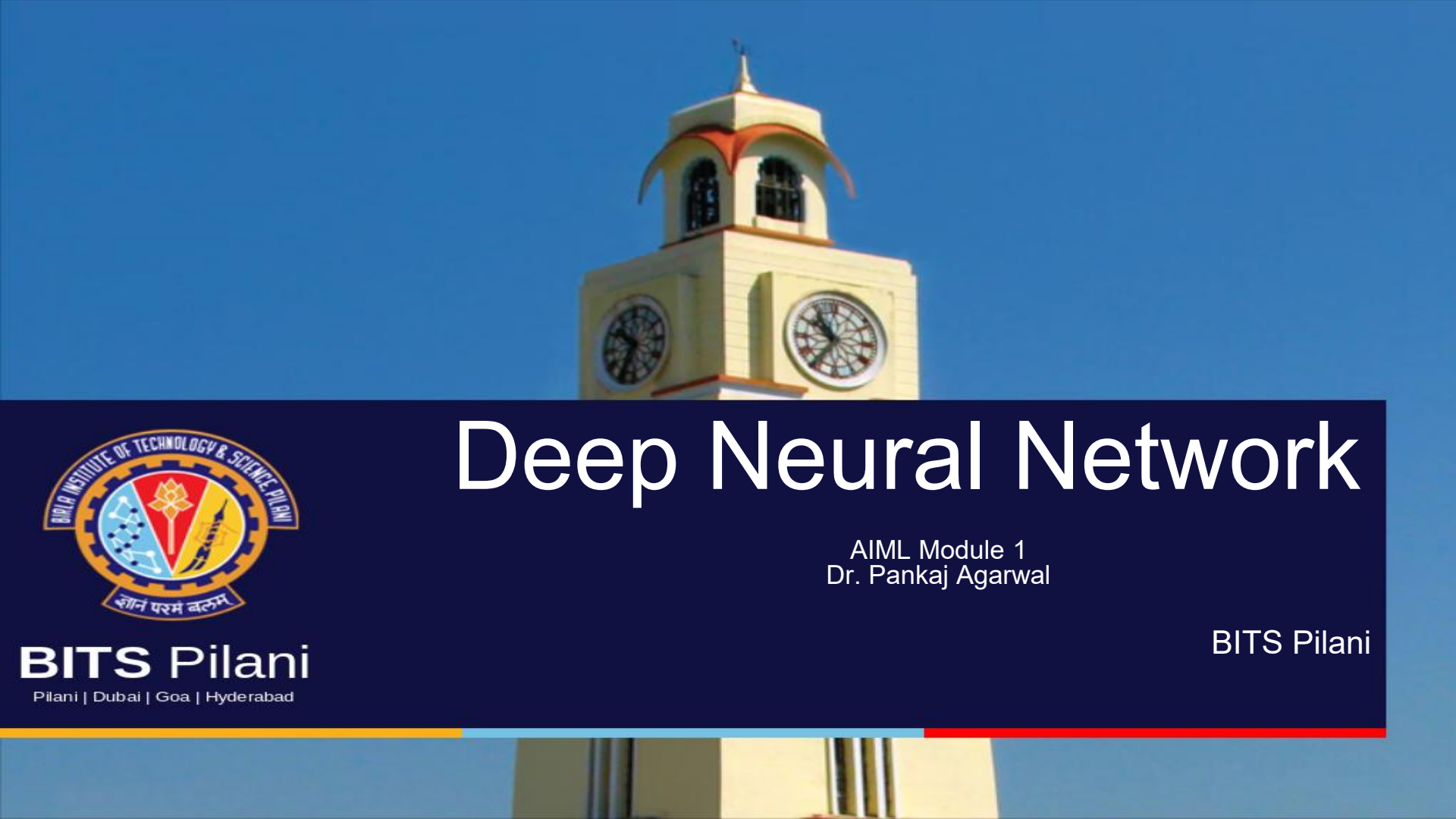


Deep Neural Network

AIML Module 1
Dr. Pankaj Agarwal

BITS Pilani

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

The Lecture Materials gratefully acknowledge the authors who made their course materials freely available online.

Course Logistics

What we Learn…. (Module Structure)

1. Fundamentals of Neural Network
2. Multilayer Perceptron
3. Deep Feedforward Neural Network
4. Improve the DNN performance by Optimization and Regularization
5. Convolutional Neural Networks
6. Sequence Models
7. Attention Mechanism
8. Neural Network Search
9. Time series Modelling and Forecasting
10. Other Learning Techniques

Books

Textbook

- Dive into Deep Learning by Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola. https://d2l.ai/chapter_introduction/index.html

Reference book

- Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville https://www.deeplearningbook.org/

Evaluation Components and Schedule

	Evaluation Component	Mode	Duration	Weightage	Date and Time
EC1	Quiz I*	Online	30 Mins	5%	Refer to LMS
	Quiz II*		30 Mins	5%	
	Assignment 01		4 Weeks	10%	
	Assignment 02		4 Weeks	15%	
EC2	Mid-sem exam (Closed book)		2 Hours	30%	
EC3	Comprehensive exam (Open book)		2.5 Hours	40%	

* Best of Two

Lab Sessions

- L1 Introduction to PyTorch
- L2 Deep Neural Network with Back-propagation and optimization
- L3 CNN
- L4 RNN
- L5 LSTM
- L6 Auto-encoders

Course Logistics

- Refer Taxila for the following
  - Handout
  - Schedule for Webinar
  - Schedule of Quiz, and Assignments.
  - Evaluation scheme
  - Session Slide Deck
  - Demo Lab Sheets
  - Quiz-I, Quiz-II
  - Assignment-I, Assignment-II
  - Sample QPs
- Lecture Recordings
  - Available on Microsoft teams

Honour Code

All submissions for graded components must be the result of your original effort. It is strictly prohibited to copy and paste verbatim from any sources, whether online or from your peers. The use of unauthorized sources or materials, as well as collusion or unauthorized collaboration to gain an unfair advantage, is also strictly prohibited. Please note that we will not distinguish between the person sharing their resources and the one receiving them for plagiarism, and the consequences will apply to both parties equally.

In cases where suspicious circumstances arise, such as identical verbatim answers or a significant overlap of unreasonable similarities in a set of submissions, will be investigated, and severe punishments will be imposed on all those found guilty of plagiarism.

In case of Queries regarding the course….

Step 1: Post in the discussion forum.

- Read through the existing post and if you find any topic similar to your concern, add on to the existing discussion.
- Avoid duplication of queries or issues.

Step 2: Email the IC at bharatesh.c@wilp.bits-pilani.ac.in if the query or issue is not resolved within 1 week. Turn around for a response to the email is 48 hours.

- In the subject pl mention the phrase ”DNN” clearly.
- Use BITS email id for correspondence. Emails from personal emails will be ignored without any reply.

PATIENCE is highly APPRECIATED

What is Deep Learning?

What is Deep Learning?

- Deep Learning is a type of **machine learning** based on **artificial neural networks** in which multiple layers of processing are used to extract progressively higher-level features from large volumes of data.
- Inspired by the human brain, these networks are designed to recognize complex patterns in unstructured data like images, text, and sound.
- Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: **learn by example**.
- Deep learning is a subset of machine learning, which is essentially a neural network with **three or more layers**.
- Deep Learning gets its name from the fact that we add more "**Layers**" to learn from the data.
- Deep learning powers many advanced artificial intelligence (AI) applications used today, such as virtual assistants and self-driving cars.

What is Deep Learning?

It uses structures called **Artificial Neural Networks (ANNs)** — systems made up of many connected layers of “neurons” (small computing units) that process information.

Each layer learns something **different** from the data:

- The first layers learn **basic details**,
- The middle layers learn **patterns or shapes**,
- The last layers learn **complex ideas or decisions**.

The more layers we have, the “**deeper**” the network becomes — that’s why it’s called **Deep Learning**.

How It Works (Step by Step)

Imagine teaching a child to recognize a **dog**:
1.The first time, the child looks at pictures — they notice simple things: eyes, ears, legs.
2.After seeing more examples, they start recognizing patterns — “dogs usually have fur, a tail, and bark.”
3.Later, they can tell the difference between a **dog**, a **cat**, and even **different breeds** of dogs.

Deep Learning does the same thing — it **learns by example**, improving as it sees more data.

Real-World Examples of Deep Learning

innovate achieve lead

Healthcare (Medical Image Diagnosis)
Deep learning models can analyze **X-rays**, **CT scans**, or **MRI images** to detect diseases such as cancer or pneumonia.

•Example: A deep learning system might learn to recognize **lung cancer** by studying thousands of X-ray images labeled as “normal” or “cancerous.”

•Layer by layer, it learns to detect shapes, tissues, and patterns related to tumors — often as accurately as human doctors.

Self-Driving Cars
Cars with **autonomous driving systems** use deep learning to “see” the road.

- One network identifies **objects** (pedestrians, traffic lights, other vehicles),
- Another predicts **movement** and decides when to **stop or turn**.
- Example: Tesla’s autopilot uses deep neural networks to process video data from cameras in real time.

Voice Assistants (Speech Recognition)
When you say “Hey Siri” or “Okay Google,” a deep learning model converts your **voice** into **text**.
•The model has learned from millions of speech samples to understand **different accents, tones, and speeds**.

Facial Recognition
Deep learning helps identify faces in photos or videos.
•Example: Your smartphone unlocking using **Face ID** works through a deep neural network that analyzes your facial features from multiple angles.

Where in AI sits DL?

AI is a general field that encompasses machine learning and deep learning, but that also includes many more approaches that don't involve any learning.

Artificial Intelligence (AI) — The Big Umbrella

•**AI** is the **broadest field** — it includes *any* technique that makes machines behave intelligently.

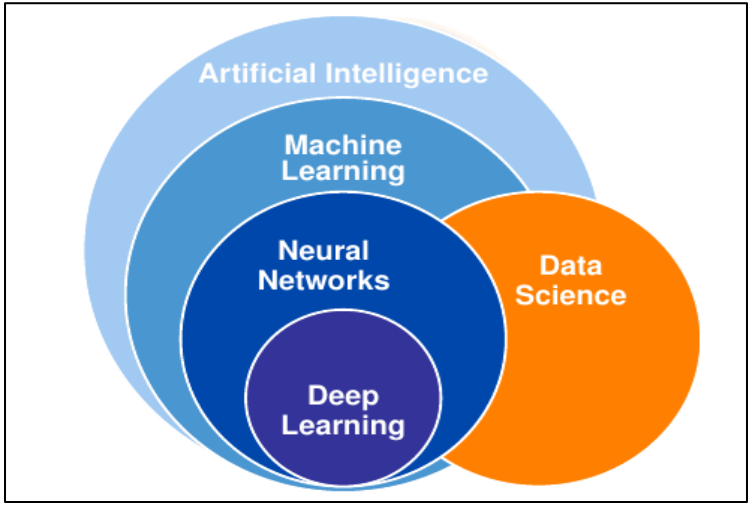
This can include:

- Rule-based systems (if-then logic)
- Expert systems
- Machine learning (learning from data)
- Robotics, natural language processing, computer vision, etc.

•In short, **AI = the science of making computers think and act like humans**.

Neural Networks (NN) — The Core Concept

- **Neural Networks** are ML models inspired by how the **human brain** works.
- They consist of **layers of neurons (nodes)** connected together that process data.
- They are especially useful for recognizing **patterns** like images, sounds, or text.



Machine Learning (ML) — A Subset of AI

- **ML** is a **way of achieving AI** — instead of hardcoding rules, the computer learns patterns from **data**.
- ML algorithms **learn from experience**, improving automatically with more data.

Where in AI sits DL?

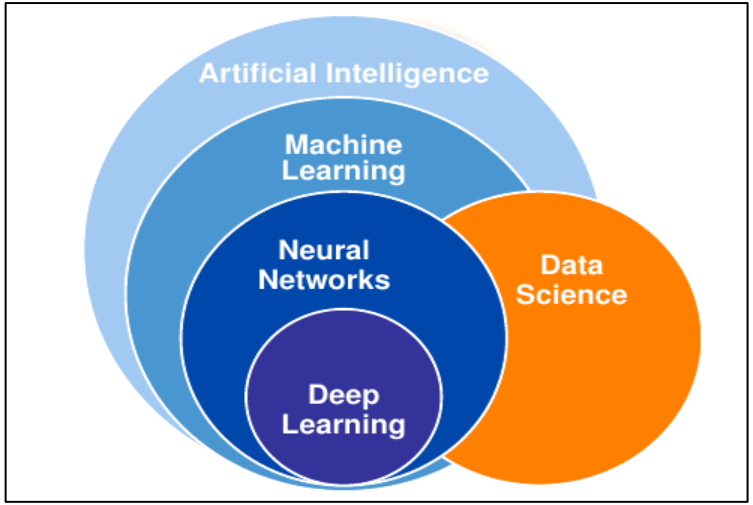
Data Science — The Overlapping Field

•**Data Science** overlaps with AI and ML but has a broader focus on **analyzing and interpreting data** to make business or scientific decisions.

•It combines:

- **Statistics**
- **Machine Learning**
- **Data visualization**
- **Programming (e.g., Python, R)**

•Data Science uses AI and ML tools, but its main goal is to **gain insights and predictions from data**, not necessarily to build "intelligent" systems.



Deep Learning (DL) is a part of **Neural Networks (NN)**, which is part of **Machine Learning (ML)**, which itself is part of **Artificial Intelligence (AI)**.
Data Science overlaps with all these — it uses AI and ML techniques to make data-driven decisions.

ML VS Deep Learning

What is Machine Learning (ML)?

Machine Learning is a subset of **Artificial Intelligence (AI)** that allows computers to **learn from data** and **improve their performance** over time **without being explicitly programmed**.

- **Core idea:**

ML systems **find patterns in data** and use those patterns to **make predictions or decisions**.

- **How it works:**

1.You feed data (inputs and outputs) into an algorithm.
2.The algorithm learns a rule or relationship between them.
3.It uses that rule to make predictions on new, unseen data.

- **Example:**

- Predicting **house prices** using features like area, location, and rooms.
- Classifying **emails** as spam or not spam.

Common ML Algorithms:

Linear Regression, Decision Trees / Random Forest
Support Vector Machines (SVM), K-Means Clustering
Naive Bayes, Gradient Boosting / XGBoost

What is Deep Learning (DL)?

Deep Learning is a subset of **Machine Learning** that uses **Artificial Neural Networks (ANNs)** — algorithms inspired by the structure of the **human brain**.

- **Core idea:**

Instead of manually extracting features, deep learning **automatically learns representations** from raw data (images, text, audio, etc.) using **multiple layers (hence "deep")**.

