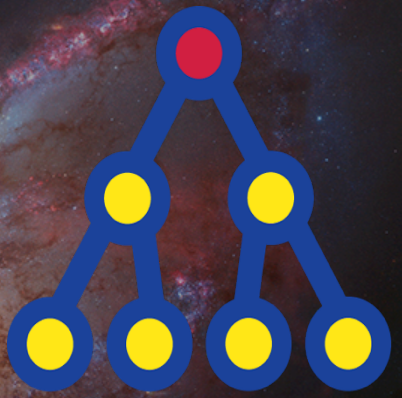
Numenta Platform for Intelligent Computing

(NuPIC)

numenta.org



Matt Taylor
OS Community Flag-Bearer
matt@numenta.org
twitter.com/rhyolight

Scott Purdy
NuPIC Engineer
scott@numenta.org
twitter.com/scottmpurdy


Agenda
- THEORY
- SOFTWARE
- COMMUNITY

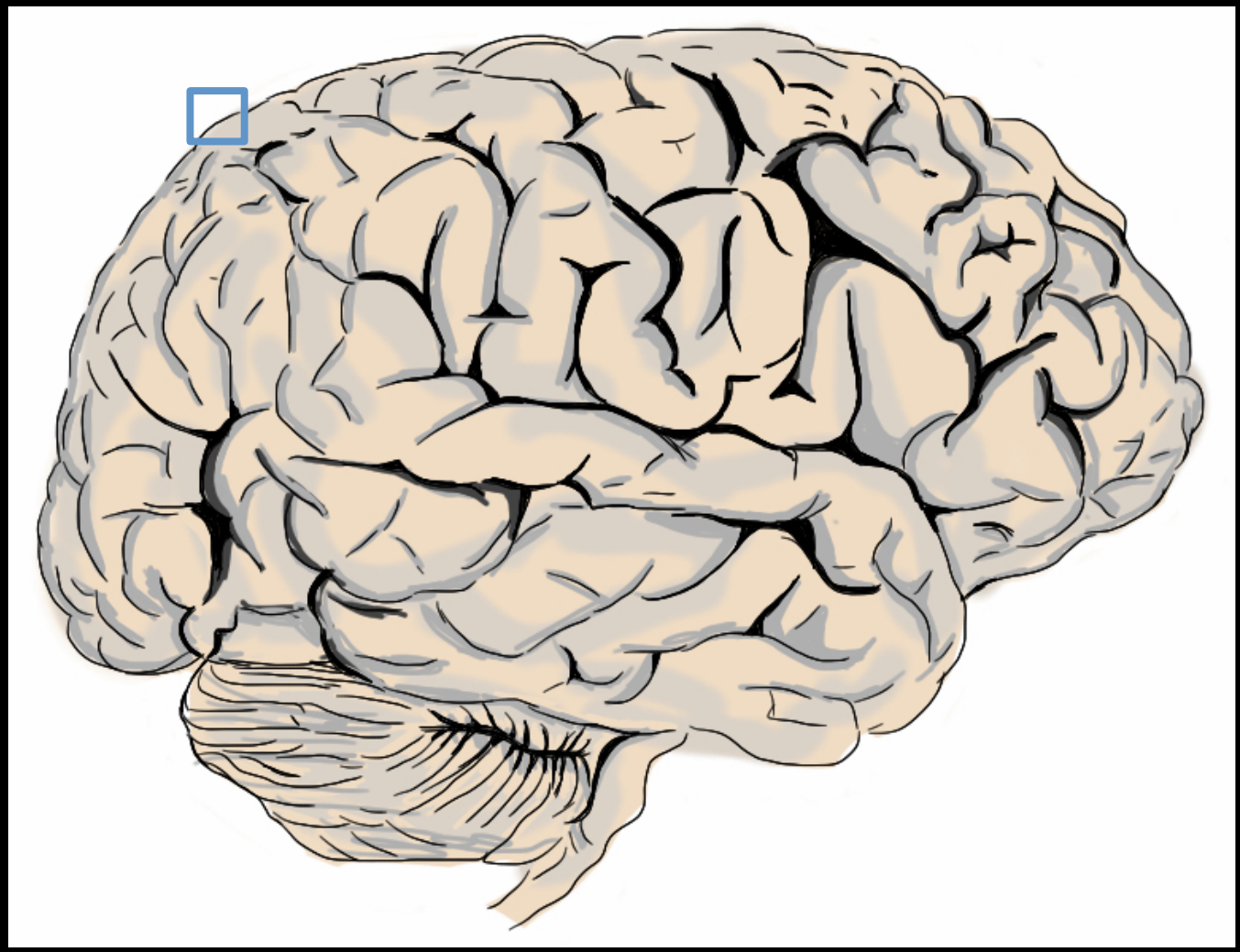
HOW A NEW UNDERSTANDING OF THE BRAIN WILL LEAD TO THE CREATION OF TRULY INTELLIGENT MACHINES

