Computer Science Engineering

M. Tech Computer Science Engineering						
Sr No	**Course Code**	**Course Name**	**L**	**T**	**P**	**C**
1	CS 402	Distributed Systems	3	0	0	6
2	CS 403	Graph Theory and Combinatorics	3	0	0	6
3	CS 410	Parallel Computing	3	0	0	6
4	CS 421	Logic for Computer Science	3	0	0	6
5	CS 426	Introduction to Blockchains	3	0	0	6
6	CS 427	Mathematics for Data Science	3	0	0	6
7	CS 438	Natural Language Processing	3	0	0	6
8	CS 439	Introduction to Sanskrit Computational Linguistics	3	0	0	6
9	CS 633	Advanced Data Structure and Algorithms	3	0	0	6
10	CS 619	Advanced Data Structure and Algorithms Lab	0	0	3	3
11	CS 618	Advanced Software Development Laboratory	1	0	4	6
12	CS 634	Combinatorics and Probability	3	0	0	6
13	CS 601	Software Development for Scientific Computing	3	0	0	6
14	CS 603	Approximation algorithms	3	0	0	6
15	CS 604	Parameterized Algorithms and Complexity	3	0	0	6
16	CS 706	Topics in Parameterized Algorithms and Complexity	3	0	0	6
17	CS 606	Advanced topics in Embedded Computing	3	0	0	6
18	CS 607	Advanced Computer Networks	3	0	0	6
19	CS 608	FPGA for communication networks prototyping.	3	0	0	6
20	CS 609	Software Defined Networking (SDN) and Network Function Virtualization (NFV)	3	0	0	6
21	CS 610	Advanced Distributed Systems	3	0	0	6
22	CS 612	Statistical Pattern Recognition Laboratory	0	0	3	3
23	CS 616	Statistical Pattern Recognition	3	0	0	6
24	CS 621	Logic and Applications	3	0	0	6
25	CS 622	Special Topics in Automata and Logics	3	0	0	6
26	CS 624	Compilers - Principles and Implementation	3	0	0	6
27	CS 810	Advanced Computer Architecture	3	0	3	9
28	EE 609	Pattern Recognition and Machine Learning (PRML)	3	0	0	6
29	EE 612	Pattern Recognition and Machine Learning (PRML) Laboratory	0	0	3	3
30	EE 606	Neural Networks and Deep Learning (NNDL)	3	0	0	6
31	EE 611	Neural Networks and Deep Learning (NNDL) Laboratory	0	0	3	3

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Distributed Systems** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	Operating Systems, Data Structures and Algorithms, Programming in C++
3	**Course content**	• Introduction to distributed systems, Message Passing, Leader Election, Distributed Models, Causality and Logical Time • Logical Time, Global State & Snapshot and Distributed Mutual Exclusion-Non- Token and Quorum based approaches • Distributed Mutual Exclusion-Token based approaches, Consensus & Agreement, Checkpointing & Rollback Recovery • Deadlock Detection, DSM and Distributed MST • Termination Detection, Message Ordering & Group Communication, Fault Tolerance and Self-Stabilization, Gossip Style communication, chord, pastry • Concurrency and Replication Control, RPCs, Transactions • Distributed Randomized Algorithms, DHT and P2P Computing • Case Studies: GFS, HDFS, Map Reduce and Spark
4	**Texts/References**	• Distributed Computing: Principles, Algorithms, and Systems- Ajay D. Kshemkalyani and Mukesh Singhal • Distributed Computing: Fundamentals, Simulations and Advanced Topics-Hagit Attiya and Jennifer Welch • Distributed Algorithms-Nancy Lynch • Elements of Distributed Computing-Vijay K. Garg • Advanced Concepts in Operating Systems-Mukesh Singhal, Niranjan G. Shivaratri

