The Machine Learning Task Zoo 🦁

A Safari Through 40+ Real-World AI Problems

Nipun Batra · IIT Gandhinagar

What We'll Explore Today

```
┌──────────────────────────────────────────────────────────────────┐
│                    THE ML TASK SAFARI MAP                          │
├──────────────────────────────────────────────────────────────────┤
│                                                                    │
│    ◆ COMPUTER VISION           ◆ NATURAL LANGUAGE PROCESSING       │
│    Classification              Sentiment Analysis                  │
│    Detection                   Named Entity Recognition            │
│    Segmentation                Translation                         │
│    Pose Estimation             Summarization                       │
│    Depth Estimation            Question Answering                  │
│                                                                    │
│                                                                    │
│    ◆ AUDIO & SPEECH            ◆ GENERATIVE MODELS                 │
│    Speech-to-Text              Image Generation                    │
│    Text-to-Speech              Text Generation                     │
│    Speaker Recognition         Video Generation                    │
│                                                                    │
│                                                                    │
│    ◆ REINFORCEMENT             ◆ MULTIMODAL                        │
│    Game Playing                Visual QA                           │
│    Robot Control               Image Captioning                    │
│                                                                    │
└──────────────────────────────────────────────────────────────────┘
```
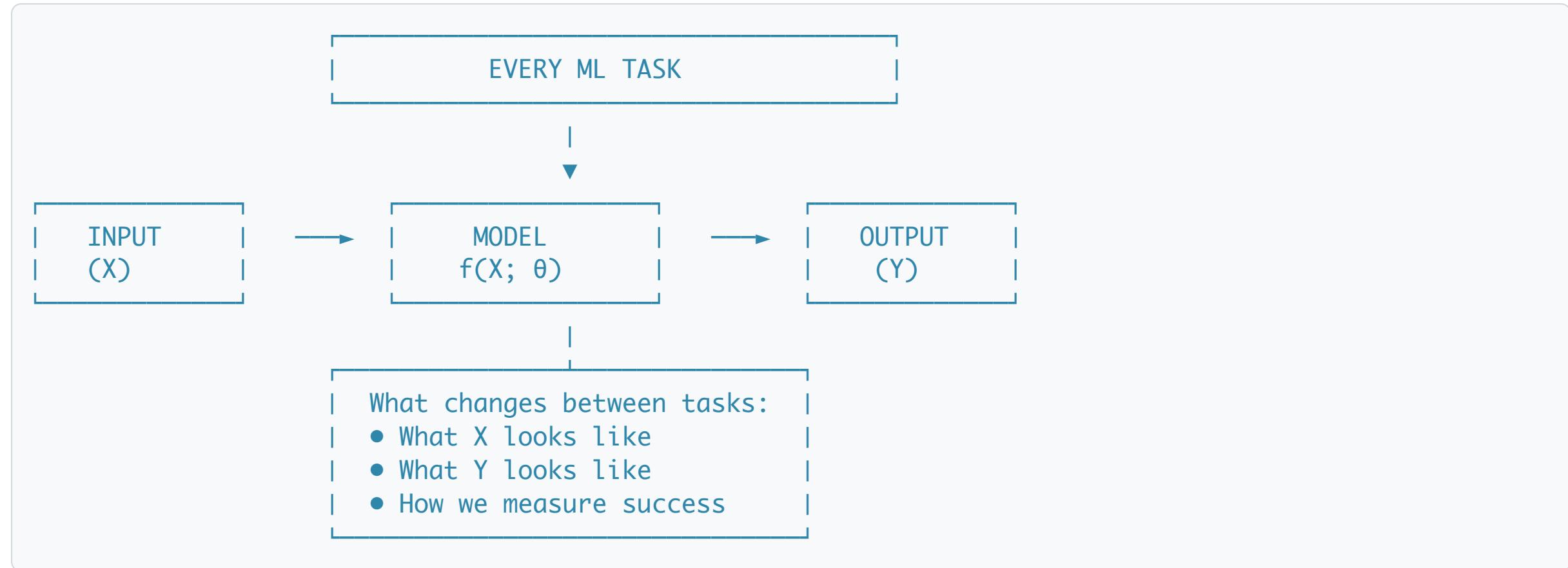
The Universal ML Recipe
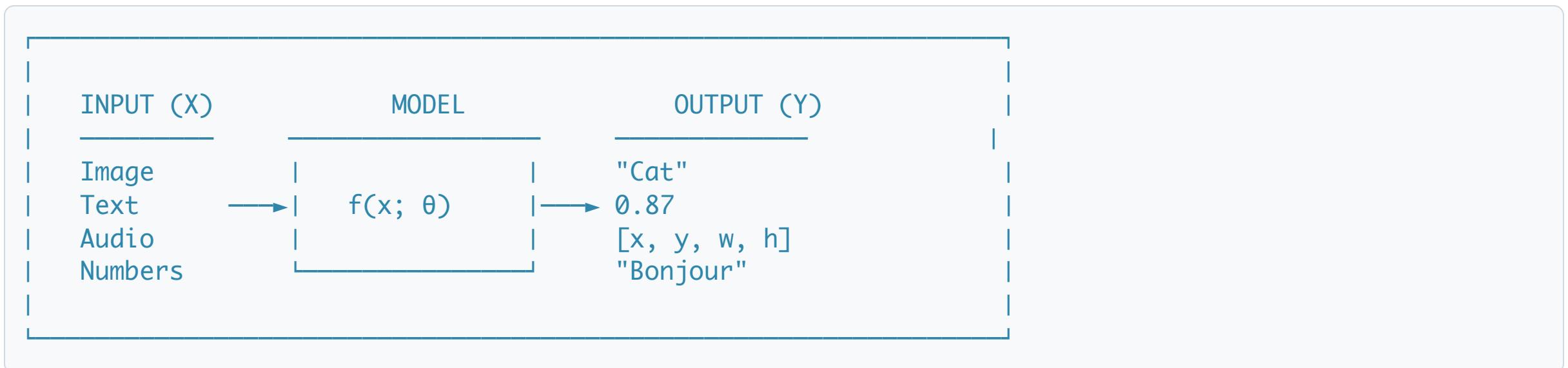
Before we dive into 40+ tasks, remember this simple pattern:

```
              ┌─────────────────────────────────┐
              │          EVERY ML TASK          │
              └─────────────────────────────────┘
                              │
                              ▼
┌─────────────┐      ┌─────────────────┐      ┌─────────────┐
│  INPUT      │ ───▶ │     MODEL       │ ───▶ │  OUTPUT     │
│  (X)        │      │     f(X; θ)     │      │   (Y)       │
└─────────────┘      └─────────────────┘      └─────────────┘
                              │
                    ┌─────────────────────────────┐
                    │  What changes between tasks: │
                    │  ● What X looks like         │
                    │  ● What Y looks like         │
                    │  ● How we measure success    │
                    └─────────────────────────────┘
```

The same Transformer architecture powers ChatGPT, DALL-E, and self-driving cars!

How to Think About ML Tasks

Every task is defined by **what goes in** and **what comes out**:

```
 ┌─────────────────────────────────────────────────────────┐
 │                                                           │
 │   INPUT (X)              MODEL              OUTPUT (Y)     │
 │   ─────────        ─────────────────        ───────────── │
 │                                                           │
 │   Image         |                  |     "Cat"            │
 │   Text      ───►|     f(x; θ)      |──►  0.87             │
 │   Audio         |                  |     [x, y, w, h]     │
 │   Numbers       └──────────────────┘     "Bonjour"        │
 │                                                           │
 └───────────────────────────────────────────────────────── ┘
```

The same model architecture can solve many different tasks — what changes is the data!

A Simple Classification

```
+----------------------------------------------------------------+
|                 ML TASKS BY INPUT/OUTPUT                       |
+----------------------------------------------------------------+
|                                                                |
|    INPUT TYPE            →          OUTPUT TYPE                |
|    _____                |
|                                                                |
|    Image                 →          Label       (Classification)|
|    Image                 →          Boxes       (Detection)    |
|    Image                 →          Pixel Labels (Segmentation) |
|    Image                 →          Text        (Captioning)   |
|    Text                  →          Label       (Sentiment)    |
|    Text                  →          Text        (Translation)  |
|    Audio                 →          Text        (Speech-to-Text)|
|    Text                  →          Audio       (Text-to-Speech)|
|    Numbers               →          Number      (Regression)   |
|    Numbers               →          Groups      (Clustering)   |
|    Noise                 →          Image       (Generation)   |
|    Text                  →          Image       (Text-to-Image)|
|    Game State            →          Action      (RL)           |
|                                                                |
+----------------------------------------------------------------+
```

5

◆ Domain 1: Computer Vision

Teaching Machines to See

"A picture is worth a thousand words... to a neural network, it's worth millions of numbers!"

The Vision Task Hierarchy

```
 _____
|                                                            |
| LEVEL 1: Classification    "There's a dog somewhere in this image" |
| _____ |
|                                                            |
|        Easiest: Just need the answer                       |
|                                                            |
| LEVEL 2: Detection         "There's a dog at position (x, y, w, h)" |
| _____ |
|                                                            |
|        Harder: Need to locate it with a box                |
|                                                            |
| LEVEL 3: Segmentation      "These exact pixels belong to the dog" |
| _____ |
|                                                            |
|        Even harder: Pixel-perfect boundaries               |
|                                                            |
| LEVEL 4: Pose Estimation   "Dog's head is at (x_1,y_1), legs at ..." |
| _____ |
|                                                            |
|        Hardest: Find specific body parts/keypoints         |
|                                                            |
|_____|

              ▲
              |
    More precision, more data, more compute needed
```
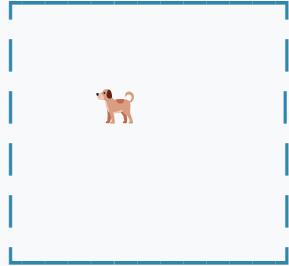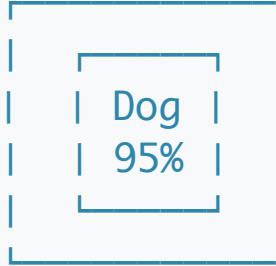
Let's See This Visually

```
SAME IMAGE, DIFFERENT TASKS:

 CLASSIFICATION        DETECTION          SEGMENTATION      POSE
 ─────────────         ─────────          ────────────      ────

                                          [segmentation     [pose
 [dog image]           | Dog |             mask]             keypoints]
                       | 95% |

 Output: "Dog"        Output:            Output:           Output:
                      [class, x,y,w,h]   Pixel mask        Keypoints
```
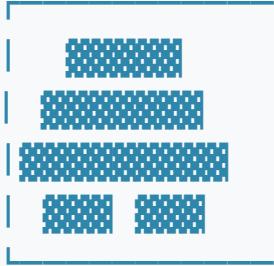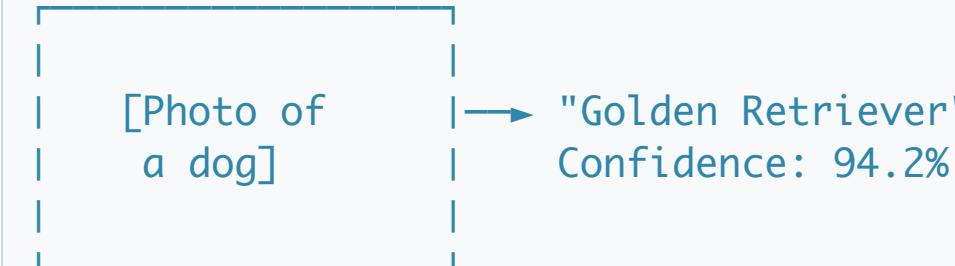
Task 1: Image Classification

What: Assign one label to an image.

```
 _____
|                   |
|   [Photo of       |——→   "Golden Retriever"
|    a dog]         |      Confidence: 94.2%
|                   |
|_____|
```

Real-world uses:

- 📱 Google Photos auto-tagging
- 🏥 Medical X-ray diagnosis
- 🏭 Quality control in factories
- 🌱 Plant disease detection

Example: MNIST Digits

```
Input:  28×28 grayscale image
Output: One of {0, 1, 2, ..., 9}




                           → "7" (98.5%)
```

10 classes, 60K training images

The "Hello World" of computer vision!

The Math Behind Classification

```
Input Image (e.g., 224 × 224 × 3 = 150,528 numbers)
                    |
                    ▼
          ┌─────────────────────┐
          |    Neural Network   |
          |    (millions of     |
          |     parameters)     |
          └─────────────────────┘
                    |
                    ▼
Raw scores (logits)
[2.5, -1.2, 8.7, 0.3, ...]  (one per class)
                    |
                    ▼
          ┌─────────────────────┐
          |      Softmax        | ← Converts to probabilities
          └─────────────────────┘
                    |
                    ▼
[0.01, 0.00, 0.94, 0.05, ...]  (sums to 1.0)
                    |
                    ▼
Prediction: Class 3 (94% confidence)
```

ImageNet: The Olympics of Image Classification

```
                        IMAGENET CHALLENGE

Dataset:  14 million images, 1000 classes

Classes include:
 ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌─────────┐
 │ 🐕      │  │ 🚗      │  │ 🎸      │  │ 🍕      │  │ 🏠      │
 │ 120 dog │  │ Cars    │  │ Musical │  │ Foods   │  │ Objects │
 │ breeds! │  │         │  │ instr.  │  │         │  │         │
 └─────────┘  └─────────┘  └─────────┘  └─────────┘  └─────────┘

Year     Winner          Top-5 Error    Note
─────────────────────────────────────────────────────────
2010     Traditional ML  28.2%          Hand-crafted features
2012     AlexNet (CNN)   16.4%          🔥 Deep learning begins!
2015     ResNet          3.6%           Superhuman performance!
2020     ViT             1.0%           Transformers enter vision
```

Multi-Label Classification

Sometimes one label isn't enough!
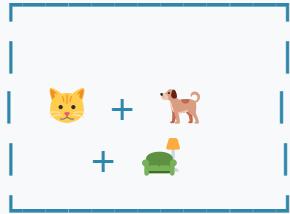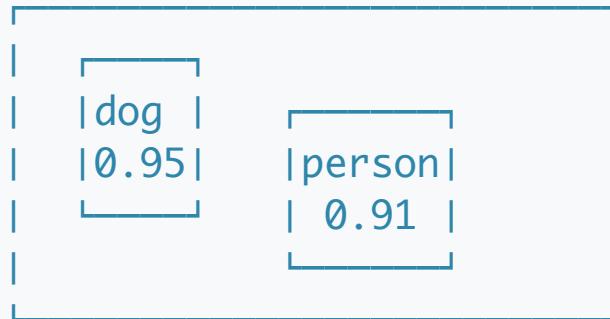
```
SINGLE-LABEL                    MULTI-LABEL
_____                    _____

   ┌──────────┐                     ┌──────────┐
   │          │                     │          │
   │   🐱     │ → "Cat"             │  🐱 + 🐕 │ → ["Cat", "Dog"]
   │          │   (one class)       │   + 🛋️   │    ["Couch"]
   └──────────┘                     └──────────┘    ["Indoors"]

Each image has                  Each image can have
exactly ONE label               MULTIPLE labels

Use: Softmax                    Use: Sigmoid (per class)
Σ probabilities = 1             Each class: 0 to 1 independently
```

Instagram uses multi-label classification for their photo tags and content moderation!

