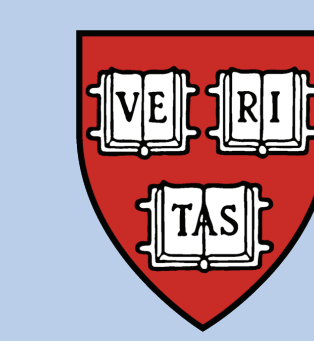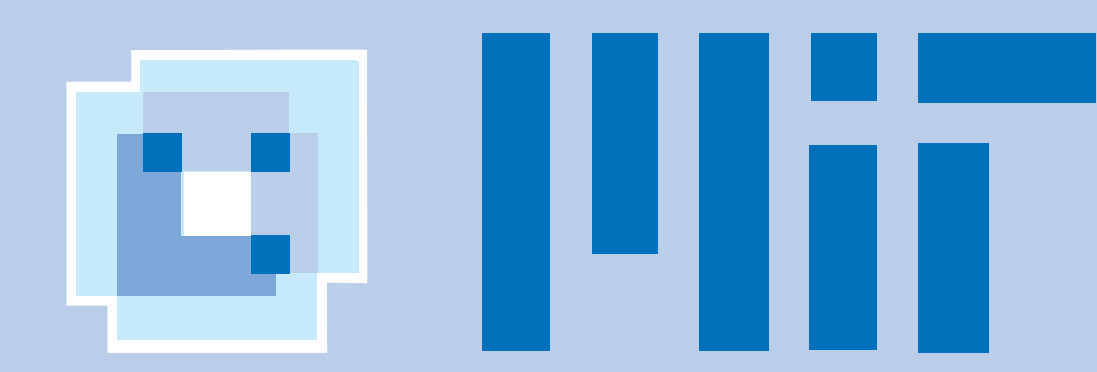# A High-Throughput Screening Approach to Discovering Good Forms of Biologically-Inspired Visual Representation

Nicolas Pinto[1], David Cox[2] and James DiCarlo[1]

1. The McGovern Institute for Brain Research and Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology
2. The Rowland Institute at Harvard

The Rowland Institute at Harvard
HARVARD UNIVERSITY

Take home message:

GPU Metaprogramming dramatically accelerates the discovery of bio-inspired vision models that beat state-of-the-art computer vision systems

