



Traffic Lights

Man

OBJECT DETECTION WITH LVLMs

From Natural Language Query to Bounding Boxes

Introduction

Can LVLM "Detect Objects" Like We Do?

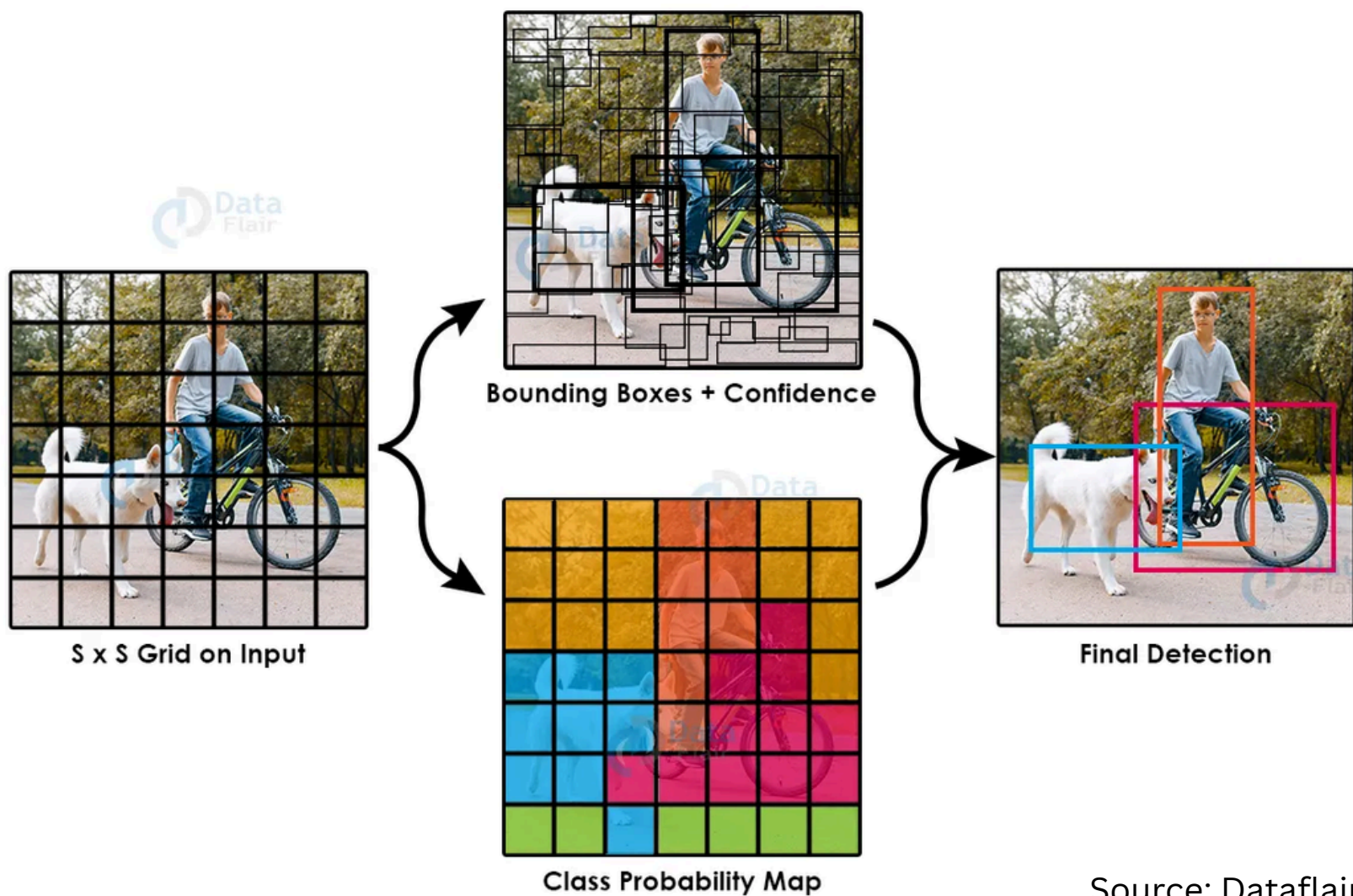
LVLMs are now automating object detection, eliminating labeled datasets and manual annotation, but how? Keep reading ;)