- **How it works:**

1.Raw data is passed through **multiple layers of neurons**.
2.Each layer learns increasingly **complex patterns** — edges → shapes → objects → meaning.
3.The final layer gives predictions (e.g., "this is a cat").

- **Example:**

•Recognizing **faces** in photos.
•Translating **languages** automatically.
•Detecting **tumors** from medical scans.

Common Deep Learning Models:

- CNN (Convolutional Neural Networks) – for images & vision.
- RNN (Recurrent Neural Networks) – for sequences & time series.
- LSTM / GRU – for text or speech data.
- Transformers (e.g., BERT, GPT) – for NLP and generative AI.

ML VS Deep Learning

Machine Learning Pipeline:

DATA → FEATURE EXTRACTION → MODEL (e.g., Decision Tree, SVM) → OUTPUT

Human helps the system by deciding which features matter

Example : Predicting House Prices

➤ **Machine Learning**

You collect data like:

- Size of the house, number of bedrooms, location, age, etc.
- You **manually select** important features (e.g., square footage, location).
- Then use algorithms like **Linear Regression** or **Random Forest**.

💡 *ML needs a human to decide what features matter.*

Deep Learning Pipeline:

DATA (e.g., images, audio) → NEURAL NETWORK (multiple layers) → OUTPUT

The network learns which features are important automatically.

Deep Learning

•You feed in raw data such as **images of houses**, **text descriptions**, and **numerical data**.

- A **deep neural network** automatically figures out that features like:
- Lawn condition, number of floors, and design aesthetics affect price.

💡 *DL learns features directly from data without human input.*

Why Deep Learning?

Why Deep Learning?

- Large amounts of data
- Lots and lots of unstructured data like images, text, audio, video
- Cheap, high-quality sensors
- Cheap computation - CPU, GPU, Distributed clusters
- Cheap data storage
- Learn by examples
- Automated feature generation
- Better learning capabilities
- Scalability
- Advance analytics can be applied

Why Deep Learning?

Deep Learning (DL) is used because it can automatically learn complex patterns and representations from large amounts of raw data — like images, speech, text, or videos — without needing humans to manually define the features.

Traditional **Machine Learning** needs you to tell it *what to look for*.
Deep Learning, however, **finds the right features on its own** — it learns directly from data.

Example: Medical Image Diagnosis

- In older ML methods, radiologists had to define features like "tumor size," "shape," or "texture."
- With **Deep Learning**, you feed thousands of X-rays or MRI scans directly to a neural network.
- The network automatically learns to detect patterns of diseases — often more accurately than humans.

💡 **Impact:** Early and accurate detection of diseases like lung cancer, diabetic retinopathy, or pneumonia.

Why Deep Learning?

📈 **2. Deep Learning Handles Big & Complex Data**

- Modern life produces **huge amounts of data** — images, videos, speech, sensor data, etc.
- Traditional ML can't handle this complexity efficiently.
- Deep Learning thrives on **big data** and learns from raw, unstructured information.

🚗 **Example: Self-Driving Cars**

- A self-driving car takes input from multiple cameras, LiDAR sensors, and radar.
- A deep neural network processes all this raw data in real-time:
  - Detects pedestrians, vehicles, traffic lights, and road signs.
  - Decides when to stop, accelerate, or turn.

💡 **Impact:** Safe autonomous driving through real-time perception and decision-making

Why Deep Learning?

3. Deep Learning Understands Natural Language

Human language is **complex, ambiguous, and contextual** — something traditional ML models struggle with.

Deep Learning models (especially **Transformers**, like those behind ChatGPT or Google Translate) can capture **meaning, tone, and context**.

💬 **Example: Virtual Assistants & Chatbots**

•Assistants like **Alexa**, **Siri**, and **Google Assistant** use deep learning to:

- Understand your voice commands ("Play my favorite song")
- Process natural language
- Respond conversationally

Impact: Machines can now communicate naturally and contextually with humans.

Why Deep Learning?

4. Deep Learning Creates New Things (Generative AI)

Deep Learning isn't just about recognizing — it can also **create** new content:

•Text, images, videos, music, and even code.

Example: Generative AI (DALL·E, Midjourney, ChatGPT)

- DALL·E can generate an image from a text prompt like:
  "A doctor robot diagnosing a patient in a futuristic hospital."
- ChatGPT (a deep learning language model) can write essays, code, or explain concepts.

💡 **Impact:** New creative possibilities in design, education, and entertainment.

Why Deep Learning?

5. Deep Learning Outperforms Humans in Accuracy
In many fields, DL models have reached or even **surpassed human-level accuracy**.

👁️ **Example: Face Recognition**

- Facebook, Apple, and security systems use deep learning for facial recognition.
- It can identify people even with different lighting, poses, or angles — much faster and more accurately than traditional ML.

💡 **Impact:** Used in security systems, smartphone unlocking (Face ID), and law enforcement.

Why Deep Learning?

innovate achieve lead

🔄 **6. Deep Learning Keeps Improving with Data**

Unlike older models that reach a performance plateau, **deep learning models improve** as they get **more data** and **more layers**.

🎬 **Example: Netflix or YouTube Recommendation System**

- Deep learning models continuously learn from millions of users' behavior:
  - What you watch
  - How long you watch
  - What you skip
- They adapt and personalize recommendations in real-time.

💡 **Impact:** Highly personalized content experience for every user

Why Deep Learning?

7. Deep Learning Enables Automation

Deep learning powers **intelligent automation** — tasks that once needed human intelligence.

🏭 **Example: Industrial Automation**

- Robots in factories use deep learning for **quality inspection** — detecting tiny defects in parts faster than humans.
- In agriculture, drones use DL to identify **crop health** and **predict yields**.

💡 **Impact:** Smarter, faster, and error-free automation in industries.

Deep Learning VS Generative AI?

Deep Learning (DL):

- A subset of Machine Learning that uses neural networks with many layers to learn patterns from data.
- Learn from data to classify, detect, or predict.
- Models: CNNs, RNNs, LSTMs, DNNs.
- Data Requirements: Large labeled datasets.

Examples

- Detecting pneumonia from X-rays
- Credit card fraud detection
- Self-driving cars
- Siri / Alexa speech-to-text

Generative AI (GenAI):

- A type of AI application built using deep learning models that can create new content — text, images, music, code, or video — that didn't exist before.
- Generative AI is built on top of Deep Learning — it uses special types of deep learning models (like Transformers, VAEs, or GANs) to create new data that looks like real data.
- Learn from data to create new data.
- GANs, VAEs, Transformers (GPT, DALL·E, Stable Diffusion).
- Large unlabeled or semi-labeled datasets.

Examples:

- Generates human-like text and conversations.
- Generates realistic videos or talking avatars.
- Writes or completes code from your natural language instructions.
- Creates images from text prompts (e.g., "A robot doctor treating a cat").
- Creates new, realistic patient data for research without exposing private information.

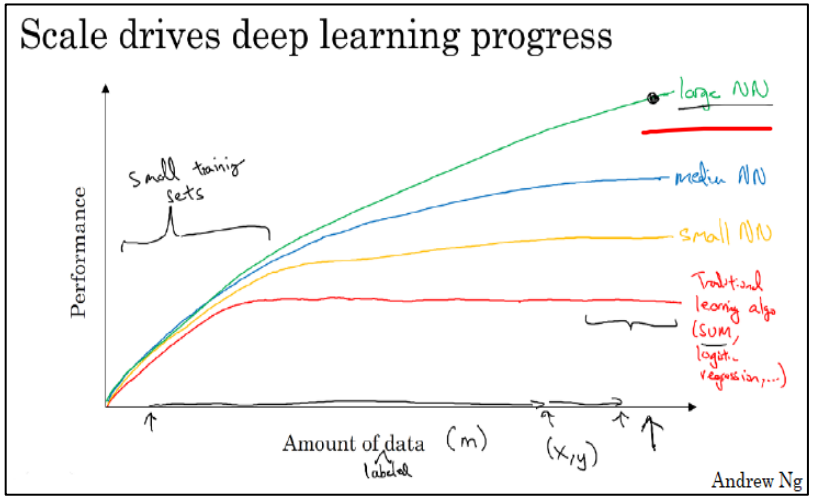
In short:

Deep Learning = The **technology (the engine)**

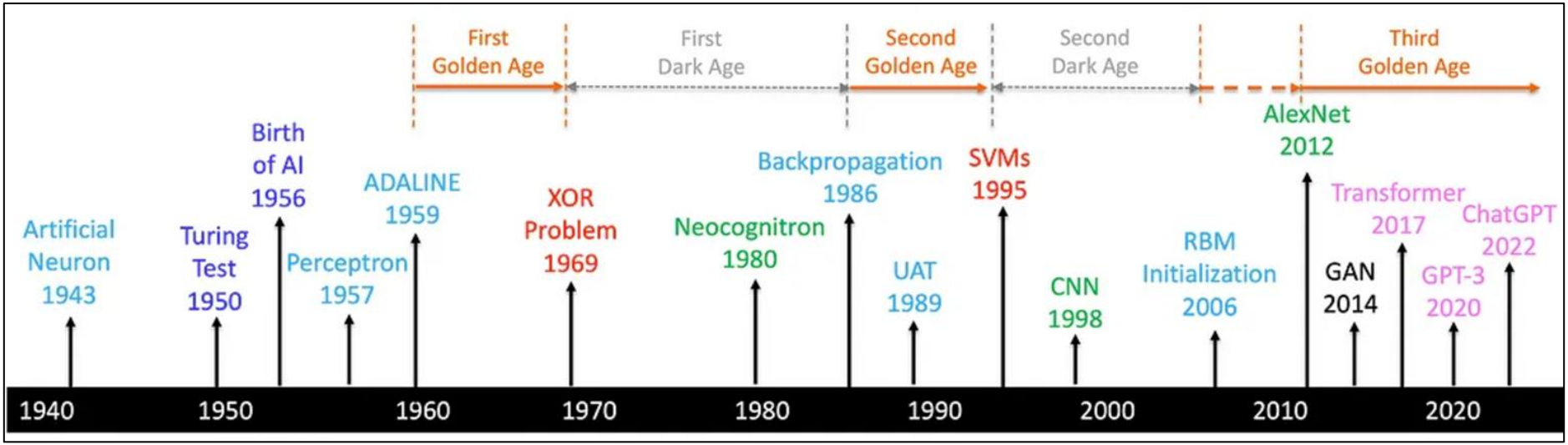
Generative AI = The **application (the car)**

Why deep learning?

Color	Model Type	Meaning
Red line	Traditional learning algorithms (e.g., SVM, Logistic Regression, Decision Trees)	These models perform well on small data but **plateau early** — they can't improve much even if you give them more data.
Yellow line	Small Neural Network	A simple (shallow) neural network that can learn patterns, but its capacity is limited.
Blue line	Medium Neural Network	A deeper, more complex neural network that can learn better from larger datasets.
Green line	Large Neural Network (Deep Learning)	A deep learning model with many layers and parameters — **keeps improving** as more data is added.



Deep **Learning** Timeline



Applications of Deep Learning

Microsoft AI Beats Humans at Speech Recognition

By Richard Adhikari
Oct 20, 2016 11:40 AM PT

Print Email

Image: Adobe Stock

Microsoft's Artificial Intelligence and Research Unit earlier this week reported that its speech recognition technology had surpassed the performance of human transcriptionists.

Most Popular Newsletters News Aler

How do you feel about Black Friday and Cyber Monday?

- They're great -- I get a lot of bargains!
- The deals are too spread out -- I'd prefer just one day.
- They're a fun way to kick off the holiday season.
- I don't like the commercialization of Thanksgiving Day.
- They're crucial for the retail industry and the economy.
- The deals typically aren't that good.

