UNIT-5 INTERACTIVE APPLICATIONS OF DEEP LEARNING AND DEEP LEARNING RESEARCH

Topics:

Interactive Applications of Deep Learning:	Deep Learning Research:
➢ Machine Vision ➢ Natural Language Processing ➢ Generative Adversarial Networks ➢ Deep Reinforcement Learning	➢ Autoencoders ➢ Deep Generative Models (e.g., Boltzmann Machines, Restricted Boltzmann Machines, Deep Belief Networks)

Machine Vision:

- Machine vision is sometimes called with the term computer vision
- The main applications of deep learning can be divided into computer vision, natural language processing (NLP), and reinforcement learning.
- Computer vision is a field of artificial intelligence (AI) that focuses on enabling computers to "see" and interpret images and videos.
- Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized computer vision, enabling machines to understand and interpret visual data with remarkable accuracy.
- Machine vision systems replicate human vision for automation and industrial tasks using hardware and software to capture and analyze visual data, enabling machines to make informed decisions.
- Deep learning in machine vision uses algorithms like CNNs and Vision Transformers to analyze images and videos for tasks such as object detection, 3D vision, and pattern recognition. Essential components include cameras, sensors, lenses, lighting, and vision processors.

Components of Machine Vision Systems:

Machine vision systems consist of several crucial components such as,

1. Hardware Components:

Machine vision hardware includes lighting for clear image capture, staging mechanisms to hold objects steady, lenses to focus light onto sensors, cameras to convert light into digital signals, and image sensors like CMOS or CCD to create digital images.

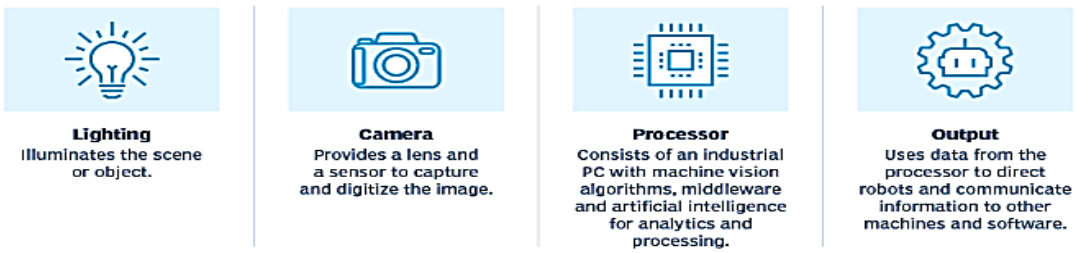
Lighting: Proper illumination is crucial for capturing clear images and enabling accurate analysis.

Staging: A staging mechanism holds objects in place for image capture and processing.

Lenses: Lenses focus light onto image sensors, capturing visual data.

Cameras: Cameras contain image sensors that convert light into digital signals (pixels).

Image Sensors: These sensors, typically CMOS or CCD, capture the light and convert it into a digital image.



2. Software and Algorithms:

Machine vision software utilizes deep learning algorithms like CNNs, RNNs, GANs, and Vision Transformers for tasks such as object detection and image segmentation. Image processing algorithms analyze captured images to identify objects and make decisions. Deep learning models handle feature extraction automatically, minimizing manual intervention. Pattern recognition algorithms classify objects and patterns effectively.

Deep Learning Algorithms:
CNNs, RNNs, GANs, and Vision Transformers are commonly used for tasks like object detection, image segmentation, and feature extraction.
Image Processing: Software algorithms process the captured images to extract features, identify objects, and make decisions based on pre-defined criteria.
Feature Extraction: Deep learning models learn to automatically extract relevant features from images, reducing the need for manual feature engineering
Pattern Recognition: Machine vision systems utilize pattern recognition algorithms to identify and classify objects or patterns in images
3. Deep Learning in Machine Vision:
Object Detection: Deep learning models are used for accurately identifying and locating objects within images.
3D Vision: Deep learning can be used to create 3D models and maps from 2D images, enabling tasks like 3D object recognition and robotic navigation.
Defect Detection: Deep learning models can be trained to automatically detect defects in products, improving quality control and reducing waste.
Augmented Reality: Machine vision can be combined with augmented reality to overlay information on real-world images.
Industrial Vision Systems: Deep learning is used in various industrial applications, such as quality control, defect detection, and robotic control.

Applications of machine vision:

Machine vision has a wide range of applications across industries, including:

- **Manufacturing and Quality Control**: Detecting defects, ensuring consistency, and automating inspections on production lines.
- **Healthcare:** Analyzing medical images for diagnosis, such as detecting tumors or identifying abnormalities in X-rays and MRIs.
- **Automotive:** Enabling autonomous vehicles to process surroundings through object recognition, lane detection, and navigation systems.
- **Retail:** Automated checkout systems that recognize products without barcodes, as well as inventory management using vision-based solutions.
- **Agriculture:** Monitoring crops for diseases, sorting produce, and optimizing harvesting processes.
- **Security and Surveillance:** Facial recognition, intruder detection, and activity monitoring in real-time.
- **Robotics:** Guiding robots in tasks like assembly, sorting, and navigation using visual inputs.

