The Foundations of Intelligence: The Mathematics Behind AI

ICMMAICS - 17 August 2026

sambath.narayanan@dataeverconsulting.com

Chief Patron

Thiru T. Narayanan,B.E., President &Secretary

Patron

Dr. M.R. Chandran, Principal(i/c)

Co-patron

Dr. S.V. Ganesan, Vice Principal

Convenor

Dr. K. Muthukumaran, Head & Associate Professor

Organizing Secretaries

Dr. K. Angaleeswari, Assistant Professor

Dr. A. Meena, Assistant Professor

Dr. A. Mohamed Ali, Assistant Professor

Executive Committee Members

Dr. P. Veerammal, Associate Professor

Dr. A. Wilson Basker, Associate Professor

Dr. S. V. Padmavathi, Associate Professor

Dr. M. Kalanithi, Associate Professor

Dr. N. Deena, Assistant Professor

Technical Sessions

Session I

Topic:

Machine Learning for Next-Generation Stability and Control Systems: from Lyapunov Certificates to Data-Driven Dynamics.

By

Dr. Nallappan Gunasekaran,
Associate Professor,
Department of Natural Sciences,
Eastern Michigan Joint College of Engineering,
Beibu Gulf University, Guangxi, China.

Session II

Topic:

Analysis of Switching the Purchase Option in a Two-Commodity Perishable Inventory System.

by

Dr. N. Anbazhagan,
Senior Professor and Head,
Department of Mathematics,
Alagappa University Karaikudi ,
Tamil Nadu, India.

Session III

Topic:

Foundations of Intelligence: The Mathematics Behind AI.

by

Dr.Sambath Parthasaraty,
Founder &Directorof Dataever Consulting,
Chennai,Tamil Nadu, India.

Session IV

Topic:

AI and Machine Learning: A Mathematical Approach.

by

Dr. C. Deisy, Professor & Head,
Department of Information Technology,
Thiagarajar College of Engineering,
Madurai,Tamil Nadu, India.

DUTY DEVOTION DISTINCTION

SARASWATHI NARAYANAN COLLEGE

(Autonomous Institution Affiliated to Madurai Kamaraj University)

(Reaccredited by NAAC in the 3rd Cycle)

Madurai 625 022,Tamil Nadu, India

PG & Research Department of

MATHEMATICS

Organizes

A One- Day

International Conference on

Mathematical Modelling, Artificial Intelligence and Computational Sciences

ICMMAICS - '26

17th August 2026

Multipurpose Hall

Saraswathi Narayanan College

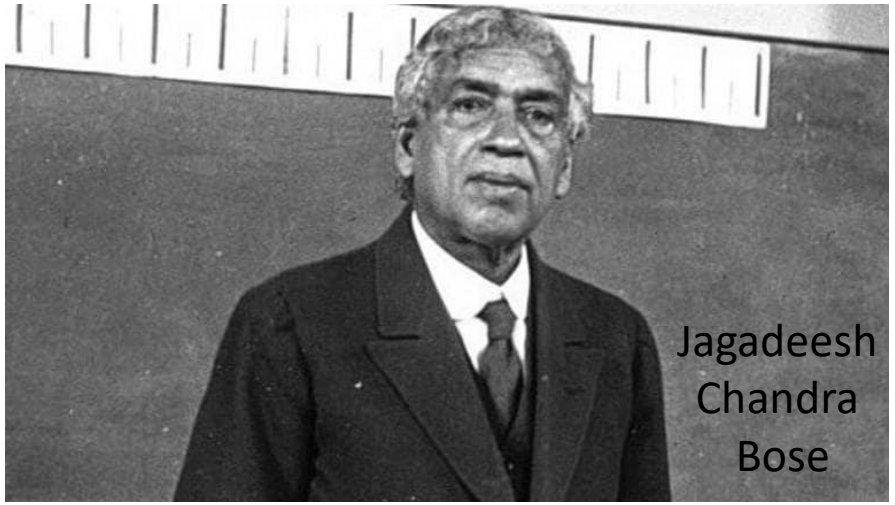
Madurai

Objective of this session

1. Explain the latest and the core concepts of AI
2. When doing that simultaneously highlight the important role played by powerful mathematics techniques

First Method of Research

Experimental

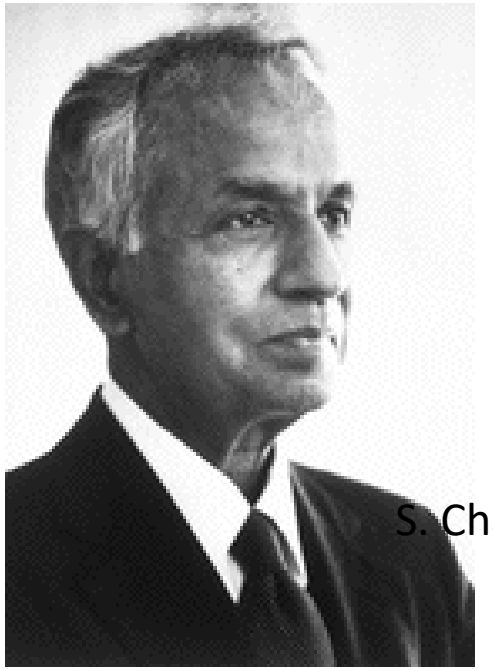


Path Breaking Discoveries in the Laboratory

Even plants have life

Second Method of Research

Mathematical



S. Chandrasekhar

Research with Pen and Paper

$M_{max} \approx 1.4 M_{\odot}$

Third Method of Research

Modeling/Numerical Simulation

Massively Parallel Supercomputers

Smoot and Mather , 2006 Nobel in Physics

COBE experiment

The Fourth Method of Research

Alan Turing

Geoffery Hinton

The Data Science & AI

Mathematics is common amongst all forms of Research


QUANTUM AI: UNIFIED ROBOTICS & COMPUTER VISION
Active CV System
Active CV System
Active CV System
UNIT-4
UNIT-4

AI's Biggest Data challenge

- Data is not easily available
- Data is raw and voluminous
- Data requires management

Can Statistics help in Exploratory Data Analysis(EDA) ? In Feature Engineering ?

AI's LLM Disruption

- **Large Language Models**

	ChatGPT	Claude	Gemini	Grok
Everyday answers	⭐	✅	✅	✅
Writing	✅	⭐	✅	✅
Coding	✅	⭐	✅	✅
Reasoning	⭐	✅	✅	✅
Deep research	⭐	✅	✅	✅
Web search	✅	✅	✅	✅
Voice chat	⭐	✅	✅	✅
Image gen	⭐	❌	✅	✅
Video gen	✅	❌	⭐	❌

The training process of AI models involves optimization, where calculus is used to minimize the loss function. Algorithms like gradient descent use derivatives to determine the direction and optimal step in updating model weights.

AI's Biggest Challenge

- **Due to Chat GPT, Gemini kind of LLM application computer time has become very very expensive**

- **It has even created the problem of cooling water shortage**

Can clever mathematics computation reduce the workload ?

Supervised Learning

Classification:

Dog

Cat

AI

(Dog or Cat)

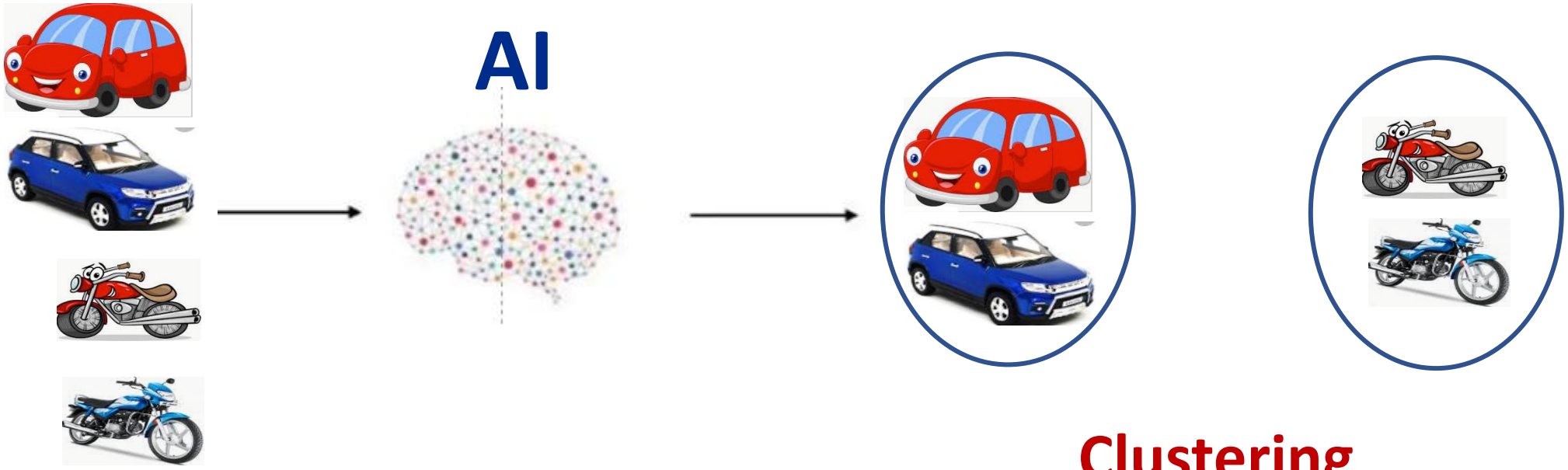
Regression:

Temperature

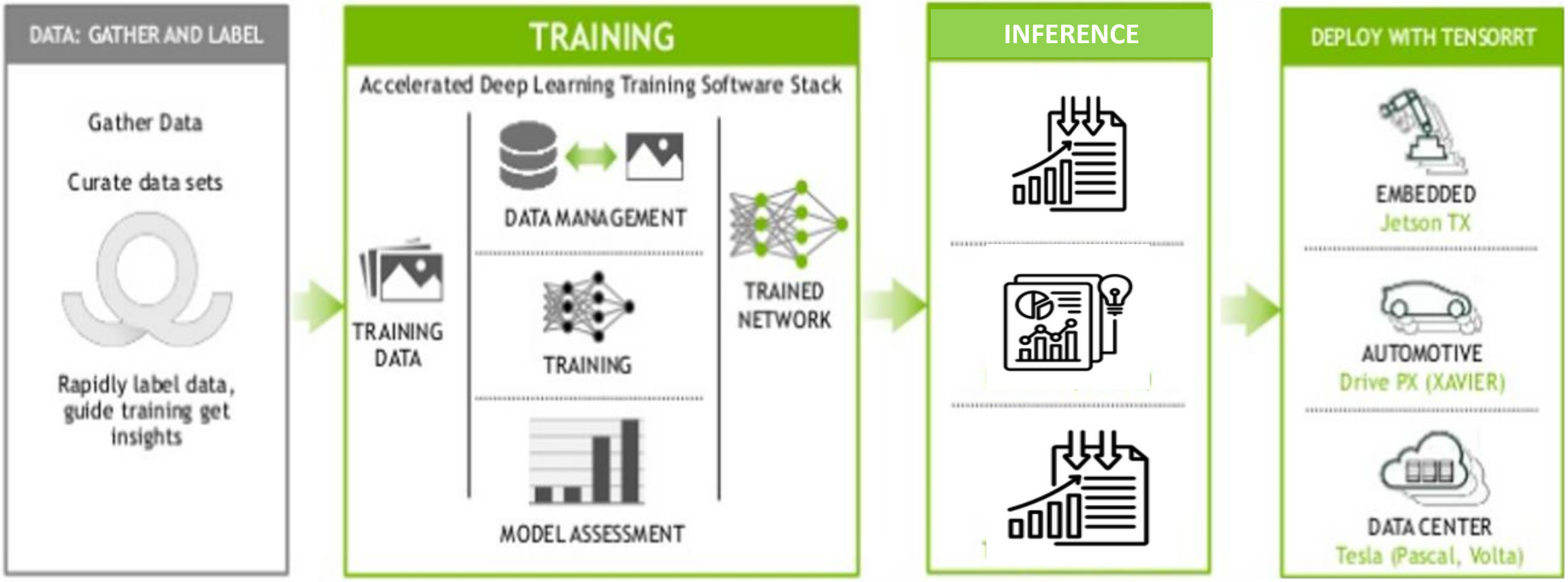
Rainfall in cm

Rainfall in cm

Unsupervised Learning



Important AI Steps



Mathematical Foundations of AI

Core mathematical disciplines driving AI

Mathematical Foundation	What It Enables in AI	Example
Linear Algebra	Represent & transform data	Vectors, Matrices, Tensors
Probability & Statistics	Handle uncertainty & learn from data	Prediction, Bayesian Models
Calculus	Optimize & improve models	Gradient Descent
Optimization	Find the best model parameters	Loss Minimization
Information Theory	Measure information & uncertainty	Entropy, Cross-Entropy

Matrix Algorithms

- PCA (Principal Component Analysis)
- SVD, (Singular Value Decomposition)
- matrix factorization for recommender systems[2]
- dimensionality reduction techniques

Calculus

- Calculus, particularly **derivatives** and integrals, is at the core of deep learning. The training process of AI models involves **optimization**, where calculus is used to minimize the loss function. Algorithms like **gradient descent** use derivatives to determine the direction and optimal step in updating model weights. For instance, artificial **neural networks use calculus to calculate gradients during backpropagation**, allowing the model to iteratively learn from data.