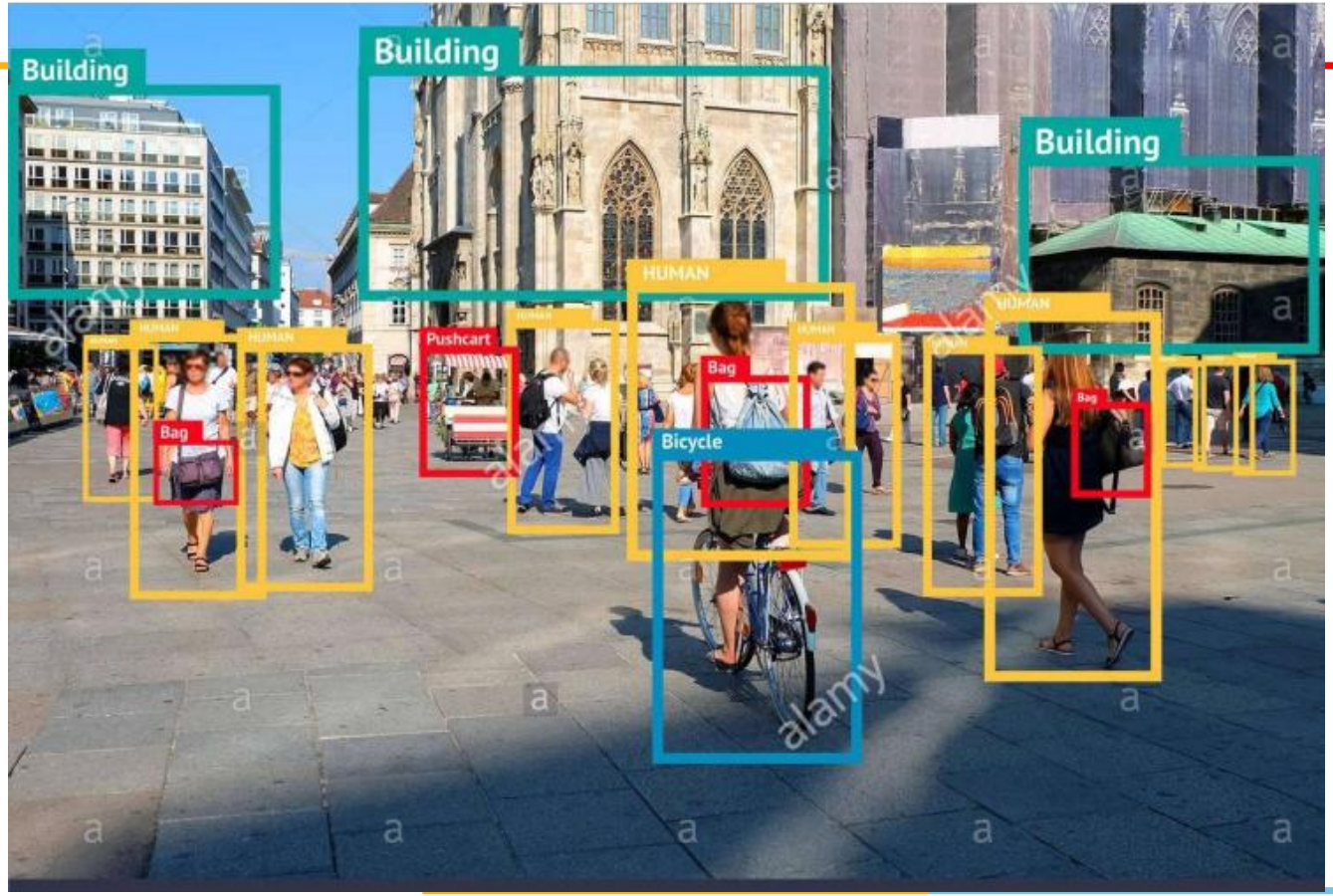
Vote to See Results

E-Commerce Times

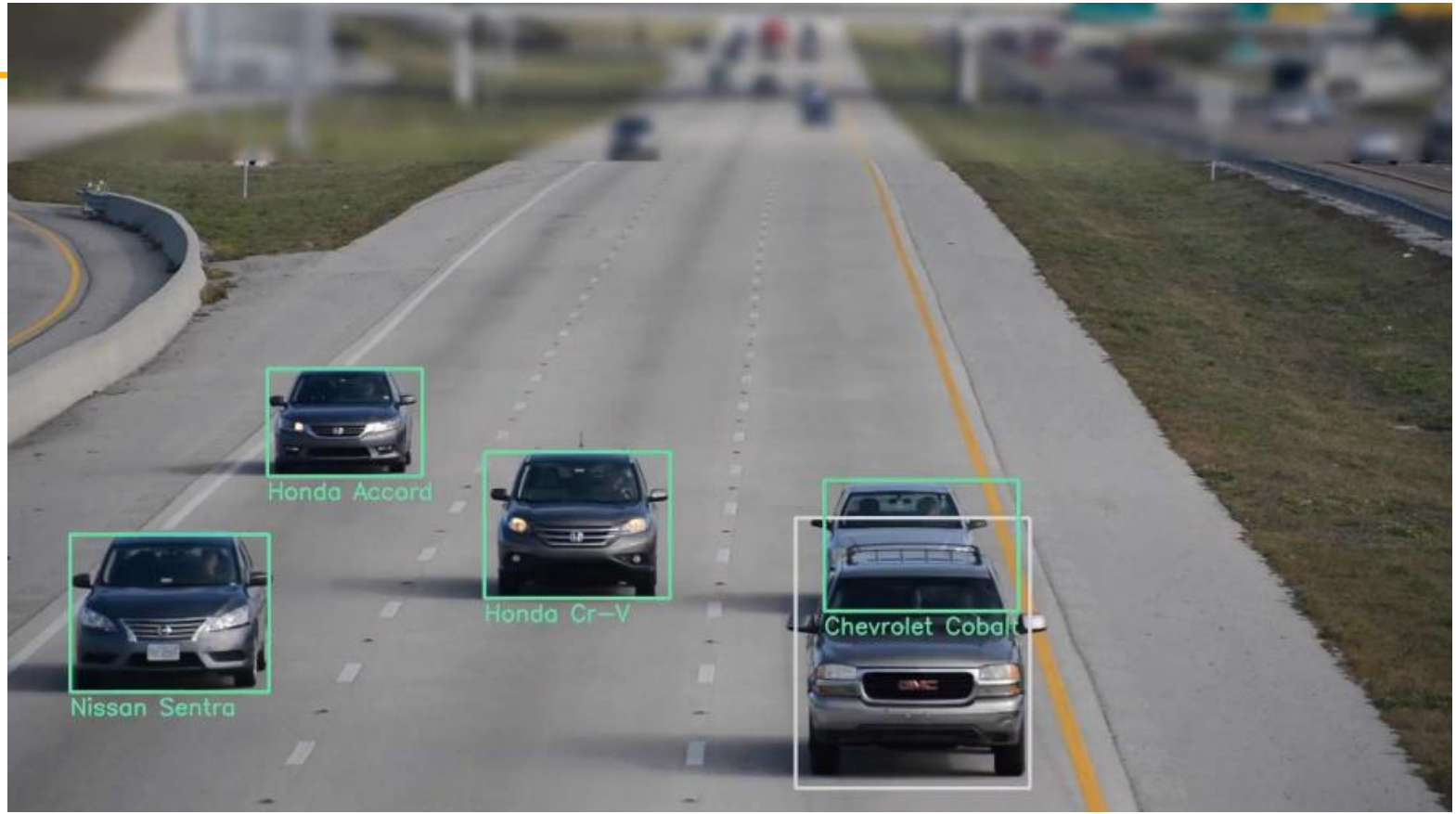
- Black Friday Shoppers Hungry for New Experiences, New Tech
- Pay TV's Newest Innovation: Giving Users Control
- Apple Celebrates Itself in $300 Coffee Table Tome
- AWS Enjoys Top Perch in IaaS, PaaS Markets
- US Comptroller Gears Up for Blockchain and


The Keyword
Latest Stories
Product News
Topics
TRANSLATE
NOV 15, 2016
Found in translation: More accurate, fluent sentences in Google Translate
Barak Turovsky
PRODUCT LEAD, GOOGLE TRANSLATE
In 10 years, Google Translate has gone from supporting just a few languages to 103, connecting strangers, teaching across language barriers and even helping

Breakthroughs with Neural Networks



Breakthroughs with Neural Networks



Breakthroughs with Neural Networks

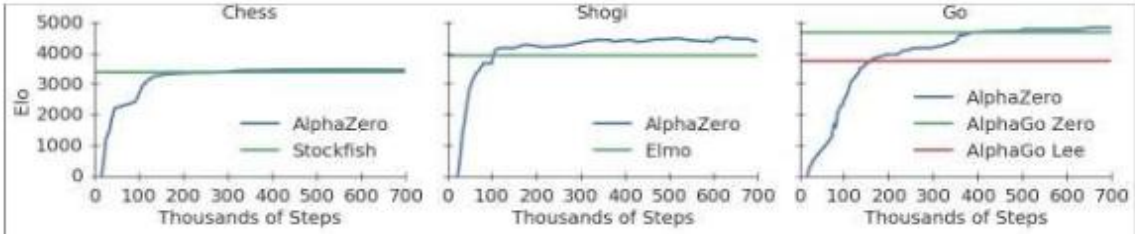
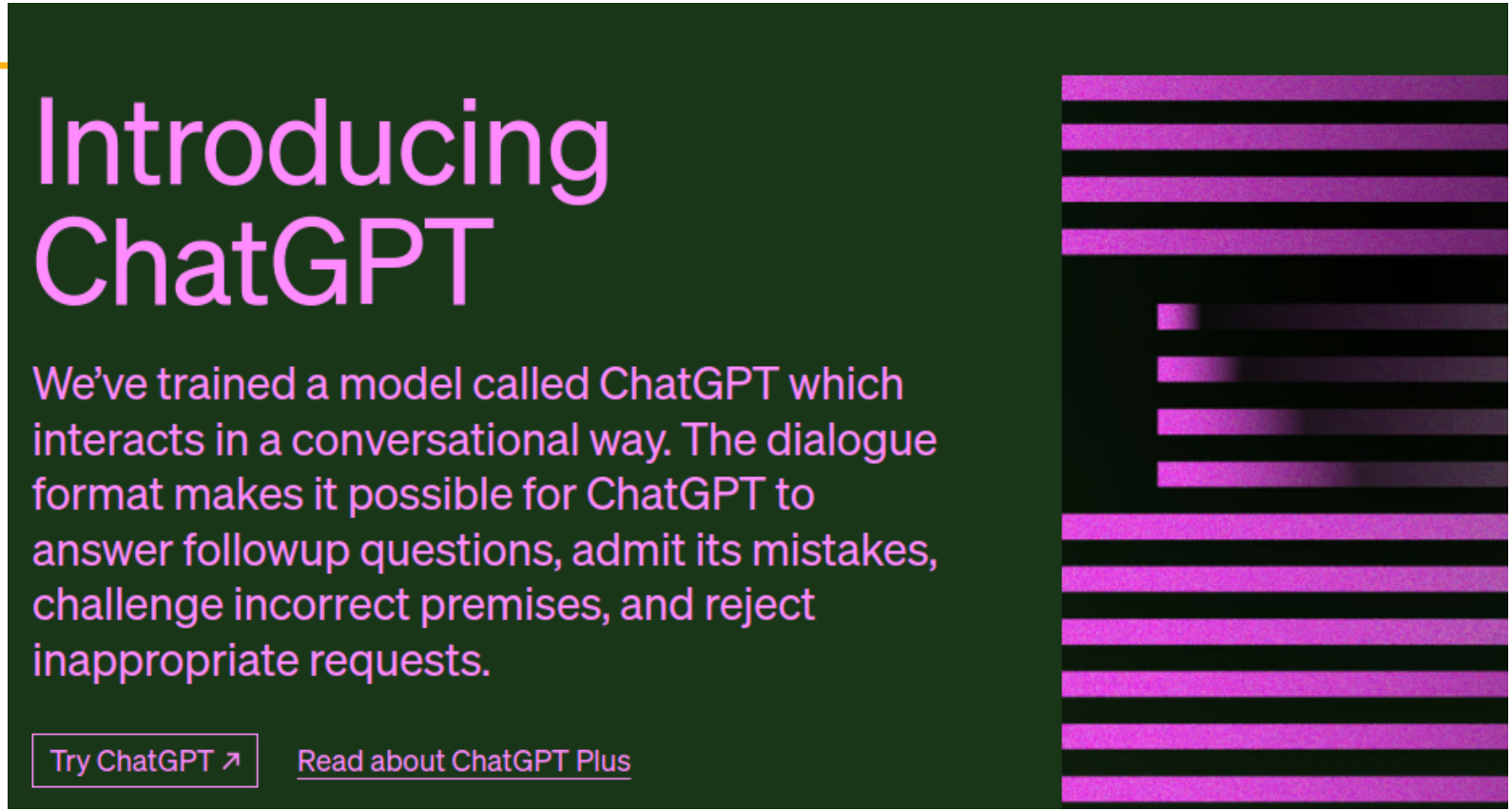


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (*29*).

Breakthroughs with Neural Networks



Applications of Deep Learning

Application	Input	Output	Neural Network
Real Estate	House features	House Price	Std NN
Photo Tagging	Image	Text	CNN
Object detection	Image	Bounding box	CNN
Speech Recognition	Audio	Text transcript	RNN
Translation	English Text	French Text	RNN
Autonomous driving	Image, Sensors Radars	Position of other cars, objects, signals	Hybrid NN

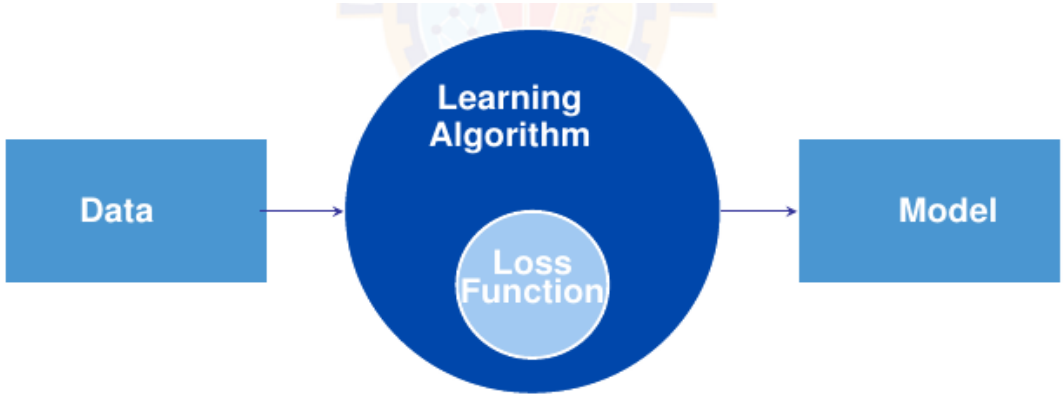
Many more applications….

- a program that predicts tomorrow's weather given geographic information, satellite images, and a trailing window of past weather.
- a program that takes in a question, expressed in free-form text, and answers it correctly.
- a program that given an image can identify all the people it contains, drawing outlines around each.
- a program that presents users with products that they are likely to enjoy but unlikely, in the natural course of browsing, to encounter.

Key components of DL problem

Core components of DL problem

1. The data that we can learn from.
2. A model of how to transform the data.
3. An objective function that quantifies how well (or badly) the model is doing.
4. An algorithm to adjust the model's parameters to optimize the objective function.



1. Data

- Collection of examples.
- Data has to be converted to an useful and a suitable numerical **representation**.
- Each example (or data point, data instance, sample) typically consists of a set of attributes called **features** (or covariates), from which the model must make its predictions.
- In the supervised learning problems, the attribute to be predicted is designated as the **label** (or target).
- Mathematically, a set of *m* examples,
- We need **right** data.

$$Data = \mathcal{D} = \{X, t\}$$

1. Data

- **Dimensionality** of data
  - Each example has the same number of numerical values. This data consist of fixed-length vectors. Eg: Image
  - The constant length of the vectors as the dimensionality of the data.
  - Text data has varying-length data.

2. Model

- Model denotes the **computational machinery** for ingesting data of one type, and spitting out predictions of a possibly different type.
- Deep learning models consist of many successive transformations of the data that are chained together top to bottom, thus the name deep learning.

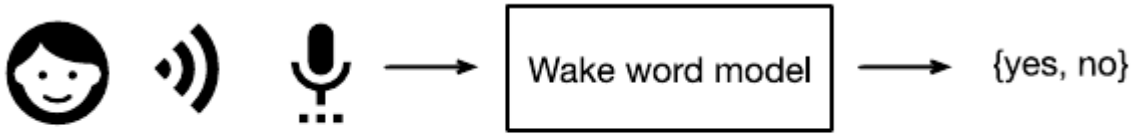


Fig. 1.1.1: Identify a wake word.

3. Objective Function

- **Learning means improving at some task over time.**
- A formal mathematical system of learning machines is defined using formal measures of how good (or bad) the models are. These formal measures are called as **objective functions**.
- By convention, objective functions are defined so that lower is better.
- Because lower is better, these functions are sometimes called **loss functions**.

3. Loss Functions

- To predict numerical values (regression), the most common loss function is squared error.
- For classification, the most common objective is to minimize error rate, i.e., the fraction of examples on which our predictions disagree with the ground truth.

