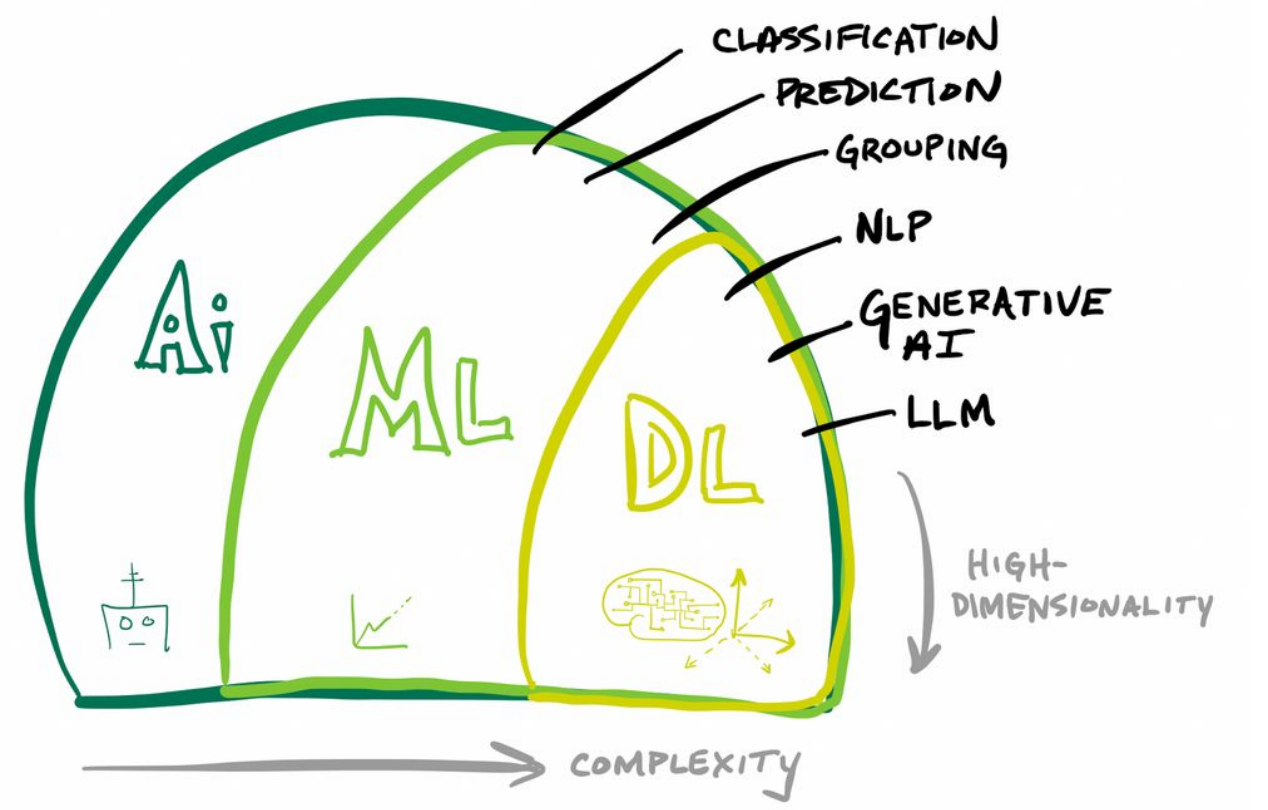
AI from Scratch:
Build your First Neural Network

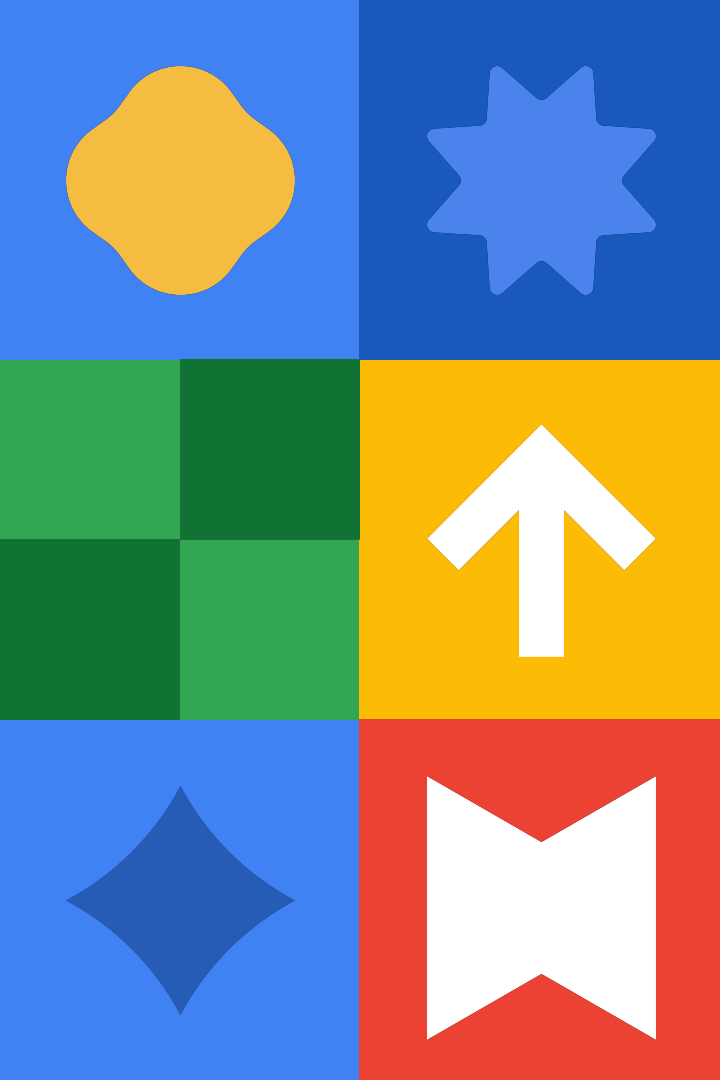
Ayush Morbar

Founder, Offbeats

Head of AI/ML, GDGoC RJIT

OFF BEATS


CLASSIFICATION
PREDICTION
GROUPING
NLP
GENERATIVE AI
LLM
AI
ML
DL
HIGH-DIMENSIONALITY
COMPLEXITY

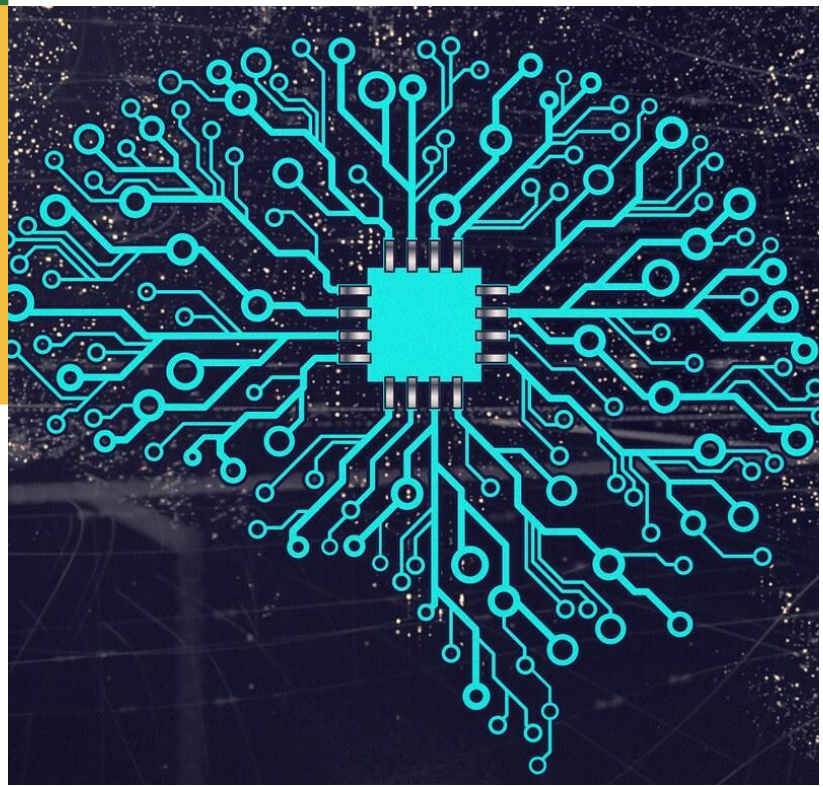


1. From Biology to Mathematics
2. TensorFlow: Your AI Toolkit
3. Hands-On Project
4. Beyond the Basics

POLL

Quick Poll: Your ML Experience

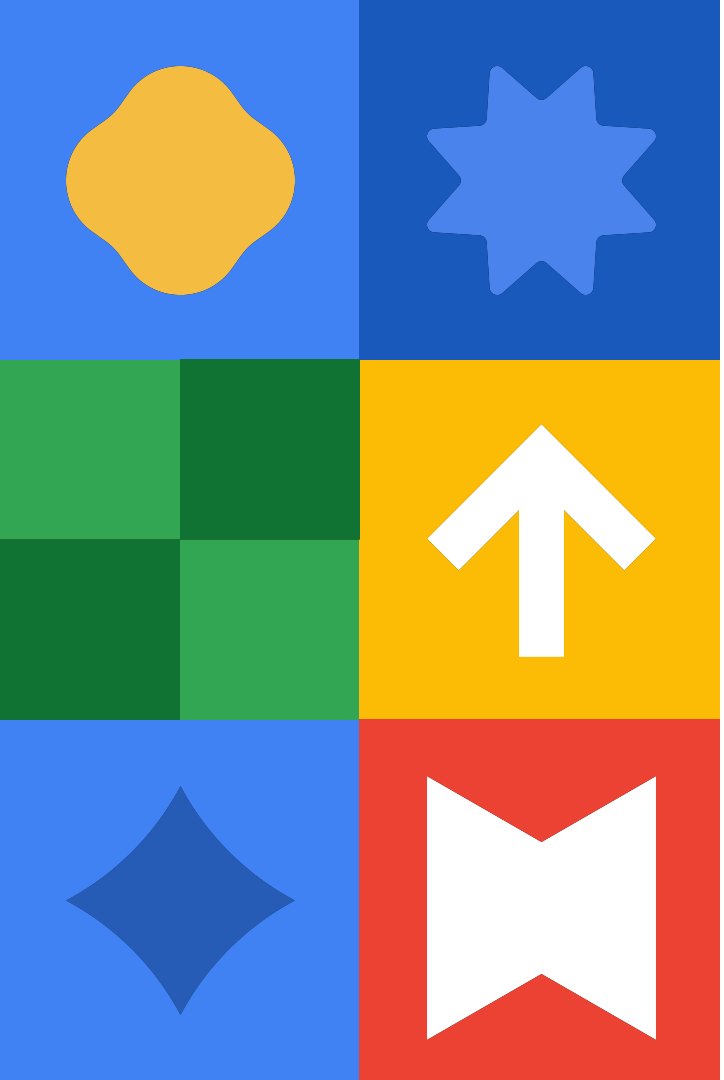
A. ✋ Complete Beginner
B. 🤓 Heard of ML/AI
C. 💻 Some Programming Experience
D. 🔥 Ready to Build Neural Networks!



