CS 598sm

Probabilistic & **A**pproximate **C**omputing

http://misailo.web.engr.Illinois.edu/courses/cs598

Probability

"the chance that something will happen"

Approximation

"an amount or figure that is almost correct and is not intended to be exact"

* Merriam-Webster Dictionary

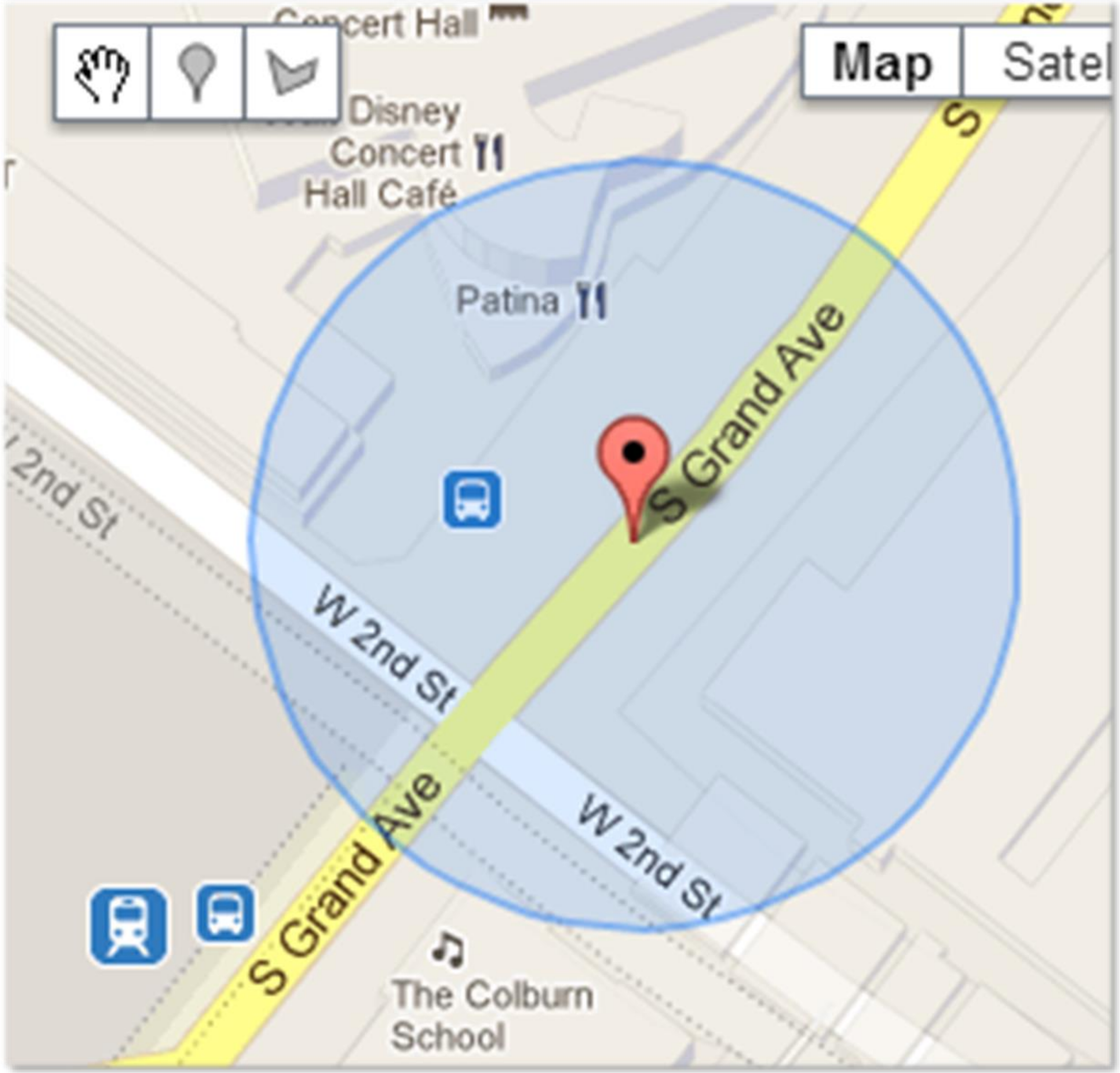
Uncertainty

“something that is doubtful or unknown“

Probability quantitatively represents uncertainty (captures the degree of belief)

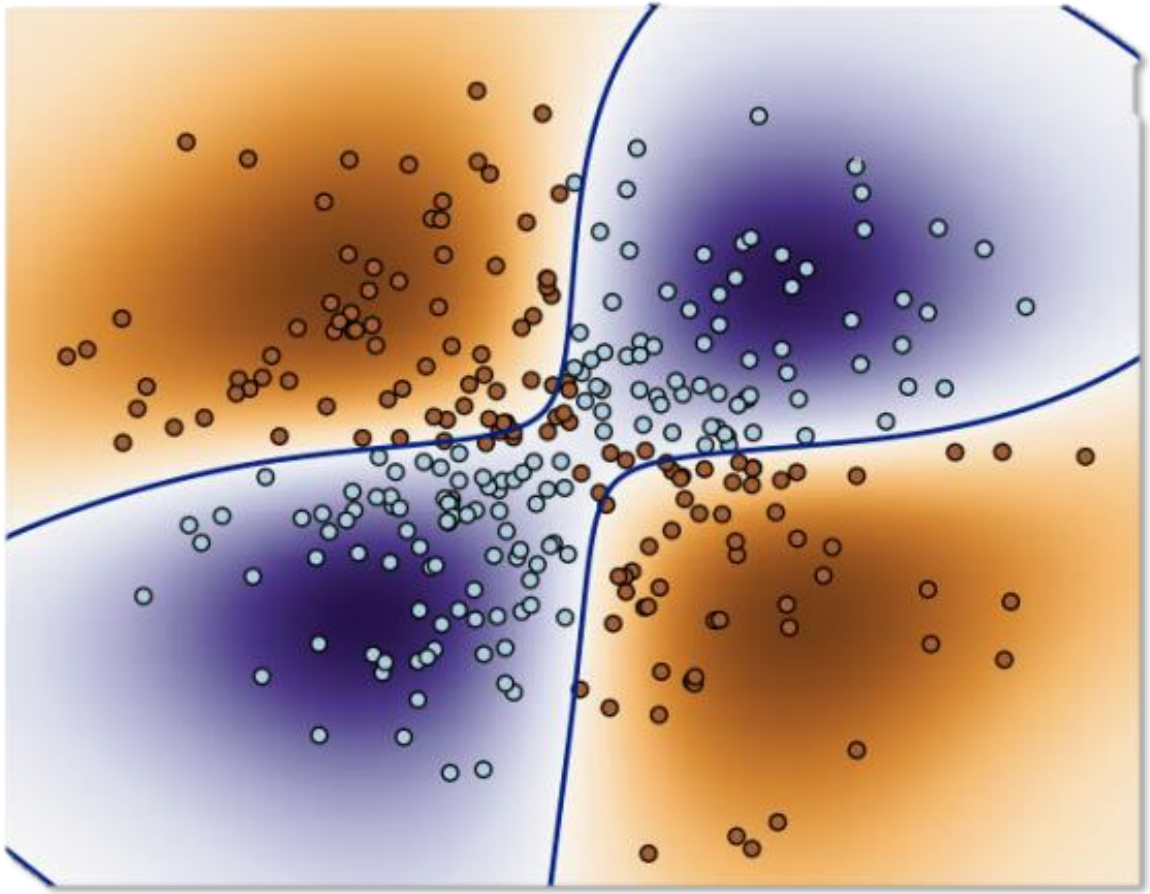
Approximation efficiently copes with uncertainty (ignores it or tractably computes with it)

Location Tracking



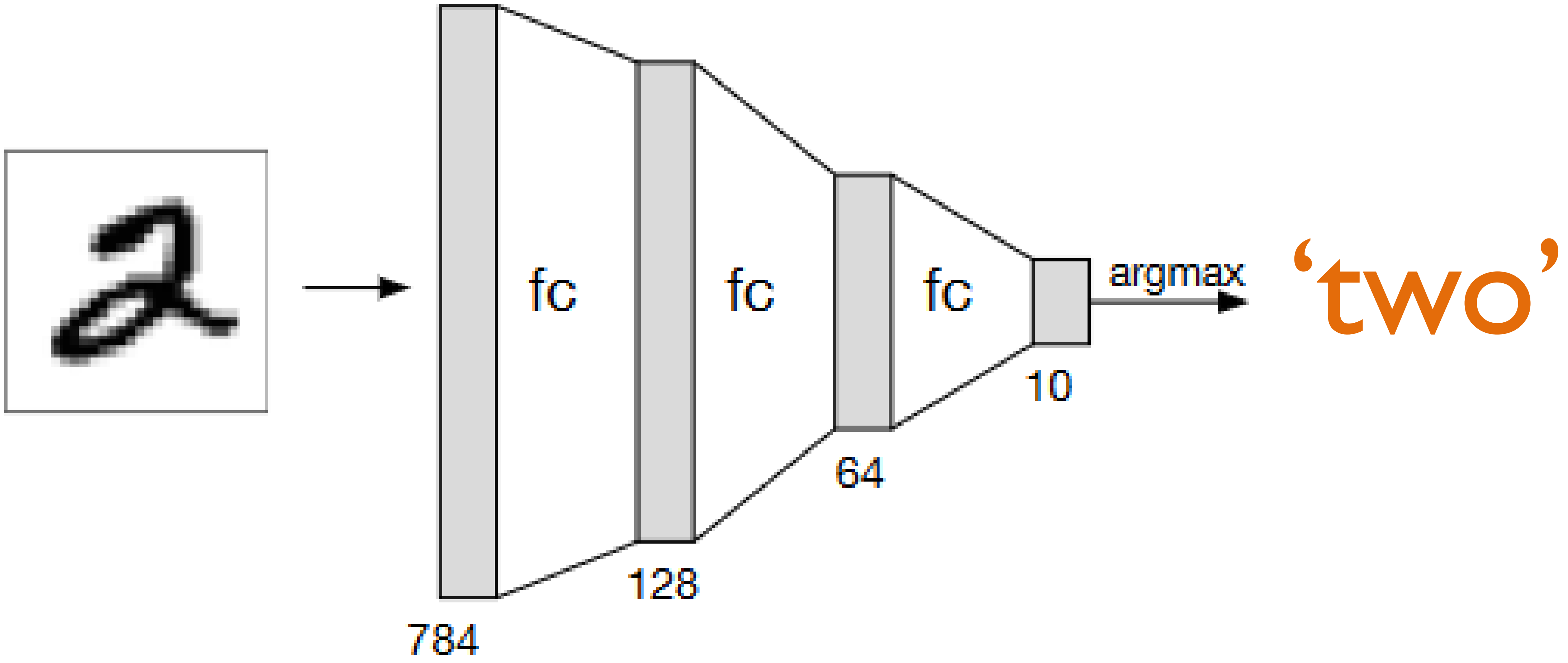
A probability distribution is hiding on the screen. Find it!