3. Loss Functions

- Loss function is defined **with respect to the model parameters** and **depends upon the dataset**.
- We learn the best values of our model parameters by **minimizing the loss** incurred on a set consisting of some number of examples collected for training. However, doing well on the training data does not guarantee that we will do well on unseen data. I.e **Model has to generalize better.**
- When a model performs well on the training set but fails to generalize to unseen data, we say that it is **overfitting**.

4. Optimization Algorithms

- Optimization Algorithm is an algorithm capable of **searching for the best possible parameters** for **minimizing the loss function**.
- Popular optimization algorithms for deep learning are based on an approach called **gradient descent**.

Example of the Framework

- We have to tell a computer explicitly how to map from inputs to outputs.
- We have to define the problem precisely, pinning down the exact nature of the inputs and outputs, and choosing an appropriate model family.
- Collect a huge dataset containing examples of audio and label those that do and that do not contain the wake word.

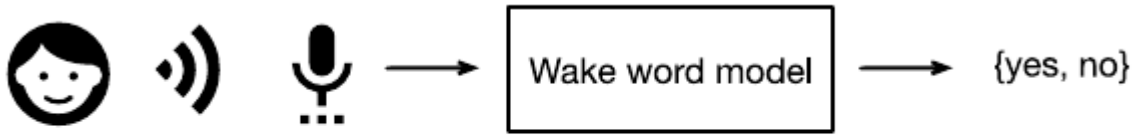


Fig. 1.1.1: Identify a wake word.

Example of the Framework

- Create a Model
  - Define a flexible program whose behavior is determined by a number of parameters.
  - To determine the best possible set of parameters, use the data. The parameters should improve the performance of the program with respect to some measure of performance on the task of interest.
  - After fixing the parameters, we call the program a model.
    - Eg: The model receives a snippet of audio as input, and the model generates a selection among yes, no as output.
  - The set of all distinct programs (input-output mappings) that we can produce just by manipulating the parameters is called a family of models.
    - Eg: We expect that the same model family should be suitable for ”Alexa” recognition and ”Hey Siri” recognition because they seem, intuitively, to be similar tasks.

Example of the Framework

- The meta-program that uses our dataset to choose the parameters is called a learning algorithm.
- In machine learning, the learning is the process by which we discover the right setting of the parameter coercing the desired behavior from our model.
- Train the model with

  data.

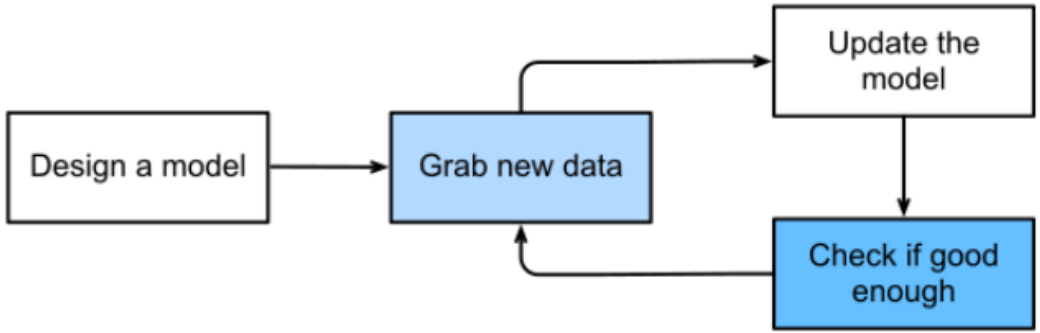


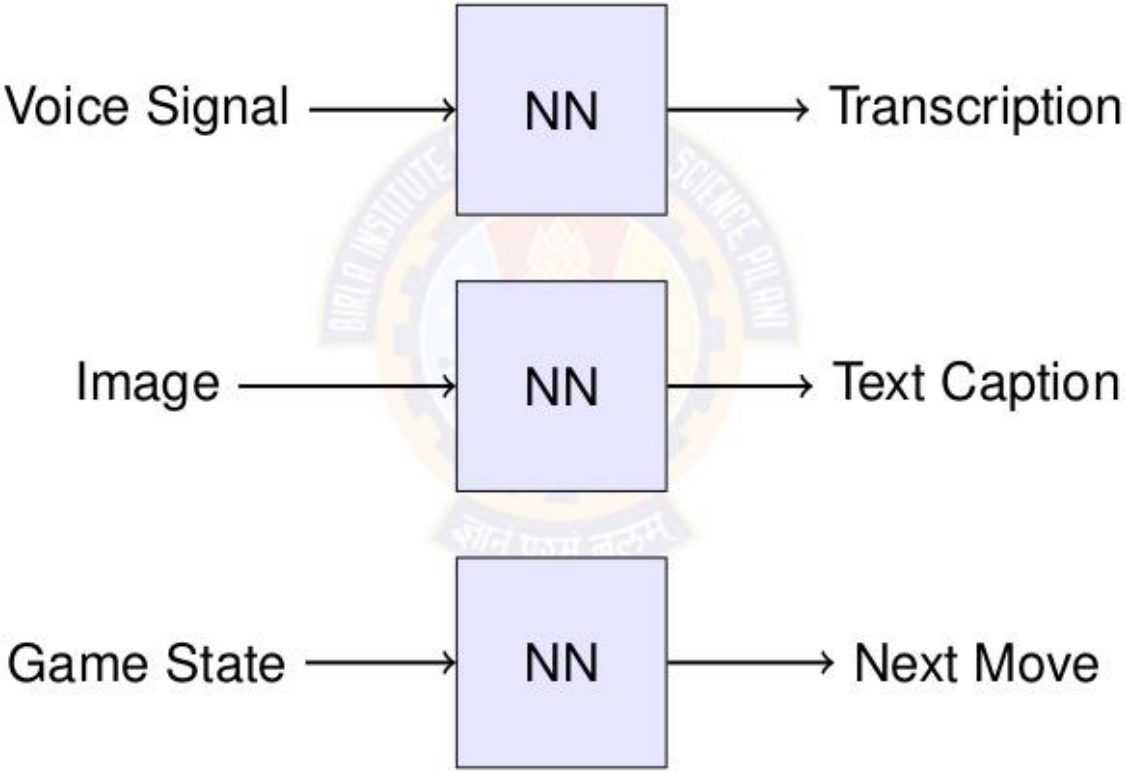
Fig. 1.1.2: A typical training process.

Reading from TB Dive into Deep Learning

- Chapter 1
- Chapter 2 for Python Prelims, Linear Algebra, Calculus, Probability

Artificial Neural Network

What are Neural Networks?



What are Neural Networks?

- It begins with human brain.

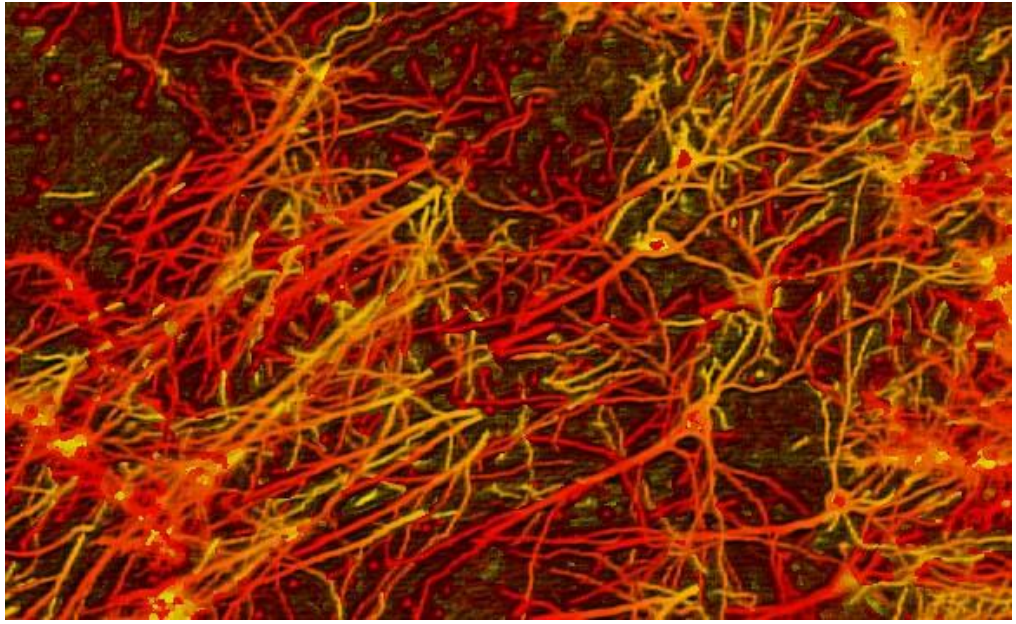
- Humans learn, solve problems, recognize patterns, create, think deeply about something, meditate and many many more.....
- Humans learn through **association**.

Observation: The Brain

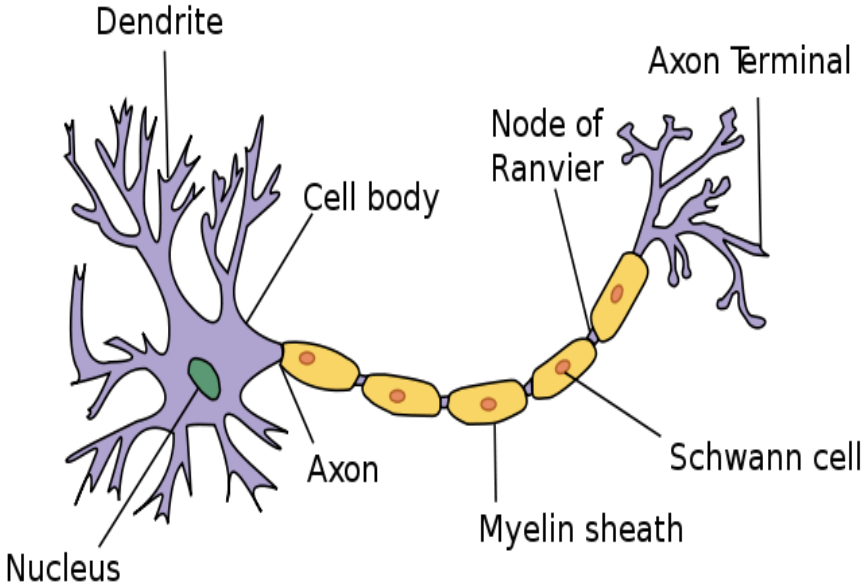
- The brain is a mass of interconnected neurons.
- Number of neurons is approximately 10^(10).
- Connections per neuron is approximately 10^(4 to 5).
- Neuron switching time is approximately 0.001 second.
- Scene recognition time is 1 second.
- 100 inference steps doesn't seem like enough. Lot of parallel computation.

Brain: Interconnected Neurons

- Many neurons connect in to each neuron.
- Each neuron connects out to many neurons.



Biological Neuron



1. Dendrites

•Biological Function:

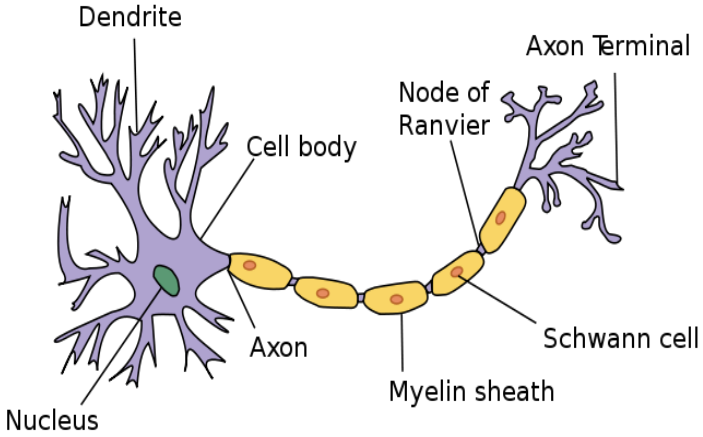
Dendrites are branch-like structures that receive incoming electrical signals (information) from other neurons and carry them toward the cell body (soma).

•Analog in ANN:

Dendrites correspond to **inputs (features)** in an artificial neuron.

Each input receives a value (e.g., $x_1, x_2, x_3, \ldots$ (representing different attributes of data.

Biological Neuron



2. Cell Body (Soma)

•Biological Function:

The soma processes incoming signals from the dendrites and determines whether the neuron should activate (fire an electrical impulse).

•Analog in ANN:

The soma corresponds to the **summation and activation function** in ANN.

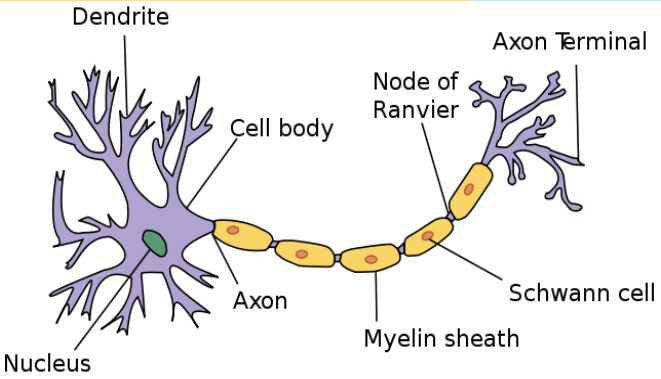
- The inputs are combined linearly (summation):

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Then, an **activation function** (like ReLU, Sigmoid, Tanh) determines the output signal strength:

$$y = f(z)$$

Biological Neuron



3. Axon

•Biological Function:

The axon is a long fiber that carries the processed electrical signal away from the cell body to other neurons or effectors.

•Analog in ANN:

The axon represents the **output signal** of an artificial neuron — the value after applying the activation function.

This output becomes the **input** to other neurons in subsequent layers.

4. Myelin Sheath

•Biological Function:

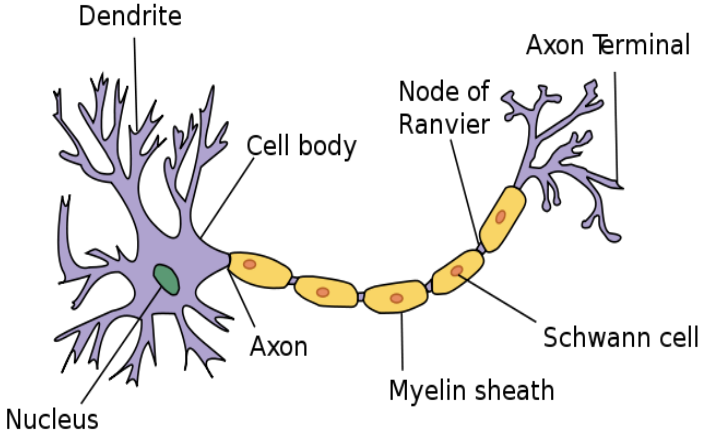
The myelin sheath is a fatty layer that insulates the axon and speeds up the transmission of electrical impulses.