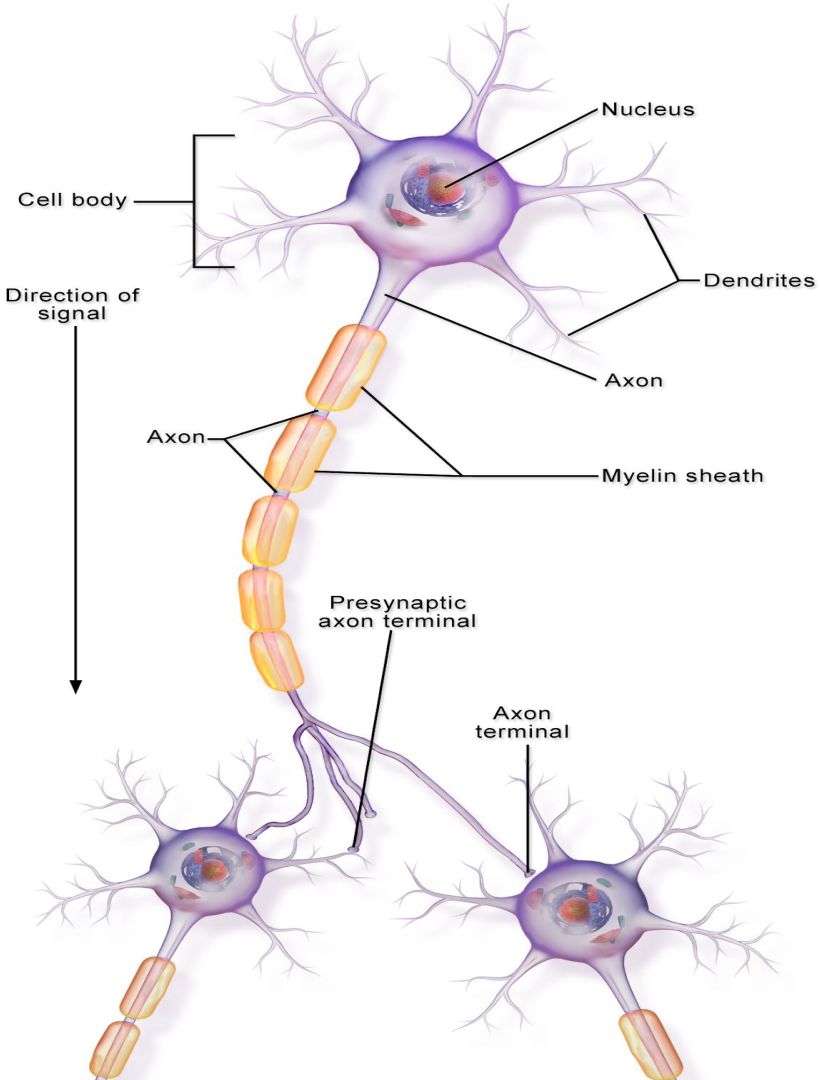
WHY NEURAL NETWORKS MATTER

Neural Networks Are Everywhere

- Your smartphone's camera (image recognition)
- Netflix recommendations
- Google Search & translation
- Self-driving cars
- Medical diagnosis
- Climate prediction



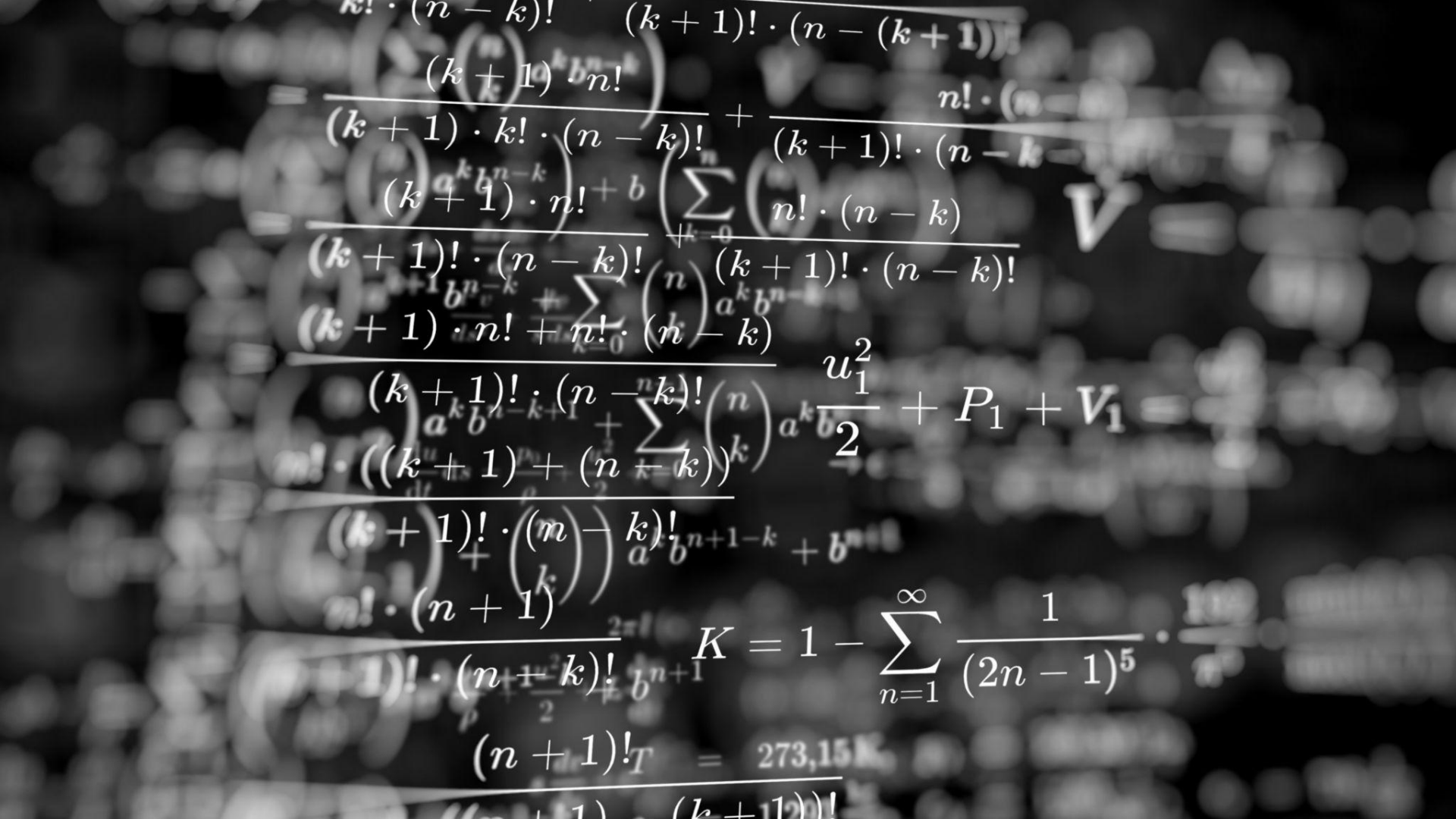
From Biology to Mathematics *Understanding the Foundation*



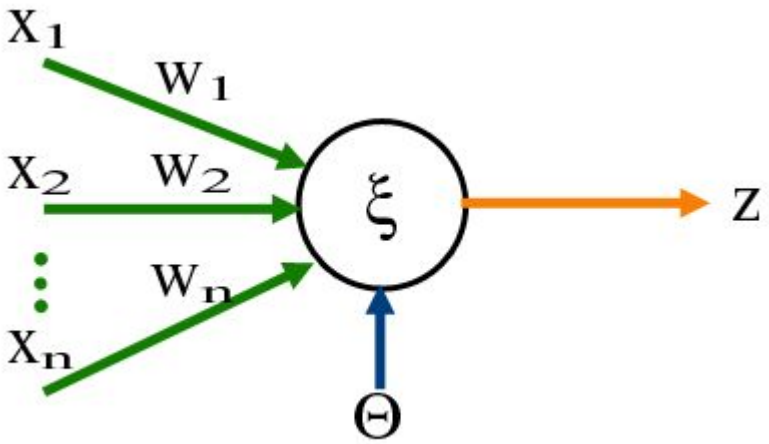
BIOLOGICAL INSPIRATION

The Human Brain: Nature's Computer

- 86 billion neurons
- Each neuron connects to ~7,000 others
- Processes information in parallel
- Learns through experience
- Inspired artificial neural networks


$$\frac{k! \cdot (n-k)!}{(k+1)! \cdot (n-(k+1))!}$$
$$\frac{(k+1) \cdot n!}{(k+1) \cdot k! \cdot (n-k)!} + \frac{n! \cdot (n-k)}{(k+1)! \cdot (n-k)!}$$
$$\frac{(k+1) \cdot n!}{(k+1)! \cdot (n-k)!} + \frac{n! \cdot (n-k)}{(k+1)! \cdot (n-k)!}$$
$$V =$$
$$\frac{(k+1) \cdot n! + n! \cdot (n-k)}{(k+1)! \cdot (n-k)!}$$
$$\frac{n! \cdot ((k+1) + (n-k))}{(k+1)! \cdot (n-k)!}$$
$$\frac{u_1^2}{2} + P_1 + V_1 =$$
$$\frac{n! \cdot (n+1)}{(k+1)! \cdot (n-k)!}$$
$$K = 1 - \sum_{n=1}^{\infty} \frac{1}{(2n-1)^5}$$
$$\frac{(n+1)!}{((k+1)+(n-(k+1)))!}$$
$$T = 273{,}15$$

```python
# Biological Neuron → Mathematical Model
def artificial_neuron(inputs, weights, bias):
    # Weighted sum (like dendrites collecting signals)
    weighted_sum = sum(x * w for x, w in zip(inputs, weights))

    # Add bias (neuron's threshold)
    z = weighted_sum + bias

    # Activation function (neuron fires or not)
    output = sigmoid(z)
    return output

def sigmoid(z):
    return 1 / (1 + math.exp(-z))
```



