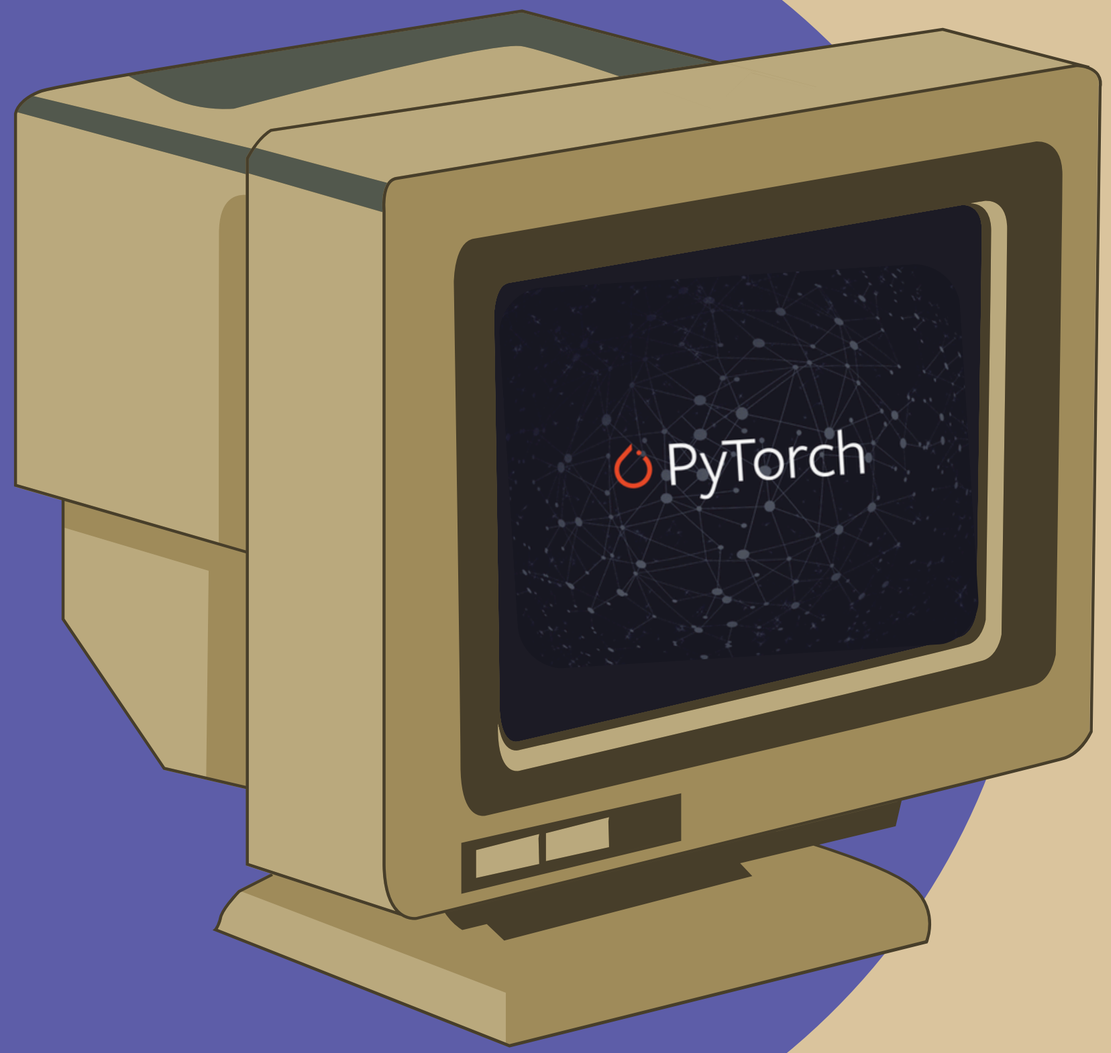


IETE STUDENT FORUM - AI & ML VERTICAL

TORCH IT UP

AI JOURNEY WITH PYTORCH

HTTPS://AVVSF.IETECBE.ORG/

BEFORE WE START...

Set Up Your Environment:
- If your local system is not set up with Python, sign up for Google Colab (https://colab.research.google.com/).
- Create an account on Kaggle (https://www.kaggle.com/).
- Create an account on GitHub to access workshop materials. (https://github.com/)
- Ensure Python version >= 3.9 is installed.

Install Required Python Libraries:
- For CPU Users:
  pip install torch torchvision
- For GPU Users: (if your system supports CUDA)
  pip install torch torchvision --index-url https://download.pytorch.org/whl/cu118
- Additional libraries (for everyone):
  pip install numpy matplotlib pandas scikit-learn tensorboard

MAKE YOUR SUBMISSIONS HERE!!

ML hackathon

IETE STUDENT FORUM - AI & ML VERTICAL

AI & ML: THE BRAIN BEHIND SMART MACHINES!

💡 **Ever wondered...**

How *Netflix* recommends your next binge-watch? 🎥

How voice assistants like *Alexa* and *Siri* get smarter over time? 🔊

How *Google Photos* organizes memories without labels? 📸

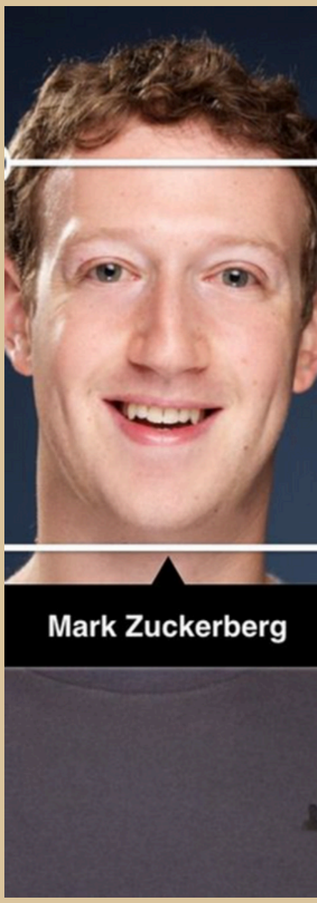
How *ChatGPT* understands and responds to you? 📱

How *Spotify* creates the perfect playlist just for you? 🎶

How AI predicts the weather or tracks hurricanes? ☁️

How *facial recognition* unlocks your phone? 😉

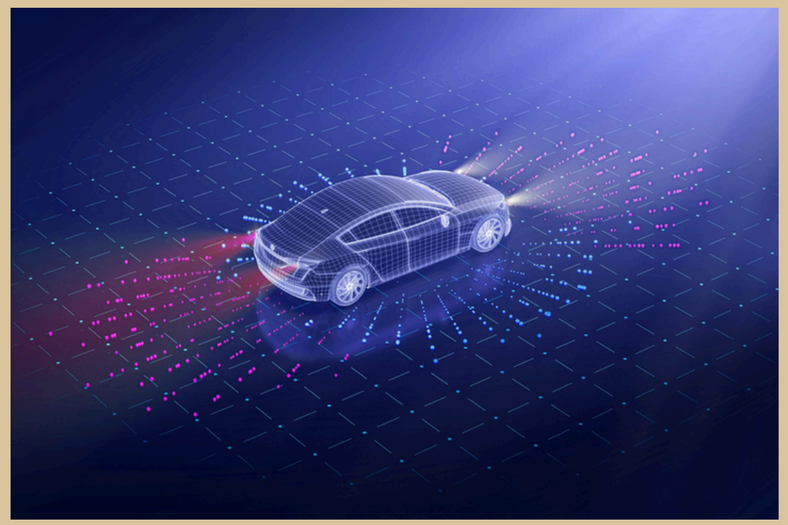
That's **AI & ML** at work!

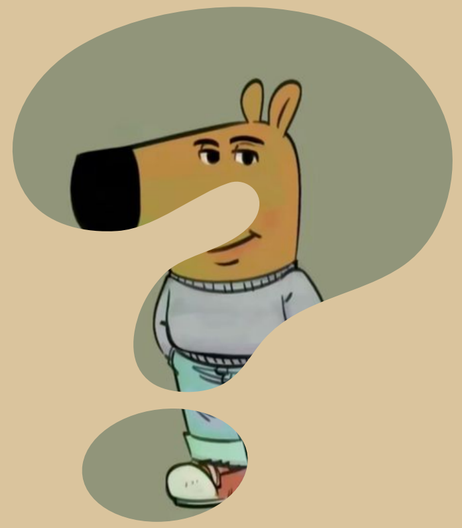


Fun Fact: The term "Artificial Intelligence" was coined in 1956 at the Dartmouth Conference!

WHAT'S THE FIRST THING THAT COMES TO YOUR MIND WHEN YOU HEAR AI?

(A ROBOT? IRON MAN'S J.A.R.V.I.S.? SELF-DRIVING CARS? LET'S HEAR FROM YOU GUYS!)