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Parallel Computing** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to C, C++ or Fortran programming
3	**Course content**	Need for High Performance Computing (HPC) and applications. Sequential Computing model, Algorithms and their complexity. Taxonomy of computer architectures – SISD, SIMD (e.g. array processors), MISD (pipelined processing, vector processors), and MIMD (shared memory and distributed memory multiprocessors, computing clusters); dataflow computing; hardware accelerators (GPUs); interconnection networks (bus, loop, mesh and hypercube); Memory hierarchy; Case Studies. Implications of computer architectures to algorithm design, synchronous processing, single program multiple data (SPMD) and multiple program multiple data (MPMD) processing; functional and data parallelism; memory hierarchies. Performance evaluation: communication and computing costs, speedup, efficiency, Amdahl's law, parallel scalability. Parallel algorithm design and case studies: numerical algorithms (linear algebra, matrix- vector and matrix-matrix multiplications, finite difference method and PDEs, Monte Carlo method), and non-numerical algorithms (search, sorting, simple tree and graph algorithms) Parallel programming platforms, OpenMP and MPI programming, GPU programming. Programing Assignments: 1. Parallel computing lab environment (system architecture, log on, hello world 2. Editors, job submission, optimization techniques for serial code. 3. MPI and simple program(s) 4. MPI and matrix-matrix multiplication 5. OpenMP and matrix-matrix multiplication OpenMP 6. Introduction to GPU programming – matrix-matrix multiplication.
4	**Texts/References**	**1.** Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar: Introduction to Parallel Computing, Addison Wesley 2003 **2.** Eric Aubanel, Elements of Parallel Computing, CRC Press, 2017. 3.https://computing.llnl.gov/tutorials/mpi/ 4.https://computing.llnl.gov/tutorials/open MP/

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Logic for Computer Science** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	Discrete Mathematics, Theory of computation.
3	**Course content**	1. Module 1 :Propositional Logic: Syntax, Semantics, Normal Forms, Boolean Functions. 2. Module 2: Computational complexity of Satisfiability P vs NP, SAT: hardest among NP. 3. Module 3: Syntactic SAT solvers : Resolution, Tableaux. 4. Module 4:proof Systems: Semantic entailment, Compactness, Soundess Completeness, Natural Deduction, Gentzen Sequent Calculus, Hilbert System. 5. Module 5: Predicate Logic. Randomized SAT solvers. Programming assignments: using SAT/SMT solver z3.
4	**Texts/References**	1. Logic in Computer Science, Michael Huth and Mark Ryan, Cambridge University Press. 2. SAT/SMT by example, Dennis Yurichev.

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Introduction to Blockchains** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	It is expected that students should have a very good programming background preferably in Java/JavaScript and Python.
3	**Course content**	Basics - What is Blockchain, Centralized vs Decentralized vs Distributed, Blockchains and Public Ledgers, Fundamentals - Cryptographically secure hash functions, Merkle Trees, Bitcoin Concept Architecture - Blockchain 2.0, smart contracts, block structure, notion of distributed consensus, challenge response to Permission-less Consensus, economics behind blockchain consensus Consensus in Blockchain - Distributed Consensus, Proof of Work, Miners in the context of Bitcoin Permissioned Blockchain - Basics, RAFT Consensus, Byzantine General Problem, Practical Byzantine Fault Tolerance Hyperledger Fabric - Introduction, Transaction flow, Membership and Identity Management, Hyperledger Composer" Ethereum Framework - Installation and subsequent execution of the use cases. Blockchain in Financial Service - Payments and Secure Trading, Compliance and Mortgage, Financial Trade Blockchain in Supply Chain Blockchain in Government Use Cases - Digital Identity" Mini Project implementation using Ethereum/Hyperledger framework on use cases related to financial, supply chain, and government sectors
4	**Texts/References**	Arvind Narayanan, Joseph Bonneu, Edward Felten, Andrew Miller, Steven Goldfeder, “Bitcoin and Cryptocurrency Technologies – A Comprehensive Introduction”, Princeton University Press, 2016. Research papers as well as several related online educational content will also be referred to for learning some of the afore-mentioned topics.

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Graph Theory and Combinatorics** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	Discrete Structures
3	**Course content**	Fundamentals of graph theory. Topics include: connectivity, planarity, perfect graphs, coloring, matchings and extremal problems. Basic concepts in combinatorics. Topics include: counting techniques, inclusion-exclusion principles, permutations, combinations and pigeon- hole principle.
4	**Texts/References**	"An Introduction to Quantum Field Theory", Michael Peskin and Daniel Schroeder (Addison Wesley) "Introduction to Quantum Field Theory", A. Zee "Quantum Field Theory", Lewis H. Ryder "Quantum Field Theory and Critical Phenomena", by Jean Zinn-Justin. "Quantum field Theory for the Gifted Amateur", T. Lancaster and Stephen J. Blundell NPTEL lectures in Quantum Field Theory (https://nptel.ac.in/courses/115106065/)

