Education Committee Meeting
April 24, 2020
3:00 PM via Zoom

Zoom link:
https://umd.zoom.us/j/96542027185?pwd=bHpmVjBINzBXYVd2cFhta05GdXFRZz09
Meeting ID: 965 4202 7185
Password: 175311

Agenda:

- Updated course list for the Machine-Learning Specialization – Jacobs (REQUIRES VOTE)

- Convert special-topic courses to regular offerings:
  - 498L: Introduction to Deep Learning – Shrivastava
  - 498F: Robotics and Perception – Tokekar
  - 498P: Introduction to Deep Learning – Goldstein

527: COMPUTER SCIENCE MAJOR

History

1. Apr 10, 2020 by Apitchaya Pimpawathin (apimpawa)

Changes saved but not submitted
Viewing: 527 : Computer Science Major
Last approved: Fri, 10 Apr 2020 17:14:27 GMT
Last edit: Tue, 21 Apr 2020 14:11:33 GMT

Proposed Action

Curriculum Change

Program Name

Computer Science Major

Program Status

Active

Effective Term

Fall 2020

Catalog Year

2020-2021

Program Level

Undergraduate Program

Program Type

Undergraduate Major

Delivery Method

On Campus

Departments

Department
Computer Science

Colleges

College
Computer, Mathematical, and Natural Sciences

Program/Major Code

07010, 0701B, 0701C, 0701D

MHEC Inventory Program

Computer Science

Degree(s) Awarded

Degree Awarded
Bachelor of Science

Proposal Contact

Apitchaya Pimpawathin; apimpawa@umd.edu

Program and Catalog Information

Provide the catalog description of the proposed program. As part of the description, please indicate any areas of concentration or specializations that will be offered.

Computer science is the study of computers and computational systems: their application, design, development and theory. Principal areas within computer science include machine learning and data science, cybersecurity and privacy, human-computer interaction, artificial intelligence, programming languages, software engineering, computer systems and networking, algorithms and theory of computing, natural language processing, high-performance and quantum computing, databases systems, bioinformatics, robotics, computer vision, information visualization, and virtual- and augmented-reality systems. A computer scientist is concerned with problem solving. Problems range from abstract questions of what problems can be solved with computers to practical matters such as the design of computer systems that are efficient, secure, and easy for people to use.

Catalog Program Requirements:

Much of the knowledge at the early stage of the degree program is cumulative. To ensure that transfer students start with the appropriate courses, the department offers exemption exams for CMSC131, CMSC132, CMSC216 and CMSC250. Students who have had CS courses prior to starting at Maryland are encouraged to schedule and take exemption exams.

A 'C-' or better must be earned in all major requirements.

Course	Title	Credits
Required Lower Level Courses (Unless Exempt)		
MATH140	Calculus I (see your advisor)	4
MATH141	Calculus II	4
CMSC131	Object-Oriented Programming I [1]	4
CMSC132	Object-Oriented Programming II [1]	4
CMSC216	Introduction to Computer Systems [1]	4
CMSC250	Discrete Structures [1]	4
Additional Required Courses		
CMSC330	Organization of Programming Languages	3
CMSC351	Algorithms	3
STAT4xx [2]		3
MATH/AMSC/STAT xxx [2]		3-4
Upper Level Computer Science Courses [3]		
Select five 400 level courses from at least three of the following areas with no more than three courses in a given area:		15
Area 1: Systems		
CMSC411	Computer Systems Architecture	
CMSC412	Operating Systems	
CMSC414	Computer and Network Security	
CMSC417	Computer Networks	
Area 2: Information Processing		
CMSC420	Advanced Data Structures	
CMSC421	Introduction to Artificial Intelligence	
CMSC422	Introduction to Machine Learning	
CMSC423	Bioinformatic Algorithms, Databases, and Tools	
CMSC424	Database Design	
CMSC426	Computer Vision	
CMSC427	Computer Graphics	
CMSC470	Introduction to Natural Language Processing	
Area 3: Software Engineering and Programming Languages		
CMSC430	Introduction to Compilers	
CMSC433	Programming Language Technologies and Paradigms	
CMSC434	Introduction to Human-Computer Interaction	
CMSC435	Software Engineering	
CMSC436	Programming Handheld Systems	
Area 4: Theory		
CMSC451	Design and Analysis of Computer Algorithms	