Natural Language Processing (NLP):

- In deep learning, Natural Language Processing (NLP) plays a crucial role in enabling computers to understand, interpret, and generate human language
- NLP integrates computer science, AI, and linguistics to help computers understand, process, and generate human language.
- It is crucial for analyzing text and automating tasks like language translation and sentiment analysis, especially with the rise of text data from various sources.
- Deep learning has transformed NLP by enabling accurate and flexible models for tasks like language translation and sentiment analysis.
- Unlike rule-based methods that depend on human expertise, deep learning models learn from large text datasets, capturing complex word relationships and generalizing them to new examples. This advancement has improved model accuracy and enabled applications like chatbots, virtual assistants, and text summarization.

Understanding Neural Network Architectures for NLP:

Neural networks in NLP include RNNs for sequential text processing and transformers with self-attention for global word relationships. They enable advanced models like GPT-3 for accurate and diverse language tasks.

Here is an overview of some of the most commonly used architectures:

- Convolutional Neural Networks (CNNs)**:** Initially developed for image processing, CNNs are adapted for NLP tasks like text classification and sentiment analysis, using convolution and pooling layers to extract and classify features.
- Recurrent Neural Networks (RNNs)**:** Designed for sequential data, RNNs process inputs one at a time, maintaining hidden states to capture long-term dependencies, with LSTMs excelling in tasks like language modeling and machine translation.
- Transformers: A recent architecture using self-attention to capture global word relationships, allowing parallel processing. Models like BERT and GPT-3 excel in tasks like text classification and language generation.
- Recursive Neural Networks (Recursive NNs): Suitable for hierarchical data, Recursive NNs apply neural modules to nodes in tree structures, making them effective for sentiment analysis and relation extraction.
- Hybrid Models: combine architectures like CNNs, RNNs, and attention mechanisms (e.g., HAN) to capture both local and global text relationships for complex NLP tasks.

NLP Techniques:

NLP encompasses a wide array of techniques that are aimed at enabling computers to process and understand human language. These tasks can be categorized into several broad areas, each addressing different aspects of language processing.

Here are some of the key NLP techniques:

1. Text Processing and Preprocessing:

- Tokenization: Dividing text into smaller units, such as words or sentences.
- Stemming and Lemmatization: Reducing words to their base or root forms.
- Stop word Removal: Removing common words (like “and”, “the”, and “is”) that may not carry significant meaning.
- Text Normalization: Standardizing text, including case normalization, removing punctuation, and correcting spelling errors.

2. Syntax and Parsing:

- Part-of-Speech (POS) Tagging: Assigning parts of speech to each word in a sentence (e.g., noun, verb, adjective).
- Dependency Parsing: Analyzing the grammatical structure of a sentence to identify relationships between words.
- Constituency Parsing: Breaking down a sentence into its constituent parts or phrases (e.g., noun phrases, verb phrases).

3. Semantic Analysis:

- Named Entity Recognition (NER): Identifying and classifying entities in text, such as names of people organizations, locations, dates, etc.
- Word Sense Disambiguation (WSD): Determining which meaning of a word is used in a given context.
- Coreference Resolution: Identifying when different words refer to the same entity in a text (e.g., “he” refers to “John”).

4. Information Extraction:

- Entity Extraction: Identifying specific entities and their relationships within the text.
- Relation Extraction: Identifying and categorizing the relationships between entities in a text.

5. Text Classification:

- Sentiment Analysis: Determining the sentiment or emotional tone expressed in a text (e.g., positive, negative, neutral).
- Topic Modeling: Identifying topics or themes within a large collection of documents.
- Spam Detection: Classifying text as spam or not spam.

6. Language Generation:

- Machine Translation: Translating text from one language to another.
- Text Summarization: Producing a concise summary of a larger text.
- Text Generation: Automatically generating coherent and contextually relevant text.

7. Speech Processing:

- Speech Recognition: Converting spoken language into text.
- Text-to-Speech (TTS) Synthesis: Converting written text into spoken language.

8. Question Answering:

- Retrieval-Based QA: Finding and returning the most relevant text passage in response to a query.
- Generative QA: Generating an answer based on the information available in a text corpus.

9. Dialogue Systems:

- Chatbots and Virtual Assistants: Enabling systems to engage in conversations with users, providing responses, and performing tasks based on user input.

10. Sentiment and Emotion Analysis:

- Emotion Detection: Identifying and categorizing emotions expressed in text.
- Opinion Mining: Analyzing opinions or reviews to understand public sentiment toward products, services, or topics.

How Natural Language Processing (NLP) Works:

1. Text Input and Data Collection
2. Text Preprocessing
3. Text Representation
4. Feature Extraction
5. Model Selection and Training
6. Model Deployment and Inference
7. Evaluation and Optimization

Applications of Natural Language Processing (NLP):

- Spam Filters
- Algorithmic Trading
- Questions Answering
- Summarizing Information

Important Natural Language Processing (NLP) Models:

- Over the years, many NLP models have made waves within the AI community, and some have even made headlines in the mainstream news.
- The most famous of these have been chatbots and language models.
- Here are some of them:
  - Eliza
  - Tay
  - BERT
  - Language Model for Dialogue Applications (LaMDA)
  - Generative Pre-Trained Transformer 3 (GPT-3)

Generative Adversarial Networks (GANs):

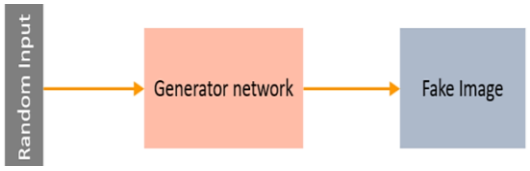
Generative Adversarial Networks (GANs) are machine learning models with two competing networks: a generator that creates synthetic data and a discriminator that evaluates its authenticity, improving the generator's output quality. Generative Adversarial Networks (GANs) were introduced in 2014 by Ian J. Goodfellow and co-authors.

