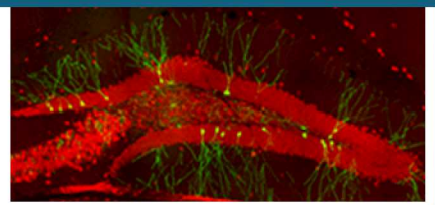
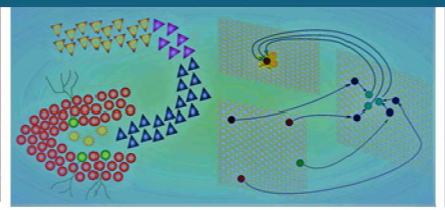
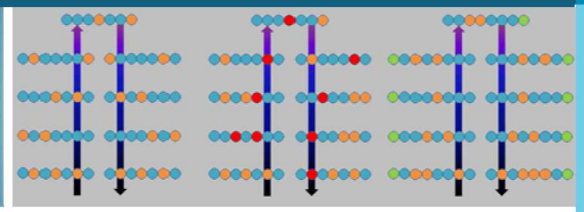
SAND2019-10694C

Fugu





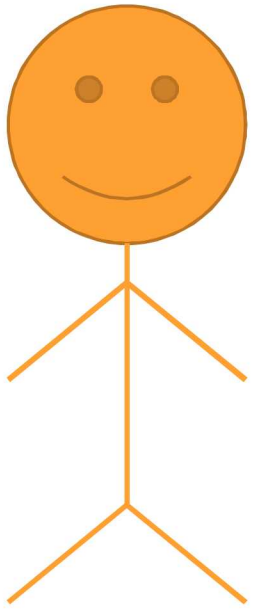


Brad Aimone, William M. Severa, Srideep Musuvathy,
Craig M. Vineyard, Leah Reeder, Frances Chance, Felix Wang

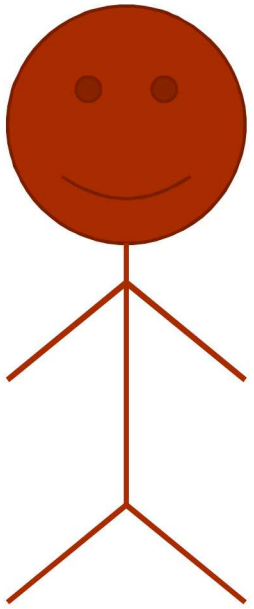
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Spiking neuromorphic hardware will have significant, widespread impact if we can both demonstrate compelling utility (algorithms), and facilitate usability (software)

