**Bansilal Ramnath Agarwal Charitable Trust's**

# Vishwakarma Institute of Technology

***(An Autonomous Institute affiliated to Savitribai Phule Pune University)***

**Structure & Syllabus**

# Honor Courses

***Offered by***

# B.Tech. (Computer Engineering)

**With Effect from Academic Year 2025-26**

**Prepared by: - Board of Studies in Computer Engineering**

**Approved by: - Academic Board, Vishwakarma Institute of Technology, Pune**

**Chairman – BOS** **Chairman – Academic Board**

## Vision of the Institution

"To be globally acclaimed Institute in Technical Education and Research for holistic Socio-economic development".

## Mission of the Institution

- To ensure that 100% students are employable and employed in Industry, Higher Studies, become Entrepreneurs, Civil / Defense Services / Govt. Jobs and other areas like Sports and Theatre.
- To strengthen Academic Practices in terms of Curriculum, Pedagogy, Assessment and Faculty Competence.
- Promote Research Culture among Students and Faculty through Projects and Consultancy.
- To make students Socially Responsible Citizen.

## Vision of the Department

"To be a leader in the world of computing education practising creativity and innovation".

## Mission of the Department

- To ensure students' employability by developing aptitude, computing, soft, and entrepreneurial skills
- To enhance academic excellence through effective curriculum blended learning and comprehensive assessment with active participation of industry
- To cultivate research culture resulting in knowledge-base, quality publications, innovative products and patents
- To develop ethical consciousness among students for social and professional maturity to become responsible citizens

## Knowledge and Attitude Profile (WK)

| WK | WK Statements |
|---|---|
| WK1 | A systematic, theory-based understanding of the natural sciences applicable to the discipline and awareness of relevant social sciences. |
| WK2 | Conceptually-based mathematics, numerical analysis, data analysis, statistics and formal aspects of computer and information science to support detailed analysis and modelling applicable to the discipline. |
| WK3 | A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline. |
| WK4 | Engineering specialist knowledge that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline. |
| WK5 | Knowledge, including efficient resource use, environmental impacts, whole-life cost, reuse of resources, net zero carbon, and similar concepts, that supports engineering design and operations in a practice area. |
| WK6 | Knowledge of engineering practice (technology) in the practice areas in the engineering discipline. |
| WK7 | Knowledge of the role of engineering in society and identified issues in engineering practice in the discipline, such as the professional responsibility of an engineer to public safety and sustainable development. |
| WK8 | Engagement with selected knowledge in the current research literature of the discipline, awareness of the power of critical thinking and creative approaches to evaluate emerging issues. |
| WK9 | Ethics, inclusive behavior and conduct. Knowledge of professional ethics, responsibilities, and norms of engineering practice. Awareness of the need for diversity by reason of ethnicity, gender, age, physical ability etc. with mutual understanding and respect, and of inclusive attitudes. |

## List of Programme Outcomes [PO]

| PO | PO Statements |
|---|---|
| PO1 | **Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified inWK1 to WK4 respectively to develop to the solution of complex engineering problems. |
| PO2 | **Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4) |
| PO3 | **Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5) |
| PO4 | **Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8). |
| PO5 | **Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6) |

**PO6** **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

**PO7** **Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

**PO8** **Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

**PO9** **Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

**PO10** **Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

**PO11** **Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

## List of Programme Specific Outcomes [PSO]

### List of PSO Statements

| | |
|---|---|
| **PSO1** | Demonstrate proficiency in programming with a sound understanding of fundamental computing principles |
| **PSO2** | Conceive well-structured design and proficient implementation to address real world challenges using software paradigms, algorithms and technologies |
| **PSO3** | Acquire and showcase expertise in emerging fields within computer science, engineering, and technology |

## Program Educational Objectives (PEOs)

- Demonstrate application of sound engineering foundations to be a committed technology workforce
- Apply mathematical and computing theory knowledge base to provide realistic computer engineering solutions
- Exhibit problem-solving skills and engineering practices to address problems faced by the industry with innovative methods, tools, and techniques
- Develop professional and ethical practices adopting effective guidelines to acquire desired soft skills in the societal and global context
- Aim for continuing education and entrepreneurship in emerging areas of computing

## Course Name Nomenclature as per NEP (For FY and SY)

| | |
|---|---|
| BSC: Basic Science Course | MDOE: Multi Disciplinary Open Elective |
| ESC: Engineering Science Course | CC: Co-curricular Course |
| PCC: Program Core Course | HSSM: Humanities Social Science and Management |
| PEC: Program Elective Course | IKS: Indian Knowledge System |
| ELC: Experiential Learning Course | FP: Field Project |
| MD: Multi Disciplinary | INT: Internship |

## Nomenclature for Teaching and Examination Assessment Scheme AY 2024-25

| Sr No. | Category | Head of Teaching/ Assessment | Abbreviation used |
|---|---|---|---|
| 1 | Teaching | Theory | Th |
| 2 | Teaching | Laboratory | Lab |
| 3 | Teaching | Tutorial | Tut |
| 4 | Teaching | Open Elective | OE |
| 5 | Teaching | Multi Disciplinary | MD |
| 6 | Teaching | Computer Science | CS |
| 7 | Assessment | Laboratory Continuous Assessment | CA |
| 8 | Assessment | Mid Semester Assessment | MSA |
| 9 | Assessment | End Semester Assessment | ESE |
| 10 | Assessment | Home Assignment | HA |
| 11 | Assessment | Course Project | CP |
| 12 | Assessment | Group Discussion | GD |
| 13 | Assessment | PowerPoint Presentation | PPT |
| 14 | Assessment | Class Test –1 | CT1 |
| 15 | Assessment | Class Test –2 | CT2 |
| 16 | Assessment | Mid Semester Examination | MSE |
| 17 | Assessment | End Semester Examination | ESE |
| 18 | Assessment | Written Examination | WRT |
| 19 | Assessment | Multiple Choice Questions | MCQ |
| 20 | Assessment | Laboratory | LAB |

## Honor Courses

| Sr. No. | Course | Page No. |
|---|---|---|
| 1 | Artificial Intelligence and Machine Learning | 11 |
| 2 | Quantum Computing | 35 |
| 3 | Advanced Algorithms | 53 |
| 4 | Computer Vision | 69 |
| 5 | Cloud Computing | 93 |
| 6 | Natural Language Processing | 118 |
| 7 | Parallel Computing | 144 |
| 8 | Cyber Security | 166 |

## Honor Course: Artificial Intelligence and Machine Learning

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Mathematics for Artificial Intelligence | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Mathematics for Machine Learning | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Optimizations in Machine Learning | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Reinforcement Learning | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Large Language Models | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |

## Honors in Artificial Intelligence and Machine Leaning

### ABCXXX: Mathematics for Artificial Intelligence

Teaching Scheme:
Theory: 2 Hours/Week; Tutorial: 1 Hour/Week
Total Credits: 3

**Course Objectives:**
Students will learn

1. To learn prepositional logic, predicate logic to reason about the relationships between propositions and predicates
2. To learn nonmonotonic reasoning to capture and represent defeasible inferences
3. To learn different trees and search algorithms to find the best possible solution inferences
4. To learn Bayesian networks to solve a problem which has uncertainty due to multiple events.
5. To learn Fuzzy logic use to imitate human reasoning and cognition
6. To learn finding minima to aids in the discovery of a function's lowest extreme values for optimization

**Course Relevance:** Since some of the mathematics used in AI is not part of a standard undergraduate curriculum, students will be learning mathematics and seeing how it is used in AI at the same time. It enhances the ability to design, analyze, and optimize AI algorithms effectively. A solid math background fosters innovation and research in advanced AI techniques

**Syllabus**
**Theory**

**Unit 1: Logic and AI** **(6 Hours)**
What Is Mathematical Logic? Logic and AI, Propositional Logic, Syntax and Semantics of Propositional Logic, Predicate Logic, Syntax and Informal Semantic of Predicate Logic, The Theory of Resolution. Truth versus Proof, Truth, Proof, Resolution and Propositional Calculus, The Resolution Method, Resolution of Horn Clauses, First-Order Predicate Calculus, Skolemization, Unification, Resolution, Soundness and Completeness, Decidability

**Unit 2: Nonmonotonic Reasoning (4 Hours)**
Types of Qualitative Nonmonotonic Reasoning, How Well Do Nonmonotonic Methods Work? Default Reasoning, Normal Default Theories, Other Modifications of Logic, Circumscription, Modal and Autoepistemic Logics, Rule Systems, Basic Concepts of Monotonic Systems, Forward versus Backward Chaining, Negation, Limiting the Effects of Contradictions, Nonmonotonicity, (Semantic Nets, Frames, Manipulating Simple Inheritance Systems, Defeasible Reasoning, The Syntax of Defeasible Reasoning, The Laws of Defeasible Reasoning)

**Unit 3: Tree and Search (6 Hours)**
Problem Spaces and Search Trees, Decision Trees, Simple Search, Iterative-Deepening Search, Heuristic Search - Partial Search: Use a Limited-Depth Search Make a Decision and Start Again, Limited-Depth Search, AND/OR Trees, Alpha-Beta Pruning, Search in game of Chess and Maze

**Unit 4: Bayesian Networks (4 Hours)**
Introduction of Bayesian Nets, Bayesian Networks- Directed Acyclic Graphs, Bayesian Networks and Proof of Theorem, Bipartite Multiple-Diagnosis Problems, Singly Connected DAGs, Deduction, Abduction, and Induction, Multiple Diagnosis in Bipartite Networks, (Irredundant Covers, An Algorithm for Singly Connected Networks, Some Theorems and Proofs - Two Theorems, Formulas for PI and Lambda, Certainty Factors, What Are Certainty Factors? An Interpretation of Certainty Factors, Limits on Certainty)

**Unit 5: Fuzziness and Belief Theory (4 Hours)**
Fuzziness, The Fuzzy Set Concept, Properties of Fuzzy Sets, Fuzzy Predicates, Fuzzy Rule Systems, Dempster-Shafer Belief Theory, Combining Independent Evidence, Looking Backward - Long Chains of Deductions, Robustness, Computational Feasibility, Psychological Validity, Choosing a System.

**Unit 6: Minimization and Feedforward Nets (4 Hours)**
Findings Minima - Types of Algorithms - Newton’s Method, Linear Methods, Quadratic Methods, Introduction to Perceptrons, Algorithm for perceptron, Backpropagation for Feedforward Nets, Assumptions and Notation, Backpropagation, Parameter Issues in Feedforward Nets, Activation Functions, Initialization, Excessive Parameter Growth and Stopping

## Syllabus
## Tutorials

**List of Tutorials**

1. Examples on Propositional Logic
2. Examples on Predicate Logic
3. Examples on Resolution Method
4. Examples on Skolemization, Unification
5. Examples on Nonmonotonic Reasoning
6. Examples on Forward and Backward Chaining
7. Examples on Simple Search Techniques
8. Examples on Heuristics Search Techniques
9. Examples on AND/OR Trees and Alph-Beta Pruning
10. Examples on Bayesian Networks
11. Examples on Bipartite Multiple-Diagnosis Problems, Singly Connected DAGs
12. Examples on Fuzzy Set Properties and Fuzzy Membership Functions
13. Examples on Findings Minima
14. Examples on Perceptron and Backpropagation for Feedforward Nets
15. Examples on Activation Functions

## Course Outcomes

**Course Outcomes**: Student will able to

1. Apply resolution methods to reason about the relationships between propositions and predicates.
2. Select appropriate nonmonotonic reasoning approach to capture and represent defeasible inferences.
3. Apply different trees and search algorithms to find the best possible solution inferences.
4. Comprehend and use Bayesian networks to solve a problem which has uncertainty due to multiple events.
5. Apply Fuzzy logic to imitate human reasoning and cognition.
6. Find minima for the purpose of optimization.

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**

1. E. A. Bender, Mathematical Methods in Artificial Intelligence, IEEE Computer Society Press Los Alamitos, California, SBN: 9780818672002, 9780818672002
2. H. Nelson, Essential Math for AI, ISBN 9781098107581, O'Reilly Media, January 2023.
3. S. Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Pearson Education.
4. K. H. Rosen, Discrete Mathematics and Its Applications, 7th Edition, McGraw-Hill Publication.

**Reference Books: (As per IEEE format)**

1. K. P. Murphy, *Machine Learning: A Probabilistic Perspective,* Cambridge, MA, USA: MIT Press, 2012.
2. S. M. Ross, *Introduction to Probability Models,* 11th edition, Amsterdam, The Netherlands: Elsevier, 2014
3. G. J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, ISBN 0-13-101171-5, Prentice-Hall Inc. Publication.
4. T. M. Mitchell, Machine Learning, ISBN 0070428077, McGraw-Hill Publication.

**For MOOCs and other learning Resources**

1. https://www.coursera.org/learn/foundational-mathematics-for-ai
2. https://www.edx.org/learn/computer-science/columbia-university-essential-math-for-ai
3. https://www.edx.org/certificates/professional-certificate/columbiax-essential-math-for-ai-programming-data-structures
4. https://www.futurelearn.com/courses/essential-mathematics-for-mla-et

# Honors in Artificial Intelligence and Machine Leaning

## ABCXXX: Mathematics for Machine Learning

Teaching Scheme:
Theory: 2 Hours/Week; Tutorial: 1 Hour/Week
Total Credits: 3

**Course Objectives:**
1. To learn Vector Spaces and Analytic Geometry
2. To learn Vector Calculus
3. To learn Linear Regression for Classifiers
4. To learn Logistics Regression for Classifiers
5. To learn Dimensionality Reduction with Principal Component Analysis
6. To learn Classification with Support Vector Machine

**Course Relevance:** Mathematics provides the theoretical foundation needed to understand how machine learning algorithms work. It enhances the ability to design, analyze, and optimize models effectively. Key concepts like linear algebra, calculus, and probability are crucial for data handling and learning mechanisms. It enables interpretation of model behavior, leading to more informed decisions and improvements. A solid math background fosters innovation and research in advanced machine learning techniques

## Syllabus
## Theory

**Unit 1: Vector Spaces and Analytic Geometry (8 Hours: 5 Th + 3 Tut)**
Systems of Linear Equations, Matrices, Solving Systems of Linear Equations, Elementary Transformations, Calculating Inverse, Algorithms for Solving a System of Linear Equations, Vector Spaces, Vector Subspaces, Linear Independence, Basis and Rank, Linear Mappings, Basis Change, Image and Kernel, Affine Spaces, Affine Subspaces, **Analytic Geometry:** Norms, Inner Products, Lengths and Distances, Angles and Orthogonality, Orthonormal Basis, Orthogonal Complement, Inner Product of Functions, Orthogonal Projections, Rotations

**Unit 2: Vector Calculus, Probability and Distributions (8 Hours: 5 Th+3 Tut)**
Differentiation of Univariate Functions, Partial Differentiation and Gradients, Gradients of Vector-Valued Functions, Gradients of Matrices, Useful Identities for Computing Gradients , Backpropagation and Automatic Differentiation, Higher-Order Derivatives, Linearization and Multivariate Taylor Series, Optimization Using Gradient Descent, Constrained Optimization and Lagrange Multipliers, Convex Optimization, **Probability and Distributions:** Construction of a Probability Space, Discrete and Continuous Probabilities, Sum Rule, Product Rule, and Bayes' Theorem, Summary Statistics and Independence, Gaussian Distribution, Conjugacy and the Exponential Family, Change of Variables/Inverse Transform

**Unit 3: Linear and Logistics Regression for Classifiers (8 Hours: 5 Th+3 Tut)**
Problem Foundations, Parameter Estimation, Maximum Likelihood Estimation, Maximum Likelihood Estimation with Features, Estimating the Noise Variance, Overfitting in Linear Regression. Maximum A Posteriori Estimation, MAP Estimation as Regularization, Bayesian Linear Regression: Model, Prior Predictions, Posterior Distribution, Posterior Predictions, Computing the Marginal Likelihood, Maximum Likelihood as Orthogonal Projection, Naive Bayes, Naive Bayes with Minimal Features, Naive Bayes with all Features, Logistic Regression, Generative and Discriminative Classifiers, Components of a probabilistic machine learning classifier, The sigmoid function, Features in Multinomial Logistic Regression, Learning in Logistic Regression, The cross-entropy loss function, Gradient Descent, The Gradient for Logistic Regression, The Stochastic Gradient Descent Algorithm,

**Unit 4: Dimensionality Reduction for Principal Component Analysis**
**(7 Hours: 5 Th+2 Tut)**
Problem Setting, Maximum Variance Perspective, Direction with Maximal Variance, M-dimensional Subspace with Maximal Variance, Projection Perspective, Setting and Objective, Finding Optimal Coordinates, Finding the Basis of the Principal Subspace, Eigenvector Computation and Low-Rank Approximations, PCA Using Low-Rank Matrix Approximations, Practical Aspects, PCA in High Dimensions, Key Steps of PCA in Practice, Latent Variable Perspective, Generative Process and Probabilistic Model, Likelihood and Joint Distribution, Posterior Distribution

**Unit 5: Density Estimation with Gaussian Mixture Models (7 Hours: 5 Th+2 Tut)**
Gaussian Mixture Model, Parameter Learning via Maximum Likelihood, Responsibilities, Updating the Means, Updating the Covariances, Updating the Mixture Weights, EM Algorithm, Latent-Variable Perspective, Generative Process and Probabilistic Model, Posterior Distribution, Extension to a Full Dataset, EM Algorithm Revisited,

.

**Unit 6: Classification with Support Vector Machine (7 Hours: 5 Th+2 Tut)**

Classification with Support Vector Machines, Separating Hyperplanes, Primal Support Vector Machine, Concept of the Margin, Traditional Derivation of the Margin, Why We Can Set the Margin to 1, Soft Margin SVM: Geometric View, Soft Margin SVM: Loss Function View, Dual Support Vector Machine, Dual SVM: Convex Hull View, Kernels, Numerical Solution,

## Syllabus
## Tutorials

**List of Tutorials**

1. Examples on Vector Spaces and Analytic Geometry
2. Examples on Vector Spaces and Analytic Geometry
3. Examples on Vector Spaces and Analytic Geometry
4. Examples on Vector Calculus, Probability and Distributions
5. Examples on Vector Calculus, Probability and Distributions
6. Examples on Vector Calculus, Probability and Distributions
7. Examples on Linear and Logistics Regression for Classifiers
8. Examples on Linear and Logistics Regression for Classifiers
9. Examples on Linear and Logistics Regression for Classifiers
10. Examples on Dimensionality Reduction for Principal Component Analysis
11. Examples on Dimensionality Reduction for Principal Component Analysis
12. Examples on Density Estimation with Gaussian Mixture Models
13. Examples on Density Estimation with Gaussian Mixture Models
14. Examples on Support Vector Machines
15. Examples on Support Vector Machines

## Course Outcomes

**Course Outcomes**: Student will able to

1. Apply Vector Spaces and Analytic Geometry for  Classification
2. Apply Vector Calculus, Probability and Distributions for Classification and Prediction
3. Apply Linear and Logistics Regression for Binary Classification
4. Apply Dimensionality Reduction to reduce Overfitting to improve Model Performance

5. Apply Density Estimation to model the underlying Probability Distribution of data in Probabilistic Modelling
6. Apply Support Vector Machines for Supervised learning mainly for Binary and Multiclass classification with High Dimensional Data

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**

1. M. P. Deisenroth, A. A. Faisal and C. S. Ong, Mathematics for Machine Learning, Cambridge, U.K.: Cambridge University Press, 2020.
2. S. Theodoridis, Machine Learning: A Bayesian and Optimization Perspective, 2nd ed., Cambridge, MA, USA: Academic Press, 2020.
3. G. Strang, *Linear Algebra and Learning from Data, Wellesley, MA, USA: Wellesley-Cambridge Press, 2019.*
4. S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms,* Cambridge, U.K.: Cambridge University Press, 2014.

**Reference Books: (As per IEEE format)**

1. K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, Cambridge, MA, USA: MIT Press, 2012.
2. R. T. Rockafellar, *Convex Analysis*, Princeton, NJ, USA: Princeton University Press, 1997.
3. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, Cambridge, MA, USA: MIT Press, 2016.
4. D. P. Bertsekas, *Nonlinear Programming*, 3rd ed., Belmont, MA, USA: Athena Scientific, 2016.
5. T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., New York, NY, USA: Springer, 2009.
6. S. M. Ross, *Introduction to Probability Models*, 11th ed., Amsterdam, The Netherlands: Elsevier, 2014

**For MOOCs and other learning Resources**

1. https://www.coursera.org/specializations/mathematics-for-machine-learning-and-data-science
2. https://onlinecourses.nptel.ac.in/noc24_ma61/preview
3. https://ocw.mit.edu/courses/18-657-mathematics-of-machine-learning-fall-2015

# Honors in Artificial Intelligence and Machine Leaning

## Optimizations in Machine Learning

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2 Hours / Week
Total Credits: 3

**Prerequisites:** Mathematics for Artificial Intelligence,
Mathematics for Machine Learning

**Course Objectives:**

1. To understand the foundational concepts and types of optimization relevant to training machine learning models.
2. To explore gradient-based methods and their enhancements for efficient optimization in machine learning.
3. To implement and compare advanced optimizers used in deep learning frameworks.
4. To apply constraint-based and regularized optimization techniques for improving model generalization.
5. To introduce and apply global optimization strategies for non-convex and complex ML problems.
6. To examine real-world applications and recent research trends in optimization for ML systems.

## Syllabus
## Theory

**Section 1: Topics/Contents**

**Unit 1: Introduction to Optimization in Machine Learning (4Hours)**

Overview of optimization in AI and ML, Mathematical foundations: functions, gradients, minima/maxima, Convex vs. non-convex optimization, Batch, mini-batch, and stochastic optimization, Role of optimization in training ML models

**Unit 2: Gradient-Based Optimization (5Hours)**

Gradient descent and its variants, Momentum, Nesterov accelerated gradient , Learning rate schedules, Challenges: vanishing/exploding gradients

**Unit 3: Advanced Optimization Algorithms (5 Hours)**
Adam, RMSProp, Adagrad, Adadelta ,Comparison and empirical behavior, Implementation in TensorFlow/ PyTorch, Hyperparameter tuning

**Section 2: Topics/Contents**
**Unit 4: Constrained and Regularized Optimization (4 Hours)**
L1, L2 regularization ,Constraint handling in optimization problems, Duality and Lagrange multipliers, Applications in sparse modelling

**Unit 5: Global Optimization Techniques (5 Hours)**
Evolutionary algorithms, Simulated annealing, Bayesian optimization, Swarm optimization methods

**Unit 6: Optimization Applications and Trends (5 Hours)**
AutoML and hyperparameter tuning, Optimization in deep learning pipelines, Optimization for large-scale systems, Recent trends and research directions

## Syllabus
## Laboratory

**List of Experiments**

1. Implement gradient descent and visualize loss surface for linear regression.
2. Compare performance of SGD, Mini-batch, and Batch gradient descent on a dataset.
3. Apply Adam, RMSProp, and Adagrad optimizers on image classification using CNN.
4. Implement L1 and L2 regularization on logistic regression.
5. Visualize the effect of learning rate schedules using a synthetic dataset.
6. Perform constrained optimization using Lagrange multipliers in Python.
7. Use evolutionary algorithm for function minimization.
8. Implement Bayesian optimization using the scikit-optimize library.
9. Hyperparameter tuning using Grid Search and Random Search.
10. Compare optimization behaviors using PyTorch on different model architectures.

## Syllabus
## Course Project

**List of Course Projects**

1. Train a deep neural network for image classification and optimize using advanced optimizers (Adam, RMSProp).
2. Design and analyze the impact of regularization on a high-dimensional dataset.
3. Develop an AutoML pipeline with hyperparameter tuning.
4. Apply Bayesian optimization to tune parameters of an ML model for text classification.
5. Implement a constrained optimization model for resource allocation.
6. Build an evolutionary algorithm-based solution for parameter tuning in reinforcement learning.
7. Create a visualization tool to compare optimizer convergence behavior.
8. Apply swarm optimization methods (e.g., PSO) for non-convex function optimization.
9. Deploy a TensorFlow model with integrated learning rate scheduling.
10. Comparative study on optimization strategies for large-scale datasets.

## Syllabus Tutorials

**List of Tutorials**

1. Mathematical foundations: derivatives, gradients, Hessians.
2. Convex vs. non-convex optimization problems.
3. Implement and plot convergence of gradient descent.
4. Visual walkthrough of learning rate decay.
5. Understanding momentum and Nesterov optimization.
6. Regularization techniques: hands-on exercises.
7. Constraint handling: KKT and Lagrangian examples.
8. Hyperparameter tuning using Scikit-learn and Keras.
9. Optimization algorithms in TensorFlow/PyTorch.
10. Exploration of optimization libraries: Optuna, Hyperopt, Scikit-Optimize.

## Course Outcomes

**Course Outcomes**: Student will able to

1. Understand the role and types of optimization techniques in ML.
2. Apply gradient-based optimization algorithms to real-world datasets.
3. Compare and analyze different advanced optimization methods.
4. Use constrained and regularized optimization techniques.
5. Implement global optimization techniques in model tuning.
6. Explore recent trends and applications of optimization in ML.

**Books and E-Resources**

**Text Books -**

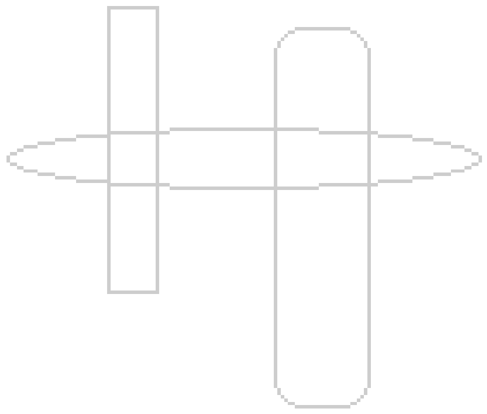
1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.https://www.deeplearningbook.org
2. Dimitri P. Bertsekas, Nonlinear Programming, 3rd Edition, Athena Scientific, 2016.
3. Stephen Boyd, Lieven Vandenberghe, Convex Optimization, Cambridge University Press, 2004.https://web.stanford.edu/~boyd/cvxbook/
4. Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning, Springer, 2nd Edition, 2009.https://web.stanford.edu/~hastie/ElemStatLearn/
5. Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly, 2019.

**For Reference Electronic Book –**

1. Yuxi Liu, Machine Learning for Beginners, Packt Publishing, 2021.
2. Jason Brownlee, Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions, Machine Learning Mastery, 2018.
3. Sebastian Raschka, Python Machine Learning, 3rd Edition, Packt Publishing, 2020.
4. Bing Liu, Learning Optimization Techniques in ML, Academic Press, 2020.
5. Andriy Burkov, The Hundred-Page Machine Learning Book, Andriy Burkov, 2019.

**For MOOCs and other learning Resources**

1. Andrew Ng; Machine Learning Specialization; Coursera
   📎 https://www.coursera.org/specializations/machine-learning-introduction
2. Sebastian Raschka; Deep Learning with PyTorch; GitHub & YouTube
   📎 https://github.com/rasbt/deeplearning-models
3. Jeremy Howard; Practical Deep Learning for Coders; fast.ai
   📎 https://course.fast.ai
4. MIT OpenCourseWare; Optimization Methods in Management Science
   📎 https://ocw.mit.edu/courses/15-053-optimization-methods-in-management-science-spring-2013/
5. Google; Machine Learning Crash Course
   📎 https://developers.google.com/machine-learning/crash-course



# Honors in Artificial Intelligence and Machine Leaning

## Reinforcement Learning

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2 Hours / Week
Total Credits: 3

**Prerequisite:** Mathematics for Artificial Intelligence, Optimization in Machine Learning

**Course Objectives**

1. To understand foundational concepts and formalism of reinforcement learning and its applications.
2. To apply value-based learning techniques like dynamic programming and TD methods.
3. To explore model-free learning through Monte Carlo and Q-learning.
4. To implement function approximation techniques for continuous state spaces.
5. To understand policy gradient methods and their implementations.
6. To apply deep reinforcement learning algorithms to practical applications.

**Syllabus**
**Theory**

### Section 1: Topics/Contents

**Unit 1: Introduction to Reinforcement Learning (4 Hours)**
What is RL, agent-environment interface, rewards, actions, policy, value function, Markov Decision Processes (MDP), goal of RL.

**Unit 2: Dynamic Programming and Planning (5 Hours)**
Bellman equations, policy evaluation, improvement and iteration, value iteration, policy iteration, planning in known MDPs.

**Unit 3: Monte Carlo and Temporal-Difference Learning (5 Hours)**
Model-free prediction and control, Monte Carlo methods, TD learning, SARSA, Q-learning, eligibility traces.

## Section 2: Topics/Contents

**Unit 4: Function Approximation (4 Hours)**
Function approximation for RL, generalization, overfitting, linear and nonlinear function approximators, neural networks in RL.

**Unit 5: Policy Gradient Methods (5 Hours)**
Policy gradients, REINFORCE algorithm, variance reduction, Actor-Critic methods, PPO.

**Unit 6: Deep Reinforcement Learning and Applications (5 Hours)**
Deep Q Network (DQN), AlphaGo, OpenAI Gym, RL for robotics/game playing, safety, ethics and trends in RL.

## Syllabus
## Laboratory

**List of Experiments**

1. Implement value iteration and policy iteration on gridworld.
2. Monte Carlo prediction using simple environment.
3. Implement SARSA and Q-learning on OpenAI Gym.
4. Use function approximation with linear models.
5. Train DQN agent on CartPole.
6. Explore policy gradients using REINFORCE.
7. Train actor-critic model using PPO.
8. Analyze learning curves and convergence behavior.
9. Tune hyperparameters in RL agents.
10. Compare value-based vs policy-based approaches.

## Syllabus
## Course Project

**List of Course Projects**

1. Build RL agent for a custom OpenAI Gym environment.
2. Implement and analyze performance of SARSA and Q-learning.
3. Fine-tune a DQN for an Atari game.
4. Apply RL for pathfinding robot simulation.
5. Design policy gradient model for continuous action space.
6. Create dashboard to visualize training of RL agents.

## Syllabus
## Tutorials

**List of Tutorials**

1. Derivation of Bellman equations.
2. Worked examples on MDPs.
3. Simulation-based policy evaluation.
4. Temporal Difference updates - hands-on.
5. Exploration-exploitation analysis.
6. Neural net design for Q-value approximation.
7. Debugging instability in RL.
8. Policy gradient exercise.
9. Comparison of value functions across methods.
10. Research reading on AlphaGo/DQNs.

