DL -NOTES

UNIT-I: INTRODUCTION TO MACHINE LEARNING AND NEURAL NETWORKS

---

1. Introduction to Machine Learning (ML)

- **Definition**: Machine Learning is a subset of AI that enables systems to learn from data and improve performance without being explicitly programmed.
- **Types of ML**:
    - **Supervised Learning**: Learns from labeled data (e.g., classification, regression).
    - **Unsupervised Learning**: Finds hidden patterns in unlabeled data (e.g., clustering).
    - **Reinforcement Learning**: Learns through rewards and punishments (e.g., game playing).
- **Applications**: Spam filtering, image recognition, speech processing, recommendation systems.
- **Key Concepts**:
    - **Model**: Mathematical representation of a process.
    - **Training**: Learning parameters from data.
    - **Generalization**: Model's ability to perform well on unseen data.

---

2. Linear Models

a. Support Vector Machines (SVMs)

- SVMs are supervised learning models used for classification and regression.
- **Concept**: Find the optimal hyperplane that maximally separates classes.
- **Key Features**:
    - Maximizes margin between support vectors.

- Can use kernel trick for non-linear separation.
- Robust to high-dimensional data.

b. Perceptrons

- **Basic Unit of Neural Networks**: A simple binary classifier.
- **Structure**:
  - Takes weighted input, applies activation function (e.g., step function).
- **Learning Rule**:
  - Adjust weights using:

$$w=w+\eta(y-\hat{y})x$$

  - where η is learning rate, y is true label, and y^ is predicted output.
- **Limitation**: Can only solve linearly separable problems.

c. Logistic Regression

- Used for binary classification.
- **Sigmoid Function**:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

- Outputs probability between 0 and 1.
- **Loss Function**: Cross-entropy loss is commonly used.
- Suitable for linearly separable problems; extended to multiclass using softmax.

---

3. Introduction to Neural Networks

a. What a Shallow Network Computes

- **Shallow Neural Network**: Has one hidden layer between input and output.
- **Structure**:
  - **Input Layer**: Features.

- **Hidden Layer**: Applies non-linear transformations.
- **Output Layer**: Gives final prediction.

- Each neuron computes:

$$y = \phi(w \cdot x + b)$$

where ϕ\is an activation function like ReLU or sigmoid.

- **Capabilities**: Can approximate simple non-linear functions.

---

4. Training a Neural Network

a. Loss Functions

- **Purpose**: Measures the error between predicted and actual output.
- **Common Loss Functions**:
  - **Mean Squared Error (MSE)**: For regression.
  - **Cross-Entropy Loss**: For classification.
  - **Hinge Loss**: Used in SVMs.

b. Backpropagation

- **Goal**: Compute gradients of loss with respect to weights.
- **Steps**:
  1. **Forward Pass**: Compute output.
  2. **Backward Pass**: Use chain rule to propagate error backward.
  3. **Gradient Calculation**: Determine how to update weights.
- Updates are applied using gradient descent.

c. Stochastic Gradient Descent (SGD)

- **Gradient Descent**: Updates weights using entire dataset.
- **SGD**: Updates weights using a small batch (even single sample) for efficiency.

- **Update Rule**:

$$w = w - \eta \frac{\partial L}{\partial w}$$

where η is learning rate.

---

5. Neural Networks as Universal Function Approximators

- **Universal Approximation Theorem**: A feedforward neural network with at least one hidden layer and non-linear activation can approximate any continuous function on compact subsets of Rn\mathbb{R}^n, given sufficient neurons.
- **Implications**:
    - Shallow networks can model complex functions.
    - Depth improves efficiency and generalization.
- **Limitations**: May require large number of neurons; training becomes harder as complexity increases.

Summary Table: Linear Models vs Neural Networks

Feature	Linear Models	Neural Networks
Learning Approach	Linear separation	Non-linear transformations
Model Complexity	Simple	Complex, layered
Feature Engineering	Manual	Often automatic
Flexibility	Limited	High (can model any function)
Interpretability	High	Low
Training Speed	Fast	Slower (due to backpropagation)
Data Requirements	Low to moderate	Often requires large datasets

---

UNIT-II: DEEP NETWORKS

---

1. History of Deep Learning

- **1950s-1980s**:
    - **Perceptron** proposed by Frank Rosenblatt (1958).
    - Limitations shown by Minsky and Papert (1969).
- **1980s-1990s**:
    - **Backpropagation** algorithm developed and popularized.
    - Early multi-layer networks trained with gradient descent.
- **2000s-Present**:
    - Advances in **computing power (GPUs)**, **large datasets**, and **efficient algorithms** (e.g., ReLU, dropout).
    - Breakthroughs in vision (ImageNet 2012 by AlexNet) and speech recognition.
- Deep Learning is now a core component of modern AI.