Task 2: Object Detection

What: Find objects AND locate them with boxes.

```
 ┌─────────────────────────────┐
 |   ┌──────┐                   |
 |   |dog   |                   |
 |   |0.95  |    ┌───────┐      |
 |   └──────┘    |person |      |
 |               | 0.91  |      |
 |               └───────┘      |
 └─────────────────────────────┘
```

Output for each detection:

- Class label ("dog")

- Confidence score (0.95)

- Bounding box: (x, y, width, height)

Example: Self-Driving Car

```
Detections in one frame:
├─ Car       at (120, 80)   conf: 0.97
├─ Car       at (400, 90)   conf: 0.89
├─ Person    at (300, 150)  conf: 0.92
├─ Bicycle   at (50, 200)   conf: 0.88
└─ Traffic   at (250, 20)   conf: 0.99
   Light

Must process 30+ frames/second!
```

Detection vs Classification: Key Differences

```
┌─────────────────────────────────────────────────────────┐
│              CLASSIFICATION vs DETECTION                  │
├───────────────────────────────────────────────────────────┤
│                                                           │
│  CLASSIFICATION              DETECTION                    │
│  ──────────────              ─────────                    │
│                                                           │
│  Input:  One image           Input:  One image           │
│  Output: One label           Output: List of (class, box)│
│                                                           │
│  Assumes: Object fills       Handles: Multiple objects,   │
│           most of image               any size, anywhere  │
│                                                           │
│  Architecture:               Architecture:                │
│  CNN → FC → Softmax          CNN → Multiple detection heads│
│                                                           │
│  Example:                    Example:                     │
│  "Is this a cat or dog?"     "Find all cats and dogs"     │
│                                                           │
│  Popular Models:             Popular Models:              │
│  ResNet, EfficientNet        YOLO, Faster R-CNN, DETR     │
│                                                           │
└───────────────────────────────────────────────────────────┘
```
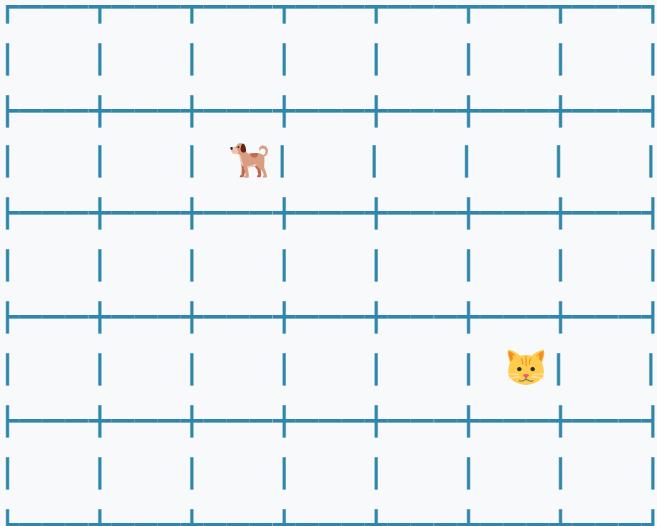
How YOLO Works (Simplified)

"You Only Look Once" - Fast single-pass detection

Step 1: Divide image into grid (e.g., 7×7)
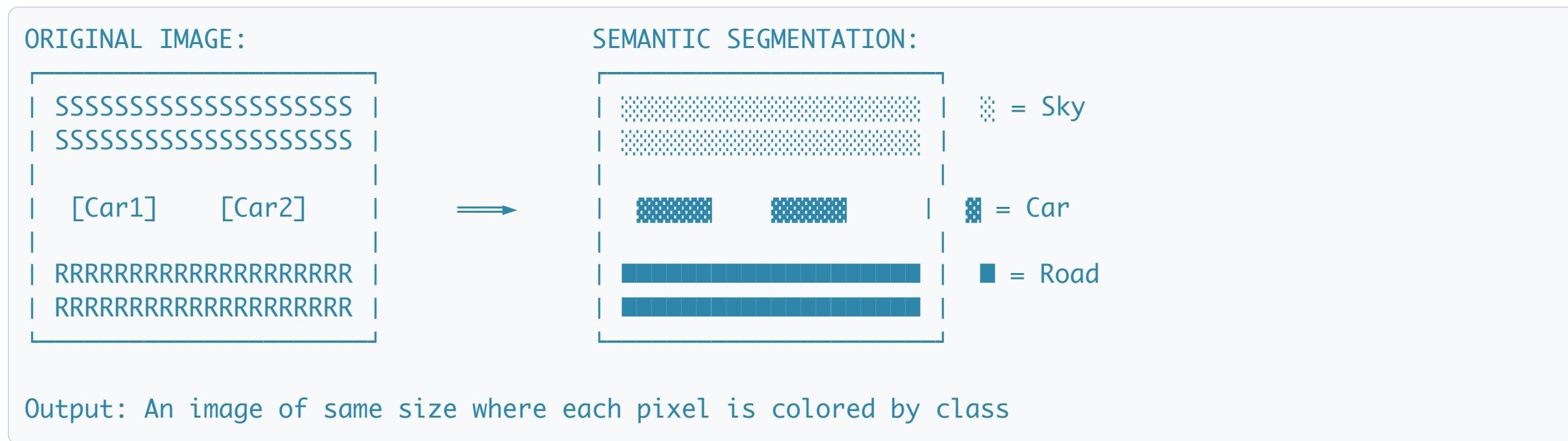
Each cell predicts:
- B bounding boxes
- Confidence scores
- C class probabilities

Step 2: Remove overlapping boxes (Non-Max Suppression)
Step 3: Output final detections

Task 3: Semantic Segmentation

What: Label every pixel with its class.

```
ORIGINAL IMAGE:                          SEMANTIC SEGMENTATION:

┌─────────────────────┐                 ┌─────────────────────┐
| SSSSSSSSSSSSSSSSSSSS |                 | ▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨ |    ▨ = Sky
| SSSSSSSSSSSSSSSSSSSS |                 | ▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨ |
|                     |                 |                     |
|  [Car1]    [Car2]   |     ⟹           |   ▨▨▨▨▨   ▨▨▨▨▨     |    ▨ = Car
|                     |                 |                     |
| RRRRRRRRRRRRRRRRRRRR |                 | ██████████████████  |    ■ = Road
| RRRRRRRRRRRRRRRRRRRR |                 | ██████████████████  |
└─────────────────────┘                 └─────────────────────┘

Output: An image of same size where each pixel is colored by class
```
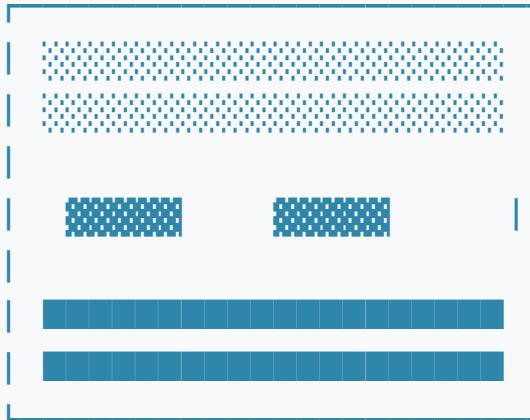
Both cars have the same color — semantic segmentation doesn't distinguish between instances of the same class!

16
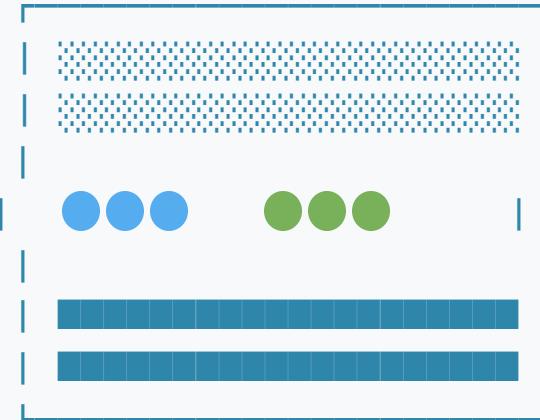
Task 4: Instance Segmentation

What: Label every pixel AND distinguish individual objects.

SEMANTIC SEGMENTATION:

INSTANCE SEGMENTATION:

vs

Both cars = same "Car" color
Can't tell them apart!

Car #1 = Blue, Car #2 = Green
Can track each car individually

Self-driving cars need instance segmentation — you must track which car is which to predict their movements!

Panoptic Segmentation: The Complete Picture

Combining everything: Semantic + Instance

```
                    PANOPTIC SEGMENTATION

    Sky (stuff - no instances)


        Car #1        Person #1        Car #2
        ●●●             ●●              ●●


    Road (stuff - no instances)
    ████████████████████████████████████████████


    "Stuff" classes: sky, road, grass (don't count instances)
    "Things" classes: cars, people (each instance gets unique ID)
```
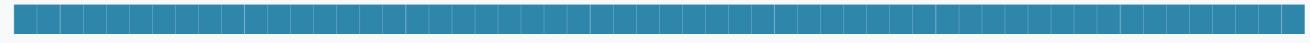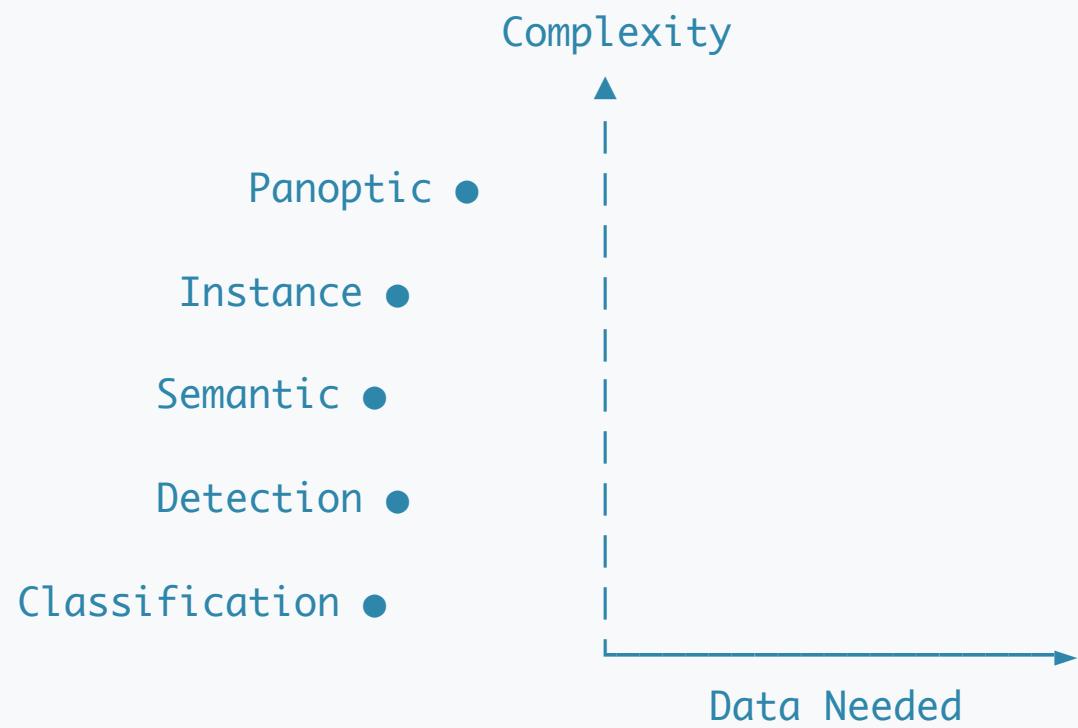
Segmentation Summary
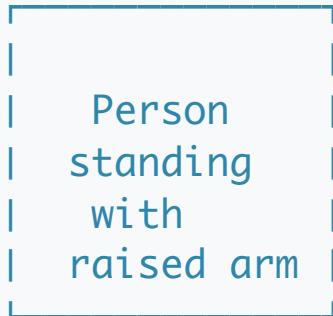
Task	What it outputs	Can count objects?	Use case
Semantic	Pixel classes	No	Land use mapping, medical imaging
Instance	Pixel + instance IDs	Yes	Object tracking, robotics
Panoptic	Both combined	Yes for "things"	Autonomous driving

```
                              Complexity
                                  ▲
                                  |
              Panoptic ●          |
                                  |
            Instance ●            |
                                  |
          Semantic ●              |
                                  |
         Detection ●              |
                                  |
     Classification ●             |
                                  └──────────────────────▶
                                         Data Needed
```
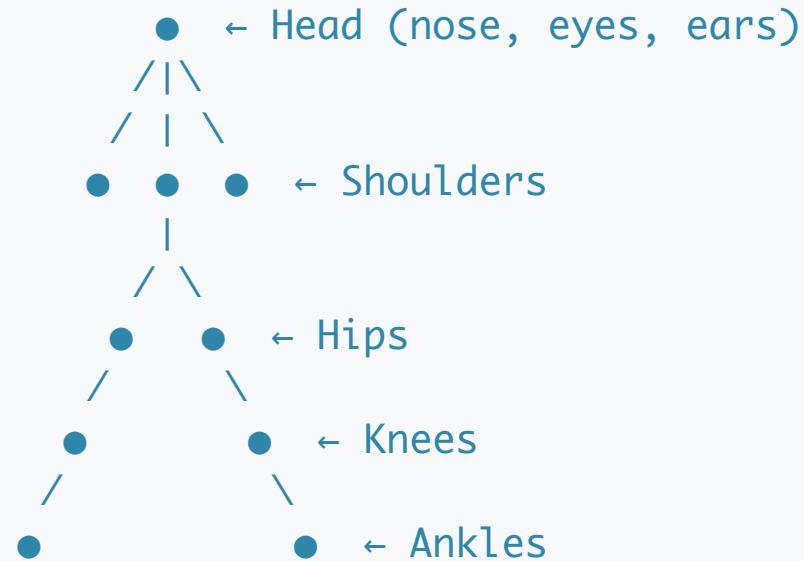
Task 5: Pose Estimation

What: Find body keypoints (skeleton) of humans or animals.

```
Original Photo:                         Detected Skeleton:

 _____                                    ● ← Head (nose, eyes, ears)
|           |                                  /|\
|  Person   |                                 / | \
| standing  |        ────→           ●  ●  ● ← Shoulders
|   with    |                                   |
| raised arm|                                  / \
|_____|                             ●    ● ← Hips
                                         /       \
                                        ●         ● ← Knees
                                       /           \
                                      ●             ● ← Ankles

Output: 17 keypoints with (x, y, confidence) each
```

Pose Estimation: Real Applications

Fitness & Sports

```
 _____
|                           |
|  Squat Form               |
|  Analysis                 |
|                           |
|            ●              |
|           /|\             |
|          / | \            |
|            |              |
|           /\              |
|          /  \             |
|         ●    ●            |
|                           |
|         ↑    ↑            |
|  "Knees too far           |
|   over toes!"             |
|_____|
```

Other Applications

- 🎬 Motion capture for movies/games

- 🏃 Running form analysis

- 🕺 Dance move recognition

- 🤟 Sign language interpretation

- 🎮 Controller-free gaming (Kinect)

- 🛡️ Fall detection for elderly

**Apple Fitness+** uses pose estimation to analyze your workout form in real-time!
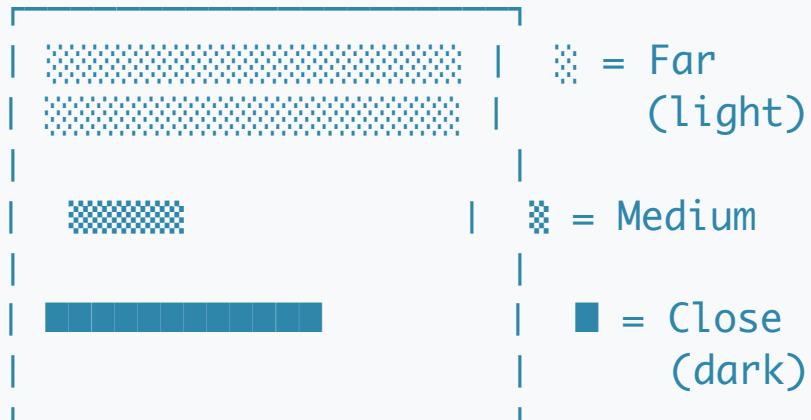
Task 6: Depth Estimation

What: Predict distance of each pixel from the camera.

```
RGB Image:                        Depth Map:

┌──────────────────┐              ┌──────────────────┐
│                  │              │ ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ │    ▓ = Far
│   🏔 (mountains)  │              │ ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ │        (light)
│                  │              │                  │
│  🌳 (trees)       │     ──→      │  ▓▓▓▓▓▓           │    ▓ = Medium
│                  │              │                  │
│  🚗 (close car)   │              │  ▇▇▇▇▇▇▇▇▇▇▇     │    ▇ = Close
│                  │              │                  │        (dark)
└──────────────────┘              └──────────────────┘

Output: Same-size image where pixel intensity = distance
```

One camera, 3D understanding! Traditional 3D sensing requires special hardware (LiDAR, stereo cameras), but AI can estimate depth from a single RGB image.