## Course Outcomes

**Course Outcomes**: Student will able to

1. Understand the foundational concepts of reinforcement learning.
2. Apply dynamic programming methods for policy learning.
3. Implement TD learning and Monte Carlo methods.
4. Use function approximation for large state spaces.
5. Implement and compare policy gradient methods.
6. Develop real-world RL applications using modern tools.

## Books and E-Resources

**Text Books**

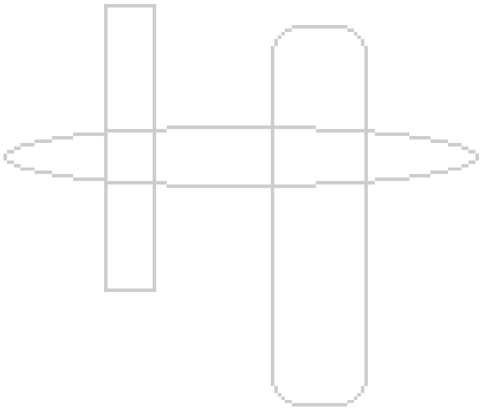
1. Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2nd Edition, 2018.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.

**Reference Books**

1. David Silver, Reinforcement Learning Lectures, DeepMind (Online).
2. Prateek Joshi, Hands-On Reinforcement Learning with Python, Packt, 2020.
3. Maxim Lapan, Deep Reinforcement Learning Hands-On, Packt, 2018.

**MOOCs and Resources**

1. David Silver, Reinforcement Learning, DeepMind: https://www.davidsilver.uk/teaching/
2. OpenAI Gym Documentation: https://www.gymlibrary.dev/
3. Coursera - Reinforcement Learning Specialization (University of Alberta)
4. Deep Reinforcement Learning Nanodegree - Udacity



# Honors in Artificial Intelligence and Machine Leaning

## Large Language Models

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2 Hours / Week
Total Credits: 3

**Prerequisites:** NLP Fundamentals, Deep Learning

**Course Objectives**

1. To introduce fundamentals and evolution of large language models.
2. To understand transformer architecture used in modern LLMs.
3. To explain pretraining and fine-tuning approaches.
4. To apply LLMs to NLP tasks like generation, summarization, QA.
5. To deploy LLMs using open libraries and APIs.
6. To explore ethical issues and trends in LLM research and applications.

## Syllabus
## Theory

**Section 1: Topics/Contents**

**Unit 1: Introduction to Large Language Models** **(5 Hours)**
History of NLP, shift to deep learning, word embeddings, limitations of RNNs, emergence of LLMs.

**Unit 2: Transformer Architecture** **(5 Hours)**
Self-attention, multi-head attention, encoder-decoder models, positional encoding, BERT vs GPT.

**Unit 3: Pretraining and Fine-Tuning** **(5 Hours)**
Masked language modeling, causal modeling, prompt tuning, transfer learning, data preprocessing.

**Section 2: Topics/Contents**

**Unit 4: Applications of LLMs-(5 Hours)**
Text generation, summarization, QA, zero/few-shot learning, code generation, real-world use cases.

**Unit 5: Tools, Libraries, and Deployment (5 Hours)**
HuggingFace Transformers, OpenAI API, tokenization, Gradio, FastAPI, hardware considerations.

**Unit 6: Ethics, Safety, and Future Trends (5 Hours)**
Bias, hallucination, safety concerns, open-source models, instruction tuning, multimodal LLMs.

## Syllabus
## Laboratory

**List of Experiments**

1. Use Hugging Face pipeline for text classification.
2. Generate text using GPT-2.
3. Fine-tune a transformer on custom dataset.
4. Use summarization model on long text.
5. Implement chatbot using OpenAI API.
6. Analyze tokenization and embeddings.
7. Compare outputs of BERT vs GPT.
8. Prompt engineering experiments.
9. Visualize attention weights.
10. Create a FastAPI wrapper for LLM service.

## Syllabus
## Course Projects

**List of Course Projects**

1. Develop an LLM-powered chatbot for domain-specific task.
2. Create summarizer for technical documents.
3. Build QA system using LLM with retrieval-based pipeline.
4. Fine-tune LLM on health/legal/coding data.
5. Build an interactive app using Gradio and HuggingFace.

## Syllabus
## Tutorials

**List of Tutorials**

1. Transformer components walkthrough.
2. Tokenization strategies comparison.
3. Pretraining vs fine-tuning tasks.
4. LLM pipeline in HuggingFace.
5. Prompt tuning techniques.
6. Use case mapping to LLM architectures.
7. Attention visualization.
8. Deployment using Gradio.
9. Error handling in API-based generation.
10. Responsible LLM usage scenarios.

## Course Outcomes

**Course Outcomes**: Student will able to

1. Understand the evolution and fundamentals of large language models.
2. Explain and apply the transformer architecture.
3. Apply pretraining and fine-tuning methods.
4. Implement LLMs for key NLP applications.
5. Utilize tools and frameworks to deploy LLMs.
6. Analyze ethical issues and trends in LLM development.

## Books and E-Resources

**Text Books**

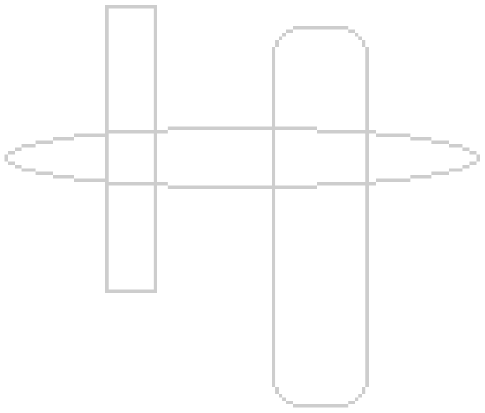
1. Jacob Eisenstein, Natural Language Processing, MIT Press, 2022.
2. Thomas Wolf et al., Transformers for NLP, Packt, 2020.

**Reference Books**

1. Delip Rao, Brian McMahan, Natural Language Processing with PyTorch, O'Reilly, 2019.
2. Dennis Rothman, Transformers for Natural Language Processing, Packt, 2021.
3. Sebastian Ruder, NLP Progress Blog (Online)

**MOOCs and Learning Resources**

1. HuggingFace Course - https://huggingface.co/course
2. OpenAI Documentation - https://platform.openai.com/docs
3. Stanford CS25 - https://web.stanford.edu/class/cs25/
4. DeepLearning.AI NLP Specialization - https://www.coursera.org/specializations/natural-language-processing
5. FastAI NLP Lectures - https://course.fast.ai



# Honors in Artificial Intelligence and Machine Leaning

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

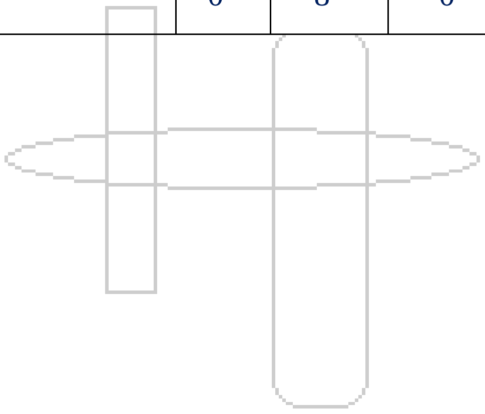
1. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
2. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

3. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
4. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
5. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
6. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
7. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
8. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Quantum Computing

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Quantum Information Theory | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Quantum Algorithms | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Quantum Programming | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Quantum Machine Learning | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Advances in Quantum Computing | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |



## Honors in Quantum Computing

# ABCXXX: Quantum Information Theory

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**

**Theory**

**Unit 1: (Title) (4 Hours)**

Introduction; Classical Shannon Theory: compression and source coding; Shannon entropy; noisy channels and channel capacities; coding; mutual information. Review of Quantum theory: state vectors, qubits, Pauli matrices, unitary transformations, measurement, composite systems and tensor products, quantum gates and circuits, entanglement and Bell inequalities.

**Unit 2: Quantum sets and Protocols ( 5 Hours)**

Noisy quantum states: ensembles and density matrices, POVMs and generalized measurements, separability and entanglement, Kraus maps and quantum instruments, noisy quantum channels, purifications. Unit quantum protocols: entanglement distribution, elementary encoding, superdense coding, quantum teleportation. Resource inequalities. Coherent protocols. Capacity regions.

**Unit 3: (Title) (6 Hours)**

Tools of Quantum Shannon Theory: distance measures, classical information and entropies, quantum information and entropies. Classical typicality: typical sets, typical sequences, Shannon compression, weak and strong typicality, joint typicality, conditional typicality. Quantum typicality: typical subspaces, bipartite and multipartite states, conditional quantum typicality, weak and strong quantum typicality, joint and conditional quantum typicality.

**Unit 4: Quantum Systems ( 5 Hours)**

Schumacher compression. The method of types for classical and quantum systems. Types, type classes and typical type classes. Entanglement manipulation and LOCC.

**Unit 5: Quantum Channels and Communication ( 5 Hours)**

Classical communication over noisy quantum channels. Holevo information, and classical capacity. Examples of quantum channels. Super additivity of classical capacity. Classical communication over entanglement-assisted quantum channels. Capacity theorem.

**Unit 6: Quantum Communication** **( 5 Hours)**

Coherent communication with noisy resources: entanglement-assisted quantum communication; private classical communication. Quantum communication. The quantum capacity theorem. Resource trade-offs and trade-off coding. Nonadditivity and other open problems.

## Syllabus
## Laboratory

To gain hands-on experience with quantum circuits, quantum gates, and entanglement using IBM's Qiskit framework. This lab will deepen understanding of key concepts in quantum information theory through simulation and visualization.

**List of Experiments**

1. Initialize a single qubit in the ∣0⟩|0\rangle∣0⟩ state. Apply a Hadamard gate to create a superposition state. Use Qiskit to simulate and visualize the statevector and measurement statistics.
2. Create a 1-qubit circuit applying a sequence of gates: Hadamard → Pauli-Z → Hadamard. Verify the output and compare it with the expected result analytically.
3. Demonstrate how a unitary matrix transforms the qubit state. Show that the matrix is indeed unitary by calculating $U^\dagger U=I$.
4. Construct a Bell state using two qubits: Apply Hadamard to qubit 0, then CNOT from qubit 0 to 1. Measure and verify that the output corresponds to the $|\Phi+\rangle$ Bell state.
5. Simulate multiple runs to collect statistics. Show that measurement outcomes are correlated.
6. Implement the quantum teleportation protocol in Qiskit. Demonstrate teleportation of a known qubit state from Alice to Bob using entanglement and classical communication.

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Understand quantum information theory
2. Understand quantum states
3. Use Tools of Quantum Theory
4. Illustrate the methods of quantum systems
5. Elaborate quantum channels for communication
6. Apply quantum information theory to solve real world problems.

## Books and E-Resources

**For Reference Print Book -**

1.Mark M. Wilde; 'Quantum Information Theory'; 2nd Edition; Cambridge University Press; 2017

**For Reference Electronic Book –**

1. John Watrous; 'The Theory of Quantum Information'; 1st Edition; Cambridge University Press; 2018; Accessed – April, 26th 2018; Available- chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cs.uwaterloo.ca/~watrous/TQI/TQI.pdf

**For MOOCs and other learning Resources**

1.Prof.Dipan Ghosh; 'Quantum Information and Computing'; Swayam NPTEL; URL – https://archive.nptel.ac.in/course.html; Accessed – August 30, 2016

**Other links**

https://faculty.iisertvm.ac.in/shaji/IISER_Home/Teaching.html

## Honors in Quantum Computing

## ABCXXX: Quantum Algorithms

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**
**Theory**

**Unit 1: Quantum Complexity Theory (5 Hours)**
Review of Quantum Computation Basics: Qubits, quantum gates, circuits, Tensor product, Dirac notation, measurement, Quantum entanglement and no-cloning theorem, Quantum circuit model and unitary evolution,
**Classical complexity classes**: P, NP, BPP, Quantum classes: BQP, QMA, Oracle problems and black-box lower bounds, Quantum query complexity

**Unit 2: Deutsch and Deutsch–Jozsa Algorithm ( 5 Hours)**
Problem setting and quantum solution, Oracle-based computation, Comparison with classical solutions, Circuit implementation,
**Simon's Algorithm**: Hidden subgroup problem, Exponential quantum advantage, Connection to Shor's algorithm, Fourier sampling over binary vectors
.

**Unit 3: Grover's Search Algorithm (5 Hours)**
Amplitude amplification and reflections, Grover iteration and search bounds, Applications to NP search problems, Variants of Grover's algorithm
.

**Unit 4: Quantum Fourier Transform ( 5 Hours)**
**Quantum Fourier Transform (QFT):**Definition and properties of QFT, Efficient circuit implementation of QFT, Role in quantum period-finding.
Shor's Factoring and Discrete Log Algorithm: Integer factorization using QFT, Order-finding problem, Discrete logarithms, Impact on RSA and cryptographic systems

**Unit 5: Quantum Walk Algorithms ( 5 Hours)**
Quantum Walk Algorithms : Quantum walks vs. classical random walks, Search algorithms on graphs, Element distinctness and hitting time.
**Amplitude Estimation and Applications:** Quantum phase estimation, Amplitude estimation algorithm, Applications in machine learning, finance, and simulation

**Unit 6: Adiabatic and Variational Algorithms ( 5 Hours)**

Quantum annealing and AQC, Variational Quantum Eigensolver (VQE), Quantum Approximate Optimization Algorithm (QAOA).

## Syllabus
## Laboratory

To gain hands-on experience with quantum circuits, quantum gates, and entanglement using IBM's Qiskit framework.

**List of Experiments**

1. Simulate quantum algorithms using IBM Qiskit, Cirq, or Q#
2. Implement Deutsch-Jozsa, Grover’s, and Shor’s algorithm
3. Run circuits on real quantum devices via IBM Quantum Experience
4. Visualize QFT and phase estimation circuits
5. Use PennyLane or Qiskit for VQE and QAOA mini projects

## Course Outcomes

**Course Outcomes**: Students will be able to

1. Understand the principles of quantum computation and distinguish between classical and quantum computational models. (Bloom’s Level: Understand)
2. CO2. Analyze the complexity and structure of quantum algorithms such as Deutsch–Jozsa, Simon’s, and Grover’s algorithms. (Bloom’s Level: Analyze)
3. Construct quantum circuits implementing foundational algorithms like Quantum Fourier Transform and phase estimation. (Bloom’s Level: Apply/Create)
4. Implement quantum algorithms using frameworks such as Qiskit or Cirq and simulate their execution. (Bloom’s Level: Apply)
5. Evaluate the computational advantages and limitations of quantum algorithms over classical counterparts. (Bloom’s Level: Evaluate)
6. Design and demonstrate simple quantum algorithmic solutions for problems in search, factorization, and optimization. (Bloom’s Level: Create)

## Books and E-Resources

**For Reference Print Book -**

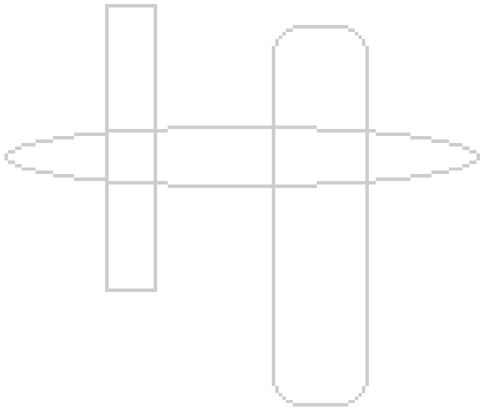
1. Michael Nielsen & Isaac Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2010
2. Phillip Kaye et al., An Introduction to Quantum Computing, Oxford University Press, 2007
3. Ashley Montanaro, Quantum Algorithms – University of Bristol notes

**For Reference Electronic Book –**

1. Ronald de Wolf, Lecture Notes on Quantum Algorithms – Available online
2. Qiskit Textbook, Learn Quantum Computation Using Qiskit – https://qiskit.org/textbook

**For MOOCs and other learning Resources**

1. Quantum Computing with Qiskit and Advanced Algorithms-Coursera - https://www.coursera.org/learn/packt-quantum-computing-with-qiskit-and-advanced-algorithms-wils7



## Honors in Quantum Computing

## ABCXXX: Quantum Programming

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

### Syllabus
### Theory

**Unit 1: Introduction to Quantum Programming (5 Hours)**
Introduction to Quantum Programming: Classical vs. quantum programming paradigms, Quantum circuit model vs. measurement-based model, Overview of quantum programming languages: Qiskit, Q#, Cirq, PennyLane,
**Quantum Data and Qubits:** Representing quantum states and operations, Bra-ket notation, matrices, and tensor products, Basic gates: X, H, Z, S, T, Multi-qubit gates: CNOT, Toffoli, SWAP, Creating and manipulating entanglement

**Unit 2: Programming Quantum Circuits ( 5 Hours)**
Programming Quantum Circuits: Writing and visualizing circuits, Gate decomposition, Circuit depth and width, Using simulators: Qiskit Aer, Cirq simulator.

**Unit 3: Measurement and Classical Control (5 Hours)**
Measurement operators and post-measurement state, Conditional logic and classical bits, Repeated execution ("shots") and probabilistic results, Controlling quantum flow with classical registers.

**Unit 4: Quantum Programming with Qiskit ( 5 Hours)**
Installing and setting up Qiskit, Circuit construction, simulation, and visualization, Working with IBM Quantum Experience, Using transpilers, backends, and noise models, Pulse programming basics.

**Unit 5: Quantum Programming with Q# and Cirq ( 5 Hours)**
Introduction to Q# syntax and quantum operations, Basic operations and simulator usage, Cirq for Google quantum processors, Comparison: Qiskit vs Q# vs Cirq.

**Unit 6: Quantum Communication ( 5 Hours)**
**Quantum Debugging, Optimization & Compilation:** Error analysis and circuit optimization, Gate fusion, circuit folding, Backend-aware compilation, Introduction to error mitigation strategies
.

## Syllabus
## Laboratory

To gain hands-on experience with quantum circuits, quantum gates, and entanglement using IBM's Qiskit framework.

**List of Experiments**

1. Simulate entangled states and run Bell test
2. Implement Grover’s search using Qiskit
3. Visualize quantum states and Bloch spheres
4. Perform variational optimization using VQE
5. Compare outputs from simulators and real quantum hardware

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Understand the fundamentals of quantum computation and the structure of quantum programs. (Bloom’s Level: Understand)
2. Construct quantum circuits using gates, registers, and measurement operations. (Bloom’s Level: Apply)
3. Implement quantum algorithms using quantum programming languages like Qiskit or Q#.→ (Bloom’s Level: Apply/Create)
4. Analyze quantum program behavior under simulation and on real quantum devices. → (Bloom’s Level: Analyze)
5. Evaluate quantum circuit performance in terms of depth, noise tolerance, and backend compatibility. (Bloom’s Level: Evaluate)
6. Design and deploy small-scale quantum applications by integrating quantum subroutines with classical logic.→ (Bloom’s Level: Create)

## Books and E-Resources

**For Reference Print Book -**

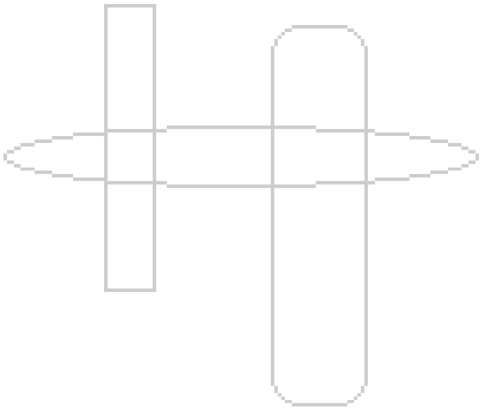
1. A. Montanaro & T. Rudolph, Introduction to Quantum Programming (Lecture notes)
2. Robert Sutor, Dancing with Qubits, Packt, 2019
3. Jack D. Hidary, Quantum Computing: An Applied Approach, Springer, 2nd Ed., 2021

**For Reference Electronic Book –**

1. Qiskit Textbook: https://qiskit.org/textbook
2. Microsoft Quantum Docs: https://docs.microsoft.com/en-us/azure/quantum/

**For MOOCs and other learning Resources**

1. Coursera- Quantum Computing with Qiskit and Advanced Algorithms- https://www.coursera.org/learn/packt-quantum-computing-with-qiskit-and-advanced-algorithms-wils7



## Honors in Quantum Computing

## ABCXXX: Quantum Machine Learning

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**

**Theory**

**Unit 1:  Introduction to Quantum Machine Learning (5 Hours)**

Overview of ○ Machine Learning, Deep Learning and Artificial Intelligence Basics, Machine Learning Algorithms , Deep Learning Algorithms,  Evolutionary Learning Algorithms
Overview of classical ML vs. quantum ML, Motivation and applications of QML, Quantum advantage in ML problems, Quantum data and quantum features

**Unit 2: Mathematical Foundations ( 5 Hours)**

Review of vector spaces, Hilbert space, Tensor products, Dirac notation, Unitary operations and measurement, Basics of supervised and unsupervised learning
**Quantum Computing Primitives:** Qubits, quantum gates, and circuits, Quantum measurement and probability, Quantum state preparation for ML, Programming QML circuits with Qiskit, Cirq, or PennyLane.

**Unit 3: Quantum Data Encoding (5 Hours)**

Amplitude encoding, Basis encoding, Angle (parametric) encoding, Circuit examples for encoding classical datasets into quantum states.
**Quantum Distance and Kernels:** Quantum distance metrics, Fidelity, trace distance, quantum inner product, Quantum kernel methods (Quantum SVMs), Kernel estimation using quantum circuits

**Unit 4: Variational Quantum Circuits ( 5 Hours)**

Variational approach and hybrid quantum-classical models, Parameterized quantum circuits (PQCs), Cost function design and optimization, Barren plateaus and training challenges,

**Unit 5: Supervised and Unsupervised Learning with QML ( 5 Hours)**

Supervised Learning with QML, Quantum classifiers using VQCs, Quantum-enhanced k-NN and perceptron models, Quantum Support Vector Machines, Use cases: image classification, simple datasets.

Unsupervised Learning with QML: Quantum k-means clustering, Principal Component Analysis (qPCA), Quantum generative models (intro to QGANs),

**Unit 6: Quantum Reinforcement and Deep Learning ( 5 Hours)**
Introduction to Quantum Reinforcement Learning, Quantum Boltzmann machines, Quantum Neural Networks, QNN implementation using PennyLane or TensorFlow Quantum.
QML for quantum chemistry, finance, pattern recognition, Current research challenges and future directions, Limitations of current quantum hardware for ML, Emerging trends: quantum advantage, noisy intermediate-scale quantum (NISQ) era

## Syllabus
## Laboratory

To gain hands-on experience with quantum circuits, quantum gates, and entanglement using IBM's Qiskit framework.

**List of Experiments**

2. Encode classical data into quantum states
3. Implement quantum SVM using Qiskit Machine Learning
4. Train VQCs on small datasets (e.g., Iris dataset)
5. Simulate QNNs using PennyLane or TensorFlow Quantum
6. Run quantum ML circuits on IBM/Qiskit

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Understand the principles of quantum computing and classical machine learning, and their intersection in quantum machine learning (QML). (Bloom’s Level: Understand)
2. Apply quantum data encoding techniques and build simple quantum circuits for machine learning tasks using Qiskit, Cirq, or PennyLane. (Bloom’s Level: Apply)
3. Construct hybrid quantum-classical models such as Variational Quantum Circuits (VQCs) for supervised and unsupervised learning problems. (Bloom’s Level: Create)
4. Analyze the performance of QML algorithms such as quantum SVMs, quantum k-means, and quantum PCA in comparison to classical counterparts. (Bloom’s Level: Analyze)

5. Evaluate the strengths and limitations of QML algorithms in real-world domains like classification, optimization, and generative modeling. → (Bloom's Level: Evaluate)
6. Design and implement a QML project involving quantum circuits for practical applications using real or simulated quantum backends.  (Bloom's Level: Create)

## Books and E-Resources

**For Reference Print Book -**

1. Maria Schuld & Francesco Petruccione, Machine Learning with Quantum Computers, Springer, 2nd Ed., 2021
2. Maria Schuld & Nathan Killoran, Introduction to Quantum Machine Learning, arXiv preprint
3. Jack D. Hidary, Quantum Computing: An Applied Approach, Springer, 2021

**For Reference Electronic Book –**

1. Qiskit Machine Learning Docs: https://qiskit.org/learn/machine-learning/
2. PennyLane Documentation: https://pennylane.ai/
3. TensorFlow Quantum Guide: https://www.tensorflow.org/quantum

**For MOOCs and other learning Resources**

## Honors in Quantum Computing

## ABCXXX: Advances in Quantum Computing

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

### Syllabus

### Theory

**Unit 1:  Quantum Computational Models Beyond Circuits (5 Hours)**

Measurement-based quantum computation (MBQC), Topological quantum computation, Adiabatic quantum computation, Hamiltonian complexity

**Quantum Error Correction & Fault Tolerance:** Basics of quantum noise and decoherence, Quantum error-correcting codes: bit-flip, phase-flip, Shor, Steane codes, Stabilizer formalism and surface codes, Fault-tolerant quantum computation, Threshold theorem and logical qubits

**Unit 2: Quantum Hardware and Physical Realizations ( 5 Hours)**

Overview of quantum hardware platforms: superconducting qubits, ion traps, photonic systems, neutral atoms, Qubit initialization, gate fidelity, and coherence time, Quantum control systems, Quantum hardware stack (firmware, compilers, control electronics)

**Unit 3: Quantum Complexity Theory (5 Hours)**

Complexity classes: BQP, QMA, QIP, QSZK, Classical vs. quantum complexity, Completeness and hardness results in quantum settings, Interactive proofs and post-quantum complexity
.

**Unit 4: Quantum Cryptography and Post-Quantum Security ( 5 Hours)**

Quantum key distribution (QKD): BB84, E91 protocols, Quantum coin flipping, bit commitment, Position-based cryptography, Post-quantum cryptography (PQC): Lattice-based, code-based, multivariate, isogeny-based cryptosystems, Security proofs under quantum adversaries,

**Unit 5: Quantum Communication and Networks ( 5 Hours)**

Quantum teleportation, Superdense coding, Quantum repeaters and entanglement swapping. Quantum network topologies, Distributed quantum computing (DQC)

**Unit 6: Quantum Simulation and Applications** **( 5 Hours)**
Quantum simulation of physical and chemical systems, Analog vs. digital quantum simulators, Fermionic simulations and second quantization, Hubbard model, quantum phase transitions
Quantum compilers and transpilers, Quantum circuit optimization techniques, Cross-platform development: Qiskit, Cirq, Q#,  Real-time interfacing with hardware (Pulse/Qiskit Pulse), Quantum benchmarking and verification.

## Syllabus
## Laboratory

To gain hands-on experience with quantum circuits, quantum gates, and entanglement using IBM's Qiskit framework.

**List of Experiments**

1. Simulate quantum error correction codes
2. Implement teleportation and superdense coding using IBM Qiskit
3. Design a simple MBQC circuit using graph state
4. Explore quantum benchmarking with Qiskit Ignis
5. Deploy small quantum networks using simulators

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Understand advanced models of quantum computation such as adiabatic, topological, and measurement-based quantum computing.→ (Bloom’s Level: Understand)
2. Analyze sources of quantum noise and implement quantum error-correcting codes using the stabilizer formalism. → (Bloom’s Level: Analyze)
3. Compare various quantum hardware platforms in terms of gate fidelity, qubit stability, and scalability. → (Bloom’s Level: Evaluate)
4. Apply quantum cryptographic protocols such as BB84 and superdense coding in simulated environments. → (Bloom’s Level: Apply)
5. Design simple quantum communication protocols and explain the principles of quantum networks and entanglement distribution.→ (Bloom’s Level: Create)
6. Evaluate quantum complexity classes and understand their implications on classical vs. quantum computational boundaries.→ (Bloom’s Level: Evaluate)

## Books and E-Resources

**For Reference Print Book -**

1. Michael Nielsen & Isaac Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 10th Anniversary Edition
2. Daniel Gottesman, Stabilizer Codes and Quantum Error Correction, Caltech Lecture Notes
3. Mark M. Wilde, Quantum Information Theory, Cambridge University Press, 2nd Edition
4. Peter Selinger, Quantum Programming Languages

**For Reference Electronic Book –**

1. IBM Qiskit Textbook, especially sections on Pulse, error correction, and advanced protocols – https://qiskit.org/textbook
2. John Preskill, Lecture Notes on Quantum Computation – http://theory.caltech.edu/~preskill/ph219/

**For MOOCs and other learning Resources**

1. ETH Zurich- Quantum Cryptography-https://www.edx.org/course/quantum-cryptography
2. Caltech (John Preskill)- Quantum Error Correction- https://www.edx.org/course/quantum-error-correction
3. TU Delft- Quantum Hardware and Realization- https://www.futurelearn.com/courses/the-quantum-internet

## Honors in Quantum Computing

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a  project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

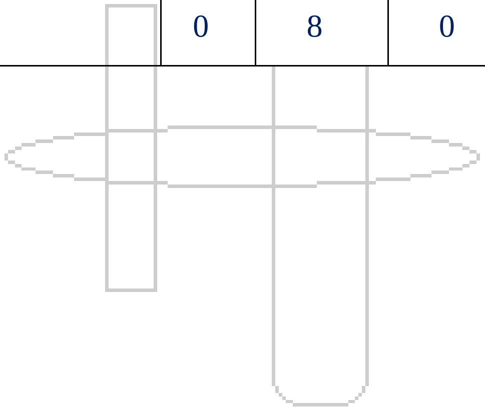
9. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
10. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

11. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
12. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
13. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
14. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
15. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
16. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well

## Honor Course: Advanced Algorithms

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Fundamentals of Algorithms and Complexity | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Advances in Algorithms | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Randomized Algorithms | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Linear Programming, Flow and Matching | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Intractability, Approximation and Heuristics | 3 | 2 | 0 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |



## Honors in Advanced Algorithms

## Fundamentals of Algorithms and Complexity

Teaching Scheme:
Theory: 2 Hours/week; Laboratory: 2 Hours/Week
Total Credits: 3

**Prerequisites:**
First course on programming, mathematical maturity.

**Syllabus**
**Theory**

**Unit 1: Introduction** **(5 Hours)**
Introduction to algorithms, role of algorithms in computing, revisiting some simple algorithms: searching (linear, Fibonacci, binary search and its variants), sorting algorithms (insertion, selection, merge, quick, heap, bucket), proving correctness of algorithms, role of invariants, formulating invariants for some simple programs, notion of time and space complexity, worst-case, best-case, average-case complexity through suitable examples. Advarsary lower bound technique, why proving lower bounds is difficult?