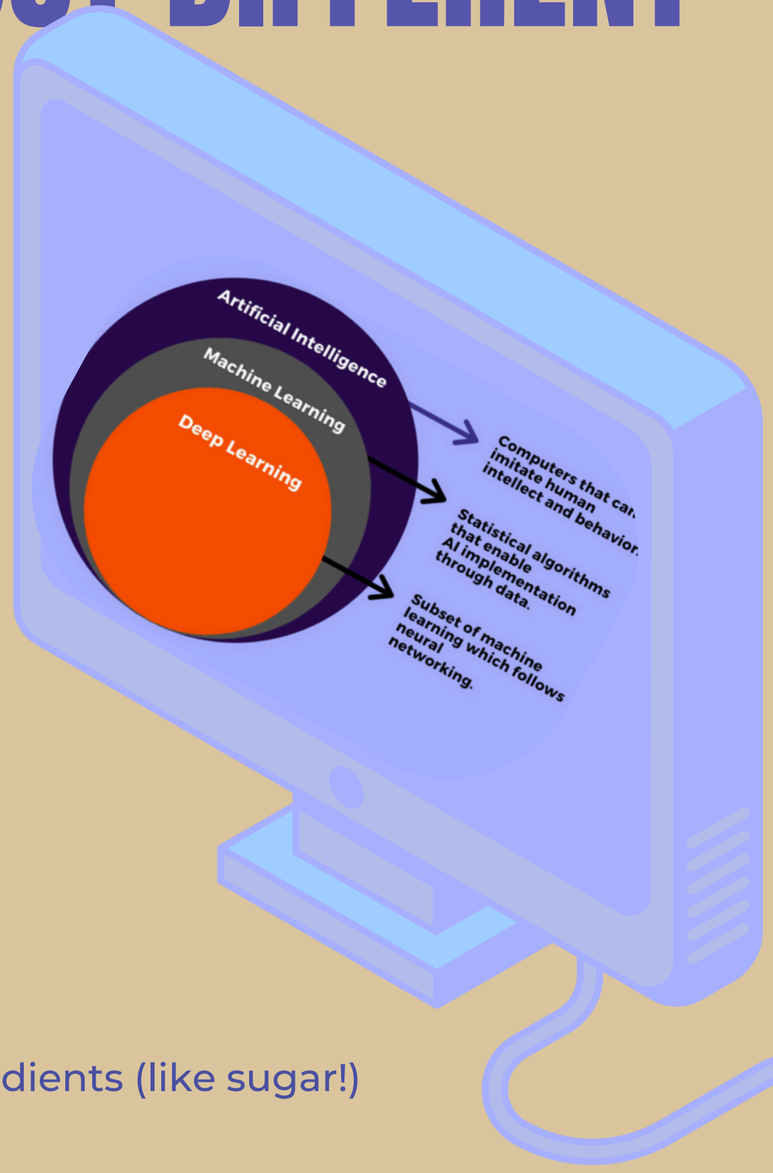
AI IS EVERYWHERE! GUESS WHERE?

- **SELF-DRIVING CARS 🚗:** *TESLA, WAYMO, AND CRUISE*
- **HEALTHCARE 🏥:** AI-POWERED DIAGNOSIS (E.G., *IBM WATSON*), DRUG DISCOVERY (E.G., *DEEPMIND'S ALPHAFOLD*)
- **FINANCE 💰:** FRAUD DETECTION IN CREDIT CARD TRANSACTIONS, ROBO-ADVISORS FOR INVESTMENT (E.G., *BETTERMENT, WEALTHFRONT*)
- **AGRICULTURE 🌾:** *PRECISION FARMING* WITH AI (E.G., AUTOMATED DRONES, SOIL ANALYSIS)
- **EDUCATION 📚:** ADAPTIVE LEARNING PLATFORMS (E.G., *DUOLINGO, KHAN ACADEMY AI TOOLS*)
- **MANUFACTURING 🏭:** PREDICTIVE MAINTENANCE USING IOT AND AI SENSORS

Fun Fact: 🤖 AI was once science fiction, now it's in your pocket *(Gemini, Siri & Alexa!)*

AI VS. ML: SAME SAME, BUT DIFFERENT

Artificial Intelligence (AI)	Machine Learning (ML)
Definition: AI is the science of making machines think & act like humans.	**Definition:** ML is a subset of AI that allows machines to learn from data without explicit programming.
Example: Chess-playing bot (AI logic & planning)	**Example:** Netflix recommendations (ML learns from your watch history)



Fun Fact: AI is the whole cake 🍰, and ML is just one of the ingredients (like sugar!)

DEEP LEARNING: AI'S SUPERPOWER! 🚀

🔍 EVER WONDERED...

HOW CAN AI RECOGNIZE FACES, TRANSLATE LANGUAGES, OR EVEN CREATE ART?

THAT'S DEEP LEARNING IN ACTION!

DEEP LEARNING: AI'S SUPERPOWER!

IF ML IS LIKE RIDING A BICYCLE, DEEP LEARNING IS LIKE DRIVING A CAR WITH AUTOPILOT!

DEEP LEARNING = MORE DATA, BETTER RESULTS!

📊 **Works best with HUGE datasets – More data = Smarter AI!**

🤖 **Learns complex patterns – No need for manual coding!**

📈 **Improves over time – More training = Better performance!**

Mind-Blowing Neural Network Examples:

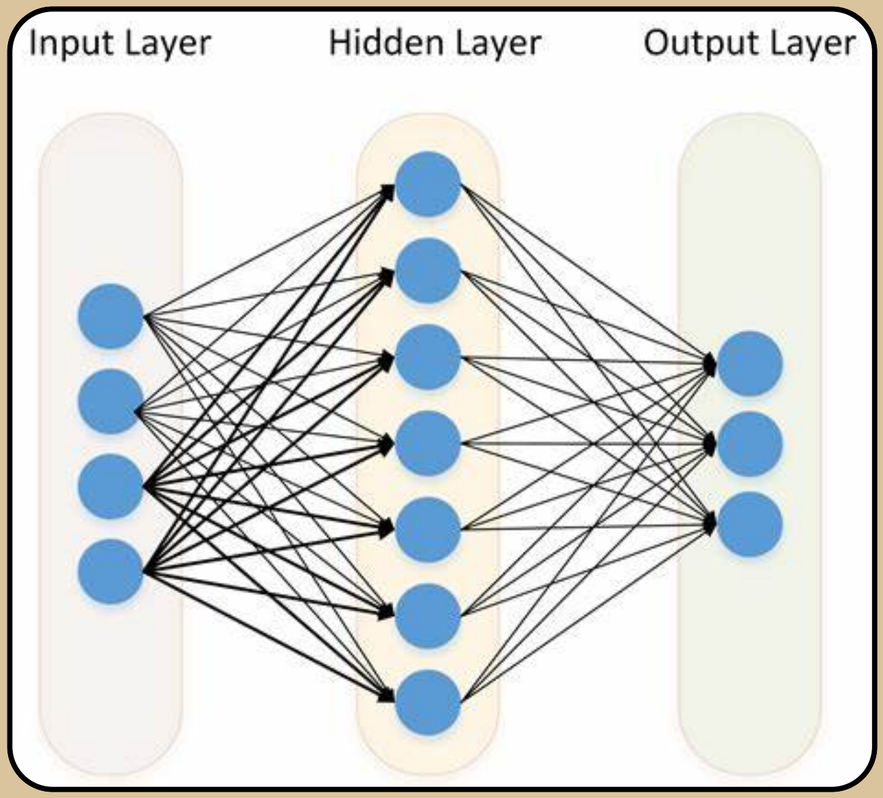
- 🌍 **Google Translate – Learns languages from millions of texts.**
- 🎮 **AI in Chess & Go – AlphaGo defeated world champions!**

Think of something AI has impressed you with. Share it! 🚀
Maybe ChatGPT? Or those AI-generated videos like this one? 😲

Fun Fact: Did you know? AI once defeated the world's best Dota 2 players using neural networks! 🤯

NEURAL NETWORKS: INSPIRED BY THE BRAIN!

Just like our **brain cells (neurons) process information**, **Artificial Neural Networks (ANNs) do the same in AI.**



How Do Neural Networks Work?

They process information layer by layer:

1. **Input Layer** → Takes raw data (images, text, numbers, etc.)
2. **Hidden Layers** → Learns patterns and makes sense of the data
3. **Output Layer** → Gives predictions (e.g., “Is this a cat or a dog?” 🐱🐶)

A neural network can have millions of neurons! The more layers it has, the "deeper" the learning.

Fun Fact: ReLu (Rectified Learning Unit) is one of most commonly used Activation Function in Neural Netowork. (More about this during the Hands-on Session)

PYTORCH: THE BACKBONE OF MODERN AI!

EVER WONDERED...

HOW DO AI MODELS LEARN TO CHAT, PREDICT, OR EVEN INVENT NEW THINGS?

THAT'S PYTORCH POWERING INNOVATION!

Fun Fact: PyTorch's name comes from Python + Torch, where Torch was an older deep learning framework! 🧠🔥

IETE STUDENT FORUM - AI & ML VERTICAL

PYTORCH: THE BACKBONE OF MODERN AI!

WHAT IS PYTORCH?

🚀 A playground for AI! A deep learning framework by Meta, built for research & production.

🐍 Pythonic & Fast – Combines flexibility with GPU-accelerated tensor computation

IMAGINE TEACHING A ROBOT TO WALK!

1️⃣ Build the neural network.
2️⃣ Train with data (e.g., 10,000 steps).
3️⃣ Test → Adjust → Repeat 🔄 – PyTorch handles gradients & optimization!

🔍 EXAMPLE: REAL-TIME OBJECT DETECTION IN SELF-DRIVING CARS.

Fun Fact: PyTorch's name comes from Python + Torch, where Torch was an older deep learning framework! 🧠🔥

IETE STUDENT FORUM - AI & ML VERTICAL

WHY PYTORCH? THE GAME-CHANGER FOR AI/ML

DYNAMIC COMPUTATION GRAPHS

⚡ Real-time model evolution – Debug & modify networks mid-training.

🚀 Perfect for research – Ideal for RNNs, GANs, and advanced architectures.

PYTHON-FIRST PHILOSOPHY

🐍 Clean, intuitive code – Feels native to Python.

🔗 Seamless integration – Works with NumPy, SciPy & more.

