1. **Python Basics :**
    - 1.1. Python Basics: Google Colaboratory
    - 1.2. Python Basics: Basic Data Types
    - 1.3. Python Basics: Lists
    - 1.4. Python Basics: Tuples, Sets, Dictionaries
    - 1.5. Python Basics: Packages
    - 1.6. Python Basics: File Handling
    - 1.7. Python Basics: Classes
    - 1.8. Python Basics: Numpy
    - 1.9. Python Basics: Plotting
2. **Expert Systems Theory :**
    - 2.1. Expert Systems & 6 Jars: Expert Systems
    - 2.2. Expert Systems & 6 Jars: Say Hi To ML
    - 2.3. Expert Systems & 6 Jars: Introduction
    - 2.4. Expert Systems & 6 Jars: Data
    - 2.5. Expert Systems & 6 Jars: Tasks
    - 2.6. Expert Systems & 6 Jars: Models
    - 2.7. Expert Systems & 6 Jars: Loss Function
    - 2.8. Expert Systems & 6 Jars: Learning Algorithm
    - 2.9. Expert Systems & 6 Jars: Evaluation
    - 2.10. Expert Systems & 6 Jars: Six Jars Summary (Part 1)
    - 2.11. Expert Systems & 6 Jars: Six Jars Summary (Part 2)
3. **Linear Algebra Basics :**
    - 3.1. Vectors & Matrices: Introduction to Vectors
    - 3.2. Vectors & Matrices: Dot product of vectors
    - 3.3. Vectors & Matrices: Unit Vectors
    - 3.4. Vectors & Matrices: Projection of one vector onto another
    - 3.5. Vectors & Matrices: Angle between two vectors
    - 3.6. Vectors & Matrices: Why do we care about vectors?
    - 3.7. Vectors & Matrices: Introduction to Matrices
    - 3.8. Vectors & Matrices: Multiplying a vector by a matrix
    - 3.9. Vectors & Matrices: Multiplying a matrix by another matrix
    - 3.10. Vectors & Matrices: An alternate way of multiplying two matrices
    - 3.11. Vectors & Matrices: Why do we care about matrices?
4. **Python Basics 2 :**
    - 4.1. Python Basics + Linear Algebra: Google Drive and Colab Integration
    - 4.2. Python Basics + Linear Algebra: Pandas
    - 4.3. Python Basics + Linear Algebra: Python Debugger
    - 4.4. Python Basics + Linear Algebra: Plotting Vectors
    - 4.5. Python Basics + Linear Algebra: Vector Addition and Subtraction
    - 4.6. Python Basics + Linear Algebra: Vector Dot Product
5. **MP Neuron :**

- **5.1. MP Neuron: Introduction**
- **5.2. MP Neuron: Model**
- **5.3. MP Neuron: Data & Task**
- **5.4. MP Neuron: Loss**
- **5.5. MP Neuron: Learning**
- **5.6. MP Neuron: Evaluation**
- **5.7. MP Neuron: Geometry Basics**
- **5.8. MP Neuron: Geometric Interpretation**
- **5.9. MP Neuron: Summary**

**6. Perceptron Model :**

- **6.1. Perceptron: Introduction**
- **6.2. Perceptron: Data & Task**
- **6.3. Perceptron: Model**
- **6.4. Perceptron: Geometric Interpretation**
- **6.5. Perceptron: Loss Function**
- **6.6. Perceptron: Learning - General Recipe**
- **6.7. Perceptron: Learning Algorithm**
- **6.8. Perceptron: Learning - Why it Works?**
- **6.9. Perceptron: Learning - Will it Always Work?**
- **6.10. Perceptron: Evaluation**
- **6.11. Perceptron: Summary**

**7. MP Neuron and Perceptron Practical :**

- **7.1. MP Neurons & Perceptron using Python: Toy Example (Perceptron)**
- **7.2. MP Neurons & Perceptron using Python: Loading Data**
- **7.3. MP Neurons & Perceptron using Python: Train-Test Split**
- **7.4. MP Neurons & Perceptron using Python: Binarization**
- **7.5. MP Neurons & Perceptron using Python: Inference And Search**
- **7.6. MP Neurons & Perceptron using Python: Inference**
- **7.7. MP Neurons & Perceptron using Python: Class**
- **7.8. MP Neurons & Perceptron using Python: Perceptron Class**
- **7.9. MP Neurons & Perceptron using Python: Epochs**
- **7.10. MP Neurons & Perceptron using Python: Checkpointing**
- **7.11. MP Neurons & Perceptron using Python: Learning Rate**
- **7.12. MP Neurons & Perceptron using Python: Weight Animation**
- **7.13. MP Neurons & Perceptron using Python: Exercises**

