

Agenda

Introduction to Deep Learning

What is Deep Learning?

Basics of neural networks

Introduction to Deep learning frameworks

Agenda

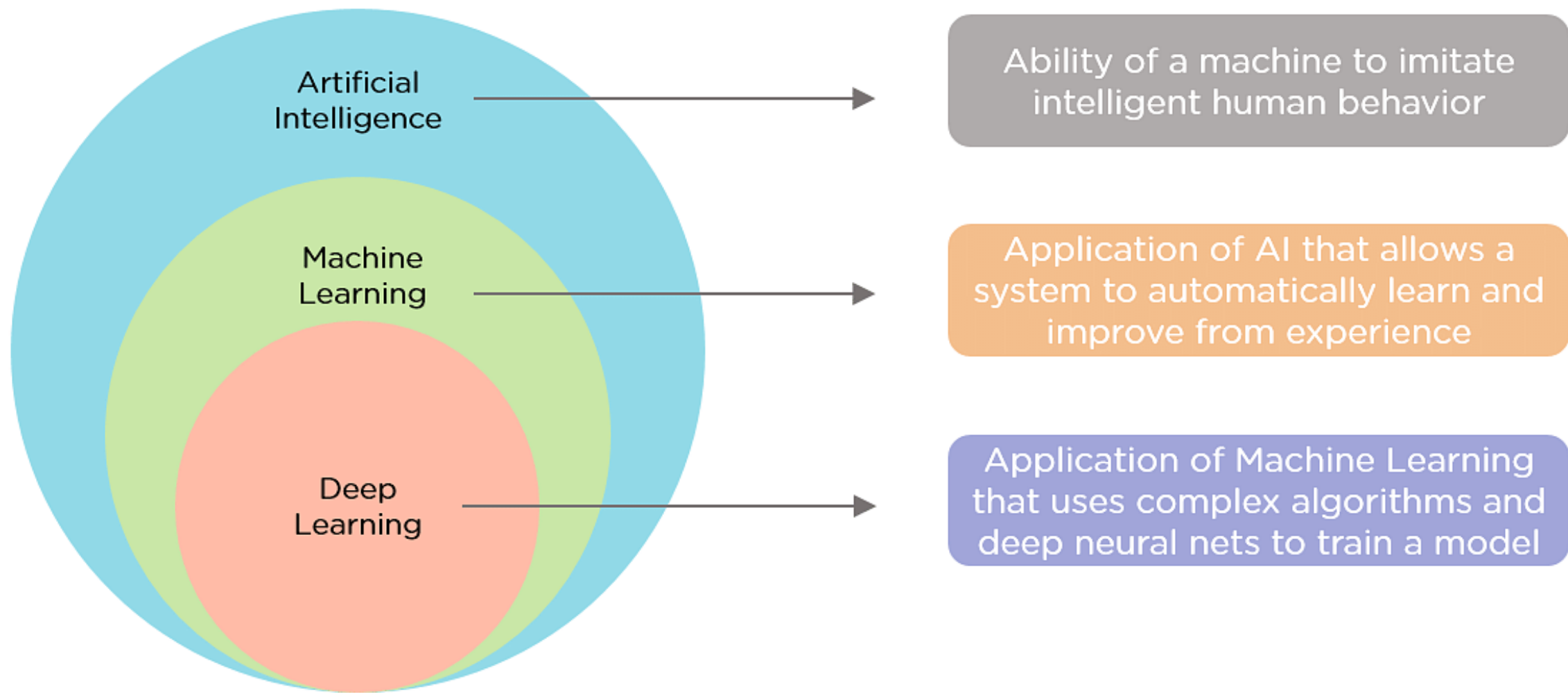
TensorFlow Introduction

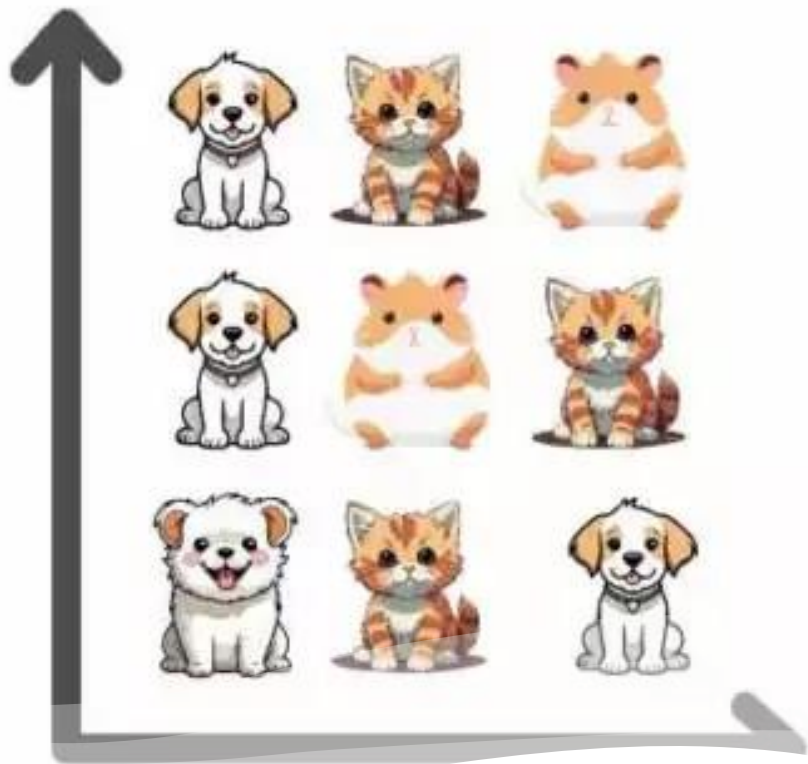
Tensorflow Python Example

Keras Introduction

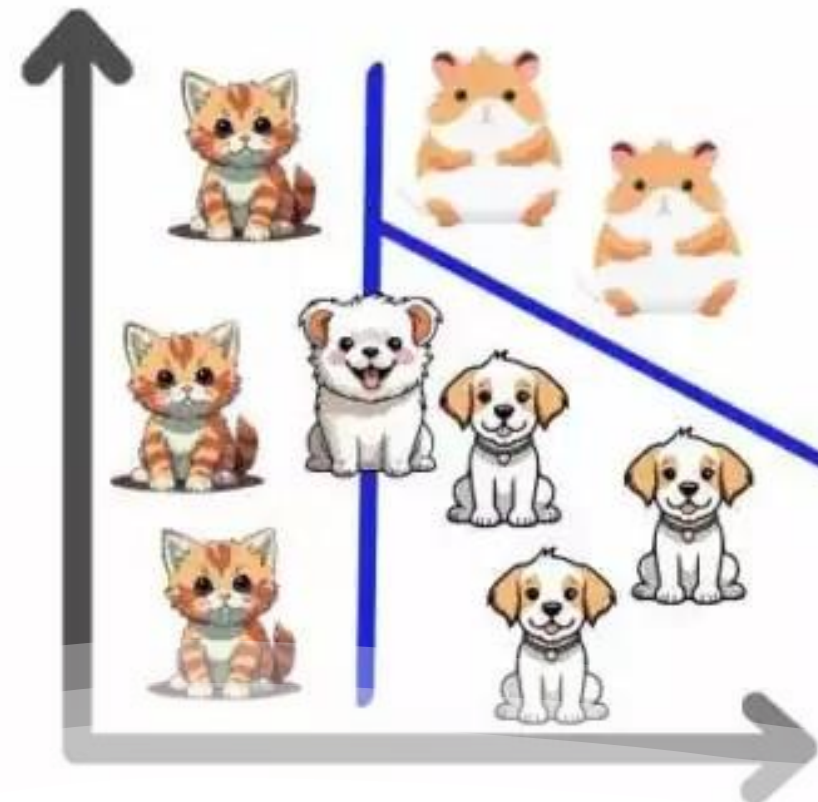
Keras Python Example

Summary and Conclusion





Representation
Learning



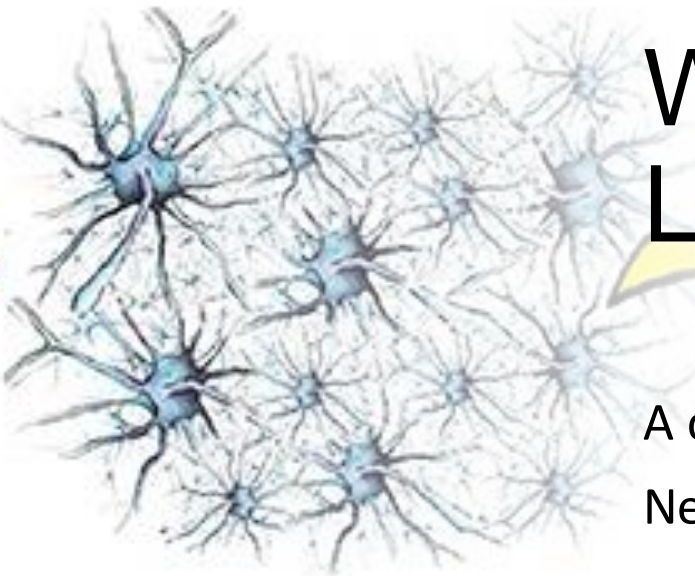
What is Deep Learning?