THE PERCEPTRON

The Simplest Neural Network

Input Layer ⟶ Weights ⟶ Activation ⟶ Output

Mathematical Formula:

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

Where:

- x = input features
- w = learned weights
- b = bias term

NEURAL NETWORK ARCHITECTURE

Building Blocks of Intelligence

Input Layer ➜ Hidden Layer(s) ➜ Output Layer

NEURAL NETWORK ARCHITECTURE

Building Blocks of Intelligence
Input Layer ➜ Hidden Layer(s) ➜ Output Layer

NEURAL NETWORK ARCHITECTURE

Building Blocks of Intelligence
Input Layer ➜ Hidden Layer(s) ➜ Output Layer

NEURAL NETWORK ARCHITECTURE

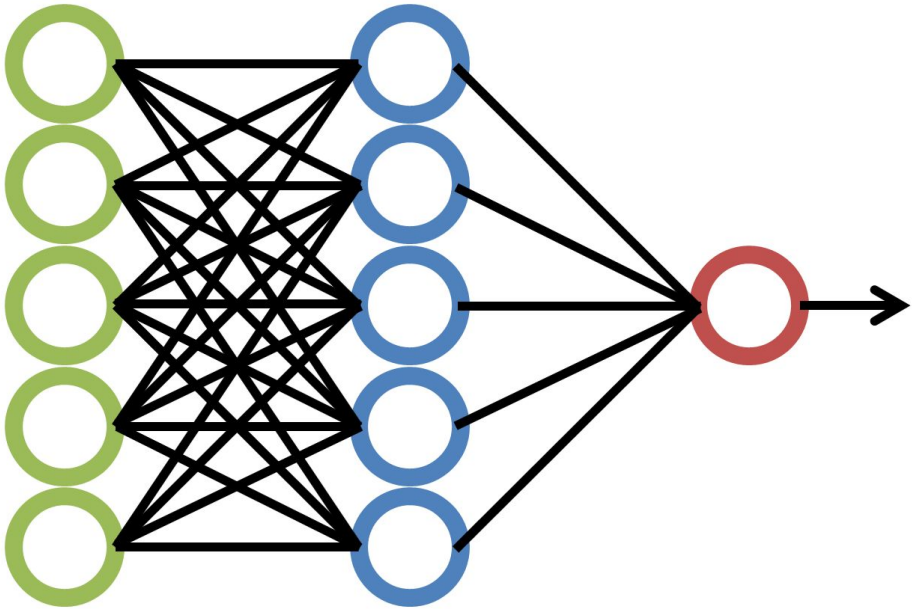
Building Blocks of Intelligence
Input Layer ➜ Hidden Layer(s) ➜ Output Layer

NEURAL NETWORK ARCHITECTURE

Building Blocks of Intelligence
Input Layer ➜ Hidden Layer(s) ➜ Output Layer

NEURAL NETWORK ARCHITECTURE

Building Blocks of Intelligence
Input Layer ➜ Hidden Layer(s) ➜ Output Layer



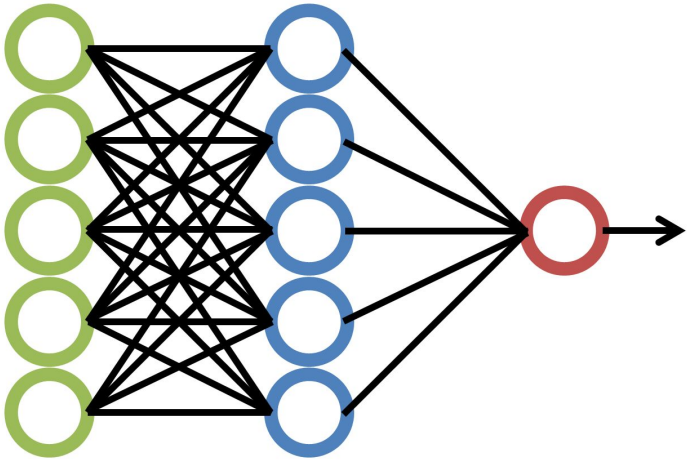
WHY DEEP NETWORKS?

Power of Depth: Learning Hierarchical Features

Layer 1: Edges & Lines

WHY DEEP NETWORKS?

Power of Depth: Learning Hierarchical Features

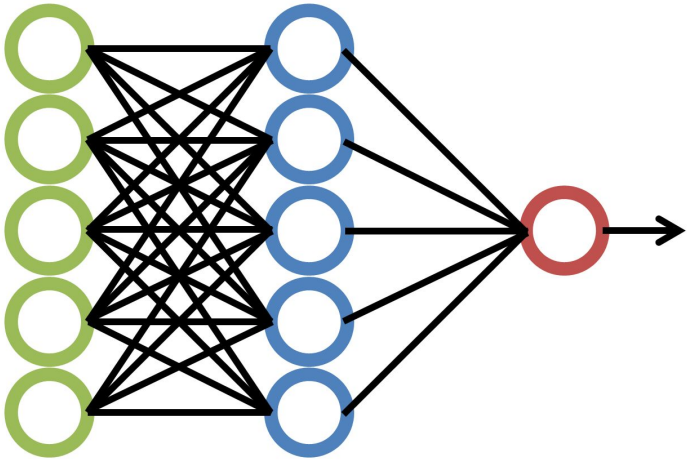


Layer 1: Edges & Lines

Layer 2: Shapes & Textures

WHY DEEP NETWORKS?

Power of Depth: Learning Hierarchical Features



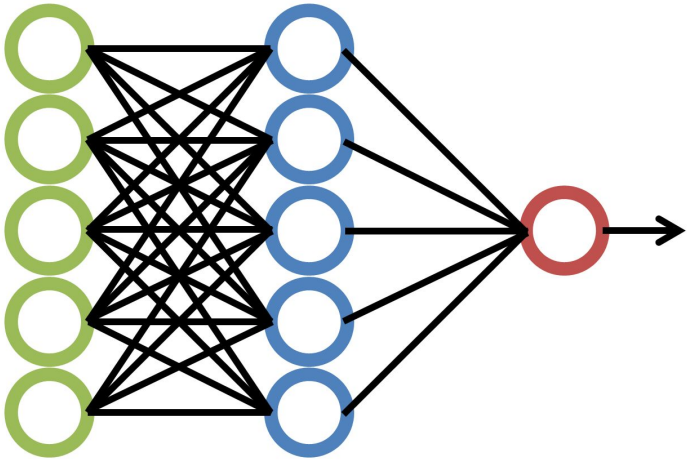
Layer 1: Edges & Lines

Layer 2: Shapes & Textures

Layer 3: Objects & Parts

WHY DEEP NETWORKS?

Power of Depth: Learning Hierarchical Features



Layer 1: Edges & Lines

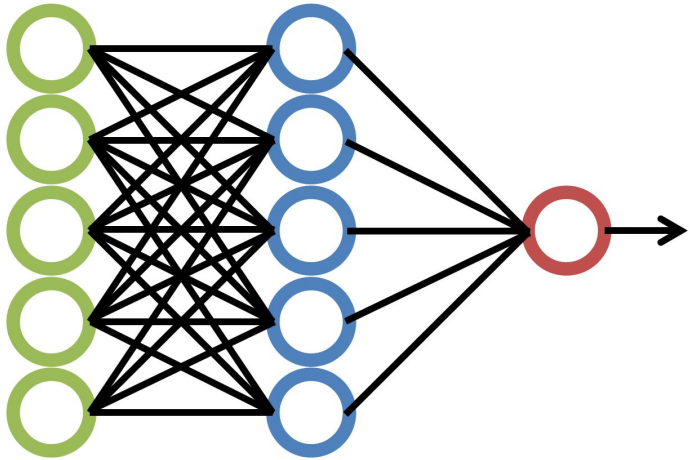
Layer 2: Shapes & Textures

Layer 3: Objects & Parts

Layer 4: Complete Objects

WHY DEEP NETWORKS?

