CS 521: Topics in PL

Probabilistic & Approximate Computing

http://misailo.web.engr.Illinois.edu/courses/cs521

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

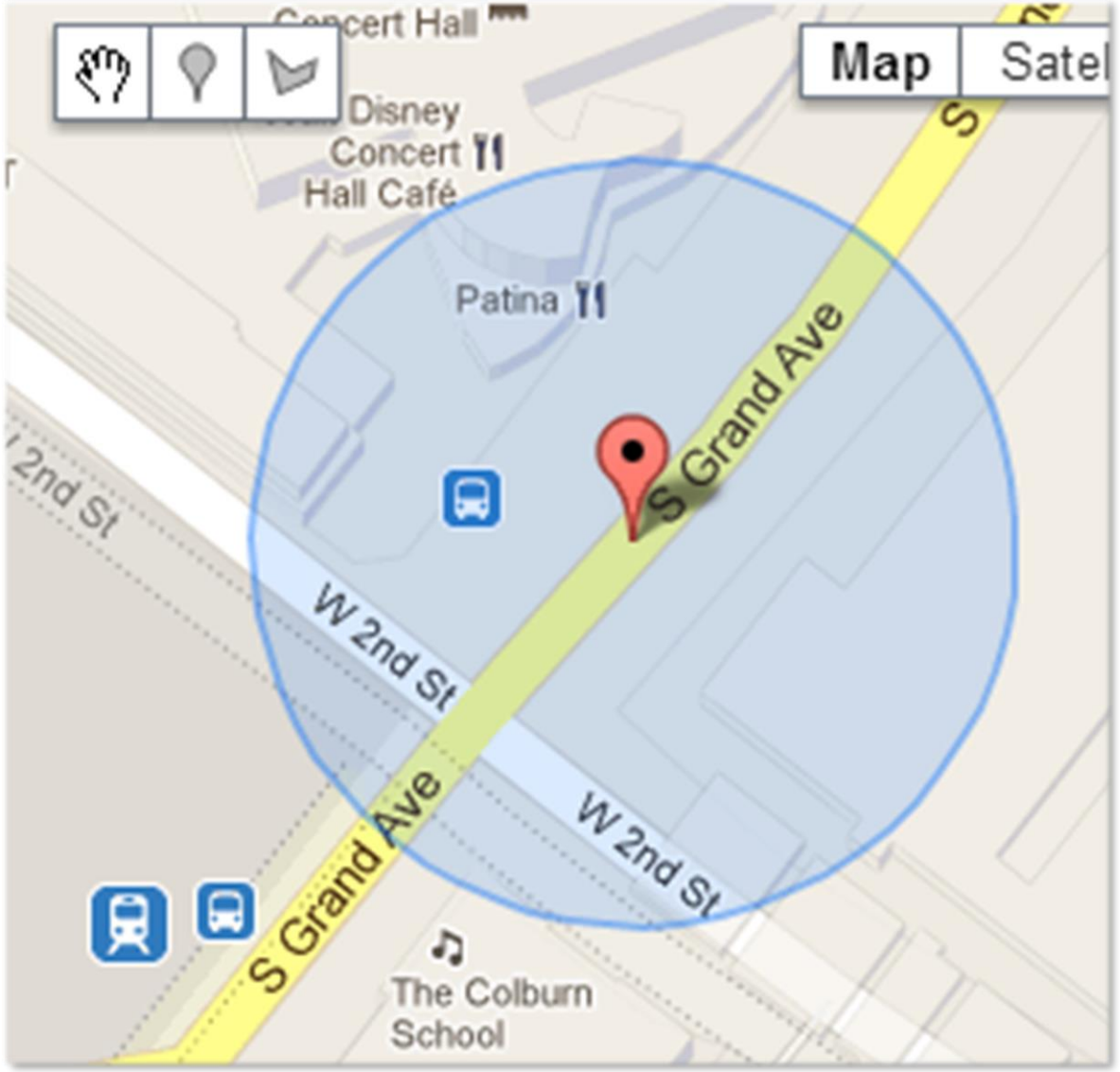
Uncertainty

"something that is doubtful or unknown"

Probability quantitatively represents uncertainty (captures the degree of belief)

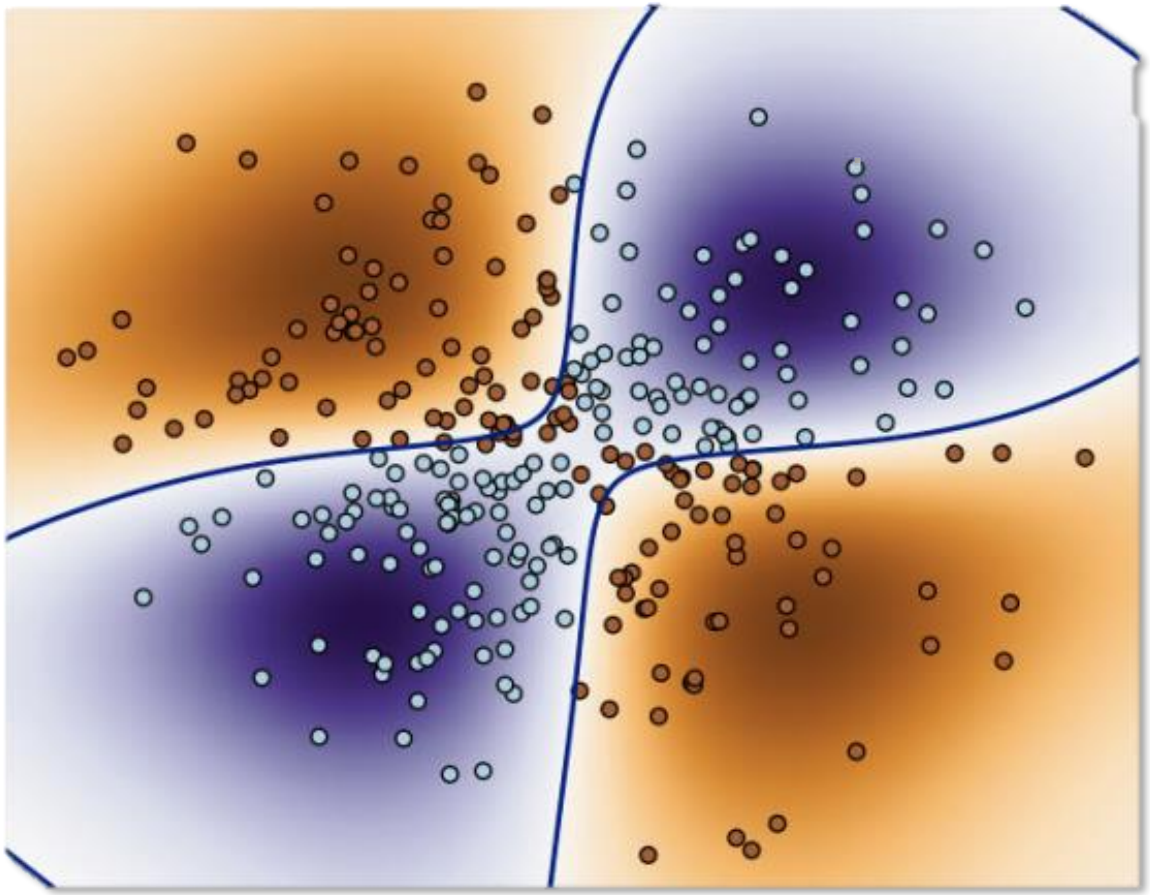
Approximation efficiently copes with uncertainty (ignores it or tractably computes in its presence)

Location Tracking



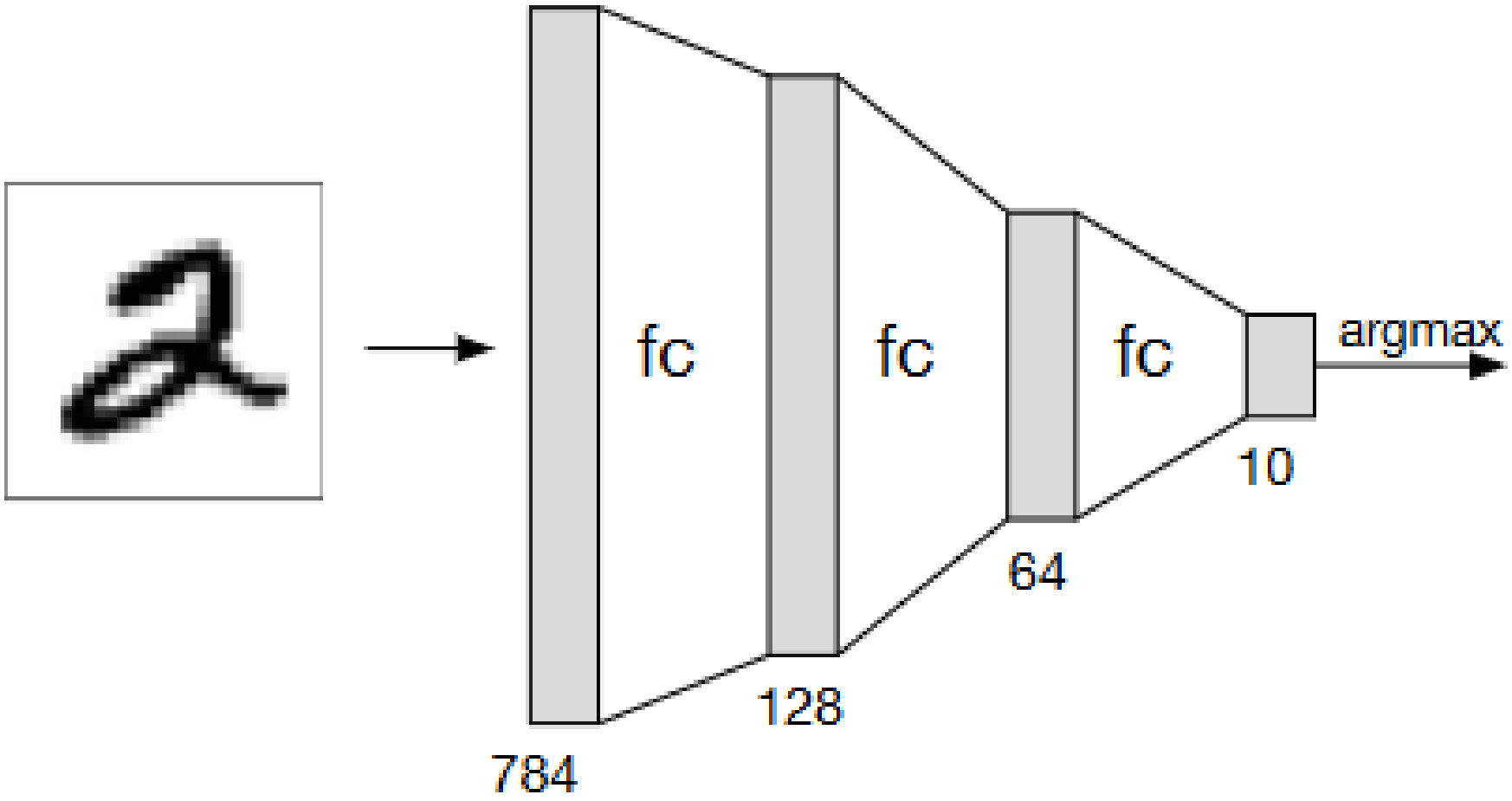
A probability distribution is hiding on the screen. Find it!

