$y = \Theta\left(W \cdot x + b\right)$$



$$\tau_s \dot{I}_s + I_s = W \cdot z_i(t)$$

$$\tau_m \dot{V}_m + V_m = I_s + b$$

$$V_m(t_j) > \theta \rightarrow \begin{cases} V_m(t_j) & \leftarrow V_m(t_j) - \theta \\ z_o(t) & \leftarrow z_o(t) + \delta(t_j) \end{cases}$$


Simulatable network
Trained network
Computational graph
Mapped parameters
Quantised parameters
Bitstream
HW configuration
Define network
Train
Convert to graph
Map to HW architecture
Quantise parameters
Deploy to HW

rockpool.ai

rockpool.ai

Rockpool

Search docs

Welcome to Rockpool

A simple introduction to spiking neurons

Spiking neural networks are considered to be the *third generation* of neural networks, preceeded by McCulloch-Pitts threshold neurons ("first generation") which produced digital outputs and Artificial Neural Networks with continuous activations, like sigmoids and hyperbolic tangets, ("second generation") that are commonly used these days.

Artificial Neuron (AN) Model

The standard artificial neuron model used most commonly in ANNs and DNNs is a simple equation

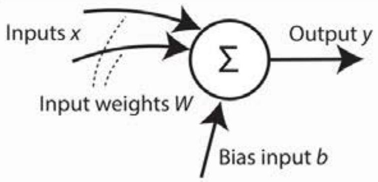
$$\vec{y} = \Theta(W.\vec{x} + b)$$

where Θ is typically a non linear activation function such as a *sigmoid* or *hyperbolic tangent* function.

[1]:
```
from IPython.display import Image

Image(filename="AN-neuron.png", width=300)
```

[1]:



Note that the output depends only on the instantaneous inputs. The neuron does not have an internal state that would affect its output.

rockpool.ai

nn.modules.LIF

class `nn.modules.LIF`(**args, **kwargs*) [source]

Bases: `rockpool.nn.modules.module.Module`

A leaky integrate-and-fire spiking neuron model

This module implements the update equations:

$$
\begin{aligned}
I_{syn} += S_{in}(t) + S_{rec} \cdot W_{rec} \\
I_{syn} *= \exp(-dt / au_{syn}) \\
V_{mem} *= \exp(-dt / au_{mem}) \\
V_{mem} += I_{syn} + b + \sigma \zeta(t)
\end{aligned}
$$

where $S_{in}(t)$ is a vector containing `1` (or a weighed spike) for each input channel that emits a spike at time t; b is a N vector of bias currents for each neuron; $\sigma\zeta(t)$ is a Wiener noise process with standard deviation σ after 1s; and τ_{mem} and τ_{syn} are the membrane and synaptic time constants, respectively. $S_{rec}(t)$ is a vector containing `1` for each neuron that emitted a spike in the last time-step. W_{rec} is a recurrent weight matrix, if recurrent weights are used. b is an optional bias current per neuron (default 0.).

On spiking:

When the membrane potential for neuron j, $V_{mem,j}$ exceeds the threshold voltage V_{thr}, then the neuron emits a spike. The spiking neuron subtracts its own threshold on reset.

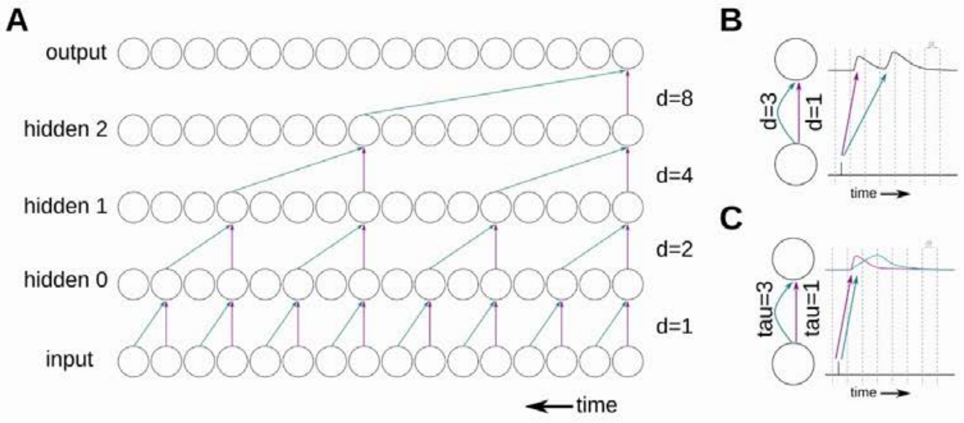
$$
\begin{aligned}
V_{mem,j} > V_{thr} \rightarrow S_{rec,j} = 1 \\
V_{mem,j} = V_{mem,j} - V_{thr}
\end{aligned}
$$

rockpool.ai

WaveSense: Training a Spiking Neural Network with Temporal Convolutions

In this notebook we will demonstrate how to create and train a WaveSense network as described in https://arxiv.org/pdf/2111.01456.pdf.

