EE-508: Hardware Foundations for Machine Learning

Lecture 1

University of Southern California

Ming Hsieh Department of Electrical and Computer Engineering

Instructor:
Arash Saifhashemi

Is this only a hardware class?

Is this only a hardware class?

- No! We will cover hardware, software, and systems for machine learning, but there will be a lot of focus on hardware.
- This course evolved from an EE599 course on Systems for Machine Learning.

What background information do I need to have?

- **Probability**
  - Basic concepts (events, random variables, etc.)
  - Probability distributions (e.g., Gaussian, Bernoulli)
  - Conditional probability and Bayes' Theorem
- **Statistics**
  - Measures of central tendency (mean, median, mode)
  - Variance and standard deviation
  - Hypothesis testing and confidence intervals
- **Python**
  - Writing and debugging Python code
  - Libraries for data science and machine learning (e.g., NumPy, pandas, matplotlib)
  - Familiarity with Jupyter notebooks (optional but helpful)
- **Computer Architecture**
  - Basic CPU architecture (e.g., cores, threads)
  - Memory organization (RAM, cache, storage)
  - Instruction sets and data representation

What background information do I need to have?

- **Linear Algebra**
  - Vector and matrix operations
  - Concepts like dot products and norms
- **Calculus**
  - Differentiation (e.g., gradients, partial derivatives)
  - Integration (basic concepts)
  - Chain rule (important for backpropagation in neural networks)
- **Optimization Basics**
  - Gradient descent and its variants (e.g., SGD, Adam)
  - Convex vs. non-convex optimization
- **Basic Programming Concepts**
  - Understanding algorithms and pseudocode
  - Data structures (lists, dictionaries, sets)
  - Basic debugging techniques
- **Algorithms and Complexity**
  - Big-O notation for understanding computational efficiency
  - Common algorithms like sorting and searching

What do We Cover?

- **Machine Learning Algorithms**
  - Review of fundamental ML algorithms and their applications
  - Mapping ML algorithms to GPUs and massive CMPs for parallelism
  - Overview of Convolutional Neural Networks (CNNs)

What do We Cover?

- **Machine Learning Algorithms**
  - Review of fundamental ML algorithms and their applications
  - Mapping ML algorithms to GPUs and massive CMPs for parallelism
  - Overview of Convolutional Neural Networks (CNNs)
- **System Constraints and Optimization**
  - Power efficiency and design constraints in ML systems
  - Efficient data processing techniques:
    - Near-data processing to reduce data movement
    - Novel memory and storage paradigms for accelerated computation

What do We Cover?

- **Machine Learning Algorithms**
  - Review of fundamental ML algorithms and their applications
  - Mapping ML algorithms to GPUs and massive CMPs for parallelism
  - Overview of Convolutional Neural Networks (CNNs)
- **System Constraints and Optimization**
  - Power efficiency and design constraints in ML systems
  - Efficient data processing techniques:
    - Near-data processing to reduce data movement
    - Novel memory and storage paradigms for accelerated computation
- **ML Accelerators**
  - Insights into various ML hardware accelerators and their architectures
  - Integration and optimization of heterogeneous hardware models
  - Deep dive into GPU architecture and its role in ML

What do We Cover?

- **Machine Learning Algorithms**
  - Review of fundamental ML algorithms and their applications
  - Mapping ML algorithms to GPUs and massive CMPs for parallelism
  - Overview of Convolutional Neural Networks (CNNs)
- **System Constraints and Optimization**
  - Power efficiency and design constraints in ML systems
  - Efficient data processing techniques:
    - Near-data processing to reduce data movement
    - Novel memory and storage paradigms for accelerated computation
- **ML Accelerators**
  - Insights into various ML hardware accelerators and their architectures
  - Integration and optimization of heterogeneous hardware models
  - Deep dive into GPU architecture and its role in ML
- **Machine Learning Pipelines**
  - Building end-to-end workflows for ML:
    - Data preparation, feature engineering, model training, and evaluation
    - Automating and scaling pipelines with modern tools (e.g., TFX, MLflow)

Attendance

- You should Participate **in-person**
  - Will be used for curving
  - If you are not able to come in person, try to participate **synchronously on Zoom** with my **approval**
- You should go to **discussion sessions**!

Holidays and Special Events

- We will observe USC holidays that fall on **Mondays**.
  - Martin Luther King's Birthday Holiday
  - President day
- Special Events
  - When there are special events on campus, due to high traffic, **we will have online lectures instead of in-person**. You are required to participate synchronously.
  - Examples:
    - Los Angeles Times Festival of Books
    - Concerts
    - USC Home Football Tailgates

Guest Lectures

- Occasionally we might have guest lectures
  - Please make sure you attend either in person or online

Grading

Assessment Tool (assignments)	**Points**	**% of Grade**
Midterm and quiz 1	20	20
Programming and homework assignments	40	40
Project	20	20
Final exam and quiz 2	20	20
TOTAL	100	100

Grading

Assessment Tool (assignments)	**Points**	**% of Grade**
Midterm and quiz 1	20	20
Programming and homework assignments	**40**	**40**
Project	20	20
Final exam and quiz 2	20	20
TOTAL	100	100

HW Late Policy

1 day late is -15 points

2 days late is -30 points

3 days late is -45 points

More than 3 days late is a 0

In case of emergency, we will decide on a case-by-case basis if an extension is warranted. You may need to show documentation depending on the extension request.

Course Platform

- Piazza
  - All the lecture material will be posted here
  - Homework assignment will be released here
  - Course material
  - Office hours
- GitHub classroom
  - **Submit your homework through GitHub before the deadline**
- Birghtspace
  - **Your grades will be posted here**

Piazza

- Everyone should've received an email invite to the Piazza for this course
  - **All** questions about assignments, course material, exams, etc. should go there
- **Only** send an email if:
  - Regrade question on Programming Assignment (email the TA who graded you)
  - Personal question (such as DSP, etc.)
- **Try** to generalize questions so that they can be public (visible to everyone)
  - ..**BUT** don't put more than 5 lines of code in a public post
- If you need more code than that for context, use a private post

Academic Integrity

- All assignments should be your **individual work**
- If you are asking classmates questions beyond the scope of what you could reasonably ask on Piazza in a “public” post, then you are not doing individual work
- **Do not** share your code files or part of your code files with your classmates (current or future students)
- **Do not** post your code on a publicly-accessible website (GitHub, course hero, etc).
- **Do not** step through anyone else’s code, if you need help debugging ask an instructor or TA or Piazza
- If you’re not sure if something is allowed: ask an instructor. Instructors/TAs are always happy to help!

An Overview of Machine Learning

History of AI Systems

- Rule Based Systems
  - Deterministic
  - Lack of learning
  - Applications:
    - Medical
    - Financial systems
    - Control systems
    - Legal advising

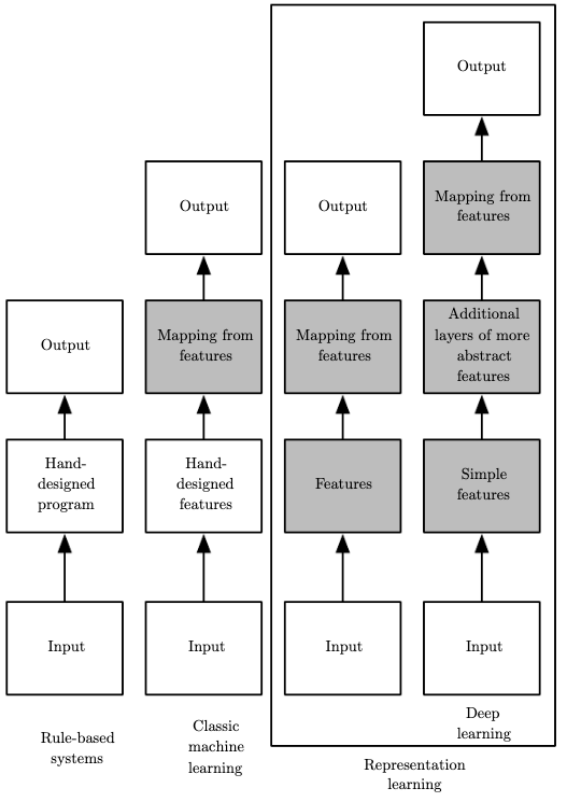
```
IF sky is clear
AND
temperature is low
THEN chance of frost
is high
```

```
% Facts
parent(john, jim). % John is a parent of Jim
parent(john, ann). % John is a parent of Ann
parent(jim, bob). % Jim is a parent of Bob
parent(ann, liz). % Ann is a parent of Liz

% Rule to deduce who is a grandparent of whom
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).

% Query to find out who is a grandparent of whom
?- grandparent(john, bob). true.
```

Example of Prolog Language

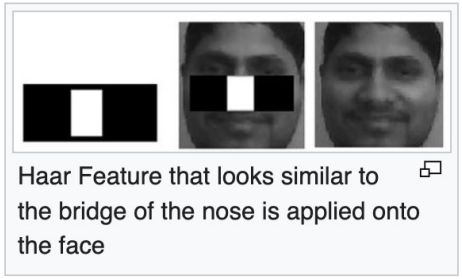


Shaded boxes indicate learnability

source: http://www.deeplearningbook.org

AI Systems

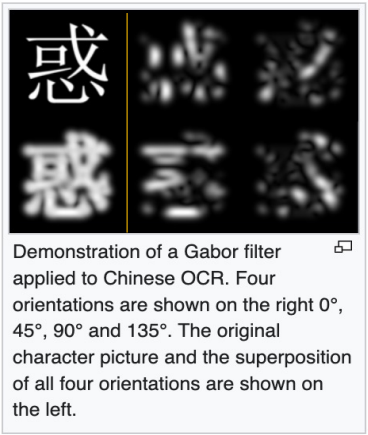
- Classic ML
  - Features are often hand-designed. Examples:
    - Data visualization
    - Transformation (e.g. normalization)
    - Feature construction(e.g. averaging)
    - Dimension Reduction
    - Feature Selection



Haar Feature that looks similar to the bridge of the nose is applied onto the face

Viola–Jones object detection

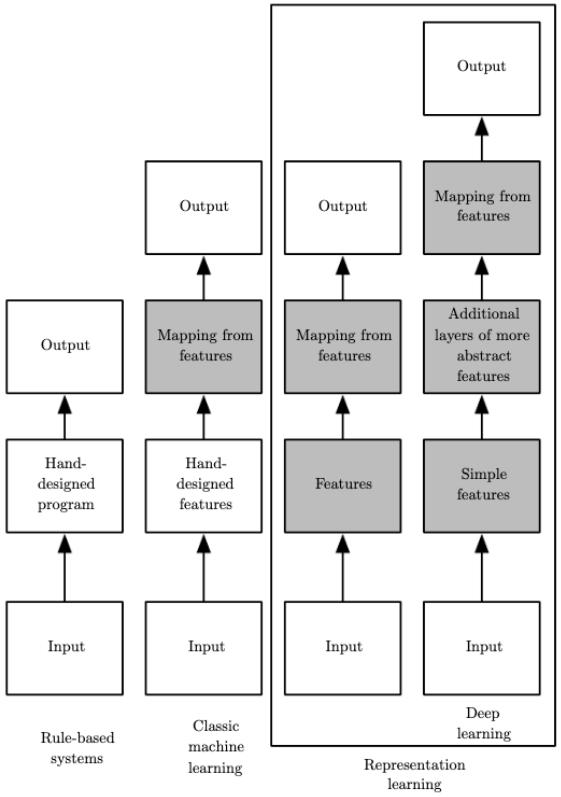
source: Wikipedia



Demonstration of a Gabor filter applied to Chinese OCR. Four orientations are shown on the right 0°, 45°, 90° and 135°. The original character picture and the superposition of all four orientations are shown on the left.

Gabor Filter

source: Wikipedia

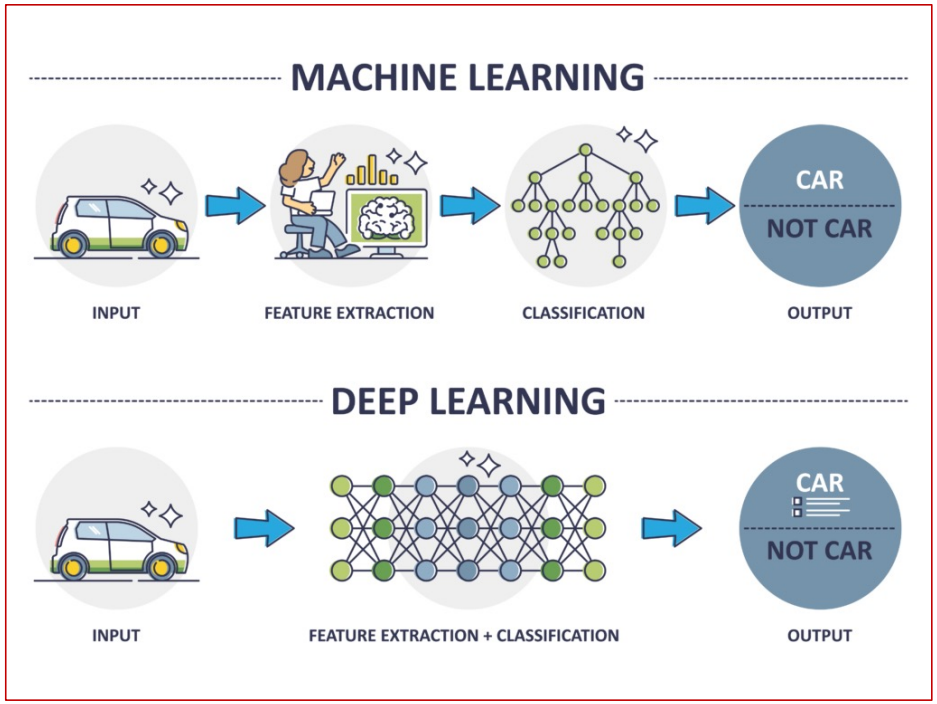


Shaded boxes indicate learnability

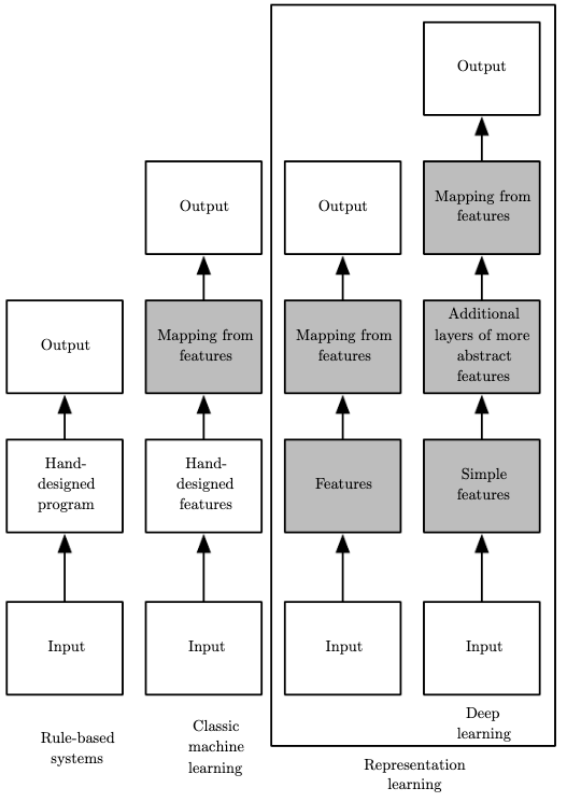
source: http://www.deeplearningbook.org

AI Systems

- Deep Learning
  - Automatic feature learning
  - End-to-End Learning: input raw data and train without manually designing intermediate features.