Computer Science Engineering

1	**Title of the course** (L-T-P-C)	**Mathematics for Data Science** **(3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to basic concepts in calculus and linear algebra
3	**Course content**	Introduction to Data science and Motivation for the course. Review of calculus, naTve set theory, notion of limits, ordering, series and its convergence. Introduction to Linear Algebra in Data science, notion of vector space, dimension and rank, algorithms for solving linear equations, importance of norms and notion of convergence, matrix decompositions and its use. Importance of optimization in data science: Birds view of Linear Regression, Multivariate Regression, Logistic Regression etc. Convex Optimization: Convex sets, convex functions, duality theory, different types of optimization problems, Introduction to linear program. Algorithms: Central of gravity method, Gradient descent methods, Nestrov acceleration, mirror descent/Nesterov dual averaging, stochastic gradient methods, Rmsprop, SIGNSGD, ADAM algorithm etc. Non-convex optimization: Demonstration of convex methods on non-convex problems; merits and disadvantages.
4	**Texts/References**	1. C. Bishop, “Pattern Recognition and Machine Learning,”Springer, 2006. 2. Cambridge university press, 2018 (reprint). for Machine Learning,” Now publisher, 2017.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Data Structure and Algorithms (3-0-0-6)**
2	**Pre-requisite courses(s)**	None
3	**Course content**	**Module 1:** Basics: asymptotic notations, recurrences, basic data structures **Module 2:** Advanced data structures: heaps, priority queues, hash tables, data structures based on trees. **Module 3:** Design paradigms and complexity analysis: divide and conquer, dynamic programming, greedy algorithms, amortized analysis. **Module 4:** Advanced topics: graph algorithms, string algorithms, geometric algorithms, complexity lower bounds, coping up with hard problems.
4	**Texts/References**	Introduction to algorithms: Cormen, Leiserson, Rivest, and Stein (Main textbook) Online lecture notes by Jeff Erickson The Algorithm Design Manual: Steven Skiena Algorithm Design: Kleinberg and Tardos Data structures and algorithm analysis in C++ (Java): Mark Weiss

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Data Structure and Algorithms Lab (0-0-3-3)**
2	**Pre-requisite courses(s)**	None
3	**Course content**	**Module 1:** Basics: asymptotic notations, recurrences, basic data structures **Module 2:** Advanced data structures: heaps, priority queues, hash tables, data structures based on trees. **Module 3:** Design paradigms and complexity analysis: divide and conquer, dynamic programming, greedy algorithms, amortized analysis. **Module 4:** Advanced topics: graph algorithms, string algorithms, geometric algorithms, complexity lower bounds, coping up with hard problems.
4	**Texts/References**	Introduction to algorithms: Cormen, Leiserson, Rivest, and Stein (Main textbook) Online lecture notes by Jeff Erickson The Algorithm Design Manual: Steven Skiena Algorithm Design: Kleinberg and Tardos Data structures and algorithm analysis in C++ (Java): Mark Weiss

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Software Development Laboratory (1-0-4-6)**
2	**Pre-requisite courses(s)**	None
3	**Course content**	Editing: Vim/emacs, Presentation: latex, beamer, Build, Integration and Deployment, Version Control, and Documentation: make, GitHub and git, doxygen Programming: HTML, CSS, Shell scripting, AWK, SED Exploring features of IDE (e.g., eclipse, vscode) using a high-level programming language such as Java, Python, C++, debugging (using gdb, using IDE). Unix/Linux basics: shell, file system, permissions, process hierarchy, process monitoring, ssh, scp, rsync, grep, find, head, tail, tar, cut, sort, I/O redirection, pipes. Profiling tools: (e.g., gprof, prof, perf, valgrind) A medium-sized project with significant weight.
4	**Texts/References**	**1.** Online tutorials for HTML/CSS, Inkscape, OODrawUnix Man Pages for all unix tools, Advanced Bash Scripting Guide from the Linux Documentation Project (www.tldp.org). **2.** The Python Tutorial Online Book (http://docs.python.org/3/tutorial/index.html). **3.** The Java Tutorials (http://docs.oracle.com/javase/tutorial/). **4.** Latex - A document preparation system, 2/e, by Leslie Lamport, Addison Wesley, 1994. **5.** Make https://www.gnu.org/software/make/manual/html_node/index.html#Top. **6.** Git - Git - Book (git-scm.com) **7.** GDB Documentation (sourceware.org) **8.** Valgrind **9.** Doxygen - My Project: Documenting the code (doxygen.nl) **10.** Gprof - GNU gprof

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Combinatorics and Probability (3-0-0-6)**
2	**Pre-requisite courses(s)**	None
3	**Course content**	Combinatorics: Principles of counting: rule of sum, rule of product, permutations, combinations. partition, modular arithmetic. Double counting, generating functions. Binomial coefficients, binomial theorem. multinomial theorem Probability theory: probability axioms and laws, random variables, binomial distribution, Poisson, exponential and normal distributions, expectations and moments, joint distribution, conditional distribution, conditional expectations, convergence of random variables, law of large numbers, central limit theorem, Markov chains.
4	**Texts/References**	**1.** W. Feller, W: An Introduction to Probability Theory and its Applications, Vol.1, John Wiley. **2.** G. R. Grimmett and D. R. Stirzaker: Probability and Random Processes, Oxford Science Publications. **3.** Biggs, N. L., Discrete Mathematics, Oxford Science Publications, 1989. **4.** Invitation to Discrete Mathematics, Jiří Matoušek, Jaroslav Nešetřil, Oxford University Press.

