Unlocking Biologically-Inspired Computer Vision: a High-Throughput Approach

Nicolas Pinto, David Cox and James DiCarlo

NVIDIA GTC | October, 2009

The Rowland Institute at Harvard
HARVARD UNIVERSITY

Unlockin… …ed
Comput…
a High-T… …n

…tute at Harvard
…SITY

Unlocking Biologically-Inspired Computer Vision: a High-Throughput Approach

=

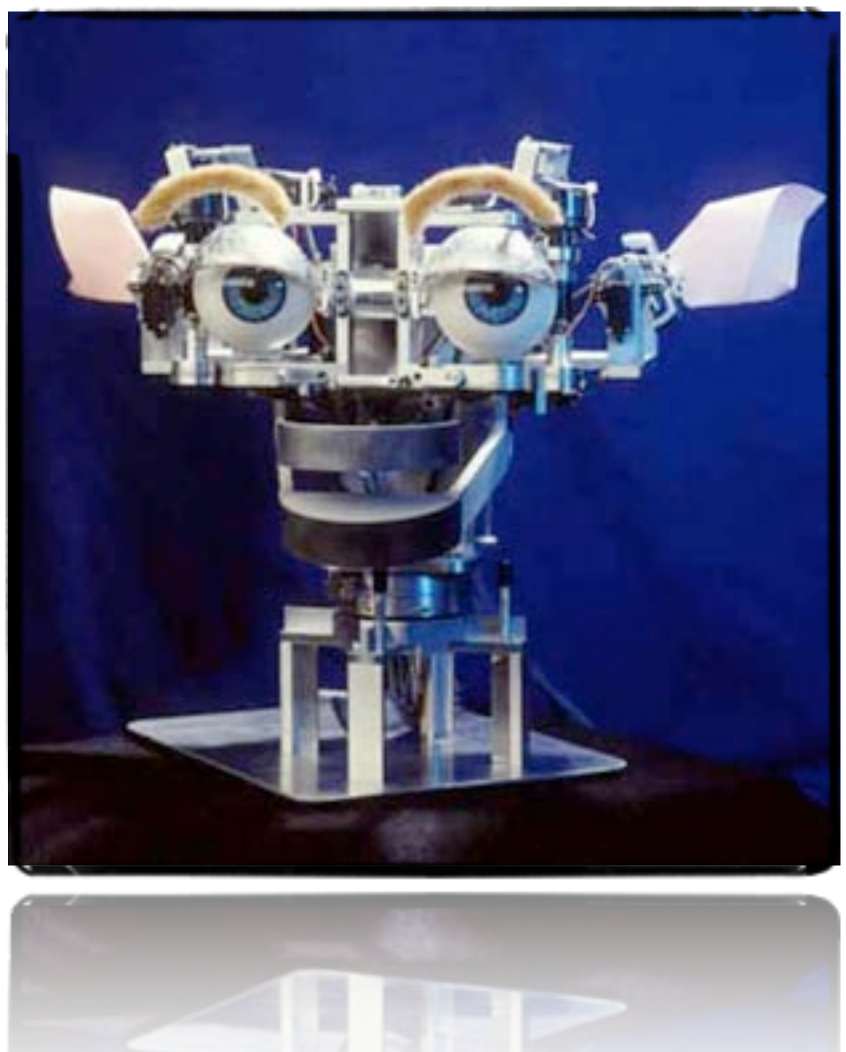
BRAIN

(NEUROSCIENCES)

Unlocking Biologically-Inspired

Computer Vision:

a High-Throughput Approach

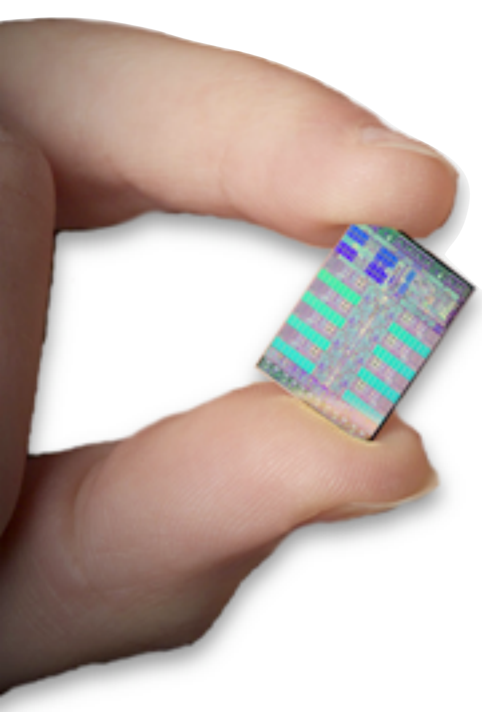


= A.I.

Unlocking Biologically-Inspired Computer Vision: a High-Throughput Approach

=

GPU



Quote to remember...

Friend: **So, what are you studying for your PhD?**

Me: **I study biological and artificial vision.**

Friend: **What?!? But vision is super easy!**

The Problem:
Visual Object Recognition

The Problem:
Visual Object Recognition

The Problem:
Visual Object Recognition

- *Fast*

The Problem:
Visual Object Recognition

- *Fast*
- *Accurate*

The Problem:
Visual Object Recognition

- *Fast*
- *Accurate*
- *Tolerant to variation*

The Problem:
Visual Object Recognition

- *Fast*
- *Accurate*
- *Tolerant to variation*
- *Effortless*

The Problem:
Visual Object Recognition

- *Fast*
- *Accurate*
- *Tolerant to variation*
- *Effortless*
- *Critical to survival*

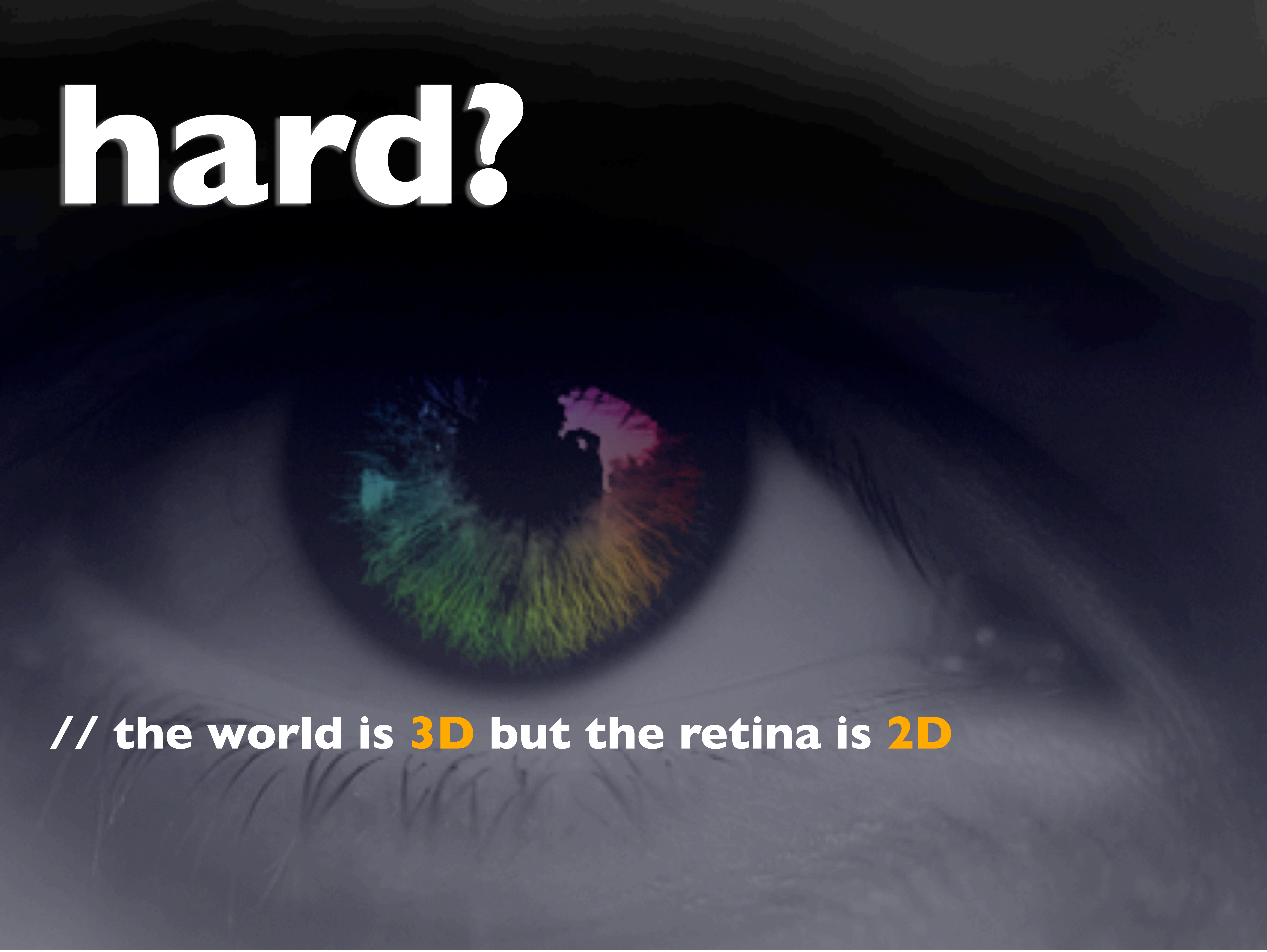
The Problem:
Visual Object Recognition

- *Fast*
- *Accurate*
- *Tolerant to variation*
- *Effortless*
- *Critical to survival*

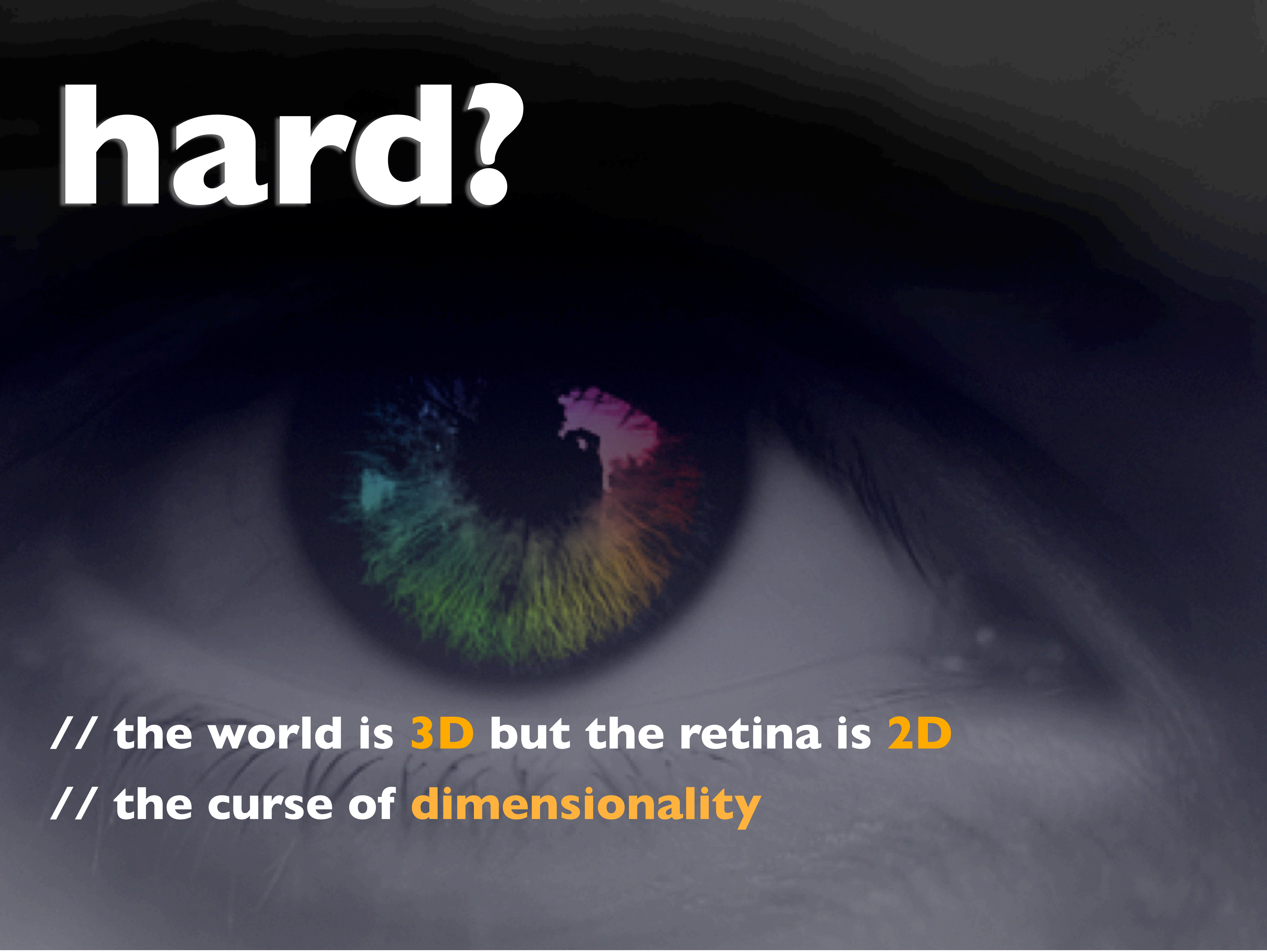
(for primates)


hard?

hard?



// the world is **3D** but the retina is **2D**


hard?
// the world is 3D but the retina is 2D
// the curse of dimensionality


hard?
// the world is 3D but the retina is 2D
// the curse of dimensionality
// considerable image variation


image variation!
do you recognize me ?
VOX

image variation!

do you recognize me ?

image variation!

do you recognize me ?

image variation!

do you recognize me ?

image variation!

do you recognize me ?

the brain!

~50% of that is for vision!

you learned it...

Need for speed

Hardware

Software

Science

The Approach:
Reverse Engineering the Brain

REVERSE

Study
Natural System

The Approach:
Reverse Engineering the Brain

REVERSE

Study
Natural System

FORWARD

Build
Artificial System

The Approach:
Reverse Engineering the Brain

REVERSE

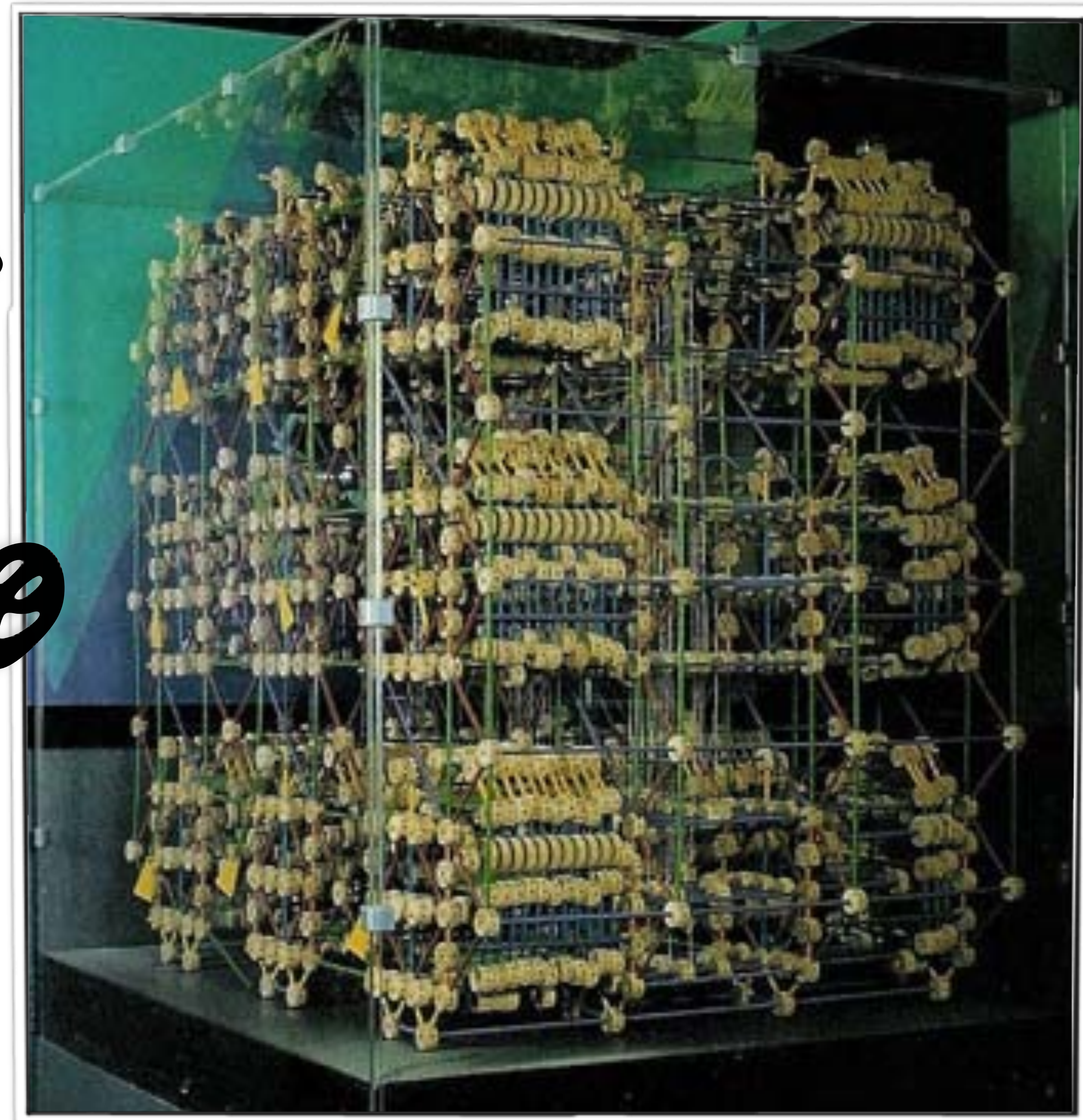
FORWARD

Study
Natural System

Build
Artificial System

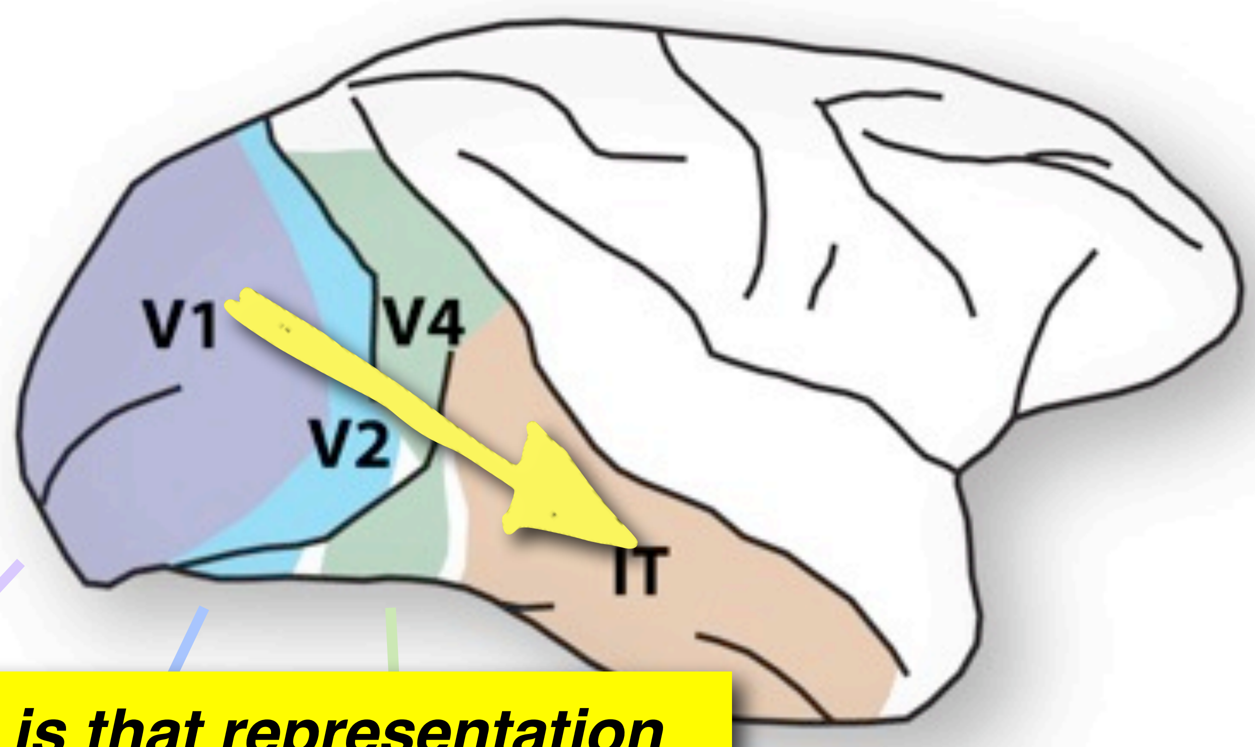
Reverse Engineering ...