Generator:

A Generator in GANs is a neural network that creates fake data to be trained on the discriminator. It learns to generate plausible data. The generated examples/instances become negative training examples for the discriminator. It takes a fixed-length random vector carrying noise as input and generates a sample.

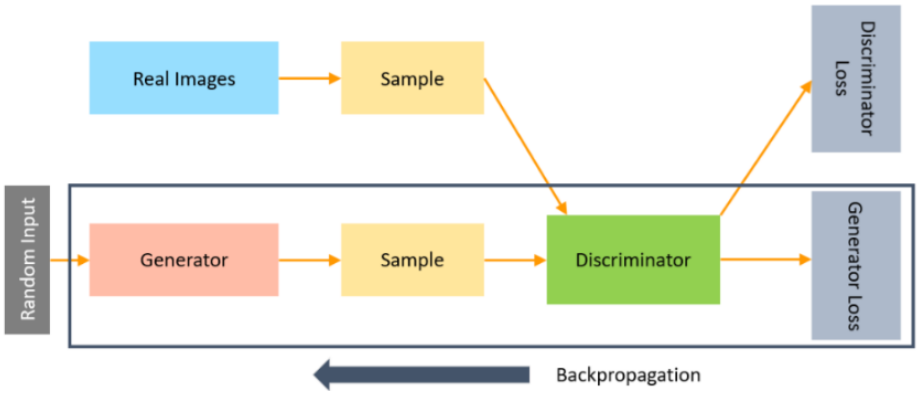
The main aim of the Generator is to make the discriminator classify its output as real.

The part of the GAN that trains the Generator includes:

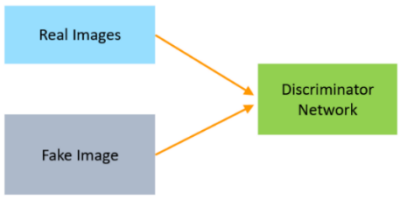


- noisy input vector
- generator network, which transforms the random input into a data instance
- discriminator network, which classifies the generated data
- generator loss, which penalizes the Generator for failing to dolt the discriminator

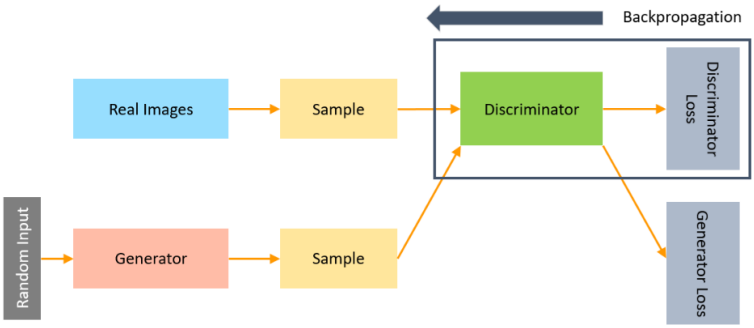
The backpropagation method is used to adjust each weight in the right direction by calculating the weight's impact on the output. It is also used to obtain gradients and these gradients can help change the generator weights.



Discriminator:



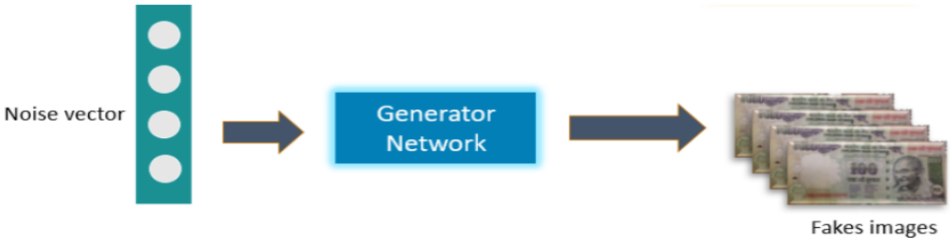
- The Discriminator is a neural network that identifies real data from the fake data created by the Generator. The discriminator's training data comes from two different sources:
- The discriminator uses real data instances, such as pictures of birds, humans, currency notes, etc., as positive samples during training.
- The fake data instances created by the Generator are used as negative examples during the training process. While training the discriminator, it connects to two loss functions.
- During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.
- In the process of training the discriminator, the discriminator classifies both real data and fake data from the generator. The discriminator loss penalizes the discriminator for misclassifying a real data instance as fake or a fake data instance as real.