CMSC452	Elementary Theory of Computation	
CMSC456	Cryptography	
CMSC457	Introduction to Quantum Computing	
Area 5: Numerical Analysis		
CMSC460	Computational Methods [4]	
or CMSC466	Introduction to Numerical Analysis I	
Upper Level Concentration Requirement [5]		
Select at least 12 credits of 300-400 level courses from one discipline outside of CMSC		12
Total Credits		63-64

Students also have the option to complete the Cybersecurity Specialization **(**https://academiccatalog.umd.edu/#cyber**),** Data Science Specialization **(**https://academiccatalog.umd.edu/#data**), or Machine Learning Specialization**

1 Students may fulfill CMSC131, CMSC132, CMSC216 or CMSC250 course requirements by passing proficiency exams before they start the sequence of classes.

2 This course must have prerequisite of MATH141 or higher; cannot be cross-listed with CMSC.

3 At the upper level, students take five (5) 400 level courses from at least three different areas with no more than three courses in a given area. An additional two (2) electives, totaling 6 credits, for the general computer science degree are also required. If students take more than three courses from an area, they will be counted as electives. Students can count one credit winter courses towards the elective requirement, as well as independent research or study with a faculty member, and other courses at the 300 or 400 level.

4 Credit will only be given for CMSC460 or CMSC466.

5 Students must also take at least 12 credits of 300-400 level courses from one discipline outside of CMSC. No course in or cross-listed with CMSC can be counted. An overall 2.0 average must be earned in these courses. Each course must be a minimum of 3 credits. Only 1 special topics or independent study course may be used.

Cybersecurity Specialization

Students looking to pursue the cybersecurity specialization are required to complete the lower level courses (MATH140, MATH141, CMSC131, CMSC132, CMSC216, CMSC250), the additional required courses (CMSC330, CMSC351, MATH/STATXXX and STAT4xx beyond MATH141), and the upper level concentration requirements as detailed above. The difference in the specialization is the upper level computer science courses. Students must fulfill their computer science upper level course requirements from at least 3 areas.[1]

Students are required to take:

Course	Title	Credits
CMSC414	Computer and Network Security	3
CMSC456	Cryptography	3
Students must choose four courses from:		12-13
CMSC411	Computer Systems Architecture	
CMSC412	Operating Systems	
CMSC417	Computer Networks	
CMSC430	Introduction to Compilers	
CMSC433	Programming Language Technologies and Paradigms	
CMSC451	Design and Analysis of Computer Algorithms	
Upper Level Elective Courses: three credits from CMSC3XX or CMSC4XX excluding CMSC330 and CMSC351 [1]		3
Total Credits		21-22

1 Students may fulfill an area requirement under the Upper Level Elective Courses requirement. Courses that fall within each area are listed in the General Track degree requirements. The five areas are: Area 1: Systems, Area 2: Information Processing, Area 3: Software Engineering and Programming Languages, Area 4: Theory, and Area 5: Numerical Analysis.

Data Science Specialization

Students looking to pursue the data science specialization are required to complete the lower level courses (MATH140, MATH141, CMSC131, CMSC132, CMSC216, CMSC250), the additional required courses (CMSC330, CMSC351, STAT400 and MATH240), and the upper level concentration requirements as detailed above. The difference in the specialization is the upper level computer science courses. Students must fulfill their computer science upper level course requirements from at least 3 areas.[1]

Students are required to take:

Course	Title	Credits
CMSC320	Introduction to Data Science	3
CMSC422	Introduction to Machine Learning	3