What is this doing?



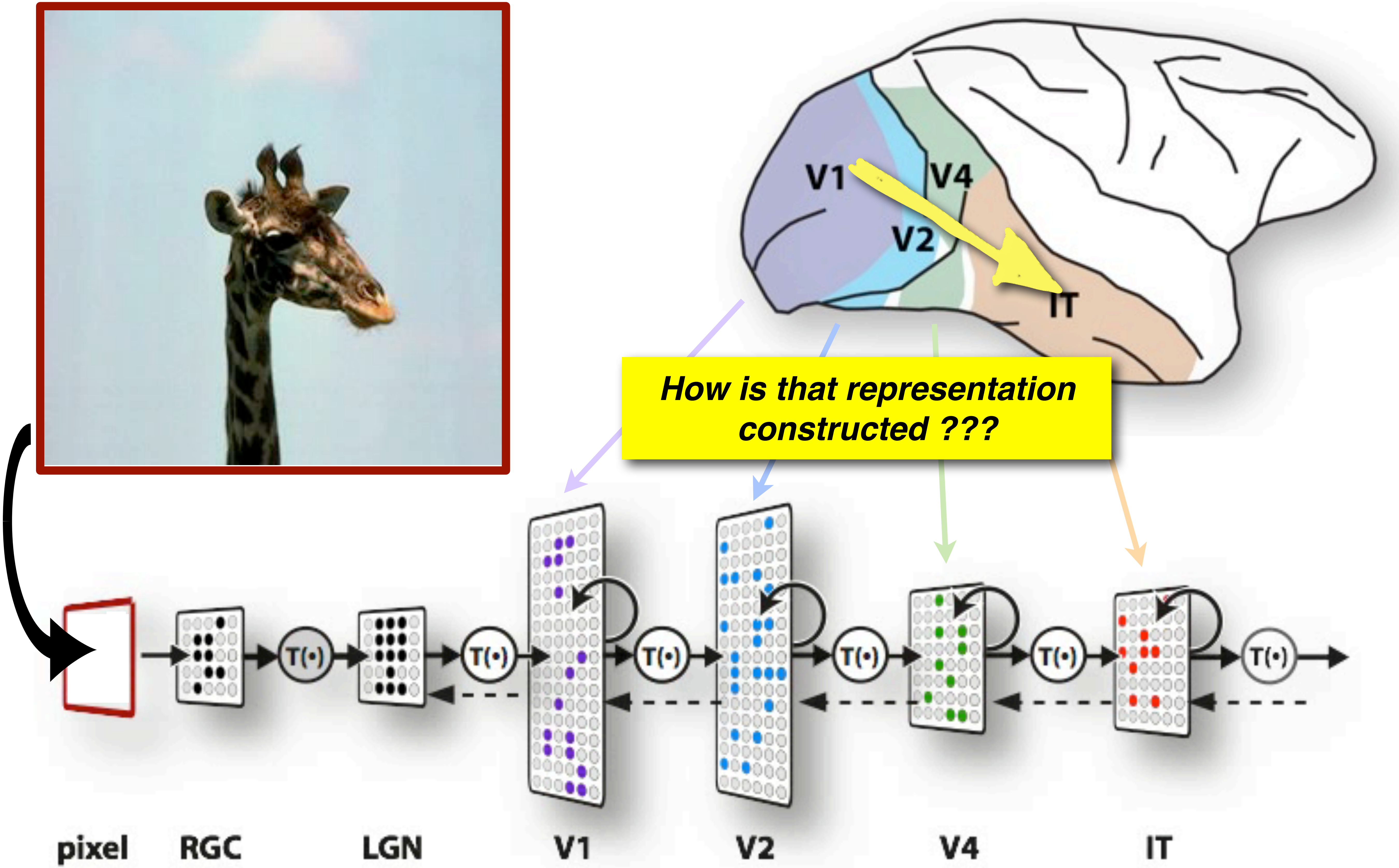
Reverse Engineering **the Brain!**

The Ventral Visual Stream

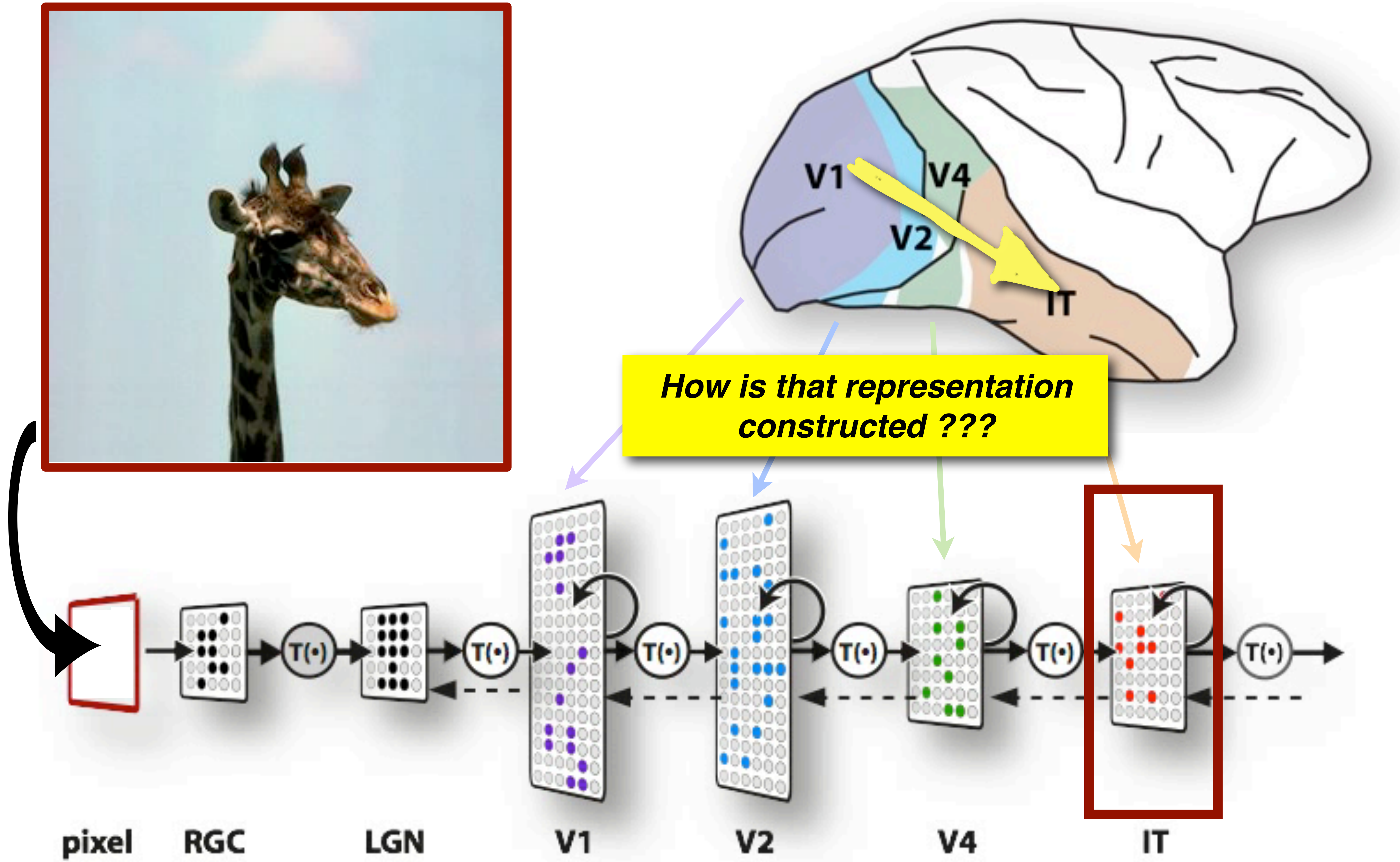


How is that representation constructed ???

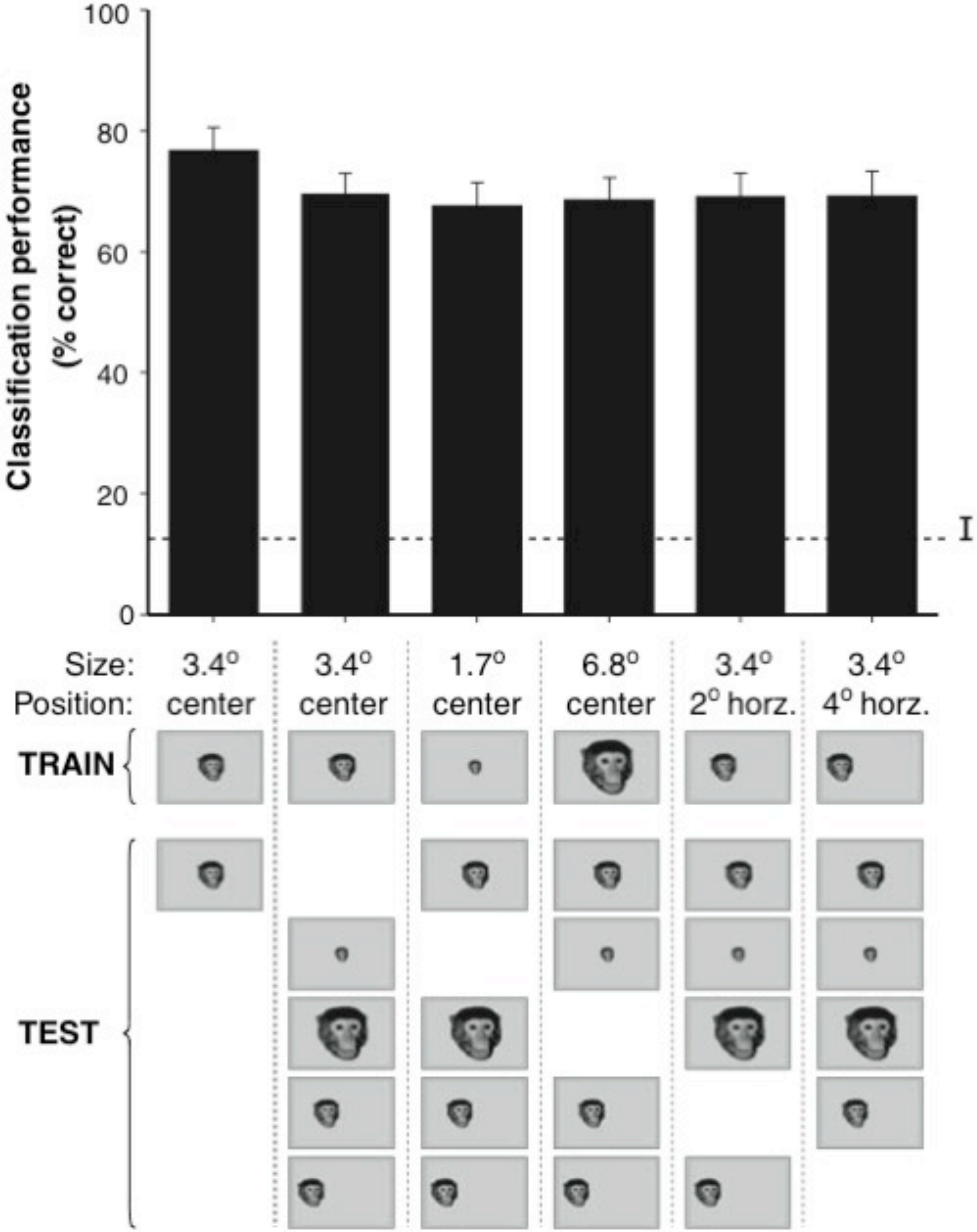
The Ventral Visual Stream

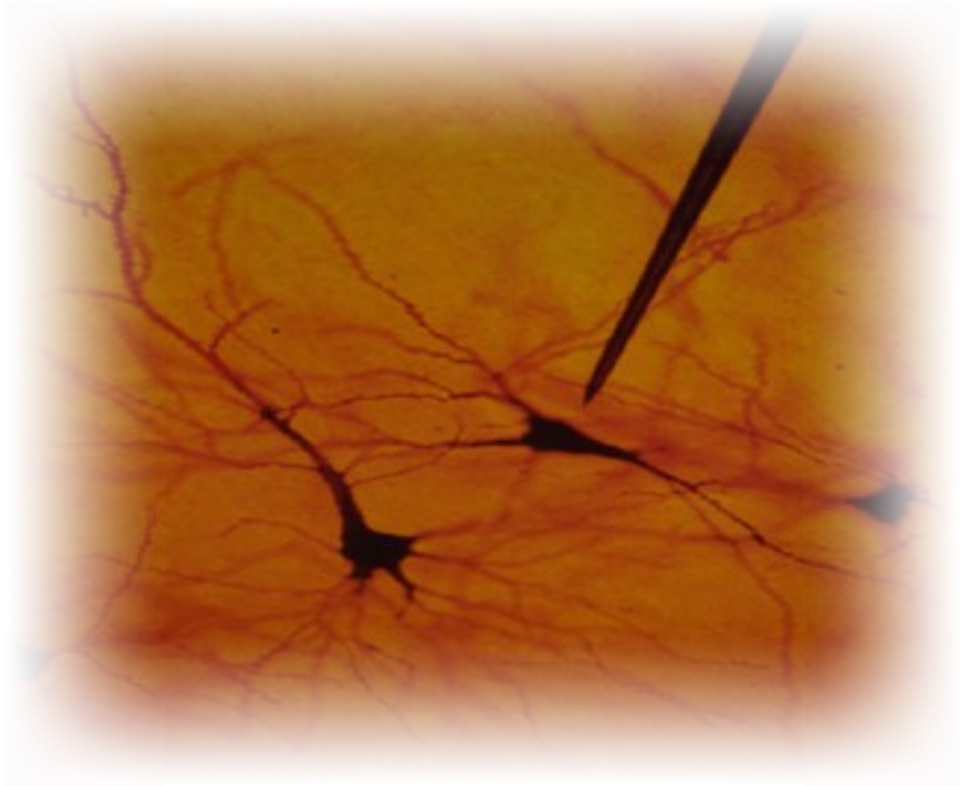


The Ventral Visual Stream



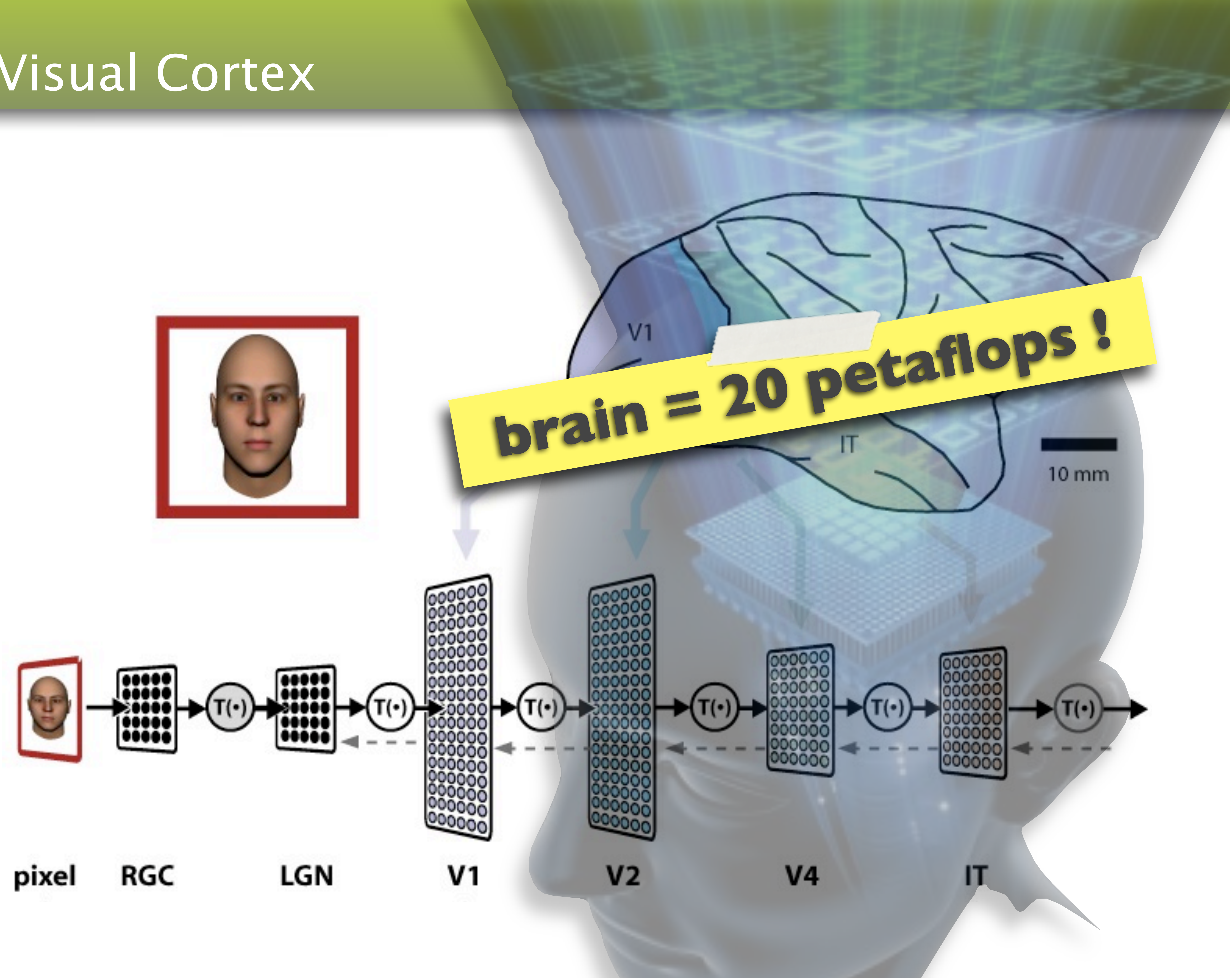
IT Cortex can do object recognition





Hung, Kreiman*, Poggio and DiCarlo,* ***Science*** *(2005)*

Visual Cortex



The need for **speed**

The need for **speed**

- **billions** of neurons and synapses

The need for **speed**

- **billions** of neurons and synapses
- **large-scale** natural evolution (“high-throughput screening” of neural architectures)

The need for **speed**

- **billions** of neurons and synapses
- **large-scale** natural evolution (“high-throughput screening” of neural architectures)
- **hours** of unsupervised learning experience

The need for **speed**

- **billions** of neurons and synapses
- **large-scale** natural evolution (“high-throughput screening” of neural architectures)
- **hours** of unsupervised learning experience
- faithful reproduction of other models (i.e. blend **many highly tuned** techniques)

Wanna Play with The Big Guys?