Computer Science Engineering

The following Table shows the electives related to CSE discipline.

Course Code	Course	L-T-P-C
CS 402	Distributed Systems	3-0-0-6
CS 403	Graph Theory and Combinatorics	3-0-0-6
CS 410	Parallel Computing	3-0-0-6
CS 421	Logic for Computer Science	3-0-0-6
CS 426	Introduction to Blockchains	3-0-0-6
CS 427	Mathematics for Data Science	3-0-0-6
CS 438	Natural Language Processing	3-0-0-6
CS 439	Introduction to Sanskrit Computational Linguistics	3-0-0-6
CS 601	Software Development for Scientific Computing	3-0-0-6
CS 603	Approximation algorithms	3-0-0-6
CS 604	Parameterized Algorithms and Complexity	3-0-0-6
CS 606	Advanced Topics in Embedded Computing	3-0-0-6
CS 607	Advanced Computer Networks	3-0-0-6
CS 608	FPGA for communication networks prototyping	3-0-0-6
CS 609	Software Defined Networking and Network Function Virtualization	3-0-0-6
CS 610	Advanced Distributed Systems	3-0-0-6
CS 612	Statistical Pattern Recognition Laboratory	0-0-3-3
CS 616	Statistical Pattern Recognition	3-0-0-6
CS 621	Logic and Applications	3-0-0-6
CS 622	Special Topics in Automata and Logics	3-0-0-6
CS 624	Compilers - Principles and Implementation	3-0-0-6
CS 810	Advanced Computer Architecture	3-0-3-9
EE 606	Pattern Recognition and Machine learning (PRML)	3-0-0-6
EE 612	Pattern Recognition and Machine learning (PRML) Laboratory	0-0-3-3
EE 620	Neural networks and deep learning (NNDL)	3-0-0-6
EE 611	Neural networks and deep learning (NNDL) Laboratory	0-0-3-3

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Software Development for Scientific Computing (3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to Data Structures and Algorithms, C / C++ / Java / Matlab
3	**Course content**	Algorithmic Patterns in Scientific Computing: dense and sparse linear algebra, structured and unstructured grid methods, particle methods (N-body, Particle-Particle, Particle-in-cell, Particle-in-a- mesh), Fast Fourier Transforms, Implementing PDEs, C++ standard template library (STL), Introduction to debugging using GDB, GMake, Doxygen, Version Control System, Profiling and Optimization, asymptotic analysis and algorithmic complexity. Mixed-language programming using C, Fortran, Matlab, and Python, Performance analysis and high-performance code, Data locality and auto tuning, Introduction to the parallel programming world.
4	**Texts/References**	• Stroustrup C++ Language Reference (https://www.stroustrup.com/4th.html) • Suely Oliveira, David Steward: Writing Scientific Software: A Guide to Good Style. Cambridge University Press, 2006 • Web references to GNU Make, GDB, Git, GProf, Gcov. • Code Complete: A Practical Handbook of Software Construction • https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006 183.html

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Approximation algorithms (3-0-0-6)**
2	**Pre-requisite courses(s)**	Data Structures and Algorithms (CS201)
3	**Course content**	Introduction, approximation schemes, design, and analysis of approximation algorithms - combinatorial algorithms, linear programming-based algorithms. Hardness of approximation.
4	**Texts/References**	**Textbook:** • Approximation algorithms. Vazirani, Vijay V. Berlin: springer, 2001. **Reference:** • The design of approximation algorithms. Williamson, David P., andDavid B. Shmoys. Cambridge university press, 2011.