The key feature of this model is its temporal convolution layer which is inspired by the famous WaveNet architecture from Google: https://arxiv.org/pdf/1609.03499.pdf and https://deepmind.com/blog/article/wavenet-generative-model-raw-audio



WaveNet uses so called "dilated convolutional layers" as depicted in panel A. Dilated convolutional layers are basically the same as a normal convolutional layers, execpt that the kernel is causal and sparse in the time domain. By choosing the sparseness (dilations) in a smart way, no information is lost but

rockpool.ai

🐝 Overview of the Xylo family

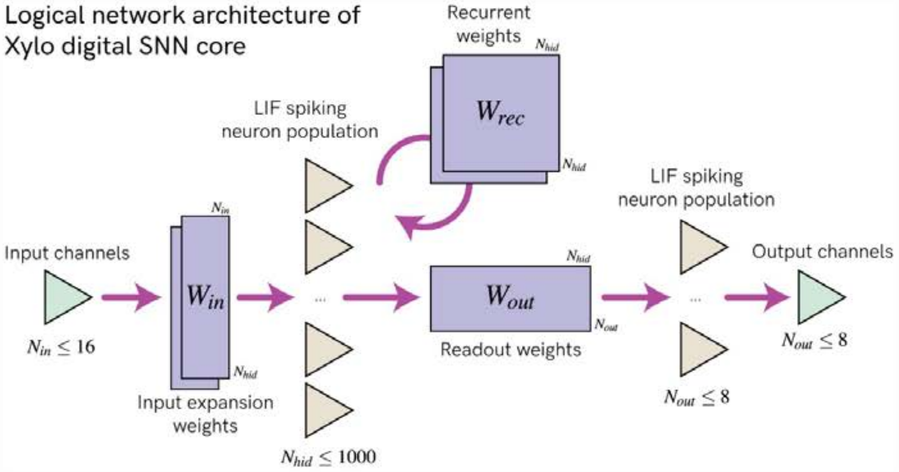
Xylo is a family of spiking neural network ASICs, for efficient simulation of spiking leaky integrate-and-fire neurons with exponential input synapses. Xylo is highly configurable, and supports individual synaptic and membrane time-constants, thresholds and biases for each neuron. Xylo supports arbitrary network architectures, including recurrent networks, residual spiking networks, and more.

Xylo is currently available in several versions with varying HW support and front-ends.

[1]:
```
# - Image display
from IPython.display import Image

Image("images/xylo_network-architecture.png")
```

[1]:



rockpool.ai

Overview of Dynap-SE2

This tutorial provides an overview of Dynap-SE2 mixed signal architecture. If you're familiar with the chip and looking for a hands-on tutorial, please see Dynap-SE2 Quick Start tutorial.

Otherwise, let's deep dive into the chip!

Introduction

Dynap-SE2, (DYnamic Neuromorphic Asynchronous Processor - ScalablE 2) inherits the event-driven nature of the DYNAP family. The mixed-signal chip uses analog spiking neurons and analog synapses as the computing units, which directly emulates biological behavior. Transistors of the neural cores operate in the subthreshold region, which results in power consumption of about one-thousandth to one-millionth of the state-of-the-art digital neuromorphic chips, below mW.

Each chip features:

- **1024 AdExpIF** (adaptive exponential integrate-and-fire) analog ultra-low-power spiking neurons,
- **64 synapses** per neuron with configurable delay, weight, and short-term plasticity.