But it's too expensive !!!

Take a chance.

Our strategy

Our strategy

Capitalizing on non-scientific high-tech markets and their $billions of R&D...

Our strategy

Capitalizing on non-scientific high-tech markets and their $billions of R&D...

– **Gaming:** GPUs, PlayStation 3 (CellBE)

Our strategy

Capitalizing on non-scientific high-tech markets and their $billions of R&D...

- **Gaming:** GPUs, PlayStation 3 (CellBE)
- **Web 2.0:** Cloud Computing (Amazon, Google)

Need for speed

Hardware

Software

Science

GPUs (since 2006)

7800 GTX (2006)	Monster16GPU (2008)	Tesla Cluster (2009)
OpenGL/Cg	*CUDA*	*CUDA/OpenCL*
C++/Python	*Python*	*Python*

Build your own!

Our 16-GPU Monster-Class Supercomputer

the world's most compact (18"x18"x18") and inexpensive ($3000) supercomputer

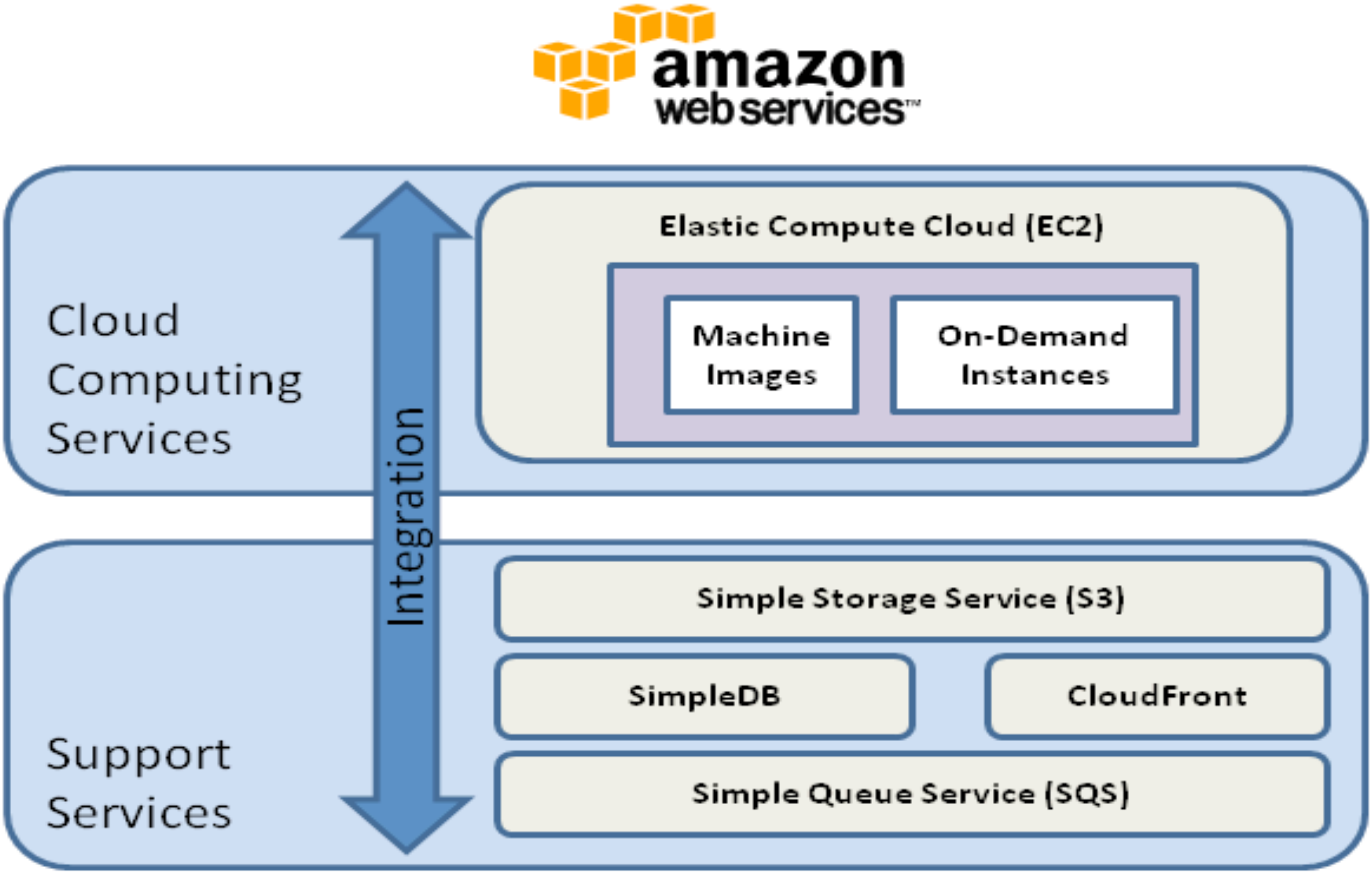
Cell Broadband Engine (since 2007)

Teraflop Playstation3 clusters:

DiCarlo Lab / MIT

Cox Lab / Harvard

Amazon Cloud Computing (since 2008)



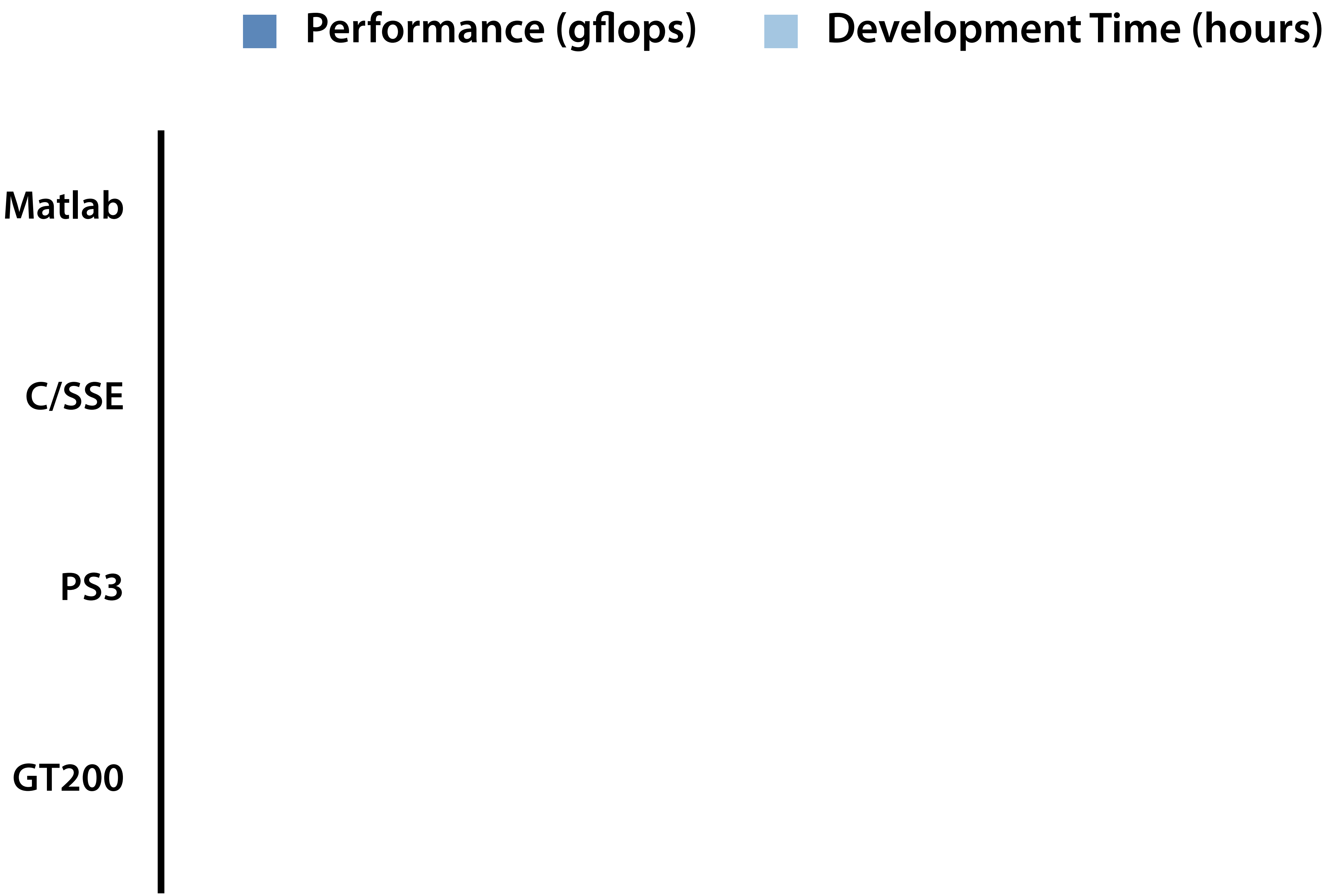
Some numbers...

3D Filterbank Convolution

Performance (gflops) | Development Time (hours)

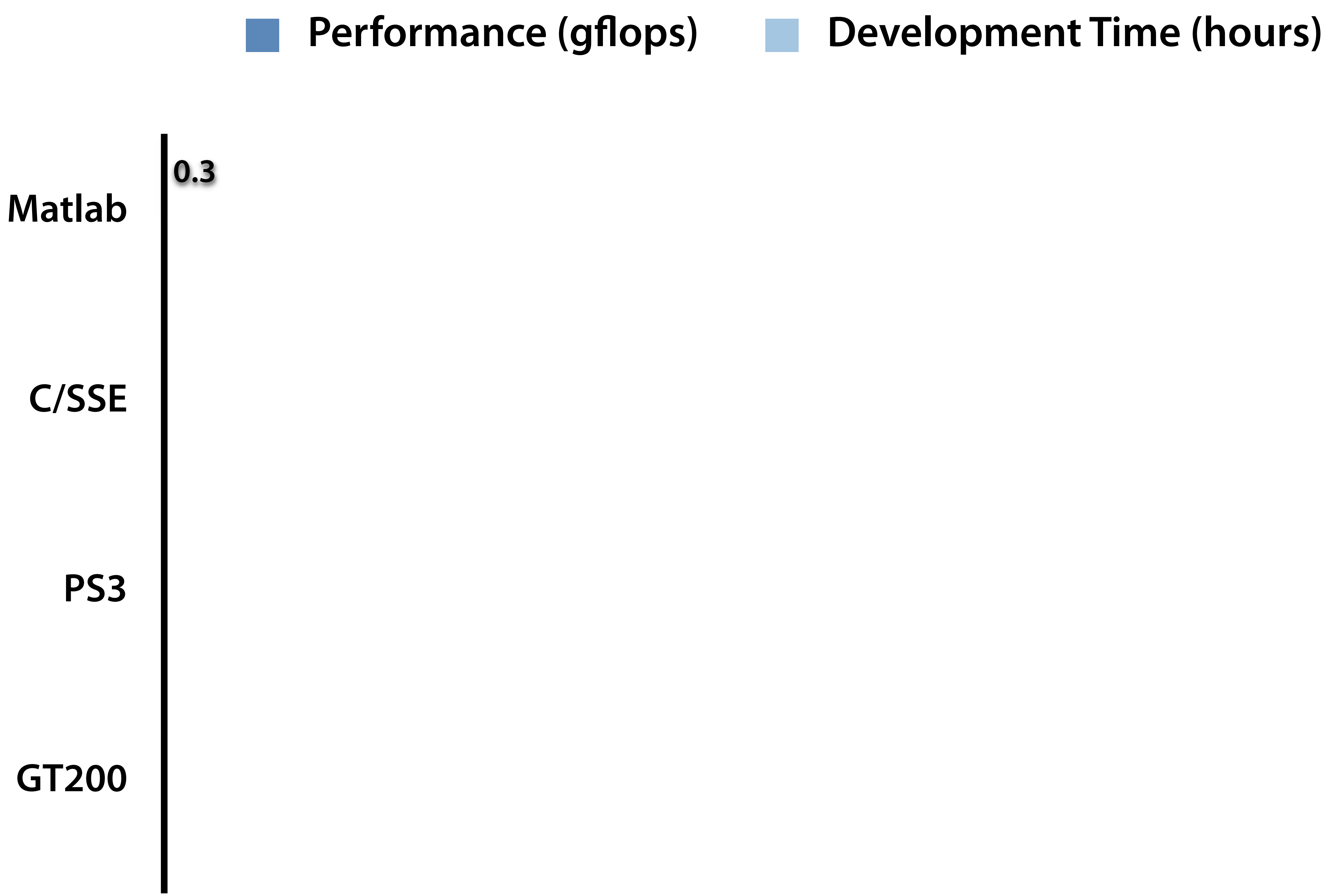
Some numbers...

3D Filterbank Convolution



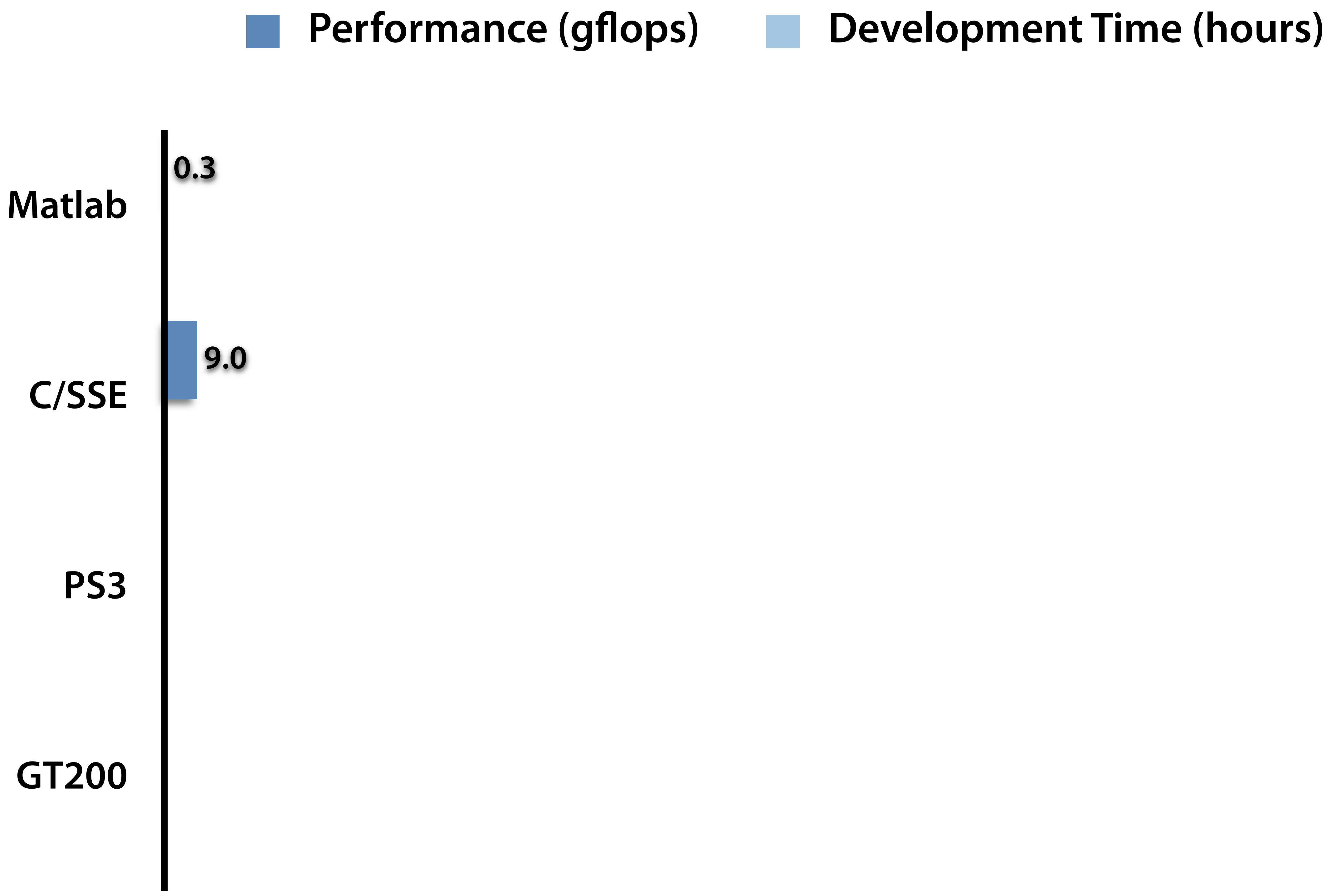
Some numbers...