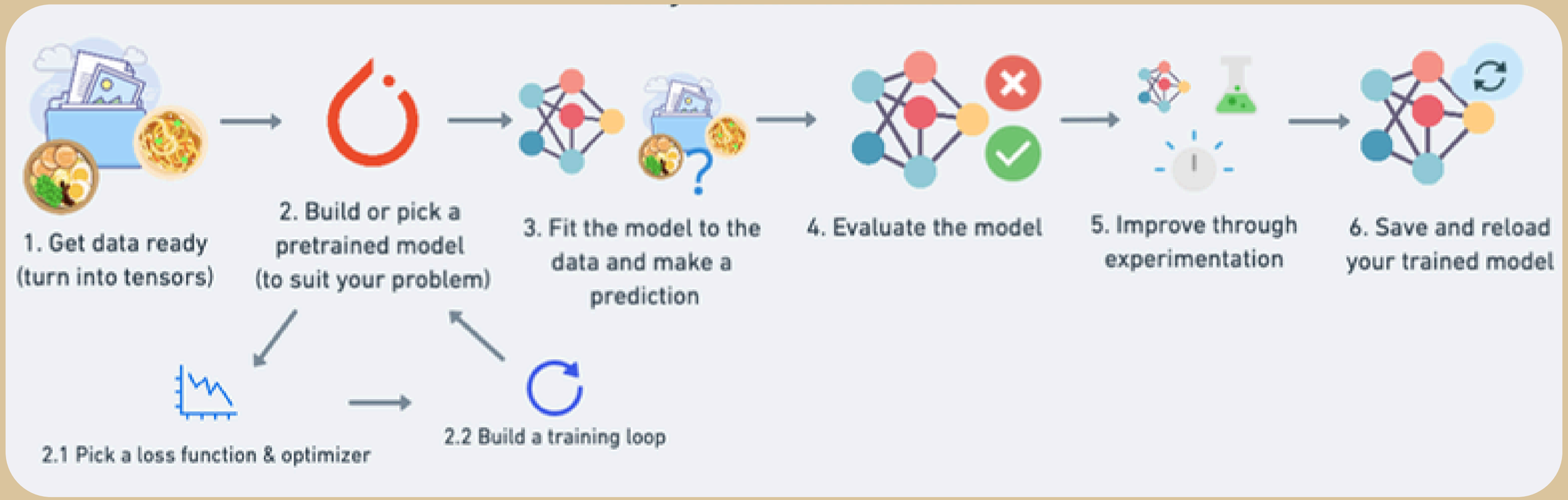
GPU ACCELERATION SIMPLIFIED

⚡One line: .to("cuda") → 10–100x faster training.

🔄 Effortless backprop with autograd.

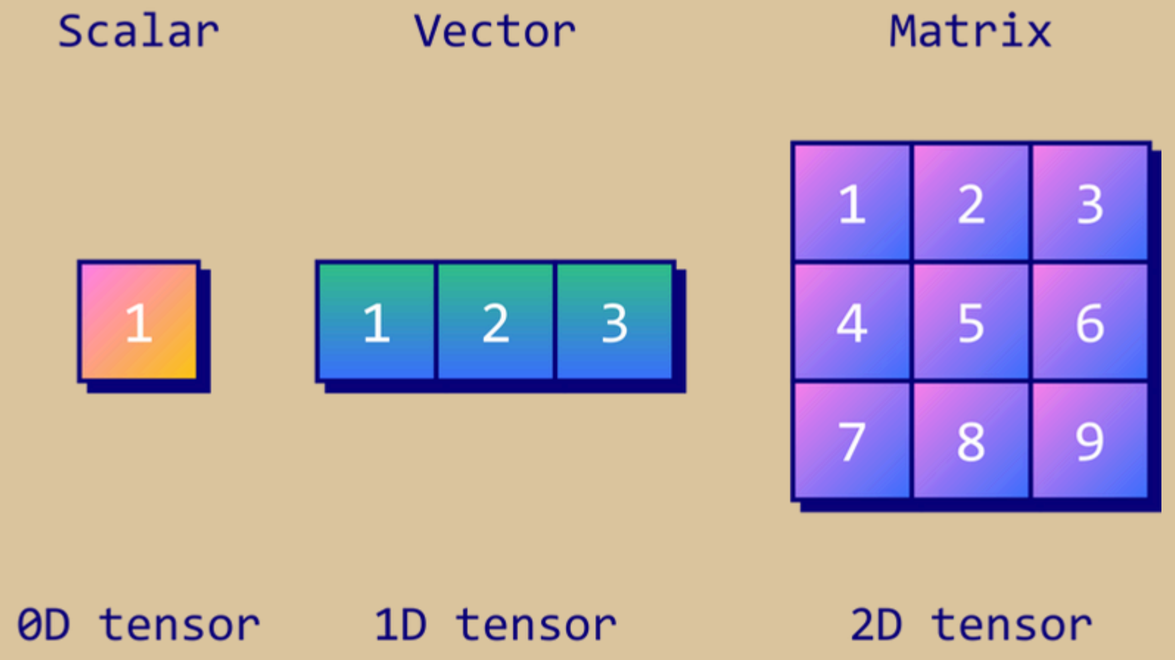
Fun Fact: PyTorch powers AI breakthroughs in healthcare, finance, and even space exploration!

FROM DATA TO INTELLIGENCE: THE PYTORCH AI/ML WORKFLOW



WHAT ARE TENSORS?

- **Tensors** are the **fundamental data structure** in AI and deep learning.
- They are **multi-dimensional arrays** used to store and manipulate numerical data efficiently.
- Tensors **generalize** scalars, vectors, and matrices to higher dimensions.



UNDERSTANDING TENSORS

The Foundation - Numbers	Vectors - Ordered Numbers	Matrices - Tables of Numbers	Tensors - Higher Dimensions
• At the lowest level, all data is represented as numbers. • AI processes these numbers to learn patterns and make decisions.	• A 1D array (list) of numbers. • Represents a point in space or features in ML.	• A 2D array with rows & columns. • Used for images, transformations, and equations.	• A multi-dimensional array extending vectors & matrices. • Used in AI for images, videos, and complex data.

WHY TENSORS?

✅ EFFICIENT DATA STORAGE

TENSORS STORE LARGE DATASETS EFFICIENTLY, OPTIMIZING MEMORY USAGE.

✅ FAST COMPUTATION

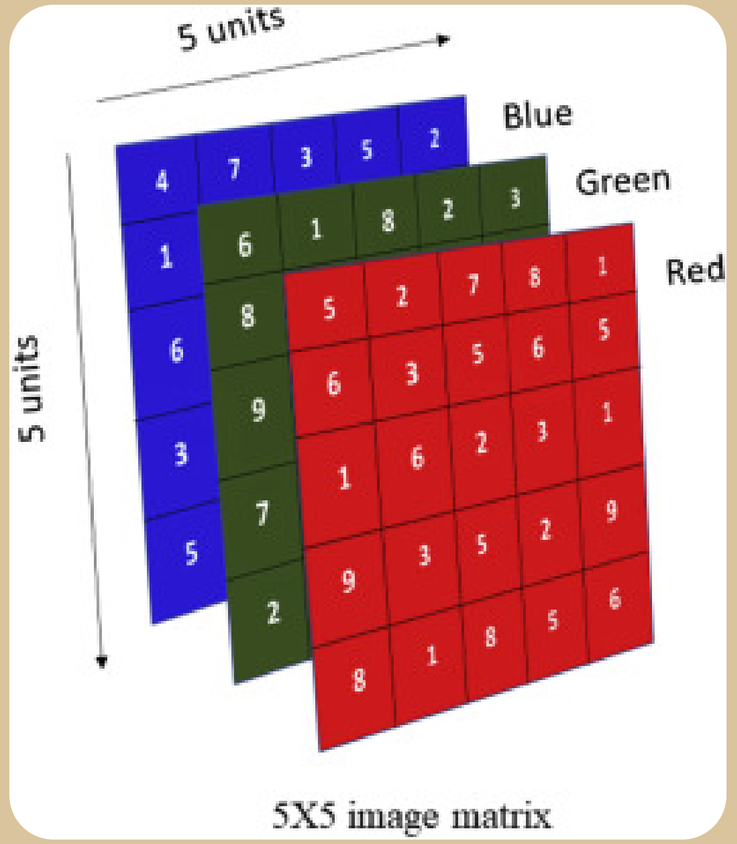
PYTORCH USES OPTIMIZED TENSOR OPERATIONS (LIKE MATRIX MULTIPLICATIONS) FOR HIGH-SPEED CALCULATIONS, ESPECIALLY ON GPUS.

✅ AUTOMATIC DIFFERENTIATION

TENSORS SUPPORT BACKPROPAGATION, ENABLING DEEP LEARNING MODELS TO LEARN EFFECTIVELY.

✅ REAL-WORLD EXAMPLE

A 5 × 5 PIXEL RGB IMAGE IS STORED AS A (5, 5, 3) TENSOR (HEIGHT × WIDTH × CHANNELS).



5X5 image matrix

TENSORS IN NEURAL NETWORKS

📥 Input Data	Images, text, and audio are converted into tensors. Example: A batch of 64 RGB images (256×256) is stored as a tensor of shape (64, 3, 256, 256).
🎛️ Weights	Neural network parameters stored as tensors. Example: A layer connecting 10 input neurons to 5 output neurons has a weight tensor of shape (5, 10).
➕ Biases	Bias terms adjust layer outputs, stored as tensors. Example: A layer with 5 neurons has a bias tensor of shape (5,).
🔄 Activations	The transformed outputs of layers after activation functions, also stored as tensors.

CREATING TENSORS IN PYTORCH

```
[1]  import torch
     # Creating a tensor from a python list
     tensor = torch.Tensor([[1, 2, 3], [4, 5, 6]])
     print((tensor))
```

```
     tensor([[1., 2., 3.],
             [4., 5., 6.]])
```

```
[2]  print(tensor.shape, tensor.dtype, tensor.device)
```

```
     torch.Size([2, 3]) torch.float32 cpu
```

- **TORCH.TENSOR()** - THE BASE CONSTRUCTOR
- CREATES A TENSOR FROM EXISTING DATA.
- CAN TAKE LISTS, NESTED LISTS, OR OTHER DATA STRUCTURES.