Data Classification



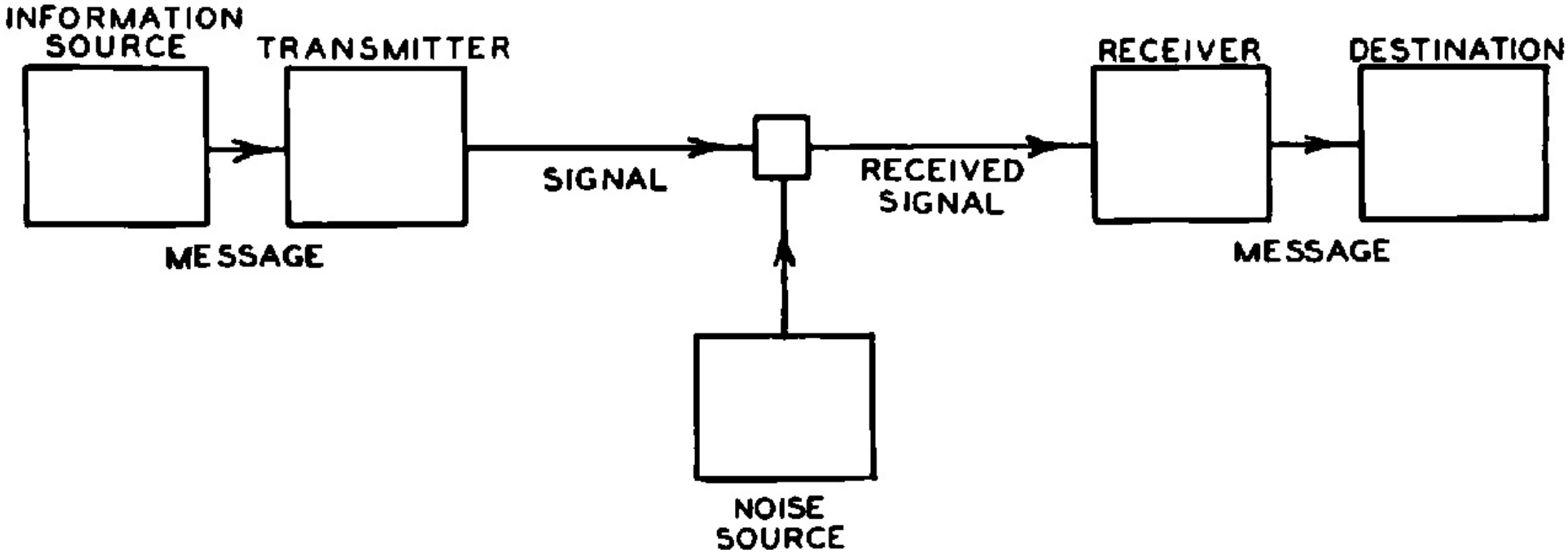
Are all red points in the same cluster?

Pattern Recognition



'two'

Communication between Agents



Big Data

Machine Vision

Extended Reality

Autonomous Systems

Numerical Simulations

...

Some Common Traits

Noisy Data coming from sensors

Redundancy in data and computation

Models that effectively compress such redundancy

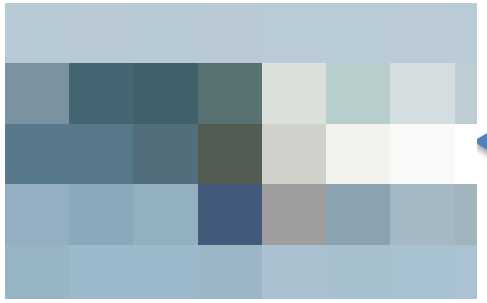
Environments that we don’t fully understand

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

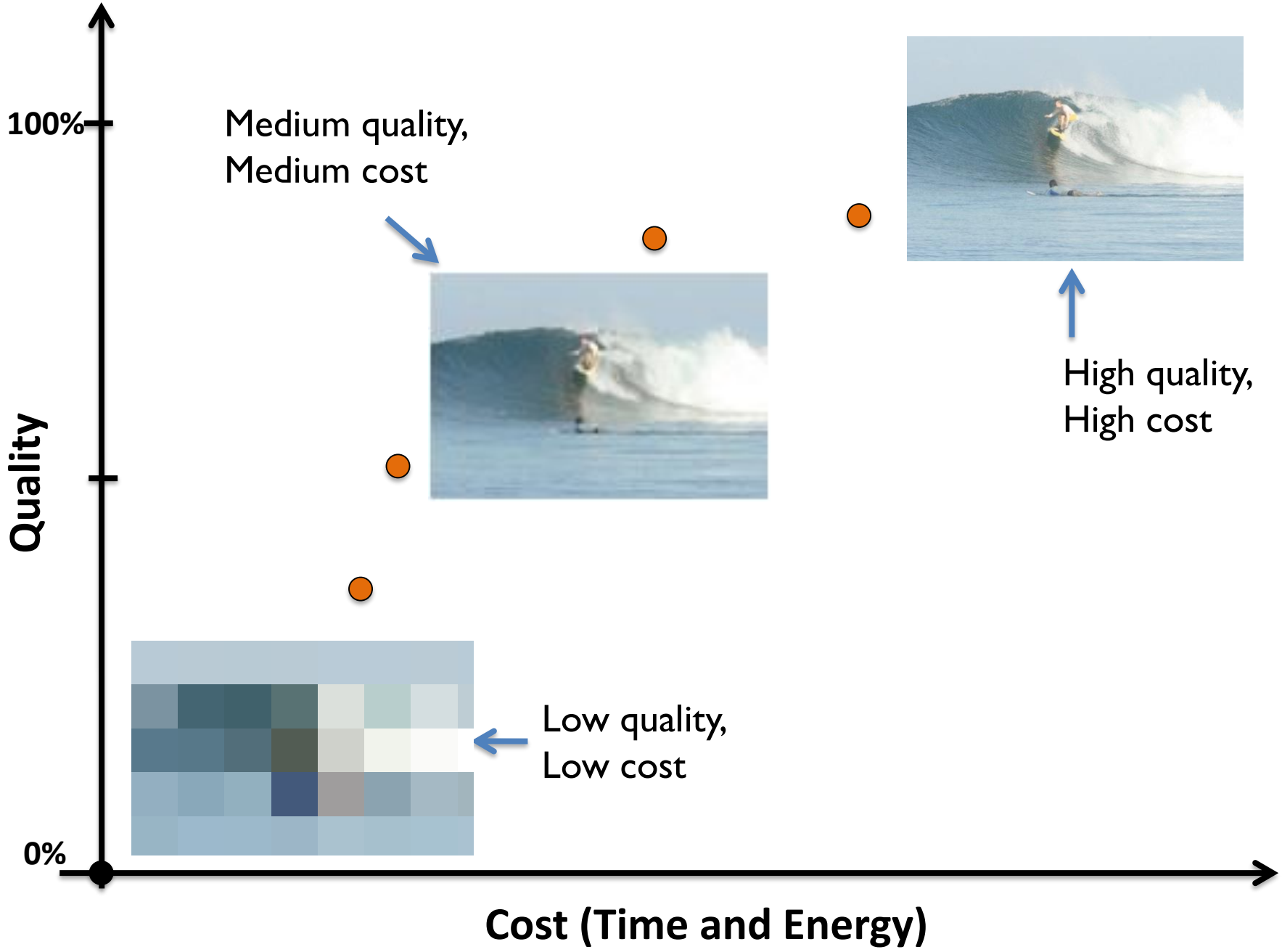
George E. P. Box

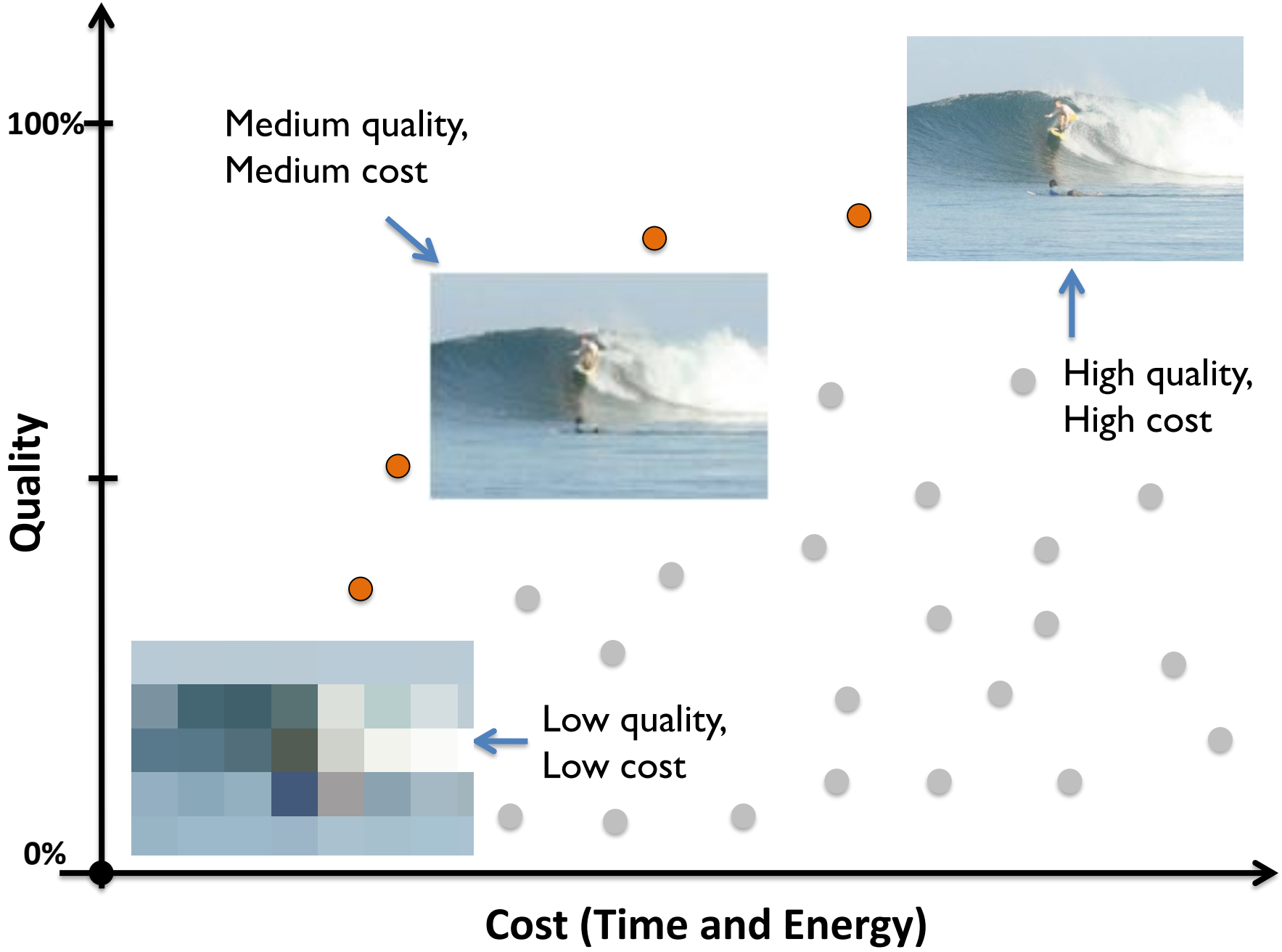
Medium quality,
Medium cost

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)