ON INTELLIGENCE

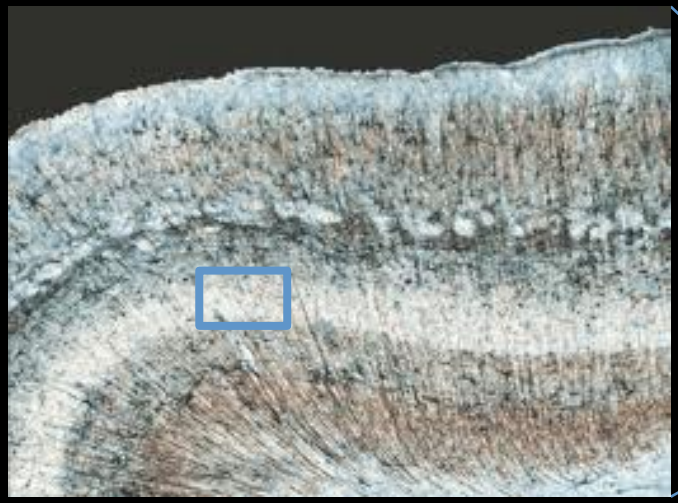
FROM THE INVENTOR OF THE ORIGINAL PALMPILOT

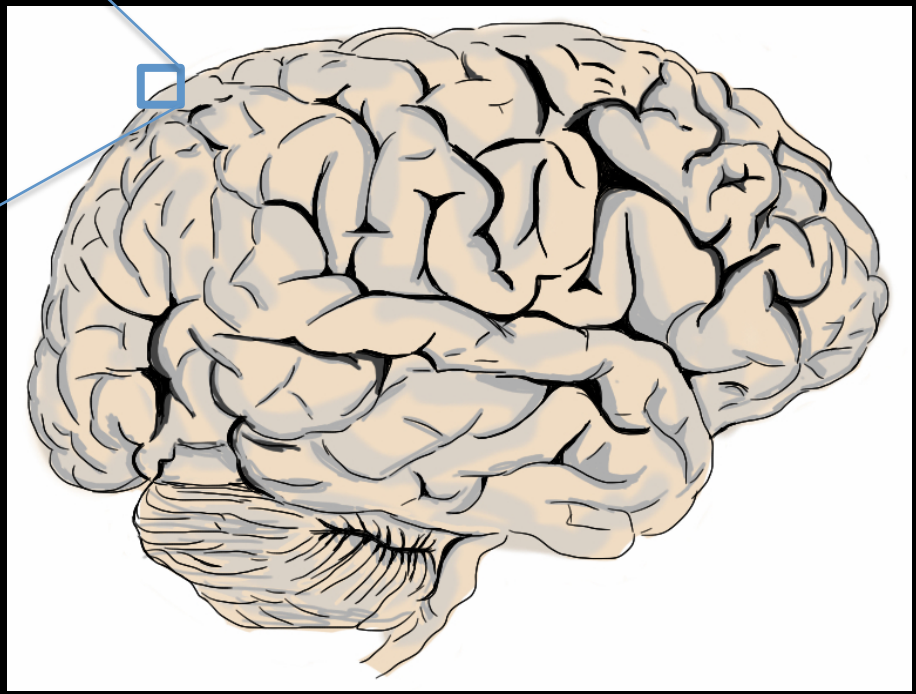
JEFF HAWKINS
with Sandra Blakeslee

Cortical Learning Algorithm

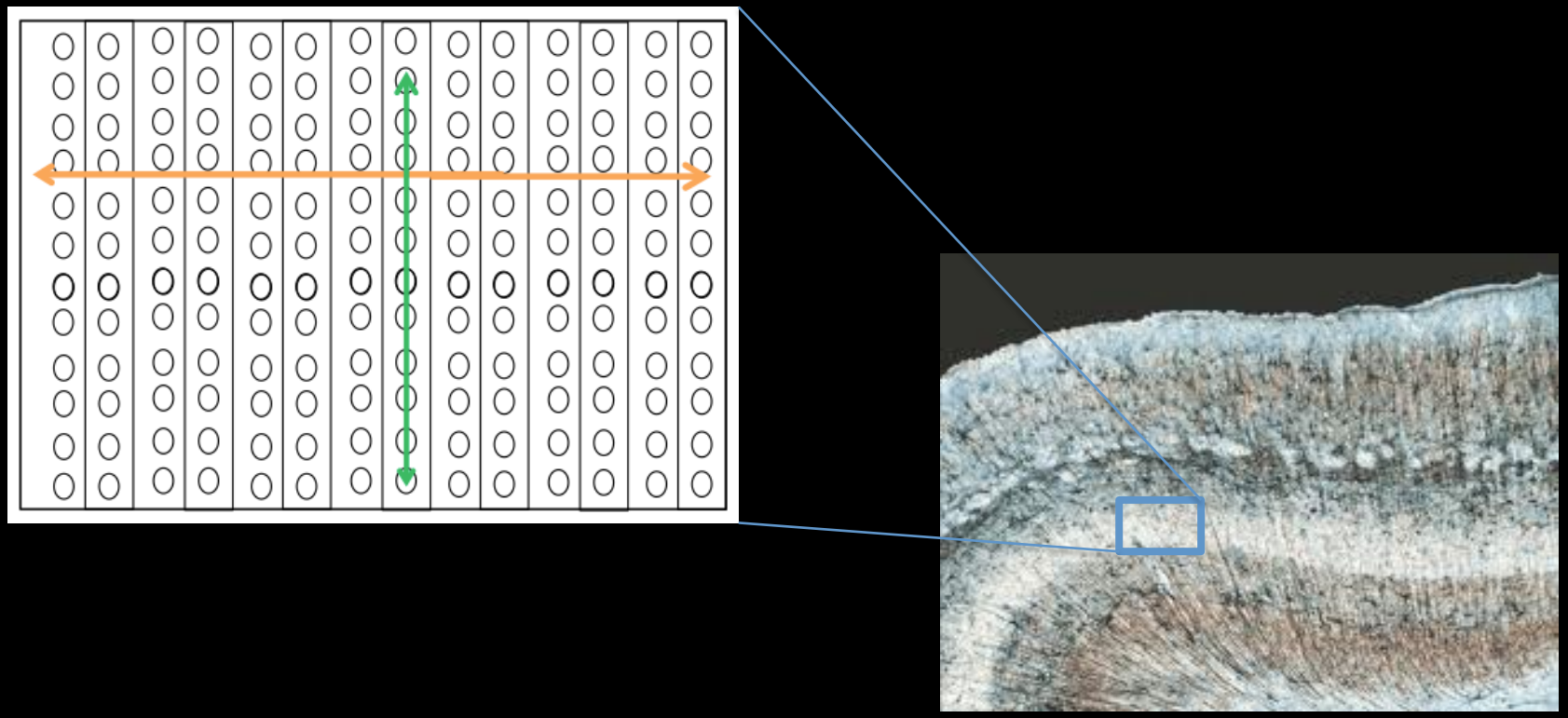


Cortical Learning Algorithm

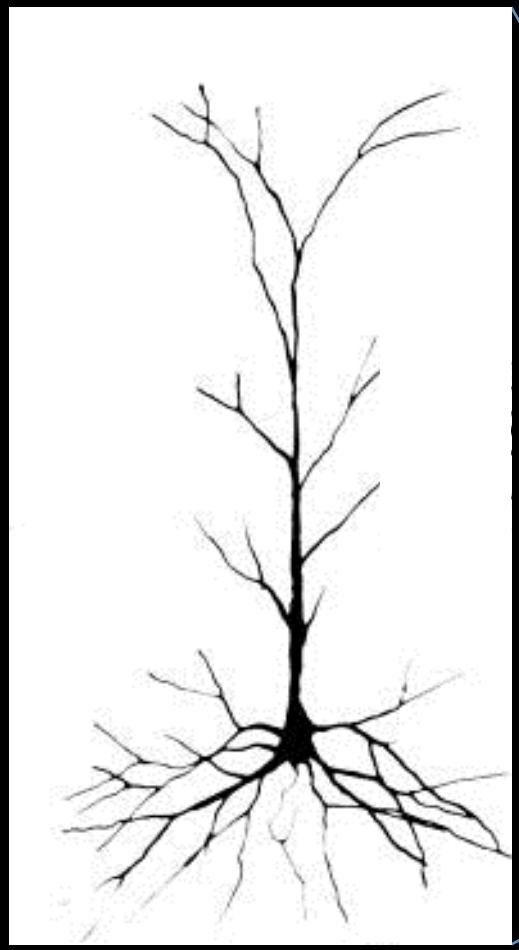


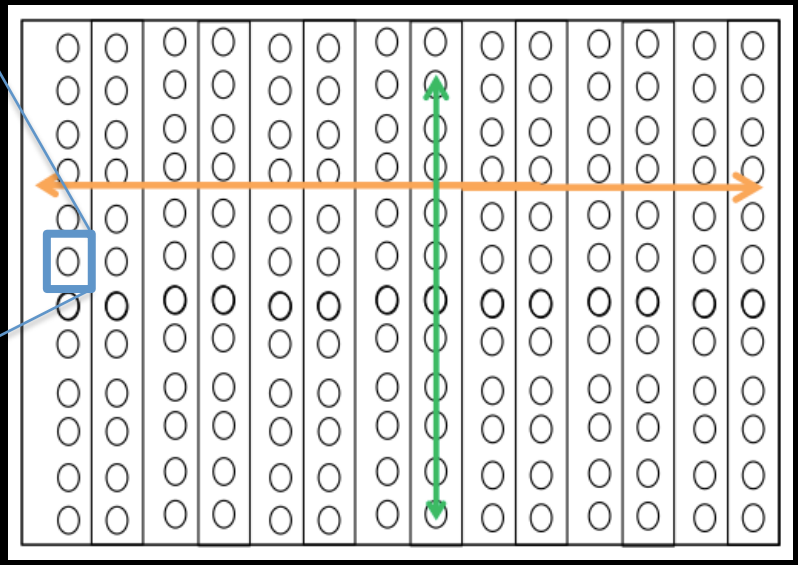


Cortical Learning Algorithm

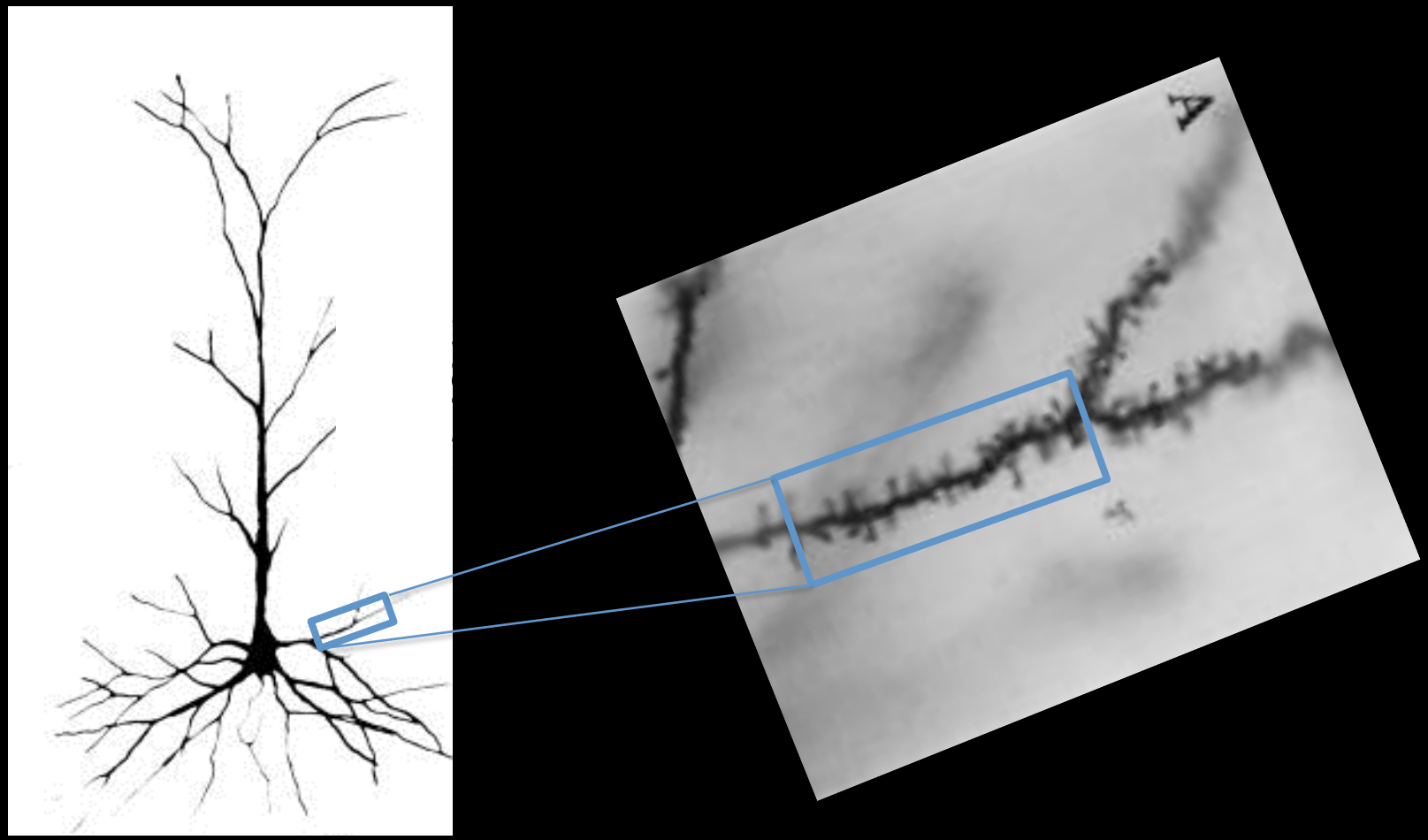


Cortical Learning Algorithm

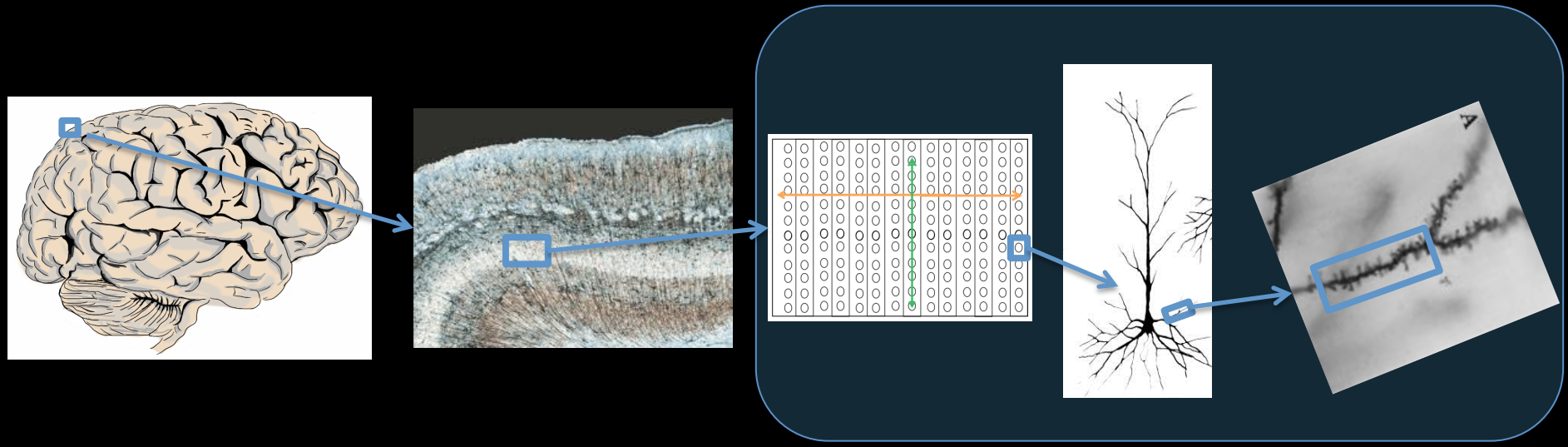




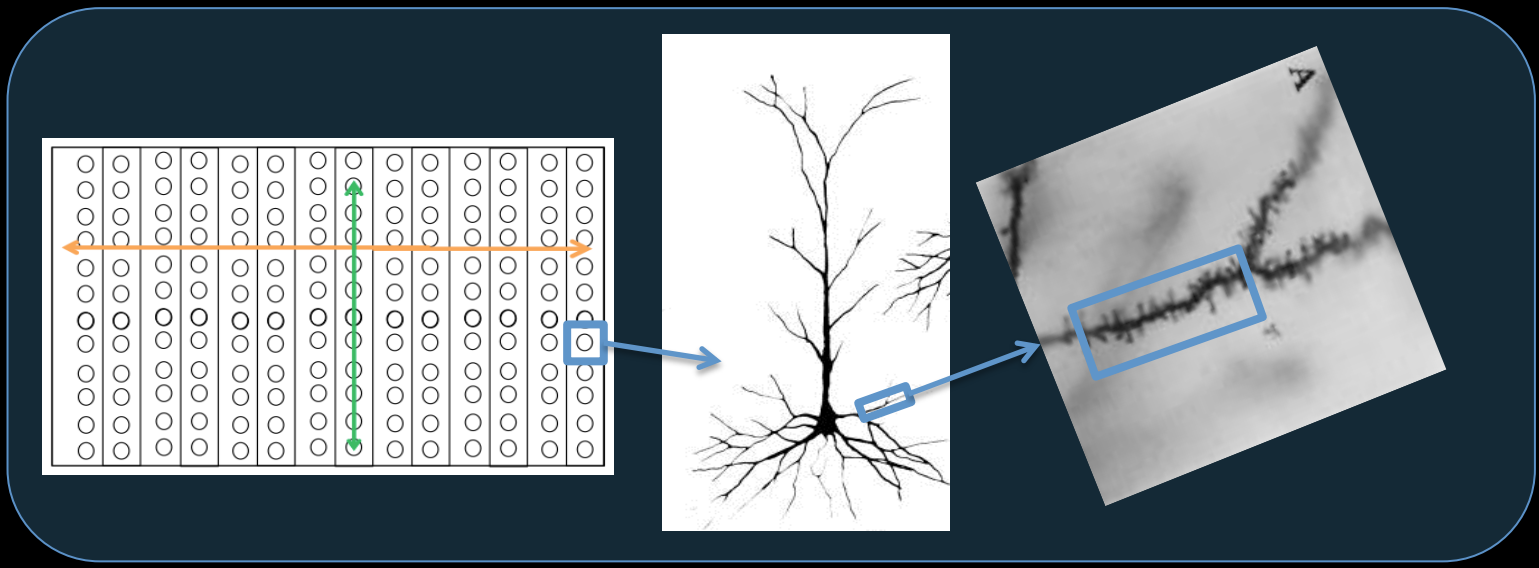
Cortical Learning Algorithm

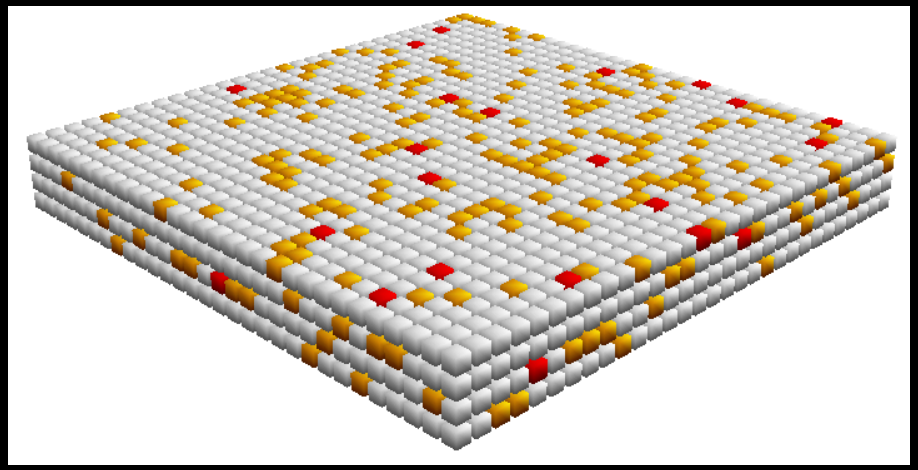


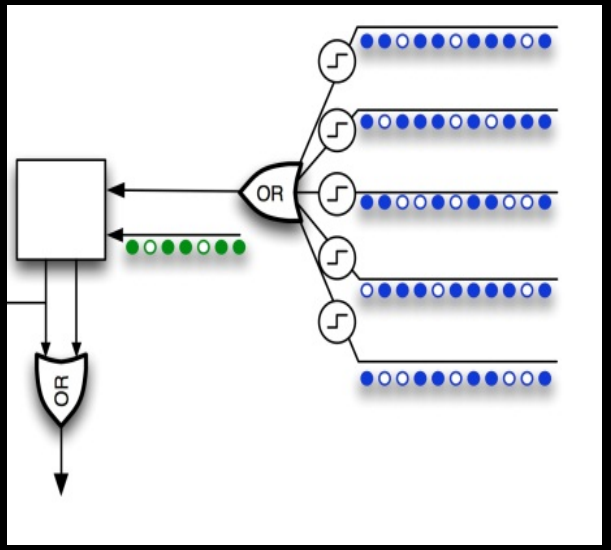
Cortical Learning Algorithm



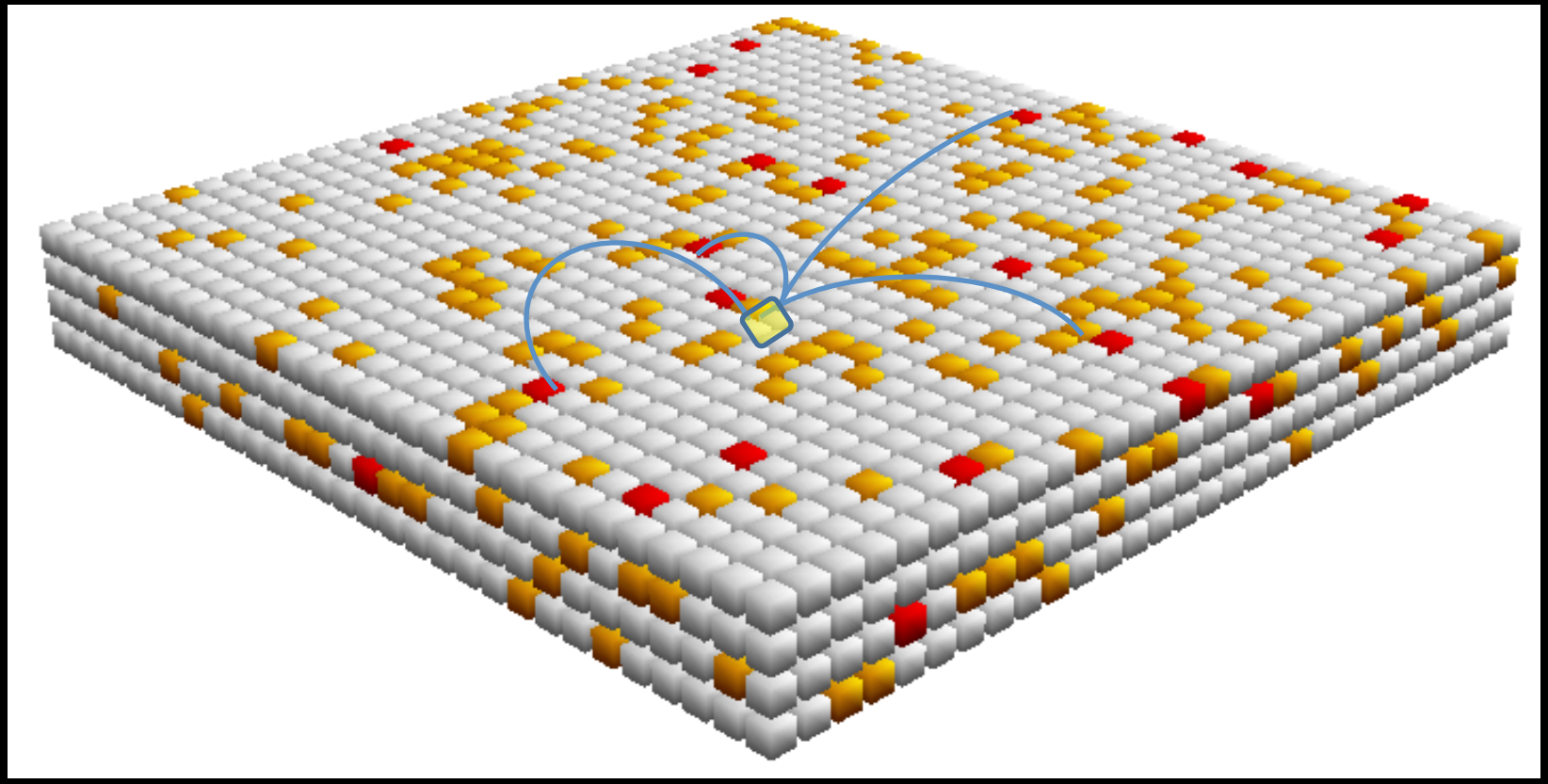