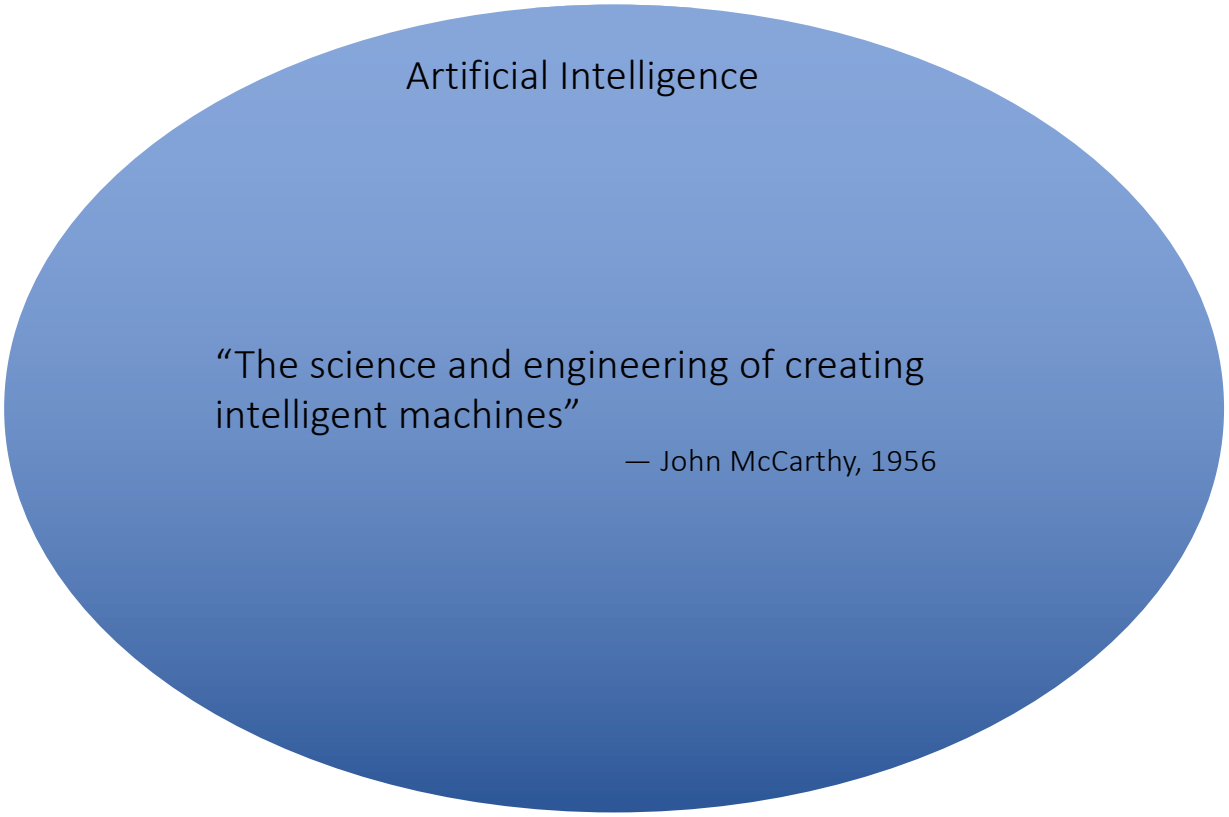
source: ait.de



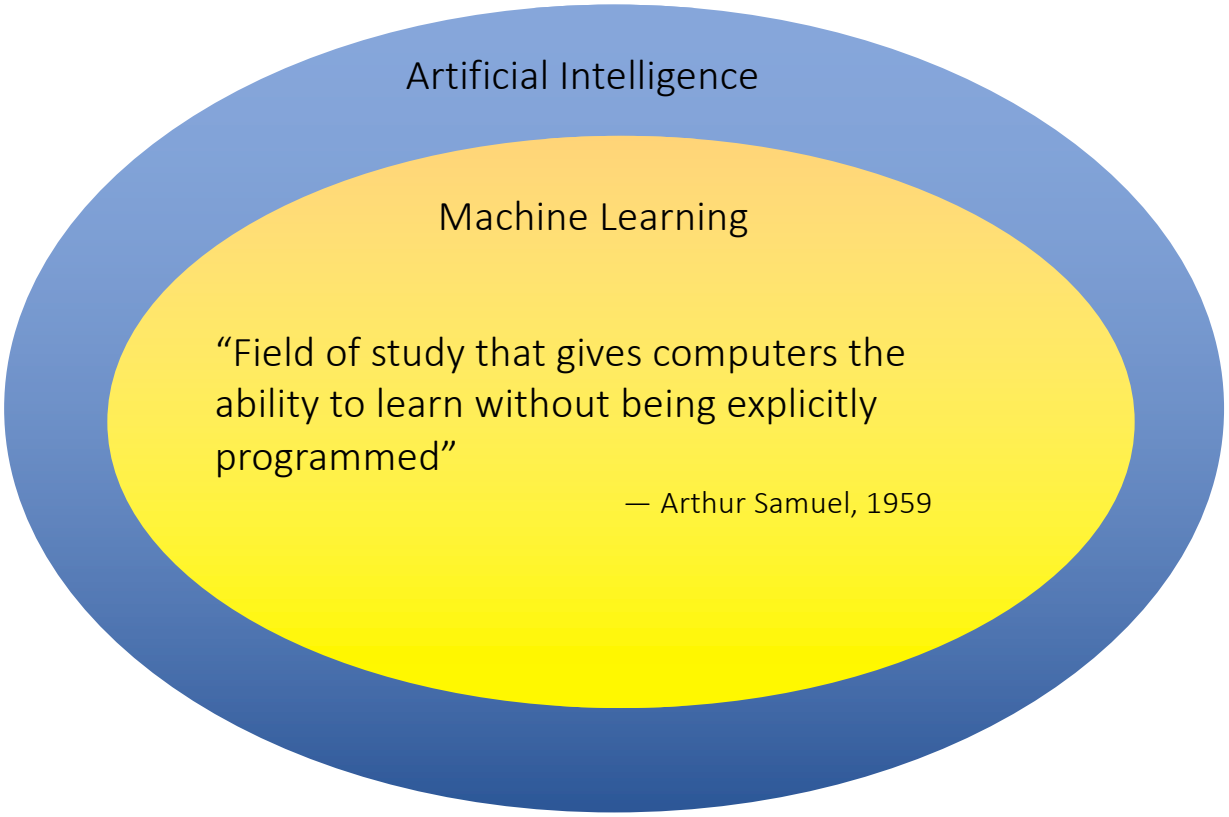
Shaded boxes indicate learnability

source: http://www.deeplearningbook.org

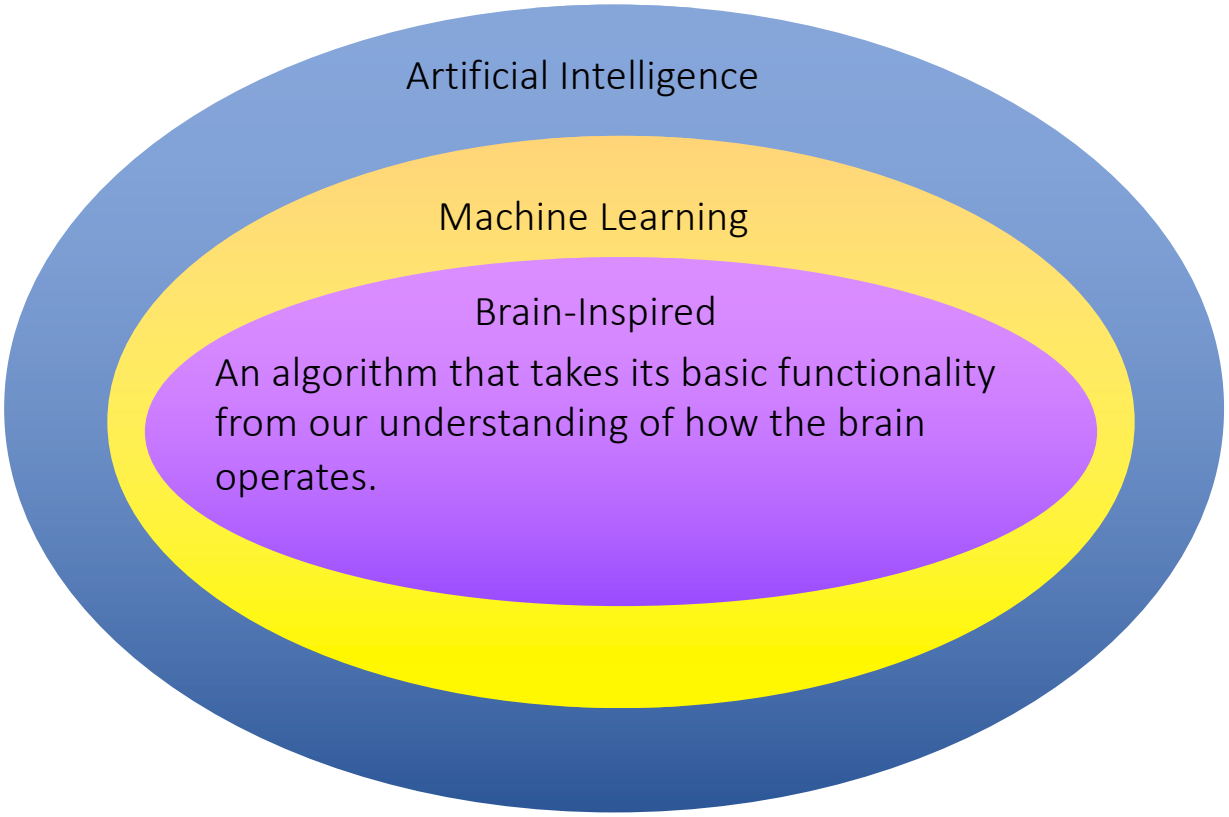
Artificial Intelligence



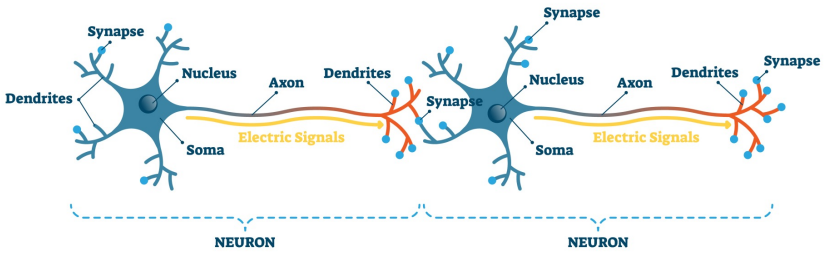
Artificial Intelligence



Artificial Intelligence



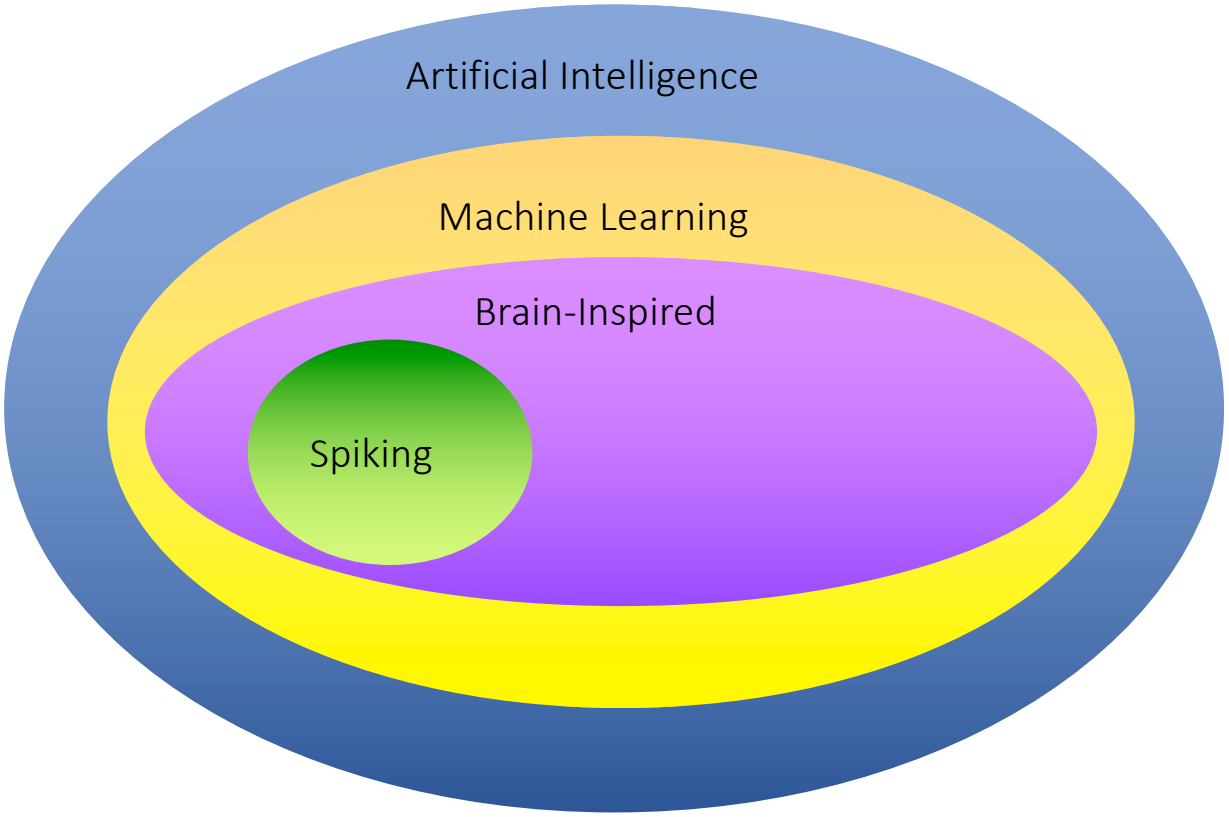
How Does the Brain Work?



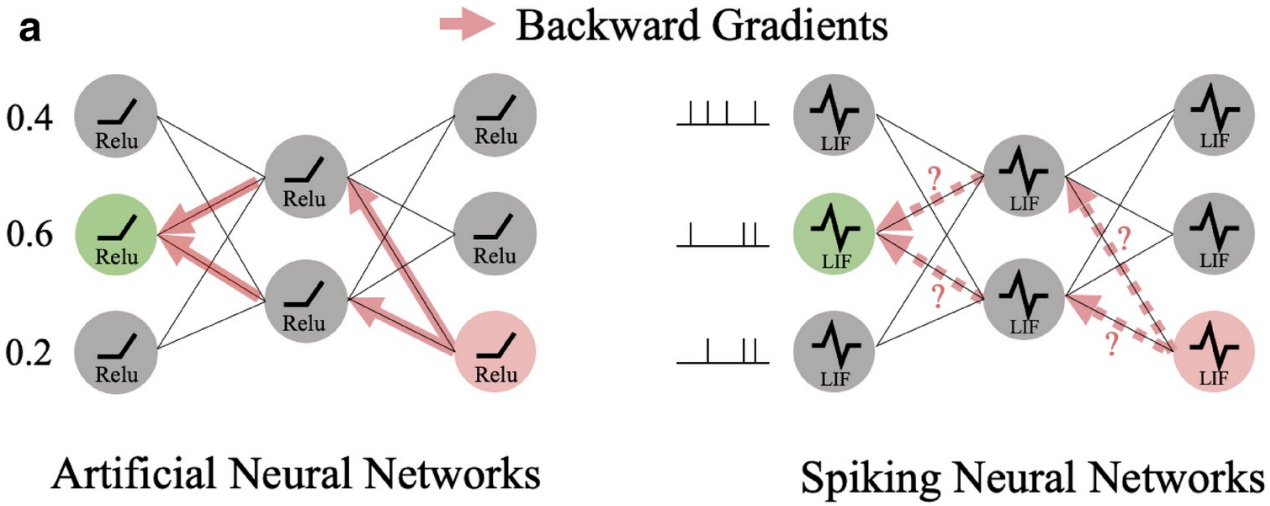
source: simplypsychology.org

- Neuron:
  - The basic computational unit of the brain
  - 86B neurons in the brain
  - Connected with nearly $10^{14} - 10^{15}$ synapses
- Receive input signal from dendrites and produce output signal along axon,
  - which interact with the dendrites of other neurons via synaptic weights
- Synaptic weights — learnable & control influence strength

Artificial Intelligence



ANN vs SNN

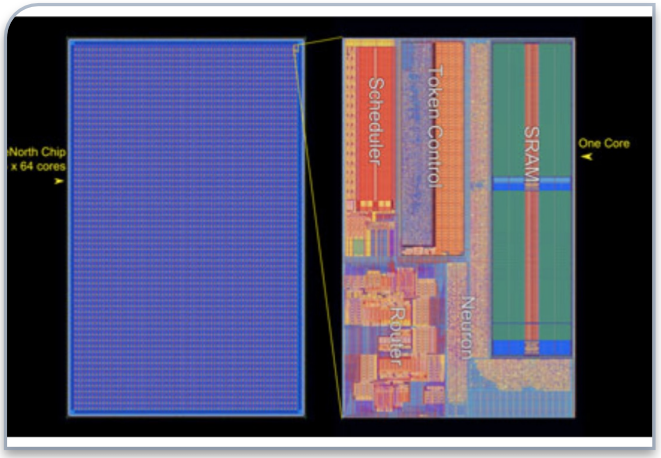


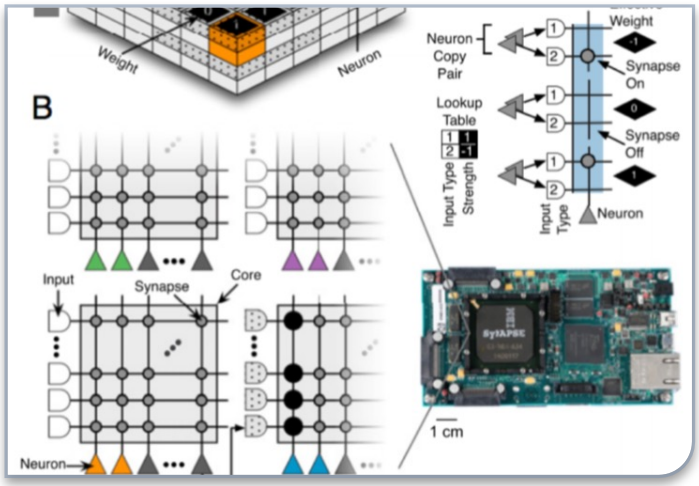
Non-differentiable spiking neural networks

source: Visual explanations from spiking neural networks using inter-spike intervals, Youngeun Kim et al.