Probability and statistics

- Probability and statistics are used to handle uncertainty in data and AI models. Probability helps in **predicting the most likely outcomes, while statistics is used to make inferences from existing data.**

- In natural language processing (NLP), probabilistic models like Naive Bayes are used for tasks such as text classification. Statistics is also used to evaluate AI model performance and ensure that the results are statistically significant.

Raw data

Documents

Images

Numbers

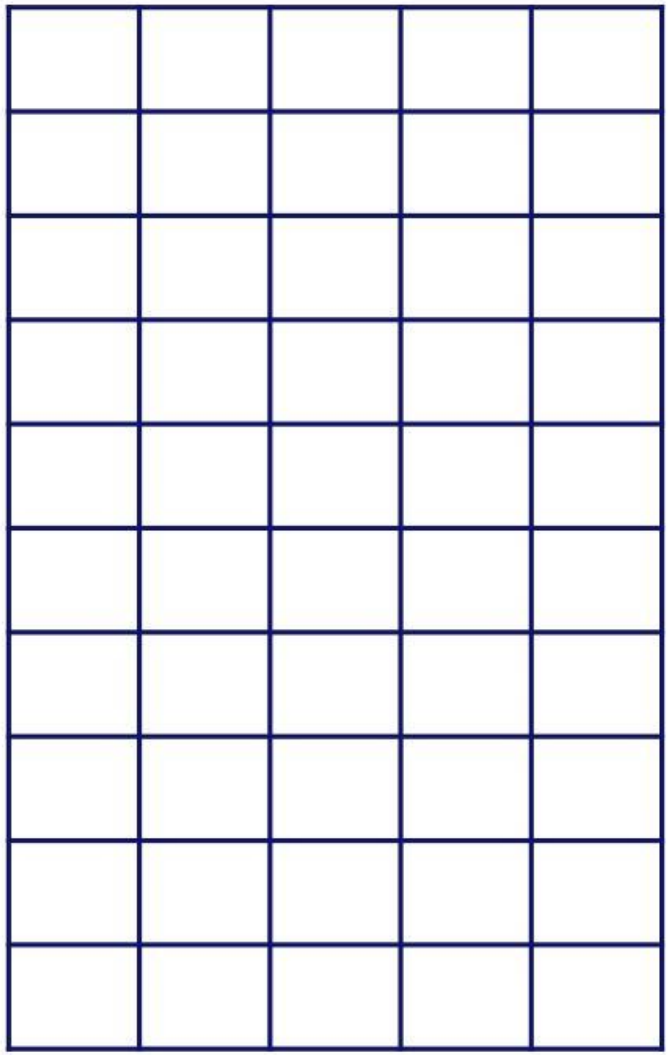
Sounds

●●●

Feature matrix (X)

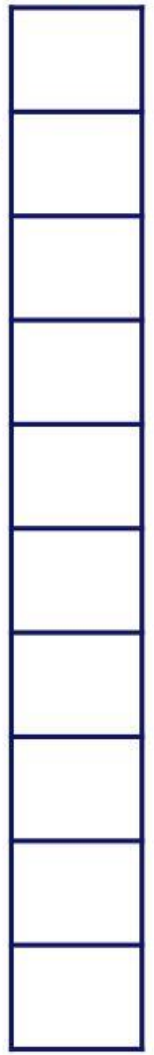
n_features ⟶

n_samples ⟵



Target (y)

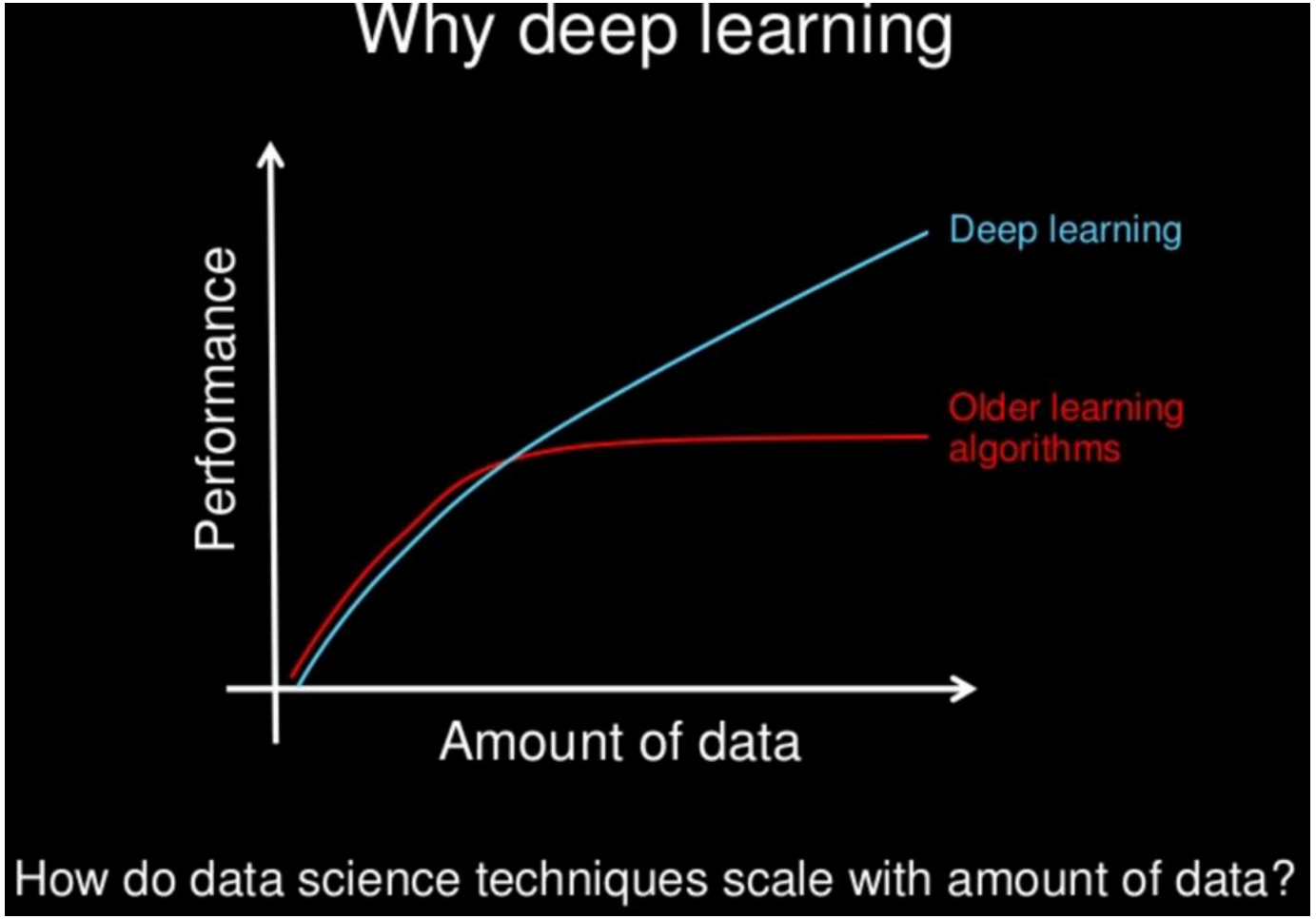
n_samples ⟵



Sample Bank Loan Processing — Structured Data

Loan_ID	Age	Income (₹)	Employment (yrs)	Credit Score	Loan Amount (₹)	Loan Term (yrs)	Existing Loans	DTI (%)	Previous Default	Loan Approved
L001	28	6,00,000	3	685	8,00,000	5	1	32	No	Yes
L002	45	12,00,000	18	780	15,00,000	7	0	21	No	Yes
L003	31	4,50,000	5	620	10,00,000	5	2	48	Yes	No
L004	52	18,00,000	25	810	20,00,000	10	1	18	No	Yes
L005	36	7,50,000	8	650	12,00,000	7	3	42	No	No
L006	29	9,00,000	4	720	10,00,000	5	0	25	No	Yes

More data higher performance



Data sources

- IoT configurations, where sensors or devices collect raw data
- Open source Data repositories
- ImageNet
- CIFAR
- MNIST
- COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features

Source: Prof. Andrew Ng

The Grammar of Math: Anatomy of a Linear Equation

W (The Learned Perspective)
The weight matrix. The model's internal memory that dictates how strongly **x** influences the output.

x (The Raw Material)
The input variable. The current state of the token's meaning.

$$y = W \cdot x + b$$

y (The Transformed Meaning)
The output. The new feature generated by the layer.

b (The Baseline Shift)
The bias. A subtle adjustment moving the meaning up or down.

This humble formula is the engine of learning. Variables (`x`) hold the data; parameters (`W, b`) hold the memory.

Module 1.1: High-Dimensional Linear Algebra & Vector Spaces

- **Core Concept:** Data as geometric points in high-dimensional vector spaces $\mathbb{R}^d$.
- **Mathematical Depth :**
  - **Linear Transformations & Matrix Operators:** How neural network layers represent affine transformations $f(x) = Wx + b$.
  - **Spectral Decomposition & SVD:** Low-rank approximations, Principal Component Analysis (PCA), and feature compression using Singular Value Decomposition:
  - $A = U\Sigma V^T$
  - **Manifold Hypothesis:** High-dimensional real-world data concentrates near lower-dimensional sub-manifolds embedded within $\mathbb{R}^d$.

Singular Value Decomposition(SVD) in AI

$$A = U \Sigma V^T$$

Decomposes matrix A (m × n) into orthogonal and diagonal matrices

Matrix U

Left singular vectors representing row feature relationships.

Matrix Σ

Diagonal matrix of singular values indicating feature importance weights.

Matrix V^T

Right singular vectors capturing column attribute patterns.

Key AI/ML Applications

Dimensionality Reduction

Powers Principal Component Analysis (PCA) by keeping top-k singular values to remove noise.

Recommender Systems

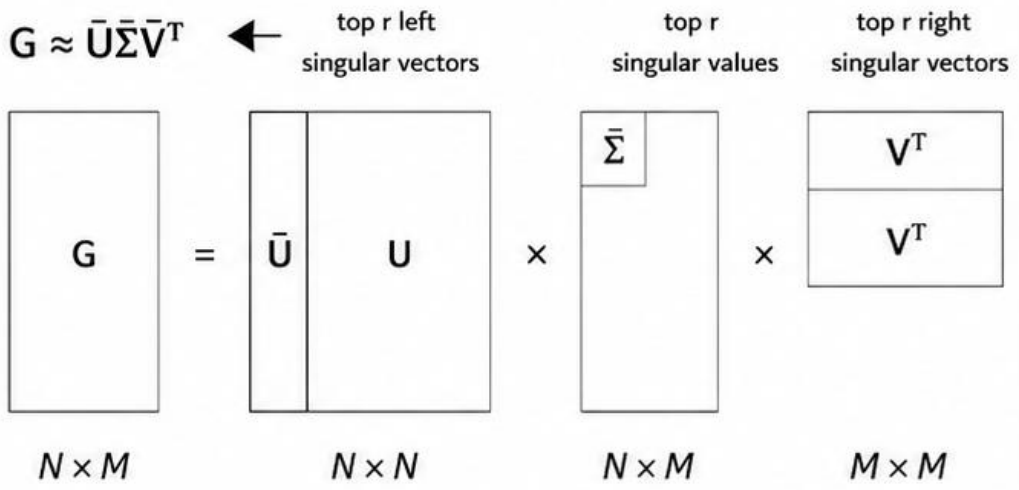
Factorizes user-item rating matrices to discover latent preference vectors (e.g., Netflix algorithm).

Image & Data Compression

Constructs low-rank matrix approximations while preserving primary visual variance.

Natural Language Processing

Underpins Latent Semantic Analysis (LSA) for extracting word-document relationships.