Cortical Learning Algorithm







Cortical Learning Algorithm



Variable order sequence memory
Multiple simultaneous predictions
High capacity

Online learning
Distributed, fault tolerant
Semantic generalization

What does it do?

- accepts streams of data
- learns spatial and temporal patterns
- creates a model of the data
- inference (pattern recognition)
  - predictions
  - anomaly detection

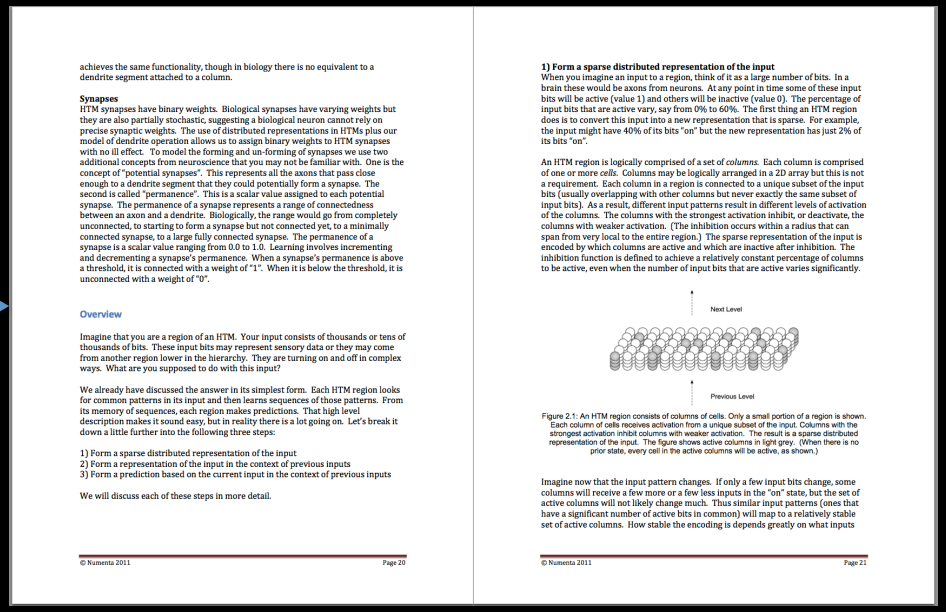
A taste of the details

- Sparse Distributed Representations

  `00000100010000000000000010000`

- very realistic neuron modeling
- layers and connections

Go to numenta.org
and download the
CLA White Paper



achieves the same functionality, though in biology there is no equivalent to a dendrite segment attached to a column.