Data Classification



Are all red points in the
same cluster?

Pattern Recognition



Multimedia

Machine Vision

Extended Reality

Autonomus Systems

. . .

Some Common Traits

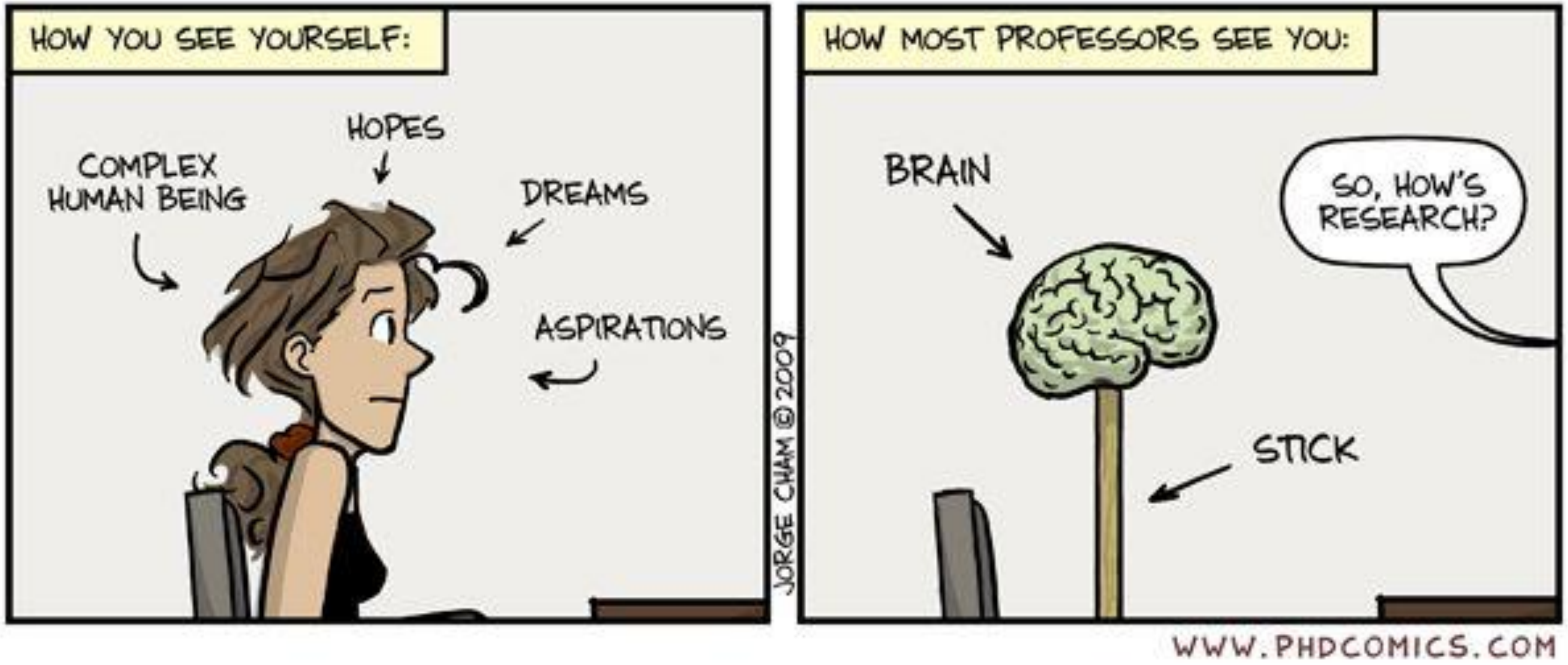
Noisy Data coming from sensors

Redundancy in data and computation

Models that effectively compress such redundancy

Environments that we don’t fully understand

For Modeling, Context is Important

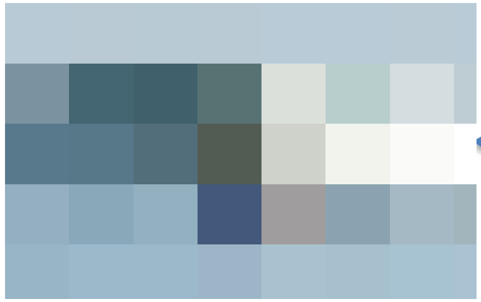


“All models are wrong,
but some are useful!”

