Towards Resource-Efficient Compound AI Systems

Gohar Irfan Chaudhry * Esha Choukse † Íñigo Goiri †
Rodrigo Fonseca † Ricardo Bianchini ‡ Adam Belay *

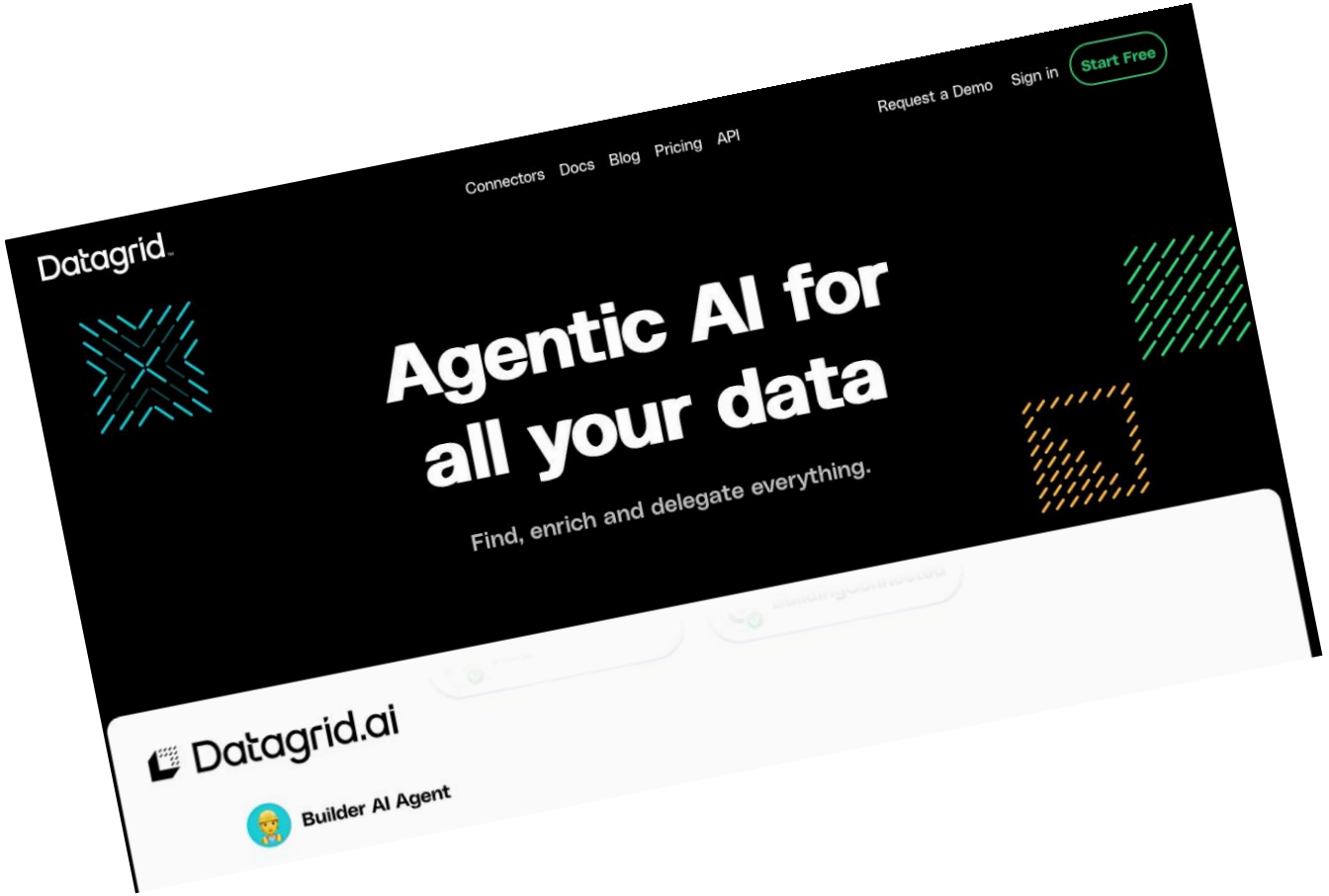
MIT CSAIL * Microsoft Azure Research – Systems † Microsoft Azure ‡