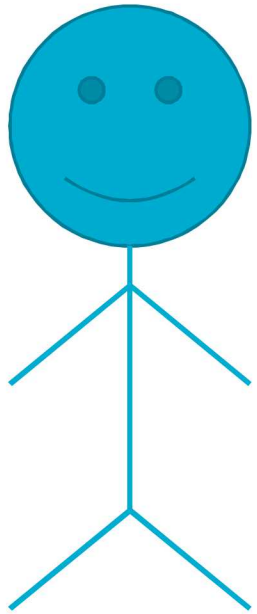
3 types of neuromorphic users



Typical Computer Scientist

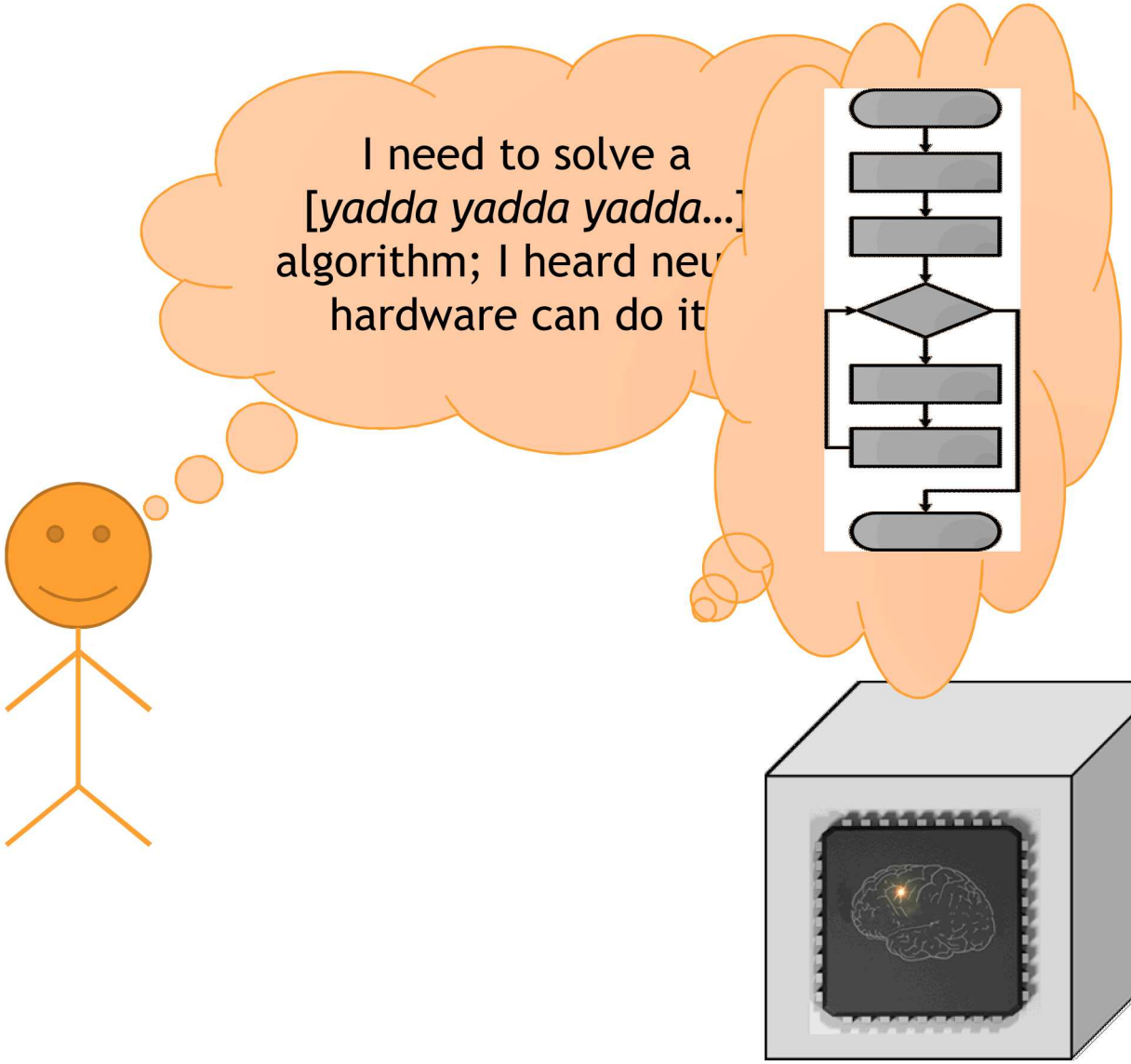


Neural Algorithms Researcher



Neural Architecture Developer

3 types of neuromorphic users…

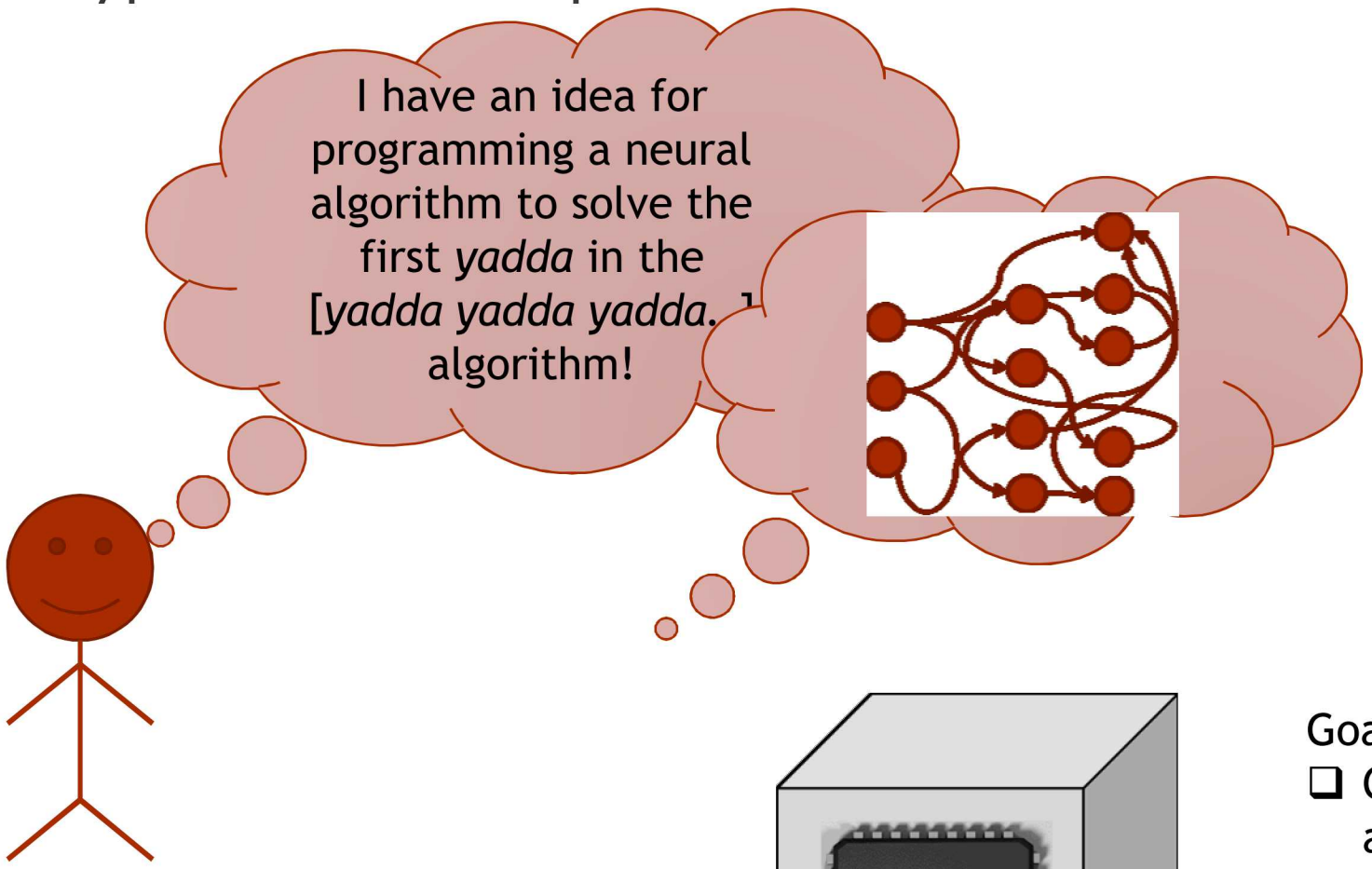



```
Algorithm yadda yadda yadda
  Initialize with data
  a=f1(start)
  b=f2(a)
  while b != 0
    c=f3(b)
    b=f4(c)
  end
```

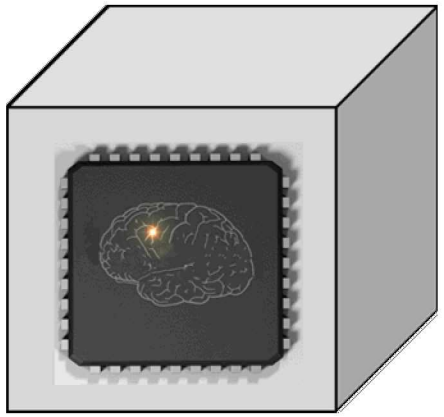
Goals for this user:

- ❑ Never have to learn anything about neurons!
- ❑ Program like any other machine
- ❑ Take advantage of libraries and great performance!

3 types of neuromorphic users…



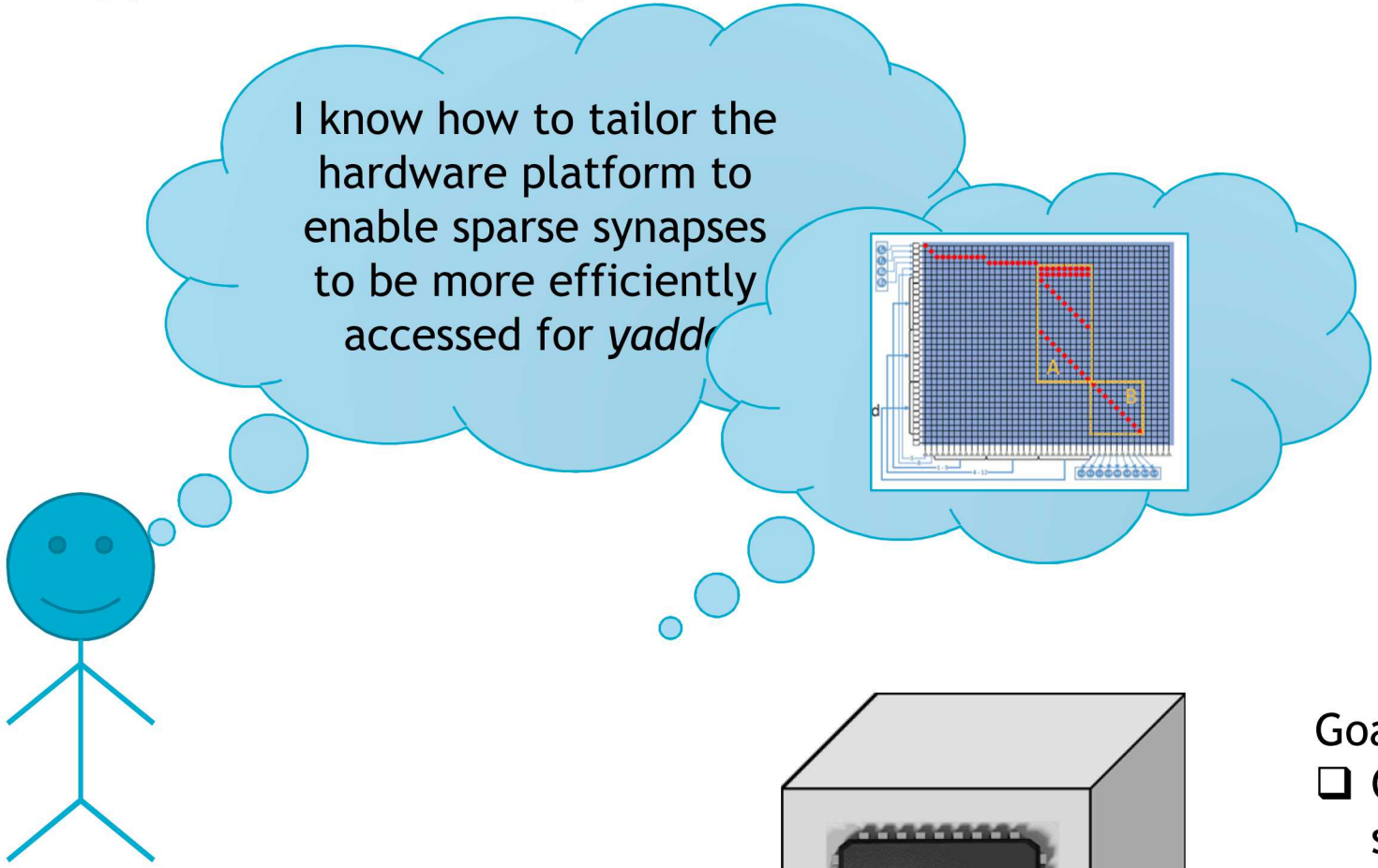

```
Neural algorithm yadda
  Initialize neurons
  pop1 = neurons(3)
  pop2 = neurons(4)
  pop3 = neurons(3)
  pop4 = neurons(2)
  conn1_2 = synapses(5)
  conn2_3 = synapses(2)
```



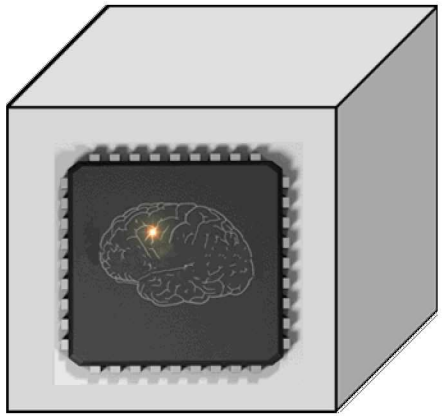
Goals for this user:

- ❑ Can create spiking neural algorithms!
- ❑ Program independently of underlying hardware
- ❑ Create libraries for first set of users.

3 types of neuromorphic users…




```
Module neuralCore(pop, syn)
for n in core
     neuron(n).v=pop(n).v_ini
     neuron(n).v_t=pop(n).v_t
for s in core
     synapse(s).w=syn(s).w
     synapse(s).source=syn(s)
     synapse(s).targ=syn(s).t
```



Goals for this user:

- ❑ Can compile algorithms onto specific hardware platforms
- ❑ Interested in optimizing algorithms for hardware constraints
- ❑ Create libraries for first set of users.