**Unit 2: Tools for analysing complexity** **(4 Hours)**
Growth of a function, asymptotic notations (Big Oh, small oh, Big Omega, Theta notations), interrelation between them, some properties of asymptotic notations such as transitivity, reflexivity, symmetry, anti-symmetry, ordering standard functions based on asymptotic complexities.
Using recurrence relations and mathematical induction to get asymptotic bounds on time complexity. Master's theorem and applications.
Amortized complexity: aggregate analysis, method of potentials.

**Unit 3: Algorithm design paradigms** **(5 Hours)**
Introduction to fundamental algorithm design paradigms through suitable interesting examples: divide and conquer, dynamic programing, greedy, backtracking, branch and bound.

**Unit 4: Combinatorial problems** **(5 Hours)**
L shaped tiling, Finding majority element, Finding median, longest increasing subsequence, patience sorting, edit distance, longest common subsequence, largest sum contiguous blocks in 1-D, 2-D arrays, counting balanced parenthesis strings, counting

binary trees, matrix chain multiplication, optimal binary search tree construction, coin change problem.

**Unit 5: Enumeration and random generation (4 Hours)**
Enumeration and random generation of different combinatorial objects: subsets, subsets of size k, permutations, combinations, binary trees, balanced parenthesis strings, derangements, binary strings.

**Unit 6: Strings (5 Hours)**
Naïve string matching algorithm, the Rabin-Karp algorithm, string matching using finite automata, the Knuth-Morris-Pratt algorithm, Problem solving on strings (palindrome check, string rotation, longest palindrome substring, roman numeral to integer conversion, integer to roman numeral conversion, edit distance between strings, anagrams), text compression and Huffman encoding.

## Syllabus
## Laboratory

**List of Experiments:**

1. Write a program to implement Merge sort and Quick sort and find time complexity.
2. Write a program to implement Josepher's problem, define recurrence relation and find time complexity.
3. Write a program to implement Huffman coding
4. Write a program to implement matrix chain multiplication
   a. Write a program to find optimal binary search tree
5. Write a program for n-queens problem.
6. Write a program for enumeration of permutation and combination
7. Implement Rabin-Karp algorithm
8. Implement Knuth-Morris-Pratt algorithm
9. Write a program for roman numeral to integer conversion

**Text Books:** ***(As per IEEE format)***

1. Cormen, Leiserson, Rivest and Stein "Introduction to Algorithms" ,PHI 3nd edition, 2009. ISBN 81-203-2141-
1. Jon Kleinberg, Eva Tardos "Algorithm Design", Pearson, 1st edition, 2005. ISBN 978-81-317-0310-6
2. Dasgupta, Papadimitriu, Vazirani "Algorithms" McGraw-Hill Education; 1 edition (September 13, 2006), ISBN-10: 9780073523408, ISBN-13: 978-0073523408

# Honors in Advanced Algorithms

## Advances in Algorithms

Teaching Scheme:
Theory: 2 Hours/week; Laboratory: 2 Hours/Week
Total Credits: 3

**Prerequisites:**
First course on algorithms.

**Unit 1: Numbers** **(5 Hours)**
Big number arithmetic (representing numbers as sequence of digits and carrying out basic arithmetic operations), Karatsuba's integer multiplication, fast modular exponentiation, perfect power testing, greatest common divisors, complexity analysis for Euclid's algorithm for gcd, modular inversion and its application, prime numbers, Miller-Rabin primality test, generation of large primes, discrete log problem.

**Unit 2: Graphs** **(5 Hours)**
Revisiting graph traversals depth first search (DFS), breadth first search (BFS), connected components of graph, detecting cycles in graph, topological sorting, directed graphs, strongly connected components, Kosaraju's algorithm, diameter of graph, testing if graph has Eulerian tour, testing if graph is planar, testing if graph is bi-partite.

**Unit 3: Shortest Path and MST algorithms** **(4 Hours)**
Shortests path problem: shortest path in directed acyclic graphs, Bellman-Ford algorithm, correctness proof for Dijkstra's algorithm, All pair shortest paths, Johnson's algorithm
Minimum Spanning Tree problem: Growing a minimum spanning tree, Efficient implementation of Prim's algorithm, Union-Find data structure and efficient implementation of Kruskal's algorithm, correctness proof for Prim's and Kruskal's algorithms.

**Unit 4: Computational geometry** **(5 Hours)**
Determining if a pair of line segments intersect, finding convex hull of points in 2-D: Graham's scan and its correctness, Jarvis's march algorithm and its correctness, closest pair of points, Voronoi diagrams and applications, Range queries in 1-D, 2-D.

**Unit 5: Introduction to streaming algorithms (5 Hours)**

The data stream model, streaming algorithms, how to measure quality of a streaming algorithms, Finding frequent items: Misra-Gries algorithm, estimating number of distinct elements: Tidemark algorithm, better estimates for distinct elements, approximate counting.

**Unit 6: Algorithms in Linear Algebra (4 Hours)**

Fast Fourier Transform (FFT) algorithm and its applications, computing determinant of integer matrix, computing rank of integer matrix, solving system of linear equations

## Syllabus
## Laboratory

**List of assignments:**

1. Implement fast modular exponentiation algorithm
2. Miller-Rabin primality test
3. Topological sorting algorithm
4. Strongly connected components of directed graph (Kosaraju’s algorithm)
5. Planarity testing algorithm
6. Johnson’s algorithm for shortest path
7. Kruskal algorithm efficient implementation using Union-Find data structure
8. Jarvis March algorithm for convex hull computation
9. Graham’s scan algorithm for convex hull computation
10. 2-D range queries
11. Visualization of Voronoi-diagram in 2-D
12. Streaming algorithms for some simple problems, computing average, median, frequent elements
13. FFT algorithm
14. Solution of system of linear equations using Gaussian elimination
15. Solution of system of linear equation: robust iterative method

## Books and E-Resources

**Text Books:** ***(As per IEEE format)***

1. Cormen, Leiserson, Rivest and Stein “Introduction to Algorithms” ,PHI 3nd edition, 2009. ISBN 81-203-2141-
2. Jon Kleinberg, Eva Tardos “Algorithm Design”, Pearson, 1st edition, 2005. ISBN 978-81-317-0310-6
3. Dasgupta, Papadimitriu, Vazirani “Algorithms” McGraw-Hill Education; 1 edition (September 13, 2006), ISBN-10: 9780073523408, ISBN-13: 978-0073523408
4. Amit Chakrabarti, “CS85: Data Stream Algorithms”, Lecture Notes, Fall 2009, https://www.cs.dartmouth.edu/~ac/Teach/CS85-Fall09/Notes/lecnotes.pdf

## Honors in Advanced Algorithms

## Randomized Algorithms

Teaching Scheme:
Theory: 2 Hours/week; Laboratory: 2 Hours/Week
Total Credits: 3

**Prerequisites:**
Prior introduction to basic probability theory and basic algorithms is helpful though it is not assumed.

**Unit I Basic Discrete Probability (5 Hours)**
Introduction to randomization in computation and some simple randomized algorithms., Revisiting concepts from basic discrete probability: definition of probability, examples, independence of events, conditional probability, union bound, inclusion exclusion, Bayes' rule, discrete random variables, expectation, variance, linearity of expectation, sum of independent random variables, standard distributions (Bernoulli, Binomial, Geometric), coupon collector problem, birthday paradox, probabilistic recurrences.
Indicator random variables and their role in algorithm analysis. Las-Vegas and Monte-Carlo algorithms (with examples: randomized quick sort, Karger's min-cut algorithm.)

**Unit 2: Tail Inequalities and applications (5 Hours)**
Moments and deviation, occupancy problem, Markov and Chebyshev inequalities and some applications, randomized selection, order statistics, weak law of large numbers, stable marriage problem and principle of deferred decision, coupon collector problem and sharp threshold, Chernoff's bound and some applications. Basic randomized complexity classes RP, Co-RP, ZPP, BPP and their interrelations, probability amplification in RP and BPP, Yao's min-max principle and application.

**Unit 3: Analyzing randomized data structures (4 Hours)**
Fundamental data structuring problem, Revisiting Random treaps, skip-lists data structures. Mulmuley games and analysis of treaps, analyzing expected space complexity of skip lists, bloom filters, hash tables, universal hash families, random graph models and some applications.

**Unit 4: Aalgebraic techniques (5 Hours)**
Ppolynomial identity testing, Schwartz-Zippel lemma and applications (with examples verifying matrix multiplication, testing equality of strings, perfect matching problem for bipartite graphs), Mulmuley-Vazirani-Vazirani isolation lemma and application to

matching problem. Number theoretic algorithms (finding quadratic non-residues, primality testing), introduction to probabilistic methods.

**Unit 5: Applications in geometric problems (4 Hours)**
Randomized incremental constructions, convex hulls, duality, half space intersection, Delaunay triangulations, diameter of point set.

**Unit 6: Markov Chains and Random Walks (5 Hours)**
Markov chains: definition, representations, randomized algorithm for 2-SAT and 3-SAT, classifying states of Markov chains, Gambler's ruin, stationary distributions. Random walks on undirected graphs, cover time, hitting time, commute time, graph connectivity, electrical networks, introduction to expander graphs. Expanders and rapidly mixing random walks.

## Syllabus
## Laboratory

**List of Assignments:**

1. Stable marriage problem
2. Randomized quick sort
3. Karger's min-cut algorithm
4. Random treaps
5. Skip lists
6. Construction of hash functions
7. Perfect matchings in bipartite graphs using Schwartz-Zippel lemma
8. Isolation Lemma and application
9. Delaunay triangulation
10. Diameter of point set
11. Randomized algorithm for 2-SAT and 3-SAT
12. Random walks on graph: computing cover-time, hitting time, commute time

## Books and E-Resources

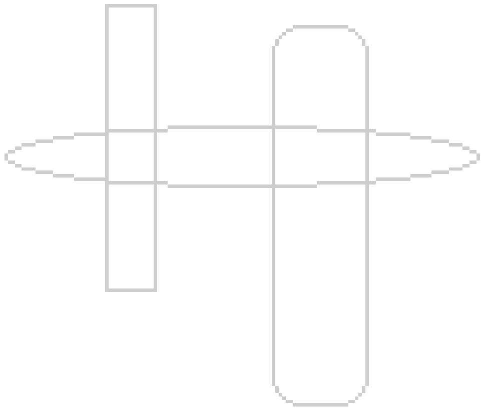
**Text books:**

1. Randomized Algorithms by Rajeev Motwani and Prabhakar Raghavan (Cambridge University Press)
2. Probability and computing by Michael Mitzenmatcher, Eli Upfal

3. Introduction to Algorithms by Thomas H. Coreman, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein ( MIT Press)

4. **References:**

1. A course in probability theory by Kai Lai Chung (Academic Press)
2. An Introduction to Probability Theory and Its Applications Vol I by William Feller (Wiley)



## Honors in Advanced Algorithms

## Linear Programming, Flow and Matching

Teaching Scheme:
Theory: 2 Hours/week; Laboratory: 2 Hours/Week
Total Credits: 3

**Prerequisites**:
Mathematical maturity, a prior introduction to discrete structures is helpful though it is not presumed.

**Unit 1: Introduction to Linear Programming (5 Hours)**
Linear optimization problem, linear programs, basic terminology: objective function, feasible solutions, geometric interpretation, LP and ILP, fundamental theorem of linear inequalities, cones, polyhedra and polytopes, structure of polyhedra.

**Unit 2: LP-duality (4 Hours)**
LP duality and its geometric interpretation, Farkas' Lemma and its variants, strict inequalities, complementary slackness, Motzkin's transposition theorem, Gordon's theorem.

**Unit 3: Solutions of Linear Programs (5 Hours)**
The simplex method, tableau form, pivot selection, cycling and complexity, worst-case and analysis of simplex, primal-dual method for solution of LP, Fourier-Motzkin elimination, introduction to ellipsoids and polynomial time algorithms for LP.

**Unit 4: Maximum flow (6 Hours)**
Flow networks, flow and its properties, maximum flow problem, cuts in the graph, max-flow min-cut theorem, Ford-Fulkerson algorithm, Edmond-Karp algorithm. Some applications of max-flow such as: disjoint paths and network connectivity, Census tabulation, Airline scheduling, Image segmentation, bipartite matchings.

**Unit 5: Matchings (4 Hours)**
Matchings, Perfect Matchings in the graph, Maximum cardinality matching, Matchings in Bipartite graphs, Vertex cover, Konig's theorem, system of distinct representatives, Hall's Marriage theorem, Matchings and flows, Matching Polytope.

**Unit 6: Algorithms for Matching problem (4 Hours)**

Flow based algorithm, Tutte’s theorem, simple randomized algorithm for perfect matching problem using Schwarz-Zipple lemma, Edmond-Gallai structure theorem and efficient algorithm.

## Syllabus

## Laboratory

**List of Assignments:**

1. Visualization of Linear Programs in 2-D
2. Simplex algorithm
3. Fourier-Motzkin elimination
4. Ford-Fulkerson algorithm
5. Edmond-Karp algorithm
6. Image segmentation using network flows
7. Matchings in bipartite graph
8. Tutte’s theorem and perfect matching in graphs
9. Edmond-Gallai structure theorem and deterministic algorithm for matching problem

## Books and E-Resources

**Reference books:**

1. Theory of Linear and Integer Programming by Alexander Schriever (John Wiley & Sons)
2. Linear Programming by Vasek Chvatal (W. H. Freeman)
3. Matching Theory by Lovasz and Plummer ( North-Holland Mathematical Studies)
4. Randomized Algorithms by Rajeev Motwani and Prabhakar Raghavan (Cambridge University Press)
5. Introduction to Algorithms by Thomas H. Coreman, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein ( MIT Press)

# Honors in Advanced Algorithms

## Intractability, Approximation and Heuristics

Teaching Scheme:
Theory: 3 Hours/week; Laboratory: 2 Hours/Week
Total Credits: 4

**Section I: NP-completeness theory and Approximation Algorithms**

**Unit 1: Complexity classes and Polynomial time reductions (4 Hours)**

Decision, Search and optimization problems, complexity classes P, NP, coNP, and their interrelation, Notion of polynomial time many one reductions reduction, properties of reductions, NP-hardness, NP-completeness, Cook-Levin theorem and implication to P versus NP question, NP-hardness of halting problem.

**Unit 2: NP-completeness (5 Hours)**

Proving NP-Completeness of some classical problems: satisfiability problem (SAT), Circuit-SAT, 3-CNF SAT, vertex cover, independent set, clique, Hamiltonian-circuit problem, subset sum, integer linear programming (ILP), closest lattice point problem**.** Modeling NP-complete problems using ILP.

**Unit 3: Introduction to Approximation Algorithms (5 Hours)**

Introduction to approximation algorithms, NP-hard optimization problems, lower bounding OPT, approximation algorithms for some classical problems: set-cover, vertex cover, traveling salesman problem(TSP), metric-TSP, knapsack, shortest vectors in integer lattices. Linear programming based algorithms, LP relaxation, LP duality, LP rounding strategy and primal-dual schema, set-cover and some other examples using LP based techniques, maximum satisfiability. Hardness of approximation for general TSP.

**Section II: Bio-inspired heuristics**

**Unit 4: Natural and Evolutionary Computing (5 Hours)**

From nature to natural computing, sample idea, Philosophy of natural computing, Natural computing approaches, Conceptualization – general concept, Problem solving as a search track, Hill climbing, Simulated annealing

Evolutionary computing: Evolutionary biology, Evolutionary computing – standard evolutionary algorithm; Genetic algorithm, evolutionary strategies, Evolutionary programming, genetic programming

**Unit 5: Swarm Intelligence (5 Hours)**
Swarm intelligence-biological motivation, from natural to artificial, standard algorithm of Ant colony optimization, Ant clustering algorithm, Particle swarm optimization
Biological motivation, from natural to artificial, standard algorithm of cuckoo search, bat algorithm, flower pollination, firefly algorithm, framework for self-tuning algorithms - case study of firefly algorithm

**Unit 6: Immune Systems (4 Hours)**
Immune system, Artificial immune systems - biological motivation, Design principles, main types of algorithms - Bone marrow, Negative selection, Clonal selection, Continuous immune network models, Discrete immune network models, Scope of artificial immune systems

## Syllabus
## Laboratory

**List of Assignments:**

1. Backtracking based algorithms for NP-complete problems
2. Backtracking based algorithms for NP-complete problems
3. Modelling NP complete problems using Integer Linear programs and using ILP solvers
4. Approximation algorithm for Vertex cover
5. Approximation algorithm for metric TSP
6. Approximation algorithm for set-cover
7. Approximation algorithm for knapsack
8. Hill climbing algorithm
9. Genetic algorithm
10. Ant colony optimization algorithm
11. Particle swarm optimization

## Syllabus
## Tutorial

Problem solving based on

1. NP Hardness
2. NP Completeness
3. Satisfiability problem (SAT)

4. Integer linear programming
5. Traveling salesman problem(TSP)
6. Genetic algorithm
7. Particle swarm optimization
8. firefly algorithm

## Books and E-Resources

**Reference books:**

1. Approximation Algorithms by Vijay V. Vazirani (Springer)
2. Introduction to Algorithms by Thomas H. Coreman, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein (MIT Press)
3. Computers and Intractability: A Guide to the Theory of NP-Completeness by Garey and Johnson (W. H. Freeman)
4. Vasuki A. - Nature Inspired Optimization Algorithms-CRC Press (2020)
5. Andreas Antoniou, Wu-Sheng Lu - Practical Optimization. Algorithms and engineering applications.-Springer (2020)
6. L. N. de Castro, “Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications”, 2006, CRC Press, ISBN-13: 978-1584886433

## Honors in Advanced Algorithms

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

17. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
18. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

19. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
20. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
21. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
22. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
23. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
24. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Computer Vision

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Computer Graphics | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Image Processing | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Computer Vision | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Introduction to Robot Learning | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Visual Learning and Recognition | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |

## Honors in Computer Vision

## ABCXXX: Computer Graphics

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

**Course Objectives:**

1. To introduce fundamental concepts and techniques of computer graphics.
2. To explore algorithms for drawing, clipping, and filling graphical primitives efficiently.
3. To develop understanding of 2D graphical transformations and their mathematical foundations
4. To understand 3D geometric transformations, their matrix representations, and study various projection techniques including perspective, orthographic, axonometric, and oblique projections.
5. To study curve and surface representation techniques, including Bezier and B-spline methods.
6. To understand rendering techniques and color models for realistic image generation.

### Syllabus
### Theory

**Unit 1: Introduction to Computer Graphics (4 Hours)**

Overview of Computer Graphics and its applications, Graphics systems and standards, Video display devices: Raster-scan systems, Random-scan systems, Input devices and graphics software, Coordinate systems, graphics pipeline, Output primitives and their attributes, Applications of computer graphics.

**Unit 2: Polygons and Clipping (5 Hours)**

Scan converting lines, mid-point criteria, Problems of aliasing, end-point ordering, Line clipping: Cyrus-Beck, Cohen-Sutherland, and Liang-Barsky algorithms, Scan converting circles and ellipses, Filling polygons, Clipping polygons and challenges with multiple components.

**Unit 3: 2D Transformation (5 Hours)**
Transformations and matrices, 2D transformations: translation, rotation, reflection, scaling, Homogeneous coordinates and matrix representation, Combined transformations, transformation of points and unit square, Solid body transformations, rotation about arbitrary points, Reflection through arbitrary lines, Window-to-viewport transformations, inverse transformations

**Unit 4: 3D Transformations and Projections (5 Hours)**
Introduction to 3D transformations: scaling, shearing, rotation, reflection, translation, Multiple transformations, rotation about an arbitrary axis in space, Reflection through an arbitrary plane, Matrix representation and composition of 3D transformations, Affine and perspective geometry, perspective transformations, Generating perspective views, vanishing points, camera models, Orthographic, axonometric, and oblique projections

**Unit 5: Plane Curves and Surfaces (5 Hours)**
Curve representation: non-parametric and parametric, Parametric representation of circle, ellipse, parabola, and hyperbola, Bezier curves, B-spline curves: fitting, subdivision, parametric cubic curves, Quadric surfaces, Bezier surfaces, Fractal line.

**Unit 6: Rendering Techniques and Color Models (4 Hours)**
Overview of rendering in computer graphics, Illumination models: ambient, diffuse, and specular reflection, Shading techniques, Visible surface detection algorithms: Z-buffer, Painter's algorithm, Introduction to ray tracing and radiosity, Color models: RGB, CMY, HSV – representation and conversions

## Syllabus
## Laboratory

**List of Experiments**

1. Implementation of basic graphics primitives: line (DDA and Bresenham algorithms) and circle (Midpoint Circle algorithm).
2. Implementation of ellipse using midpoint algorithm.
3. Implementation of line clipping algorithms: Cohen-Sutherland and Liang-Barsky.
4. Implementation of polygon clipping using Sutherland-Hodgman algorithm.
5. 2D geometric transformations: translation, scaling, rotation, reflection.
6. Composite transformations and window-to-viewport mapping
7. Scanline polygon filling algorithm
8. Drawing ellipse using midpoint algorithm

9. 3D transformations: translation, rotation, scaling
10. Implementation of 3D perspective and orthographic projections
11. Bezier curve generation and rendering
12. B-spline curve generation and comparison with Bezier curves
13. Implementation of Z-buffer algorithm for hidden surface removal

## Syllabus
## Course Project

**List of Course Projects**

1. Terrain Generation System.
2. 2D Game Development.
3. 3D Car Simulation.
4. Solar System Animation.
5. Curve Editor for Designers.
6. Architectural Walkthrough.
7. Virtual Art Gallery.
8. Flight Simulator Viewer.
9. Ray Tracing Renderer.
10. Water Ripple Simulation.
11. 3D Chess Game Visualization.
12. Pixel Art Drawing Tool.
13. Interactive Fractal Explorer.
14. 3D Avatar Builder.
15. Digital Clock with Animated Effects.
16. Weather Visualization Dashboard.
17. Cloth Simulation Using Meshes.
18. Heatmap Visualizer for Real-Time Data.
19. Virtual Aquarium
20. Procedural City Generator

## Course Outcomes

**Course Outcomes:** Students will be able to:

1. Understand and apply fundamental concepts of computer graphics.
2. Implement various output primitives and scan conversion algorithms.
3. Apply 2D and 3D geometric transformations in graphical applications.
4. Use clipping and projection techniques in 2D and 3D graphics.
5. Generate curves and surfaces using Bezier and B-spline methods.
6. Demonstrate understanding of rendering techniques and color models

## Books and E-Resources

**For Reference Print Book -**

1. "Computer Graphics", D. Hearn, M. Baker, 2nd Edition, Pearson Education, 2002, ISBN 81-7808-794-4.
2. "Procedural Elements for Computer Graphics", D. Rogers, 2nd Edition, Tata McGraw-Hill Publication, 2001, ISBN 0-07-047371-4.
3. "Computer Graphics", S. Harrington, 2nd Edition, McGraw-Hill Publications, ISBN 0 - 07 -100472 -6.
4. "Computer Graphics Principles and Practice", J. Foley, V. Dam, S. Feiner, J. Hughes, 2nd Edition, Pearson Education, 2003, ISBN 81-7808-038-9

**For MOOCs and other learning Resources**

1. https://onlinecourses.nptel.ac.in/noc20_cs90/preview

## Honors in Computer Vision

## ABCXXX: Image Processing

Teaching Scheme:
Theory: 28 Hours / Week ; Laboratory: 2 Hours / Week;
Total Credits: (3)

**Course Prerequisites:**

1. Digital logic Design
2. Microprocessor
3. Computer Organization

**Course Objectives:**

1. To learn the basics of image processing and different color models.
2. To analyze image condition and learn image  enhancement algorithms in a spatial domain.
3. To learn image enhancement algorithms in the frequency domain.
4. Segment images using modern techniques to extract meaningful regions..
5. To study image morphological operations and their applications.
6. Implement standard image compression techniques and analyze their efficiency

**Course Relevance:**

Vision is one of the most powerful human senses. In an era of intelligent automation, image processing is a vital research and development domain. In Industry 4.0, image processing systems using industrial cameras play a crucial role in automated production — from raw material inspection to final quality checks. In the entertainment industry, demands such as 4K video streaming require effective compression techniques. Additionally, image manipulation is central to social media and mobile applications. This course enables students to explore and innovate in these rapidly evolving fields.

### Syllabus

### Theory

**Unit 1: Introduction to Image Processing (5 Hours)**

Elements of an image processing system, Scenes and images,  Human visual system, color vision , Color models: RGB, HVS, YUV, CMYK, YCbCr ,Basic pixel relationships ,Linear and nonlinear operations ,Image sampling and quantization

**Unit 2: Image Enhancement Techniques (5 Hours)**
Memory-less operations,Spatial domain enhancement, smoothing, sharpening ,Contrast stretching and enhancement,Histogram and histogram equalization , Denoising filters

**Unit 3: Frequency Domain Processing (4Hours)**
Fourier transform for images, Discrete Fourier Transform (DFT),Frequency domain filtering: low-pass and high-pass filters ,Homomorphic filtering, Applications in image enhancement

**Unit 4: Image Segmentation (5 Hours)**
Classification of image segmentation techniques: Edge-based Segmentation, Region based techniques. Binarization: Global Thresholding, Adaptive thresholding. Types of Edge detector: derivative filters, Sobel, Canny. Edge linking. Feature Extraction.

**Unit 5: Morphological Operations & Object Recognition (5 Hours)**
Binary morphology: erosion, dilation, opening, and closing ,Feature detection: lines, circles, corners ,Hough transform and RANSAC ,Harris corner detector , Feature descriptors: SIFT, boundary representation (chain code) ,Descriptor matching techniques

**Unit 6: Image Compression Techniques (4 Hours)**
Need for image compression, Coding redundancy, Classification: lossy and lossless compression, Run Length Encoding (RLE), Huffman Coding, Shannon-Fano Coding, JPEG and JPEG2000 standards

## Syllabus
## Laboratory

**List of Experiments**

1. Introduction to image processing tools (MATLAB / Python – OpenCV)
2. To implement reading and displaying grayscale and color images
3. To implement histogram equalization and contrast enhancement
4. To implement Spatial filtering – smoothing and sharpening
5. To implement Fourier transform and frequency domain filtering
6. To implement adding noise and noise reduction techniques
7. To implement Edge detection using Sobel, Prewitt, Canny
8. To implement Image restoration using Wiener filter
9. To implement Image compression using JPEG
10. To implement basic Morphological operations

11. To implement Image segmentation – thresholding and region growing
12. To implement Color Space Conversion and Analysis

## Syllabus
## Course Project

**List of Course Projects**

1. Real-time edge detection system
2. License plate recognition
3. Face detection and recognition
4. Medical image enhancement (e.g., X-ray/CT scans)
5. OCR system using image segmentation
6. Object tracking in video
7. Image inpainting
8. Document scanning and binarization
9. Texture classification using GLCM
10. Image mosaic generation
11. Image watermarking
12. Smart crop recommender from aerial images

## Course Outcomes

**Course Outcomes:** Students will be able to:

1. Recognize basic concepts and applications of digital image processing.
2. Apply spatial domain techniques for image enhancement.
3. Analyse and implement methods for image restoration and noise reduction.
4. Build image segmentation techniques to identify region of interest.
5. Perform morphological operations and interpret their outcomes.
6. Implement standard image compression techniques and analyse their efficiency.

## Books and E-Resources

**For Reference Print Book –**

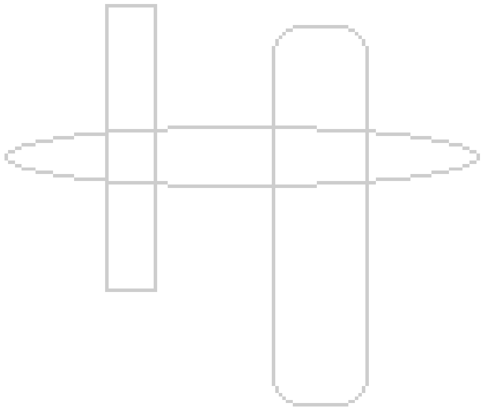
1. R. C. Gonzalez, R. E. Woods; "Digital Image Processing" ; 4th Edition; Pearson; 2018
2. Rafael Gonzalez & Richard Woods, "Digital Image Processing," ;3rd Edition, Pearson publications, ISBN 0132345633.

**For Reference Electronic Book –**

1. W. K. Pratt; "Digital Image Processing:" PIKS Scientific Inside; 4th Edition; Wiley;   2007. Accessed: Jun. 3, 2025. [Online]. Available: https://ebookcentral.proquest.com
2. K.R. Castleman, "Digital Image Processing,";3rd Edition, Prentice Hall: Upper Saddle River, NJ, 3, ISBN 0-13-211467 -4.

**For MOOCs and other learning Resources**

1. Prof. Prabir Kumar Biswas, IIT Kharagpur "Digital Image Processing";https://onlinecourses.nptel.ac.in/noc22_ee116/preview



# Honors in Computer Vision

## ABCXXX: Computer Vision

Teaching Scheme:
Theory: 28 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

**Course Prerequisites:**

2. Image Processing (recommended)
3. Basic Programming in Python
4. Linear Algebra, Probability, and Statistics

**Course Objectives:**

1. To understand the principles and components of modern computer vision systems.
2. To apply geometric and algebraic approaches for camera modeling and image formation.
3. To implement feature detection, description, and matching for visual correspondence.
4. To analyse motion in video sequences using optical flow and tracking.
5. To explore 3D vision, depth estimation, and scene reconstruction.
6. To develop and evaluate vision-based applications using deep learning.

**Course Relevance:**
Computer vision empowers machines to "see" and make decisions based on visual data. With applications in autonomous vehicles, robotics, AR/VR, surveillance, and medical imaging, it's a core pillar of AI. This course provides essential skills to design intelligent systems capable of perception and understanding through cameras.