Computer Science Engineering

Computer Science Engineering

1	**Title of thecourse (L-T-P-C)**	**Topics in Parameterized Algorithms and Complexity (3-0-0-6)**
2	**Pre-requisite courses(s)**	Data Structures and Algorithms, Design and Analysis of Algorithms
3	**Course content**	Introduction. Kernelization, Bounded Search Trees, Iterative Compression, Treewidth, Advanced kernelization algorithms. Lower bounds: Fixed-parameter intractability, lower bounds based on ETH, lower bounds for kernelization. Parameterized Algorithms, Kernelization, and Complexity of Graph Modification Problems
4	**Texts/References**	**Textbook:** • Parameterized Algorithms, Marek Cygan, Fedor V. Fomin, Lukasz kowalki. Daniel Lokshtanov, Daniel Mark, Marcin Pilipaczuk, Michal Pilipczuk, and Saket Sourabh. Springer. 2015 **Reference:** • Parameterized Complexity, R. G. Downey, and M. R. Fellows. Springer Science and Business Media. 2012

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced topics in Embedded Computing (3-0-0-6)**
2	**Pre-requisite courses(s)**	CS 301
3	**Course content**	Introduction to systems software in embedded platforms Boot loader Embedded Linux kernel (Processes, Threads, Interrupts) Device Drivers Scheduling Policies (including Real Time) Memory Management Optimizations (Data level and Memory level) Embedded Systems Security Introduction to Embedded GPUs and Accelerators Embedded Heterogenous Programming with Open CLApplication Case Study on Embedded Platforms – eg. Neural Network inferencing on Embedded Platforms, Advanced Driver Assistance Systems
4	**Texts/References**	• Building Embedded Linux Systems, 2nd Edition by Gilad Ben-Yossef, Jon Masters, Karim Yaghmour, Philippe Gerum, O'Reilly Media, Inc. 2008 • Linux Device Drivers, Third Edition by Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, O'Reilly Media, Inc. 2005 • Embedded Systems: ARM Programming and Optimization by Jason D Bakos, Elsevier, 2015 • Learning Computer Architecture with Raspberry Pi by Eben Upton, Jeff Duntemann, Ralph Roberts, Tim Mamtora, Ben Everard, Wiley Publications, 2016 • Real Time Systems by Jane S. Liu, 1 edition, Prentice Hall; 2000 • Practical Embedded Security: Building Secure Resource-Constrained Systems by Timothy Stapko, Elsevier, 2011

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Computer Networks (3-0-0-6)**
2	**Pre-requisite courses(s)**	
3	**Course content**	**1.** Circuit, Packet and Virtual Circuit Switching, MPLS **2.** Switch Architectures, Buffering Strategies, Input and Output Queuing, IP Buffer Sizing **3.** Quality of Service and Scheduling Algorithms **4.** IP Address Lookup and IP Packet Classification algorithms **5.** Software Defined Networking **6.** Next Generation Network Architectures, Network Provisioning and Design, and "Green" (Energy- Efficient) Networking **7.** Data Driven Networking **8.** Wireless Networks - manets, Sensor Networks, Cellular Networks, Personal Area Networks **9.** Content Based Delivery Networks - Principles of data dissemination, aggregation and caching that are applied to sensor networks, Internet of Things, and other content-based paradigms. Students will survey recent research publications on opportunistic networks and next generation content-based networking ideas. **10.** Delay tolerant networks. **11.** Network security - authentication, access control, privacy preservation, intrusion detection and prevention. **12.** Performance analysis of new Networking ideas using simulation (such as Network Simulator (ns3), GENI testbed, Simulink, Open LTE and Open C-RAN frameworks)
4	**Texts/References**	**Textbook:** Computer Networks: A Systems Approach, Larry Peterson, and Bruce Davie, 2011. Performance Evaluation of Computer Systems, by Raj Jain, Wiley, 1991. Computer Networking, Kurose and Ross, Addison-Wesley, 2012. **Reference:** **1.** An Engineering Approach to Computer Networking by S. Keshav, 1997, Addison-Wesley Professional Series. **2.** Network Routing, by Deepankar Medhi and Karthikeyan Ramasamy, Morgan Kaufmann, 2007. **3.** SDN: Software Defined Networks, by Thomas D. Nadeau, Ken Gray, O'Reilly Media, 2013. **4.** High Performance Switches and Routers, By H. Jonathan Chao and Bin Liu, Wiley, 2007. Network Algorithmics, by George Varghese, Morgan Kaufmann, 2005