3D Filterbank Convolution



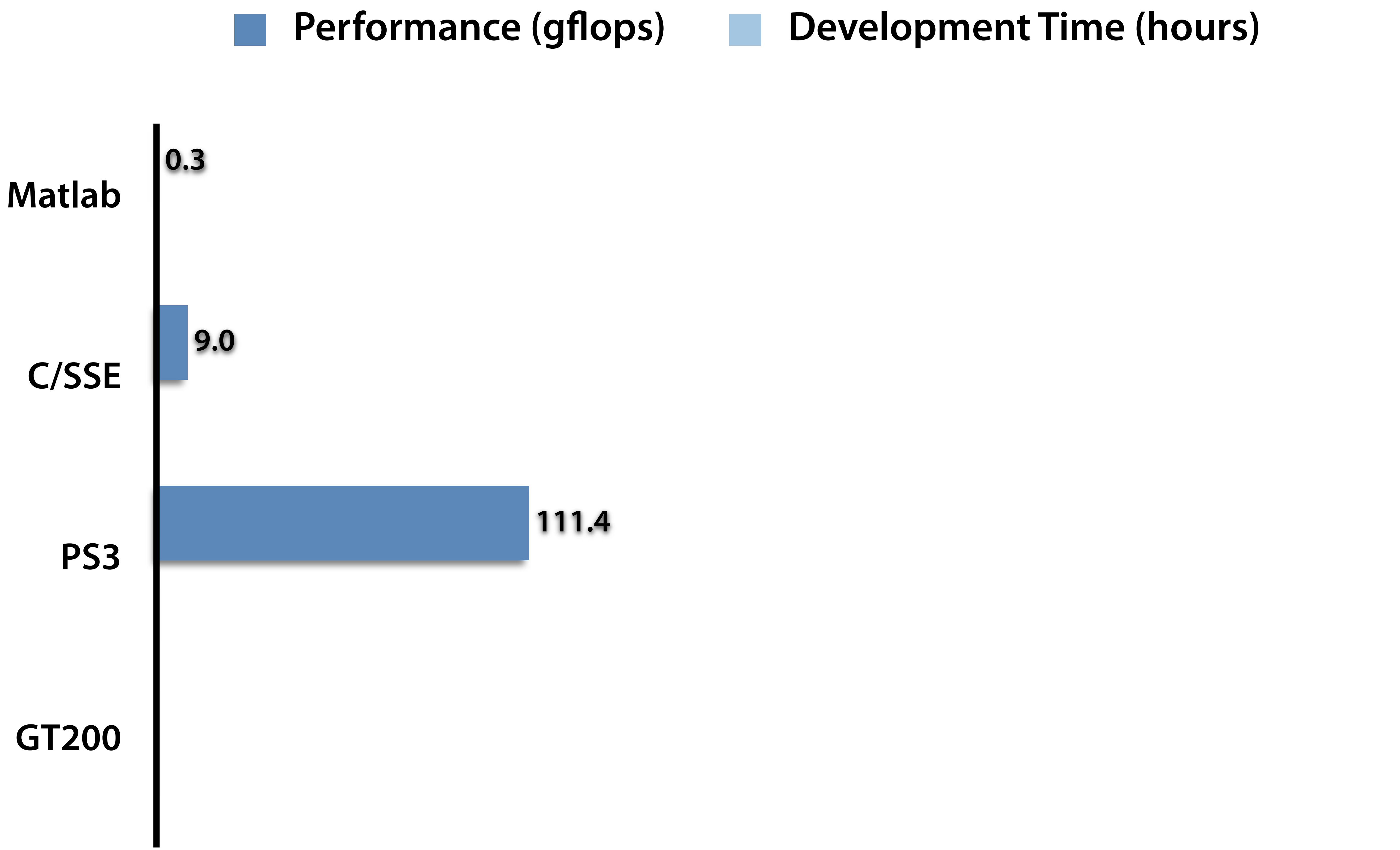
Some numbers...

3D Filterbank Convolution



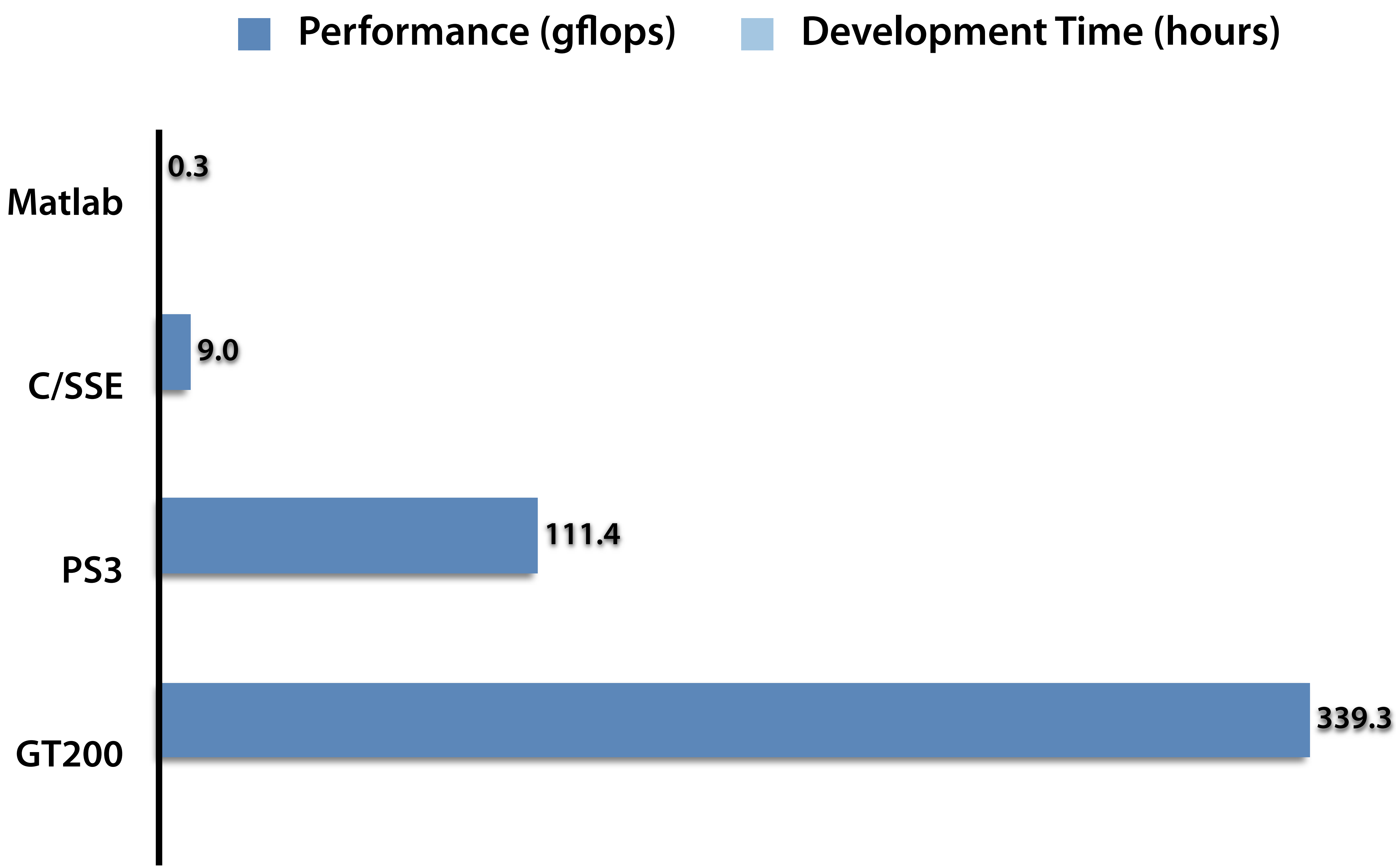
Some numbers...

3D Filterbank Convolution

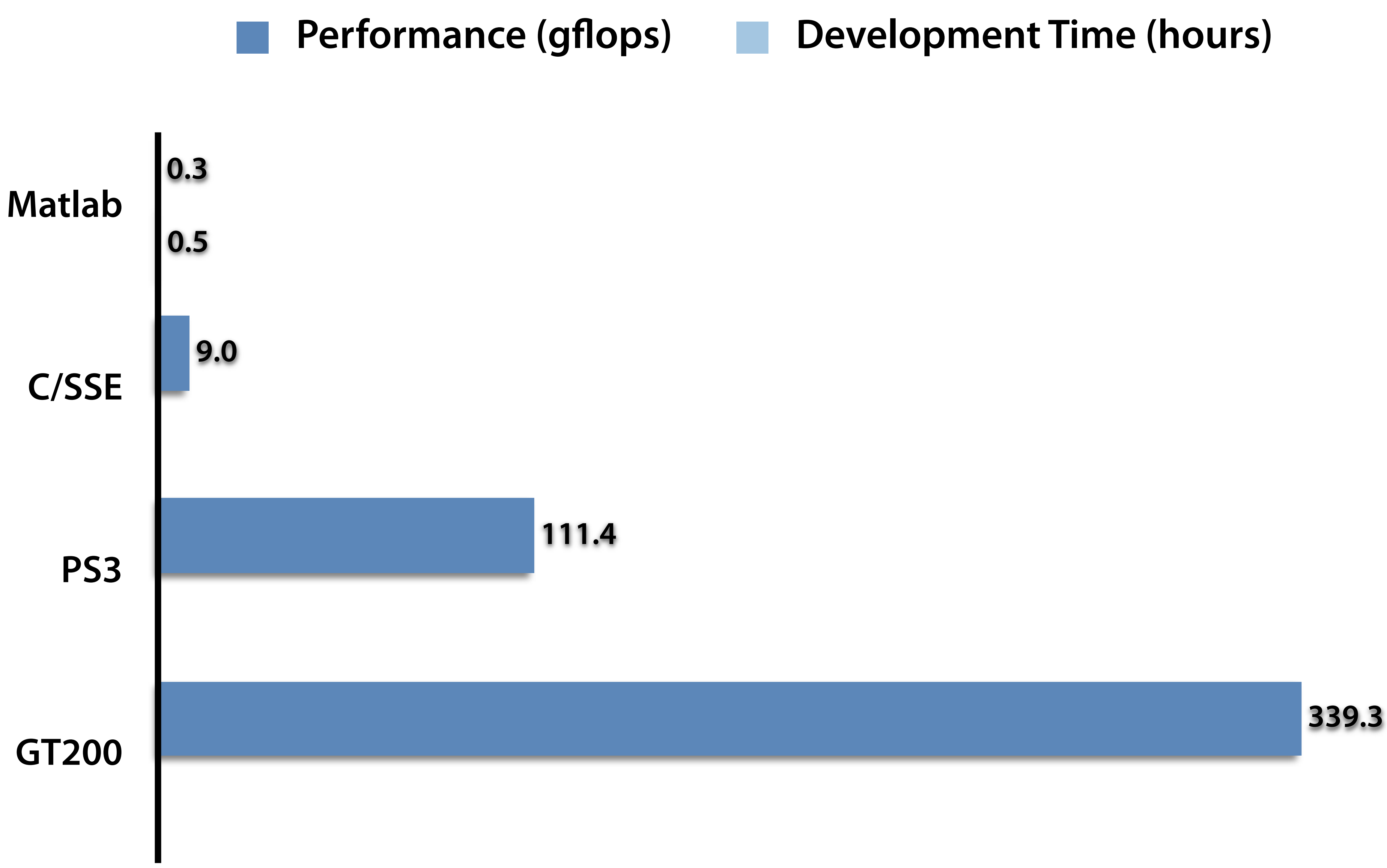


Some numbers...

3D Filterbank Convolution

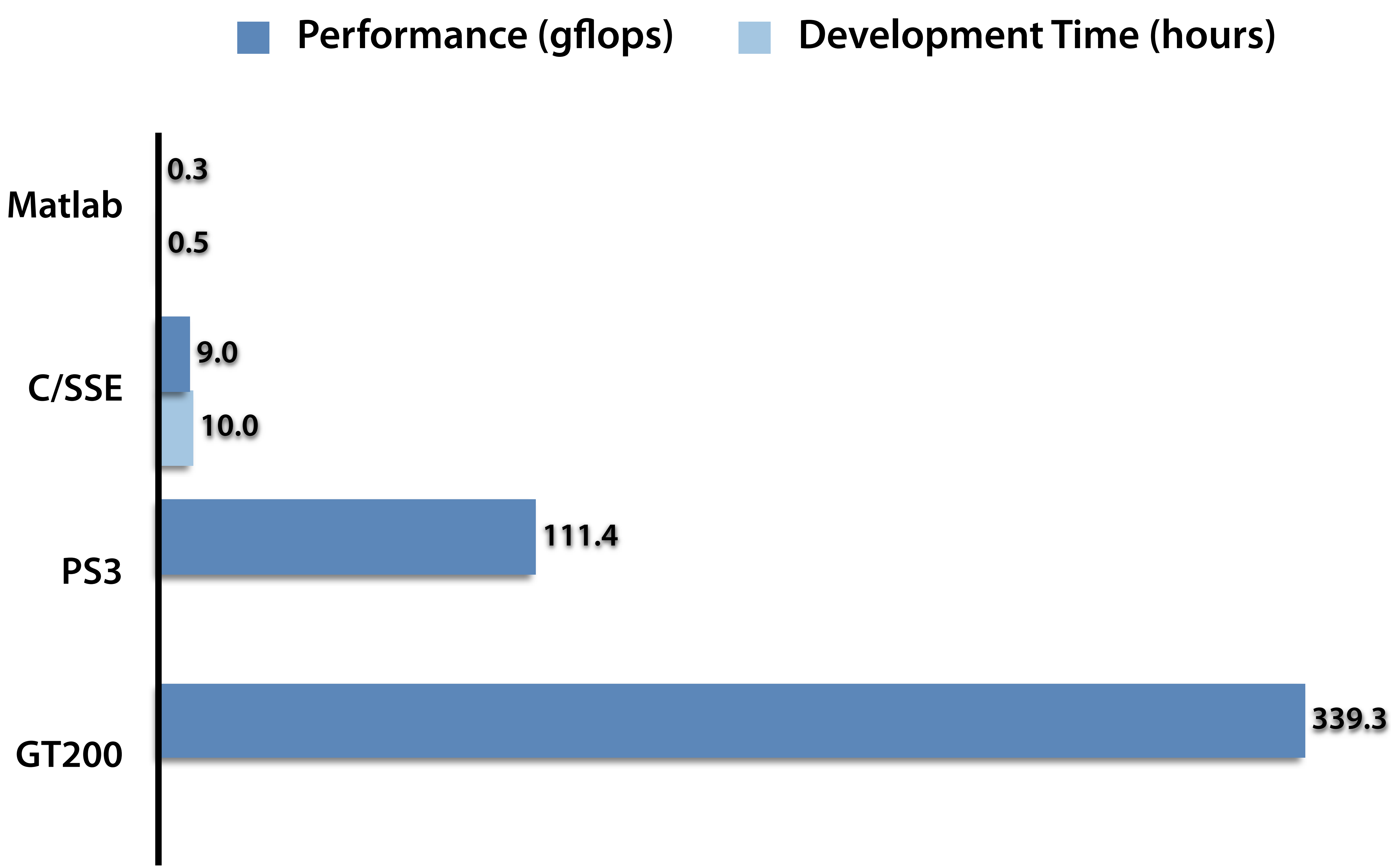


Some numbers...



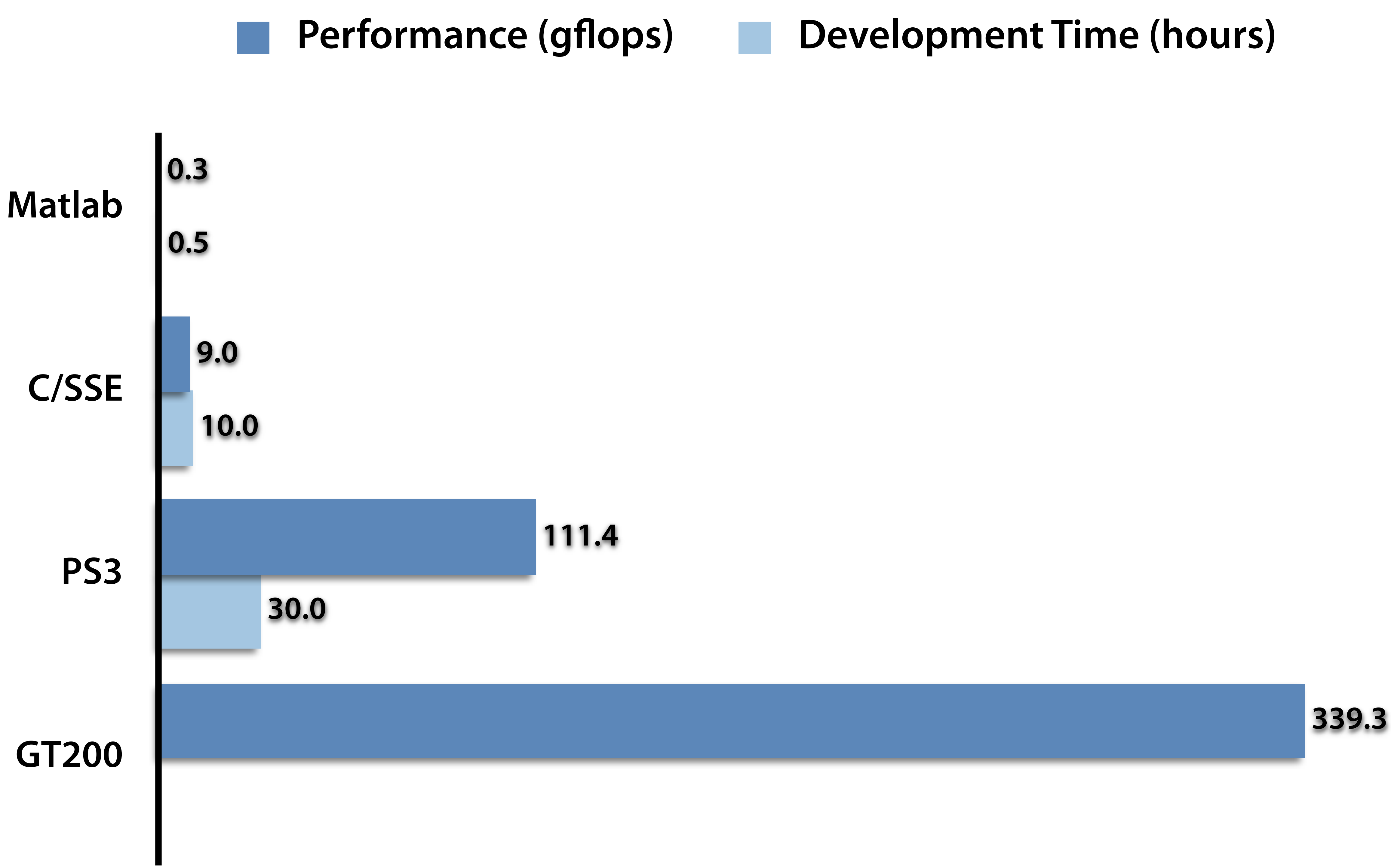
Some numbers...