### Syllabus
### Theory

**Unit 1: Introduction to Computer Vision** **(5 Hours)**
Overview of computer vision tasks, Applications and challenges,  Vision vs Image Processing , Image formation: perspective projection, pinhole camera model , Intrinsic and extrinsic parameters ,Camera calibration

**Unit 2: Feature Detection and Matching (5 Hours)**
Edges, corners, and blobs ,Harris corner detector, FAST ,Scale-Invariant Feature Transform (SIFT) , Oriented FAST and Rotated BRIEF (ORB) , Feature matching: brute force, FLANN, RANSAC for geometric consistency

**Unit 3: Image Alignment and Homography (4Hours)**
Planar homography , Image stitching and mosaicing , Geometric transformations: affine and projective ,Panorama generation

**Unit 4: Motion and Tracking (5 Hours)**
Optical flow: Horn-Schunck, Lucas-Kanade ,Background subtraction ,Kalman Filter basics, Object tracking algorithms: KLT, Median Flow, CSRT, Multi-object tracking basics

**Unit 5: 3D Vision and Depth Estimation (4 Hours)**
Stereo vision and disparity map ,Structure from motion (SfM) ,Epipolar geometry and essential/fundamental matrices ,Triangulation ,Depth sensors (RGB-D, LiDAR)

**Unit 6: Deep Learning for Vision (5 Hours)**
Introduction to CNNs for vision ,Object detection: YOLO, SSD ,Image segmentation: semantic vs instance ,Pretrained networks (ResNet, MobileNet, etc.) ,Transfer learning and fine-tuning with datasets (COCO, PASCAL VOC)

**Syllabus**
**Laboratory**

**List of Experiments**

1. To implement Camera calibration and distortion removal
2. To implement Feature detection and matching with OpenCV
3. To implement Image stitching using homography
4. To implement Motion estimation using optical flow
5. To implement Tracking moving objects in video
6. To implement 3D reconstruction from stereo images
7. To implement  Object detection using pretrained YOLOv5 or MobileNet
8. To implement   Semantic segmentation with DeepLab or U-Net
9. To implement   Real-time face detection and tracking

10. To implement Custom classification with transfer learning (e.g., with PyTorch or TensorFlow)
11. To implement Facial Landmark Detection and Head Pose Estimation
12. To implement Human Activity Recognition using Video Sequences

## Syllabus
## Course Project

**List of Course Projects**

1. Smart Surveillance System
2. Vision-based Attendance System (Face Recognition)
3. Augmented Reality Overlay System
4. Driver Drowsiness Detection
5. Visual Navigation for Robot
6. Real-time Object Counting in Video
7. Gesture Recognition System
8. AI-based Virtual Fitting Room
9. Lane Detection in Autonomous Vehicles
10. 3D Scene Reconstruction from Stereo Images
11. Emotion Recognition from Facial Expressions
12. Real-Time Sign Language Recognition

## Course Outcomes

**Course Outcomes:** Students will be able to:

1. Understand fundamental concepts in camera modeling and image formation.
2. Identify and match visual features across images.
3. Perform image registration and alignment for stitching and tracking.
4. Estimate motion and apply tracking algorithms in video.
5. Analyze 3D vision problems like stereo and SfM.
6. Implement deep learning models for vision applications.

## Books and E-Resources

**For Reference Print Book –**

1. R. Szeliski; "*Computer Vision: Algorithms and Applications*"; Springer; 2010
2. R. Hartley, A. Zisserman; "*Multiple View Geometry in Computer Vision*"; Cambridge University Press; 2nd Edition; 2004

**For Reference Electronic Book –**

1 Simon J.D. Prince; "*Computer Vision: Models, Learning, and Inference*"; Cambridge University Press, 2012. Accessed: Jun. 3, 2025. [Online]. Available: https://ebookcentral.proquest.com

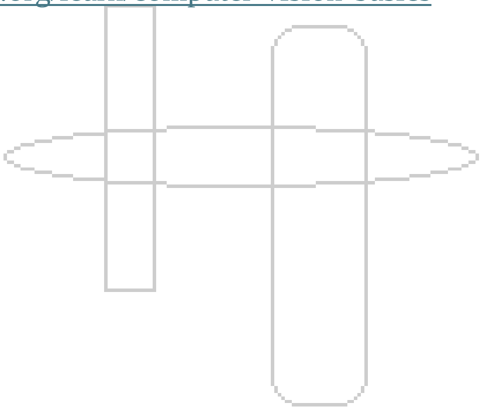
.

**For MOOCs and other learning Resources**

1. J. Malik, S. Savarese; *Computer Vision*; edX (UC Berkeley, Stanford); https://www.edx.org/course/computer-vision

A. Karpathy; *Convolutional Neural Networks for Visual Recognition*; Stanford CS231n; http://cs231n.stanford.edu

2. S. Lazebnik; *Computer Vision Basics*; Coursera; https://www.coursera.org/learn/computer-vision-basics



## Honors in Computer Vision

## ABCXXX: Introduction to Robot Learning

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits:  3

**Course Objectives:**

1. To introduce the fundamental principles of robotics.
2. To develop an understanding of core components in robot learning systems, such as perception, motion control, and various learning paradigms.
3. To provide hands-on knowledge of the Robot Operating System (ROS).
4. To impart knowledge on localization, mapping, and path planning techniques, including SLAM and various search algorithms.
5. To explore advanced reinforcement learning and deep learning methods applied to robotics.
6. To analyze real-world applications and emerging trends in robot learning

### Syllabus
### Theory

**Unit 1: Fundamentals of Robotics** **(4 Hours)**
Introduction to Robotics and Mechanisms, Types of robots, Degrees of freedom, kinematics and dynamics basics, Human-Robot Systems-Human-Robot Interaction (HRI), Cognitive models, task sharing, safety in HRI, HRI Interfaces

**Unit 2: Components of Robot Learning Systems** **(5 Hours)**
Perception: Sensors and sensor fusion (LIDAR, camera, IMU), Object recognition and environmental understanding
Action: Motion planning and control, Inverse kinematics,
Learning & Adaptation: Supervised, unsupervised, and reinforcement learning, Online and continual learning in robotics

**Unit 3: Robot Operating System** **(5 Hours)**
Introduction to ROS, ROS architecture, nodes, topic and services, ROS Tools and Simulation
Writing custom ROS packages, Integration with Python and C++

**Unit 4: Localization, Mapping, and Navigation (5 Hours)**
Simultaneous Localization and Mapping (SLAM): EKF-SLAM, Graph-SLAM
Path Planning Algorithms: A*, D*, RRT, Dynamic Window Approach
Robot Navigation Stack in ROS: Sensor integration, cost maps, planners

**Unit 5: Advanced Reinforcement Learning and Deep Learning (5 Hours)**
Policy gradient methods and actor-critic algorithms, Deep Q-Networks (DQN), Use of neural networks for policy/value function approximation, Continuous action spaces and policy optimization, Transfer learning and meta-learning in robotics.

**Unit 6: Applications and Current Trends in Robot Learning (4 Hours)**
Autonomous navigation and manipulation, Robot locomotion learning, Human-robot interaction learning, Research trends: Sim2Real transfer, lifelong learning, meta-learning, Case studies and future directions.

## Syllabus
## Laboratory

**List of Experiments**

1. Implement supervised learning algorithms for robot perception tasks.
2. Sensor data collection and pre-processing simulation.
3. Implementation of Q-learning on grid-world robot navigation.
4. Policy gradient algorithm implementation for simple control tasks.
5. Deep Q-Network (DQN) simulation for a robot control environment.
6. Imitation learning from recorded demonstrations.
7. Simulation of robot path planning using learned models.
8. Safe learning experiment with constraints in simulation.
9. Multi-agent learning simple cooperative task implementation.
10. Final project: Integrating perception and learning for a robotics task (e.g., navigation or pick-and-place).

## Syllabus
## Course Project

**List of Course Projects**

1. Autonomous robot navigation in a maze environment.
2. Learning-based robotic arm manipulation.
3. Imitation learning for robot handwriting.
4. Reinforcement learning for drone flight control.

5. Deep learning-based obstacle avoidance for mobile robots.
6. Multi-agent cooperative exploration.
7. Robot learning for human-robot collaboration tasks.
8. Sim2Real transfer learning for robot control.
9. Learning to balance a robot on uneven terrain.
10. Adaptive robot grasping using reinforcement learning.

## Course Outcomes

**Course Outcomes:** Students will be able to:

1. Understand and apply fundamental robot learning concepts.
2. Describe the key components of robot learning systems such as perception, motion planning, control, and various learning techniques.
3. Develop basic robotic applications using Robot Operating System (ROS).
4. Apply localization, mapping, and navigation algorithms.
5. Design reinforcement learning models using methods like policy gradients, DQN, and actor-critic for robotic control tasks.
6. Develop practical robot learning projects using simulation tools.

## Books and E-Resources

**For Reference Print Book -**

1. *Reinforcement Learning: An Introduction* by Richard S. Sutton and Andrew G. Barto, 2nd Edition, MIT Press, 2018.
2. *Probabilistic Robotics* by Sebastian Thrun, Wolfram Burgard, and Dieter Fox, MIT Press, 2005.
3. *Deep Reinforcement Learning Hands-On* by Maxim Lapan, Packt Publishing, 2018.
4. *Robot Learning* edited by Sergey Levine, Aude Billard, Davide Scaramuzza, Springer, 2023.

**For MOOCs and other learning Resources**

1. https://onlinecourses.nptel.ac.in/noc20_de11/preview

## Honors in Computer Vision

## ABCXXX: Visual Learning and Recognition

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week; Tutorial: 1 Hours / Week
Total Credits:  4

**Course Prerequisites:**

1. Machine Learning
2. Linear Algebra
3. Python Programming

**Course Objectives:**

1. To learn the basics of computer vision, including image formation, color spaces, and key applications.
2. To learn classical image representation using features like SIFT, HOG, BoVW, and apply machine learning for visual recognition.
3. To learn to build and evaluate CNN-based deep learning models for image classification using architectures like VGG and ResNet.
4. To learn deep learning methods for object detection and segmentation using R-CNN, YOLO, and U-Net, with performance evaluation.
5. To learn advanced visual learning methods like self-supervised learning, vision transformers, GANs, and few-/zero-shot learning.
6. To learn to apply advanced interpretability techniques like Grad-CAM, SHAP, and LIME, and tackle robustness, ethical issues, and deployment challenges in visual recognition systems.

**Course Relevance:**
This course provides a comprehensive foundation in visual learning and recognition, covering core computer vision principles, feature extraction, and state-of-the-art deep learning architectures. It enables students to develop, implement, and evaluate advanced models for image classification, object detection, segmentation, and interpretability, addressing critical challenges such as robustness and ethical deployment. This technical expertise is vital for applications in autonomous systems, medical diagnostics, surveillance, and AI-driven image analysis.

**Section 1: Topics/Contents**

**Unit 1: Introduction to Visual Recognition (4 Hours)**
Basics of computer vision and image understanding, Challenges in visual learning, Applications: object recognition, face detection, OCR, Overview of datasets (ImageNet, COCO, MNIST), Image formation, color spaces, pre-processing basics.

**Unit 2: Classical Methods for Image Representation (4 Hours)**
Handcrafted features: SIFT, HOG, SURF,  Bag of Visual Words (BoVW), Spatial pyramids, Machine learning models: k-NN, SVMs, decision trees, Dimensionality reduction: PCA, t-SNE

**Unit 3: Deep Learning for Image Classification (5 Hours)**
Convolutional Neural Networks (CNNs), Key architectures: LeNet, AlexNet, VGG, ResNet, Training techniques: data augmentation, regularization, Transfer learning and fine-tuning, Evaluation metrics: accuracy, precision, recall

**Section2: Topics/Contents**

**Unit 4: Object Detection and Image Segmentation (5 Hours)**
Object detection: R-CNN, Fast R-CNN, YOLO, SSD, Image segmentation: semantic vs instance, U-Net, Mask R-CNN, Evaluation: IoU, mAP

**Unit 5: Advanced Visual Learning Techniques (5 Hours)**
Self-supervised learning (SimCLR, MoCo), Vision Transformers (ViT) and attention mechanisms, Autoencoders, GANs (DCGAN, CycleGAN), Few-shot and zero-shot learning

**Unit 6: Interpretability, Robustness, and Real-World Applications (5 Hours)**
Interpretability in vision using methods like Grad-CAM, SHAP, LIME, LRP, and Integrated Gradients; robustness through adversarial defenses (FGSM, PGD), training, and noise handling; with applications in medical imaging, autonomous driving, agriculture, and AR/VR.

**Syllabus**
**Laboratory**

**List of Lab Assignments:**

1. Implement a Python program to load, display, and preprocess images. Perform color space conversions (RGB, HSV, Grayscale), resizing, cropping, and apply filters like Gaussian blur and Sobel edge detection.

2. Explore popular vision datasets (e.g., MNIST, CIFAR-10, COCO). Implement and visualize basic image augmentation techniques such as flipping, rotation, scaling, and brightness adjustment.
3. Extract image features using SIFT and HOG descriptors. Visualize keypoints and descriptors, and compare their effectiveness on various images.
4. Build an image classification pipeline using Bag of Visual Words. Train SVM or k-NN classifiers and evaluate performance using confusion matrix and accuracy metrics.
5. Design and implement a Convolutional Neural Network (CNN) using TensorFlow/Keras or PyTorch. Train the model on MNIST or CIFAR-10 dataset and plot training/validation accuracy and loss.
6. Fine-tune a pre-trained deep learning model (e.g., VGG16, ResNet50) on a custom image dataset. Evaluate and compare results with a baseline CNN.
7. Train a U-Net model for semantic segmentation using a dataset like Cityscapes or a medical imaging dataset. Visualize input images, predicted masks, and ground truth.
8. Apply a pre-trained Mask R-CNN model to perform instance segmentation. Visualize the segmented objects with corresponding masks, bounding boxes, and class labels.
9. Implement or use a self-supervised learning framework like SimCLR. Train the model on unlabeled data and evaluate using a linear classifier on top of learned features.
10. Use a Vision Transformer (ViT) model for image classification using PyTorch or Hugging Face Transformers. Train on a small dataset and compare with CNN performance.
11. Generate Grad-CAM heatmaps for a CNN model to explain classification decisions. Analyze and interpret the visual explanations for different classes.
12. Implement FGSM or PGD adversarial attacks on an image classifier. Apply simple defenses like adversarial training or input denoising and evaluate model robustness.
13. Use CycleGAN or StyleGAN to perform unpaired image-to-image translation (e.g., horses to zebras, summer to winter). Tasks:  Train a GAN model on the selected domain,

    Evaluate visual quality and FID score,  Experiment with latent space interpolation.

**List of Course Projects:**

1. Build a Python-based GUI application that allows users to upload images and apply preprocessing operations such as resizing, filtering, color adjustments, and edge detection in real-time using OpenCV.
2. Create a dashboard that analyzes and visualizes dataset statistics (class distribution, image resolutions, color histograms) for a given image dataset like CIFAR-10 or Fashion-MNIST.
3. Develop a system to detect and recognize brand logos in real-world images or videos using keypoint-based feature descriptors and homography matching.
4. Build a classification model for popular landmarks using SIFT/HOG features and a Bag-of-Words pipeline with SVM. Dataset can include tourist landmark images (e.g., Eiffel Tower, Taj Mahal).
5. Create a deep learning model to classify crop or plant diseases from leaf images. Use data augmentation and train a custom CNN using TensorFlow or PyTorch.
6. Design and train a CNN to recognize human facial expressions (e.g., happy, sad, angry) using datasets like FER2013. Deploy the model as a web or mobile app.
7. Build a real-time object detection system (YOLOv5/SSD) for detecting safety equipment (helmets, masks, vests) in workplace environments using surveillance video.
8. Develop a model to detect and classify traffic signs using an object detection framework. Utilize datasets like GTSDB or Indian traffic sign datasets.
9. Implement a semantic segmentation model (e.g., U-Net or DeepLabV3+) to segment road lanes and obstacles from dashcam videos for autonomous driving support.
10. rain a U-Net model for segmenting organs or lesions in medical images (e.g., lungs in X-rays or tumors in MRI scans). Use publicly available medical imaging datasets.
11. Implement SimCLR or BYOL to learn visual representations from an unlabeled dataset. Evaluate the features using a downstream classification task.
12. Develop a Visual Question Answering (VQA) system using multimodal transformers (e.g., ViLT or LXMERT) that can answer questions about images in natural language.

## Syllabus
## Tutorials

**List of Tutorials:**

**1.** Self-Supervised and Unsupervised Visual Representation Learning
**2.** Transformer Architectures for Vision Tasks**.**
**3.** Graph Neural Networks (GNNs) for Visual Data
**4.** Explainable AI (XAI) in Visual Recognition
**5.** Few-Shot and Zero-Shot Learning for Image Recognition**.**
6. Adversarial Attacks and Defenses in Vision Models
7. Generative Adversarial Networks (GANs) for Image Synthesis and Augmentation
**8.** Multimodal Learning and Visual Question Answering (VQA)**.**
9. Domain Adaptation and Generalization in Visual Recognition
10. Neural Architecture Search (NAS) for Vision Models
**11.** Video Understanding and Action Recognition**.**
**12.** Edge and Embedded Visual Recognition Systems

## Course Outcomes

**Course Outcomes:** Students will be able to

**1.** Select foundational concepts of visual learning and recognize their practical relevance in real-world applications.
**2.** Implement and apply classical and deep learning-based methods for effective visual data representation and recognition.
**3.** Analyze and compare the effectiveness of different image classification and object detection algorithms on real-world datasets.
**4.** Design and develop convolution neural networks and state-of-the-art architectures for image recognition and segmentation tasks.
**5.** Evaluate advanced visual recognition techniques, including transformer models and self-supervised learning, in various application scenarios.
**6.** Collaborate effectively to develop industry-relevant visual recognition projects, demonstrating professional communication and ethical considerations.

## Books and E-Resources

**For Reference Print Book -**

1. R. Szeliski; 'Computer Vision: Algorithms and Applications'; 1st Edition; Springer; 2010
2. M. S. Nixon, A. S. Aguado; 'Feature Extraction and Image Processing for Computer Vision'; 3rd Edition; Academic Press; 2019
3. Goodfellow, Y. Bengio, A. Courville; 'Deep Learning'; 1st Edition; MIT Press; 2016

**For Reference Electronic Book –**

1. S. D. Joshi, M. M. Trivedi; *Image Processing and Pattern Recognition: Fundamentals and Techniques*; 1st Edition; PHI Learning Pvt. Ltd.; 2017
2. A.K. Sharma; *Digital Image Processing*; 3rd Edition; Tata McGraw-Hill Education; 2018
3. R. C. Gonzalez, R. E. Woods; *Digital Image Processing*; 4th Edition; Pearson India; 2018

**For MOOCs and other learning Resources**

1. Deep Learning for Visual Computing - Course- https://onlinecourses.nptel.ac.in/noc22_ee54/preview?utm_source=chatgpt.com
2. Deep Learning for Computer Vision I Stanford Online- https://online.stanford.edu/courses/cs231n-deep-learning-computer-vision?utm_source=chatgpt.com

# Honors in Computer Vision

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

25. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
26. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

27. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
28. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
29. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
30. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
31. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
32. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Cloud Computing

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Cloud Infrastructure and Architecture | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Big Data Analytics in cloud | 2 | 2 | 0 | | | | | 3 |
| C3 | V | DevOps for Cloud | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Cloud Security | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Cloud-Based Application Development and Serverless Computing | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |

## Honors in Cloud Computing

## ABCXXX: Cloud Infrastructure and Architecture

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

**Course Prerequisites:**
Basic knowledge of computer networks, operating systems, and virtualization concepts.
**Course Objectives:**

1. To introduce the core concepts of cloud computing and infrastructure.
2. To explore the architecture and deployment models of cloud systems.
3. To gain insights into virtualization, storage, and networking in the cloud.
4. To examine modern cloud services, automation, and orchestration tools.
5. To understand cloud security, compliance, and performance optimization.
6. To analyze and evaluate cloud cost management and emerging trends.

**Course Relevance:**
The **Cloud Infrastructure and Architecture (Honors)** course is highly relevant in today's cloud-driven IT landscape, where businesses rely on scalable, secure, and efficient cloud solutions. It equips students with both theoretical knowledge and practical skills in areas like virtualization, networking, DevOps, and cloud security using platforms such as AWS, Azure, and GCP. This course prepares students for industry roles like Cloud Engineer and Solutions Architect, supports certification readiness, and lays a strong foundation for innovation in emerging fields like cloud-native development and edge computing, thereby enhancing employability and future readiness.

**Section 1: Topics/Contents**
**Unit 1: Introduction to Visual Recognition (4 Hours)**
Cloud Computing Overview: Characteristics, Benefits, Challenges,Cloud Service Models (IaaS, PaaS, SaaS), Cloud Deployment Models (Public, Private, Hybrid, Multi-cloud) Overview of Cloud Providers (AWS, Azure, GCP)

**Unit 2: Cloud Architecture and Design (5 Hours)**
Core Components: Compute, Storage, Network, Multi-tier Cloud Architecture, Micro services and Containerization Basics, Design for High Availability, Scalability, and Disaster Recovery, Cloud-native vs Traditional Architecture

**Unit 3: Virtualization and Storage (5 Hours)**
Virtual Machines vs Containers, Hypervisors: Type 1 and Type 2, Container Orchestration: Docker, Kubernetes, Cloud Storage: Block, Object, and File Storage, Storage Access and Management in Cloud Environments

**Section 2: Topics/Contents**
**Unit 4: Networking and Load Balancing in Cloud (4 Hours)**
Virtual Networks, Subnets, and IP Management, Cloud Load Balancers and Traffic Distribution, CDN and Edge Networks, VPN, Direct Connect, and Peering Options
DNS Management and High Availability Networking

**Unit 5: Automation, DevOps, and Monitoring (5 Hours)**
Infrastructure as Code: Terraform, AWS CloudFormation, CI/CD Pipelines with Jenkins, GitHub Actions, Configuration Management: Ansible, Chef, Puppet, Monitoring Tools: CloudWatch, Prometheus, Grafana, Log Management and Alerting

**Unit 6: Cloud Security, Cost Optimization, and Trends (5 Hours)**
Security Principles: IAM, Encryption, Key Management, Compliance and Regulatory, Standards (GDPR, HIPAA), Cost Management Tools: Cost Explorer, Budgets, Performance Optimization and Auto-scaling, Emerging Trends: Serverless, Edge Computing, Green Cloud

**Syllabus**
**Laboratory**

**List of Lab Assignments:**

1. Design and deploy a multi-cloud architecture using AWS and Azure free-tier accounts to host a simple web application, demonstrating deployment models (public, hybrid). Document the pros and cons of each platform's service model.
2. Perform a detailed cost-benefit analysis by provisioning similar resources (compute, storage) across IaaS, PaaS, and SaaS models in Azure and compare their performance, scalability, and cost-effectiveness.
3. Architect and implement a highly available, scalable multi-tier web application on Azure using Virtual Machines, Azure App Services, and Azure SQL Database with automated failover configurations.
4. Design and simulate a disaster recovery strategy for a cloud infrastructure using Azure Site Recovery and Backup services, ensuring minimal downtime and data loss.

5. Deploy and manage containerized applications using Azure Kubernetes Service (AKS), including setting up namespaces, deployments, and scaling based on traffic load.
6. Implement an Infrastructure as Code (IaC) project using Azure Resource Manager templates or Terraform to provision a complete cloud environment with networking, compute, and storage components.
7. Configure a secure, isolated virtual network with multiple subnets, route tables, and Network Security Groups (NSGs) in Azure, and test connectivity and security rules among resources.
8. Set up and test Azure Load Balancer and Application Gateway for a web app, including configuring session persistence, SSL termination, and Web Application Firewall (WAF) policies.
9. Build and automate a CI/CD pipeline using Azure DevOps or GitHub Actions to deploy a micro services application to AKS, incorporating unit tests and deployment approval workflows.
10. Implement monitoring and alerting on Azure using Azure Monitor and Log Analytics, creating dashboards and setting up alerts based on custom performance and security metrics.
11. Configure and enforce role-based access control (RBAC) and Conditional Access policies in Azure Active Directory to secure cloud resources, and demonstrate auditing and reporting for compliance.
12. Develop a cost management strategy by setting budgets, creating cost alerts, and optimizing resource utilization in Azure, and then analyze how server less computing (Azure Functions) can reduce costs for event-driven workloads.

## Syllabus
## Course Projects

**List of Course Projects:**

1. Develop a multi-cloud deployment strategy and prototype for a global e-commerce platform, utilizing AWS and Azure, analyzing the benefits and challenges of each cloud service and deployment model to optimize performance and cost.

2. Create a detailed comparative study and implementation of IaaS, PaaS, and SaaS solutions for a real-world business scenario (e.g., online education platform), including hands-on deployment and performance benchmarking on Azure.
3. Design and implement a fault-tolerant, auto-scaling web application architecture on Azure that uses microservices deployed with Azure Kubernetes Service, integrated with Azure Cosmos DB for globally distributed data management.
4. Architect and simulate a cloud disaster recovery and business continuity plan for a healthcare application using Azure Site Recovery, Backup, and multi-region deployment strategies, ensuring compliance with data privacy regulations.
5. Build and deploy a containerized multi-tier application on Azure AKS, including persistent storage management with Azure Blob and Azure Disk Storage, and implement rolling updates with zero downtime.
6. Develop an Infrastructure as Code (IaC) automation framework using Terraform and Azure DevOps to provision, configure, and maintain a secure cloud environment for a financial services application.
7. Design and implement a secure Azure virtual network with hybrid connectivity that connects on-premises infrastructure via VPN or Express Route, incorporating subnet segmentation, network security policies, and DNS management.
8. Build a scalable load balancing solution for a media streaming service using Azure Application Gateway and Traffic Manager, ensuring high availability, SSL offloading, and global traffic distribution
9. Develop a fully automated CI/CD pipeline for a cloud-native application using Azure DevOps**,** incorporating automated testing, infrastructure provisioning, container orchestration, and deployment monitoring.
10. Implement a comprehensive monitoring and alerting system using Azure Monitor, Log Analytics, and Application Insights to provide real-time operational intelligence and security auditing for a deployed cloud service.
11. Design and enforce a cloud security governance framework for an enterprise using Azure Active Directory, RBAC, Azure Policy, and Azure Security Center**,** including compliance reporting and incident response simulations.

12. Create a cloud cost management and optimization project that models workloads using Azure Cost Management tools**,** integrates serverless computing (Azure Functions) for cost savings, and provides strategic recommendations for sustainable cloud adoption.

## Course Outcomes

**Course Outcomes**: Students will be able to

1. Understand and explain the fundamental principles and key components of cloud infrastructure, demonstrating comprehension of their roles and interconnections in modern computing environments.
2. Design methodologies to develop cloud architectures that ensure scalability and resilience, incorporating best practices to address real-world application requirements.
3. Develop virtualization technologies and storage management solutions in cloud environments, showcasing practical skills in configuring and optimizing resource utilization.
4. Evaluate security frameworks and compliance standards pertinent to cloud systems, assessing their impact on data protection and regulatory adherence.
5. Utilize and integrate cloud automation and DevOps tools to streamline deployment, management, and monitoring processes, enhancing operational efficiency.
6. Critically assess cost and performance trade-offs in cloud deployments, making informed decisions to optimize resource allocation and maximize value.

## Books and E-Resources

**For Reference Print Book –**

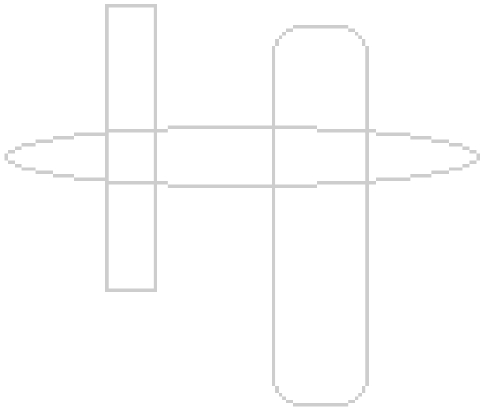
1. T. E. Anderson, M. Dahlin; *'Cloud Computing: Principles, Systems and Applications'*; 1st Edition; Springer; 2014
2. R. Erl, K. Puttini, Z. Mahmood; *'Cloud Computing: Concepts, Technology & Architecture'*; 1st Edition; Prentice Hall; 2013
3. M. J. Kavis; *'Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)'*; 1st Edition; Wiley; 2014

**For Reference Electronic Book –**

1. A. Velte, T. Velte, R. Elsenpeter; *'Cloud Computing: A Practical Approach'*; 1st Edition; McGraw-Hill Education; 2010
2. J. J. Paz, R. Zamora; *'Mastering Cloud Computing: Foundations and Applications Programming'*; 1st Edition; McGraw-Hill Education; 2014
3. C. J. Date; *'Cloud Computing: Principles, Systems and Applications'*; 1st Edition; Morgan Kaufmann; 2016

**For MOOCs and other learning Resources**

1. A. N. Coursera, J. G. University of Illinois; *‘Cloud Computing Specialization’*; Online Course; Coursera; 2020
2. M. Microsoft, A. Azure Team; *‘Microsoft Azure Fundamentals (AZ-900)’*; Online Course; Microsoft Learn; 2023
3. G. Cloud, K. Coursera Team; *‘Architecting with Google Cloud Platform Specialization’*; Online Course; Coursera; 2021



## Honors in Cloud Computing

## ABCXXX: Big Data Analytics in Cloud

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits:  3

**Course Prerequisites:**

1. Basic knowledge of programming (preferably Python, Java, or Scala)
2. Understanding of database concepts and SQL
3. Fundamentals of data structures and algorithms
4. Introductory knowledge of cloud computing concepts
5. Basic familiarity with statistics and machine learning principles

**Course Objectives:**

1. Understand the core concepts of big data and cloud computing and their integration for scalable analytics.
2. Apply modern cloud storage and data management solutions including data lakes and streaming pipelines.
3. Utilize advanced big data processing frameworks and serverless platforms for real-time analytics.
4. Implement and deploy AI/ML models on cloud-based big data with a focus on explainability and automation.
5. Assess and apply cloud security, privacy, and ethical standards in big data analytics environments.
6. Investigate emerging trends such as edge computing, multi-cloud strategies, and future technologies in big data analytics.

**Course Relevance:**

This course is essential for understanding how big data analytics and cloud computing combine to handle large-scale data efficiently and securely. It equips students with skills in modern technologies like AI/ML, real-time processing, and serverless computing; preparing them for advanced roles in data science and cloud-based analytics across various industries.

## Section 1: Topics/Contents

**Unit 1: Foundations of Big Data and Cloud Analytics (4 hours)**

Review of Big Data Concepts: 5Vs (Volume, velocity, Variety, Veracity and Value) and Challenges, Cloud Computing Evolution & Models (IaaS, PaaS, SaaS), Introduction to Cloud-Native Big Data Architectures, Latest Cloud Platforms for Big Data (AWS, Azure, Google Cloud updates), Role of Containerization (Docker, Kubernetes) in Cloud Analytics

**Unit 2: Modern Big Data Storage & Management in Cloud (5 hours)**

Advances in Cloud Storage: Object Stores, Data Lakes (Lakehouse Architecture), Next-Gen NoSQL and NewSQL Databases (e.g., Google Bigtable, AWS DynamoDB, CockroachDB), Data Integration using Streaming Pipelines (Apache Kafka, AWS Kinesis), Cloud-native Data Catalogs & Metadata Management, Deploying a Data Lake with Cloud Services.