Compound AI systems are everywhere…

Agentic workflows composed from LLMs, ML models, tools, web APIs…

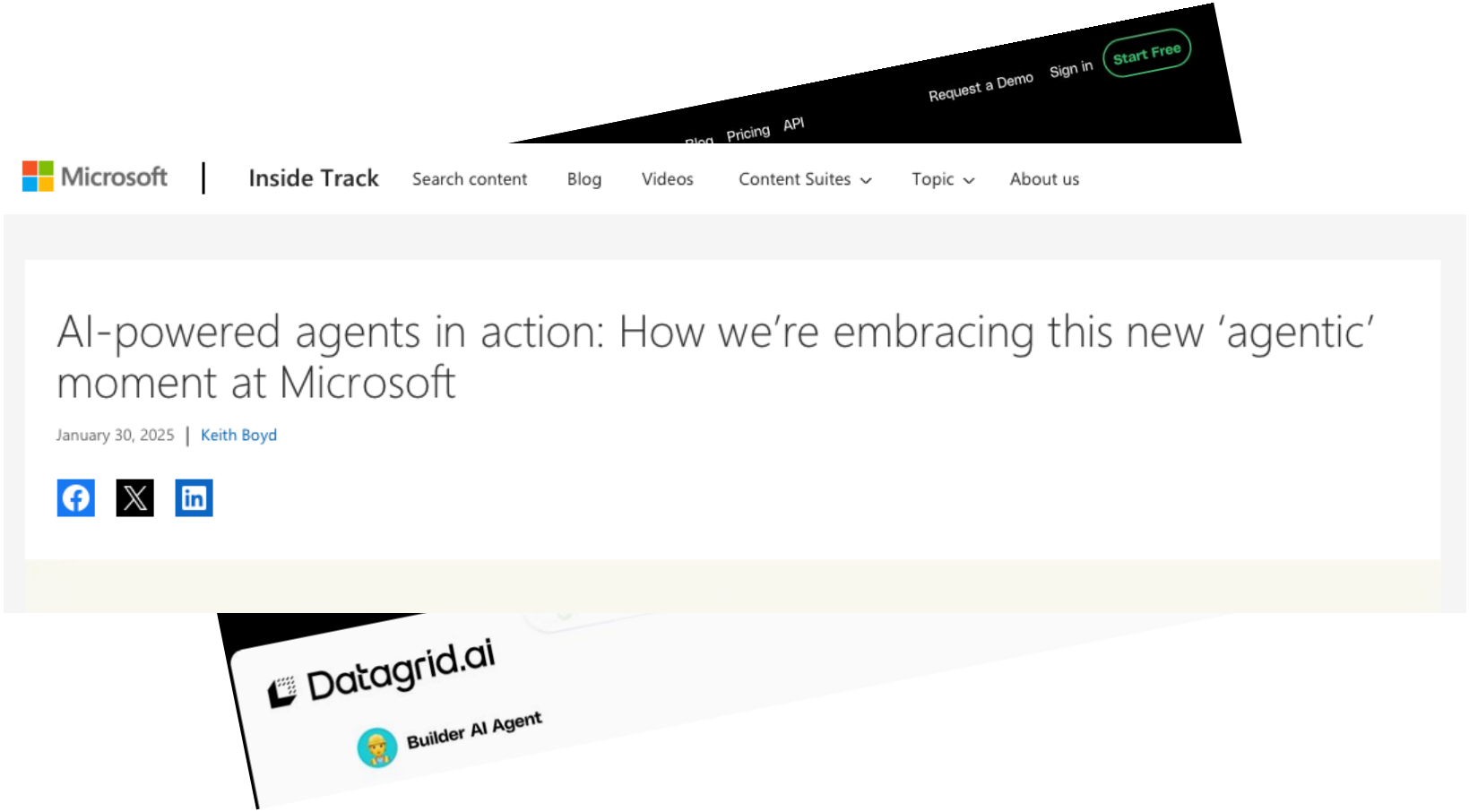
Compound AI systems are everywhere…

Agentic workflows composed from LLMs, ML models, tools, web APIs…



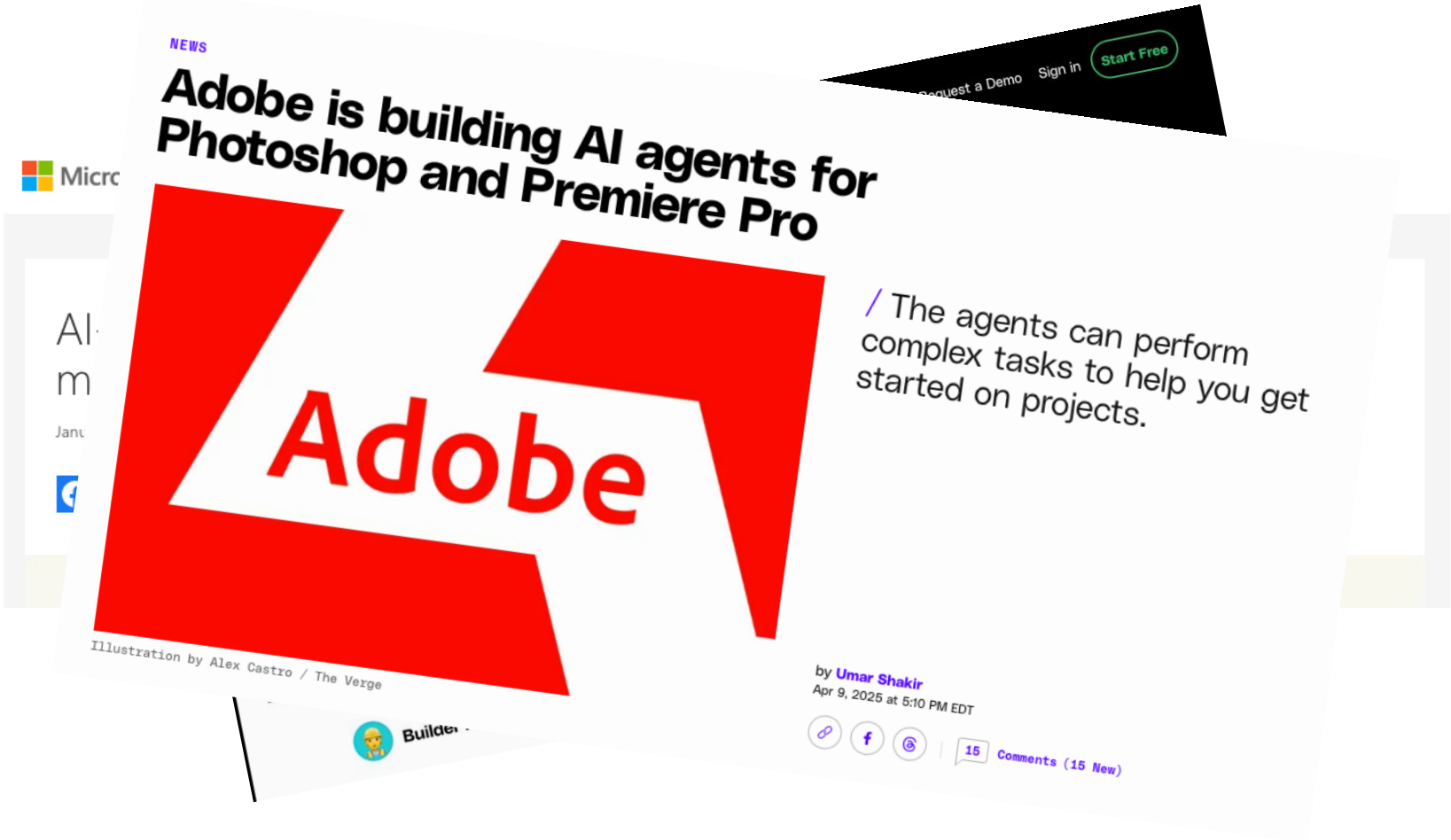
Compound AI systems are everywhere…

Agentic workflows composed from LLMs, ML models, tools, web APIs…



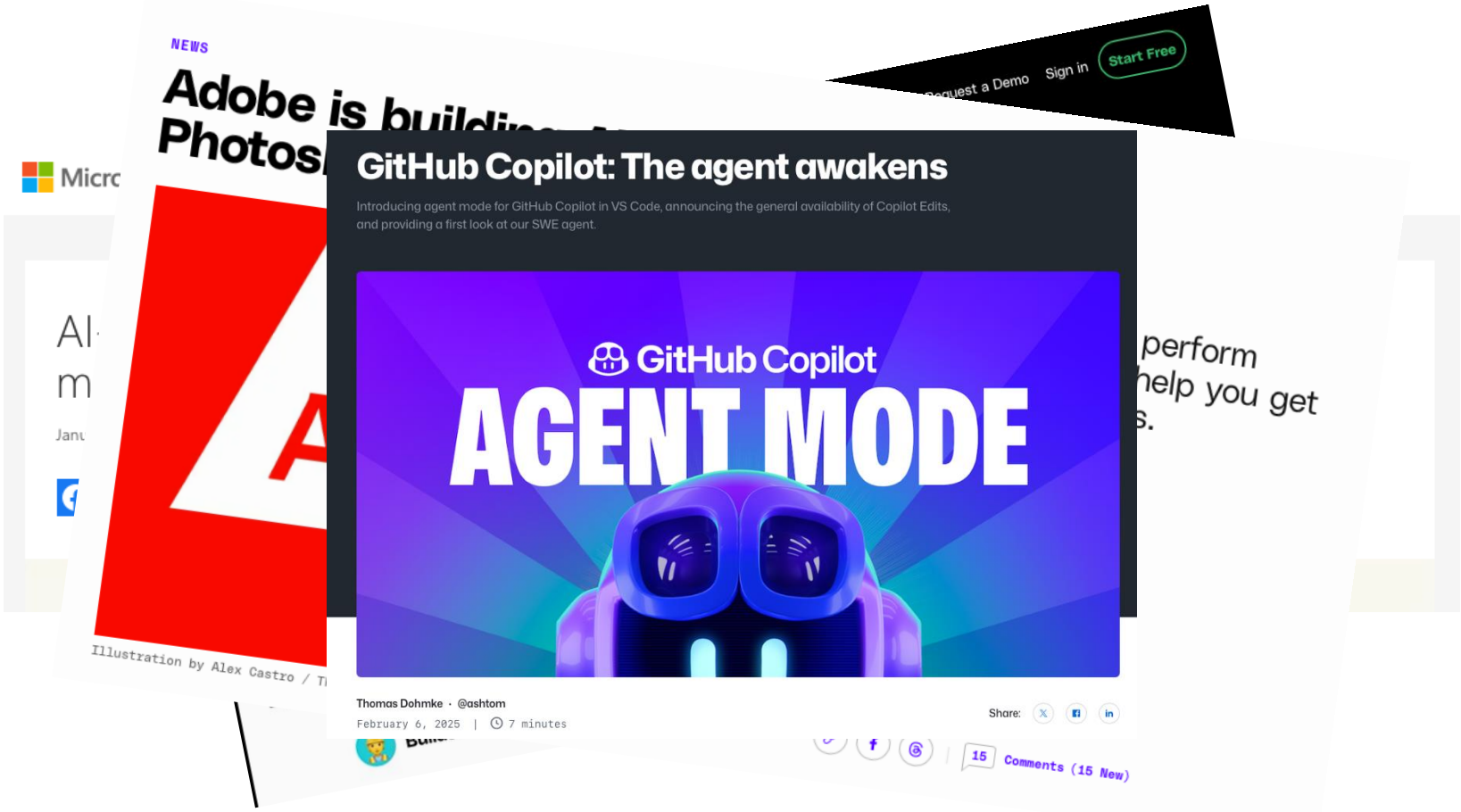
Compound AI systems are everywhere…

Agentic workflows composed from LLMs, ML models, tools, web APIs…