- Subset of machine learning
- Based on Artificial Neural Networks
- with representation learning.

brain



neural network



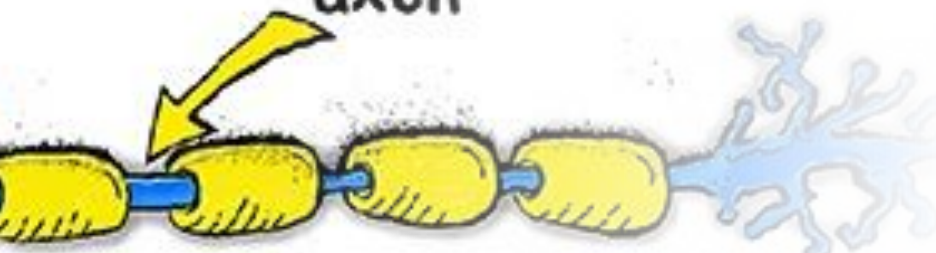
What is Deep Learning?

A computer software Mimics
Network of Neurons in a brain.

nucleus



axon



myelin sheath

dendrites



What is Deep Learning?



For Data Scientists



Deep learning is indispensable



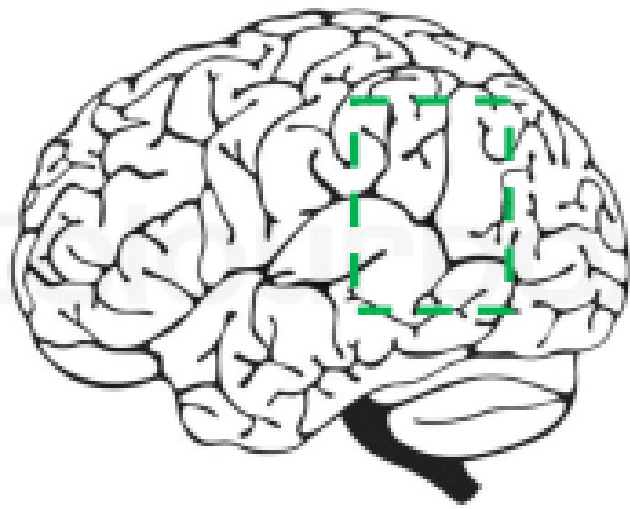
Facilitates and accelerates



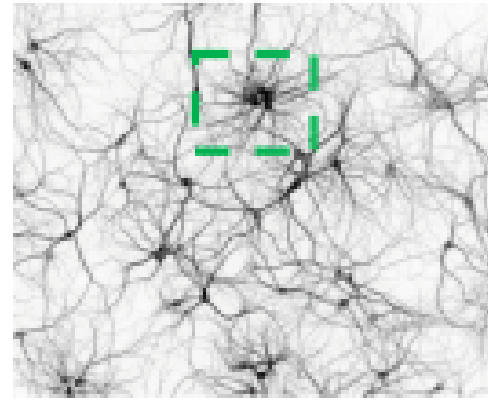
Collecting, Analyzing



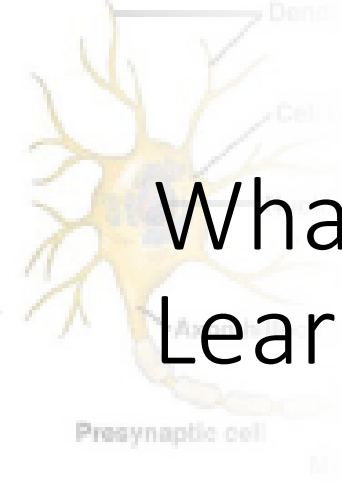
Processing of large amounts of data.



(a) Brain

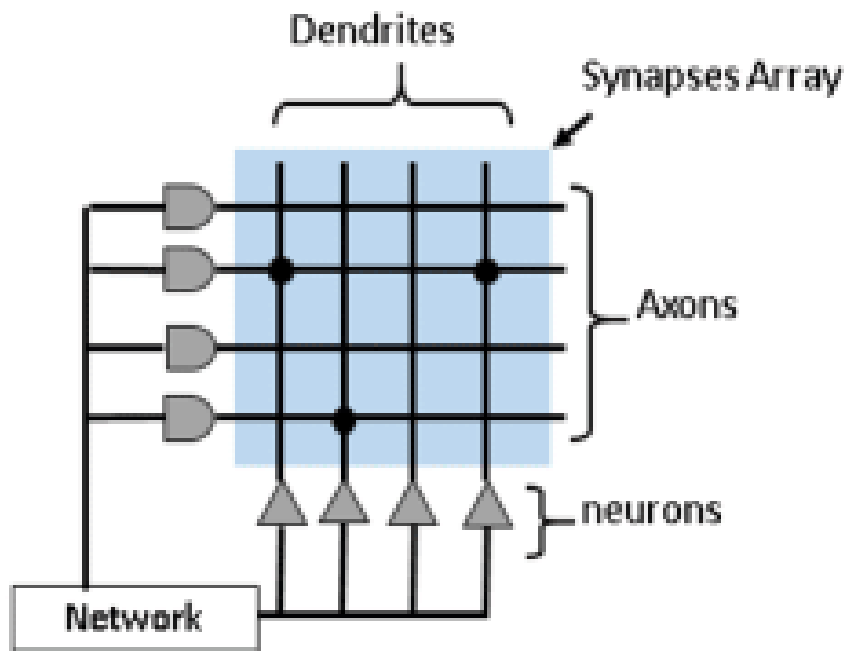


(b) Neuron network

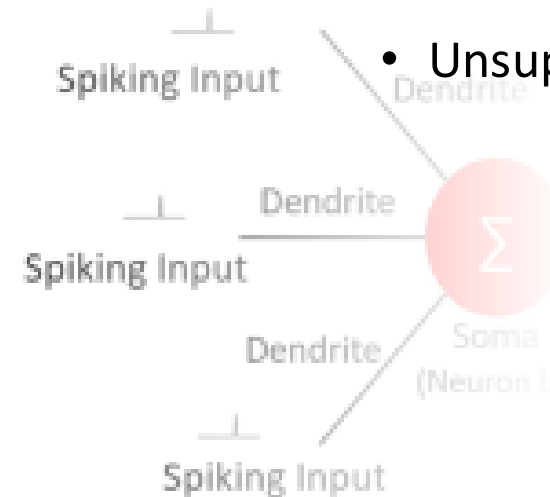


What is Deep Learning?

- Makes use of Deep Neural Networks.
- Learning can be supervised,
- Semi-supervised or
- Unsupervised.



(e) Neuron Network



(d) ?

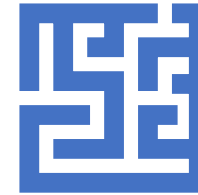
What is Deep Learning?



A subset of machine
learning

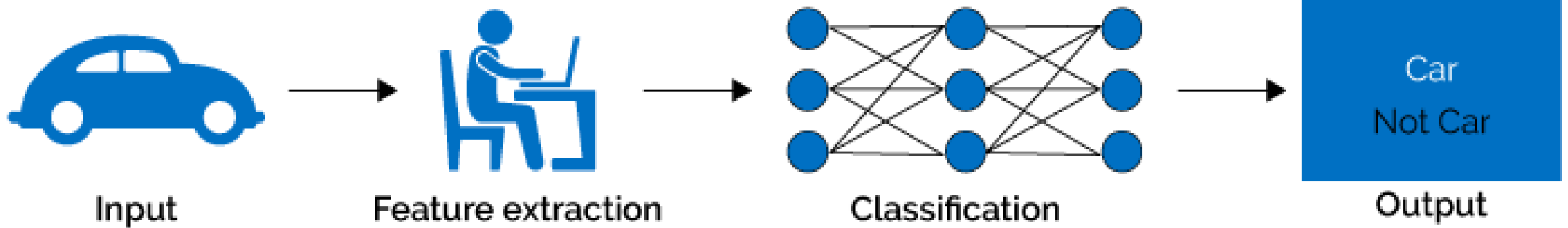


Use of artificial neural
networks

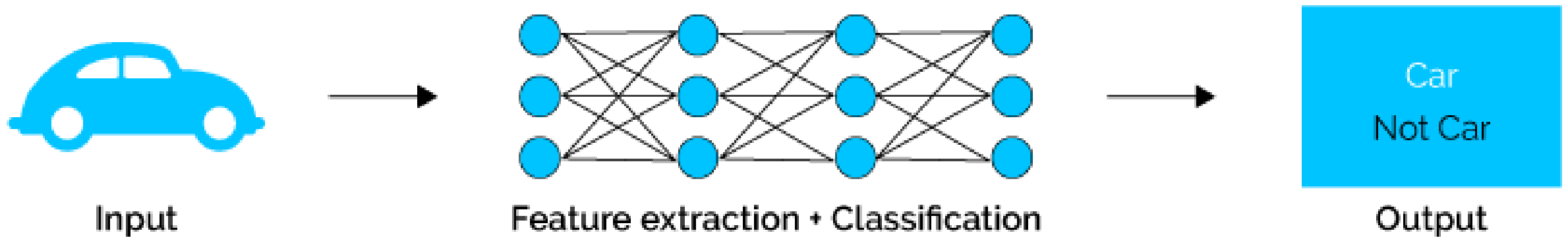


to model and solve
complex problems.