**Unit 3: Advanced Big Data Processing Frameworks & Real-Time Analytics (5 hours)**

Unified Analytics Engines: Apache Spark 3.x, Apache Flink Enhancements, Serverless Big Data Processing (AWS Lambda, Azure Functions with Spark), Real-time Analytics & Complex Event Processing, AI-driven Data Pipelines and AutoML in Cloud, Real-time data streaming & analytics with Apache Kafka , Spark on Cloud.

**Unit 4: AI/ML and Deep Learning Integration in Big Data Analytics (5 hours)**

ML/DL Techniques for Big Data on Cloud, Tools & Frameworks: TensorFlow, PyTorch, Spark MLlib, SageMaker, Explainable AI and Model Interpretability for Big Data, Automated Machine Learning (AutoML) and MLOps in Cloud, Case Study: Building scalable AI pipelines for big data,

**Unit 5: Security, Privacy, and Ethical Considerations (4 hours)**

Zero Trust Security Models in Cloud Big Data Analytics, Privacy-Preserving Techniques: Federated Learning, Differential Privacy, Homomorphic Encryption, and Regulatory Compliance: GDPR, CCPA, HIPAA updates, Data Governance Frameworks & Ethical AI, Best practices for secure data sharing and compliance audits.

**Unit 6: Emerging Trends & Future Directions in Big Data Cloud Analytics (5 hours)**

Edge Computing and IoT-enabled Big Data Analytics, Cloud-Native Serverless Architectures and their impact on Big Data, Quantum Computing prospects for Big Data Analytics, Multi-Cloud and Hybrid Cloud Strategies for Big Data, Industry 4.0 and AI-driven Automation in Analytics.

## Syllabus
## Laboratory

**List of Lab Assignments:**

1. Set up and configure a cloud storage bucket (AWS S3/Azure Blob) and upload/download datasets.
2. Create and query a NoSQL database (e.g., MongoDB or DynamoDB) on the cloud for big data storage.
3. Develop and execute a simple Hadoop MapReduce job on a cloud-managed cluster.
4. Run a Spark batch processing job on cloud platform (AWS EMR, Azure HDInsight, or Google Dataproc).
5. Implement a real-time data stream processing application using Apache Kafka and Spark Streaming.
6. Train and evaluate a machine learning model using Spark MLlib on cloud-hosted big data.
7. Deploy a TensorFlow or PyTorch deep learning model on a cloud service like AWS SageMaker or Azure ML.
8. Implement role-based access control (RBAC) for cloud big data resources.
9. Apply encryption to data at rest and in transit within a cloud big data pipeline.
10. Set up an edge computing environment for data collection and analysis with cloud integration.
11. Develop a serverless data processing pipeline using AWS Lambda or Azure Functions with cloud storage triggers.
12. Complete a mini project combining cloud big data storage, processing, and ML model

## Syllabus
## Course Projects

**List of Course Projects:**

1. Design and deploy scalable cloud data lake architecture for multi-source heterogeneous data ingestion, storage, and cataloging.
2. Develop a real-time data ingestion pipeline using Apache Kafka and NoSQL databases to process streaming data from IoT devices on a cloud platform.

3. Build and optimize a distributed batch-processing pipeline using Hadoop and Spark on a cloud-managed cluster for large-scale log analytics.
4. Implement a hybrid real-time analytics system combining Apache Flink and Spark Streaming for fraud detection in financial transactions.
5. Create an end-to-end machine learning pipeline on cloud platforms to predict customer churn, including data preprocessing, model training, tuning, and deployment.
6. Develop a deep learning-based image recognition system using Tensor Flow/PyTorch on cloud GPUs, integrating with big data storage for large datasets.
7. Design and implement a secure multi-tenant big data analytics environment on cloud with role-based access, encryption, and audit logging.
8. Build a privacy-preserving analytics system using federated learning or differential privacy techniques on distributed cloud datasets.
9. Develop an edge-cloud integrated analytics solution for real-time environmental monitoring using edge devices and cloud processing.
10. Build a serverless big data workflow using AWS Lambda (or Azure Functions), Step Functions, and cloud storage for automated data processing and reporting.
11. Investigate and prototype a multi-cloud big data analytics platform, orchestrating data and processing across AWS, Azure, and Google Cloud.
12. Explore quantum-inspired algorithms for big data clustering or optimization, simulating on classical cloud resources and evaluating scalability.

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Explain the integration of big data and cloud computing for building scalable analytics solutions.
2. Apply cloud-based storage technologies and data management techniques such as data lakes and NoSQL databases.
3. Utilize big data processing frameworks and serverless computing models to perform real-time analytics on cloud platforms.

4. Build and deploy machine learning and deep learning models for big data using cloud services with a focus on automation and explainability.
5. Assess cloud security models and apply privacy-preserving techniques to ensure compliance with data protection standards.
6. Investigate and propose solutions involving emerging trends like edge computing, multi-cloud systems, and quantum-inspired analytics.

## Books and E-Resources

**For Reference Print Book –**

1. T. E. Anderson, M. Dahlin; *Cloud Computing: Principles, Systems and Applications*; 1st Edition; Springer; 2014
2. Martin Kleppmann; *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*; 1st Edition; O'Reilly Media; 2017
3. Vignesh Prajapati; *Big Data Analytics with R and Hadoop*; 1st Edition; Packt Publishing; 2013

**For Reference Electronic Book –**

1. Tom White; *Hadoop: The Definitive Guide*; 4th Edition; O'Reilly Media; 2015
2. Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee; *Learning Spark: Lightning-Fast Big Data Analysis*; 2nd Edition; O'Reilly Media; 2020
3. Mark Garcia; *Getting Started with Google Cloud Platform: Big Data and Machine Learning*; 1st Edition; Packt Publishing; 2018

**For MOOCs and other learning Resources**

1. **Big Data Computing – NPTEL (IIT Patna)**
   This course offers an in-depth understanding of big data concepts, including distributed computing, machine learning, and graph processing, with a focus on practical applications. https://onlinecourses.nptel.ac.in/noc23_cs112/preview
2. **Big Data Analytics – edX (University of Adelaide)**
   Part of the Big Data MicroMasters program, this course covers cloud-based big data analysis, predictive analytics, and the application of large-scale data analysis techniques. https://www.mooc-list.com/course/big-data-analytics-edx
3. **Big Data Specialization – Coursera (University of California San Diego)**
   This specialization provides a comprehensive introduction to big data, including data modeling, management, and analysis using tools like Hadoop and Spark. https://www.coursera.org/specializations/big-data

## Honors in Cloud Computing

## ABCXXX: DevOps for Cloud

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits:  3

**Prerequisites:** Basic understanding of Cloud Computing, Operating Systems, and Software Development

**Course Objectives**

1. Understand the Fundamentals of DevOps and Cloud Computing.
2. Enable Continuous Integration and Continuous Deployment (CI/CD).
3. Implement Infrastructure as Code (IaC).
4. Integrate Monitoring and Logging.
5. Apply Containerization and Orchestration Concepts.
6. Ensure Security and Compliance in DevOps Pipelines.

**Syllabus**

**Theory**

**Unit 1: Introduction to DevOps and Cloud Computing (4 Hours)**
DevOps principles and practices, Cloud computing models (IaaS, PaaS, SaaS),DevOps benefits in cloud environments,Overview of public, private, and hybrid clouds

**Unit 2: Source Control and CI/CD Fundamentals (5 Hours)**
Git and GitHub basics, Branching, merging, and pull requests, Introduction to Continuous Integration,Setting up Jenkins for CI/CD

**Unit 3: Containerization and Orchestration (5 Hours)**
Docker fundamentals and images,Docker Compose for multi-container applications, Kubernetes basics (pods, deployments, services),Container orchestration in cloud environments

**Unit 4: Infrastructure as Code and Configuration Management (5 Hours)**
Infrastructure as Code (IaC) concepts, Using Terraform for cloud provisioning, Configuration management with Ansible, State management and automation scripts

**Unit 5: Monitoring, Logging, and Security (5 Hours)**
Monitoring with Prometheus and Grafana,Logging with ELK stack,DevSecOps principles and security integration,Alerting and incident management in the cloud

**Unit 6: Advanced DevOps and Cloud Practices (4 Hours)**
Serverless computing and cloud functions, GitOps and automated deployments, Blue-green and canary deployments, Site reliability engineering (SRE) basics

## Syllabus
## Laboratory

**List of Experiments**

1. Set up Git and GitHub repository with branching workflow
2. Install and configure Jenkins for CI
3. Build and run Docker containers for a sample application
4. Create a multi-container application using Docker Compose
5. Deploy an application to Kubernetes using kubectl
6. Write Terraform scripts to provision AWS/GCP infrastructure
7. Automate server configuration with Ansible playbooks
8. Integrate Jenkins with Git and Docker for a CI/CD pipeline
9. Monitor application metrics using Prometheus and Grafana
10. Analyze logs with ELK stack (Elasticsearch, Logstash, Kibana)
11. Secure a CI/CD pipeline using DevSecOps tools (e.g., Snyk)
12. Set up blue-green deployment for a cloud application

## Syllabus
## Course Projects

**List of Course Projects**

1. Build and deploy a microservices app using Docker and Kubernetes
2. Create a CI/CD pipeline for a web app using Jenkins and Git
3. Automate infrastructure provisioning using Terraform
4. Configure a secure cloud environment with IAM and Ansible
5. Build a DevOps dashboard with monitoring and alerting
6. Integrate DevSecOps tools in CI/CD pipeline

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Explain the principles of DevOps and cloud computing models.
2. Use version control and CI/CD tools to automate software delivery.
3. Implement containerized and orchestrated deployments using Docker and Kubernetes.
4. Automate cloud infrastructure provisioning and configuration using IaC tools.
5. Monitor and secure cloud-based systems using observability tools and DevSecOps.
6. Design and execute a complete cloud-native DevOps workflow for real-world applications.

## Books and E-Resources

**Text Books and References**

1. Gene Kim, Jez Humble, Patrick Debois, John Willis – The DevOps Handbook
2. Len Bass, Ingo Weber, Liming Zhu – DevOps: A Software Architect's Perspective
3. Kelsey Hightower – Kubernetes: Up and Running
4. James Turnbull – The Docker Book
5. Yevgeniy Brikman – Terraform: Up & Running

**MOOC References**

1. Coursera: Continuous Delivery & DevOps by University of Virginia
2. edX: Introduction to DevOps by Microsoft
3. Udemy: DevOps Projects – CI/CD with Jenkins, Kubernetes, Docker
4. LinkedIn Learning: Learning Ansible and Terraform
5. Pluralsight: DevOps Paths (CI/CD, Containers, Monitoring)

## Honors in Cloud Computing

## ABCXXX: Cloud Security

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week
Total Credits: 03

**Prerequisites:** Cloud Fundaments, Computer Network

**Course Objectives**

1. To understand the fundamental principles, models, and shared responsibility architecture of cloud security.
2. To explore the major threats, vulnerabilities, and risks in cloud computing environments.
3. To enable students to design secure identity and access management (IAM) systems using cloud-native tools.
4. To apply data protection mechanisms such as encryption, key management, and secure storage in cloud platforms.
5. To secure cloud network infrastructures including virtual private networks (VPNs), firewalls, and container security.
6. To introduce security operations practices such as threat detection, logging, auditing, compliance, and incident response in cloud environments.

### Syllabus
### Theory

**Unit 1: Introduction to Cloud Security (4 Hours)**

Overview of Cloud Computing Models (IaaS, PaaS, SaaS), Shared Responsibility Model, Security Challenges in Cloud Environments, Threat Landscape: Data breaches, account hijacking, insider threats

**Unit 2: Identity and Access Management (IAM) (4 Hours)**

IAM Fundamentals, Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC),Federated Identity, Single Sign-On (SSO), and Multi-Factor Authentication (MFA),IAM Implementation in AWS, Azure, and GCP

**Unit 3: Data Security and Encryption (5 Hours)**

Data-at-Rest, Data-in-Transit, and Data-in-Use Protection, Encryption Techniques and Key Management, Cloud Key Management Services (AWS KMS, Azure Key Vault, Google Cloud KMS),Tokenization and Masking

**Unit 4: Network Security and Virtualization Risks (5 Hours)**
Cloud Network Architecture and Security Groups, Virtual Private Clouds (VPCs), VPNs, Firewalls, Secure APIs and Service Mesh, Container and Hypervisor Security

**Unit 5: Security Monitoring, Logging, and Incident Response (5 Hours)**
Security Information and Even,t Management (SIEM)Audit Trails and Logging (CloudTrail, Azure Monitor, GCP Logging),Intrusion Detection and Prevention Systems (IDPS),Incident Response in Cloud Environments

**Unit 6: Compliance, Governance, and Emerging Trends (5 Hours)**
Cloud Compliance Standards (ISO 27001, NIST, GDPR, HIPAA),Cloud Security Posture,Management (CSPM),DevSecOps in Cloud,Zero Trust Architecture, SASE, Confidential Computing

## Syllabus Laboratory

**List of Lab Experiments**

1. Setup and Configure IAM Policies on AWS/GCP
2. Implement Multi-Factor Authentication (MFA)
3. Create and Secure a Virtual Private Cloud (VPC)
4. Deploy Encrypted Storage Volumes
5. Use AWS KMS or Azure Key Vault for Data Encryption
6. Configure Firewalls and Network ACLs in the Cloud
7. Set Up Monitoring with AWS CloudWatch or GCP Monitoring
8. Detect and Block Suspicious Logins Using IAM Logs
9. Simulate a Cloud-Based Attack and Mitigation
10. Set Up SIEM (e.g., Splunk or OSSIM) with Cloud Logs
11. Use DevSecOps Tools (e.g., Checkov, Aqua, or Snyk)
12. Audit Cloud Resources Against CIS Benchmarks

## Syllabus
## Course Projects

**List of Course Projects**

1. IAM Policy Enforcement Tool for Cloud
2. Real-Time Cloud Threat Detection Dashboard
3. Secure File Storage Application using AWS S3 + KMS
4. Automated Incident Response Workflow with Lambda
5. DevSecOps Pipeline with Vulnerability Scanning
6. SIEM Integration with Multi-Cloud Environments

## Course Outcomes

**Course Outcomes:** Students will be able to

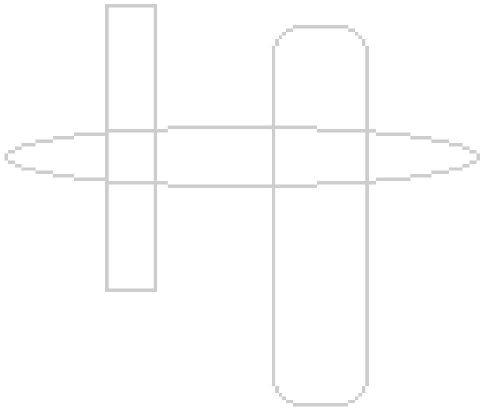
1. Explain the fundamentals of cloud security models and identify major risks and threats in cloud ecosystems.
2. Implement secure Identity and Access Management policies using cloud-native tools.
3. Apply encryption, data protection, and key management strategies to secure cloud-stored data.
4. Analyze and secure cloud network infrastructure including firewalls, VPCs, and containerized environments.
5. Monitor and respond to security incidents using logging, auditing, and SIEM tools.
6. Design a cloud security framework that aligns with compliance standards and integrates with modern DevSecOps workflows.

**Textbook References**

1. "Cloud Security and Privacy"By Tim Mather, Subra Kumaraswamy, and Shahed Latif Publisher: O'Reilly Media ISBN: 978-0596802769
2. "Cloud Computing: Concepts, Technology & Architecture" By Thomas Erl, Ricardo Puttini, Zaigham Mahmood Publisher: Prentice Hall ISBN: 978-0133387520
3. "Cloud Security: A Comprehensive Guide to Secure Cloud Computing" By Ronald L. Krutz and Russell Dean Vines Publisher: Wiley ISBN: 978-0470589873
4. "Architecting the Cloud: Design Decisions for Cloud Computing Service Models" By Michael J. Kavis Publisher: Wiley ISBN: 978-1118617618

**MOOC References**

1. Coursera – Cloud Security by University of Maryland
2. edX – Cloud Security Fundamentals (by IBM)
3. Udemy – Cloud Security Fundamentals
4. LinkedIn Learning – Cloud Security Considerations
5. AWS Training – AWS Cloud Security Essentials



# Honors in Cloud Computing

## ABCXXX: Cloud-Based Application Development and Serverless Computing

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week; Tutorials: 1 Hours/ Week
Total Credits: 4

**Prerequisites**: Programming Fundaments, Web Technology, Operating System, Computer Network

**Course Objectives**

1. Understand the principles of cloud-native and serverless application development.
2. Explore Function-as-a-Service (FaaS) platforms such as AWS Lambda, Azure Functions, and Google Cloud Functions.
3. Develop event-driven applications using cloud-based services and APIs.
4. Integrate storage, authentication, and messaging services into cloud applications.
5. Implement CI/CD pipelines and version control for serverless deployments.
6. Evaluate the performance, scalability, and cost-efficiency of serverless architectures.

### Syllabus
### Theory

**Unit 1: Introduction to Cloud and Serverless Computing (4 Hours)**
Overview of cloud computing paradigms: IaaS, PaaS, SaaS, What is Serverless Computing?, Benefits and limitations, Cloud service models and providers: AWS, Azure, GCP, Use cases and real-world applications

**Unit 2: Cloud-Native Application Design (5 Hours)**
12-factor app principles, Microservices architecture, Containers vs Serverless, CI/CD in cloud environments, Infrastructure as Code (IaC)

**Unit 3: Serverless Platforms and Functions (6 Hours)**
AWS Lambda, Azure Functions, Google Cloud Functions, Function lifecycle and triggers, Stateless programming model, Event-driven architecture, Pricing models

**Unit 4: Data Management in Cloud Applications (4 Hours)**
Cloud storage: S3, Blob, Cloud Storage, Serverless databases: DynamoDB, Firebase, Cosmos DB, Event streaming (e.g., Kinesis, Pub/Sub), Data security and privacy

**Unit 5: API Management and Backend Services (6 Hours)**
Building and deploying RESTful APIs with serverless, API Gateway integration, Authentication and authorization (JWT, OAuth2), Rate limiting, throttling, monitoring

**Unit 6: Real-World Case Studies and Emerging Trends (4 Hours)**
Case studies: Netflix, Uber, Airbnb, Edge computing and FaaS, Monitoring and debugging serverless applications, Cost optimization strategies, Green computing and sustainability

## Syllabus
## Laboratory

**List of Lab Experiments**

1. Deploying a static website on AWS S3
2. Creating and invoking a basic AWS Lambda function
3. Building a serverless API using AWS Lambda + API Gateway
4. Integrating serverless with a database (e.g., DynamoDB)
5. Writing an Azure Function with Blob Trigger
6. Google Cloud Function for file processing
7. Serverless image processing pipeline
8. Creating a CI/CD pipeline for serverless apps
9. Monitoring Lambda performance using CloudWatch
10. Securing APIs with AWS Cognito
11. Event-driven email notification service
12. Cost analysis and optimization of serverless workloads

## Syllabus
## Tutorials

**List of Tutorials**

1. Introduction to cloud environments
2. Hands-on with serverless architecture diagramming
3. AWS CLI basics and account setup
4. Designing microservices for serverless

5. Writing handler functions with Node.js/Python
6. Exploring IAM roles and permissions
7. Trigger mechanisms: HTTP, S3, Scheduler
8. Introduction to Firebase Functions
9. Real-time data streaming using Pub/Sub
10. Testing and logging for Lambda

## Syllabus
## Course Projects

**List of Course Projects**

1. Smart attendance system using serverless backend
2. Chatbot using AWS Lex and Lambda
3. Serverless blog platform
4. IoT data ingestion with serverless processing
5. Real-time chat app using Firebase
6. AI image classifier with Lambda and S3
7. Social media sentiment analyzer
8. E-commerce backend with API Gateway and DynamoDB

**Course Outcomes:**

1. Understand the fundamentals and architecture of cloud and serverless computing.
2. Design and implement cloud-native applications using serverless paradigms.
3. Work with serverless platforms such as AWS Lambda, Azure Functions, and GCF.
4. Integrate serverless functions with cloud storage, APIs, and databases.
5. Apply CI/CD and monitoring practices in serverless application lifecycle.
6. Analyze real-world use cases and optimize performance and cost of solutions.

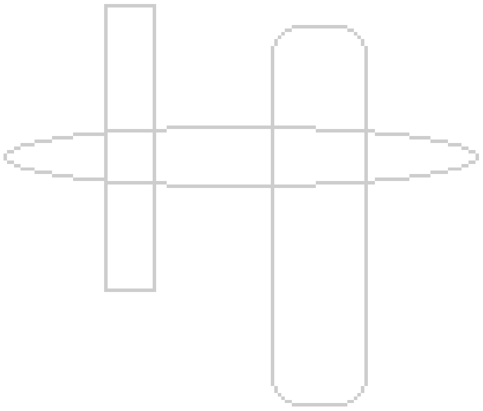
**Textbook References**

1. Serverless Architectures on AWS: With examples using AWS Lambda by Peter Sbarski Publisher: Manning Publications
2. Cloud Native Applications with Node.js and AWS Lambda by Roland Bischofberger Publisher: Packt Publishing

3. Learning AWS: Design, Build, and Deploy Responsive Applications using AWS Cloud Components by Aurobindo Sarkar Publisher: Packt Publishing
4. Programming AWS Lambda: Build and Deploy Serverless Applications with Java by John Chapin, Mike Roberts
   Publisher: O'Reilly Media

**MOOC References**

1. Serverless Framework Bootcamp – Udemy
   https://www.udemy.com/course/serverless-framework
2. Developing Cloud Apps with Node.js and React – Coursera (IBM)
   https://www.coursera.org/learn/cloud-app-development
3. Serverless Machine Learning with AWS Lambda – Coursera (AWS)
   https://www.coursera.org/learn/serverless-machine-learning
4. AWS Serverless Applications – AWS Skill Builder (Free Course)
   https://explore.skillbuilder.aws/learn/course/164/aws-serverless-applications



## Honors in Cloud Computing

## ABCXXX: Project

Teaching Scheme:
Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

1. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
2. The project requires the students to conceive, design, implement and operate a mechanism (the design problem). The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts. If the mechanism

incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

3. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
4. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
5. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
6. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
7. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
8. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Natural Language Processing

| Course No. | Year and Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | Credits |
|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | Total | |
| C1 | III | Foundations of Natural Language Processing | 2 | 2 | 0 | | | | 3 |
| C2 | IV | Classification and Vector Spaces | 2 | 2 | 0 | | | | 3 |
| C3 | V | Probabilistic and Sequence Models | 2 | 2 | 0 | | | | 3 |
| C4 | VI | Large Language Models | 2 | 2 | 0 | | | | 3 |
| C5 | VII | Natural Language Processing Applications | 2 | 2 | 1 | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | 4 |
| | | Total | | | | | | | |

## Honors in Natural Language Processing

### ABCXXX: Foundations of Natural Language Processing

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week
Total Credits: 3

**Course Objectives:**
1. To understand morphology for given natural language
2. To learn how to design lexical analyzer for given natural language
3. To learn how to design Syntactic Analyzer for given natural language
4. To learn how to design type dependency parser using pragmatic approach for given natural language

**Course Relevance:** Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

**Syllabus**
**Theory**

**Unit 1: Introduction** **(4 Hours)**
Introduction, what is natural language processing? Applications of NLP, Origins of NLP, Challenges of NLP, Language and Knowledge, Language and Grammar, Processing Indian Languages

**Unit 2: Language Modelling** **(6 Hours)**
Grammar-based language models, lexical functional Grammar (LFG), Government and Binding (GB), Lexical functional Grammar Model, Generative grammars, Statistical Language Model. N-gram Language Models - N-Grams, Evaluating Language Models, Sampling sentences from a language model, Generalization and Zeros, Smoothing, Kneser-Ney Smoothing, Huge Language Models and Stupid Backoff, Advanced: Perplexity's Relation to Entropy.

**Unit 3: Regular Expressions and Automata (5 Hours)**
Formal Language Theory: Basic Notions, Basic Regular Expression Patterns, Disjunction, Grouping and Precedence, Advanced Operators, Substitution, Finite State Automata, NFSA. Words, Corpora, Text Normalization, Minimum Edit Distance.

**Unit 4: Words and Transducers (5 Hours)**
Morphology, Inflectional Morphology, Derivational Morphology, Finite State Morphological Parsing, Construction of Finite State Lexicon, Finite State Transducers

**Unit 5: Syntactic Analysis (5 Hours)**
Context Free Grammar, parsing, Top-down Parsing, Bottom-up parsing, Probabilistic parsing, Indian Languages parsing

**Unit 6: Semantic Analysis (5 Hours)**
Meaning Representation, Lexical Semantic, Ambiguity, Word Sense Disambiguation, Discourse processing, Natural Language Generation

## Syllabus
## Laboratory

**List of Experiments**

1. Manual Word Analysis and Word Generation on plain paper
2. Manual  Morphology on plain paper
3. Manual  N-Grams and N-Grams Smoothing on plain paper
4. Manual  Chunking on plain paper
5. Manual Building Chunker on plain paper
6. Write a program for Word Analysis
7. Write a program for Word Generation
8. Write a program for Morphology
9. Write a program for N-Gram generations
10. Write a program for N-Grams Smoothing
11. Write a program for Chunking
12. Write a program for to build Building Chunker

## Course Outcomes

**Course Outcomes**: Student will able to

7. Interpret language and grammar for given natural language
8. Infer grammar based language modelling for given natural language
9. Perform normalization for given natural language
10. Prepare Finite State Lexicons for Finite State Transducers
11. Construct shallow depth lexical analyzer and syntactic analyzer
12. Develop shallow depth type dependency parser for given natural language

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**

1. Tanveer Siddiqui and U S Tiwary, "Natural Language Processing and Information Retrieval" Fourth Impression, Oxford,  ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

**Reference Books: (As per IEEE format)**

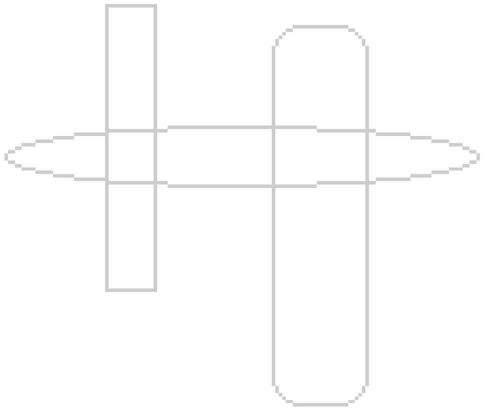
1. Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8
2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

**For Reference Electronic Book –**

1. Dan Jurafsky and James H. Martin; " Speech and Language Processing (3rd ed. draft)"; Jan 2025.
   Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

**For MOOCs and other learning Resources**

2. www.nptelvideos.in
3. www.nfnlp.com
4. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
5. https://www.cse.iitb.ac.in/~cs626-460-2012/



# Honors in Natural Language Processing

## ABCXXX: Classification and Vector Spaces

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week
Total Credits: 3

**Course Objectives:**
1. To learn how to use Naive Bayes for Text Classification
2. To learn how to use logistic regression sentiment analysis
3. To learn how to use represent words as vectors
4. To learn how to use different embeddings in NLP applications

**Course Relevance:** Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

**Syllabus**

**Theory**

**Unit 1: Naive Bayes Text Classification with Minimal Features (5 Hours)**
Naive Bayes Classifiers, Training the Naive Bayes Classifier, Sentiment Analysis - Worked example, Optimizing for Sentiment Analysis

**Unit 2: Naive Bayes Text Classification with all Word Features (5 Hours)**
Naive Bayes for other text classification tasks- Spam Detection, Naive Bayes as a Language Model, Evaluation: Precision, Recall, F-measure, Test sets and Cross-validation, Statistical Significance Testing, Avoiding Harms in Classification

**Unit 3: Logistic Regression (5 Hours)**
Generative and Discriminative Classifiers, Components of a probabilistic machine learning classifier, The sigmoid function, Classification with Logistic Regression: Sentiment Classification, Other classification tasks and features, Designing versus learning features, Scaling input features, Processing many examples at once, Choosing a classifier

**Unit 4: Multinomial Logistic Regression (5 Hours)**
Features in Multinomial Logistic Regression, Learning in Logistic Regression, The cross-entropy loss function, Gradient Descent, The Gradient for Logistic Regression, The Stochastic Gradient Descent Algorithm, Working through an example, Mini-batch training, Regularization, Learning in Multinomial Logistic Regression, Interpreting models, Advanced: Deriving the Gradient Equation

**Unit 5: Vector Semantics (5 Hours)**
Lexical Semantics: Lemmas and Senses, Synonymy, Word Similarity, Word Relatedness, semantic field, Semantic Frames and Roles, Connotation. Vector Semantics: Words and Vectors, Vectors and documents, Words as vectors: document dimensions, Words as vectors: word dimensions

**Unit 6: Embeddings (5 Hours)**
Cosine for measuring similarity, TF-IDF: Weighing terms in the vector, Point-wise Mutual Information (PMI), Applications of the TF-IDF or PPMI vector models, Word2vec, The classifier using Similarity, Learning skip-gram embeddings, Other static embeddings: fasttext, GloVe, Visualizing Embeddings, Semantic properties of embeddings, Bias and Embeddings, Evaluating Vector Models

## Syllabus Laboratory

**List of Experiments**

1. Build a Naive Bayes classifier using only 10–15 key sentiment words from the IMDb movie reviews dataset.
   - Select top sentiment words (e.g., “good,” “bad,” “love”).
   - Create feature vectors based on these words only.
   - Train and test the model with Laplace smoothing.
   - Evaluate accuracy, precision, recall, and F1-score.

2. Classifying movie reviews from IMDb. Predict if a review is positive or negative. Use a dataset of the IMDb review dataset.
   - Create a bag-of-words model.
   - Apply the Naive Bayes formula
   - Visualize key sentiment words using word clouds

3. Build a Naive Bayes classifier for Sentiment Analysis using all word features from the full vocabulary of IMDb movie reviews.
   - Use all words after pre-processing (tokenization, optional stop word removal).
   - Vectorize using count **or TF-IDF.**
   - Train/test with Laplace smoothing, evaluate with metrics and ROC curve.
   - Compare performance against minimal feature models.