Spiking Architecture

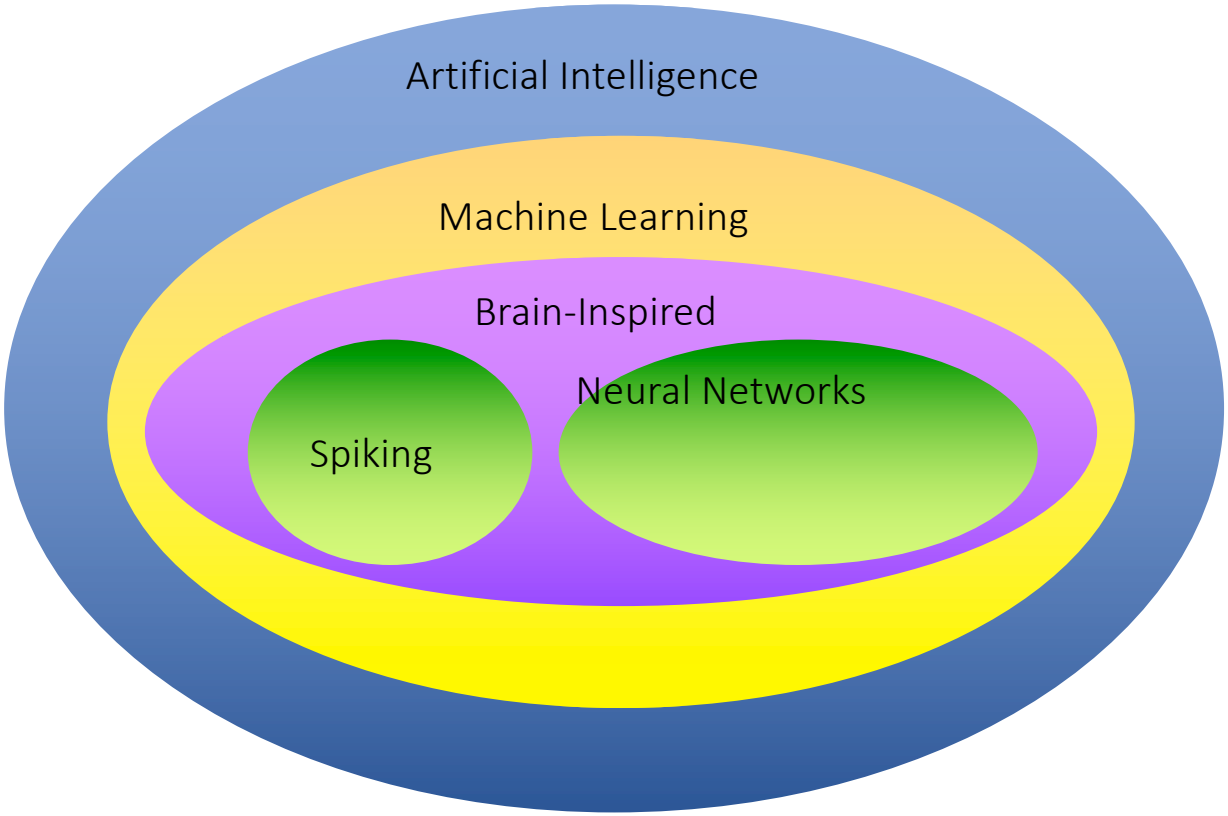
- Brain-inspired
- Integrate and fire
- Example: IBM TrueNorth
  - A neuromorphic computing architecture
  - Mimics the structure and functionality of the human brain.
  - It is energy efficient and parallel processing, making it well-suited for real-time processing tasks.

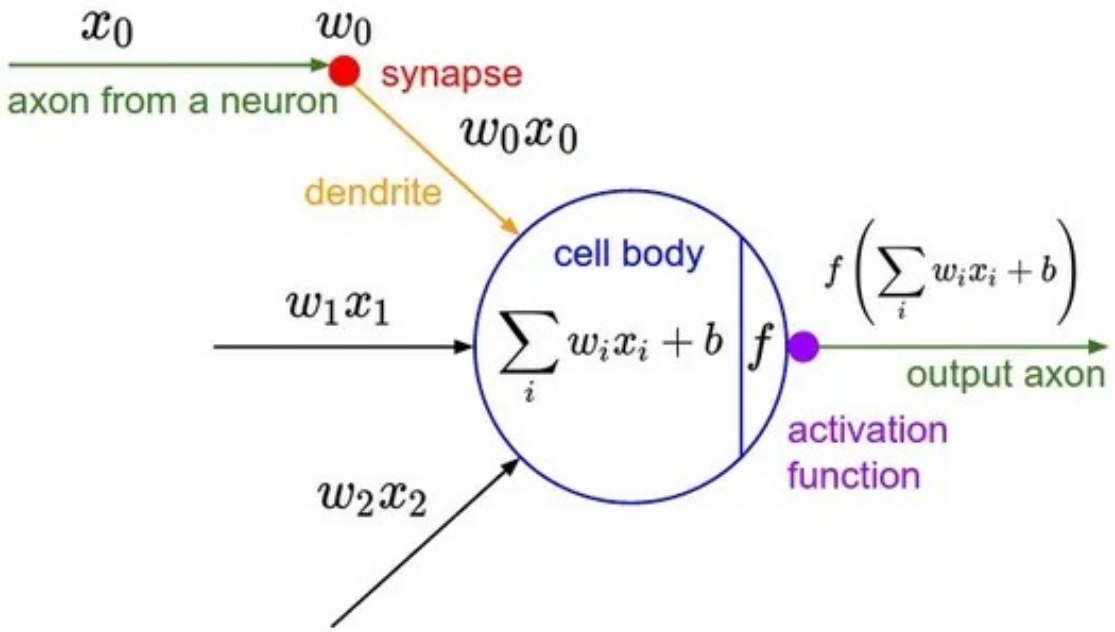




TrueNorth chip
source: pnas.org

Artificial Intelligence



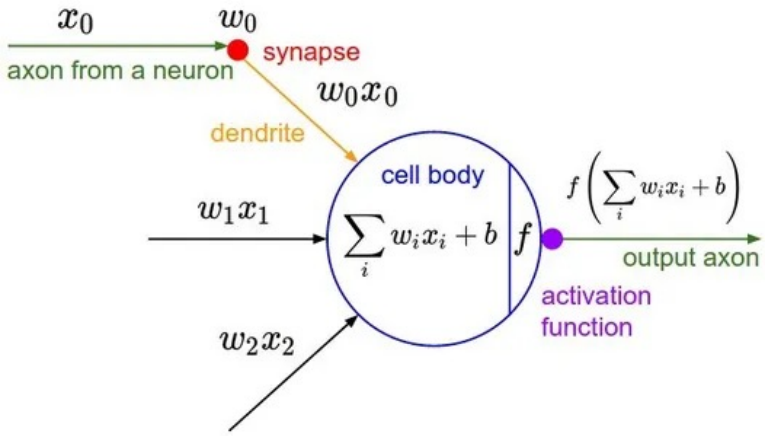


source: analyticsvidhya.com

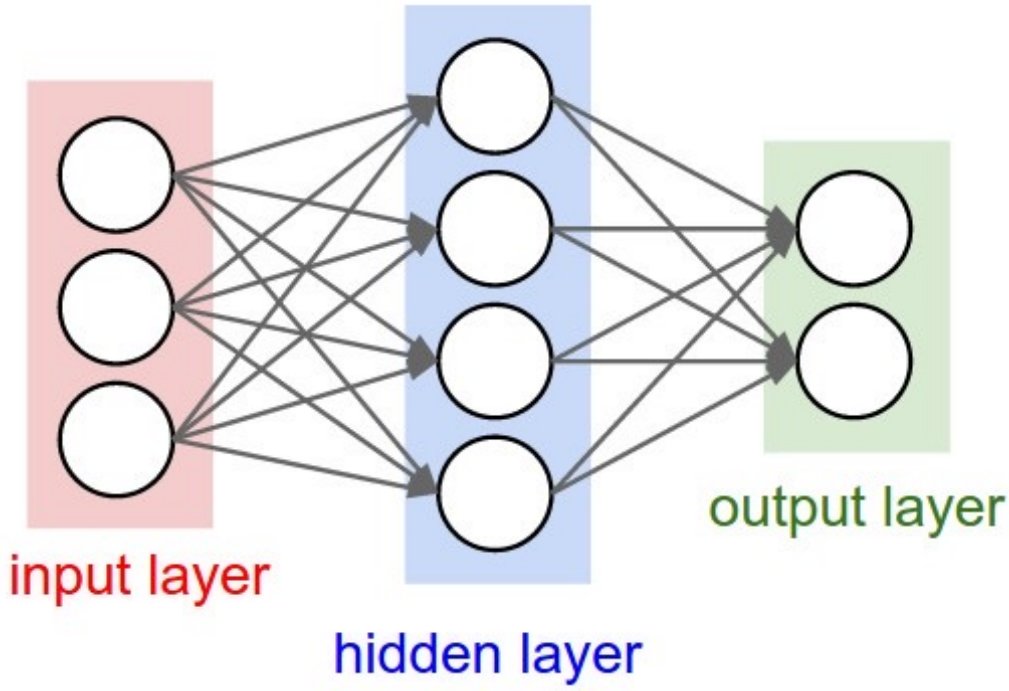
Perceptron Rosenblatt 1957

Role for Non-Linearity

- If there is no non-linearity
  - Multiple layers of weights multiplied by input activations can be reduced to a single layer
- Hence, non-linearity at the end of a layer provides the true ability of DNNs to learn complex functions

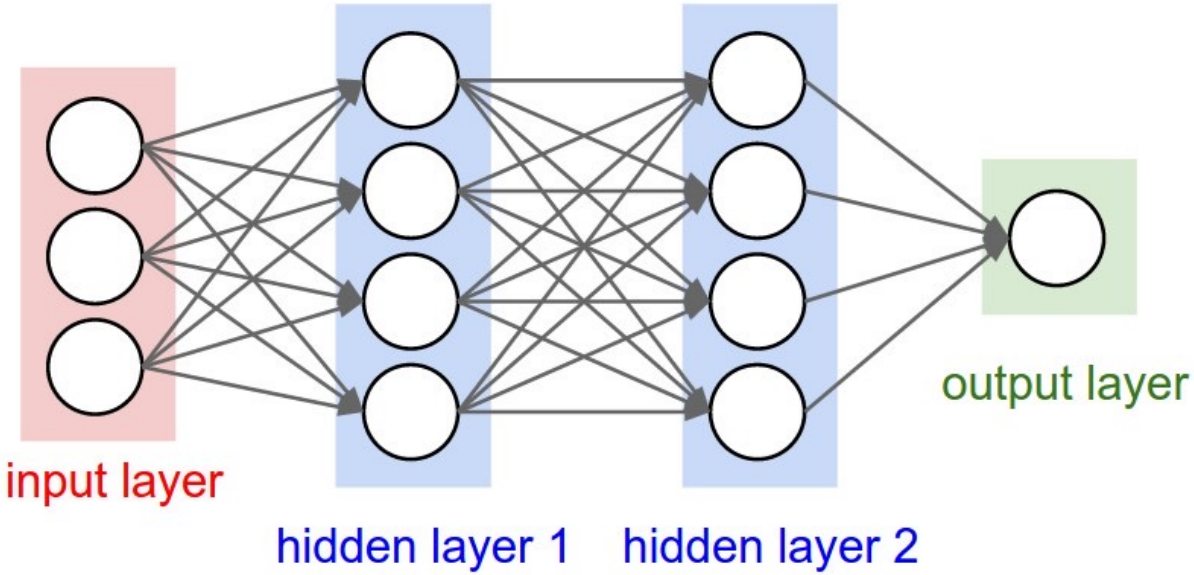


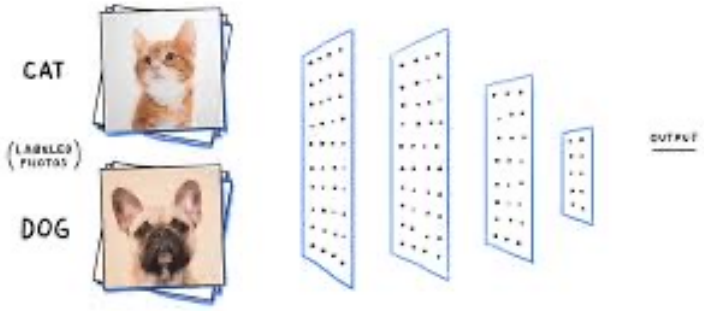
Multi-Layer Perceptron



source: cs231n.github.io

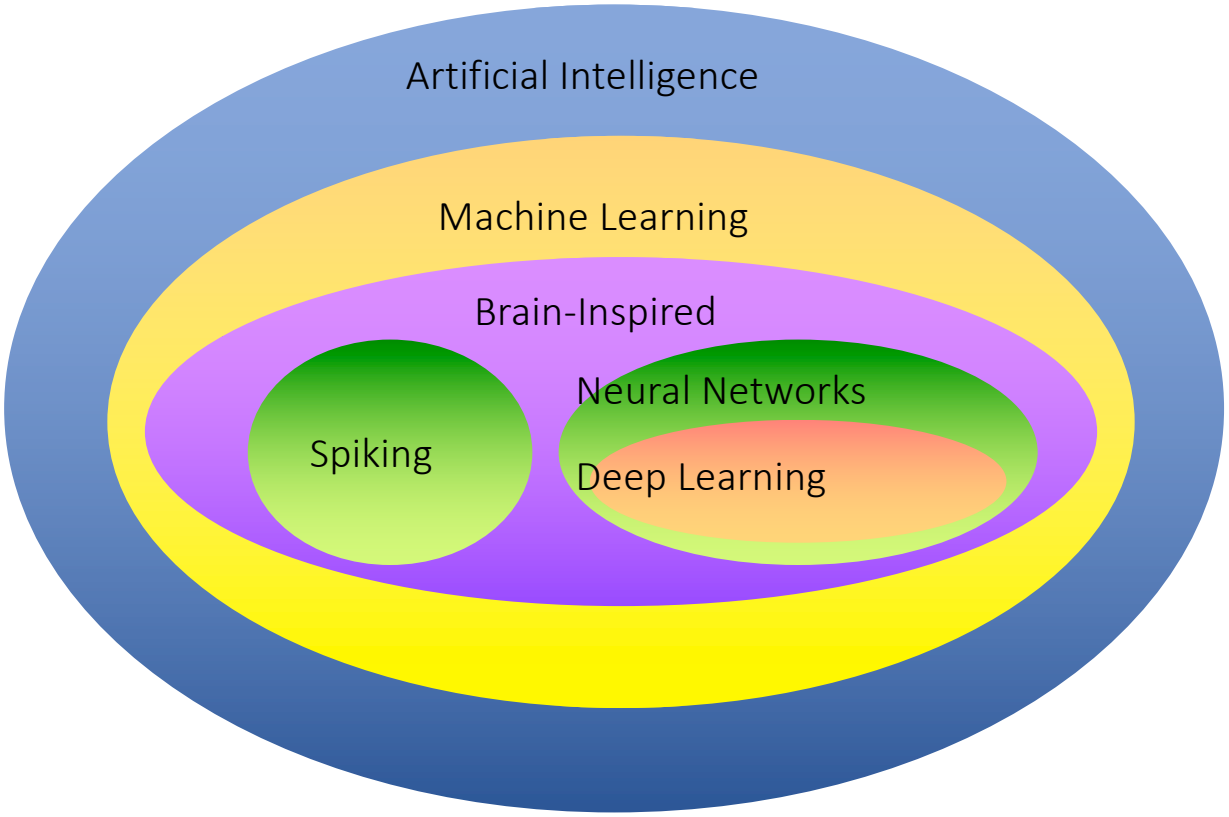
Multi-Layer Perceptron





Source: cs231n.github.io

Artificial Intelligence



Types of Learning

Supervised Learning

- **Description**: Models are trained on **labeled** data, learning to predict the output from the input data.
- **Examples**:
  - Classification tasks (e.g., spam detection).
  - Regression tasks (e.g., house price prediction).

Types of Learning

Supervised Learning

- **Description**: Models are trained on **labeled** data, learning to predict the output from the input data.
- **Examples**:
  - Classification tasks (e.g., spam detection).
  - Regression tasks (e.g., house price prediction).

Unsupervised Learning

- **Description**: Models learn patterns and structures from **unlabeled** data.
- **Examples**:
  - Clustering (e.g., customer segmentation).
  - Association (e.g., market basket analysis).

Types of Learning

Supervised Learning

- **Description**: Models are trained on **labeled** data, learning to predict the output from the input data.
- **Examples**:
  - Classification tasks (e.g., spam detection).
  - Regression tasks (e.g., house price prediction).

Unsupervised Learning

- **Description**: Models learn patterns and structures from **unlabeled** data.
- **Examples**:
  - Clustering (e.g., customer segmentation).
  - Association (e.g., market basket analysis).

Reinforcement Learning

- **Description**: Models learn to make decisions by performing actions and receiving **feedback** in the form of **rewards** or **penalties**.
- **Examples**:
  - Gameplay strategies (e.g., AlphaGo).
  - Robotics (e.g., pathfinding algorithms).

Types of Learning

Supervised Learning

- **Description**: Models are trained on **labeled** data, learning to predict the output from the input data.
- **Examples**:
  - Classification tasks (e.g., spam detection).
  - Regression tasks (e.g., house price prediction).

Unsupervised Learning

- **Description**: Models learn patterns and structures from **unlabeled** data.
- **Examples**:
  - Clustering (e.g., customer segmentation).
  - Association (e.g., market basket analysis).

Reinforcement Learning

- **Description**: Models learn to make decisions by performing actions and receiving **feedback** in the form of **rewards** or **penalties**.
- **Examples**:
  - Gameplay strategies (e.g., AlphaGo).
  - Robotics (e.g., pathfinding algorithms).

Generative Learning

- **Description**: Models learn to **generate new data** that's similar to the training data.
- **Examples**:
  - Generative Adversarial Networks (GANs) for image generation.
  - Text generation (e.g., chatbots).