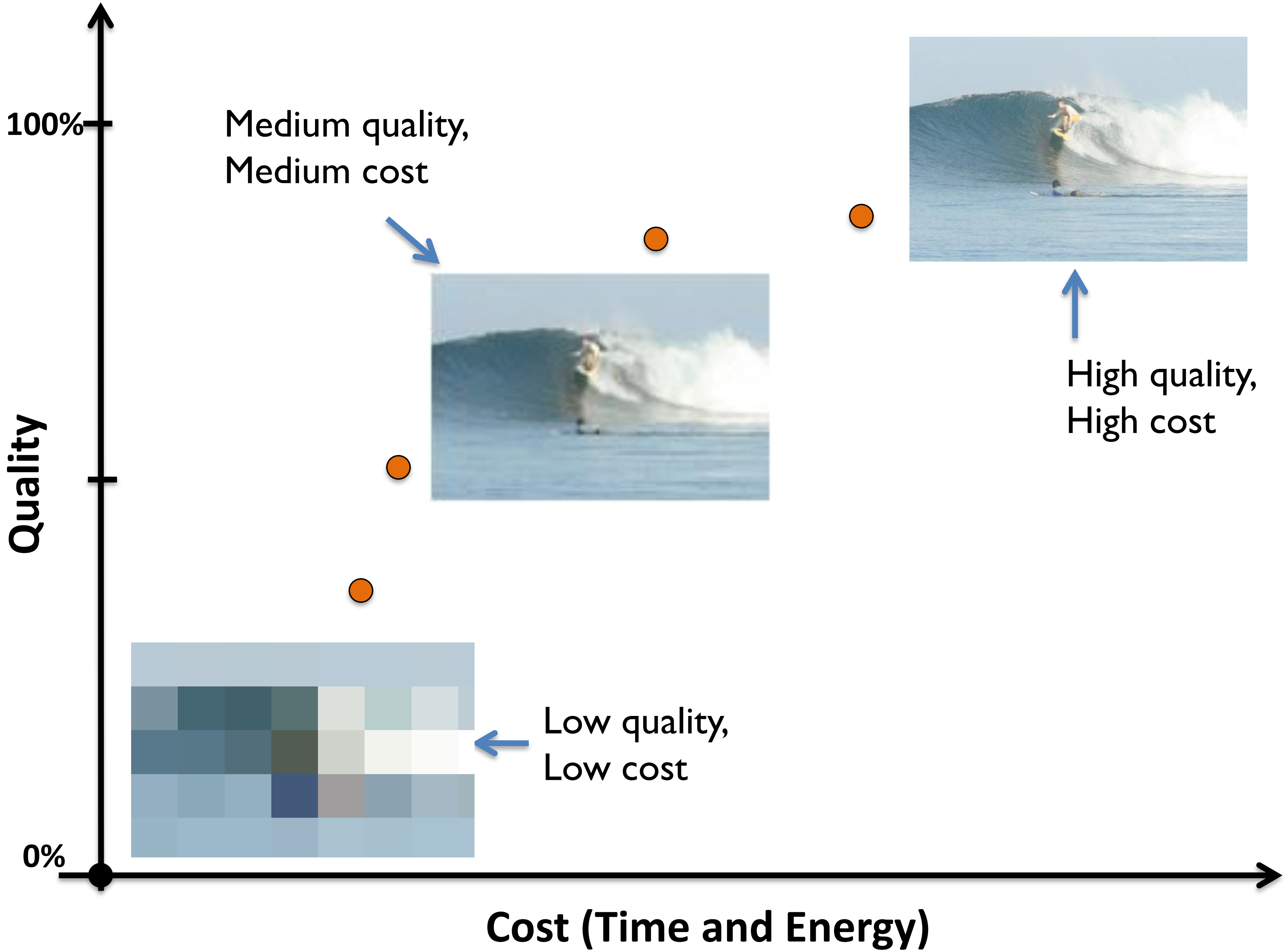
George E. P. Box

Medium quality,
Medium cost

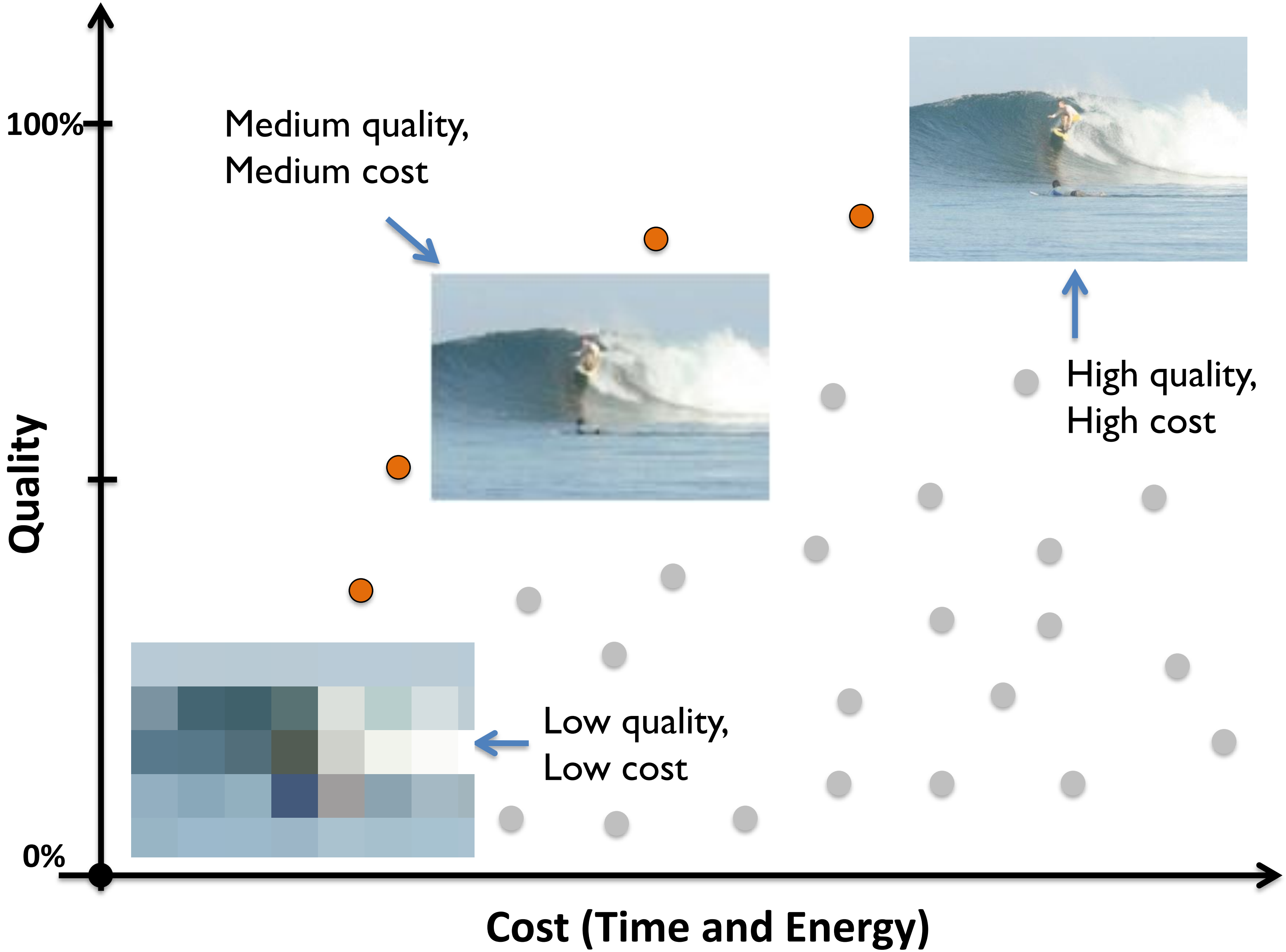
High quality,
High cost



Low quality,
Low cost



100%
0%
Quality
Medium quality,
Medium cost
High quality,
High cost
Low quality,
Low cost
Cost (Time and Energy)

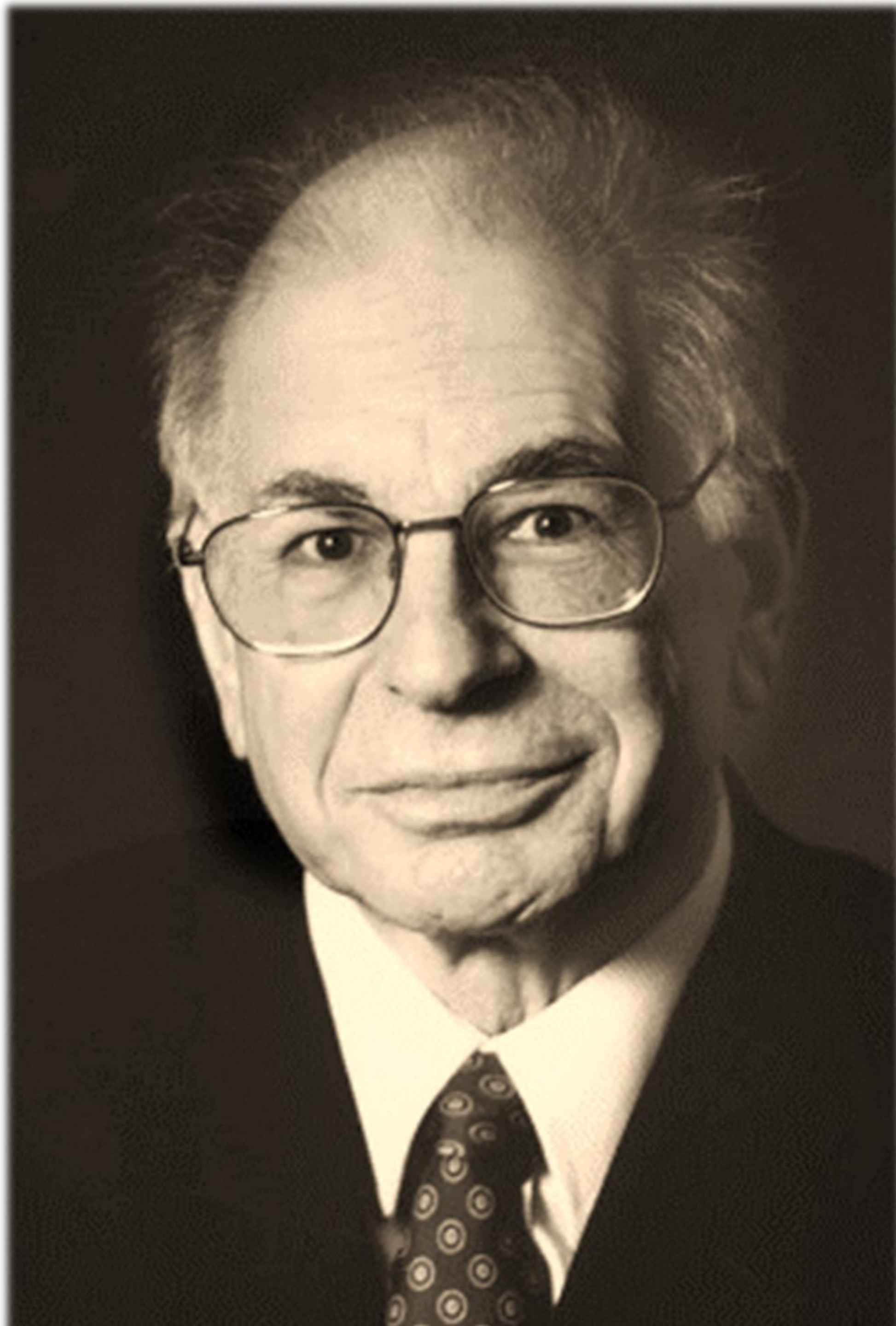

Quality
100%
0%
Medium quality,
Medium cost
High quality,
High cost
Low quality,
Low cost
Cost (Time and Energy)

Two ways the brain forms thoughts:

System 1: fast, instinctive, and emotional

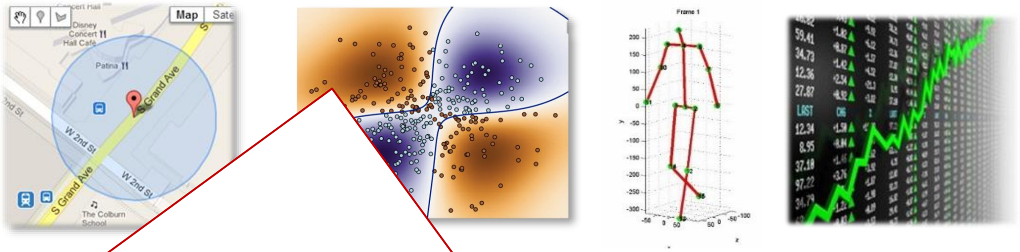
System 2: slow, conscious, and logical

Daniel Kahneman



Classical Computing Stack

Applications



Languages and Compilers

...he program code to ...*al* machine code

Syst...
S...

Dynamically schedules program to minimize time

...ware Architectures

Execute the machine code ***in unambiguous manner***

Perfection!

Observation (1965):

“The number of gates on a chip doubles every 24 months”

Gordon Moore

Dennard Scaling (1974)*:

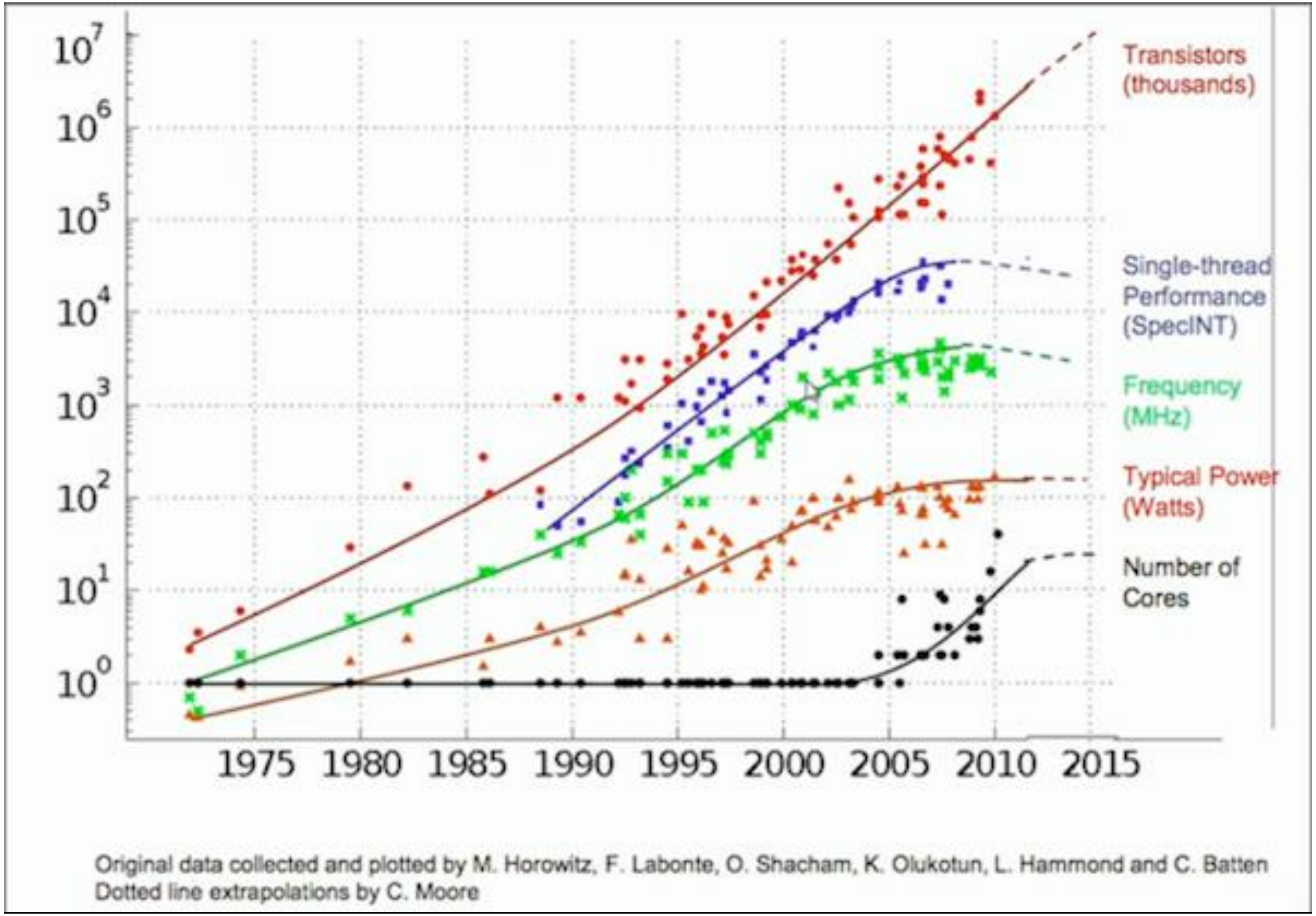
Voltage and current should be proportional to the linear dimensions of a transistor.