---

2. A Probabilistic Theory of Deep Learning

- Views deep learning through the lens of **Bayesian inference** and **probabilistic models**.
- Neural networks can be interpreted as **hierarchical probabilistic models**.
- **Each layer** in a deep net can be seen as learning a **representation** of the data distribution.
- **Posterior Approximation**: Deep networks can approximate complex posterior distributions through learned features.
- Helps explain **why depth helps generalization** — deeper models better capture complex data distributions.

---

3. Backpropagation and Regularization

a. Backpropagation

- Algorithm for training deep networks via **gradient descent**.
- Uses **chain rule** to compute gradients from output layer back to input layer.
- Efficiently updates all weights in the network.

b. Regularization

- Techniques to reduce **overfitting** and improve generalization.
- Common methods:
    - **L1/L2 Regularization**: Adds penalty on weight magnitudes.
    - **Dropout**: Randomly drops neurons during training.
    - **Early Stopping**: Stops training when validation loss increases.
    - **Data Augmentation**: Expands dataset artificially.

---

4. Batch Normalization

- **Normalizes activations** of a layer to zero mean and unit variance.
- Applied between layers to:
    - Reduce **internal covariate shift**.
    - Allow higher learning rates.
    - Speed up convergence.
- Computation involves:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad y = \gamma \hat{x} + \beta$$

- Parameters γ\gamma and β\beta are learnable.

---

5. VC Dimension and Neural Networks

- **VC (Vapnik–Chervonenkis) Dimension**: Measures capacity/complexity of a model.

- High VC dimension ⇒ High capacity to fit data.
- Neural networks with many parameters have high VC dimension.
- But good **generalization** is still possible due to **regularization** and **data fitting** behavior.

---

6. Deep vs Shallow Networks

Feature	**Shallow Networks**	**Deep Networks**
Layers	1 hidden layer	Multiple hidden layers
Feature Learning	Requires manual feature engineering	Learns hierarchical features automatically
Representation Power	Limited	Higher, can represent complex functions
Training Complexity	Simpler	More complex, needs tuning
Generalization	Lower (for complex tasks)	Higher (with sufficient data)

- Deep networks are more **parameter-efficient** at approximating complex functions.

---

7. Convolutional Neural Networks (CNNs)

- Special type of neural network designed for **grid-like data** (e.g., images).
- **Key Components**:
    - **Convolutional Layers**: Use filters to extract spatial features.
    - **Pooling Layers**: Downsample feature maps (e.g., max pooling).
    - **Fully Connected Layers**: Final layers for classification.
- **Advantages**:
    - **Parameter sharing** (same filter used across image).
    - **Local connectivity** reduces the number of weights.

---

8. Generative Adversarial Networks (GANs)

- Introduced by **Goodfellow et al., 2014**.
- Consist of two neural networks:
    - **Generator (G)**: Tries to produce realistic data.
    - **Discriminator (D)**: Tries to distinguish between real and fake data.
- **Training**: A minimax game:

$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$

- Applications: Image generation, super-resolution, data augmentation.

---

9. Semi-Supervised Learning

- **Definition**: Combines a small amount of labeled data with a large amount of unlabeled data.
- **Why it matters**:
    - Labeling data is expensive.
    - Can improve performance with limited supervision.
- **Common Techniques**:
    - **Self-training**: Model labels unlabeled data and retrains.
    - **Consistency regularization**: Model predictions should be stable to perturbations.
    - **GAN-based SSL**: Use GANs to learn data distribution with limited labels.

UNIT-III: DIMENSIONALITY REDUCTION & CONVOLUTIONAL NETWORKS

---

1. Dimensionality Reduction

a. Purpose:

- Reduce the number of input variables (features) while preserving important information.
- Helps in:
    - Improving model performance.
    - Reducing overfitting.
    - Better visualization of data.