Depth Estimation: How It's Used

AR/VR

Place virtual
furniture in
your room!

🛋️

(knows it's
on the floor)

Portrait Mode

Blur background
based on depth

👤 (sharp)
░░░ (blurred)

Robotics

Navigate without
bumping into
objects

🤖 → 📦
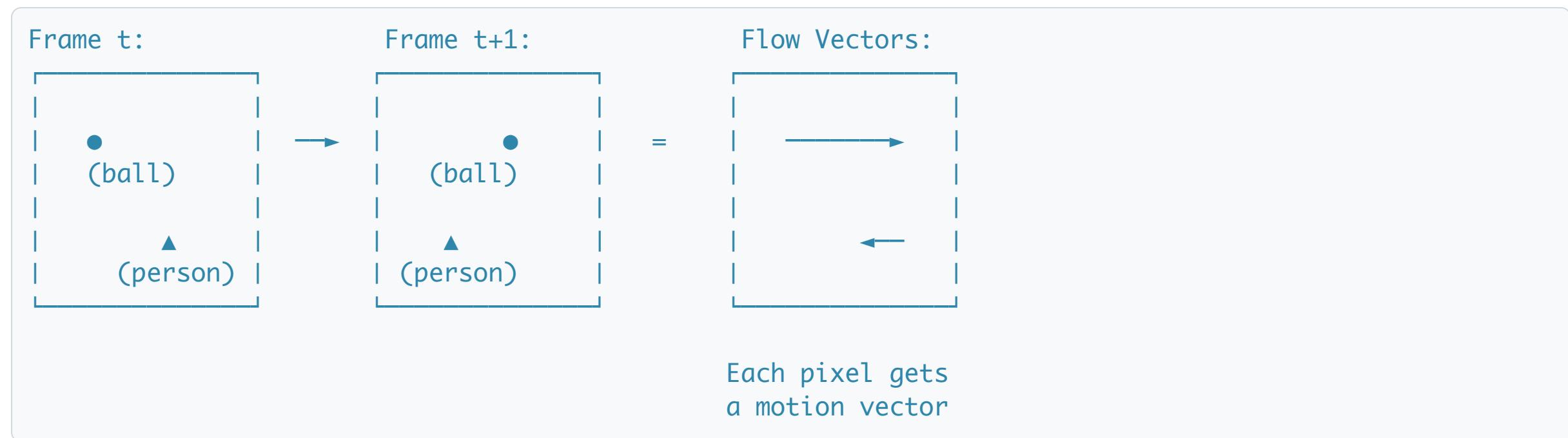
(knows distance)

The iPhone's Portrait Mode uses a combination of depth sensors AND neural network depth estimation!

Task 7: Optical Flow

What: Track how each pixel moves between video frames.

```
Frame t:                Frame t+1:              Flow Vectors:

 _____         _____         _____
|               |       |               |       |               |
|   ●           |  ──>   |       ●       |   =   |   ──────────> |
|  (ball)       |       |     (ball)     |       |               |
|               |       |               |       |               |
|      ▲        |       |    ▲          |       |       <────    |
|  (person)     |       |  (person)     |       |               |
 ───────────────         ───────────────         ───────────────

                                                 Each pixel gets
                                                 a motion vector
```

Key insight: Every pixel gets a (dx, dy) vector showing where it moved!

Optical Flow Applications

```
┌─────────────────────────────────────────────────────┐
│                OPTICAL FLOW USE CASES                 │
├─────────────────────────────────────────────────────┤
│                                                       │
│  🎬 VIDEO COMPRESSION                                 │
│  Instead of storing every frame, store keyframes + motion vectors │
│  Result: 10x smaller file sizes                       │
│                                                       │
│  🏃 ACTION RECOGNITION                                │
│  "Running" = specific pattern of flow vectors         │
│  "Waving" = different pattern                         │
│                                                       │
│  🚗 AUTONOMOUS DRIVING                                │
│  Objects moving towards you → collision warning       │
│  Everything moving left → you're turning right        │
│                                                       │
│  🎮 VIDEO GAMES                                       │
│  Frame interpolation: turn 30fps into 60fps           │
│                                                       │
└─────────────────────────────────────────────────────┘
```
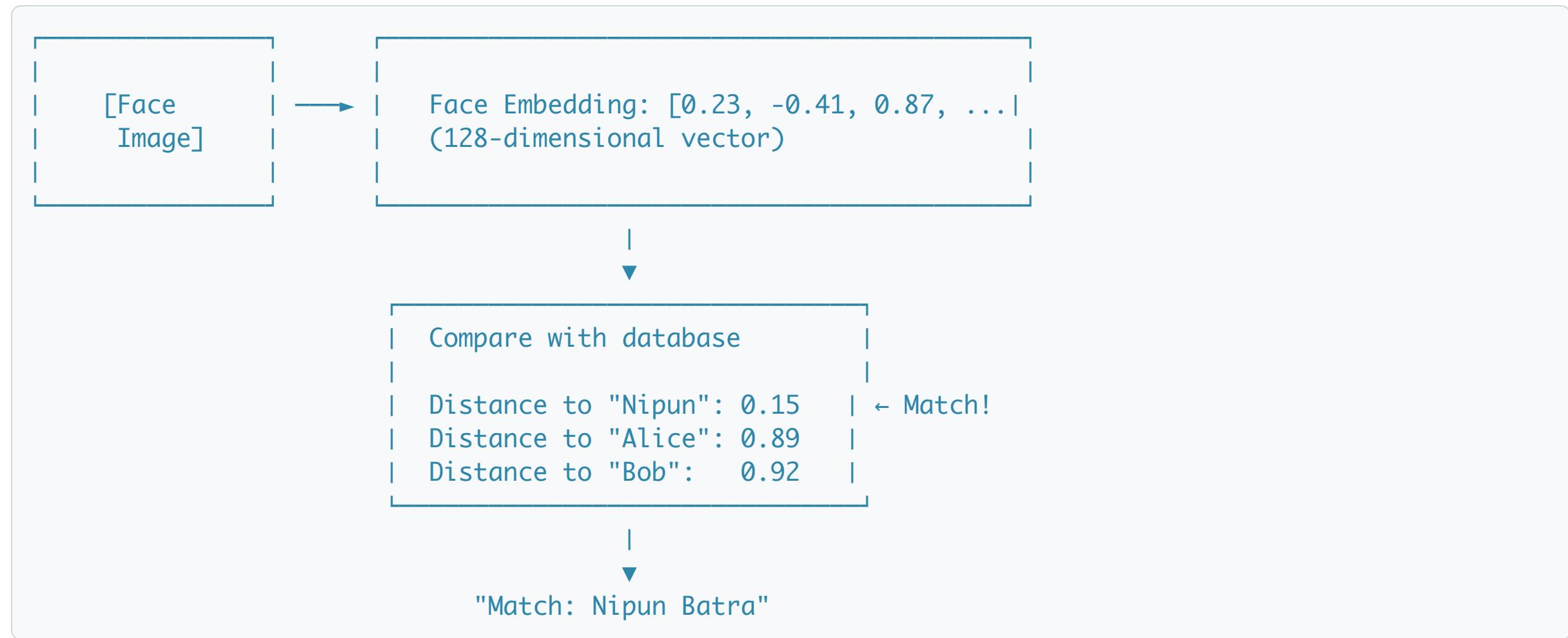
Task 8: Face Recognition
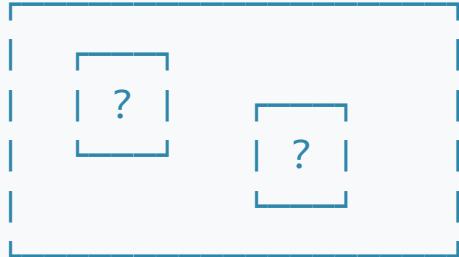
What: Identify WHO a face belongs to.

```
+-----------+        +-------------------------------------+
|           |        |                                     |
| [Face     |  --->  | Face Embedding: [0.23, -0.41, 0.87, ...|
|  Image]   |        | (128-dimensional vector)            |
|           |        |                                     |
+-----------+        +-------------------------------------+
                                        |
                                        ▼
                     +-----------------------------------+
                     | Compare with database             |
                     |                                   |
                     | Distance to "Nipun": 0.15   | ← Match!
                     | Distance to "Alice": 0.89   |
                     | Distance to "Bob":   0.92   |
                     +-----------------------------------+
                                        |
                                        ▼
                            "Match: Nipun Batra"
```

Face Detection ≠ Face Recognition

```
FACE DETECTION                    FACE RECOGNITION
_____                    _____


Question: "Where are            Question: "Who is
          the faces?"                      this person?"


  ┌─────────────┐                  ┌─────────────┐
  │ ┌───┐       │                  │    ┌─[👤]─┐  │         │
  │ │ ? │ ┌───┐ │                  │    └─────┘  │         │
  │ └───┘ │ ? │ │                  │      ↓      │         │
  │       └───┘ │                  │  "This is Nipun" │
  └─────────────┘                  └─────────────┘


Output: Bounding boxes          Output: Identity label

Used BEFORE recognition         Requires database of
(find faces first)              known faces
```

Vision Tasks Summary

Task	Input	Output	Example Use
Classification	Image	Label	"Is this spam?"
Detection	Image	Boxes + labels	Self-driving cars
Segmentation	Image	Pixel mask	Medical imaging
Pose Estimation	Image	Keypoints	Fitness apps
Depth Estimation	Image	Depth map	AR furniture
Optical Flow	2 frames	Motion vectors	Video compression
Face Recognition	Face	Identity	Phone unlock

◆ Domain 2: Natural Language Processing

Teaching Machines to Read & Write

"Language is the dress of thought." — Samuel Johnson

The NLP Task Landscape

```
┌────────────────────────────────────────────────────────┐
│                        NLP TASKS                         │
├────────────────────────────────────────────────────────┤
│                                                          │
│  UNDERSTANDING TASKS         GENERATION TASKS            │
│  ─────────────────           ────────────────           │
│                                                          │
│  • Sentiment Analysis        • Text Generation           │
│    (Is this positive?)         (Write like Shakespeare)  │
│                                                          │
│  • Named Entity Recognition  • Summarization             │
│    (Find names, places, dates) (Shorten this article)    │
│                                                          │
│  • Question Answering        • Translation               │
│    (Find the answer)           (English → Hindi)         │
│                                                          │
│  • Topic Classification      • Paraphrasing              │
│    (Sports? Politics? Tech?)   (Same meaning, new words)  │
│                                                          │
│  ──────────────────────────────────────────────         │
│                                                          │
│  MODERN LLMS CAN DO ALL OF THESE WITH A SINGLE MODEL!    │
│                                                          │
└────────────────────────────────────────────────────────┘
```

Task 9: Sentiment Analysis

What: Classify text by emotion/opinion.

```
 _____
| "This movie was absolutely  |
|  amazing! Best film of the  |
|  year. A masterpiece!"      |
 _____
             |
             ▼
       _____
      | POSITIVE  |
      |  (0.96)   |
       _____
```

Output Options:

```
Binary:      Positive / Negative

3-class:     Positive / Neutral /
             Negative

5-class:     ★ to ★★★★★

Continuous: -1.0 to +1.0
             (very negative →
              very positive)
```

Sentiment Analysis: Real World

Brand Monitoring

```
Twitter Stream:
├─ "Love the new iPhone!"  →  😊
├─ "Battery dies so fast"  →  😠
├─ "Just bought one!"      →  😐
├─ "Worst purchase ever"   →  😠
└─ "Camera is incredible"  →  😊


Daily Sentiment: 67% positive
```

Customer Feedback

```
Support Tickets:
┌─────────────────────────────┐
│ Urgent (Negative) ●●●●  │
│ Normal  (Neutral) ●●     │
│ Praise  (Positive) ●      │
└─────────────────────────────┘

→ Route angry customers
  to senior support!
```

Amazon analyzes millions of reviews using sentiment analysis to understand product reception!

The Challenge: Sarcasm & Context

```
                    WHY SENTIMENT IS HARD

  SARCASM:
  "Oh great, another software update. Just what I needed."
   Words: positive ("great", "needed")
   Meaning: NEGATIVE

  NEGATION:
  "This movie is not bad."
   Contains "bad" → but overall POSITIVE

  CONTEXT:
  "The battery lasts forever" (phone review) → POSITIVE
  "This movie lasts forever" (movie review) → NEGATIVE

  MIXED:
  "The food was great but the service was terrible."
   Overall? Positive? Negative? Neutral? Depends on priority!
```
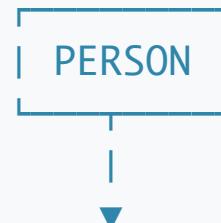
Task 10: Named Entity Recognition (NER)

What: Find and label names, places, dates, organizations, etc.

```
Input:   "Elon Musk announced that Tesla will open a factory
          in Berlin by March 2025."


            ┌─────────┐                    ┌─────────┐
            | PERSON  |                    |   ORG   |
            └─────────┘                    └─────────┘
                 |                              |
                 ▼                              ▼
Output: "Elon Musk announced that Tesla will open a factory


                                    ┌─────────┐ ┌─────────┐
                                    |   LOC   | |  DATE   |
                                    └─────────┘ └─────────┘
                                         |           |
                                         ▼           ▼
          in Berlin by March 2025."
```

NER: Entity Types

```
+--------------------------------------------------------------------+
|                    COMMON NER ENTITY TYPES                         |
+--------------------------------------------------------------------+
|                                                                    |
|   TYPE              EXAMPLES               COLOR CODE               |
|   ————              ————————               ——————————              |
|                                                                    |
|   PERSON            Elon Musk, Marie Curie    ■ Blue               |
|   ORGANIZATION      Tesla, Google, UN         ■ Green              |
|   LOCATION          Berlin, Mount Everest     ■ Yellow             |
|   DATE              March 2025, last Tuesday  ■ Orange             |
|   MONEY             $5 million, €100          ■ Brown              |
|   PERCENT           15%, three percent        ■ White              |
|   TIME              3:30 PM, midnight         ■ Purple             |
|   PRODUCT           iPhone 15, Model S        ■ Red                |
|                                                                    |
|   Domain-specific:                                                 |
|   - Medical: DISEASE, DRUG, SYMPTOM                                |
|   - Legal: CASE_NUMBER, COURT, JUDGE                               |
|   - Finance: TICKER, EXCHANGE, CURRENCY                            |
|                                                                    |
+--------------------------------------------------------------------+
```