> Thus, as transistors shrink, so do necessary voltage and current.
>
> Power is proportional to the area of the transistor (while the transistor is still reliable)

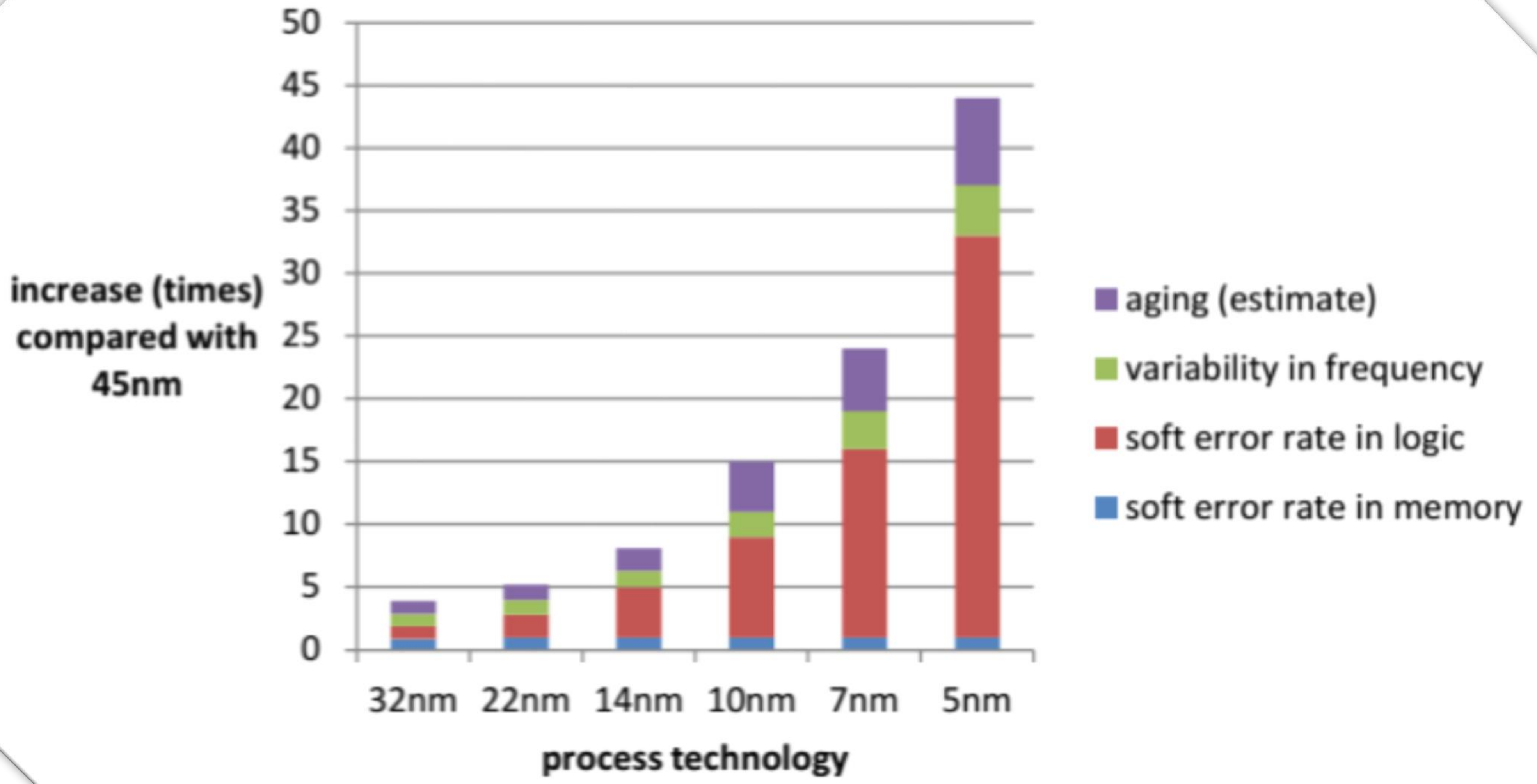
Robert Dennard

* Robert Dennard (from W. Gropp; UIUC CS 598WG)

End of Dennard Scaling (2005-now)



How Much Can We Shrink?

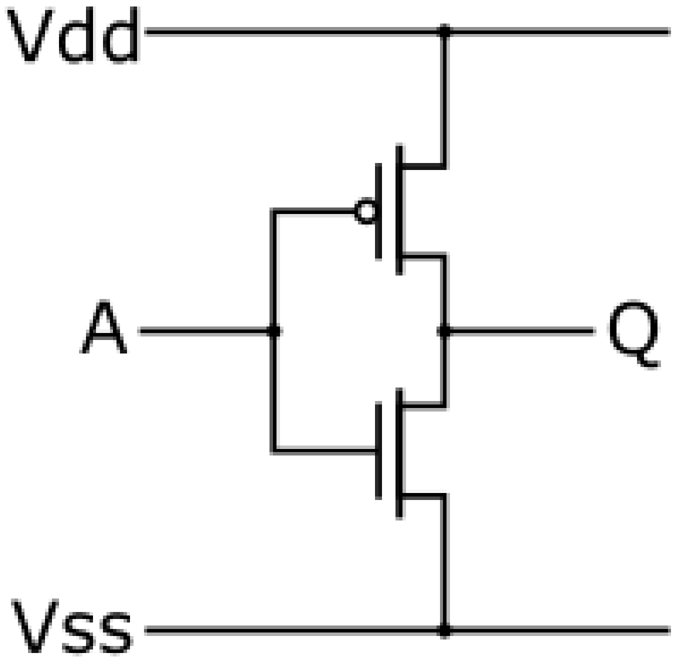


Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, Feb 2012.

GET READY FOR OUR FIRST THOUGH EXPERIMENT…

CMOS Transistors

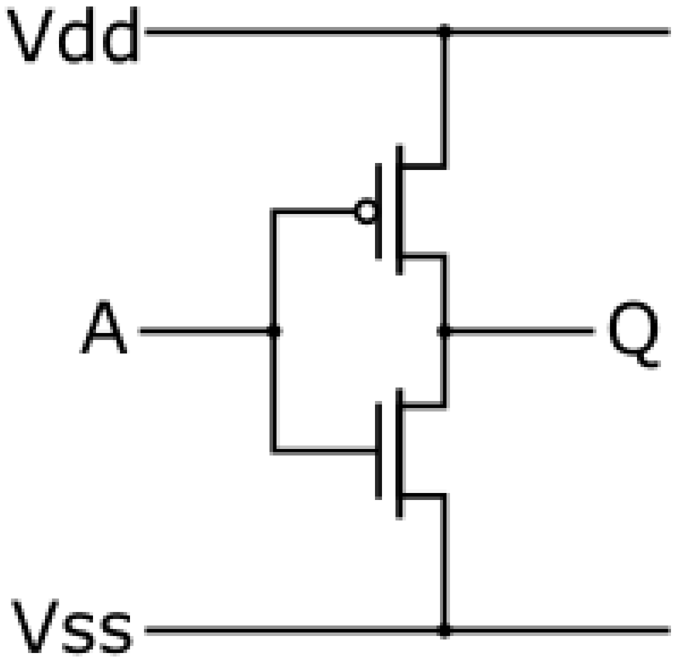
Simple Invertor



A	Q
0	1
1	0

Probabilistic CMOS Transistors

Simple Invertor



A	Q	
0	1	P>0.99
0	0	P<0.01
1	0	P>0.99
1	1	P<0.01

See [Probabilistic CMOS Technology: A Survey and Future Directions (VLSI 2006)]()

Breaking Digital Abstraction

Approximate and Unreliable Hardware

- Process Variation and Aging

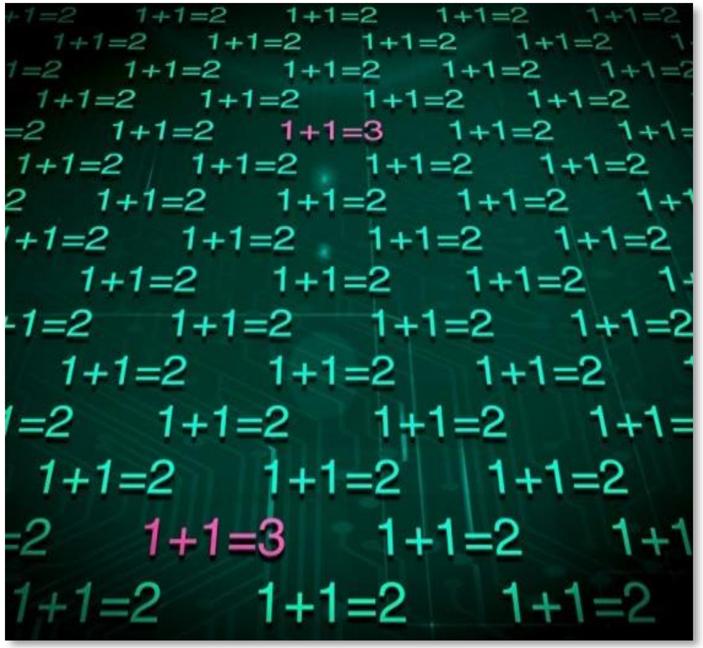
  Tiwari et al., ISCA ’07; Mohapatra et al., ISLPED ’09; Rahimi et al., DAC ’13; Karpuzcu et al., DNS ‘12; Namaki-Shoushtari et al., CODES+ISSS ‘13…

- Timing Errors & Soft Faults

  Ernst et al., MICRO ‘03; Sarangi et al., MICRO ‘08; Kruijf et al., ISCA ’10; Leem et al. DATE ’10; Sampson et al., PLDI ’11; Ku He et al., DATE ’11; Esmaeilzadeh et al., ASPLOS ’12 …

- Inexact Circuits & Storage

  Palem et al., SSDM ’04; Narayanan et al., DATE ’10; Chippa et al., DAC ’10; Liu et al., ASPLOS ’11; Esmaeilzadeh et al., ASPLOS ’12; Esmaeilzadeh et al., MICRO ’12; Sampson et al., MICRO ’13; Venkataramani et al., MICRO ’13; Venkataramani et al., ISLPED ‘14; Kozhikkottu et al., ISLPED ’14; Miao et al., ICCAD ‘14; St Amant et al., ISCA ‘14; Düben et al., Phil. Trans. R. Soc. ‘14 …



Programming Language Support

Specifications

```
<0.99*R(x)> f(x) {...}
```
Rely (OOPLSA'13)

```
@approx int x = …
```
EnerJ (PLDI'11)

```
x := Gaussian (0,1);
```
Kozen (JCSS'81)

Verification

```
assert Pr[ Error ] < 0.001

assert Expected[ Error ] = 0
```

Key Concept **Probability Theory**

Compilers Pick Approximations

Loop Perforation (ICSE'10, FSE'11)

```
for (i = 0; i < n; i++) { … }
```

```
for (i = 0; i < n; i += 2) { … }
```

Compilers Pick Approximations

Loop Perforation (ICSE'10, FSE'11)

```
for (i = 0; i < n; i++) { … }
```

```
for (i = 0; i < n; i += 2) { … }
```

It's not going to work!