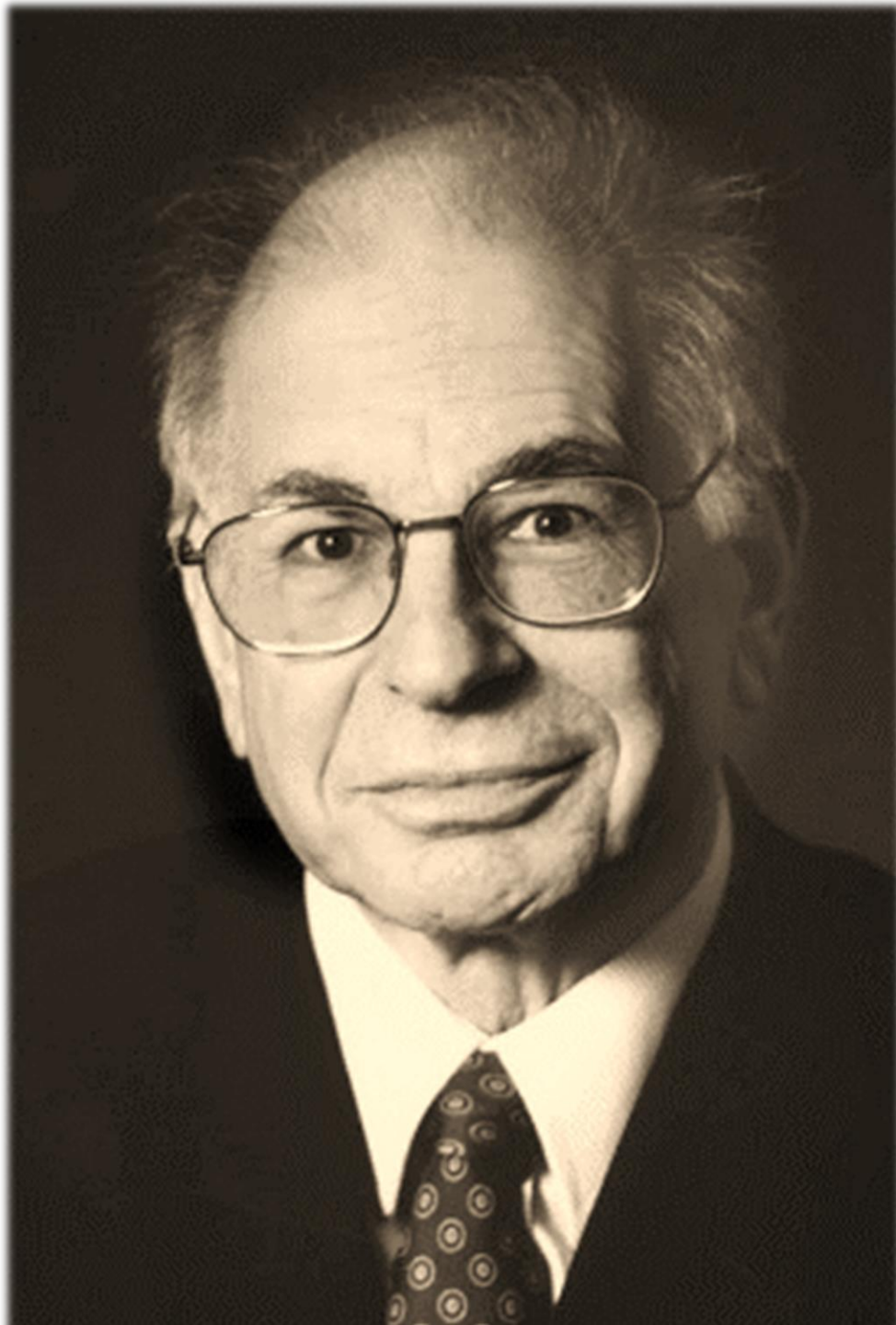

100%
0%
Quality
Medium quality,
Medium cost
High quality,
High cost
Low quality,
Low cost
Cost (Time and Energy)

Two ways the brain forms thoughts:

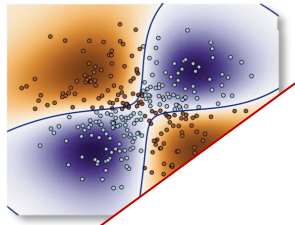
System 1: fast, instinctive, and emotional

System 2: slow, conscious, and logical

Daniel Kahneman



Classical Computing Stack

Applications	  
Languages and Compilers	Trans am code to *id* e code
Systems Softwar	nically schedules program to nimize time
H Ar tures	Execute the machine code ***in unambiguous manner***

Perfection! System 2!

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.

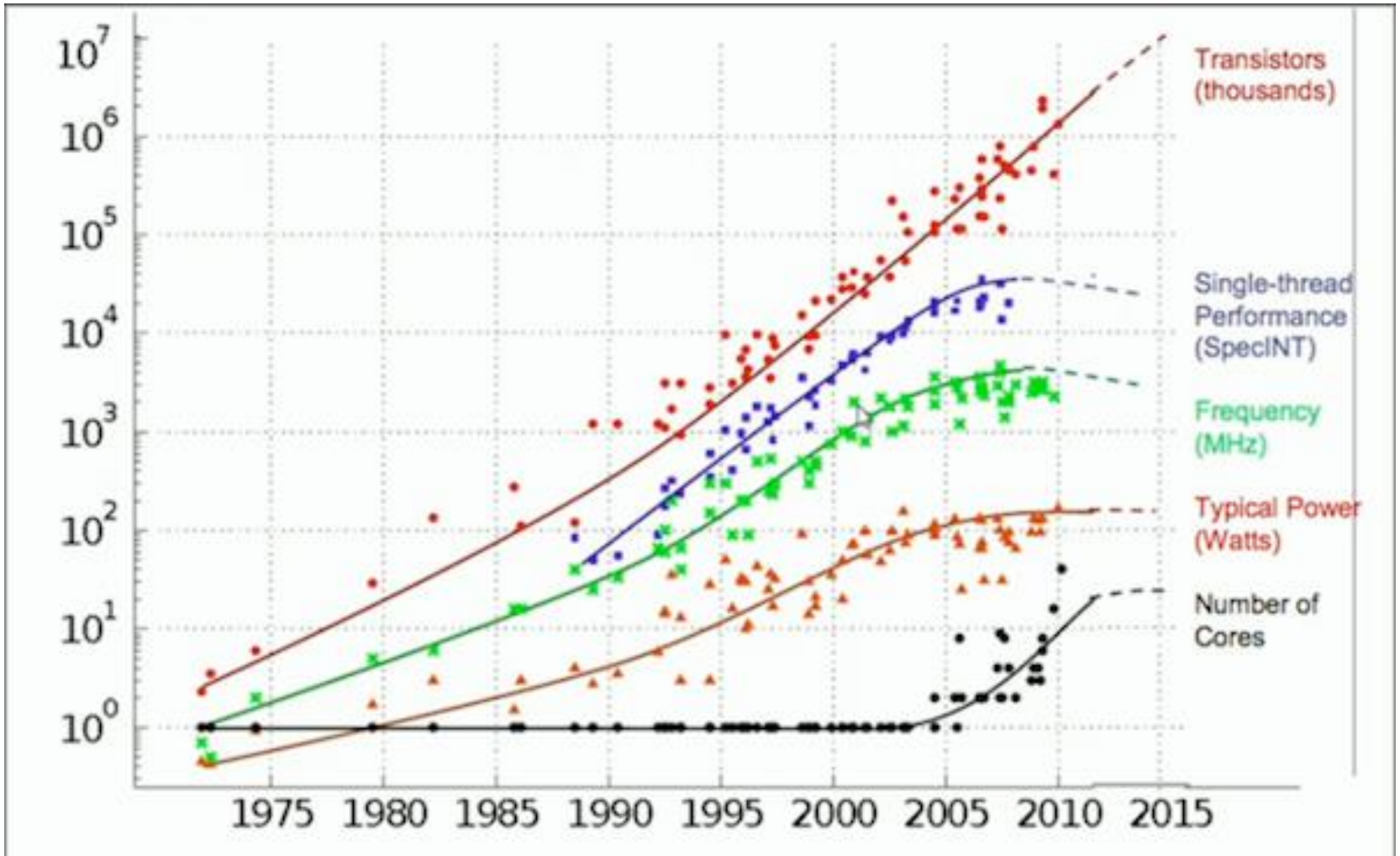
Thus, as transistors shrink, so do necessary voltage and current.

Power is proportional to the area of the transistor (while the transistor is still reliable)

Robert Dennard

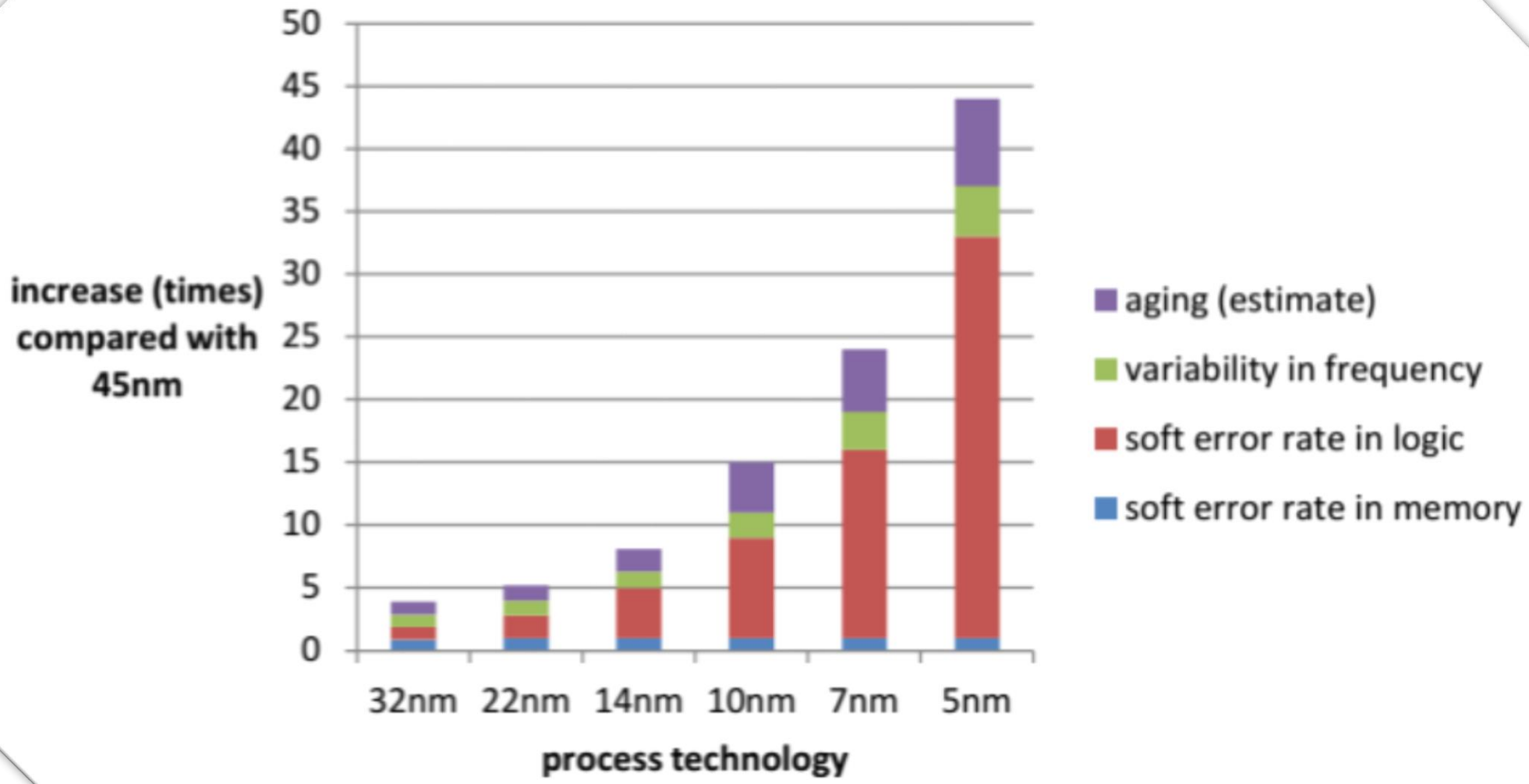
* Robert Dennard (from W. Gropp; UIUC CS 598WG)

End of Dennard Scaling (2005-now)



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

How Much Can We Shrink?



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

Meanwhile…

Cores that don't count

Peter H. Hochschild, Paul Turner, Jeffrey C. Mogul — Google, Sunnyvale, CA, US
Rama Govindaraju, Parthasarathy Ranganathan — Google, Sunnyvale, CA, US
David E. Culler, Amin Vahdat — Google, Sunnyvale, CA, US

Abstract

We are accustomed to thinking of computers as fail-stop, especially the cores that execute instructions, and most system software implicitly relies on that assumption. During most of the VLSI era, processors that passed manufacturing tests and were operated within specifications have insulated us from this fiction. As fabrication pushes towards smaller feature sizes and more elaborate computational structures, and as increasingly specialized instruction-silicon pairings are introduced to improve performance, we have observed e
computational errors that were not detected duri
facturing tests. These defects cannot always be miti
techniques such as microcode updates, and may be c
to specific components within the processor, allowi
code changes to effect large shifts in reliability. Wo
failures are often "silent" – the only symptom is an c
computation.