Synapses
HTM synapses have binary weights. Biological synapses have varying weights but they are also partially stochastic, suggesting a biological neuron cannot rely on precise synaptic weights. The use of distributed representations in HTMs plus our model of dendrite operation allows us to assign binary weights to HTM synapses with no ill effect. To model the forming and un-forming of synapses we use two additional concepts from neuroscience that you may not be familiar with. One is the concept of "potential synapses". This represents all the axons that pass close enough to a dendrite segment that they could potentially form a synapse. The second is called "permanence". This is a scalar value assigned to each potential synapse. The permanence of a synapse represents a range of connectedness between an axon and a dendrite. Biologically, the range would go from completely unconnected, to starting to form a synapse but not connected yet, to a minimally connected synapse, to a large fully connected synapse. The permanence of a synapse is a scalar value ranging from 0.0 to 1.0. Learning involves incrementing and decrementing a synapse's permanence. When a synapse's permanence is above a threshold, it is connected with a weight of "1". When it is below the threshold, it is unconnected with a weight of "0".

Overview

Imagine that you are a region of an HTM. Your input consists of thousands or tens of thousands of bits. These input bits may represent sensory data or they may come from another region lower in the hierarchy. They are turning on and off in complex ways. What are you supposed to do with this input?

We already have discussed the answer in its simplest form. Each HTM region looks for common patterns in its input and then learns sequences of those patterns. From its memory of sequences, each region makes predictions. That high level description makes it sound easy, but in reality there is a lot going on. Let's break it down a little further into the following three steps:

1) Form a sparse distributed representation of the input
2) Form a representation of the input in the context of previous inputs
3) Form a prediction based on the current input in the context of previous inputs

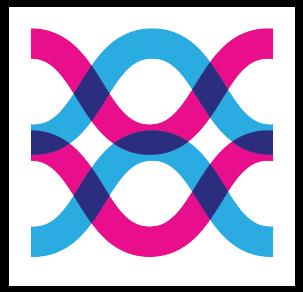
We will discuss each of these steps in more detail.

1) Form a sparse distributed representation of the input
When you imagine an input to a region, think of it as a large number of bits. In a brain these would be axons from neurons. At any point in time some of these input bits will be active (value 1) and others will be inactive (value 0). The percentage of input bits that are active vary, say from 0% to 60%. The first thing an HTM region does is to convert this input into a new representation that is sparse. For example, the input might have 40% of its bits "on" but the new representation has just 2% of its bits "on".

An HTM region is logically comprised of a set of *columns*. Each column is comprised of one or more *cells*. Columns may be logically arranged in a 2D array but this is not a requirement. Each column in a region is connected to a unique subset of the input bits (usually overlapping with other columns but never exactly the same subset of input bits). As a result, different input patterns result in different levels of activation of the columns. The columns with the strongest activation inhibit, or deactivate, the columns with weaker activation. (The inhibition occurs within a radius that can span from very local to the entire region.) The sparse representation of the input is encoded by which columns are active and which are inactive after inhibition. The inhibition function is defined to achieve a relatively constant percentage of columns to be active, even when the number of input bits that are active varies significantly.

Figure 2.1: An HTM region consists of columns of cells. Only a small portion of a region is shown. Each column of cells receives activation from a unique subset of the input. Columns with the strongest activation inhibit columns with weaker activation. The result is a sparse distributed representation of the input. The figure shows active columns in light grey. (When there is no prior state, every cell in the active columns will be active, as shown.)

Imagine now that the input pattern changes. If only a few input bits change, some columns will receive a few more or a few less inputs in the "on" state, but the set of active columns will not likely change much. Thus similar input patterns (ones that have a significant number of active bits in common) will map to a relatively stable set of active columns. How stable the encoding is depends greatly on what inputs

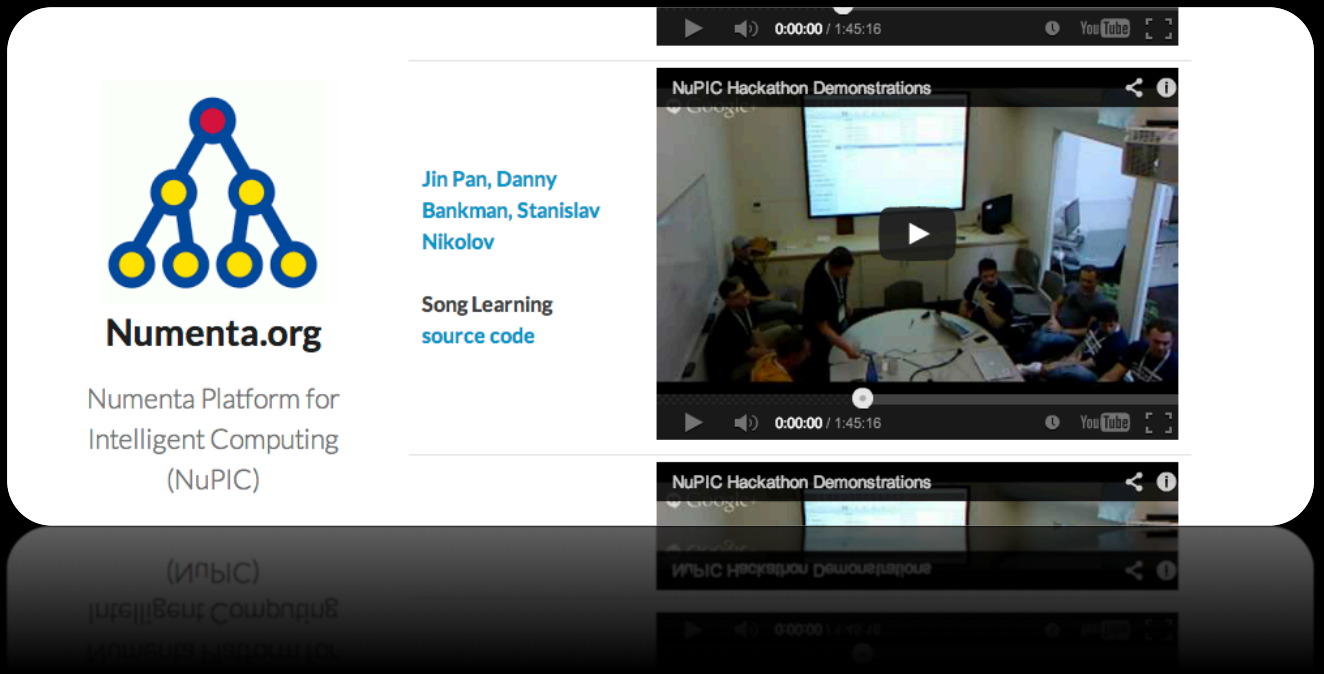


Current Application: GROK

- SAAS product
- High velocity, streaming input data
- NuPIC at the core
- Focus on
  - Industrial automation
  - IT applications

Possible Future Applications

- Vision
- Robotics
- Audio
- Natural language processing
- Fraud detection
- Supply / Demand
- IT
- Automated actions