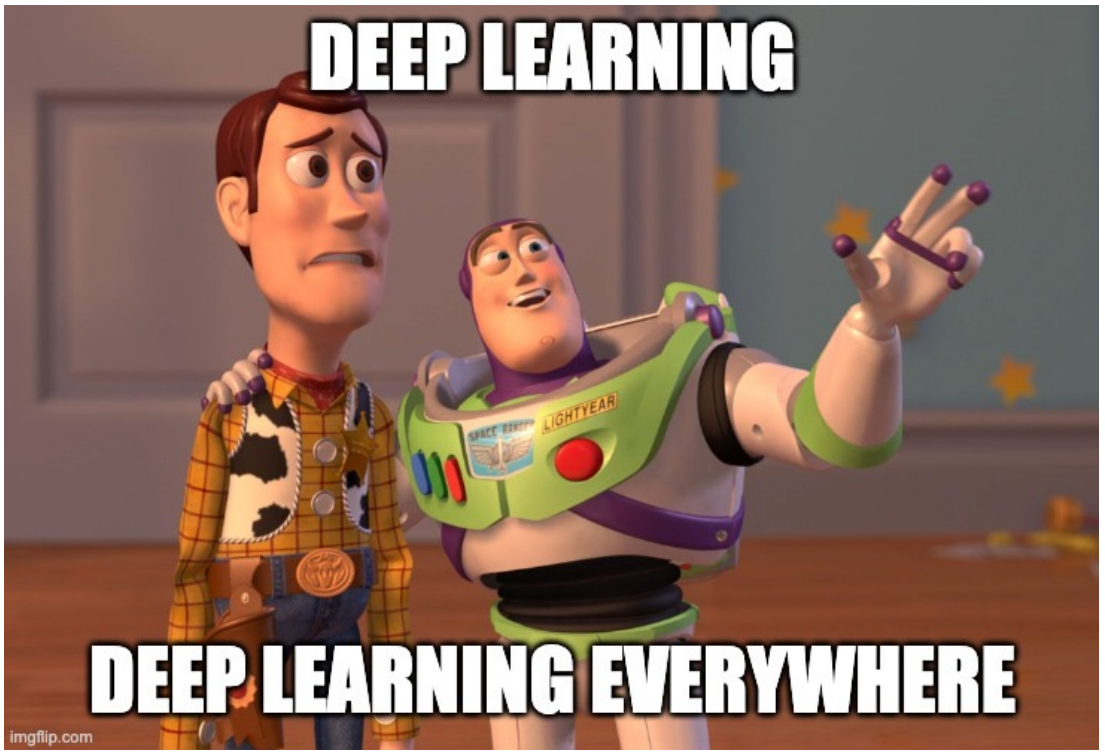
Machine Learning Terminology

Term	Explanation	Example
Algorithm	A set of rules or processes followed in calculations or problem-solving.	Decision Tree, Support Vector Machine
Model	A mathematical representation of what a machine learning system has learned.	A trained neural network
Training	The process of fitting a machine learning model to a dataset.	Using house prices to train a regression model
Inference	The process of making predictions using a trained model.	Predicting the price of a house
Feature	An individual measurable property or characteristic of a phenomenon.	Pixel values in an image, words in a document
Feature Engineering	The process of using domain knowledge to create features that make ML algorithms work.	Creating a 'time of day' feature from timestamp data
Label	In supervised learning, the output or target variable that the model is trying to predict.	The 'spam' or 'not spam' designation in a spam detection task
Classification	A type of supervised learning where the output is a category.	Identifying if an email is spam or not
Regression	A type of supervised learning where the output is a continuous value.	Predicting the selling price of a house
Accuracy	A measure of a model's performance, typically used for classification models.	The percentage of emails correctly classified as spam or not
Loss Function	A method of evaluating how well a machine learning model performs.	Mean Squared Error for regression tasks
Overfitting	A modeling error that occurs when a model is too complex and captures the noise in the training data.	A model that performs well on training data but poorly on unseen data
Underfitting	A scenario where a machine learning model is too simple to capture the underlying pattern in the data.	A linear model used for a non-linear data distribution
Cross-Validation	A technique for evaluating ML models by partitioning the original sample into a training set and a test set.	k-fold cross-validation
Hyperparameters	The configuration settings used to structure machine learning models.	The learning rate in neural networks
Epoch	One complete presentation of the entire dataset to the learning algorithm during training.	Running through all the images in a dataset once while training a CNN

Additional Machine Learning Terminology…

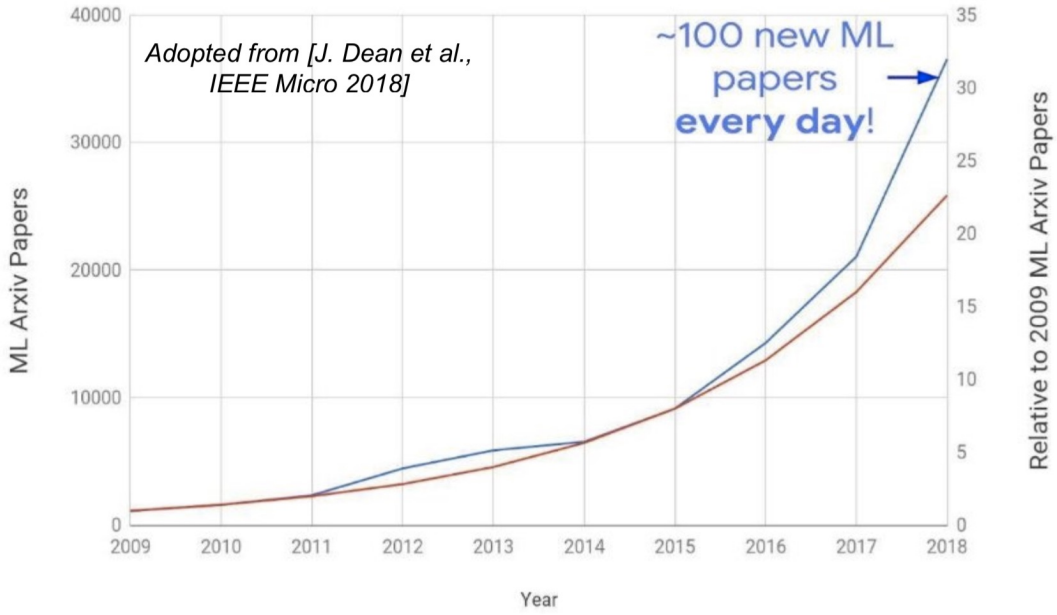
Term	Explanation	Example
Edge AI Device	A physical device with limited computing power and resources that runs AI algorithms locally.	Smartphone, intelligent camera, wearable device
Edge Computing	Performing data processing and analysis at the edge of the network, close to the data source.	Analyzing sensor data directly on a robot instead of sending it to the cloud
Federated Learning	Training a machine learning model collaboratively across multiple edge devices while preserving data privacy.	Each smartphone contributing to a spam detection model without sharing individual email data
Low-latency Inference	Making predictions with an AI model in real-time with minimal delay.	Analyzing drone video to avoid obstacles as it flies
Resource-constrained Environment	Operating with limited processing power, memory, and battery life.	Running an anomaly detection model on a wearable device with minimal power consumption

Why is Deep Learning so Popular Nowadays?



Why ML?

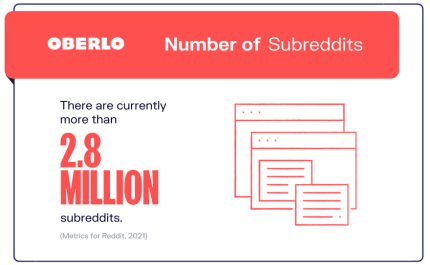
- ML's meteoric raise
- After 2 hype-disillusionment cycles, the Third Wave has crossed the threshold

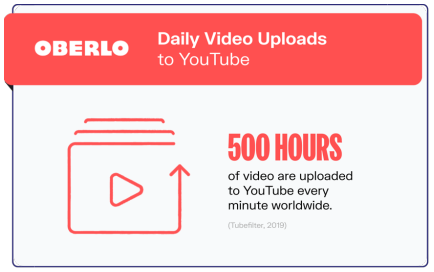


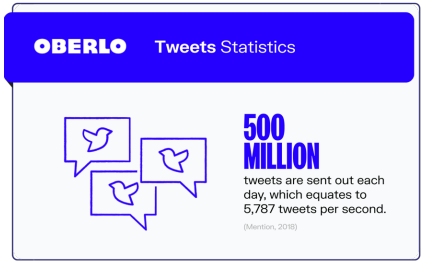
Why is Deep Learning so Popular Nowadays?

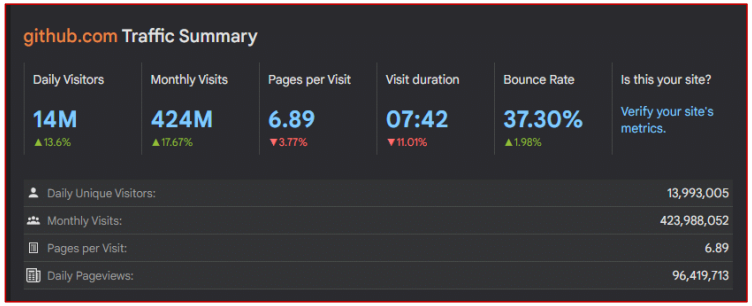
1. Tons of Data

Social Network	Active Users
Facebook	2.853 billion
YouTube	2.291 billion
WhatsApp	2 billion
Instagram	1.386 billion
Facebook Messenger	1.3 billion
Weixin / WeChat	1.242 billion
TikTok	1 billion
QQ	606 million
Douyin	600 million
Telegram	550 million
Sina Weibo	530 million
Snapchat	514 million
Kuaishou	481 million
Pinterest	478 million
Reddit	430 million
Twitter	397 million
Quora	300 million





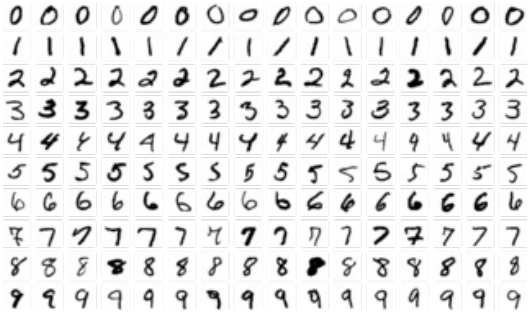


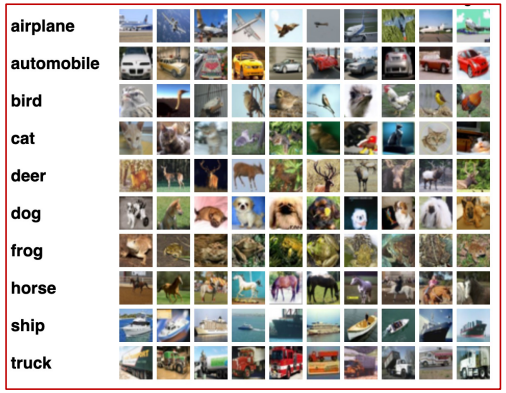


Why is Deep Learning so Popular Nowadays?

Availability of Datasets For Training

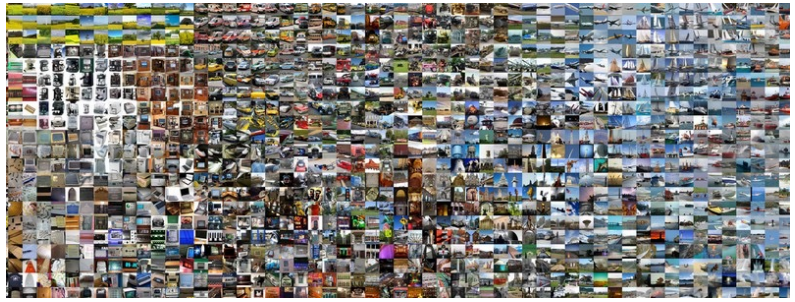
- Some of datasets in `torchvision.datasets`
  - **Image Classification Datasets**:
    - CIFAR10: 60,000 images in 10 classes for classification.
    - CIFAR100: Similar to CIFAR10 but with 100 classes.
    - MNIST: Handwritten digits dataset.
    - FashionMNIST: Fashion product images, often used as an alternative to MNIST.
    - EMNIST: Extended MNIST with more classes like letters.
    - STL10: An image recognition dataset for developing unsupervised feature learning.
    - SVHN: Street View House Numbers, real-world images of house numbers from Google Street View.
  - **Object Detection and Segmentation Datasets**:
    - CocoCaptions: COCO dataset with image caption annotations.
    - CocoDetection: COCO dataset for object detection.
    - VOCDetection: PASCAL VOC dataset used for object detection.
    - VOCSegmentation: PASCAL VOC dataset but with segmentation masks.
    - Cityscapes: For semantic urban scene understanding.
  - **Transfer Learning and Fine-tuning**:
    - ImageNet: A large scale dataset for general-purpose image classification.
    - Caltech101: Images in 101 categories for fine-tuning and transfer learning.
    - Caltech256: Extension of Caltech101 with more categories.



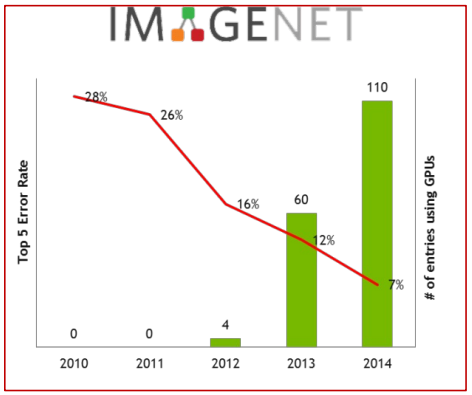


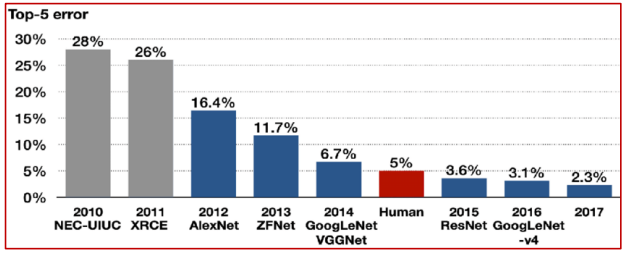
https://pytorch.org/vision/0.8/datasets.html

Why is Deep Learning so Popular Nowadays?



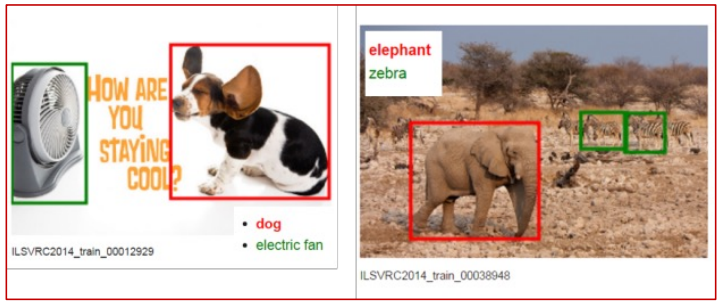
ImageNet



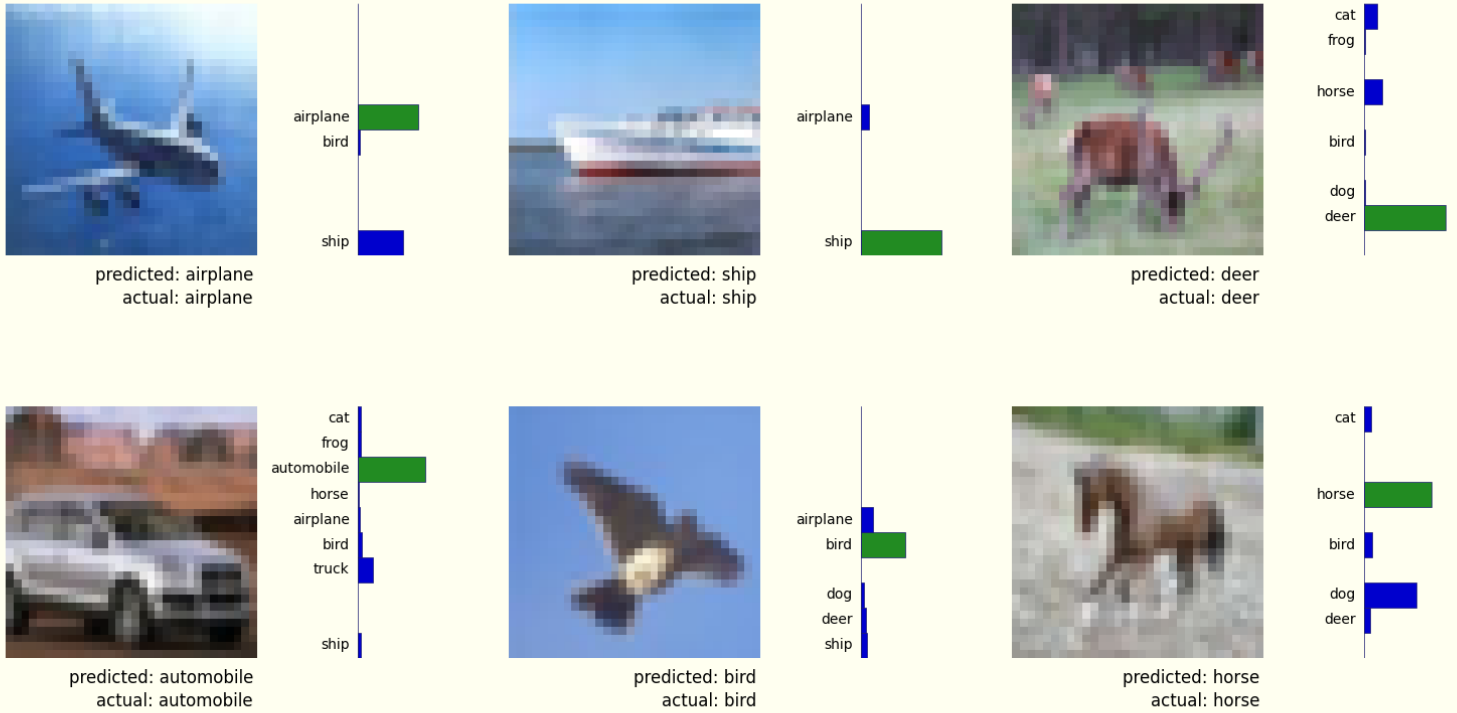


Source: NVDIA

- 14 million hand-annotated images.
- Over 20,000 categories
- **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**:
  - Annual competition held from 2010 to 2017.
  - Notable for its role in the rise of deep learning, especially
    the Convolutional Neural Network (CNN) model "AlexNet" in 2012,
    which significantly outperformed previous methods.