We refer to a core that develops such behavior as
rial." Mercurial cores are extremely rare, but in a la
of servers we can observe the disruption they cau
enough to see them as a distinct problem – one tha
quire collaboration between hardware designers, p
vendors, and systems software architects.

This paper is a call-to-action for a new focus in sy
search; we speculate about several software-based ap
to mercurial cores, ranging from better detection a
ing mechanisms, to methods for tolerating the sil
corruption they cause.

ACM Reference Format:
Peter H. Hochschild, Paul Turner, Jeffrey C. Mogul, Ran
daraju, Parthasarathy Ranganathan, David E. Culler, and A
dat. 2021. Cores that don't count. In *Workshop on Hot
Operating Systems (HotOS '21), May 31–June 2, 2021, A*

MI, USA. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3458336.3465297

1 Introduction

Imagine you are running a massive-scale data-analysis pipeline in production, and one day it starts to give you wrong answers – somewhere in the pipeline, a class of computations are yielding corrupt results. Investigation fingers a surprising cause: an innocuous change to a low-level library. The change itself was

Silent Data Corruptions at Scale

Harish Dattatraya Dixit — Facebook, Inc. — hdd@fb.com
Sneha Pendharkar — Facebook, Inc. — spendharkar@fb.com
Matt Beadon — Facebook, Inc. — mbeadon@fb.com
Chris Mason — Facebook, Inc. — clm@fb.com
Tejasvi Chakravarthy — Facebook, Inc. — teju@fb.com
Bharath Muthiah — Facebook, Inc. — bharathm@fb.com
Sriram Sankar — Facebook Inc. — sriramsankar@fb.com

ABSTRACT

machine learning inferences, ranking and recommendation systems. However, it is our observation that computations are not always *accurate*. In some cases, the CPU can perform computations incorrectly. For example, when you perform 2x3, the CPU may give a result of 5 instead of 6 silently under certain microarchitectural conditions, without an indication of the miscomputation in system event or error logs. As a result, a service utilizing the CPU is potentially unaware of the computational accuracy and keeps consuming the incorrect values in the application. This paper predominantly focuses on scenarios where datacenter CPUs exhibit such silent data corruption. We dive deep into a real-world application-level impact of a corruption, the processes used in debugging such corruption, and conclude with detection and mitigation strategies for silent data corruptions. While we present one case study, we have observed several scenarios, data paths and architectural blocks where SDCs manifest, and hence it is a systemic problem that the industry should tackle collectively.

Prior work [11], [24], [28], [14], [15], [18] within this domain focused on soft errors due to radiation or synthetic fault injection. In contrast, we observe that silent data corruptions are not limited to soft errors due to radiation and environmental effects with probabilistic models. Silent data corruptions can occur due to device characteristics and are repeatable at scale. We observe that these failures are reproducible and not transient. Techniques like Error Correction Code (ECC) are beneficial for reducing the error rates in SRAM. However not all the blocks within a datacenter CPU have

DESIGNLINES | AUTOMOTIVE DESIGNLINE

Toyota Case: Single Bit Flip That Killed

By Junko Yoshida 10.25.2013 0

MADISON, Wis. — Could bad code kill a person? It could, and it apparently did.

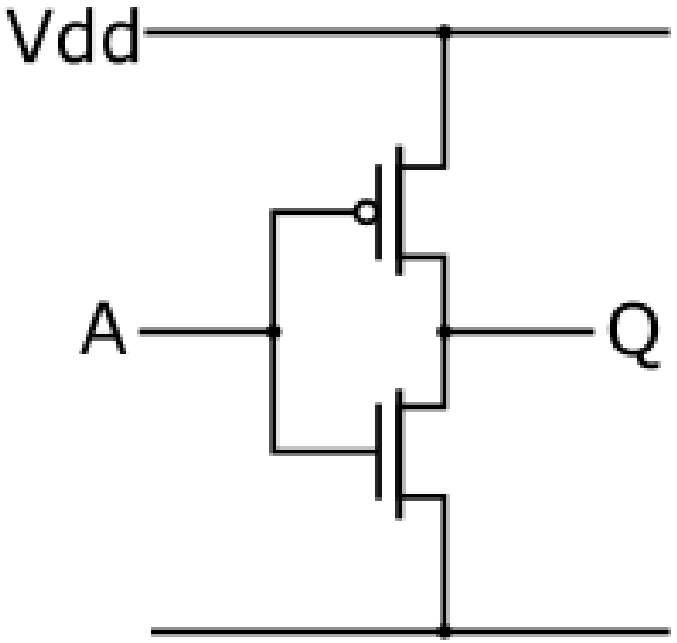
The Bookout v Toyota Motor Corp. case, which blamed sudden acceleration in a Toyota Camry for wrongful death, touches the issue directly.

This case — one of several hundred contending that Toyota's vehicles inadvertently accelerated — was the first in which a jury heard the plaintiffs' attorneys supporting their argument with extensive testimony from embedded systems experts. That testimony focused on Toyota's electronic throttle control system — specifically, its source code.

GET READY FOR OUR FIRST THOUGH EXPERIMENT…

CMOS Transistors

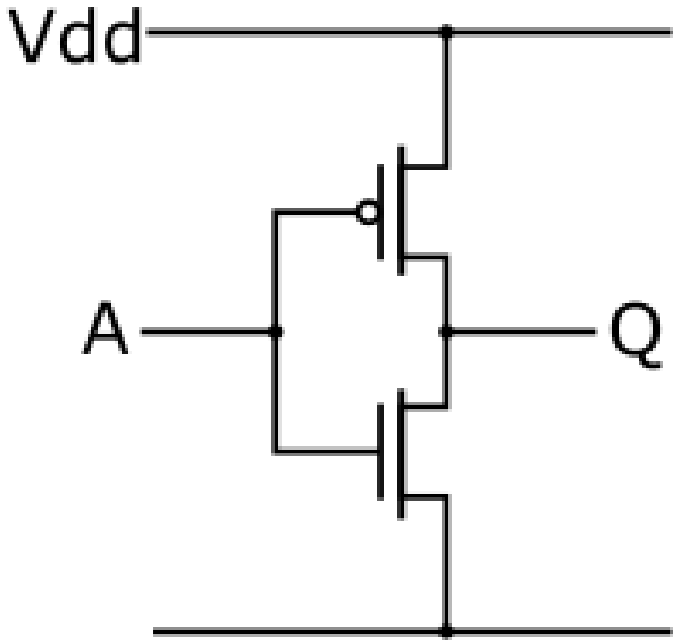
Simple Invertor



A	Q
0	1
1	0

Probabilistic CMOS Transistors

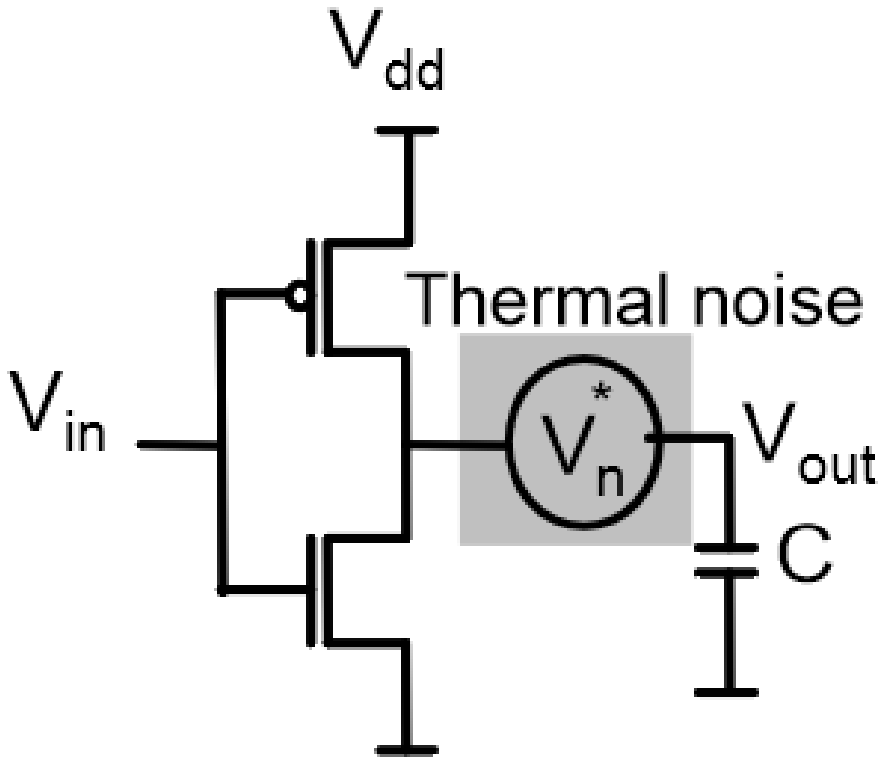
Simple Invertor

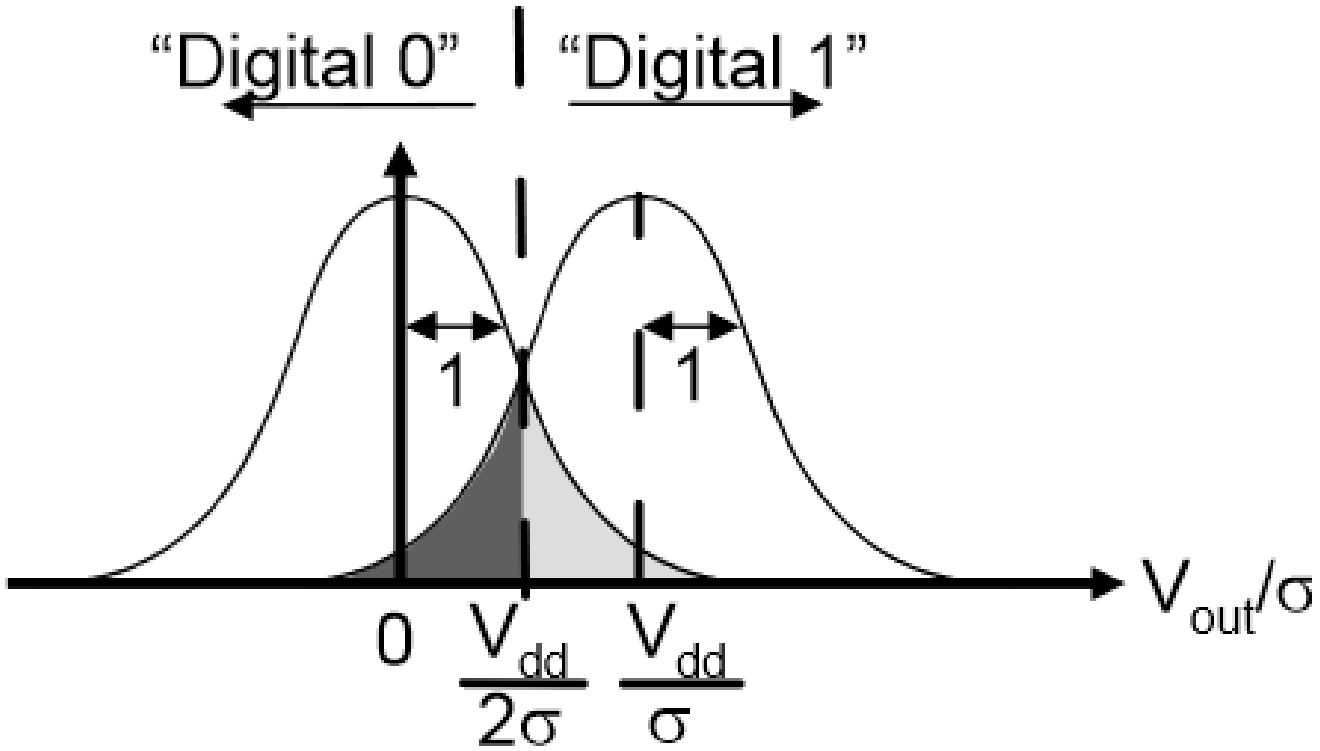


A	Q	
0	1	P>0.99
	0	P<0.01
1	0	P>0.99
	1	P<0.01

See: Probabilistic Design: A Survey of Probabilistic CMOS Technology and Future Directions for Terascale IC Design (VLSI 2006)