What you'll learn:

- How LVLMs interpret natural language prompts to find objects
- The hybrid workflow that combines language models with traditional computer vision
- Challenges and practical solutions for implementation
- Real-world applications across industries



1. Conventional Object Detection



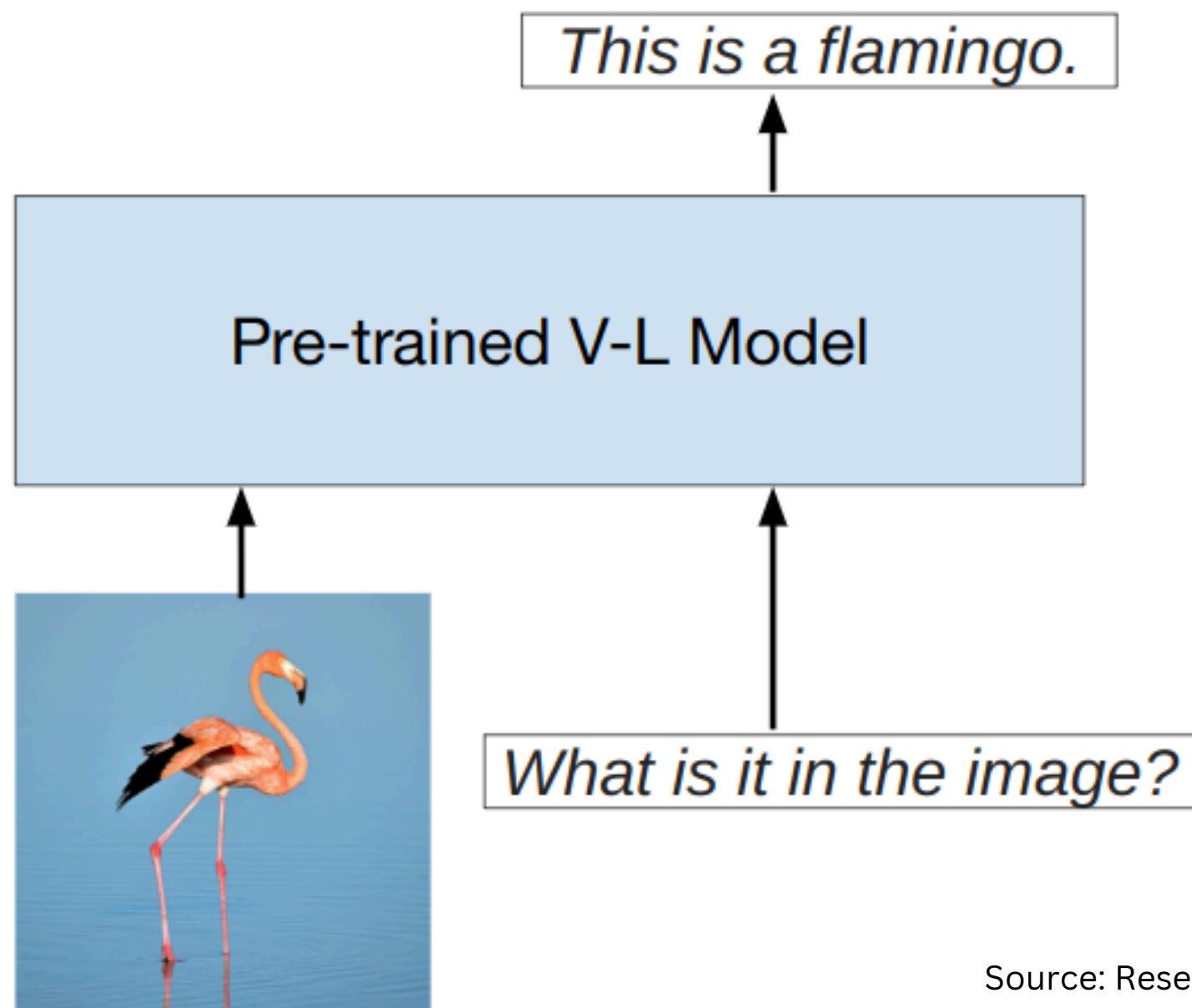
Traditional approach mainly involves:

- **Tedious Data Annotation:** Manually drawing boxes around objects, one by one.
- **Lengthy Model Training:** Training models for weeks on annotated data.
- It includes supervised models like **YOLO** (You Only Look Once), **Faster R-CNN**, and **SSD** (Single Shot Detector)

The challenges with this approach:

- **Slow & Expensive:** Data collection and annotation are HUGE bottlenecks.
- **Limited Generalization:** Models only worked well on what they are trained on but fail to detect unseen objects.
- **Inflexible:** Adding new objects meant starting from scratch.

2. What are LVLMs?

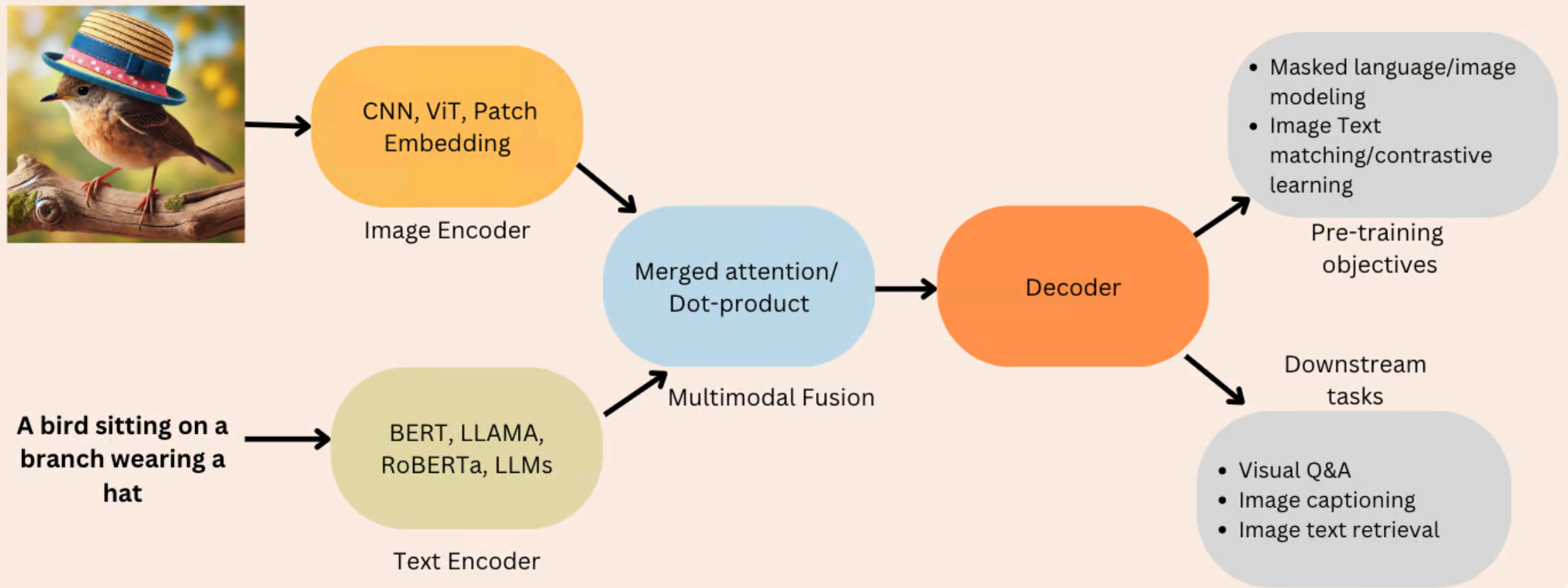


Source: ResearchGate

LVLMs are AI systems designed to understand and connect visual content (images) with text. They can analyze an image and generate meaningful descriptions or answer questions about it by combining visual recognition and language understanding.