- The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

GANs Working:

- GANs consist of two neural networks. There is a Generator G(x) and a Discriminator D(x). Both of them play an adversarial game. The generator aims to fool the discriminator by producing data that are similar to those in the training set.
- The discriminator will try not to be fooled by identifying fake data from real data. Both of them work simultaneously to learn and train complex data like audio, video, or image files.
- The Generator network takes a sample and generates a fake sample of data. The Generator is trained to increase the Discriminator network's probability of making mistakes.
- Below is an example of a GAN trying to identify if the 100-rupee notes are real or fake. So, first, a noise vector or the input vector is fed to the Generator network. The generator creates fake 100-rupee notes.
- The real images of 100-rupee notes stored in a database are passed to the discriminator along with the fake notes. The Discriminator then identifies the notes classifying them as real or fake.



We train the model, calculate the loss function at the end of the discriminator network, and backpropagate the loss into both discriminator and generator models.

Mathematical Equation:

Here,

G = Generator

D = Discriminator

Pdata(x) = distribution of real data

p(z) = distribution of generator

x = sample from Pdata(x)

z = sample from P(z)

D(x) = Discriminator network

G(z) = Generator network

The mathematical formula for working on GANs can be represented as:

$$V(D, G) = E_{x \sim Pdata(x)} [\log D(x)] + E_{z \sim p(z)} [\log(1 - D(G(z))]$$

Steps for Training GAN:

1. Define the problem
2. Choose the architecture of GAN
3. Train discriminator on real data
4. Generate fake inputs for the generator
5. Train discriminators on fake data
6. Train the generator with the output of the discriminator

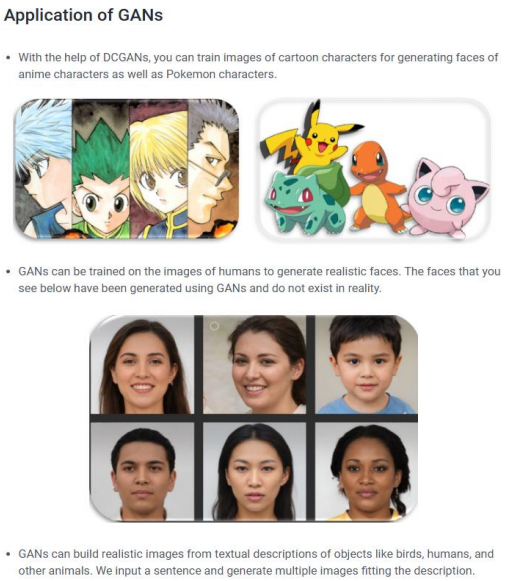
Types of generative adversarial networks:

1. Vanilla GAN
2. Conditional GAN
3. Deep convolutional GAN
4. Super-resolution GAN
5. Laplacian Pyramid GAN
6. StyleGAN
7. CycleGAN

GAN examples/Applications:

GANs are used to generate a wide range of data types, including images, music, and text. The following are popular real-world examples of GANs:

- Generating human faces: GANs can produce accurate representations of human faces. For example, StyleGAN2 from Nvidia can produce photorealistic images of people who don't exist. These pictures are so lifelike that many people believe they're real individuals.
- Developing new fashion designs: GANs can be used to create new fashion designs that reflect existing ones. For instance, clothing retailer H&M uses GANs to create new apparel designs for its merchandise.
- Generating realistic animal images: GANs can also generate realistic images of animals. For example, BigGAN, a GAN



model developed by Google researchers, can produce high-quality images of animals such as birds and dogs.

- Creating video game characters: GANs can be used to create new characters for video games. For example, Nvidia created new characters using GANs for the well-known video game Final Fantasy XV.
- Generating realistic 3D objects: GANs are also capable of producing actual 3D objects. For example, researchers at MIT have used GANs to create 3D models of chairs and other furniture that appear to have been created by people. These models can be applied to architectural visualization or video games.

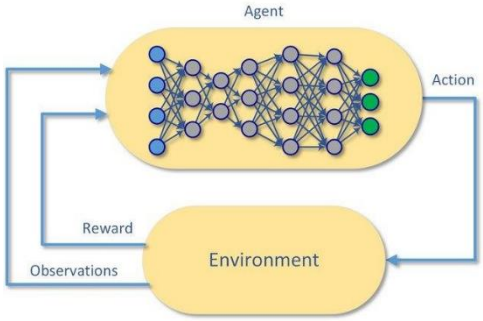
Deep Reinforcement Learning (DRL):

- Deep Reinforcement Learning (DRL) combines reinforcement learning with deep learning, enabling agents to make decisions in complex environments by learning from rewards and penalties.
- It uses deep neural networks to learn sophisticated policies and value functions, allowing agents to solve problems without needing explicit state space engineering.
- Reinforcement Learning: It's a learning framework where an agent interacts with an environment, makes decisions (actions), and learns from rewards or penalties to maximize overall rewards.

How Does DRL Combine RL and Deep Learning: DRL uses neural networks to represent the agent's decision-making strategy (policy) and/or reward predictions (value function). It enables learning in high-dimensional, unstructured environments.