3D Filterbank Convolution



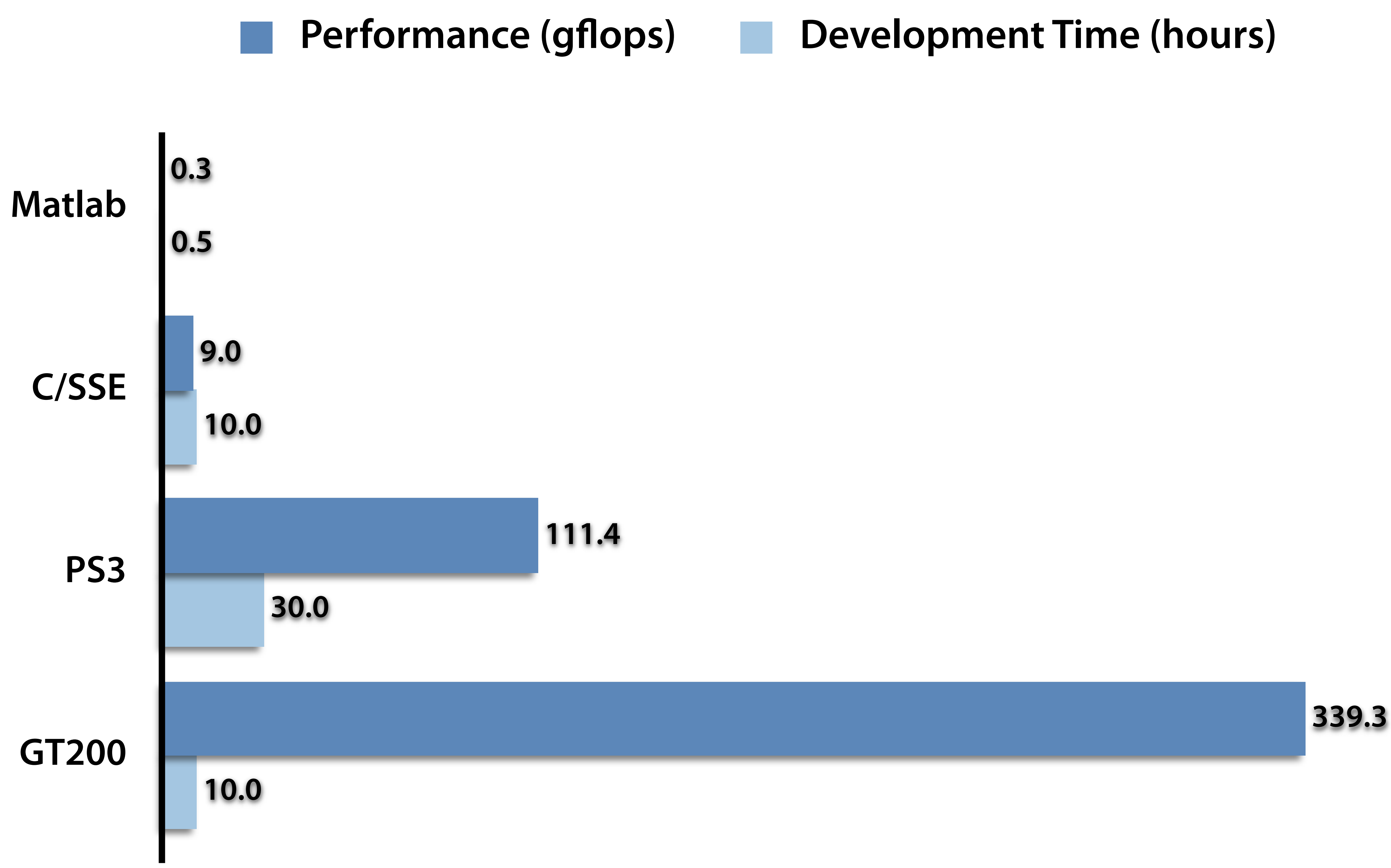
Some numbers...

3D Filterbank Convolution



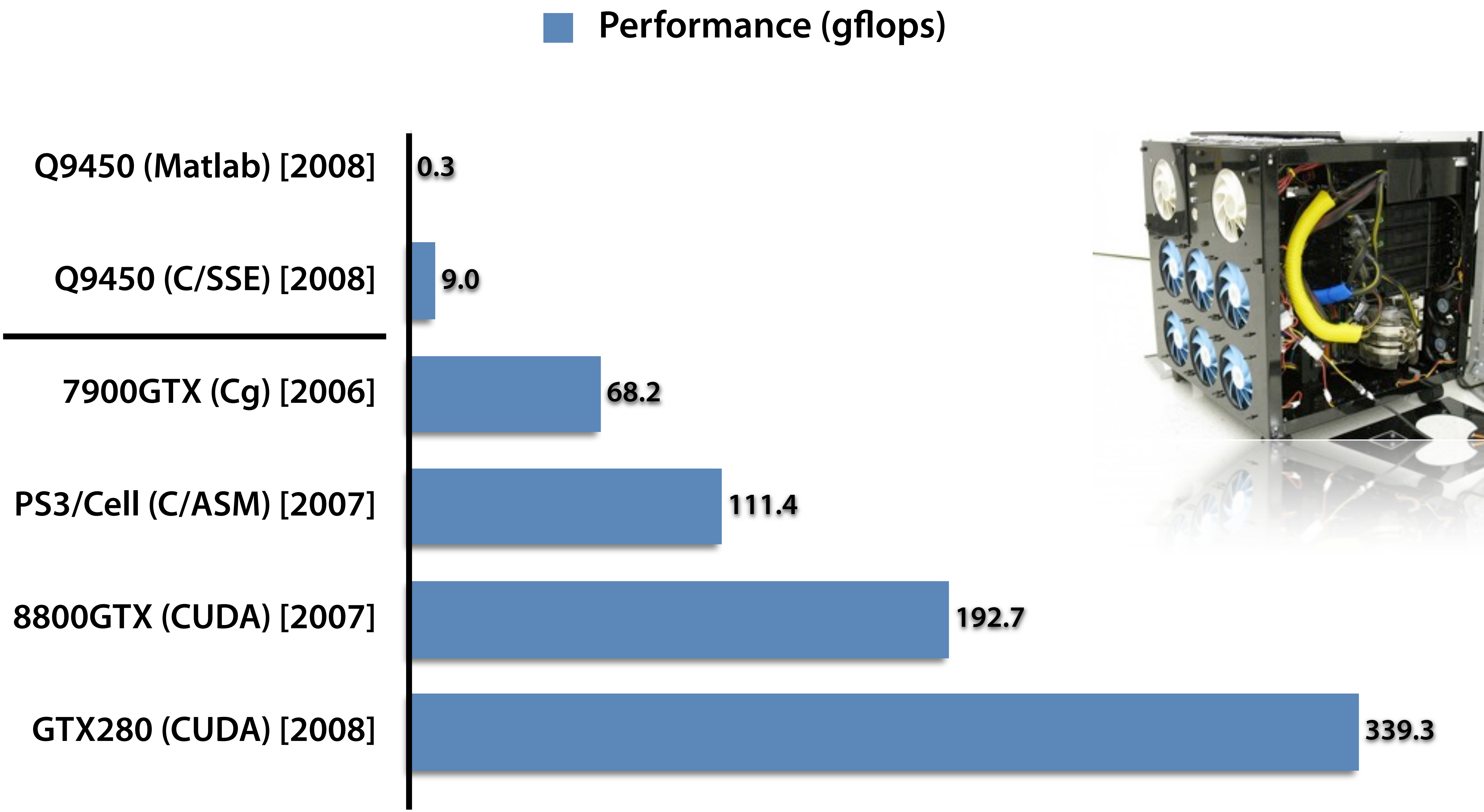
Some numbers...

3D Filterbank Convolution



Some numbers...

3D Filterbank Convolution



Need for speed
Hardware
Software
Science

What do we all want?

- Ease of use
- **Maximum raw speed**
- Ease of extension
- Hardware “agnostic”

A little story

You just finished your code...

A little story

You just finished your code...

1. You run it on one image: it works! 

A little story

You just finished your code...

1. You run it on one image: it works! 
2. You adjust your parameters: it's slow! 

A little story

You just finished your code...

1. You run it on one image: it works! 

2. You adjust your parameters: it’s slow! 

3. Your optimize your code: it’s fast now! 

A little story

You just finished your code...

1. You run it on one image: it works!

2. You adjust your parameters: it’s slow!

3. Your optimize your code: it’s fast now!

4. You run it on another image: it’s slow now!

A little story

You just finished your code...

1. You run it on one image: it works!

2. You adjust your parameters: it’s slow!

3. Your optimize your code: it’s fast now!

4. You run it on another image: it’s slow now!

5. You repeat or you stop...


Here are the keys
to Easy-High-Performance !

Meta-programming?

Meta-programming !

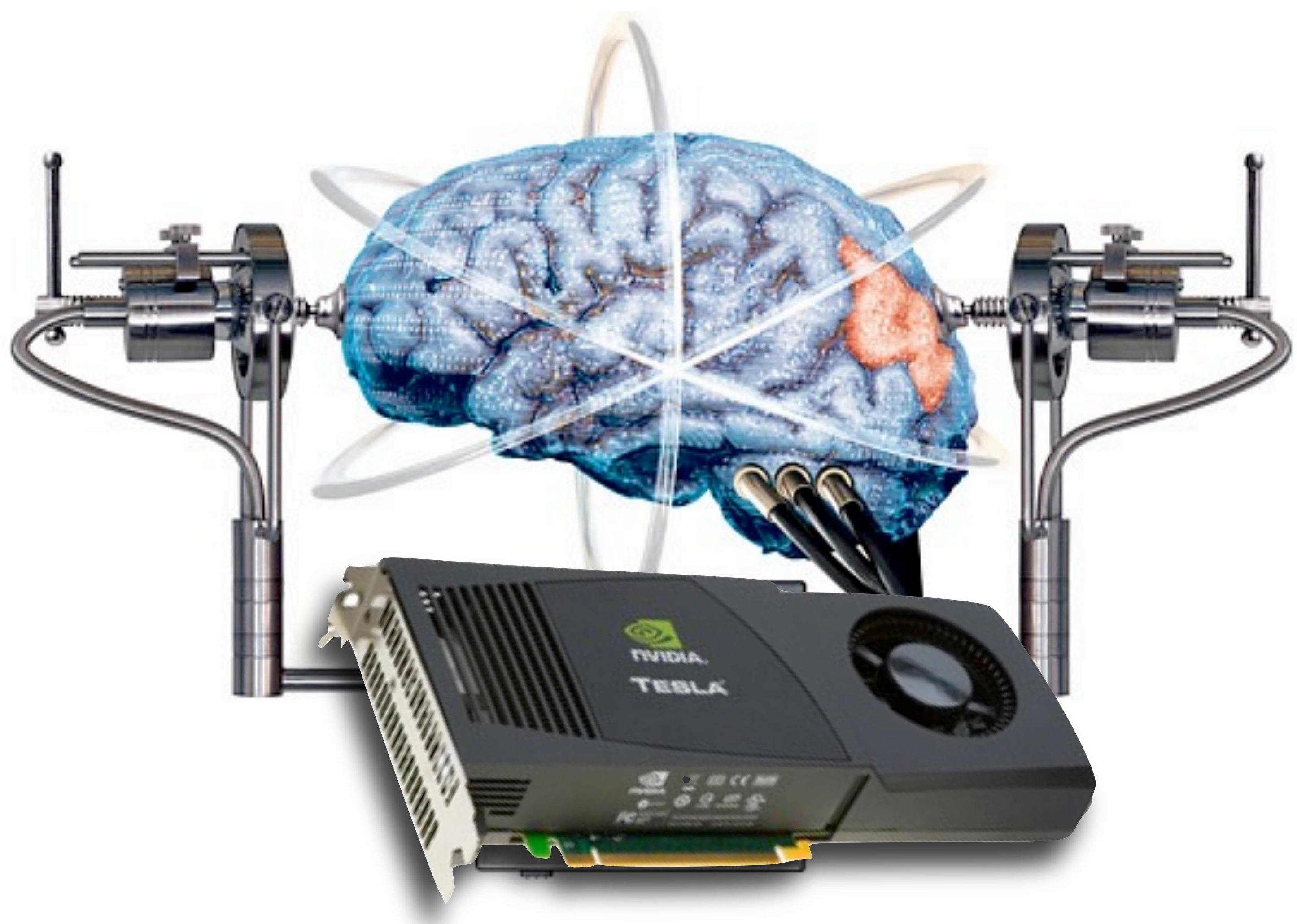
Leave the **grunt-programming** to the computer (i.e. auto-tuning like ATLAS or FFTW)

- Dynamically compile **specialized versions** of the same kernel for different conditions (~Just-in-Time Compilation (JIT))
- **Smooth** syntactic ugliness: unroll loops, index un-indexable registers
- **Dynamic**, empirical run-time **tuning**

Meta-programming!

"Instrumentalize" your solutions:

- Block size
- Work size
- Loop unrolling
- Pre-fetching
- Spilling
- **etc.**



Meta-programming!

Let the computer find the **optimal code**:

- brute–force search with a **global objective**
- machine–learning approach with **local objectives** and **hidden variables** (advanced)
  - eg. PyCuda makes this easy:
    - Access properties of compiled code: `func.{registers,lmem,smem}`
    - Exact GPU timing via events
    - Can calculate hardware-dependent MP occupancy

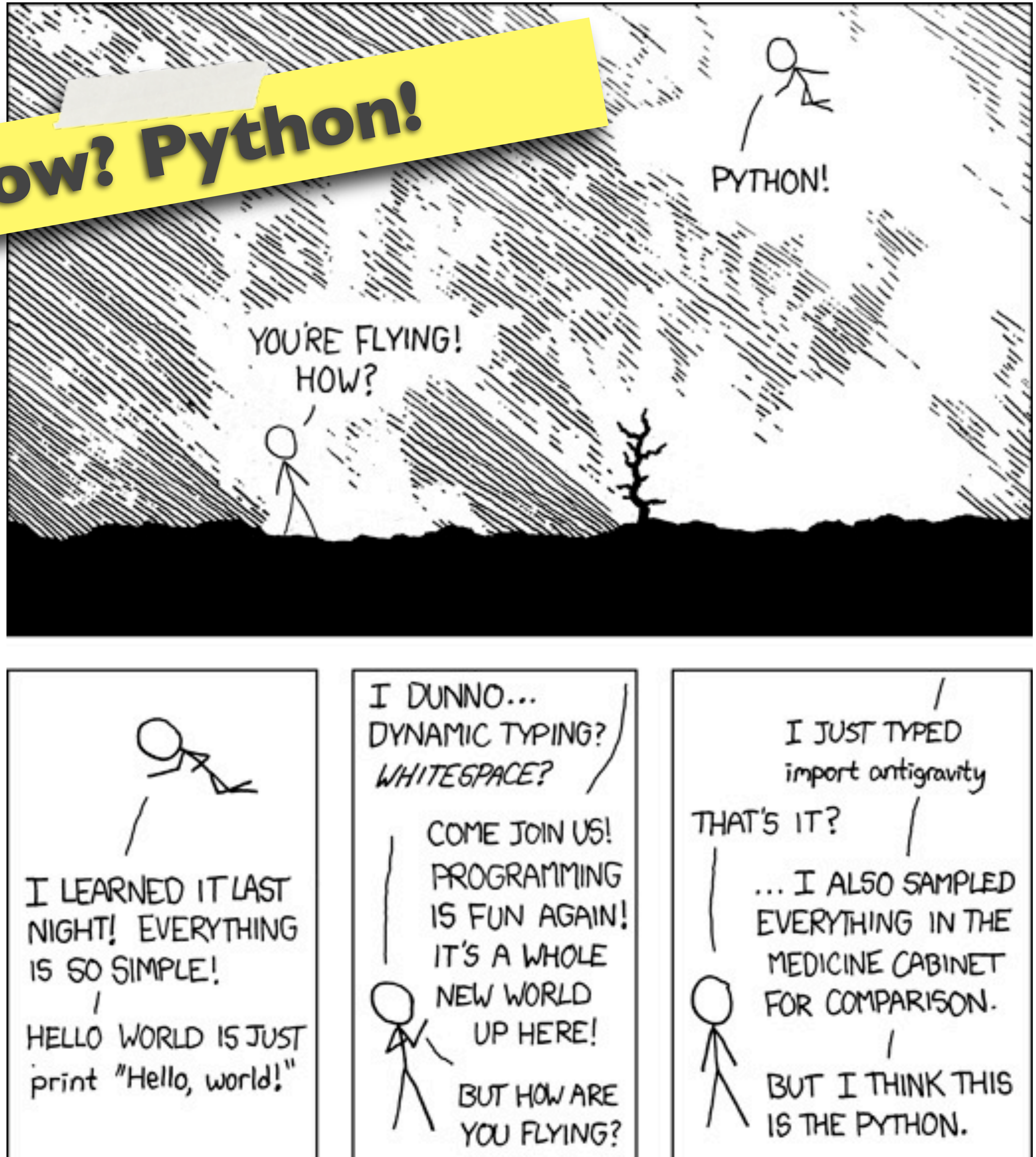
Meta-programming!

- GPU Metaprogramming using **PyCUDA**: Methods & Applications
  - Andreas Kloeckner (Brown)
  - Friday 1pm @ Empire

How ?

Our mantra: always use the right tool !