CMSC424	Database Design	3
Select one of the following:		3
CMSC402	Bioinformatic Algorithms and Methods for Functional Genomics and Proteomics	
CMSC420	Advanced Data Structures	
CMSC421	Introduction to Artificial Intelligence	
CMSC423	Bioinformatic Algorithms, Databases, and Tools	
CMSC425	Game Programming	
CMSC426	Computer Vision	
CMSC427	Computer Graphics	
CMSC470	Introduction to Natural Language Processing	
Select one of the following:		
CMSC451	Design and Analysis of Computer Algorithms	3
or CMSC460	Computational Methods	
Select two of the following:		6-7
CMSC411	Computer Systems Architecture	
CMSC412	Operating Systems	
CMSC414	Computer and Network Security	
CMSC417	Computer Networks	
CMSC430	Introduction to Compilers	
CMSC433	Programming Language Technologies and Paradigms	
CMSC434	Introduction to Human-Computer Interaction	
CMSC435	Software Engineering	
Total Credits		21-22

[1] Courses that fall within each area are listed in the General Track degree requirements. The five areas are: Area 1: Systems, Area 2: Information Processing, Area 3: Software Engineering and Programming Languages, Area 4: Theory, and Area 5: Numerical Analysis.

Machine Learning Specialization

Students looking to pursue the machine learning specialization are required to complete the lower level courses (MATH140, MATH141, CMSC131, CMSC132, CMSC216, CMSC250), the additional required courses (CMSC330, CMSC351, STAT4xx beyond MATH141, and MATH240), and the upper level concentration requirements as detailed above. The difference in the specialization is the upper level computer science courses. Students must fulfill their computer science upper level course requirements from at least 3 areas.[1]

Students are required to take:

Course	Title	Credits
CMSC320	Introduction to Data Science	3
CMSC421	Introduction to Artificial Intelligence	3
CMSC422	Introduction to Machine Learning	3
Select two of the following:		6
CMSC426	Computer Vision	
CMSC/AMSC460	Computational Methods	
or CMSC/AMSC466	Introduction to Numerical Analysis I	
or MATH401	Applications of Linear Algebra	
CMSC470	Introduction to Natural Language Processing	
CMSC474	Introduction to Computational Game Theory	
CMCS498F	Course CMCS498F Not Found (CMSC498F: Robotics and Perception)	
CMSC498L	Course CMSC498L Not Found (CMSC498L: Introduction to Deep Learning)	
CMSC498P	Course CMSC498P Not Found (CMSC498P: Machine Learning Capstone)	
Upper Level Elective Courses: six credits from CMSC3XX or CMSC4XX excluding CMSC330 and CMSC351 [1]		6
Total Credits		21

[1] Students may fulfill an area requirement under the Upper Level Elective Courses requirement. Courses that fall within each area are listed in the General Track degree requirements. The five areas are: Area 1: Systems, Area 2: Information Processing, Area 3: Software Engineering and Programming Languages, Area 4: Theory, and Area 5: Numerical Analysis.

Sample plan. Provide a term by term sample plan that shows how a hypothetical student would progress through the program to completion. It should be clear the length of time it will take for a typical student to graduate. For undergraduate programs, this should be the four-year plan.

Freshman Year

Semester 1	Credits	Semester 2	Credits
CMSC100	1	CMSC132	4
CMSC131	4	MATH141	4
MATH140	4	ANTH222 (NSDL, DVUP)	4
ENGL101	3	AASP100 (DSHS, DVUP)	3
COMM107	3		
	15		15

Sophomore Year

Semester 1	Credits	Semester 2	Credits
CMSC216	4	CMSC330	3
CMSC250	4	CMSC351	3
MATH240	4	STAT400	3
ARCH271 (DSSP)	3	ASTR100 (DSNS)	3
		AAST233 (DSHU, DVUP)	3
	15		15