Program will produce incorrect result!

Original

Perforated

Encoded 3x faster

Original

Perforated

Original

Perforated

Any pixel difference

Original

Perforated

> 1% pixel difference

Original

Perforated

> 5% pixel difference

Not a correctness issue

Accuracy issue

John Von Neumann, 1952

PROBABILISTIC LOGICS AND THE SYNTHESIS OF RELIABLE
ORGANISMS FROM UNRELIABLE COMPONENTS

J. von Neumann

1. INTRODUCTION

The paper that follows is based on notes taken by Dr. R. S. Pierce on five lectures given by the author at the California Institute of Technology in January 1952. They have been revised by the author but they reflect, apart from minor changes, the lectures as they were delivered.

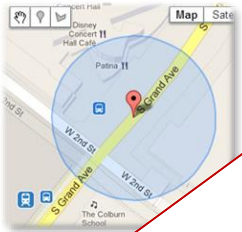
The subject-matter, as the title suggests, is the role of error in logics, or in the physical implementation of logics — in automata-synthesis. Error is viewed, therefore, not as an extraneous and misdirected or misdirecting accident, but as an essential part of the process under consideration — its importance in the synthesis of automata being fully comparable to that of the factor which is normally considered, the intended and correct logical structure.

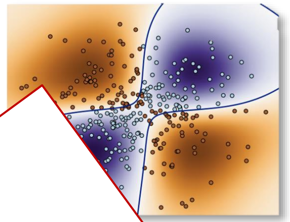
Our present treatment of error is unsatisfactory and ad hoc. It is the author's conviction, voiced over many years, that error should be treated by thermodynamical methods, and be the subject of a thermodynamical theory, as information has been, by the work of L. Szilard and C. E. Shannon [Cf. 5.2]. The present treatment falls far short of achieving this, but it assembles, it is hoped, some of the building materials, which will have to enter into the final structure.

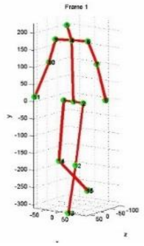
The author wants to express his thanks to K. A. Brueckner and M. Gell-Mann, then at the University of Illinois, to whose discussions in 1951 he owes some important stimuli on this subject; to Dr. R. S. Pierce at the California Institute of Technology, on whose excellent notes this exposition is based; and to the California Institute of Technology, whose invitation to deliver these lectures combined with the very warm reception by the audience, caused him to write this paper in its present form, and whose cooperation in connection with the present publication is much appreciated.

Classical Computing Stack

Applications







Languages and Compilers — ...he program code to ***...al*** machine code

Syst... S... — Dynamically schedules program to minimize time

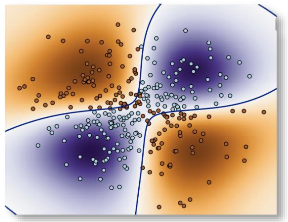
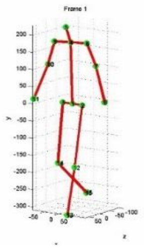
...ware Architectures — Execute the machine code ***in unambiguous manner***

Perfection!

New-Reality Computing Stack

Approximate Computing

Applications
(tolerate errors)


Languages and Compilers

Translates the program code to ***automatically approximate*** code

Systems Software

Dynamically schedules program to minimize time ***by trading accuracy***

Hardware Architectures

Execute the code ***approximately and non-deterministically***

Fault-Tolerant Computing

Goal of the Course

Embark on a journey to

rethink computing in the world

where absolute correctness is elusive

- Learn from recent papers
- Discuss new research ideas
- **Do a fun project**

PEEK PREVIEW

Tentative Lectures

Date	Topic	Presenter	Notes
8/25	Introduction ***Background Read:*** Exploiting Errors for Efficiency: A Survey from Circuits to Algorithms (CSUR 2020)	Sasa Slides	***Fun Facts:*** J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components (Automata Studies, 1956)
8/27	Approximations in Software Systems ***Background Read:*** Exploiting Errors for Efficiency: A Survey from Circuits to Algorithms (CSUR 2020)	Sasa Slides	***Fun Facts:*** Computing, Approximately -- Ravi Nair's talk (WACI@ASPLOS 2008)
9/1	Approximations: Numerical Computations ***Background Read:*** What Every Computer Scientist Should Know About Floating-Point Arithmetic (ACM 1991)	Sasa Slides	***Additional Read:*** Towards a Compiler for Reals (TOPLAS 2017) ***Software:*** Precimonious ***Fun Facts:*** How Accurate is Scientific Software? (IEEE 1994)
9/3	Approximations: Machine Learning ***Background Read:*** Neural Network Quantization Survey ***Background Read:*** A Survey of Model Compression and Acceleration for Deep Neural Networks (IEEE Signal Processing Magazine, 2020)	Sasa Slides	***Additional read:*** Tensorflow: A system for large-scale machine learning (OSDI 2016) ***Additional read:*** Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis (ACM CSUR 2019) ***Software:*** Intel Distiller
9/8	Approximations: Non-deterministic ***Background Read:*** Approximate Communication: Techniques for Reducing Communication Bottlenecks in Large-Scale Parallel Systems (CSUR 2018) ***Background Read:*** Probabilistic CMOS Technology: A Survey and Future Directions (VLSI 2006)	Sasa Slides	***Additional Read:*** EnerJ: Approximate Data Types for Safe and General Low-Power Computation (PLDI 2011) ***Additional Read:*** Unconventional Parallelization of Nonterministic Applications (ASPLOS 2018)
9/10	**Submit Your Paper Choices:** Link		
9/10	Quality-Aware Optimization Systems ***Background Read:*** Metaheuristics Book (Ch.1, Ch.2, Ch.3, Ch.7)	Sasa Slides	***Additional Read:*** Evolutionary Algorithms for Solving Multi-Objective Problems (Ch.1) ***Additional Read:*** Language and Compiler Support for Auto-Tuning Variable-Accuracy Algorithms (CGO 2011) ***Software:*** OpenTuner
9/15	Probabilistic Programming (1) ***Background Read:*** Probabilistic Programming (ICSE/FoSE 2014) ***Background Read:*** An Introduction to Probabilistic Programming	Sasa Slides	***Additional Read:*** Probabilistic Models of Cognition (online book)
9/17	Probabilistic Programming (2) ***Background Read:*** Probabilistic Models of Cognition, Ch.7 ***Examples:*** Examples worked out in class ***Background Read:*** PPAML Summer School 2016	Sasa Slides	***Additional Read:*** Modeling Agents with Probabilistic Programs (online book)
9/22	Probabilistic Programming (3) ***Background Read:*** Bayesian inference using data flow analysis (FSE 2013)	Sasa Slides	***Additional Read:*** The Design and Implementation of Probabilistic Programming Languages (online book)
9/24	Testing and Verifying Approximate, Nondeterministic, and Probabilistic Software ***Background Read:*** A Comprehensive Study of Real-World Numerical Bug Characteristics (ASE 2017) ***Background Read:*** Testing Probabilistic Programming Systems (FSE 2018) ***Background Read:*** A practical guide for using statistical tests to assess randomized algorithms in software engineering (ICSE 2011)	Sasa Slides	

Tentative Paper List

9/29 Approximate Systems (1)

Primary Read: Managing Performance vs. Accuracy Trade-offs With Loop Perforation (FSE 2011)

Secondary Read: Best-Effort Parallel Execution Framework for Recognition and Mining Applications (IPDPS 2009)

10/1 Approximate Systems (2)

Primary Read: Input responsiveness: using canary inputs to dynamically steer approximation (PLDI 2016)

Secondary Read: Crayon: Saving Power through Shape and Color Approximation on Next-Generation Displays (EuroSys 2016)

10/6 Approximate Systems (3)

Primary Read: Proactive control of approximate programs (ASPLOS 2016)

Secondary Read: JouleGuard: Energy Guarantees for Approximate Applications (SOSP 2015)

10/8 Approximate Systems (4)

Primary Read: Rigorous floating-point mixed-precision tuning (POPL 2017)

Secondary Read: Chisel: Reliability- and Accuracy-Aware Optimization of Approximate Computational Kernels (OOPSLA 2014)

10/13 Accuracy-Aware DNN Inference

Primary Read: ApproxHPVM: A Portable Compiler IR for Accuracy-aware Optimizations (OOPSLA 2019)

Secondary Read: PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions (NIPS 2016)

10/15 Pruning/Distilling in Neural Networks

Primary Read: Scalpel: Customizing DNN Pruning to the UnderlyingHardware Parallelism (ISCA 2017)

Secondary Read: ExTensor: An Accelerator for Sparse Tensor Algebra (MICRO 2019)

10/20 Quantization of Neural Networks

Primary Read: Deep Compression: Compressing Neural Networks With Pruning, Trained Quantization, and Huffman Coding (ICLR 2016)

Secondary Read: Quantized neural networks: training neural networks with low precision weights and activations (JMLR 2017)

10/22 Training DNNs in Low-Precision

Primary Read: Understanding and Optimizing Asynchronous Low-Precision Stochastic Gradient Descent (ISCA 2017)

Secondary Read: Training Deep Neural Networks with 8-bit Floating Point Numbers (NeurIPS 2018)

10/27 Testing: Deep Learning Applications

Primary Read: DeepXplore: automated whitebox testing of deep learning systems (SOSP 2017)

Secondary Read: Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks? (FSE 2020)

10/29 Testing: Numerical Applications

Primary Read: Efficient Generation of Error-Inducing Floating-Point Inputs via Symbolic Execution (ICSE 2020)