Power of Depth: Learning Hierarchical Features

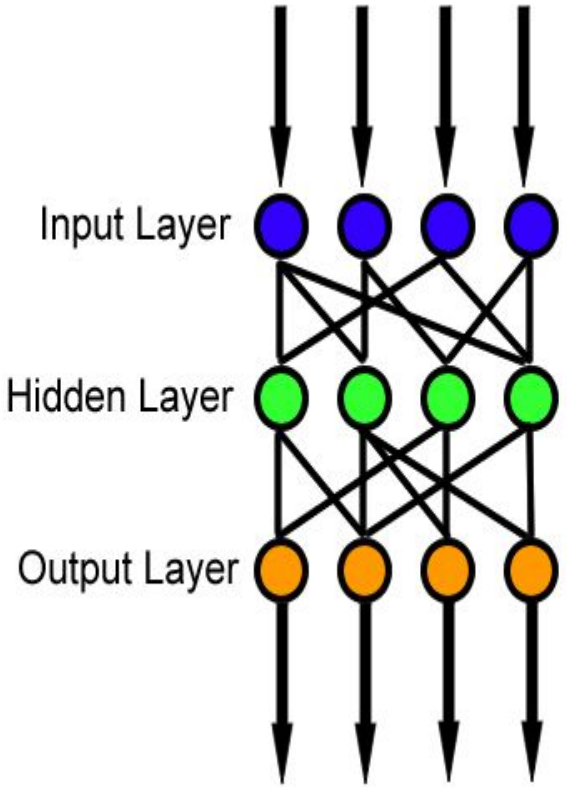


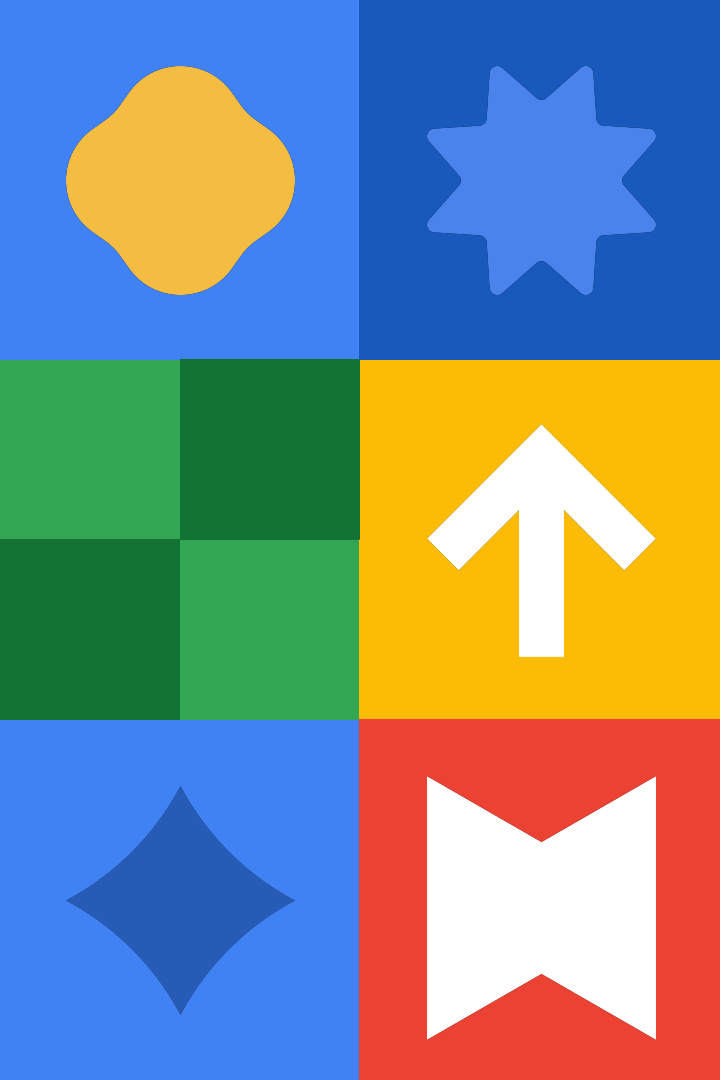
```python
import numpy as np

def forward_pass(X, W1, b1, W2, b2):
    # Layer 1: Input to Hidden
    z1 = np.dot(X, W1) + b1
    a1 = sigmoid(z1)  # Activation

    # Layer 2: Hidden to Output
    z2 = np.dot(a1, W2) + b2
    a2 = sigmoid(z2)  # Final output

    return a2, a1  # Return outputs and hidden activations
```

FORWARD PROPAGATION

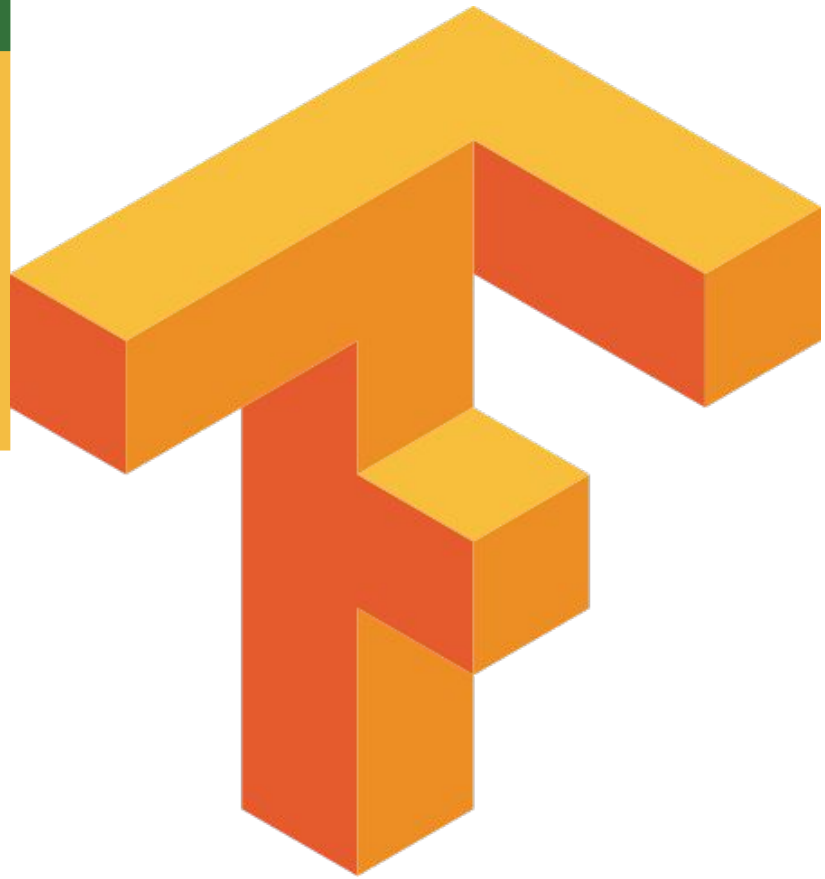
```python
import numpy as np

def forward_pass(X, W1, b1, W2, b2):
    # Layer 1: Input to Hidden
    z1 = np.dot(X, W1) + b1
    a1 = sigmoid(z1)  # Activation

    # Layer 2: Hidden to Output
    z2 = np.dot(a1, W2) + b2
    a2 = sigmoid(z2)  # Final output

    return a2, a1  # Return outputs and hidden activations
```

FORWARD PROPAGATION





TensorFlow: Your AI Toolkit *From Theory to Implementation*



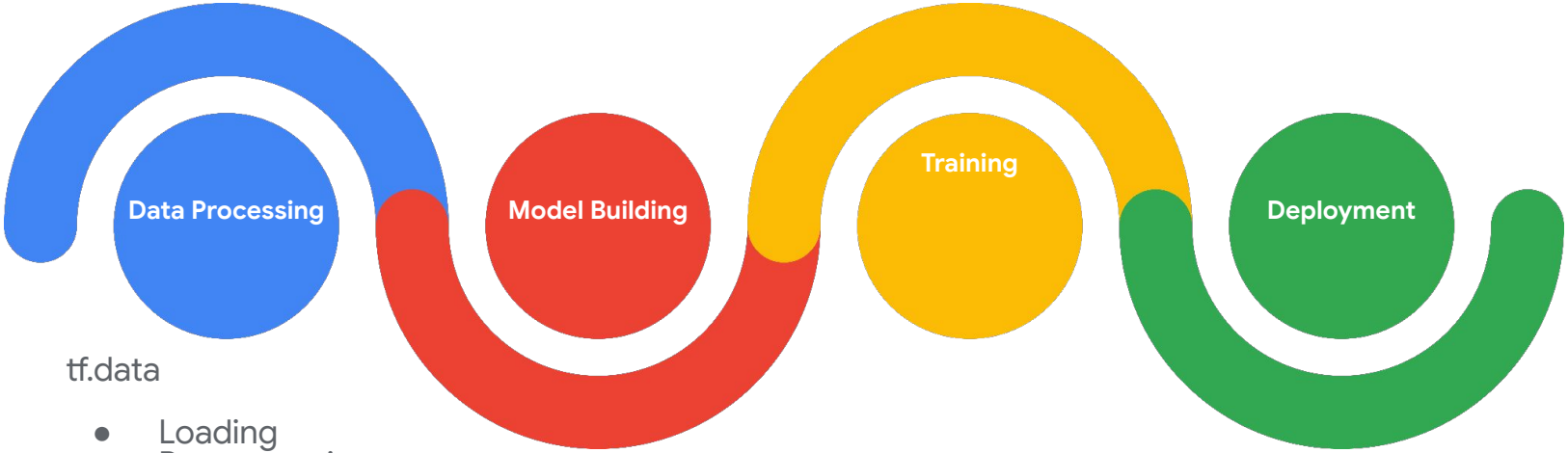
WHAT IS TENSORFLOW?

TensorFlow: Google's Open-Source AI Library

- ✓ Developed by Google Brain team
- ✓ Powers Google Search, Gmail, Photos
- ✓ Supports CPUs, GPUs, and TPUs
- ✓ Used by 50+ million developers
- ✓ Keras: High-level API for easy development