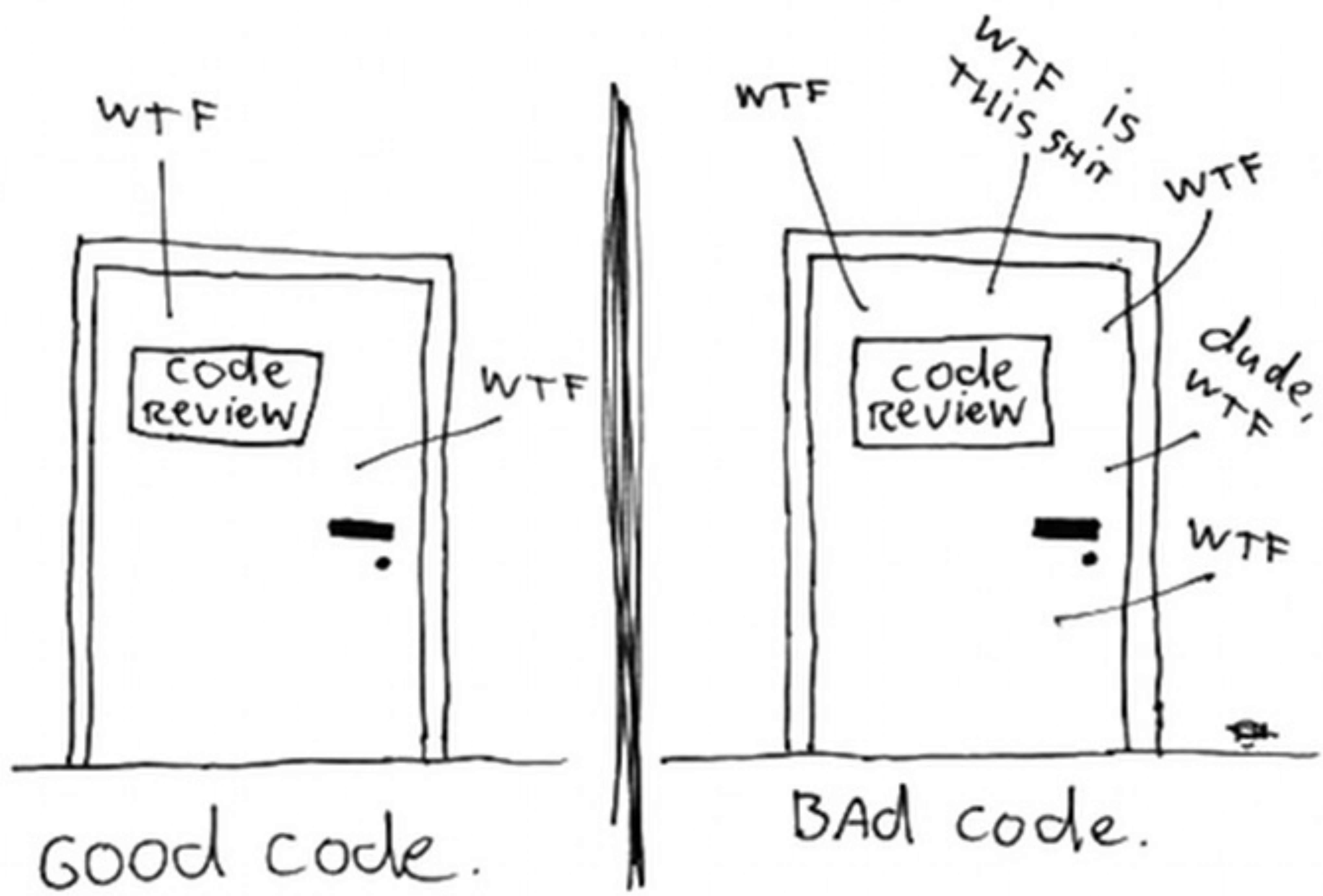
How? Python!



PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Meta-programming requires **careful engineering**

Meta-programming requires **careful engineering**



Need for speed
Hardware
Software
Science

The Approach:
Forward Engineering the Brain

REVERSE

Study
Natural System

⇄

FORWARD

Build
Artificial System

The Approach:
Forward Engineering the Brain

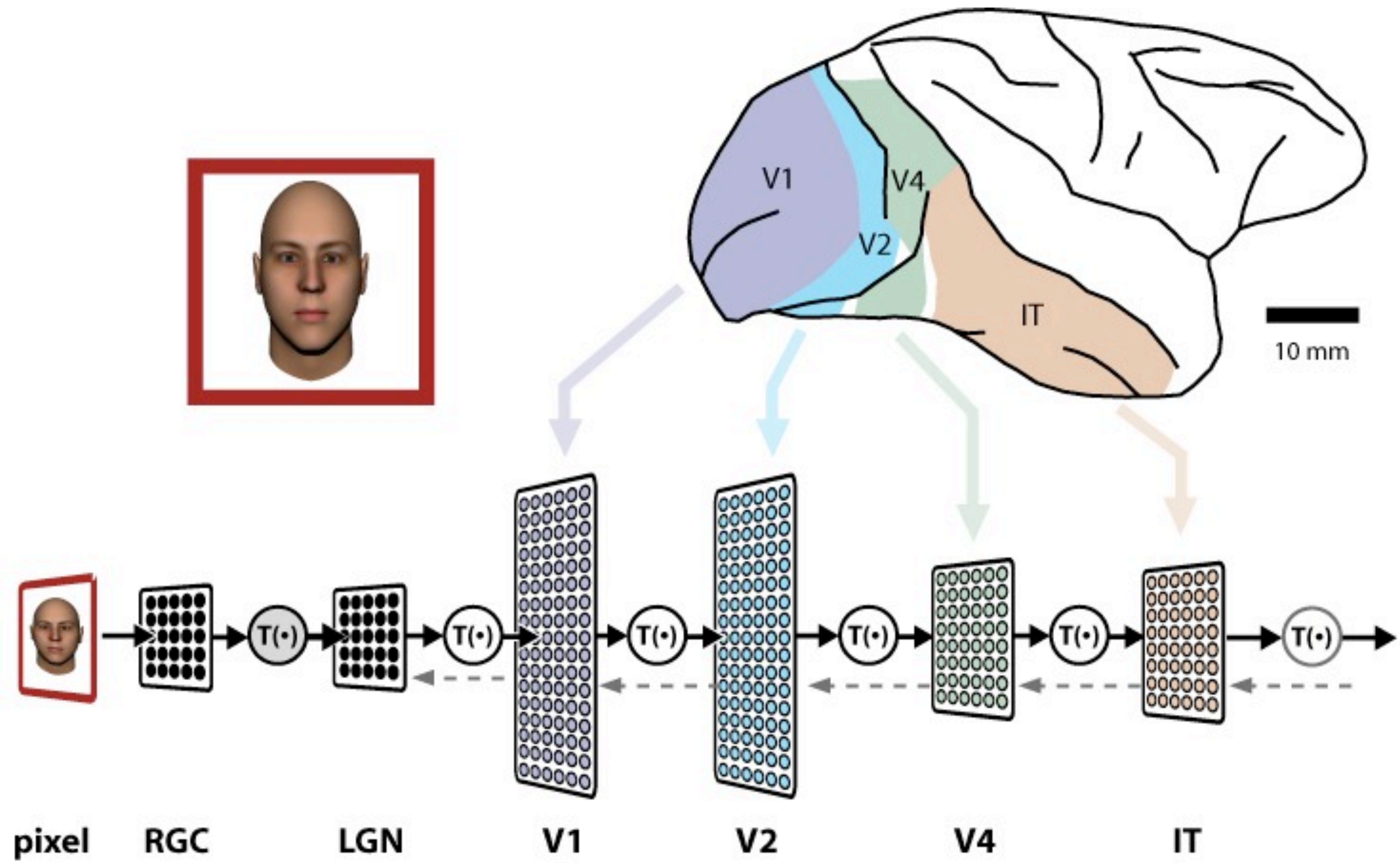
REVERSE

Study
Natural System

FORWARD

Build
Artificial System

Visual System



How are things done normally?

How are things done normally?

Usual Formula:

How are things done normally?

Usual Formula:

1) One grad student

How are things done normally?

Usual Formula:

1) **One grad student**

2) **One Model** (size limited by runtime)

How are things done normally?

Usual Formula:

1) One grad student

2) One Model (size limited by runtime)

3) Performance numbers on a few standard test sets

How are things done normally?

Usual Formula:

1) One grad student

2) One Model (size limited by runtime)

3) Performance numbers on a few standard test sets

4) yay. we. rock.

How are things done normally?

Usual Formula:

1) One grad student

2) One Model (size limited by runtime)

3) Performance numbers on a few standard test sets

4) yay. we. rock.

5) One Ph.D.

How are things done normally?

Usual Formula:

1) One grad student

2) One Model (size limited by runtime)

3) Performance numbers on a few standard test sets

4) yay. we. rock.

5) One Ph.D.

Doing things a little bit differently

Doing things a little bit differently

1) One grad student

Doing things a little bit differently

1) **One grad student**

2) **~~One~~ Hundreds of Thousands of BIG Models**

Doing things a little bit differently

1) One grad student

2) ~~One~~ Hundreds of Thousands of BIG Models

3) Performance numbers on a few standard test sets

Doing things a little bit differently

1) One grad student

2) ~~One~~ Hundreds of Thousands of BIG Models

3) Performance numbers on a few standard test sets

Doing things a little bit differently

1) One grad student

2) ~~One~~ Hundreds of Thousands of BIG Models

3) Performance numbers on a few standard test sets

4) yay. we. rock.

Doing things a little bit differently

1) One grad student

2) ~~One~~ Hundreds of Thousands of BIG Models

3) Performance numbers on a few standard test sets

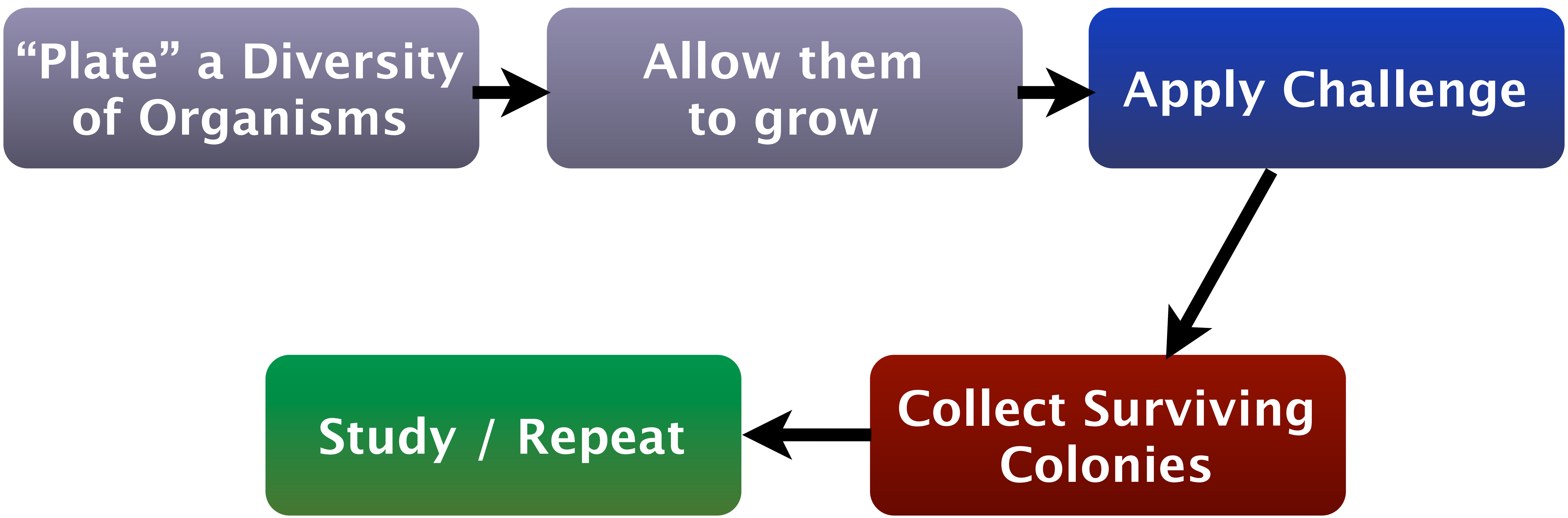
4) yay. we. rock.

5) ~~Hundreds of Thousands~~ One PhD ?