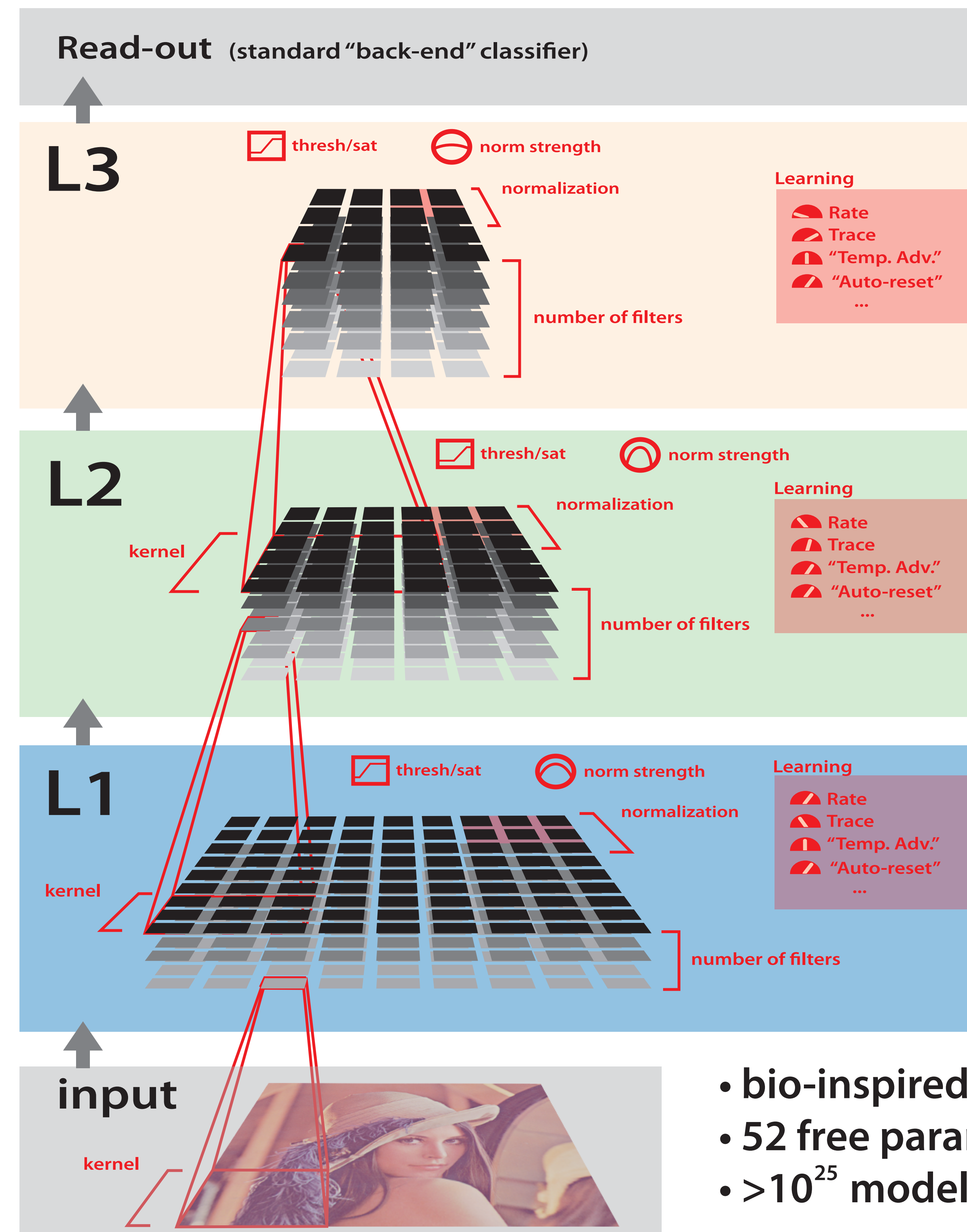| Hardware	CPUs			GPUs			
Manufacturer	Intel	Intel	Intel	NVIDIA	Sony, IBM, Toshiba	NVIDIA	NVIDIA
Model	Q9450	Q9450	Q9450	7900 GTX	PlayStation 3	8800 GTX	GTX 280
# cores used	1	4	4	4x96	2+6	4x128	4x240
Implementation	MATLAB	MATLAB	SSE2	Cg	Cell SDK	CUDA	CUDA
Year	2008	2008	2008	2006	2007	2007	2008
Performance / Cost							
Full System Cost (approx.)	$1,500**	$2,700**	$1,000	$3,000*	$400	$3,000*	$3,000*
Relative Speedup	1x	4x	80x	544x	222x	1544x	2712x
Relative Perf. / $	1x	2x	120x	272x	833x	772x	1356x

1 Abstract

While many models of biological object recognition share a common set of "broad-stroke" properties, the performance of any one model depends strongly on the choice of parameters in a particular instantiation of that model - e.g. the number of units per layer, the size of pooling kernels, exponents in normalization operations, etc. Since the number of such parameters (explicit or implicit) is typically large, and the computational cost of evaluating one particular parameter set is high, the space of possible model instantiations goes largely unexplored. Thus, when a model fails to approach the abilities of biological visual systems, we are left uncertain whether this failure is because we are missing a fundamental idea, or because the correct "parts" have not been tuned correctly, assembled at sufficient scale, or provided with enough training.

Here, we present a high-throughput approach to the exploration of such parameter sets, leveraging recent advances in stream processing hardware (high-end NVIDIA graphic cards and the PlayStation 3's IBM Cell Processor). In analogy to high-throughput screening approaches in molecular biology and genetics, we explored thousands of potential network architectures and parameter instantiations, screening those that show promising object recognition performance for further analysis. We show that this approach can yield significant, reproducible gains in performance in across an array of basic object recognition tasks, consistently outperforming a variety of state-of-the-art vision systems from the literature.