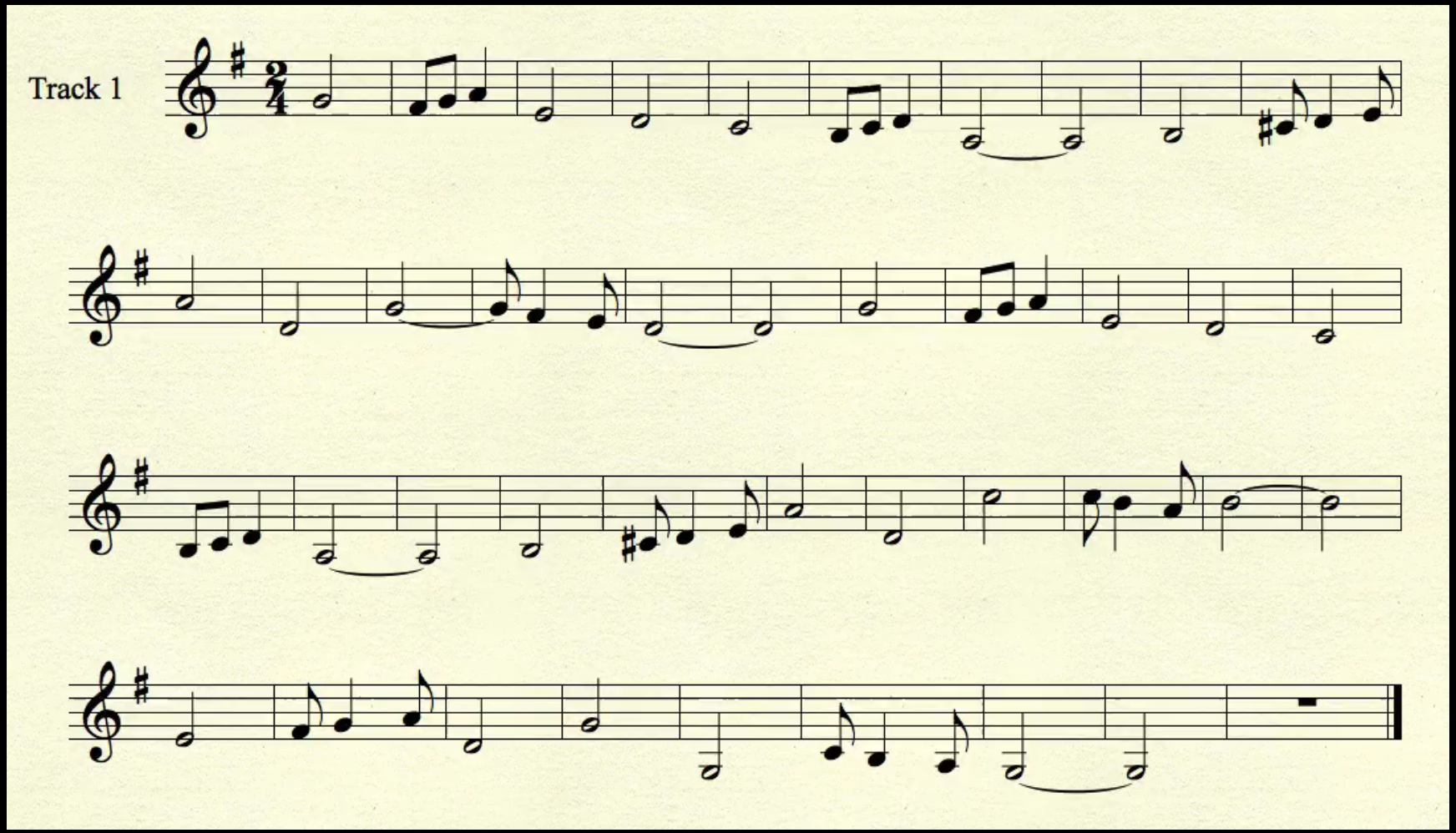


demonstration

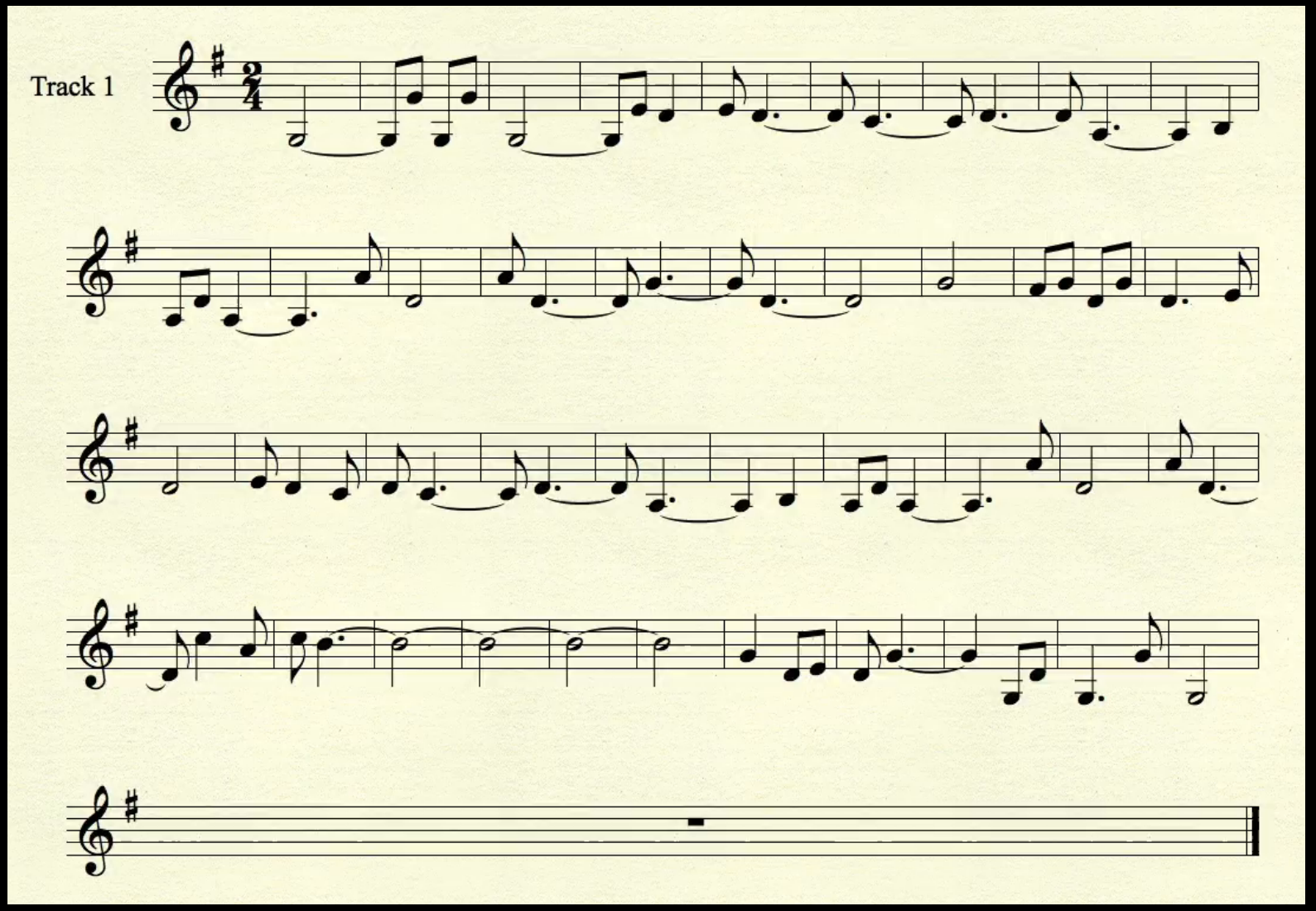
ONLINE LEARNING

Danny Bankman
Stanislav Nikolov
Jin Pan

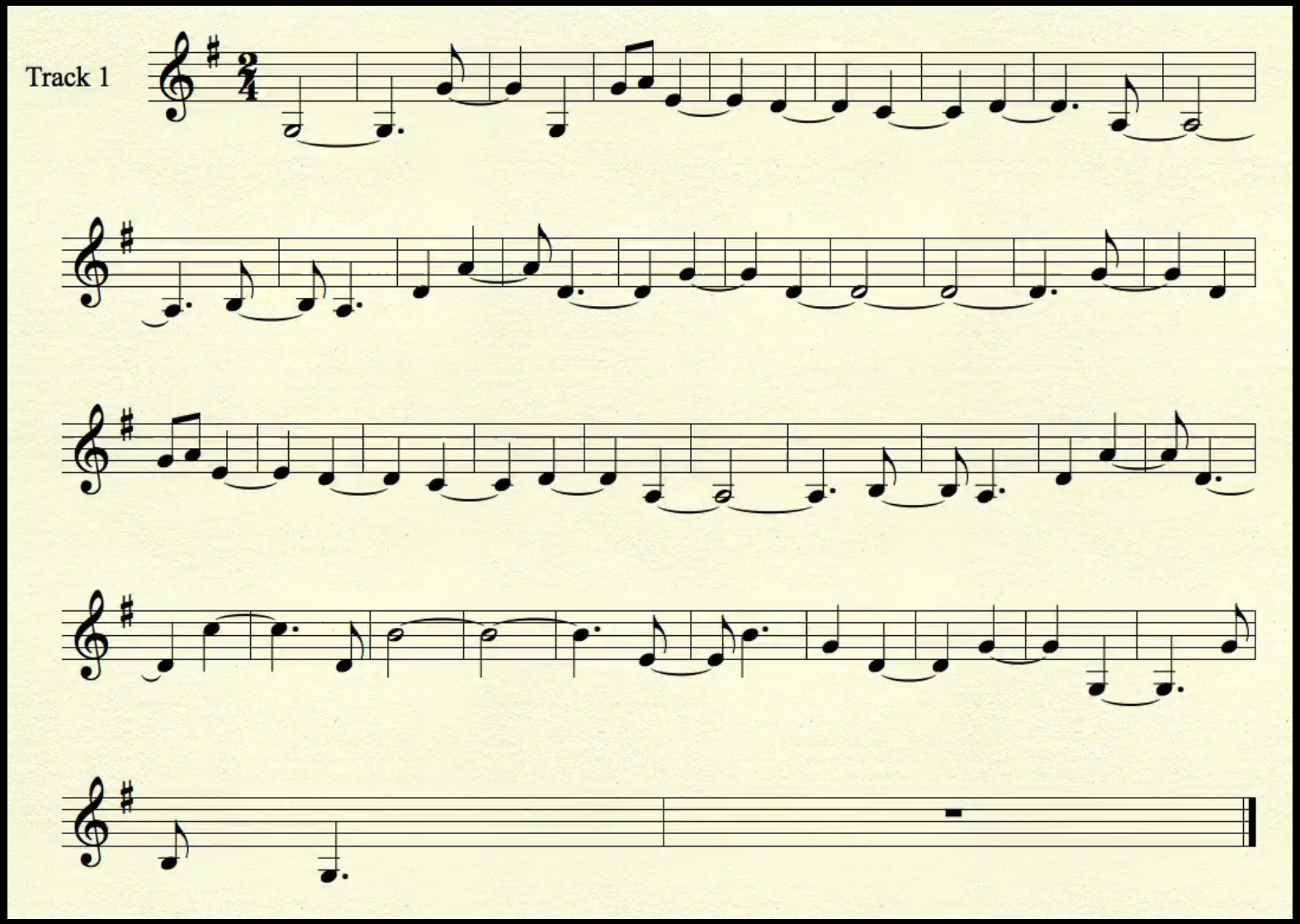
The Original Song



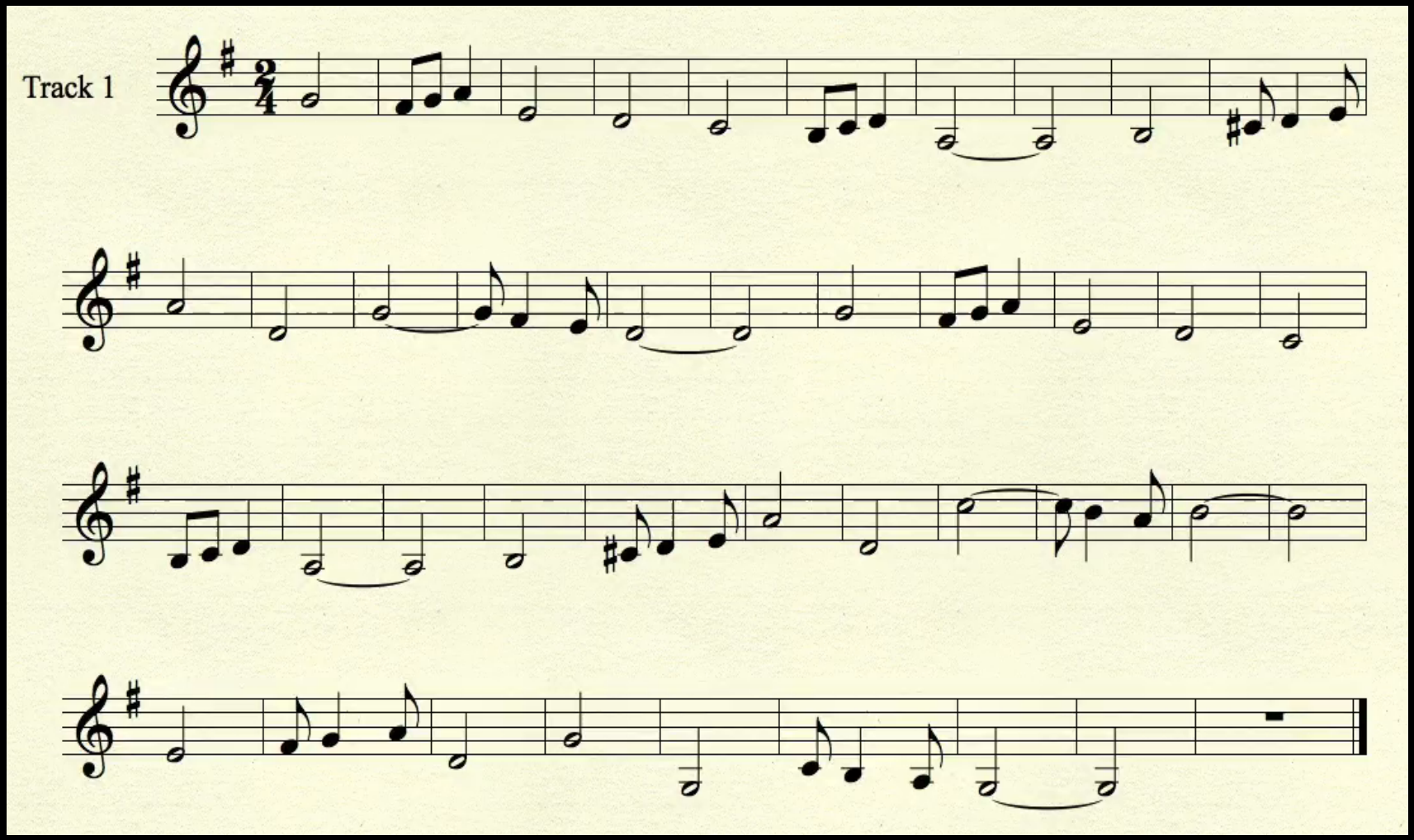
After 10 listens



After 20 listens



After 25 listens



Powerful Sequence Memory

- online learning
- temporal patterns
- common phrases
- just like your brain

Scott

NUTS AND BOLTS

NuPIC

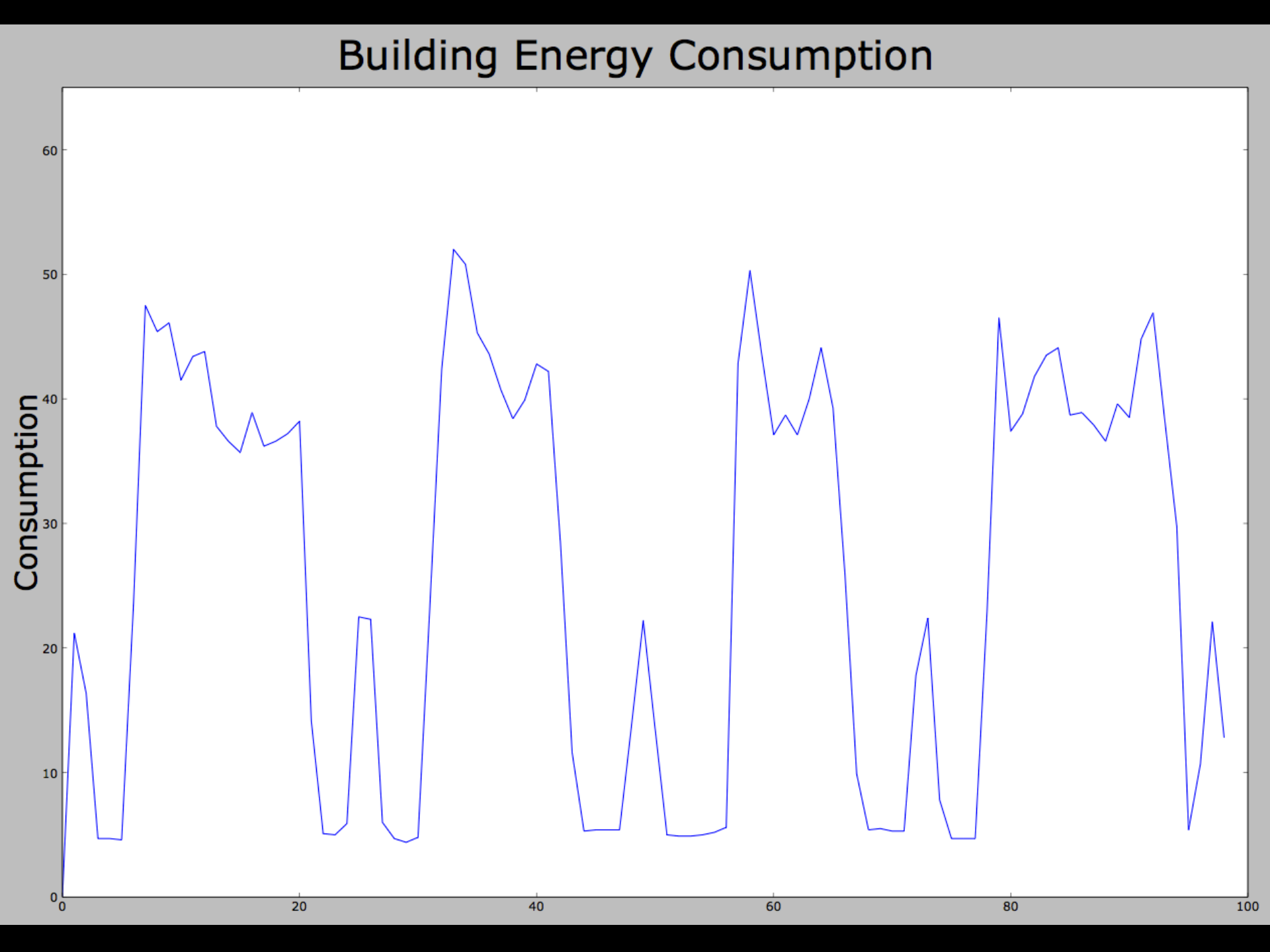
- Python API (some C++)

- Dual-license with GPL v3 option

- Code on Github

- Travis CI + Github = ❤

Getting Started Walkthrough


Building Energy Consumption
Consumption
0
10
20
30
40
50
60
0
20
40
60
80
100

Create the model

```
# Create a model
model = ModelFactory.create(model_params.MODEL_PARAMS)
model.enableInference({'predictedField': 'consumption'})
```

Model Parameters

```
'sensorParams': {
    'encoders': {
        'consumption': {
            'type': 'AdaptiveScalarEncoder',
            'n': 100,
            'w': 21
        },
    },
},
'tpParams': {
    'columnCount': 2048,
    'cellsPerColumn': 32,

    'permanenceInc': 0.1,
    'permanenceDec' : 0.1,
},
'clParams': {
    'steps': '1,5',
},
```

Iterate over the data

```
# Create a model
model = ModelFactory.create(model_params.MODEL_PARAMS)
model.enableInference({'predictedField': 'consumption'})

reader = csv.reader(open(_DATA_PATH))
headers = reader.next()

# Iterate over the data file
for i, record in enumerate(reader, start=1):
```

Pass the data to the model

```
# Create a model
model = ModelFactory.create(model_params.MODEL_PARAMS)