---

2. Linear Techniques

a. Principal Component Analysis (PCA)

- Unsupervised method that finds **directions (principal components)** that maximize variance.
- **Steps**:
    1. Standardize the data.
    2. Compute covariance matrix.
    3. Find eigenvalues and eigenvectors.
    4. Project data on top-k eigenvectors.
- Reduces dimensionality while retaining maximum variance.

b. Linear Discriminant Analysis (LDA)

- Supervised method that maximizes **class separability**.
- Optimizes ratio of **between-class scatter** to **within-class scatter**.
- Projects data to lower dimensions with maximum class separation.

---

3. Manifold Learning

- Non-linear dimensionality reduction techniques.
- Assumes data lies on a low-dimensional manifold within a higher-dimensional space.
- Examples:

    - **t-SNE (t-distributed stochastic neighbor embedding)**.
    - **Isomap**, **Locally Linear Embedding (LLE)**.
- Better for visualizing complex, high-dimensional datasets.

---

4. Metric Learning

- Learns a **distance function** that reflects semantic similarity.
- Applications: Face verification, clustering, information retrieval.
- Algorithms:
    - **Contrastive Loss**, **Triplet Loss** used with Siamese Networks.
    - **Mahalanobis distance** can be learned for better separation.

---

5. Autoencoders & Dimensionality Reduction in Networks

a. Autoencoders

- Neural networks trained to **reconstruct input**.
- Consist of:
    - **Encoder**: Maps input to a lower-dimensional latent space.
    - **Decoder**: Reconstructs input from latent space.
- Loss: Mean Squared Error (MSE).
- Types:
    - **Denoising Autoencoders**
    - **Sparse Autoencoders**
    - **Variational Autoencoders (VAE)**

b. Dimensionality Reduction in Networks

- Autoencoders reduce dimensionality by learning compressed representations.
- Useful for **pre-training**, **feature extraction**, and **noise removal**.

---

6. Introduction to ConvNets (Convolutional Neural Networks)

- Specialized for processing **grid-like data** such as images.
- Consist of:
    - **Convolutional layers**: Apply filters to detect patterns.
    - **Activation functions** (e.g., ReLU).
    - **Pooling layers**: Downsample features.
    - **Fully connected layers**: Final classification.

---

7. CNN Architectures

a. AlexNet (2012)

- 8 layers (5 conv + 3 FC).
- Used ReLU activation and dropout for regularization.
- First model to win ImageNet with deep learning.

b. VGGNet (2014)

- Uses **3×3 convolutions** and **max pooling**.
- Simple and deep (16 or 19 layers).
- Known for its uniform architecture.

c. Inception (GoogLeNet, 2014)

- Introduced **Inception modules**: multiple filters (1×1, 3×3, 5×5) in parallel.
- Improves efficiency by **factorizing convolutions**.
- Uses **global average pooling** at the end.

d. ResNet (2015)

- Introduced **Residual connections (skip connections)**:

$$y = F(x) + x$$

- Solves the vanishing gradient problem.

- Enables training of extremely deep networks (e.g., 50, 101, 152 layers).

---

8. Training a ConvNet

a. Weight Initialization

- Proper initialization avoids vanishing/exploding gradients.
- Common methods:
    - **Xavier Initialization**: For sigmoid/tanh.
    - **He Initialization**: For ReLU activations.

b. Batch Normalization

- Normalizes layer inputs to have zero mean and unit variance.
- Stabilizes learning, allows higher learning rates.

c. Hyperparameter Optimization

- Key hyperparameters:
    - Learning rate
    - Batch size
    - Number of layers/filters
    - Dropout rate
    - Optimizer (e.g., SGD, Adam)
- Methods:
    - **Grid Search**
    - **Random Search**
    - **Bayesian Optimization**
- **Early stopping** and **cross-validation** used for better tuning.

Summary Table: Linear vs Non-linear Dimensionality Reduction

Technique	Type	Supervised	Handles Non-linearity	Example Algorithms
PCA	Linear	No	No	Principal Component Analysis
LDA	Linear	Yes	No	Linear Discriminant Analysis
t-SNE, Isomap	Non-linear	No	Yes	Manifold learning techniques
Autoencoders	Non-linear	No	Yes	Neural Network based encoding

🔷 1. Principal Component Analysis (PCA)

Definition:

- PCA is an **unsupervised linear dimensionality reduction technique**.
- It transforms the data to a new coordinate system such that the greatest variance lies on the first principal component, the second greatest on the second component, and so on.