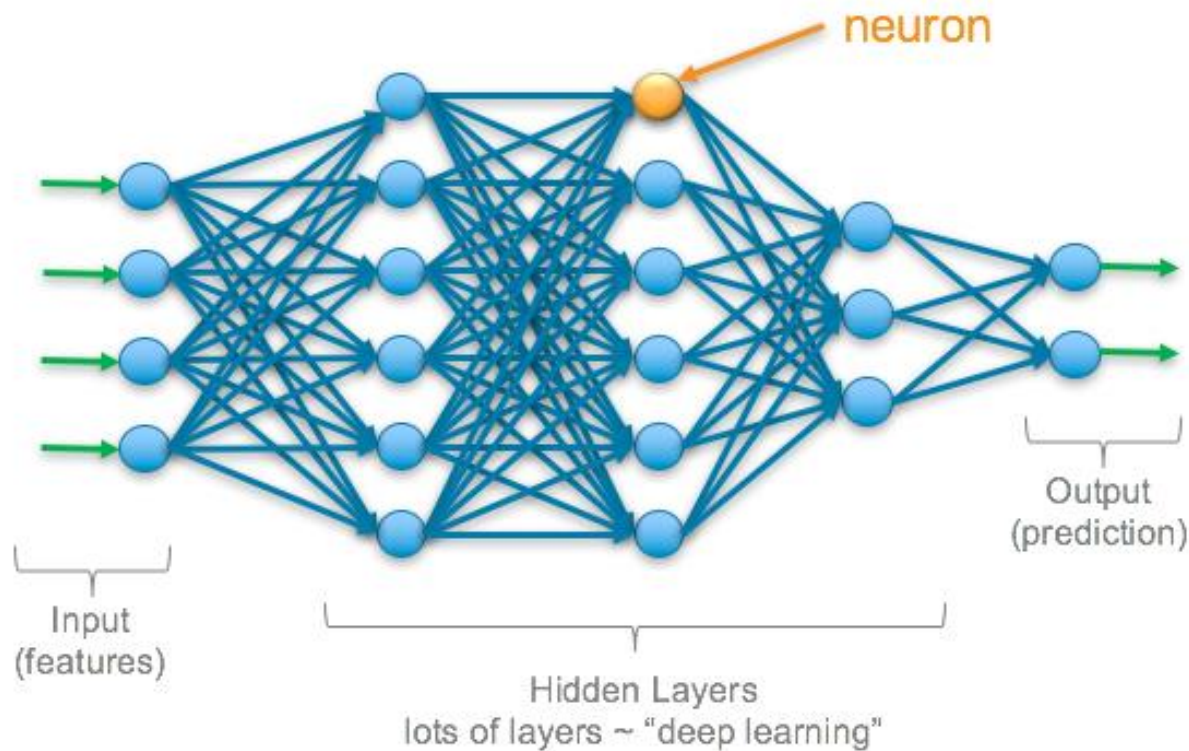
Machine Learning



Deep Learning



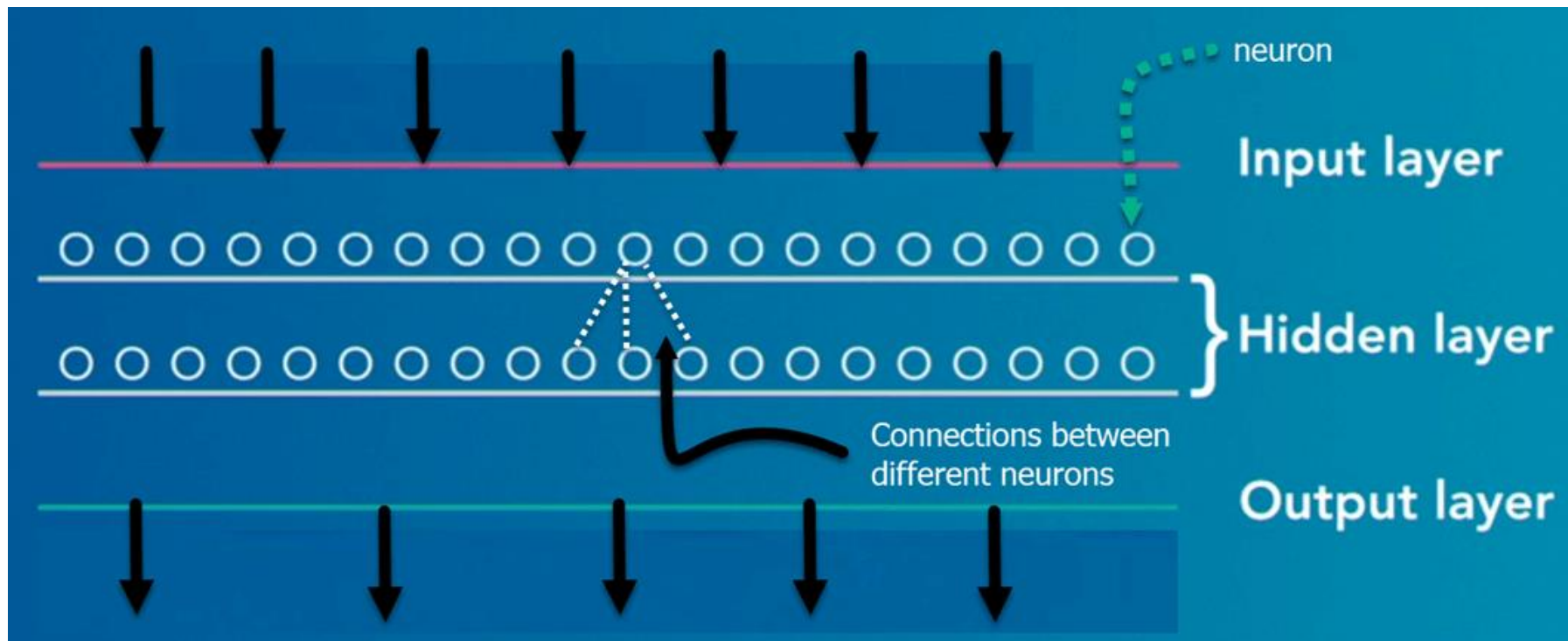
What is Deep Learning?



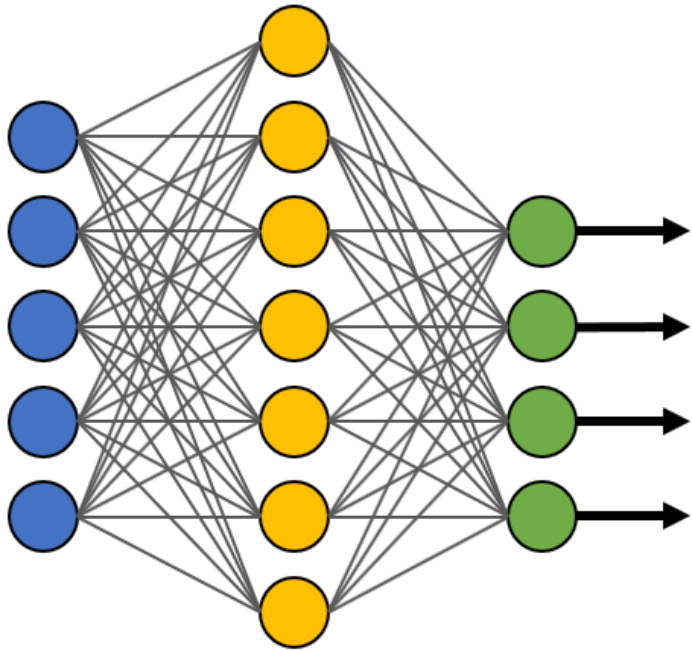
- DL Algorithms are constructed with connected layers.
- First Layer / Input Layer
- Last layer / Output Layer
- Middle Layer / Hidden Layers.

What is Deep Learning?

- The word deep means
- the network join neurons in
- more than two layers.