Fugu aims to bring neuromorphic solutions to general computing world

Typical Computer Scientists

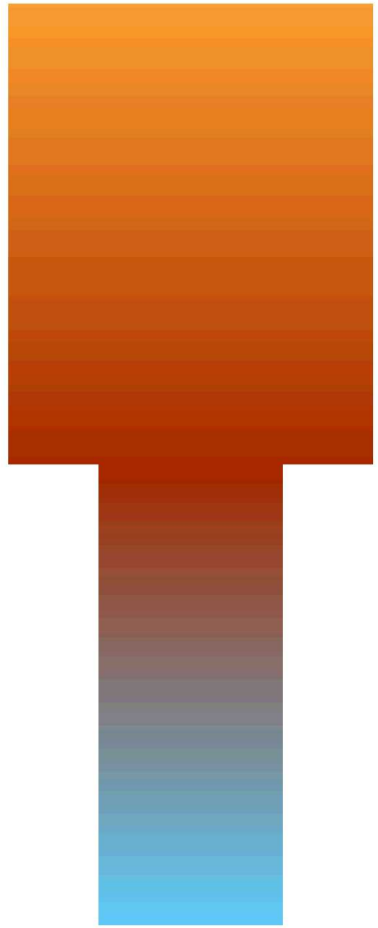
Wants to program with libraries

Neural Algorithm Researcher

Wants to program with neurons

Neural Architecture Developer

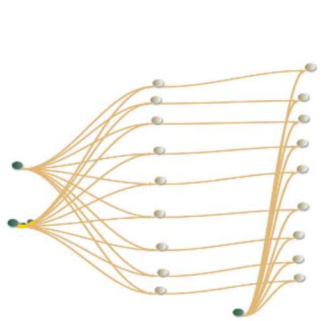
Wants to program hardware directly

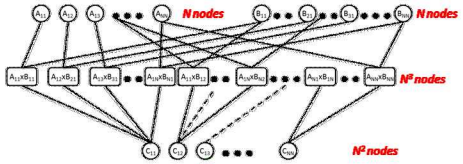


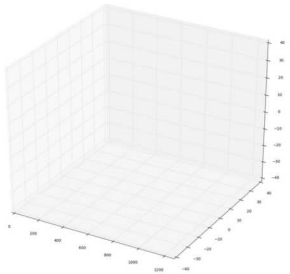
Concept and Approach

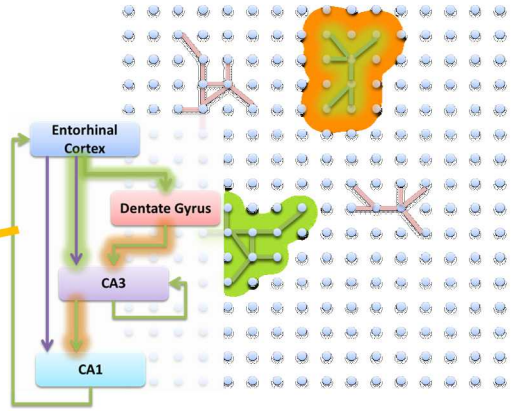
Spiking neuromorphic hardware will have significant, widespread impact if we can both demonstrate compelling utility (algorithms), and facilitate usability (software)

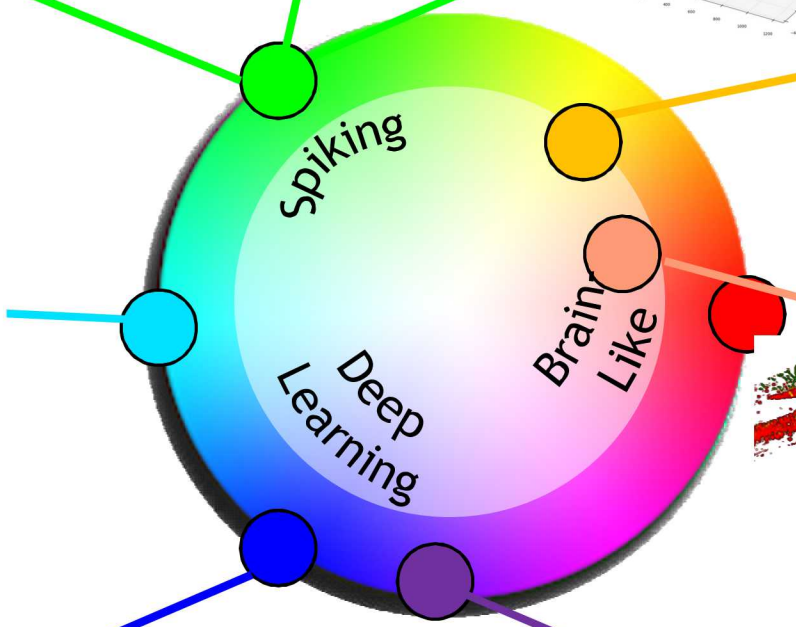
Potential for neuromorphic computing may extend farther than anticipated