35

NER: Real Applications

Search Engines

```
Query: "restaurants near
        Eiffel Tower"

NER finds:
└ LOCATION: "Eiffel Tower"

→ Shows map of Paris
→ Lists nearby restaurants
```
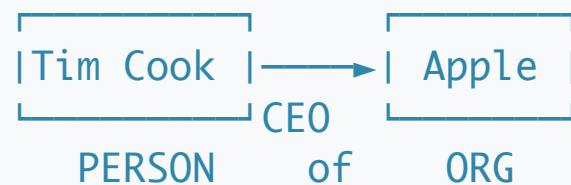
Knowledge Graphs

```
Text: "Tim Cook is the
       CEO of Apple"

Extracted:
      ┌─────────┐         ┌───────┐
      |Tim Cook |───────→ | Apple |
      └─────────┘CEO      └───────┘
       PERSON    of        ORG
```
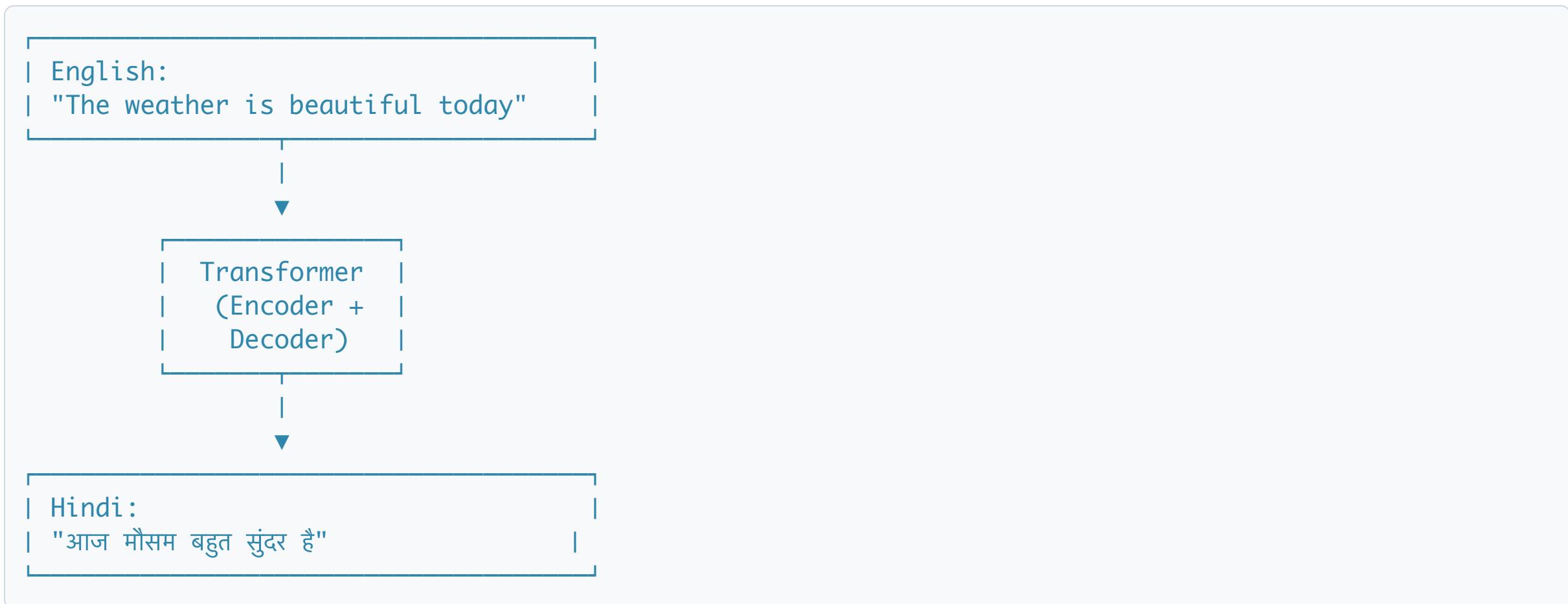
Google Search uses NER to understand your queries and build its Knowledge Graph!

Task 11: Machine Translation

What: Convert text from one language to another.

```
+-------------------------------------+
| English:                            |
| "The weather is beautiful today"    |
+-------------------------------------+
                  |
                  ▼
        +-------------------+
        |    Transformer    |
        |    (Encoder +     |
        |     Decoder)      |
        +-------------------+
                  |
                  ▼
+-------------------------------------+
| Hindi:                              |
| "आज मौसम बहुत सुंदर है"              |
+-------------------------------------+
```

Translation Challenges

```
┌─────────────────────────────────────────────────────────┐
│                  WHY TRANSLATION IS HARD                  │
├─────────────────────────────────────────────────────────┤
│                                                           │
│ WORD ORDER:                                               │
│ English: "I eat an apple"     (Subject-Verb-Object)       │
│ Japanese: " はりんごを べる"   (Subject-Object-Verb)      │
│                                                           │
│ IDIOMS:                                                   │
│ "It's raining cats and dogs" → Not about animals!         │
│ Must translate the MEANING, not words                     │
│                                                           │
│ CONTEXT:                                                  │
│ "The bank is by the river"                                │
│ - bank = financial institution? river bank?              │
│                                                           │
│ CULTURAL CONCEPTS:                                        │
│ Some words have no direct translation                     │
│ - Japanese "  れ " (komorebi): sunlight filtering through trees │
│                                                           │
│ GENDER/FORMALITY:                                         │
│ "You" in English = tu/vous in French (formal vs informal) │
│                                                           │
└─────────────────────────────────────────────────────────┘
```

Task 12: Text Summarization

Extractive: Pick important sentences verbatim.

Abstractive: Generate new text (paraphrase).

```
Long Article:

┌─────────────────────┐
| Sentence 1          |  ← Keep
| Sentence 2          |
| Sentence 3          |
| Sentence 4          |  ← Keep
| Sentence 5          |
| Sentence 6          |  ← Keep
| ...                 |
└─────────────────────┘


Just highlights!
No new words created.
```

```
Long Article:

┌─────────────────────┐
| [Original 1000      |
|   words about       |
|   climate change...]|
└─────────────────────┘
             | AI rewrites
             ▼
┌─────────────────────┐
| "Climate scientists |
|   warn that global  |
|   temperatures..."  |
|   (100 words)       |
└─────────────────────┘
```

LLMs like GPT-4 and Claude do **abstractive** summarization — they truly understand and paraphrase!

Task 13: Question Answering

Extractive QA:

Find answer span in given text.

```
Context: "Albert Einstein
was born in Ulm, Germany
on March 14, 1879."

Question: "Where was
Einstein born?"

Answer: "Ulm, Germany"
          ▲
          └── Highlighted from
              the context text
```

Generative QA:

Generate free-form answer.

```
Question: "Explain
quantum entanglement
to a 5-year-old."

Answer: "Imagine you have
two magic coins. When
you flip one and it
lands on heads, the
other one ALWAYS lands
on heads too, even if
it's on the moon!"
              ▲
              └── Created new text
                  (not from any doc)
```

QA: The Evolution

```
┌──────────────────────────────────────────────────────────┐
│                QUESTION ANSWERING EVOLUTION                │
├──────────────────────────────────────────────────────────┤
│                                                            │
│   ERA 1: Rule-Based (1960s-2000s)                          │
│   ─────────────────────────────                            │
│                                                            │
│   Keywords → Database lookup → Template answer             │
│   "Very brittle, only worked for specific domains"         │
│                                                            │
│   ERA 2: Extractive (2016-2020)                            │
│   ─────────────────────────────                            │
│                                                            │
│   BERT-style: "Find the answer IN the text"                │
│   Great for reading comprehension tasks                    │
│                                                            │
│   ERA 3: Generative (2020-present)                         │
│   ─────────────────────────────                            │
│                                                            │
│   LLMs: Generate answers from learned knowledge            │
│   Can answer questions about ANYTHING                      │
│   Can reason, explain, and elaborate                       │
│                                                            │
│   ERA 4: RAG (Retrieval-Augmented Generation)              │
│   ───────────────────────────────────────                 │
│                                                            │
│   LLM + Search = Best of both worlds                       │
│   Accurate, up-to-date, with sources                       │
│                                                            │
└──────────────────────────────────────────────────────────┘
```

Task 14: Text Generation (LLMs)

What: Predict and generate the next tokens, one at a time.

```
Prompt:   "The secret to happiness is"
                |
                ▼
         ┌─────────────────┐
         |       LLM       |
         |   (GPT, Claude,  |
         |    Gemini, etc.) |
         └─────────────────┘
                |
                ▼
Token 1: "not"      P("not" | prompt) = 0.15
                |
                ▼
Token 2: "in"       P("in" | prompt + "not") = 0.42
                |
                ▼
Token 3: "wealth"  P("wealth" | ... + "in") = 0.23
                |
                ▼
         ... continues until <END> token

Output: "The secret to happiness is not in wealth but in
         meaningful connections with others."
```

42
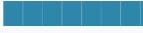
How LLMs Generate Text

```
┌─────────────────────────────────────────────────────┐
│                AUTOREGRESSIVE GENERATION              │
├─────────────────────────────────────────────────────┤
│                                                       │
│  Step 1: "The"        → predict next → "secret"       │
│  Step 2: "The secret" → predict next → "to"           │
│  Step 3: "The secret to" → predict next → "happiness" │
│  ...                                                  │
│                                                       │
│  Each step:                                           │
│  ┌─────────────────────────────────────────┐         │
│  │  Probability Distribution Over Vocabulary │        │
│  │                                           │        │
│  │  P("the")      = 0.02  ░                  │        │
│  │  P("happiness")= 0.25  ████████           │        │
│  │  P("success")  = 0.18  █████              │        │
│  │  P("life")     = 0.12  ███                │        │
│  │  P("love")     = 0.08  ██                 │        │
│  │  ...           = ...                      │        │
│  └─────────────────────────────────────────┘         │
│                                                       │
│  Sample from this distribution (or take argmax)       │
│                                                       │
└─────────────────────────────────────────────────────┘
```

NLP Tasks Summary

Task	Input	Output	Key Challenge
Sentiment	Text	Positive/Negative	Sarcasm, context
NER	Text	Entity spans + types	Ambiguous names
Translation	Text (lang A)	Text (lang B)	Word order, idioms
Summarization	Long text	Short text	Keeping key info
QA	Question + context	Answer	Finding relevant info
Generation	Prompt	Continued text	Coherence, factuality

◆ Domain 3: Audio & Speech
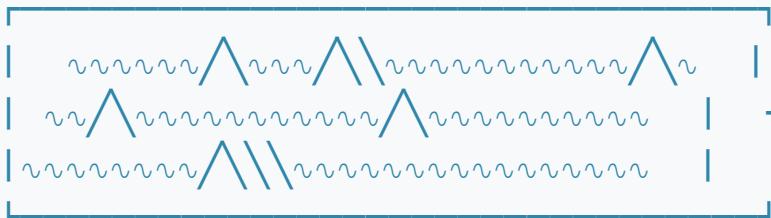
Teaching Machines to Hear

"The human voice is the most beautiful instrument of all." — Arvo Pärt

Task 15: Speech-to-Text (ASR)

What: Convert spoken audio to text.
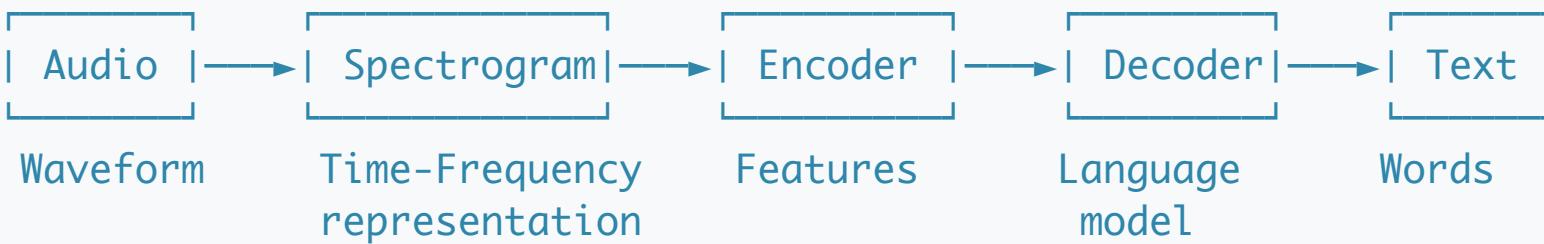
Audio Waveform: Text Output:

```
|  ～～～～～／\／～～～／\／\～～～～～～～～～／\～  |
|  ～／\／～～～～～～～～～／\／\～～～～～～～～～～  |        ──→    "Hello, how are
|  ～～～～～～／\／\\～～～～～～～～～～～～～～～～  |                    you today?"
```

Pipeline:

```
|  Audio  |──→| Spectrogram|──→| Encoder  |──→| Decoder|──→| Text |

  Waveform    Time-Frequency     Features      Language      Words
              representation                     model
```
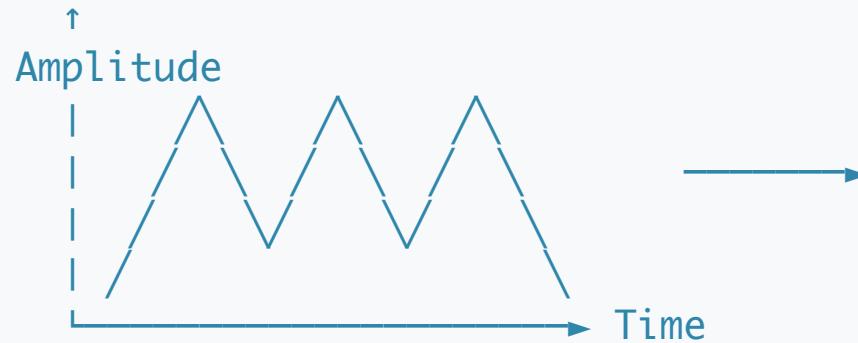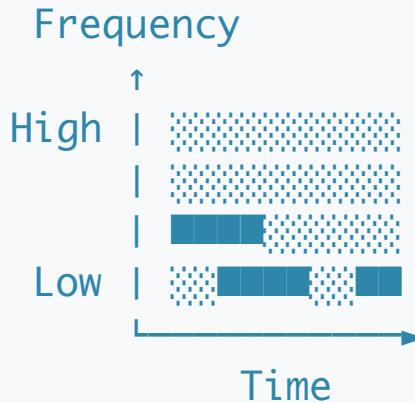
ASR: The Spectrogram

Converting sound to "images" that neural networks can process

```
Audio Wave:                          Spectrogram:
                                     (time → frequency "image")

 ↑
Amplitude                            Frequency
 |                                        ↑
 |     /\      /\                  High |  ░░░░░░░░░░░░░░
 |    /  \    /  \    /\                 |  ░░░░░░░░░░░░░░
 |   /    \  /    \  /  \                |  ██████░░░░░░░░
 |  /      \/      \/    \               |
 | /                      \       Low  |  ░░████████░░██
 |_____→             |_____→
                     Time                        Time
```

Bright areas = loud frequencies at that moment
Pattern = unique "fingerprint" of each word!