Key Concepts:

- **Variance**: PCA retains directions with the highest variance.
- **Orthogonality**: Principal components are orthogonal (uncorrelated).
- **Eigenvectors & Eigenvalues**: Directions (eigenvectors) of maximum variance; magnitude (eigenvalues) tells how important that direction is.

Steps:

1. Standardize the data.
2. Compute the **covariance matrix**.
3. Compute **eigenvalues and eigenvectors**.
4. Sort eigenvectors by decreasing eigenvalues.
5. Choose top-k components to project the data.

Applications:

- Noise reduction
- Visualization (2D or 3D)
- Preprocessing for ML algorithms

🔷 2. Linear Discriminant Analysis (LDA)

Definition:

- LDA is a **supervised linear dimensionality reduction** technique that aims to find a feature space that maximizes class separability.

Key Concepts:

- Maximizes the **ratio of between-class variance to within-class variance**.
- Projects data onto a line (or hyperplane) that best separates multiple classes.

Steps:

1. Compute the **mean vectors** of each class.
2. Compute **within-class scatter** and **between-class scatter** matrices.
3. Compute **eigenvalues and eigenvectors** of SW−1SBS_W^{-1} S_B.
4. Sort eigenvectors by eigenvalues and select top-k.

Applications:

- Face recognition (e.g., Fisherfaces)
- Classification problems with labeled data
- Feature extraction in supervised learning

🔷 3. Autoencoders

Definition:

- An **autoencoder is a type of neural network** used to learn compressed representations of input data (unsupervised learning).

Structure:

- **Encoder**: Compresses input into a lower-dimensional latent space.
- **Decoder**: Reconstructs the input from the compressed representation.

Architecture:

Input → [Encoder] → Bottleneck (latent space) → [Decoder] → Reconstructed Output

Loss Function:

- Usually Mean Squared Error (MSE):

$$\mathcal{L} = \|x - \hat{x}\|^2$$

Types of Autoencoders:

Type	Description
Denoising Autoencoder	Learns to reconstruct input from a corrupted version
Sparse Autoencoder	Adds sparsity constraint on hidden activations
Variational Autoencoder (VAE)	Learns probabilistic latent space, suitable for generation

Applications:

- Data compression
- Anomaly detection
- Pre-training deep networks
- Image denoising

🔶 **Comparison Table: PCA vs LDA vs Autoencoder**

Feature	PCA	LDA	Autoencoder
Supervision	Unsupervised	Supervised	Unsupervised
Linearity	Linear	Linear	Non-linear (if deep network used)

Feature	PCA	LDA	Autoencoder
Goal	Maximize variance	Maximize class separation	Minimize reconstruction loss
Outputs	Principal Components	Discriminant Axes	Encoded Latent Representation
Class Label Use	No	Yes	No
Flexibility	Limited	Limited	High (with deep nets)
Interpretability	High	Moderate	Low

✅ UNIT-IV: OPTIMIZATION AND GENERALIZATION

🔷 1. Optimization in Deep Learning

- Deep learning models are trained by **minimizing a loss function** using optimization algorithms.
- Most optimization methods rely on **gradient descent**.

🔹 **Common Techniques:**

- **Gradient Descent**:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

- **Variants**:
  - **Stochastic Gradient Descent (SGD)**
  - **Mini-batch Gradient Descent**
  - **Momentum**
  - **Adam (Adaptive Moment Estimation)**
  - **RMSProp**

🔷 **2. Non-convex Optimization for Deep Networks**

- Loss surfaces in deep networks are **non-convex**, meaning:
  - Multiple **local minima**, **saddle points**, and **flat regions**.
- **Challenges**:
  - Hard to find global minimum.
  - Stuck in saddle points or poor local minima.
- **Solutions**:
  - Use of SGD with momentum.
  - **Batch normalization** smooths the landscape.
  - **Good initialization** helps avoid bad regions.

🔷 **3. Stochastic Optimization**

- Instead of computing gradient on full dataset, it uses **random samples** (mini-batches).
- **Benefits**:
  - Faster convergence.
  - Introduces noise → helps escape local minima.
- Common optimizers: **SGD**, **Adam**, **RMSProp**.