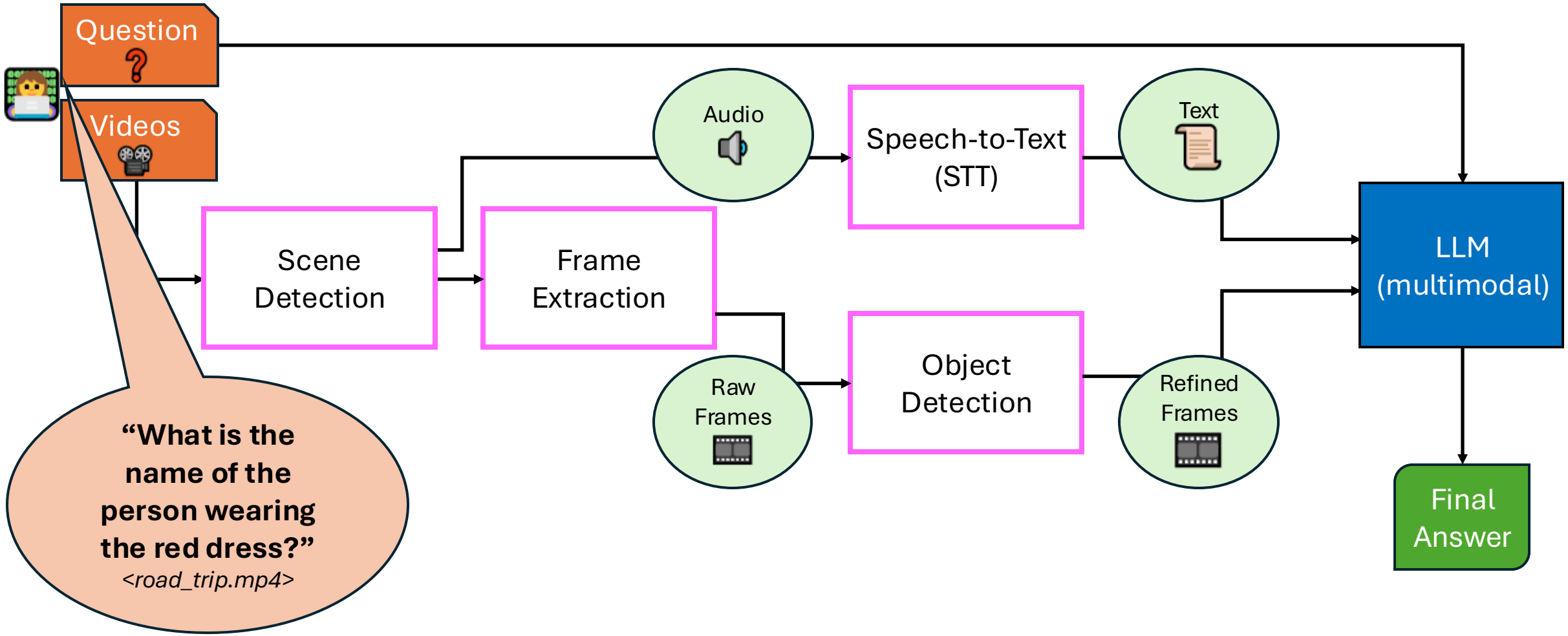
Compound AI systems are everywhere…

Agentic workflows composed from LLMs, ML models, tools, web APIs…

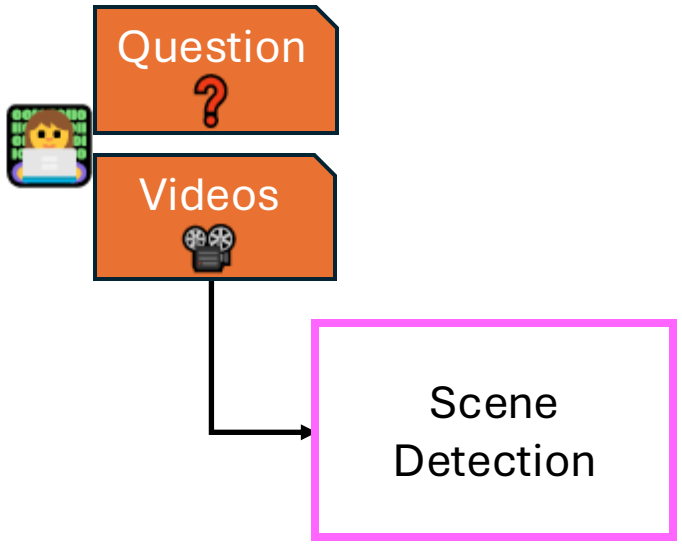


Are these systems efficient?

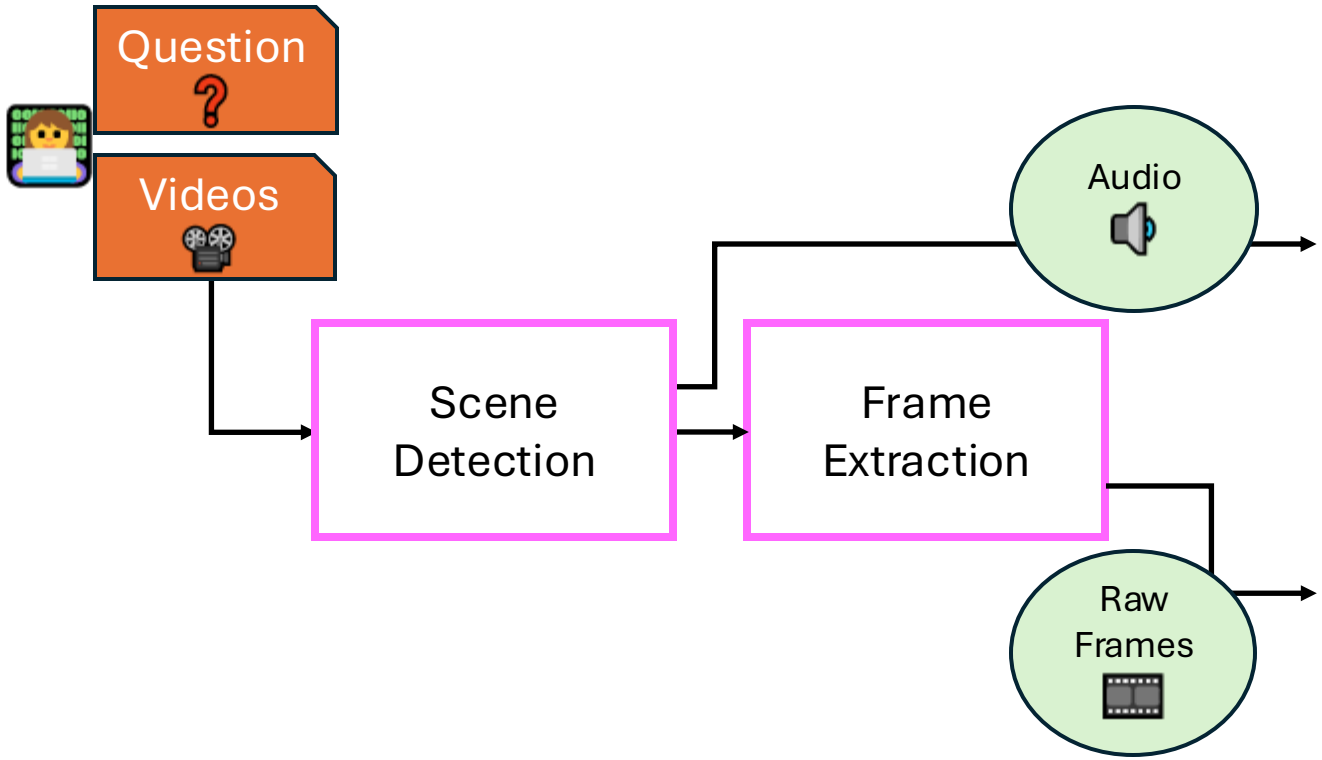
Video Q/A


Question
Videos
Audio
Speech-to-Text (STT)
Text
Scene Detection
Frame Extraction
LLM (multimodal)
Raw Frames
Object Detection
Refined Frames
Final Answer
“What is the name of the person wearing the red dress?”
<road_trip.mp4>