[9]:

```
from IPython.display import Image
Image("images/dynapse2.jpeg")
```

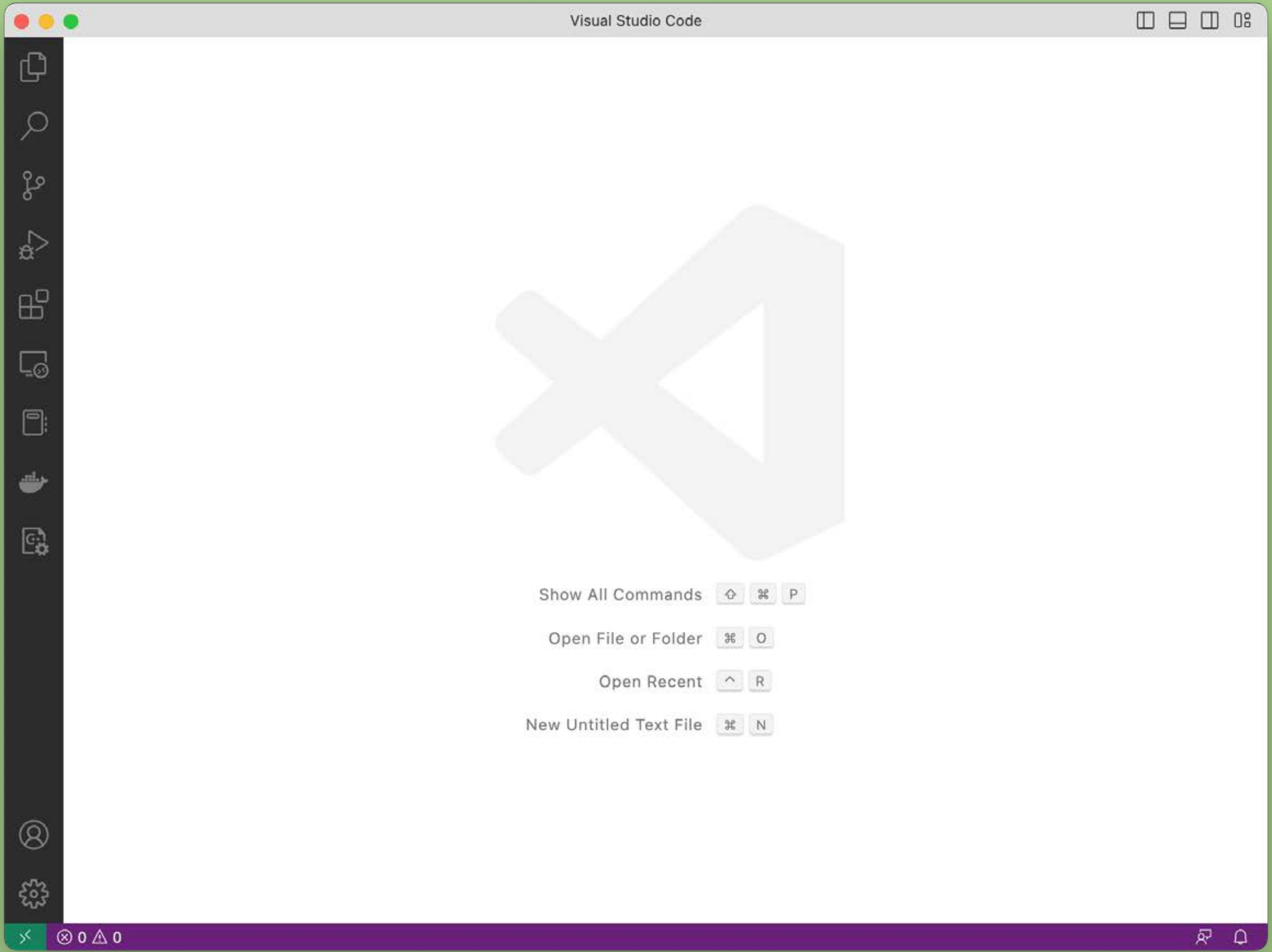
[9]:

Terminal

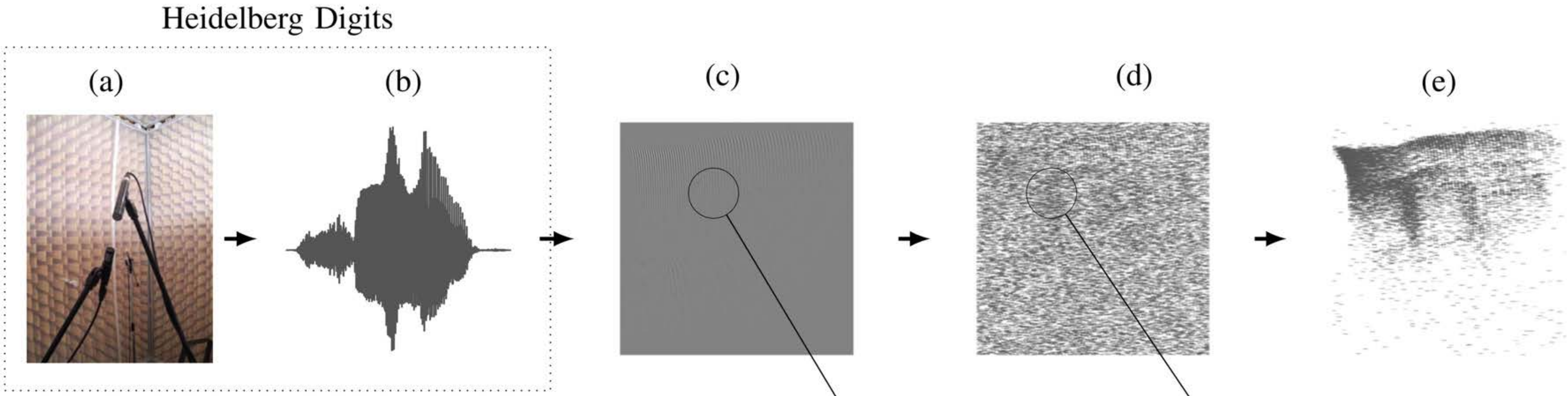
```
(base) dylan@Statics ~ % conda create --name rp rockpool -c conda-forge
```