model.enableInference({'predictedField': 'consumption'})

reader = csv.reader(open(_DATA_PATH))
headers = reader.next()

# Iterate over the data file
for i, record in enumerate(reader, start=1):
  # Create a dictionary to pass in to the model
  modelInput = dict(zip(headers, record))
  modelInput["consumption"] = float(modelInput["consumption"])
  modelInput["timestamp"] = datetime.datetime.strptime(
      modelInput["timestamp"], "%m/%d/%y %H:%M")

  # Run the model on a single record and retrieve the result
  result = model.run(modelInput)
```

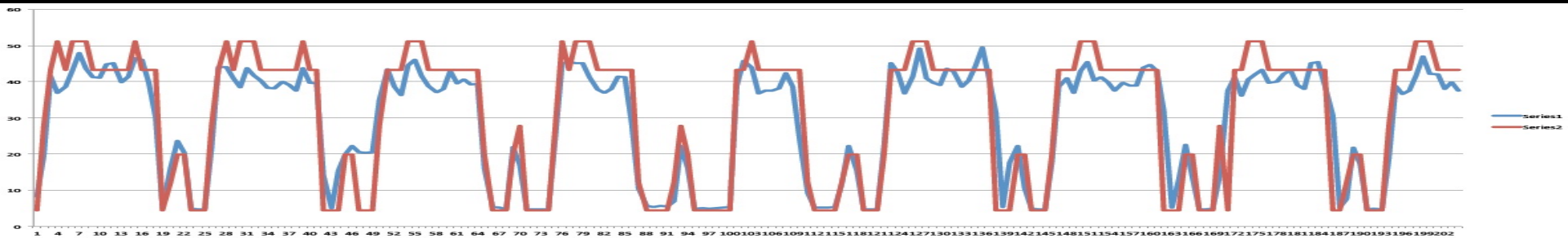
ModelResult

```
ModelResult(
    inferences={
        'multiStepPredictions': {
            1: {
                5.2825868514199987: 0.69999516634971859,
                10.699999999999999: 0.076012570549651950,
                22.100000000000001: 0.055294648127235196,
                22.899999999999999: 0.052690624183750749,
            },
            5: {
                38.188079999999999: 0.22754381767774520,
                47.359999999999992: 0.19538808382423584,
                37.399999999999999: 0.12597931862094047,
                45.399999999999999: 0.09912326127203159 6,
                37.089999999999996: 0.08291321593693275 2,
                39.280000000000001: 0.07793578193551516 1,
                43.629999999999995: 0.07640528916418928 8
            }
        },
        'multiStepBestPredictions': {1: 5.2825868514199987, 5: 38.188079999999999},
    }
    metrics={...}
    predictedFieldIdx=0
    predictedFieldName=consumption
)
```