Whisper by OpenAI can transcribe audio in 99 languages with near-human accuracy!
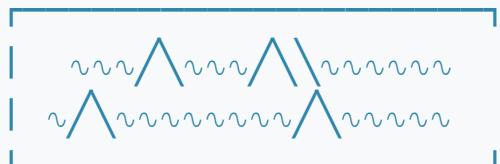
Task 16: Text-to-Speech (TTS)

What: Convert text to natural-sounding audio.
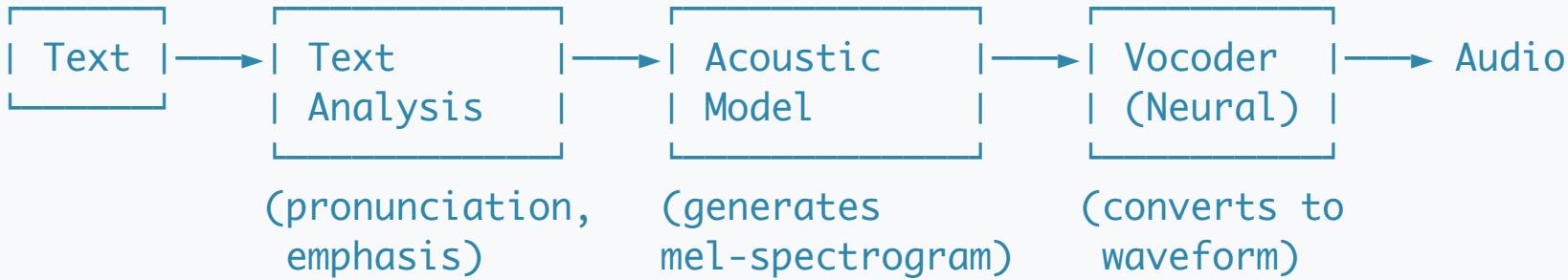
```
Text Input:                        Audio Output:

 _____                  _____
| "Welcome to the |               |                    |
|  future of AI.  |    ----->     | ∿∿∿/\∿∿∿/\∿∿∿∿∿∿   |
|  This is exciting!" |           | ∿/\∿∿∿∿∿∿∿∿∿/\∿∿∿∿ |
|_____|                |_____|


Modern TTS Pipeline:

 _____        _____        _____        _____
| Text  |----->| Text   |----->| Acoustic |----->| Vocoder  |-----> Audio
|_____|      | Analysis|     | Model    |      | (Neural) |
               |_____|      |_____|      |_____|

               (pronunciation,  (generates       (converts to
                 emphasis)      mel-spectrogram)   waveform)
```

TTS: Then vs Now

```
┌─────────────────────────────────────────────────┐
│                  TTS EVOLUTION                    │
├─────────────────────────────────────────────────┤
│                                                   │
│   1990s: Concatenative TTS                        │
│   ─────────────────────────                       │
│                                                   │
│   Splice together recorded phonemes               │
│   Result: Robotic, unnatural "The-wea-ther-to-day-is..." │
│                                                   │
│   2010s: Statistical Parametric TTS               │
│   ─────────────────────────────────               │
│                                                   │
│   HMM-based models, smoother but still artificial │
│                                                   │
│   2016: WaveNet (DeepMind)                        │
│   ─────────────────────────                       │
│                                                   │
│   Neural network generates audio sample by sample │
│   Human-like quality, but VERY slow               │
│                                                   │
│   2020s: Parallel Neural TTS                      │
│   ──────────────────────────                      │
│                                                   │
│   Real-time, expressive, can clone voices         │
│   Examples: ElevenLabs, Bark, XTTS                │
│                                                   │
└─────────────────────────────────────────────────┘
```
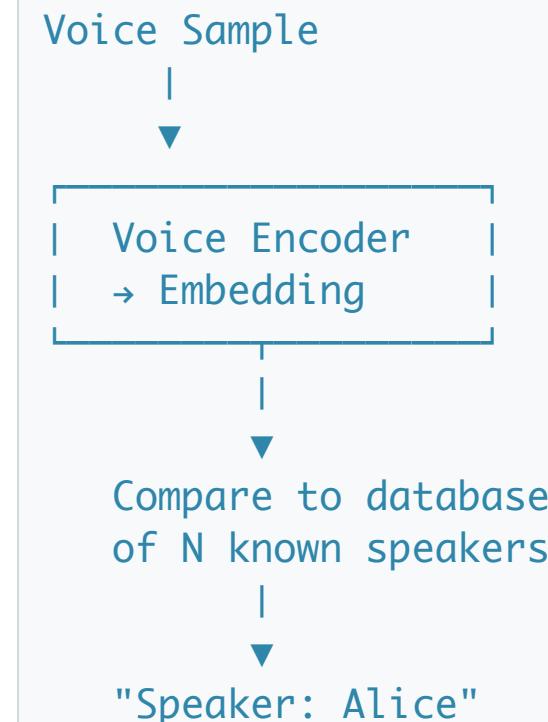
Task 17-18: Speaker Recognition
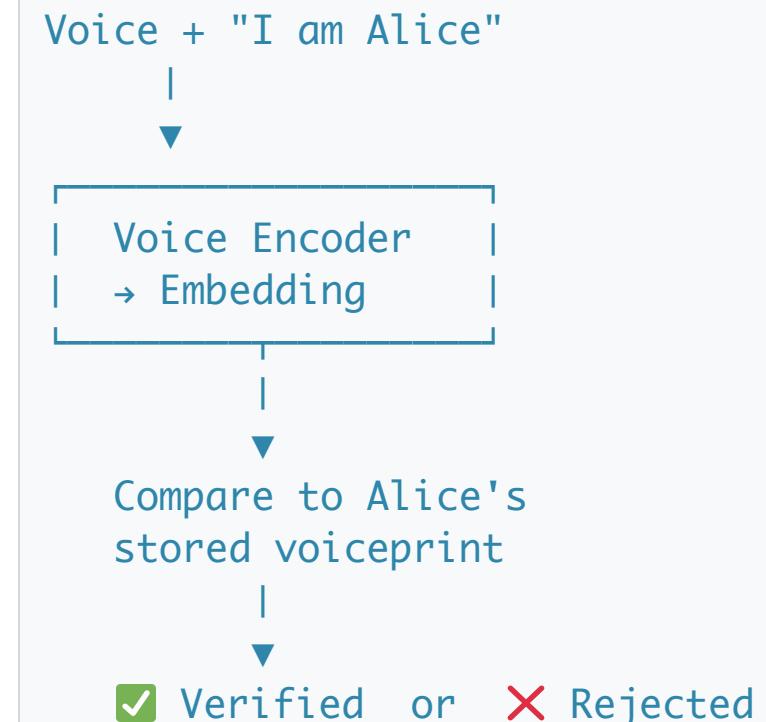
Speaker Identification:

Who is speaking? (1-of-N)

```
Voice Sample
     |
     ▼
|  Voice Encoder  |
|  → Embedding    |
     |
     ▼
Compare to database
of N known speakers
     |
     ▼
"Speaker: Alice"
```

Speaker Verification:

Is this who they claim to be?

```
Voice + "I am Alice"
     |
     ▼
|  Voice Encoder  |
|  → Embedding    |
     |
     ▼
Compare to Alice's
stored voiceprint
     |
     ▼
✅ Verified  or  ❌ Rejected
```

"Hey Siri" uses **speaker verification** — it only responds to the device owner's voice!

◆ Domain 4: Unsupervised Learning
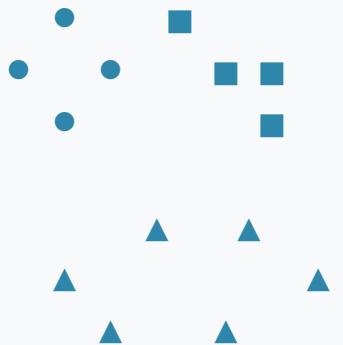
Finding Patterns Without Labels

"The goal is to find structure in chaos."
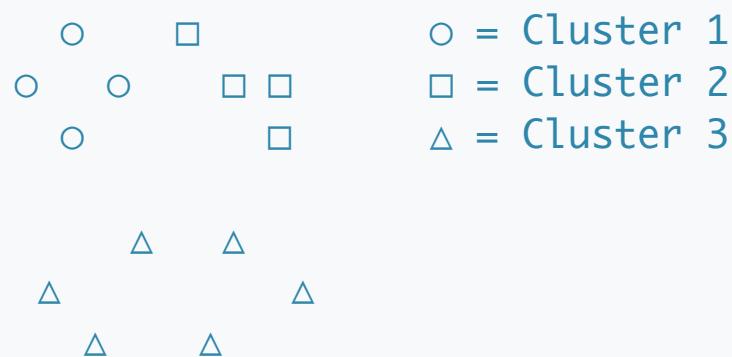
Task 19: Clustering

What: Group similar items together automatically (no labels needed!).

```
Before (unlabeled data):              After (3 clusters found):

    ●      ■                              ○      □           ○ = Cluster 1
  ●    ●    ■  ■                        ○    ○    □ □        □ = Cluster 2
    ●         ■                          ○         □         △ = Cluster 3


      ▲    ▲                                △    △
  ▲         ▲                           △         △
    ▲    ▲                                △    △


Algorithm (K-Means) figures out:
- There are 3 natural groups
- Which points belong to which group
```
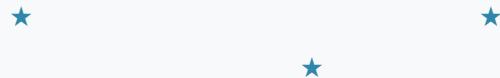
Clustering: The K-Means Algorithm

```
                    K-MEANS: STEP BY STEP


  Step 1: Pick K random "centroids" (cluster centers)
            ★                      ★

                    ★


  Step 2: Assign each point to nearest centroid
         ○○○ near ★1      □□□ near ★2      △△△ near ★3


  Step 3: Move centroids to center of their points
         ★1 moves to average of ○○○
         ★2 moves to average of □□□
         ★3 moves to average of △△△


  Step 4: Repeat steps 2-3 until centroids stop moving

  Done! Points are now clustered.
```

Clustering: Real Applications

Customer Segmentation

```
Cluster 1: "VIPs"
├─ High spending
├─ Infrequent visits
└─ Premium products

Cluster 2: "Regulars"
├─ Medium spending
├─ Weekly visits
└─ Staple items

Cluster 3: "Bargain Hunters"
├─ Low spending
├─ Sale days only
└─ Discounted items
```

Image Compression

```
Original: 16 million colors

After K-Means (K=16):
Only 16 colors!


| 🔴🟠🟡🟢🔵🟣⬛⬜ |
| and 8 more shades |


File size: 10x smaller
Quality: Still looks good!
```
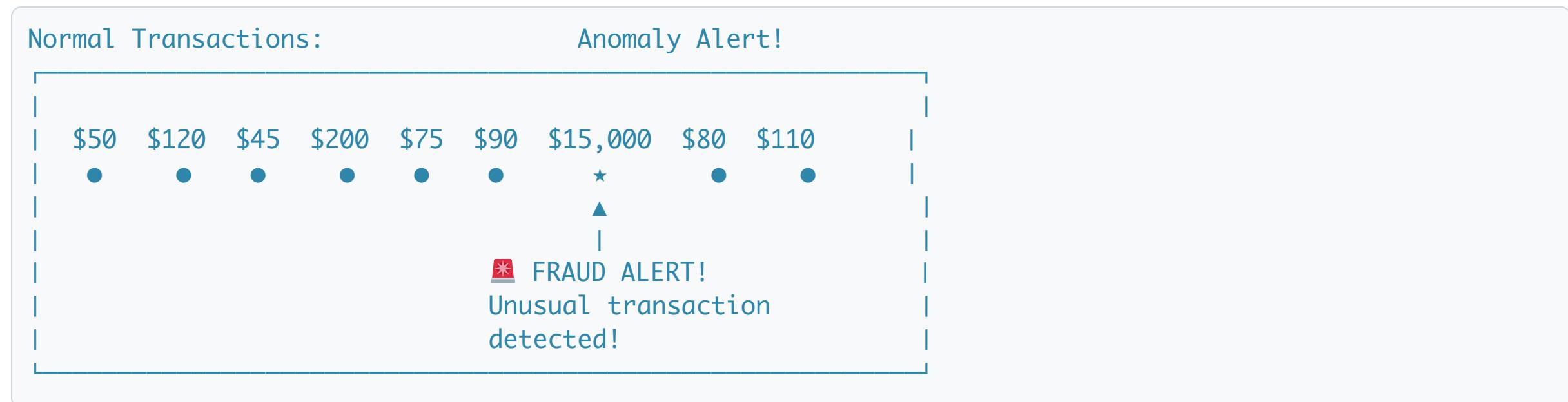
Task 20: Anomaly Detection

What: Find the outliers / unusual patterns.

```
Normal Transactions:                    Anomaly Alert!

   $50    $120   $45    $200   $75    $90   $15,000   $80    $110
    ●      ●      ●      ●      ●      ●       ★        ●      ●
                                                ▲
                                                |
                          🚨 FRAUD ALERT!
                          Unusual transaction
                          detected!
```

Anomaly detection is "learning what's normal, then flagging what's not."

Anomaly Detection: Methods

```
+------------------------------------------------------------+
|                  ANOMALY DETECTION APPROACHES              |
+------------------------------------------------------------+
|                                                            |
|  1. STATISTICAL                                            |
|     If value is > 3 standard deviations from mean → Anomaly|
|     Simple but assumes normal distribution                 |
|                                                            |
|  2. DISTANCE-BASED                                         |
|     If point is far from all other points → Anomaly        |
|     Works for any shape of data                            |
|                                                            |
|  3. DENSITY-BASED                                          |
|     If point is in a low-density region → Anomaly          |
|     Good for varying cluster sizes                         |
|                                                            |
|  4. AUTOENCODER (Neural Network)                           |
|     Train to reconstruct normal data                       |
|     High reconstruction error → Anomaly                    |
|     Best for complex, high-dimensional data                |
|                                                            |
+------------------------------------------------------------+
```
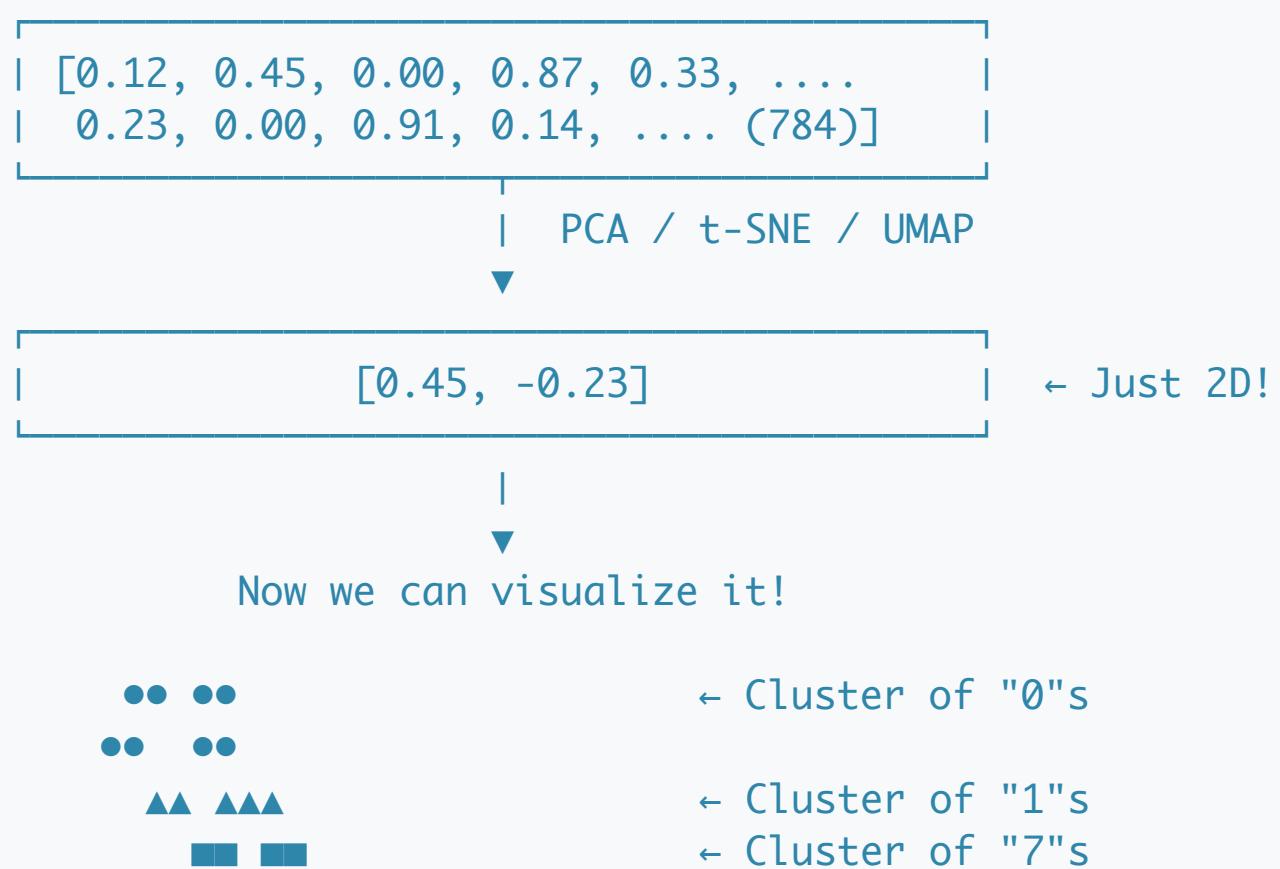
Task 21: Dimensionality Reduction