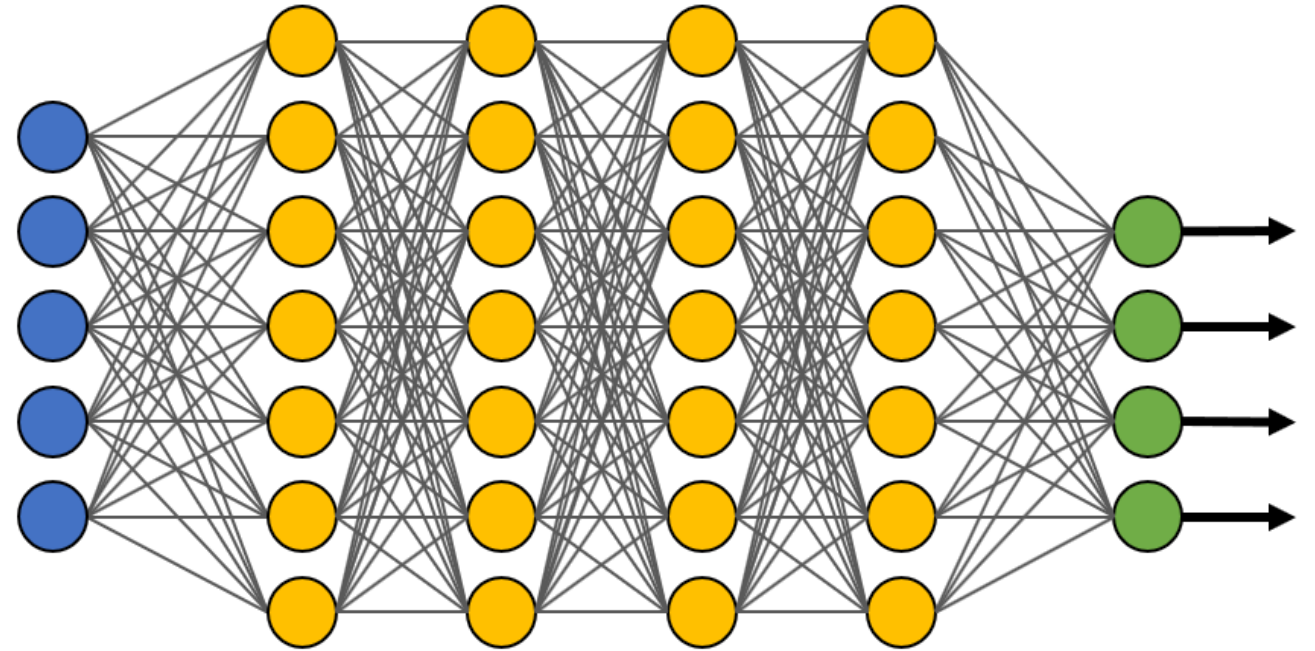


Simple Neural Network



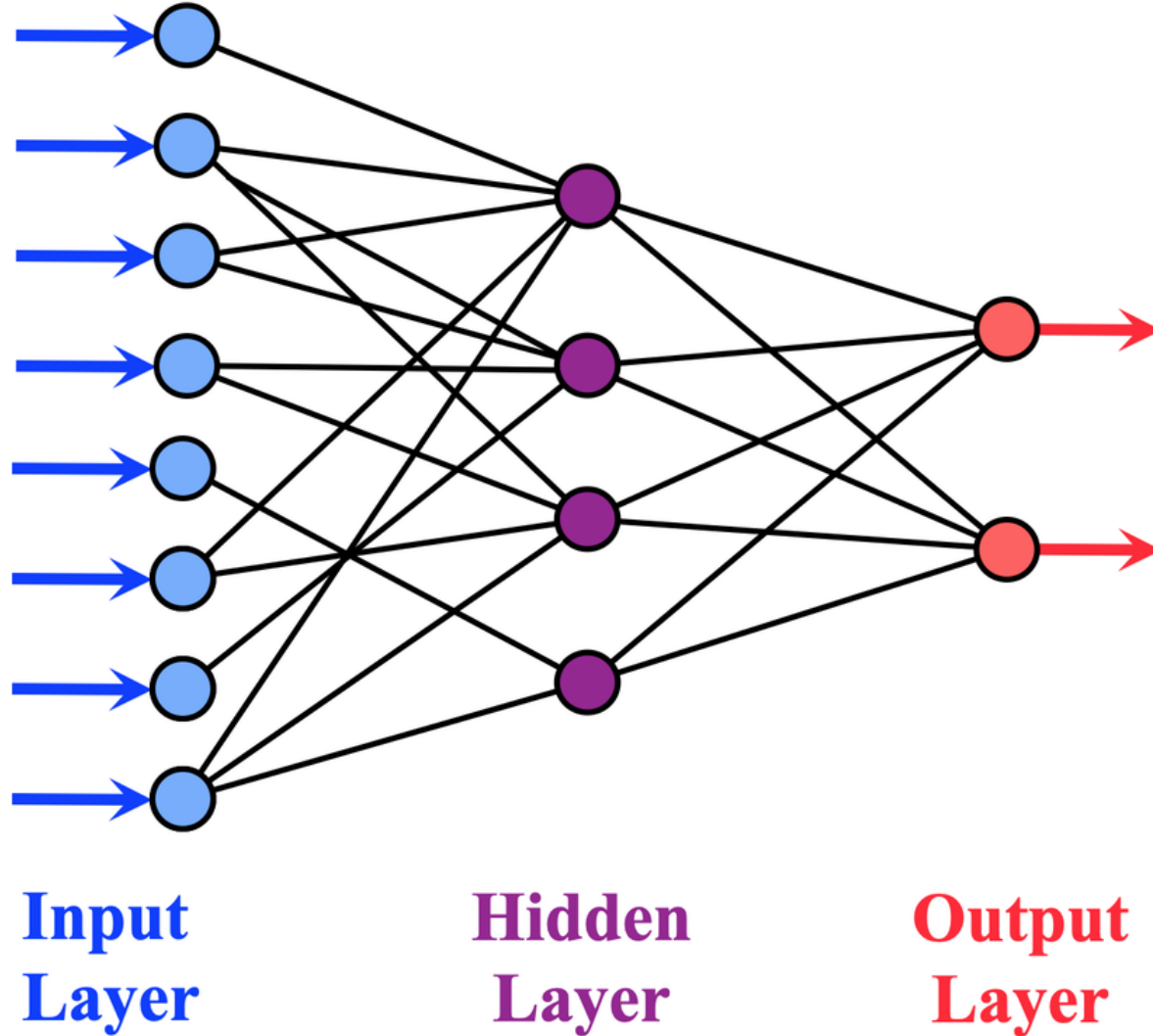
● Input Layer

Deep Learning Neural Network



● Hidden Layer

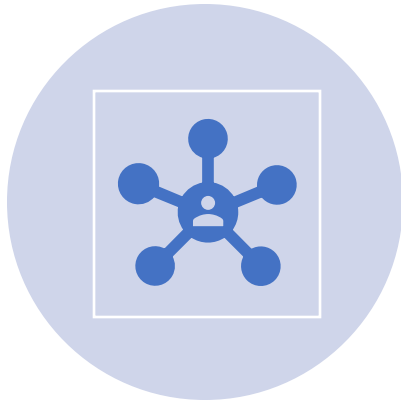
● Output Layer



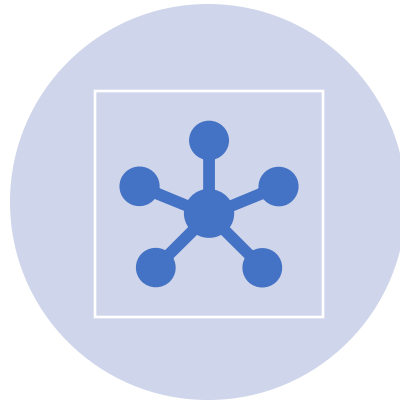
What is Deep Learning?

- NN Consumes
- Large amounts of input data
- Operates them through
- Multiple layers

What is Deep Learning?