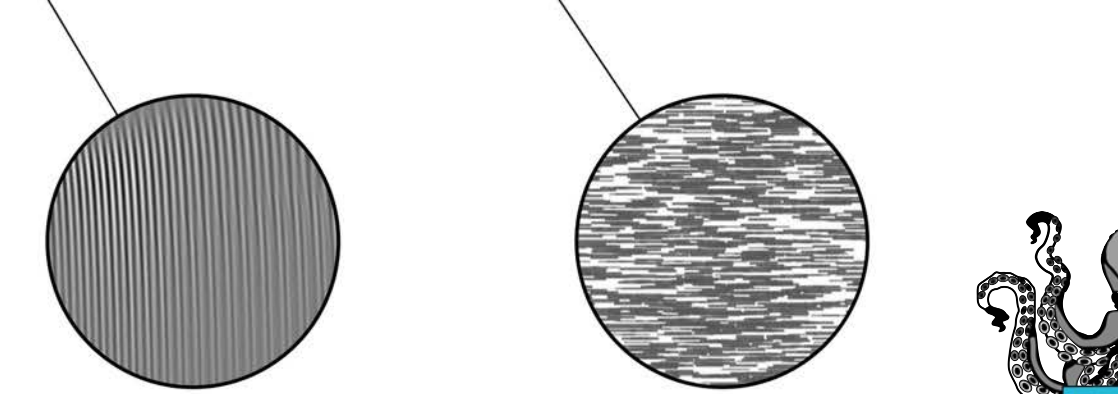
Terminal

```
(base) dylan@Statics ~ % pip install rockpool
```


Visual Studio Code
Show All Commands ⇧ ⌘ P
Open File or Folder ⌘ O
Open Recent ^ R
New Untitled Text File ⌘ N
0 0