What: Compress high-dimensional data while preserving structure.

```
Original: 784 dimensions (28×28 MNIST image)

    ┌─────────────────────────────────────────┐
    │ [0.12, 0.45, 0.00, 0.87, 0.33, ....     │
    │  0.23, 0.00, 0.91, 0.14, .... (784)]     │
    └─────────────────────────────────────────┘
                    │   PCA / t-SNE / UMAP
                    ▼
    ┌─────────────────────────────────────────┐
    │           [0.45, -0.23]                  │   ← Just 2D!
    └─────────────────────────────────────────┘
                    │
                    ▼
            Now we can visualize it!

        ●● ●●                      ← Cluster of "0"s
        ●● ●●

        ▲▲ ▲▲▲                     ← Cluster of "1"s
          ■■ ■■                    ← Cluster of "7"s
```

57

Why Reduce Dimensions?

```
┌─────────────────────────────────────────────────────────────┐
│              BENEFITS OF DIMENSIONALITY REDUCTION             │
├─────────────────────────────────────────────────────────────┤
│                                                               │
│  1. VISUALIZATION                                             │
│     Can't plot 784D data, but can plot 2D!                    │
│     Reveal clusters and patterns to humans                    │
│                                                               │
│  2. SPEED                                                     │
│     ML on 10 features is 100x faster than 1000 features       │
│     Less computation, less memory                             │
│                                                               │
│  3. NOISE REMOVAL                                             │
│     Lower dimensions often capture signal, remove noise       │
│     Can improve model accuracy!                               │
│                                                               │
│  4. THE CURSE OF DIMENSIONALITY                               │
│     As dimensions ↑, distance between points → same           │
│     Need exponentially more data in high dimensions           │
│                                                               │
└─────────────────────────────────────────────────────────────┘
```
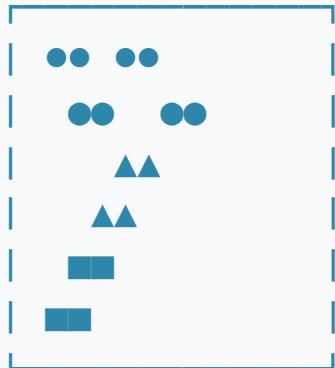
PCA vs t-SNE vs UMAP

Method	Speed	Preserves	Best For
PCA	Very fast	Global structure	Initial exploration
t-SNE	Slow	Local clusters	Visualizing clusters
UMAP	Fast	Both local + global	Best overall
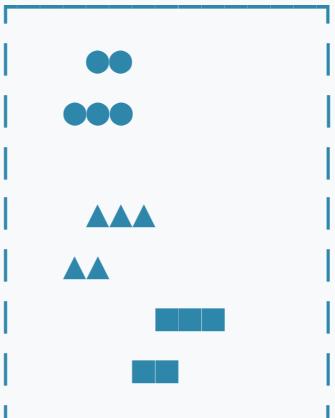
Same data, different methods:

PCA: t-SNE: UMAP:
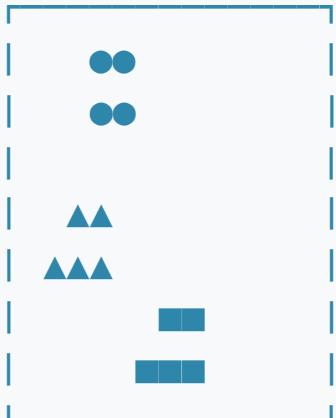
Linear projection   Clusters tight   Clusters + structure

◆ Domain 5: Generative Models

Creating New Content

"Creativity is just connecting things." — Steve Jobs

The Generative AI Revolution

```
                        GENERATIVE AI TIMELINE


    2014: GANs invented
          First realistic image generation

    2020: GPT-3 launches
          Text generation goes mainstream

    2022: DALL-E 2, Stable Diffusion, Midjourney
          Anyone can generate images from text

    2022: ChatGPT launches
          100M users in 2 months (fastest ever)

    2023: GPT-4, Claude, Gemini
          Multimodal: text + images + code

    2024: Sora (video), Suno (music)
          Generate any media type from text
```

Task 22: Image Generation
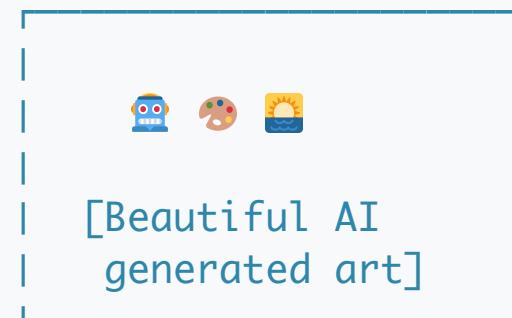
What: Create new images from noise, text, or other images.

```
Text-to-Image (Stable Diffusion, DALL-E, Midjourney):

Prompt: "A robot painting              Generated Image:
         a sunset, oil
         painting style"           ┌─────────────────┐
              |                    |                 |
              |                    |   🤖 🎨 🌅       |
              └──────────────────────>|                 |
                                   |                 |
                                   |  [Beautiful AI  |
                                   |   generated art]|
                                   └─────────────────┘



Noise-to-Image (GANs, Diffusion):

Random Noise ──> Generator ──> Realistic Image
 [z ~ N(0,1)]                  (faces, landscapes, art...)
```

62

How Diffusion Models Work

```
┌────────────────────────────────────────────────────────┐
│                  DIFFUSION: THE CORE IDEA              │
├────────────────────────────────────────────────────────┤
│                                                        │
│  TRAINING (Forward process):                           │
│  ─────────────────────────                             │
│  Take image → gradually add noise → pure noise         │
│                                                        │
│  🖼 ⟶ 🖼+⋯ ⟶ 🖼+⋯ ⟶ 🖼+▦ ⟶ ▦▦▦         │
│  Clean    Slight   Medium   Heavy    Pure              │
│  image     noise    noise    noise    noise            │
│                                                        │
│  GENERATION (Reverse process):                         │
│  ────────────────────────────                          │
│  Start with noise → gradually denoise → clean image    │
│                                                        │
│  ▦▦▦ ⟶ 🖼+▦ ⟶ 🖼+⋯ ⟶ 🖼+⋯ ⟶ 🖼          │
│  Pure      Heavy    Medium   Slight   Clean            │
│  noise      noise    noise    noise    image!          │
│                                                        │
│  The model learns: "Given noisy image, predict the noise" │
│                                                        │
└────────────────────────────────────────────────────────┘
```
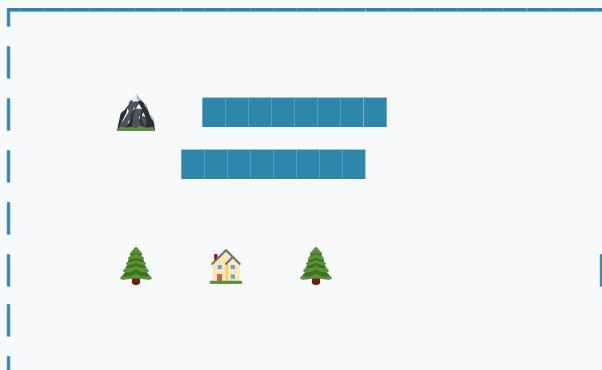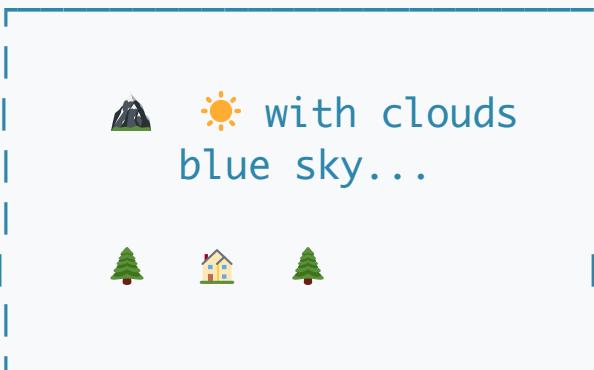
Task 23: Image Inpainting

What: Fill in missing or masked regions intelligently.

```
Original with hole:                Inpainted result:

    🏔 ████████                →        🏔  ☀️ with clouds
       ██████                            blue sky...

    🌲  🏠  🌲                           🌲  🏠  🌲

    (user painted a mask)              (AI filled it in)
```
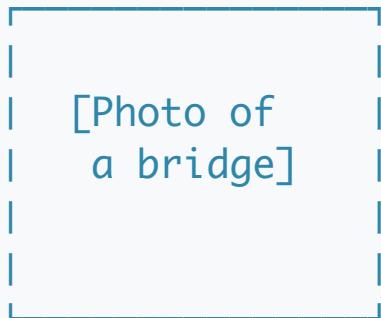
Uses:

- 🖌️ Remove unwanted objects (photobombers!)
- 🖼️ Restore damaged/old photos
- ⬆️ Extend image boundaries (outpainting)

Task 24: Style Transfer

What: Apply the artistic style of one image to the content of another.
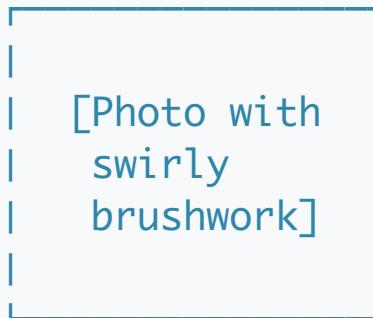
```
Content Image:          Style Image:          Result:
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│                 │     │                 │     │                 │
│   [Photo of     │  +  │   [Van Gogh's   │  =  │   [Photo with   │
│    a bridge]    │     │     Starry      │     │     swirly      │
│                 │     │     Night]      │     │    brushwork]   │
│                 │     │                 │     │                 │
└─────────────────┘     └─────────────────┘     └─────────────────┘


The model learns to separate:
- CONTENT: What objects are in the image (bridge, sky, water)
- STYLE: How they're rendered (brushstrokes, colors, texture)
```
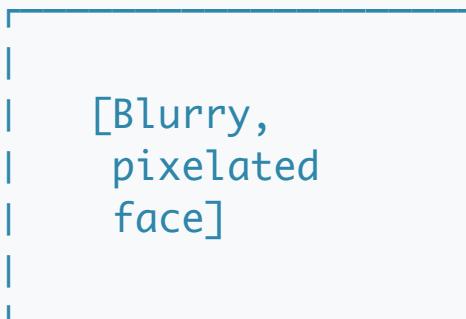
Task 25: Super Resolution

What: Upscale low-resolution images while adding realistic detail.

```
Low Resolution (64×64):          High Resolution (512×512):

 ┌──────────────────┐             ┌──────────────────────┐
 │                  │             │                      │
 │   [Blurry,       │             │   [Sharp, detailed image │
 │    pixelated     │     ──▶     │    with realistic texture, │
 │    face]         │  AI Magic   │    pores, hair strands] │
 │                  │             │                      │
 └──────────────────┘             └──────────────────────┘


Traditional upscaling: Just makes pixels bigger (still blurry)
AI upscaling: Adds plausible detail that wasn't there!
```

Ethical note: Super resolution "hallucinates" detail. The added details are plausible but not necessarily accurate. Not suitable for forensics or legal evidence!

◆ Domain 6: Self-Supervised Learning

The Secret Sauce of Modern AI

"Give me a lever long enough and I shall move the world." — Archimedes

The Self-Supervised Revolution

```
┌─────────────────────────────────────────────────────────┐
│                  THE LABELING BOTTLENECK                  │
├─────────────────────────────────────────────────────────┤
│                                                           │
│   SUPERVISED LEARNING:                                    │
│   ─────────────────────                                   │
│                                                           │
│   Need millions of labeled examples                       │
│   Labeling is EXPENSIVE and SLOW                          │
│   Limited by human annotation capacity                    │
│                                                           │
│   ImageNet: 14 million images                             │
│   Cost: ~$500,000 and years of work!                      │
│                                                           │
│   SELF-SUPERVISED LEARNING:                               │
│   ──────────────────────────                              │
│                                                           │
│   Create labels FROM the data itself                      │
│   "Free" supervision from data structure                  │
│   Can use BILLIONS of examples                            │
│                                                           │
│   GPT-3: 45 TB of text                                    │
│   Cost: Compute only (no labeling!)                       │
│                                                           │
└─────────────────────────────────────────────────────────┘
```
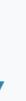
Task 26: Masked Language Modeling (BERT-style)

What: Predict the hidden word(s) — fill in the blank.

```
Training example:

Original:  "The cat sat on the mat."
Masked:    "The cat sat on the [MASK]."
                                  |
                                  ▼
                         ┌─────────────────┐
                         |      BERT       |
                         └─────────────────┘
                                  |
                                  ▼
Predictions:    "mat"    (0.45)  ← Correct!
                "floor"  (0.22)
                "couch"  (0.15)
                "bed"    (0.08)
                ...
```

BERT was trained on **3.3 billion words** from Wikipedia and books, just playing fill-in-the-blank!

Task 27: Next Token Prediction (GPT-style)

What: Predict what comes next in a sequence.

```
Input:  "The capital of France is"
                              |
                              ▼
                    ┌──────────────────┐
                    |        GPT       |
                    └──────────────────┘
                              |
                              ▼
Next token distribution:
       "Paris"  (0.89)  ← Most likely
       "the"    (0.03)
       "in"     (0.02)
       "a"      (0.01)
       ...
```

GPT, Claude, Gemini, and all LLMs are trained with just this one objective — repeated trillions of times! The simplicity is the brilliance.