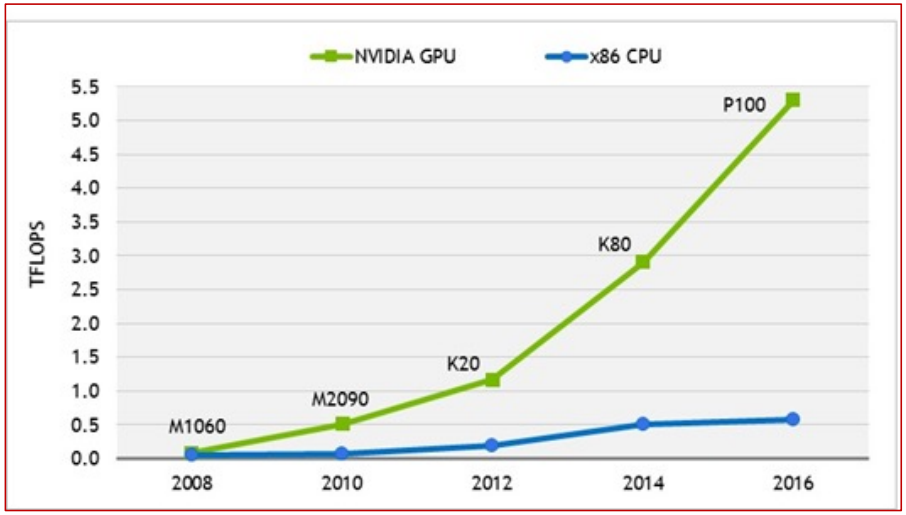
Examples of ImageNet images demonstrating classification with localization.


airplane
bird
ship
predicted: airplane
actual: airplane
airplane
ship
predicted: ship
actual: ship
cat
frog
horse
bird
dog
deer
predicted: deer
actual: deer
cat
frog
automobile
horse
airplane
bird
truck
ship
predicted: automobile
actual: automobile
airplane
bird
dog
deer
ship
predicted: bird
actual: bird
cat
horse
bird
dog
deer
predicted: horse
actual: horse

Why is Deep Learning so Popular Nowadays?

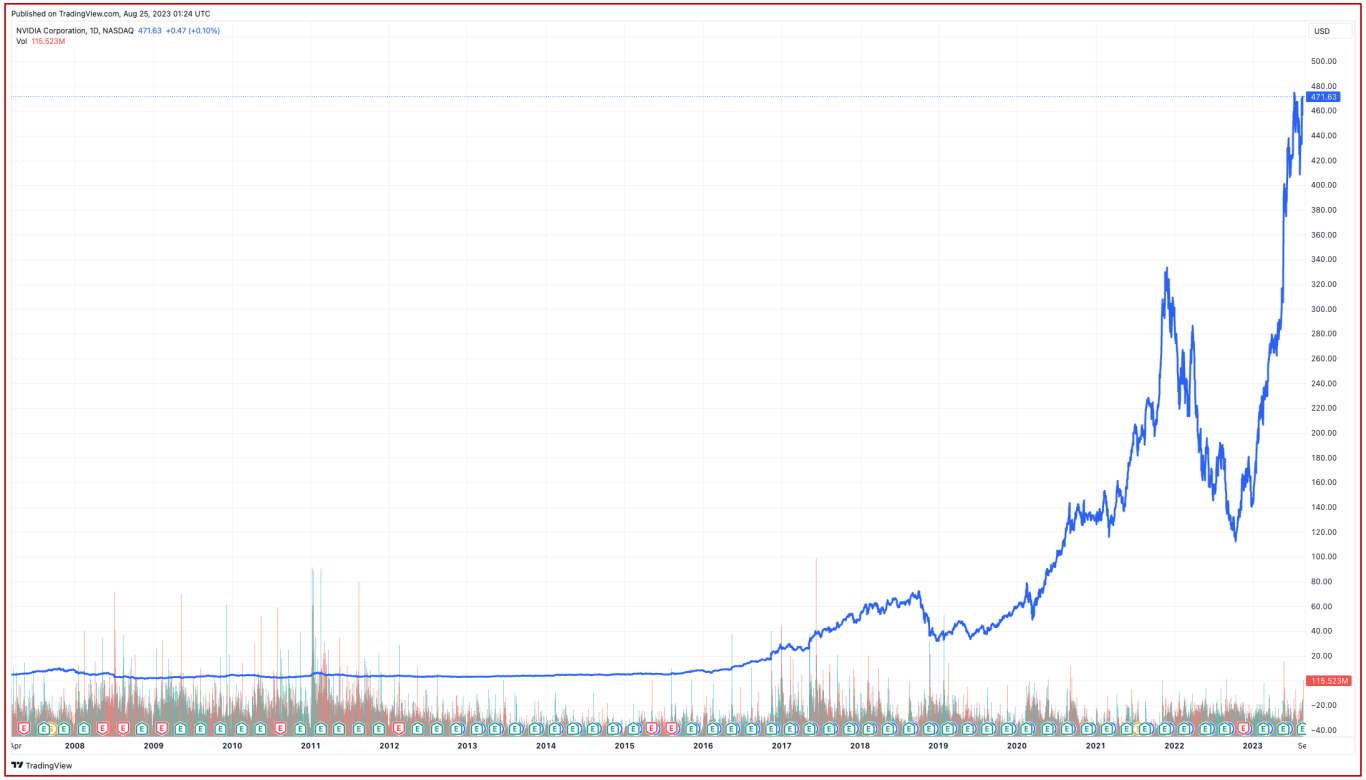
2. Computational Power

NVDA GPU



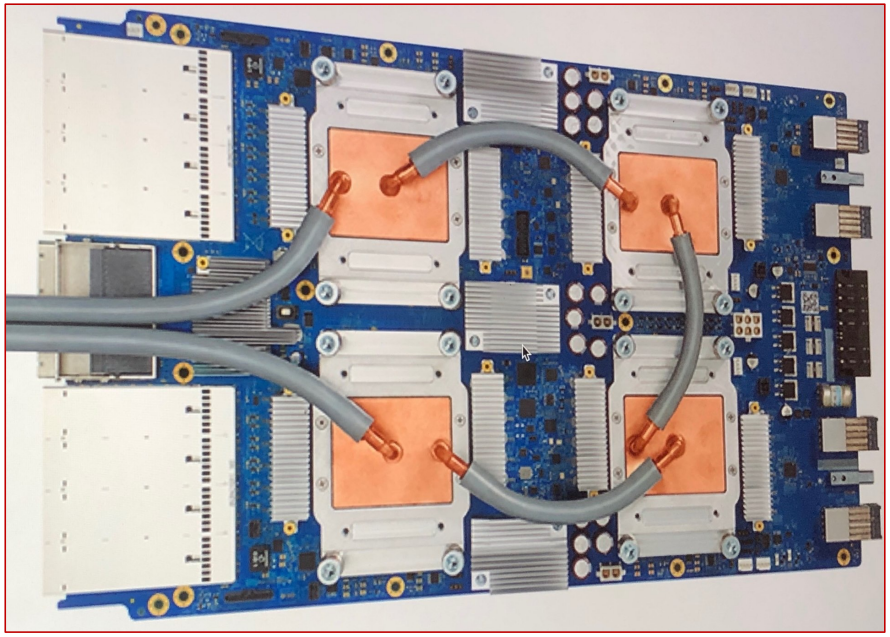
Source: Nvidia

Why is Deep Learning so Popular Nowadays?



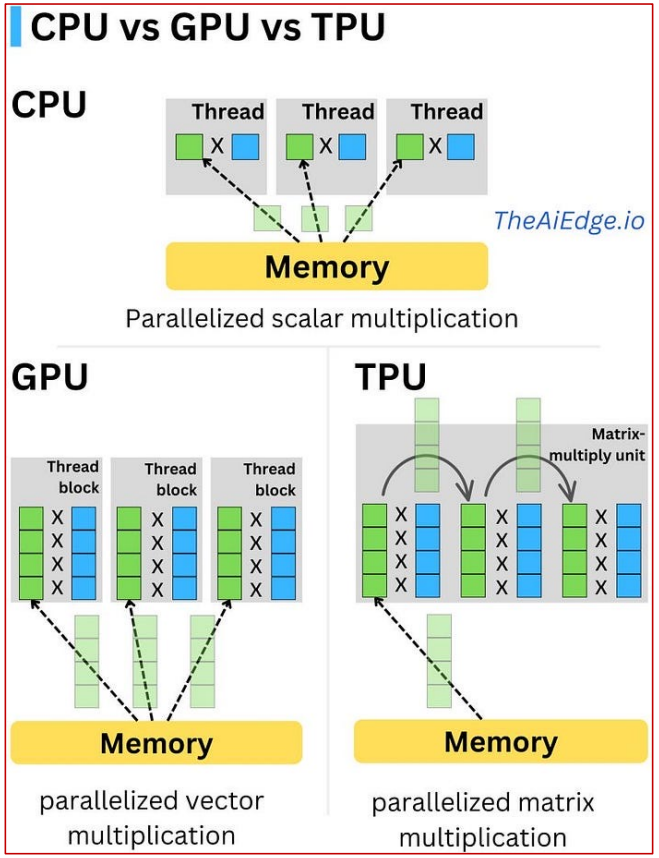
NVDA Stock Price

Why is Deep Learning so Popular Nowadays?



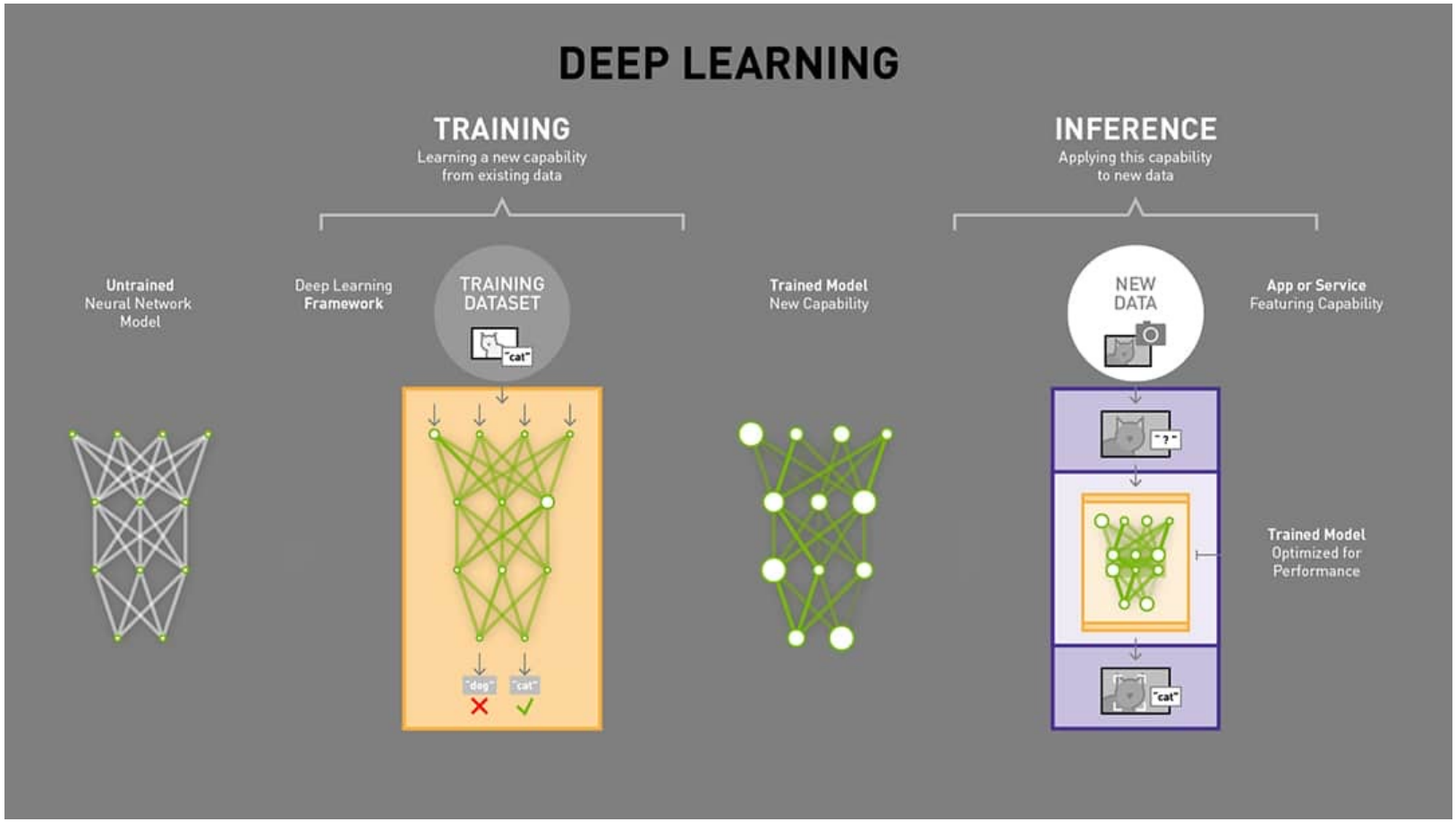
Google Tensor Processing Unit 3.0

Source: Wikipedia



Where Does the Computation Go?

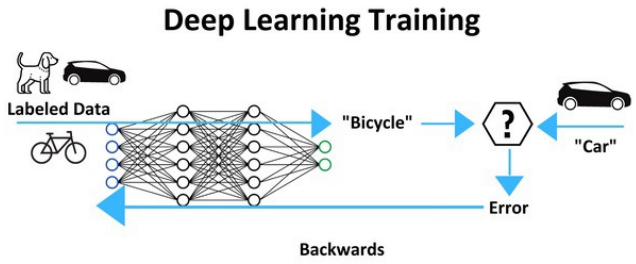
Computation and Its Cost



Source: blogs.nvidia.com

Computation and Its Cost

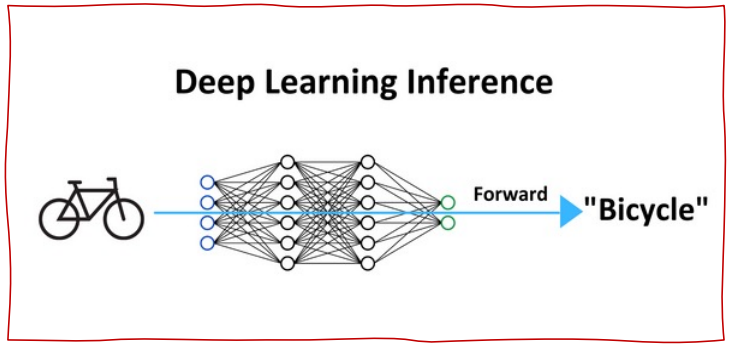
- **Training** requires:
  - **Multiple Passes**: The entire dataset is typically processed multiple times (an epoch).
    - Each epoch involves a forward pass and a backward pass (for error backpropagation).
  - **Parameter Updates**
  - **Storage of Intermediate Values**: During the backward pass, we need to store intermediate outputs for gradient computation. For deep networks, this can be memory-intensive.
- Computation Cost in **Training**:
  - **Time**: Can take from hours to weeks
  - **High Memory**
  - **Hardware**: Training benefits significantly from parallelization using GPUs or TPUs



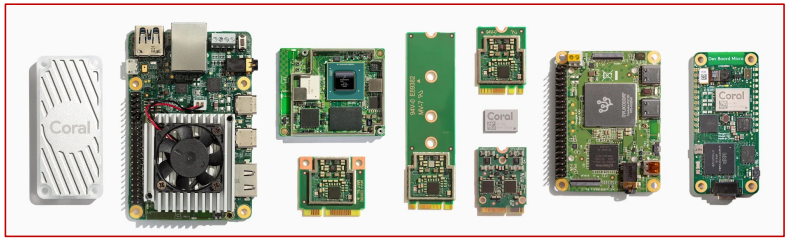
Source: premioinc.com

Computation and Its Cost

- **Inference** requires:
  - **Single** Pass: Data only needs a forward pass through the model.
  - **Fixed Parameters**: Since the model is already trained
  - **No Intermediate Storage**: We typically don't need to store intermediate layers' outputs
- Computation Cost in **Inference**:
  - **Time**: Generally, much faster than training.
  - **Memory**: Typically, lower than during training
  - **Hardware**: While GPUs can speed up inference, it's possible to run inference on smaller devices



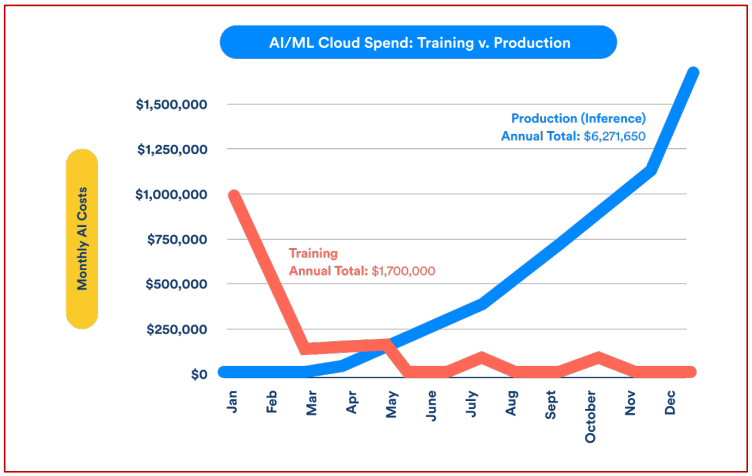
Source: premioinc.com



Source: coral.ai

Computation and Its Cost

- Inference for a single data point is fast
- But you might need to make predictions for millions of data points.