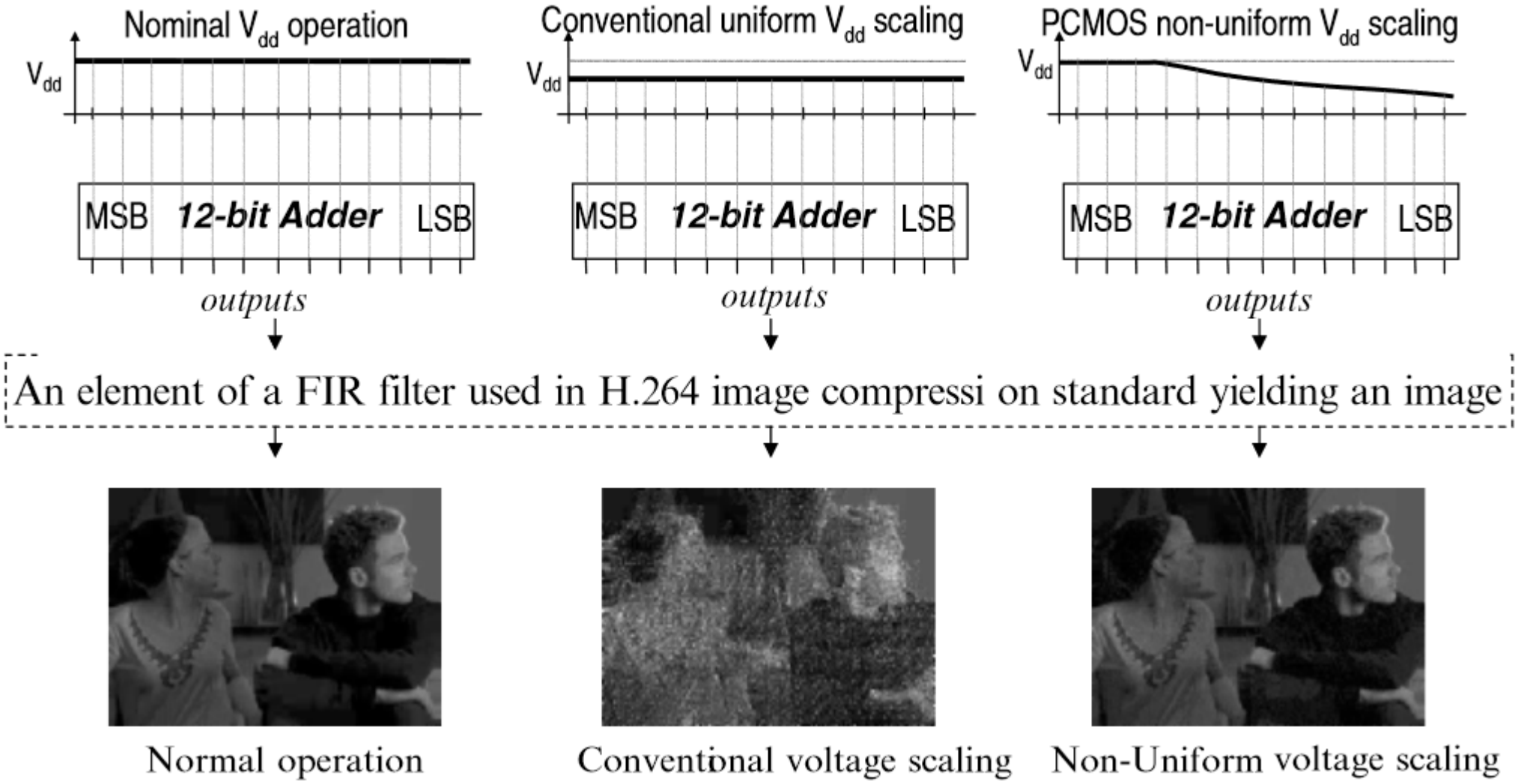
Probabilistic CMOS Transistors





See: Probabilistic Design: A Survey of Probabilistic CMOS Technology and Future Directions for Terascale IC Design (VLSI 2006)

Probabilistic CMOS Transistors



See: Probabilistic Design: A Survey of Probabilistic CMOS Technology and Future Directions for Terascale IC Design (VLSI 2006)

Breaking Digital Abstraction

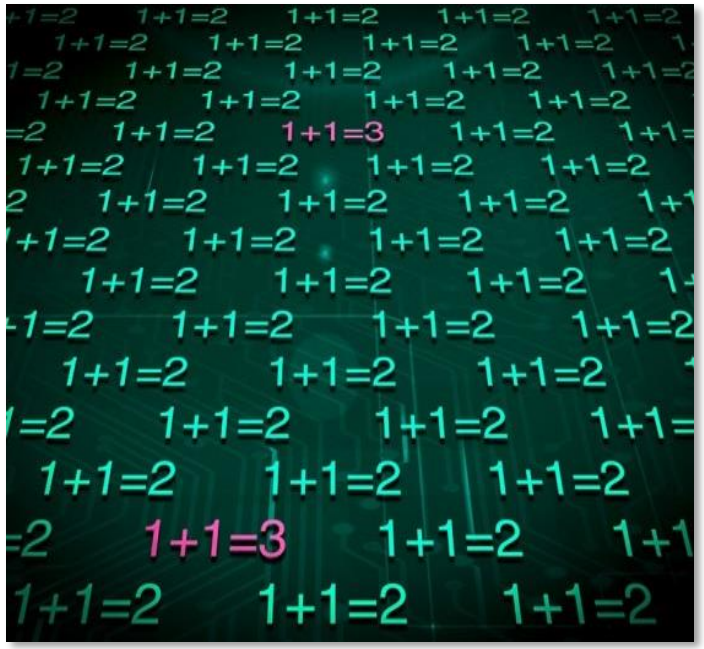
Approximate and Unreliable Hardware

- 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 …



© MIT News

- 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 (OOPSLA'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**

LET’S DO ANOTHER THOUGH EXPERIMENT…

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

Not a correctness issue

Accuracy and Safety issue

PROBABILISTIC LOGICS AND THE SYNTHESIS OF RELIABLE ORGANISMS FROM UNRELIABLE COMPONENTS

J. von Neumann

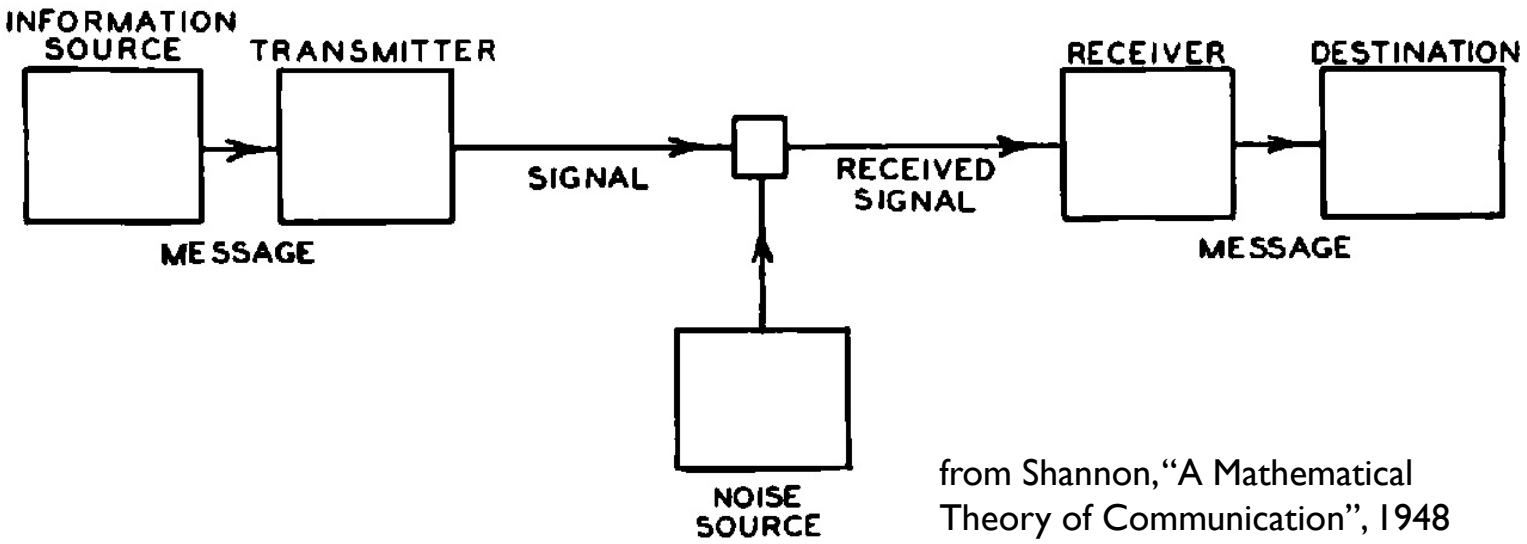
1. INTRODUCTION

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.

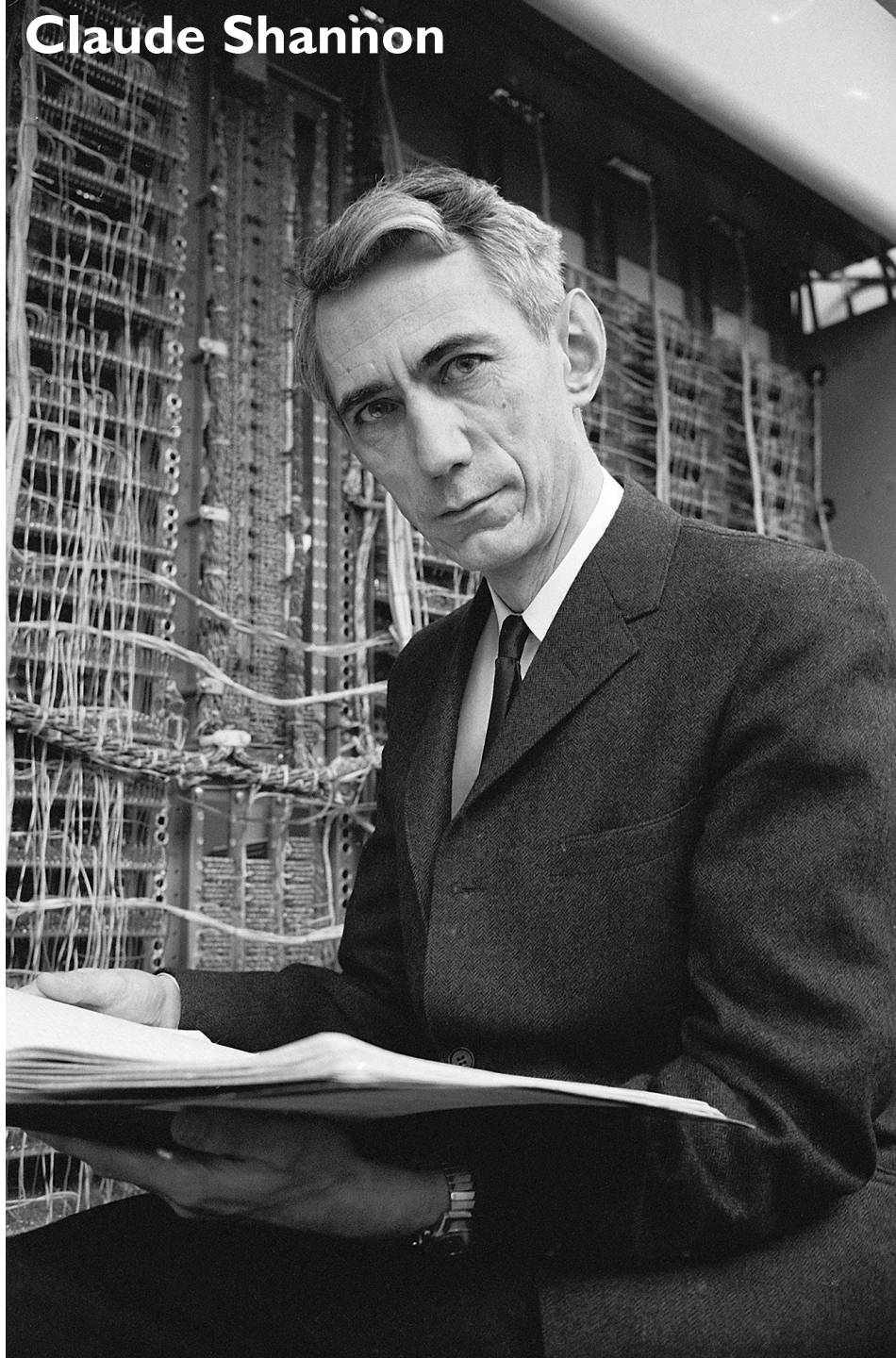
The author wants to express his thanks to K. A. Brueckner and M. Gell-Mann, then at the University of Illinois, to whose discussions in 1951 he owes some important stimuli on this subject; to Dr. R. S. Pierce at the California Institute of Technology, on whose excellent notes this exposition is based; 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.