4. Automated Customer Feedback Routing Using Naive Bayes Classification

   **Dataset :**https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter/data
   - Pre-process and label a representative dataset of customer tweets.
   - Train a Naive Bayes classifier (Multinomial or Bernoulli) to classify tweets into:
     - ➢ Technical Support
     - ➢ Billing Issues
     - ➢ General Queries
   - Compute the confidence/probability of classification for each incoming tweet.
   - Evaluate model accuracy using precision, recall, F1-score, and confusion matrix.
   - Create a visualization dashboard showing real-time routing of incoming tweets.

5. Compare Naive Bayes (generative) and Logistic Regression (discriminative) for spam detection using the Enron Email Dataset. Analyze and report on accuracy and feature importance

   **Dataset :** https://www.kaggle.com/datasets/mohinurabdurahimova/maildataset/data

6. Build a Binary Sentiment Classification Using Logistic Regression Model to classify product reviews as positive or negative based on review text.
   - Use sigmoid function
   - feature scaling
   - Evaluate the model using precision, recall, and F1-score.

**Dataset** :https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products

7. Use multinomial logistic regression to classify sentences into one of six emotions: *joy, sadness, anger, fear, love, surprise*.
   - Split the training data into small batches (e.g., 32 or 64 samples per batch).
   - For each batch:
     - Compute the gradient of the cross-entropy loss with respect to the weights.
     - Update the model parameters using the gradient.
   - Repeat until convergence over several epochs.
   - Apply L2 regularization.
   - Compare the model performance with regularization and without regularization

8. Conduct a multi-class classification task (e.g., classifying Stack Exchange questions by topic) and analyze learning curves for different hyper-parameters

   **Dataset** : https://www.kaggle.com/datasets/nazeboan/stackoverflow-questions-classification-challenge

9. Develop a search engine that can retrieve the most relevant legal documents (contracts, terms, or clauses) based on a user's query, by computing semantic similarity between documents and queries.
   - Collect or use a dataset of legal texts (e.g., EU Legislation Dataset).
   - Pre-process texts (lemmatization, stop word removal).
   - Build a TF-IDF matrix for documents.
   - Convert the user query into a TF-IDF vector.
   - Compute cosine similarity between query vector and all document vectors.
   - Rank and return top-N matching documents

10. Analyze pre-trained word embeddings to detect and quantify gender bias associated with job titles.

- Select gender words (e.g., he, she, man, woman) and job titles (e.g., doctor, nurse, engineer, teacher).
- Use pre-trained embeddings like GloVe or Word2Vec Google News.
- Extract vectors for selected words and compute cosine similarities between job titles and gender words.
- Identify job titles closer to male or female terms to reveal gender bias.
- Visualize results using charts or tables.

11. Create a recommendation engine that suggests similar products based on customer reviews using both TF-IDF (sparse) and Word2Vec (dense) vectors. Compare their effectiveness
    - Clean and Vectorize reviews using TF-IDF and Word2Vec.
    - Calculate similarity between reviews of different products.
    - Recommend top 3 semantically similar products.
    - Compare recommendation performance using both vector methods.

**Dataset :** https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

12. Classify tweets into emotion labels (e.g., joy, anger, sadness) using pre-trained Word2Vec embeddings and cosine similarity between tweet vectors and class centroids.

- Preprocess the tweets and tokenize.
- Use pre-trained Word2Vec (Google News or GloVe vectors).
- Average word vectors to obtain sentence embeddings.
- Compute centroid vectors for each emotion class.
- Classify new tweets based on cosine similarity to emotion centroids.

Dataset : https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp

## Course Outcomes

**Course Outcomes:** Student will be able to

1. Apply Naive Bayes for Text Classification with minimal features
2. Design and develop Apply Naive Bayes for Text Classification with all features
3. Apply logistic regression for sentiment analysis
4. Design and develop multinomial logistic regression for sentiment analysis
5. Construct vectors from words
6. Utilize different embeddings in NLP applications

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**

1. Tanveer Siddiqui and U S Tiwary, "Natural Language Processing and Information Retrieval" Fourth Impression, Oxford, ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

**Reference Books: (As per IEEE format)**

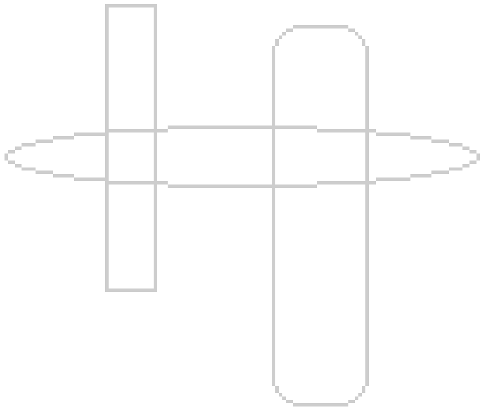
1. Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8
2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

**For Reference Electronic Book –**

1. Dan Jurafsky and James H. Martin; " Speech and Language Processing (3rd ed. draft)"; Jan 2025. Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

**For MOOCs and other learning Resources**

2. www.nptelvideos.in
3. www.nfnlp.com
4. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
5. https://www.cse.iitb.ac.in/~cs626-460-2012/



## Honors in Natural Language Processing

### ABCXXX: Probabilistic and Sequence Models

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week
Total Credits: 3

**Course Objectives:**

1. To learn how to use Sequence Labeling for POS and Named Entities
2. To learn how to use Conditional Random Fields for named entity recognition
3. To learn how to use Neural Network for language modelling
4. To learn how to Feed-forward Neural for Language Modeling
5. To learn how to RNN for Language Modeling
6. To learn how to use Feed-forward mechanism for Language Modeling

**Course Relevance:** Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

**Syllabus**

**Theory**

**Unit 1: Sequence Labeling for POS and Named Entities (5 Hours)**
English Word Classes, Part-of-Speech Tagging, Named Entities and Named Entity Tagging, Markov Chains, The Hidden Markov Model, The components of an HMM tagger, HMM tagging as decoding, The Viterbi Algorithm, Working through an example

**Unit 2: Conditional Random Fields (CRFs) (5 Hours)**
Conditional Random Fields, Features in a CRF POS Tagger, Features for CRF Named Entity Recognizers, Inference and Training for CRFs, Evaluation of Named Entity Recognition,

**Unit 3: Neural Language Modeling (5 Hours)**
Neural Networks, Neural Single Computational Unit, The XOR problem, The solution: neural networks, Feed-forward Neural Networks, details on Feed-forward networks, Feed-forward networks for NLP, Training Neural Nets, Loss function, Computing the Gradient, Computation Graphs, Backward differentiation on computation graphs, details on learning

**Unit 4: Feed-forward Neural Language Modeling (5 Hours)**
Feed-forward Neural Language Modeling, Forward inference in the neural language model, Training the neural language model

**Unit 5: RNN Language Model (5 Hours)**
Recurrent Neural Networks, Inference in RNNs, Training, RNNs as Language Models, Forward Inference in an RNN language model, Training an RNN language model, Weight Tying, RNNs for other NLP tasks: Sequence Labeling, RNNs for Sequence Classification, Generation with RNN-Based Language Models, Stacked and Bidirectional RNN architectures, Stacked RNNs, Bidirectional RNNs

**Unit 6: RNN-LSTM NLP Architectures (5 Hours)**
The LSTM, Gated Units, Layers and Networks, Common RNN NLP Architectures, The Encoder-Decoder Model with RNNs, Training the Encoder-Decoder Model, Attention

## Syllabus
## Laboratory

**List of Experiments**

1. Implement the forward algorithm and run it with HMM to compute the probability of the observation sequences 331122313 and 331123312 which is more likely?
2. Implement the Viterbi algorithm and run it with the HMM to compute the most likely weather sequences for each of the two observation sequences above,331122313 and 331123312.
3. Implement a CRF model to tag POS in informal texts (e.g., tweets) and evaluate how well the model handles slang and abbreviations
4. Develop a Named Entity Recognition (NER) system using Conditional Random Fields (CRF) to extract structured medical entities—such as disease names, medications, and patient identifiers—from anonymized electronic health records (EHRs)..
5. Create a CRF with custom features (capitalization, legal stop words) to identify contract parties and obligations in PDF agreement
6. Build and train a simple feed-forward neural network to predict the next word in a sentence from a given text corpus.
7. Implement and visualize backward differentiation for a small neural network trained to classify text sentiment.

8. Develop a feed-forward neural network to predict the relevance between user search queries and product titles in an e-commerce setting. Use pre-trained word embeddings (e.g., Word2Vec, GloVe, or BERT embeddings) to convert queries and product titles into fixed-length vectors, and train the model to output a relevance score
9. Train a feed-forward language model to predict word sequences in product reviews and evaluate performance on domain-specific vocabulary.
10. Use a bidirectional RNN to classify user comments as toxic or non-toxic for content moderation applications.
11. Design an LSTM-based sequence labeller to extract legal entities such as parties, case numbers, and statutes from case summaries.
12. Implement LSTM encoder-decoder architecture with attention to translate short English phrases into Hindi.
13. Develop a Bi-LSTM model to extract clinical events and symptom onset dates from patient reports to construct event timelines.

## Course Outcomes

**Course Outcomes**: Student will be able to
1. Apply Sequence Labeling for POS and Named Entities
2. Design and develop Named Entity Recognition using Conditional Random Fields
3. Apply Neural Network for language modelling
4. Design and develop applications using Feed-forward Neural Language Modeling
5. Apply RNN for Language Modeling
6. Design and develop applications using Feed-forward mechanism for Language Modeling

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**
1. Tanveer Siddiqui and U S Tiwary, “Natural Language Processing and Information Retrieval” Fourth Impression, Oxford,  ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

**Reference Books: (As per IEEE format)**

1. Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8
2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

**For Reference Electronic Book –**

1. Dan Jurafsky and James H. Martin; " Speech and Language Processing (3rd ed. draft)"; Jan 2025.

Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

**For MOOCs and other learning Resources**

2. www.nptelvideos.in
3. www.nfnlp.com
4. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
5. https://www.cse.iitb.ac.in/~cs626-460-2012/

## Honors in Natural Language Processing

## ABCXXX: Large Language Models

Teaching Scheme:
Theory: 2 Hours/Week; Laboratory: 2/Week
Total Credits: 3

**Course Objectives:**

1. To learn various attention mechanisms in Transformer model
2. To learn Large Language Models
3. To learn how to train and evaluating Large Language Models
4. To learn Masked Language Models
5. To learn MLM Fine-Tuning for Classification
6. To learn how to use Model Alignment, Prompting, and In-Context Learning

**Course Relevance:** Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

### Syllabus
### Theory

**Unit 1: The Transformer** **(5 Hours)**
The Transformer , Attention, Attention more formally , Simplified version of attention , A single attention head using query, key, and value matrices, Multi-head Attention, Transformer Blocks, Parallelizing computation using a single matrix, Parallelizing attention, Masking out the future, Parallelizing multi-head attention, The input: embeddings for token and position, The Language Modeling Head

**Unit 2: Large Language Models** **(5 Hours)**
Large Language Models: Large Language Models with Transformers, Sampling for LLM Generation; Top-k sampling, Nucleus or top-p sampling, Temperature sampling, Pre-training Large Language Models, Self-supervised training algorithm,

**Unit 3: Training and Evaluating Large Language Models (5 Hours)**
Training corpora for large language models, Evaluating Large Language Models : Perplexity, Fairness, HELM protocol, Dealing with Scale, Scaling laws, KV Cache, Parameter Efficient Fine Tuning, Potential Harms from Language Models

**Unit 4: Masked Language Models (5 Hours)**
Masked Language Models: Bidirectional Transformer Encoders, The architecture for bidirectional masked models, Training Bidirectional Encoders, Masking Words, Next Sentence Prediction, Training Regimes, Contextual Embeddings, Contextual Embeddings and Word Sense, Word Sense Disambiguation, Contextual Embeddings and Word Similarity,

**Unit 5: MLM Fine-Tuning for Classification (5 Hours)**
Fine-Tuning for Classification, Sequence Classification, Sequence-Pair Classification, Fine-Tuning for Sequence Labelling: Named Entity Recognition, Named Entities, BIO Tagging, Sequence Labeling, Evaluating Named Entity Recognition,

**Unit 6: Model Alignment, Prompting, and In-Context Learning (5 Hours)**
Prompting, Learning from Demonstrations: Few-Shot Prompting, In-Context Learning and Induction Heads, Post-training and Model Alignment, Model Alignment: Instruction Tuning
Instruction, Instructions as Training Data, Evaluation of Instruction-Tuned Models, Chain-of-Thought Prompting, Automatic Prompt Optimization, Evaluating Prompted Language Models, Model Alignment with Human Preferences: RLHF and DPO

## Syllabus
## Laboratory

**List of Experiments**

1. Implement a mini-Transformer with multi-head attention, positional encodings, and masking to train on a small corpus for next-word prediction.
2. Create interactive visualizations of sinusoidal vs learned positional embeddings using Stack Overflow Q&A data
3. Compare Top-k vs Nucleus sampling for generating drug interaction responses using the DrugQA dataset
4. Pre-train a Transformer model on a domain-specific corpus (e.g., medical or legal text) using self-supervised learning to observe language understanding improvements.

5. Train or fine-tune a Transformer on a small corpus and evaluate it using perplexity and fairness indicators such as gender bias.
6. Apply LoRA or Adapter techniques for parameter-efficient fine-tuning on a classification task, comparing memory and accuracy trade-offs.
7. Train a masked language model on a domain-specific dataset to predict missing tokens and improve contextual word understanding.
8. Use contextual embeddings from BERT to disambiguate words in ambiguous sentences (e.g., "bank" as a riverbank or financial bank).
9. Fine-tune a masked language model like BERT to classify customer reviews into positive, neutral, and negative sentiment classes.
10. Fine-tune a model to classify whether a given pair of sentences are logically entailed, contradictory, or neutral.
11. Design few-shot prompts using chain-of-thought reasoning for solving grade-school math problems using an open-source LLM.
12. Evaluate LLM responses generated via prompting using human-aligned metrics (e.g., coherence, helpfulness), and explore how reward models or DPO refine outputs.

## Course Outcomes

**Course Outcomes**: Student will be able to

1. Apply various attention mechanisms in Transformer model
2. Select appropriate Large Language Models
3. Train and evaluating Large Language Models
4. Select Masked Language Models
5. Perform MLM Fine-Tuning for Classification
6. Design and develop prompt engineering application

## Books and E-Resources

**<u>For Reference Print Book -</u>**

**Text Books: (As per IEEE format)**

1. Tanveer Siddiqui and U S Tiwary, "Natural Language Processing and Information Retrieval" Fourth Impression, Oxford,  ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

**Reference Books: (As per IEEE format)**

1. Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8
2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

**For Reference Electronic Book –**

1. Dan Jurafsky and James H. Martin; "Speech and Language Processing (3rd ed. draft)"; Jan 2025.
   Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

**For MOOCs and other learning Resources**

2. www.nptelvideos.in
3. www.nfnlp.com
4. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
5. https://www.cse.iitb.ac.in/~cs626-460-2012/

## Honors in Natural Language Processing

## ABCXXX: Natural Language Processing Applications

Teaching Scheme:
Theory: 3 Hours/Week; Laboratory: 2/Week
Total Credits: 4

**Course Objectives:**
1. To learn how to use Encoder-Decoder Model for Machine Translation
2. To learn how to evaluate Machine Translation
3. To learn how to design Information Retrieval and Question Answering Systems
4. To learn how to design Chatbots & Dialogue Systems
5. To learn how to design Automatic Speech Recognition
6. To learn how to design Text To Speech System

**Course Relevance:** Although Natural Language Processing (NLP) has been with us for quite some time, it has only recently gained industry-wide attention, thanks to Deep Learning. Today, NLP is a core competence area in Data Science and IT, with applications spanning across sectors that rely on harnessing language data's potential. Essentially, GEN based NLP applications are designed to extract relevant and meaningful information from natural human language data and impart machines with the ability to interact with humans.

### Syllabus
### Theory

**Unit 1: Encoder-Decoder Model for Machine Translation (5 Hours)**
Introduction, Language Divergences and Typology: Word Order Typology, Lexical Divergences, Morphological Typology, Referential density, Machine Translation using Encoder-Decoder: Tokenization, Creating the Training data; Details of the Encoder-Decoder Model

**Unit 2: Decoder and Evaluation of Machine Translation (5 Hours)**
Decoding in MT: Beam Search, Minimum Bayes Risk Decoding, Translating in low-resource situations, Data Augmentation, Multilingual models, Socio-technical issues, MT Evaluation: Using Human Raters to Evaluate MT, Automatic Evaluation: Automatic Evaluation by Character Overlap: chrF, Alternative overlap metric: BLEU, Statistical Significance Testing for MT evals, chrF: Limitations, Automatic Evaluation: Embedding-Based Methods, Bias and Ethical Issues

**Unit 3: Information Retrieval and Question Answering (5 Hours)**
Information Retrieval, Term weighting and document scoring, Document Scoring, Inverted Index, Evaluation of Information-Retrieval Systems, Information Retrieval with Dense Vectors, Answering Questions with RAG: Retrieval-Augmented Generation, Question Answering Datasets, Evaluating Question Answering,

**Unit 4: Chatbots & Dialogue Systems (5 Hours)**
Introduction, Properties of Human Conversation, Frame-Based Dialogue Systems, Dialogue Acts and Dialogue State, Chatbots, Training chatbots, Fine Tuning for Quality and Safety, Learning to perform retrieval as part of responding, RLHF for Reinforcement Learning from Human Feedback, Evaluating Chatbots; Dialogue System Design, Ethical Issues in Dialogue System Design

**Unit 5: Automatic Speech Recognition (5 Hours)**
Introduction, Automatic Speech Recognition Task, Feature Extraction for ASR: Log Mel Spectrum, Discrete Fourier Transform, Mel Filter Bank and Log, Speech Recognition Architecture,

**Unit 6: Connectionist Temporal Classification and Text To Speech (5 Hours)**
Connectionist Temporal Classification (CTC), CTC Training, Combining CTC and Encoder-Decoder, Streaming Models: RNN-T for improving CTC, ASR Evaluation: Word Error Rate,
TTS Pre-processing, TTS: Spectrogram prediction, TTS Vocoding, TTS Evaluation,

## Syllabus
## Laboratory

**List of Experiments**

1. Build a basic machine translation model to convert sentences from English to any target language using an encoder-decoder architecture and attention.
2. Analyze how differences in word order and morphological complexity affect translation accuracy and tokenization strategy.
3. Train a translation model and use beam search decoding. Evaluate its performance using BLEU and chrF metrics to compare output quality.
4. Evaluate translated text using BLEU, chrF, and embedding-based metrics to understand their correlation with human judgment.
5. Create an information retrieval system using dense embeddings (e.g., BERT) to rank and retrieve relevant research abstracts based on a query.

6. Implement a retrieval-augmented generation (RAG) model to answer questions on COVID-19 research articles
7. Develop a frame-based dialogue system to handle banking customer queries
8. Fine-tune a conversational model (like DialoGPT) and evaluate response coherence, relevance, and safety.
9. Pre-process audio using Log Mel spectrograms and train a model to transcribe spoken digits or commands.
10. Build an ASR system to transcribe customer service calls using the Common Voice dataset
11. Train a CTC-based model for speech recognition in noisy environments
12. Develop a TTS system to convert input text to spectrograms and use vocoding techniques to synthesize speech.

## Course Outcomes

**Course Outcomes**: Student will be able to
1. Apply Encoder-Decoder Model for Machine Translation
2. Evaluate Machine Translation
3. Design and develop Information Retrieval and Question Answering Systems
4. Design and develop Chatbots & Dialogue Systems
5. Design and develop Automatic Speech Recognition
6. Design and develop Text To Speech System

## Books and E-Resources

**For Reference Print Book -**

**Text Books: (As per IEEE format)**
1. Tanveer Siddiqui and U S Tiwary, “Natural Language Processing and Information Retrieval” Fourth Impression, Oxford,  ISBN-13:978-019-569232-7.
2. Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4

**Reference Books: (As per IEEE format)**
1. Alexander Clark, Chris Fox and Shalom Lappin “The Handbook of Computational Linguistics and Natural Language Processing”,Wiley-Blackwell-2013, ISBN-978-1-118-34718-8

2. Allen, James, Natural Language Understanding, Second Edition, Benjamin /Cumming, 1995.
3. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Radford, Andrew et. al., Linguistics, An Introduction, Cambridge University Press, 1999.
6. Journals : Computational Linguistics, Natural Language Engineering, Machine Learning, Machine Translation, Artificial Intelligence
7. Conferences : Annual Meeting of the Association of Computational Linguistics (ACL), Computational Linguistics (COLING), European ACL (EACL), Empirical Methods in NLP (EMNLP), Annual Meeting of the Special Interest Group in Information Retrieval (SIGIR), Human Language Technology (HLT).

**For Reference Electronic Book –**

1. Dan Jurafsky and James H. Martin; “Speech and Language Processing (3rd ed. draft)"; Jan 2025.
   Available: web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

**For MOOCs and other learning Resources**

2. www.nptelvideos.in
3. www.nfnlp.com
4. https://nlp-iiith.vlabs.ac.in/List%20of%20experiments.html?domain=Computer%20Science
5. https://www.cse.iitb.ac.in/~cs626-460-2012/

## Honors in Natural Language Processing

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

33. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
34. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

35. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
36. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
37. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
38. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).

39. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
40. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**

The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Parallel Computing

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | GPU Architecture and GPU Programming | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Parallel Algorithms for GPUs | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Parallel Programming Languages and Models I | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Parallel Programming Languages and Models II | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | High-Performance Computing | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |

## Honors in Parallel Computing

## ABCXXX: GPU Architecture and GPU Programming

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

**Syllabus**
**Theory**

**Unit 1: Introduction to GPUs** **(4 Hours)**
Evolution of GPU computing, Comparison of CPU and GPU architectures; Key GPU components: Streaming Multiprocessors (SMs), warps, schedulers, and cores; Instruction-level parallelism and thread-level parallelism; GPU execution model: SIMT (Single Instruction Multiple Thread); Memory hierarchy in GPU systems: registers, shared, global, local, texture, and constant memory.(Fermi Architecture)

**Unit 2:  CUDA Programming Basics** **(5 Hours)**
Overview of CUDA development environment; Structure of a CUDA program; Kernel functions and thread execution; Thread organization: threads, blocks, and grids; Launch configuration and kernel calls; Synchronization and barriers; Introduction to memory allocation and transfers between host and device.

**Unit 3:  GPU Memory Management and Optimization** **(5 Hours)**
Detailed study of CUDA memory types and access latency; Global vs shared memory; Memory coalescing; Bank conflicts in shared memory; Constant and texture memory usage; Shared memory usage and optimization strategies; Register spilling and occupancy tuning; Data alignment and access patterns for efficient memory utilization.

**Unit 4: Thread Scheduling and Warp Execution** **(5 Hours)**
Thread scheduling at warp level; Warp divergence and control flow behavior; SIMT execution model and predication; Role of warp schedulers and instruction issue logic; Managing divergence using conditional code structuring; Execution efficiency and its effect on performance; Occupancy calculator and hardware utilization.

**Unit 5: Advanced CUDA Features and Multi-GPU Programming (5 Hours)**

CUDA streams for concurrency; Asynchronous memory transfer and kernel execution; Event synchronization; Dynamic parallelism; Unified Memory and memory prefetching; Multi-GPU programming using CUDA-aware libraries; Peer-to-peer communication and multi-GPU memory handling.

**Unit 6: Case Studies and Real-World Applications (4 Hours)**

Performance benchmarking of GPU kernels; Applications in scientific computing: matrix operations, FFT, N-body simulation; Use of libraries: cuBLAS, cuFFT, Thrust; Deep learning inference acceleration using cuDNN; Optimization case studies: memory-bound vs compute-bound kernels; GPU architecture trends: Ampere, Hopper and beyond.

## Syllabus
## Laboratory

**List of Experiments**

1. Write and run your first CUDA kernel for vector addition
2. Analyze thread and block indexing in a 2D grid and print thread IDs
3. Matrix multiplication on GPU using naive and optimized shared memory version
4. Experiment with global and shared memory access; measure and compare latency
5. Implement a kernel with warp divergence and optimize its execution
6. Use of constant memory in CUDA for broadcasting values efficiently
7. Experiment with CUDA streams for concurrent kernel execution
8. Profile a kernel using Nsight Systems and identify memory bottlenecks
9. Measure performance impact of memory coalescing and bank conflicts
10. Use Thrust library for parallel sorting and reduction operations
11. Implement an application using Unified Memory
12. Develop a multi-GPU matrix addition program with peer-to-peer communication

## Syllabus
## Course Project

**List of Course Projects**

1. Real-Time Image Filtering and Enhancement using CUDA-Optimized GPU Kernels”
2. (Title) GPU-Accelerated K-Means Clustering for Large Datasets (Title)
3. CUDA-Based Parallel Prefix Sum and Histogram Equalization for Images (Title)
4. Real-Time Fluid Simulation Using GPU and CUDA (Title)
5. GPU-Accelerated Convolutional Neural Network (CNN) Inference Engine (Title)
6. 3D Game of Life Simulation on GPU (Title)
7. GPU-Based PageRank Algorithm for Web Graphs
8. Ray Tracing Renderer using GPU Parallelism
9. Monte Carlo Simulation for Option Pricing (Financial Computing)
10. GPU-Accelerated Sudoku Solver using Parallel Backtracking
11. Real-Time Face Detection using CUDA and Haar Cascades
12. GPU-Accelerated Weather Data Visualization and Interpolation

## Course Outcomes

**Course Outcomes**: Students will be able to

1. Explain the evolution of GPU architecture, the GPU execution model, and the memory hierarchy, distinguishing it from traditional CPU-based computing systems.*(Understand)*
2. Develop and execute parallel programs using CUDA by utilizing threads, blocks, grids, and kernel invocations for data-parallel tasks. *(Apply)*
3. Optimize CUDA programs by applying advanced memory management techniques including shared memory, memory coalescing, and minimizing warp divergence. *(Analyze)*
4. Evaluate the impact of warp-level thread scheduling, SIMT model, and occupancy on GPU performance through profiling and tuning. *(Evaluate)*

5. Implement applications using advanced CUDA features such as streams, asynchronous memory transfer, unified memory, and multi-GPU programming. *(Apply/Analyze)*
6. Analyze and solve real-world problems by applying GPU programming principles and CUDA libraries (cuBLAS, cuFFT, Thrust, cuDNN) in domains like image processing, scientific computing, and deep learning. *(Analyze/Create)*

## Books and E-Resources

**For Reference Print Book -**

4. D.B. Kirk, W. W. Hwu, 'Programming Massively Parallel Processors: A Hands-on Approach'; 4th Edition; Morgan Kaufmann;2022
5. J. Sanders, E Kandrot; 'CUDA by Example: An Introduction to General-Purpose GPU Programming'; 1st Edition; Addison-Wesley; 2010
6. Author initials, Author Surname (for Author 1), Author initials, Author Surname (for Author 2); 'Title of the Book'; Edition; Publisher; Year of Publication
   Example - R. E. Zeimer, W. H. Tranter; 'Principles of Communications: Systems, Modulation and Noise'; 7th Edition; Wiley; 2015

**For Reference Electronic Book –**

1. S Cook; 'CUDA Programming: A Developer's Guide to Parallel Computing with GPUs'; 1st Edition; Morgan Kaufmann; 2012
2. 'NVIDIA CUDA C++ Programming Guide; https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

**For MOOCs and other learning Resources**

1. D. Luebke; *Intro to Parallel Programming with CUDA*; Udacity; https://www.udacity.com/course/intro-to-parallel-programming--cs344;
   Accessed: Jun. 16, 2025
2. B. Catanzaro, M. Garland; *Intro to CUDA Programming*; NVIDIA Developer Zone; https://developer.nvidia.com/cuda-education-training;
   Accessed: Jun. 16, 2025
3. T. Mattson; *Heterogeneous Parallel Programming*; Coursera; https://www.coursera.org/learn/heterogeneous-parallel-programming;
   Accessed: Jun. 16, 2025

## Honors in Parallel Computing

## ABCXXX: Parallel Algorithms for GPUs

Teaching Scheme:

Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week;

Total Credits: 3

**Syllabus**

**Theory**

**Unit 1: Introduction to Parallel Algorithms and GPU Platforms (4 Hours)**

Need for parallel algorithms; Basic concepts: speedup, efficiency, scalability; Overview of GPU computing platforms (CUDA, O penCL); GPU architecture features relevant to algorithms – SIMT execution, thread blocks, memory hierarchy; GPU algorithm design considerations: massive parallelism, coalesced memory access, avoiding divergence.

**Unit 2: Fundamental Parallel Algorithms on GPU (5 Hours)**

Parallel prefix sum (scan); Reduction; Histogram computation; Parallel sorting: bitonic sort, radix sort (brief); Mapping sequential algorithms to parallel GPU kernels; Analyzing computational and memory complexity on GPU architectures.

**Unit 3: Matrix Computations on GPU (5 Hours)**

Parallel implementation of dense matrix operations: matrix addition, multiplication; Tiled matrix multiplication and shared memory optimization; Sparse matrix-vector multiplication (SpMV); Memory alignment and coalescing; Performance tuning using CUDA tools.

**Unit 4: Graph Algorithms on GPU (5 Hours)**

Graph representation for parallel processing (CSR, edge list); Parallel BFS and DFS; Parallel PageRank; Load balancing techniques for irregular parallelism; Optimization techniques for memory-bound graph problems.

**Unit 5: Scientific Computing Algorithms (5 Hours)**

Parallel numerical methods: Jacobi, Gauss-Seidel, and Conjugate Gradient methods; FFT on GPU using cuFFT and custom implementation; N-body simulation (brute-force and optimized); Use of CUDA libraries: cuBLAS, cuSPARSE, cuRAND.

**Unit 6: Case Studies and Performance Evaluation (4 Hours)**

Real-world case studies: image processing (convolution, filtering), machine learning (k-means); Benchmarking and profiling GPU algorithms using Nsight and Visual Profiler; Identifying performance bottlenecks; Guidelines for writing scalable GPU algorithms; Trends in GPU-based parallel algorithm research.

## Syllabus
## Laboratory

**List of Experiments**

1. Implement prefix sum using inclusive and exclusive scan on CUDA.
2. Perform parallel reduction with different tree-based approaches.
3. Implement and compare bitonic and radix sort on GPU.
4. Design tiled matrix multiplication using shared memory.
5. Implement Sparse Matrix-Vector multiplication using CSR format.
6. Perform histogram equalization on grayscale images using CUDA.
7. Simulate a simple N-body system with brute-force approach.
8. Compute FFT on signal data using cuFFT and compare with CPU implementation.
9. Implement parallel BFS for unweighted graphs on GPU.
10. Benchmark and analyze kernel performance using Nsight.
11. Optimize memory access patterns to avoid divergence.
12. Implement Conjugate Gradient method for linear system solving.