OTHER WAYS TO CREATE TENSORS IN PYTORCH

✅ **torch.zeros()** – Creates a tensor filled with zeros.
➡ **torch.zeros(2, 3)** → 2×3 tensor of zeros

✅ **torch.ones()** – Creates a tensor filled with ones.
➡ **torch.ones(3, 4)** → 3×4 tensor of ones

✅ **torch.rand()** – Random values from a uniform distribution (0 to 1).
➡ **torch.rand(5, 5)** → 5×5 tensor with random values

✅ **torch.randn()** – Random values from a normal distribution (mean 0, std 1).
➡ **torch.randn(4, 2)** → 4×2 tensor with normally distributed values

✅ **torch.arange()** – Creates a sequence of numbers.
➡ **torch.arange(0, 10, 2)** → [0, 2, 4, 6, 8]

CONVERTING BETWEEN NUMPY AND PYTORCH

🔢 WHY NUMPY?

✅ **NUMPY** IS A POPULAR PYTHON LIBRARY FOR NUMERICAL COMPUTING.

✅ BOTH **NUMPY** AND **PYTORCH** HANDLE ARRAYS/TENSORS EFFICIENTLY.

✅ **KEY SIMILARITY:** BOTH USE MULTI-DIMENSIONAL ARRAYS FOR COMPUTATIONS.

🚀 **BUT IN THIS WORKSHOP, WE'LL FOCUS ON PYTORCH, WHICH IS OPTIMIZED FOR DEEP LEARNING!**

CONVERTING BETWEEN NUMPY AND PYTORCH

🟢 NUMPY → PYTORCH

✅ USE TORCH.FROM_NUMPY() TO CREATE A TENSOR FROM A NUMPY ARRAY.

```
import numpy as np

numpy_array = np.array([[1, 2], [3, 4]])
torch_tensor = torch.from_numpy(numpy_array)

print(torch_tensor, type(torch_tensor))
```

```
tensor([[1, 2],
        [3, 4]]) <class 'torch.Tensor'>
```

🔵 PYTORCH → NUMPY

✅ USE .NUMPY() TO CONVERT A PYTORCH TENSOR BACK TO A NUMPY ARRAY.

```
torch_tensor = torch.tensor([[1, 2], [3, 4]])
numpy_array = torch_tensor.numpy()

print(numpy_array, type(numpy_array))
```

```
[[1 2]
 [3 4]] <class 'numpy.ndarray'>
```

TENSOR DATA TYPES (DTYPE)

🚀 WHY DATA TYPES MATTER?

✅ AFFECTS MEMORY USAGE AND COMPUTATION SPEED IN PYTORCH.

✅ CHOOSING THE RIGHT DTYPE IMPROVES EFFICIENCY IN DEEP LEARNING MODELS.

👨‍💻 COMMON DATA TYPES IN PYTORCH

- **TORCH.FLOAT32** → 32-BIT FLOATING-POINT (DEFAULT, GENERAL-PURPOSE).
- **TORCH.FLOAT16** → 16-BIT FLOATING-POINT (LOWER MEMORY, FASTER ON SOME GPUS).
- **TORCH.INT64** → 64-BIT INTEGER (USEFUL FOR INDEXING, LABELS).
- **TORCH.INT32** → 32-BIT INTEGER (EFFICIENT FOR LOWER-RANGE INTEGER VALUES).
- **TORCH.BOOL** → BOOLEAN TENSOR (TRUE/FALSE VALUES).

SPECIFYING DATATYPE IN PYTORCH

```
[4]  # Step 1: Create a tensor with dtype (float16)
     tensor = torch.tensor([[1.5, 2.2], [3.1, 4.5]], dtype = torch.float16)
     print(tensor.dtype, end = ":\n")  # Output: torch.float16
     print(tensor, end = "\n\n")

     # Step 2: Change the dtype to int32
     tensor_int32 = tensor.to(torch.int32)
     print(tensor_int32.dtype, end = ":\n")  # Output: torch.int32
     print(tensor_int32)
```

```
torch.float16:
tensor([[1.5000, 2.1992],
        [3.0996, 4.5000]], dtype=torch.float16)

torch.int32:
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
```

MOVING TENSORS - CPU VS. GPU

TWO MEMORY SPACES FOR TENSORS

✅ **CPU (CENTRAL PROCESSING UNIT) –** GOOD FOR GENERAL COMPUTATIONS.

✅ **GPU (GRAPHICS PROCESSING UNIT) –** OPTIMIZED FOR PARALLEL PROCESSING, MUCH FASTER FOR DL.

```
[7]  # Check if GPU is available (cuda is API for Nvidia GPU's)
     device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

     # Create a tensor on CPU
     tensor = torch.tensor([1.0, 2.0, 3.0])
     print("default device:\n\t" + str(tensor.device))

     # Move tensor to GPU (if available)
     tensor = tensor.to(device)
     print("GPU:\n\t" + str(tensor.device))  # Output: cuda or cpu
```

```
default device:
        cpu
GPU:
        cuda:0
```

NOTE: CUDA WORKS ONLY WITH NVIDIA GPUS

TENSOR MANIPULATIONS - RESHAPING

🔄 RESHAPING TENSORS

✅ **RESHAPE() –** CHANGES THE SHAPE WITHOUT MODIFYING DATA.

```
[9]  tensor = torch.arange(16)
     print(tensor)
     reshaped_tensor = tensor.reshape(4, 4)  # Reshapes into (4, 4)
     print(reshaped_tensor)
     flattened_tensor = tensor.reshape(1, -1) # Reshapes into (1, -1)
     print(flattened_tensor)
```

```
tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11],
        [12, 13, 14, 15]])
tensor([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]])
```

✅ **VIEW() –** SIMILAR TO RESHAPE(), BUT SHARES THE SAME MEMORY.

📌 **KEY NOTE:** VIEW() IS MORE MEMORY-EFFICIENT BUT AFFECTS THE ORIGINAL TENSOR! 🚀

TENSOR MANIPULATIONS - SLICING TENSORS

✂ SLICING TENSORS

✅ EXTRACT SPECIFIC PORTIONS OF A TENSOR.

```
[13] tensor = torch.arange(16).reshape(4, 4)  # 4x4 matrix
     print(tensor)

     slice_tensor = tensor[0:2, -2:]
     print("\ntaking the first two rows and last two columns:")
     print(slice_tensor)
```

```
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11],
        [12, 13, 14, 15]])

taking the first two rows and last two columns:
tensor([[2, 3],
        [6, 7]])
```

ELEMENT-WISE OPERATIONS

⚡ FAST COMPUTATIONS ON MATRICES

✅ ARITHMETIC OPERATIONS PERFORMED ELEMENT-BY-ELEMENT BETWEEN TENSORS OF THE SAME SHAPE.

```
[2]  import torch
     A = torch.tensor([[1, 2], [3, 4]])
     B = torch.tensor([[5, 6], [7, 8]])

     # Element-wise addition
     C = A + B  # [[6, 8], [10, 12]]
     print(C)
     # Element-wise multiplication
     D = A * B  # [[5, 12], [21, 32]]
     print(D)
```

```
tensor([[ 6,  8],
        [10, 12]])
tensor([[ 5, 12],
        [21, 32]])
```

```
[6]  # Element-wise subtraction
     E = A + B
     print(E)
     # Element-wise division
     F = A * B
     print(F)
     # Element-wise power
     G = A ** B
     print(G)
```

```
tensor([[ 6,  8],
        [10, 12]])
tensor([[ 5, 12],
        [21, 32]])
tensor([[    1,    64],
        [ 2187, 65536]])
```

BROADCASTING

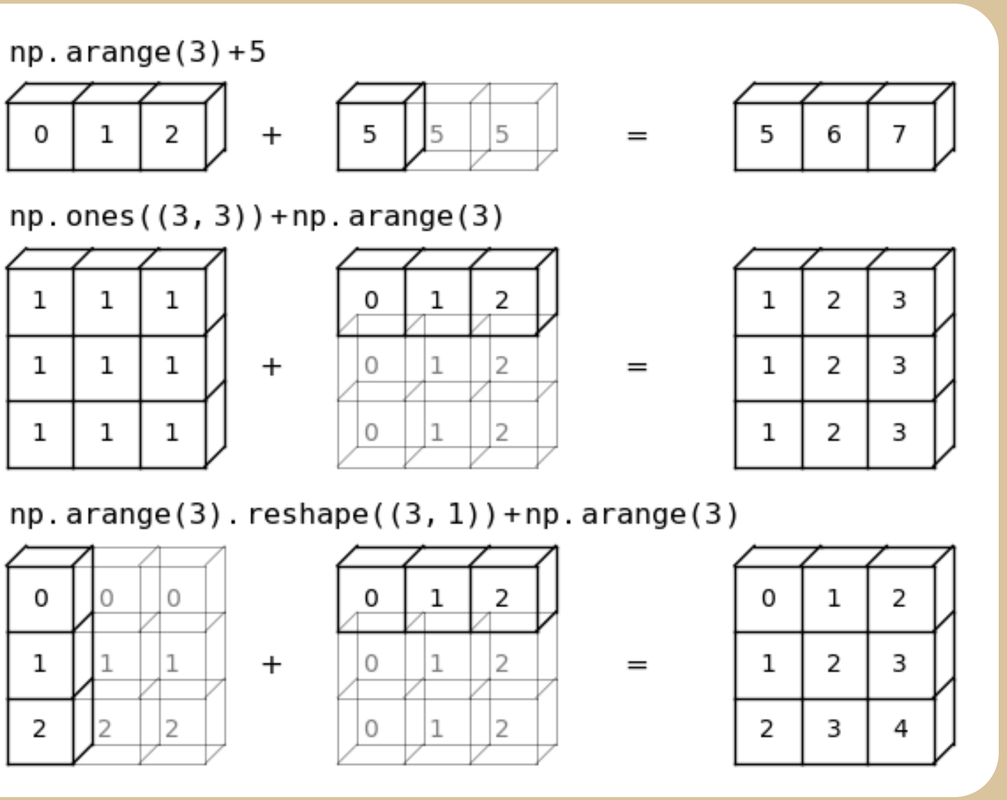
📡 MAKING TENSORS COMPATIBLE

✅ PYTORCH AUTOMATICALLY EXPANDS SMALLER TENSORS TO MATCH LARGER ONES WITHOUT COPYING DATA.

```
A = torch.tensor([[1, 2, 3], [4, 5, 6]])
B = torch.tensor([1, 2, 3])  # Shape: (3,)

# B is broadcasted to (2,3) before addition
C = A + B  # [[2, 4, 6], [5, 7, 9]]
print(C)
```

```
tensor([[2, 4, 6],
        [5, 7, 9]])
```