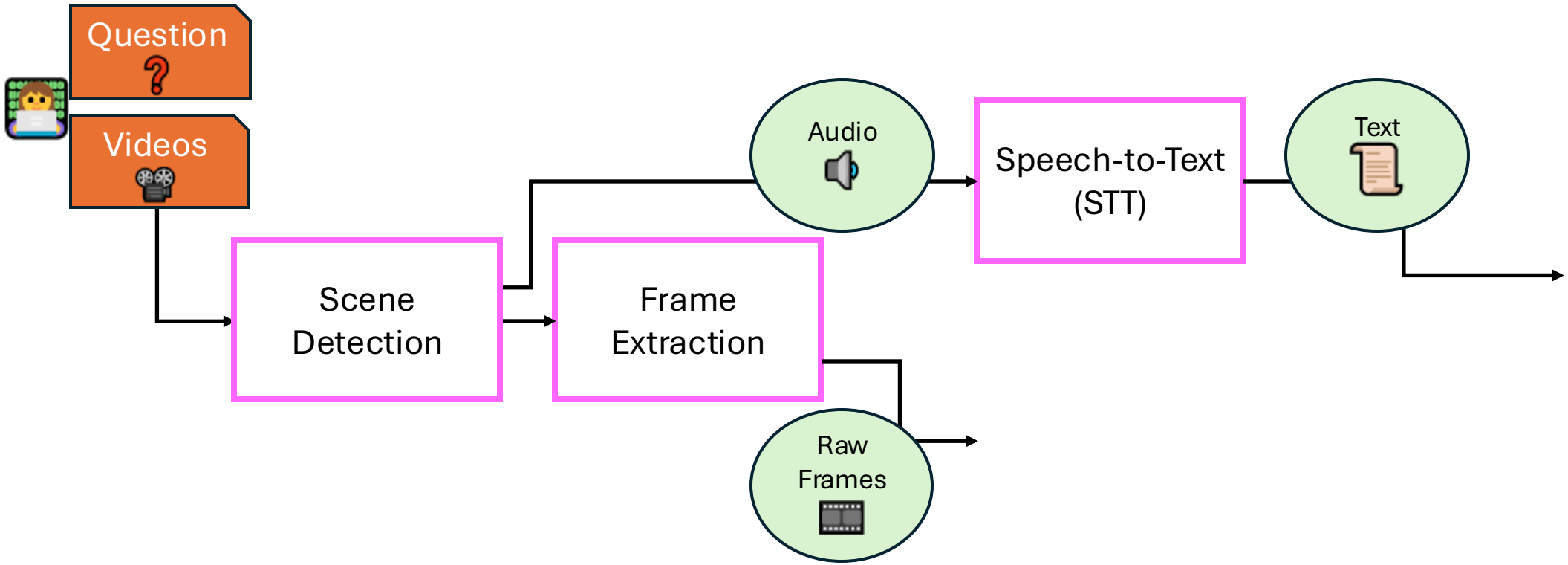
Video Q/A



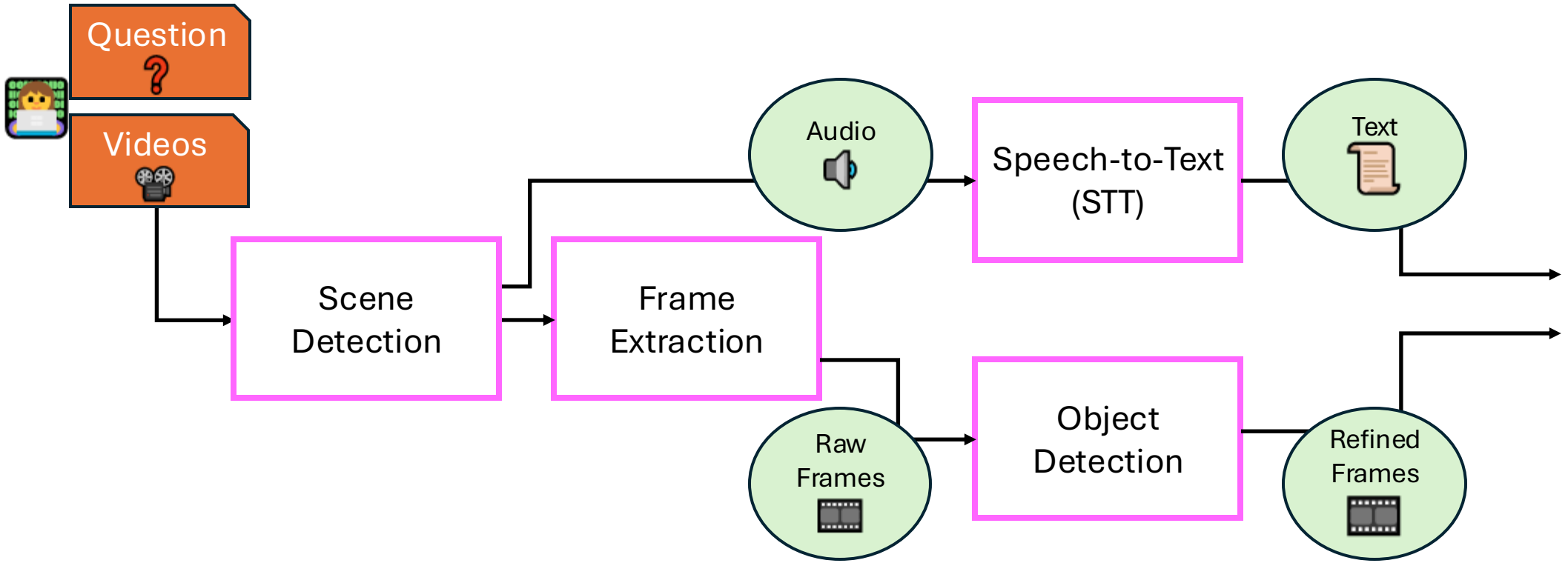
Video Q/A


Question
Videos
Audio
Scene Detection
Frame Extraction
Raw Frames

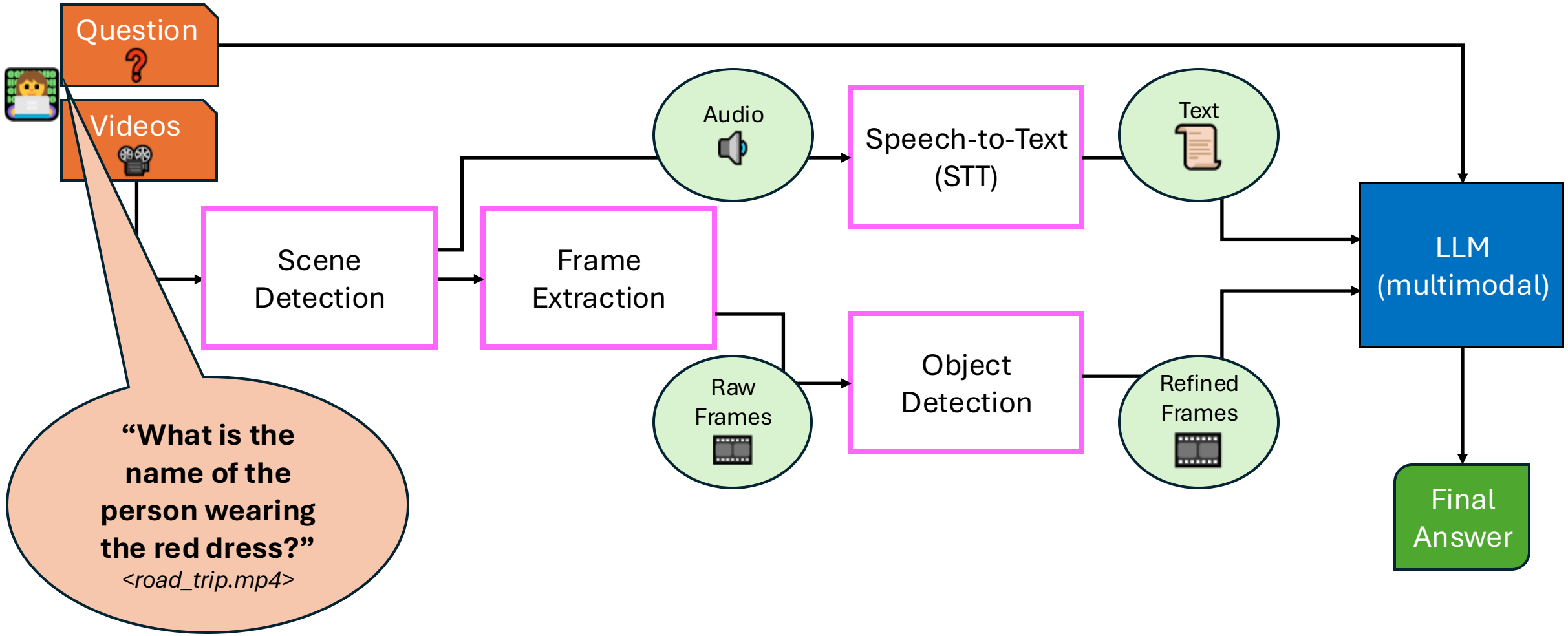
Video Q/A


Question
Videos
Scene Detection
Frame Extraction
Audio
Raw Frames
Speech-to-Text (STT)
Text