Junior Year

Semester 1	Credits	Semester 2	Credits
CMSC320	3	CMSC422	3
CMSC421	3	ENGL393	3
PHIL100 (DSHU)	3	CMSC470	3
GVPT200 (DSHS, DVUP)	3	ASTR340 (Upper Level Concentration 2)	3
ASTR315 (Upper Level Concentration 1 & DSSP)	4	ENGL125 (SCIS)	3
	16		15

Senior Year

Semester 1	Credits	Semester 2	Credits
CMSC417 (UL CS Elective)	3	CMSC451 (UL CS Elective)	3
CMSC426	3	EDSP476 (Elective)	3
ASTR350 (Upper Level Concentration 3)	3	PLCY215 (Elective)	3
ASTR380 (Upper Level Concentration 4)	3	ARTT260 (SCIS)	3
		KNES287 (Elective)	3
	12		15

Total Credits 118

List the intended student learning outcomes. In an attachment, provide the plan for assessing these outcomes.

Learning Outcomes
Graduates will be able to create, augment, debug, and test computer software. These skills will be built progressively through the courses in the introductory sequence and in some courses beyond that.
Graduates will develop mathematical and reasoning skills that are needed for computer science.
Graduates will be able to design and implement programming projects that are similar to those seen in the real world.
Graduates will gain skills in communication.
Academic Research (Optional): Graduates will be able to work independently on a project.

Program Modification Information

Linked Programs

Key: 527

CMSC4XX
Title: Introduction to Deep Learning

Credits: 3
Formerly Offered As: CMSC498L: Introduction to Deep Learning
Formerly Offered in: Spring 2019, Spring 2020

Description:
An introduction to deep learning, a machine learning technique, as well as its applications to a variety of domains. Provides a broad overview of deep learning concepts including neural networks, convolutional neural networks, recurrent neural networks, generative models, and deep reinforcement learning, and an intuitive introduction to basics of machine learning such as simple models, learning paradigms, optimization, overfitting, importance of data, and training caveats. These concepts are understood in the context of advanced applications in computer vision (image classification, object detection, and dense prediction) and natural language processing (sentiment analysis and review classification).

List of prerequisites and/or course restrictions:
Prerequisite: Minimum grade of C- in CMSC330 and CMSC351; and 1 course with a minimum grade of C- from (MATH240, MATH461);
Restriction: Permission of CMNS-Computer Science department. Or must be in (Computer Science (Doctoral), ComputerScience (Master's) program.

Learning Outcomes:
- Acquire the fundamental machine learning concepts that are required to understand and train neural networks. Assessed in the assignments, exams, and final project.
- Understand a variety of neural networks and how they are utilized in different applications. Assessed in the assignments and final project.
- Understanding of the inner workings of state-of-the-art deep learning algorithms and associated machine learning techniques. Assessed in the assignments, exams, and final project.
- Implement and train state-of-the-art deep learning algorithms for computer vision and language processing applications. Assessed in the assignments and final project.

Syllabus:
The course will cover fundamental deep learning techniques, using computer vision and language processing as core applications as a running theme.
- Machine learning basics
    - Simple models: linear and logistic regression
    - Paradigms of learning
- Neural networks basics
    - Terminology, simple neural networks, non-linearities
    - Problem setup, labels, and loss functions
- Optimization basics
    - Loss functions derivatives, local/global minimas

    - Gradient descent, stochastic gradient descent
- Training neural networks
    - Initialization, backpropagation
    - Training caveats: overfitting, bias/variance trade-offs
    - Optimization, hyperparameters, and tuning performance
- Convolutional neural networks
    - Task: Image classification
    - Popular architectures, intuitions and key-insights, design principles
    - Visualizing features and neurons, inversion techniques
    - Applications
        - Object detection
        - Dense prediction
- Recurrent neural networks
    - Recurrent architectures, including gated recurrent units and long-short term memory
    - Language modelling, text/language applications
    - Self-attention and transformers
- Advanced topics: vision plus language topics
    - Models, tasks, and training
    - Examples: image captioning, visual question answering
- Advanced topics: image generative models
    - Auto-regressive models
    - Generative adversarial networks, pix2pix, cycleGANs
    - Variational auto-encoders
    - Text-generation, self-supervised learning
- Advanced topics: brief introduction to deep reinforcement learning
- Ethics and bias