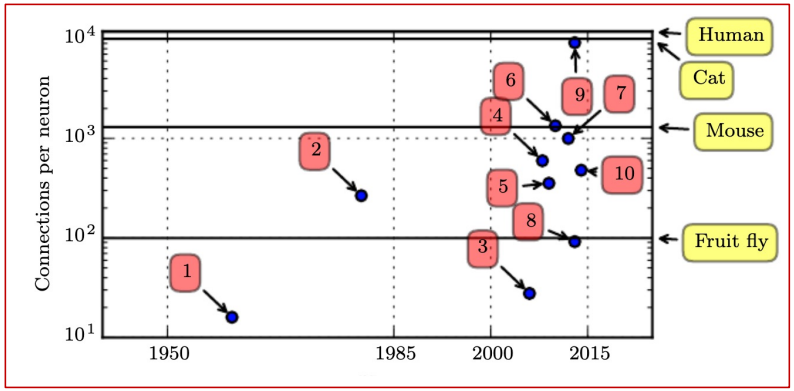


Cost of Training vs Production

Source: octoml.ai

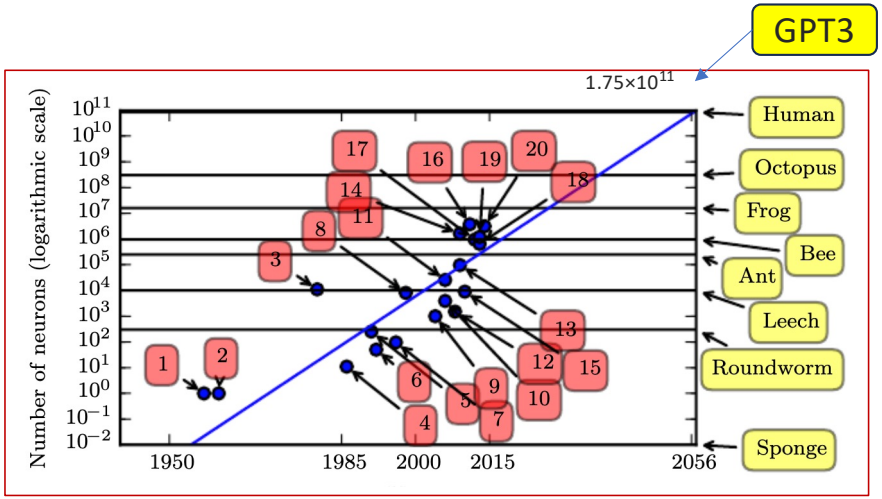
- Example:
  - A GPT-3 inference takes approximately 1 second on an A100 GPU
    - Cost per 1,000 tokens: $0.0002 and $0.0014
  - A user generating **100** inference requests a day can cost **$5** per year.
  - ChatGPT currently has about **100M** users.

Computing is Growing



Number of connections per neuron over time

Source: deeplearningbook.org

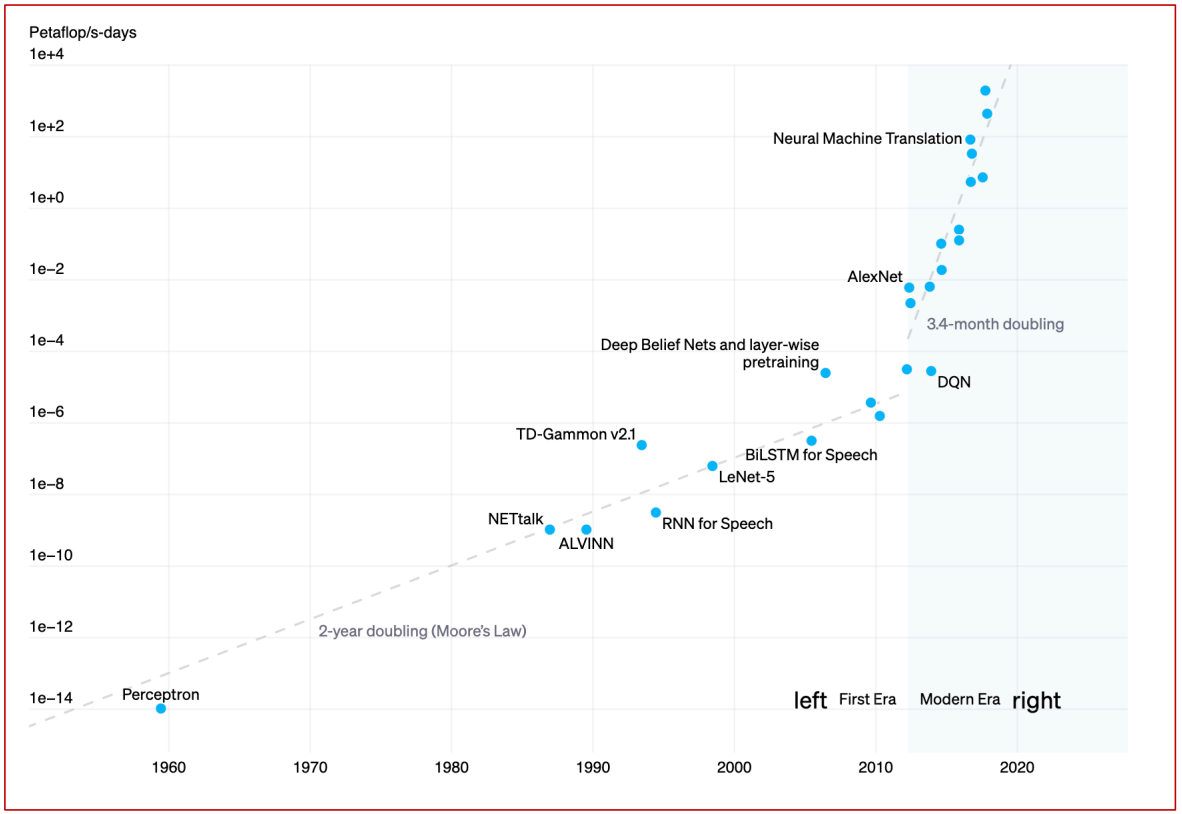


Number of neurons

Source: deeplearningbook.org

Rapid increase due to the availability of faster hardware

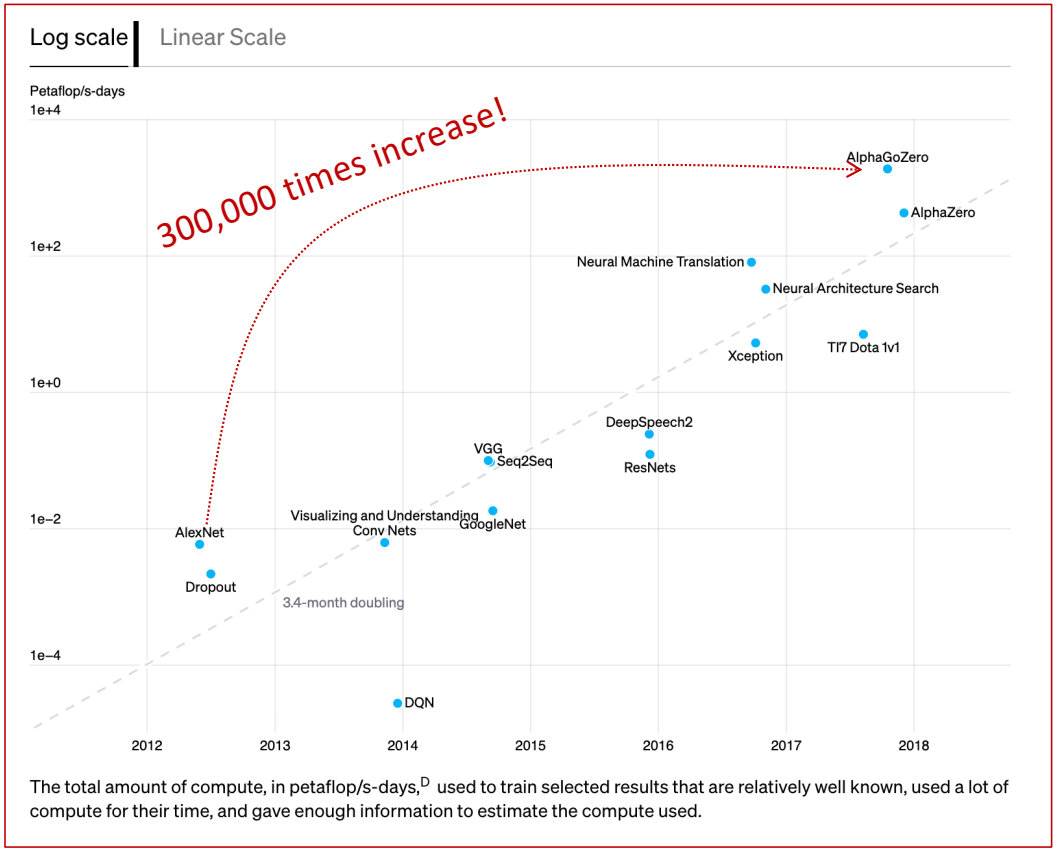
Computing is Growing



Two distinct eras of compute usage in training AI systems

Source: openai.com

Computing is Growing



The total amount of compute, in petaflop/s-days,[D] used to train selected results that are relatively well known, used a lot of compute for their time, and gave enough information to estimate the compute used.

Source: openai.com

Let’s take a look at the training process

What is Gradient?

- **Definition**: A vector representing the partial derivatives of a **scalar function** with respect to its **input variables**.
- **Use**: Measures the rate of change of a scalar function in each direction.
- **Input/Output**:
  - Input: A scalar-valued function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is a vector.
  - Output: A vector of size equal to the number of input variables.

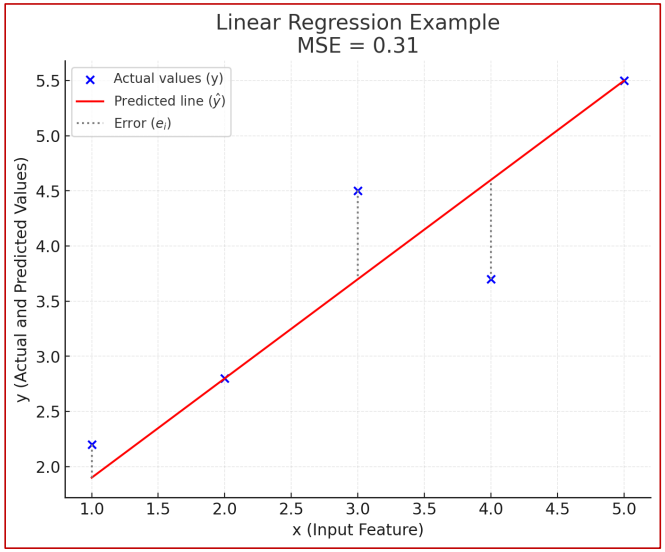
Example: For $f(x, y) = x^2 + y^2$,

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}.$$

Liner Regression

$\hat{y} = \theta_0 + \theta_1 \cdot x$:

- $\hat{y}$: Predicted value or output from the linear regression model.
- θ_0: Intercept of the line (bias term).
- θ_1: Slope of the line (weight or coefficient for x).
- x: Input feature or independent variable.



Blue points: Actual values (y) from the data.

Red line: Predicted values ($\hat{y}$) from the regression model ($\hat{y} = \theta_0 + \theta_1 \cdot x$).

Gray dotted lines: Errors ($e_i = \hat{y}_i - y_i$) between the actual and predicted values.

MSE: Mean Squared Error, representing the average of squared errors.

Liner Regression

$\hat{y} = \theta_0 + \theta_1 \cdot x$:

- $\hat{y}$: Predicted value or output from the linear regression model.
- θ_0: Intercept of the line (bias term).
- θ_1: Slope of the line (weight or coefficient for x).
- x: Input feature or independent variable.

Cost Function $J(\theta_0, \theta_1)$:

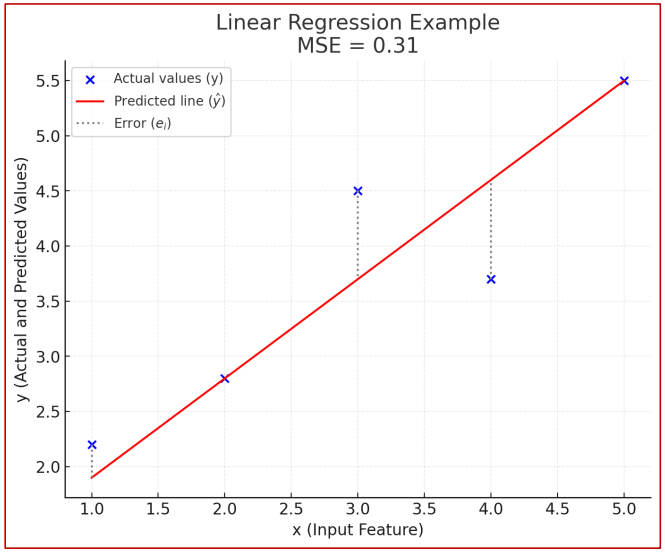
- Represents the **Mean Squared Error (MSE)**, a measure of the model's error.
- Formula:

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

- n: Number of data points (samples) in the dataset.
- $\hat{y}_i$: Predicted value for the i-th data point.
- y_i: Actual value (ground truth) for the i-th data point.

Error (e_i):

- $e_i = \hat{y}_i - y_i$: The difference between the predicted value and the actual value for the i-th sample.



Blue points: Actual values (y) from the data.

Red line: Predicted values ($\hat{y}$) from the regression model ($\hat{y} = \theta_0 + \theta_1 \cdot x$).

Gray dotted lines: Errors ($e_i = \hat{y}_i - y_i$) between the actual and predicted values.

MSE: Mean Squared Error, representing the average of squared errors.

Parameters
(Weights and biases in a NN)

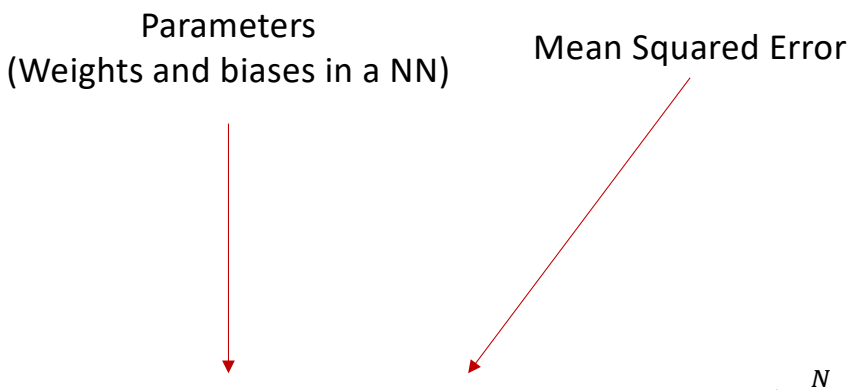
Mean Squared Error

$$J(\theta_1, \theta_2, \dots, \theta_m) = MSE(\theta_1, \theta_2, \dots, \theta_m) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2$$

$$e_i = y_i - \widehat{y_i}$$

$\widehat{y_i}$: Predicted value of i^{th} sample

y_i: Actual value of i^{th} sample



$$J(\theta_1, \theta_2, \dots, \theta_m) = MSE(\theta_1, \theta_2, \dots, \theta_m) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2$$

$$e_i = y_i - \widehat{y_i}$$

Squaring the errors has the effect of amplifying larger errors more than smaller ones.