**8. Contest 1 :**

- 8.1. Contest - Mobile Phone Like/Dislike predictor: Introduction to Contests.
- 8.2. Contest - Mobile Phone Like/Dislike predictor: Creating a Kaggle account
- 8.3. Contest - Mobile Phone Like/Dislike predictor: Data preprocessing
- 8.4. Contest - Mobile Phone Like/Dislike predictor: Submitting Entries
- 8.5. Contest - Mobile Phone Like/Dislike predictor: Clarifications

**9. Sigmoid Neuron :**

9.1. 6 Jars of Sigmoid Neuron (I): Recap from last lecture
9.2. 6 Jars of Sigmoid Neuron (I):Revisiting limitations of perceptron model
9.3. 6 Jars of Sigmoid Neuron (I): Sigmoid Model Part 1
9.4. 6 Jars of Sigmoid Neuron (I): Sigmoid Model Part 2
9.5. 6 Jars of Sigmoid Neuron (I): Sigmoid Model Part 3
9.6. 6 Jars of Sigmoid Neuron (I): Sigmoid Model Part 4
9.7. 6 Jars of Sigmoid Neuron (I): Sigmoid: Data and Tasks
9.8. 6 Jars of Sigmoid Neuron (I): Sigmoid: Loss Function
9.9. 6 Jars of Sigmoid Neuron (I): Learning: Intro to Learning Algorithm
9.10. 6 Jars of Sigmoid Neuron (I): Learning: Learning by guessing
9.11. 6 Jars of Sigmoid Neuron (I): Learning - Error Surfaces for learning
9.12. 6 Jars of Sigmoid Neuron (I): Learning - Mathematical setup for the learning algorithm
9.13. 6 Jars of Sigmoid Neuron (I): Learning - The math-free version of learning algorithm
9.14. 6 Jars of Sigmoid Neuron (II): Learning - Introducing Taylor Series
9.15. 6 Jars of Sigmoid Neuron (II): Learning - More intuitions about Taylor series
9.16. 6 Jars of Sigmoid Neuron (II): Learning - Deriving the Gradient Descent Update rule
9.17. 6 Jars of Sigmoid Neuron (II): Learning - The complete learning algorithm
9.18. 6 Jars of Sigmoid Neuron (II): Learning - Computing Partial Derivatives
9.19. 6 Jars of Sigmoid Neuron (II): Learning - Writing the code
9.20. 6 Jars of Sigmoid Neuron (II): Sigmoid - Dealing with more than 2 parameters
9.21. 6 Jars of Sigmoid Neuron (II): Sigmoid - Evaluation
9.22. 6 Jars of Sigmoid Neuron (II): Summary and take-aways

10. Sigmoid Neuron using Python :
10.1. Sigmoid Neurons Using Python (I): Plotting Sigmoid 2D
10.2. Sigmoid Neurons Using Python (I): Plotting Sigmoid 3D
10.3. Sigmoid Neurons Using Python (I): Plotting Loss
10.4. Sigmoid Neurons Using Python (I): Contour Plot
10.5. Sigmoid Neurons Using Python (I): Class
10.6. Sigmoid Neurons Using Python (I): Toy Data Fit
10.7. Sigmoid Neurons Using Python (I): Toy Data Plot - 1/2
10.8. Sigmoid Neurons Using Python (I): Toy Data Plot - 2/2
10.9. Sigmoid Neurons Using Python (II): Loading Data
10.10. Sigmoid Neurons Using Python (II): Standardisation
10.11. Sigmoid Neurons Using Python (II): Test/Train Split (1/2)
10.12. Sigmoid Neurons Using Python (II): Test/Train Split (2/2)
10.13. Sigmoid Neurons Using Python (II): Fitting Data
10.14. Sigmoid Neurons Using Python (II): Loss Plot
10.15. Sigmoid Neurons Using Python (II): Progress Bar
10.16. Sigmoid Neurons Using Python (II): Exercises