🔷 **4. Generalization in Neural Networks**

- **Generalization**: Ability of a model to perform well on unseen data.
- Neural networks generalize well **despite being overparameterized**.

🔹 **Factors Affecting Generalization:**

- **Model complexity**
- **Amount and quality of training data**

- **Regularization** (dropout, weight decay)
- **Early stopping**
- **Data augmentation**

🔹 **Implicit regularization:**

- Optimization methods like **SGD** act as regularizers even without explicit penalties.

🔷 **5. Spatial Transformer Networks (STN)**

- Allow neural networks to **learn spatial transformations** of input data.
- Introduced by Jaderberg et al., 2015.

🔹 **Key Components:**

1. **Localization Network**: Learns parameters of transformation.
2. **Grid Generator**: Creates sampling grid.
3. **Sampler**: Applies grid to input and produces transformed output.

- Makes networks more robust to translation, scaling, rotation.

🔷 **6. Recurrent Networks & LSTM**

🔹 **Recurrent Neural Networks (RNN):**

- Designed for **sequential data** (e.g., time series, language).
- Maintains **hidden state** that captures past information:

$$h_t = f(W x_t + U h_{t-1} + b)$$

🔹 **Limitations:**

- **Vanishing/exploding gradients** during backpropagation through time (BPTT).

🔹 **Long Short-Term Memory (LSTM):**

- Solves vanishing gradient problem using **gates**:
    - **Forget Gate**: Decides what to discard.

- **Input Gate**: Decides what to update.
- **Output Gate**: Decides what to output.

- LSTM cell maintains a **cell state** for long-term memory.

🔷 7. Recurrent Neural Network Language Models

- RNNs used to model **probability distribution over sequences of words**.
- For word sequence w1,w2,...,wTw_1, w_2, \dots, w_T:

$$P(w_1, w_2, ..., w_T) = \prod_{t=1}^{T} P(w_t | w_1, ..., w_{t-1})$$

- RNNs capture long-term dependencies better than traditional n-gram models.

🔷 8. Word-Level RNNs

- Input is **one word at a time** (often represented using word embeddings).
- Used for:
  - Language modeling
  - Text generation
  - Sentiment analysis

🔹 **Training:**

- Predict next word given previous sequence.
- Use **cross-entropy loss** for training.

🔷 9. Deep Reinforcement Learning

- Combines deep learning with reinforcement learning (RL).
- **Agent learns** to take actions to maximize cumulative reward.

- **Components:**
  - **Environment**
  - **Agent**
  - **Policy**: Maps state to actions.
  - **Reward Function**
- **Key Architectures:**
  - **DQN (Deep Q-Network)**: Uses CNN to approximate Q-value.
  - **Policy Gradient Methods**: Learn the policy directly.
- **Applications:**
  - Game playing (e.g., AlphaGo)
  - Robotics
  - Autonomous driving

---

10. Computational and Artificial Neuroscience

- **Computational Neuroscience:**
  - Studies **biological neural systems** using mathematical models.
  - Focuses on:
    - Spiking neurons
    - Neural coding
    - Brain simulation
- **Artificial Neuroscience:**
  - Inspired by how the **brain processes information**.
  - Develops algorithms like **neural networks**, **synaptic plasticity**, **unsupervised learning** (e.g., Hebbian learning).
  - Bridges biology and machine learning.

✅ Summary Table

Topic	Key Idea
Optimization	Minimize loss using gradient-based methods
Non-convex Optimization	Loss surface has local minima, saddle points
Stochastic Optimization	Uses mini-batches; faster and more robust
Generalization	Model's ability to perform on unseen data
STN	Learns spatial transformations
RNN	Handles sequences via recurrent connections
LSTM	Solves vanishing gradient with memory gates
RNN Language Models	Predict next word using RNNs
Word-level RNN	Input/output is at word level
Deep RL	Learns optimal actions using rewards
Computational Neuroscience	Models biological neurons and brain behavior
Artificial Neuroscience	Builds brain-inspired AI systems

✅ UNIT-V: CASE STUDIES AND APPLICATIONS

🔷 1. ImageNet

- **ImageNet** is a large visual database designed for use in visual object recognition research.
- Contains **over 14 million labeled images** across **20,000+ categories**.
- Powered the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**:
    - Landmark for progress in **image classification and object detection**.
    - **AlexNet (2012)** was the first deep CNN to win, triggering the deep learning revolution.