•Analog in ANN:

The myelin sheath doesn't have a direct mathematical counterpart, but conceptually it relates to **optimization and efficiency mechanisms** — for example:

- Techniques that speed up training (like **batch normalization**, **momentum**, or **efficient architectures**)
- **Regularization methods** that maintain smooth signal flow and reduce "noise."

Biological Neuron



5. Node of Ranvier

•Biological Function:

These are small gaps in the myelin sheath where electrical impulses are regenerated, ensuring signal strength is maintained over long distances.

•Analog in ANN:

The Nodes of Ranvier are conceptually similar to **activation points** between layers — where intermediate computations occur to maintain "signal strength" across the network.

Each neuron in an ANN reprocesses and reactivates the signal before passing it forward.

6. Axon Terminals (Synaptic Terminals)

•Biological Function:

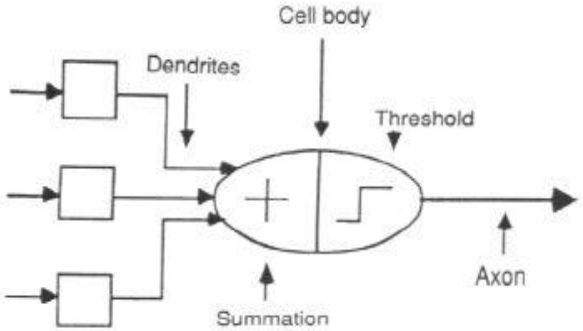
Axon terminals connect to dendrites of other neurons through synapses, transferring signals chemically or electrically to the next neuron.

•Analog in ANN:

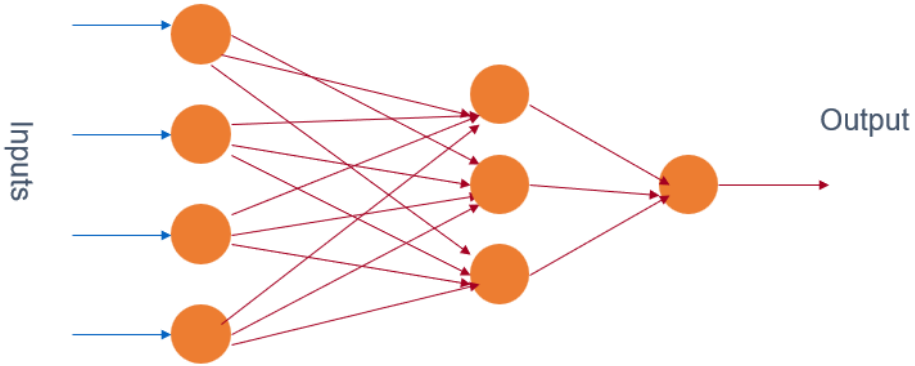
These represent the **connections (weights)** between neurons in different layers.

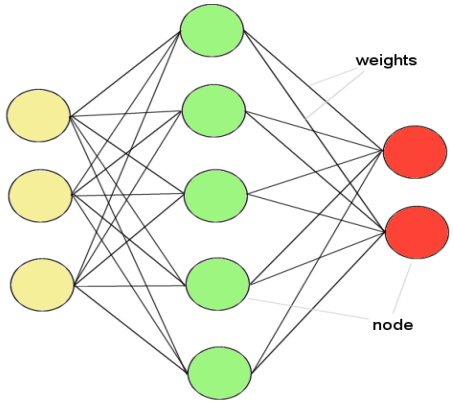
The "signal transmission" at the synapse corresponds to multiplying the output by a **weight**, adjusting the strength of the connection during learning.

Artificial Neural Networks



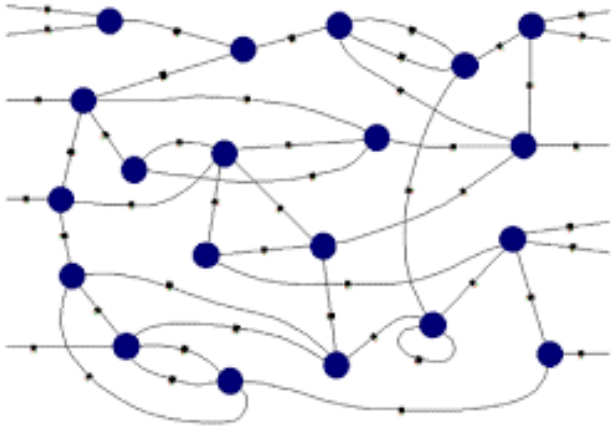
Key Insight:
Artificial Neural Networks mimic the **signal processing** behavior of biological neurons — but with **mathematical operations** instead of electrical impulses.
While a biological neuron fires when its potential exceeds a threshold, an artificial neuron activates when its **weighted sum** surpasses a certain level defined by the **activation function**.





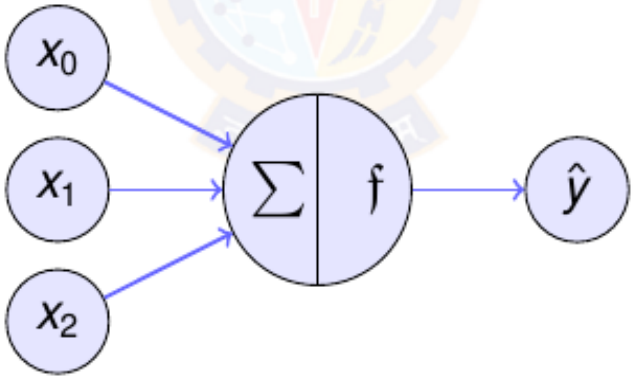
Connectionist Machines

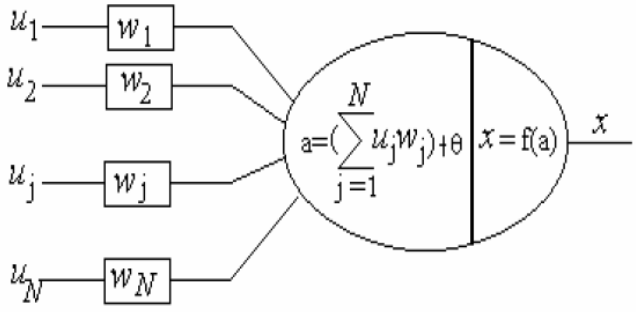
- Network of processing elements, called artificial neural unit.
- The neurons are interconnected to form a network.
- All world knowledge is stored in the connections between these elements.
- Neural networks are connectionist machines.



What are Artificial Neurons?

- Neuron is a processing element inspired by how the brain works.
- Similar to biological neuron, each artificial neuron will be do some computation. Each neuron is interconnected to other neurons.
- Similar to brain, the interconnections between neurons store the knowledge it learns. The knowledge is stored as parameters.





Properties of Artificial Neural Nets (ANNs)

- **Nonlinearity**: ANNs can model **non-linear relationships** between inputs and outputs because of their activation functions (e.g., ReLU, Sigmoid, Tanh).
- Unlike traditional linear models, ANNs can learn complex patterns such as curved surfaces, boundaries, and chaotic relationships.
- Predicting house prices: price doesn't increase linearly with size — factors like location, amenities, and neighborhood cause nonlinear effects.

Learning Ability (Adaptivity)

ANNs learn from examples using **training data**. Through **backpropagation and optimization algorithms (like gradient descent)**, they automatically adjust internal weights to minimize errors.
This adaptability allows them to improve performance over time.
•**Example:** A **spam detection system** learns from examples of spam and non-spam emails.

Generalization

Once trained, ANNs can **generalize** — i.e., they can make correct predictions on **new, unseen data**, not just the examples they were trained on.
Good generalization means the model has learned the **underlying pattern**, not just memorized the training data.
•**Example:** A model trained to recognize **cats and dogs** from thousands of images can correctly classify a **new, unseen** picture of a cat taken from a different angle or lighting condition.

Properties of Artificial Neural Nets (ANNs)

Fault Tolerance / Robustness

ANNs are **fault-tolerant** — minor errors or noise in input data don't drastically affect the output because of their **distributed representation** of knowledge.

Information is stored across many neurons, so damage to a few does not destroy performance.

•**Example:**

- In **speech recognition**, if there's slight background noise, the ANN can still recognize spoken words accurately.
- Biological analogy: even if some neurons in the brain fail, we still perform most tasks normally.

Parallelism

Neural networks perform computations in **parallel** — all neurons in a layer process information simultaneously. This property makes ANNs highly efficient when implemented on GPUs or TPUs that support parallel processing.

•**Example:**

- When recognizing a face, hundreds of neurons may simultaneously analyze eyes, nose, mouth, and contours — all at once — rather than sequentially.

Properties of Artificial Neural Nets (ANNs)

Distributed Representation of Knowledge

Knowledge in an ANN is **not stored in one location** (like a database). Instead, it's **distributed across the network's weights and biases**.

Each connection contributes partially to different learned features.

•**Example:**

- In an image recognition model, one set of neurons might learn to detect **edges**, another **textures**, and others **objects** — together forming a complete understanding of the image.

Ability to Handle Noisy or Incomplete Data

•**Explanation:**

ANNs are good at handling **noisy, missing, or imprecise data**, as they can approximate patterns rather than requiring perfect input.

•**Example:**

- In **medical diagnosis**, if a few test results are missing, a trained ANN can still predict the likelihood of a disease using available data.

Properties of Artificial Neural Nets (ANNs)

Self-Organization (especially in unsupervised learning)
Some types of ANNs, like **Self-Organizing Maps (SOMs)** and **Autoencoders**, can **discover structures or clusters** in data **without supervision**.
They organize input data into meaningful patterns based on similarity.

•**Example:**

- A SOM can automatically group customers with similar purchasing habits — useful in **market segmentation**.

Fault Resilience through Redundancy
Because multiple neurons can perform similar computations, ANNs inherently include **redundancy**, which makes them **resilient** to small failures or fluctuations in data.

•**Example:**

- In an autonomous vehicle's vision system, even if one neuron fails to detect a road line, others can still recognize it correctly.

When to consider Neural Networks?

Neural Networks should be considered when:

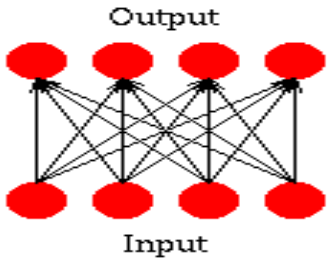
1.The problem is complex and nonlinear.

2.The dataset is large and rich in features.

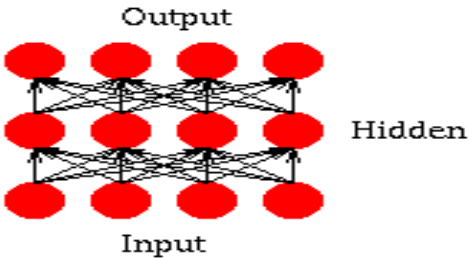
3.The relationships between input and output are difficult to model with traditional algorithms.

4.You need high accuracy and can afford computation time.

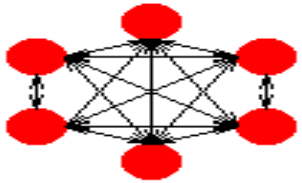
5.Feature extraction is difficult or should be learned automatically.



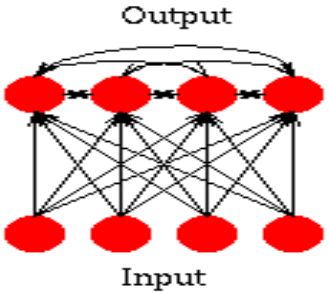
Single Layer Feedforward



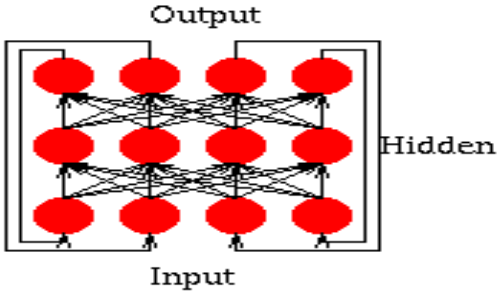
Multi Layer Feedforward



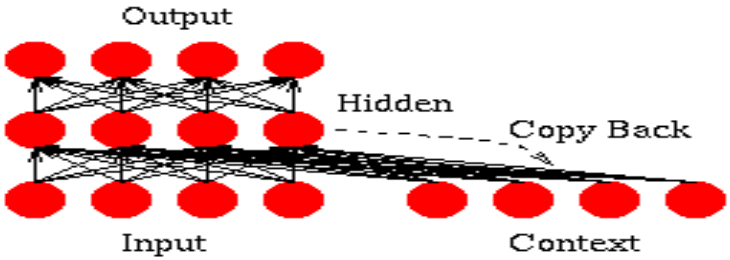
Fully Recurrent Network



Competitive Network



Jordan Network



Simple Recurrent Network

Learning in ANNs

ANNs are capable of learning and they need to be trained. There are several learning strategies −

- Supervised Learning − It involves a teacher that is scholar than the ANN itself. For example, the teacher feeds some example data about which the teacher already knows the answers.
- Unsupervised Learning − It is required when there is no example data set with known answers. For example, searching for a hidden pattern. In this case, clustering i.e. dividing a set of elements into groups according to some unknown pattern is carried out based on the existing data sets present.
- Reinforcement Learning − This strategy built on observation. The ANN makes a decision by observing its environment. If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time.

Classification of Learning Algorithms in ANN

1. Error-Correction Learning (Delta Rule / Perceptron Learning Rule)

- Based on the principle of **reducing the error** between the actual output and desired output.
- The weights are adjusted in the direction that minimizes this error.
- Learning by minimizing prediction error iteratively.

2. Hebbian Learning Concept:

- Proposed by psychologist Donald Hebb (1949).
- Known as the “cells that fire together, wire together” principle.
- If two neurons activate simultaneously, their connection strengthens.
- Strengthen connections between co-active neurons (unsupervised learning).

3. Competitive Learning

- Neurons **compete** among themselves to be activated.
- Only one neuron (or a small subset) wins and gets its weights updated.
- Helps in **clustering and feature discovery**.
- Learning by competition — one neuron “wins,” others “lose.”

Classification of Learning Algorithms in ANN

4. Boltzmann Learning

- Used in **stochastic and energy-based networks**, like **Boltzmann Machines** and **Restricted Boltzmann Machines (RBMs)**.
- Neurons update their states probabilistically to **minimize an energy function**.
- Learning by minimizing system energy — probabilistic and generative.