Why SVD is Essential in AI

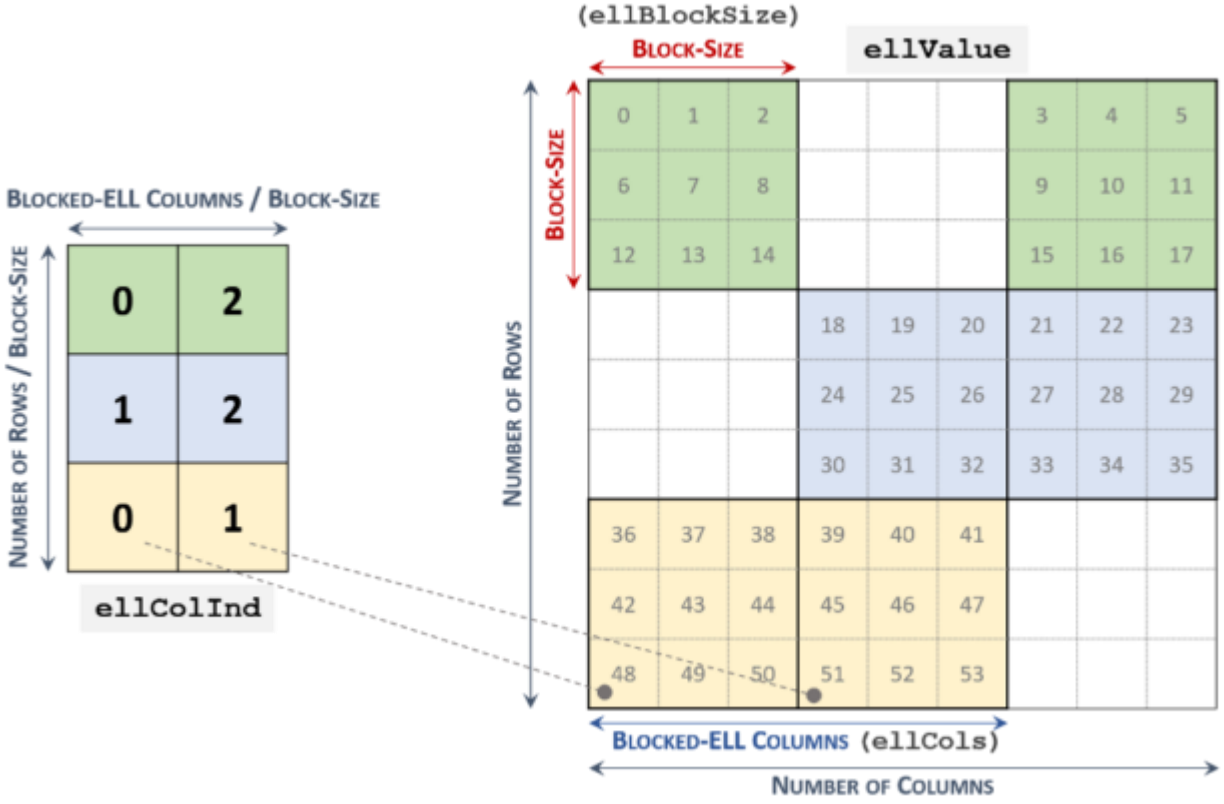
SVD provides the mathematical bedrock for discovering hidden low-rank structures in massive high-dimensional datasets. By filtering out non-essential dimensions, models run faster, require less memory, and generalize significantly better.

Matrix Sparsity

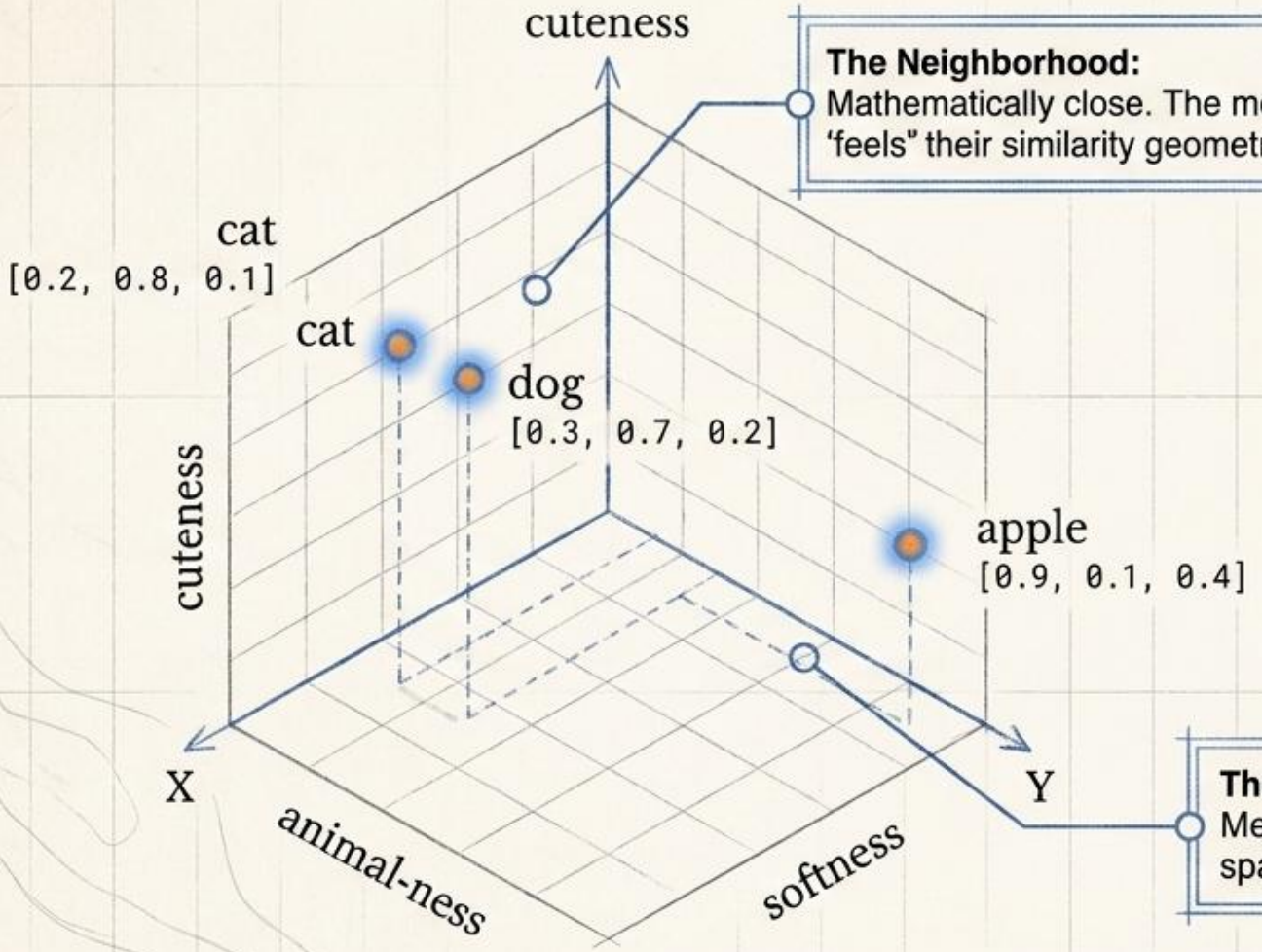
- **Core Concept:** sparsity
- **Mathematical Depth :**
  - **Linear Transformations & Matrix Operators:** How neural network layers represent affine transformations $f(x) = Wx + b$.
- The fine-grained structured sparsity, can accelerate inference workloads by processing only non-zero values in matrix multiplication, potentially doubling compute throughput compared to dense matrix multiplication.
- Tencent's Machine Learning Platform department achieved 1.3-1.8x acceleration in offline services by combining structured sparsity with quantization techniques, and sparse-QAT (quantization-aware training) can be used to further optimize models by combining sparsity with quantization and knowledge distillation.

SpMM

- **Sparse-matrix dense-matrix multiplication (SpMM) is a linear algebra operation and a building block** for more complex algorithms such as finding the solutions of linear systems. SpMM is also an important kernel used in AI. In the specific context of deep learning, **sparsity is one of the leading approaches** for **increasing training and inference performance as well as reducing the model sizes** while maintaining the accuracy.



Bundling Meaning: Vectors as Semantic Coordinates



The Neighborhood:
Mathematically close. The model "feels" their similarity geometrically.

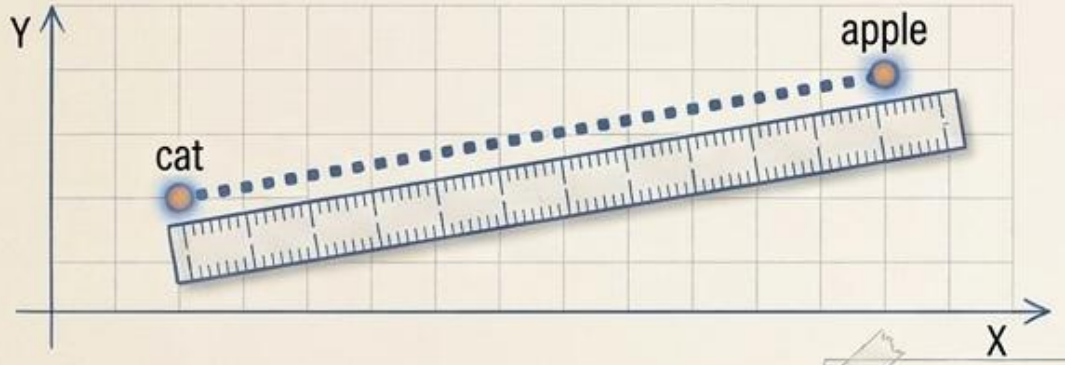
```
# Imagine a tiny "embedding" for words
embedding = {
    "cat": [0.2, 0.8, 0.1],
    "dog": [0.3, 0.7, 0.2],
    "apple": [0.9, 0.1, 0.4]
}

print("cat ->", embedding["cat"])
```

The Void: "Apple" sits far away. Meaning is derived entirely from spatial relationships.

Navigating the Space: Distance vs. Similarity

Euclidean Distance



Measures the raw **straight-line proximity** between two points. Good for absolute magnitude.

Cosine Similarity

Measures the angle between vectors. Even if scales differ, a small angle means they point in the same direction of meaning.

```
Import math

def euclidean(a, b):
    return math.sqrt(sum((x - y)**2 for x, y in zip(a, b)))

def cosine_similarity(a, b):
    dot = sum(x * y for x, y in zip(a, b))
    norm_a = math.sqrt(sum(s**2 for x in a))
    norm_b = math.sqrt(sum(x**2 for x in b))
    return dot / (norm_a + norm_b)

cat = [8.2, 0.8]
dog = [8.3, 0.?]
apple = [8.9, 0.1]

print("cat-dog distance:", euclidean(cat, dog))
print("cat-apple distance:", euclidean(cat, apple))
```

Takeaway: Meanings often grow in scale, but direction carries the essence.

Multivariate Calculus & Automatic Differentiation

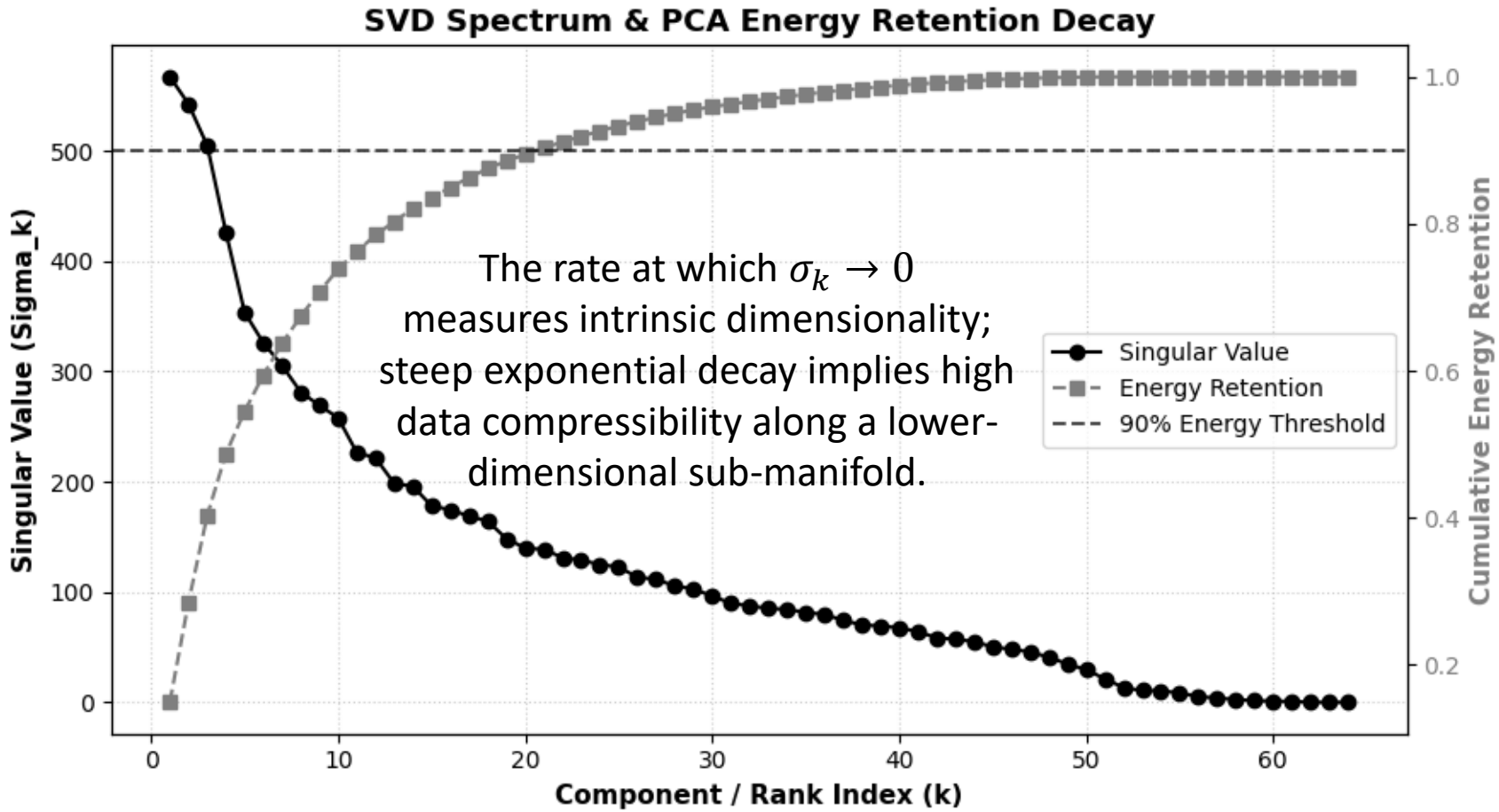
- **Core Concept:** How models compute update direction via gradient vectors and tensor operations.
- **Mathematical Depth :**
  - **Jacobian Matrices:** Role of the Jacobian Jin backpropagation
  - **Chain Rule:** Differential geometry perspective on nested function compositions $f_n\left(f_{n-1}\left(\ldots f_1(x)\right)\right)$.
  - **Reverse-Mode Automatic Differentiation:** How modern frameworks (like MATLAB's dlgradient) evaluate exact derivatives in $\mathcal{O}(1)$time relative to output scalar loss, avoiding numerical finite-difference errors.