🔷 2. Object Detection

- **Goal**: Identify and locate objects in an image.
- Outputs: Bounding boxes + class labels.

🔹 Key Methods:

- **R-CNN (Region-based CNN)**: Uses region proposals + CNN features.
- **Fast R-CNN / Faster R-CNN**: Improves speed by integrating region proposals into the network.
- **YOLO (You Only Look Once)**: Real-time object detection using a single CNN.
- **SSD (Single Shot Detector)**: Detects objects in one pass, faster than R-CNN.

🔷 3. Audio – WaveNet

- **WaveNet**: A deep generative model for raw audio signals, introduced by DeepMind.
- **Autoregressive model**: Predicts each audio sample based on previous samples.
- Uses **dilated causal convolutions** for receptive field growth.

🔹 Features:

- Produces high-fidelity human-like speech.
- Used in text-to-speech systems (e.g., Google Assistant).
- Can be extended to music and other audio tasks.

🔷 4. Natural Language Processing (NLP)

🔹 Word2Vec:

- A neural network model that learns **vector representations of words** (embeddings).
- Words with similar meaning are closer in vector space.

Architectures:

- **CBOW (Continuous Bag of Words)**: Predicts target word from surrounding context.
- **Skip-gram**: Predicts surrounding words from a given word.

🔹 **Applications:**

- Sentiment analysis
- Machine translation
- Information retrieval
- Question answering

🔷 **5. Joint Detection**

- Combines multiple detection tasks (e.g., object + attribute detection).
- Example: Detect object **and** its location **and** its semantic properties.
- Used in:
  - **Multi-task learning**
  - **Context-aware models**
  - **Panoptic segmentation**

🔷 **6. Bioinformatics**

- Application of deep learning in **biology and medical science**.

🔹 **Use Cases:**

- **DNA sequence analysis**
- **Protein structure prediction** (e.g., AlphaFold)
- **Medical imaging** (X-ray, MRI classification)
- **Drug discovery** using molecule representations
- Uses CNNs, RNNs, and GNNs.

🔷 7. Face Recognition

- Identify or verify a person from an image.

🔹 **Techniques:**

- **FaceNet**, **DeepFace**, and **OpenFace**:
  - Learn embeddings using **Triplet Loss** or **Contrastive Loss**.
- **Applications**:
  - Security and surveillance
  - Social media tagging
  - Access control

🔷 8. Scene Understanding

- High-level interpretation of visual scenes:
  - Objects, actions, relationships, and context.

🔹 **Key Tasks:**

- **Semantic segmentation**: Label every pixel with a class.
- **Instance segmentation**: Segment each object instance separately.
- **Panoptic segmentation**: Combines semantic and instance segmentation.
- **Image captioning**, **visual question answering** also fall under this domain.

🔷 9. Image Captioning

- Generate textual descriptions for images.
- Combines **CNN (for image feature extraction)** + **RNN/LSTM (for language generation)**.

🔹 **Workflow:**

1. Use CNN (e.g., ResNet) to extract image features.
2. Feed features to LSTM to generate sentence word-by-word.

3. Trained using datasets like **MS COCO**.

🔹 **Applications:**

- Assistive technology for the visually impaired
- Automatic tagging in social media
- Content-based image retrieval

✅ **Textbooks and Resources**

Book Title	Author(s)
Deep Learning	Ian Goodfellow, Yoshua Bengio, Aaron Courville
Neural Networks and Deep Learning	Michael Nielsen (Free online book)
Pattern Recognition and Machine Learning	Christopher M. Bishop
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow	Aurélien Géron

---

✅ **Summary Table of Applications**

Application	Deep Learning Method	Key Model/Approach
Image Classification	CNN	AlexNet, VGG, ResNet
Object Detection	CNN + Region Proposals	YOLO, R-CNN, SSD
Audio Generation	Causal CNNs	WaveNet
NLP	Embeddings + RNN/LSTM/Transformers	Word2Vec, BERT, GPT
Face Recognition	Metric Learning	FaceNet, DeepFace
Scene Understanding	CNN + RNN	Mask R-CNN, DenseNet
Image Captioning	CNN + LSTM	Show & Tell, Show & Attend

Application	Deep Learning Method	Key Model/Approach
Bioinformatics	CNN, RNN, Graph Nets	AlphaFold, DeepBind