TENSORFLOW ECOSYSTEM

Complete ML Pipeline

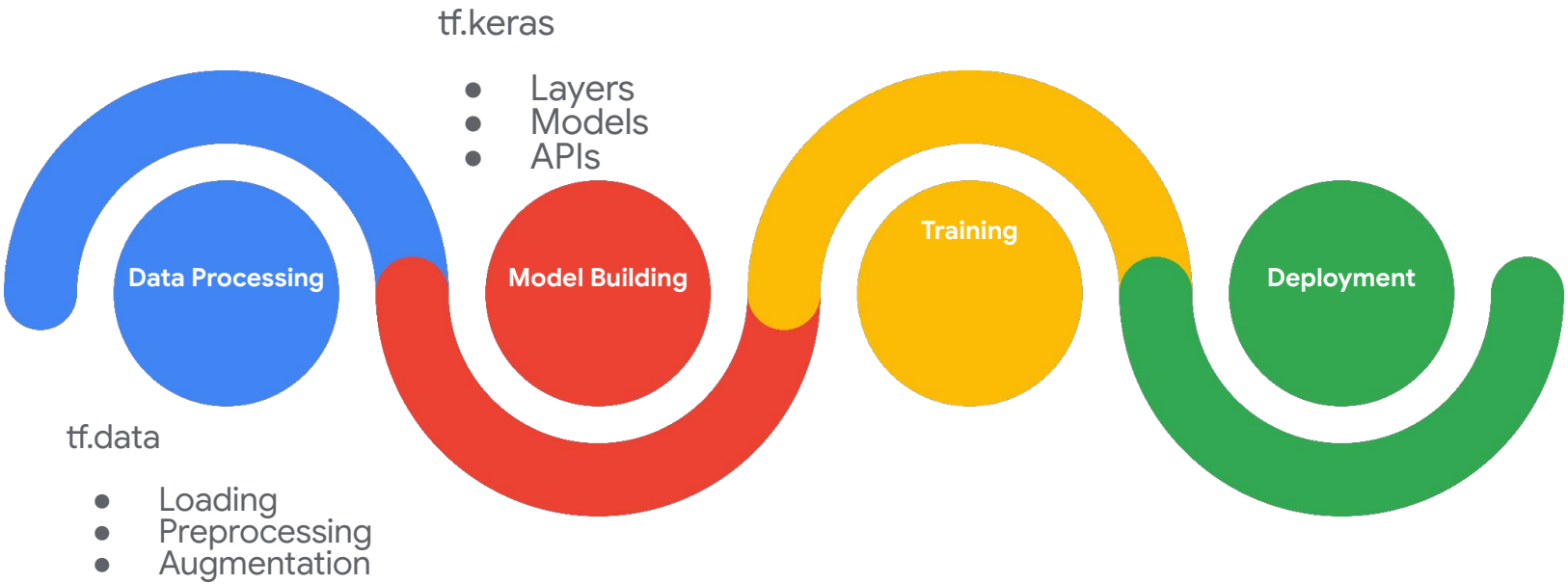


tf.data

- Loading
- Preprocessing
- Augmentation

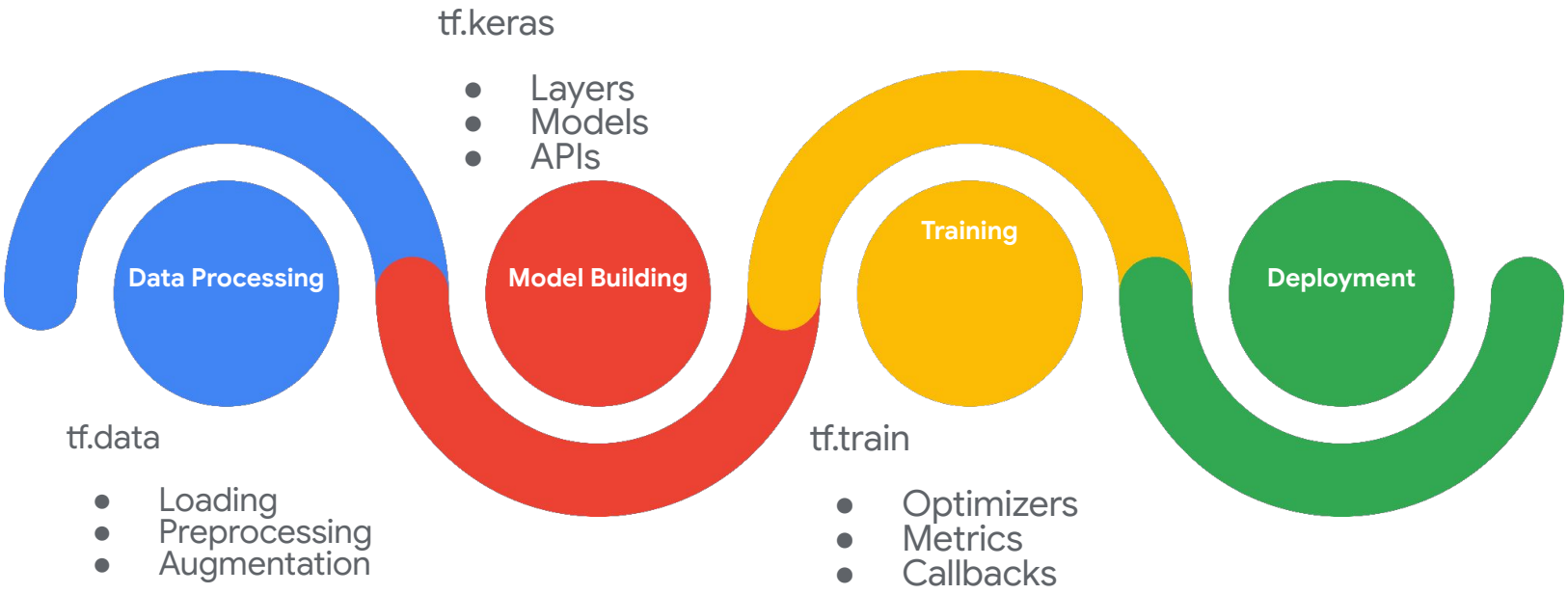
TENSORFLOW ECOSYSTEM

Complete ML Pipeline



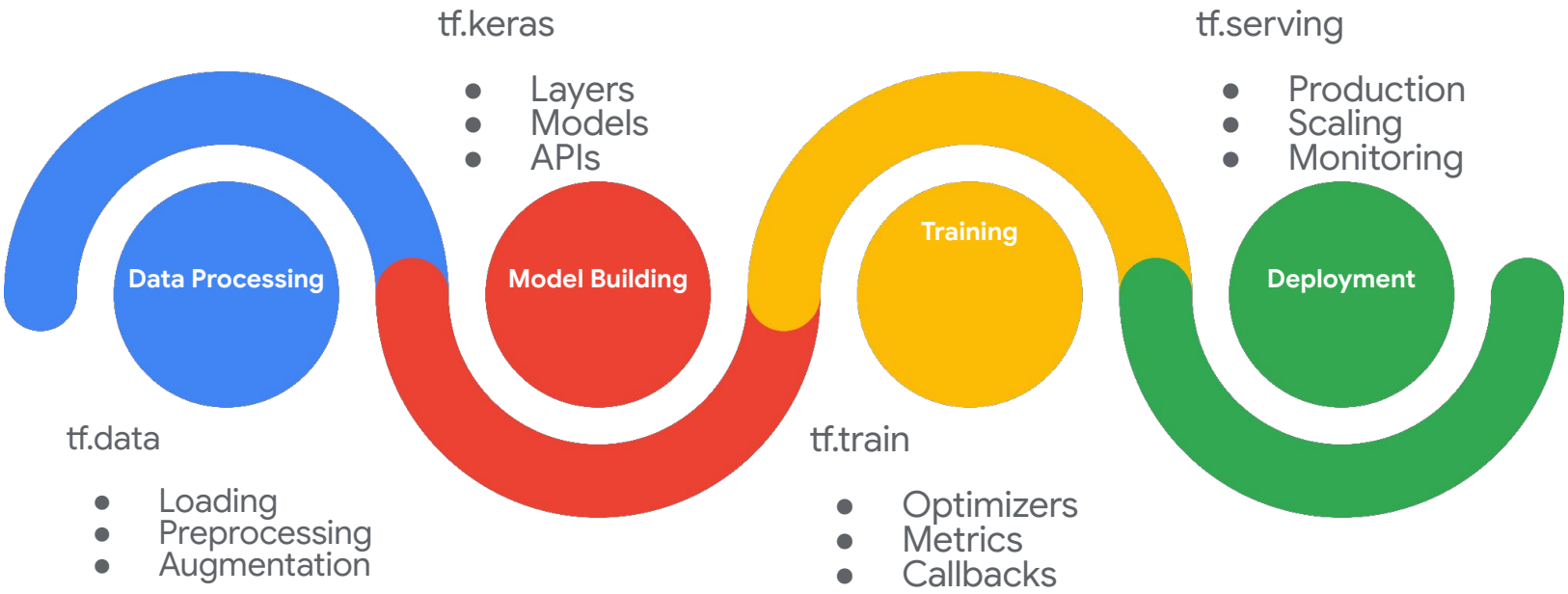
TENSORFLOW ECOSYSTEM

Complete ML Pipeline



TENSORFLOW ECOSYSTEM

Complete ML Pipeline



YOUR FIRST NEURAL NETWORK

```
import tensorflow as tf
from tensorflow import keras

# 1. Create the model
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(784,)),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# 2. Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 3. Train the model
model.fit(train_images, train_labels, epochs=5)
```

YOUR FIRST NEURAL NETWORK

UNDERSTANDING LAYERS

Neural Network Layers Explained

Dense Layer: Fully connected neurons

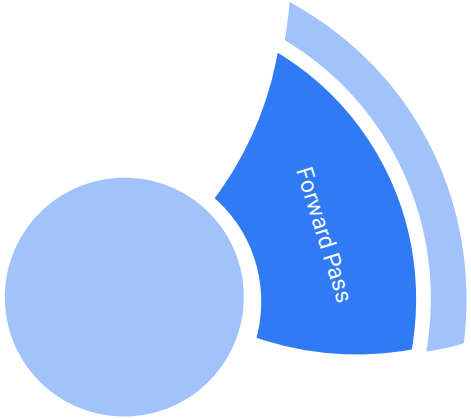
- Every input connected to every output
- Most common layer type
- Great for general problems

Activation Functions:

- ReLU: f(x) = max(0, x) - Most popular
- Sigmoid: f(x) = 1/(1+e^-x) - Output probability
- Softmax: Converts to probability distribution