from Shannon, "A Mathematical Theory of Communication", 1948

"Before Shannon, the problem of communication was primarily viewed as a deterministic signal-reconstruction problem: how to transform a received signal, distorted by the physical medium, to reconstruct the original as accurately as possible. **Shannon's genius lay in his observation that the key to communication is uncertainty**. After all, if you knew ahead of time what I would say to you in this column, what would be the point of writing it?"

https://www.quantamagazine.org/how-claude-shannons-information-theory-invented-the-future-20201222/



Classical Computing Stack

Applications

Languages and Compilers

ne program code to
al machine code

Syste
S

Dynamically schedules program to minimize time

ware
Architectures

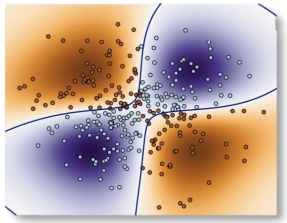
Execute the machine code
in unambiguous manner

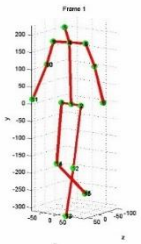
Perfection! Or not?

Accuracy-Aware 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**

Tentative Lectures

Date	Topic	Presenter	Notes
1/18	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)
1/20	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)
1/25	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)
1/27	Approximations: Machine Learning ***Background Read:*** Approximate Computing Techniques for Deep Neural Networks Book Chapter ***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) ***Additional Read:*** Neural Network Quantization Survey ***Software:*** Intel Distiller
2/1	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 Nondeterministic Applications (ASPLOS 2018)
2/3	Quality-Aware Optimization Systems (1) ***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
2/8	**Submit Your Paper Choices: Link**		
2/8	Quality-Aware Optimization Systems (2) ***Background Read:*** Managing Performance vs. Accuracy Trade-offs With Loop Perforation (FSE 2011) ***Background Read:*** Best-Effort Parallel Execution Framework for Recognition and Mining Applications (IPDPS 2009)	Sasa Slides	***Additional Read:*** Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures (2009) ***Additional Read:*** Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation (PLDI 2010)
2/10	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)
2/15	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)
2/17	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)
2/22	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 List for Paper Discussion

2/24 Approximate Systems (1)
Primary Read: Input responsiveness: using canary inputs to dynamically steer approximation (PLDI 2016)
Secondary Read: ApproxIt: A Quality Management Framework of Approximate Computing for Iterative Methods (IEEE TCAD 2020) (Sections 1 and 2)

3/1 Approximate Systems (2)
Primary Read: Proactive control of approximate programs (ASPLOS 2016)
Secondary Read: JouleGuard: Energy Guarantees for Approximate Applications (SOSP 2015)

3/3 Approximate Systems (3)
Primary Read: Rigorous floating-point mixed-precision tuning (POPL 2017)
Secondary Read: Chisel: Reliability- and Accuracy-Aware Optimization of Approximate Computational Kernels (OOPSLA 2014)

3/8 Accuracy-Aware DNN Inference (1)
Primary Read: Deep Compression: Compressing Neural Networks With Pruning, Trained Quantization, and Huffman Coding (ICLR 2016)
Secondary Read: ApproxTuner: A Compiler and Runtime System for Adaptive Approximations (PPOPP 2021)

3/10 Accuracy-Aware DNN Inference (2)
Primary Read: Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism (ISCA 2017)
Secondary Read: What is the state of Neural Network Prunnig? (MLSYS 2020)

3/22 Accuracy-Aware DNN Inference (3)
Primary Read: The Lottery Ticket Hypothesis: Finding Small, Trainable Neural Networks (ICLR 2019)
Secondary Read: NPAS: A Compiler-aware Framework of Unified Network Pruning and Architecture Search for Beyond Real-Time Mobile Acceleration (CVPR 2021)

3/24 DNN Accuracy vs. Privacy
Primary Read: Shredder: Learning Noise Distributions to Protect Inference Privacy (ASPLOS 2020)
Secondary Read: Defensive approximation: securing CNNs using approximate computing (ASPLOS 2021)

3/29 Testing: Deep Learning Applications
Primary Read: DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars (ICSE 2018)
Secondary Read: Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks? (FSE 2020)

3/31 Testing: Probabilistic Applications
Primary Read: TERA: Optimizing Stochastic Regression Tests in Machine Learning Projects (ISSTA 2021)
Secondary Read: FLEX: Fixing Flaky Tests in Machine Learning Projects by Updating Assertion Bounds (FSE 2021)

4/5 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)

4/7 Testing and Analysis: Robustness
Primary Read: TBD (TBD)
Secondary Read: TBD (TBD)

4/19 Probabilistic Programming Systems (2)
Primary Read: Gen: a general-purpose probabilistic programming system with programmable inference (PLDI 2019)
Secondary Read: Pyro: Deep Universal Probabilistic Programming (JMLR 2018) (read full paper)

4/21 Probabilistic Programming Systems (3)
Primary Read: Reactive probabilistic programming (PLDI 2020)
Secondary Read: Scenic: a language for scenario specification and scene generation (PLDI 2019)

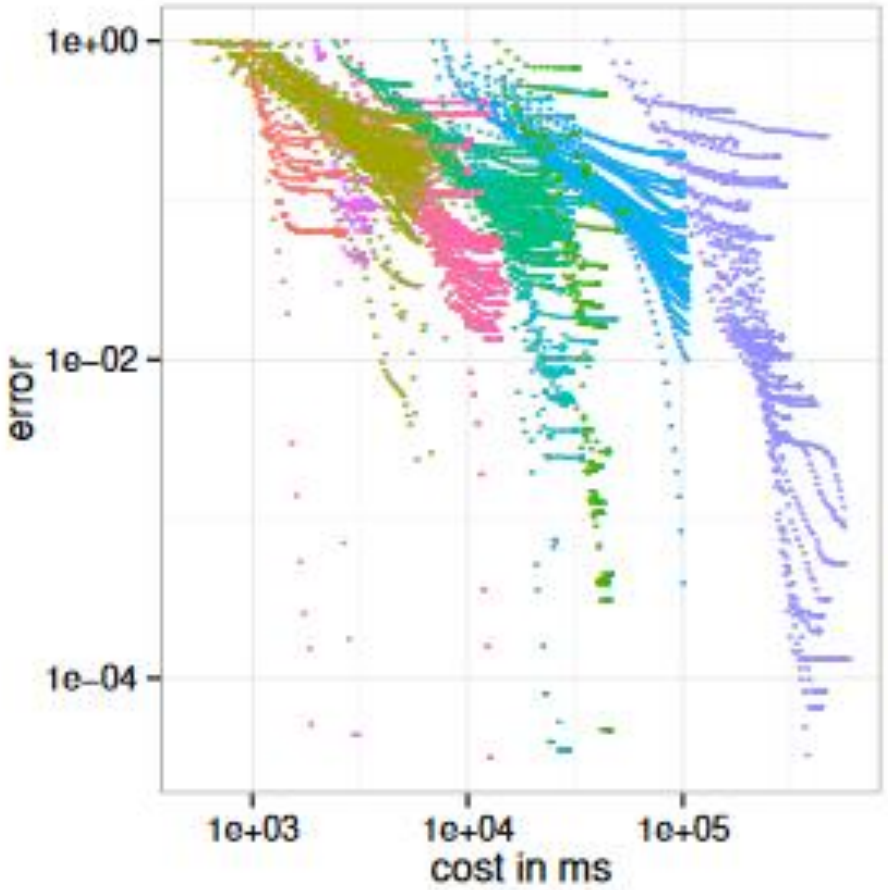
4/26 Probabilistic Programming Systems (4)
Primary Read: Probabilistic Programming with Densities in SlicStan: Efficient, Flexible and Deterministic (POPL 2019)
Secondary Read: Static Analysis for Probabilistic Programs: Inferring Whole Program Properties from Finitely Many Paths (PLDI 2013)

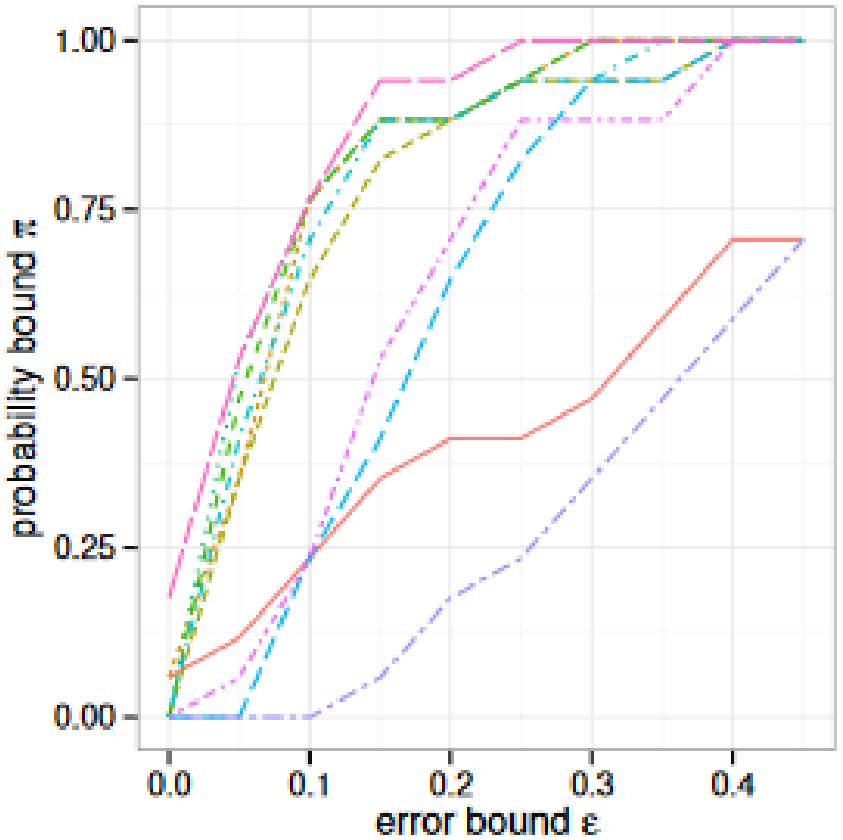
4/28 Probabilistic Programming Systems (5)
Primary Read: Transforming Probabilistic Programs for Model Checking (FODS 2020)
Secondary Read: FairSquare: probabilistic verification of program fairness (OOPSLA 2017)