BERT vs GPT: Key Differences

```
BERT vs GPT

BERT (Bidirectional)          GPT (Autoregressive)
_____          _____

"The [MASK] sat on mat"       "The cat sat on"→ "the"→ "mat"
      ↑↓↑↓↑                    ─────────────────────────→
Looks both directions         Only looks backward

Best for:                     Best for:
● Understanding text          ● Generating text
● Classification              ● Chat/dialogue
● Question answering          ● Code completion
● Named entity recognition    ● Creative writing

Can't generate text well      Can generate, but slower
(doesn't predict in order)    (one token at a time)
```
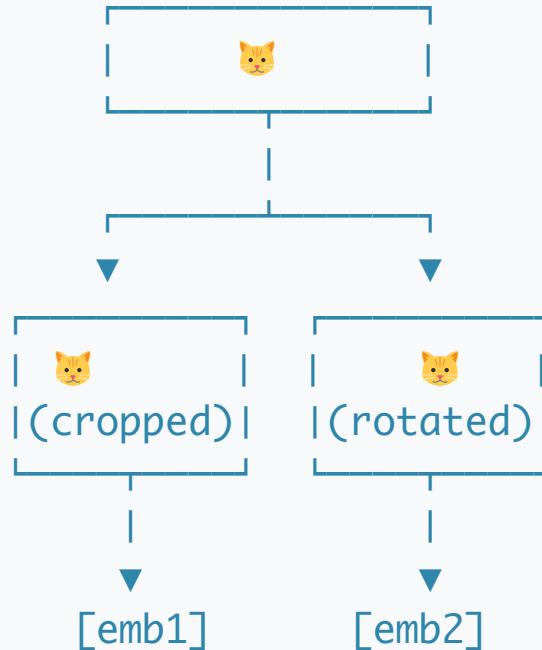
Task 28: Contrastive Learning

What: Learn that different views of the same image should have similar embeddings.

```
Original Image:
    ┌───────────────┐
    |      🐱       |
    └───────────────┘
            |
    ┌───────┴───────┐
    ▼               ▼              Create different "views"
                                   (augmentations)
 ┌───────┐       ┌───────┐
 |  🐱   |       |  🐱   |
 |(cropped)|     |(rotated)|
 └───────┘       └───────┘
     |               |
     ▼               ▼
  [emb1]          [emb2]       These should be SIMILAR!

Meanwhile: embeddings of DIFFERENT images should be DIFFERENT!

  [cat_emb] ←──────────────────────→ [dog_emb]
           Push apart!
```

Contrastive Learning: The Big Picture

```
┌─────────────────────────────────────────────────┐
│              CONTRASTIVE LEARNING                 │
├─────────────────────────────────────────────────┤
│                                                   │
│  Key Insight:                                     │
│  _____                                      │
│                                                   │
│  We don't need labels to learn good representations! │
│  Just need to know: "These are the same" vs "These are different" │
│                                                   │
│  Training:                                        │
│  _____                                         │
│                                                   │
│  • Take image → create 2 augmented versions (positive pair) │
│  • Other images in batch = negative pairs         │
│  • Learn: positives close, negatives far          │
│                                                   │
│  Result:                                          │
│  _____                                           │
│                                                   │
│  An encoder that maps similar images to similar embeddings │
│  Can then use these embeddings for ANY downstream task! │
│  Often matches supervised learning with just 1% of labels! │
│                                                   │
│  Famous methods: SimCLR, MoCo, CLIP               │
│                                                   │
└─────────────────────────────────────────────────┘
```
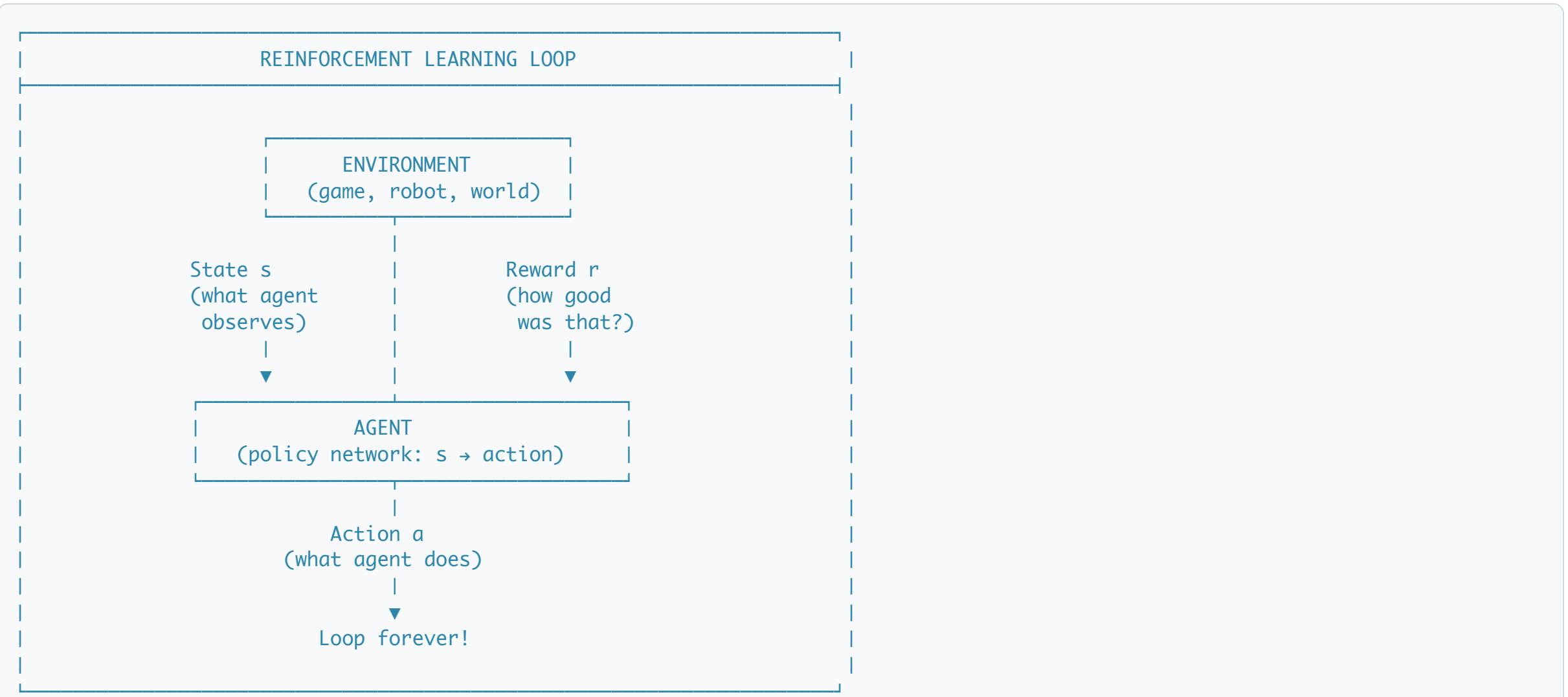
◆ Domain 7: Reinforcement Learning

Learning by Doing

"Experience is the teacher of all things." — Julius Caesar

The RL Framework

```
┌─────────────────────────────────────────────────────────────┐
│                 REINFORCEMENT LEARNING LOOP                   │
├─────────────────────────────────────────────────────────────┤
│                                                               │
│                   ┌─────────────────────┐                     │
│                   │     ENVIRONMENT      │                     │
│                   │  (game, robot, world) │                    │
│                   └─────────────────────┘                     │
│                              │                                │
│          State s             │         Reward r               │
│          (what agent         │         (how good              │
│           observes)          │          was that?)            │
│              │               │             │                  │
│              ▼               │             ▼                  │
│          ┌───────────────────────────────────┐               │
│          │              AGENT                 │               │
│          │   (policy network: s → action)     │               │
│          └───────────────────────────────────┘               │
│                          │                                    │
│                     Action a                                  │
│                  (what agent does)                            │
│                          │                                    │
│                          ▼                                    │
│                    Loop forever!                              │
│                                                               │
└─────────────────────────────────────────────────────────────┘
```
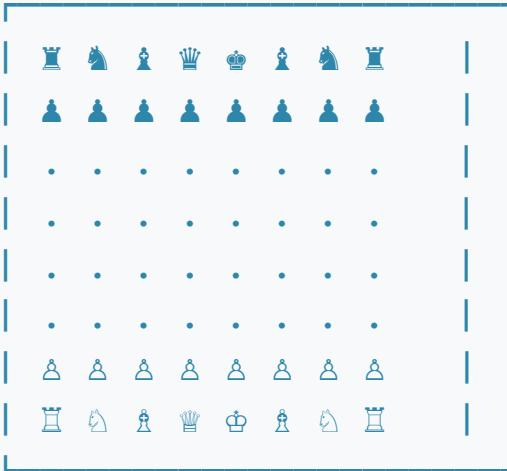
Task 29: Game Playing

What: Learn optimal strategy through trial and error.

```
Game State (Chess):              Agent Decision:

  ┌─────────────────────┐
  | ♜ ♞ ♝ ♛ ♚ ♝ ♞ ♜ |            ┌─────────────────────┐
  | ♟ ♟ ♟ ♟ ♟ ♟ ♟ ♟ |   ──▶     |  Best move:         |
  | . . . . . . . . |            |  e2 → e4            |
  | . . . . . . . . |            |                     |
  | . . . . . . . . |            |  Evaluation:        |
  | . . . . . . . . |            |  +0.3 pawns         |
  | ♙ ♙ ♙ ♙ ♙ ♙ ♙ ♙ |            └─────────────────────┘
  | ♖ ♘ ♗ ♕ ♔ ♗ ♘ ♖ |
  └─────────────────────┘


AlphaGo/AlphaZero: Learned by playing MILLIONS of games against itself!
No human games needed — pure self-play!
```

RL Milestones in Games

```
┌─────────────────────────────────────────────────────────────┐
│                  RL GAME-PLAYING ACHIEVEMENTS                 │
├─────────────────────────────────────────────────────────────┤
│                                                               │
│  1992: TD-Gammon                                              │
│  ───────────────                                              │
│                                                               │
│  Backgammon at world champion level                           │
│  First major RL success!                                      │
│                                                               │
│  2013: DQN (Atari)                                            │
│  ─────────────────                                            │
│                                                               │
│  Superhuman at 29 Atari games                                 │
│  Raw pixels as input!                                         │
│                                                               │
│  2016: AlphaGo                                                │
│  ─────────────                                                │
│                                                               │
│  Beats world champion Lee Sedol at Go                         │
│  Game with 10^170 possible positions!                         │
│                                                               │
│  2019: AlphaStar                                              │
│  ───────────────                                              │
│                                                               │
│  Grandmaster level at StarCraft II                            │
│  Real-time, imperfect information, complex strategy           │
│                                                               │
└─────────────────────────────────────────────────────────────┘
```
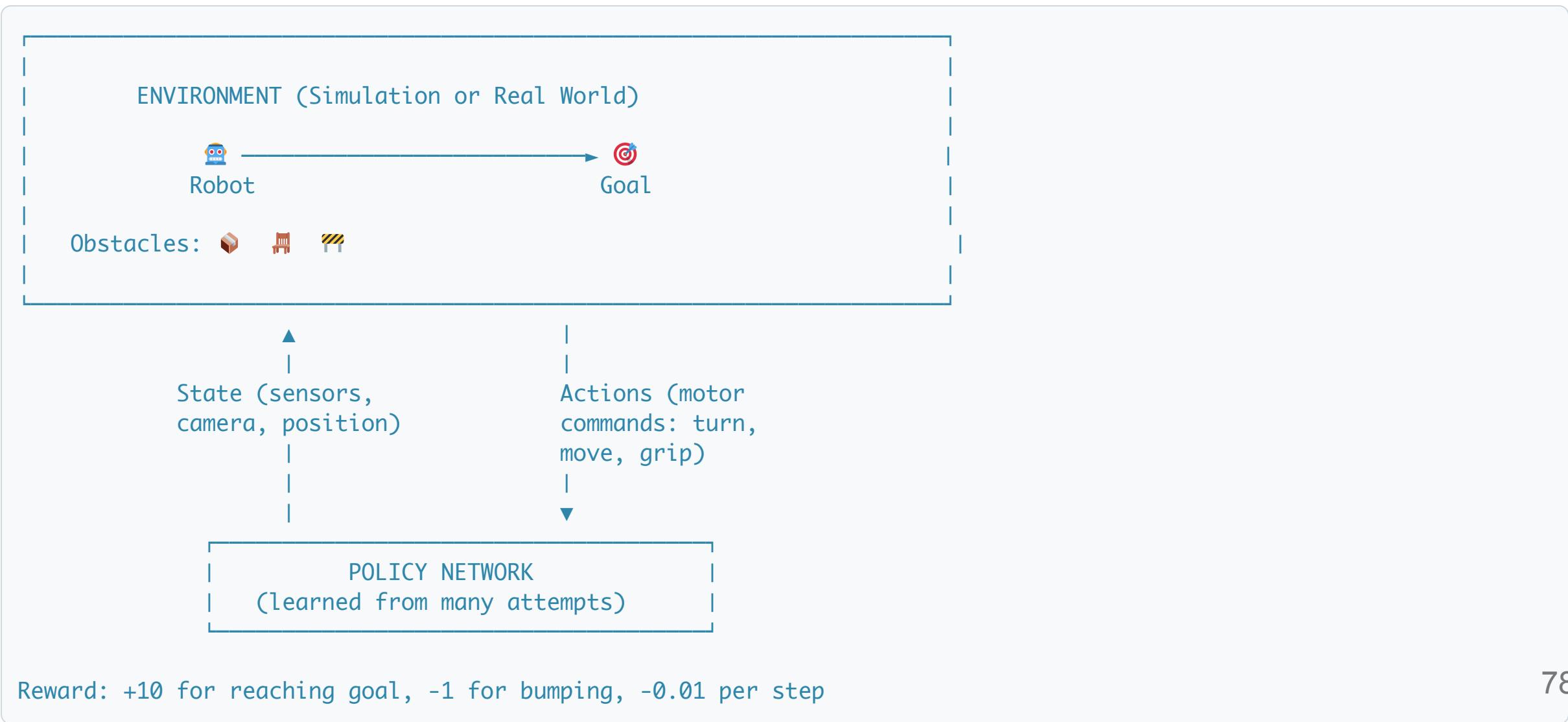
Task 30: Robot Control

What: Learn to move and interact in the physical world.

```
┌──────────────────────────────────────────────────────────┐
│                                                            │
│   ENVIRONMENT (Simulation or Real World)                   │
│                                                            │
│       🤖 ─────────────────────────▶ 🎯                    │
│       Robot                          Goal                  │
│                                                            │
│   Obstacles:  📦   🪑   🚧                                 │
│                                                            │
└──────────────────────────────────────────────────────────┘
              ▲                    │
              │                    │
         State (sensors,      Actions (motor
         camera, position)    commands: turn,
              │               move, grip)
              │                    │
              │                    ▼
        ┌───────────────────────────────┐
        │      POLICY NETWORK            │
        │  (learned from many attempts)  │
        └───────────────────────────────┘
```

Reward: +10 for reaching goal, -1 for bumping, -0.01 per step

Real-World RL Applications

Robotics

- Robot manipulation (picking objects)
- Drone navigation
- Self-balancing robots
- Walking robots (Boston Dynamics)

Games

- Video game AI
- Board game engines
- Game testing automation

Beyond Games

- **Data center cooling** (Google: 40% energy savings)
- **Chip design** (Google, NVIDIA)
- **Trading** (quantitative finance)
- **Recommendations** (long-term engagement)
- **RLHF** (making LLMs helpful & safe!)

ChatGPT uses RLHF (Reinforcement Learning from Human Feedback) to learn to be helpful rather than just predicting text!

◆ Domain 8: Multimodal Tasks

Combining Vision + Language

"The whole is greater than the sum of its parts." — Aristotle

The Multimodal Revolution

```
MULTIMODAL = MULTIPLE MODALITIES


MODALITY = Type of data
_____

• Text (words, sentences)
• Images (photos, diagrams)
• Audio (speech, music)
• Video (sequences of images + audio)

MULTIMODAL MODELS:
_____

• GPT-4V: Text + Images
• Gemini: Text + Images + Audio + Video
• CLIP: Connects text and images
• Claude: Text + Images + Documents

The trend: One model to rule them all!
```

Task 31: Visual Question Answering (VQA)

What: Answer questions about images using both visual and language understanding.

```
Image:                         Questions & Answers:

+-----------------------+
|                       |      Q: "How many people are
|   👤 👩 🐕             |              in the image?"
|                       |      A: "Two people"
|                       |
|  [People walking      |
|   a dog in a park]    |      Q: "What animal is there?"
|                       |      A: "A dog"
|                       |
|   🌳      🌳          |
|                       |      Q: "What are they doing?"
|                       |      A: "Walking their dog
|                       |              in a park"
|                       |
+-----------------------+
```

Requires BOTH:
- Understanding image content (computer vision)
- Understanding question (NLP)
- Reasoning to connect them!