TRAINING PROCESS

How Neural Networks Learn

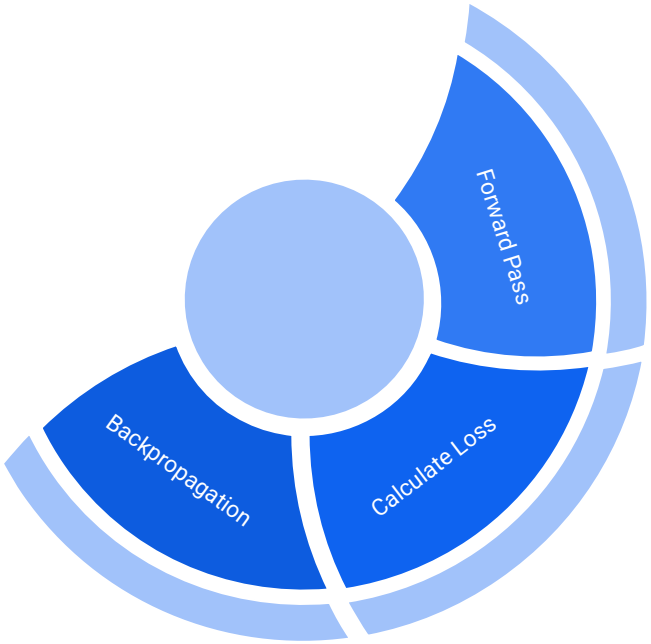


TRAINING PROCESS

How Neural Networks Learn

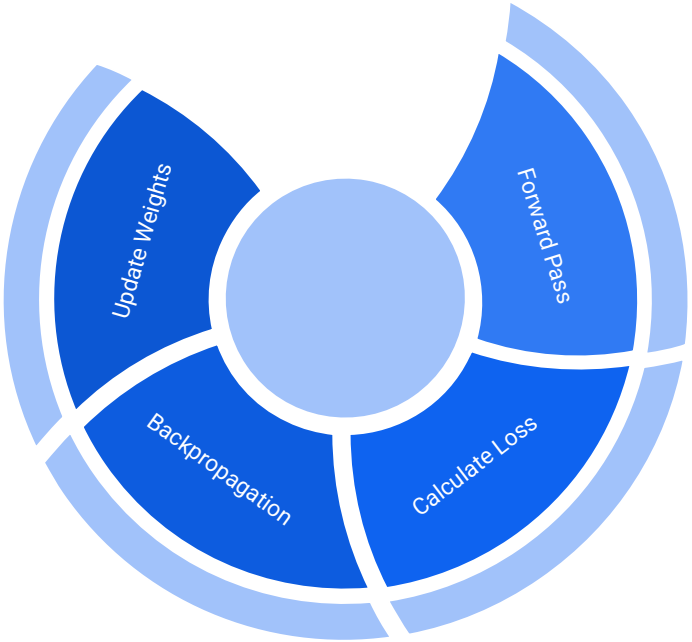
TRAINING PROCESS

How Neural Networks Learn



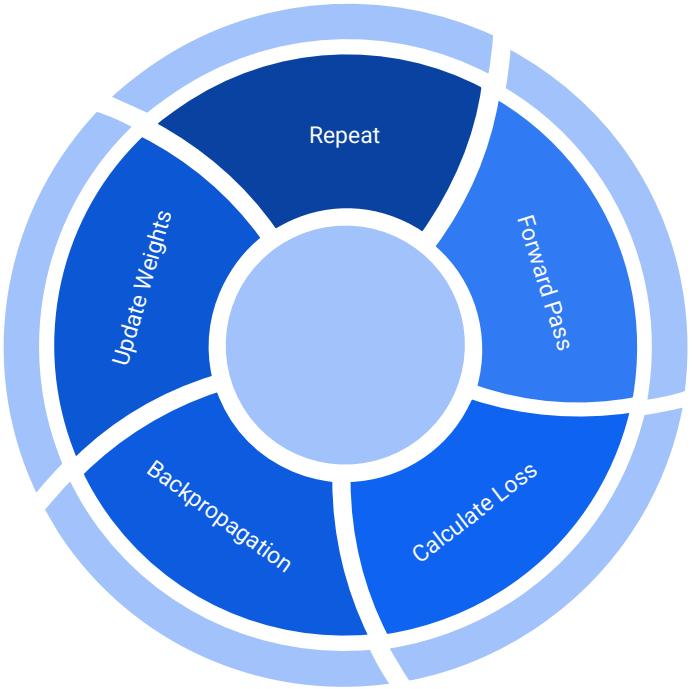
TRAINING PROCESS

How Neural Networks Learn



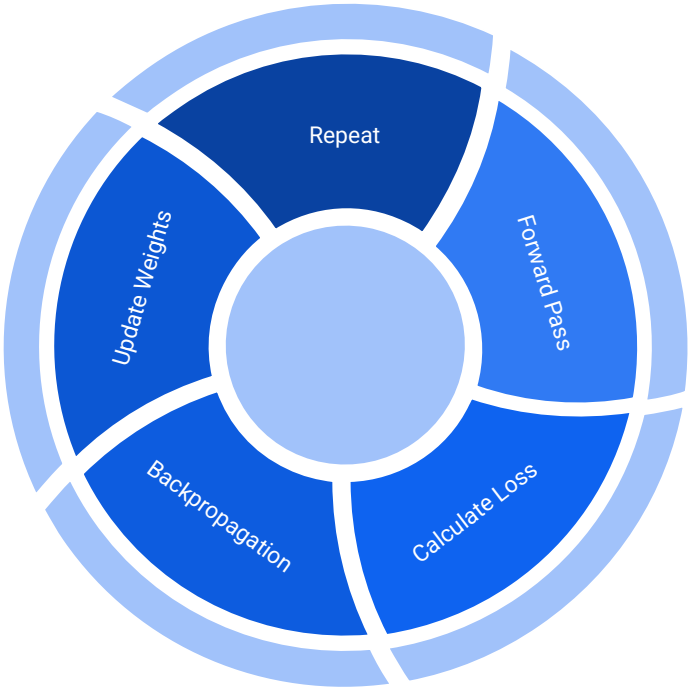
TRAINING PROCESS

How Neural Networks Learn



TRAINING PROCESS

How Neural Networks Learn



TRAINING PROCESS

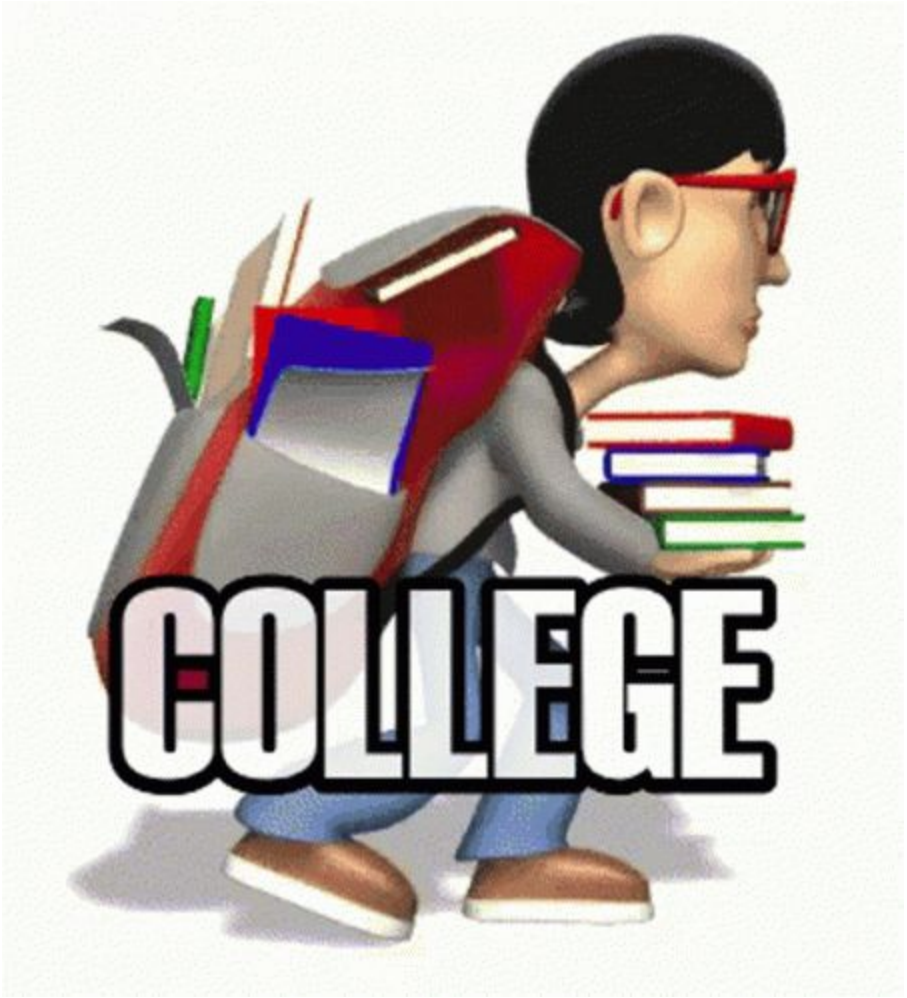
How Neural Networks Learn

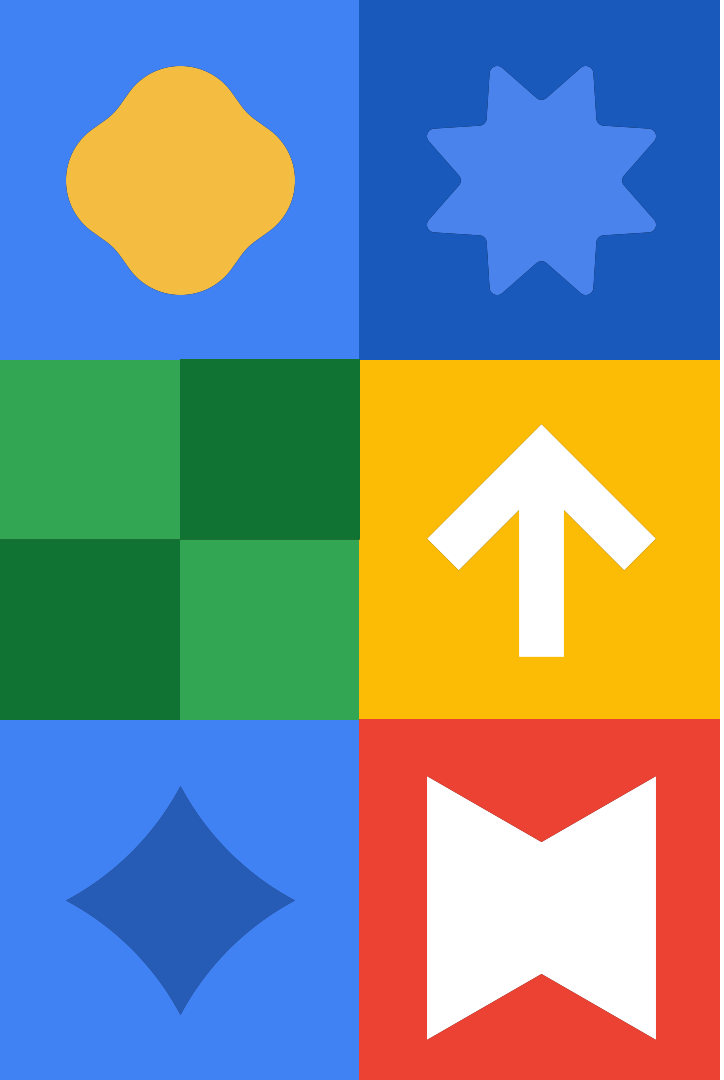
Epoch 1: Accuracy: 65%

Epoch 2: Accuracy: 78% — Learning!