Spiking Hiedelberg Digits





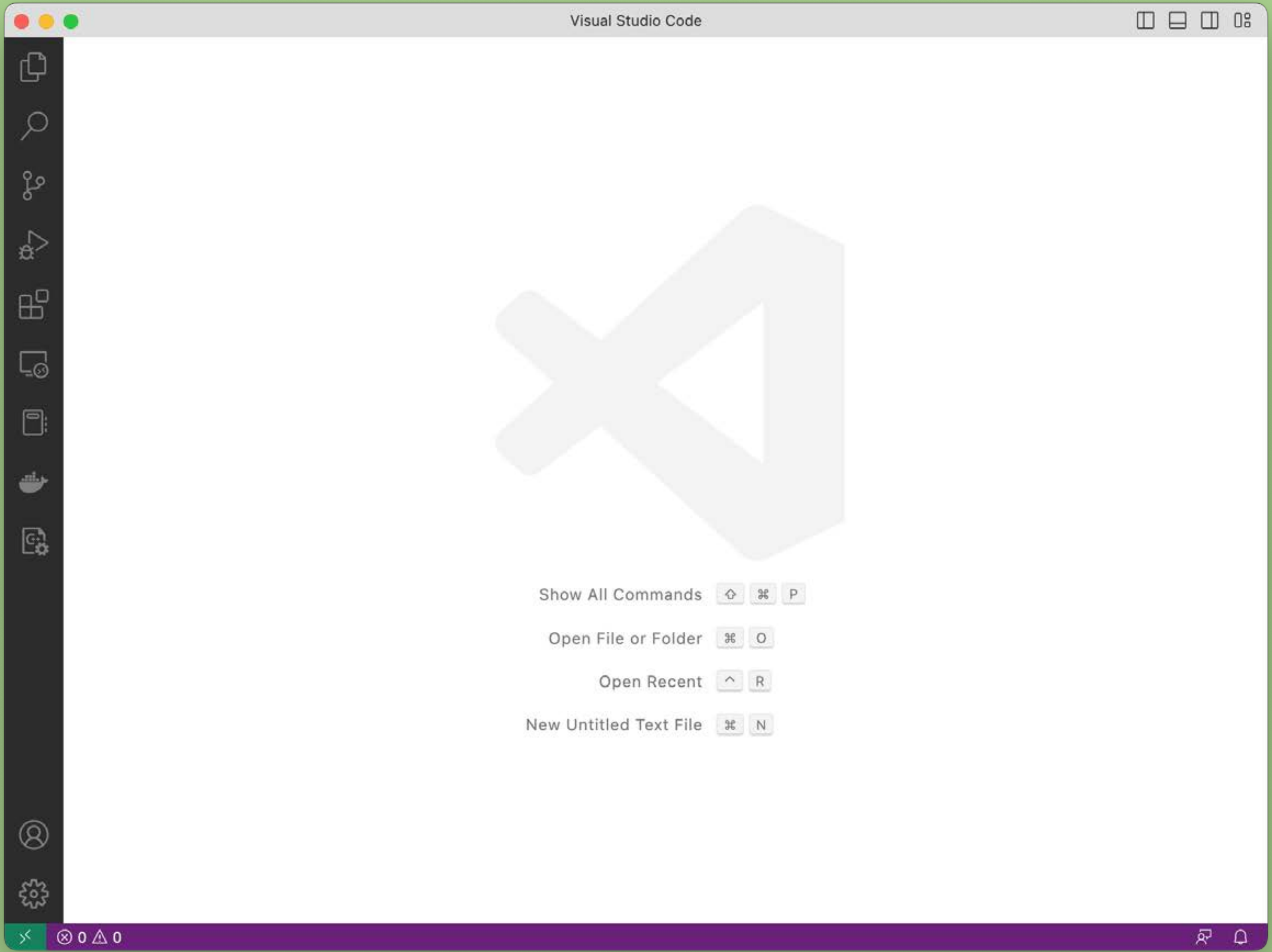
- Spoken digits, English and German
- 20 classes
- Pre-processed with highly detailed basilar membrane / cochlea model
- Input data provided as spike events

Cramer et al. 2022. IEEE Trans. NNLS 33 2744-2757. [10.1109/TNNLS.2020.3044364](https://doi.org/10.1109/TNNLS.2020.3044364)