MATRIX MULTIPLICATION

💡 TORCH.MATMUL() – CORE OF NEURAL NETWORKS

✅ MATRIX MULTIPLICATION REQUIRES DIMENSION COMPATIBILITY.

```
A = torch.rand(3, 5)  # Shape: (3,5)
B = torch.rand(5, 4)  # Shape: (5,4)

C = torch.matmul(A, B)  # Shape: (3,4)
print(str(A.shape) + " * " + str(B.shape) + " = " + str(C.shape))
```

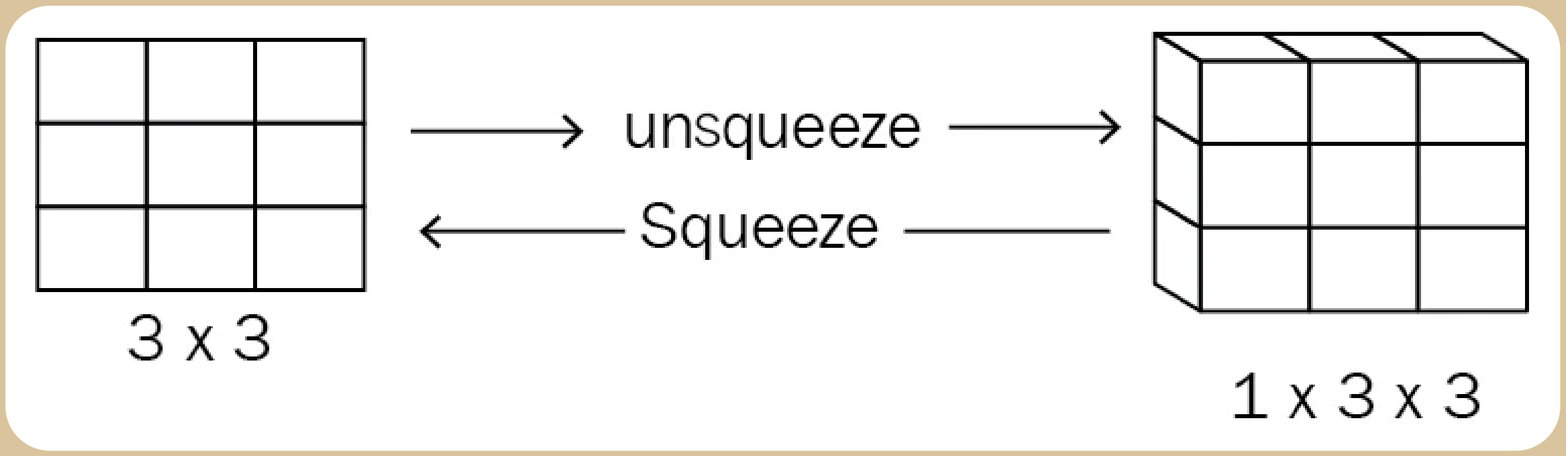
```
torch.Size([3, 5]) * torch.Size([5, 4]) = torch.Size([3, 4])
```

📌 **KEY RULE:** IF A IS (M, N) AND B IS (N, P), THE OUTPUT IS (M, P).

SQUEEZE & UNSQUEEZE - ADJUSTING TENSORS

🔄 WHY ADJUST DIMENSIONS?

- ✅ SOMETIMES, TENSORS HAVE UNNECESSARY DIMENSIONS (E.G., (1, 3, 1, 5)).
- ✅ SQUEEZE() REMOVES DIMENSIONS OF SIZE 1.
- ✅ UNSQUEEZE() ADDS A NEW DIMENSION AT A SPECIFIED POSITION.



SQUEEZE & UNSQUEEZE - ADJUSTING TENSORS

✂ **SQUEEZE():** REMOVING EXTRA DIMENSIONS

```
tensor = torch.rand(1, 3, 1, 5)  # Shape: (1,3,1,5)
print(tensor.shape)
squeezed = tensor.squeeze()  # Removes all dimensions of size 1
print(squeezed.shape)  # Output: torch.Size([3, 5])
```

```
torch.Size([1, 3, 1, 5])
torch.Size([3, 5])
```

📌 **BY DEFAULT, SQUEEZE() REMOVES ALL DIMENSIONS OF SIZE 1!**

SQUEEZE & UNSQUEEZE - ADJUSTING TENSORS

+ UNSQUEEZE(): ADDING A DIMENSION

```
tensor = torch.rand(3, 5)  # Shape: (3,5)
print(tensor.shape)
unsqueezed = tensor.unsqueeze(1)  # Adds a dimension at index 1
print(unsqueezed.shape)  # Output: torch.Size([3, 1, 5])
```

```
torch.Size([3, 5])
torch.Size([3, 1, 5])
```

📌 USEFUL FOR BATCH PROCESSING IN NEURAL NETWORKS! 🚀

CONCATENATION - COMBINING TENSORS

🔗 WHY CONCATENATE TENSORS?

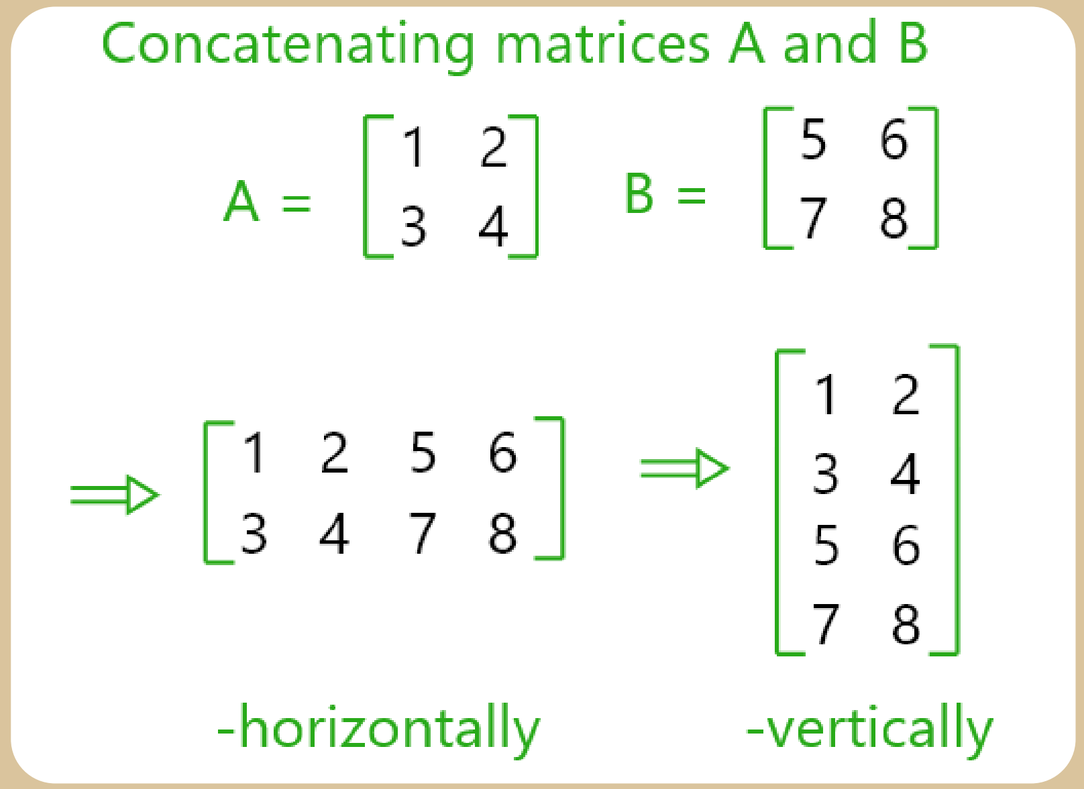
✅ MERGES MULTIPLE TENSORS ALONG A SPECIFIC DIMENSION.

✅ USED IN DATA PROCESSING, FEATURE FUSION, AND NEURAL NETWORKS.

📍 WHEN TO USE CONCATENATION?

✅ MERGING BATCHES OF DATA IN DEEP LEARNING.

✅ COMBINING FEATURES FROM DIFFERENT SOURCES.