As the scale of available computational power continues to expand, we argue that this approach has the potential to greatly accelerate progress in both artificial vision and our understanding of the computational underpinning of biological vision.
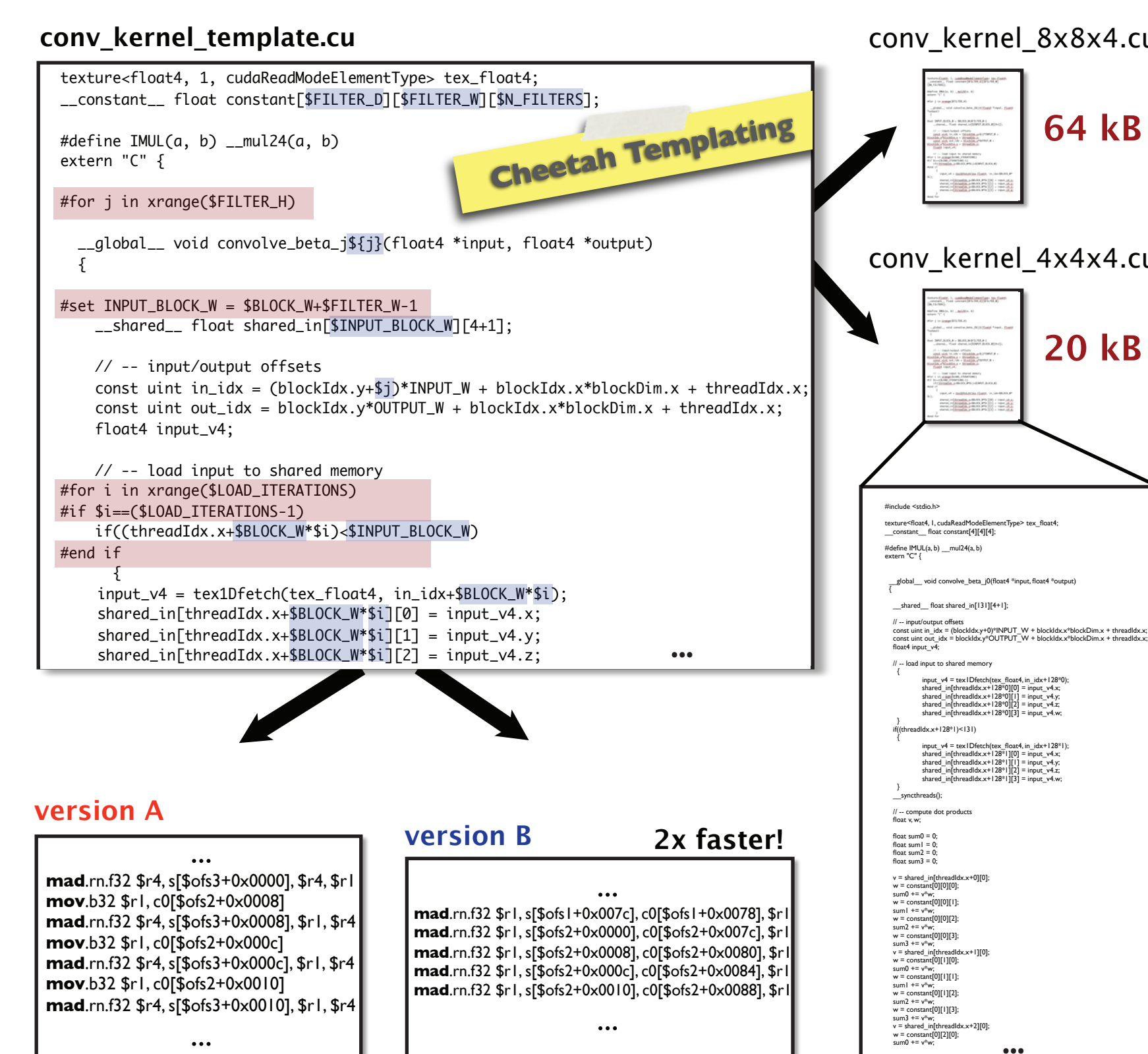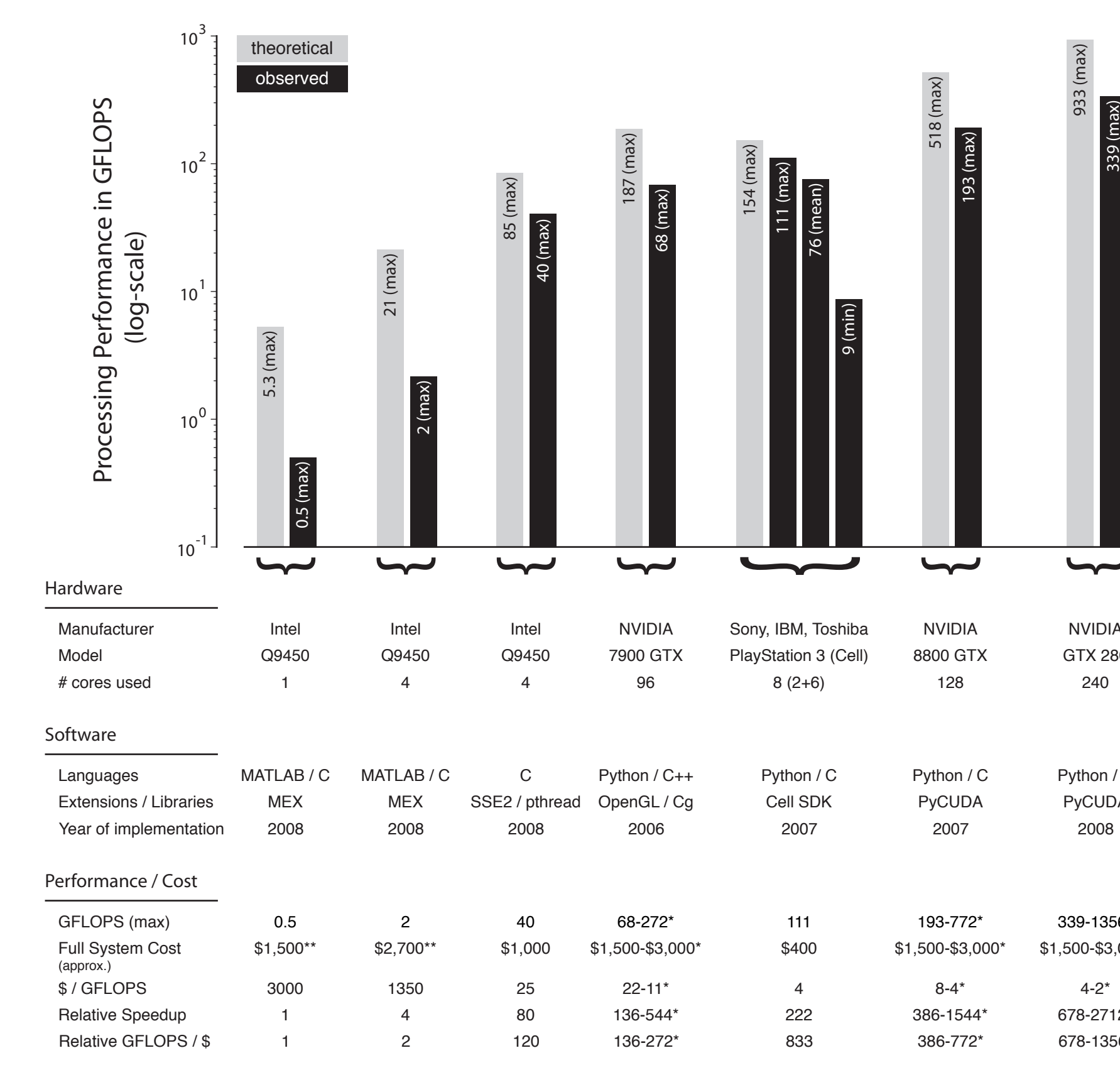
2 A Vast space of models to explore