Tonic

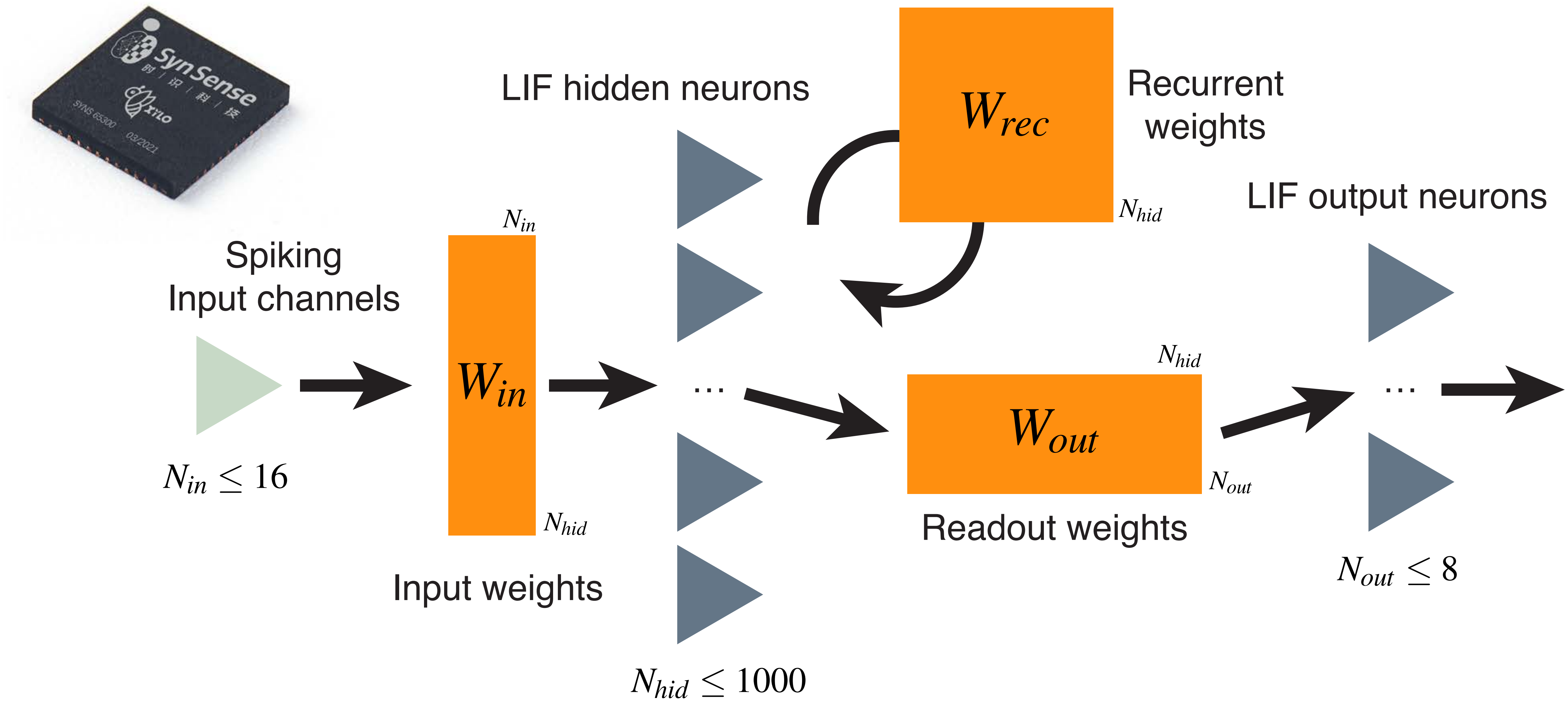
tonic.readthedocs.io

- Neuromorphic datasets
  - SHD, S-MNIST, DVSGesture, …
- Data transformations
- Data augmentation
- Caching
- Open source ❤️
- …

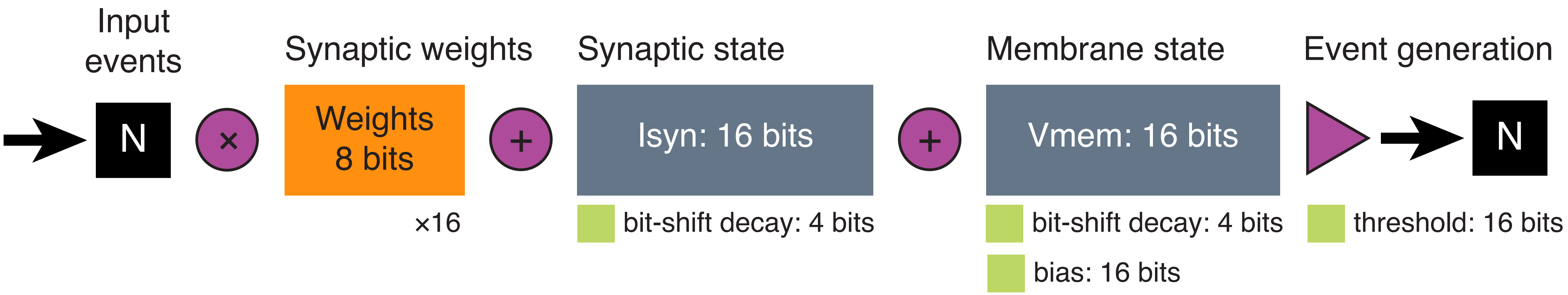

Visual Studio Code
Show All Commands ⇧ ⌘ P
Open File or Folder ⌘ O
Open Recent ^ R
New Untitled Text File ⌘ N
0 0