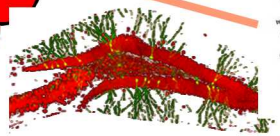


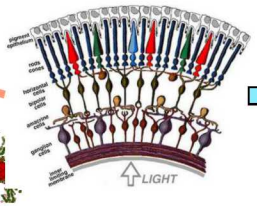


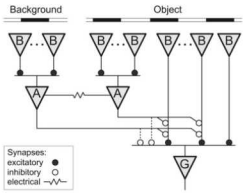


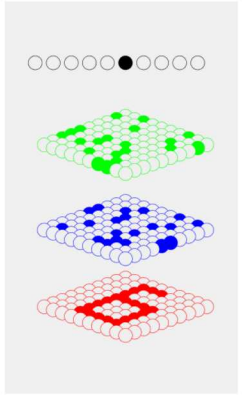


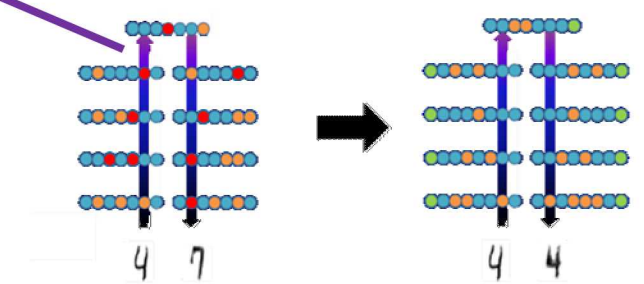












Ever growing list of reasonable spiking neural algorithms and kernels

Machine Learning

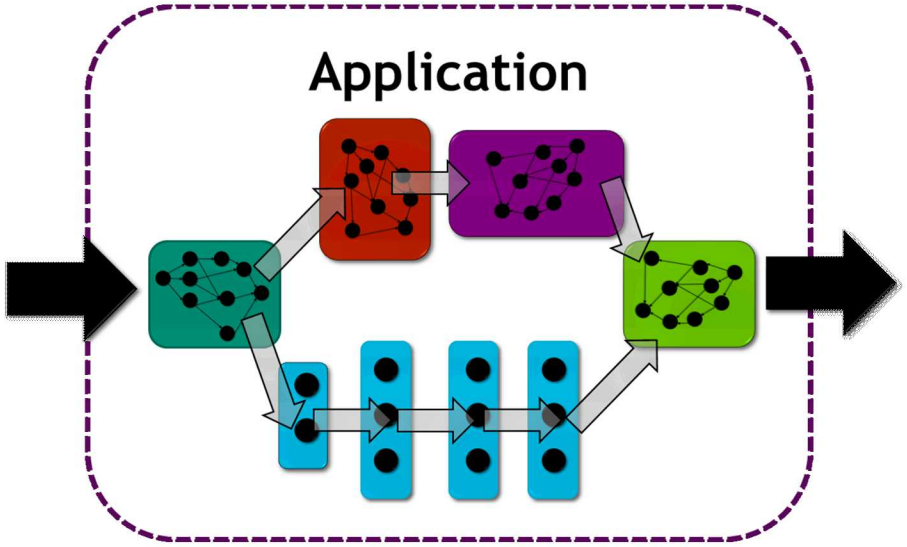
Whetstone

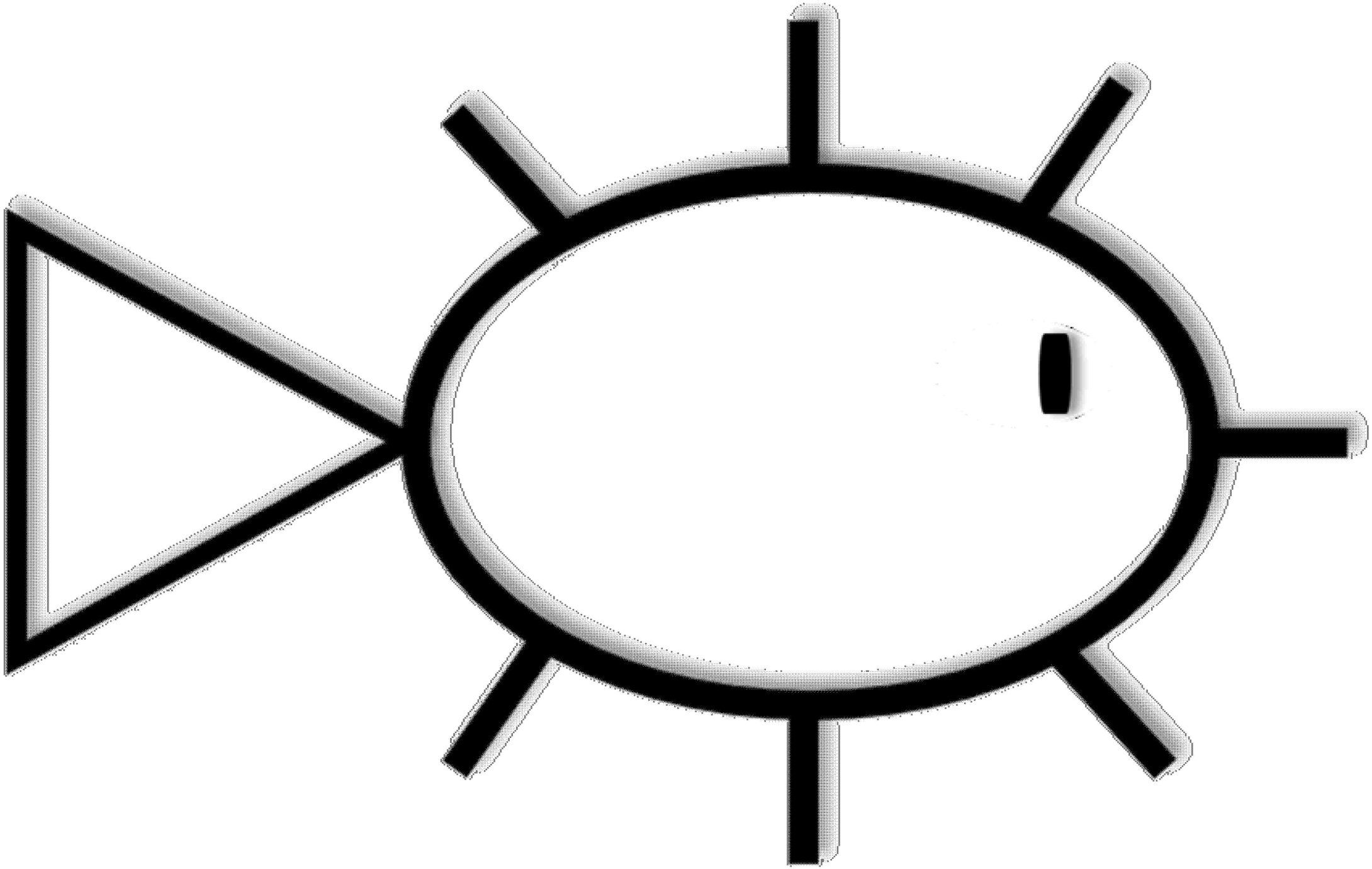
Convolutions

k-Nearest Neighbor

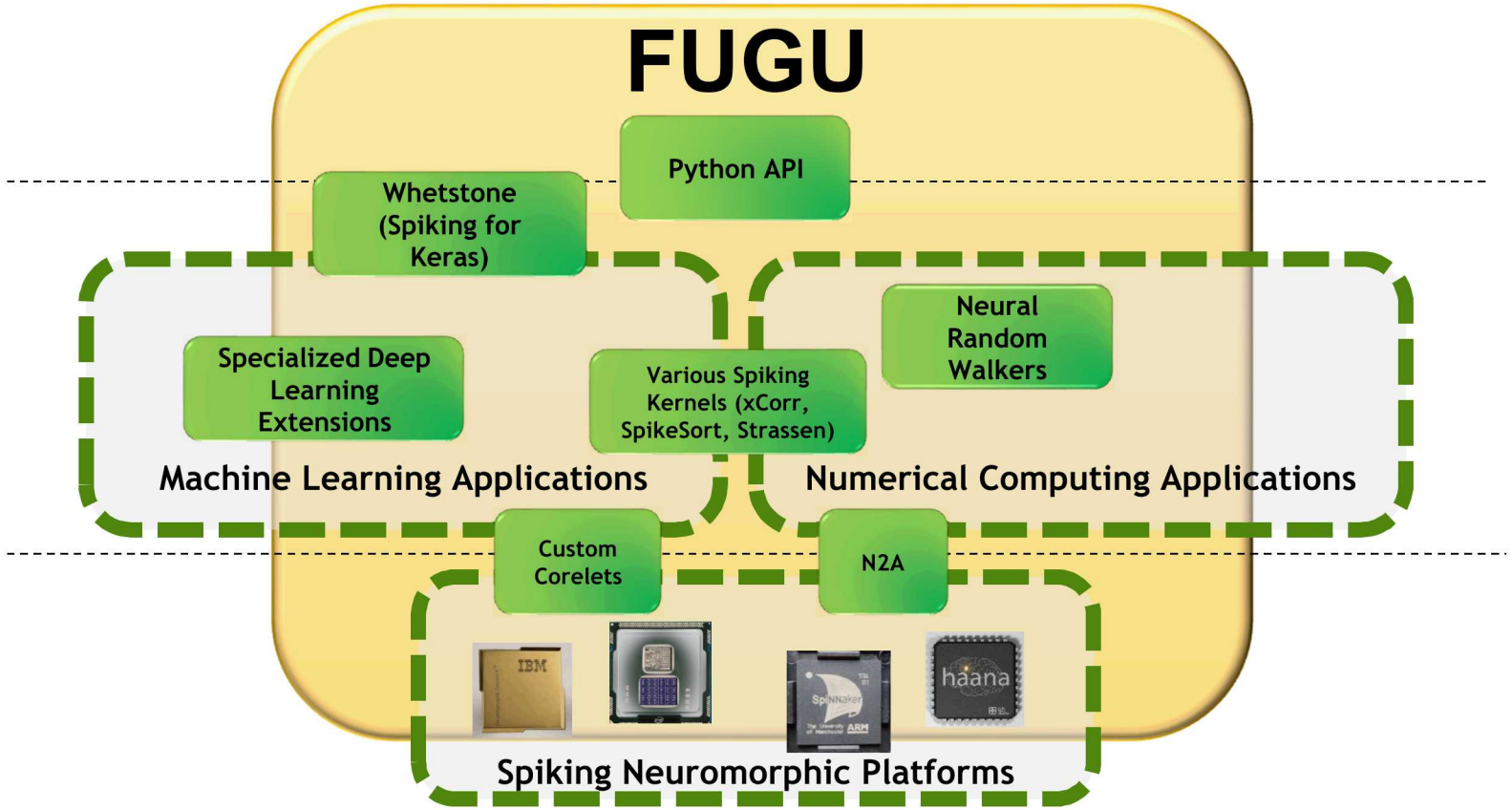
Support Vector Machines

- Many 'kernels' used for common neural computation are important for conventional algorithms as well
- Neural hardware is capable of reasonable performance on many non-ML kernels as well
- View neural algorithms as ***composable*** from linking together neural circuits to solve broadly useful kernels