Mathematical Optimization & Loss Landscapes

- **Core Concept:** Navigating complex loss surfaces to find global or near-optimal parameter settings.
- **Mathematical Depth :**
  - **Convex vs. Non-Convex Optimization:** Why deep learning works despite losing global convexity guarantees.
  - **First-Order Optimization Variants:** Gradient Descent update step with momentum and adaptive learning rates (e.g., Adam, RMSprop):
  - $\theta_{t+1} = \theta_t - \eta \cdot \frac{m_t}{\sqrt{v_t}+\epsilon}$

SVD_Demo.ipynb

https://drive.google.com/file/d/1o2WwaqN4R7XSeH7dGF2CWW_QxeBpfKvK/view?usp=sharing



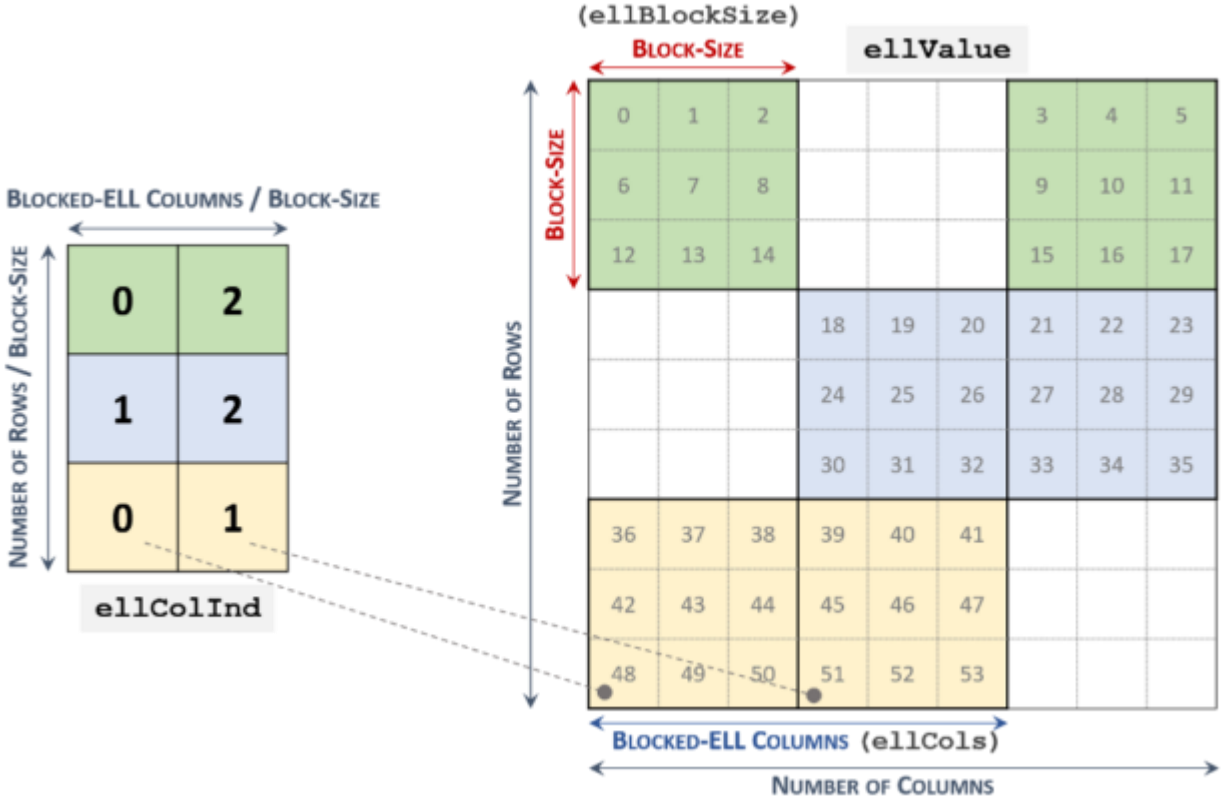
The rate at which $\sigma_k \to 0$ measures intrinsic dimensionality; steep exponential decay implies high data compressibility along a lower-dimensional sub-manifold.

Matrix Sparsity

- **Core Concept:** sparsity
- **Mathematical Depth :**
  - **Linear Transformations & Matrix Operators:** How neural network layers represent affine transformations $f(x) = Wx + b$.
- The fine-grained structured sparsity, can accelerate inference workloads by processing only non-zero values in matrix multiplication, potentially doubling compute throughput compared to dense matrix multiplication.
- Tencent's Machine Learning Platform department achieved 1.3-1.8x acceleration in offline services by combining structured sparsity with quantization techniques, and sparse-QAT (quantization-aware training) can be used to further optimize models by combining sparsity with quantization and knowledge distillation.

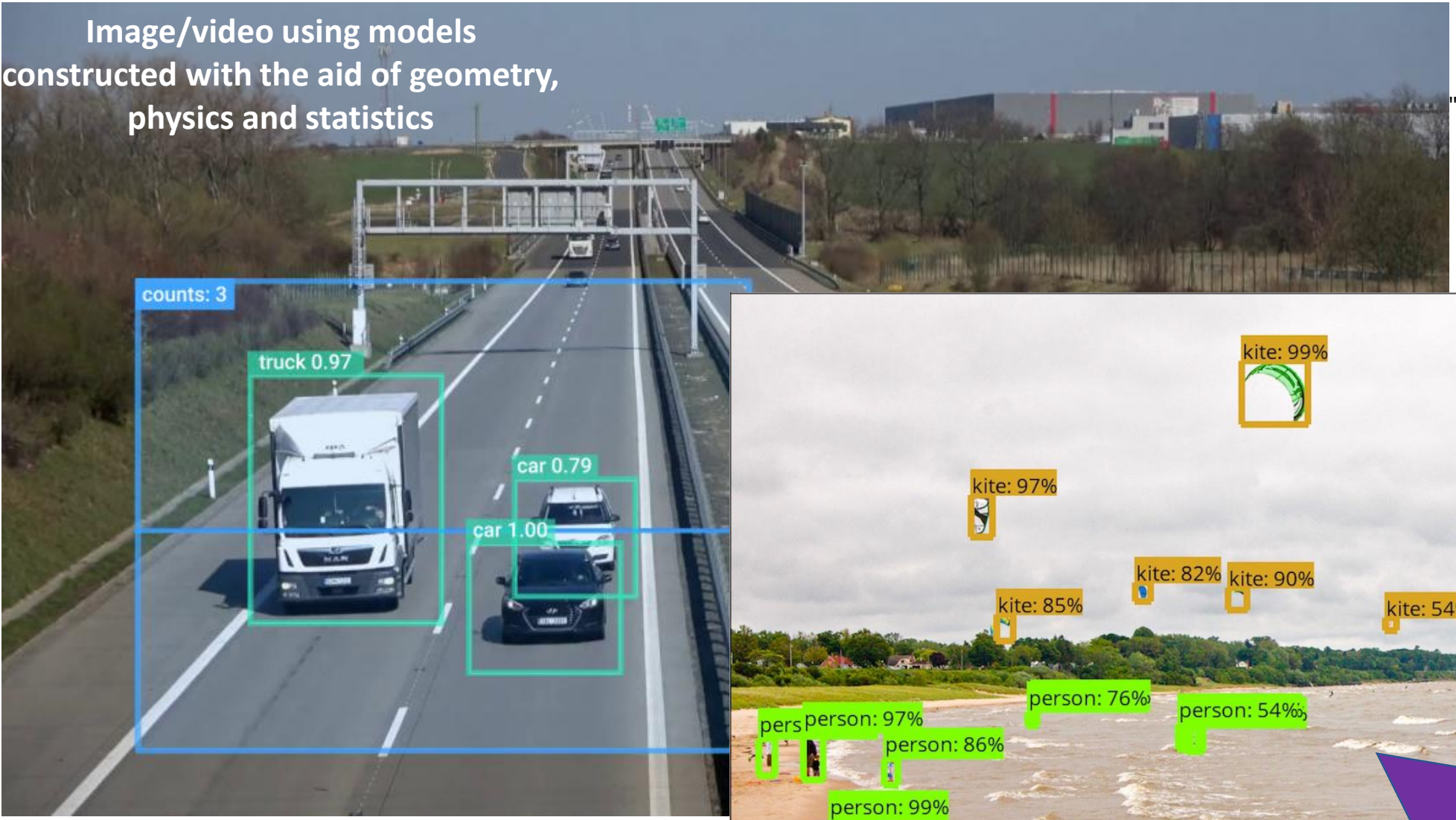
SpMM

- **Sparse-matrix dense-matrix multiplication (SpMM) is a linear algebra operation and a building block** for more complex algorithms such as finding the solutions of linear systems. SpMM is also an important kernel used in AI. In the specific context of deep learning, **sparsity is one of the leading approaches** for **increasing training and inference performance as well as reducing the model sizes** while maintaining the accuracy.



Module 4: Probability, Statistics & Information Theory

- **Core Concept:** Quantifying model uncertainty, information transfer, and maximum likelihood.
- **Mathematical Depth:**
  - **Maximum Likelihood Estimation (MLE) & MAP:** Formulating loss functions (e.g., MSE, Cross-Entropy) from Gaussian and Bernoulli log-likelihoods.
  - **Information Measures:** Entropy
  - $H(P)\mathcal{L}_{CE} = -\sum_x P\,(x) \log Q\,(x)$



"Machine Vision Fundamentals, How to Make Robots See". *NASA Tech Briefs Magazine*. **35** (6)

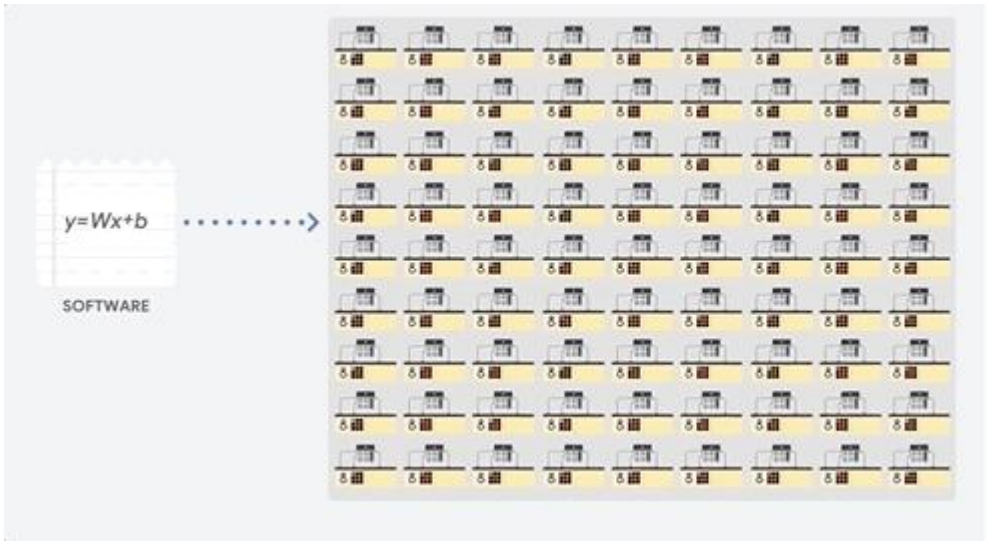
Computer Vision tasks

Fused multiply-add(FMA)

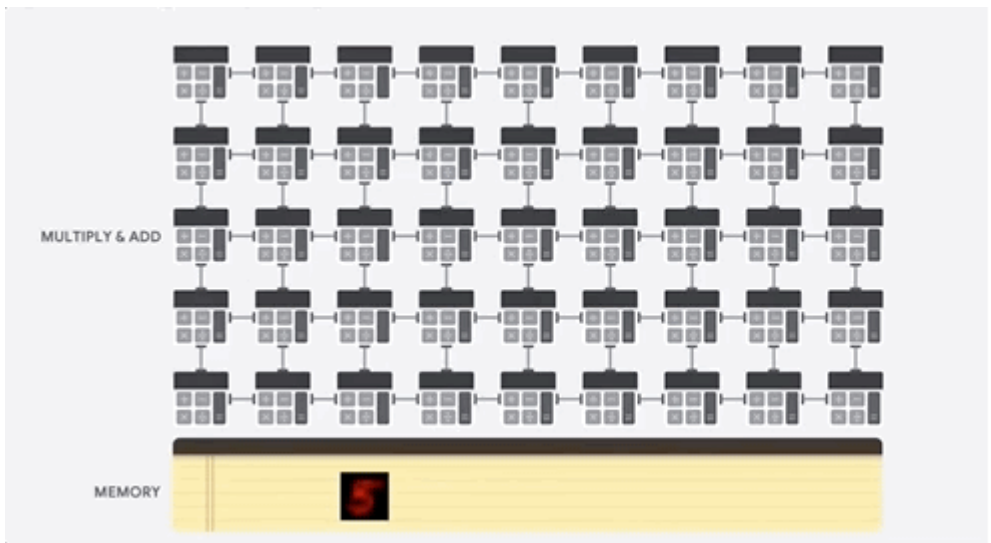
- Multiply-add operation
- Computes the product of two numbers and adds that product to an accumulator
- $a \leftarrow a + (b \times c)$