Core components of Deep Reinforcement Learning (DRL):

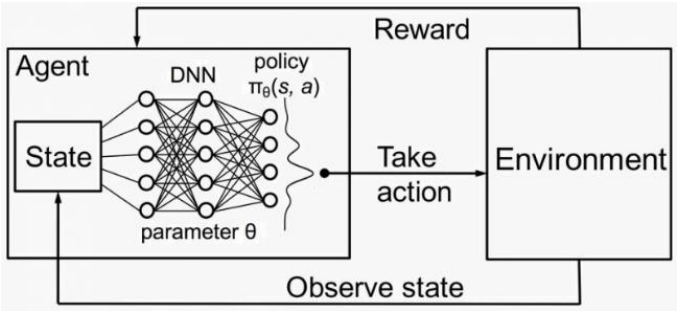
- Agent: The decision-maker interacts with the environment to learn from experiences and improve decision-making.
- Environment: External system providing feedback (rewards or penalties) based on the agent's actions.
- State: The current situation or condition of the environment at any moment.
- Action: The choice made by the agent to alter the environment's state, guided by its policy.
- Reward: Feedback signal indicating how desirable the agent's action was in a given state.
- Policy: A strategy that maps states to actions to maximize cumulative rewards.



- Value Function: Estimates the expected cumulative reward from a state while following a specific policy.
- Model: Simulates the environment to predict outcomes of actions for planning.
- Exploration-Exploitation Strategy: Balances learning new actions (exploration) and optimizing known actions for rewards (exploitation).
- Learning Algorithm: Adjusts the policy or value function based on experiences, e.g., Q-learning, policy gradients.
- Deep Neural Networks: Approximate complex input-to-output mappings for high-dimensional states and actions.
- Experience Replay: Reuses past experiences during training to improve learning stability.
- These elements work together to create powerful learning systems for diverse applications.

How Deep Reinforcement Learning Works:

- Initialization: Set up the agent and define the problem.
- Interaction: The agent performs actions in the environment, receiving states and rewards as feedback.
- Learning: Based on its experiences, the agent updates its strategy for decision-making.
- Policy Update: Algorithms refine the agent's strategy to improve performance.



- Exploration-Exploitation: The agent balances trying new actions (exploration) and sticking to known successful ones (exploitation).
- Reward Maximization: The agent learns to choose actions that maximize cumulative rewards.
- Convergence: Over time, the agent stabilizes its strategy and performs optimally.
- Evaluation: The agent is tested in new or unknown environments to assess its performance.
- Practical Use: The trained agent is deployed in real-world tasks.

Real-World Example: Self-Driving Cars

- Problem: Teach a car to navigate complex traffic environments.
- Implementation: The agent (car) interacts with its environment (road) by sensing its surroundings using cameras and LIDAR. Actions like steering, braking, and accelerating are taken based on the current state. Rewards are given for safe driving, following traffic rules, and efficient navigation, while penalties are applied for errors like collisions. Through iterations, the agent learns to optimize its driving strategy.
- Outcome: The car becomes capable of making safe and reliable decisions, enabling autonomo us navigation.

DRL algorithms:

Some prominent DRL algorithms include Deep Q-Networks (DQNs), Double Deep Q-Learning (DDQL), Deep Deterministic Policy Gradient (DDPG), and Trust Region Policy Optimization (TRPO).

1. Deep Q-Networks (DQNs): Uses deep neural networks to approximate the Q-function. Learns optimal policies for environments with high-dimensional state spaces.
2. Double Deep Q-Learning (DDQL): Addresses overestimation bias in DQN using a separate target network. Improves stability and performance.
3. Deep Deterministic Policy Gradient (DDPG): Suitable for continuous action spaces.
4. Uses actor-critic architecture: the actor learns the policy, and the critic evaluates it.
5. Trust Region Policy Optimization (TRPO): Constrains policy updates within a trust region for stability. Robust in learning complex policies for challenging environments. These algorithms have advanced DRL's ability to tackle complex, real-world tasks across diverse domains.

Applications:

- Robotics: Training robots to perform complex tasks like grasping objects, walking, or assembling products.
- Example: Boston Dynamics uses RL to improve robot agility and adaptability.
- Gaming: Creating AI agents that can play video games at superhuman levels.
- Example: DeepMind's AlphaGo and Alpha Zero, which mastered Go, Chess, and Shogi.
- Autonomous Vehicles: Enhancing decision-making for self-driving cars, such as lane changes, obstacle avoidance, and route optimization.
- Example: Waymo and Tesla use RL for autonomous driving systems.
- Healthcare: Optimizing treatment plans, drug discovery, and robotic surgery.
- Example: RL is used to personalize cancer treatment schedules.
- Finance: Portfolio management, algorithmic trading, and fraud detection.
- Example: RL algorithms are used to predict stock market trends and optimize investments.
- Energy Management: Optimizing energy consumption in smart grids and buildings.
- Example: Google DeepMind reduced energy usage in data centers using RL.
- Manufacturing: Improving production efficiency, predictive maintenance, and quality control.
- Example: RL is used in assembly lines to optimize workflows.

Autoencoders:

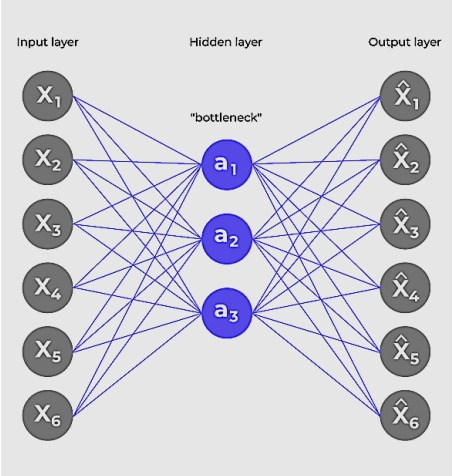
- Autoencoders are a type of deep learning algorithm that is designed to receive an input and transform it into a different representation. They play an important part in image construction. Let's learn about autoencoders in detail.
- Autoencoders are very useful in the field of unsupervised machine learning. You can use them to compress the data and reduce its dimensionality.
- An Autoencoder is a type of neural network that can learn to reconstruct images, text, and other data from compressed versions of themselves.