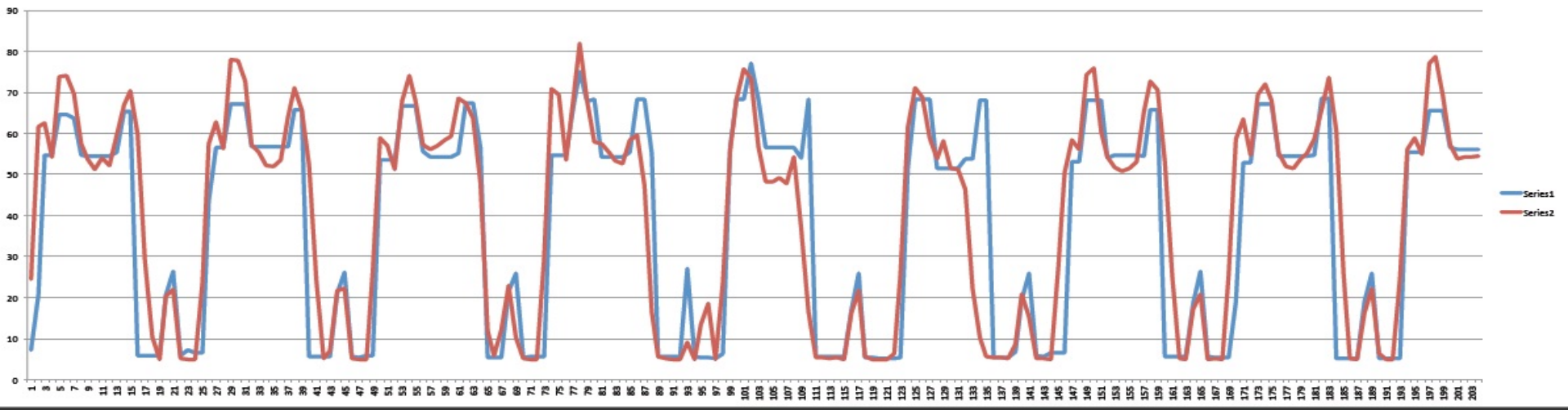
Live Demo

Changing patterns

Time epoch 1



Time epoch 2



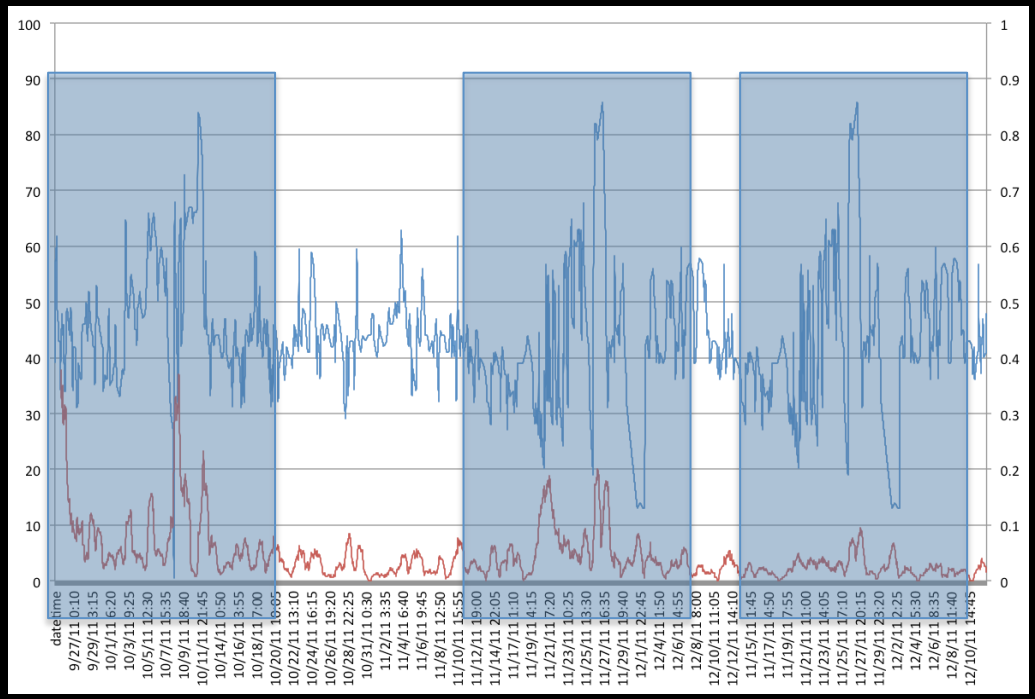
Anomaly Detection

CLA used to detect anomalies in gear bearing temperature

Detects anomalies based on temporal characteristics

Thresholds are not sufficient

Reduces downtime and maintenance costs



Gear bearing temperature & anomaly score

Quick Review

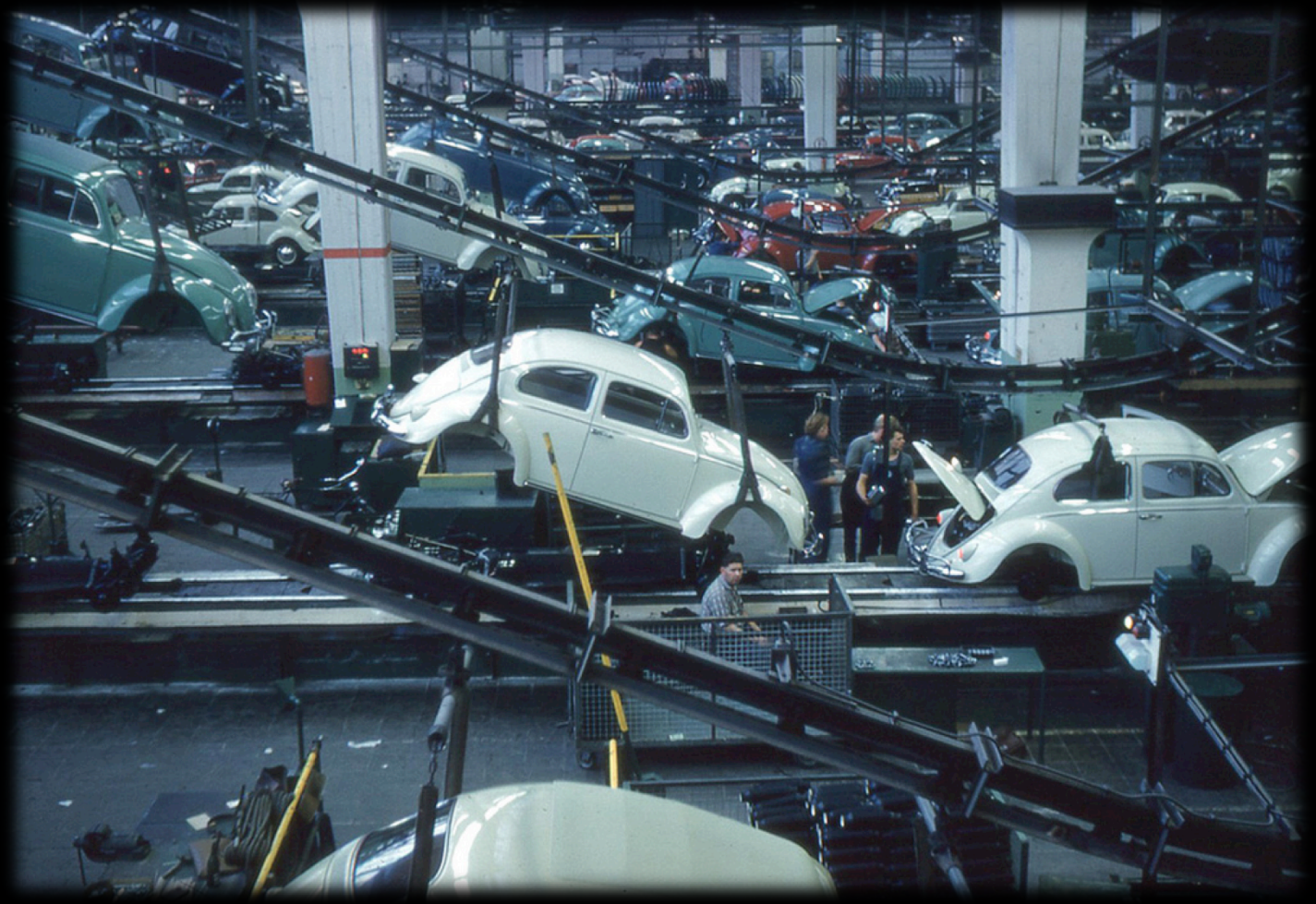
- streamed energy consumption data into CLA
- it learned spatial and temporal patterns in data
  - created a model of the data
- inference (pattern recognition)
  - predictions (1 hour, 5 hours)
  - anomalies

Ideas to inspire

Matt

OPEN SOURCE COMMUNITY

Why this OS project is different

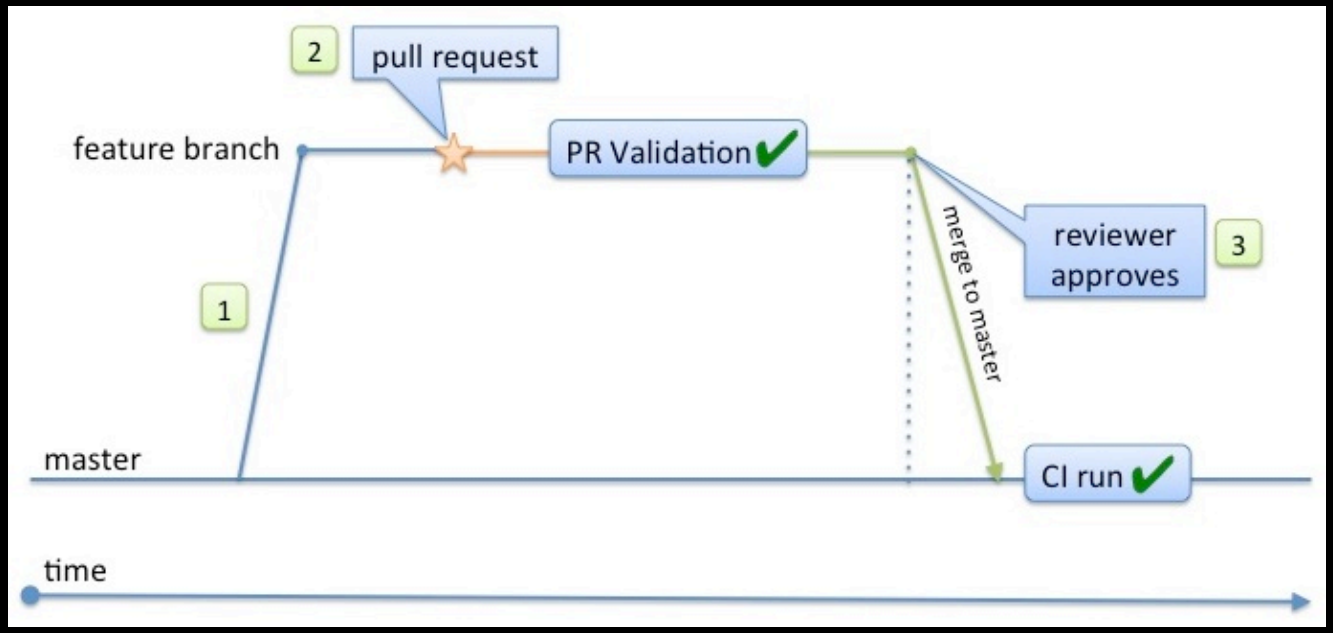


Why open source the tool...

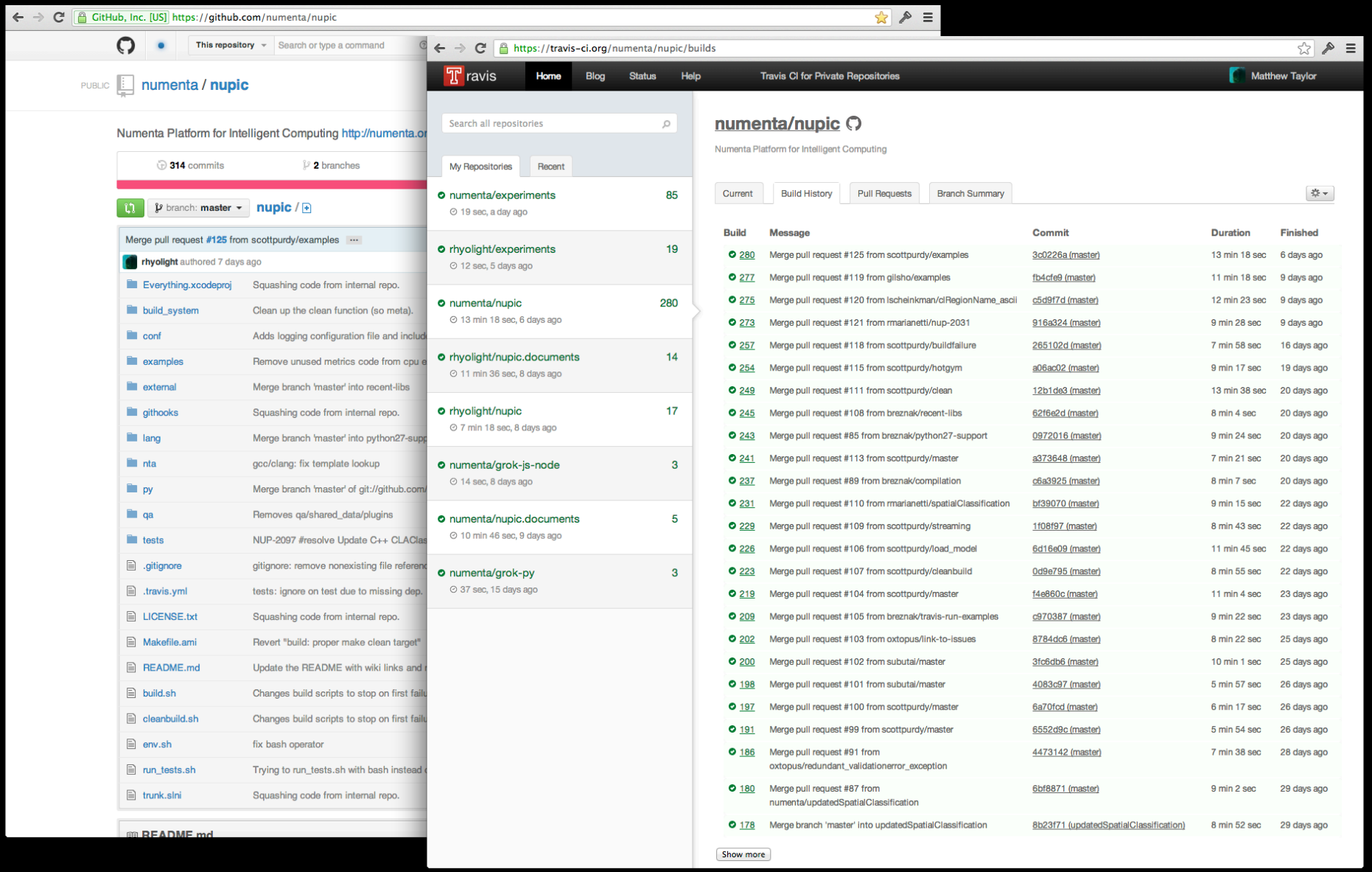
When you can open source the engine?