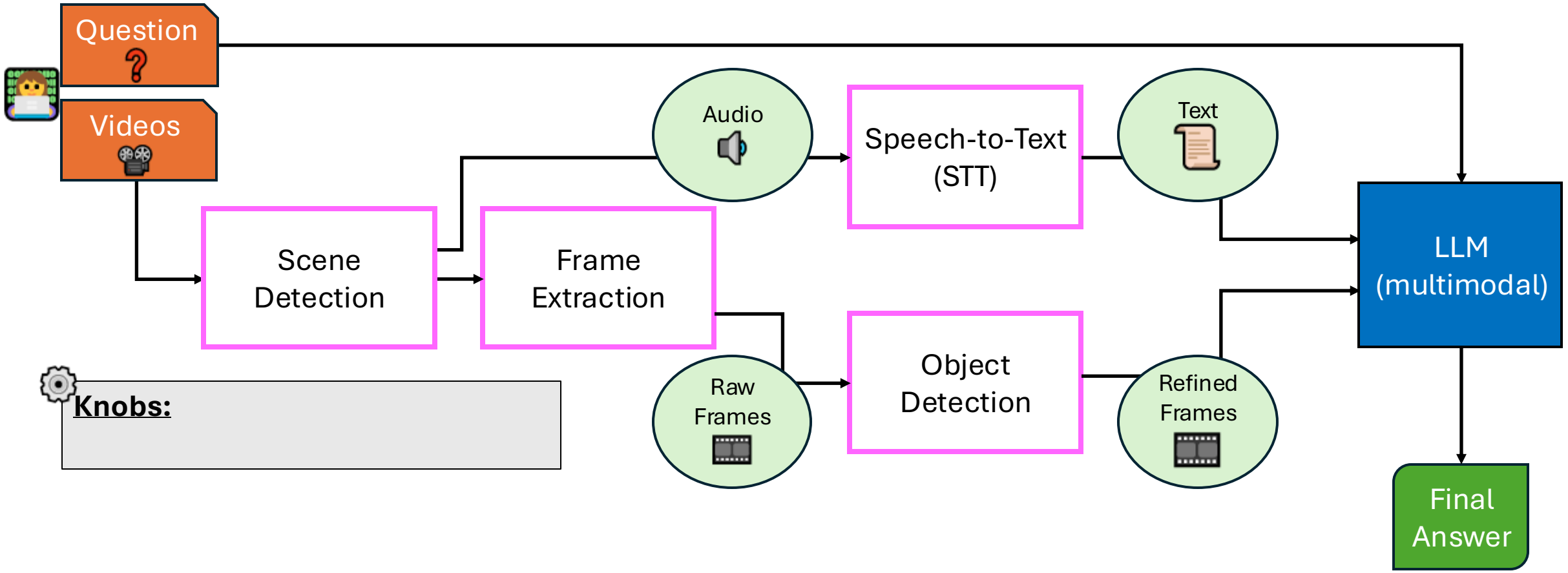
Video Q/A


Question
Videos
Scene Detection
Frame Extraction
Audio
Speech-to-Text (STT)
Text
Raw Frames
Object Detection
Refined Frames

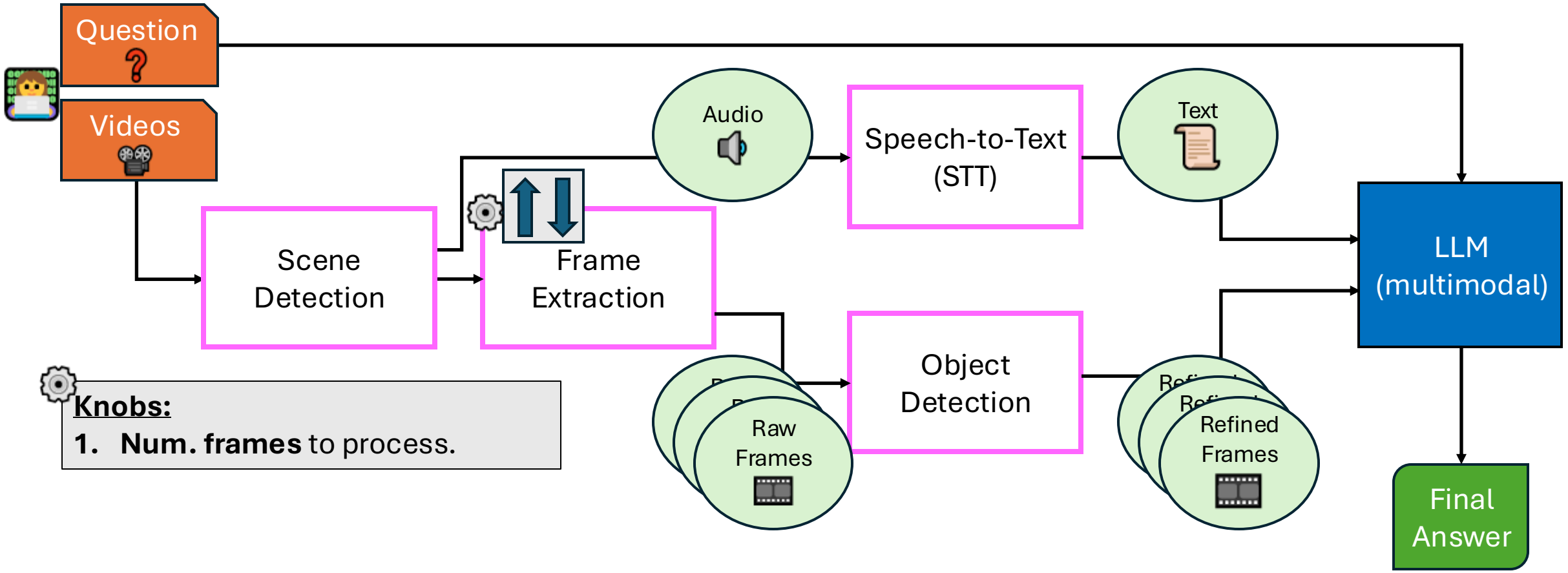
Video Q/A


Question
Videos
"What is the name of the person wearing the red dress?"
<road_trip.mp4>
Scene Detection
Frame Extraction
Audio
Speech-to-Text (STT)
Text
Raw Frames
Object Detection
Refined Frames
LLM (multimodal)
Final Answer