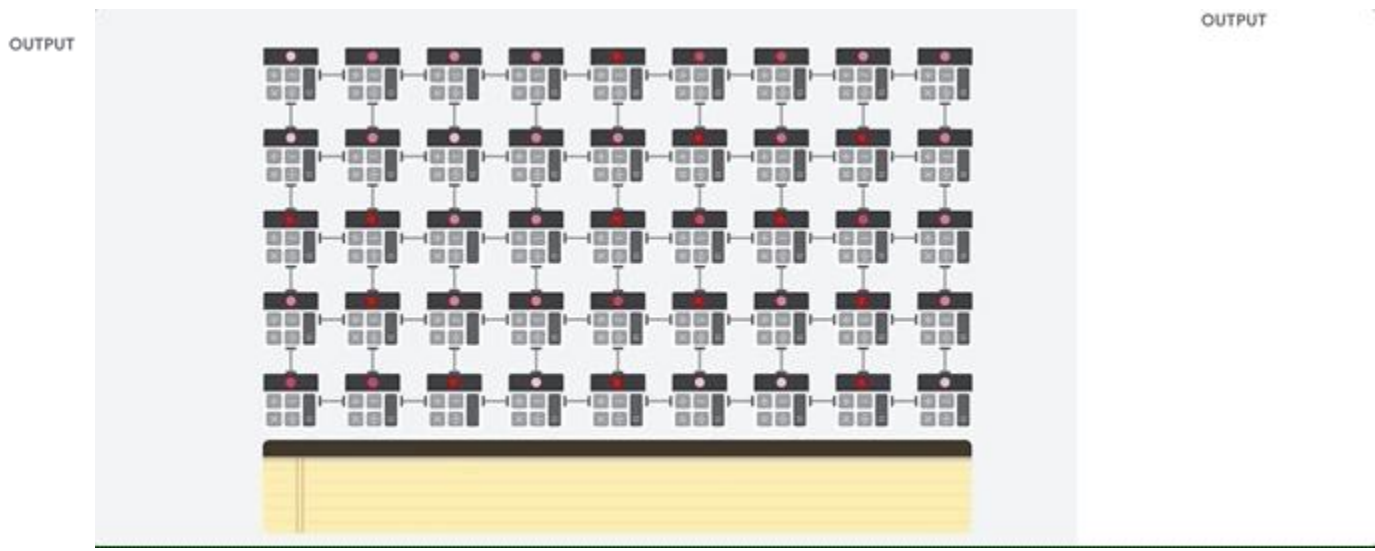
https://cloud.google.com/tpu/docs/intro-to-tpu

How a GPU works

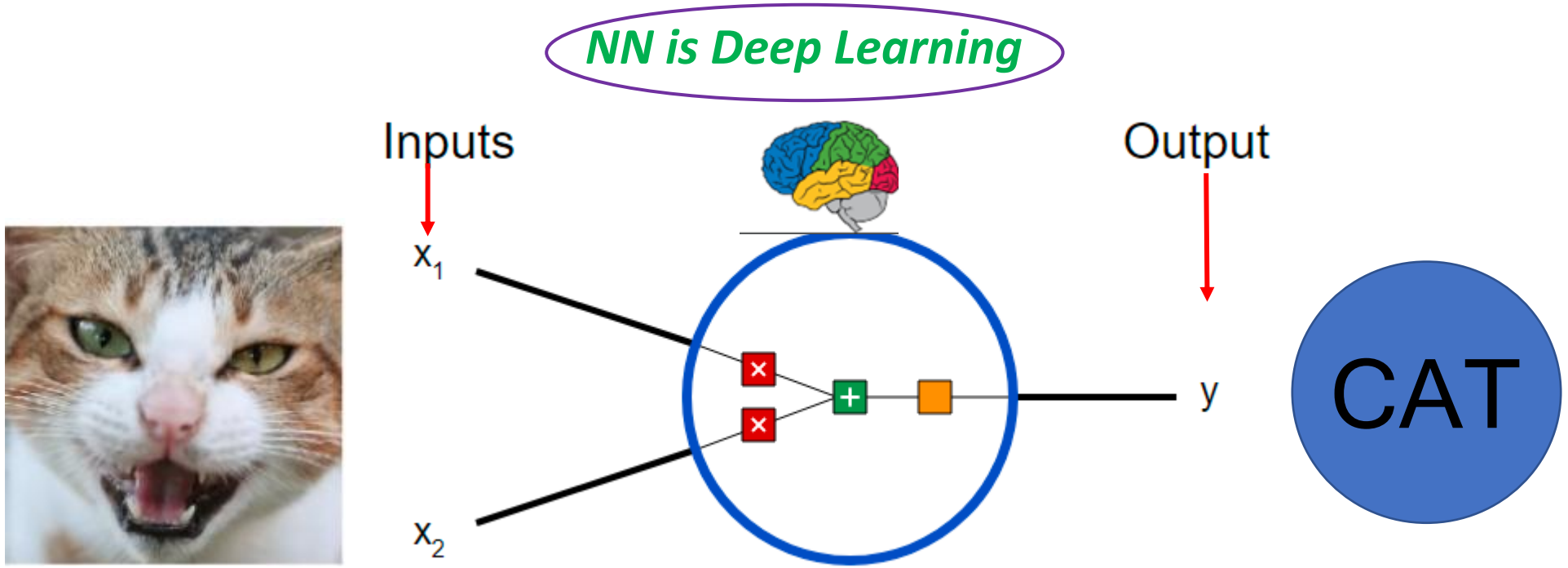


How a TPU works





Building Blocks Neurons - combine Neurons into Neural Network(NN)



Input, processing and output together is also known as ***Perceptron***

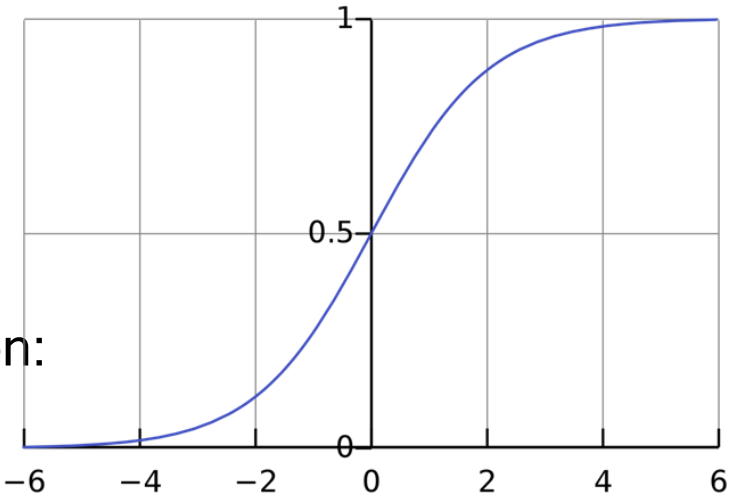
Example: Does the passengers Biometrics identity matches? which is a binary classification (Yes it matches 1 or No it doesn't match 0)

Typical AI workload pattern

Three things are happening here

$$f(x) = \frac{1}{1 + e^{-x}}$$

1. First, each input is multiplied by a weight:
   - x1 → x1 * **w1**
   - x2 → x2 * **w2**
2. all weighted inputs are added together with a bias **b**:
   - (x1 * **w1**) + (x2 * **w2**) + **b**
3. Finally, the sum is passed through an activation function:
   - y = **S** (x1 * w1 + x2 * w2 + b)



- The activation function is used to turn an unbounded input into an output that has a nice, predictable form. A commonly used activation function is the sigmoid function:

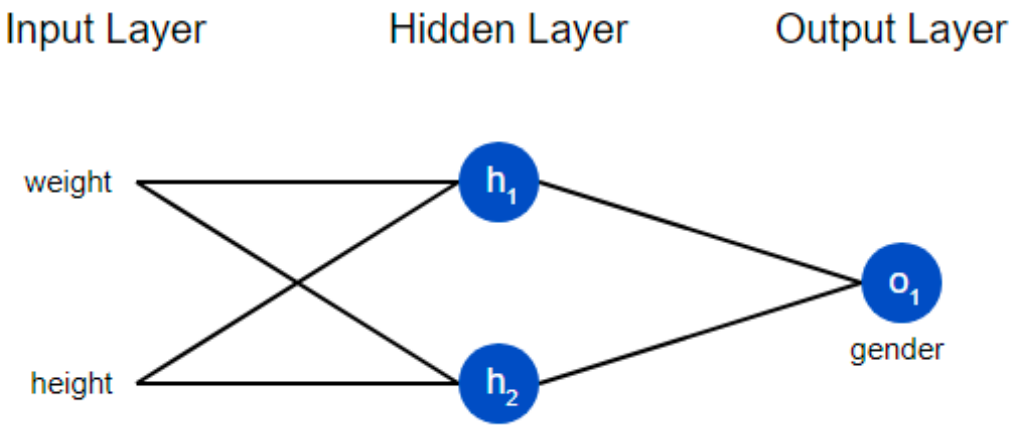
The Neural Networks(NN) from Scratch

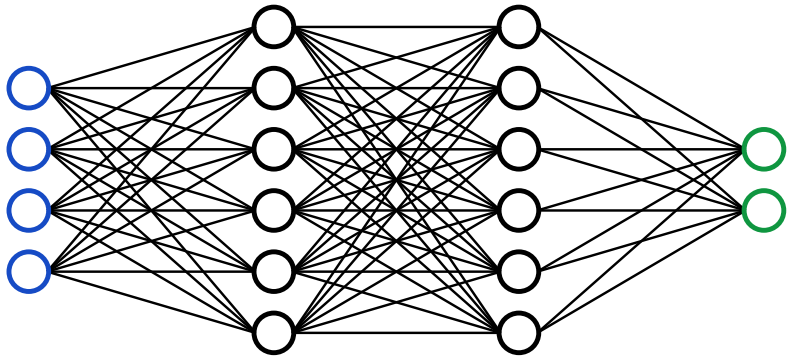
Name	Weight (lb)	Height (in)	Gender
Alice	133	65	F
Bob	160	72	M
Charlie	152	70	M
Diana	120	60	F

Male = 0

Female = 1

Let's train our network to predict someone's gender given their weight and height:





Minimize Loss - Mean Squared Error (MSE)

- First quantify how “good” it’s doing so that it can try to do “better”.
- We’ll use the mean squared error (MSE) Loss:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

Where,

- n is the number of samples, which is 44 (Alice, Bob, Charlie, Diana).
- y represents the variable being predicted, which is Gender.
- y_{true} is the true value of the variable (the “correct answer”).
- For example, y_{true} for Alice would be 11 (Female).
- y_{pred} is the predicted value of the variable. It’s whatever our network outputs.
- $(y_{true} - y_{pred})^{**} 2$ is known as the squared error.
- Our loss function is simply taking the average over all squared errors .

Better predictions = Lower loss.

Training a network = trying to minimize its loss.

Neurons – the building block

- A **neuron** takes inputs, applies weights, adds a bias, and passes the result through an **activation function.**
- Mathematically:

$$y = f(w_1 x_1 + w_2 x_2 + b)$$

- Common activation: **sigmoid**

$$f(x) = \frac{1}{1 + e^{-x}}, \quad f(x) \in (0, 1)$$

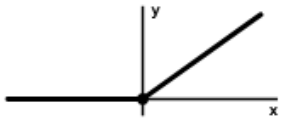
- This process of computing outputs from inputs is called **feedforward.**

ReLU, Tanh, and GELU

More powerful activation functions

ReLU

PIECEWISE LINEAR



EQUATION

$$f(x) = \max(0, x)$$

OUTPUT RANGE $[0, +\infty)$

KEY TRAITS

- Computationally ultra-efficient & sparse activation.
- Mitigates vanishing gradient for positive values.
- Prone to "Dying ReLU" when inputs are negative.

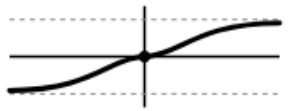
PRIMARY USE CASES

Vision Networks (CNNs): Standard hidden layer activation (ResNet, VGG).

Feedforward MLPs: Default baseline choice for deep tabular neural networks.

Tanh

S-SHAPED SIGMOIDAL



EQUATION

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

OUTPUT RANGE $(-1, +1)$

KEY TRAITS

- Zero-centered output aids optimization convergence.
- Stronger gradients near zero than standard Sigmoid.
- Saturates at extremes, causing vanishing gradients.

PRIMARY USE CASES

Recurrent Architectures: Core gating mechanism in LSTMs and GRU cells.

Generative Models: Final layer activation in GAN generators for normalized outputs.

GELU

SMOOTH STOCHASTIC

EQUATION

$$f(x) = x \cdot \Phi(x) = 0.5x1 + \text{erf}\frac{x}{\sqrt{2}}$$

OUTPUT RANGE $[\approx -0.17, +\infty)$

KEY TRAITS

- Smooth, non-monotonic curvature without hard zero drop.
- Probabilistic gating scales inputs by Gaussian CDF.
- Avoids dead neurons while maintaining gradient flow.

PRIMARY USE CASES

Transformer Models: Default activation in BERT, GPT-3/4, ViT, and modern LLMs.