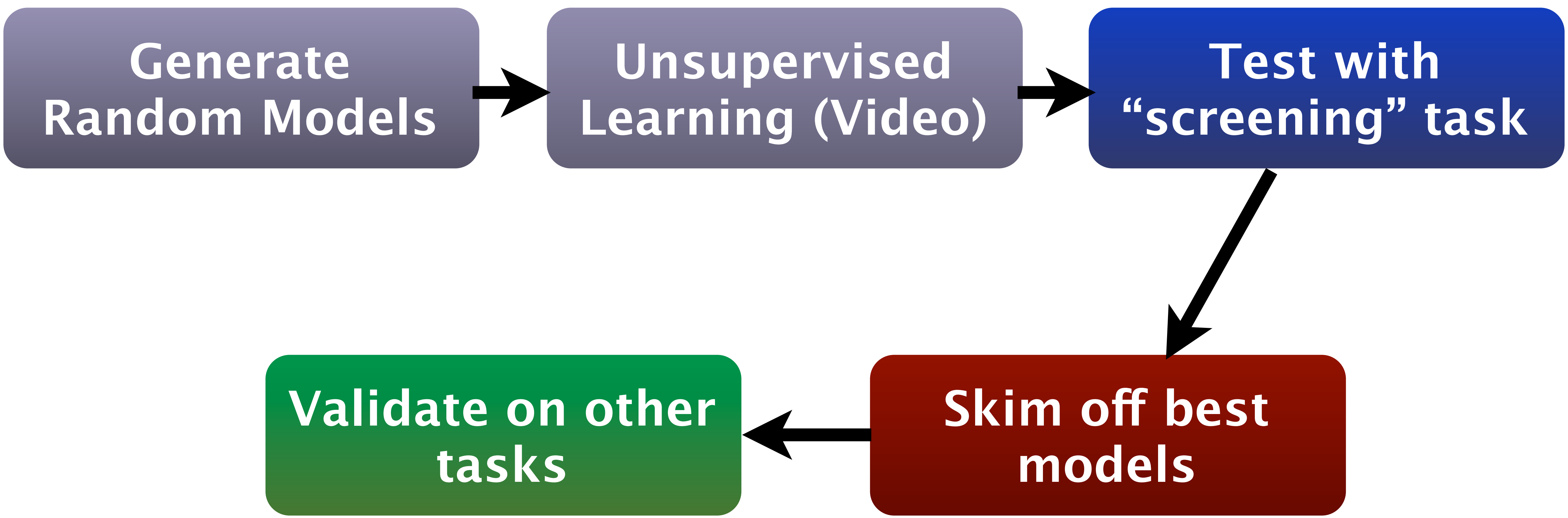
High-Throughput Screening

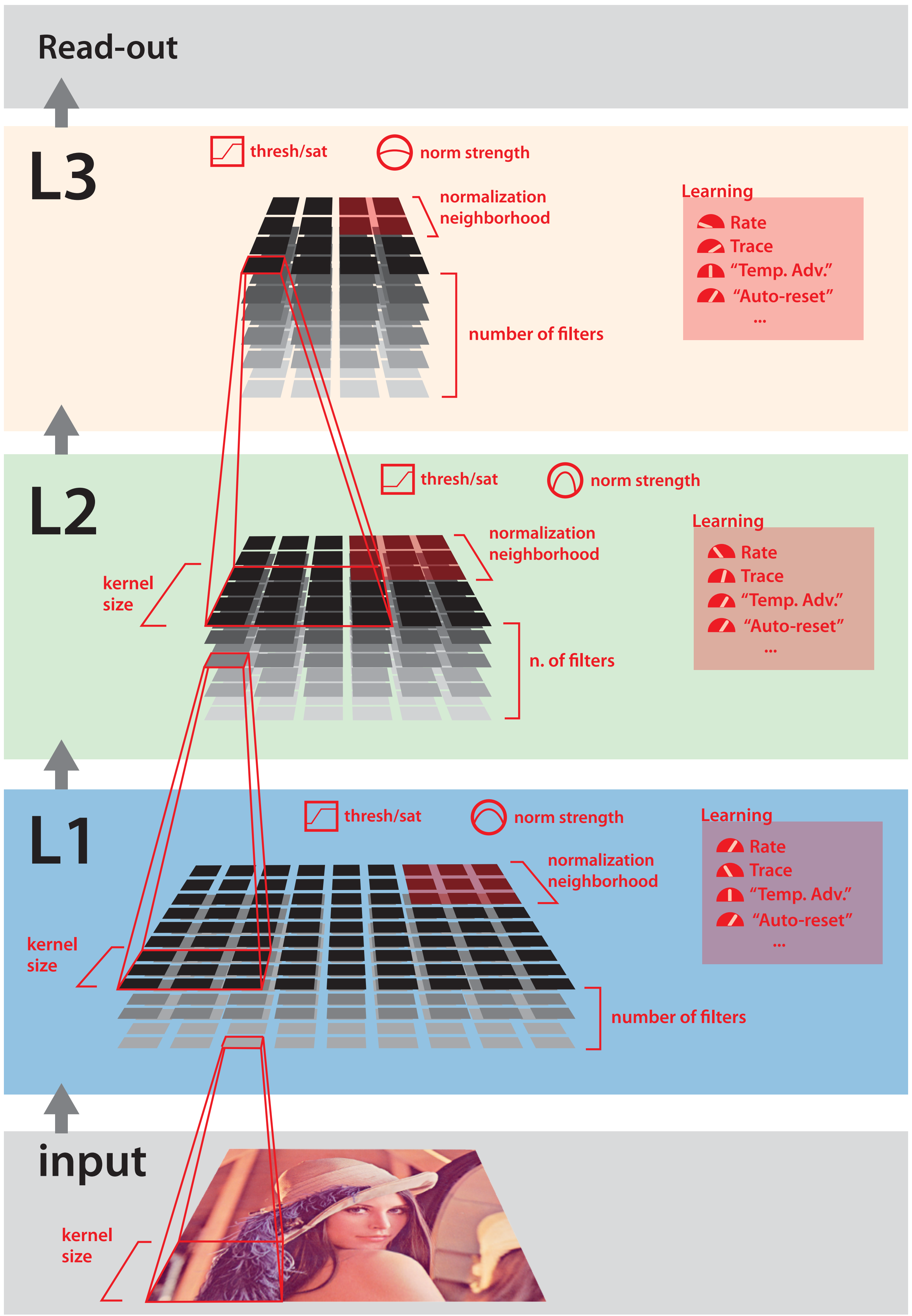
High-Throughput Screening

Pipeline: Biology

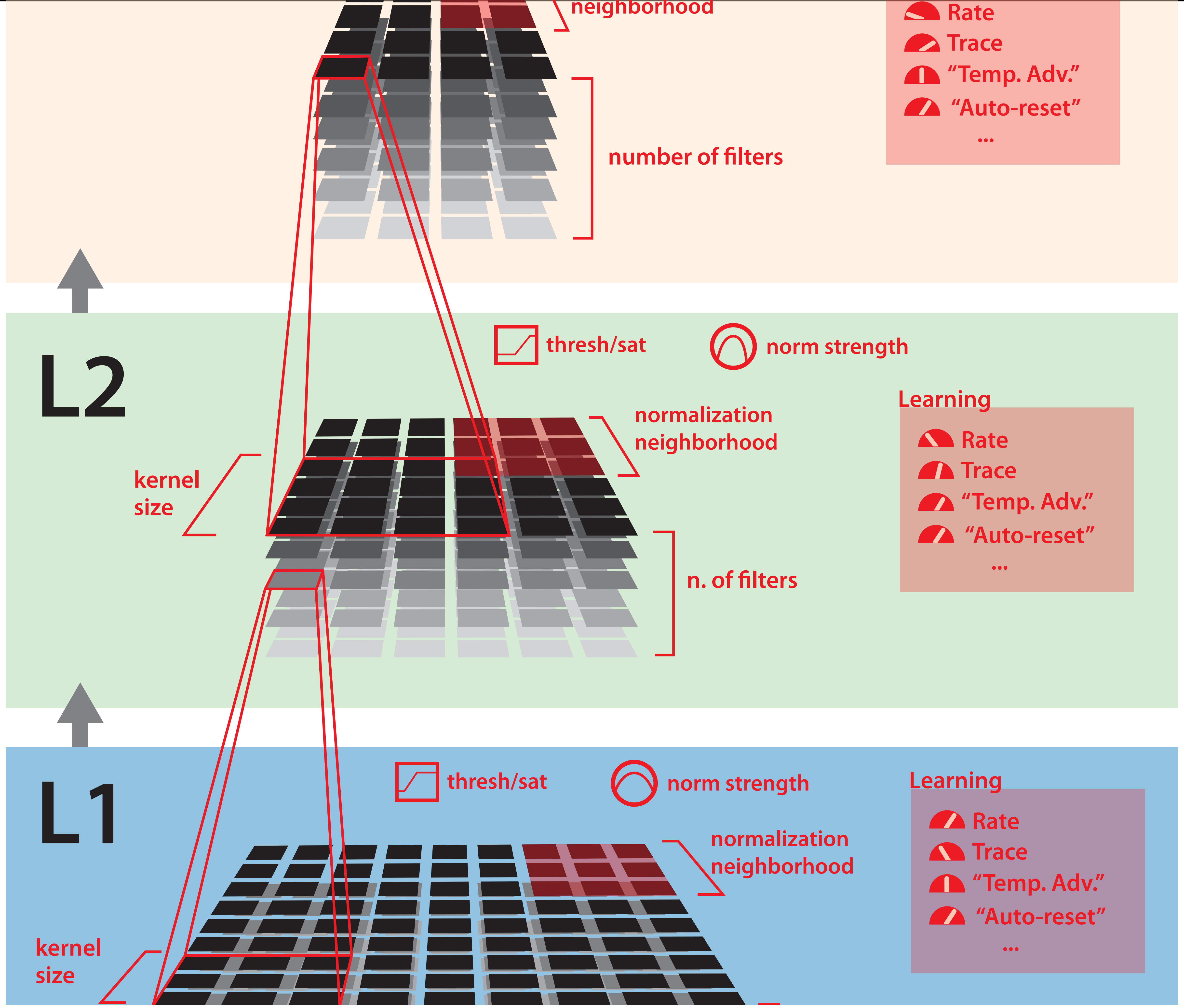


Pipeline: Biology-Inspired Vision





Read-out
L3
thresh/sat
norm strength
normalization neighborhood
number of filters
Learning
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
L2
thresh/sat
norm strength
normalization neighborhood
kernel size
n. of filters
Learning
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
L1
thresh/sat
norm strength
normalization neighborhood
kernel size
number of filters
Learning
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
input
kernel size



neighborhood
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
number of filters
L2
thresh/sat
norm strength
Learning
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
normalization neighborhood
kernel size
n. of filters
L1
thresh/sat
norm strength
Learning
Rate
Trace
"Temp. Adv."
"Auto-reset"
...
normalization neighborhood
kernel size

A Broad **Parametric** Model

Normalize

$N_i = Input_i / norm(Input_{neighborhd})$

Compute Filter Responses

$R_i = F_i \otimes N$

$R_i < thresh: R_i = thresh$

$R_i > sat: R_i = sat$

Determine a "Winning Filter"

$R_i' = (\sum T_k * H_k) * R_i$

winner: $max(R_i')$

Update Filter

$F_{winning} = F_{winning} + \text{learning rate} * N$

A Broad **Parametric** Model

Normalize

$N_i = Input_i / norm(Input_{neighborhd})$

Compute Filter Responses

$R_i = F_i \otimes N$

$R_i < thresh: R_i = thresh$

$R_i > sat: R_i = sat$

Determine a "Winning Filter"

$R_i' = (\sum T_k * H_k) * R_i$

winner: $max(R_i')$

Update Filter

$F_{winning} = F_{winning} + learning\ rate * N$

- **Optimize "Coverage"**

A Broad **Parametric** Model

Normalize

$N_i = Input_i / norm(Input_{neighborhd})$

Compute Filter Responses

$R_i = F_i \otimes N$

$R_i < thresh: R_i = thresh$

$R_i > sat: R_i = sat$

Determine a "Winning Filter"

$R_i' = (\sum T_k * H_k) * R_i$

winner: $max(R_i')$

Update Filter

$F_{winning} = F_{winning} + learning\ rate * N$

- **Optimize "Coverage"**

(filters span the range of observed inputs)

A Broad **Parametric** Model

Normalize
$N_i = Input_i / norm(Input_{neighborhd})$

Compute Filter Responses
$R_i = F_i \otimes N$
$R_i < thresh: R_i = thresh$
$R_i > sat: R_i = sat$

Determine a "Winning Filter"
$R_i' = (\sum T_k * H_k) * R_i$
winner: $max(R_i')$

Update Filter
$F_{winning} = F_{winning} + learning\ rate * N$

- **Optimize "Coverage"**
(filters span the range of observed inputs)

- **Privilege movement of filters in certain directions using temporal information**

A Broad **Parametric** Model

Normalize

$N_i = Input_i / norm(Input_{neighborhd})$

Compute Filter Responses

$R_i = F_i \otimes N$

$R_i < thresh: R_i = thresh$

$R_i > sat: R_i = sat$

Determine a "Winning Filter"

$R_i' = (\sum T_k * H_k) * R_i$

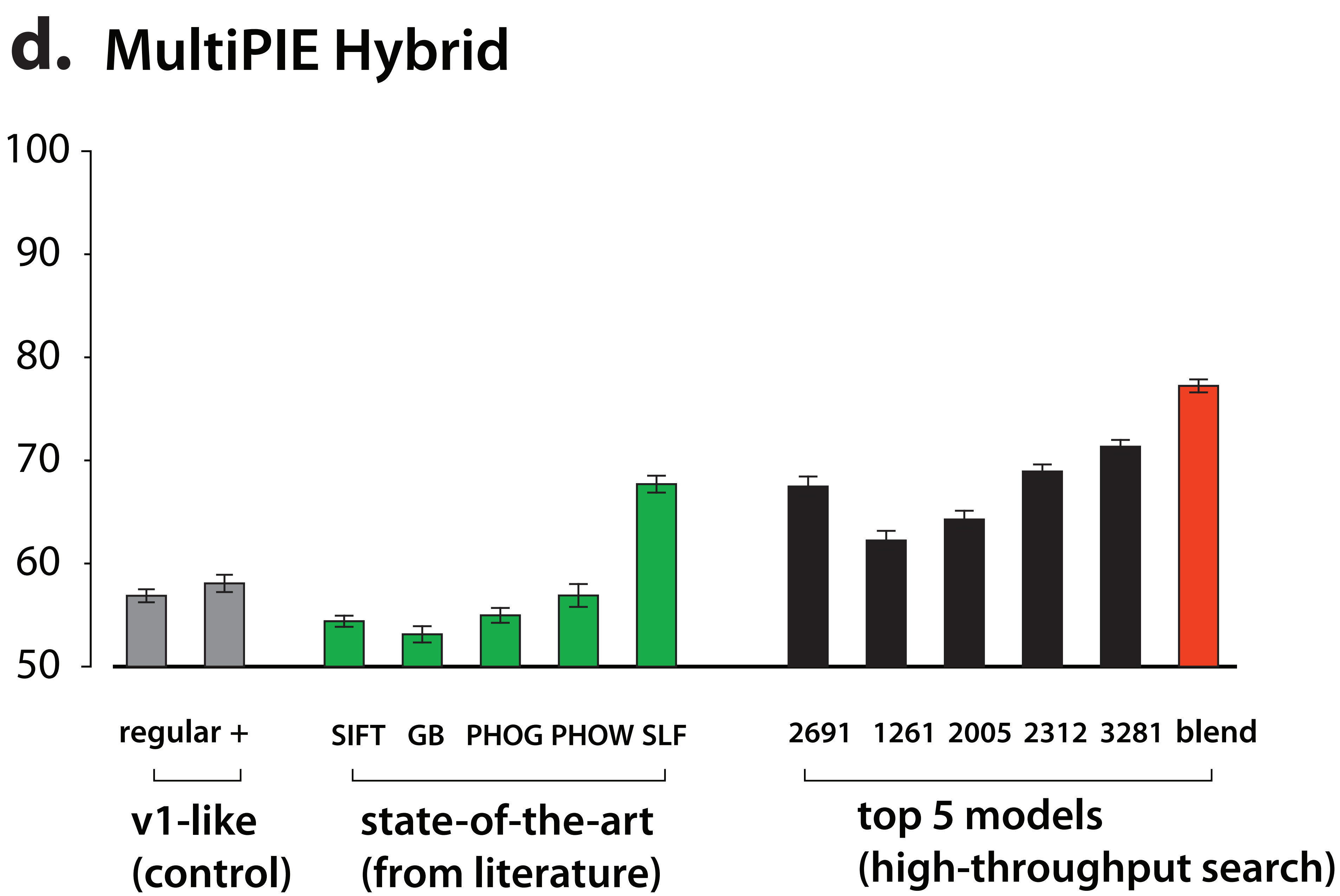
winner: $max(R_i')$

Update Filter

$F_{winning} = F_{winning} + \text{learning rate} * N$

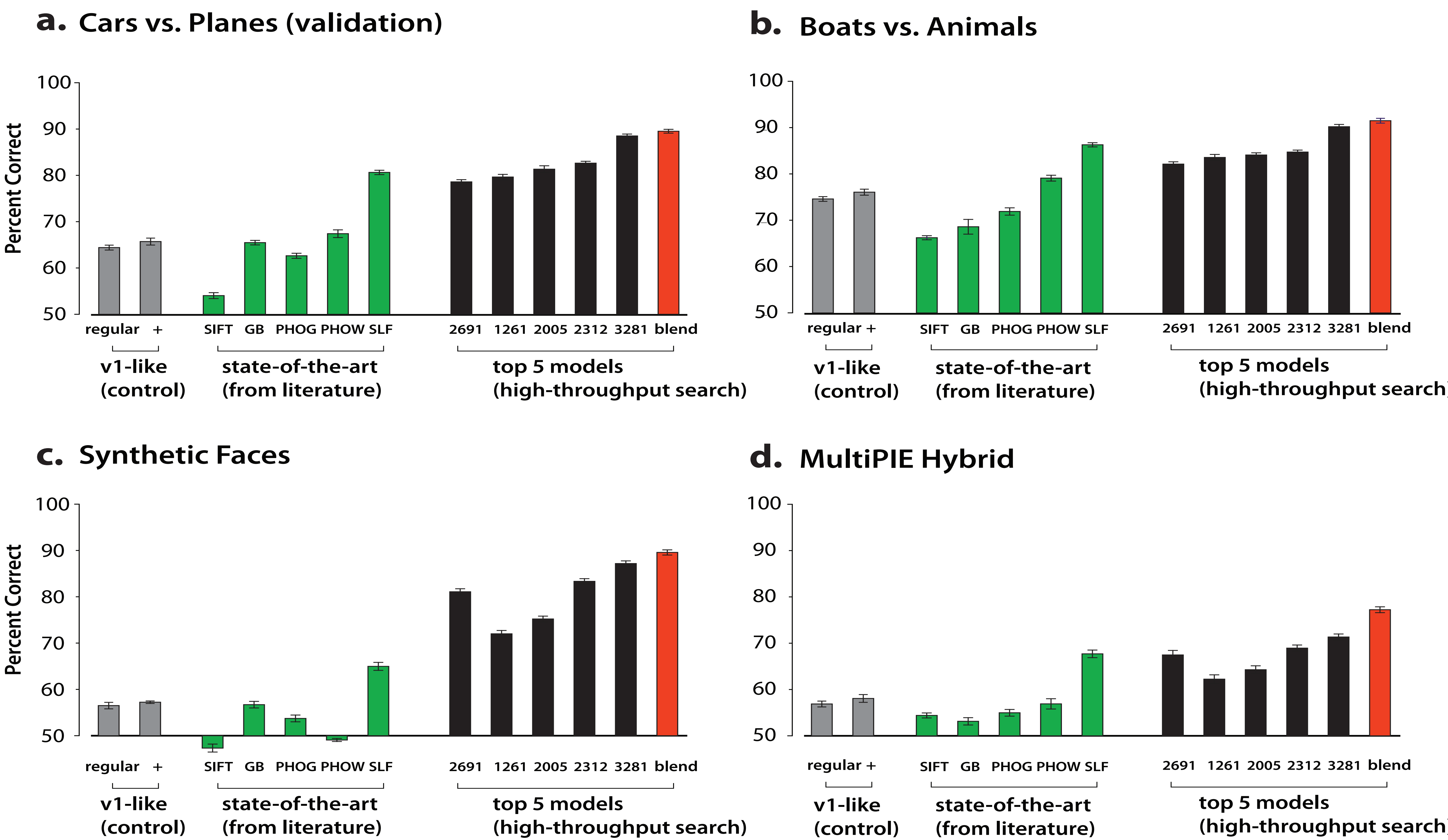
- **Optimize "Coverage"**
  (filters span the range of observed inputs)
- **Privilege movement of filters in certain directions using temporal information**
- **Expand dimensionality greatly and then scale back as layers progress**

State-of-the-art performance



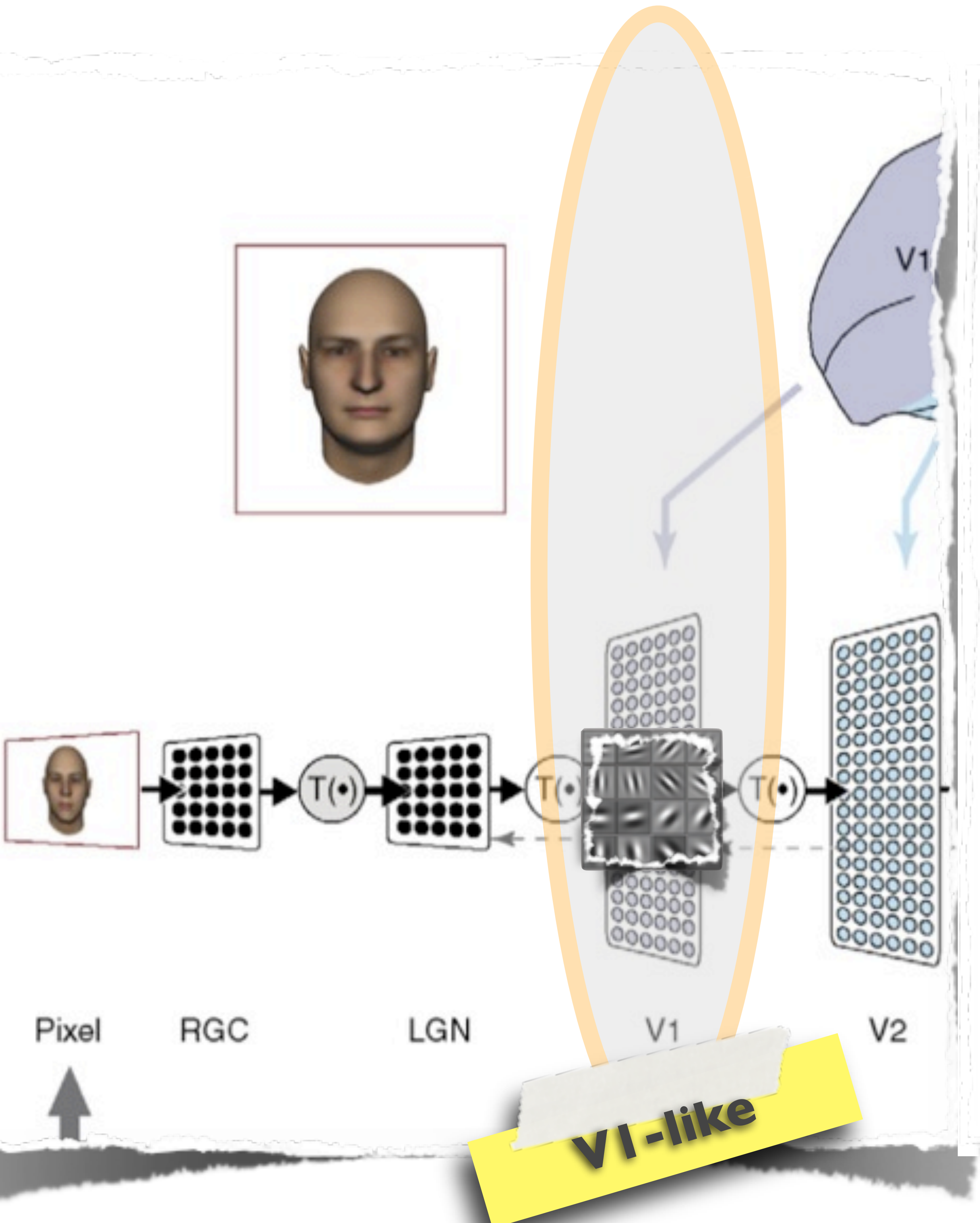
Pinto, DiCarlo, Cox (in review)