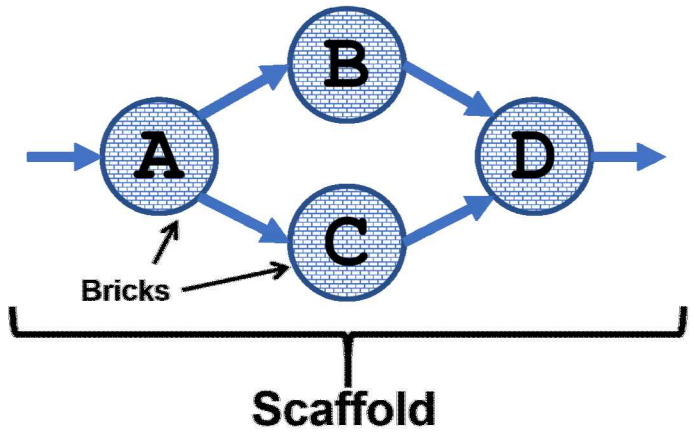
Fugu Overview

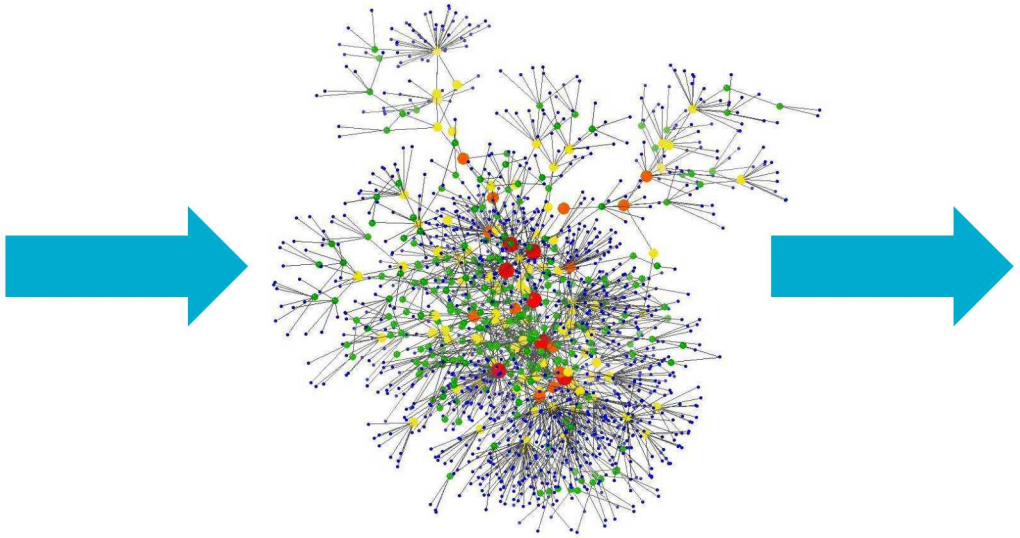


Fugu Overview - Challenges

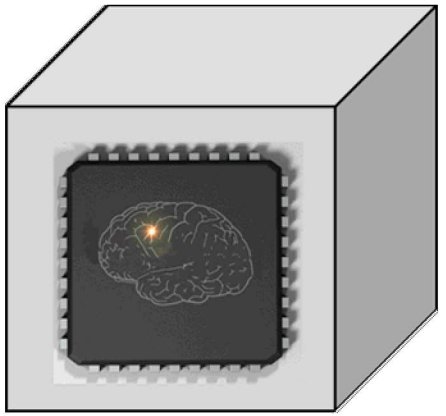
- For spiking neural networks, it is (very) hard to
  - Implement someone else's network
  - Integrate multiple kernels into an algorithm
  - Port networks designed for one platform to another

```
Algorithm
Insert Brick A, input IN     //A=f_A(in)
Insert Brick B, input A      //B=f_B(A)
Insert Brick C, input A      //C=f_C(A)
Insert Brick D, input B, C   //D=f_D(B, C)
```





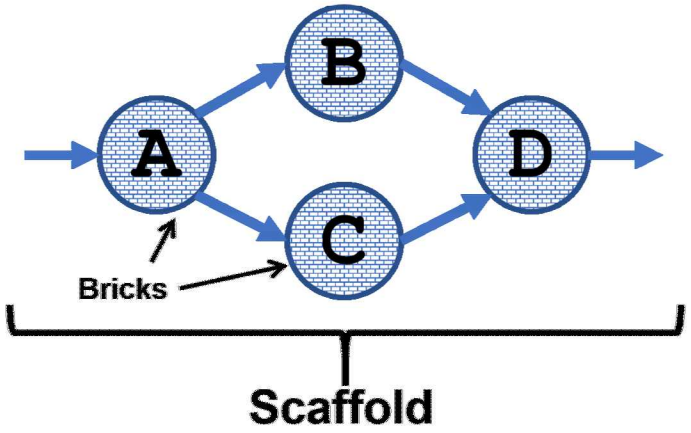
Not actual syntax or network

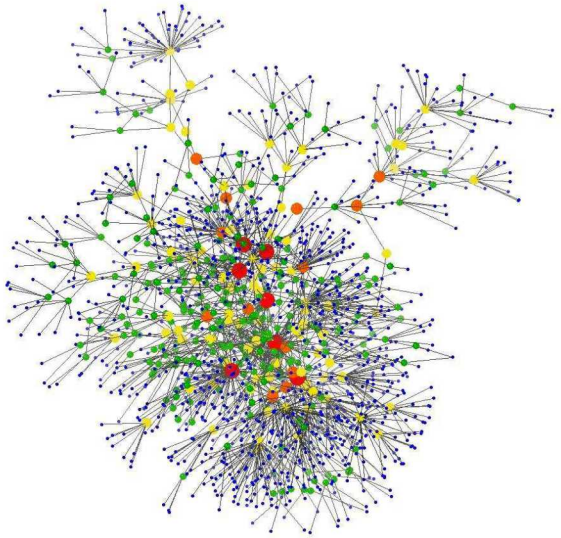


Fugu Overview - Goals

- *Common linking framework for implementing spiking algorithms*
  - Leverage and combine community's recent progress
- *Hardware-independent intermediate representation*
  - Ability to rapidly change platforms
- *Procedural definition of network connectivity*
  - Kernels and algorithms can scale according to problem size
- *Flexible, pre-determined communication methods*
  - Simple interactivity

```
Algorithm
Insert Brick A, input IN    //A=f_A(in)
Insert Brick B, input A     //B=f_B(A)
Insert Brick C, input A     //C=f_C(A)
Insert Brick D, input B, C  //D=f_D(B, C)
```





Not actual syntax or network

Fugu Overview - What Fugu *is*

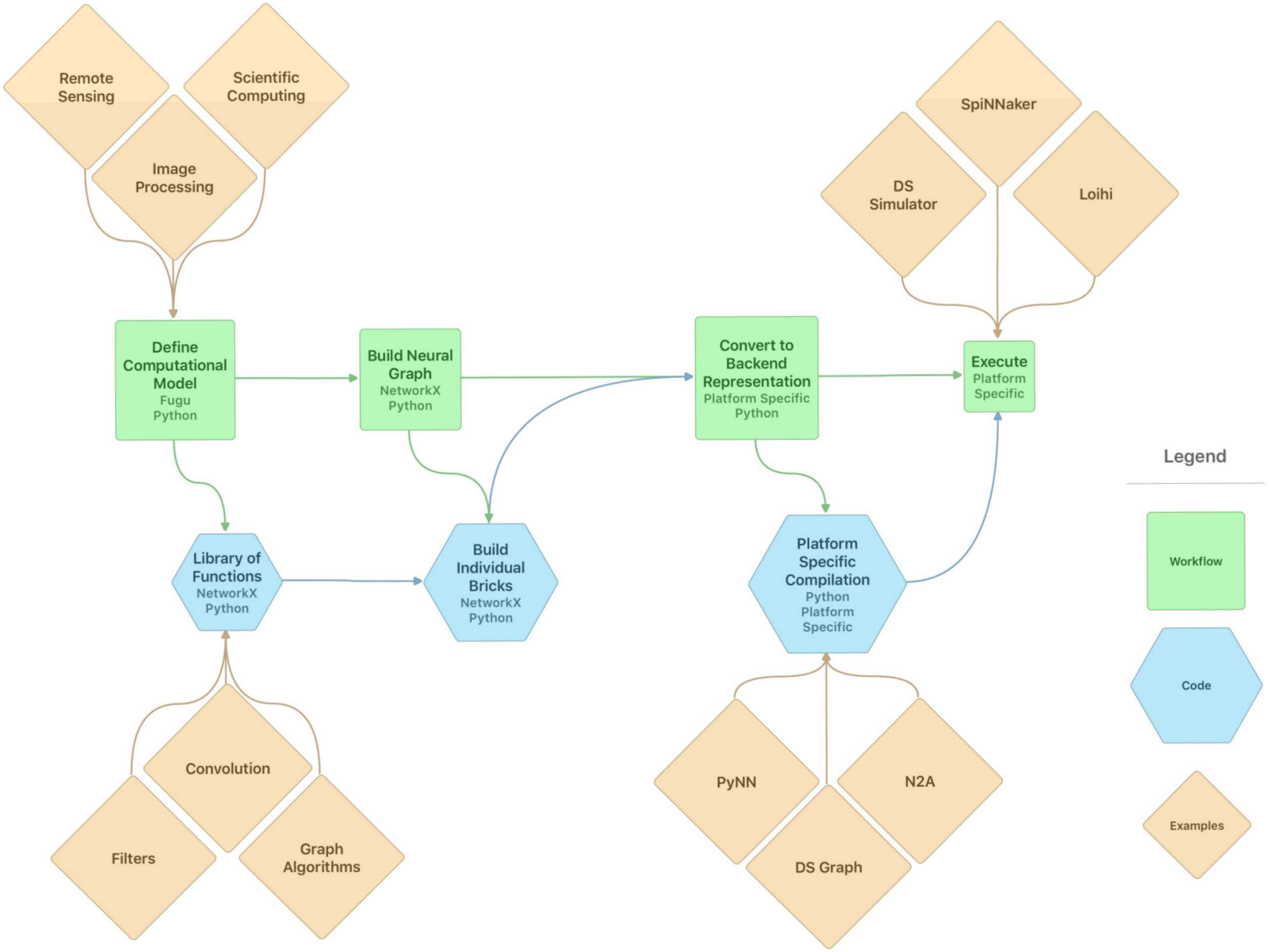
- Fugu is a linking framework
  - It's "easy" to build spiking circuits for a single computation
  - It's hard to do application-level computation on neuromorphic
  - We provide a mechanism to combine small computational kernels (Bricks) into large computational graphs
- Fugu is a spec
  - For the Bricks to transfer information, we need to agree on data formatting
  - For computation to be consistent, we need to agree on neuron behavior (lowest common denominator*)
  - For this to be useful, we need a hardware independent intermediate representation

*Usually, for most cases

Fugu Overview – What Fugu *is not*

- Fugu includes but is NOT a simulator
  - Uses a reference simulator ‘ds’ which can quickly run small-medium sized spiking networks
  - ds instantiates the fugu neuron model (discrete time, point synapses)
  - Fugu is designed to support a variety of backends including hardware platforms
- Fugu includes but is NOT a spiking algorithm
  - The goal of Fugu is to have a library of Fugu Bricks for many kernels
  - *We’re hoping that the community will help contribute*
- Fugu includes but is NOT a graph utility
  - NetworkX provides (nearly) all of our graph functionality
  - Node and edge properties are inherent in NetworkX and only become meaningful when interpreted by a backend