5/3 Student Project Presentations All

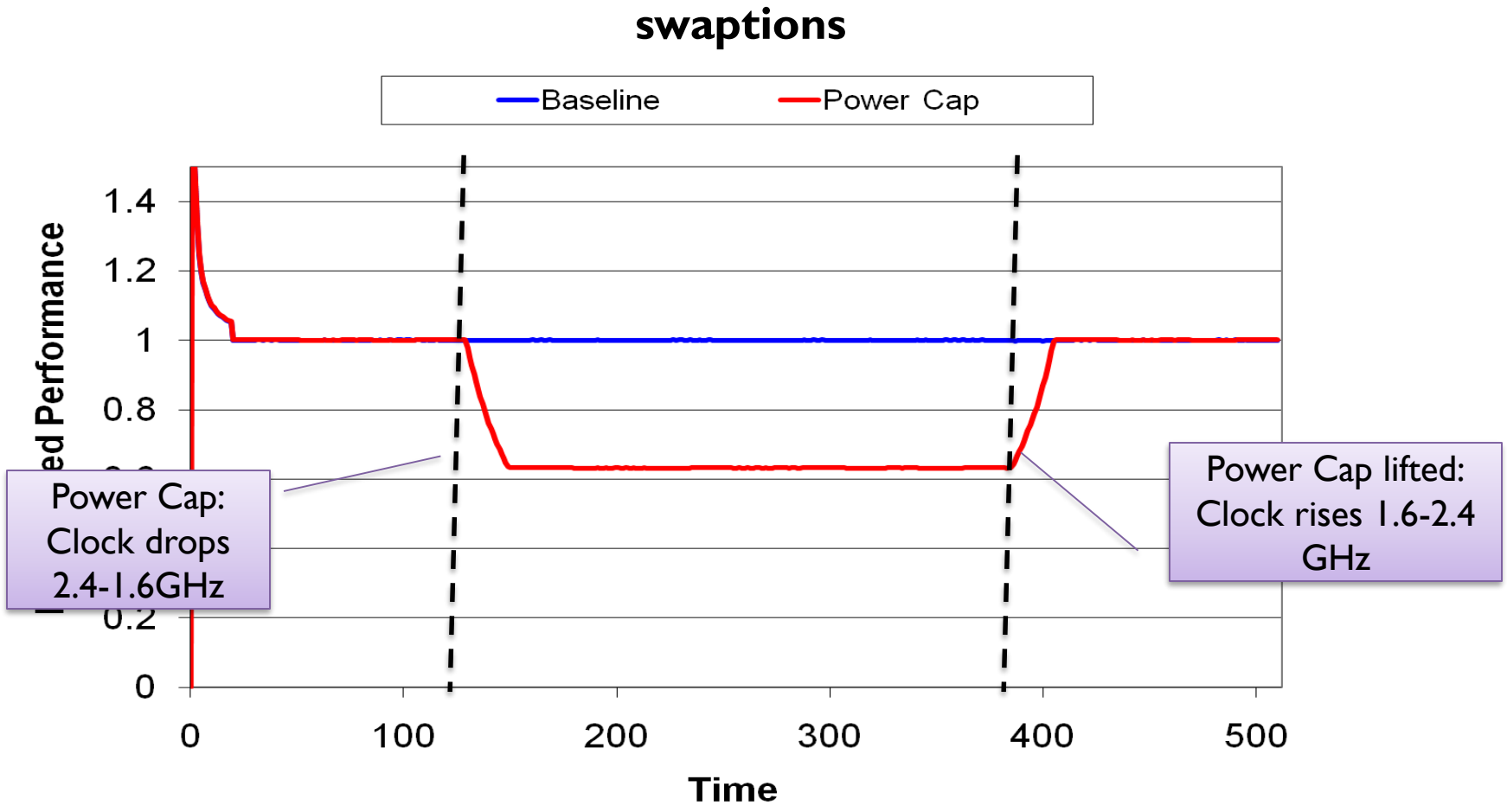
Systematically Exploring Tradeoffs of Approximation

Capri, ASPLOS 2016



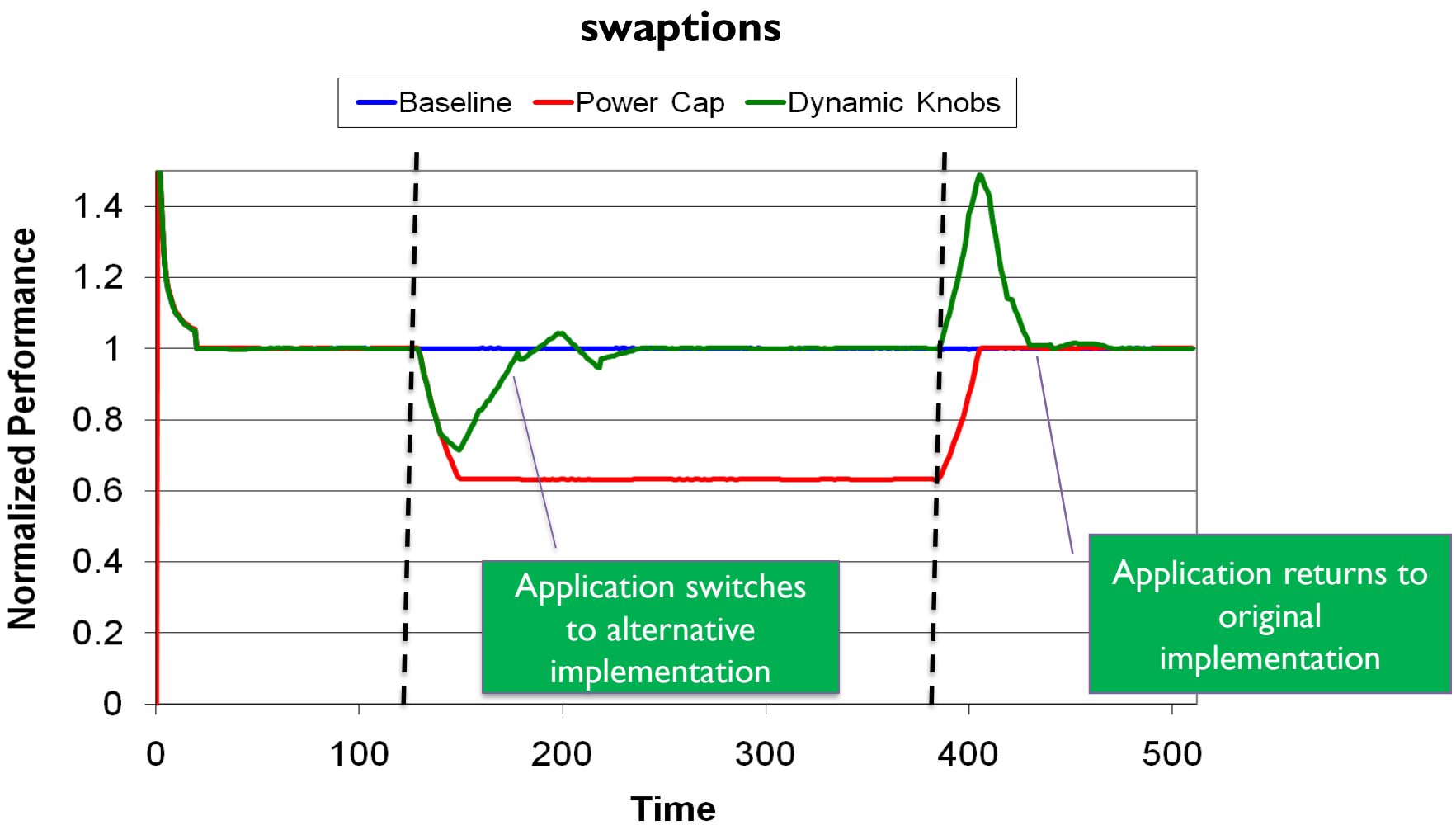


Study Dynamic Approximation



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

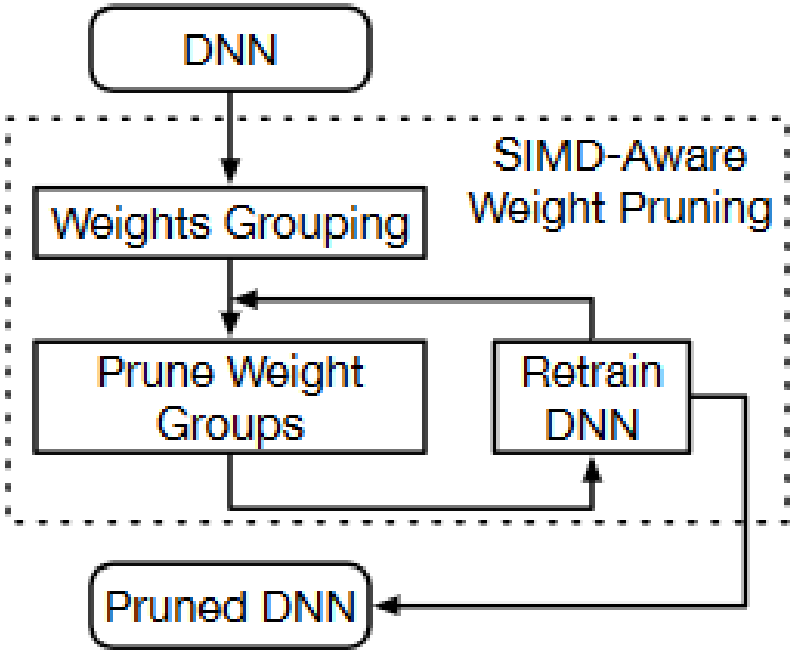


Figure 7: Main steps of SIMD-aware weight pruning.

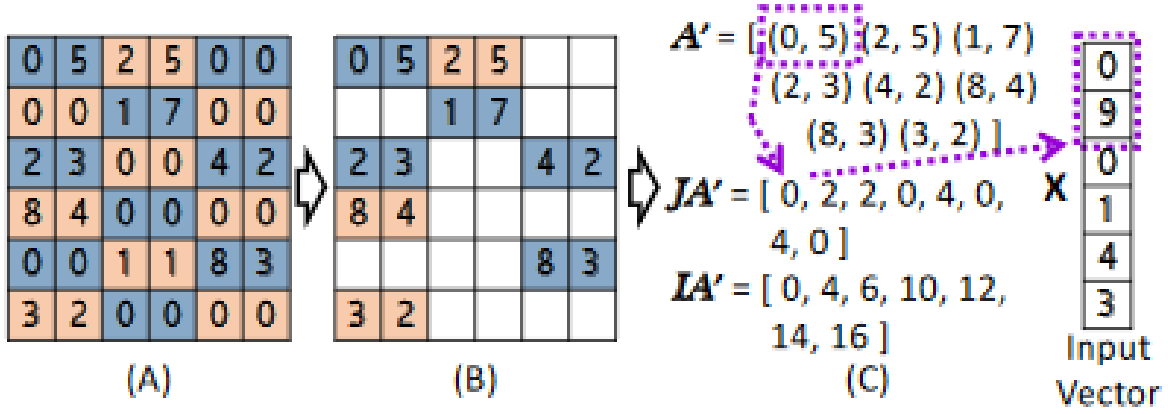
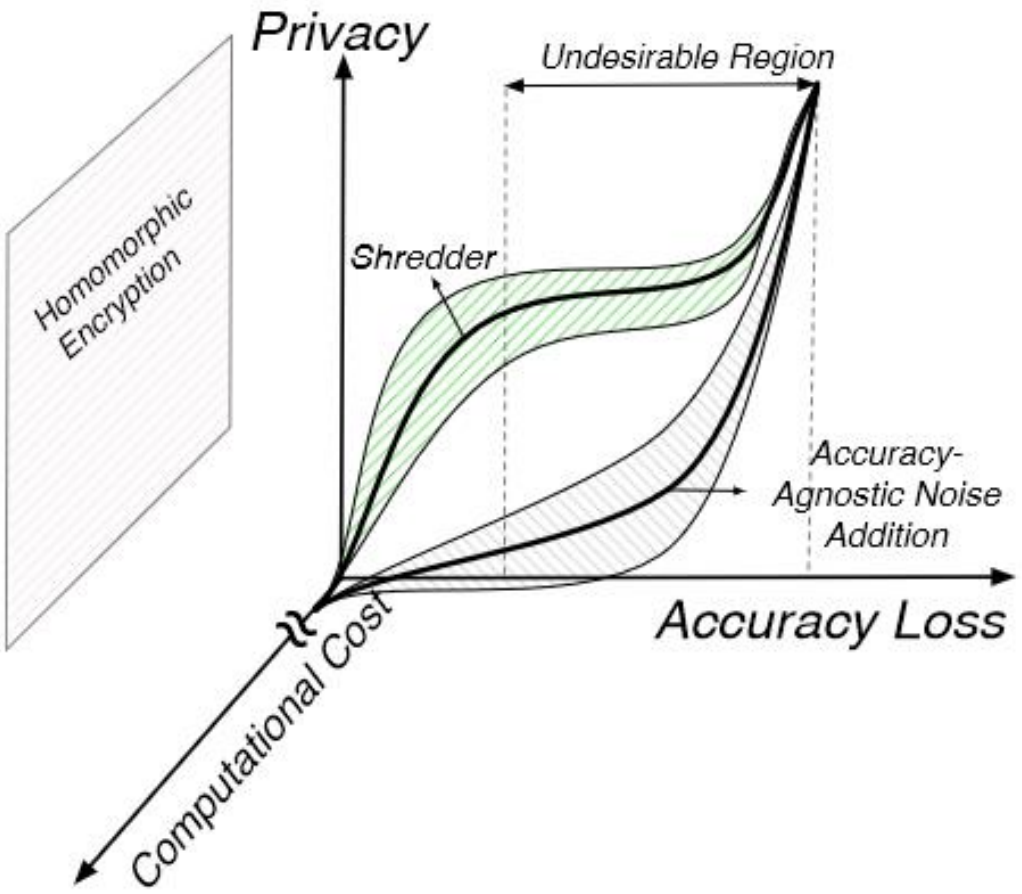


Figure 8: (A) Weights grouping; (B) Sparse weight matrix after pruning weight groups; (C) Modified CSR format for SIMD-aware weight pruning.