An Autoencoder consists of three layers:

- Encoder
- Code/Latent Space/Bottleneck
- Decode

Encoder:

The encoder is the part of the network that takes the input data and compresses it into a smaller lower-dimensional representation. Input Layer: This is where the original data enters the network e.g., an image or a set of features Hidden Layers: These layers apply transformations to the input data. The encoder's goal is to extract essential features and reduce the data's dimensionality.



Output of Encoder (Latent Space): The encoder outputs a compressed version of the data often called the latent representation or encoding. This is a condensed version of the input retaining only the important features.

Code/Latent Space/Bottleneck:

The bottleneck is the smallest layer of the network where the data is represented in its most compressed form. It's often referred to as the latent space or code. This layer contains a reduced set of features representing the most important information from the input. The idea is that through this compression the network learns the key patterns and structures of the input data.

Decoder:

The decoder is responsible for taking the compressed representation from the latent space and reconstructing it back into the original data form.

Hidden Layers: The decoder uses a series of layers to gradually expand the compressed data back to the original input's dimensions. Output Layer: This layer produces the reconstructed data and aims to closely resemble the input data.

In summary, we can say that The Encoder layer compresses the input image into a smaller latent space, reducing its dimensions but distorting it slightly. The Code layer holds this compressed representation and passes it to the decoder. The Decoder layer reconstructs the original-sized image from the latent space, but the output is a lossy version of the original.

Types of autoencoders:

1. Undercomplete Autoencoders: Compress input data into a latent space (bottleneck) and reconstruct the same data. Useful for creating a compressed representation of data that can be decompressed later.
2. Sparse Autoencoders: Limit neuron activation in hidden layers to regularize the network. Use L1 Loss (adds magnitude) or KL-divergence (constraints average activation) as penalties to reduce excessive activation.
3. Contractive Autoencoders: Employ bottleneck representations and include regularization terms. Prevents the network from simply copying input to output by minimizing activation derivatives concerning the input.
4. Denoising Autoencoders: Reconstruct clean data from noisy input. Effective for removing noise automatically using L1/L2 loss functions.
5. Variational Autoencoders (VAEs): Represent latent space as a probability distribution, ensuring continuity for interpolation. Ideal for sampling and generating data like images or text.

Applications:

- Dimensionality Reduction: Compress high-dimensional data into lower dimensions for visualization or preprocessing.
- Feature Extraction: Extract meaningful features for tasks like classification and clustering.
- Anomaly Detection: Identify deviations from normal patterns, such as fraud detection or network security.
- Data Denoising: Remove noise from images, audio, or other data types.
- Image Compression: Reduce image size while preserving essential features.
- Generative Modeling: Create new data samples, such as generating realistic images or text.
- Medical Imaging: Analyze and compress medical scans like MRI or CT images for efficient storage and processing.

- Natural Language Processing (NLP): Learn embeddings for text data to improve sentiment analysis or text classification.
- Recommender Systems: Predict user preferences by learning latent representations of user-item interactions.

Deep Generative Models (e.g., Boltzmann Machines, Restricted Boltzmann Machines, Deep Belief Networks):

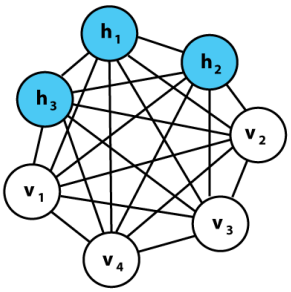
Deep Generative Models e.g., Boltzmann Machines, Restricted Boltzmann Machines, and Deep Belief Networks these models are foundational in the evolution of deep learning and generative modeling, paving the way for more advanced architectures like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs).

- Deep Generative Models (DGMs) are a category of generative models that use deep neural networks to learn the distribution of training data and then generate new data points from that learned distribution.
- They are crucial in deep learning for tasks like generating new images, text, or audio, and for understanding complex data patterns.

Boltzmann machine:

A Boltzmann machine is an unsupervised deep-learning model in which every node is connected to every other node. It is a type of recurrent neural network, and the nodes make binary decisions with some level of bias.

- These machines are not deterministic deep learning models, they are stochastic or generative deep learning models. They are representations of a system.
- A Boltzmann machine has two kinds of nodes.
    - Visible nodes: These are nodes that can be measured and are measured.
    - Hidden nodes: These are nodes that cannot be measured or are not measured.