CONCATENATION - COMBINING TENSORS

TORCH.CAT() – CONCATENATING TENSORS

✅ JOINS TENSORS ALONG A GIVEN DIMENSION (AXIS).

```
A = torch.tensor([[1, 2], [3, 4]])  # Shape: (2,2)
B = torch.tensor([[5, 6], [7, 8]])  # Shape: (2,2)

# Concatenating along rows (dim=0)
C = torch.cat((A, B), dim=0)
print(C)

# Concatenating along columns (dim=1)
D = torch.cat((A, B), dim=1)
print(D)
```

```
tensor([[1, 2],
        [3, 4],
        [5, 6],
        [7, 8]])
tensor([[1, 2, 5, 6],
        [3, 4, 7, 8]])
```

📌 DIM=0 → STACKS VERTICALLY | DIM=1 → STACKS HORIZONTALLY

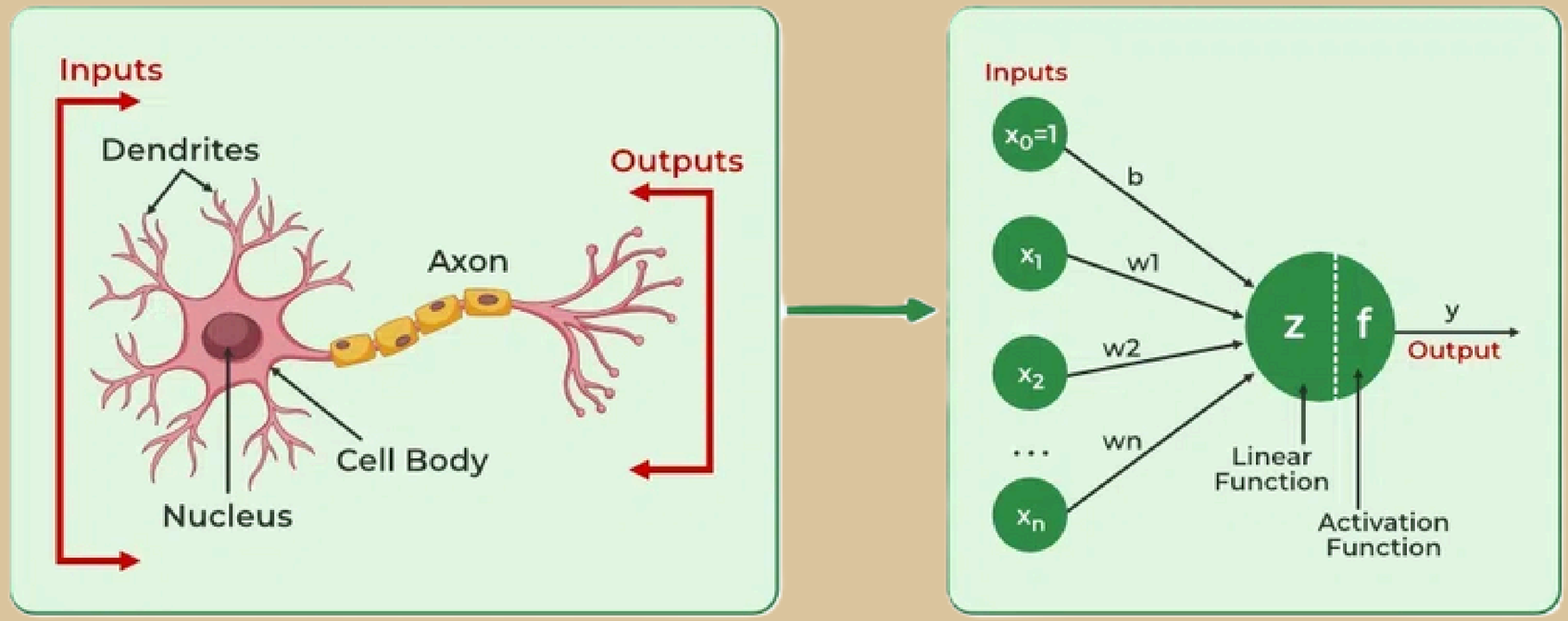
UNDERSTANDING NN - INTUITION

🧠 WHAT IS A NEURAL NETWORK?

✅ A NEURAL NETWORK IS A COMPUTATIONAL MODEL INSPIRED BY THE HUMAN BRAIN.

✅ IT CONSISTS OF LAYERS OF NEURONS THAT LEARN PATTERNS FROM DATA.

✅ EACH NEURON APPLIES A MATHEMATICAL OPERATION TO TRANSFORM INPUTS INTO OUTPUTS.

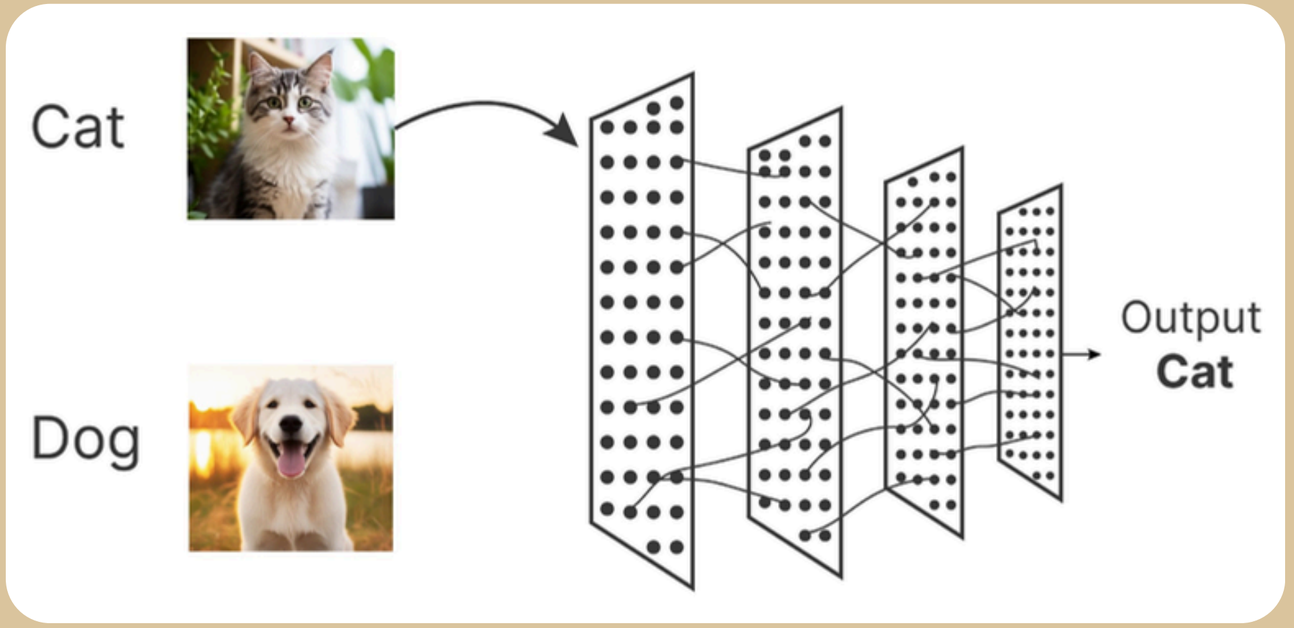


📌 **ANALOGY:** THINK OF A NEURAL NETWORK AS A STUDENT LEARNING – THE MORE PRACTICE (TRAINING), THE BETTER IT GETS!

NN AS A SERIES OF TRANSFORMATIONS

KEY COMPONENTS:

1. **INPUT LAYER –** RECEIVES RAW DATA (E.G., IMAGES, TEXT, AUDIO).
2. **HIDDEN LAYERS** – APPLY MATHEMATICAL TRANSFORMATIONS TO EXTRACT FEATURES.
3. **OUTPUT LAYER –** PRODUCES THE FINAL RESULT (E.G., CLASSIFICATION, PREDICTION).

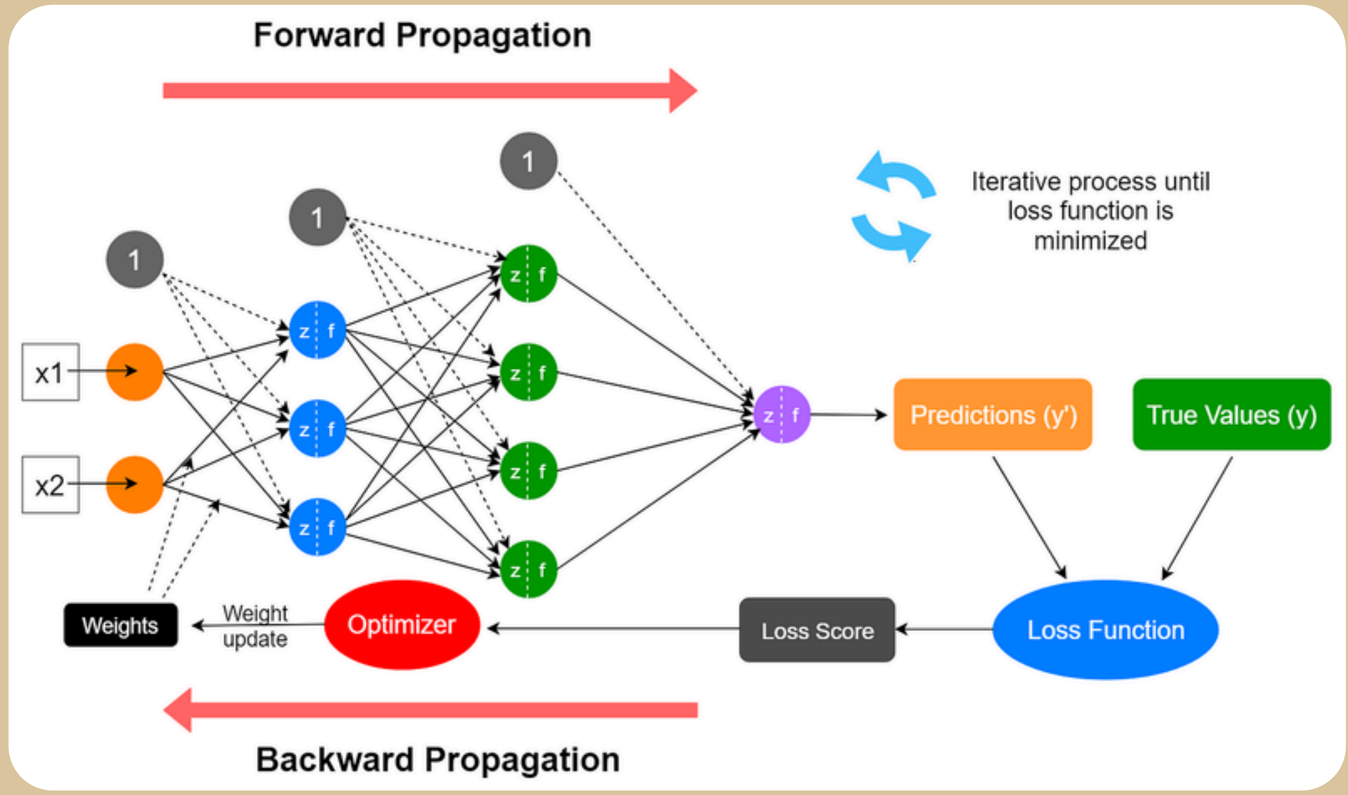


🚀 **NEURAL NETWORKS LEARN BY ADJUSTING THEIR INTERNAL PARAMETERS (WEIGHTS & BIASES) BASED ON ERRORS!**

HOW NEURAL NETWORKS LEARN

🔄 LEARNING = ADJUSTING WEIGHTS & BIASES

- ✅ WEIGHTS DETERMINE INPUT IMPORTANCE.
- ✅ BIASES SHIFT ACTIVATION THRESHOLDS.
- ✅ BACKPROPAGATION UPDATES THEM USING OPTIMIZATION (E.G., GRADIENT DESCENT).