THE NETWORK CAN
LEARN

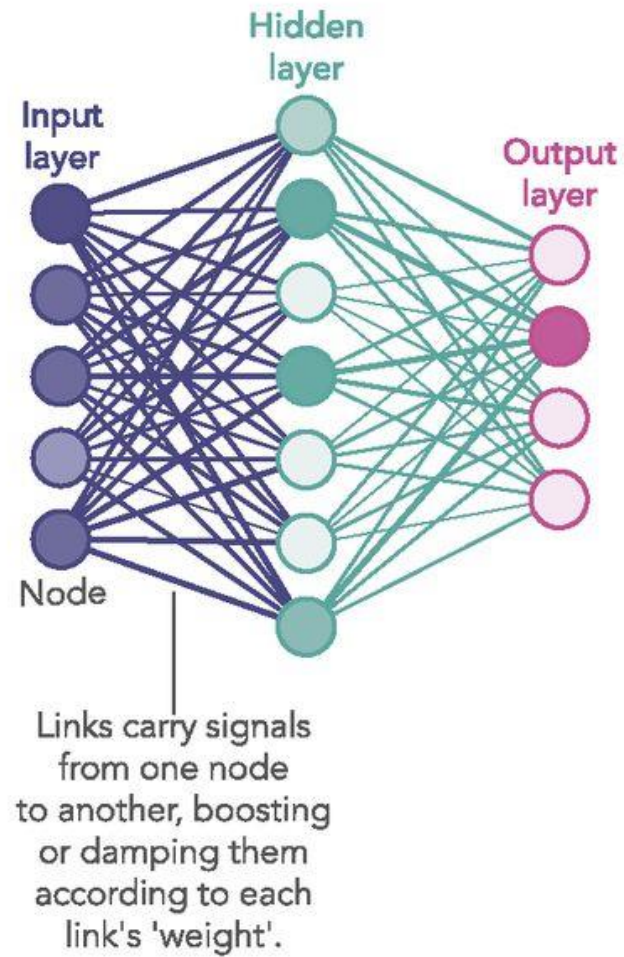


INCREASINGLY COMPLEX
FEATURES

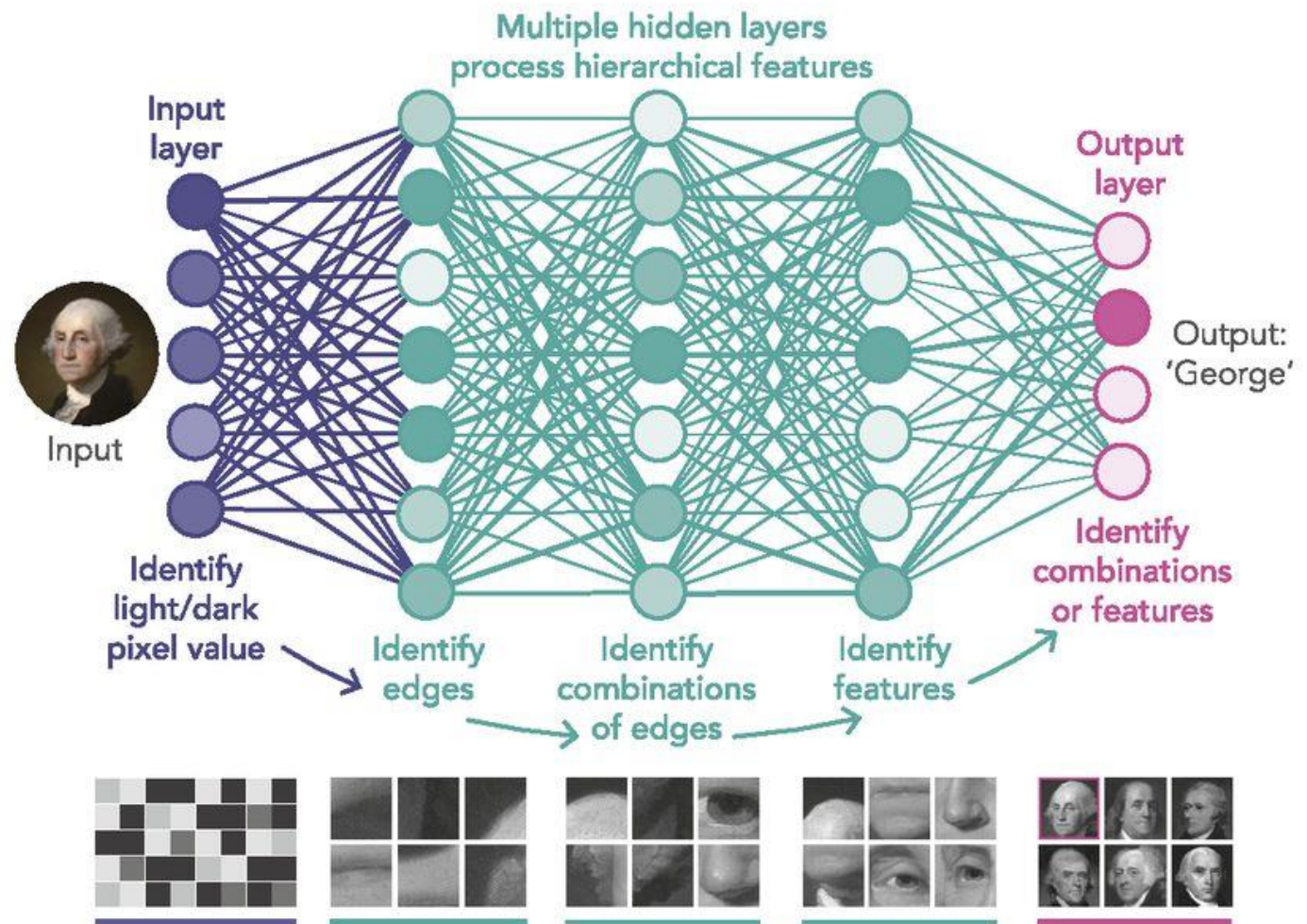


OF THE DATA AT EACH
LAYER.

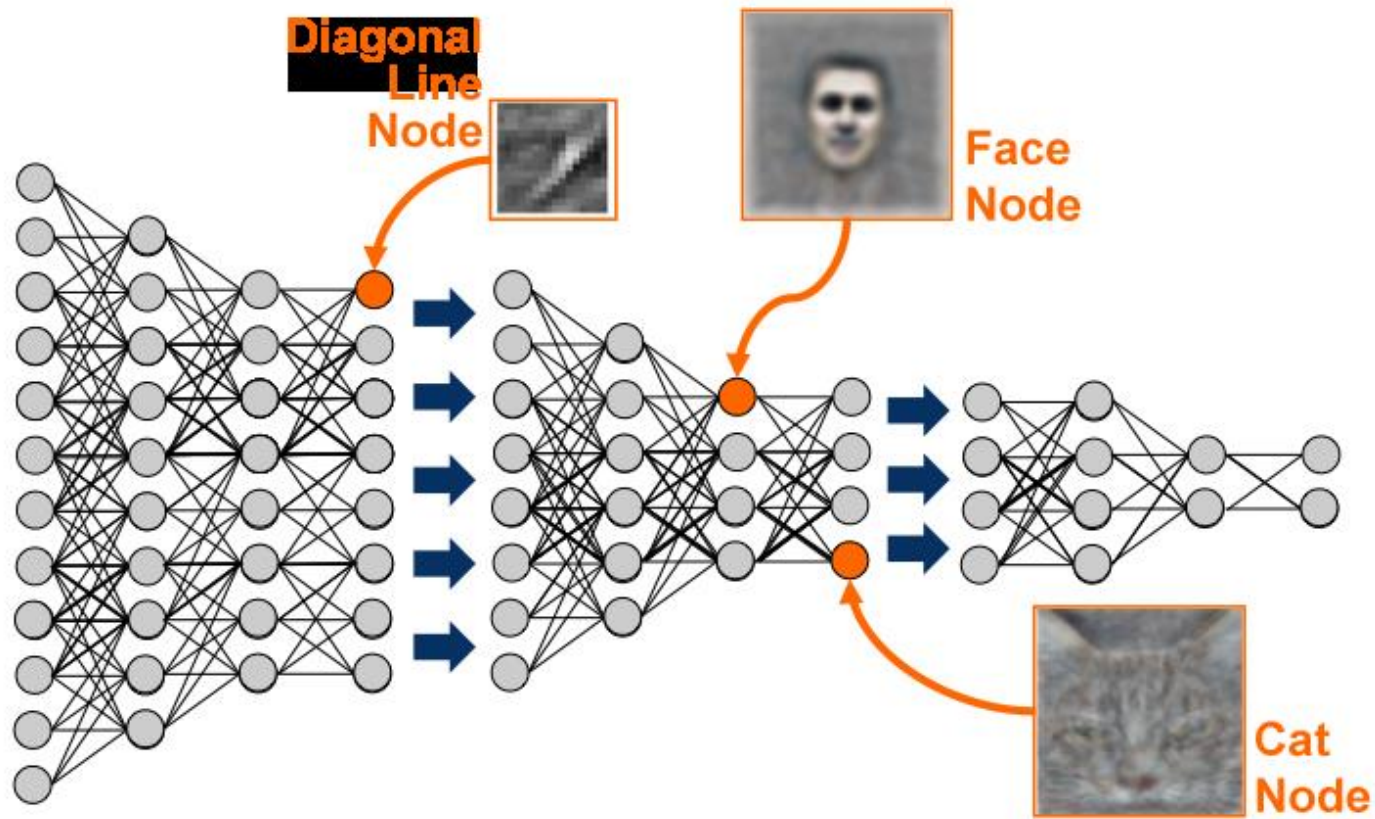
1980S-ERA NEURAL NETWORK



DEEP LEARNING NEURAL NETWORK



Deep Neural Network



- Provides state-of-the-art
- accuracy in many tasks,
- from object detection
- to speech recognition

Deep Learning Process



DEEP LEARNING IS A CLASS OF MACHINE LEARNING ALGORITHMS THAT USES MULTIPLE LAYERS TO PROGRESSIVELY EXTRACT HIGHER-LEVEL FEATURES FROM THE RAW INPUT.

STEP 1



Understand the
Problem

STEP 2



Identify Data

STEP 3



Select Deep
Learning
ALgorithms

STEP 4



Training the
Model

STEP 5



Test the
Model



Basics of Neural Networks

Surendra Panpaliya
International Corporate Trainer

What is a Neural Network?



Inspired by human brain structure.



Composed of interconnected layers.



Processes data and learns patterns.



Basis of artificial intelligence systems.



Used in deep learning models.

Structure of a Neural Network



Input layer for raw data.



Hidden layers for computations.



Output layer for final predictions.



Nodes connected via weights.



Bias added for flexibility.

How Neural Networks Work

Data flows through network layers.

Weights adjust during training process.

Uses backpropagation to minimize errors.

Activations determine node output.

Learns patterns from input data.

Types of Neural Networks



Feedforward: Data moves one way.



Convolutional: For image recognition tasks.



Recurrent: Processes sequential data.



Generative: Creates new data samples.



Deep: Many hidden computation layers.

Applications of Neural Networks



Image and speech recognition.



Natural language processing tasks.



Medical diagnosis and drug discovery.



Autonomous vehicles and robotics.



Predictive analytics in businesses.

Advantages of Neural Networks

Handles complex data relationships.

Learns without explicit programming.

Adaptive to changing data trends.

Scales for large datasets effectively.

Enables advancements in AI research.

Limitations of Neural Networks



Requires large labeled datasets.



Computationally expensive to train.



Interpretable models are challenging.



Risk of overfitting on small data.



Sensitive to hyperparameter tuning.

Summary of Neural Networks



Mimics brain's learning process.



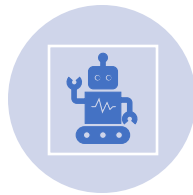
Solves complex real-world problems.



Powers advancements in AI fields.