5. Gradient-Based (Stochastic) Learning

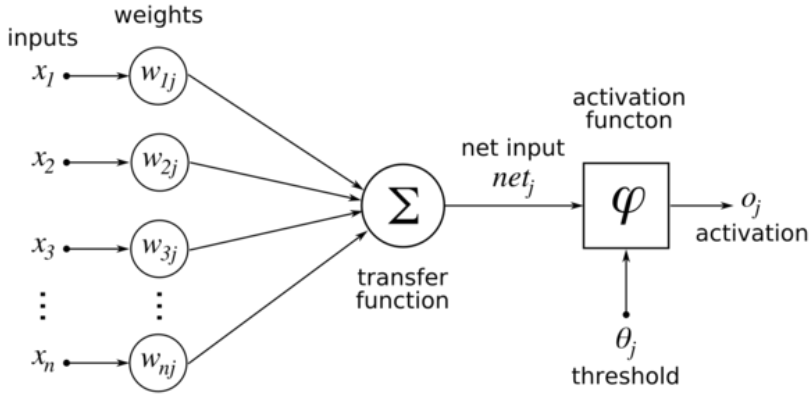
- This is the **most common and powerful** category used in **Deep Learning** today.
- It minimizes the **loss function (E)** by adjusting weights along the **negative gradient** direction.

6. Reinforcement Learning-Based Algorithms

- The system learns by **trial and error** using **reward signals** instead of explicit labels.
- The goal is to maximize long-term rewards.

What are activation functions & its role?

- An **activation function** is a **mathematical function** applied to the **output of each neuron** in an Artificial Neural Network.
- It decides whether a neuron should be **activated or not**, i.e., whether the information should be passed forward in the network.
- An activation function introduces **non-linearity** into a neural network, enabling it to **learn complex relationships** between input and output.



Think of a neuron as a **decision maker**:

- It receives inputs (signals),
- Computes a weighted sum,

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

The final output is: $y = f(z)$

Passes this through an **activation function**, which decides:

- Should it fire strongly?
- Should it suppress the signal?
- Should it partially activate?

What are activation functions & its role?

Types of Activation Functions

Activation functions are generally categorized into three groups:

1.Linear Activation Function

2.Non-Linear Activation Functions

1. Sigmoid Family
2. ReLU Family
3. Other modern functions

3.Specialized Output Layer Activations

What are activation functions & its role?

innovate achieve lead

Activation Function	Formula	Range	Common Use	Example Application	Limitation
Linear (Identity)	$f(x) = x$	$-\infty, +\infty$	Output layer for **regression tasks**	Predicting house prices, temperature, or sales	Cannot learn non-linear patterns; same output for multiple layers
Sigmoid (Logistic)	$f(x) = \frac{1}{1+e^{-x}}$	(0, 1)	Output layer for **binary classification**	Disease detection (Yes/No), Spam detection	Saturates for large +/– x → **vanishing gradients**, not zero-centered
Tanh (Hyperbolic Tangent)	$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	(-1, 1)	Hidden layers of **RNNs / simple ANNs**	Time series forecasting, signal prediction	Vanishing gradients for large
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$	[0, ∞)	Default activation for **CNNs & DNNs**	Image classification (ResNet, AlexNet)	"**Dead neurons**" problem for negative x values
Leaky ReLU	$f(x) = \begin{cases} x, & x > 0 \\ 0.01x, & x \leq 0 \end{cases}$	(-∞, ∞)	Hidden layers in **deep networks**	Object detection, deep CNNs	Small slope may still cause instability

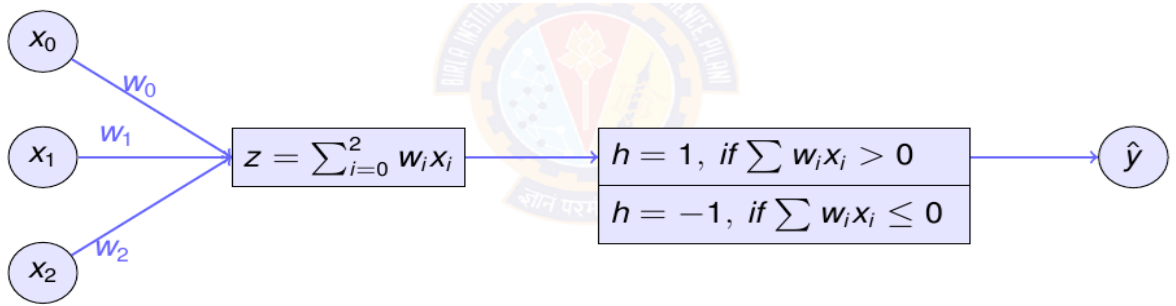
What are activation functions & its role?

Parametric ReLU (PReLU)	$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$	(-∞, ∞)	Adaptive **CNN architectures**	Face recognition (He et al., ResNet)	Extra parameter (α) adds complexity
ELU (Exponential Linear Unit)	$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$	(-α, ∞)	Deep CNNs for **faster convergence**	Medical image segmentation	Computationally more expensive
Swish (Google's Function)	$f(x) = x \cdot \text{sigmoid}(x)$	(-∞, ∞)	**Advanced deep networks**	EfficientNet, Vision Transformers	Slightly slower due to sigmoid computation
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	(0, 1), sum=1	**Output layer** for multiclass classification	MNIST digit classification, NLP models	Sensitive to outliers; numerical instability for large inputs
GELU (Gaussian Error Linear Unit)	$f(x) = x \cdot \Phi(x)$ (where Φ is Gaussian CDF)	(-∞, ∞)	**Transformer architectures**	BERT, GPT, LLMs	Complex computation (not as fast as ReLU)
Softplus	$f(x) = \ln(1 + e^x)$	(0, ∞)	Smooth ReLU alternative	Regression tasks with smooth gradients	Still has vanishing gradient for large negative x

Perceptron

Perceptron

- A **Perceptron** is the **simplest form of an artificial neural network neuron** — a computational model that mimics how a single biological neuron works.
- **A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, and then outputs a 1 if the result is greater than some threshold and -1 otherwise.**



Perceptron-Mathematical Representation

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

$$f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

Where:

- x_i = input features
- w_i = weights (importance of each input)
- b = bias (shifts the decision boundary)
- $f(\cdot)$ = activation function
- y = output (predicted class: 0 or 1)

The **bias** is an additional parameter in a neuron that allows the activation function to be shifted **left or right** on the input axis. bias acts like an intercept (c) in a straight-line equation y=mx+c.

So, the perceptron makes a **binary decision**.

Goal of a Perceptron

To **find the correct weights (w)** and **bias (b)** so that it correctly separates the data points of different classes. It essentially learns a **linear decision boundary** like:

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = 0$$

This line (or plane) separates one class from another.

Use of bias & Learning rate

- Imagine a decision boundary (a line) separating two classes.
- Without a bias, this line **must always pass through the origin (0,0)** — that's a big limitation.
- 👉 The **bias term allows shifting** this line **up or down** (or hyperplane in higher dimensions) so the model can fit the data better.

◆ **Example (AND Gate)**

Suppose we have:

$$y = f(w_1x_1 + w_2x_2)$$

If both weights are 1, this gives:

x_1	x_2	$z = x_1 + x_2$	f(z)
0	0	0	?
0	1	1	?
1	0	1	?
1	1	2	?

To make this behave like an AND gate (output 1 **only if** both inputs are 1), we need:

$$f(z) = 1 \text{ if } z \geq 1.5$$

So we set bias $b = -1.5$:

$$y = f(x_1 + x_2 - 1.5)$$

Now the perceptron gives correct AND gate outputs.

- Bias adds **flexibility** to the neuron's output.
- It ensures the neuron **can activate even when inputs are zero**.
- It acts like a "default activation trigger" independent of inputs.

Use of Learning rate

The learning rate (η) is a small positive constant that controls how much the weights are adjusted during training.

Weight update rule in perceptron:

$$w_i^{new} = w_i^{old} + \eta(t - y)x_i$$

$$b^{new} = b^{old} + \eta(t - y)$$

Here:

- $(t - y)$ = error (difference between target and predicted output)
- η = learning rate

When the model makes an error, it updates the weights.
The learning rate decides **how big that correction step** should be.

- If **η is too large** → model overshoots, may oscillate or diverge.
- If **η is too small** → learning becomes extremely slow, may get stuck in local minima.

Suppose error = +1 and input = 2.

If:

- η = 1 → update = +2 (large step)
- η = 0.1 → update = +0.2 (small step)

So η controls the **speed and stability** of learning.

- Large learning rate → big jumps → may miss the valley.
- Small learning rate → tiny steps → safe, but slow.

Representing Logic Gates using Perceptron

- **Perceptron** can represent all of the primitive Boolean functions AND, OR, NAND, and NOR – Linearly separable data
- Some Boolean functions cannot be represented by a single perceptron, such as the XOR function – Linearly nonseparable data
- Perceptron learning rule finds a successful weight vector when the training examples are linearly separable, **it can fail to converge if the examples are not linearly separable.**

Perceptron Learning Algorithm

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

Convergence of Perceptron Learning Algorithm

It can be proved that the algorithm will converge

- If training data is linearly separable.
- Learning rate is sufficiently small
  - The role of the learning rate is to moderate the degree to which weights are changed at each step.
  - It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases.

Example 4.1 Develop a perceptron for the AND function with bipolar inputs and targets

Solution The training pattern for AND function can be,

Input			Target
X_1	X_2	b	t
1	1	1	1
−1	1	1	−1
1	−1	1	−1
−1	−1	1	−1

Step 1: Initial weights $w_1 = w_2 = 0$ and $b = 0$, $\alpha = 1$, $\theta = 0$.

Step 2: Begin computation.

Step 3: For input pair (1, 1): 1, do Steps 4–6

Step 4: Set activations of input units

$$x_i = (1, 1).$$

Step 5: Calculate the net input.

$$y_{-in} = b + \sum x_i w_i = 0 + 1 \times 0 + 1 \times 0 = 0$$

Applying the activation,

$$y = f(y_{-in}) = \begin{cases} 1, & \text{if } y_{-in} > 0 \\ 0, & \text{if } -0 \le y_{-in} \le 0 \\ -1, & \text{if } y_{-in} < -0 \end{cases}$$

Therefore $y = 0$.

Step 6: $t = 1$ and $y = 0$

Since $t \neq y$, the new weights are,

$$w_{i(new)} = w_{i(old)} + \alpha t x_i$$

$$w_{1(new)} = w_{i(old)} + \alpha t x_1 = 0 + 1 \times 1 \times 1 = 1$$

$$w_{2(n)} = w_{2(o)} + \alpha t x_2 = 0 + 1 \times 1 \times 1 = 1$$

$$b_{(new)} = b_{(old)} + \alpha t$$

$$b_{(n)} = b_{(0)} + \alpha t = 0 + 1 \times 1 = 1$$

The new weights and bias are [1 1 1].

The algorithmic steps are repeated for all the input vectors with their initial weights as the previously calculated weights.

By presenting all the input vectors, the updated weights are shown in table below:

Input			Net	Output	Target	Weight Changes			Weights		
x_1	x_2	B	y_{in}	y	t	Δw_1	Δw_2	Δb	w_1	w_2	B
								(0	0	0)	
1	1	1	0	0	1	1	1	1	1	1	1
−1	1	1	1	1	−1	1	−1	−1	2	0	0
1	−1	1	2	1	−1	−1	1	−1	1	1	−1
−1	−1	1	−3	−1	−1	0	0	0	1	1	−1

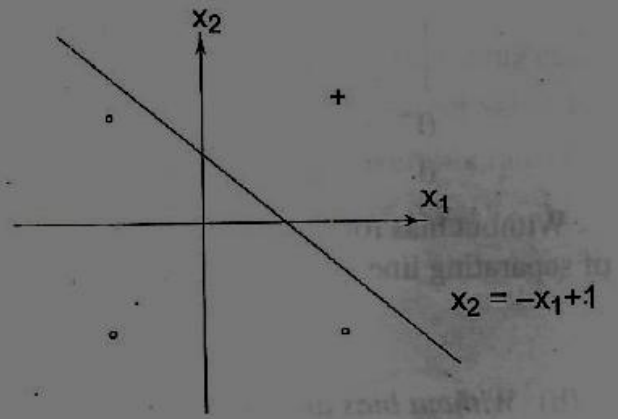
This completes one epoch of the training.

The final weights after the first epoch is completed are, $w_1 = 1, w_2 = 1, b = -1$

We know that $b + x_1 w_1 + x_2 w_2 = 0$

$$x_2 = -x_1 \frac{w_1}{w_2} - \frac{b}{w_2}$$

$$x_2 = -x_1 \frac{1}{1} - \frac{(-1)}{1}$$



$x_2 = -x_1 + 1$ is the separating line equation.

The decision boundary for AND function trained by perceptron network is given as,

In a similar way, the perceptron network can be developed for logic functions OR, NOT, AND NOT etc.

Example 4.2 Develop a perceptron for the AND function with binary inputs and bipolar targets without bias up to 2 epochs. (Take first with (0,0) and next without (0,0)).

Perceptron for AND with Binary inputs & Bipolar Targets

Epoch 1:

Input		Net	Output	Target	Weight Changes		Weights	
x_1	x_2	y_{in}	y	t	Δw_1	Δw_2	W_1 (0	W_2 0)
1	1	0	0	1	1	1	1	1
1	0	1	1	−1	−1	0	0	1
0	1	1	1	−1	0	−1	0	0
0	0	0	0	−1	0	0	0	0

The separating line for 1st and 2nd input are, $x_1 + x_2 = 0$ and $x_2 = 0$ respectively.

Epoch 2:

The initial weights used are the final weights from the pervious iteration.