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**FPGA for communication networks prototyping. (3-0-0-6)**
2	**Pre-requisite courses(s)**	EE 224 Digital System Exposure on Computer Network
3	**Course content**	History and evaluation of FPGAs; FPGA architecture; Introduction to Quartus Prime (vendors and design tools; vendors and programmable logic); Exploiting Simulation tools (e.g., ModelSim); Exploiting FPGAs for multi-domain technologies; Introduction to radio access networks-fronthaul (e.g., common public radio interface); optical network; metro and core networks; Cross-layer design; The role of FPGA in the specified network segments and use case scenarios; In and Out; Clocks and Registers; State Machines; Modular Design; Memories Managing Clocks; I/O Flavors; Exploiting Qsys and Nios II tools
4	**Texts/References**	**1.** C. Maxfield, "The Design Warrior's Guide to FPGAs: Devices, Tools and Flows," Jun. 2004, eISBN 9780080477138 **2.** FPGAs For Dummies, 2nd Intel Special Edition. Published by. John Wiley & Sons, Inc **3.** William J. Dally, R. Curtis Harting, "Digital Design: A Systems Approach 1st Edition", Cambridge University Press, September 2012, ISBN 9780521199506 **4.** Verilog by Example: A Concise Introduction for FPGA Design, Blaine C. Readler **5.** Course materials: Slides; Notes; Tutorials from Altera website https://www.altera.com/support/training/university/materials-tutorials.html **6.** R. Ramaswami, K. Sivarajan, G. Sasaki; "Optical Networks: A Practical Perspective," 3rd Ed., Morgan Kaufmann, ISBN: 9780123740922

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Software Defined Networking (SDN) and Network Function Virtualization (NFV) (3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to Computer Networks
3	**Course content**	History and evolution of SDN; SDN Architecture (Application, Control, Infrastructure Layer); SDN Interfaces (East/West/North/South-bound interfaces); SDN Security; SDN routing; SDN standards; SDN Controllers; Network Operating Systems and Languages; OpenFlow; Software Switches (e.g. OpenVSwitch); SDN Simulation/Emulation Platforms (e.g. Mininet); Federated SDN networks; SDN Applications and Use Cases; Programming assignment/project. Need for NFV; NFV and SDN Relationship; Virtual Network Functions; Service Function Chaining; NFV Specifications; NFV Architecture; NFV Use Cases; NFV Management and orchestration (MANO); Open-source NFV; Hands-on exercises based on OpenStack/Docker.
4	**Texts/References**	• Software Defined Networks: A Comprehensive Approach by Paul Goransson and Chuck Black, Morgan Kaufmann Publications, 2014 • SDN – Software Defined Networks by Thomas D. Nadeau & Ken Gray, O'Reilly, 2013 • Software Defined Networking with OpenFlow, By Siamak Azodolmolky, Packt Publishing, 2013 • Gray, Ken, and Thomas D. Nadeau. Network function virtualization. Morgan Kaufmann, 2016. • Zhang Ying. Network Function Virtualization: Concepts and Applicability in 5G Networks. John Wiley & Sons, 2018. • Foundations of modern networking- SDN, NFV, QoE, IoT, and Cloud, William Stallings • James Kurose and Keith Ross, "Computer Networking, A Top-Down Approach"

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Distributed Systems (3-0-0-6)**
2	**Pre-requisite courses(s)**	Operating Systems, Data Structures and Algorithms, Programming in C++
3	**Course content**	Synchronization, Global Snapshot and Distributed Mutual Exclusion, Consensus & Agreement, Checkpointing & Rollback Recovery, Deadlock Detection, Termination Detection, Message Ordering & Group Communication, Fault Tolerance and Self- Stabilization, Peer to Peer Systems Mining Data Streams in a distributed system: filtering data streams, queries on streams, pattern detection Key-Value Storage: Cassandra, HBase Virtualization and Cloud Computing: virtual machines containers Message oriented communication, Publish Subscribe Systems (use case Apache Kafka) Security: Distribution of security mechanisms, access control, and security management.
4	**Texts/References**	**1.** Distributed Computing: Principles, Algorithms, and Systems- Ajay D. Kshemkalyani and Mukesh Singhal **2.** Mining Massive data sets- Jure Leskovec, Anand Rajaraman, Jeff Ullman **3.** Distributed Algorithms – An Intuitive Approach (The MIT Press) by Wan Fokkink **4.** Distributed Algorithms-Nancy Lynch

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Statistical Pattern Recognition Laboratory (0-0-3-3)**
2	**Pre-requisite courses(s)**	Currently taking statistical pattern recognition theory course
3	**Course content**	The lab will closely follow the theory course. The idea is to have the students implement the basic algorithms on different topics studied in the statistical pattern recognition theory course.
4	**Texts/References**	**1.** R.O. Duda, P.E. Hart and D.G. Stork, Pattern Classification, John Wiley, 2001. **2.** C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Statistical Pattern Recognition (3-0-0-6)**
2	**Pre-requisite courses(s)**	Multivariate Calculus and Linear Algebra, Probability, Programming
3	**Course content**	Ayesian Decision Making and Bayes Classifier, Parametric And Non-Parametric Estimation Of Densities, General Linear Models, Discriminative Learning Based Models, Dimensionality Reduction Techniques, Empirical And Structural Risk Minimization, Ensemble Methods - Bagging, Boosting, Pattern Clustering, Graphical Models, Statistical Learning Theory
4	**Texts/References**	**1.** R O Duda, P E Hart and D G Stork, Pattern Classification, John Wiley, 2001. **2.** C.M. Bishop, Pattern Recognition and Machine Learning, Springer,2006

