

# Introduction to CS & AI

**Tao LIN**

September 2, 2025



- 1 General Introduction
  - How to Become a Good AI Researcher

# Table of Contents

- 1 General Introduction
  - How to Become a Good AI Researcher

Indeed “introduction to CS & AI” DOES NOT indicate that  
we will go through all basic CS & AI materials step-by-step in this course!

# If you are still unfamiliar with

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

# If you are still unfamiliar with

- programming, e.g., Python

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

# If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

# If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>



## If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

## If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

## If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs<sup>1</sup>.

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

## If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs<sup>1</sup>.
- UC Berkeley, CS 61B: Data Structures<sup>2</sup>.

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

## If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs<sup>1</sup>.
- UC Berkeley, CS 61B: Data Structures<sup>2</sup>.
- (Even) UC Berkeley, CS 61C: Great Ideas in Computer Architecture (Machine Structures)<sup>3</sup>.

---

<sup>1</sup><https://inst.eecs.berkeley.edu/~cs61a/sp22/>

<sup>2</sup><https://sp23.datastructur.es/>

<sup>3</sup><https://cs61c.org/fa23/>

# The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

## Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)



Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- 1/13/20: [Course overview + the shell](#)
- 1/14/20: [Shell Tools and Scripting](#)
- 1/15/20: [Editors \(Vim\)](#)
- 1/16/20: [Data Wrangling](#)
- 1/21/20: [Command-line Environment](#)
- 1/22/20: [Version Control \(Git\)](#)
- 1/23/20: [Debugging and Profiling](#)
- 1/27/20: [Metaprogramming](#)
- 1/28/20: [Security and Cryptography](#)
- 1/29/20: [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling
- Virtual machines

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)

Build your research weapons and wheels

*Check it out:* <https://missing.csail.mit.edu/>

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling
- Virtual machines
- Security

## The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the [motivation behind this class](#).

### Schedule

- **1/13/20:** [Course overview + the shell](#)
- **1/14/20:** [Shell Tools and Scripting](#)
- **1/15/20:** [Editors \(Vim\)](#)
- **1/16/20:** [Data Wrangling](#)
- **1/21/20:** [Command-line Environment](#)
- **1/22/20:** [Version Control \(Git\)](#)
- **1/23/20:** [Debugging and Profiling](#)
- **1/27/20:** [Metaprogramming](#)
- **1/28/20:** [Security and Cryptography](#)
- **1/29/20:** [Potpourri](#)



# Advanced CS: System, Theory, and AI

## System

- Computer systems

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems



# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

# Advanced CS: System, Theory, and AI

## System

## Theory

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra



# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning



# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning



# Advanced CS: System, Theory, and AI

System	Theory	AI
<ul style="list-style-type: none"><li>• Computer systems</li><li>• Distributed systems</li><li>• Network systems</li><li>• Embedded systems</li><li>• Compiler</li><li>• Software security</li><li>• System-on-chip</li><li>• Internet-of-Things systems</li><li>• etc</li></ul>	<ul style="list-style-type: none"><li>• Distributed algorithms</li><li>• Cryptography and security</li><li>• Formal verification</li><li>• Computational complexity</li><li>• Information theory and coding</li><li>• Computational linear algebra</li><li>• Statistical theory</li><li>• Quantum information theory and computation</li><li>• Dynamical system theory</li><li>• Advanced probability theory</li><li>• etc</li></ul>	<ul style="list-style-type: none"><li>• Statistical Machine Learning</li><li>• Deep Learning</li><li>• Learning theory</li><li>• Optimization for Machine Learning</li><li>• Multi-agents</li><li>• Computational photography</li><li>• Computer vision</li><li>• Virtual reality</li><li>• Computational neurosciences</li><li>• Reinforcement learning</li><li>• Online learning in games</li></ul>

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning
- Online learning in games
- Natural Language Processing

# Advanced CS: System, Theory, and AI

## System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

## Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

## AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning
- Online learning in games
- Natural Language Processing
- etc