📌 KEY INTUITION:

IMAGINE ADJUSTING THE VOLUME OF DIFFERENT MUSICAL INSTRUMENTS 🎵 TO GET THE PERFECT SOUND – THAT'S WHAT A NETWORK DOES WITH ITS WEIGHTS & BIASES!

CAN AI BE BIASED?

🤖 AI learns from data, but if the data has hidden biases, AI can make unfair decisions!

Real-Life Example:

🏛️ Hiring AI Favoring Certain Resumes – A job recruitment AI once prioritized certain keywords, unintentionally favoring some candidates. Why? It learned from past biased hiring data!

CAN AI BE BIASED?

Why Does This Happen?

- **Data Bias** – Unbalanced training data leads to unfair AI decisions.
- **Algorithmic Bias** – AI design may unintentionally favor certain outcomes.
- **Human Influence** – AI mirrors past biased decisions.

The Solution? Responsible AI! 🌍

✅ **Fairness** – Train AI with diverse, balanced data.
✅ **Transparency** – AI decisions should be clear & explainable.
✅ **Accountability** – Humans must monitor & correct AI mistakes.

Fun Fact: What if AI made the rules for a fair society? Would it do a better job than humans?

QUIZZEEE TIMEE!!

WHAT IS A TENSOR IN PYTORCH?

A) A neural network architecture

B) A multi-dimensional array

C) A gradient calculation method

D) A model optimization technique

WHAT IS A TENSOR IN PYTORCH?

A) A neural network architecture

B) A multi-dimensional array

C) A gradient calculation method

D) A model optimization technique

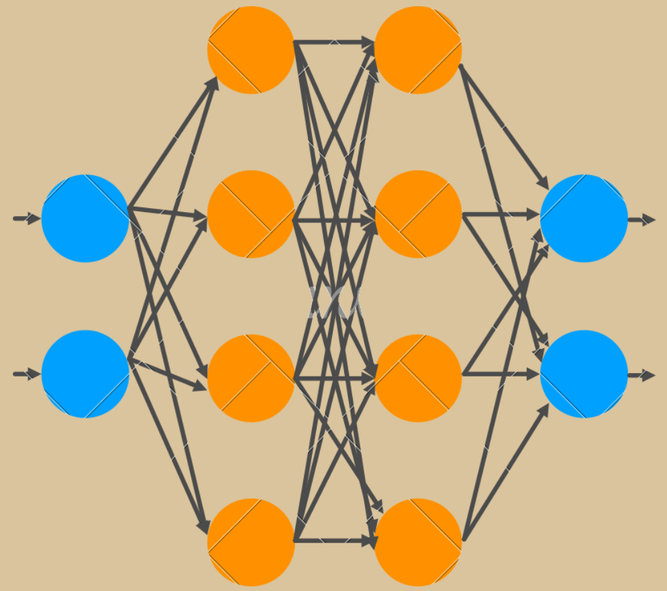
WHICH FUNCTION IS COMMONLY USED AS AN ACTIVATION FUNCTION IN NEURAL NETWORKS?

A) CONCAT

B) RELU

C) SORT

D) MEAN



WHICH FUNCTION IS COMMONLY USED AS AN ACTIVATION FUNCTION IN NEURAL NETWORKS?

A) CONCAT

B) RELU

C) SORT

D) MEAN

WHAT IS THE PURPOSE OF THE OPTIMIZER IN A NEURAL NETWORK TRAINING PROCESS?

A) To prepare the input data

B) To update the model parameters based on gradients

C) To evaluate model performance

D) To structure the neural network layers

WHAT IS THE PURPOSE OF THE OPTIMIZER IN A NEURAL NETWORK TRAINING PROCESS?