- Sensitive to outliers.
- Sometimes this is desirable when we want to punish large errors
- Alternatives:
  - MAE (Mean absolute value).
  - Root Mean Squared Error (RMSE)
  - Log Loss (Cross-Entropy Loss)

Gradient Descent

$$\hat{y} = \theta_0 + \theta_1 \cdot x$$

Cost:

$$J(\theta_0, \theta_1) = \text{MSE}(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2$$

where

$$e_i = \hat{y}_i - y_i$$

Gradient:

$$\nabla J(\theta_0, \theta_1) = \left(\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}\right)$$

$$\nabla J(\theta_0, \theta_1) = \left(\sum_{i=1}^{n}(\frac{\partial J}{\partial e_i} \cdot \frac{\partial e_i}{\partial \theta_0}), \sum_{i=1}^{n}(\frac{\partial J}{\partial e_i} \cdot \frac{\partial e_i}{\partial \theta_1})\right)$$

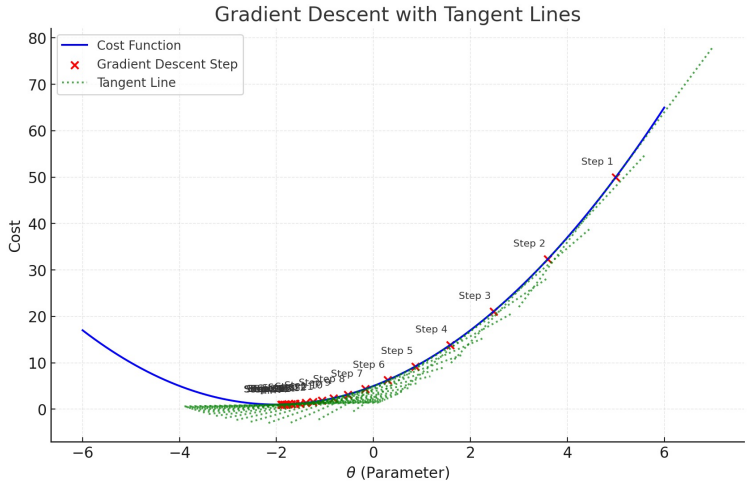
Replacing e_i:

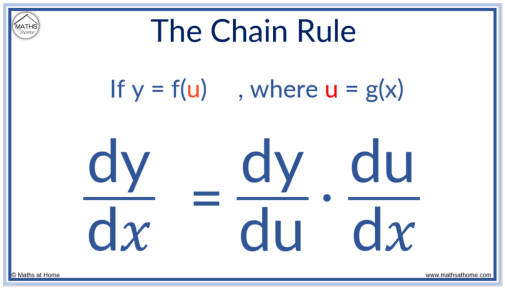
$$\nabla J(\theta_0, \theta_1) = \left(\frac{2}{n}\sum_{i=1}^{n}(e_i) \times 1, \frac{2}{n}\sum_{i=1}^{n}(e_i) \times x_i\right)$$

Final Gradient Vector:

$$\nabla J(\theta_0, \theta_1) = \left(\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i), \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \times x_i\right)$$

Gradient Descent for a linear regression with only two parameters (HW1)





The chain rule

Gradient Descent

$$\hat{y} = \theta_0 + \theta_1 \cdot x$$

Cost:

$$J(\theta_0, \theta_1) = \text{MSE}(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2$$

where

$$e_i = \hat{y}_i - y_i$$

Gradient:

$$\nabla J(\theta_0, \theta_1) = \left(\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}\right)$$

$$\nabla J(\theta_0, \theta_1) = \left(\sum_{i=1}^{n}(\frac{\partial J}{\partial e_i} \cdot \frac{\partial e_i}{\partial \theta_0}), \sum_{i=1}^{n}(\frac{\partial J}{\partial e_i} \cdot \frac{\partial e_i}{\partial \theta_1})\right)$$

Replacing e_i:

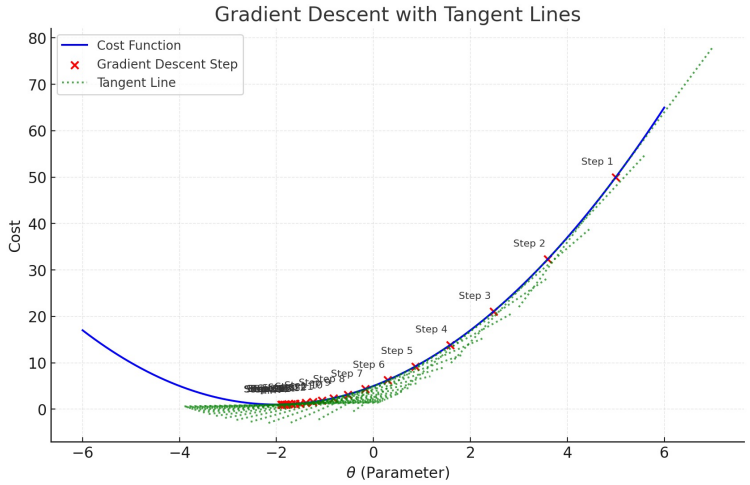
$$\nabla J(\theta_0, \theta_1) = \left(\frac{2}{n}\sum_{i=1}^{n}(e_i) \times 1, \frac{2}{n}\sum_{i=1}^{n}(e_i) \times x_i\right)$$

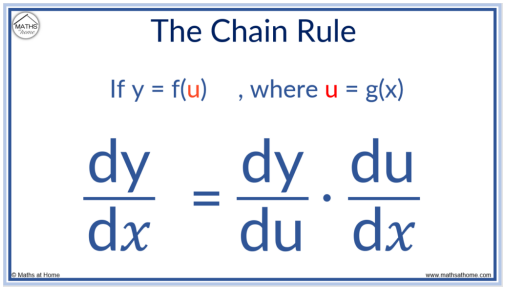
Final Gradient Vector:

$$\nabla J(\theta_0, \theta_1) = \left(\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i), \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \times x_i\right)$$

$$\theta_0 := \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta_0}, \quad \theta_1 := \theta_1 - \alpha \cdot \frac{\partial J}{\partial \theta_1} \quad \text{where } \alpha \text{ is the learning rate.}$$

Gradient Descent for a linear regression with only two parameters (HW1)

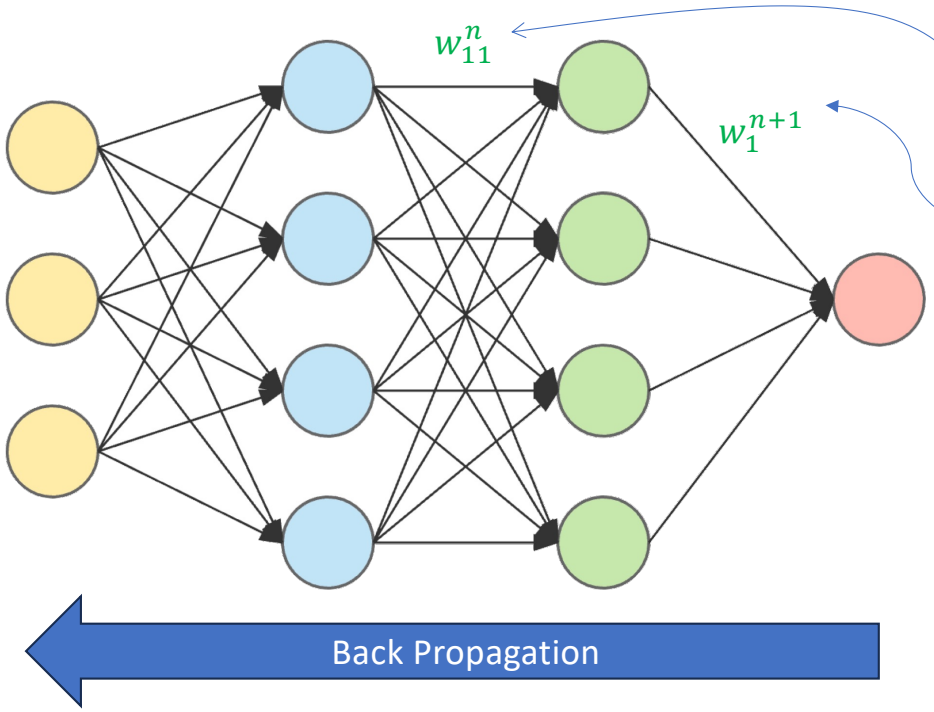


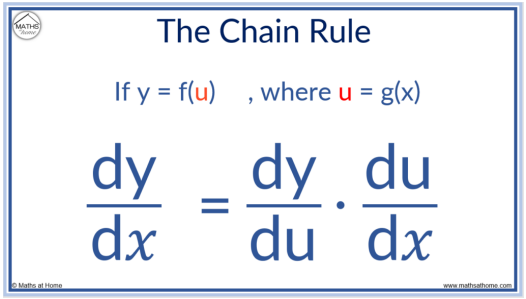


The chain rule

How is GD Run on DNNs?

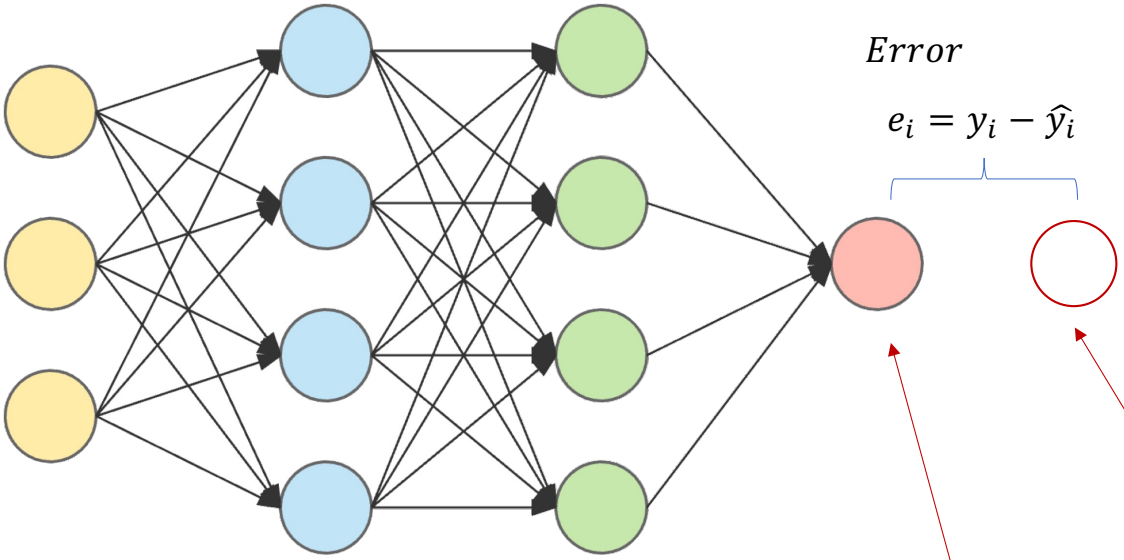
- Backward propagation of errors:
  - A method of calculating the gradient of the loss function concerning each weight in the network by applying the chain rule of calculus.





Forward Path

N Samples



$Error$

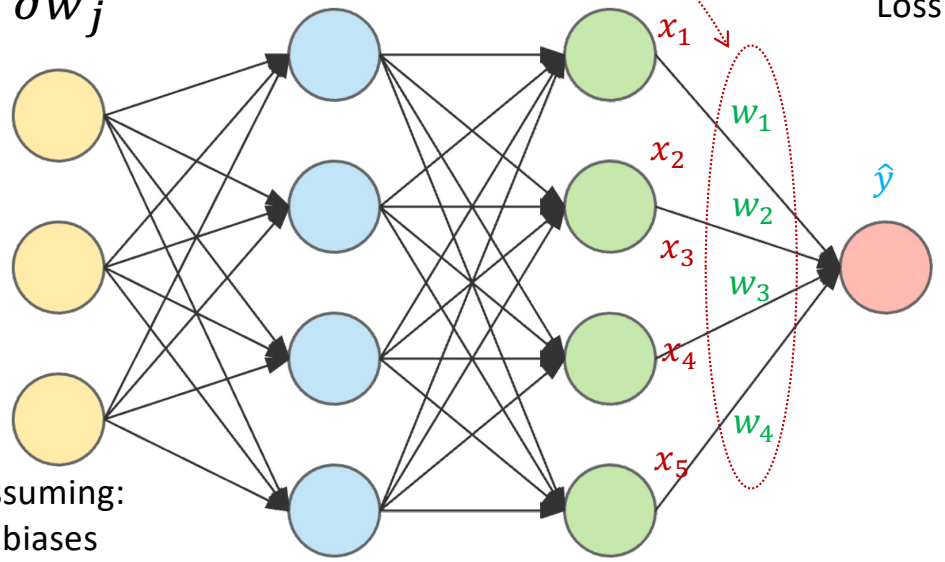
$e_i = y_i - \hat{y}_i$

Back Propagation

y_i: Actual value of ith sample

$\hat{y}_i$: Predicted value of ith sample

Goal: Find $\frac{\partial L}{\partial w_j}$



For simplicity, assuming:

- There are no biases
- There are no activation functions

Number of samples

Loss Function $L = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

$\hat{y} = \sum_{1}^{5} x_j\, w_j$

$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_j}$

x_i

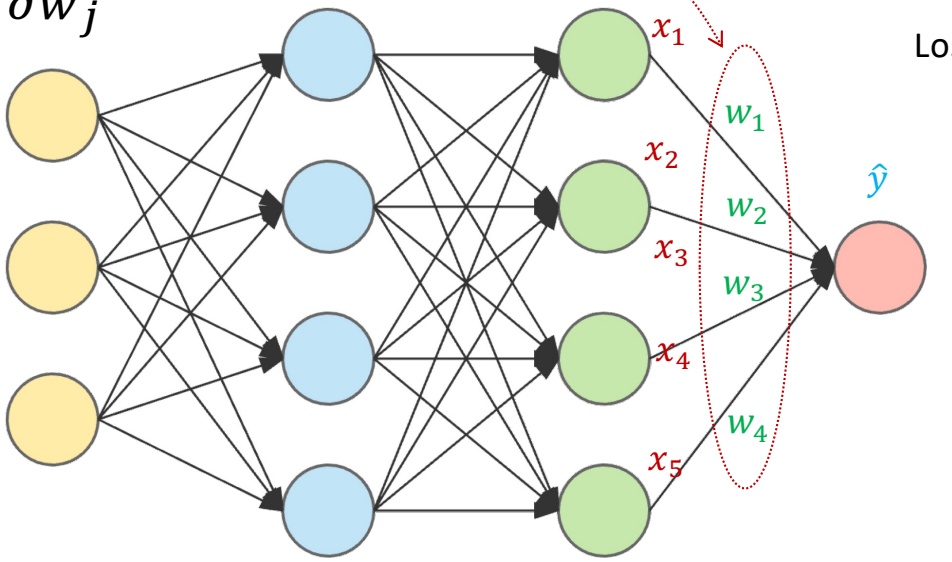
$\frac{\partial \hat{y}}{\partial w_j} = x_j$

$\frac{\partial L}{\partial \hat{y}} = \frac{\partial \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\partial \hat{y}} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)$

$\frac{\partial L}{\partial w_j} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i) \times x_j$

Goal: Find $\frac{\partial L}{\partial w_j}$

N Samples



Loss Function $L = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

Number of samples

$$\frac{\partial L}{\partial w_j} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i) \times x_j$$

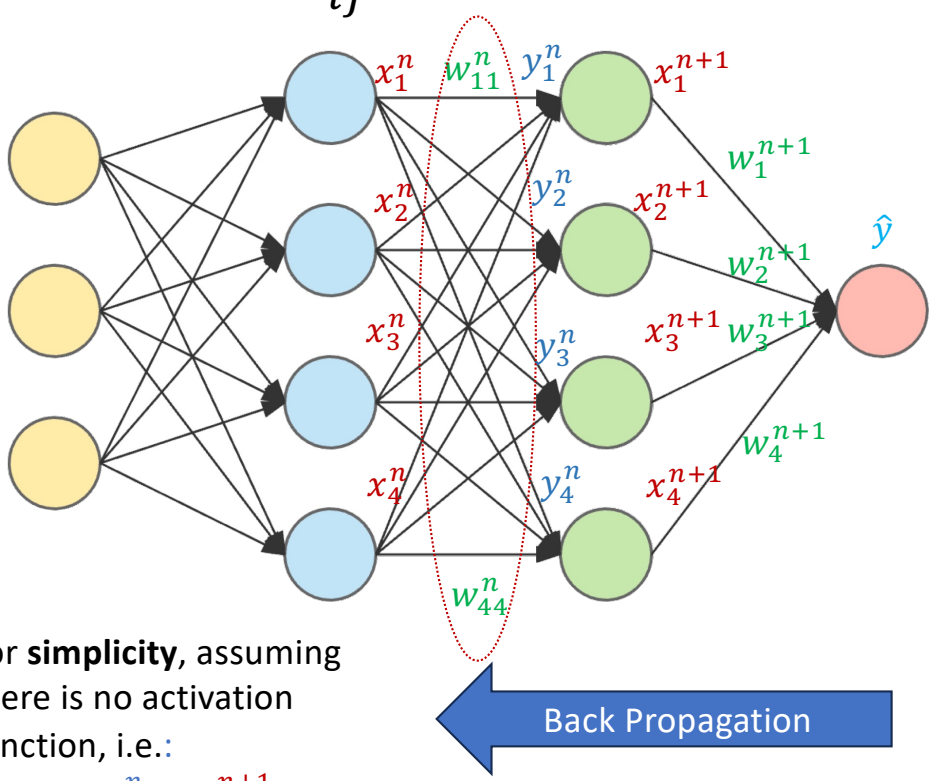
Having the gradient, now we can update each w_j
We do this in multiple steps (AKA epochs)

Gradient

$$w_j{}^{t+1} = w_j{}^{t} - \alpha \frac{\partial L}{\partial w_j}$$

Learning Rate

Goal: Find $\frac{\partial L}{\partial w_{ij}^n}$



Back Propagation

For **simplicity**, assuming there is no activation function, i.e.:

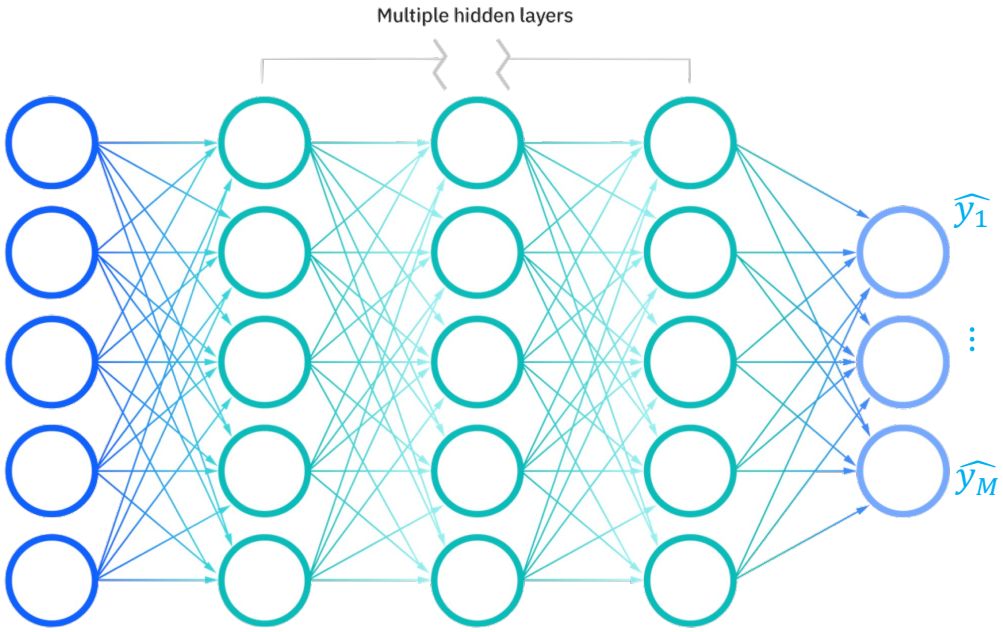
$$y_i^n = x_i^{n+1}$$

Loss Function $L = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$ (Number of samples)

$$\frac{\partial L}{\partial w_{ij}^n} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial y_j^n} \times \frac{\partial y_j^n}{\partial w_{ij}^n}$$

$$\frac{\partial L}{\partial w_{ij}^n} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i) \times w_j^{n+1} \times x_j^n$$