11. **Probability :**
    - 11.1. **Probability Theory Introduction**
    - 11.2. **Probability Theory: Random Variable - Intuition**
    - 11.3. **Probability Theory: Random Variable - Formal Definition**
    - 11.4. **Probability Theory: Random Variable - Continuous and Discrete**
    - 11.5. **Probability Theory: Probability Distribution**
    - 11.6. **Probability Theory: True and Predicted Distribution**
    - 11.7. **Probability Theory: Certain Events**
    - 11.8. **Probability Theory: Why Do we Care About Distributions**
12. **Information Theory & Cross Entropy :**
    - 12.1. **Information Theory and Cross Entropy: Expectation**
    - 12.2. **Information Theory and Cross Entropy: Information Content**
    - 12.3. **Information Theory and Cross Entropy: Entropy**
    - 12.4. **Information Theory and Cross Entropy: Relation To Number Of Bits**
    - 12.5. **Information Theory and Cross Entropy: KL-Divergence and Cross Entropy**
    - 12.6. **Information Theory and Cross Entropy: Sigmoid Neuron and Cross Entropy**
    - 12.7. **Information Theory and Cross Entropy: Using Cross Entropy With Sigmoid Neuron**
    - 12.8. **Information Theory and Cross Entropy: Learning Algorithm for Cross Entropy loss function**
    - 12.9. **Information Theory and Cross Entropy: Computing partial derivatives with cross entropy loss**
    - 12.10. **Information Theory and Cross Entropy: Code for Cross Entropy Loss function**
13. **Contest 2 :**
    - 13.1. Contest - Text, Non-Text Classification: Overview
14. **Representation Power of Function :**
    - 14.1. Representation Power of function: Why do we need complex functions
    - 14.2. Representation Power of function: Complex functions in real world examples
    - 14.3. Representation Power of function: A simple recipe for building complex functions
    - 14.4. Representation Power of function: Illustrative Proof of Universal Approximation Theorem
    - 14.5. Representation Power of function: Summary
15. **Deep Neural Networks :**
    - 15.1. Deep Neural Networks: Setting the context
    - 15.2. Deep Neural Networks: Data and Tasks
    - 15.3. Deep Neural Networks: Model: A simple deep neural network
    - 15.4. Deep Neural Networks: Model: A generic deep neural network
    - 15.5. Deep Neural Networks: Model: Understanding the computations in a deep neural network
    - 15.6. Deep Neural Networks: Model: The output layer of a deep neural network

21.5. Variants of Gradient Descent: Exercise: Guess the 3D surface
21.6. Variants of Gradient Descent: Visualizing gradient descent on a 2D contour map
21.7. Variants of Gradient Descent: Intuition for momentum based gradient descent
21.8. Variants of Gradient Descent: Dissecting the update rule for momentum based gradient descent
21.9. Variants of Gradient Descent: Running and visualizing momentum based gradient descent
21.10. Variants of Gradient Descent: A disadvantage of momentum based gradient descent
21.11. Variants of Gradient Descent: Intuition behind nesterov accelerated gradient descent
21.12. Variants of Gradient Descent: Running and visualizing nesterov accelerated gradient descent
21.13. Variants of Gradient Descent: Summary and what next

**22. Optimisation Algorithms Theory 2:**

22.1. Variants of Gradient Descent: The idea of stochastic and mini-batch gradient descent
22.2. Variants of Gradient Descent: Running stochastic gradient descent
22.3. Variants of Gradient Descent: Running mini-batch gradient descent
22.4. Variants of Gradient Descent: Epochs and Steps
22.5. Variants of Gradient Descent: Why do we need an adaptive learning rate ?
22.6. Variants of Gradient Descent: Introducing Adagrad
22.7. Variants of Gradient Descent: Running and Visualizing Adagrad
22.8. Variants of Gradient Descent: A limitation of Adagrad
22.9. Variants of Gradient Descent: Running and visualizing RMSProp
22.10. Variants of Gradient Descent: Running and visualizing Adam
22.11. Variants of Gradient Descent: Summary