Read-out (standard "back-end" classifier)

L3 — thresh/sat — norm strength — normalization — number of filters — Learning: Rate, Trace, "Temp. Adv.", "Auto-reset"

L2 — thresh/sat — norm strength — normalization — kernel — number of filters — Learning: Rate, Trace, "Temp. Adv.", "Auto-reset"

L1 — thresh/sat — norm strength — normalization — kernel — number of filters — Learning: Rate, Trace, "Temp. Adv.", "Auto-reset"

input — kernel

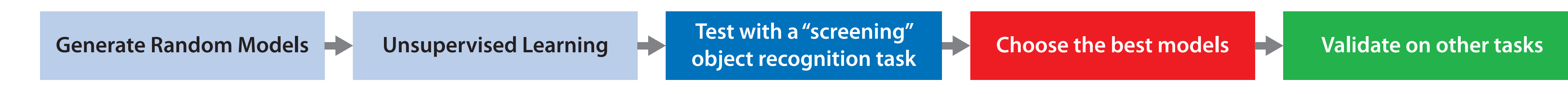- bio-inspired architecture
- 52 free parameters
- >10^{25} models

3 Metaprogramming

Recipe:
1) Use scripting (e.g. Python) and templates,
2) Instrumentalize your code,
3) Let the computer auto-tune it!

conv_kernel_template.cu

Cheetah Templating

conv_kernel_8x8x4.cu — 64 kB
conv_kernel_4x4x4.cu — 20 kB

version A | version B | 2x faster!