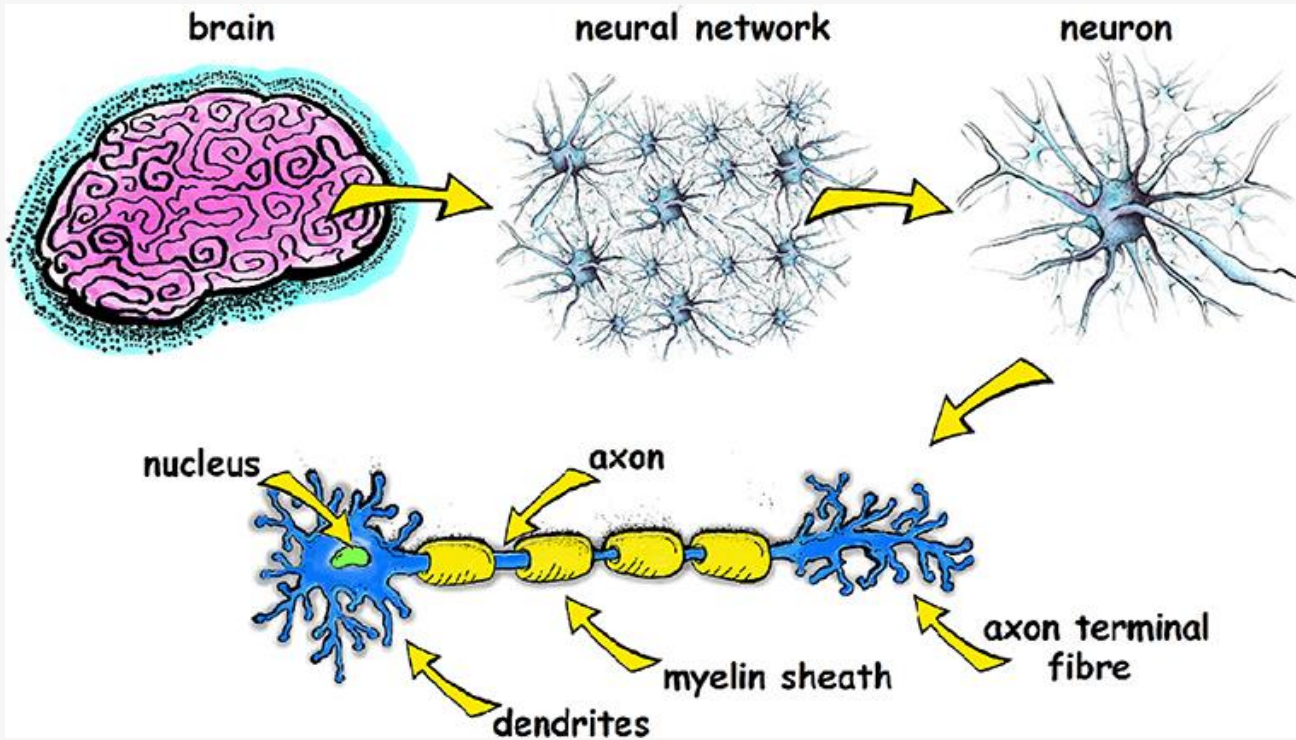
Used in diverse industry applications.



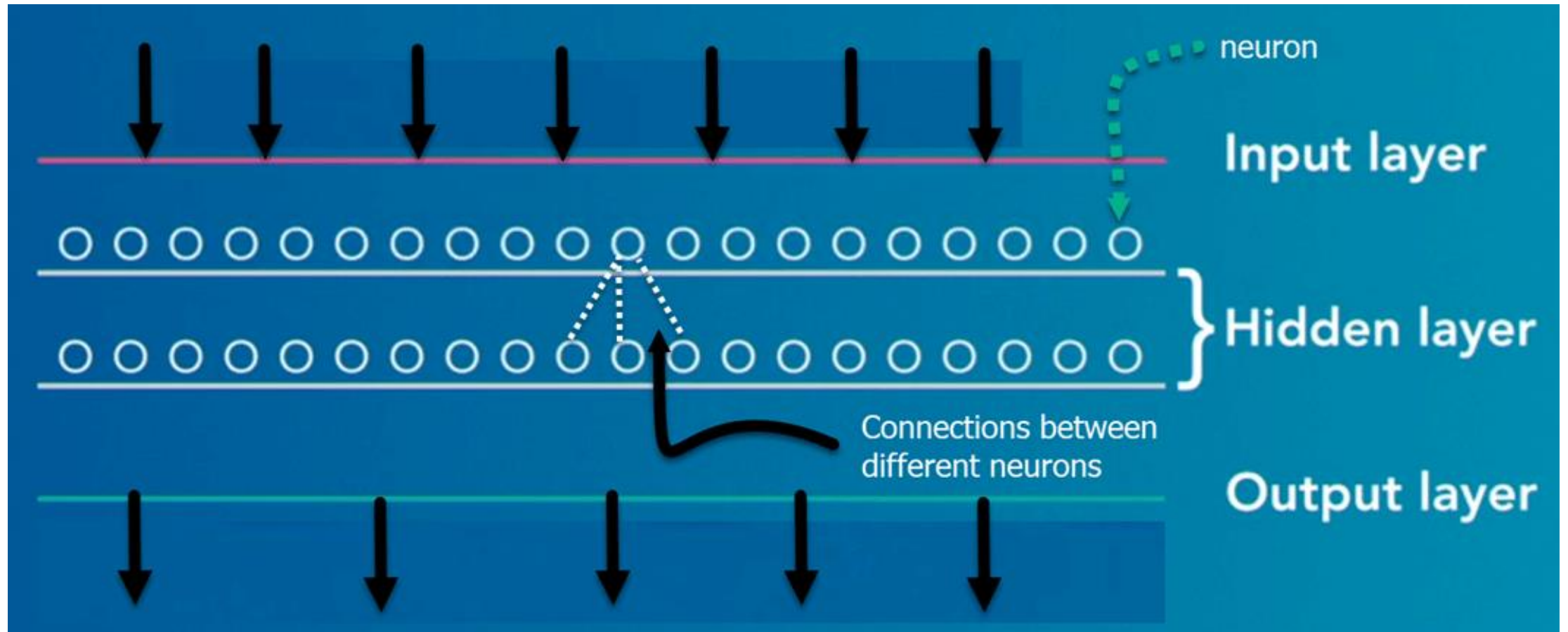
Revolutionizing technology and research.

1. Neurons

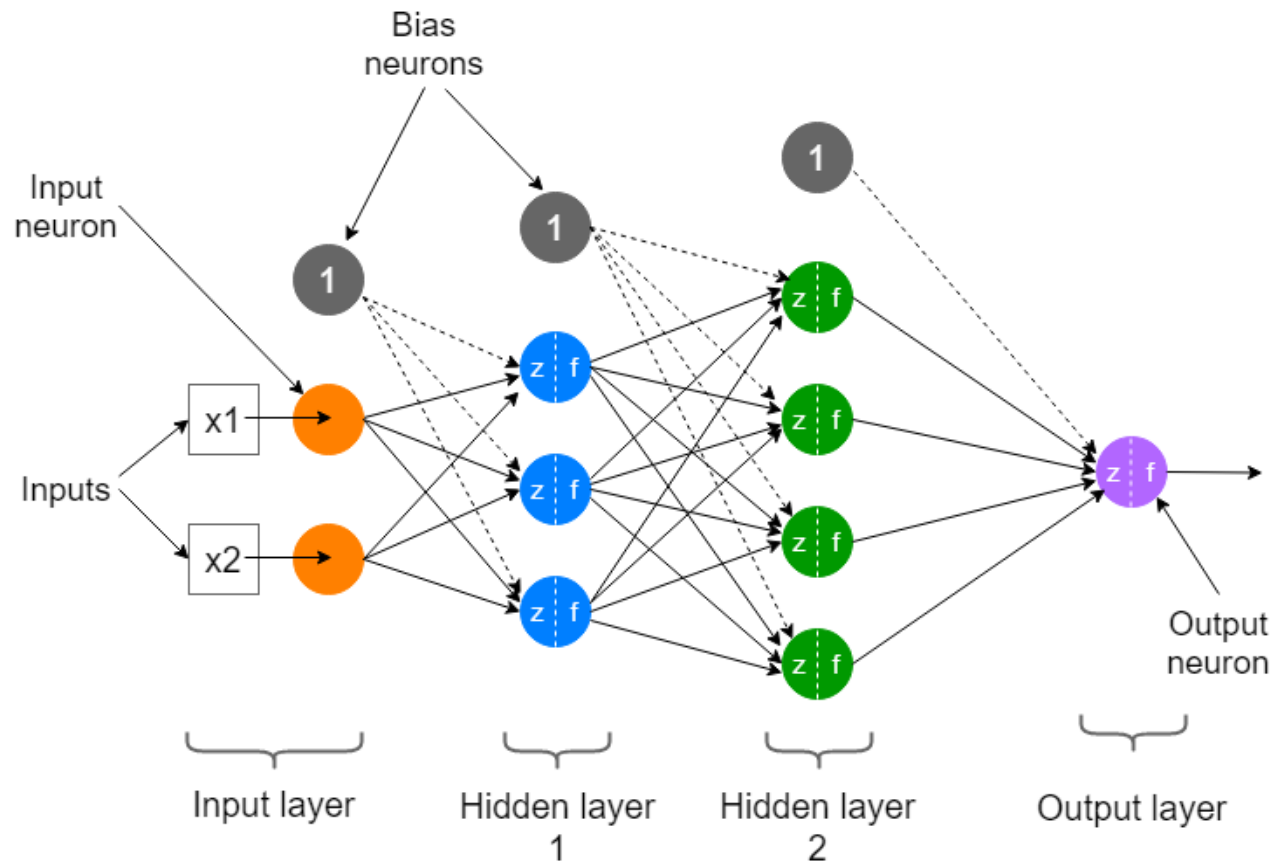
Each neuron receives input,
Performs a computation
Produces an output.