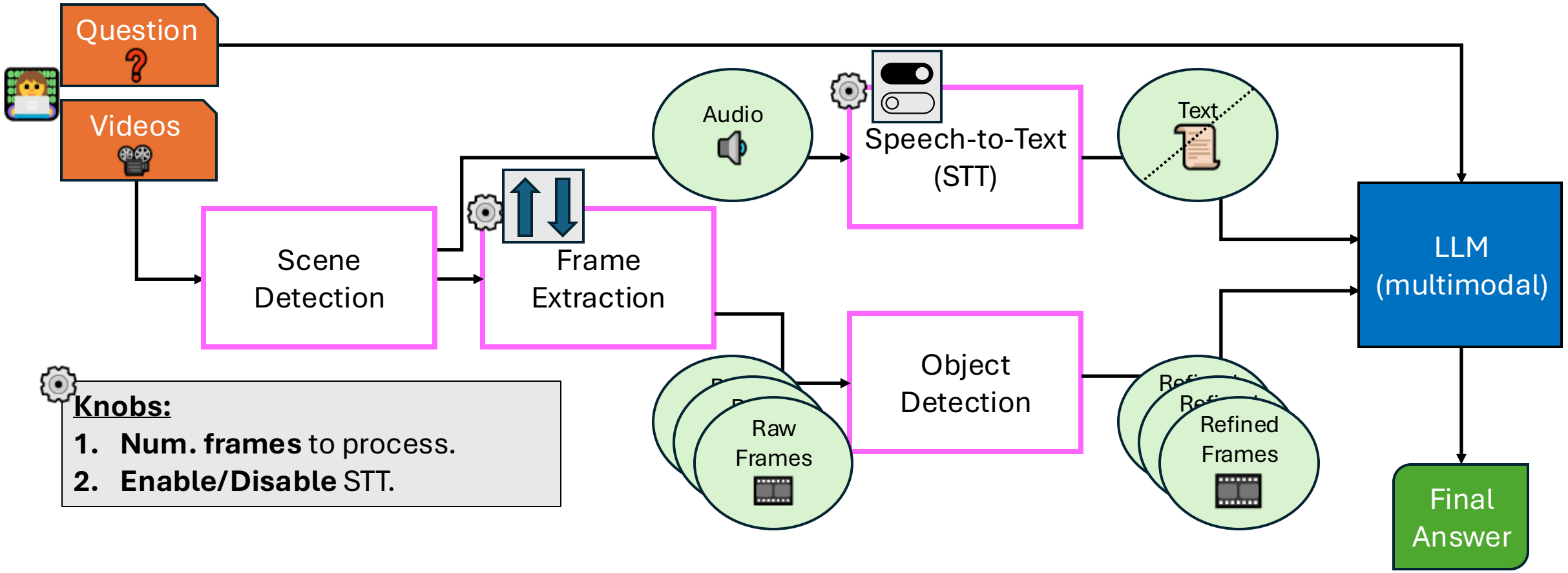
Video Q/A: workflow-specific knobs



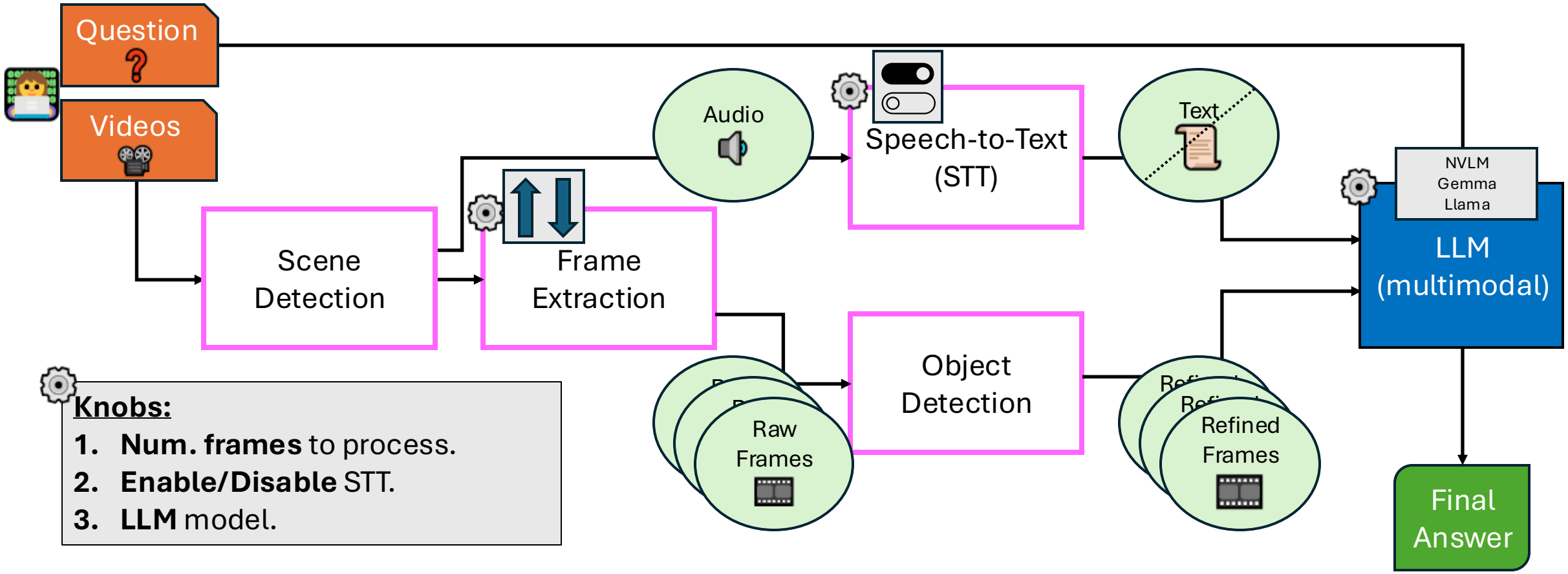
Video Q/A: num. frames



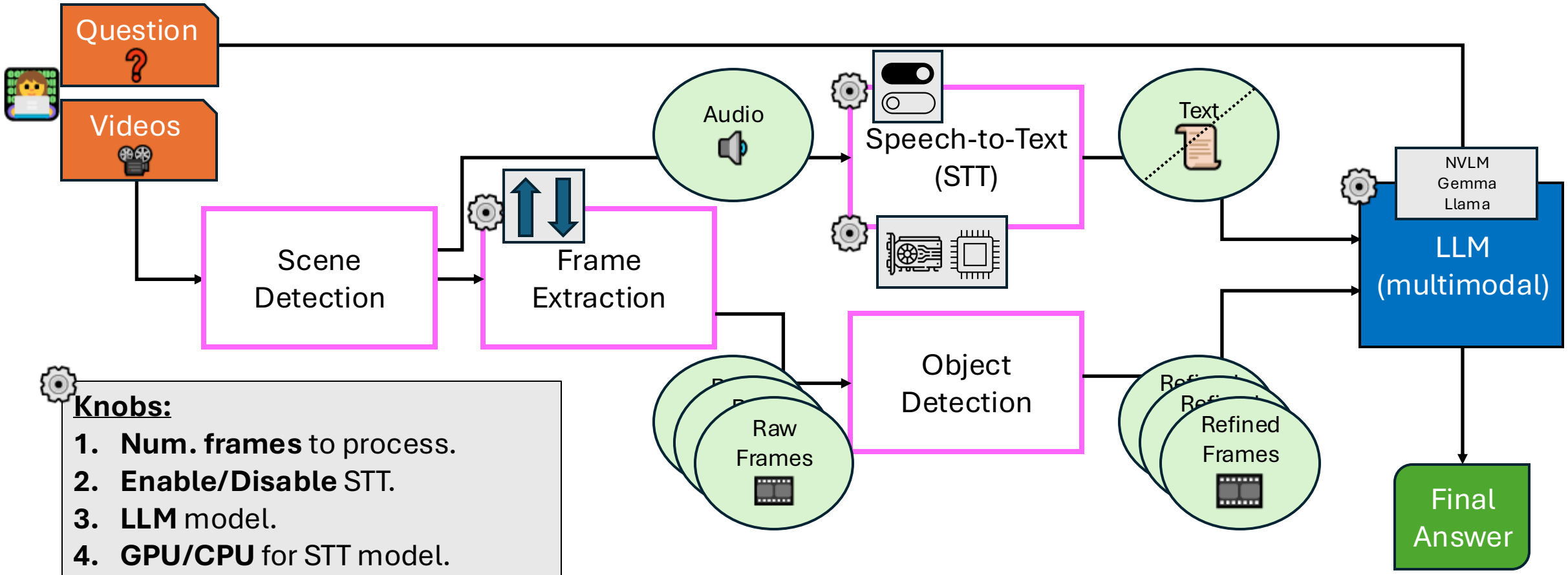
Video Q/A: STT


Question
Videos
Audio
Speech-to-Text (STT)
Text
Scene Detection
Frame Extraction
LLM (multimodal)
Object Detection
Raw Frames
Refined Frames
Final Answer
Knobs:
1. Num. frames to process.
2. Enable/Disable STT.