Assignments and projects

Assignments explore key concepts and simple applications and will ask students to implement and evaluate neural networks for computer vision and language processing applications as well as evaluate students' background. Example projects:

Assignment 0: Math background: probability, derivatives, vector calculus

Assignment 1: Implement two simple neural networks without using a deep learning library (e.g., using numpy and python): (a) a neural network to do image classification (computer vision), and (b) a neural network to do review classification (language processing).

Assignment 2: Re-implement the neural networks from assignment 2 in a deep learning library (e.g., PyTorch or TensorFlow).

Assignment 3: Write modules for building a convolutional neural network for object detection (basic code structure provided)

Assignment 4: Write modules for building a recurrent neural network for sequence translation (basic code structure provided)

Assignment 5: Write modules for building a neural network for driving a game car using reinforcement learning (basic code structure provided)

Course final project allows an in-depth exploration of a particular application area as chosen by the student. These are performed in groups of 3-5 students.

Grading scheme:
40% assignment projects
30% final project
30% exams (1 mid-term, 1 final)
Bonus points for top-N ranks for each challenge assignment.

Readings:
The course should be self contained, but if you need additional reading material, you can consult the following:

- Deep learning, Goodfellow, Bengio and Courville, 2016 (book)
- Machine Learning Crash Course, an interactive online course by folks at Google
- Dive into Deep Learning, an interactive online book by folks at Amazon
- Neural Networks and Deep Learning, an online book by Michael Nielsen

Course Title: Introduction to Robotics with Perception

Formerly: CMSC 498F Robotics and Perception (2016, 2017)

Description:
Introduction to the programming of robots with perception. Topics covered include navigation using vision and 3D depth sensors, localization and map making, image processing for visual navigation and recognition, and basic vision and depth-based manipulation. Develop algorithms and learn how to use vision and software tools, such as OpenCV, MoveIt and the Point Cloud Library. Programming is done in Python and C++ under the Robotic Operating System (ROS).

Additional Course Information:
The course will be organized around a few projects, starting with navigation in a map, then localization using the map, then finding objects, and finally a project of object manipulation. The software will first be developed in simulation, before testing it on the platform, where students will work in groups.

Course Prerequisites: MATH240

Learning Outcomes:
1. Ability to implement on a robot path planning and motion control for algorithms for navigation;
2. Understand and use SLAM algorithms based on range sensing;
3. Ability to integrate on a mobile robot the different modules for a complete visual search pipeline

Syllabus:

Date	Topic
Week 1	Introduction, Linear Algebra Review Overview of ROS
Week 2	Hardware and Locomotion Coding with ROS
Week 3	Coordinate System Transformations: Representing Position and Orientation Mobile Robot Kinematics
Week 4	Basic Control and Mobile Robot Control Inverse Kinematics
Week 5	Path Planning: Map Representation, Potential Field Methods Path Planning: Graph Algorithms, Obstacle Avoidance
Week 6	Sensors for Robotics
Week 7	Vision: Image Formation Filtering, Edge Detection, Features
Week 8	Vision: Motion Perception Midterm
Week 9	Depth from Stereo Probabilistic Robotics – Statistical Prerequisites

Week 10	Uncertainty and Error Propagation Localization with Sensors (Markov Localization)
Week 11	Localization with Particle Filters Localization with Kalman Filters
Week 12	Depth from Vision: Multiple View Geometry 3D Motion Perception
Week 13	Vision: Object Recognition: Specific Objects Visual Object Recognition of Object Classes
Week 14	Visual SLAM Project Discussion
Week 15	Grasping Visual Servoing