Single Source Tree

- Grok builds directly from NuPIC
- master is always green
- testing occurs within Pull Requests



Github / Travis-CI



September 2013 Archives by thread

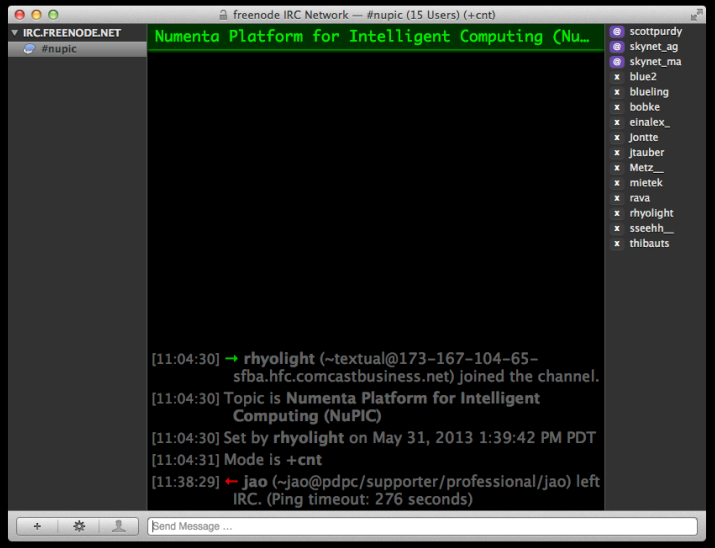
- **Messages sorted by:** [subject] [author] [date]
- More info on this list...

Starting: *Sun Sep 1 05:10:25 EDT 2013*
Ending: *Tue Sep 24 17:52:29 EDT 2013*
Messages: 321

- [nupic-dev] Some comments on *Tim McNamara*
  - [nupic-dev] Some comments on *Matthew Taylor*
    - [nupic-dev] Some comments on *Tim McNamara*
  - [nupic-dev] Some comments on *Austin Marshall*
    - [nupic-dev] Some comments on *Tim McNamara*
      - [nupic-dev] Some comments on *Matthew Taylor*
  - [nupic-dev] Some comments on *Erik Blas*
- [nupic-dev] Inter-layer plumbing *Patrick Higgins*
  - [nupic-dev] Inter-layer plumbing *Jeff Hawkins*
    - [nupic-dev] Inter-layer plumbing *Tim Boudreau*
  - [nupic-dev] Inter-layer plumbing *Tim Boudreau*
    - [nupic-dev] Inter-layer plumbing *Jeff Hawkins*
      - [nupic-dev] Inter-layer plumbing *Chetan Surpur*
      - [nupic-dev] Inter-layer plumbing *Fergal Byrne*
      - [nupic-dev] Inter-layer plumbing *Jeff Hawkins*
      - [nupic-dev] Inter-layer plumbing *Chetan Surpur*
      - [nupic-dev] Inter-layer plumbing *Jeff Hawkins*
      - [nupic-dev] Inter-layer plumbing *Chetan Surpur*
      - [nupic-dev] Inter-layer plumbing *Fergal Byrne*
  - [nupic-dev] Inter-layer plumbing *Ivan Sytenko*
- [nupic-dev] Brain Protein *may2003*
- [nupic-dev] NuPIC for Model Predictive Control *Ralph Dratman*
  - [nupic-dev] NuPIC for Model Predictive Control *Ian Danforth*
    - [nupic-dev] NuPIC for Model Predictive Control *Matthew Taylor*
      - [nupic-dev] NuPIC for Model Predictive Control *Pedro Tabacof*
      - [nupic-dev] NuPIC for Model Predictive Control *Chetan Surpur*
      - [nupic-dev] NuPIC for Model Predictive Control *Chetan Surpur*
      - [nupic-dev] NuPIC for Model Predictive Control *Chetan Surpur*
      - [nupic-dev] NuPIC for Model Predictive Control *Ian Danforth*
      - [nupic-dev] NuPIC for Model Predictive Control *Erik Blas*
- [nupic-dev] Documenting models *Tim McNamara*
  - [nupic-dev] Documenting models *Matthew Taylor*
- [nupic-dev] Python 2.7 build with 2.6 Error *Matthew Taylor*
  - [nupic-dev] Python 2.7 build with 2.6 Error *Marek Otahal*
    - [nupic-dev] Python 2.7 build with 2.6 Error *Matthew Taylor*
      - [nupic-dev] Python 2.7 build with 2.6 Error *Marek Otahal*
    - [nupic-dev] Python 2.7 build with 2.6 Error *Sinon Galvin*
  - [nupic-dev] Python 2.7 build with 2.6 Error *Joshua Auerbach*
    - [nupic-dev] Python 2.7 build with 2.6 Error *Erik Blas*
    - [nupic-dev] Python 2.7 build with 2.6 Error *Sinon Galvin*

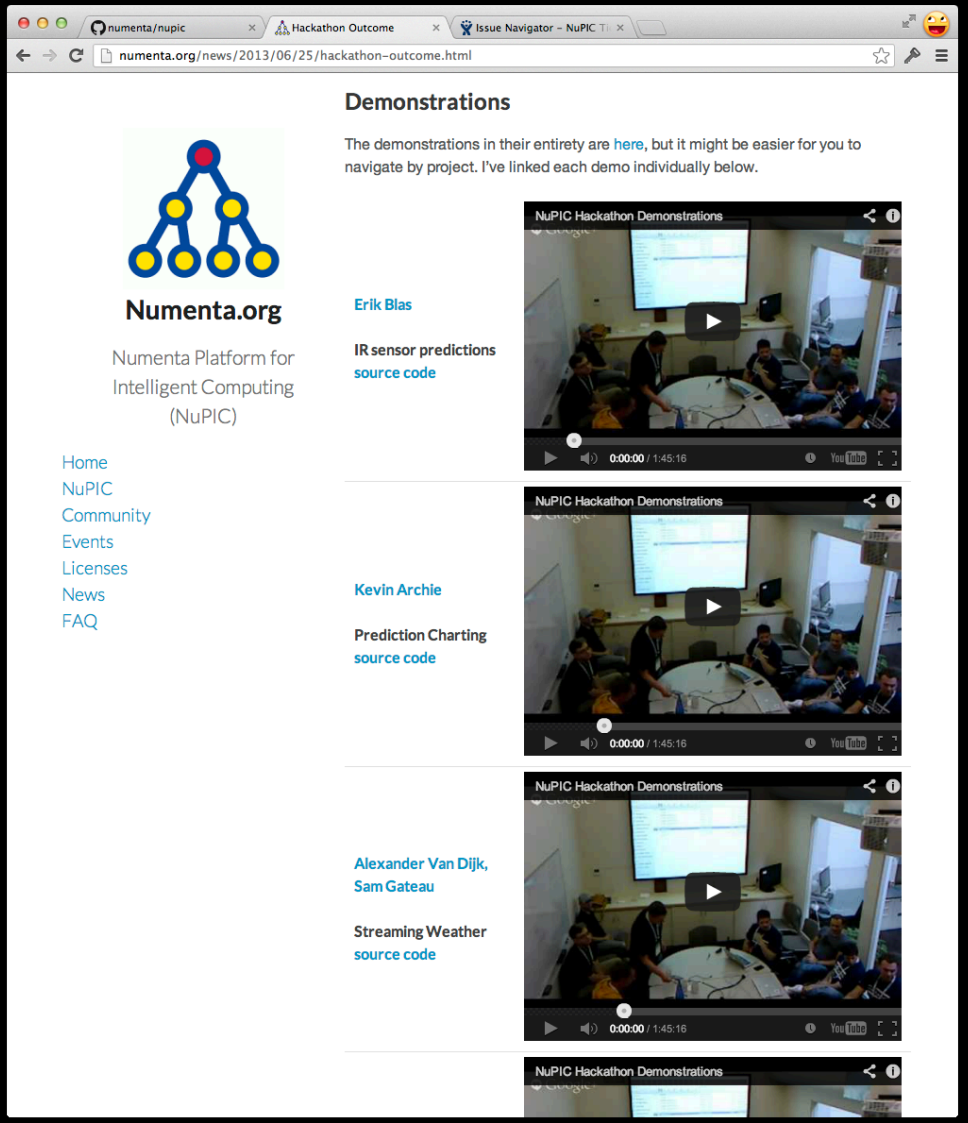
Community

- 63 contributors
- 227 list members
- 10 daily messages
- public issue tracker
- Bi-weekly sprints
- IRC #nupic



June 2013 Hackathon

- Robot movement
- IR Sensors
- Weather
- Human movement
- Tic Tac Toe
- CPU usage



Getting Involved

- Community Page on numenta.org

- Public JIRA for ticket tracking
- You can always tell what we're doing
- Newbie tasks

- Hackathon in SF this fall
  - Nov 2 - 3
  - numenta.org/events.html

Next Steps for you

- Read the CLA White Paper
- Read *On Intelligence*
- Join our mailing list
  - nupic@lists.numenta.org
- Videos: numenta.org/media.html

Contacts

matt@numenta.org
scott@numenta.org
jhawkins@numenta.org

http://numenta.org

We'll be outside after

numenta