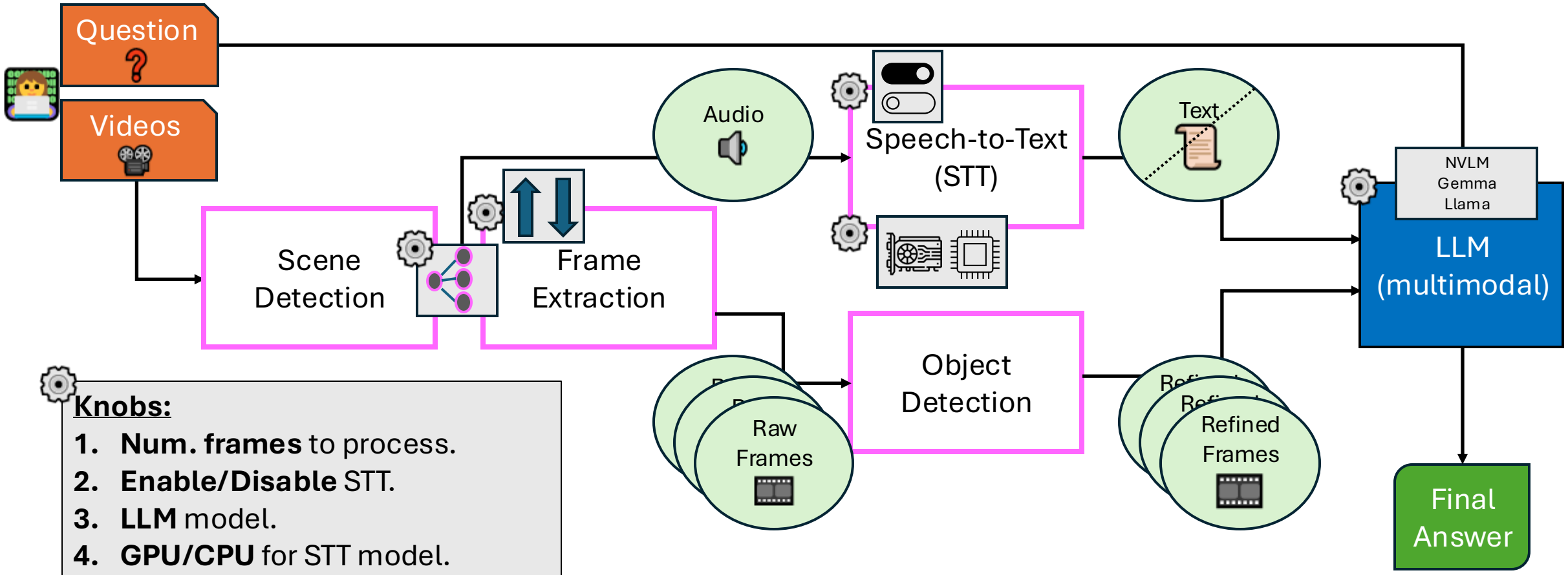
Video Q/A: LLM


Question
Videos
Audio
Speech-to-Text (STT)
Text
NVLM
Gemma
Llama
LLM (multimodal)
Scene Detection
Frame Extraction
Object Detection
Raw Frames
Refined Frames
Final Answer
Knobs:
1. Num. frames to process.
2. Enable/Disable STT.
3. LLM model.

Video Q/A: hardware configuration



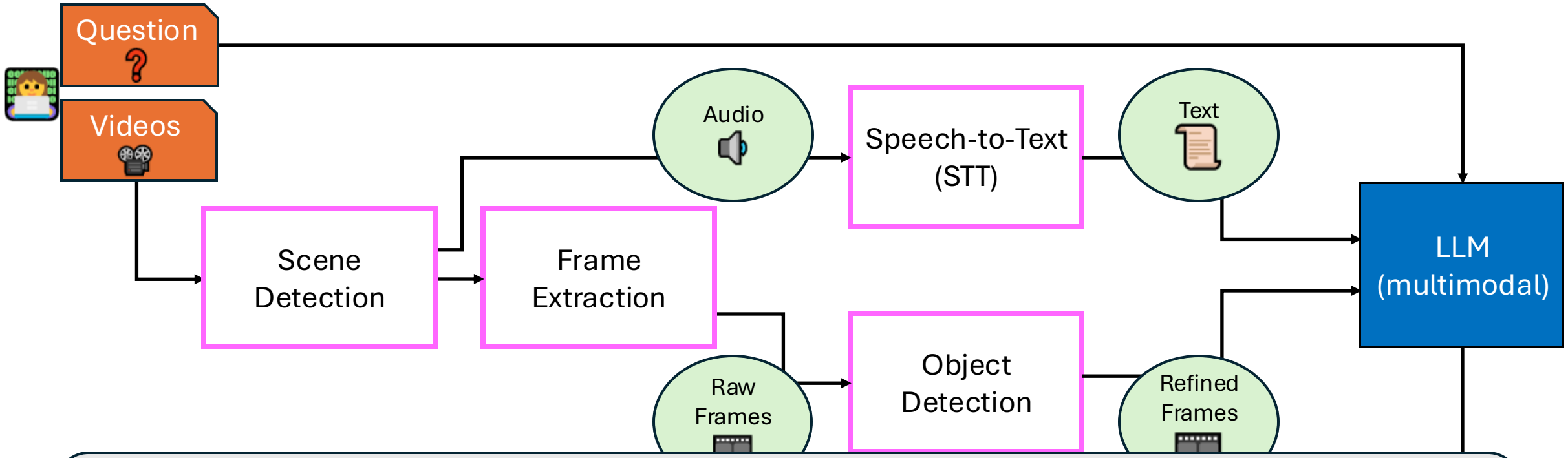
Video Q/A: parallelism



Knobs:

1. **Num. frames** to process.
2. **Enable/Disable** STT.
3. **LLM** model.
4. **GPU/CPU** for STT model.
5. **Parallelize** scene processing.

Video Q/A

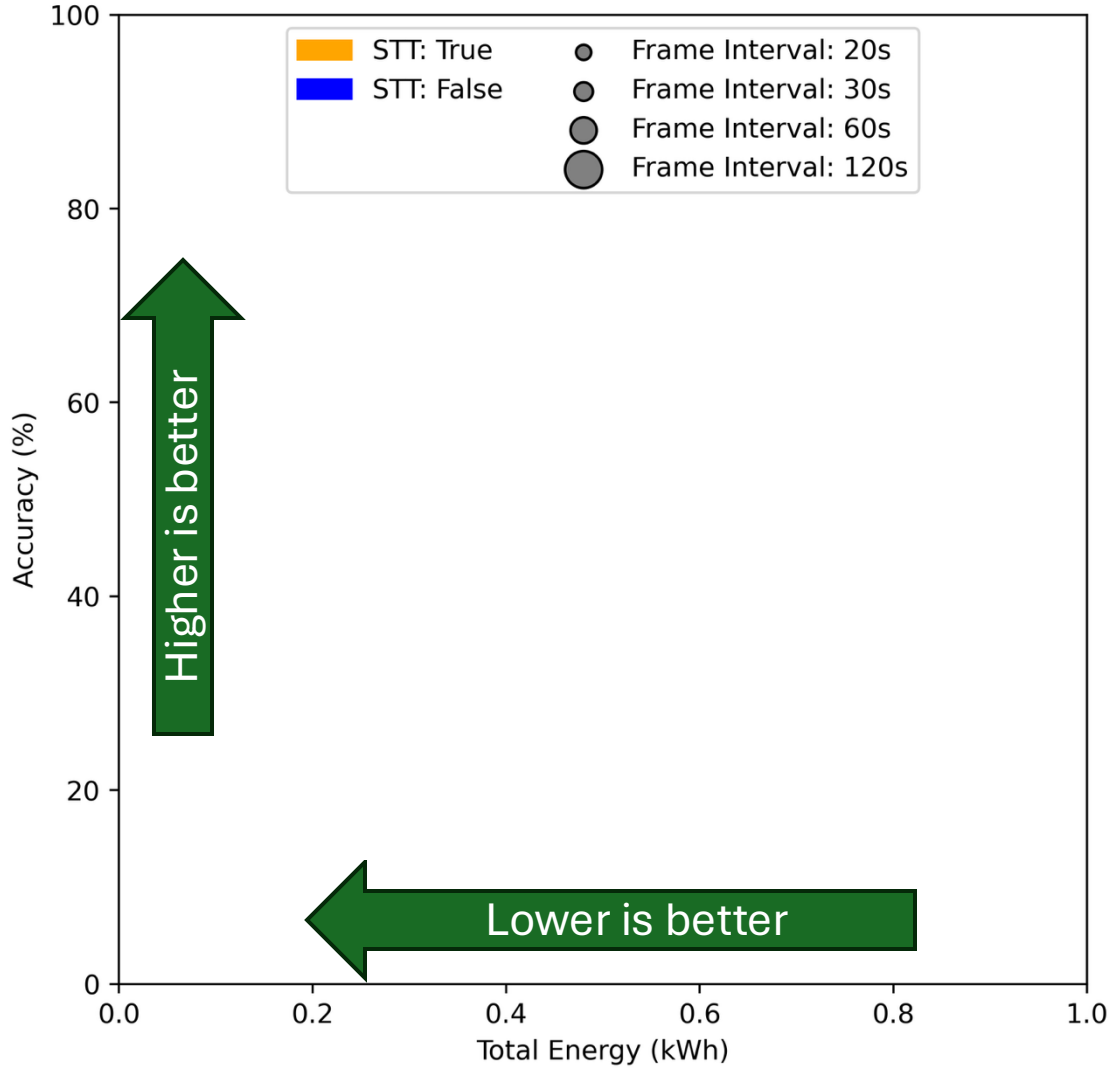


Many possible configurations for a single workflow.