Xylo digital SNN architecture

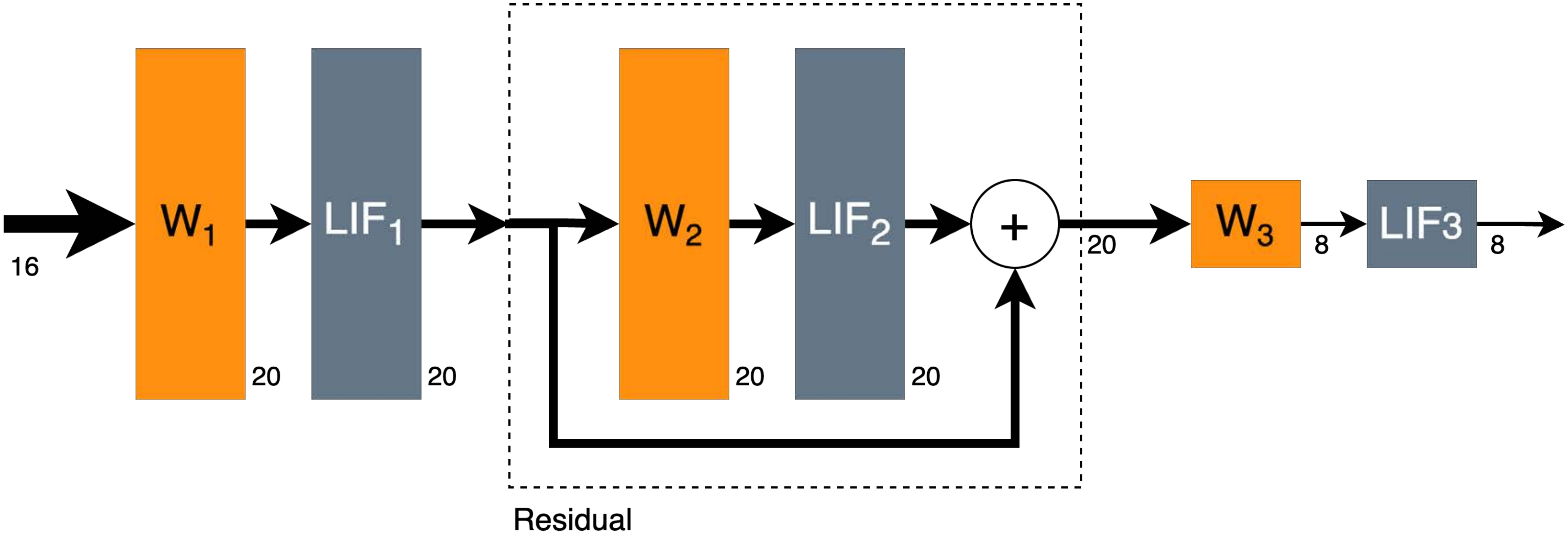
Logical network architecture



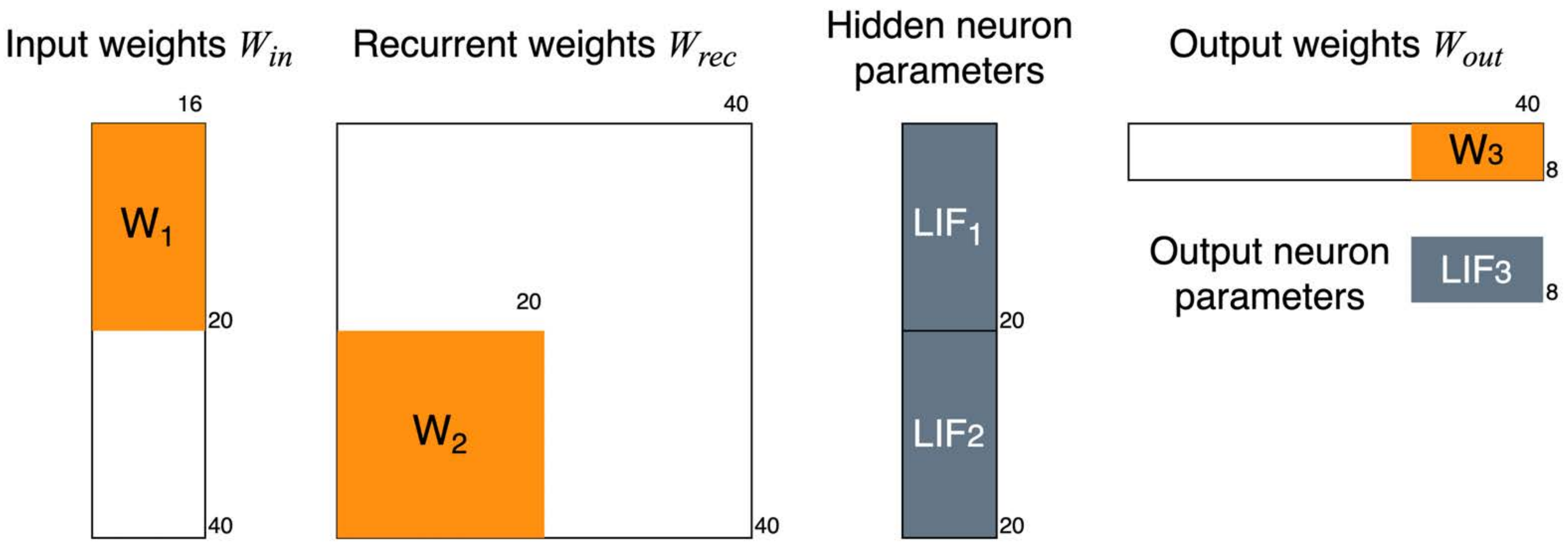
Xylo digital SNN architecture
Digital LIF Neuron