**23. Optimisation Algorithms Practical:**

23.1. Implementing Optimization Algorithms Using Python: Outline
23.2. Implementing Optimization Algorithms Using Python: Modified Sigmoid Neuron Class
23.3. Implementing Optimization Algorithms Using Python: Setup for Plotting
23.4. Implementing Optimization Algorithms Using Python: Gradient Descent Algorithm
23.5. Implementing Optimization Algorithms Using Python: GD Algorithm - Contour Plot
23.6. Implementing Optimization Algorithms Using Python: Momentum
23.7. Implementing Optimization Algorithms Using Python: Nesterov Accelerated GD
23.8. Implementing Optimization Algorithms Using Python: Mini-Batch GD
23.9. Implementing Optimization Algorithms Using Python: AdaGrad
23.10. Implementing Optimization Algorithms Using Python: RMSProp
23.11. Implementing Optimization Algorithms Using Python: Adam
23.12. Implementing Optimization Algorithms Using Python: Vectorised Class Recap
23.13. Implementing Optimization Algorithms Using Python: Vectorised GD Algorithms

23.14. Implementing Optimization Algorithms Using Python: Performance of Different Algorithms
23.15. Implementing Optimization Algorithms Using Python: Good solutions and Exercise

**24. Activation Functions Theory:**

24.1. Activation Functions & Initialization Methods: Setting the context
24.2. Activation Functions & Initialization Methods: Saturation in logistic neuron
24.3. Activation Functions & Initialization Methods: Zero centered functions
24.4. Activation Functions & Initialization Methods: Introducing Tanh and ReLU activation functions
24.5. Activation Functions & Initialization Methods: Tanh and ReLU Activation Functions
24.6. Activation Functions & Initialization Methods: Symmetry Breaking Problem
24.7. Activation Functions & Initialization Methods: Xavier and He initialization
24.8. Activation Functions & Initialization Methods: Summary and what next

**25. Activation Functions Practical :**

25.1. Hands-on - activation functions & initialization methods: Introduction and Activation Functions
25.2. Hands-on - activation functions & initialization methods: Activation Functions
25.3. Hands-on - activation functions & initialization methods: Plotting Setup
25.4. Hands-on - activation functions & initialization methods: Sigmoid
25.5. Hands-on - activation functions & initialization methods: Tanh
25.6. Hands-on - activation functions & initialization methods: ReLu
25.7. Hands-on - activation functions & initialization methods: Leaky ReLU
25.8. Hands-on - activation functions & initialization methods: Exercises

**26. Regularization Theory:**

26.1. Regularization: Simple v/s complex models
26.2. Regularization: Analysing the behavior of simple and complex models
26.3. Regularization: Bias and Variance
26.4. Regularization: Test error due to high bias and high variance
26.5. Regularization: Overfitting in deep neural networks
26.6. Regularization: A detour into hyperparameter tuning
26.7. Regularization: L2 regularization
26.8. Regularization: Dataset Augmentation and Early Stopping
26.9. Regularization: Summary

**27. Regularization Practical :**

27.1. Hands-on - Regularization: Outline and Libraries
27.2. Hands-on - Regularization: L2 Regularisation in Code
27.3. Hands-on - Regularization: Bias on Increasing Model Complexity
27.4. Hands-on - Regularization: L2 Regularisation in Action
27.5. Hands-on - Regularization: Adding Noise to Input Features
27.6. Hands-on - Regularization: Early Stopping and Exercises

**28. Basics of Pytorch :**

- **28.1. Basics of Pytorch: Outline**
- **28.2. Basics of Pytorch: PyTorch Tensors**
- **28.3. Basics of Pytorch: Simple Tensor Operations**
- **28.4. Basics of Pytorch: NumPy vs PyTorch**
- **28.5. Basics of Pytorch: GPU PyTorch**
- **28.6. Basics of Pytorch: Automatic Differentiation**
- **28.7. Basics of Pytorch: Loss Function with AutoDiff**
- **28.8. Basics of Pytorch: Learning Loop GPU (Plays on Safari)**

**29. FNNs using Pytorch :**

- **29.1. FNNs using Pytorch: Outline**
- 29.2. FNNs using Pytorch: Forward Pass With Tensors
- 29.3. FNNs using Pytorch: Functions for Loss, Accuracy, Backpropagation
- 29.4. FNNs using Pytorch: PyTorch Modules - NN and Optim
- 29.5. FNNs using Pytorch: NN Sequential and Code Structure
- 29.6. FNNs using Pytorch: GPU Execution
- 29.7. FNNs using Pytorch: Exercises and Recap