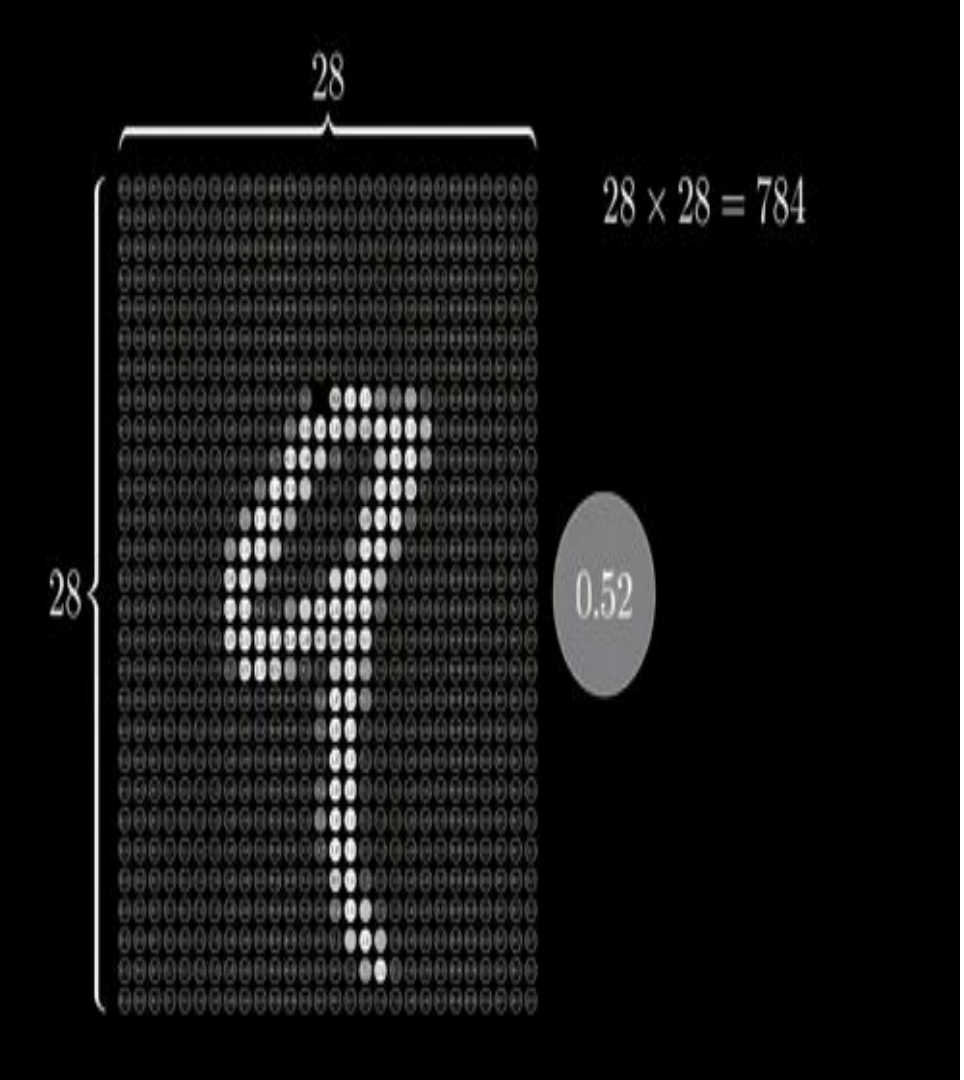
Epoch 3: Accuracy: 84%

Epoch 4: Accuracy: 89%


COLLEGE



Hands-On Project *Building a Real Neural Network*



PROJECT OVERVIEW

Handwritten Digit Recognition

Goal: Classify handwritten digits (0-9)

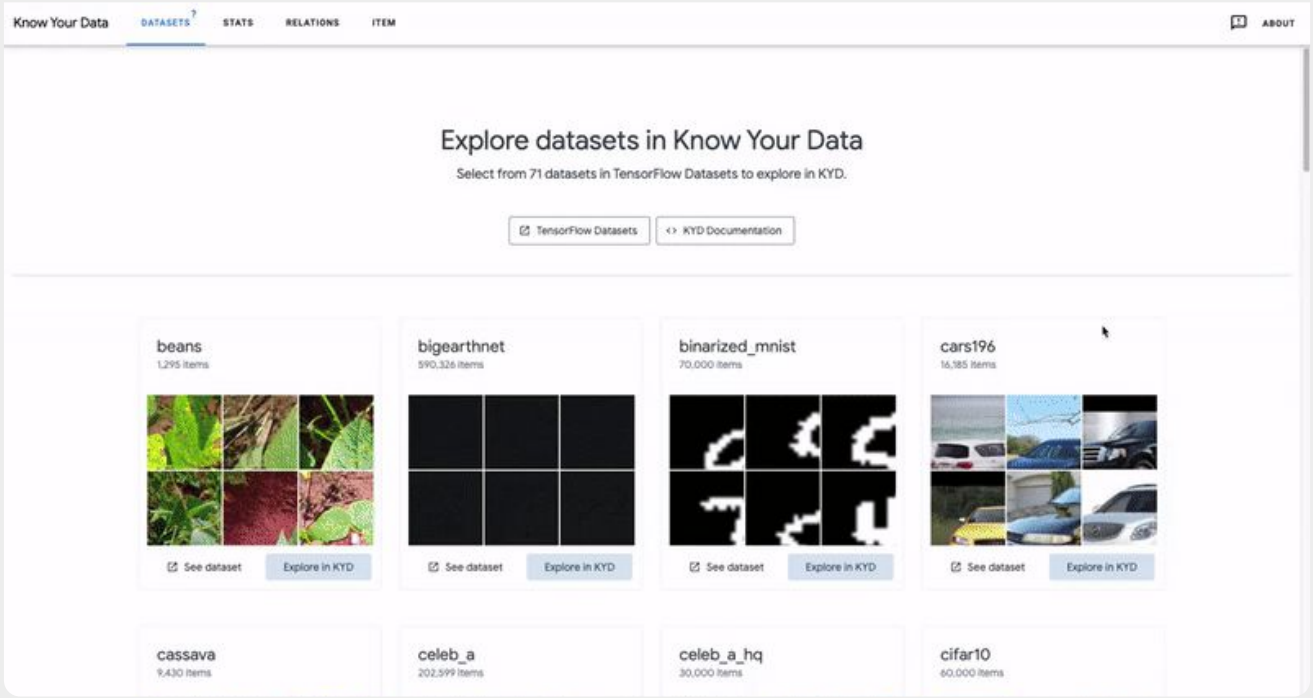
Dataset: MNIST (60,000 training images)

Input: 28×28 pixel grayscale images

Output: Digit class (0, 1, 2, ..., 9)

Success Metric: >87% accuracy

TensorFlow Datasets



kaggle

Search datasets

Filters

Computer Science | Education | Classification | Computer Vision | NLP | Data Visualization | Pre-Trained Model

Trending Datasets

See All

F1 Schedule 2022

Vivo Vinco · Updated 4 hours ago
Usability **9.4** · 2 kB
2 Files (CSV)

7

World Happiness Report 2022

Mathurin Aché · Updated 12 hours ago
Usability **10.0** · 2 MB
4 Files (other, CSV)

13

Vinello.eu Wine Catalogue

efimpolianskii · Updated 13 hours ago
Usability **9.4** · 42 kB
1 File (CSV)

5

Youtube Videos Dataset (~3400 videos)

Rajat Chaudhari · Updated 15 hours ago
Usability 8.8 · 661 kB
1 File (CSV)

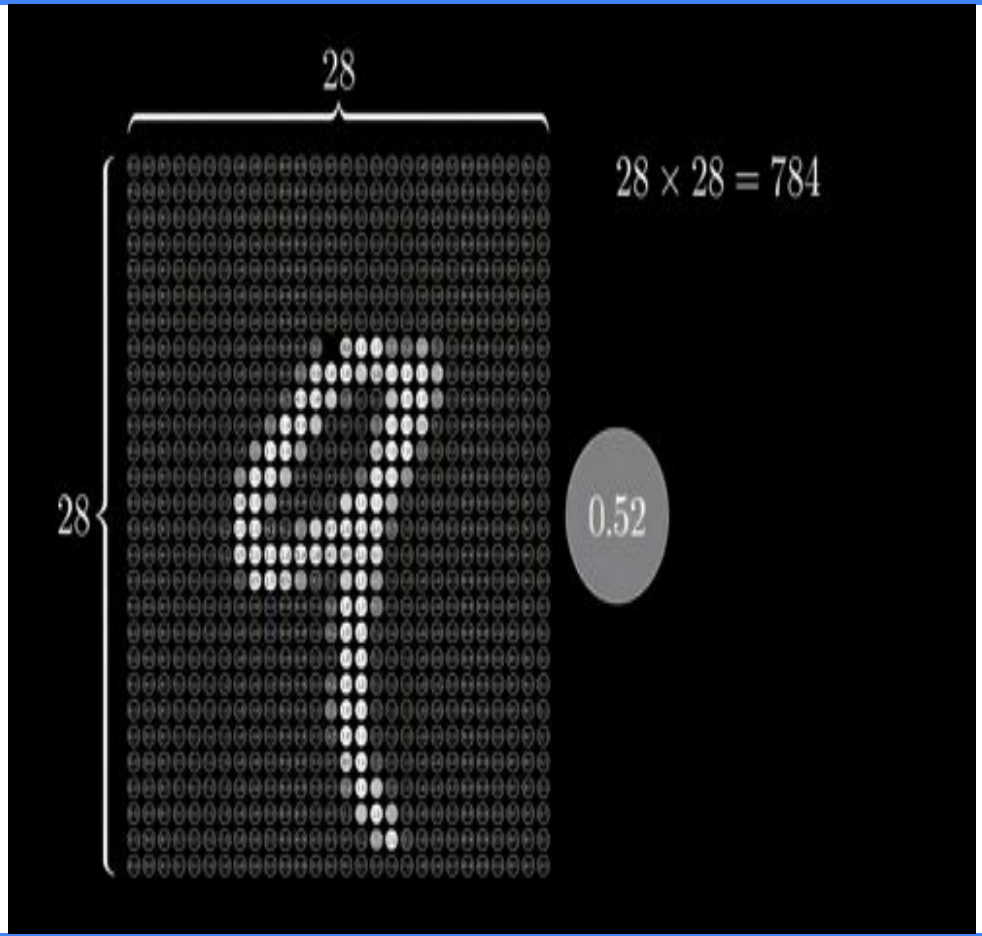
6

Popular Datasets

See All



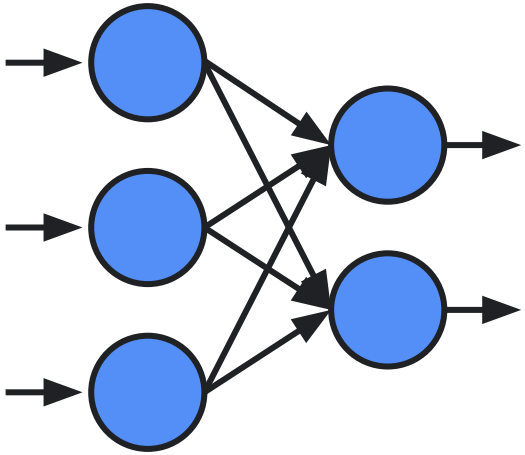

DATA EXPLORATION


28
28
28 × 28 = 784
0.52

```python
# Load and explore MNIST dataset
import tensorflow as tf

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

print(f"Training images: {x_train.shape}")  # (60000, 28, 28)
print(f"Training labels: {y_train.shape}")  # (60000,)
print(f"Label range: {y_train.min()} to {y_train.max()}")  # 0 to 9

# Normalize pixel values to 0-1 range
x_train = x_train / 255.0
x_test = x_test / 255.0
```