2. Layers



2. Layers



- Neurons in hidden layers
- Perform computations
- Learn representations from the data.
- Produces the final output of the network.

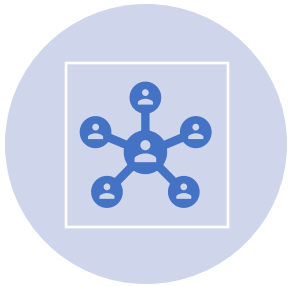
3. Connections and Weights



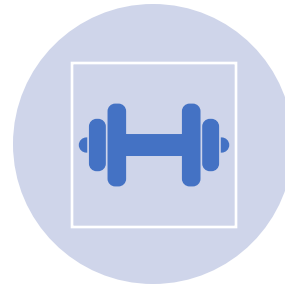
Neurons are connected
to neurons



in the next layer



through connections,



each associated with a
weight.

3. Connections and Weights



The weights represent



the strength of the connections



Learned during the training process.

- **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$
- **Sigmoid:** $f(x) = \frac{1}{1+e^{-x}}$
- **Tanh:** $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$

4. Activation Function

Each neuron has an activation function
Determines its output
Based on the weighted sum of its inputs.

5. Feedforward Process



Input data is passed through the network



Computations are performed layer by layer



Until the final output is produced.



This process transforms input data into a prediction.

6. Loss Function

Measures difference between

the predicted output

the actual target.

Quantifies the error of

the model's predictions

7. Backpropagation

An optimization algorithm

used to train neural networks.

involves adjusting the weights

based on the error (loss)

calculated during the feedforward process.

7. Backpropagation



This process is iteratively performed



to minimize the loss and



improve the model's performance.

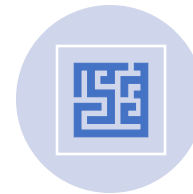
8. Training Data



Neural networks
require



Labelled training
data



To learn and
generalize patterns.



Consists of input
features



Corresponding
target labels.



9. Gradient Descent



An optimization algorithm



used in the backpropagation process

9. Gradient Descent



It adjusts weights in the direction that



minimizes the loss function,



gradually converging towards



the optimal set of weights.



10. Epochs and Batch Size



Training is typically done in epochs,



where one epoch is a complete pass



through the entire training dataset.



10. Epochs and Batch Size



Batch size determines



the number of samples
processed



before updating the model's
weights.

11. Hyperparameters

Such as the learning rate

the number of hidden layers

the number of neurons in each layer,

are set before training

influence the model's learning process.

12. Overfitting and Regularization

Neural networks may overfit the training data,
capturing noise and reducing generalization.

Techniques like dropout and

regularization are used

to mitigate overfitting.

13. Validation and Testing

The model's performance is assessed

on a separate validation set during training

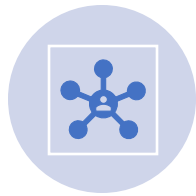
final evaluation is done

on a test set not seen during training.

Activation Functions



Transforms inputs to outputs.



Adds non-linearity to networks.



Examples: Sigmoid, ReLU, Tanh.



Determines node activation level.



Crucial for learning complex patterns.

Forward Propagation



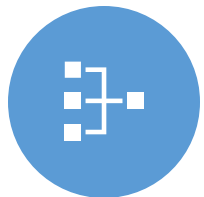
Data flows through the network.



Calculates predictions from inputs.



Weights and biases influence results.



Layer-by-layer computations occur.



Foundation of neural network training.

Backpropagation

Adjusts weights using error gradients.

Minimizes prediction errors iteratively.

Relies on chain rule calculations.

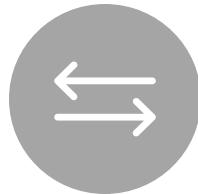
Updates propagate back through layers.

Essential for supervised learning tasks.

Gradient Descent



Optimizes network parameters iteratively.



Minimizes loss function value.



Uses learning rate for step size.

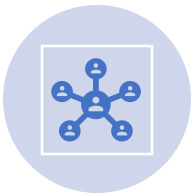


Variants:
Stochastic, Mini-
batch, Batch.



Drives convergence
to optimal
solutions.

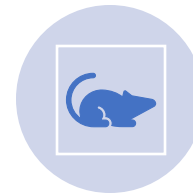
Summary of Key Concepts



Activation: Adds network flexibility.



Forward Propagation: Calculates predictions.



Backpropagation: Refines model accuracy.



Gradient Descent: Optimizes parameters.



Core to training neural networks.

Summary of Neural Networks



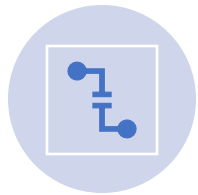
Neural networks
can take



Various
architectures



feedforward,
convolutional, and



recurrent
networks,



depending on the
nature of



the data and the
problem at hand

Summary of Neural Networks



Understanding
Neural networks



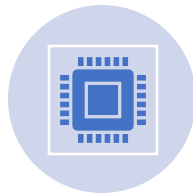
crucial for
working with



Deep learning
models



developing
solutions for a

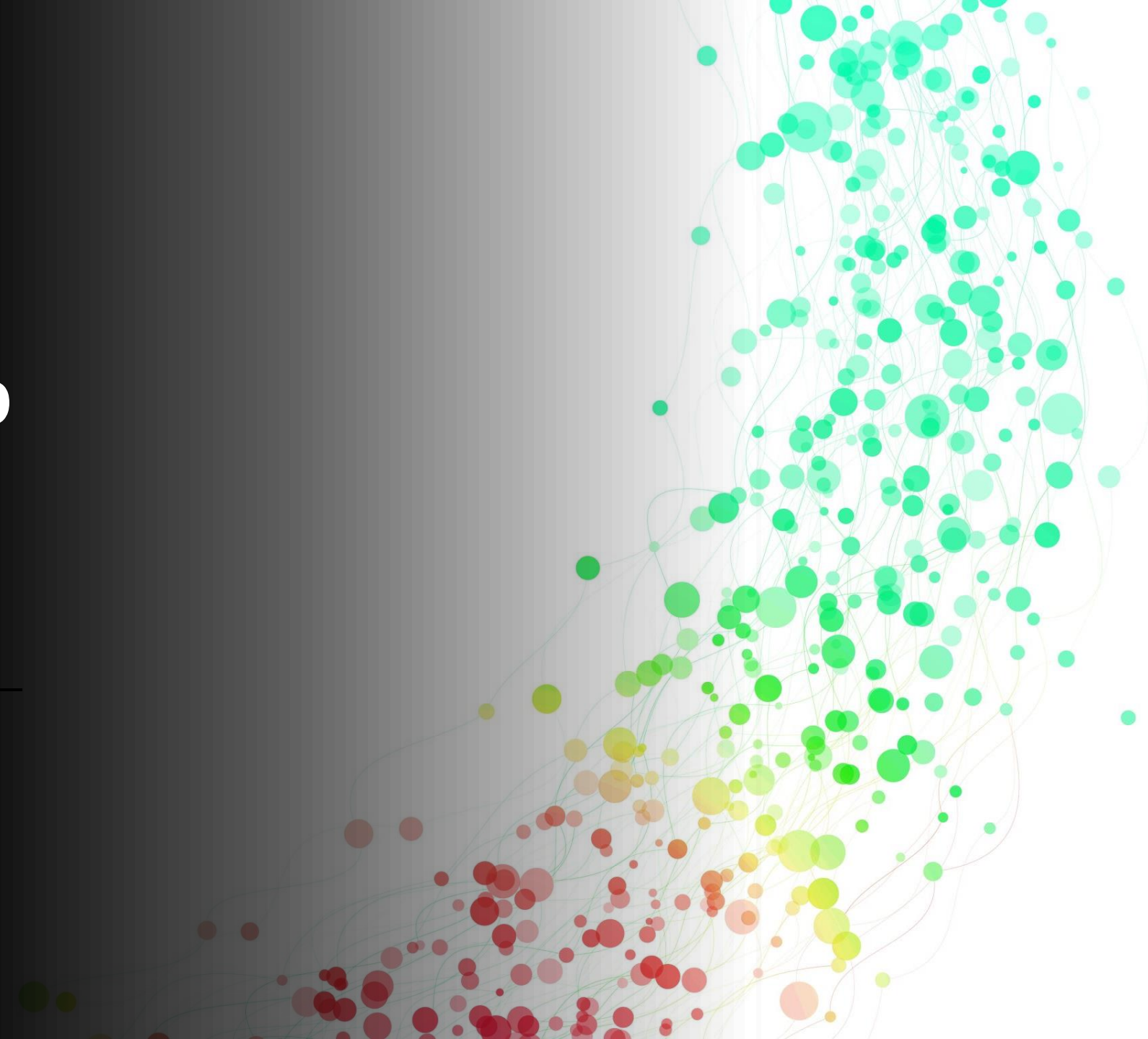



wide range of
applications.




Introduction to Deep learning frameworks

Surendra Panpaliya





Deep learning frameworks



Software libraries or tools



Provide a set of pre-built functions



Modules to simplify the development,



Training, Deployment of



Deep learning models

Deep learning frameworks



These frameworks offer an abstraction layer



for working with neural networks,



making it easier for researchers and developers



to implement complex architectures and



algorithms without having



to build everything from scratch.

TensorFlow



Developed by the Google Brain team

TensorFlow is an open-source

machine learning library that

supports both deep learning and

traditional machine learning.



