1. Deep Reinforcement Learning: Combines deep learning and reinforcement learning to help agents learn how to act in complex environments.

2. Markov Decision Process (MDP): A mathematical model that helps in decision-making where outcomes are partly random and partly under the agents control.

3. Basic Framework of Reinforcement Learning: An agent interacts with an environment, gets rewards, and learns to choose actions that maximize long-term reward.

4. Challenges of Reinforcement Learning: It can be slow to learn, needs lots of data, and struggles with changing environments or long-term goals.

5. Dynamic Programming Algorithms for Reinforcement Learning: These use known models of the environment to compute the best actions using methods like value iteration and policy iteration.

6. Q-Learning and Deep Q-Networks (DQN): Q-Learning is a method to learn the value of actions; DQNs use deep neural networks to scale it to complex problems.

7. Deep Q Recurrent Networks: These combine Q-learning with memory (recurrent networks) to handle sequences and partial observability.

8. Simple Reinforcement Learning for Tic-Tac-Toe: A basic example where an agent learns to win or draw by playing many games and learning from rewards.

9. Deep Generative Models: These models learn to create new data similar to the training data (like fake images or text).

10. Boltzmann Machine: A network of neurons that learns patterns by simulating energy levels of configurations.

11. Deep Belief Networks (DBN): A stack of Boltzmann machines trained layer-by-layer to learn deep features in data.

12. Generative Adversarial Network (GAN): Consists of a generator that makes fake data and a discriminator that tries to detect itboth improve through competition.

13. Discriminator Network: Tries to tell real data from fake data made by the generator.

14. Generator Network: Tries to produce fake data that looks like the real thing.

15. Types of GAN: Variants like DCGAN, CycleGAN, and StyleGAN modify GANs for different data types or tasks.

16. Applications of GAN Networks: Used for generating realistic images, videos, voices, and even data for training AI models.

17. Unfolding Computational Graphs: Shows how a recurrent network works across time steps by unrolling it into a

feed-forward structure.

18. Recurrent Neural Networks (RNNs): Neural networks with loops that let them remember information over timeuseful for sequences.

19. Bidirectional RNNs: Processes sequences both forward and backward to get more context.

20. Encoder-Decoder Sequence-to-Sequence Architectures: One part of the network encodes input into a summary; another decodes it to produce output (used in translation).

21. Deep Recurrent Networks: RNNs with multiple layers for learning complex patterns in sequences.

22. Recursive Neural Networks: Apply the same set of weights recursively on a hierarchical input (like trees).

23. Challenge of Long-Term Dependencies: RNNs struggle to remember distant information due to vanishing or exploding gradients.

24. Echo State Networks: A type of RNN with fixed random connections and a trained readout layer.

25. Leaky Units and Strategies for Multiple Time Scales: Help networks learn both short- and long-term dependencies by slowing down how quickly they forget.

26. Long Short-Term Memory (LSTM) and Gated RNNs: Special RNNs that use gates to control memory and overcome long-term memory issues.

27. Optimization for Long-Term Dependencies: Techniques like better initialization or gradient clipping to help learning over long sequences.

28. Explicit Memory: Neural networks with separate memory storage that helps them remember and use past information.

29. Performance Metrics: Used to evaluate model quality (e.g., accuracy, precision, recall).

30. Default Baseline Models: Simple models used to compare if your complex model is really better.

31. Determining Whether to Gather More Data: Check if adding data will improve learning or if the model has hit its performance limit.

32. Selecting Hyperparameters: Choosing the best settings like learning rate or number of layers to improve performance.

33. CNN Architecture Overview: CNNs are designed for image data, using layers to detect patterns and features.

34. Basic Structure of CNN: Padding, Strides, Settings: Padding adds borders, strides control movement, and settings

decide layer details.

35. ReLU Layer: Applies a function that keeps only positive values to help networks learn faster.

36. Pooling: Reduces the size of the data to make the network faster and less sensitive to small changes.

37. Fully Connected Layers: Every neuron connects to every neuron in the next layerused at the end of CNNs.

38. Interleaving Between Layers: CNNs alternate between convolution, activation, pooling, and fully connected layers for effective learning.

39. Local Response Normalization: Helps improve generalization by normalizing neuron outputs based on neighbors.

40. Training a Convolutional Network: Use data and backpropagation to adjust weights and learn to classify images.

41. Introduction to Neural Networks: Neural networks are inspired by the brain and used to model complex patterns.

42. Biological Neuron: A nerve cell that fires when it receives enough signalsused as a model for artificial neurons.

43. The Perceptron: The simplest neural network model that makes decisions by weighing inputs.

44. Multilayer Feed-Forward Networks: Neural networks with layers stacked one after anotherno loops.

45. Training Neural Networks: Forward and Backpropagation: Forward pass makes predictions, backpropagation adjusts weights to improve them.

46. Activation Functions: These functions decide how much a neuron fires; different ones suit different tasks.

47. Loss Functions: Measure how far predictions are from true values; help the network learn.

48. Loss for Regression: Used when predicting numbers (e.g., mean squared error).

49. Loss for Classification: Used when predicting classes (e.g., cross-entropy loss).

50. Loss for Reconstruction: Used when trying to recreate input (e.g., in autoencoders).

51. Hyperparameters: Learning Rate, Regularization, etc.: Settings that affect training behavior, like how fast it learns or how much it avoids overfitting.

52. Deep Feedforward Networks XOR Example: Shows how deep networks can solve problems simple ones can't, like the XOR logic problem.

53. Hidden Units: Neurons in the middle layers that learn features from data.

54. Cost Functions: Used to evaluate how wrong the network is and guide learning.

55. Error Backpropagation: Algorithm to adjust weights by moving errors backward through the network.

56. Gradient-Based Learning: Learning by changing weights in the direction that reduces error.

57. Implementing Gradient Descent: A step-by-step method to update weights using gradients.

58. Vanishing and Exploding Gradient Descent: Problems where gradients become too small or large, making learning unstable.

59. Sentiment Analysis: Determines if text expresses positive or negative feelings using machine learning.

60. Deep Learning with PyTorch, Jupyter, Colab: PyTorch is a deep learning library; Jupyter and Colab help write and run code easily.

61. What is Machine Learning and Deep Learning?: Machine learning lets computers learn from data; deep learning uses neural networks for this.

62. Supervised and Unsupervised Learning: Supervised uses labeled data, unsupervised finds patterns without labels.

63. Bias-Variance Tradeoff: Balance between a model being too simple (bias) or too sensitive (variance).

64. Hyperparameters, Under/Overfitting, Regularization: Hyperparameters tune models; regularization prevents overfitting (too closely fitting training data).

65. Limitations of Machine Learning: Needs lots of data, can be biased, and may not explain decisions well.

66. History of Deep Learning: Evolved from neural network research with key breakthroughs in the 2000s.

67. Advantages and Challenges of Deep Learning: Great at recognizing patterns but needs data, power, and can be hard to understand.

68. Learning Representations from Data: Deep learning learns useful features automatically instead of needing hand-crafted ones.

69. How Deep Learning Works in 3 Figures: Layers of neurons extract features step by step to transform raw input to output.

70. Common Architectural Principles of Deep Networks: Includes using many layers, nonlinear activations, and large datasets.

71. Architecture Design: Choosing the number of layers, types, and connections for best results.

72. Applications of Deep Learning: Used in speech, vision, robotics, healthcare, and more.

73. Popular Tools: TensorFlow, Keras, PyTorch, Caffe, Shogun: These are libraries that help build and train deep learning models easily.