Fugu Overview



Software Design and Organization

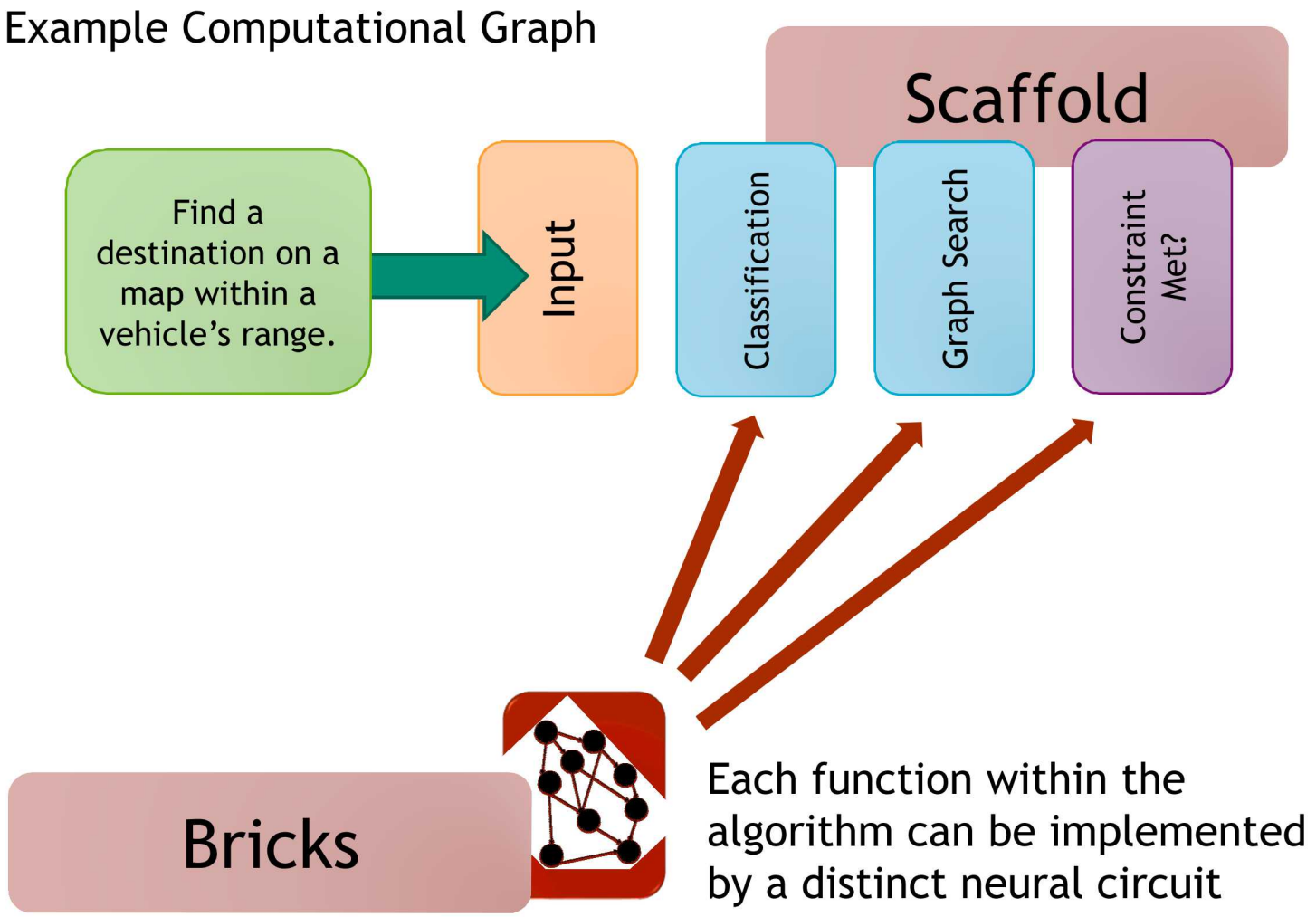
Software Design

- Fugu exists!
  - Still a ***work in progress***
  - We have begun to partner with other Sandia and external laboratory groups (LLNL and LANL)
  - We will eventually open source and integrate with other tools

Software Design

- Building a computational graph
- Fugu algorithms are designed using computational directed acyclic graphs
  - Nodes → Functions
  - Edges → Data flow

- Overall Fugu algorithm graph → *Scaffold*
- Neural circuits for functions → *Bricks*

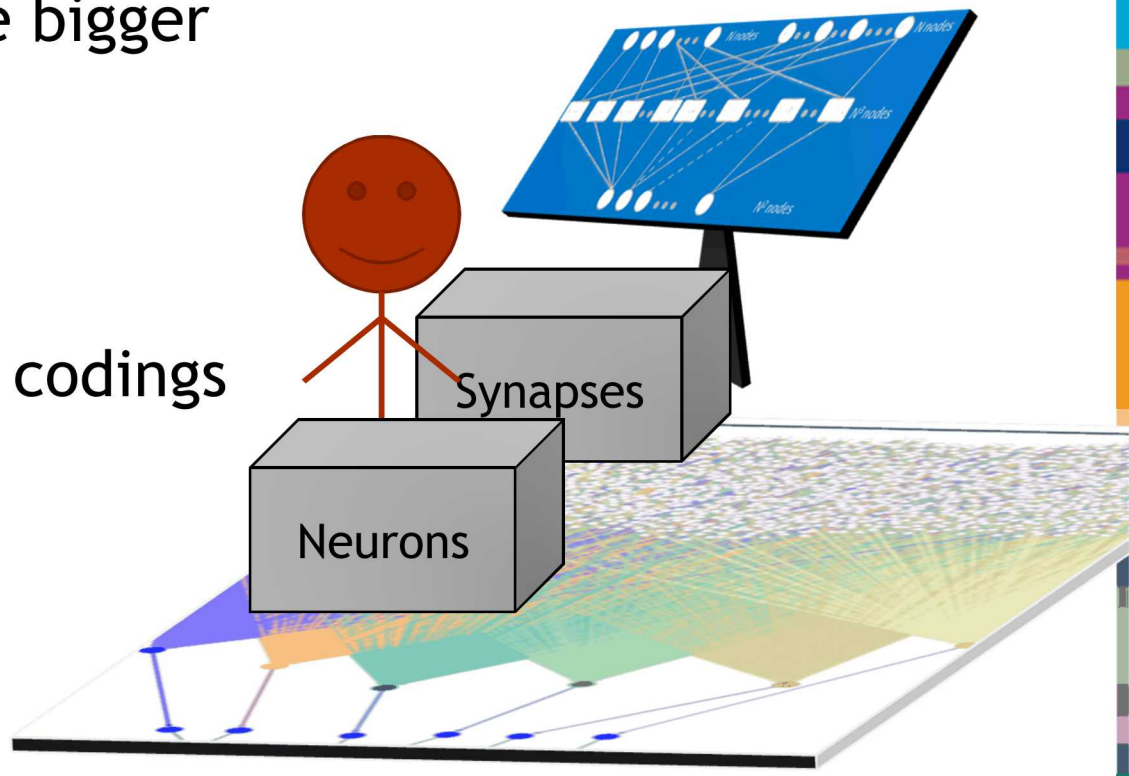


Software Design

- Key Classes:
  - Scaffold
    - Provides the main entry point for people using Fugu
    - Manages the computational graph (`Scaffold.circuit`), metadata, backend, and network graph (`Scaffold.graph`)
    - `Fugu.Scaffold` in `Fugu.py`
  - Brick
    - Represents a fundamental spiking computational kernel
    - Spiking algorithms should inherit from Brick (or one of its subclasses); `Fugu.Brick` is an abstract class
    - Responsible for building a portion of the network graph
    - `Fugu.Brick` in `Fugu.py`