State-of-the-Art NLP/Vision: High-capacity networks requiring fine probabilistic tuning.

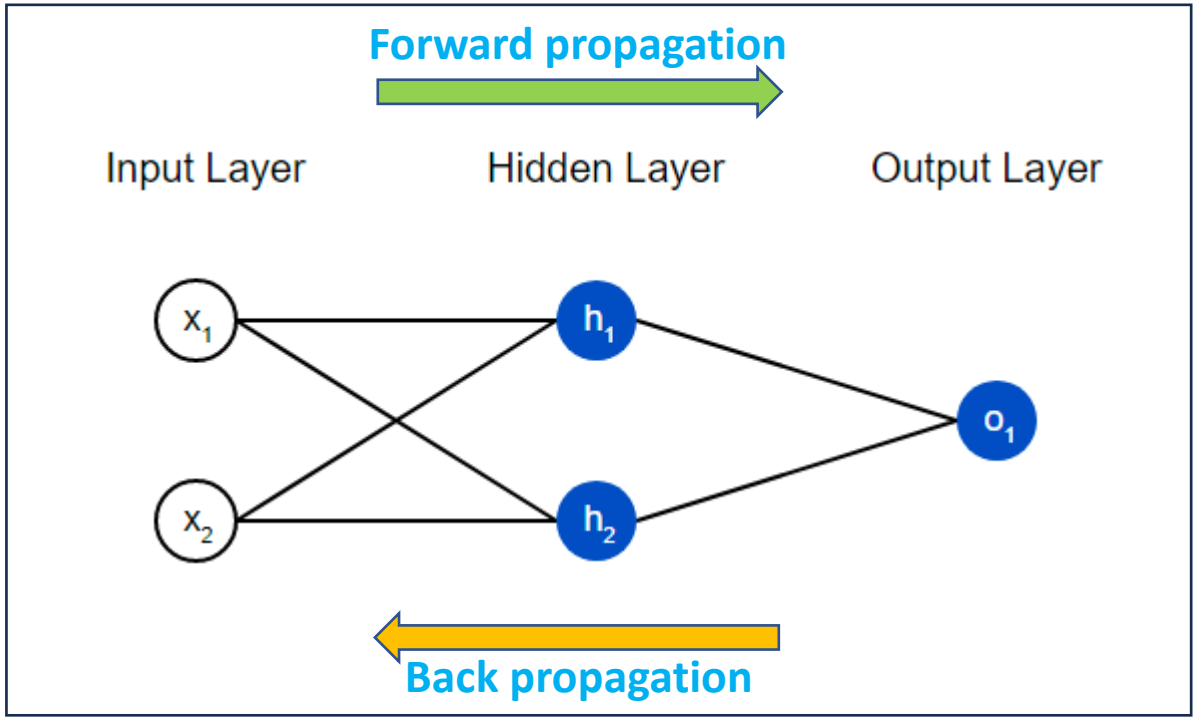
Neural Network Architecture

- A **neural network** = multiple neurons connected in layers.
- Example architecture:
  - 2 inputs → **hidden layer** (2 neurons: h_1, h_2) → **output layer** (1 neuron: o_1).
- Feedforward example:

$$h_1 = f(w \cdot x + b), \quad h_2 = f(w \cdot x + b)$$
$$o_1 = f(w \cdot [h_1, h_2] + b)$$

- Networks can have **any number of layers and neurons**; the principle remains the same.

Combining Neurons into Neural Network(NN)



This network has 2 inputs, a hidden layer with 2 neurons (*h*1 and *h*2), and an output layer with 1 neuron (*o*1). Notice that the inputs for *o*1 are the outputs from *h*1 and *h*2 - that's what makes this a network.

Loss function – Measuring performance

- A **neural network** = multiple neurons connected in layers.
- Example architecture:
  - 2 inputs → **hidden layer** (2 neurons: h_1, h_2) → **output layer** (1 neuron: o_1).
- Feedforward example:

$$h_1 = f(w \cdot x + b), \quad h_2 = f(w \cdot x + b)$$
$$o_1 = f(w \cdot [h_1, h_2] + b)$$

- Networks can have **any number of layers and neurons**; the principle remains the same.

Back propagation – computing gradients

- To minimize loss, compute how loss changes with each weight:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot \frac{\partial y_{\text{pred}}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

- Uses the **chain rule** of calculus.
- Sigmoid derivative (used repeatedly):

$$f'(x) = f(x) \cdot (1 - f(x))$$

- This backward pass to compute partial derivatives is called **backpropagation.**

Training - Stochastic Gradient Descent(SGD)

- Update rule for each weight/bias:

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

  where η = **learning rate.**

- **SGD process:**

  1. Pick one training sample.
  2. Compute all partial derivatives via backprop.
  3. Update weights and biases.
  4. Repeat over many **epochs.**

- Result: loss decreases, network learns to predict accurately.

GRADIENT DESCENT

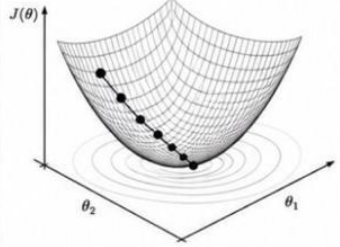
MATHEMATICS CHEAT SHEET

1. THE BIG PICTURE

Goal: Find parameters θ^* that minimize a loss function $J(\theta)$.

$$\theta^* = \arg\min_{\theta} J(\theta)$$

Idea: Start anywhere, move in the direction that decreases the loss the most.



- Start
- Gradient Descent Path
- ★ Minimum

2. GRADIENT DESCENT UPDATE RULE

At iteration t, update parameters as:

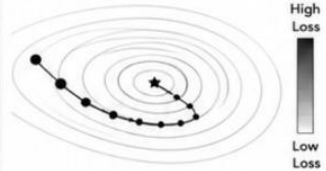
$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

θ_t : parameter vector at step t
η : learning rate (step size)
$\nabla J(\theta_t)$: gradient of the loss w.r.t. θ

In component form:

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} - \eta \frac{\partial J(\theta_t)}{\partial \theta^{(i)}}$$

We move in the opposite direction of the gradient (steepest ascent) to minimize the loss.



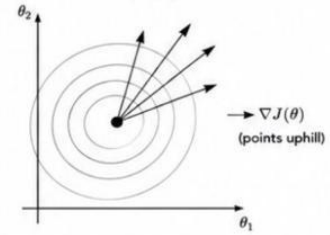
3. GRADIENT

The gradient points in the direction of steepest increase of the function.

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \in \mathbb{R}^n$$

For a function $J(\theta_1, \theta_2)$:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

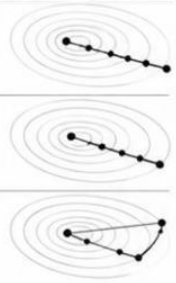


4. LEARNING RATE (η)

Too small η
- Very slow convergence
- Many steps

Good η
- Fast convergence
- Stable

Too large η
- Overshoot minimum
- May diverge



Choosing η is crucial for efficient optimization.

5. CONVERGENCE

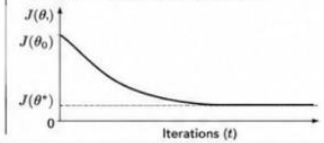
If $J(\theta)$ is convex and η is small enough, gradient descent converges to the global minimum.

For L-smooth convex functions:
$0 < \eta < \frac{2}{L}$ ensures convergence.

Rate of convergence (roughly):

$$J(\theta_t) - J(\theta^*) \leq (1 - \eta\mu)^t \left(J(\theta_0) - J(\theta^*)\right)$$

where μ = strong convexity constant.



6. COMMON VARIANTS

Batch Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

Uses all training data to compute the gradient.

Stochastic Gradient Descent (SGD)

$$\theta_{t+1} = \theta_t - \eta \nabla J_i(\theta_t)$$

Uses one random training example i.

Mini-Batch Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla J_{\mathcal{B}}(\theta_t)$$

Uses a small batch $\mathcal{B}$ of examples.

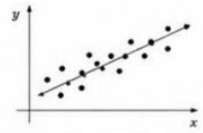
7. EXAMPLES OF GRADIENTS

(a) Linear Regression (MSE)

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

where $\hat{y}_i = wx_i + b$

$$\frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) x_i$$
$$\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

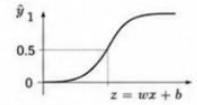


(b) Logistic Regression (Binary Cross-Entropy)

$$J(w,b) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

$$\hat{y}_i = \sigma(wx_i + b)$$

$$\frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) x_i$$
$$\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

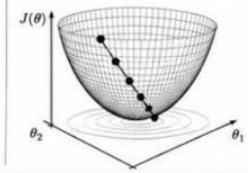


(c) Quadratic Function

$$J(\theta) = \frac{1}{2} \theta^T A \theta - b^T \theta + c$$

(A symmetric positive definite)

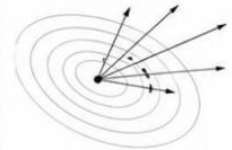
$$\nabla J(\theta) = A\theta - b$$
$$\theta_{t+1} = \theta_t - \eta (A\theta_t - b)$$



(d) Regularized Loss (L2)

$$J(\theta) = L(\theta) + \frac{\lambda}{2} \|\theta\|_2^2$$

$$\nabla J(\theta) = \nabla L(\theta) + \lambda\theta$$
$$\theta_{t+1} = \theta_t - \eta \left(\nabla L(\theta_t) + \lambda\theta_t\right)$$



AI Intelligence = Mathematics + Data + Computation

The Core Idea
Data → Mathematical Representation → Learning → Prediction

Example: Neural Network
`Input (X)` → **Weights (W)** → `Σ(WX) + b` → **Activation** → `Prediction (ŷ)`

Key Takeaway
AI does not "think" magically — it learns patterns by applying mathematical operations to data.

Linear algebra

- Linear algebra is the backbone of AI, especially in machine learning. Data is typically represented in the form of matrices and vectors, which allow for manipulation and analysis of data on a large scale. Operations like matrix multiplication are used in artificial neural networks to compute the output of each layer. For example, in image analysis, image pixels are represented as matrices, and matrix transformations are used to detect patterns. Additionally, methods such as Singular Value Decomposition (SVD) are used in data compression and recommendation systems.

Singular Value Decomposition(SVD) in AI

$$A = U \Sigma V^T$$

Decomposes matrix A (m × n) into orthogonal and diagonal matrices

Matrix U

Left singular vectors representing row feature relationships.

Matrix Σ

Diagonal matrix of singular values indicating feature importance weights.

Matrix V^T

Right singular vectors capturing column attribute patterns.

Key AI/ML Applications

Dimensionality Reduction

Powers Principal Component Analysis (PCA) by keeping top-k singular values to remove noise.

Recommender Systems

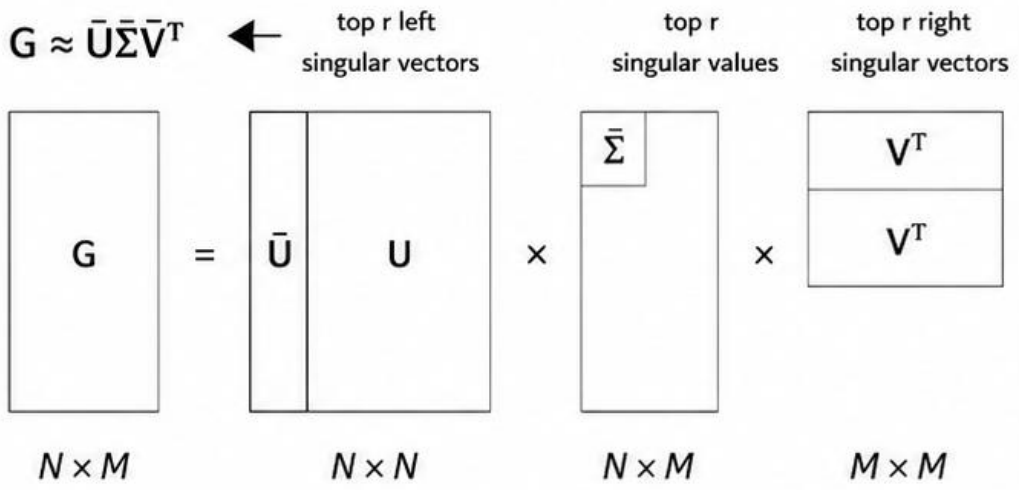
Factorizes user-item rating matrices to discover latent preference vectors (e.g., Netflix algorithm).

Image & Data Compression

Constructs low-rank matrix approximations while preserving primary visual variance.

Natural Language Processing

Underpins Latent Semantic Analysis (LSA) for extracting word-document relationships.



Why SVD is Essential in AI

SVD provides the mathematical bedrock for discovering hidden low-rank structures in massive high-dimensional datasets. By filtering out non-essential dimensions, models run faster, require less memory, and generalize significantly better.

Evolution of AI Architectures

HISTORICAL STANDARD

Sequential Processing

RNNs & LSTMs

Designed for text, time-series, and audio. Processes data step-by-step while maintaining an internal memory state.

- **Sequential Bottleneck:** Cannot train efficiently in parallel across GPUs.
- **Vanishing Gradients:** Struggles with long-range context across long documents.
- **Status:** Largely replaced by Transformers in modern NLP.

MODERN PARADIGM

Unified Self-Attention

Transformers

Replaces recurrence and convolutions with global self-attention mechanisms across sequence tokens or image patches.