# Table of Contents

- 1 General Introduction
  - How to Become a Good AI Researcher

# How to Become a Good AI Researcher

# How to Become a Good AI Researcher

- Master math and CS skills

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)



# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk
  - etc

# How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- <https://csdiy.wiki/>
- books e.g., [Algebra](#), [Topology](#), [Differential Calculus](#), and [Optimization Theory For CS and ML](#)

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk
  - etc
- Practice these skills and make steady progress in your research

# Course Schedule

Week	Date	Topics
1	2025. Sep. 02	How to communicate
2	2025. Sep. 09	How to do presentation
3	2025. Sep. 16	How to be a good AI researcher (I): doing research I
4	2025. Sep. 23	How to be a good AI researcher (II): productivity and career
5	2025. Sep. 30	How to be a good AI researcher (III): academic paper writing and peer reviews
6	2025. Oct. 14	Sharing the experience of writing excellent academic papers and rebuttal
7	2025. Oct. 21	Practice course I
8	2025. Oct. 28	Practice course II

# How to communicate

- 1 A General Guide
  - Why Communication Matters?
  - The 7 **C**'s of Communication
- 2 How to Communicate With Your Collaborator?
  - How to Work With Your Advisor Effectively
  - How to Share Progress With Your Mentors/Collaborators?
  - How to Work With a Busy Advisor?
  - How to Work With Your Senior Advisor(s)?
- 3 How to Ask Questions The Smart Way (From CS Perspective)?
  - Before You Ask
  - When You Ask
- 4 How to Do Presentation

# How to do presentation

- 1 Reminder: Principle of Effective Communication
- 2 How to Present—A General Guideline
  - A General Guide
  - Before the Talk / Preparing Your Talk
  - The Beginning of the Talk
  - The Body of the Talk
  - The End of the Talk
- 3 Others
  - How to Handle Questions in a Presentation?
  - How to Present a Line Plot?
  - How to Make a Research Poster?
  - How to Present a Poster at a Conference?
  - How to Present a Paper in Theoretical Computer Science: A Speaker's Guide for Students?



# How to be a good AI researcher (I): doing research I

- 1 Course Logistics
- 2 Recitation
- 3 How to Do Research
  - The Illustrated Guide to a Ph.D.
  - 10 Easy Ways to Fail a Ph.D.
  - How to Make Steady Progress?
  - How to Keep Track With the Literature?
  - How to Read Papers?
  - How to Come up With Research Ideas?
  - How to Do Experiments?
  - How to Create More Impact
- 4 Concluding Remarks

# How to be a good AI researcher (II): productivity and career

## 1 Recitation

## 2 How to Do Research

- More on How to Read Papers
- 12 Resolutions for Grad Students
- How to Manage Your Time?
- How to Be Productive?
- Tips for Work-Life Balance (WLB)
- Others Career Tips

# How to be a good AI researcher (III): academic paper writing and peer reviews

- 1 How to Write a Great Research Paper?
  - A General Guideline
  - How to Write the Introduction?
  - How to Write Papers That Are Easy to Read?
  - Tips to Create a Good Table
- 2 How to Write a Rebuttal for a Conference?
- 3 Summary

# Parallel course

## GAMES003: 科研基本素养

2024 年秋季学期（在线直播）

### 课程介绍



本课程为初学者展示了一条全面的学术研究路径，旨在引领大家以系统性的方法探索计算机视觉和图形学领域的科学前沿。我们将指导大家从建立领域视野，到选择科研课题、设计技术方案，再到设计实验、优化方案、管理论文投稿、设计论文图表、撰写论文、自我评审与rebuttal，以及学习做学术报告的技巧，覆盖了科研过程中的每一个关键步骤。课程中，我们将结合具体案例，分享科研经验，同时鼓励学生提出问题，以实现具象的科研素养教学。

具体课程内容请参见课程大纲。

# Thanks & Question Time!