Trading off Accuracy, Performance and Privacy

Shredder,ASPLOS 2020



Impacts of Approximate Answers on Decision-Making

DeepTest, ICSE 2018

1.1 original

1.2 with added rain

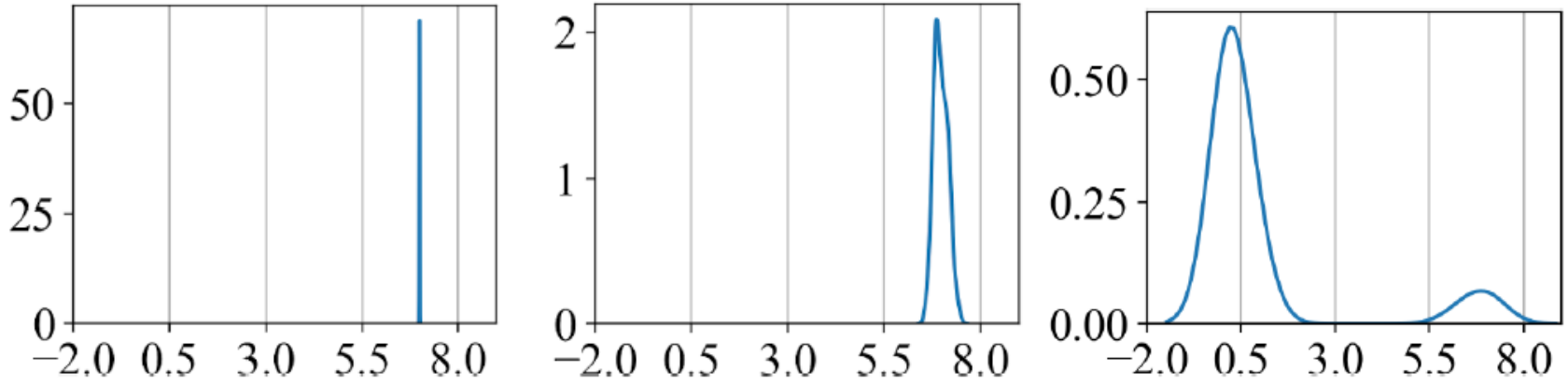
Figure 1: A sample dangerous erroneous behavior found by DeepTest in the *Chauffeur* DNN.

Testing Probabilistic Applications

Probfuzz/Flex/Tera

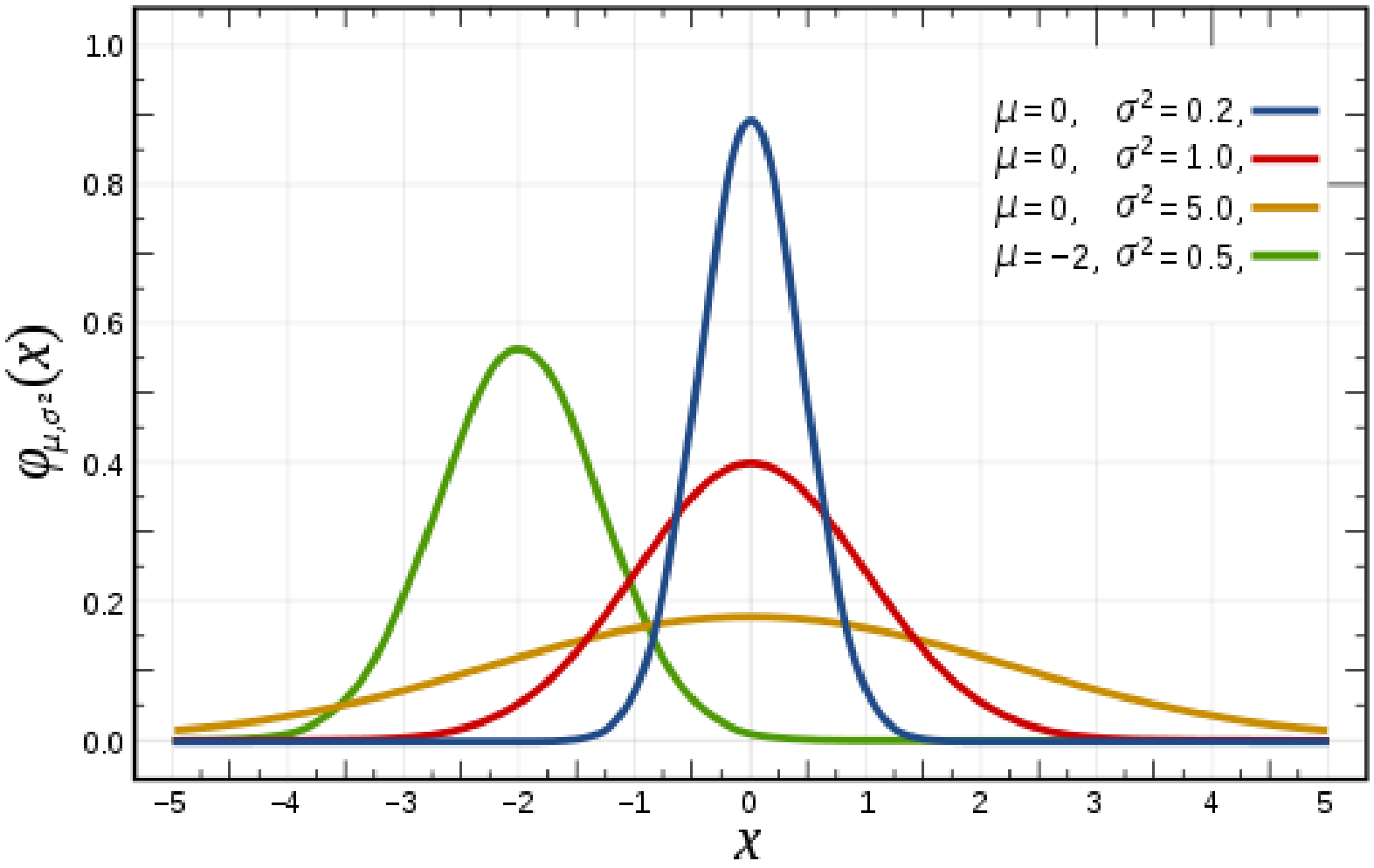
```
1 x̄ = [1.0, 2.0, ...]
2 ȳ = [7.0, 14.0, ...]
3 w = Gamma(97.5, 86.2)
4 p = Beta(44.0, 44.0)
5 observe(
6   Normal(w · x̄, p), ȳ)
7 posterior(w)
```

Probabilistic program



Edward **Pyro** **Stan**

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

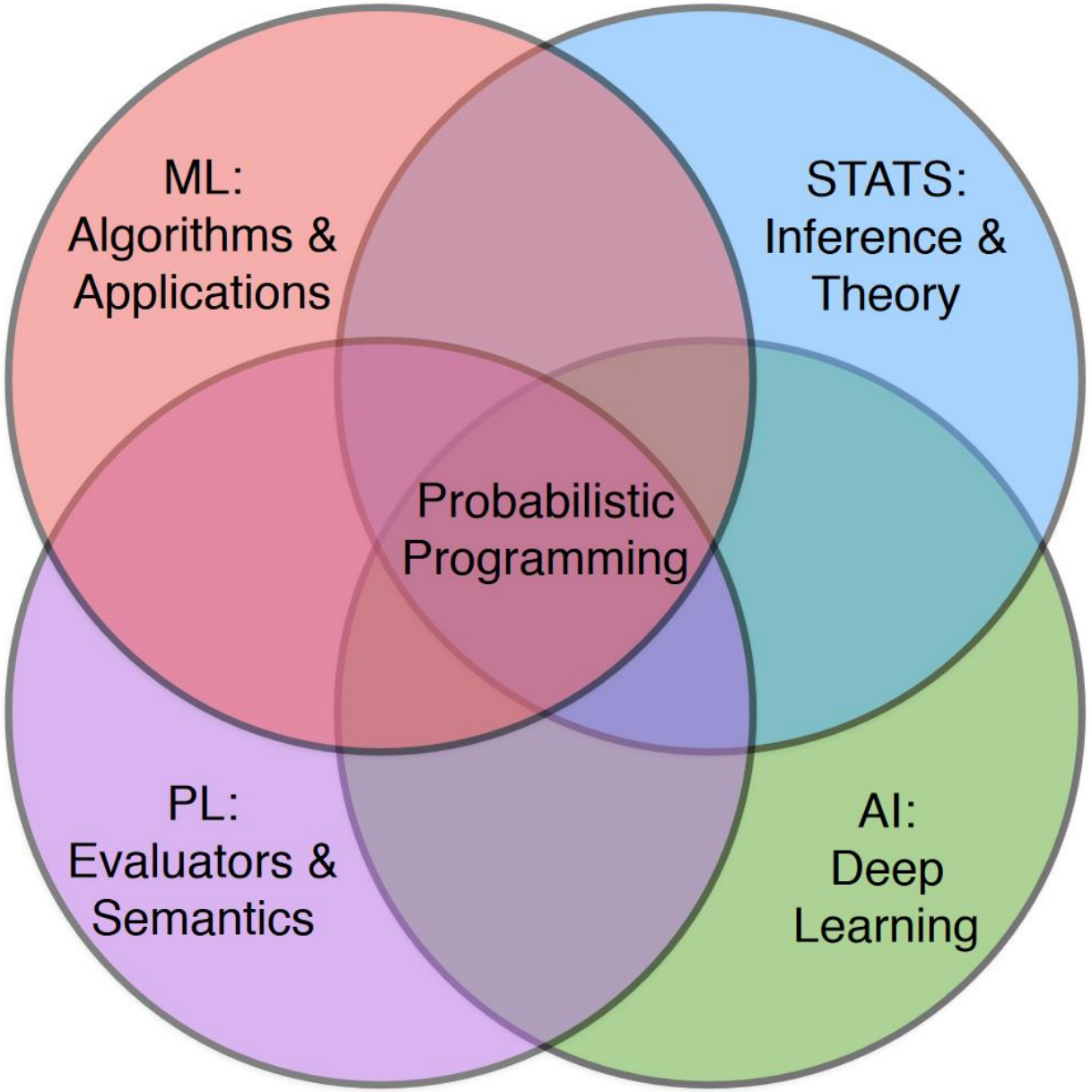


Diagram by Frank Wood

CS 521

COURSE LOGISTICS

Schedule

Twice a week – Tuesdays and Thursdays 3:30-4:45

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

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…

Miss a lecture, lecture cancelled or postponed or online

Bad Internet connection, Zoom, other sites, 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	20%
Project	50%
Homework	10% + 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 **February 10**

- Link: http://misailo.web.engr.Illinois.edu/courses/521/
- 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 2nd 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 **first half of March (Date TBD)**

- 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	20%
Project	50%
Hon	10% + 10% XC

Grading on an absolute scale (no curve!)

CS 521

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?