## Syllabus
## Course Projects

**Course Project Ideas**

1. GPU-Accelerated PageRank Algorithm
2. Parallel Implementation of 2D/3D Heat Diffusion using CUDA
3. Sparse Matrix Operations Library for CUDA
4. Fast Fourier Transform-based Signal Analyzer on GPU
5. GPU-Accelerated Genetic Algorithm for Optimization
6. Deep Learning Inference Engine for CNNs on GPU
7. Image Segmentation Using Graph Cuts on GPU
8. Parallel Simulation of Cellular Automata (e.g., Game of Life)
9. Hybrid CPU-GPU Solver for Partial Differential Equations
10. Parallel K-means with CUDA Thrust and Custom Kernels

## Course Outcomes

**Course Outcomes**: Students will be able to

1. **Understand** GPU hardware architecture and constraints that influence algorithm design and execution.
2. **Apply** parallel programming techniques to implement basic GPU algorithms such as scan, reduction, and sorting.
3. **Analyze** matrix and graph algorithms in terms of parallelism, memory usage, and performance on GPUs.
4. **Evaluate** the computational efficiency of different GPU implementations using profiling tools and metrics.
5. **Design** optimized and scalable GPU-based solutions for scientific, image processing, and graph-based applications.
6. **Create** real-world GPU algorithm implementations using CUDA libraries and performance-tuned kernels.

## Books and E-Resources

**For Reference Print Book –**

1. D.B. Kirk, W. W. Hwu; *Programming Massively Parallel Processors: A Hands-on Approach*; 4th Edition; Morgan Kaufmann; 2022
2. S. Cook; *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*; 1st Edition; Morgan Kaufmann; 2012
3. M. J. Quinn; *Parallel Programming in C with MPI and OpenMP*; 1st Edition; McGraw-Hill; 2003

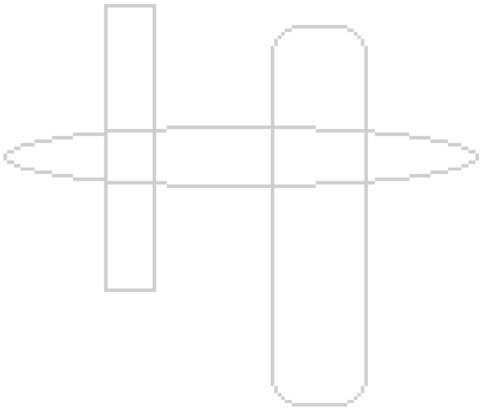
**For Reference Electronic Book –**

1. Jason Sanders, Edward Kandrot; CUDA by Example: An Introduction to General-Purpose GPU Programming; 1st Edition; Addison-Wesley; 2010
2. . NVIDIA Developer Resources; CUDA Toolkit Documentation; https://docs.nvidia.com/cuda/

**MOOCs and Learning Resources –**

1.  B. Catanzaro, M. Garland; *Intro to Parallel Programming with CUDA*; Udacity; https://www.udacity.com/course/intro-to-parallel-programming--cs344; Accessed: Jun. 16, 2025

2. V. Sarkar; *High Performance Scientific Computing*; Coursera; https://www.coursera.org/learn/hpc; Accessed: Jun. 16, 2025



## Honors in Parallel Computing

## ABCXXX: Parallel Programming Languages and Models I

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

**Syllabus**

**Theory**

**Unit 1: Introduction to Parallel Programming Paradigms (4 Hours)**

Need for parallelism and concurrency; Flynn's taxonomy of parallel architectures (SISD, SIMD, MISD, MIMD); Types of parallelism – task vs data parallelism; Shared vs distributed memory models; Overview of multicore, manycore, cluster and cloud environments; Goals and challenges in parallel programming – correctness, scalability, load balancing.

**Unit 2: Shared Memory Programming with OpenMP (5 Hours)**

Introduction to OpenMP: Purpose and advantages; Compilation and execution basics; OpenMP programming model; Parallel regions, work-sharing constructs (for, section, single), Loop parallelism and scheduling (static, dynamic, guided); Variable scoping: `private`, shared, firstprivate, lastprivate)

**Unit 3: Synchronization and Data Handling (5 Hours)**

Synchronization constructs: critical, atomic, barrier, flush, ordered; Data race conditions and prevention; Reduction operations; Performance considerations in synchronization; Parallelizing an existing code using OpenMP.

**Unit 4 : Advanced Parallelism Techniques (5 Hours)**

Nested parallel regions, Controlling number of threads; Affinity and thread placement basics; Tasking model: #pragma omp task, taskwait; nowait clause Default scoping and advanced data-sharing attributes

**Unit 5: Performance Tuning in OpenMP (5 Hours)**

Measuring speedup and scalability; profiling tools for OpenMP; Performance tuning : minimizing overhead, optimizing cache usage; Performance issues in OpenMP: false sharing, overheads; Load balancing and scheduling strategies; Code refactoring for better performance

**Unit 6 Case Studies and Applications (4 Hours)**
Case studies in scientific computing, image processing, and matrix operations; Integration with legacy codebases; Real-world applications using OpenMP; overview of hybrid models MPI+OpenMP

## Syllabus
## Laboratory

**List of Experiments**

1. **Vector Addition using OpenMP -**
   Write a program to add two vectors using #pragma omp parallel for.
2. **Matrix Multiplication** - Implement matrix multiplication using OpenMP and measure speedup over the sequential version.
3. **Sum of Large Array Elements using Reduction**- Use #pragma omp parallel for reduction(+:sum) to compute the sum of an array with millions of elements.
4. **Compare Static vs Dynamic Scheduling**
   Implement a loop with uneven workload per iteration and compare performance using schedule(static), schedule(dynamic).
5. **Parallel Prefix Sum (Inclusive Scan)**
   Implement a prefix sum algorithm using OpenMP for loop-based and tree-based approaches.
6. **Finding Maximum Element with Critical Section**
   Write an OpenMP program to find the maximum value in an array using critical and then reduction
7. **Image Convolution using OpenMP**
   Apply a 3×3 filter to a grayscale image using OpenMP parallel loops.
8. **Histogram Calculation with Synchronization**
   Generate a histogram of image intensity values with race condition management using atomic or critical.
9. **Monte Carlo Simulation for Estimating Pi**
   Implement Monte Carlo technique with OpenMP parallel region to estimate π.
10. **Implement Parallel Merge Sort using OpenMP Tasks**
    Use #pragma omp task and taskwait for a recursive merge sort implementation.

11. **False Sharing Demonstration and Fix**
    Write a program where false sharing degrades performance, and modify it to eliminate the problem using padding or proper memory layout.
12. **Stencil Computation (2D Heat Diffusion)**
    Implement iterative averaging over a 2D grid using OpenMP and shared memory optimization.
13. **Parallelize String Matching Algorithm**
    Use OpenMP to speed up pattern matching (e.g., naive string search) in a large text.
14. **Use of OpenMP Locks**
    Simulate a ticket counter system where multiple threads (customers) access shared counter using OpenMP locks.
15. **Measure Scalability using Varying Threads**
    Take any computation-intensive task (e.g., matrix multiplication or FFT) and run it with 1, 2, 4, 8... threads to plot scalability graph

**Syllabus**
**Course Project**

**List of Course Projects**

1. Parallel Matrix Multiplication with Performance Benchmarking .
2. OpenMP-Based Parallel Sorting Algorithms
3. Real-Time Image Processing Filters using OpenMP
4. Weather Simulation Using Stencil Computation
5. Monte Carlo Simulation for Risk Assessment or Pi Estimation
6. Parallel Game of Life
7. OpenMP-Based Web Log Analysis Tool
8. Parallel Prefix Sum and Histogram Equalization for Image Enhancement
9. OpenMP-Based Word Frequency Counter
10. Performance Analysis Toolkit for OpenMP Programs

## Course Outcomes

**Course Outcomes**: Students will be able to

1. Explain parallel programming paradigms, Flynn’s taxonomy, and memory models used in modern architectures. (Understand)
2. Apply OpenMP constructs to implement shared memory parallelism in C/C++ programs. (Apply)
3. Implement synchronization techniques and manage variable scoping to avoid race conditions and ensure program correctness. (Apply)
4. Analyze and utilize advanced OpenMP features like nested parallelism, tasking, and thread affinity to enhance program flexibility.( Analyze)
5. Evaluate and optimize the performance of OpenMP programs using profiling tools and tuning strategies.nd collective communication, synchronization, and data distribution. (Evaluate)
6. Develop real-world parallel applications and perform integration with existing codebases using OpenMP and hybrid models.(Create)

## Books and E-Resources

**For Reference Print Book -**

1. B. Chapman, G. Jost, R. van der Pas; *Using OpenMP: Portable Shared Memory Parallel Programming*; 1st Edition; MIT Press; 2008
2. M. J. Quinn; *Parallel Programming in C with MPI and OpenMP*; 1st Edition; McGraw-Hill; 2003

**For Reference Electronic Book –**

OpenMP Architecture Review Board; *OpenMP Specification and Resources*; OpenMP.org; https://www.openmp.org; Accessed: Jun. 16, 2025

**For MOOCs and other learning Resources**

V. Sarkar, R. Barik; *High Performance Scientific Computing*; Coursera; https://www.coursera.org/learn/hpc; Accessed: Jun. 16, 2025

# Honors in Parallel Computing

## ABCXXX: Parallel Programming Languages and Models II

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

**Syllabus**

**Theory**

**Unit 1: Introduction to MPI (4 Hours)**

Introduction to MPI: goals, features, and programming model; MPI program structure: mpi.h, mpicc, mpirun, compiling and executing; MPI environment functions: MPI_Init, MPI_Finalize, MPI_Comm_size, MPI_Comm_rank;Simple MPI programs: Hello world, rank-based messages

**Unit 2:  Point-to-Point Communication in MPI (5 Hours)**

Blocking vs non-blocking communication ;  MPI point-to-point functions: MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv; Use of MPI_Status, MPI_Request;  Message matching and buffer considerations;  Communication modes: standard, buffered, synchronous, ready;  Examples of ring communication, ping-pong, broadcast via point-to-point

**Unit 3: Collective Communication in MPI (5 Hours)**

Introduction to collective operations and their advantages; Broadcast (MPI_Bcast);Reduction operations (MPI_Reduce, MPI_Allreduce); Scatter and gather operations (MPI_Scatter, MPI_Gather, MPI_Allgather); Barrier synchronization (MPI_Barrier);  Performance implications of collective operations;  Examples: parallel sum, parallel histogram using collectives

**Unit 4 : Derived Data Types and Communicators (5 Hours)**

Motivation for derived data types in MPI; Creating derived data types: MPI_Type_contiguous, MPI_Type_vector, MPI_Type_struct ; Use of MPI_Type_commit and MPI_Type_free;  Communicators and groups: MPI_COMM_WORLD, MPI_Comm_split, MPI_Comm_create; Use cases for custom communicators (e.g., subgrouping processes)

**Unit 5: Process Topologies and Synchronization (5  Hours)**

Virtual topologies: Cartesian and graph topologies; Creating topologies with MPI_Cart_create, MPI_Graph_create; Topology utilities: MPI_Cart_coords, MPI_Cart_shift; .Synchronization methods in MPI: barriers, fences, ordering of communication; Deadlock and how to avoid it in MPI; Debugging MPI programs (tools and techniques)

**Unit 6: Performance, File I/O, and Case Studies (4 Hours)**

Performance considerations: latency, bandwidth, communication/computation overlap; Load balancing and granularity in MPI; Introduction to MPI I/O: MPI_File_open, MPI_File_read, MPI_File_write;Parallel I/O design considerations; Case studies:Parallel matrix-vector multiplication,Image processing with MPI,Domain decomposition methods in MPI; Summary of best practices and MPI limitations

## Syllabus
## Laboratory

**List of Experiments**

1. **Hello World MPI Program -** Write an MPI program where each process prints its rank and total number of processes.
2. **Point-to-Point Communication (Ping-Pong) -**Implement a ping-pong message exchange between two MPI processes using MPI_Send and MPI_Recv.
3. **Ring Communication -**Design a program where each process sends a message to the next process in a ring fashion.
4. **MPI Broadcast -**Implement MPI_Bcast to broadcast a message or an array from the root process to all others.
5. **MPI Reduction (Sum of Array Elements) -**Parallelize an array sum operation using MPI_Reduce.
6. **MPI Scatter and Gather -**Use MPI_Scatter to distribute array chunks and MPI_Gather to collect processed results.
7. **Parallel Matrix-Vector Multiplication-**Implement matrix-vector multiplication where rows are distributed among processes.
8. **Parallel Prefix Sum (Scan) -**Write an MPI program to compute prefix sum (inclusive/exclusive) using custom logic or MPI_Scan.
9. **MPI Barrier Synchronization-** Demonstrate the use of MPI_Barrier to synchronize processes before and after a critical section.
10. **File I/O using MPI -**Use MPI_File_open, MPI_File_read, and MPI_File_write to read/write a file in parallel.

11. **Deadlock Scenario and Resolution-** Create a scenario that leads to deadlock using blocking sends/receives and modify it to prevent the deadlock.
12. **MPI Cartesian Topology -** Set up a 2D Cartesian topology and allow each process to communicate with its neighbors.
13. **Custom Derived Datatypes -**Create a derived MPI datatype using MPI_Type_vector and use it for structured communication.
14. **Monte Carlo π Estimation in Parallel -**Estimate the value of π using a parallel Monte Carlo simulation with reduction.
15. **Parallel Sorting using MPI (Odd-Even Transposition or Bucket Sort) -**Implement a parallel sorting algorithm using multiple MPI processes.

## Syllabus
## Course Project

**List of Course Projects**

1. Parallel Matrix Multiplication (2D Decomposition)
2. Parallel PageRank Algorithm using MPI
3. MPI-based Parallel File Compression Tool
4. Parallel Image Convolution/Filtering
5. MPI-based Distributed Game of Life Simulation
6. Weather Data Analysis using MPI
7. Parallel N-body Simulation using MPI
8. Parallel File Compression Utility
9. MPI-based Parallel Web Log Analyzer
10. Parallel image processing or sorting using MPI.
11. Parallel Sudoku Solver with Work Sharing

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Explain the goals, programming model, and essential components of MPI. (Understand)
2. Implement point-to-point and non-blocking communication techniques in MPI programs using standard and advanced communication modes. (Apply)

3. Develop parallel applications using collective communication functions such as broadcast, reduction, scatter, and gather in MPI. (Apply)
4. Construct MPI programs utilizing derived datatypes and custom communicators for complex structured data and subgroup processing. (Apply)
5. Analyze MPI process topologies, synchronization techniques, and deadlock scenarios to enhance communication efficiency and correctness. (Analyze)
6. Design and evaluate the performance of MPI-based parallel applications involving file I/O, load balancing, and real-world case studies. (Evaluate)

## Books and E-Resources

**For Reference Print Book -**

1. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra; *MPI: The Complete Reference, Volume 1 – The MPI Core*; 2nd Edition; MIT Press; 1998
2. M. J. Quinn; *Parallel Programming in C with MPI and OpenMP*; 1st Edition; McGraw-Hill; 2003

**For Reference Electronic Book –**

MPI Forum; *MPI Standard and Documentation*; mpi-forum.org; https://www.mpi-forum.org; Accessed: Jun. 16, 2025

**For MOOCs and other learning Resources**

V. Sarkar, R. Barik; *High Performance Scientific Computing*; Coursera; https://www.coursera.org/learn/hpc; Accessed: Jun. 16, 2025

# Honors in Parallel Computing

## ABCXXX: High-Performance Computing

Teaching Scheme:
Theory: 2 Hours / Week; Laboratory: 2 Hours / Week;
Total Credits: 3

**Syllabus**
**Theory**

**Unit 1: Introduction to High-Performance Computing (4 Hours)**
Evolution and importance of HPC; Applications in science, engineering, and AI; Basics of computer architecture for HPC: memory hierarchy, pipelining, instruction-level parallelism; Classification of parallel systems – Flynn's taxonomy (SISD, SIMD, MIMD, MISD); Parallel vs distributed computing; Performance metrics: speedup, efficiency, Amdahl's law, Gustafson's law.

**Unit 2: Parallel Programming Models (5 Hours)**
Overview of shared and distributed memory models; Message Passing Interface (MPI); Shared memory programming with OpenMP; GPU computing overview with CUDA; Hybrid programming models; Execution models, synchronization, communication costs; Scalability and load balancing.

**Unit 3: Architecture of HPC Systems (5 Hours)**
Multicore CPUs and manycore GPUs; Vector processors; SMP, clusters, and distributed systems; Network topologies (ring, mesh, torus, hypercube); Interconnects and bandwidth; Cache coherence, NUMA systems; Trends in supercomputing architectures.

**Unit 4: Parallel Algorithms and Patterns (5 Hours)**
Decomposition strategies – domain, functional, data-parallel; Parallel algorithm design principles; Common computation patterns – map, reduce, scan, stencil; Parallel algorithms: matrix multiplication, sorting, FFT, prefix sum; Performance analysis and bottlenecks.

**Unit 5: Performance Optimization and Tools (5 Hours)**
Profiling and benchmarking tools (gprof, perf, MPI tools); Performance tuning: memory access patterns, communication reduction, vectorization, thread affinity; Power and thermal considerations; Roofline model; Debugging parallel applications.

**Unit 6: Case Studies and Emerging Trends in HPC (4 Hours)**
HPC applications in climate modeling, genomics, finance, AI, astrophysics; Exascale computing; Green computing in HPC; Cloud-based HPC; Quantum computing overview; National and international supercomputers; Software stacks and libraries for HPC.

## Syllabus
## Laboratory

**List of Experiments**

1. Analyze system architecture using hardware info tools (e.g., lscpu, hwloc).
2. Parallelize numerical integration using OpenMP.
3. Parallel matrix multiplication with OpenMP.
4. MPI program for computing sum across processes.
5. MPI implementation of matrix-vector multiplication.
6. Use MPI collectives (Broadcast, Reduce, Scatter, Gather).
7. Use hybrid MPI+OpenMP for vector operations.
8. Use nvprof or Nsight to analyze CUDA programs.
9. Apply loop unrolling and vectorization techniques in C/C++.
10. Profile and optimize OpenMP code using perf and gprof.
11. Implement a parallel merge sort using OpenMP/MPI.
12. Performance comparison of serial, OpenMP, and MPI versions of a program.

## Syllabus
## Course Projects

**Course Project Ideas**

1. Performance analysis of scientific kernels on multicore CPU and GPU.
2. HPC-based weather forecasting model using OpenMP/MPI.
3. Parallel Monte Carlo simulation for risk estimation.
4. Distributed database query engine using MPI.
5. GPU-based fluid dynamics simulation.
6. Parallel recommendation system using matrix factorization.
7. Genetic algorithm implementation using OpenMP.
8. Parallel BFS or PageRank on a graph dataset.

9. Molecular dynamics or protein folding simulation.
10. Hybrid solution for big-data analytics (MPI + Hadoop/Spark).

## Course Outcomes

**Course Outcomes**: Sstudents will be able to:

1. **Describe** the architecture and characteristics of modern HPC systems. *(Understand)*
2. **Apply** parallel programming models (OpenMP, MPI, CUDA) to design scalable applications. *(Apply)*
3. **Analyze** parallel algorithms and identify performance bottlenecks. *(Analyze)*
4. **Evaluate** the efficiency and scalability of parallel applications using standard profiling tools. *(Evaluate)*
5. **Optimize** HPC applications through memory, computation, and communication tuning. *(Apply/Evaluate)*
6. **Implement** real-world scientific and engineering applications on HPC platforms. *(Create)*

## Books and E-Resources

**For Reference Print Books**

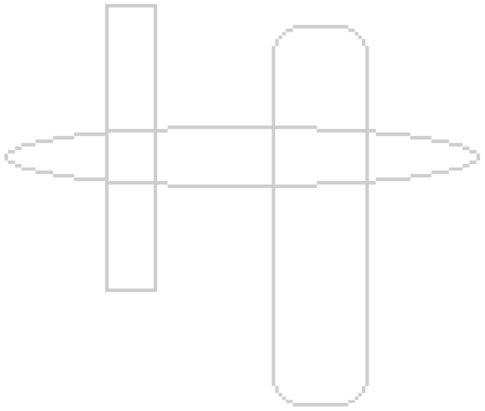
1. G. E. Karniadakis, R. M. Kirby; *Parallel Scientific Computing in C++ and MPI*; 1st Edition; Cambridge University Press; 2003
2. M. J. Quinn; *Parallel Programming in C with MPI and OpenMP*; 1st Edition; McGraw-Hill; 2003
3. D. B. Kirk, W. W. Hwu; *Programming Massively Parallel Processors: A Hands-on Approach*; 4th Edition; Morgan Kaufmann; 2022
4. B. Chapman, G. Jost, R. van der Pas; *Using OpenMP: Portable Shared Memory Parallel Programming*; 1st Edition; MIT Press; 2008

**For Reference Electronic Resources**

1. MPI Forum; *MPI Standard Documentation*; https://www.mpi-forum.org; Accessed: Jun. 16, 2025
2. OpenMP Architecture Review Board; https://www.openmp.org; Accessed: Jun. 16, 2025
3. NVIDIA CUDA Toolkit Documentation; https://docs.nvidia.com/cuda; Accessed: Jun. 16, 2025

**For MOOCs and Learning Platforms**

1. V. Sarkar, R. Barik; *High Performance Scientific Computing*; Coursera; https://www.coursera.org/learn/hpc; Accessed: Jun. 16, 2025
2. *Parallel Programming with MPI*; Udacity; https://www.udacity.com/course/parallel-programming--cs344; Accessed: Jun. 16, 2025



## Honors in Parallel Computing

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**
This course addresses the issues associated with the successful management of a   project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**
The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

1. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
2. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.
3. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.

4. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
5. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
6. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).

7. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
8. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate's performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.

## Honor Course: Cyber Security

| Course No. | Semester | Course Name | Teaching Scheme (Hours/Week) | | | Examination Scheme | | | | Credits |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Th | Lab | Tut | | | | Total | |
| C1 | III | Cryptography | 2 | 2 | 0 | | | | | 3 |
| C2 | IV | Networks and Systems Security | 2 | 2 | 0 | | | | | 3 |
| C3 | V | Application Security | 2 | 2 | 0 | | | | | 3 |
| C4 | VI | Cloud Security and Cyber Forensic | 2 | 2 | 0 | | | | | 3 |
| C5 | VII | Blockchain | 2 | 2 | 1 | | | | | 4 |
| C6 | VIII | Project | 0 | 8 | 0 | | | | | 4 |

## Honors in Cyber Security

## ABCXXX: Cryptography

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

### Syllabus
### Theory

**Unit 1: (Title)  -** **(4 Hours)**

Key Security Properties: Confidentiality, Integrity, Availability, Non-repudiation
Introduction: What is modern cryptography, Historical ciphers and their cryptanalysis, The heuristic versus the rigorous approach; adversarial models and principles of defining security. Classical Cryptosystem, Block Cipher. Motivation for the feistel Cipher structure, the feistel Cipher, Simplified Data Encryption standard. Data Encryption Standard (DES), Triple DES, Modes of Operation, Stream Cipher.

**Unit 2:** **( 5 Hours)**

LFSR based Stream Cipher, Mathematical background, Abstract algebra, Number Theory.  Modular arithmetic, Modular Inverse, Extended Euclid Algorithm, Fermat's Little Theorem, Euler Phi-Function, Euler's theorem.  Simplified Advanced Encryption Standard (AES), Advanced Encryption Standard (AES).

**Unit 3: (Title)** **(6 Hours)**

Algebraic Structure, Introduction to Public Key Cryptosystem, Diffie-Hellman Key Exchange, Knapsack Cryptosystem, RSA Cryptosystem. Primarily Testing, ElGamal Cryptosystem, Elliptic curve arithmetic, abelian groups,  elliptic curves over Zp, elliptic curves over GF(2m), Elliptic curve cryptography, Elliptic Curve over the Reals, Elliptic curve Modulo a Prime.

**Unit 4:** **( 5 Hours)**

Generalized ElGamal Public Key Cryptosystem, Rabin Cryptosystem. Cryptanalysis, Time-Memory Trade-off Attack, Differential and Linear Cryptanalysis.

**Unit 5:** **( 5 Hours)**

Symmetric key distribution using Symmetric encryption, A key distribution scenario, Hierarchical key control, session key lifetime, a transparent key control scheme, Decentralized key control, controlling key usage, Symmetric key distribution using asymmetric encryption, simple secret key distribution, secret key distribution with confidentiality and authentication, A hybrid scheme, distribution of public keys, public announcement of public keys, publicly available directory, public key authority, public keys certificates.

**Unit 6: (5 Hours)**
Cryptanalysis on Stream Cipher, Modern Stream Ciphers, Shamir's secret sharing and BE, Identity-based Encryption (IBE), Attribute-based Encryption (ABE). Quantum Cryptography: Quantum key distribution (QKD), Post-quantum cryptography

## Syllabus
## Laboratory

**List of Experiments**

1. Implementation of Hill Cipher
2. Implementation of Simplified DES
3. Implementation of Simplified AES
4. Implementation of RSA algorithm
5. Implementation of ECC algorithm
6. Implementation of Diffie -Hellman Key exchange Algorithm
7. Implementation of ElGamal Cryptosystem
8. Implementation of Knapsack Cryptosystem
9. Study of Quantum Cryptography

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Explain the fundamental concepts and goals of cryptography, including confidentiality, integrity, authentication, and non-repudiation.[Knowledge Level: Understand]
2. Analyze and compare classical encryption techniques and evaluate their strengths and vulnerabilities.[Knowledge Level: Analyze]
3. Apply symmetric and asymmetric cryptographic algorithms (e.g., AES, RSA) to secure communication systems.[Knowledge Level: Apply]
4. Demonstrate the use of cryptographic algorithms in ensuring data integrity and authentication.[Knowledge Level: Apply]
5. Evaluate key management techniques and secure protocol mechanisms used in real-world systems.[Knowledge Level: Evaluate]

6. Design and implement secure solutions using cryptographic techniques for modern applications.[Knowledge Level: Create]

## Books and E-Resources

**For Reference Print Book -**

1. William Stallings, *Cryptography and Network Security: Principles and Practice*, 7th Edition, Pearson Education, 2017.
2. Behrouz A. Forouzan, *Cryptography and Network Security,* 1st Edition, McGraw Hill Education, 2007.
3. Bernard Menezes, "Network Security and Cryptography", 1st Edition, Cengage Learning, 2010, ISBN 81-315-1349-1.
4. V.K. Pachghare, "Cryptography and Information Security", 2nd Edition, PHI, 2015, ISBN-978-81-203-5082-3.
5. Atul Kahate, Cryptography and Network Security, 3rd Edition, McGraw Hill Education, 2018.

**For Reference Electronic Book –**

1. **Alfred J. Menezes, Paul C. van Oorschot & Scott A. Vanstone**, *Handbook of Applied Cryptography*, 1st ed., CRC Press, 1996
2. **Jonathan Katz & Yehuda Lindell**, *Introduction to Modern Cryptography*, 2nd ed., CRC Press, 2015 (e-book PDF ISBN 978-1-4665-7027-6)

**For MOOCs and other learning Resources**

Prof. Sourav Mukhopadhyay; 'Cryptography And Network Security'; Swayam NPTEL; URL – https://onlinecourses.nptel.ac.in/noc22_cs90/preview; Accessed – 25 Jul 2022

# Honors in Cyber Security

## ABCXXX: Networks and Systems Security

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**
**Theory**

**Unit 1: (Title) (4 Hours)**

Principles of network security, Security services and mechanisms, Threats and vulnerabilities, OSI Security Architecture, Security policies and attack types,
Certificates and Hashing: Properties of hash functions, HASH + SALT, hashing algorithms (SHA1, SHA2).
Authentication and Authorization: Access control models: DAC, MAC, RBAC, Network access control, SHA-512, Kerberos, multifactor authentication.
Secure login mechanisms, Password storage best practices (hashing, salting), Session management techniques, Attacks: session hijacking, fixation, brute force, password spraying,

**Unit 2:: (Title) ( 5 Hours)**

Transport-Level Security: Web security considerations, Secure Sockets Layer (SSL), Transport Layer Security (TLS), HTTPS standard, Secure Shell (SSH) application, Secure IP protocols: IPSec (AH & ESP), VPNs, Email security: PGP and S/MIME, DNS Security (DNSSEC)

**Unit 3: (Title) (6 Hours)**

Application Security: Security by design, writing secure code, static and dynamic application security testing (SAST and DAST), interactive application security testing (IAST), Integrated Security in DevOps, Application Security Services, Digital signatures and certificates, Firewalls and Intrusion Detection Systems , Secure Network Architecture, Secure Network Design, Intrusion Detection and Prevention, Host-based and Network-based IDS, Signature vs Anomaly-based detection. Secure Software Development Life Cycle (SDLC). Firewall types: Packet-filtering, Stateful, Proxy, NGFW, Firewall architectures and design

**Unit 4: Mobile Security ( 5 Hours)**

Introduction to Mobile Security, Mobile ecosystem overview (devices, OSs, apps, stores), Mobile threat landscape, Attack surface of mobile devices, Security models of Android and iOS, Android architecture and security components, iOS architecture and sandboxing, Secure boot, encryption, and keychain/key store, Device-level security (rooting/jailbreaking threats), Securing mobile app APIs, OAuth 2.0 and mobile API authentication, Using Mobile App Security Verification Standards (MASVS)

**Unit 5: ( 5 Hours)**
IEEE 802.11 Security (WEP, WPA, WPA2, WPA3), Threats in wireless networks (eavesdropping, spoofing), Bluetooth and mobile device security, Secure configuration of access points, Wireless Security: Modes and Protocols for wireless communication, WEP, WPA 2, Hacking Wireless signals, Satellite communication, GSM communication
Fundamentals of Pen-testing: Web, IoT, Mobile

**Unit 6: ( 5 Hours)**
**IoT Security:** Definitions of OT, IoT, IIoT, & ICS), Convergence of OT/IoT and IT domains. Functional difference between security measures for IT and OT/IoT, Introduction to most widely used protocols in IoT environment - MQTT and CoAP, KNX, BACnet, BLE, LoRa, ZigBee, SNMP , Embedded/ Hardware device security, Attack vectors and attack surfaces for IoT devices, Layered and Dynamic security measures for IoT Industrial Ecosystem, Introduction to IoT security Standards and its importance, Side Channel Attacks
**Cyber Physical System Attacks and Social Engineering**

## Syllabus
## Laboratory

**List of Experiments will be conducted using tools given below**

1. Implementation of SHA-1/SHA-2
2. MobSF, Drozer, Frida, Burp Suite, APKTool,
3. Android Studio, Xcode
4. Wireshark, Genymotion, Charles Proxy
5. Scanning and Enumeration- tools-Nmap/ Netcat
   a. Conduct port scanning and service detection
   b. Use Nmap scripts to find vulnerabilities
   c. Use Netcat for banner grabbing
6. Firewall Configuration and Testing-tools used pfSense, iptables
   a. Set up basic firewall rules
   b. Block/allow specific ports and protocols

   c. Test rule effectiveness using Nmap

7. Intrusion Detection System (IDS)
   a. Install and configure Snort
   b. Create and test custom rules
   c. Analyze alerts for suspicious traffic
8. Web Application Security Basics-identify web Vulnerability (tools- OWASP Juice Shop / DVWA)
   a. Perform SQL Injection and XSS attacks
   b. Capture cookies and perform session hijacking
   c. Suggest defenses against these attacks

## Course Outcomes

1. Use appropriate methods of authentication and access control **[Level: Apply]**
2. Identify security threats and vulnerabilities in mobile applications and mobile platforms.**[Level: Analyze]**
3. Illustrate the architecture and communication protocols used in Application layers of the network model and their associated security challenges. **[Level: Understand]**
4. Demonstrate the use of network security tools such as firewalls, IDS/IPS, vulnerability scanners, and packet analyzers.**[Bloom's Level: Apply]**
5. Evaluate security policies and mechanisms in real-world network environments, including VPNs, wireless, and Mobile and IoT Security **[Bloom's Level: Evaluate]**
6. Design a secure network architecture and recommend security measures for organizational networks. **[Bloom's Level: Create]**

## Books and E-Resources

**For Reference Print Book -**

1. William Stallings; Network Security Essentials: Applications and Standards; 6th Edition; Pearson Education; 2023.
2. Behrouz A. Forouzan; Cryptography and Network Security; 1st Edition; McGraw-Hill Education; 2007.
3. Atul Kahate; Cryptography and Network Security; 3rd Edition; McGraw-Hill Education; 2017.
4. Charlie Kaufman, Radia Perlman, Mike Speciner; Network Security: Private Communication in a Public World; 2nd Edition; Prentice Hall; 2002.