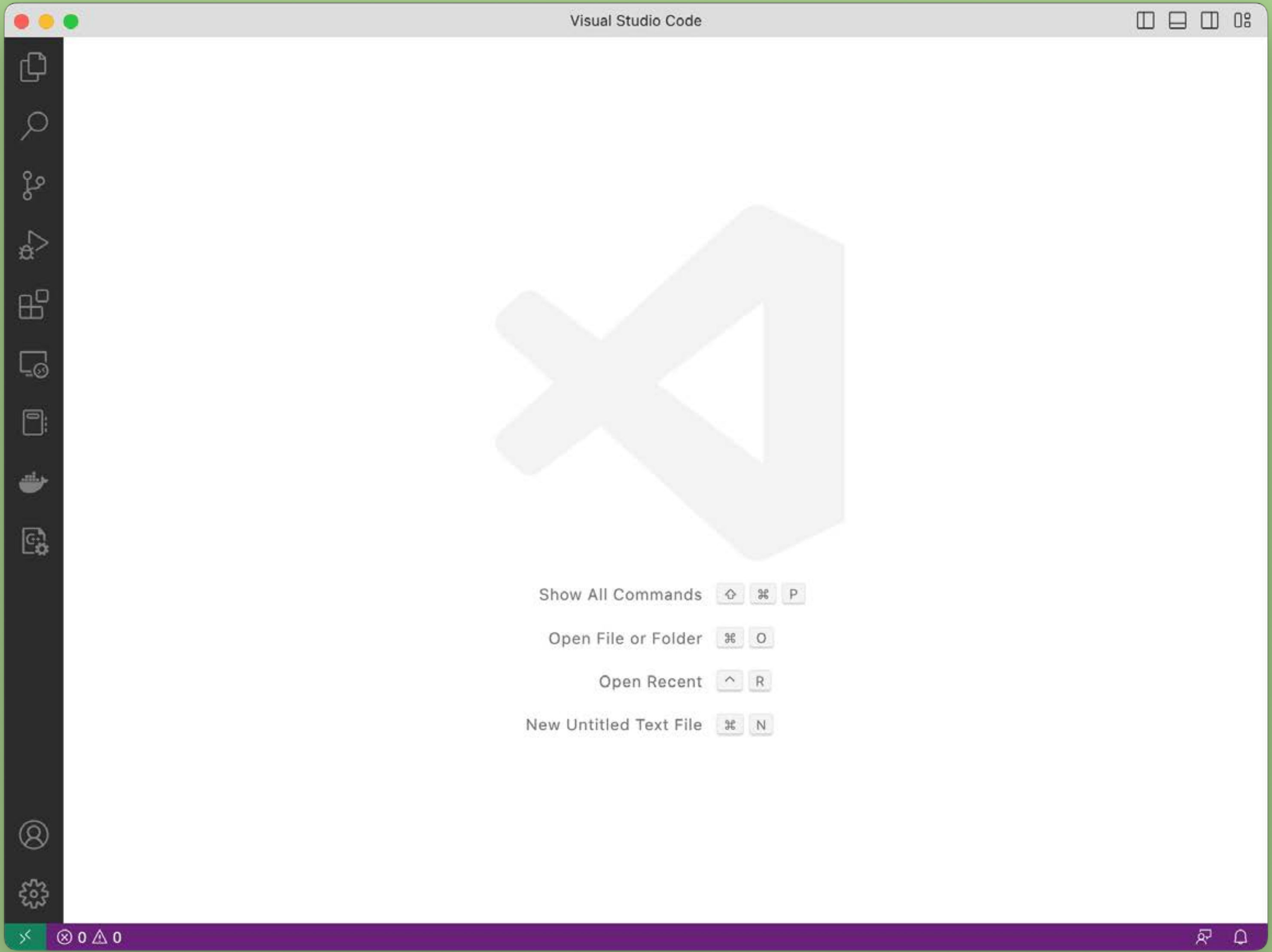


Network architecture



Mapping process




Visual Studio Code
Show All Commands ⇧ ⌘ P
Open File or Folder ⌘ O
Open Recent ^ R
New Untitled Text File ⌘ N
0 0

Rockpool

Est. 2019

Artisanal signal processing

- PyTorch, Jax backends
- GPU/CUDA, TPU, MPS acceleration
- Constrained optimization for SNNs
- Time constant & threshold training
- Deployable adaptive LIF models
- Quantization-aware training
- Mixed-signal HW-aware training
- Easily extensible
- Open source ❤️
- ...