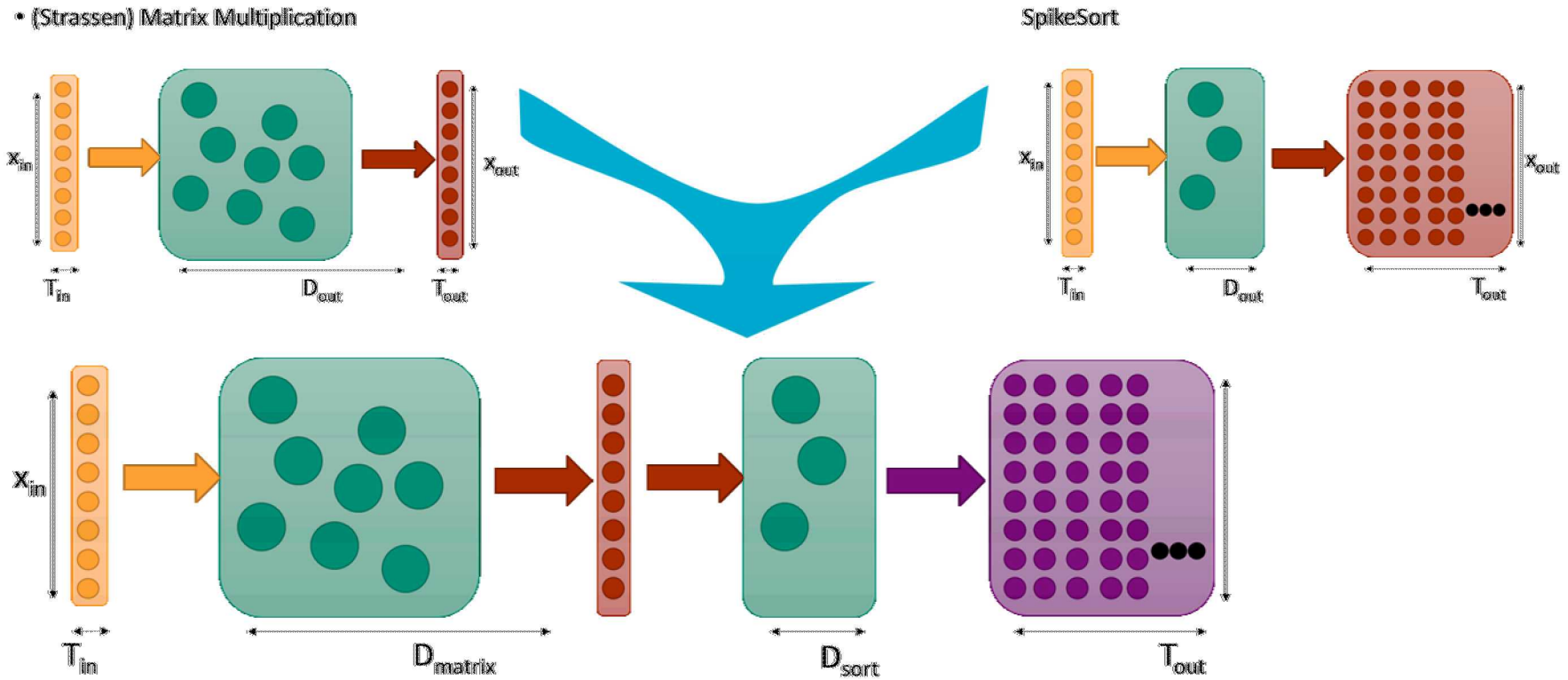
Software Design – More about Bricks

- Fugu will contain a growing library of bricks
- Bricks can be linked together to compose bigger algorithms
- Bricks are individually responsible for
  - Building their portion of the graph
  - Adapting to a list of acceptable input codings (unary, binary, temporal, etc)
  - Scaling to the input dimensionality
  - Providing an output in a standard representation
  - Incorporating any specialized components (e.g., learning)



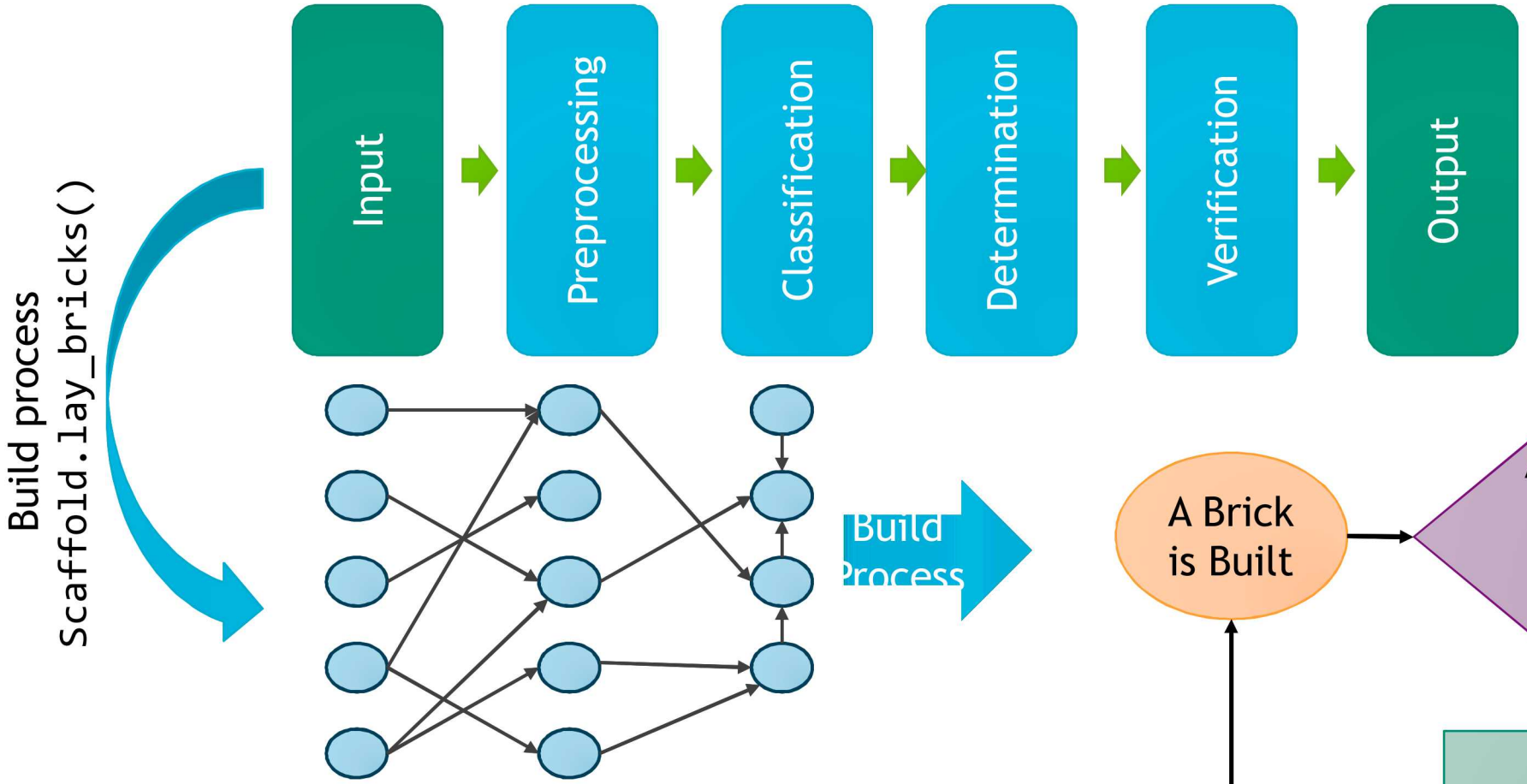
Example of Linking

You can think of the scaffold as linking bricks' graphs together and those graphs adjust as needed.



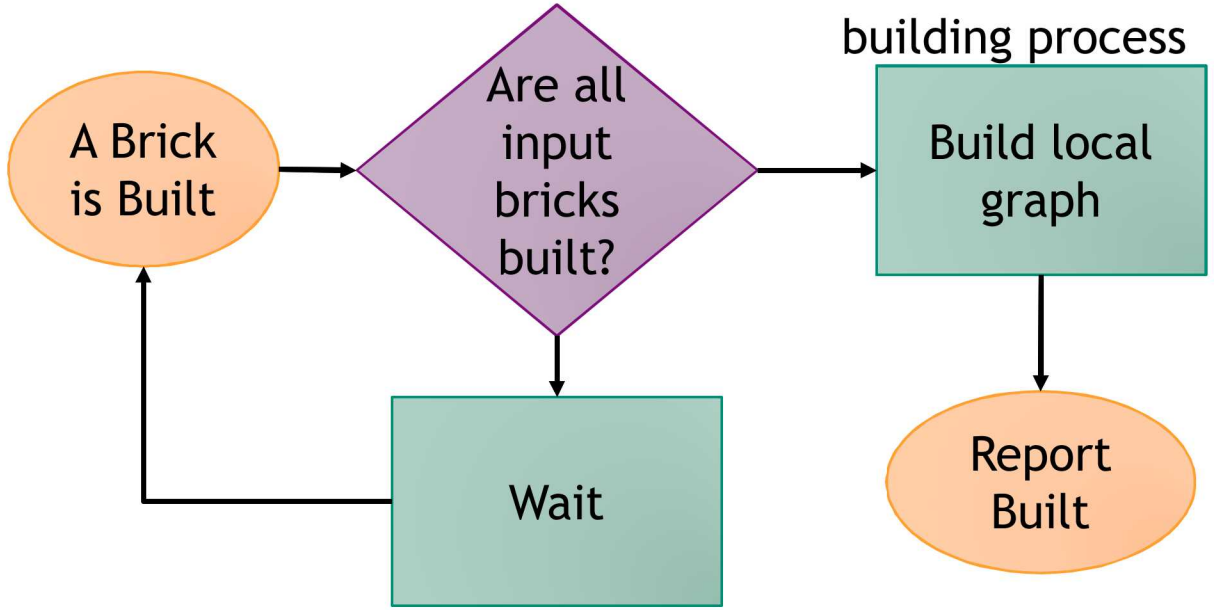
Example of Linking

The scaffold holds references to each brick, 'lays' the bricks iteratively, and each brick builds its portion of the graph when all build conditions are satisfied.