**30. The Convolution Operation :**

- 30.1. The convolution operation: Setting the Context
- 30.2. The convolution operation: The 1D convolution operation
- 30.3. The convolution operation: The 2D Convolution Operation
- 30.4. The convolution operation: Examples of 2D convolution
- 30.5. The convolution operation: 2D convolution with a 3D filter
- 30.6. The convolution operation: Terminology
- 30.7. The convolution operation: Padding and Stride

**31. Convolution to Neural Networks :**

- 31.1. From convolution operation to neural networks: How is the convolution operation related to Neural Networks - Part 1
- 31.2. From convolution operation to neural networks: How is the convolution operation related to Neural Networks - Part 2
- 31.3. From convolution operation to neural networks: How is the convolution operation related to Neural Networks - Part 3
- 31.4. From convolution operation to neural networks: Understanding the input/output dimensions
- 31.5. From convolution operation to neural networks: Sparse Connectivity and Weight Sharing
- 31.6. From convolution operation to neural networks: Max Pooling and Non-Linearities
- 31.7. From convolution operation to neural networks: Our First Convolutional Neural Network (CNN)
- 31.8. From convolution operation to neural networks: Training CNNs
- 31.9. From convolution operation to neural networks: Summary and what next

**32. CNNs in Pytorch :**

32.1. CNNs in Pytorch: Outline
32.2. CNNs in Pytorch: Loading Data Sets
32.3. CNNs in Pytorch: Visualising Weights
32.4. CNNs in Pytorch: Single Convolutional Layer
32.5. CNNs in Pytorch: Deep CNNs
32.6. CNNs in Pytorch: LeNet
32.7. CNNs in Pytorch: Training Le Net
32.8. CNNs in Pytorch: Visualising Intermediate Layers, Exercises

**33. CNN Architectures :**

33.1. CNN Architectures - Part 1: Setting the context
33.2. CNN Architectures - Part 1: The Imagenet Challenge
33.3. CNN Architectures - Part 1: Understanding the first layer of AlexNet
33.4. CNN Architectures - Part 1: Understanding all layers of AlexNet
33.5. CNN Architectures - Part 1: ZFNet
33.6. CNN Architectures - Part 1: VGGNet
33.7. CNN Architectures - Part 1: Summary
33.8. CNN Architectures - Part 2: Setting the context
33.9. CNN Architectures - Part 2: Number of computations in a convolution layer
33.10. CNN Architectures - Part 2: 1x1 Convolutions
33.11. CNN Architectures - Part 2: The Intuition behind GoogLeNet
33.12. CNN Architectures - Part 2: The Inception Module
33.13. CNN Architectures - Part 2: The GoogleNet Architecture
33.14. CNN Architectures - Part 2: Average Pooling
33.15. CNN Architectures - Part 2: Auxiliary Loss for training a deep network
33.16. CNN Architectures - Part 2: ResNet

**34. Building CNNs :**

34.1. Building CNN Architectures Using Pytorch: Outline
34.2. Building CNN Architectures Using Pytorch: Image Transforms
34.3. Building CNN Architectures Using Pytorch: VGG
34.4. Building CNN Architectures Using Pytorch: Training VGG
34.5. Building CNN Architectures Using Pytorch: Pre-trained Models
34.6. Building CNN Architectures Using Pytorch: Checkpointing Models
34.7. Building CNN Architectures Using Pytorch: ResNet
34.8. Building CNN Architectures Using Pytorch: Inception Part 1
34.9. Building CNN Architectures Using Pytorch: Inception Part 2
34.10. Building CNN Architectures Using Pytorch: Exercises

**35. Visualising CNNs :**

35.1. Visualising CNNs: Receptive field of a neuron
35.2. Visualising CNNs: Identifying images which cause certain neurons to fire
35.3. Visualising CNNs: Visualising filters
35.4. Visualising CNNs: Occlusion experiments
35.5. Visualising CNNs Using Python: Outline

35.6. Visualising CNNs Using Python: Custom Torchvision Dataset
35.7. Visualising CNNs Using Python: Visualising inputs
35.8. Visualising CNNs Using Python: Occlusion
35.9. Visualising CNNs Using Python: Visualising filters
35.10. Visualising CNNs Using Python: Visualising filters - code