- **Parallel Scaling:** Trains efficiently across massive GPU clusters.
- **Language (LLMs):** Powers GPT-4, LLaMA, and Claude.
- **Vision (ViT):** Divides images into visual patches for state-of-the-art vision tasks.

How Large Language Models (LLMs) Work

A step-by-step breakdown from user prompt to generated text response

STEP 01

Tokenization

WORDS TO NUMBERS

Splits prompt into small word pieces ("tokens") and converts each into a unique numerical ID numbers.

INPUT MAPPING

"The cat sat"

➔ [464, 3797, 3374]

STEP 02

Embeddings

MAPPING MEANING

Maps numbers into high-dimensional space where words with similar meanings sit close together.

VECTOR RELATIONSHIP

King - Man + Woman

= "Queen"

STEP 03

Self-Attention

CONNECTING CONTEXT

Weighs relationships between words to resolve ambiguities (e.g., matching "it" to the right subject).

CONTEXT LINKAGE

"The animal crossed because **it** was tired."

('it' ➔ 'animal')

STEP 04

Next-Word Prediction

PROBABILITY LOOP

Calculates statistical odds for every possible next word, picks the best one, and repeats the loop.

PROBABILITY ODDS

"mat" 85%

"rug" 10%

THE ONE IDEA THAT MADE MODERN AI POSSIBLE

"ATTENTION IS ALL YOU NEED": HOW TRANSFORMERS REWIRED AI

BEFORE ATTENTION: SEQUENTIAL (RNNs, LSTMs)

Slow, word-by-word reading

Forgot the start

Struggled with long sentences

AFTER ATTENTION: PARALLEL (TRANSFORMER)

Looks at everything at once

Attention

Decides what to care about for each word

Keeps track of long context

MULTI-HEAD ATTENTION: MANY PERSPECTIVES AT ONCE

By letting AI process text in parallel and focus on what matters most using Attention, Transformers **enabled the large-scale** models behind **ChatGPT, Gemini,** and **Claude**.

WHY THIS UNLOCKED GIANT MODELS

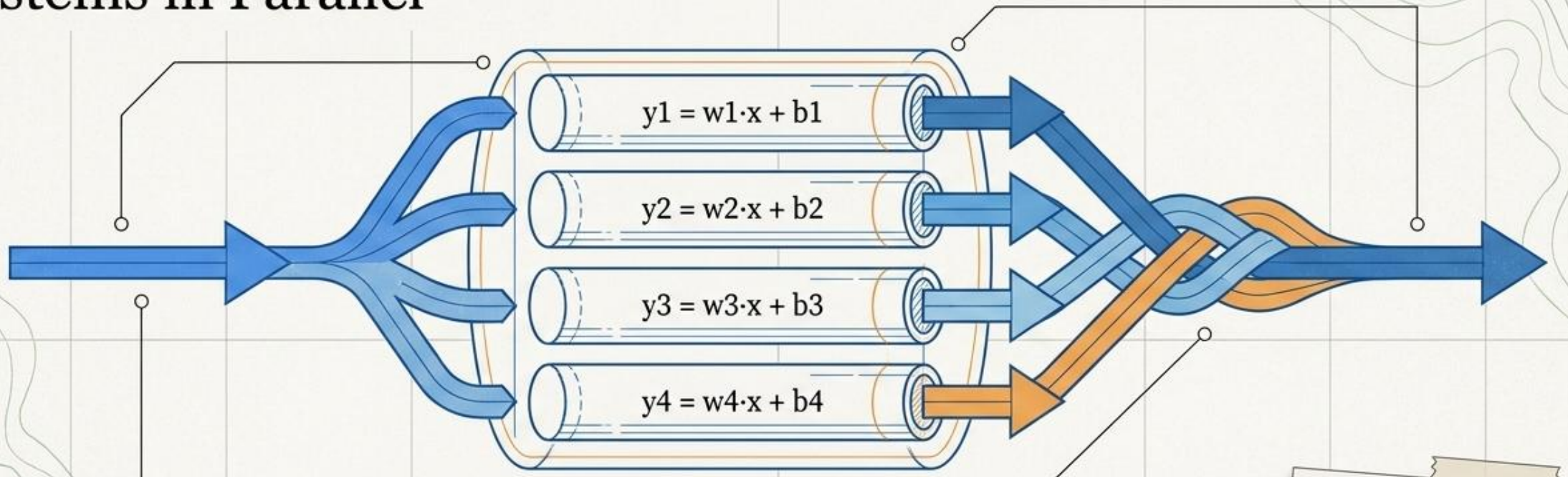
1) HANDLES LONG CONTEXT WELL

GPUs

2) RUNS FAST ON MODERN HARDWARE

AI stores knowledge (weights), learns (loops), thinks (layers), and uses attention to decide what to care about before next-word prediction.

The Architecture of a Layer: Systems in Parallel



The System

Every neural layer is a giant system of equations sharing the same inputs.

The Division of Labor

One equation might detect a verb. Another might detect an animal. Another traces sentence position.

The Synthesis

Each neuron sees the world slightly differently. Together, they build a shared, multi-dimensional understanding.

```
import numpy as np

# Two equations, two variables
W = np.array([
    [2, 3],  # y1 = 2x1 + 3x2
    [1, 4]   # y2 = 1x1 + 4x2
])
x = np.array([1, 2]) # x1 = 1, x2 = 2
b = np.array([0.5, 1.0]) # biases

y = W.dot(x) + b
print("Outputs:", y)
```

Gemini Notebook

Evolution of AI Architectures

EFFICIENT VISION CORE

Spatial Grid Processing
CNNs

Designed for 2D/3D images and video. Uses sliding filters to extract local hierarchical visual features.

- **Spatial Inductive Bias:** Highly effective at detecting edges, textures, and shapes.
- **High Efficiency:** Low memory footprint; ideal for mobile & edge devices.
- **Status:** Widely used today, alongside Vision Transformers (ViT).

MODERN PARADIGM

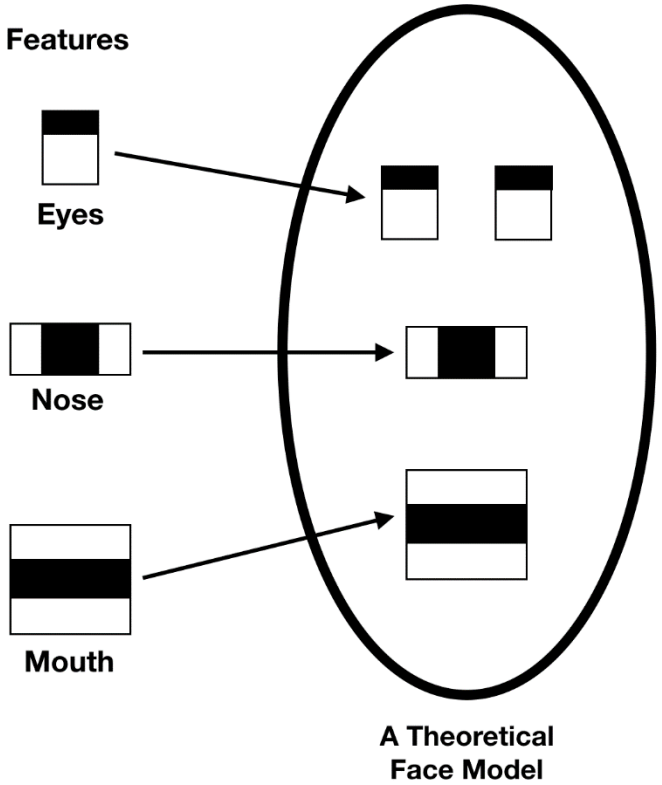
Unified Self-Attention
Transformers

Replaces recurrence and convolutions with global self-attention mechanisms across sequence tokens or image patches.

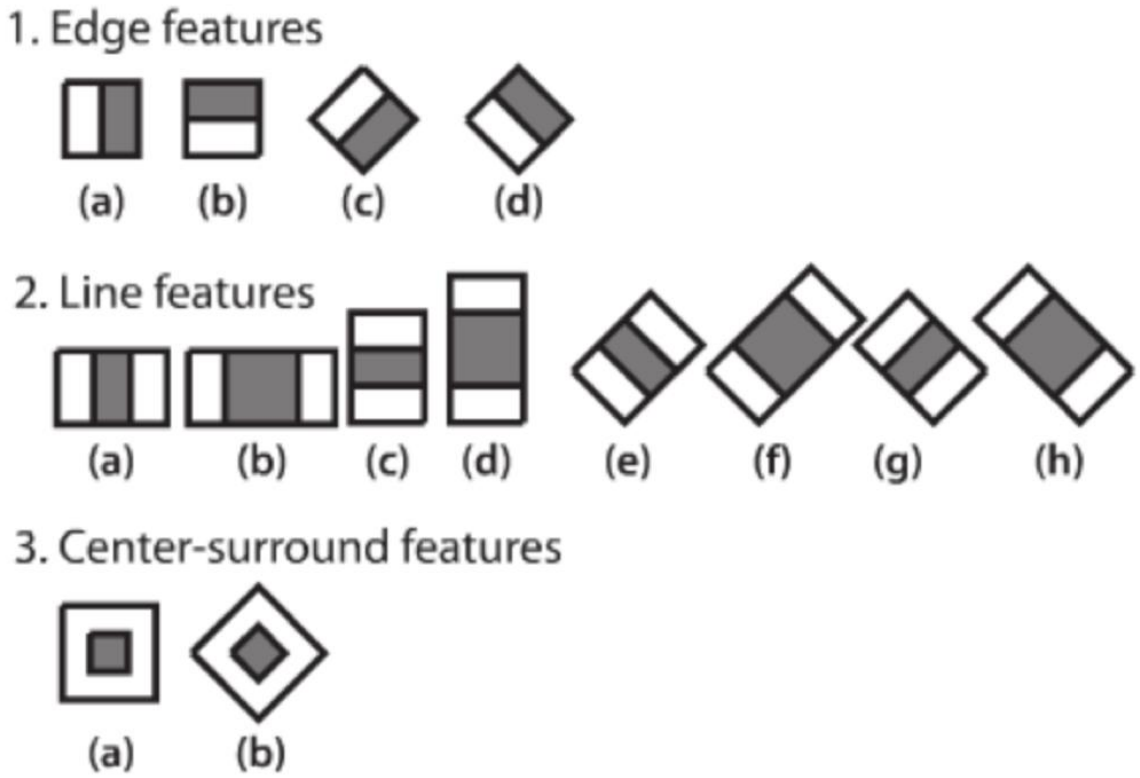
- **Parallel Scaling:** Trains efficiently across massive GPU clusters.
- **Language (LLMs):** Powers GPT-4, LLaMA, and Claude.
- **Vision (ViT):** Divides images into visual patches for state-of-the-art vision tasks.

Face Model

- A theoretical face Model



Difference between Face detection and Face Recognition



Face Detection determines the locations and sizes of human faces in arbitrary (digital) images.

In **Face Recognition**, the use of Face Detection comes first to determine and isolate a face before it can be recognized.

Face_Recognition_Demo

cd 'D:\EDGE AI\DAY2\opencv-course-master\opencv-course-master\Section #3 - Faces’

python face_detect.py

Linear Algebra for Face Recognition

$$\Omega = U^T \Gamma - \Psi$$

Projecting mean-centered face vector $(\Gamma - \Psi)$ onto Eigenface space U^T

1. Vectorization of Pixel Grids

Grayscale face images ($N \times N$) are flattened into high-dimensional vectors $\Gamma \in \mathbb{R}^{N^2}$ representing points in face space.

2. Covariance & Eigenfaces (PCA)

Computes the covariance matrix $C = AA^T$. Eigenvectors with highest eigenvalues capture fundamental facial features (eyes, nose, jawline).

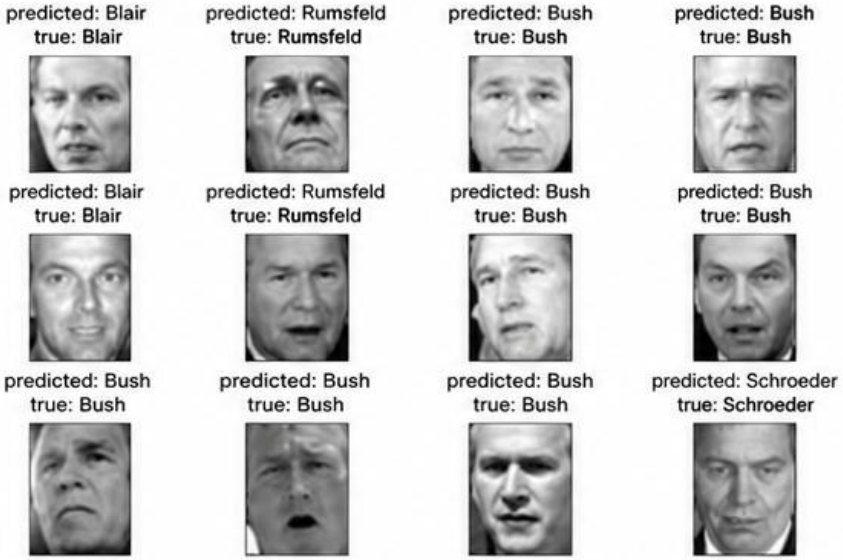
3. Dimensionality Reduction

Retains top k eigenvectors to compress thousands of pixel dimensions into a lightweight weight vector Ω.

