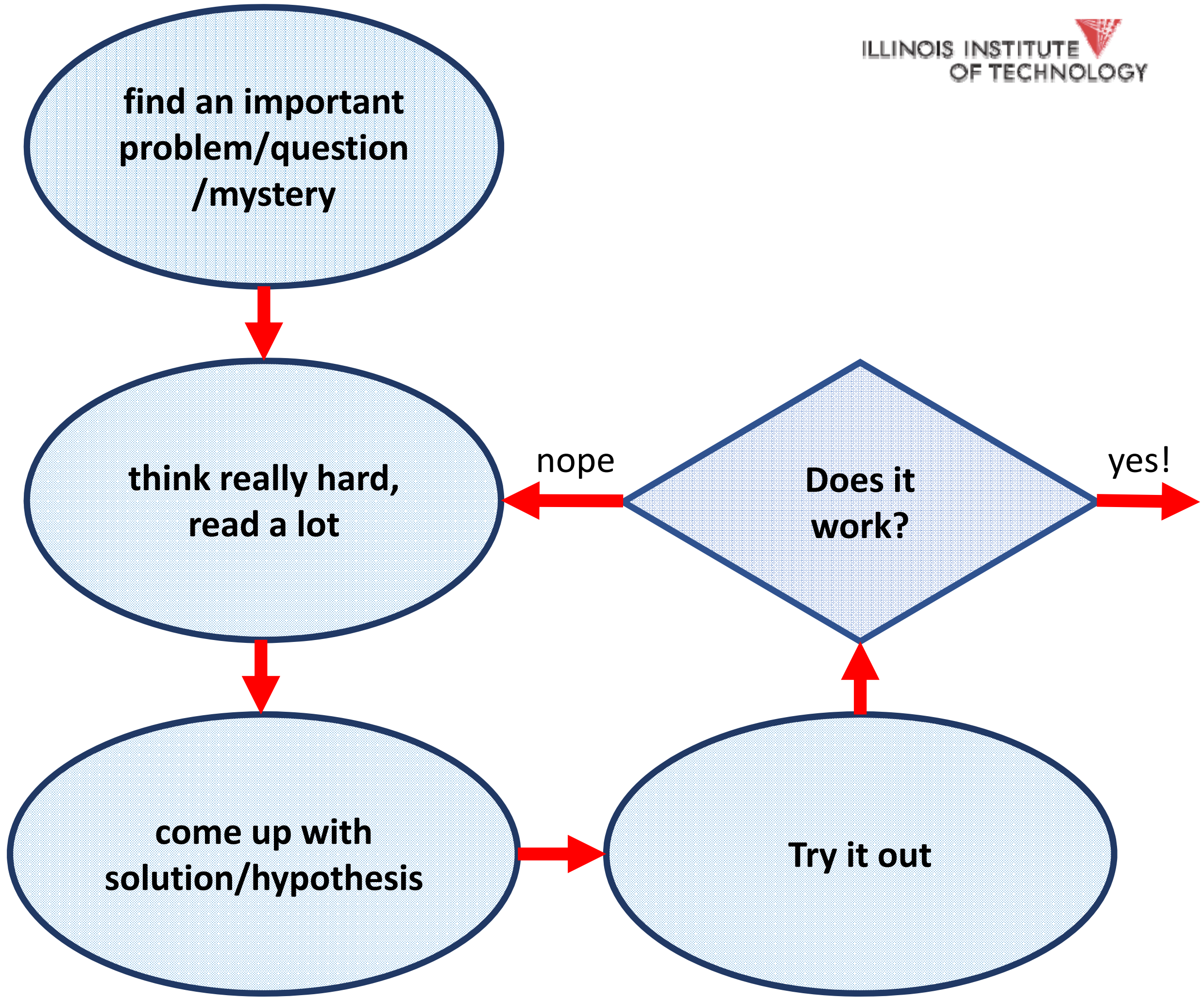
Undergraduate Research @ IIT

Kyle C. Hale

What is research?