5. Mark Stamp; Information Security: Principles and Practice; 2nd Edition; Wiley; 2011.
6. Nina Godbole, Sunit Belapure; Cyber Security: Understanding Cyber Crimes, Computer Forensics and Legal Perspectives; 1st Edition; Wiley India; 2011.
7. Eric Maiwald; Network Security: A Beginner's Guide; 3rd Edition; McGraw-Hill Education; 2012.

**For Reference Electronic Book –**

1. William Stallings; Cryptography and Network Security: Principles and Practice; 8th Edition; Pearson Education; 2023.
2. Charlie Kaufman, Radia Perlman, Mike Speciner; Network Security: Private Communication in a Public World; 2nd Edition; Pearson Education; 2002.
3. Mark Stamp; Information Security: Principles and Practice; 2nd Edition; Wiley; 2011.
4. Bruce Schneier; Applied Cryptography: Protocols, Algorithms, and Source Code in C; 2nd Edition; Wiley; 2015.
5. Michael T. Goodrich, Roberto Tamassia; Introduction to Computer Security; 1st Edition; Pearson Education; 2011.

**For MOOCs and other learning Resources**

7. Cybersecurity: Network Security- University of Colorado System- https://www.coursera.org/learn/network-security
8. Network Security- Rochester Institute of Technology (RIT)- https://www.edx.org/course/network-security
9. Network Security-IIT Madras- NPTEL / SWAYAM- https://swayam.gov.in/nd1_noc20_cs56

## Honors in Cyber Security

## ABCXXX: Application Security

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**
**Theory**

**Unit 1: (Title) (6 Hours)**
Introduction to Security: Vulnerabilities, Threats, Threat Modeling, Risk, attack and attack types, countermeasures - Avoiding attacks, Security services. Protocol Vulnerabilities: DoS and DDoS, session hijacking, ARP spoofing, Software vulnerabilities: Phishing, buffer overflow, Cross-site scripting attack, ransomware, SYN-Flooding, SQL- injection, DNS poisoning, Sniffing.

**Unit 2: ( 5 Hours)**
Threats, vulnerabilities, and risks, Vulnerability lifecycle and CVSS (Common Vulnerability Scoring System), Common Vulnerabilities and Exposures (CVE) database, Attack surface and attack vectors, Case studies of major cyberattacks due to known vulnerabilities
Manual vs automated discovery, Static and dynamic analysis, Fuzz testing, Code review and debugging for vulnerability detection, format string attack.
Exploit development fundamentals, Reverse engineering basics, Metasploit framework and usage, Privilege escalation techniques, Post-exploitation analysis.

**Unit 3: (Title) 6 Hours)**
Patch management strategies, Hardening systems and software, Secure coding practices, IDS/IPS role in vulnerability management, Security Information and Event Management (SIEM), Responsible disclosure and bug bounty programs, Legal implications of vulnerability analysis , Standards and frameworks: OWASP Top 10, NIST, ISO/IEC 27001, Ethics in security testing

**Unit 4: ( 5 Hours)**
Web architecture overview (client-server model, HTTP/S, cookies, sessions), Threat landscape in web applications, Review of Cross site scripting attack, Command injection, LDAP injection, XPath injection, Parameterized queries and ORM protections, Secure input handling.
**Cross-Site Request Forgery (CSRF):** Understanding CSRF attacks and preconditions, Exploitation techniques, Token-based mitigation (anti-CSRF tokens), SameSite cookie

attribute. Risks of unrestricted file uploads, MIME type and file extension spoofing, Horizontal and vertical privilege escalation

**Unit 5: ( 5 Hours)**
Web Server and API Security, Common server misconfigurations, HTTPS, TLS/SSL, and certificate management, REST API and JSON security concerns, API rate limiting, input validation, Authentication protocols: OAuth 2.0, JWT, Content Security Policy (CSP), Secure cookie attributes, CORS policies, Secure deployment and DevSecOps,

**Unit 6: ( 5 Hours)**
**Incident Response and Compliance**
Application-level logging and monitoring, Detection and response strategies, Regulatory compliance (GDPR, HIPAA, PCI-DSS), Vulnerability disclosure and patching practices, Bug bounty programs and responsible disclosure,

## Syllabus
## Laboratory

**List of Experiments and tools**

1. Network scanning: Nmap, Nessus, OpenVAS
2. Perform vulnerability scanning on sample web apps
3. Fix OWASP Top 10 vulnerabilities in code
4. Write secure login and session management code Bandit, Brakeman (for Python and Ruby apps)
5. Configure DevSecOps pipeline with GitHub/GitLab and scanners (GitHub Actions/GitLab CI for DevSecOps)
6. Web application scanning: Burp Suite, Nikto, Acunetix
7. Source code analysis: SonarQube, Bandit
8. OS & patch scanning: Microsoft Baseline Security Analyzer (MBSA), Lynis
9. sqlmap, Microsoft Threat Modeling Tool,
10. OWASP ZAP, OWASP Dependency-Check
11. w3af, SonarQube, Checkmarx, Fortify
12. Browser developer tools

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Explain the fundamental principles of application security and the need for secure software development practices.**[Level: Understand]**
2. Identify and analyze common application vulnerabilities such as those listed in OWASP Top 10 and CWE/SANS Top 25.**[Level: Analyze]**
3. Apply secure coding practices to mitigate risks associated with authentication, session management, input validation, and error handling.**[Level: Apply]**
4. Perform static and dynamic security testing using appropriate tools to detect vulnerabilities in web and mobile applications.**[Level: Apply]**
5. Integrate security into DevOps pipelines and adopt Secure DevOps (DevSecOps) practices in software development environments. **[Level: Evaluate]**
6. Design and implement secure application components that comply with relevant security standards, regulations, and threat models.**[Level: Create]**

## Books and E-Resources

**For Reference Print Book -**

1. Mark Curphey, Window Snyder; Threat Modeling: Designing for Security; 1st Edition; Wiley; 2014.
2. Andrew Hoffman; Web Application Security: Exploitation and Countermeasures for Modern Web Applications; 1st Edition; O’Reilly Media; 2020.
3. Tanya Janca; Alice and Bob Learn Application Security; 1st Edition; Wiley; 2020.
4. Bryan Sullivan, Vincent Liu; Web Application Security: A Beginner's Guide; 1st Edition; McGraw-Hill Education; 2011.
5. Joel Scambray, Vincent Liu, Caleb Sima; Hacking Exposed Web Applications; 3rd Edition; McGraw-Hill Education; 2010.
6. Michael Cross; Developer’s Guide to Web Application Security; 1st Edition; Syngress Publishing; 2011.
7. Jason Andress, Ryan Linn; Coding for Penetration Testers: Building Better Tools; 2nd Edition; Syngress; 2016.

**For Reference Electronic Book –**

1. Mark Curphey, Window Snyder; Threat Modeling: Designing for Security; 1st Edition; Wiley; 2014.
2. Andrew Hoffman; Web Application Security: Exploitation and Countermeasures for JavaScript Apps; 1st Edition; O'Reilly Media; 2020.
3. Tanya Janca; Alice and Bob Learn Application Security; 1st Edition; Wiley; 2020.
4. Michael Cross; Practical Web Penetration Testing; 1st Edition; Packt Publishing; 2021.
5. Bryan Sullivan, Vincent Liu; Web Application Security: A Beginner's Guide; 1st Edition; McGraw-Hill Education; 2011.
6. Joel Scambray, Vincent Liu, Caleb Sima; Hacking Exposed Web Applications: Web Application Security Secrets and Solutions; 3rd Edition; McGraw-Hill Education; 2010.
7. Jason Andress, Ryan Linn; Coding for Penetration Testers: Building Better Tools; 2nd Edition; Syngress/Elsevier; 2016.
8. OWASP Foundation; OWASP Secure Coding Practices – Quick Reference Guide; Latest Edition; OWASP; [Online Resource, Updated Regularly].

**For MOOCs and other learning Resources**

10. EC Council , Application Security – The Complete Guide- https://www.edx.org/course/application-security-the-complete-guide
11. Sunil Gupta, Web Application Security for Absolute Beginners, - https://www.udemy.com/course/web-application-security-for-absolute-beginners/
12.

## Honors in Cyber Security

## ABCXXX: d.  Cloud Security and Cyber Forensic

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week
Total Credits: 3

**Syllabus**

**Theory**

**Unit 1: (Title) (4 Hours)**

Introduction to Cloud Computing and Security Concepts, Overview of cloud computing: IaaS, PaaS, SaaS, Cloud security basics: confidentiality, integrity, availability (CIA), Cloud-specific attacks: data breaches, DoS, account hijacking, Top threats (CSA Top Threats to Cloud Computing), Multi-tenancy and virtualization risks, Insider threats and insecure APIs

Identity and Access Management (IAM): IAM principles: least privilege, RBAC, ABAC, Federated identity and SSO, MFA (Multi-Factor Authentication), Cloud-native IAM services (e.g., AWS IAM, Azure Active Directory), Identity federation protocols (SAML, OAuth 2.0, OpenID Connect),

**Unit 2: ( 5 Hours)**

Data Security and Encryption in the Cloud: Data lifecycle in cloud: in-transit, at-rest, in-use, Encryption techniques and key management, Cloud provider encryption services (AWS KMS, Azure Key Vault), Data loss prevention (DLP), Tokenization and anonymization,

Secure coding and DevSecOps in cloud environments, API security best practices, Container and Kubernetes security, CI/CD pipeline security, Serverless application security, Network Logging and monitoring (e.g., AWS CloudTrail, Azure Monitor)

**Unit 3: (Title) (6 Hours)**

Cloud governance and risk management, Compliance standards: ISO 27001, GDPR, HIPAA, PCI-DSS, Data sovereignty and jurisdiction issues, Cloud audit and assessment frameworks (CCM, STAR, FedRAMP), Vendor risk management,

Security Operations Center (SOC) in cloud, Cloud-native security monitoring and SIEM tools, Incident response plan and forensic investigation, Cloud automation and remediation, Business continuity and disaster recovery in cloud,

**Unit 4: ( 5 Hours)**

Introduction to Cyber Forensics: Definition and scope of cyber/digital forensics, Types of digital evidence, Cybercrime categories: financial fraud, data theft, cyberterrorism, identity theft, Forensics vs incident response, Legal considerations and cyber laws (IT Act, GDPR basics), ules of evidence and admissibility in court, Chain of custody and documentation, Phases of digital forensics: Identification, Preservation, Collection, Examination, Analysis, Presentation,  FAT, NTFS, EXT file systems, File metadata, timestamps, and carving deleted files, Windows artifacts: Registry, Prefetch, Event logs, Recycle Bin, Linux forensic artifacts and shell history, Timeline analysis

**Unit 5: ( 5 Hours)**

Disk imaging and verification (bit-by-bit copies), Tools: FTK Imager, dd, Autopsy, Volatile memory acquisition (RAM dump), Memory analysis tools: Volatility Framework, Extracting passwords, process lists, injected code,
Browser history, cookies, cache, downloads, Cloud-based email analysis (Gmail, Outlook), Email forensic tools (e.g., MailXaminer, Xplico),

**Unit 6: ( 5 Hours)**

**Mobile Device Forensics:** Mobile OS file systems (Android/iOS), SIM card, memory card, and app data analysis, Tools: Cellebrite, MOBILedit, Oxygen Forensic Suite, Extracting SMS, contacts, call logs, GPS data, Legal implications and consent for mobile forensics,
Structuring a forensic investigation report, Writing clear, precise, and legally valid reports, Role of expert witness in court, Ethics and responsibilities of a forensic investigator,

## Syllabus
## Laboratory

**List of Experiments**

1. Configure IAM roles and policies in AWS/Azure
2. Encrypt and decrypt cloud data using KMS
3. Analyze traffic using VPC flow logs
4. Secure a cloud-hosted web app using WAF
5. Simulate a cloud data breach and perform incident response
6. Use open-source tools like CloudSploit, ScoutSuite for auditing
7. Create and verify forensic disk images
8. Recover deleted files and hidden data
9. Perform memory analysis to identify malware

10. Analyze browser artifacts and email headers
11. Capture and investigate network traffic for anomalies
12. Mobile data extraction and forensic reporting

## Course Outcomes

1. Illustrate the principles, models, and architecture of cloud computing and assess the security challenges associated with various cloud deployment models. **[Level: Understand]**
2. Analyze cloud-specific threats, vulnerabilities, and risk mitigation strategies including access control, encryption, and network security. **[Level: Analyze]**
3. Apply cloud security tools and best practices to design and implement secure cloud applications and infrastructure. **[Level: Apply]**
4. Explain the principles, procedures, and legal aspects involved in digital evidence handling and cybercrime investigation. **[Level: Understand]**
5. Analyze digital evidence from computers, mobile devices, and networks using forensic tools and techniques. **[Level: Analyze]**
6. Apply appropriate forensic methods to collect, examine, and present digital evidence in a legally admissible manner. **[Level: Apply]**

## Books and E-Resources

**For Reference Print Book -**

1. Tim Mather, Subra Kumaraswamy, Shahed Latif; Cloud Security and Privacy; 1st Edition; O'Reilly Media; 2009.
2. Vic (J.R.) Winkler; Securing the Cloud: Cloud Computer Security Techniques and Tactics; 1st Edition; Syngress; 2011.
3. Yuri Diogenes, Tom Shinder; Microsoft Azure Security Infrastructure; 1st Edition; Microsoft Press; 2016.
4. Ben Potter, Scott Ward; AWS Security Best Practices; AWS Whitepaper; Latest Edition.
5. Cloud Security Alliance (CSA); Security Guidance for Critical Areas of Focus in Cloud Computing v4.0; CSA; 2017.
6. NIST Special Publication 800-144; Guidelines on Security and Privacy in Public Cloud Computing; NIST; 2011.
7. Nelson, Bill; Phillips, Amelia; Steuart, Christopher; Guide to Computer Forensics and Investigations; 6th Edition; Cengage Learning; 2018.

8. Marjie T. Britz; Computer Forensics and Cyber Crime: An Introduction; 3rd Edition; Pearson Education; 2013.

**For Reference Electronic Book –**

1. Tim Mather, Subra Kumaraswamy, Shahed Latif; Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance; 1st Edition; O'Reilly Media; 2009.
2. Ronald L. Krutz, Russell Dean Vines; Cloud Security: A Comprehensive Guide to Secure Cloud Computing; 1st Edition; Wiley; 2010.
3. Nelson, Bill; Phillips, Amelia; Steuart, Christopher, "Guide to Computer Forensics and Investigations; 6th Edition; Cengage Learning; 2018."
4. Marjie T. Britz , "Computer Forensics and Cyber Crime: An Introduction; 3rd Edition; Pearson Education; 2013.

**For MOOCs and other learning Resources**

1. Cloud Computing and Security- IIT Kharagpur / IIT Madras-NPTEL
2. Cloud Security Fundamentals- Cybrary- https://www.cybrary.it/course/cloud-security-fundamentals/
3. Cloud Security- University of Minnesota- https://www.coursera.org/learn/cloud-security

## Honors in Cyber Security

## ABCXXX: Blockchain

Teaching Scheme:
Theory: 2 Hours / Week ; Laboratory: 2 Hours / Week; Tutorials: 1 Hours/Week
Total Credits: 4

**Prerequisites:**
Programming Fundamentals, Data Structures, Operating Systems, Computer Networks, Basic Cryptography

**Course Objectives**

1. Understand the core principles and architecture of blockchain technology.
2. Explore cryptographic foundations and consensus algorithms that secure blockchain networks.
3. Gain practical knowledge of major blockchain platforms including Bitcoin, Ethereum, and Hyperledger Fabric.
4. Develop and deploy smart contracts and decentralized applications (DApps).
5. Analyze real-world blockchain applications and emerging trends.
6. Evaluate security, scalability, and regulatory challenges in blockchain ecosystems.

**Syllabus**
**Theory**

**Unit 1: Blockchain Fundamentals and Cryptography (5 Hours)**
Introduction to Blockchain: History, Concepts, Types (Public, Private, Permissioned), Blockchain Architecture: Blocks, Chains, Transactions, Ledgers, Cryptographic Foundations: Hash Functions (SHA-256), Digital Signatures (ECDSA), Public/Private Keys, Distributed Systems Basics: Consensus Problem, Byzantine Generals Problem, Game Theory (Nash Equilibrium)

**Unit 2: Consensus Mechanisms and Security (5 Hours)**
Consensus Algorithms: Proof of Work (PoW), Proof of Stake (PoS), Delegated PoS, Byzantine Fault Tolerance (BFT), Mining and Incentive Structures, Security Aspects: Double Spending, 51% Attacks, Sybil Attacks, Privacy Techniques: Zero-Knowledge Proofs, Anonymity in Blockchain (Monero, Zcash)

**Unit 3: Blockchain Platforms and Protocols (6 Hours)**

Bitcoin Protocol: Transactions, Wallets, Mining, Scripting, Ethereum: Ethereum Virtual Machine (EVM), Smart Contracts, Solidity Basics, Enterprise Blockchains: Hyperledger Fabric Architecture, Chaincode, Other Platforms Overview: Ripple, Multichain, Stellar

**Unit 4: Smart Contracts and Decentralized Applications (DApps) (4 Hours)**
Smart Contract Development: Solidity Language, Contract Lifecycle, Security Best Practices, Development Tools: Remix IDE, Truffle, Hardhat, DApp Architecture and Development: Web3.js, Ethers.js, Metamask Integration, Deployment and Testing of Smart Contracts

**Unit 5: Blockchain Use Cases and Industry Applications (4 Hours)**
Use Cases: Supply Chain Management, Finance, Healthcare, Voting, Identity Management, Blockchain Integration with IoT, AI, and Cybersecurity, Case Studies: Food Industry Supply Chain, Land Records, Digital Identity

**Unit 6: Emerging Trends, Challenges, and Future Directions (4 Hours)**
Scalability and Interoperability Challenges, Layer 2 Solutions and Sidechains, Regulatory, Legal, and Ethical Considerations, Future Trends: NFTs, DeFi, Web3, Cross-chain Technologies

## Syllabus
## Laboratory

**List of Lab Experiments**

1. **Setting up Bitcoin and Ethereum Wallets**
   Create and configure wallets (e.g., MetaMask, Trust Wallet), perform test transactions to simulate real cryptocurrency transfers as done on exchanges like Binance or Coinbase.
2. **Exploring Blockchain Explorers and Transaction Tracing**
   Use tools like Etherscan and Blockchain.com to trace Bitcoin and Ethereum transactions, similar to how blockchain analysts investigate transaction flows for compliance or fraud detection.
3. **Writing and Deploying Smart Contracts using Solidity on Remix IDE**
   Develop and deploy a smart contract simulating a real-world escrow agreement or crowdfunding contract on Ethereum testnet.
4. **Building a Decentralized Application (DApp) with Web3.js and Metamask**
   Create a simple web-based voting system where users interact with a blockchain

through Metamask, similar to how users vote in DAOs (Decentralized Autonomous Organizations).

5. **Implementing Consensus Simulation: Proof of Work (PoW) and Proof of Stake (PoS)**
   Simulate how consensus works by building small-scale PoW (Bitcoin-like) and PoS (Ethereum 2.0-like) systems demonstrating block creation and validation.
6. **Hyperledger Fabric Network Setup and Chaincode Deployment**
   Set up a permissioned blockchain to simulate a food supply chain network for tracking produce, replicating real-world Hyperledger Fabric use by Walmart or IBM Food Trust.
7. **Developing and Deploying a Token (ERC-20 or ERC-721) on Ethereum Testnet**
   Create and deploy a cryptocurrency token (ERC-20) or NFT (ERC-721), similar to tokens used in loyalty programs or NFT art platforms like OpenSea.
8. **Security Testing of Smart Contracts with Common Vulnerabilities**
   Use tools like Mythril or Slither to identify security loopholes (e.g., reentrancy attacks) that have caused real-world incidents such as the infamous DAO hack.
9. **Building a Simple NFT Marketplace Prototype**
   Simulate listing, buying, and transferring NFTs on a mini-marketplace, similar to platforms like OpenSea or Rarible.
10. **Blockchain Integration with IoT: Real-Time Data Logging**
    Simulate an IoT sensor (e.g., environmental temperature) sending data to a blockchain smart contract for immutable record-keeping, similar to real-world applications by IBM or VeChain.
11. **Performance and Gas Cost Analysis of Blockchain Applications**
    Analyze transaction fees and optimize smart contract code to reduce gas consumption, similar to how developers minimize costs for DeFi or NFT platforms.
12. **Interacting with a Layer 2 Solution (e.g., Polygon) for Scalable Blockchain Applications**
    Deploy smart contracts and interact with them on Polygon's testnet to experience reduced transaction fees and increased scalability, as done in popular blockchain games and dApps.

## Syllabus
## Course Projects

**List of Course Projects**

1. Decentralized voting system using smart contracts
2. Supply chain tracking on Hyperledger Fabric
3. Land record management with blockchain
4. IoT data logging on blockchain network
5. NFT marketplace for digital asset trading
6. DeFi lending and borrowing platform
7. Decentralized identity management system
8. Healthcare data sharing using blockchain
9. Cross-border payments with stablecoins
10. Blockchain-based crowdfunding platform
11. Peer-to-peer energy trading using blockchain

## Syllabus
## Tutorials

**List of Tutorials**

1. Comparative study of blockchain ecosystem and its real-world industry impact
2. Role of Game Theory in blockchain incentive models beyond Byzantine Generals Problem
3. Cryptographic attack vectors on blockchain and their real-world prevention strategies
4. Evolution of blockchain platforms beyond Bitcoin and Ethereum (e.g., Polkadot, Cosmos)
5. Analysis of real blockchain governance models (e.g., Ethereum DAO, Tezos voting)
6. Token design and cryptocurrency economic models for sustainable blockchain networks
7. Regulatory challenges and policy evolution for blockchain in India and globally
8. Ethical dilemmas in blockchain adoption: privacy vs transparency debate
9. Interoperability case studies: how different blockchains communicate (e.g., Polkadot, Chainlink)

10. Future research directions: blockchain in quantum-resistant systems and green energy applications

## Course Outcomes

**Course Outcomes:** Students will be able to

1. Apply the core principles of blockchain, distributed ledgers, and cryptographic foundations to design basic blockchain-based solutions.
2. Analyze consensus mechanisms and blockchain security models used in public and permissioned networks.
3. Demonstrate the use of major blockchain platforms like Bitcoin, Ethereum, and Hyperledger Fabric.
4. Develop and deploy smart contracts and decentralized applications using industry-relevant tools.
5. Evaluate real-world blockchain applications and assess integration challenges with IoT, AI, and existing systems.
6. Assess scalability solutions, interoperability approaches, and emerging trends in blockchain ecosystems.

## Books and E-Resources

**Textbook References**

1. A.Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder; *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*; 1st Edition; Princeton University Press; 2016.
2. A.M. Antonopoulos; *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*; 2nd Edition; O'Reilly Media; 2017.
3. A.M. Antonopoulos, G. Wood; *Mastering Ethereum: Building Smart Contracts and DApps*; 1st Edition; O'Reilly Media; 2018.
4. D.Drescher; *Blockchain Basics: A Non-Technical Introduction in 25 Steps*; 1st Edition; Apress; 2017.
5. N.Gaur, L. Desrosiers, V. Ramakrishna, P. Novotny, A. Singh, M. Bocarsly; *Hyperledger Fabric In Action*; 1st Edition; Manning Publications; 2020.

**MOOC References**

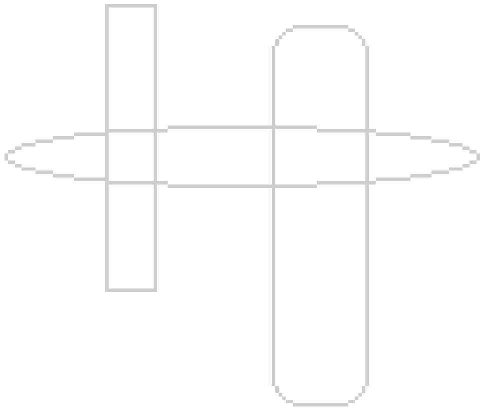
1. A. Narayanan, J. Bonneau; *Bitcoin and Cryptocurrency Technologies*; Coursera; https://www.coursera.org/learn/cryptocurrency

2. University at Buffalo; *Blockchain Basics*; Coursera; https://www.coursera.org/learn/blockchain-basics

3. Prof. Sandip Chakrborty and Prof. Shamik Sural ; *Blockchain and its Applications*; NPTEL; https://nptel.ac.in/courses/106105235

4. Stipen Grider; *Ethereum and Solidity: The Complete Developer's Guide*; Udemy; https://www.udemy.com/course/ethereum-and-solidity-the-complete-developers-guide/

5. Linux Foundation; *Hyperledger Fabric Fundamentals*; edX; https://www.edx.org/learn/hyperledger



## Honors in Cyber Security

## ABCXXX: Project

Teaching Scheme:  Laboratory: 8 Hours/Week
Total Credits: 4

### Syllabus

**Aim**

This course addresses the issues associated with the successful management of a  project. The course emphasizes project life cycle phase's requirement engineering, system analysis and system design. A further aim is for students to heighten personal awareness of the importance of developing strategies for themselves and working with peers to create desired outcomes. The Project Work can lead to:

- Transform existing Ideas into conceptual models.
- Transform conceptual models into determinable models.
- Use determinable models to obtain system specifications.
- Select optimum specifications and create physical models.
- Apply the results from physical models to create real target systems.

**Project Group and Topic Selection and Synopsis:**

The project work needs to be undertaken by a group of maximum FOUR and minimum of THREE students. The Project work will be jointly performed by the project team members. The student needs to identify a technological problem in the area of Computer Engineering or Information Technology of their choice and address the problem by formulating a solution for the identified problem. The Project Group will prepare a synopsis of the project work which will be approved by the concerned faculty member. The project should not be a reengineering or reverse engineering project. In some cases, reverse engineering projects will be permissible based on the research component involved in it. The project work aims at solving a real world technical problem. Hence ample literature survey is required to be done by the students. Application-oriented projects will not be acceptable. Low-level custom User Interface development and its allied mapping with a particular technology will not be accepted.

**Overview of the Course:**

41. The Student Project Group is expected to make a survey of situation for identifying the requirements of selected Technological Problem. The Student Project Group will be monitored by Internal Guides and External Guides (if any).
42. The project requires the students to conceive, design, implement and operate a mechanism (the design problem).  The mechanism may be entirely of the student's own design, or it may incorporate off-the-shelf parts.  If the mechanism incorporates off-the-shelf parts, the students must perform appropriate analysis to show that the parts are suitable for their intended purpose in the mechanism.

43. The project must be based on a Fresh Idea or Implementation of a Theoretical Problem – meaning that there is not a known Solution to the design problem Or Create a Better Solution.
44. The project must have an experimental component. Students must conceive, design, implement and operate an appropriate experiment as part of the project. The experiment might be to collect data about some aspect of the design (i.e., to verify that the design will work as expected). Alternatively, the experiment could be to verify that the final mechanism performs as expected.
45. Upon receiving the approval, the Student Project Group will prepare a preliminary project report consisting, Feasibility Study Document, System Requirement Specification, System Analysis Document, Preliminary System Design Document. All the documents indicated will have a prescribed format.
46. Upon project completion, the Student Project Group will prepare a detailed Project Report consisting Semester I Preliminary Project document along with Detailed System Design Document, Implementation and Testing Document with conclusion and future scope of the Project Work. All the documents indicated will have a prescribed format. The Project Report ideally should consist of following documents : (Exceptions may be there based on the nature of the project, especially if some of the following documents are not applicable to a particular project as determined by the project guide, coordinator and head of department).
47. The Project Work will be assessed jointly by a panel of examiners consisting faculty and industry experts. The Project Groups will deliver the presentation and demonstration of the Project Work which will be assessed by the panel.
48. The Student Project Group needs to actively participate in the presentation. The panel of examiners will evaluate the candidate’s performance based on presentation skills, questions based on the Project Work and overall development effort taken by the candidates.

**Note:**
The student needs to design and develop solution for the identified technological problem in the area of Computer Engineering or Information Technology of their choice. The Project Implementation needs to be completed using best possible use of available technologies as applicable to deal with the complexity of the project. The Project Group will prepare a detailed report of the project work which will be approved by the concerned faculty member. The Project Report need to be submitted both in Hard form and Soft form in CD. The Soft Copy of the Project Report must accompany other project deliverables as well.