4. Vector Distance Classification

Recognizes unknown faces by measuring Euclidean distance $||\Omega_{new} - \Omega_{known}||$ or Cosine Similarity in vector space.

Eigenfaces facial recognition visual



Why Linear Algebra Drives Vision AI

By treating images as mathematical vectors, Linear Algebra turns subjective visual matching into deterministic matrix operations. Modern face recognition (including Deep Face Embeddings like FaceNet) relies on these exact inner-product and vector-space transformations.

Eigenvectors | Matrix Projections | Euclidean Distance | Latent Space

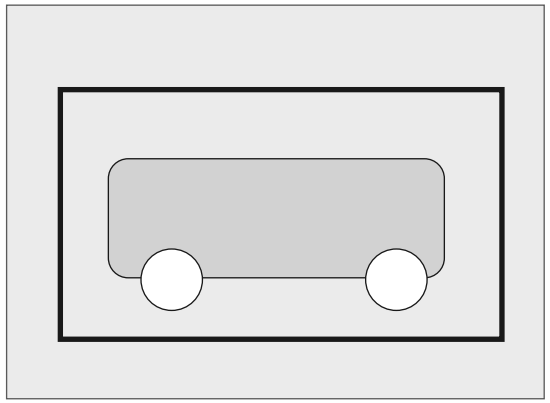
The Role of Mathematics in Object Detection

From pixels and data → mathematical representation → prediction

OBJECT DETECTION DATASET — **LINEAR ALGEBRA** — **CONVOLUTION** — **PROBABILITY** — **GEOMETRY**

Images • Labels • Bounding Boxes | Pixels → Vectors / Tensors | Kernels → Visual Features | P(class) → What is it? | Bounding Box + IoU → Where is it?

1 DATA BECOMES NUMBERS



Bounding box = (x, y, w, h)

2 MATHEMATICS TURNS DATA INTO LEARNING

LINEAR ALGEBRA
Image $\in \mathbb{R}^{H \times W \times c}$
Matrix / tensor operations

CONVOLUTION
$Y(i,j)=\Sigma X \cdot K$
Extract visual features

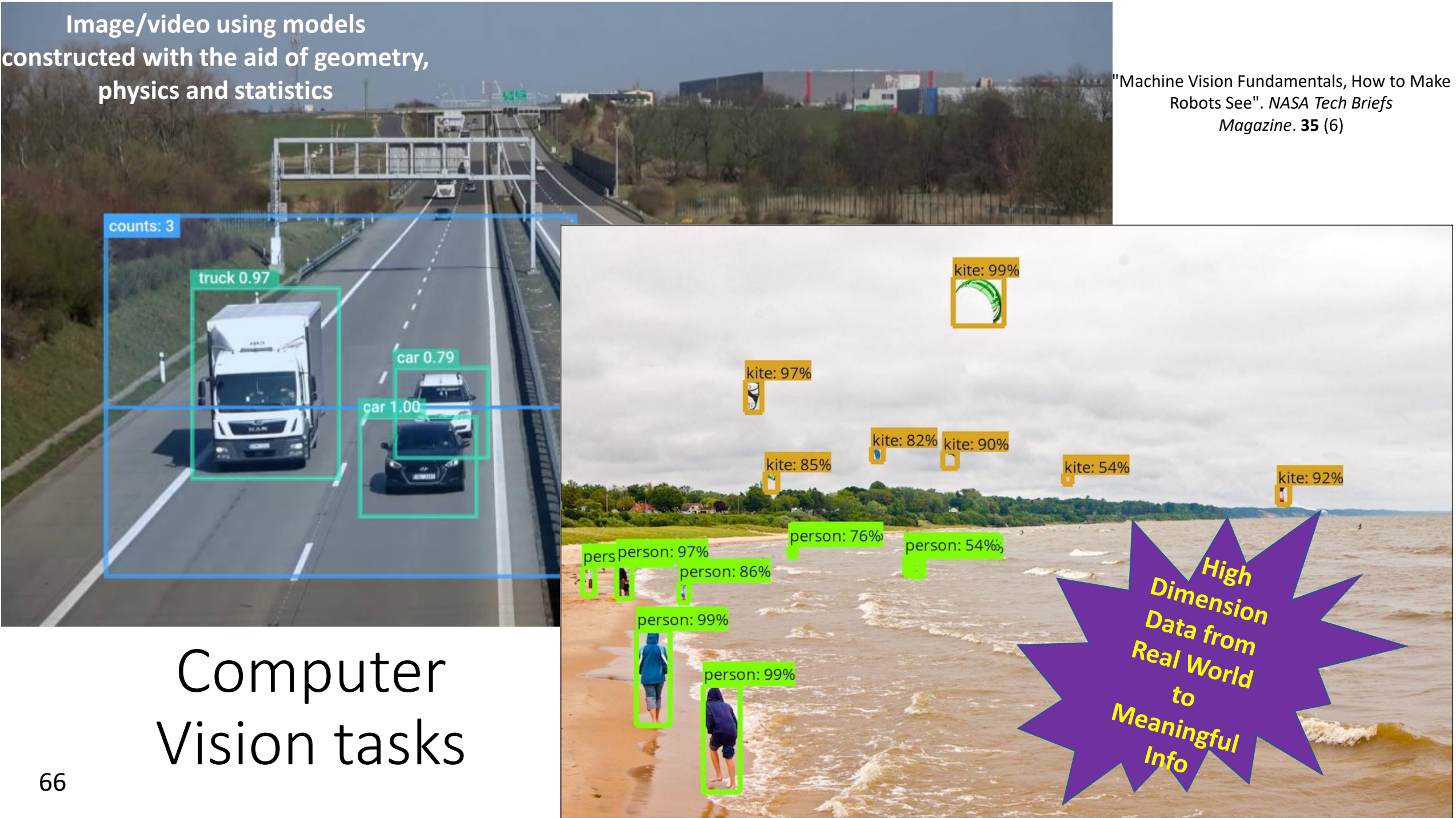
PROBABILITY
P(class | image)
“What object is it?”

GEOMETRY
IoU = Intersection / Union
“How well is it located?”

OPTIMIZATION
Loss → Gradient → Update
“How does it improve?”

DEEP LEARNING
Features → Prediction
YOLO / R-CNN / SSD

MATHEMATICS = REPRESENT • DETECT • COMPARE • OPTIMIZE • PREDICT



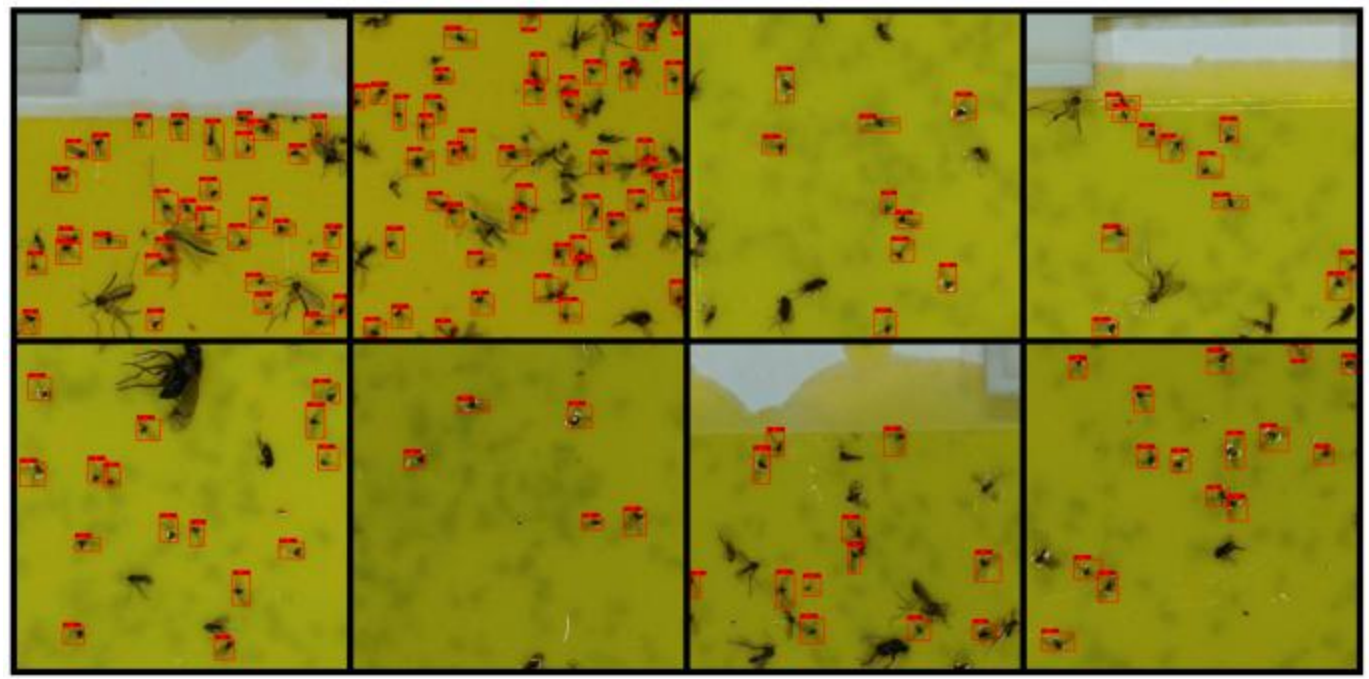
"Machine Vision Fundamentals, How to Make Robots See". *NASA Tech Briefs Magazine*. **35** (6)

Computer Vision tasks

AI based

early pest warning system

Large agricultural projects in India depend on manual counting of pests and pose challenges with identifying pests accurately and offering guidance to the farmers



Key Takeaway

- **AI offers many research innovation possibilities for the student of mathematics**
- **Promising mathematics areas with research potential are**
  - **Optimization**
  - **Data compression**
  - **Performance enhancement**
  - **New algorithms**

Important References

1. **Mathematics and Artificial Intelligence: The Foundation of Future Technology** https://math.fmipa.ugm.ac.id/en/matematika-dan-kecerdasan-buatan-fondasi-yang-membentuk-teknologi-masa-depan/, December 13th, 2024
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
3. Strang, G. (2009). Introduction to Linear Algebra. Wellesley-Cambridge Press.
4. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
5. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
6. CS231n Deep Learning for Computer Vision - https://cs231n.github.io/convolutional-networks/
7. https://www.ibm.com/adobe/dynamicmedia/deliver/dm-aid--8503d244-b389-47ae-825b-b3f35c71ea2d/svd.png?preferwebp=true
8. https://arxiv.org/html/2501.10465v1
9. **Numpy**, https://numpy.org/
10. **Scipy**, https://scipy.org/
11. https://victorzhou.com/series/neural-networks-from-scratch/

Thank you !

The Foundations of Intelligence: The Mathematics Behind AI

ICMMAICS - 17 August 2026

sambath.narayanan@dataeverconsulting.com

The Grammar of Math: Anatomy of a Linear Equation

W (The Learned Perspective)
The weight matrix. The model's internal memory that dictates how strongly **x** influences the output.

x (The Raw Material)
The input variable. The current state of the token's meaning.

$$y = W \cdot x + b$$

y (The Transformed Meaning)
The output. The new feature generated by the layer.

b (The Baseline Shift)
The bias. A subtle adjustment moving the meaning up or down.

This humble formula is the engine of learning. Variables (`x`) hold the data; parameters (`W, b`) hold the memory.

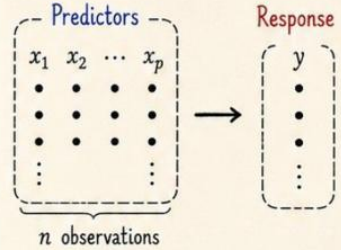
GENERAL LINEAR MODEL

Sir Ronald A. Fisher

(February 17, 1890 – July 29, 1962)

General concept of the model

$$y = X\beta + \varepsilon$$



X: design matrix $(n \times (p+1))$

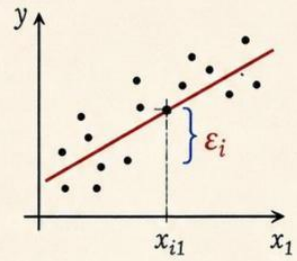
β: coefficient vector $((p+1) \times 1)$

y: response vector $(n \times 1)$

ε: error vector $(n \times 1)$

For the i-th observation:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i$$



Regression line

Observed data

Matrix representation of the general linear model

$$= \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{n \times 1} \times \underbrace{\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}}_{X,\ n \times (p+1)} : \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}}_{\beta,\ (p+1) \times 1} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}}_{\varepsilon,\ n \times 1}$$

Ordinary least squares (OLS) estimator

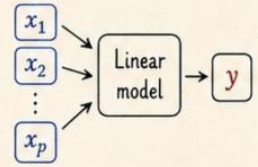
$$\hat{\beta} = (X^T X)^{-1} X^T y$$

$\hat{\beta}$: estimator of β

1. What does the model describe?

The general linear model describes the relationship between a response variable and a set of explanatory variables through a linear combination and an error term.

- y: response vector
- X: design matrix
- β: parameter vector
- ε: error vector



2. Basic assumptions:

- Linearity
- $E(\varepsilon) = 0$
- $\mathrm{Var}(\varepsilon) = \sigma^2 I$
- Independence
- Normality (optional for inference)