TensorFlow Features



Flexible and scalable for a wide range of applications.



TensorFlow offers a high-level API



Called Keras for easy model building.



TensorFlow Lite enables



deployment on mobile and edge devices.



TensorFlow Use Cases



Image and speech recognition.



Natural language processing.



Reinforcement learning.



Website: TensorFlow

2. PyTorch



An open-source deep learning framework



developed by Facebook's AI Research lab (FAIR).



It has gained popularity



for its dynamic computation graph



ease of use.

2. PyTorch Features



Dynamic computation graph allows for more intuitive model building and debugging.



PyTorch Lightning provides a lightweight PyTorch wrapper for high-level abstractions



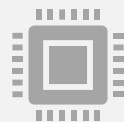
Extensive community support and a strong ecosystem.



2. PyTorch Use Cases



Natural language processing.



Computer vision.



Generative models.



Website: PyTorch

3. Keras

Initially developed as a high-level API

for building neural networks

on top of other frameworks,

Keras is now integrated as

the official high-level API for TensorFlow.



3. Keras Features



Simple and user-friendly
syntax



for building



experimenting with models.

3. Keras Features

Supports Convolutional,

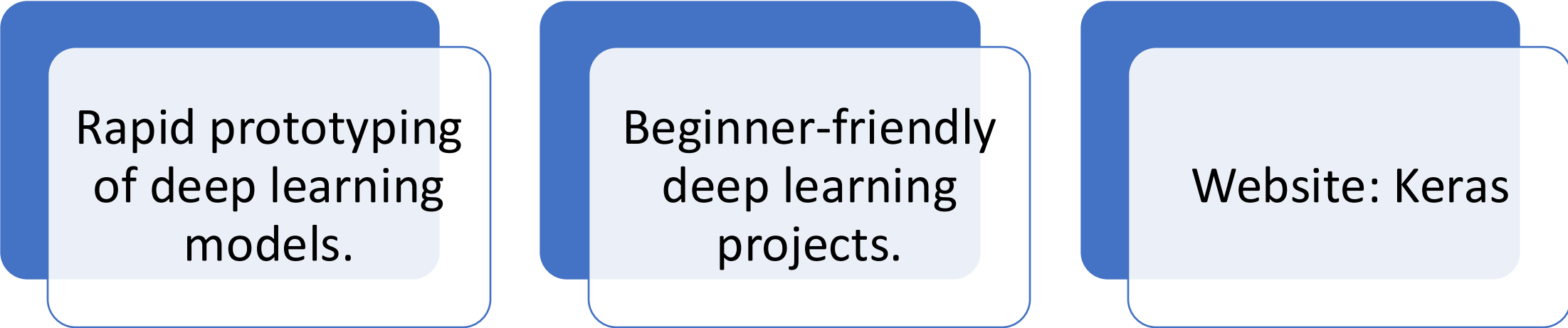
Recurrent, and

Dense networks.

Easy model deployment

using TensorFlow.

3. Keras Use Cases



Rapid prototyping
of deep learning
models.

Beginner-friendly
deep learning
projects.

Website: Keras

4. Apache MXNet

An open-source deep learning framework

supports both symbolic and

imperative programming.

Designed for flexibility and efficiency



4. Apache MXNet Features



Supports symbolic and imperative programming models.



Efficient for distributed computing and multi-GPU training.



Gluon API provides a concise and dynamic approach



to building models.

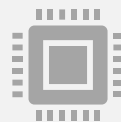




4. MXNet Use Cases



Natural language processing.



Computer vision.



Recommendation systems.



Website: Apache MXNet

5. Caffe



Deep learning framework developed by the Berkeley Vision and



Learning Center.



Known for its speed and efficiency,



Especially in Convolutional Neural Networks (CNNs)



5. Caffe Features



Fast and efficient for image classification tasks.



Supports GPU acceleration.



Pre-trained models available in the Caffe Model Zoo.



5. Caffe Use Cases

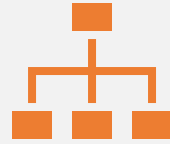


Image classification.



Object detection.



Website: Caffe

6. Theano (Deprecated):

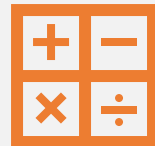
Early deep learning framework

Developed by the Montreal Institute for Learning Algorithms

It is no longer actively maintained



6. Theano (Deprecated) Features



Efficient computation of mathematical expressions.



GPU acceleration support.



Integration with NumPy.



6. Theano (Deprecated) Use Cases



Early deep learning research
and experimentation.



Website: Theano

Summary and Conclusion

Choosing a deep learning framework

depends on factors such as

personal preference,

project requirements, and

community support.

Summary and Conclusion

**TensorFlow
and PyTorch**

The most
popular

Widely used
frameworks,

Each with its
strengths

Unique
features.

Summary and Conclusion

Frameworks offer

Compatibility with popular hardware accelerators like

Graphics Processing Unit (GPU)

Tensor Processing Units (TPUs)



TensorFlow Overview



Open-source deep learning framework.



Developed by Google Brain team.



Supports neural network modeling.



Scalable for production environments.



Ideal for machine learning tasks.



PyTorch Overview



Flexible deep learning framework.



Developed by Facebook AI Research.



Known for dynamic computation graphs.



Popular for research and prototyping.



Supports distributed training efficiently.

TensorFlow Key Features

Eager execution for quick debugging.

Supports TensorFlow Extended (TFX).

Offers TensorFlow Lite for devices.

Provides TensorBoard for visualization.

Seamless deployment on various platforms.

PyTorch Key Features

Dynamic computation graph flexibility.

Supports TorchScript for deployment.

Integration with Pythonic tools.

Highly compatible with NumPy libraries.

Strong support for custom models.

Comparison: TensorFlow vs PyTorch



TensorFlow:
Production scalability
focus.



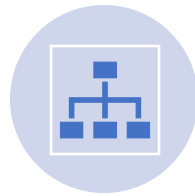
PyTorch: Research
prototyping
flexibility.



TensorFlow: Static
and dynamic graphs.



PyTorch: Purely
dynamic
computation graphs.



Choose based on
project needs.



Deep Learning Frameworks Summary



TensorFlow: Robust and production-ready.



PyTorch: Flexible and researcher-friendly.



Both support deep learning tasks.



Extensive community and ecosystem support.



Key tools for AI advancements.

What is PyTorch?



Open-source deep learning framework.



Developed by Facebook AI Research.



Supports dynamic computation graphs.



Ideal for research and prototyping.



Widely used for AI tasks.



PyTorch Key Features



Dynamic computation graph flexibility.



Supports custom neural network models.



Integration with Pythonic libraries.



Simplifies debugging and experimentation.



Efficient for distributed training.

PyTorch Use Cases

Computer vision and image analysis.

Natural language processing (NLP).

Time-series data analysis tasks.

Reinforcement learning for agents.

Custom AI and ML applications.



Advantages of PyTorch



Easy to learn and use.



Highly flexible for experiments.



Strong support from open community.



Works seamlessly with NumPy tools.



Efficient model deployment options.

Summary of PyTorch



Dynamic and flexible framework.



Ideal for research and prototyping.



Extensive library and tool support.



Simplifies AI and ML development.



Powers advancements in deep learning.



Introduction to TensorFlow



Open-source deep learning framework.



Developed by Google Brain team.



Supports both static, dynamic graphs.



Widely used for production models.



Extensive tools and libraries.

Introduction to PyTorch



Open-source deep learning framework.



Developed by Facebook AI Research.



Uses dynamic computation graphs.



Ideal for research and prototyping.



Flexible for custom model building.

Graph Handling

TensorFlow: Static, dynamic supported.

PyTorch: Dynamic graph approach.

TensorFlow: Optimized for production.

PyTorch: Easy debugging flexibility.

Dynamic graphs aid experimentation.

Ease of Use

TensorFlow: Steeper learning curve.

PyTorch: More intuitive design.

TensorFlow: Extensive community support.

PyTorch: Growing research preference.

Choose based on experience level.

Deployment

TensorFlow: TensorFlow Serving available.



```
graph TD; A[TensorFlow: TensorFlow Serving available.] --> B[PyTorch: TorchScript aids deployment.]; B --> C[TensorFlow: Production-ready pipelines.]; C --> D[PyTorch: Increasing deployment tools.]; D --> E[Both enable scalable solutions.]
```

PyTorch: TorchScript aids deployment.

TensorFlow: Production-ready pipelines.

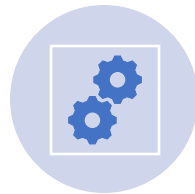
PyTorch: Increasing deployment tools.

Both enable scalable solutions.

Use Cases



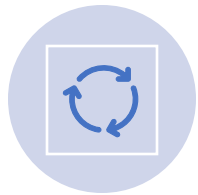
TensorFlow:
Enterprise-scale
applications.



PyTorch: Rapid
prototyping for
research.



TensorFlow:
Comprehensive
ecosystem tools.



PyTorch: Custom,
experimental
projects.



Tailored to specific
project goals.

Summary



TensorFlow: Production-focused framework.



PyTorch: Research-friendly and flexible.




TensorFlow: Larger tool ecosystem.



PyTorch: Simpler debugging workflows.



Choose based on project needs.



Surendra Panpaliya
Founder and CEO
GKTCS Innovations

<https://www.gktcs.com>