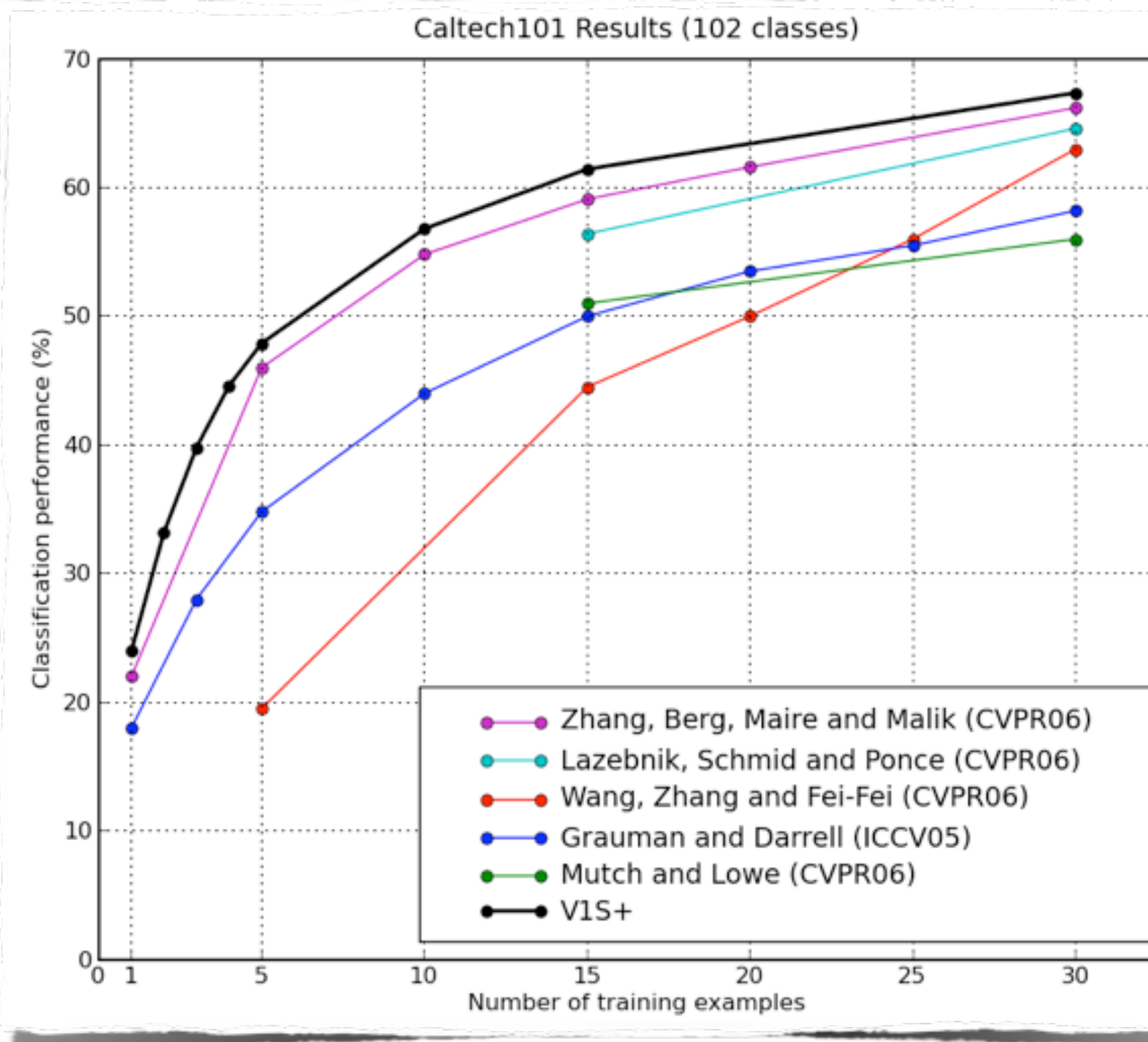
State-of-the-art performance



Pinto, DiCarlo, Cox (in review)

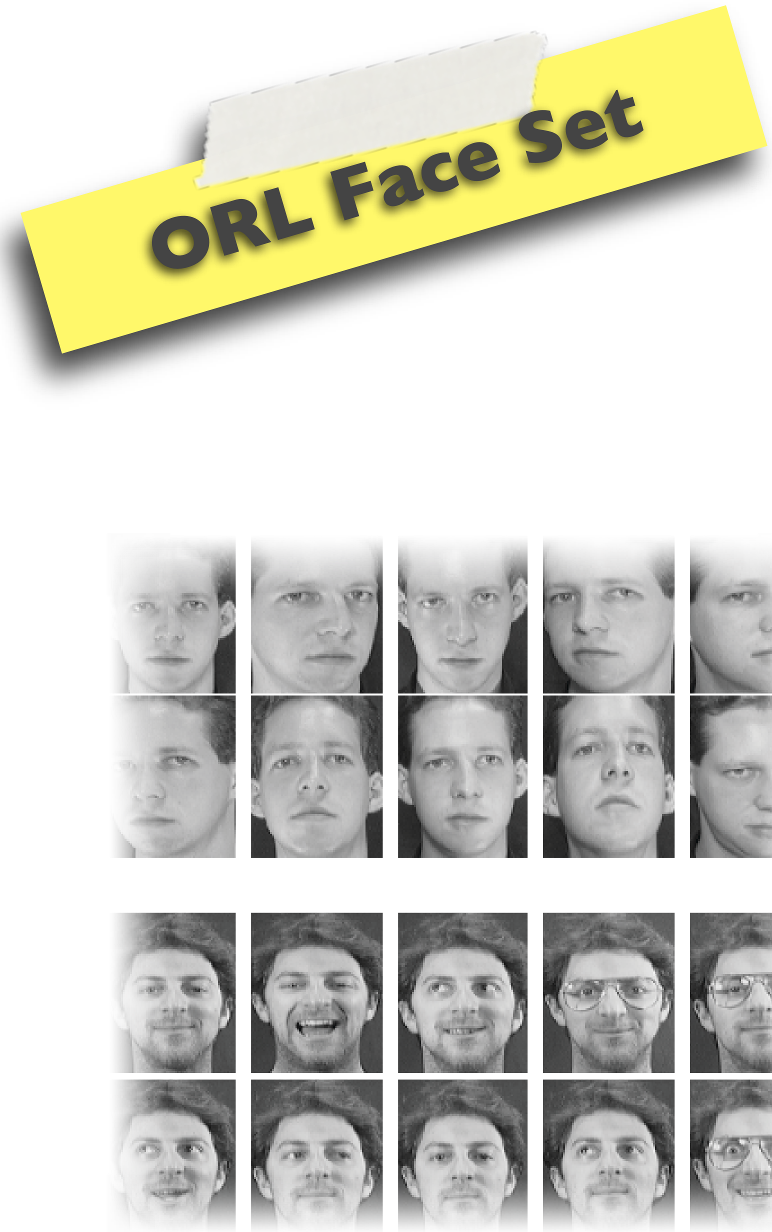
State-of-the-art performance

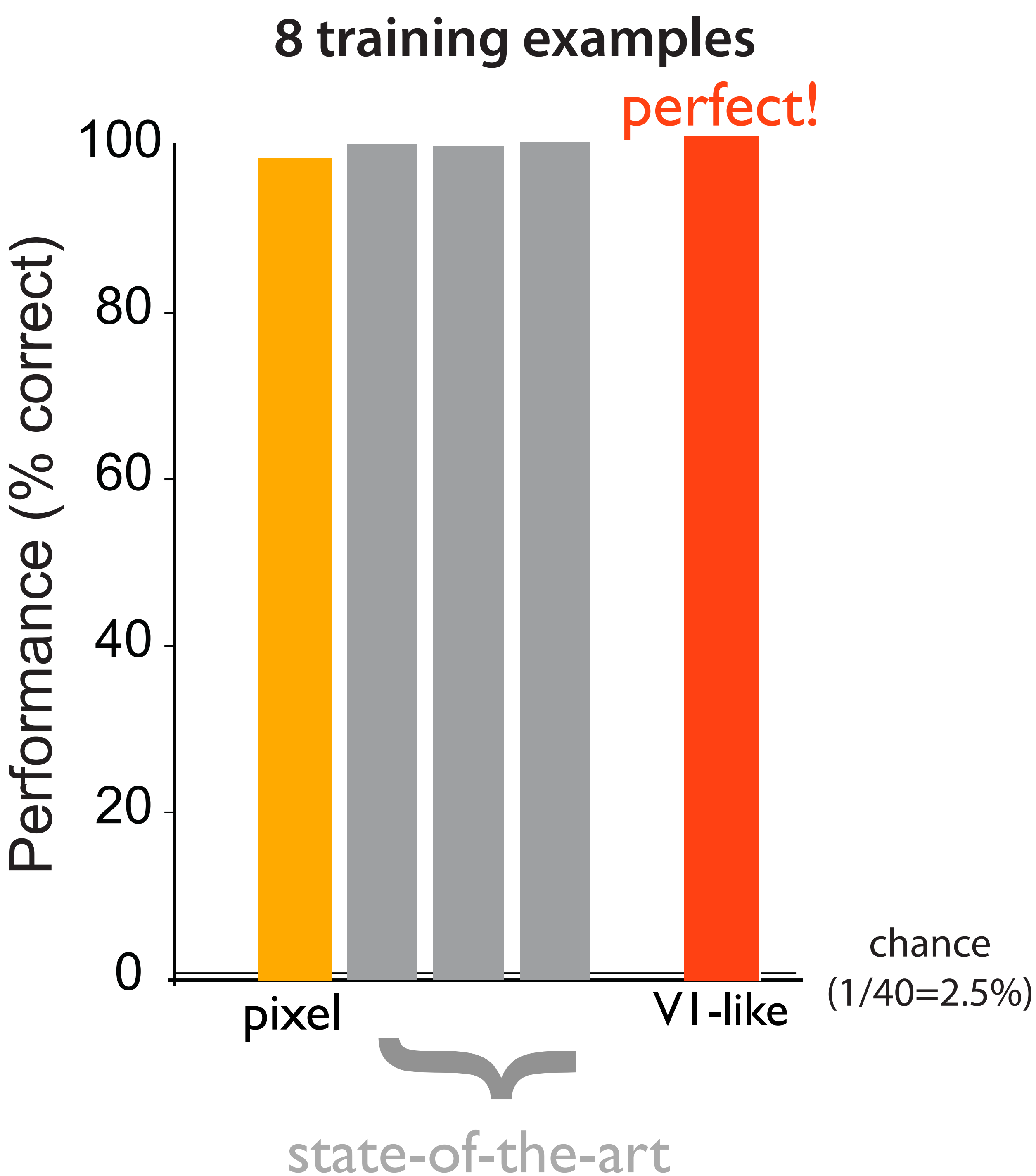




Pinto, Cox, DiCarlo PLoS08

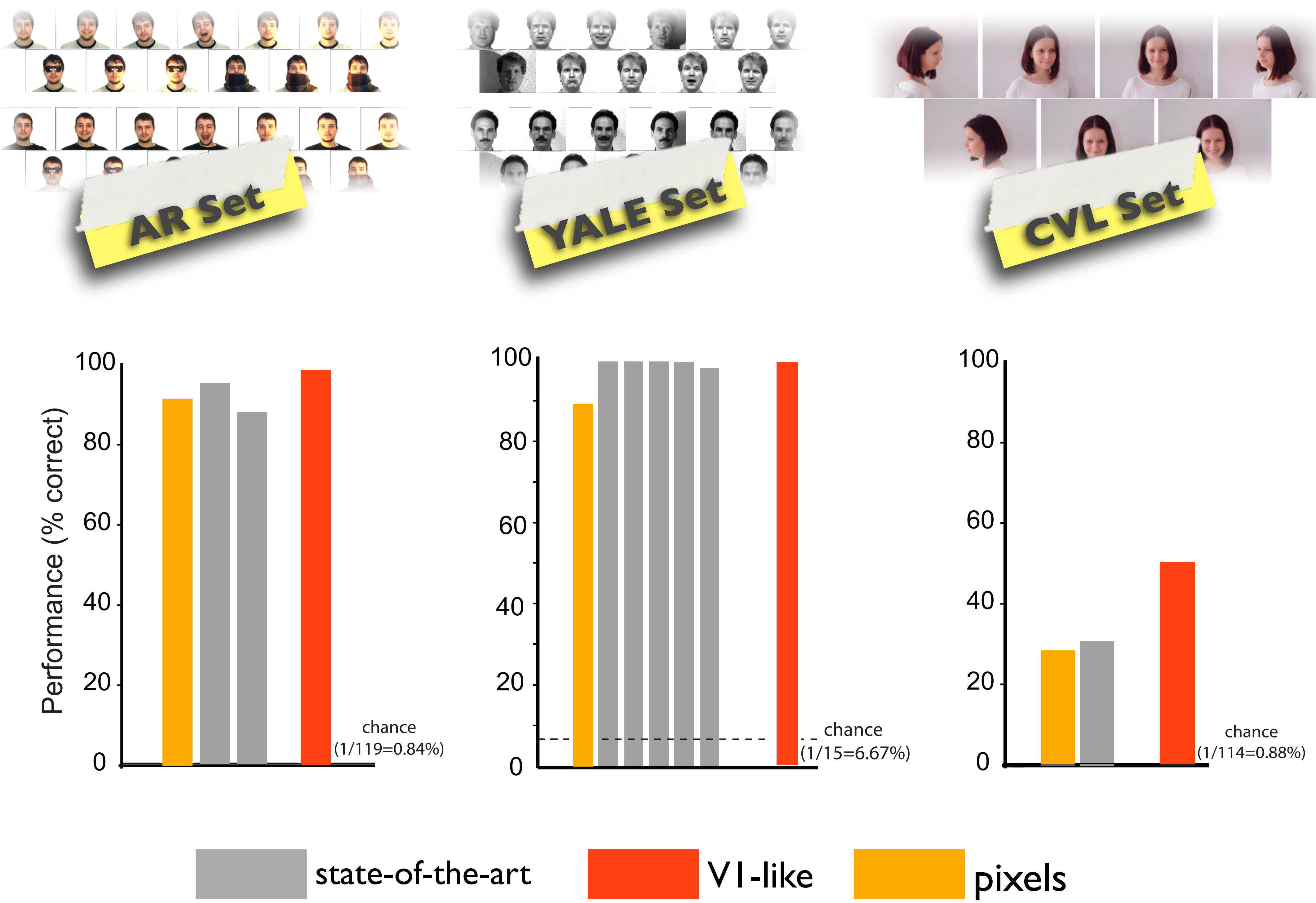
State-of-the-art performance





Pinto, DiCarlo, Cox ECCV08

State-of-the-art performance



Pinto, DiCarlo, Cox ECCV08

State-of-the-art performance

LFW Face Set

Reference	Methods	Performance
Huang08 [6]	Nowak [8]	73.93%±0.49
	MERL	70.52%±0.60
	Nowak+MERL	76.18%±0.58
Wolf08 [17]	descriptor-based	70.62%±0.57
	one-shot-learning*	76.53%±0.54
	hybrid*	78.47%±0.51
This paper	Pixels/MKL	68.22%±0.41
	V1-like/MKL	**79.35%±0.55**

Table 3. Average performance comparison with the current state-of-the-art on LFW. *note that the "one-shot-learning" and "hybrid" methods from [17] can't directly be compared to ours as they exploit the fact that individuals in the training and testing sets are mutually exclusive (i.e. using this property, you can build a powerful one-shot-learning classifier knowing that each test example is *different* from all the training examples, see [17] for more details. Our decision not to use such techniques effectively handicaps our results relative to reports that use them).

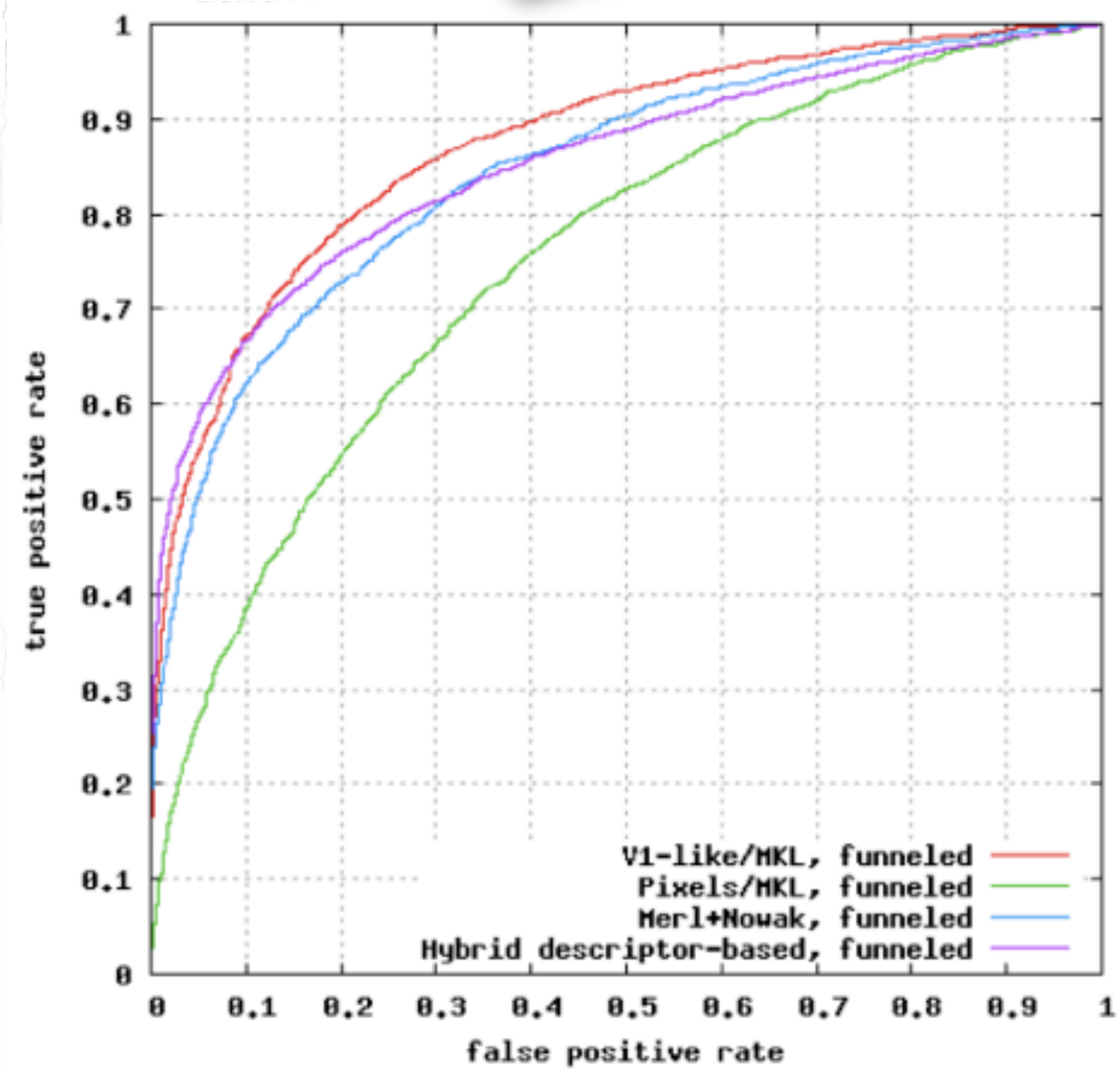


Figure 1. ROC curve comparison with the current state-of-the ar
on LFW. These curves were generated using the standard proce
dure described in [24].

Pinto, DiCarlo, Cox CVPR09

Acknowledgements

Jim DiCarlo

DiCarlo Lab @ MIT

David Cox

The Visual Neuroscience Group
@ The Rowland Institute at Harvard

Acknowledgements

NVIDIA®


"Thank You"
COME
AGAIN
and bring
a Friend!
VOX

Back Pocket Slides

slide by David Cox