Example: Given the image of a flamingo, an LVLM could process the image and respond to the prompt "What is it in the image?" with: "This is a flamingo." The model identifies the bird visually and generates a text response based on its training

3. How LVLMs Work?



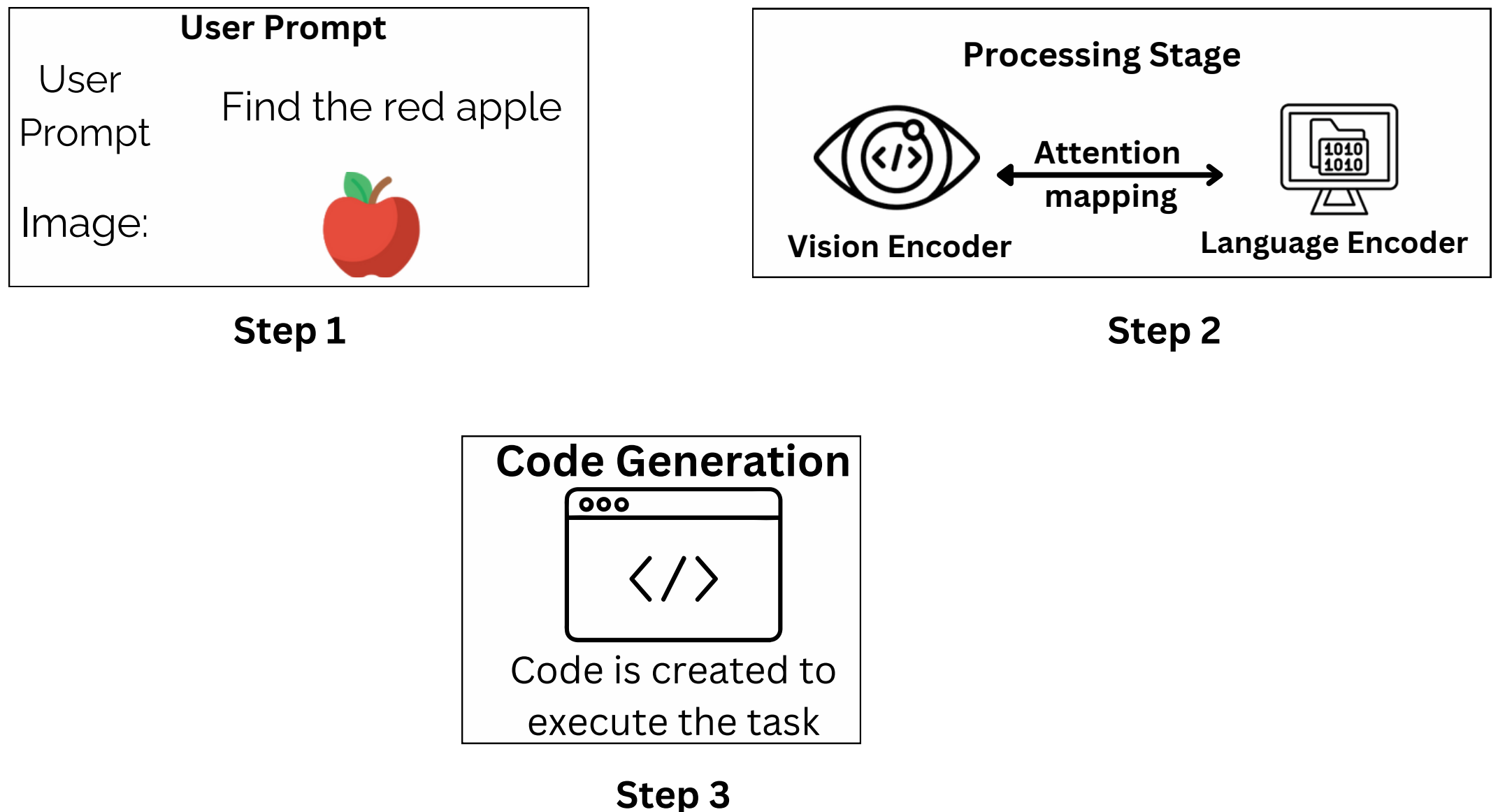
Source: Datacamp

LVLMs integrate visual and textual data to generate context-aware outputs. They function through:

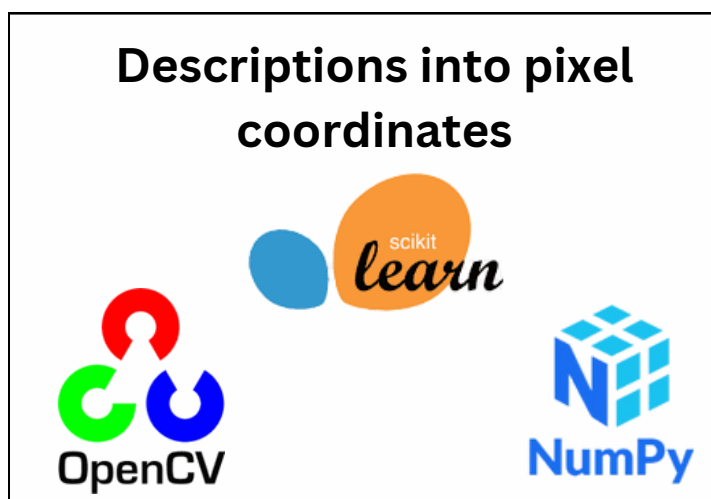
- **Multimodal Training:** Learning from image-text datasets to link visual elements with descriptions.
- **Transformer Architecture:** Using self-attention to align visual features with text tokens.
- **Tokenization & Embedding:** Mapping text tokens and visual embeddings into a shared space.
- **Fine-Tuning:** Adapting for tasks like image captioning, visual question answering, and object localization.

4. How LVLMs achieve Pixel-Precise Localization?

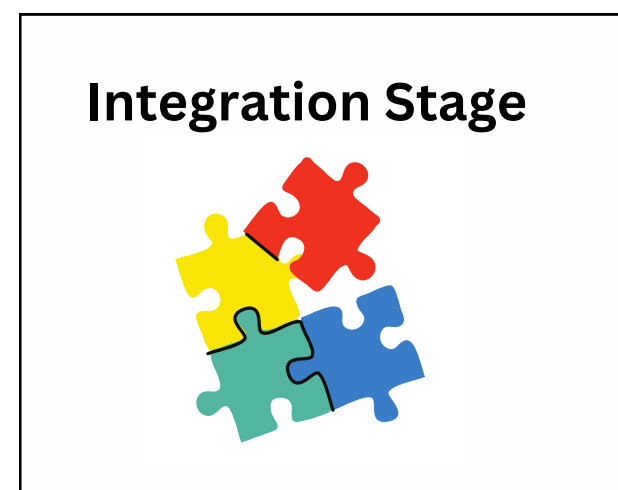
Workflow for LVLM Object Detection & Localization:



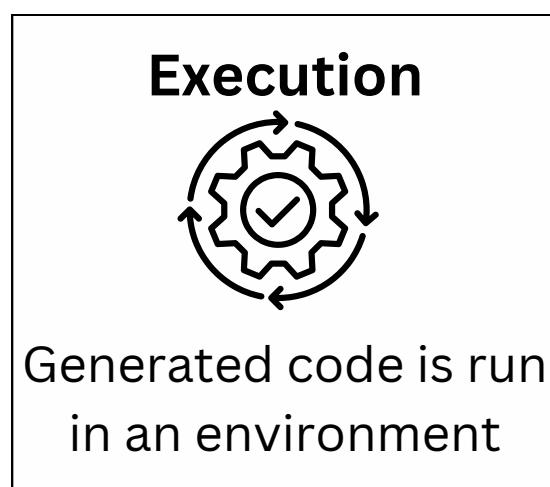
- 1. User Prompt:** User provides a text prompt (e.g., "Find the red apple") and an image.
- 2. Processing Stage:**
 - a. Vision Encoder:** Extracts visual features from the image.
 - b. Language Encoder:** Understands the text prompt.
 - c. Attention Mapping:** Aligns text with relevant regions in the image.
- 3. Code Generation:** The model generates executable code to process the task, including steps for object localization.