Secondary Read: Effective Floating-Point Analysis via Weak-Distance Minimization (PLDI 2019)

11/3 Testing: Probabilistic Applications

Primary Read: Statistical Algorithmic Profiling for Randomized Approximate Programs (ICSE 2019)

Secondary Read: Detecting Flaky Tests in Probabilistic and Machine Learning Applications (ISSTA 2020)

11/5 Probabilistic Programming Systems (1)

Primary Read: Design and Implementation of Probabilistic Programming Language Anglican (IFL 2016)

Secondary Read: Stan: A Probabilistic Programming Language (Journal of Statistical Software)

11/10 Probabilistic Programming Systems (2)

Primary Read: Compiling Markov Chain Monte Carlo Algorithms for Probabilistic Modeling (PLDI 2017)

Secondary Read: AcMC 2: Accelerating Markov Chain Monte Carlo Algorithms for Probabilistic Models (ASPLOS 2019)

11/12 Probabilistic Programming Systems (3)

Primary Read: Gen: a general-purpose probabilistic programming system with programmable inference

Secondary Read: Pyro: Deep Universal Probabilistic Programming (JMLR 2018)

11/17 Probabilistic Programming Systems (4)

Primary Read: R2: An Efficient MCMC Sampler for Probabilistic Programs (AAAI 2014)

Secondary Read: Probabilistic Programming with Densities in SlicStan: Efficient, Flexible and Deterministic (POPL 2019)

12/1 Application of Probabilistic Analysis 1

Primary Read: Static Analysis for Probabilistic Programs: Inferring Whole Program Properties from Finitely Many Paths (PLDI 2013)

Secondary Read: Bayonet: Probabilistic Inference for Networks (PLDI 2018)

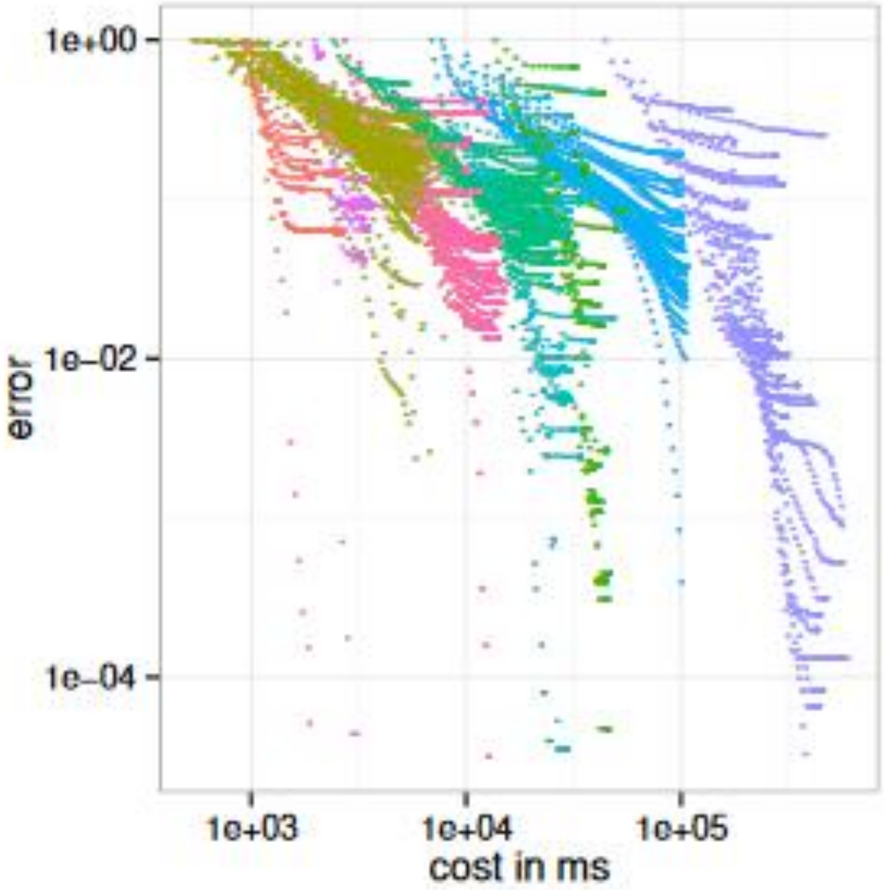
10/6 Application of Probabilistic Analysis 2

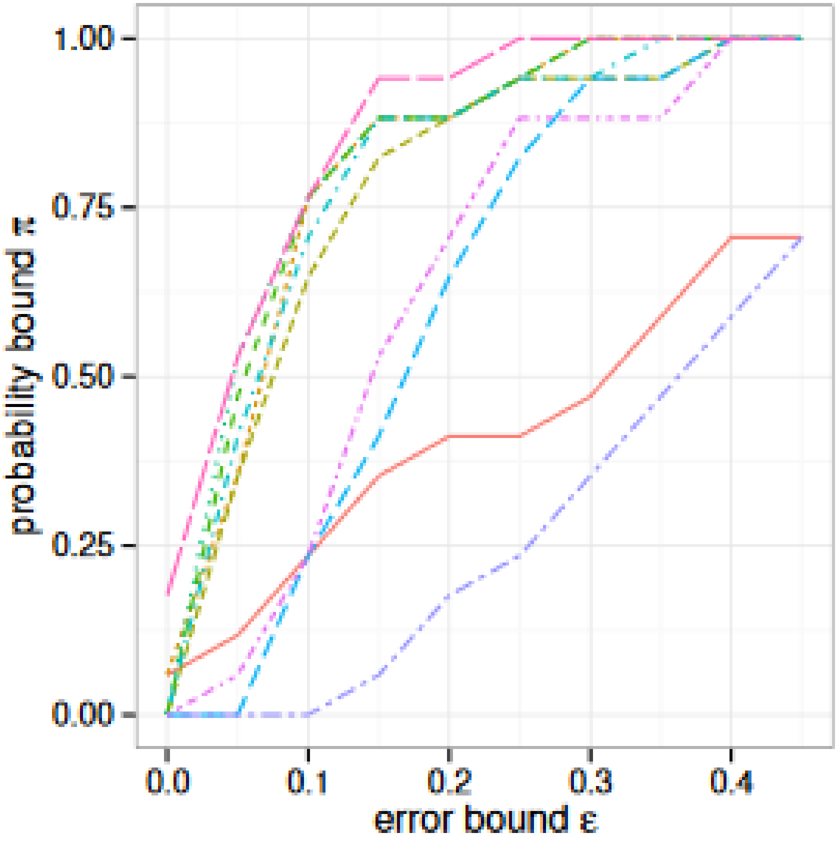
Primary Read: Verifying Quantitative Reliability for Programs that Execute on Unreliable Hardware (OOPSLA 2013)

Secondary Read: Approxilyzer: Towards A Systematic Framework for Instruction-Level Approximate Computing and its Application to Hardware Resiliency (MICRO 2016)

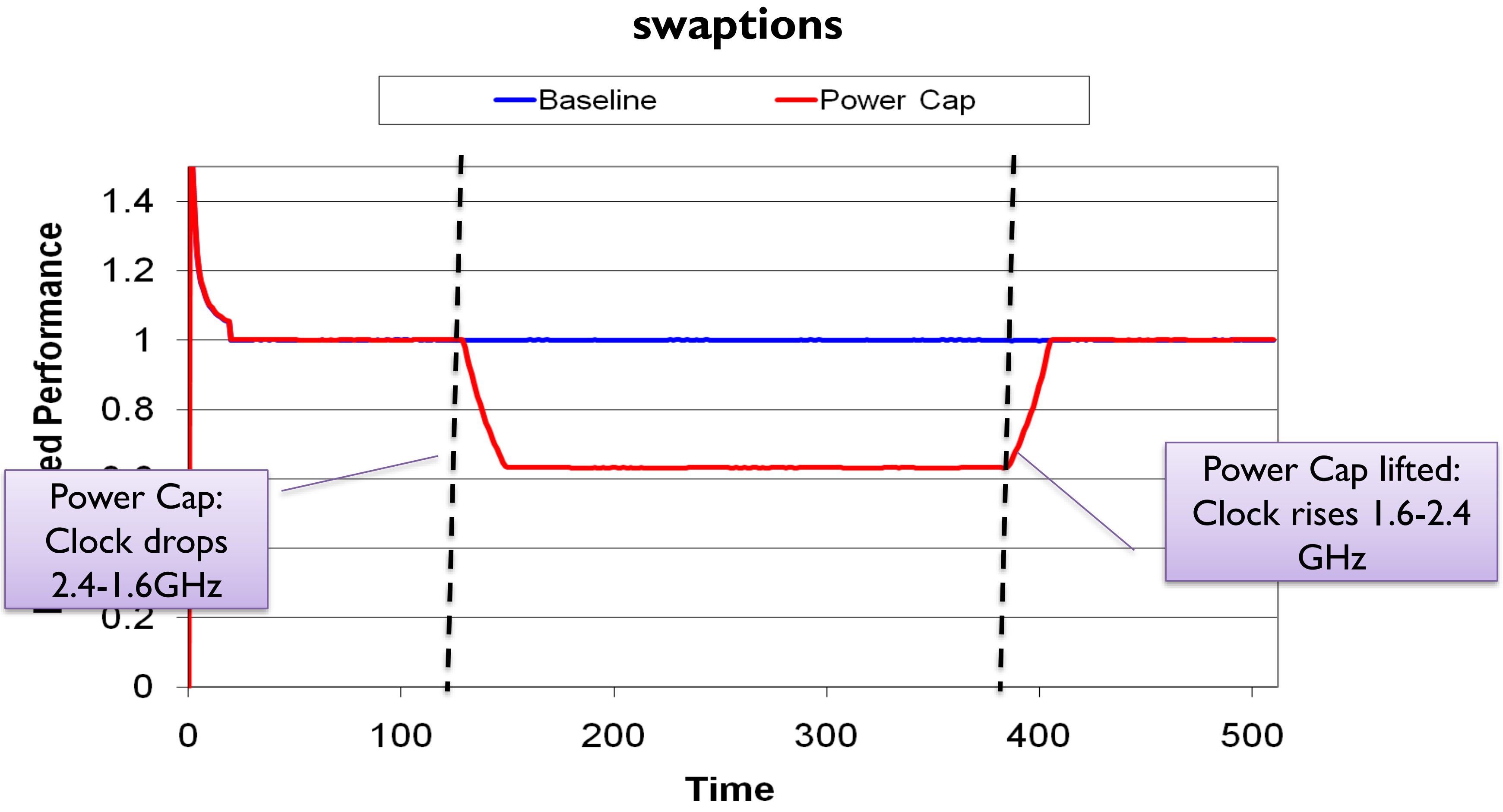
Systematically Exploring Tradeoffs of Approximation