ILLINOIS INSTITUTE OF TECHNOLOGY

ASPLOS Submission #289 – Confidential Draft – Do Not Distribute!!

ASC: Automatically Scalable Computation

Abstract

We present an architecture designed to transparently and automatically scale the performance of sequential programs as a function of the hardware resources available. The architecture is predicated on a model of computation that views program execution as a walk through the enormous state space composed of the memory and registers of a single-threaded processor. Each instruction execution in this mode… system from one state to a deterministically-detern… quent state. We can parallelize such execution by … partitioning the complete path and speculativel… each partition in parallel. Accurately partitioning… a challenging prediction problem. We have imple… system in an architectural simulator of an x86 pro… a collection of state predictors and a mechanism … tively executing threads that explore potential stat… execution path. We demonstrate that we can achi… of 256 speedup on 1024 cores while running an … sequential program.

1. Introduction

The primary design challenge in realizing this architecture is accurately predicting points that partition a trajectory. We break this challenge into two parts: (1) recognizing states from which accurate prediction is possible and will result in useful speedup, and (2) predicting future states of the system when the current state of execution is recognized as one from which prediction is possible.

230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

rejoice! and tell the whole world!

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable

PROC. OF THE IEEE, NOVEMBER 1998 1

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract— **Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.**

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

I. INTRODUCTION

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images.

A Case for Redundant Arrays of Inexpensive Disks (RAID)

David A Patterson, Garth Gibson, and Randy H Katz

Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720

1 Background: Rising CPU and Memory Performance

What does it involve?

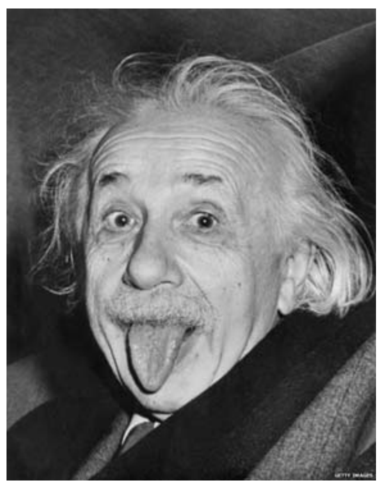
- Thinking (creatively, critically, sometimes wrongly, sometimes rightly) *a lot
- Reading *a lot
- Learning *a lot
- Questioning *a lot
- Brainstorming/Collaborating *a lot
- Experimenting *a lot
- Programming (these days in most fields, esp. CS) *a lot
- Writing *a lot
- Challenging yourself *a lot
- Growing *a lot
- Traveling, meeting interesting people *a lot
- Joining communities
- Having fun! *a lot

What *isn't* research?

- Boredom
- Someone telling you exactly what to do
- Just another 3 credit hours
- Working on a problem someone else has the answer for
- A 9-5 job

Why should I care?

- It's ***fun***! You have the ***freedom*** to explore
- You get to be on the ***cutting edge***. Your path to becoming an expert
- You can ***contribute something*** to the world, creating new knowledge
- Leads to flexible, fulfilling, and well-paying ***careers***



What about research *in CS*?

- One of the most important and game-changing fields currently
- Technology ***is everywhere***, it’s growing, it ain’t slowing down, and it’s all **driven by computing**
- A lot of open problems, challenges
- Fast-moving: things change quickly, will never be bored!

CS Research Areas (broadly construed)

- **AI, machine learning, data science**
- **Systems and Networking**
- **Theory**
- **Programming Languages**
- **CS + X**

What does it take to do research?