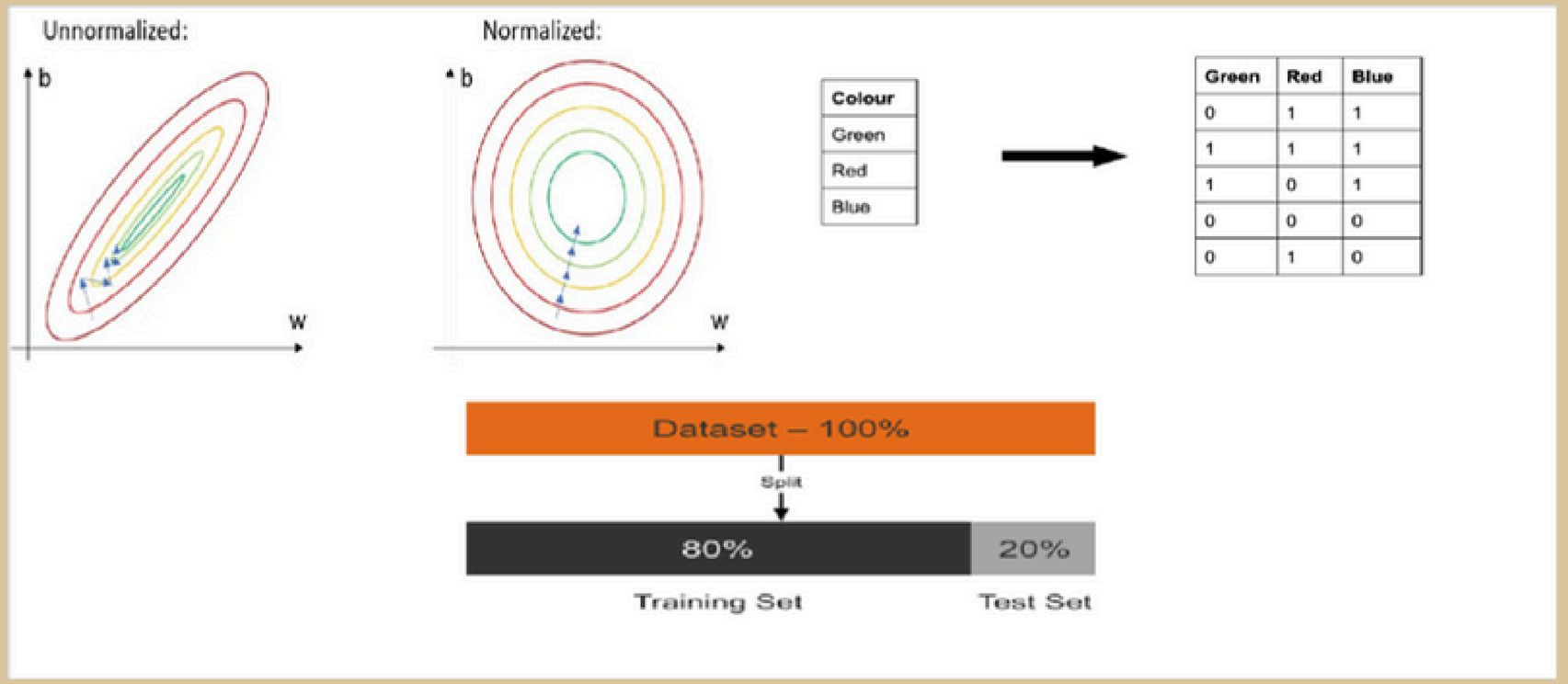
A) To prepare the input data

B) To update the model parameters based on gradients

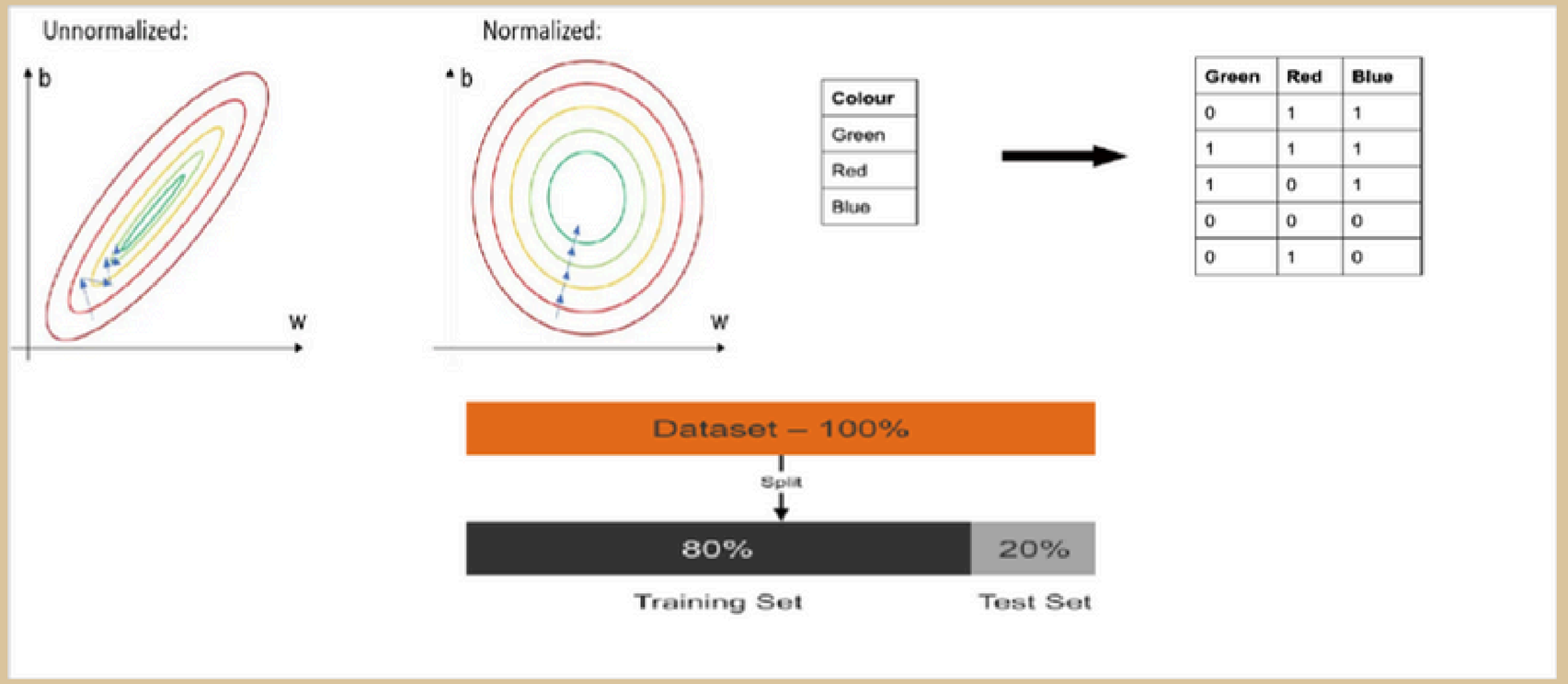
C) To evaluate model performance

D) To structure the neural network layers

WHAT'S THIS IMAGE IS ABOUT?



WHAT'S THIS IMAGE IS ABOUT?



Data Processing

UNSCRAMBLE THE FOLLOWING MACHINE LEARNING TERMS:

Vursepised Lraneing

SLSO UNNCTIOF

UNSCRAMBLE THE FOLLOWING MACHINE LEARNING TERMS:

Vursepised Lraneing

Supervised Learning

SLSO UNNCTIOF

UNSCRAMBLE THE FOLLOWING MACHINE LEARNING TERMS:

Vursepised Lraneing

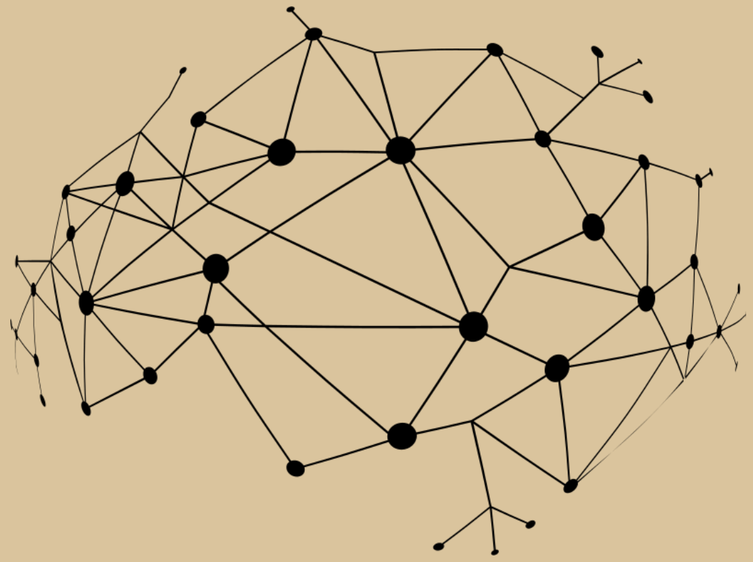
Supervised Learning

SLSO UNNCTIOF

Loss Function

ML PICTIONARY

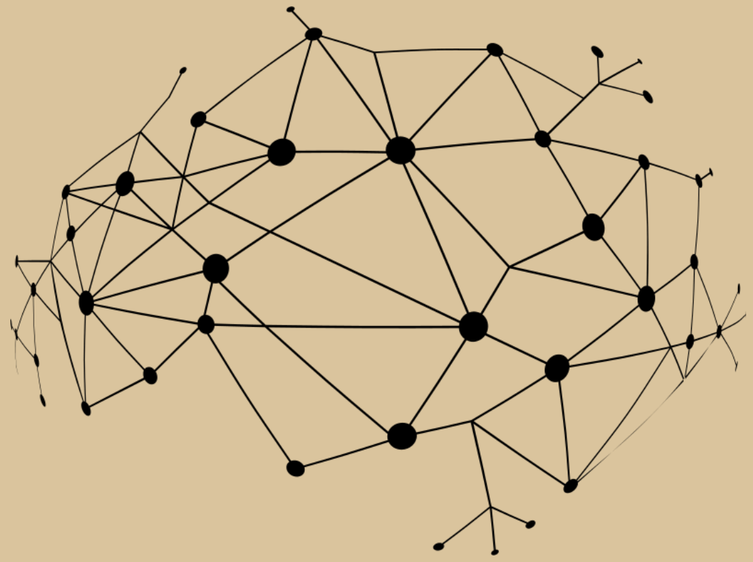
+



ML PICTIONARY

(BRAIN)

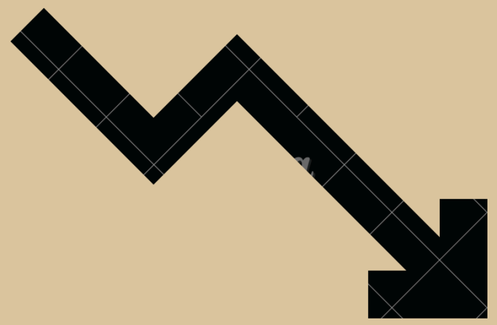
+



(NETWORK)

NEURAL NETWORK

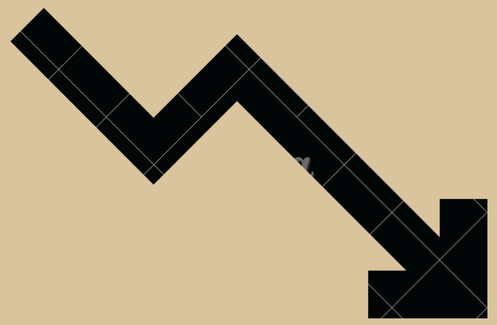
ML PICTIONARY



+

$$f(x)$$

ML PICTIONARY



+

$f(x)$

(LOSS) (FUNCTION)

LOSS FUNCTION

IETE STUDENT FORUM - AI & ML VERTICAL

TO DOWNLOAD THE PRESENTATION....

SCAN THISSSSS....

OUR TEAM

Sai Nivedh

AIE - II yr

Dharsini Sri

AIE - II yr

Baranidharan

AIE - II yr

OUR TEAM

Akhilesh M

AIE - II yr

Saran

AIE - III yr

Amritha

AIE - III yr

IETE STUDENT FORUM - AI & ML VERTICAL

OUR TEAM

Keerthivasan

AIE - II yr

Ashwin V A

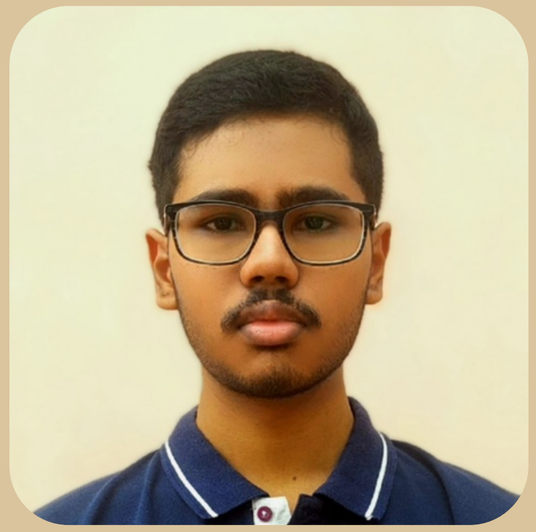
Jayadev D

CSE - II yr

Nithish

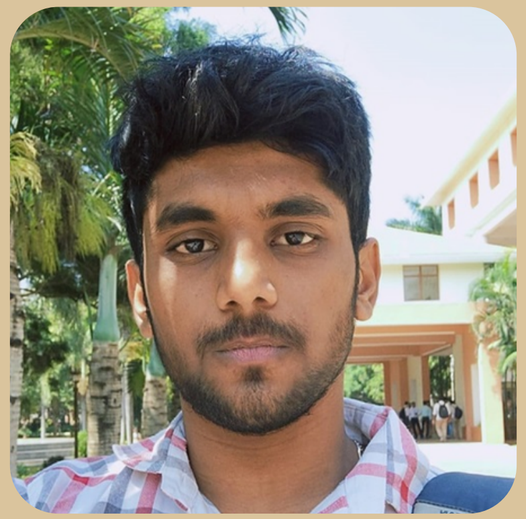
AIE - III yr

OUR TEAM



Suriya K P

AIE - III yr



L S Thosi Babu

AIE - III yr

@iete-amrita-sf

@IETEAmritaChapter

@IETE_AMRITA

BUILDING A SIMPLE NEURAL NETWORK

HTTPS://GITHUB.COM/IETEAMRITACHAPTER/TORCH-IT-UP

IETE STUDENT FORUM - AI & ML VERTICAL

FEEDBACK FORM

THANK YOU!