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Logic and Applications (3-0-0-6)**
2	**Pre-requisite courses(s)**	Discrete Mathematics, Theory of computation
3	**Course content**	**Module 1**: Propositional Logic: Natural deduction, semantics, soundness, completeness, compactness, normal forms, Horn clauses and satisfiability. **Module 2:** Predicate Logic: Natural deduction, resolution, undecidability, expressiveness. **Module 3:** Some decidable fragments of first-order logic and their decision procedures: propositional logic, equality with uninterpreted functions, linear arithmetic, Presburger logic, bit vectors, arrays, pointer logic. **Module 4:** SAT and SMT solvers: theory and practice: Decision procedures for combinations of first-order theories: Nelson-Oppen, Shostak, Satisfiability Modulo Theories (SMT) Combination with SAT solvers: eager, lazy approaches. Student is required to do a small project using a SAT/SMT solver.
4	**Texts/References**	**1.** Logic in Computer Science, Michael Huth and Mark Ryan, Cambridge University Press. **2.** Mathematical Logic for Computer science, Mordechai Ben-Ari, Springer. **3.** Logic for Computer Scientists, Uwe Schoning, Birkhauser. **4.** SAT/SMT by example, Dennis Yurichev.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Special Topics in Automata and Logics (3-0-0-6)**
2	**Pre-requisite courses(s)**	Discrete Mathematics, Theory of computation, Logic, and its applications.
3	**Course content**	This course aims at giving an introduction to the theory of automata working on infinite words and infinite trees and connections thereof to logic. These automata and related logics are of fundamental importance in the areas of formal specification and verification of reactive systems. If time permits, we will also discuss some basic results in finite model theory. Below is a list of topics which will be discussed on this course. Automata on finite words - equivalence of MSO and automata; Automata on infinite words â€“ different acceptance conditions; Closure properties and equivalence of different acceptance conditions and related translations Determinization and complementation results; Equivalence of automata and MSO and decidability of MSO; Automata on infinite trees - different acceptance conditions; Closure properties and comparison of expressive power of different acceptance conditions and related translations Complementation result for tree automata via parity games; Equivalence of MSO and tree automata; Decidability of MSO over tress; Parity games and determinacy; Ehrenfeucht-Frasse games in logics and applications
4	**Texts/References**	**1.** Wolfgang Thomas: Automata on infinite objects, Handbook of theoretical computer science (vol B): formal methods and semantics, Elsevier. **2.** Wolfgang Thomas: Languages, automata, and logic, Handbook of formal languages, vol. 3: beyond words, Springer-Verlag. **3.** Dominique Perrin, Jean-Eric Pin: Infinite words, Elsevier **4.** Erich Gradel, Wolfgang Thomas, Thomas Wilke: Automata, logics, and infinite games: a guide to current research. LNCS, Springer-Verlag. **5.** Leonid Libkin: Elements of finite model theory, Springer-Verlag.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Compilers - Principles and Implementation (3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to Data Structures and Algorithms, Computer Architecture, Automata Theory
3	**Course content**	Structure of a compiler, the compiled and interpreted execution models. Lexical analysis and parsing using lex and yacc. Scope and visibility analysis. Data layout and lifetime management of data. Runtime environment. Dynamic memory allocation and Garbage collection. Translation of expressions, control structures, and functions. Code generation and local optimizations, Lattice theory, register allocation, instruction scheduling, optimizations - dataflow, control flow, reaching definitions, and liveness analysis, code transformation-tiling, fusion.
4	**Texts/References**	**1.** Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman: Compilers: Principles, Techniques, and Tools, 2/E, AddisonWesley 2007. **2.** Andrew Appel: Modern Compiler Implementation in C/ML/Java, Cambridge University Press, 2004 **3.** Dick Grune, Henri E. Bal, Cerial J.H. Jacobs, and Koen G. Langendoen: Modern Compiler Design, John Wiley & Sons, Inc. 2000. **4.** Michael L. Scott: Programming Language Pragmatics, Morgan Kaufman Publishers, 2006. 5. Fisher and LeBlanc: Crafting a Compiler in C.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Advanced Computer Architecture (3-0-3-9)**
2	**Pre-requisite courses(s)**	Computer Architecture
3	**Course content**	Instruction-level parallelism: out-of-order pipelines; Thread-level parallelism: multi-core, multi-threading, memory hierarchies, coherence and consistency, on-chip networks; Data-level parallelism: vector processing, GPUs; optimizations and enhancements: modern branch predictors, instruction, and data prefetchers, value speculation.
4	**Texts/References**	**Textbook:** **1.** Computer Architecture: A Quantitative Approach, David Patterson, and John L. Hennesy, Elsevier, Sixth edition. 2017 **Reference:** **1.** Processor Microarchitecture: An Implementation Perspective. Antonio Gonzalez, Fernando Latorre, and Grigorios Magklis. Synthesis Lectures on Computer Architecture. 2011. (available online) **2.** A Primer on Memory Consistency and Cache Coherence, Daniel Sorin, Mark Hill, and David Wood, Morgan, and Claypool Publishers, 2011 **3.** On-chip Networks: Second edition, Natalie Enright Jerger, Tushar Krishna, Li-Shiuan Peh, Morgan and Claypool Publishers, 2017 **4.** Parallel Computer Architecture, David Culler, Jaswinder Pal Singh, Anoop Gupta, Elsevier, 1998