CMSC 43X: Capstone in Machine Learning
Fall 2018

Description
CMSC 43X is a semester-long project course in which each student will identify and carry out a project related to machine learning, with the goal of publishing a research paper or software tool that will benefit their community. Students will...
• Understand machine learning principles and software related to their chosen application area.
• Develop publicly releasable code. Code projects may require
    o Modularity, portability, goog memory management practices
    o Post-processing, restarting, and writing to databases
    o Interactivity
    o Scientific visualization
    o Documentation and version management tools
    o Debugging and profiling tools
    o Validation
• Build verification methods and unit tests

Students will be paired with project advisors from the UMD faculty, or alternatively an industry advisor. Students are encouraged to plan for project results that are publishable at academic conferences, or will impact academic research.

Logistics
Lectures: TBD
Location: TBD
Instructors:
    Tom Goldstein: Irbie 4121

Prerequisites
You *must* have taken either 421 or 422, and further exposure to machine learning topics is strongly encouraged.

Course requirements
The first semester will have the following deliverables: proposal document, proposal presentation, midterm presentation, final presentation, final report. You will also deliver code and documentation for your project.

Your faculty advisor is required to participate in a meeting with 663 supervisors, and must attend your final presentation.

Grading

This course does NOT have an “A for effort” grading policy. Projects will be graded on the quality of code, documentation, presentation, and writing. Projects that do not meet high standards of quality will not receive an “A”, and in some circumstances will not receive a passing grade.

I**mportant dates and grading events (tentative)**

Aug 30-Sept 11: Personal presentations
Sept 24: Proposal report and advisor meetings must be completed:
Sept 25 - Oct 11: Proposal presentations
Oct 23-25: Code reviews
Oct 30 - Nov 8: Midterm update presentations
Nov 13 - 15: Code review
Nov 27th - done: final presentations
Dec 10: Final report

Requirements for deliverables

Personal presentation (15 mins):

- Where are you from?
- What are you interested in professionally?
- What are you interested in personally?
- What do you hope to get out of this class?

Proposal document and presentation should include:

- Background on the problem: why is it important, what is the state of the art (with citations)?
- Project Goals: what are you hoping to achieve?
- Approach: How will you achieve these goals? What components will need to be implemented to get there?
- Describe specific algorithms and how they will be implemented
- Describe hardware/software platform you target. What programming language?
- How will it be *documented* and *distributed*? Github? User guide?
- Validation methods: complete suite of unit tests!
- Deliverables: what specific components/code/data/visualizations will you generate?
- Milestones, and a rough timeline of when you expect to accomplish them?

The oral presentation should last no more than 30 minutes including questions and discussions. A sample presentation document is available. You will be evaluated on the quality of your presentation skills, and clarity of your slides.

The proposal document should be at most 5 pages, and fewer is acceptable.

Midterm update (15 mins):

- Brief overview of project

- Brief overview of approach, goals, milestones
- Update on current status and accomplishments
- Figures or code snippets or demos to show what your code can do
- Explanation and description of any changes/updates to approach

Final Presentation and report (30 mins):

- Build on the proposal and midterm presentations.
- Overview of proposal content
- Detailed description of what has been accomplished and changes to the approach
- Description of what did not get accomplished and why
- Overview of deliverables, such as code, data, code documentation and distribution
- Description of plans for the next semester

Your code must be...

- Well documented, with a user guide (if appropriate)
- Well commented. This includes having descriptive headers and usage information at the top of every file, and having important description information within the code body. Code should also be clearly written, and "self-documenting" when appropriate.
- Well organized, clean, loosely coupled, and extensible
- Portable to different systems
- Thoroughly unit tested and validated
- Distributed using online tools, such as Gitlab, Github, etc...

Academic integrity

Be aware of the UMD code of academic integrity, found here http://www.president.umd.edu/policies/iii100a.html. All sources must be properly and clearly cited. Text or code can never be copied verbatim from the sources, this is plagiarism and a serious offense. You can, of course, import and use third-party libraries in your code (provided you give full and due credit and discuss the third party in your reports and presentations).