4 GPU performance

Processing Performance in GFLOPS (log-scale)

theoretical / observed

Hardware						
Manufacturer	Intel	Intel	Intel	NVIDIA	Sony, IBM, Toshiba	NVIDIA	NVIDIA
Model	Q9450	Q9450	Q9450	7900 GTX	PlayStation 3 (Cell)	8800 GTX	GTX 280
# cores used	1	4	4	4		128	240
Software							
Languages	MATLAB / C	MATLAB / C	C	Python / C++	Python / C	Python / C	Python / C
Extensions / Libraries	MEX	MEX	SSE2 / pthread	OpenGL / Cg	Cell SDK	PyCUDA	PyCUDA
Year of implementation	2008	2008	2008	2006	2007	2007	2008
Performance / Cost							
GFLOPS (max)	0.5	2	40	68-272*	111	193-772*	339-1356*
Full System Cost (approx.)	$1,500**	$2,700**	$1,000	$1,500-$3,000*	$400	$1,500-$3,000*	$1,500-$3,000*
$ / GFLOPS	3000	1350	25	22-11*	4	8-4*	4-2*
Relative Speedup	1	4	80	136-544*	222	386-1544*	678-2712*
Relative GFLOPS / $	1	2	120	136-272*	833	386-772*	678-1356*

5 High-throughput screening for good models

Generate Random Models → Unsupervised Learning → Test with a "screening" object recognition task → Choose the best models → Validate on other tasks

Unsupervised Learning — Law & Order "World"

Screen Distribution — N=2500 — top 5 models — number of models vs Percent Correct

Best Models — Cars vs. Planes — top 5 models (screen) — different initial weights — different videos

6 Result

Validation

Cars vs. Planes (validation) | Boats vs. Animals | Synthetic Faces | MultiPIE Hybrid