- According to some experts, a Boltzmann machine can be called a stochastic Hopfield network which has hidden units. It has a network of units with an 'energy' defined for the overall network.
- Boltzmann machines seek to reach thermal equilibrium. It essentially looks to optimize the global distribution of energy. However, the temperature and energy of the system are relative to the laws of thermodynamics and are not literal.
- A Boltzmann machine is made up of a learning algorithm that enables it to discover interesting features in datasets composed of binary vectors. The learning algorithm tends to be slow in networks that have many layers of feature detectors but it is possible to make it faster by implementing a learning layer of feature detectors.
- They use stochastic binary units to reach probability distribution equilibrium (to minimize energy). It is possible to get multiple Boltzmann machines to collaborate to form far more sophisticated systems like deep belief networks.
- The Boltzmann machine is named after Ludwig Boltzmann, an Austrian scientist who came up with the Boltzmann distribution. However, this type of network was first developed by Geoff Hinton, a Stanford Scientist.

Boltzmann Distribution:

A probability distribution gives the likelihood of a system being in a specific state, based on its energy and temperature. Formulated by Ludwig Boltzmann in 1868, also called the Gibbs distribution.

Uses of Boltzmann Machines:

Optimizes solutions by learning patterns, structures, and mappings in data.

Useful for unsupervised learning tasks like clustering, dimensionality reduction, anomaly detection, and generative modeling. Widely applied in imaging, image processing, and solving complex quantum or statistical physics problems.

How Boltzmann Machines Work:

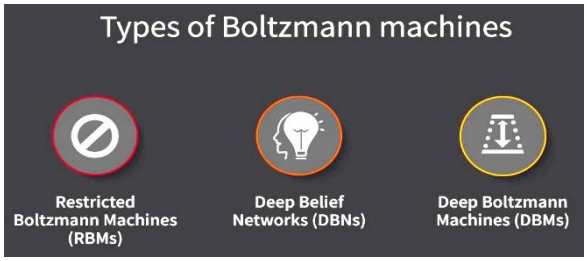
Nodes: Include visible (measured) and hidden (latent) nodes.

Connections: Every node is connected to all others, enabling information sharing.

Process: Learns data patterns without deterministic outputs using stochastic gradient descent.

Captures parameters and correlations to self-generate subsequent data, making it a type of deep generative model under unsupervised learning.

Types of Boltzmann machines:



There are three types of Boltzmann machines. These are:

1. Restricted Boltzmann Machines (RBMs)
2. Deep Belief Networks (DBNs)
3. Deep Boltzmann Machines (DBMs)

1. Restricted Boltzmann Machines (RBMs):

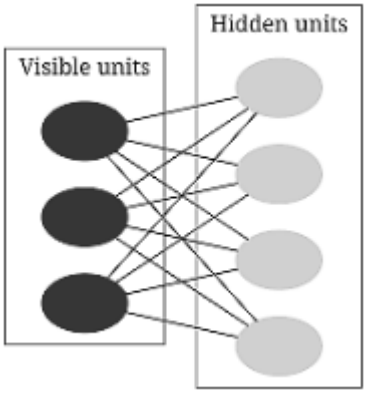
- A type of neural network used for unsupervised learning and generative modeling.
- Consists of two layers: the visible layer (input data) and the hidden layer (learned features).
- Called "restricted" because neurons within the same layer are not connected.
- Trained using contrastive divergence, a variant of stochastic gradient descent.
- While in a full Boltzmann machine, all the nodes are connected and the connections grow exponentially, an RBM has certain restrictions concerning node connections.

Boltzmann Machines:

A network where all neurons are interconnected, consisting of visible (input) and hidden layers.
No output layer, making them stochastic generative models.
Optimize internal representations and solve combinatorial problems.
Based on Boltzmann distribution (Gibbs Distribution), related to statistical mechanics and thermodynamics.
Developed by Geoffrey Hinton and Terry Sejnowski in 1985.
In a Restricted Boltzmann Machine, hidden nodes cannot be connected while visible nodes are connected.
Structure: RBMs consist of two layers: a visible layer (representing the input data) and a hidden layer (representing latent features).
Connections: There are no connections between neurons within the same layer (visible-to-visible or hidden-to-hidden).
Functionality: RBMs learn a probability distribution over the input data by learning the relationships between the visible and hidden units.

Restricted Boltzmann Machines (RBMs) work:

Phases of RBM Operation:
Feed Forward Pass: The input layer activates the hidden layer using weights and biases.
Positive Association: Strong link between visible and hidden units.
Negative Association: Weak link between visible and hidden units.
Feed Backward Pass: The hidden layer reconstructs the input layer from activated hidden states.
Calculates error: Error = Reconstructed Input - Actual Input.
Adjusts weights: Adjust Weight = Input × Error × Learning Rate.
How it Learns:
By identifying patterns responsible for activating hidden neurons, the RBM can backtrack and match characteristics of similar inputs.



Applications:

- RBMs are used for dimensionality reduction, feature learning, and generating new data similar to the input. Computer vision, natural language processing, speech recognition, and as part of architectures l ike Deep Belief Networks

Types of RBMs:

There are mainly two types of Restricted Boltzmann Machine (RBM) based on the types of varia bles they use:
Binary RBM: In a binary RBM, the input and hidden units are binary variables. Binary RBMs are often used in modeling binary data such as images or text.

Gaussian RBM: In a Gaussian RBM, the input and hidden units are continuous variables that follow a Gaussian distribution. Gaussian RBMs are often used in modeling continuous data such as audio signals or sensor data.