Task 32: Image Captioning

What: Generate natural language description of an image.

```
Image:                          Generated Caption:

 _____
|                   |
|  🏃 🏃 🏃          |    ──→    "A group of runners
|                   |               participating in a city
|  [Marathon scene  |               marathon on a sunny day,
|   with crowds and |               with spectators cheering
|   city buildings] |               along the street and tall
|                   |               buildings in the background."
|  👥 👥 👥 👥 🏢🏢  |
|                   |
|_____|


The inverse of VQA:
Instead of answering questions, generate descriptions!
```
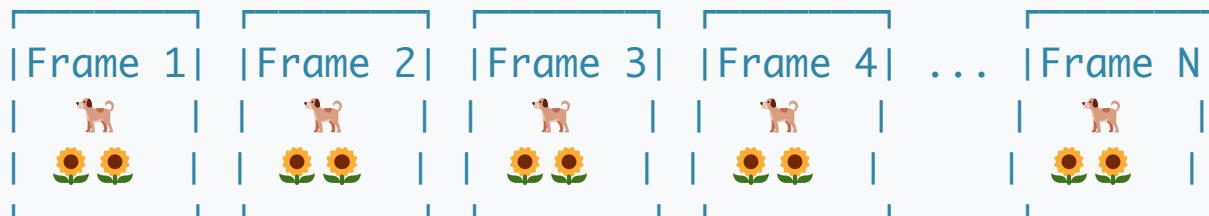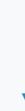
Task 33: Text-to-Video

What: Generate video from text description.

```
Prompt: "A golden retriever running through a field
         of sunflowers on a sunny day, slow motion"
                        |
                        ▼
            +-----------------------+
            |    Video Model        |
            |   (Sora, Runway,      |
            |     Pika, etc.)       |
            +-----------------------+
                        |
                        ▼

  |Frame 1|  |Frame 2|  |Frame 3|  |Frame 4|  ...  |Frame N|
  |  🐕   |  |  🐕   |  |  🐕   |  |  🐕   |       |  🐕   |
  |  🌻🌻  |  |  🌻🌻  |  |  🌻🌻  |  |  🌻🌻  |       |  🌻🌻  |


            Temporally consistent video!
```
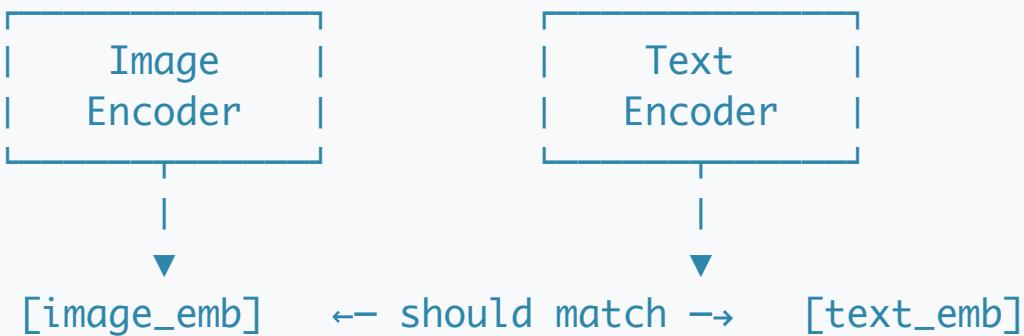
CLIP: The Foundation of Multimodal AI

```
┌──────────────────────────────────────────────────────────────┐
│              CLIP: Connecting Text and Images                  │
├──────────────────────────────────────────────────────────────┤
│                                                                │
│  Training Data: 400 million (image, caption) pairs from internet │
│                                                                │
│   ┌─────────────────┐      ┌─────────────────┐                 │
│   │     Image       │      │      Text       │                 │
│   │    Encoder      │      │    Encoder      │                 │
│   └─────────────────┘      └─────────────────┘                 │
│            │                        │                          │
│            ▼                        ▼                          │
│     [image_emb]    ←— should match —→    [text_emb]            │
│                                                                │
│  "A dog playing       →     [0.23, -0.41, 0.87, ...]          │
│   in the snow"              Shared embedding space!            │
│                                                                │
│  Result: Can search images with text, or text with images!    │
│  Powers: DALL-E, Stable Diffusion, image search, and more     │
│                                                                │
└──────────────────────────────────────────────────────────────┘
```

◆ Domain 9: Tabular & Time Series

The Classic ML Tasks

"Not everything that counts can be counted, but data often helps."

Task 34-35: Regression & Classification on Tables

Tabular Regression:

```
| Beds    | SqFt  | Price?  |
|---------|-------|---------|
| 3       | 1500  | ???     |
| 4       | 2200  | ???     |
| 2       | 900   | ???     |

        |
        ▼
  Predict: $425,000

Output: Continuous number
```

Tabular Classification:

```
| Age  | Income| Default?|
|------|-------|---------|
| 35   | 75K   | ???     |
| 52   | 120K  | ???     |
| 28   | 45K   | ???     |

        |
        ▼
  Predict: Yes / No

Output: Category
```
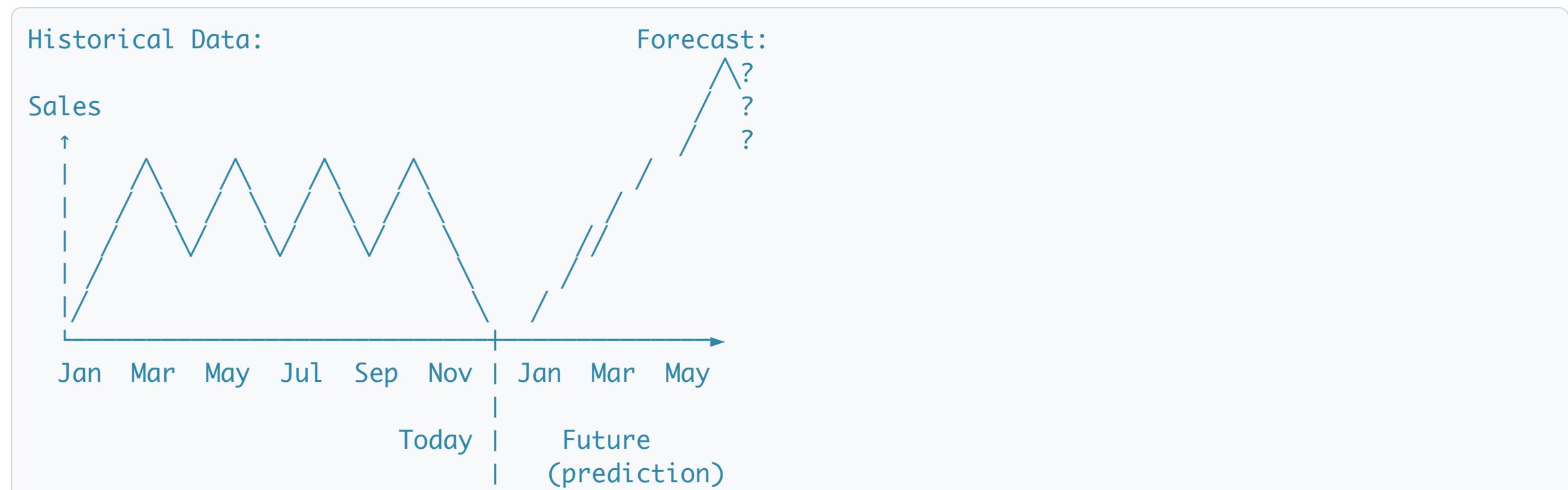
For tabular data, **gradient boosting (XGBoost, LightGBM)** often beats deep learning! Simpler, faster, and more interpretable.

Task 36: Time Series Forecasting

What: Predict future values from historical patterns.

```
Historical Data:                          Forecast:
                                                  /\ ?
Sales                                            /  \ ?
  ↑                                             /    ?
  |     /\      /\       /\        /\          /
  |    /  \    /  \     /  \      /  \       //
  |   /    \  /    \   /    \    /    \     //
  |  /      \/      \ /      \  /      \   /
  | /                \/       \/        \ /
  |/
  └────────────────────────────────────────→
  Jan  Mar  May  Jul  Sep  Nov | Jan  Mar  May
                               |
                       Today   |    Future
                               |  (prediction)
```
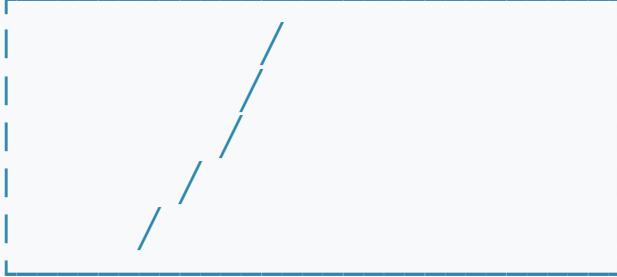
Components to model:
- Trend (overall direction)
- Seasonality (repeating patterns)
- Noise (random variation)

Time Series: Key Patterns

```
┌──────────────────────────────────────────────────────┐
│                  TIME SERIES COMPONENTS                │
├──────────────────────────────────────────────────────┤
│                                                        │
│  TREND: Long-term direction                            │
│  ─────────────────────────                             │
│  ┌──────────────────────────┐                          │
│  │                    /      │   Upward trend           │
│  │                   /       │   (e.g., company growth) │
│  │                  /        │                          │
│  │                 /         │                          │
│  │                /          │                          │
│  └──────────────────────────┘                          │
│                                                        │
│  SEASONALITY: Repeating patterns                       │
│  ───────────────────────────────                       │
│  ┌──────────────────────────┐                          │
│  │   /\    /\    /\    /\    │   Weekly, monthly, yearly│
│  │  /  \  /  \  /  \  /  \   │   (e.g., holiday shopping)│
│  │ /    \/    \/    \/    \  │                          │
│  └──────────────────────────┘                          │
│                                                        │
└──────────────────────────────────────────────────────┘
```

Task 37: Recommendation Systems

What: Predict what users will like based on their history.

```
User-Item Matrix:                    Recommendations:

        Movie1 Movie2 Movie3 Movie4 | For User A:           |
User A   ★★★★★    ?     ★★★     ?    | ● Movie2 (pred: 4.2)|
User B   ★★★★   ★★★★★   ?      ★★   | ● Movie4 (pred: 3.8)|
User C     ?    ★★★★   ★★★★★  ★★★ |                       |
                                    | "Because you liked    |
                                    |  Movie1 and Movie3"   |


Two main approaches:
● Collaborative: "Users like you also liked..."
● Content-based: "Similar items to ones you liked..."
```

Recommendation: The Netflix Problem

```
┌─────────────────────────────────────────────────────────┐
│                 RECOMMENDATION CHALLENGES                 │
├─────────────────────────────────────────────────────────┤
│                                                           │
│ THE COLD START PROBLEM:                                   │
│ ──────────────────────                                    │
│                                                           │
│ New user: No history → What to recommend?                 │
│ New item: No ratings → Who might like it?                 │
│ Solution: Ask preferences, use demographics, popular items│
│                                                           │
│ THE SPARSITY PROBLEM:                                     │
│ ────────────────────                                      │
│                                                           │
│ Netflix: 200M users × 15K movies = 3 trillion possible ratings│
│ Actual ratings: ~5 billion → 0.17% filled!                │
│ Solution: Matrix factorization, embeddings                │
│                                                           │
│ THE FILTER BUBBLE:                                        │
│ ─────────────────                                         │
│                                                           │
│ Only showing similar content → User misses diverse content│
│ Solution: Exploration vs exploitation, diversity metrics  │
│                                                           │
└─────────────────────────────────────────────────────────┘
```

Summary: The ML Task Landscape

```
ML TASK FAMILIES

SUPERVISED              UNSUPERVISED            SELF-SUPERVISED
_____             _____            _____

• Classification        • Clustering            • Masked LM (BERT)
• Regression            • Dim. Reduction        • Next Token (GPT)
• Detection             • Anomaly Det.          • Contrastive (CLIP)
• Segmentation
• Seq2Seq

GENERATIVE              REINFORCEMENT           MULTIMODAL
_____              _____           _____

• Image Gen             • Game Playing          • VQA
• Text Gen              • Robotics              • Captioning
• Video Gen             • RLHF                  • Text-to-Image
• Style Transfer                                • Text-to-Video
```
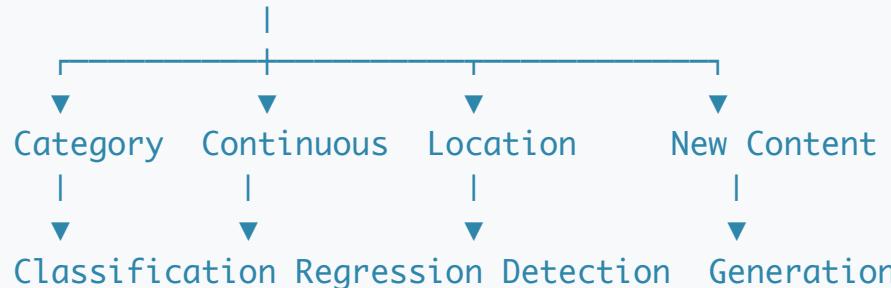
Choosing the Right Task

```
┌──────────────────────────────────────────────────┐
│                 DECISION FLOWCHART                 │
├──────────────────────────────────────────────────┤
│                                                    │
│   What do you want to predict?                     │
│              │                                     │
│      ┌───────┼───────────────┬──────────────┐      │
│      ▼       ▼               ▼              ▼      │
│  Category  Continuous     Location      New Content │
│      │        │               │              │      │
│      ▼        ▼               ▼              ▼      │
│ Classification Regression Detection   Generation    │
│                                                    │
│   Have labels?                                     │
│   ├── Yes → Supervised                             │
│   ├── No  → Unsupervised (clustering, anomaly)     │
│   └── Can create from data → Self-supervised       │
│                                                    │
│   Need to take actions?                            │
│   └── Yes → Reinforcement Learning                 │
│                                                    │
│   Multiple input types?                            │
│   └── Yes → Multimodal                             │
│                                                    │
└──────────────────────────────────────────────────┘
```

93

Key Takeaways

The 5 Things to Remember

1. **Every task = Input type + Output type**
   Define these clearly and you've defined your problem

2. **Same architectures work across domains**
   Transformers power text, images, audio, and more

3. **Self-supervised learning powers modern AI**
   GPT, BERT, CLIP — all learned from unlabeled data

4. **Start with the task → then choose the model**
   Don't pick a model first and force it to fit

5. **Real-world ML often combines multiple tasks**
   Self-driving cars: detection + segmentation + prediction + control

The ML Practitioner's Toolkit

Task Type	Go-To Models (2024)
Image Classification	ResNet, EfficientNet, ViT
Object Detection	YOLOv8, DETR, RT-DETR
Segmentation	SAM, Mask R-CNN
Text Classification	BERT, RoBERTa
Text Generation	GPT-4, Claude, Llama
Speech-to-Text	Whisper
Image Generation	Stable Diffusion, DALL-E
Tabular	XGBoost, LightGBM, TabNet
Time Series	Prophet, N-BEATS, TimeGPT
Recommendations	Two-Tower, DLRM

What's Next?

In the Labs:

- Lab 1-2: sklearn basics
- Lab 3: PyTorch & neural nets
- Lab 4-5: Build your own LLM
- Lab 6-7: Object detection

The Bigger Picture:

- Most tasks share core principles
- Transfer learning is key
- Start simple, add complexity
- The best model is the one you ship!

Pick a task, find a dataset, and start building! The best way to learn ML is by doing.

Thank You!

"The best way to predict the future is to invent it." — Alan Kay

Resources

- Papers With Code (paperwithcode.com) — State-of-the-art models

- Hugging Face (huggingface.co) — Pre-trained models

- Kaggle (kaggle.com) — Datasets and competitions

Questions?