v1-like (control) | state-of-the-art (from literature) | top 5 models (high-throughput search)
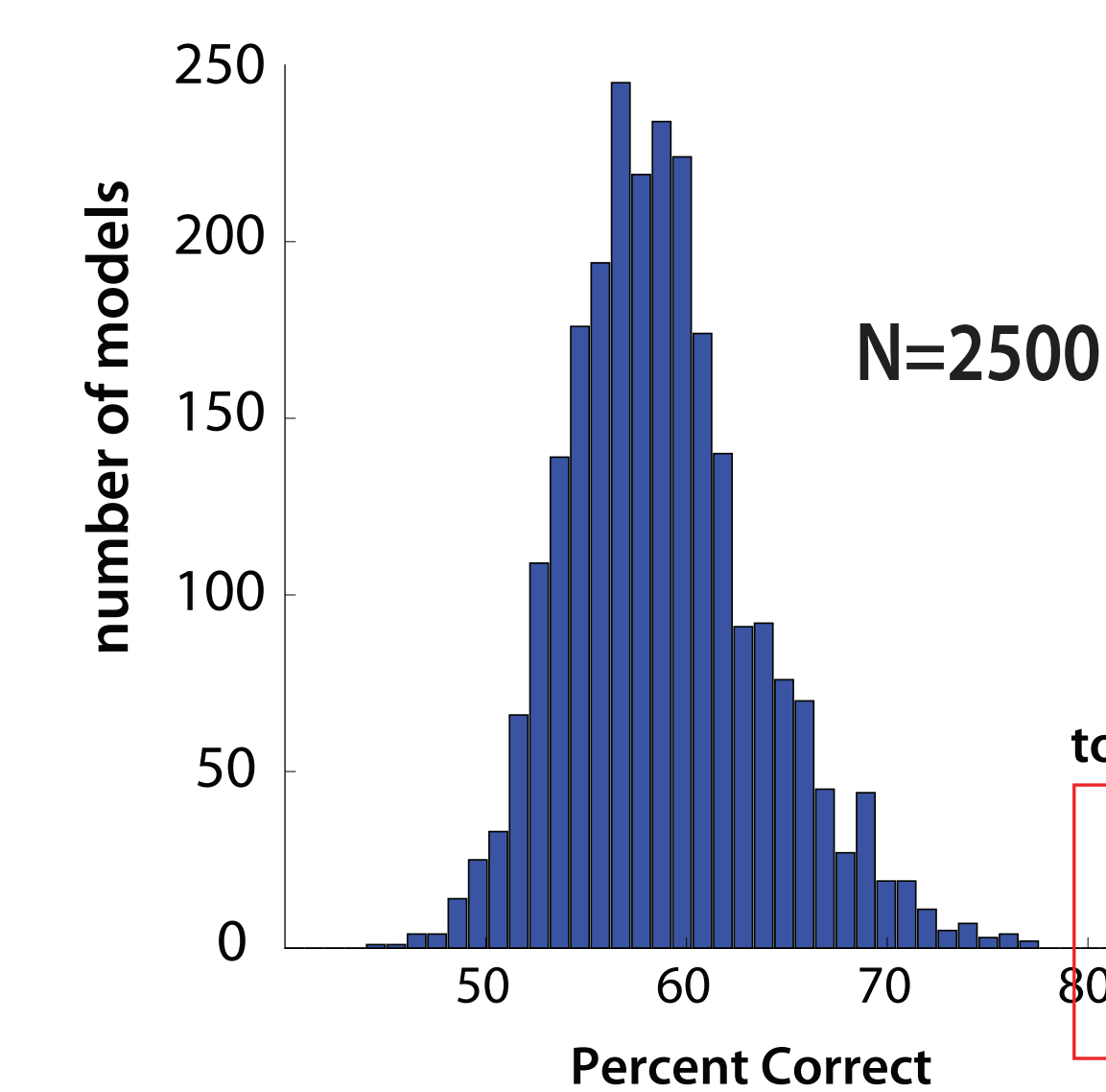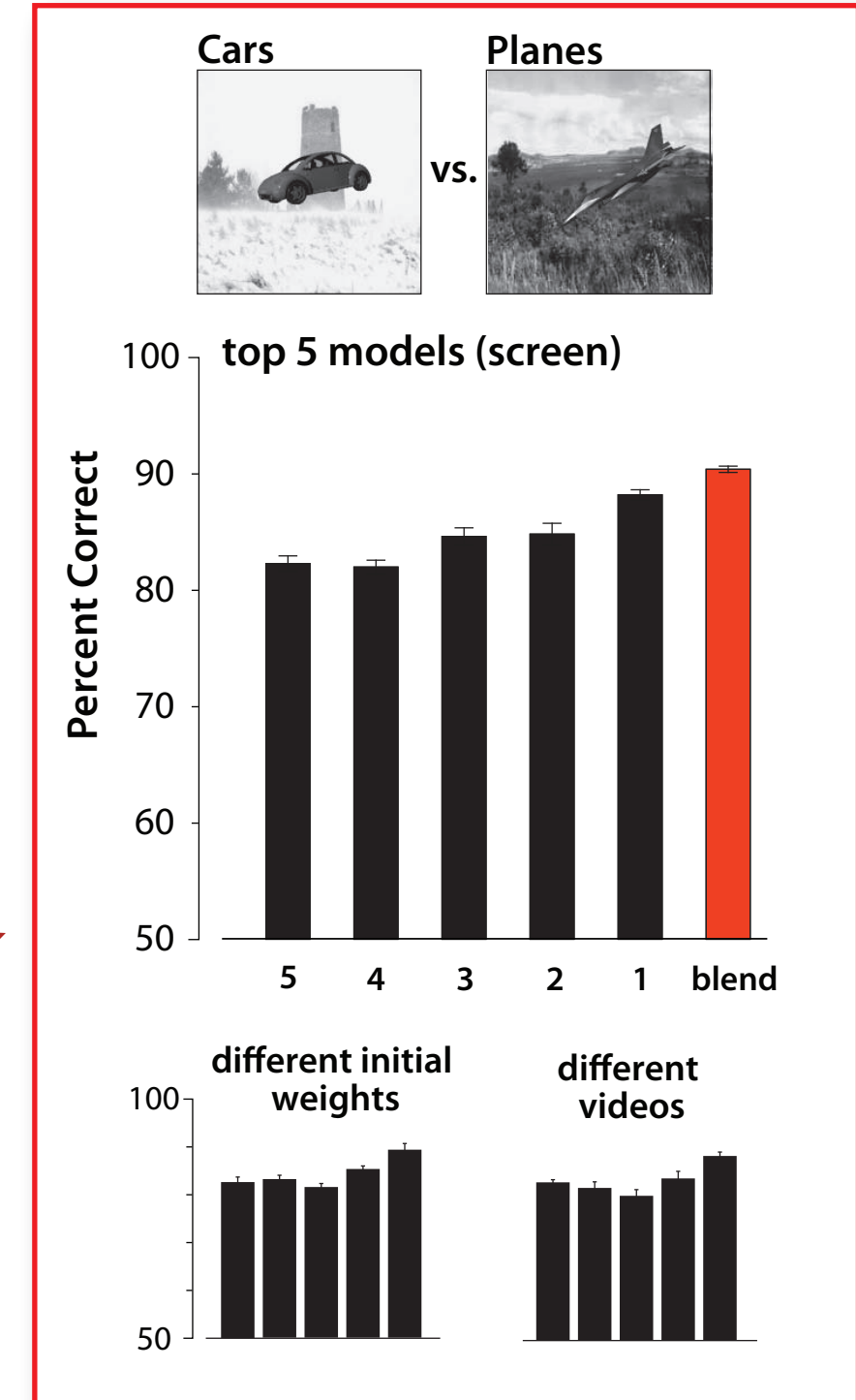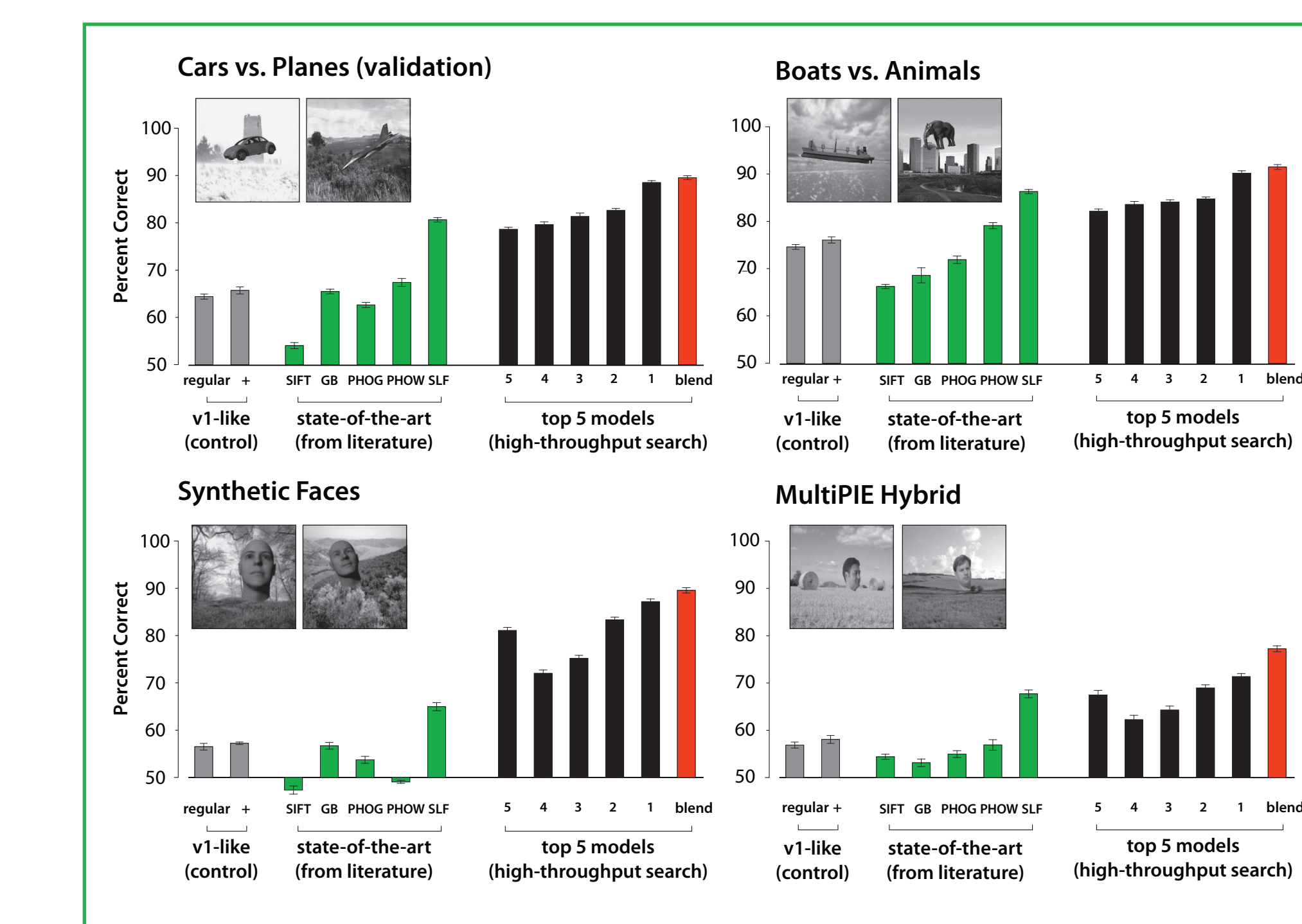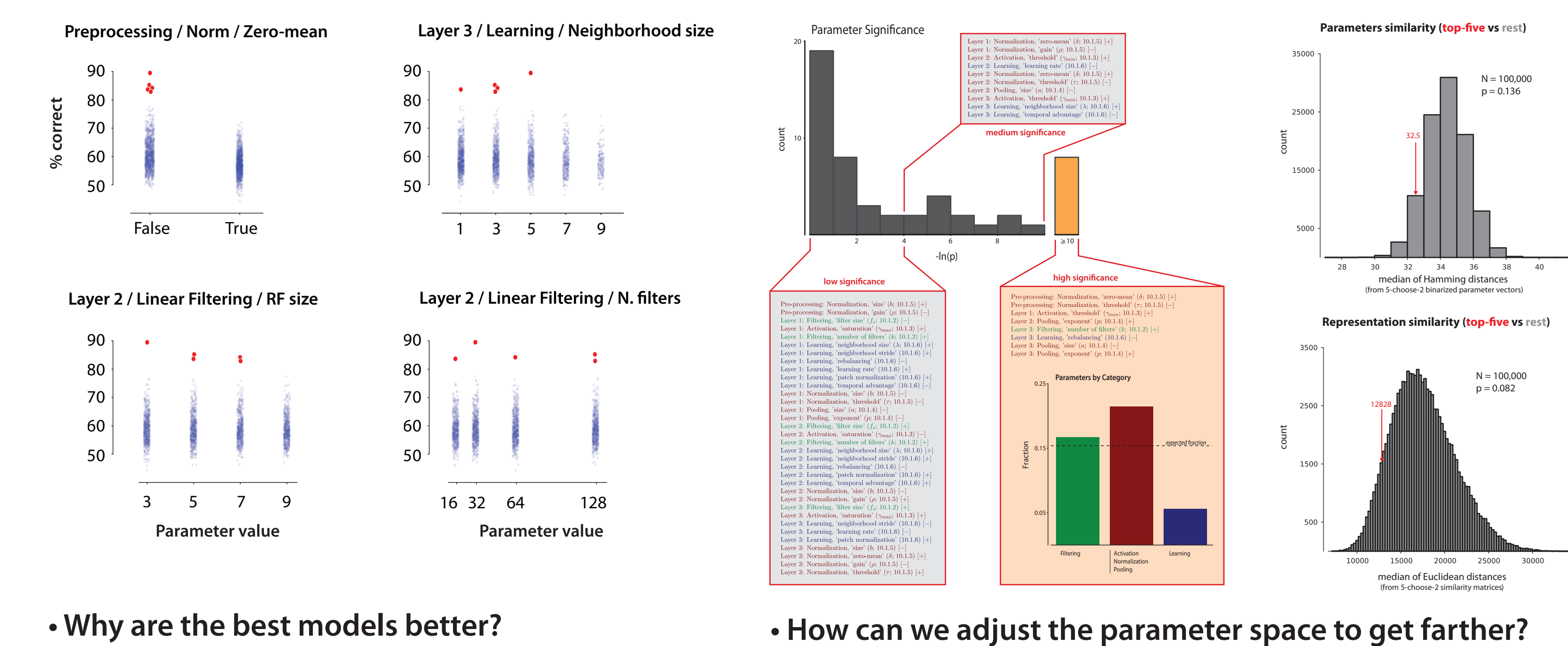
We discovered models that beat state-of-the-art computer vision (object and face recognition)

7 Analysis: Why are these models good?

Preprocessing / Norm / Zero-mean — % correct vs False / True

Layer 3 / Learning / Neighborhood size — 1 3 5 7 9

Parameter Significance

Parameters similarity (top-five vs rest) — N = 100,000 p = 0.136

Layer 2 / Linear Filtering / RF size — 3 5 7 9

Layer 2 / Linear Filtering / N. filters — 16 32 64 128

Representation similarity (top-five vs rest) — N = 100,000 p = 0.682

- Why are the best models better?
- How can we adjust the parameter space to get farther?