**36. Batch Normalization and Dropout :**

36.1. Batch Normalization and Dropout: Normalizing inputs
36.2. Batch Normalization and Dropout: Why should we normalize the inputs
36.3. Batch Normalization and Dropout: Batch Normalization
36.4. Batch Normalization and Dropout: Learning Mu and Sigma
36.5. Batch Normalization and Dropout: Ensemble Methods
36.6. Batch Normalization and Dropout: The idea of dropout
36.7. Batch Normalization and Dropout: Training without dropout
36.8. Batch Normalization and Dropout: How does weight sharing help ?
36.9. Batch Normalization and Dropout: Using dropout at test time
36.10. Batch Normalization and Dropout: How does dropout act as a regularizer ?
36.11. Batch Normalization and Dropout: Summary and what next ?

**37. Batch Normalization and Dropout using Python :**

37.1. Batch Normalization and Dropout Using Python: Outline and Dataset
37.2. Batch Normalization and Dropout Using Python: Batch Norm Layer
37.3. Batch Normalization and Dropout Using Python: Batch Norm Visualisation
37.4. Batch Normalization and Dropout Using Python: Batch Norm 2d
37.5. Batch Normalization and Dropout Using Python: Dropout layer
37.6. Batch Normalization and Dropout Using Python: Dropout Visualisation and Exercises

**38. Hypermeter Tuning and MLFlow:**

38.1. Hyperparameter Tuning and MLFlow: Outline
38.2. Hyperparameter Tuning and MLFlow: Colab on Local Runtime
38.3. Hyperparameter Tuning and MLFlow: MLFlow installation and basic usage
38.4. Hyperparameter Tuning and MLFlow: Hyperparameter Tuning
38.5. Hyperparameter Tuning and MLFlow: Refined Search for Hyperparameters
38.6. Hyperparameter Tuning and MLFlow: Logging Image Artifacts
38.7. Hyperparameter Tuning and MLFlow: Logging and Loading Models
38.8. Hyperparameter Tuning and MLFlow: One Last Visualisation

**39. Sequence Learning Problems :**

39.1. Sequence Learning Problems: Setting the context
39.2. Sequence Learning Problems: Introduction to sequence learning problems
39.3. Sequence Learning Problems: Some more examples of sequence learning problems
39.4. Sequence Learning Problems: Sequence learning problems using video and speech data

39.5. Sequence Learning Problems: A wishlist for modelling sequence learning problems
39.6. Sequence Learning Problems: Intuition behind RNNs - Part 1
39.7. Sequence Learning Problems: Intuition behind RNNs - Part 2
39.8. Sequence Learning Problems: Introducing RNNs
39.9. Sequence Learning Problems: Summary and what next

**40. Recurrent Neural Networks :**

40.1. Recurrent Neural Networks: Setting the context
40.2. Recurrent Neural Networks: Data and Tasks - Sequence Classification - Part 1
40.3. Recurrent Neural Networks: Data and Tasks - Sequence Classification - Part 2
40.4. Recurrent Neural Networks: A clarification about padding
40.5. Recurrent Neural Networks: Data and Tasks - Sequence Labelling
40.6. Recurrent Neural Networks: Model
40.7. Recurrent Neural Networks: Loss Function
40.8. Recurrent Neural Networks: Learning Algorithm
40.9. Recurrent Neural Networks: Learning Algorithm - Derivatives w.r.t. V
40.10. Recurrent Neural Networks: Learning Algorithm - Derivatives w.r.t. W
40.11. Recurrent Neural Networks: Evaluation
40.12. Recurrent Neural Networks: Summary and what next

**41. Vanishing and Exploding Gradients :**

41.1. Vanishing and Exploding Gradients: Revisiting the gradient wrt W
41.2. Vanishing and Exploding Gradients: Zooming into one element of the chain rule - Part 1
41.3. Vanishing and Exploding Gradients: Zooming into one element of the chain rule - Part 2
41.4. Vanishing and Exploding Gradients: A small detour to calculus
41.5. Vanishing and Exploding Gradients: Looking at the magnitude of the derivative
41.6. Vanishing and Exploding Gradients: Exploding and vanishing gradients
41.7. Vanishing and Exploding Gradients: Summary and what next