DATA EXPLORATION

Keras



```
model = tf.keras.Sequential([
  layers.Flatten(),
  layers.Dense(3, activation='relu'),
  layers.Dense(2)
])
```

```
# Build our neural network architecture
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),    # 784 features
    tf.keras.layers.Dense(128, activation='relu'),    # Hidden layer 1
    tf.keras.layers.Dropout(0.2),                     # Prevent overfitting
    tf.keras.layers.Dense(64, activation='relu'),     # Hidden layer 2
    tf.keras.layers.Dense(10, activation='softmax')   # Output layer
])

# Print model architecture
model.summary()
```

Building Model

```
# Configure training process
model.compile(
    optimizer='adam',                    # Smart learning rate adjustment
    loss='sparse_categorical_crossentropy',    # For multi-class classification
    metrics=['accuracy']                 # Track accuracy during training
)

# Train the model
history = model.fit(
    x_train, y_train,
    epochs=10,                # 10 complete passes through data
    validation_split=0.2,       # Use 20% of training data for validation
    batch_size=32,            # Process 32 images at once
    verbose=1                 # Show training progress
)
```

TRAINING CONFIGURATION

```
# Configure training process
model.compile(
    optimizer='adam',                    # Smart learning rate adjustment
    loss='sparse_categorical_crossentropy',    # For multi-class classification
    metrics=['accuracy']                 # Track accuracy during training
)

# Train the model
history = model.fit(
    x_train, y_train,
    epochs=10,                # 10 complete passes through data
    validation_split=0.2,       # Use 20% of training data for validation
    batch_size=32,            # Process 32 images at once
    verbose=1                 # Show training progress
)
```

TRAINING CONFIGURATION

LIVE DEMO TIME!

🚀 LIVE CODING DEMONSTRATION
Let's build this neural network together!

Demo Environment: Google Colab
Repository: Link
Follow along: Colab Notebook Link

LIVE DEMO TIME!

Training Results

Final Training Accuracy: 98.7%
Final Validation Accuracy: 97.2%
Test Accuracy: 96.8%

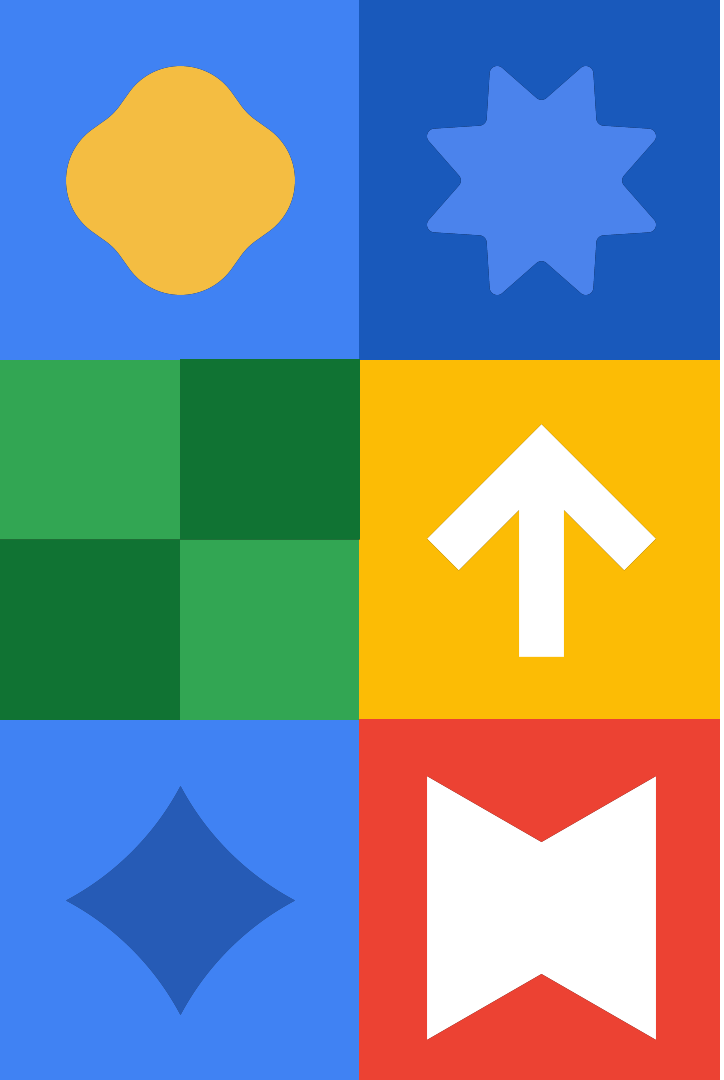
Model Performance:

- Epoch 1: 92.1% accuracy
- Epoch 5: 96.5% accuracy
- Epoch 10: 97.2% accuracy

Success!

```python
# Make predictions on new data
import matplotlib.pyplot as plt

# Get predictions for test set
predictions = model.predict(x_test)

# Visualize a prediction
def plot_prediction(image, prediction, true_label):
    plt.figure(figsize=(6, 3))
    plt.subplot(1, 2, 1)
    plt.imshow(image, cmap='gray')
    plt.title(f'True Label: {true_label}')
    plt.subplot(1, 2, 2)
    plt.bar(range(10), prediction)
    plt.title(f'Predicted: {np.argmax(prediction)}')
    plt.xlabel('Digit Class')
    plt.ylabel('Probability')

# Test on a random image
idx = 42
plot_prediction(x_test[idx], predictions[idx], y_test[idx])
```

MAKING PREDICTIONS



Beyond the Basics *Where to Go Next*

ADVANCED ARCHITECTURES

Next-Level Neural Networks

Convolutional Neural Networks (CNNs)

→ *Image recognition, computer vision*

Recurrent Neural Networks (RNNs)

→ *Text processing, language models*

Transformers

→ *Large language models, ChatGPT*

Generative Adversarial Networks (GANs)

→ *Image generation, deepfakes*

REAL-WORLD APPLICATIONS

Neural Networks in Action

Healthcare: Medical image analysis, drug discovery

Autonomous Vehicles: Object detection, path planning

Finance: Fraud detection, algorithmic trading

Entertainment: Game AI, content recommendation

Climate: Weather prediction, climate modeling

Education: Personalized learning, automated grading

Your AI Learning Journey

Month 1-2: Master the Fundamentals

- Linear algebra & calculus review
- Python programming proficiency
- Complete Andrew Ng's courses

Month 3-4: Specialized Architectures

- CNNs for computer vision
- RNNs for natural language processing
- Practice with real datasets

Month 5-6: Advanced Topics

- Transfer learning & fine-tuning
- Model optimization & deployment
- Contributing to open source projects

ESSENTIAL LEARNING RESOURCES

📚 **Courses:**
- Andrew Ng's Deep Learning Specialization (Coursera)
- MIT 6.034 Artificial Intelligence
- Google's Machine Learning Crash Course

📖 **Books:**
- "Deep Learning" by Ian Goodfellow
- "Hands-On Machine Learning" by Aurélien Géron
- "Neural Networks and Deep Learning" by Michael Nielsen

💻 **Practice:**
- Kaggle competitions and datasets
- Google Colab notebooks
- TensorFlow tutorials and documentation

CONNECT & CONTINUE LEARNING

🌐 Online Communities:

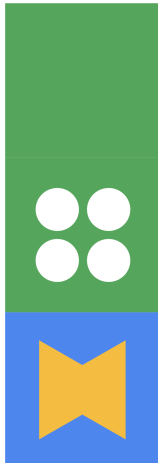
- ➔ *r/MachineLearning (Reddit)*
- ➔ *AI/ML Discord servers (e.g. Kaggle)*
- ➔ *TensorFlow Community Forum*
- ➔ *Join Offbeats!*

🏢 Local Groups:

- ➔ *AI/ML Meetups in your city*
- ➔ *University research groups*
- ➔ *Tech company events and workshops*

📱 Follow & Learn:

- ➔ *@AndrewYNg (Twitter)*
- ➔ *Google AI Blog*
- ➔ *Distill.pub for visual explanations*
- ➔ *Newsletters*



Gracias!
Questions?

Ayush Morbar

Founder, Offbeats
AI/ML Lead, GDGoC - RJIT

🐙 GitHub: @ayushmorbar

🔗 LinkedIn: @ayushmorbar

📱 Twitter/X: @ayush_morbar

BONUS - WHAT'S NEXT

Your Next Challenge

Build a Neural Network for:

🖼️ Image Classification (cats vs dogs)

📝 Text Analysis (sentiment classification)

📈 Time Series Prediction (stock prices)

🎮 Game AI (tic-tac-toe)

The Journey Continues...

Start Coding Today! 🚀


THANK YOU
dhanyavad
THANK YOU
THANK YOU
THANK YOU
ock 5 minutes before work ends:
THANK YOU