Capri, ASPLOS 2016



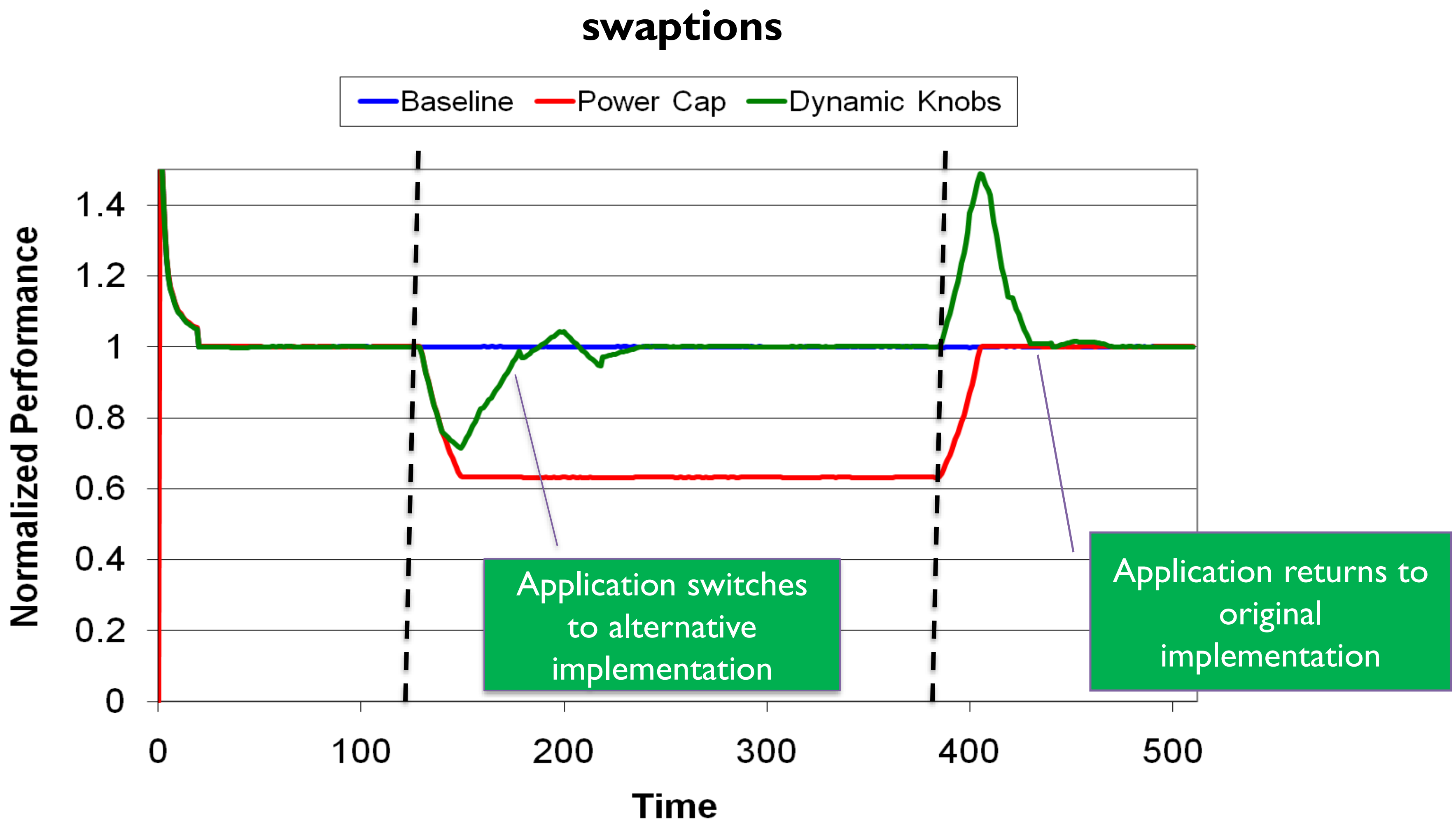


Study Dynamic Approximation



During the power cap, we either restart or suffer through poor performance.

Study Dynamic Approximation



Approximations in ML Frameworks in software and hardware

Scalpel ISCA 2017

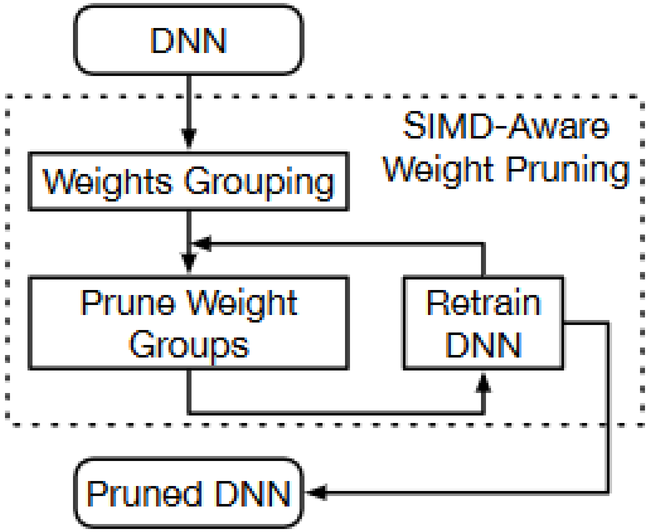


Figure 7: Main steps of SIMD-aware weight pruning.

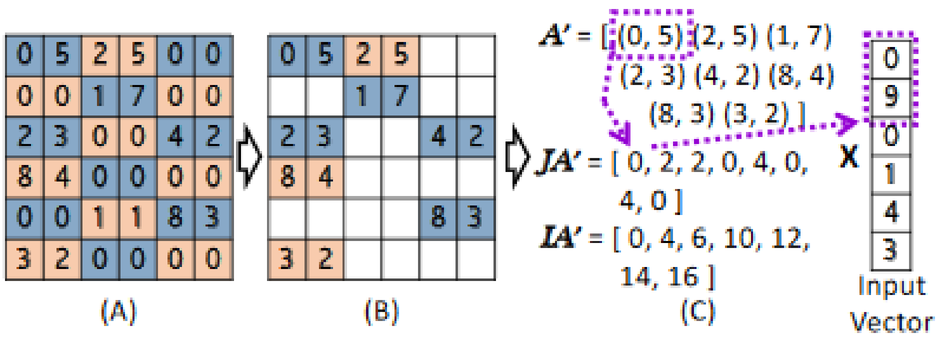


Figure 8: (A) Weights grouping; (B) Sparse weight matrix after pruning weight groups; (C) Modified CSR format for SIMD-aware weight pruning.

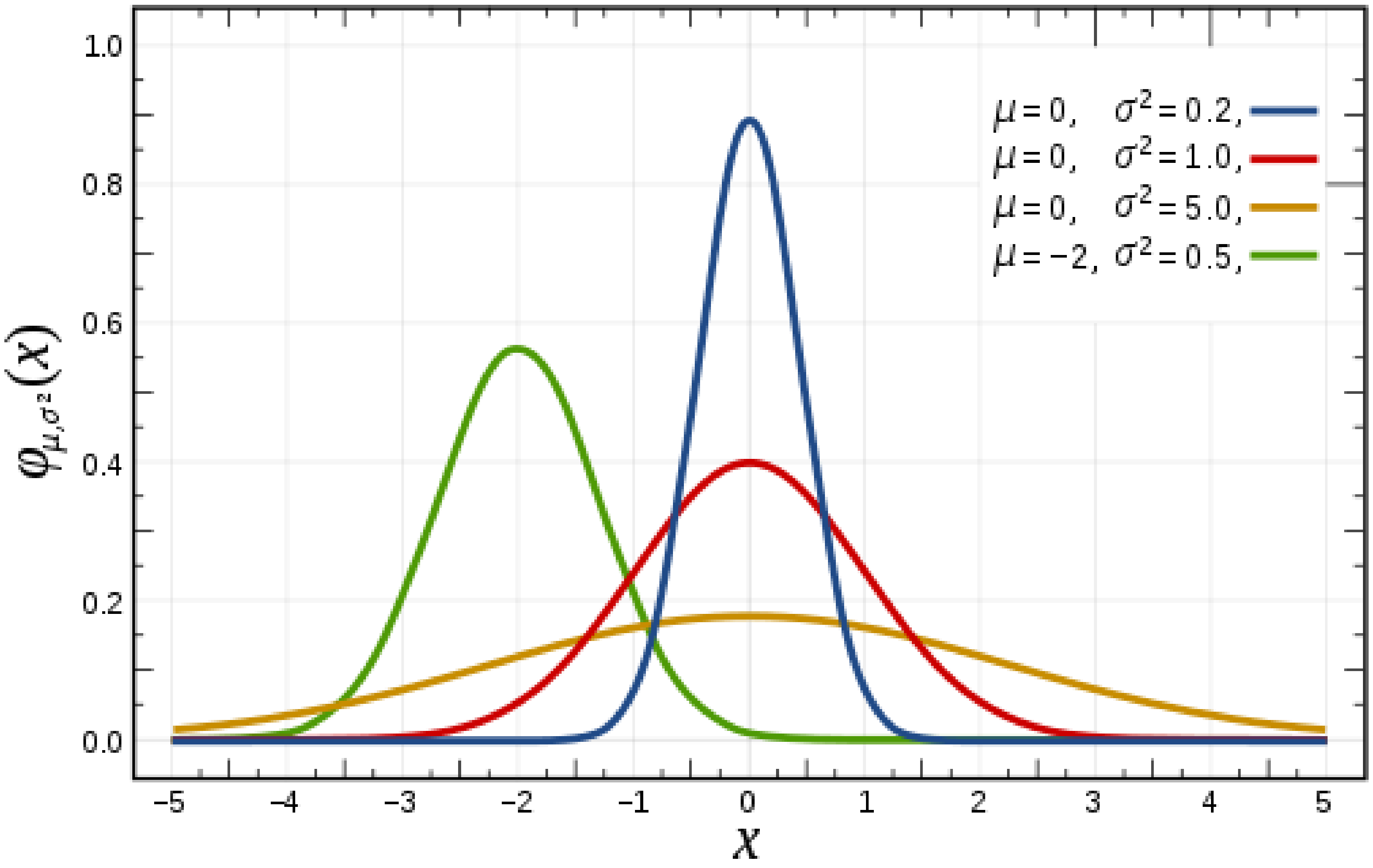
Testing Approximate and Probabilistic Programs

DeepXplore, SOSP 2017

(a) Input 1 (b) Input 2 (darker version of 1)

Figure 1: An example erroneous behavior found by DeepXplore in Nvidia DAVE-2 self-driving car platform. The DNN-based self-driving car correctly decides to turn left for image (a) but incorrectly decides to turn right and crashes into the guardrail for image (b), a slightly darker version of (a).