**42. LSTMs and GRUs:**

42.1. LSTMs and GRUs: Dealing with longer sequences
42.2. LSTMs and GRUs: The white board analogy
42.3. LSTMs and GRUs: Real world example of longer sequences
42.4. LSTMs and GRUs: Going back to RNNs
42.5. LSTMs and GRUs: Selective Write - Part 1
42.6. LSTMs and GRUs: Selective Write - Part 2
42.7. LSTMs and GRUs: Selective Read
42.8. LSTMs and GRUs: Selective forget
42.9. LSTMs and GRUs: An example computation with LSTMs
42.10. LSTMs and GRUs: Gated recurrent units
42.11. LSTMs and GRUs: Summary and what next

**43. Sequence Models in Pytorch :**

43.1. Sequence Models in PyTorch: OutlineSequence Models in PyTorch: Outline
43.2. Sequence Models in PyTorch: Dataset and Task
43.3. Sequence Models in PyTorch: RNN Model
43.4. Sequence Models in PyTorch: Inference on RNN
43.5. Sequence Models in PyTorch: Training RNN
43.6. Sequence Models in PyTorch: Training Setup
43.7. Sequence Models in PyTorch: LSTM
43.8. Sequence Models in PyTorch: GRU and Exercises

**44. Addressing the problem of vanishing and exploding gradients :**

44.1. Addressing the problem of vanishing and exploding gradients: Quick Recap
44.2. Addressing the problem of vanishing and exploding gradients: Intuition: How gates help to solve the problem of vanishing gradients
44.3. Addressing the problem of vanishing and exploding gradients: Revisiting vanishing gradients in RNNs
44.4. Addressing the problem of vanishing and exploding gradients: Dependency diagram for LSTMs
44.5. Addressing the problem of vanishing and exploding gradients: Computing the gradient
44.6. Addressing the problem of vanishing and exploding gradients: When do the gradients vanish?
44.7. Addressing the problem of vanishing and exploding gradients: Dealing with exploding gradients
44.8. Addressing the problem of vanishing and exploding gradients: Summary and what next

**45. Batching for Sequence Models in Pytorch :**

45.1. Batching for Sequence Models in Pytorch: Overview
45.2. Batching for Sequence Models in Pytorch: Recap on Sequence Models
45.3. Batching for Sequence Models in Pytorch: Batching for Sequence Models
45.4. Batching for Sequence Models in Pytorch: Padding Vector Representations
45.5. Batching for Sequence Models in Pytorch: Packing in PyTorch
45.6. Batching for Sequence Models in Pytorch: Training with Batched Input

**46. Neural Encoders and Decoders :**

46.1. Neural Encoders and Decoders: Setting the context
46.2. Neural Encoders and Decoders: Revisiting the task of language modelling
46.3. Neural Encoders and Decoders: Using RNNs for language modelling
46.4. Neural Encoders and Decoders: Introducing Encoder Decoder Model
46.5. Neural Encoders and Decoders: Connecting encoder decoder models to the six jars
46.6. Neural Encoders and Decoders: A compact notation for RNNs, LSTMs and GRUs
46.7. Neural Encoders and Decoders: Encoder decoder model for image captioning
46.8. Neural Encoders and Decoders: Six jars for image captioning

46.9. Neural Encoders and Decoders: Encoder decoder for Machine translation
46.10. Neural Encoders and Decoders: Encoder decoder model for transliteration
46.11. Neural Encoders and Decoders: Summary

**47. Attention Mechanism :**

47.1. Attention Mechanism: Motivation for attention mechanism
47.2. Attention Mechanism: Attention mechanism with an oracle
47.3. Attention Mechanism: A model for attention
47.4. Attention Mechanism: The attention function
47.5. Attention Mechanism: Machine translation with attention
47.6. Attention Mechanism: Summary and what next

**48. Encoder Decoder Models and Attention Mechanism using PyTorch :**

48.1. Encoder Decoder Models and Attention Mechanism Using Pytorch: Outline
48.2. Encoder Decoder Models and Attention Mechanism Using Pytorch: Data set and Task
48.3. Encoder Decoder Models and Attention Mechanism Using Pytorch: Data Ingestion - XML processing
48.4. Encoder Decoder Models and Attention Mechanism Using Pytorch: Encoder Decoder Model - 1
48.5. Encoder Decoder Models and Attention Mechanism Using Pytorch: Encoder Decoder Model - 2
48.6. Encoder Decoder Models and Attention Mechanism Using Pytorch: Adding Attention - 1
48.7. Encoder Decoder Models and Attention Mechanism Using Pytorch: Adding Attention - 2
48.8. Encoder Decoder Models and Attention Mechanism Using Pytorch: Model Evaluation and Exercises