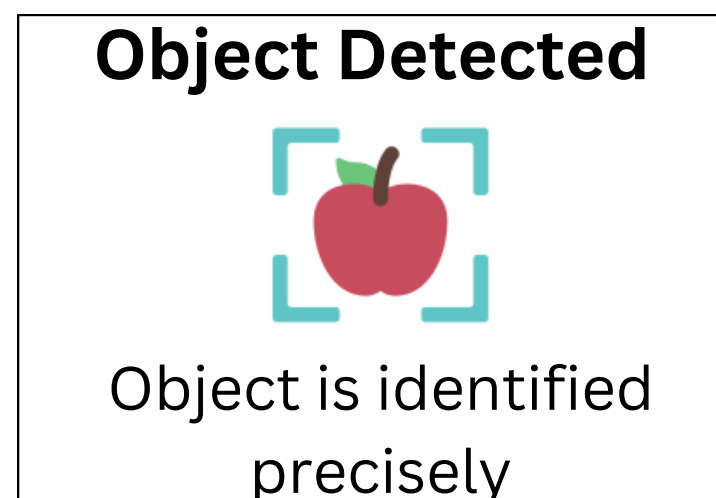
Step 4



Step 5



Step 6



Step 7

4.Descriptions into Pixel Coordinates: Tools like OpenCV, NumPy, and scikit-learn convert attention maps into precise pixel coordinates for bounding boxes.

5.Integration Stage: Combines visual and textual data to refine object localization.


6.Execution: The generated code is executed in an environment to identify the object.

7.Object Detection: The object is localized with bounding boxes and displayed precisely.

Step 1




User Prompt

User Prompt Find the red apple

Image: 

Step 2




Processing Stage

Vision Encoder Language Encoder

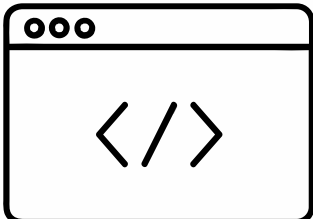
Step 4

Descriptions into pixel coordinates

Step 3


Code Generation



Code is created to execute the task

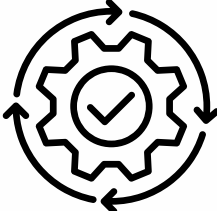
Step 5

Integration Stage



Step 6


Execution



Generated code is run in an environment

Step 7

Object Detected



Object is identified precisely

5. Tradeoffs in Object Detection

Conventional Pipeline

VS

LVLM Approach

Aspects

Conventional Pipeline

LVLM Approach

Data Handling



Needs extensive data collection & annotation

Utilizes prompts and reducing data needs

Inference



Time-efficient & used in real-time

Slow due to LVLM latency during code generation

Human Resources



High demand for human annotators

Minimal human intervention required

Flexibility



Rigid structure, difficult to adapt

Highly adaptable to new asks via prompts

Performance



Highly optimized for well-defined tasks with enough data

Depends on prompt quality and domain coverage

6. Applications Across Industries

Retail



Inventory management systems that identify products from natural language descriptions (40% faster stocktaking).



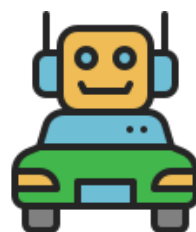
Healthcare

Medical imaging tools that locate anomalies based on radiologist descriptions (30% improvement in screening efficiency).

Manufacturing



Quality control systems that detect defects from verbal specifications without reprogramming



Autonomous Vehicles

CLIP and Grounding DINO models enable identification of unexpected road obstacles from simple descriptions.



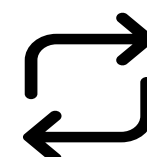
**Follow to stay updated on
Generative AI**



LIKE



COMMENT



REPOST