Input		Net	Output	Target	Weight Changes		Weights	
x_1	x_2	y_{in}	y	t	Δw_1	Δw_2	W_1 (0	W_2 0)
1	1	0	0	1	1	1	1	1
1	0	1	1	−1	−1	0	0	1
0	1	1	1	−1	0	−1	0	0
0	0	0	0	1	0	0	0	0

Without bias for the given inputs, the final weights obtained are same as that for with bias and the equation of separating line also remains same. Thus the equations remain same and are given by,

for 1st input : $x_1 + x_2 = 0$

for 2nd input : $x_2 = 0$

OR Gate Function

Epoch 1

Iter	x_1	x_2	t	$y_{in} = b + w_1x_1 + w_2x_2$	y	$e = t - y$	$\Delta w_1 = \eta e x_1$	$\Delta w_2 = \eta e x_2$	$\Delta b = \eta e$	New w_1	New w_2	New b
1	1	1	1	0	0	1	+1	+1	+1	1	1	1
2	−1	1	1	1	1	0	0	0	0	1	1	1
3	1	−1	1	1	1	0	0	0	0	1	1	1
4	−1	−1	−1	−1	−1	0	0	0	0	1	1	1

End of Epoch 1 weights: $w_1 = 1,\ w_2 = 1,\ b = 1$

XOR Gate Function

Iter	x_1	x_2	t	$y_{in} = b + w_1x_1 + w_2x_2$	y	$e = t - y$	Δw_1	Δw_2	Δb	**New** w_1	**New** w_2	**New** b
1	1	1	−1	0	0	−1	−1	−1	−1	**−1**	**−1**	**−1**
2	−1	1	1	−1	−1	2	−2	2	2	**−3**	**1**	**1**
3	1	−1	1	−3	−1	2	2	−2	2	**−1**	**−1**	**3**
4	−1	−1	−1	5	1	−2	2	2	-2	1	1	1

End of Epoch 1: $w_1 = 1,\ w_2 = 1,\ b = 1$

XOR Gate Function

Epoch 2 (weights cycle back—still misclassifies some points)

Iter	x_1	x_2	t	y_{in}	y	e	Δw_1	Δw_2	Δb	New w_1	New w_2	New b
1	1	1	−1	3	1	−2	−2	−2	−2	−1	−1	−1
2	−1	1	1	−1	−1	2	−2	2	2	−3	1	1
3	1	−1	1	−3	−1	2	2	−2	2	−1	−1	3
4	−1	−1	−1	5	1	−2	2	2	−2	1	1	1

End of Epoch 2: back to $w_1 = 1,\ w_2 = 1,\ b = 1$ — exactly the same state as after Epoch 1.

- The perceptron is a linear classifier, therefore it will never get to the state with all the input vectors classified correctly if the training set *D* is not linearly separable, i.e. if the positive examples can not be separated from the negative examples by a hyperplane.
- In this case, no "approximate" solution will be gradually approached under the standard learning algorithm, but instead learning will fail completely.
- Hence, if linear separability of the training set is not known a priori, one of the training variants below should be used.

Perceptron Learning Algorithm for NOT gate

- Assume $w_0 = w_1 = 0$. Let the learning rate$= \eta = 1$.
- Equations are

$$\text{Hypothesis} \quad z = w_0 + w_1 x_1$$

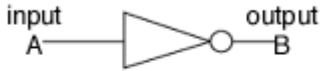
$$\text{Activation} \quad h = sign(z)$$

$$\text{Compute output} \quad \hat{y} = h = sign(w_0 + w_1 x_1)$$

$$\text{Compute } \Delta w = \eta(t - \hat{y})x$$

$$\text{Update } w_{new} \leftarrow w_{old} + \Delta w$$

NOT
(Inverter)



A	B
0	1
1	0

Perceptron Learning Algorithm for NOT gate

w1=w0=0, η =1

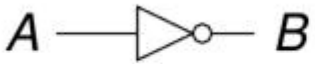
X1	t	W1	W0	z	h	Isequal(t,h)	Δw	New w
-1	1	0	0	0+0=0	-1	No	Δw1=1(1+1)*(-1)=-2 Δw0=1(1+1) =2	w1 0-2=-2 w0 0+2=2
1	-1	-2	2	2-2=0	-1	Yes	Δw1=1(-1+1)1=0 Δw0=1(-1+1) =0	w1 -2+0=-2 w0 2+0=2

X1	t	W1	W0	z	h	Isequal(t,h)	Δw	New w
-1	1	-2	2	2+2=4	1	Yes	Δw1=1(1-1)*(-1)=0 Δw0=1(1-1) =0	w1 -2+0=-2 w0 2+0=2

NOT Logic Gate

Question:

- How to represent NOT gate using a perceptron?
- What are the parameters for the NOT perceptron?
- Data is given below.



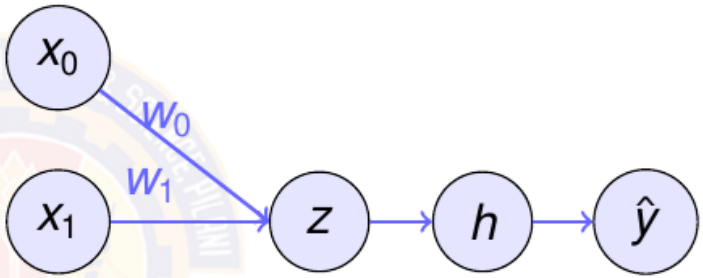
A	B
0	1
1	0

Rewrite as

x_1	t
-1	1
1	-1

Perceptron for NOT gate

- Perceptron equation is $\hat{y} = w_0 x_0 + w_1 x_1$.
- $x_0 = 1$ always.
- $h > 0$ for output to be 1.
- For each row of truth table, the equations are given.
- One solution is $w_0 = 1$ and $w_1 = -1$. (Intuitive solution)
- This give a beautiful linear decision boundary.

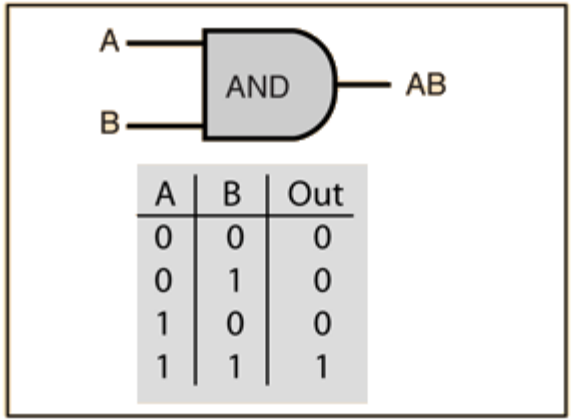


x_1	t_1	h	h
-1	1	$w_0 x_0 + w_1(-1) > 0$	$w_0 - w_1 > 0$
1	-1	$w_0 x_0 + w_1(1) < 0$	$w_0 + w_1 < 0$

AND Logic Gate

Question:

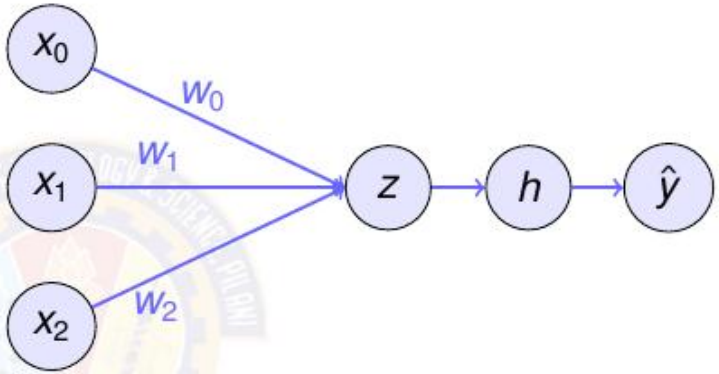
- How to represent AND gate using a perceptron?
- What are the parameters for the AND perceptron?
- Data is given below.



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

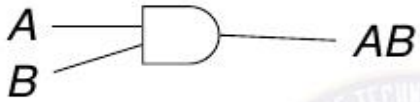
Perceptron for AND gate

- Perceptron equation is $\hat{y} = w_0 x_0 + w_1 x_1 + w_2 x_2$.
- $h > 0$ for output to be 1.
- For each row of the truth table, the equations are given.
- Solve for the inequalities.
- One solution is $w_1 = w_2 = 2, w_0 = (-1)$.
- This give a beautiful linear decision boundary.



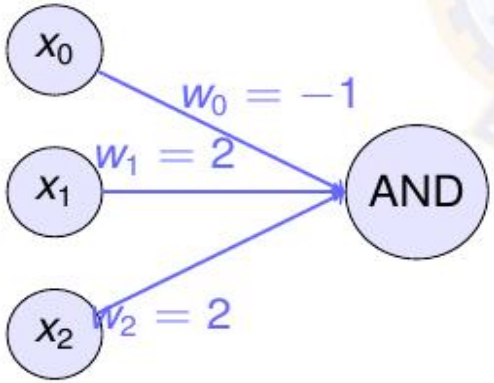
x_1	x_2	t	h
-1	-1	-1	$w_0 + w_1(-1) + w_2(-1) < 0$
-1	1	-1	$w_0 + w_1(-1) + w_2(1) < 0$
1	-1	-1	$w_0 + w_1(1) + w_2(-1) < 0$
1	1	1	$w_0 + w_1(1) + w_2(1) > 0$

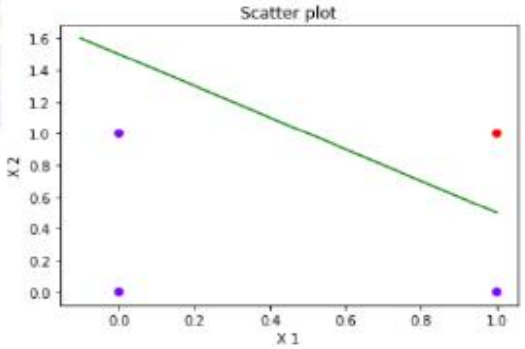
Perceptron for AND gate



A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

DECISION BOUNDARY



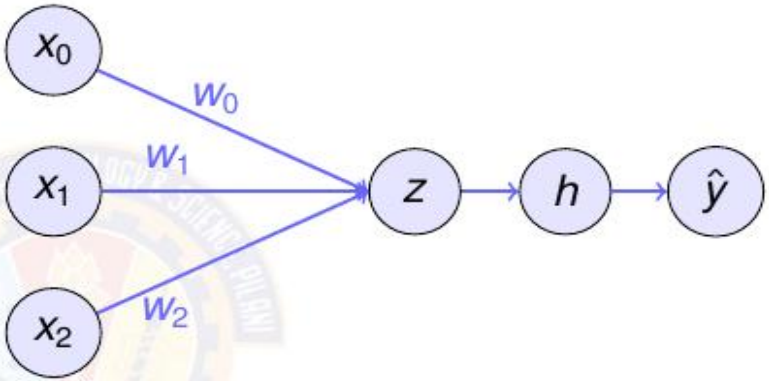


Exercise

1. Represent OR gate using Perceptron. Compute the parameters of the perceptron.
2. Represent NOR gate using Perceptron. Compute the parameters of the perceptron.
3. Represent NAND gate using Perceptron. Compute the parameters of the perceptron.

Perceptron for OR gate

- Perceptron equation is $\hat{y} = w_0 x_0 + w_1 x_1 + w_2 x_2$.
- $h > 0$ for output to be 1.
- For each row of the truth table, the equations are given.
- Solve for the inequalities.
- One solution is $w_0 = w_1 = w_2 = 2$.
- This give a beautiful linear decision boundary.

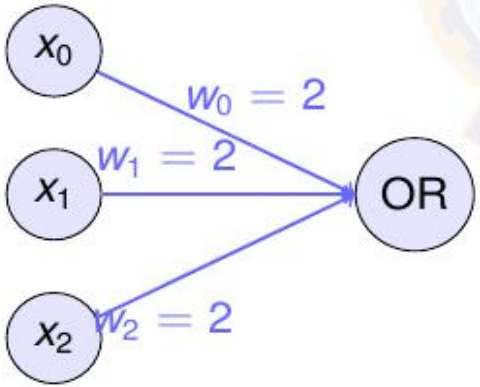


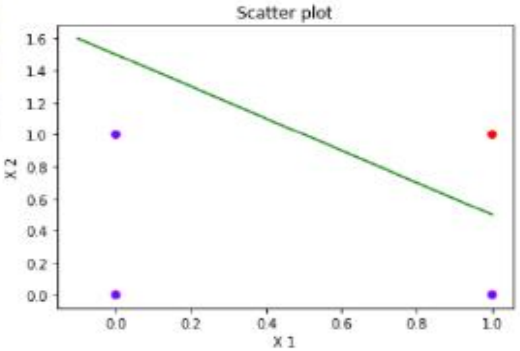
x_1	x_2	t	h
-1	-1	-1	$w_0 + w_1(-1) + w_2(-1) < 0$
-1	1	-1	$w_0 + w_1(-1) + w_2(1) > 0$
1	-1	-1	$w_0 + w_1(1) + w_2(-1) > 0$
1	1	1	$w_0 + w_1(1) + w_2(1) > 0$

Perceptron for OR gate

A	B	AB
0	0	0
0	1	1
1	0	1
1	1	1

DECISION BOUNDARY



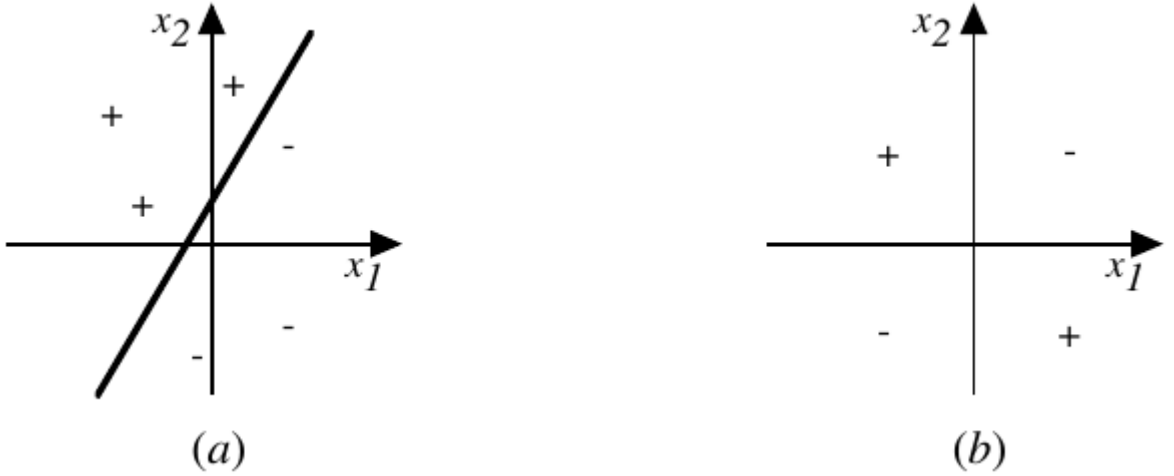


Exercise

1. Represent OR gate using Perceptron. Compute the parameters of the perceptron using perceptron learning algorithm.
2. Represent AND gate using Perceptron. Compute the parameters of the perceptron using perceptron learning algorithm.