Probabilistic Programming



Probabilistic Programs

Extend Standard (Deterministic) Programs

Distribution	`X := Uniform(0, 1);`
Assertion	`assert ( X >= 0 );`
Observation	`observe ( X >= 0.5 );`
Query	`return X;`

Microsoft

facebook

UBER

Google

Tensorflow
Probability

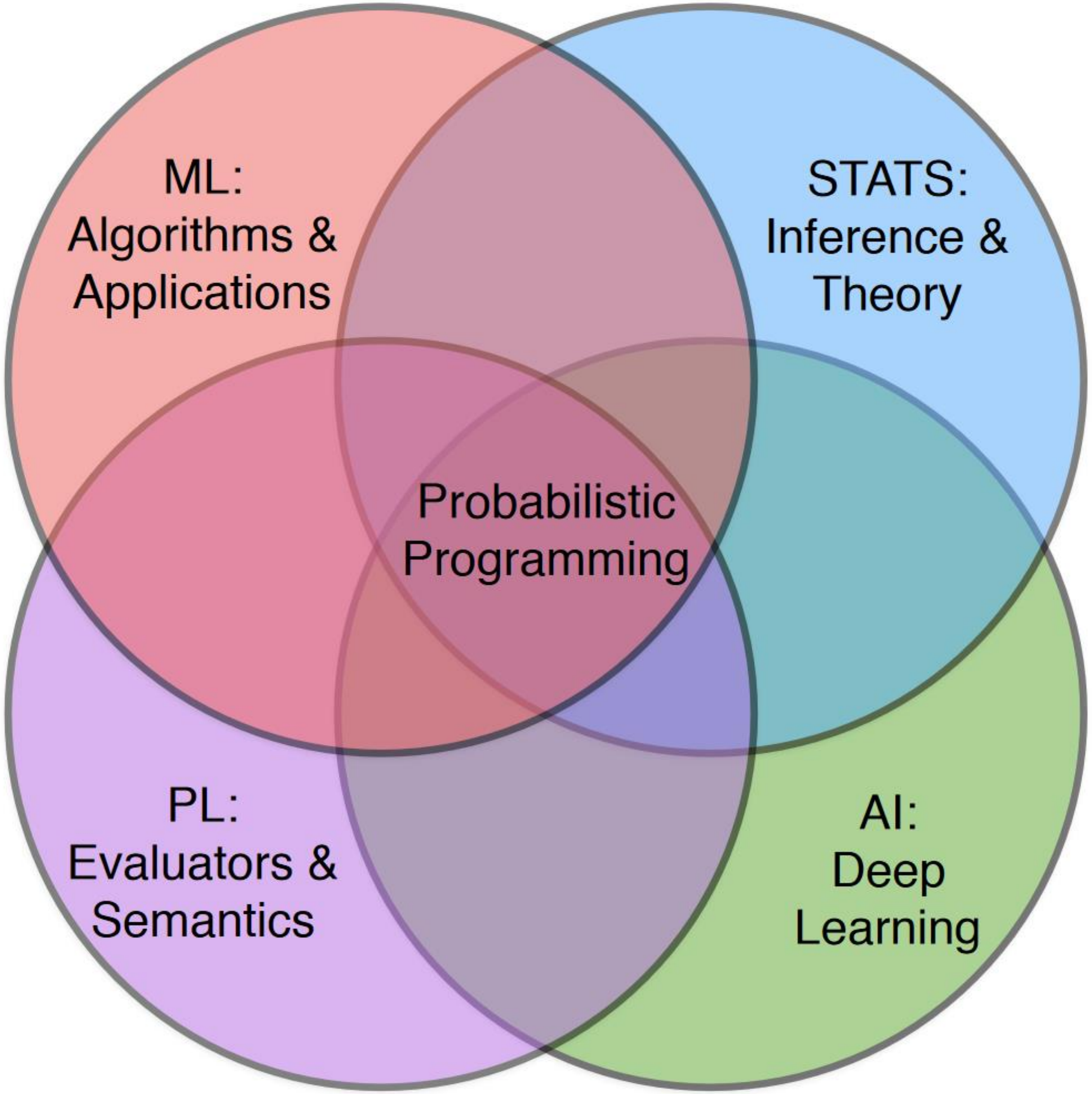


Diagram by Frank Wood

CS 598 SM

COURSE LOGISTICS

Schedule

Twice a week – Tuesdays and Thursdays 11am-12:15pm

We first do several lectures: tutorial style introductions to

- Approximate computing
- Probabilistic programming
- Covers key ideas and classical results

In the majority of the course, we will discuss recent papers

- Typically, discussion focus is on one paper at a time
- One student presents the paper
- Everyone participates in the discussion

Course Format

Research-oriented Course:

- Discussing latest research
- Reading from primary literature (papers)
- Focus on finding new ideas and building new systems, not lecturing and grading

Research project is the main outcome of the course

- Be able to publish your work at a conference
- It is **hard**! **Unpredictable + requires time and effort**

The Show Must Go On…

Bad Internet connection, Zoom, other sites, students cannot access some services in their area, life happens…

Lesson from approximate computing:

Something will fail occasionally
What matters is how we respond and recover.

Prerequisites

Basic Probability (e.g., CS 361)

Basic Compilers and/or PL course (e.g., CS 421, CS 426)

Basic Computer Architecture (e.g., CS 233)

Basic Machine Learning (e.g., CS 446)

(or a commitment to learn as you go)

Real Prerequisites

Being comfortable with the idea of doing research

(If you don't know what you're getting into,
talk to me after the class)

Grading

Reviews & Discussion	20%
Paper Presentations	25%
Project	50%
Homework	5% + 10% XC

Papers

For the majority of the class, we will **jointly read and discuss** recent research results

We focus on one paper in each class discussions

- but for most part we put it in context with at least another related paper

Make sure you can make it to the class on the day you're presenting the paper!

Selecting Papers

Submit at least 5 candidate papers you'd like to present

- List of papers is on the website (use the week and number)
- Split in the ***primary read*** – the one we will focus the most in the discussion and ***secondary read*** – the one we will use to expand our understanding of the topic
- If you'd like a paper outside of the list, email me and make a case

Submission deadline is **September 10**

- Link: http://misailo.web.engr.Illinois.edu/courses/598sm/
- Will get back with the assignments by the class after

Reviews and Discussions

For each paper, write a review of between 500-1000 words:

- Summarize the **primary** paper:
  state main contributions in one/two paragraphs (use your own words!)
- Evaluate the contribution and discuss pros and cons:
  give a honest critique of the approach (main part)
- Two questions:
  about the paper, the general topic, its impact, or extensions (key!)
- Summarize the **secondary** paper after reading only its introduction (and maybe example) sections
  State main contributions in one/two paragraphs
- Relate the two papers
  What is similar and what is different (extrapolate what the 2^{nd} paper does from the first two sections)

Reviews and Discussions

Send reviews before the lecture

- By midnight two days before the lecture
- Submission forms: we will use HotCRP

Discuss papers online before the class

- You will be able to see everyone else's reviews
- Improve your understanding on the paper's pros/cons
- Free to update your reviews after reading the other reviews.
- The lead student moderates the discussion

Participate in the discussion during the class

- We try to reach
- Purpose: practice how to be loud
  (at the conferences, board meetings, home…)

Paper Lead: Presentation

Week before: Meet with the instructor

- **Mandatory!** (we can set up 30min meeting)
- Discuss outline and questions you have so far (ok if still rough!)
- Send the recorded video by the time the reviews are due

30 minute slot per presentation (ok to have video):

- Explain motivation for the work
- Clearly present the technical solution and results
- Use your own example (not the one from the paper)
- Outline limitations / improvements
- Focus on concepts, leave out nonessential details
- Discuss the impact on the related/follow-up work

Paper Lead: Roles (Continued)

Before the class:

- Lead the online discussion (I will often help)
- Prepare the response to the questions raised by other students (by pointing into the paper or other related work)

After your presentation:

- Lead the discussion
  (take the online discussion as a starting point)
- Help reach the verdict about the main points of the paper
- Summarize the discussion on the online system after the class

Grading Presentations

Presentation quality:

- How well did you understand the work?
- How well did you present it (clarity and grace)?
- How well did you answer the questions?
- How well did you write the after-class report?

We will take into account the **paper difficulty**

Project

Teams of two but individual is also ok this semester

- Teamwork is a great experience!
- But this year is strange in so many ways!

Research projects, some ideas:

- New Software and/or hardware approximations
- Dynamic or input-aware approximations
- Optimize approximate inference algorithms
- New program analysis for probabilistic programs
- New probabilistic analysis of approximate programs
- Implement and compare existing approaches
- Survey literature on an emerging topic

Grading Projects

Proposal by **October 13**

- Meet with instructor for a quick discussion

Deliverables:

- Short paper – up to 5 pages ACM 10pt format
- Think of e.g., DATE (https://www.date-conference.com/)
- Project overview – 10/15 minutes
- Officially, due **last week of classes (Tuesday)**

Real outcome:

- Prepare (or make a good step toward) a publishable research paper

Grading

Reviews & Discussion	
Paper Presentations	25%
Project	50%
Ho	5% + 10% XC

Grading on an absolute scale (no curve!)

CS 598 SM

RESOURCES FOR READING, WRITING AND PRESENTING

Reading Papers

"How to Read a Research Paper",
by Michael Mitzenmacher

http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf

"How to Read an Engineering Research Paper",
by William Griswold

http://cseweb.ucsd.edu/~wgg/CSE210/howtoread.html

Advice compiled by Tao Xie:

http://taoxie.cs.illinois.edu/advice.htm#review

Writing Reviews

"The Task of the Referee", by Allan Smith

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.177.3844

"Constructive and Positive Reviewing",
by Mark Hill and Kathryn McKinley

http://www.cs.utexas.edu/users/mckinley/notes/reviewing.html

Presenting Research

“How to give strong technical presentations”
by Markus Püschel

http://users.ece.cmu.edu/~pueschel/teaching/guides/guide-presentations.pdf

Patrick Winston’s talk @ MIT:

https://www.youtube.com/playlist?list=PL9F536001A3C605FC

Jean Luc Doumont’s talk

https://www.youtube.com/watch?v=meBXuTIPJQk

CS 598 SM

QUESTIONS SO FAR?