2. Deep Belief Networks (DBNs):

- In a Deep Belief Network, you could say that multiple Restricted Boltzmann Machines are stacked, such that the outputs of the first RBM are the inputs of the subsequent RBM.
- The connections within individual layers are undirected, while the connections between layers are directed. However, there is an exception here. The connection between the top two layers is undirected.
- A deep belief network can either be trained using a Greedy Layer-wise Training Algorithm or a Wake-Sleep Algorithm.

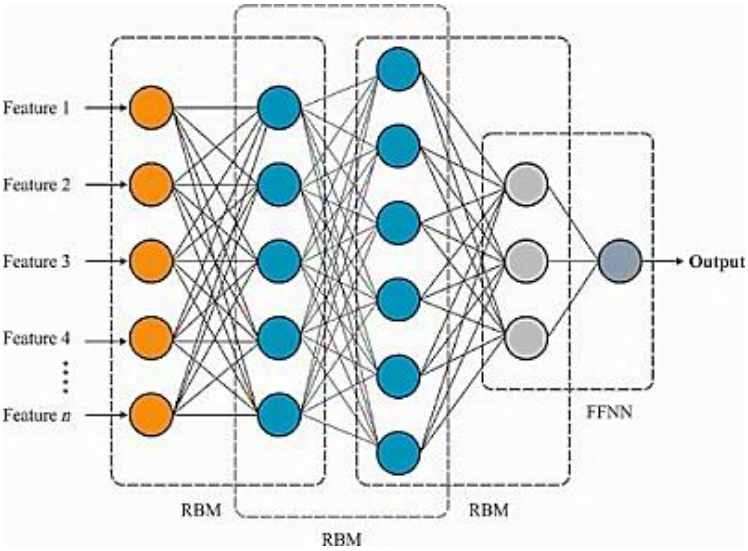
Structure: DBNs are built by stacking multiple RBMs on top of each other, creating a deep network with multiple hidden layers.

Functionality: Each RBM learns to represent the input of the previous layer, enabling the network to learn hierarchical features.

Training: DBNs can be trained in a greedy layer-by-layer manner, with each RBM trained individually, or in a fine-tuning stage using supervised learning.

How Does DBN Work?

- Getting from pixels to property layers is not a straightforward process.
- First, we train a property layer that can directly gain pixel input signals. Then we learn the features of the preliminarily attained features by treating the values of this sub-caste as pixels.
- The lower bound on the log-liability of the training data set improves every time a fresh sub-caste of parcels or features that we added to the network. The deep belief network's operational pipeline is as follows:
- First, we run numerous steps of Gibbs sampling in the top two hidden layers.
- The top two hidden layers define the RBM. Thus, this stage effectively extracts a sample from it.
- Then generate a sample from the visible units using a single pass of ancestral sampling through the rest of the model.
- Finally, we'll use a single bottom-up pass to infer the values of the latent variables in each layer. In the bottom layer, greedy pretraining begins with an observed data vector. It then oppositely fine-tunes the generative weights.



Applications:

- DBNs are used for feature extraction, classification, and generating complex data representations.
- Deep Belief Networks work similarly to deep neural networks and can handle large datasets and various data types.

- Therefore, they are great for tasks like:
  - Image Classification.
  - Text Classification.
  - Image Generation.
  - Speech Recognition.

3. Deep Boltzmann Machines (DBMs):

Deep Boltzmann Machines are very similar to Deep Belief Networks. The difference between these two types of Boltzmann machines is that while connections between layers in DBNs are directed, in DBMs, the connections within layers, as well as the connections between the layers, are all undirected.

Structure: DBMs also have multiple hidden layers, but unlike DBNs, they have bidirectional connections between all layers.

Functionality: DBMs learn a probability distribution over the data by capturing complex relationships between all units in the network.

Training: DBMs are trained using a combination of unsupervised and supervised learning techniques, such as contrastive divergence and backpropagation.

What are DBMs:

DBMs are a type of neural network that learns complex internal representations from data, especially from limited labeled data inputs. They can be thought of as stacking multiple Restricted Boltzmann Machines (RBMs) to create deeper networks.

How they work:

DBMs consist of multiple layers of hidden units that capture probabilities of various patterns in the data. The "deep" aspect refers to the multiple layers, allowing for a deeper understanding of the data. Each layer captures increasingly abstract representations.

Key features:

Unsupervised learning: They learn features without explicit labels.

Feature extraction: They can extract hidden features from data, making them suitable for tasks like object or speech recognition.

Data generation: They can learn to produce new data examples similar to the original data.

Applications:

DBMs are used for generative modeling, where they learn to generate new data samples that are similar to the input data.

- Speech recognition: DBMs can learn complex speech patterns and representations.
- Object recognition: They can learn to identify objects in images by capturing their underlying features.
- Text analysis: DBMs can be used for tasks like language modeling and sentiment analysis.
- Data synthesis and augmentation: They can generate new data samples, which is useful for tasks like training machine learning models with limited data.
- Molecular modeling: DBMs have been used in bioinformatics for tasks like protein structure prediction.

Advantages:

Complex feature extraction: DBMs can capture intricate relationships and dependencies in data.

Generative capabilities: They can generate new data samples, which is useful for various applications.

Disadvantages:

Computational complexity: DBMs can be computationally expensive to train, especially for large datasets.

Training challenges: Exact maximum likelihood learning is intractable for DBMs, making it necessary to use approximate methods.