**49. RCNN :**

49.1. RCNN: More clarity on regression
49.2. RCNN: Setting the context
49.3. RCNN: A typical pipeline for object detection
49.4. RCNN: Region Proposal
49.5. RCNN: Feature Extraction
49.6. RCNN: Classification
49.7. RCNN: Regression
49.8. RCNN: Training

**50. YOLO :**

50.1. YOLO: Introduction to YOLO
50.2. YOLO: The Output of YOLO
50.3. YOLO: Training
50.4. Object Detection: Summary and what next

**51. Capstone :**

51.1. Capstone Project: Project Details

Watch lectures and then try them out in separate lab(sort of) time. Organise your schedule according to that. Because most of the theory part we already know, it's the implementation part we need to sort and that can be done with better resonance of time meant for research and implementation. There can be different pace for the two imo. The lectures can be seen as their input to you and then your experiments should be independent of the exact lecture words. Rather they should be more open ended and broad.

Spend good time on RL as well. Learning that and being able to implement those concepts could really help you moving forward. Other concepts can be picked up with time, on weekends or at times when you want to do something absolutely different. You have a lot of options to pick out from for that. Flying car nanodegree is the cornerstone of all that and we are finally finding something that has been scalable, is scalable and will be scalable in the foreseeable future, unlike RL or DL and so on, and that is robotics. Spend more time on robotics in the coming future and that will be very rewarding imo. Work on NLP and CV as well.

Completing deep learning should be a good priority, because once that is done, then we can go about miscellaneous sources like research papers and then explore them for other stuff that is blocked right now for some reason. Like computer vision and natural language processing is something we should properly focus on working and exploring and then implementing stuff.

Let’s split the course into theory and practical. Theory includes all lectures and their note making and then there is the practical slot wherein all related exploration and basic code implementation needs to be done related to the code and the concepts. I really think that having a mix in the practical part would not be as such a bad thing, and if we can finish with the theory part by the time the btp is done with, I think we would have a good amount of time before the year is up to finish the practical aspect of exploration and learning things related to deep learning. And then from next semester we can properly focus on the application part of things.

We need to be okay with the process being slow and then going about getting better every day rather than wanting fast progress. We need to manage time better, and then make time for both the theory and practical parts of the course, along with other lecture series and resources. Reading, exploring and coding stuff is the only way to get better at this stuff, and to breaking inhibitions.

We need to be okay investing some good amount of time for the same and not rush through stuff. Rather it should be a mix of new stuff and then exploring on the knowledge already compiled, so as to grow in different forms, and not limiting ourselves in any way. But this process requires time and patience to pay off for sure and also some long nights along with perfect discipline.

This should be a mix of reading, watching independent videos and lecture series as well. But one thing we need to be sure is to complete whatever we pick up. The big problem with me is that I pick up stuff and then I don’t complete it properly and then I get bored and pick something else up as well. Which really hurts the entire process. Rather having a mix of things and then focusing them should be a good way to go at things.

Thinking that you are not doing adequate work is also one reason for unwarranted stress that can surely be avoided. Focus on 1-2 activities and finish them properly and then pick up something new. But in the 1-2 activities make sure that you immerse yourself properly. And also remember that you can’t consume everything, there will always be something that you can’t get to and something that you will miss for sure. And also there are things which you would benefit more from by focussing on now, rather than something else.

1. Machine Learning
    a. Coding Ninjas
2. Deep Learning
    a. One Fourth Labs Course

    b. Lecture Series by fast.ai
3. Self Driving Cars
    a. Course by Coursera
    b. ARES pipeline research and development
4. Reinforcement Learning
    a. Udacity Course
    b. CS285 by UC Berkeley
5. Robotics
6. Aerial Vehicles
7. Random Algorithms
8. Mathematics
9. Natural Language Processing
10. Computer Vision
11. Web Development
    a. Colt Steele Course
12. Open Source Development
13. Google Summer of Code

Deep Learning, do properly and Web Development do properly. Extending them won't be that hard. Mix it up in between. Like on weekends, do something something entirely different.

Implementing papers, reading one paper per week and making notes on them.