`Scaffold.circuit` is a NetworkX Digraph of subclasses of `Fugu.Brick`

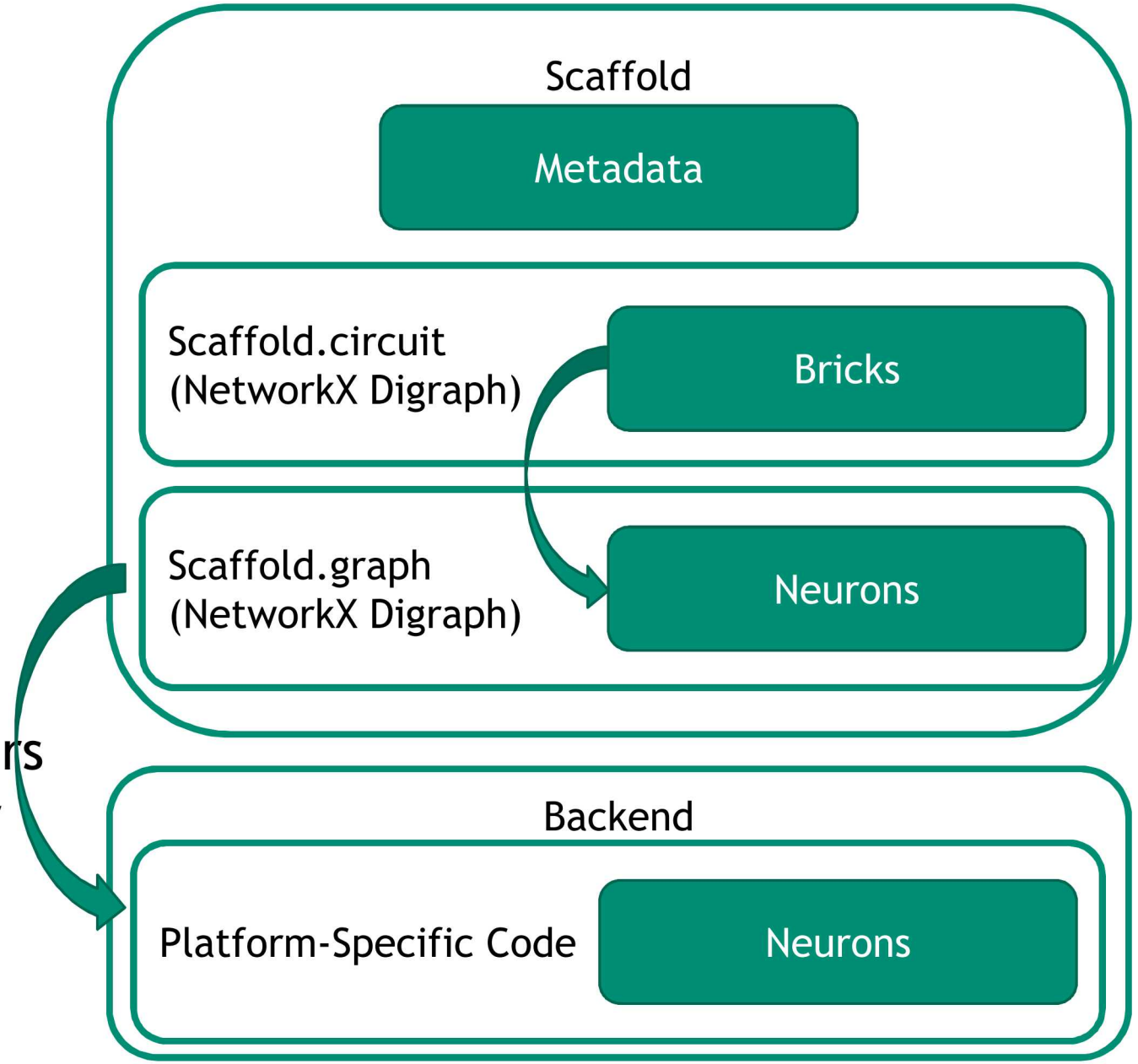
Control Flow for a Bricks' building process



These bricks have been built (i.e. their portions of `Scaffold.graph` are complete).

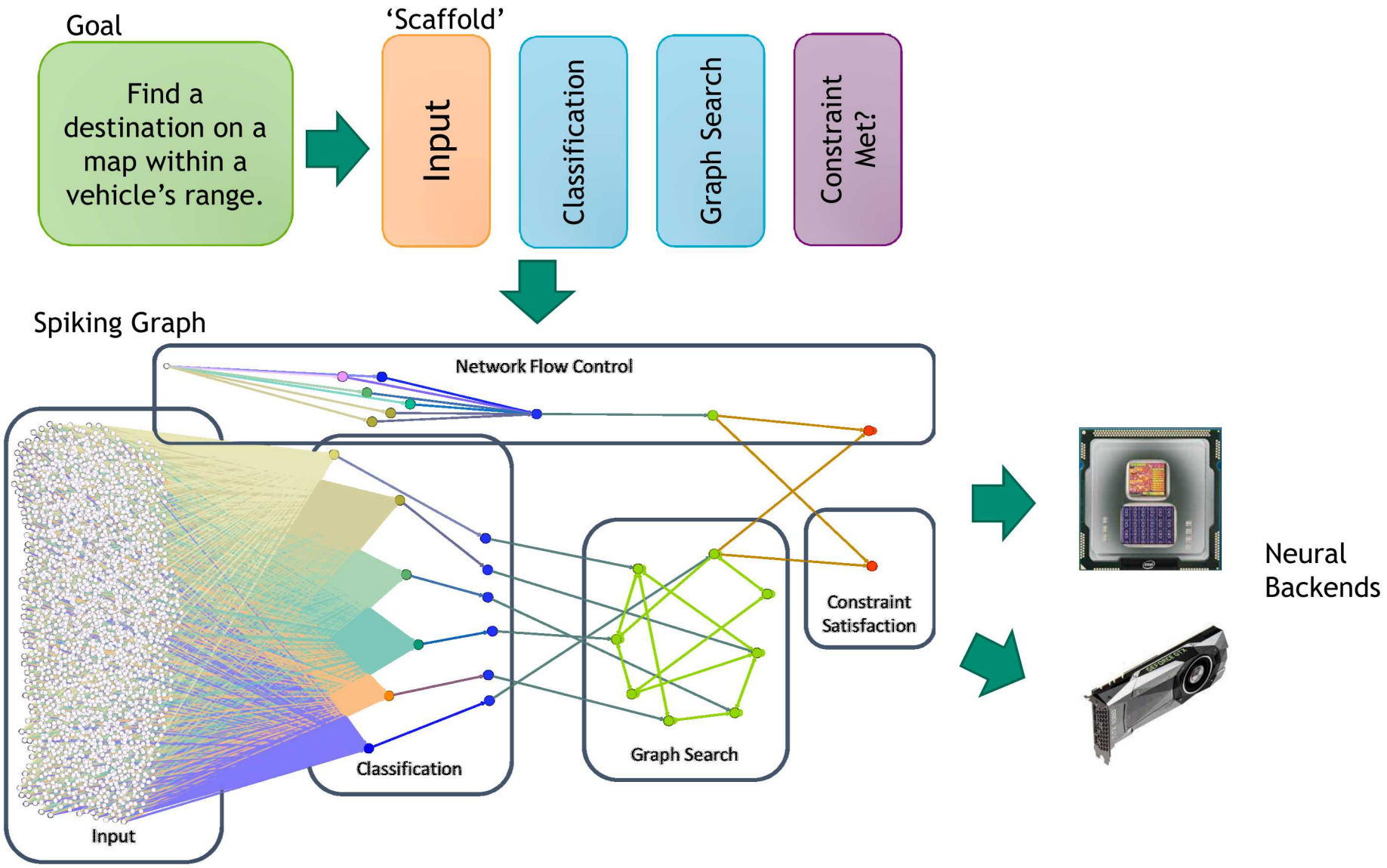
Software Design – More about backends

- A backend generates platform-specific code from the platform-independent network graph (`Scaffold.graph`)
- Included in Fugu today is a basic reference simulator ‘ds’
- The backend handles inputs (represented by input bricks)
- Hardware platform backends can be developed by hardware partners (though we hope to provide a few as well)



Some Sample Results

What the end-to-end Fugu Process looks like




Dash
127.0.0.1:8050
Search
Sandia National Laboratories
scaffold.circuit
decode results
output neurons
Select...
view decoded results here
spike raster
neuron number
time
scaffold.graph
hover over a node to view its attributes
timestep
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energyâ€™s National Security Administration under contract DE-NA0003525.

Questions?