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Pattern Recognition and Machine Learning (PRML) (3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to basic concepts in calculus and probability
3	**Course content**	**Overview of** Probability Theory, Linear Algebra, Convex Optimization. **Introduction:** History of pattern recognition & machine learning, distinction in focus of pattern recognition and machine learning. **Regression**: Linear Regression, Multivariate Regression, Logistic Regression. **Clustering:** Partitional Clustering, Hierarchical Clustering, Birch Algorithm CURE Algorithm, Density-based Clustering **PCA and LDA:** Principal Component Analysis, Linear Discriminant Analysis. **Kernel methods:** Support vector machine **Graphical Models**: Gaussian mixture models and hidden Markov models **Introduction to Bayesian Approach:** Bayesian classification, Bayesian Learning, Bayes Optimal Classifier, Naive Bayes Classifier and Bayesian Network.
4	**Texts/References**	**1.** C. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006. **2.** S. Theodoridis and K. Koutroumbas, "Pattern Recognition" Second Edn, Elsivier, 2003 **3.** B. Yegnanarayana, "Artificial Neural Networks," PHI, 1999. **4.** Simon Hayking, "Neural Networks and Learning Machines,"Pearson, 1999.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Pattern Recognition and Machine Learning (PRML) Laboratory (0-0-3-3)**
2	**Pre-requisite courses(s)**	Currently taking or already taken PRML theory course
3	**Course content**	The lab will closely follow the theory course. The idea is to have the students implement the basic algorithms on different topics studied in the PRML theory course.
4	**Texts/References**	**1.** C. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006. **2.** S. Theodoridis and K. Koutroumbas, "Pattern Recognition" Second Edn, Elsivier, 2003 **3.** B. Yegnanarayana, "Artificial Neural Networks," PHI, 1999. **4.** Simon Hayking, "Neural Networks and Learning Machines," Pearson, 1999.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Neural Networks and Deep Learning (NNDL) (3-0-0-6)**
2	**Pre-requisite courses(s)**	Exposure to basic concepts in calculus and probability
3	**Course content**	**Introduction to Artificial Neural Networks (ANN) and Deep Learning (DL):** Motivation, basics of ANN, overview of PRML, evolution deep learning and different architectures. Applications of ANN vs DL. **Feedforward Neural Networks (FFNN):** Working principle, basic architecture, analysis of FFNN for different PRML tasks. **Feedback Neural Networks (FBNN):** Working principle, basic architecture, Boltzmann machine, analysis of FFNN for different PRML tasks. **Competitive learning Neural Networks (CLNN):** Working principle, basic architecture, analysis of CLNN for different PRML tasks. **Deep Learning (DL) Architectures:** Deep FFNN, Convolutional neural networks (CNN), Recurrent neural network (RNN), Longterm shortterm memory (LSTM), Generative adversarial network (GAN), DL architectures with attention mechanism. Some recent DL architectures.
4	**Texts/References**	**1.** B. Yegnanarayana, Artificial Neural Networks, PHI, 1999. **2.** Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.

Computer Science Engineering

1	**Title of the course (L-T-P-C)**	**Neural Networks and Deep Learning (NNDL) Laboratory (0-0-3-3)**
2	**Pre-requisite courses(s)**	Currently taking or already taken NNDL theory course
3	**Course content**	The lab will closely follow the theory course. The idea is to have the students implement the basic algorithms on different topics studied in the NNDL theory course.
4	**Texts/References**	**1.** B. Yegnanarayana, Artificial Neural Networks, PHI, 1999. **2.** Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.