- **Curiosity**: If you don't ask questions, you won't go exploring
- **Drive + Passion**: You must genuinely *care* if you are to make progress
- **Creativity**: It's not just for the arts! Some of the best research involves new and original ways of rethinking old problems
- **Courage**: You have to learn that **failure is both normal and okay**
  - **This is, of course, *unlike* your classes, where you should definitely not fail**
- **Dedication:** The best researchers pick a problem and work hard on it

Notice what I didn't list…

- **Genius/Talent**: Overrated; it sometimes helps, but **curiosity, passion, and perseverance win out big-time**
- **Straight A's**: The collective sphere of human knowledge couldn't care less about your GPA
  - But *it does* help a professor determine how serious you are about learning…and therefore whether to let you join the lab…so don't ignore them ☺
- **Knowledge**: I'm more interested in working with people who are **eager and willing to learn** more than with those who already know a lot

Career Paths

- **After undergrad -> grad school (PhD)**
  - Yes, it's more school, but it's ***not the same***
- **Research Scientist/Engineer**
  - National Labs/Government entities
  - Think Tanks
  - Industry
- **Academic (Professor)**
  - Research + Teaching usually, sometimes just one
- **Consulting**

Opportunities in the Department

- **Informal arrangements**: Generally professors are willing to let you work with them if you're hard-working and motivated
  - e.g., take CS 497 with a prof (independent study)
  - We sometimes have ad hoc summer (or regular semester funding) to pay you for it
- **CS Honors Research Specialization**: **this is new**! Perform research as you progress through ugrad, write and present thesis at end. Ideally, publish! Talk to us more if you have ??s
- Research Experience for Undergrad – paid (more on this after)
  - At another institution
  - Here at IIT

OK, sounds great; how do I start?

- **Be engaged in class!**
  - Ask *a lot* of questions; **BE CURIOUS.** Don't be afraid to look dumb!
  - Sometimes seemingly obvious questions lead to adventures
  - Do these things and **we will for sure notice!**
- **Get to know your professors**
  - We love talking about research! (we're nerds)
  - Don't be intimidated, we're just people
  - **We will make time for you**
- Find out what you're passionate about, and **approach one of us about doing research**

Important questions in CS

Systems (not exhaustive)

- How to deal with explosion of data (“big data”)?
- How to make extreme amounts of data easy to store, easy to analyze, and easy to compose *efficiently*? -> database systems
- How to make our systems secure?
- How to tame complexity of systems?
- How to design better chip architectures to meet todays challenges?
- How to handle growth of heterogeneous hardware?
- How to make systems *scalable*? (1000s of cores, 1000s of machines)
- How to make systems reliable in the face of failure?

Theory (not exhaustive)

- How to design efficient algorithms for large swaths of (changing) data?
- How to use algorithms to predict behavior and inform decisions?
- Improving computational efficiency across the board (e.g. graph isomorphism, SAT, bin packing, scheduling, etc.)
- P = NP? (Just how hard are certain classes of problems? Can we do better?)
- Someone else would know better ☺

AI + Machine Learning + Data Science (not exhaustive)

- How to build (and understand) intelligent machines
  - That have same capabilities of human intelligence: strong AI
  - That will help make our lives better: increase pattern matching, insight abilities of machines
- How to help machines make insights on large amounts of data?
- How to help machines effectively interact with and understand humans? e.g. Natural language processing, knowledge representation, automated reasoning, problem solving
- How to help machines understand the world and interpret data? e.g. computer vision, knowledge representation, etc.
- How to make sure they AI makes positive (and ethical) impact on humanity
- How to make intelligent machines efficient?
- Transparent machine learning: how do we understand *why* and *how* a machine did what it did?
- Are thinking machines conscious? How is our brain different? Philosophy of mind

Programming Languages and Compilers (not exhaustive)

- How do we make it easier to write efficient programs?
- How do we make compilers generate better code?
- How do we automatically port code from one architecture to another?
- What are the right abstractions to current (and new) machines?
- How to create languages that prevent (or at least discourage) bugs?
- How to make programs more secure?
- How to verify that a program does what we think? (verification)
- How to apply tools like machine learning to compilation and program verification?