Representational Power of Perceptrons

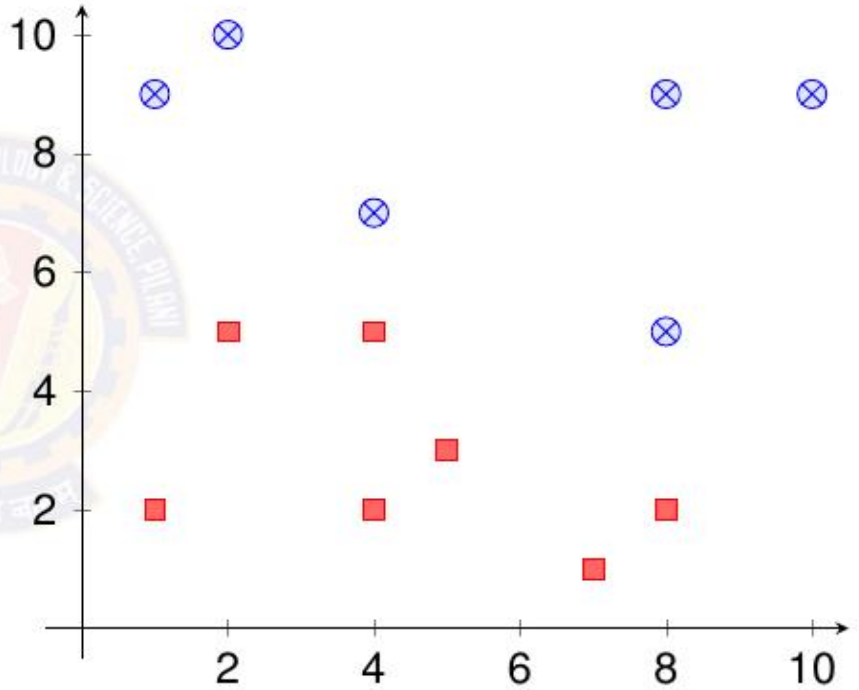
- A perceptron represents a hyperplane decision surface in the n-dimensional space of examples.
- The perceptron outputs a 1 for examples lying on one side of the hyperplane and outputs a -1 for examples lying on the other side.



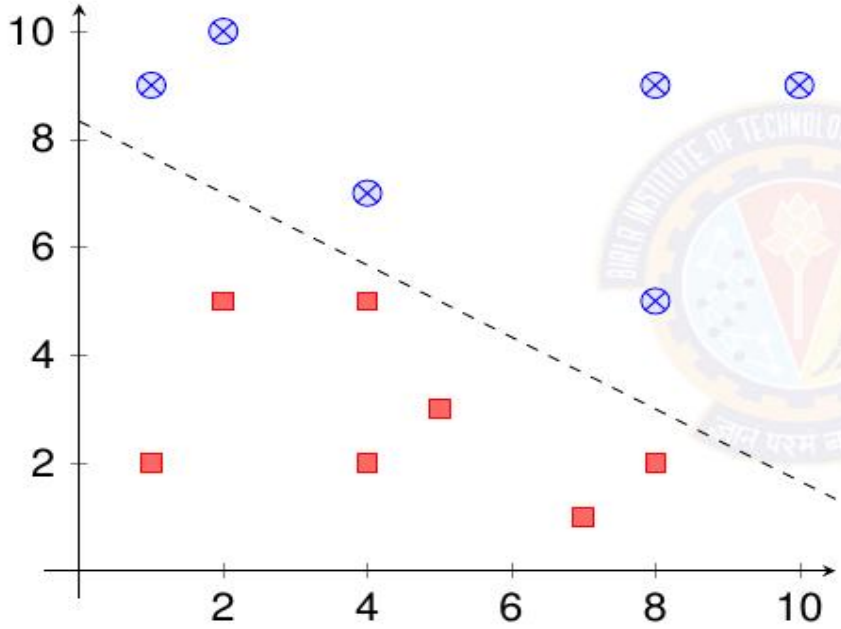
Single Layer Perceptron

Example of Linearly Separable data

x_1	x_2	y
1	9	white
10	9	white
4	7	white
4	5	pink
5	3	pink
8	9	white
4	2	pink
2	5	pink
7	1	pink
2	10	white
8	5	white
1	2	pink
8	2	pink



Example of Linearly Separable data



- Linearly separable data.
- Line equation say

$$2x_1 + 3x_2 - 25 = 0$$

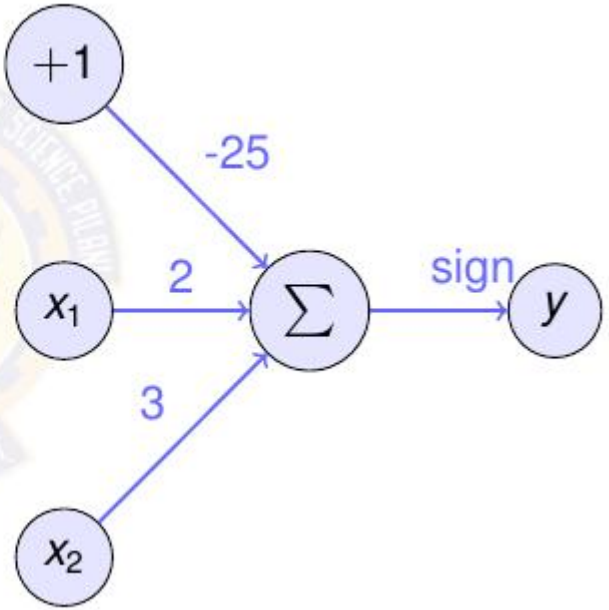
- The line becomes the decision boundary.
- New point $(0, 0) < 25$ so pink interior.
- New point $(10, 10) > 25$ so white interior.

Example of Linearly Separable data

- Linearly separable data.
- Line equation say

$$2x_1 + 3x_2 - 25 = 0$$

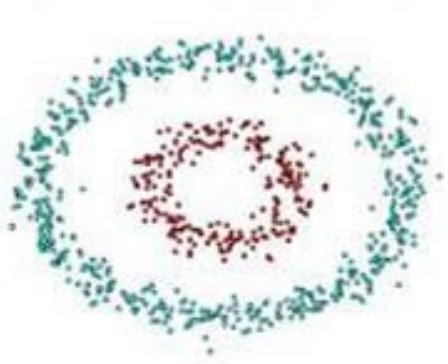
- The line becomes the decision boundary.

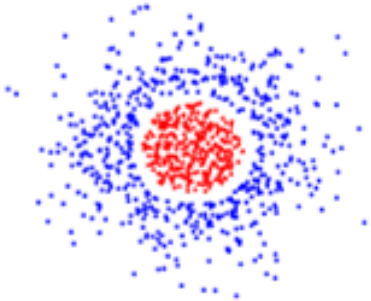


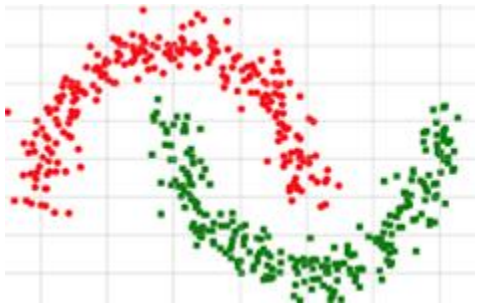
Non-Linearly Separable Data

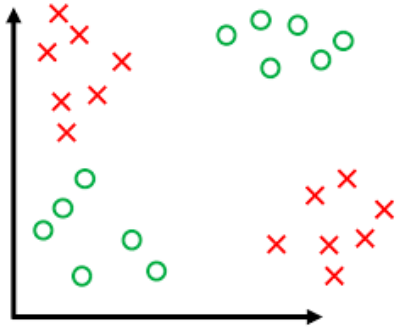
Non-linearly Separable Data

- Two groups of data points are non-linearly separable in a 2-dimensional space if they cannot be easily separated with a linear line.









Perceptron for XOR Gate

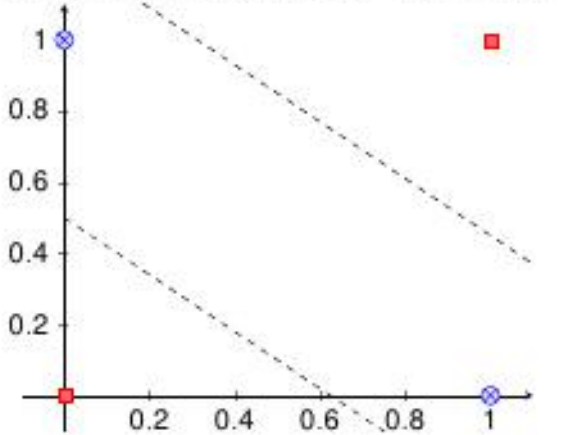
Question:

- How to represent XOR gate using a Perceptron?
- What are the parameters for the XOR Perceptron?
- Data is given below.

A	B	AB
0	0	0
0	1	1
1	0	1
1	1	0

Challenge: Data is non-linearly separable.

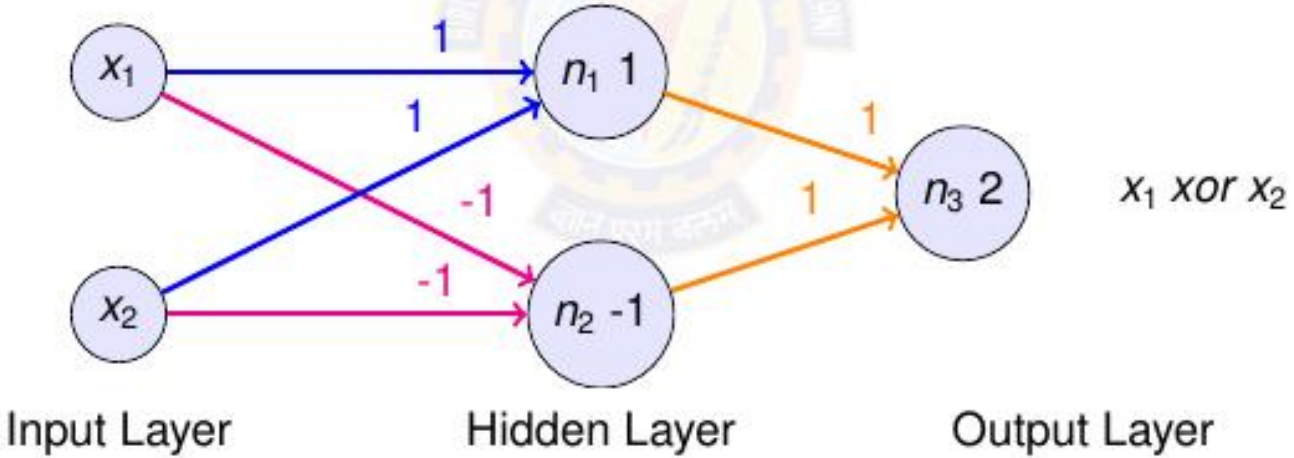
Decision boundary for XOR



Solution for XOR

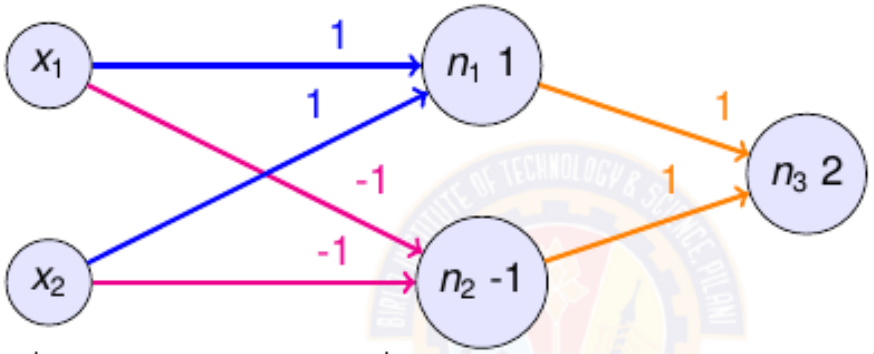
Qn: How to represent XOR gate using a Perceptron?

- Use Multilayer Perceptron (MLP)
- Introduce another layer in between the input and output.
- This in-between layer is called hidden layer.



x_1 *xor* x_2

MLP for XOR



x_1 *xor* x_2

x_1	x_2	n_1	n_2	n_3
0	0	$0 \cdot 1 + 0 \cdot 1 = 0 \not\geq th$ $n_1 = 0$	$0 \cdot (-1) + 0 \cdot (-1) = 0 > th$ $n_2 = 1$	$0 \cdot 1 + 1 \cdot 1 = 1 \not\geq th$ $n_3 = 0$
0	1	$0 \cdot 1 + 1 \cdot 1 = 1 \geq th$ $n_1 = 1$	$0 \cdot (-1) + 1 \cdot (-1) = -1 \geq th$ $n_2 = 1$	$1 \cdot 1 + 1 \cdot 1 = 2 \geq th$ $n_3 = 1$
1	0	$1 \cdot 1 + 0 \cdot 1 = 1 \geq th$ $n_1 = 1$	$1 \cdot (-1) + 0 \cdot (-1) = -1 \geq th$ $n_2 = 1$	$1 \cdot 1 + 1 \cdot 1 = 2 \geq th$ $n_3 = 1$
1	1	$1 \cdot 1 + 1 \cdot 1 = 2 \geq th$ $n_1 = 1$	$1 \cdot (-1) + 1 \cdot (-1) = -2 \not\geq th$ $n_2 = 0$	$1 \cdot 1 + 0 \cdot 1 = 1 \not\geq th$ $n_3 = 0$

Ref:
Chapter 4 of Book: Machine Learning by Tom M. Mitchell

Thank You All!

What is PyTorch?

- **PyTorch** is an **open-source machine learning and deep learning framework** developed by **Facebook's AI Research Lab (FAIR)**.
- It helps you **build, train, and deploy neural networks** easily using Python.

You can use PyTorch for:

- **Machine Learning Models** (e.g., regression, classification)
- **Deep Learning Models**
  - Image Recognition (CNNs)
  - Text Processing (RNNs, Transformers)
  - Generative AI (GANs, diffusion models)
- **Transfer Learning**
- **Reinforcement Learning**
- **Building Chatbots, AI agents, NLP systems**

Sample Code


```
import torch
import torch.nn as nn
import torch.optim as optim

# Example: Simple Neural Network
class SimpleModel(nn.Module):
    def __init__(self):
        super(SimpleModel, self).__init__()
        self.linear = nn.Linear(2, 1)  # 2 inputs -> 1 output

    def forward(self, x):
        return self.linear(x)

# Create model
model = SimpleModel()
```

```
# Dummy data
inputs = torch.tensor([[1.0, 2.0], [3.0, 4.0]])
targets = torch.tensor([[5.0], [11.0]])

# Define loss and optimizer
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Train
for epoch in range(100):
    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, targets)
    loss.backward()
    optimizer.step()

print("Final Loss:", loss.item())
```