Multiple Outputs

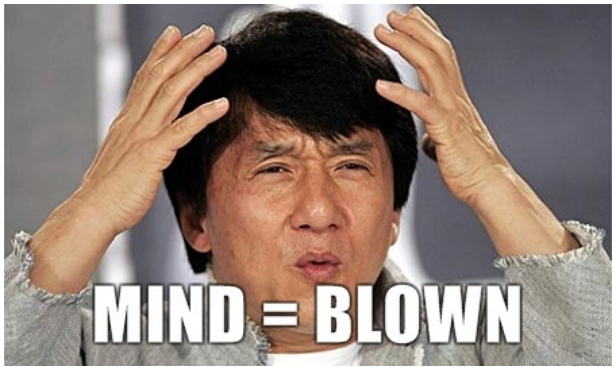


Number of outputs

Number of samples

$$L = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N}(\widehat{y_{ij}} - y_{ij})^2\right)$$

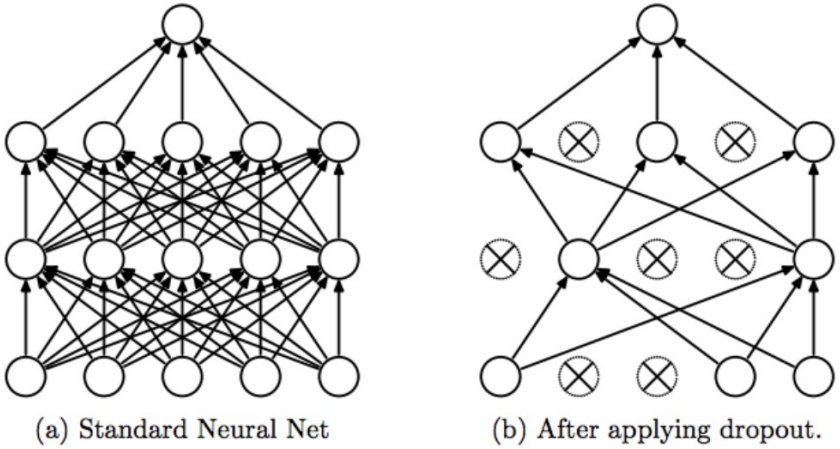
Loss Function Is the average of MSE on all outputs



We simplified many things. In reality, there are multiple outputs and activation functions, which makes backpropagation even more complicated.

Why is Deep Learning so Popular Nowadays?

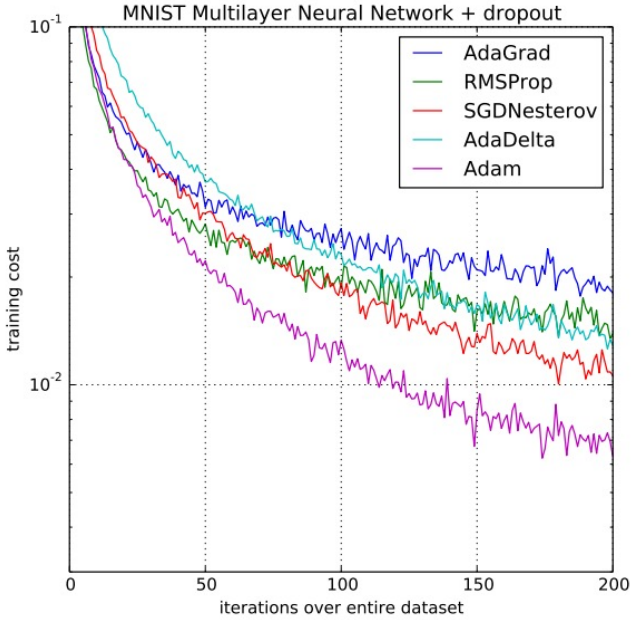
3. Advanced Algorithms



(a) Standard Neural Net (b) After applying dropout.

Google Tensor Processing Unit 3.0

Source: Wikipedia

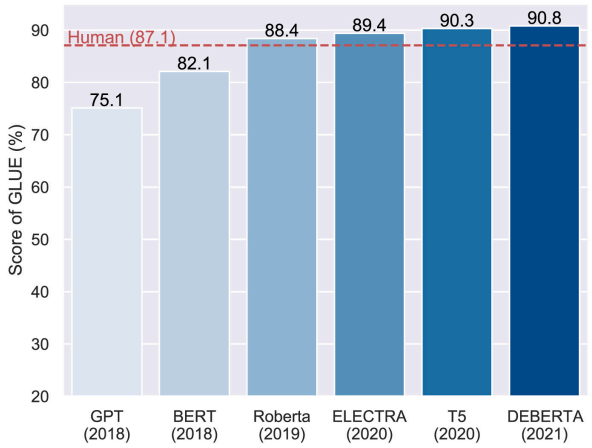


Comparison of optimizers used for the optimization training of a multilayer neural network on MNIST images.

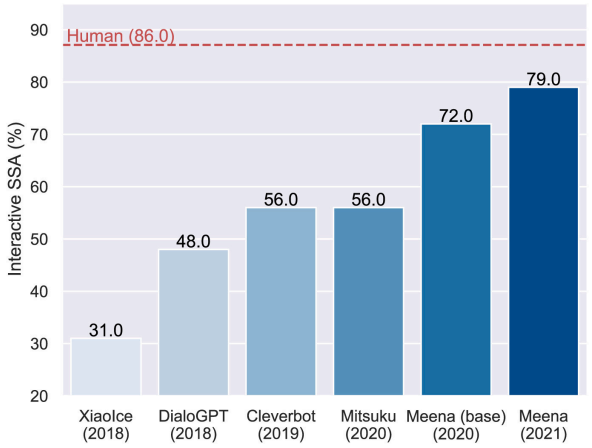
Source: Adam: A Method for Stochastic Optimization, 2015

Why is Deep Learning so Popular Nowadays?

4. Pre-trained Models



(a) Evaluation on language understanding benchmark GLUE.



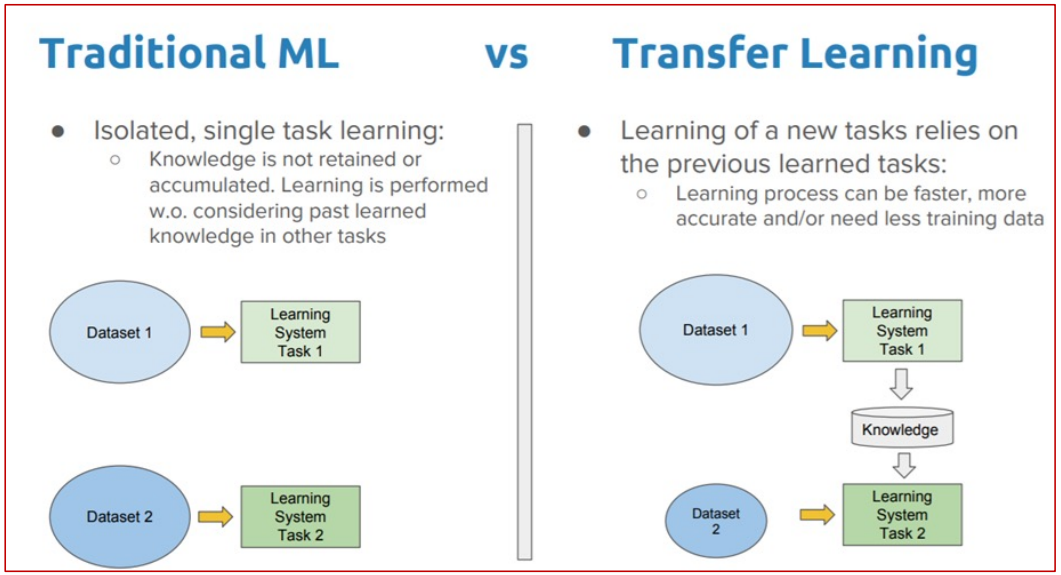
(b) Manual evaluation on dialogue systems.

Improvement on performance of both language understanding and language generation after using large-scale PTMs

Source: Pre-trained models: Past, present and future, 2021

Why is Deep Learning so Popular Nowadays?

5. Transfer Learning



Transfer Learning

Source: Dipanjar Sarkar

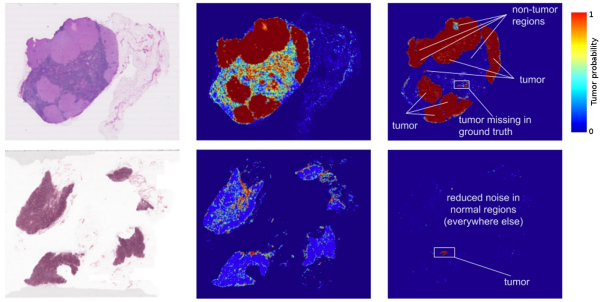
Why is Deep Learning so Popular Nowadays?

6. Open Source Software

TensorFlow

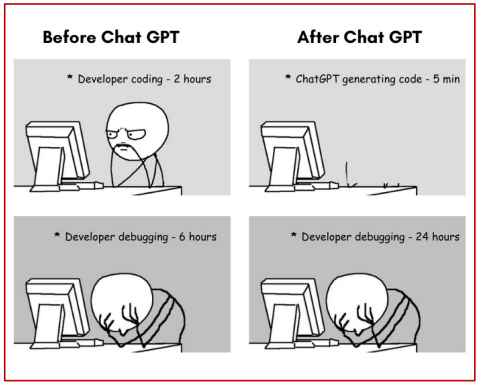
Why is Deep Learning so Popular Nowadays?

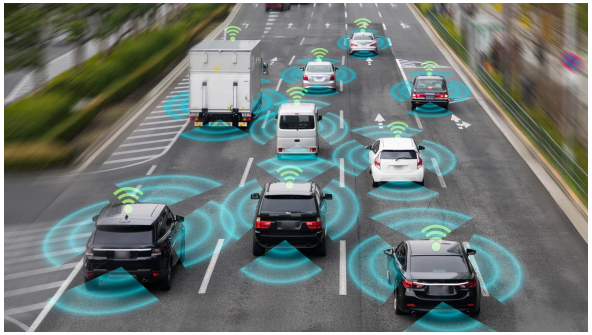
7. Expansion Into New Areas



Cancer Detection

Source: Google Blog

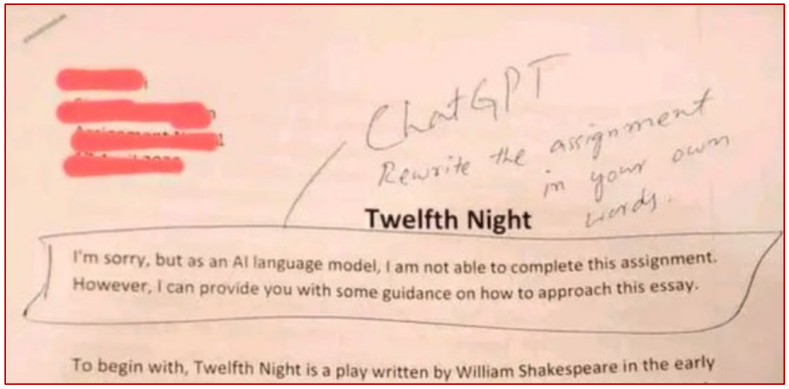




Autonomous Driving

Source: topgear.com

Why is Deep Learning so Popular Nowadays?



COMPANY NEWS > TECH SECTOR NEWS

ChatGPT Passes CPA Exam on Second Attempt. Is It Coming For Accounting Jobs?

The latest version of the chatbot passed the CPA exam with a score of 85.1, just weeks after an earlier version had failed

Tech

ChatGPT Can Pass the Bar Exam. Does That Actually Matter?

The question is what happens next.

GPT-3 Overview:

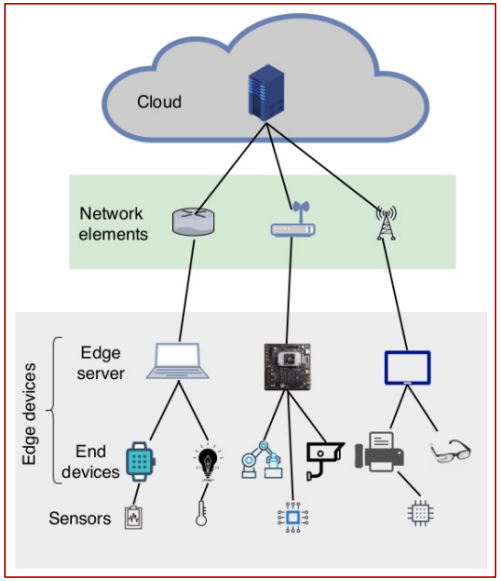
- **Layers**: 96 layers in its largest variant.
- **Parameters**: 175 billion parameters.

The supercomputer developed for OpenAI is a single system with more than **285,000** CPU cores, **10,000** GPUs and **400** gigabits per second of network connectivity for each GPU server

https://news.microsoft.com/source/features/innovation/openai-azure-supercomputer/

Edge AI

- **Latency:** Edge AI reduces latency by processing data on-device, crucial for real-time applications.
- **Bandwidth & Costs:** On-device processing saves significant bandwidth and associated costs.
- **Privacy & Security:** Edge AI offers better privacy as data stays on-device and isn't transmitted to the cloud.
- **Operational Continuity:** Edge devices can operate independently of internet connectivity.



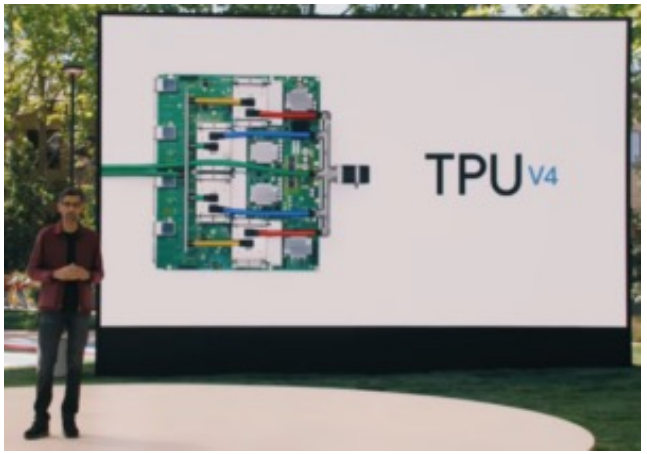
Source: M.G. SARWAR MURSHED et al.

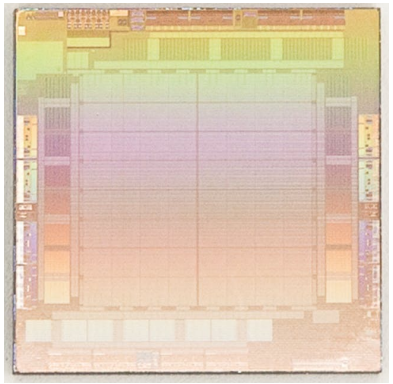
Google Edge AI

	Edge (Devices/nodes, Gateways, Servers)	Google Cloud
Tasks	ML inference	ML training and inference
Software, services	Linux, Windows	AI Platform, Kubernetes Engine, Compute Engine, Cloud IoT Core
ML frameworks	TensorFlow Lite, NN API	TensorFlow, scikit-learn, XGBoost, Keras
Hardware accelerators	Edge TPU, GPU, CPU	Cloud TPU, GPU, and CPU

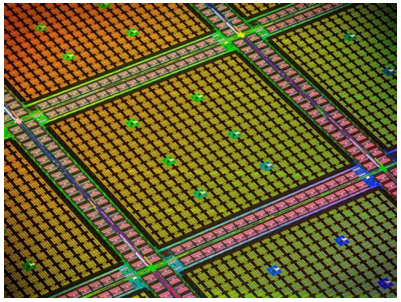
Source: coral.ai

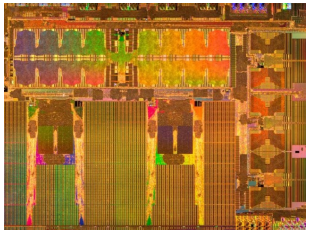
Everyone is Building Chips These Days





Meta Training and Inference Accelerator (**MTIA**)





Tesla Dojo and FSD Chips

AI Accelerators: A Hot Market

- **AI & ML Growth**: Rising demand due to expansive AI applications.
- **Edge Computing**: Need for on-device AI processing.
- **Cloud Providers**: Improve data center efficiency and performance.
- **Startup Boom**: Venture-backed companies innovating in AI hardware.
- **Industry Demand**: Automotive, healthcare, finance, and more.
- **Academic Push**: Research demands powerful AI hardware.