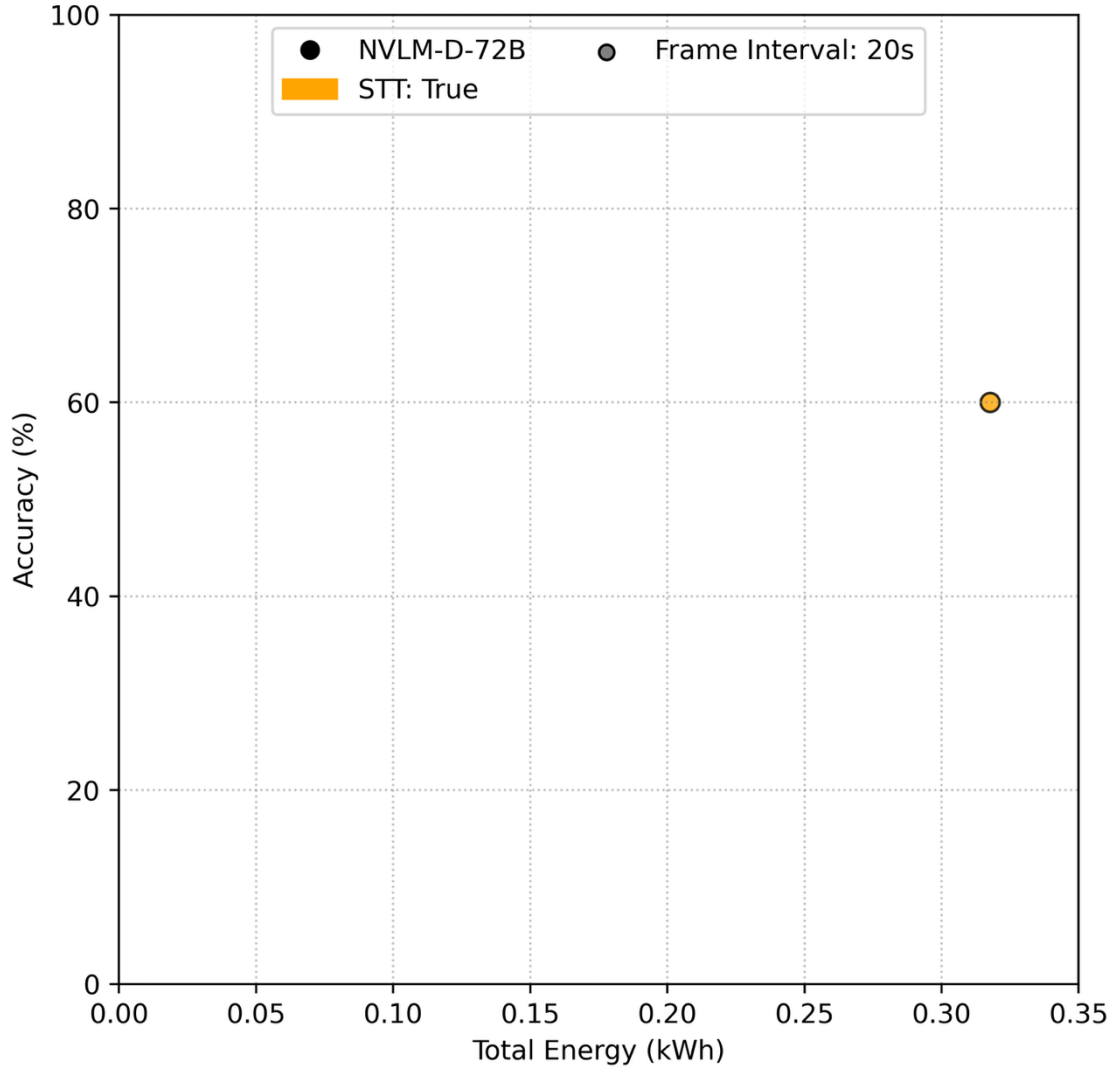
Outline

1. Compound AI workflow: Video Q/A
   a. Possible configurations.
   b. Workflow implementation.
   c. Workflow execution.
2. Our vision.
3. Murakkab.

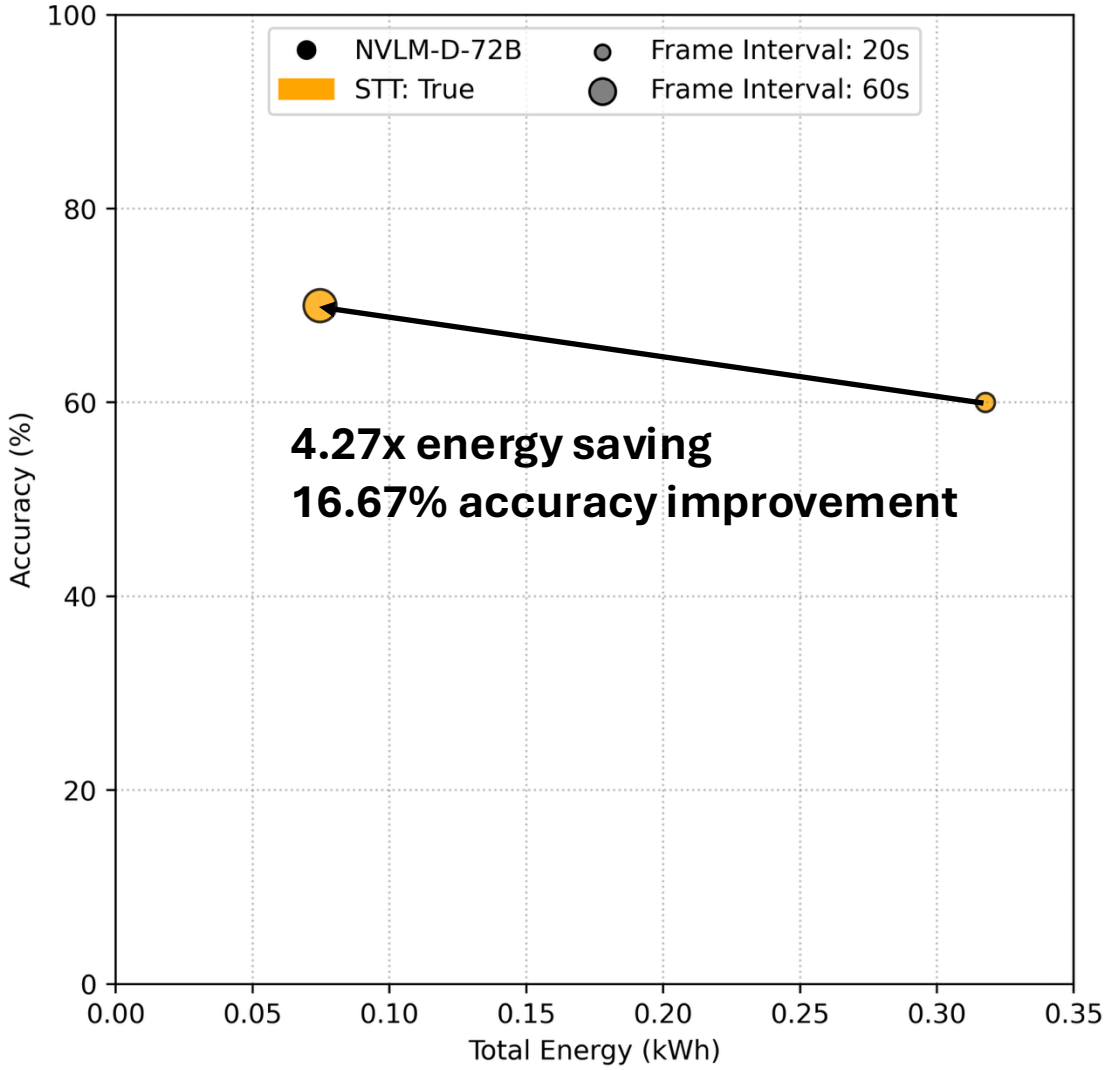
Video Q/A: configuring with only 2 knobs



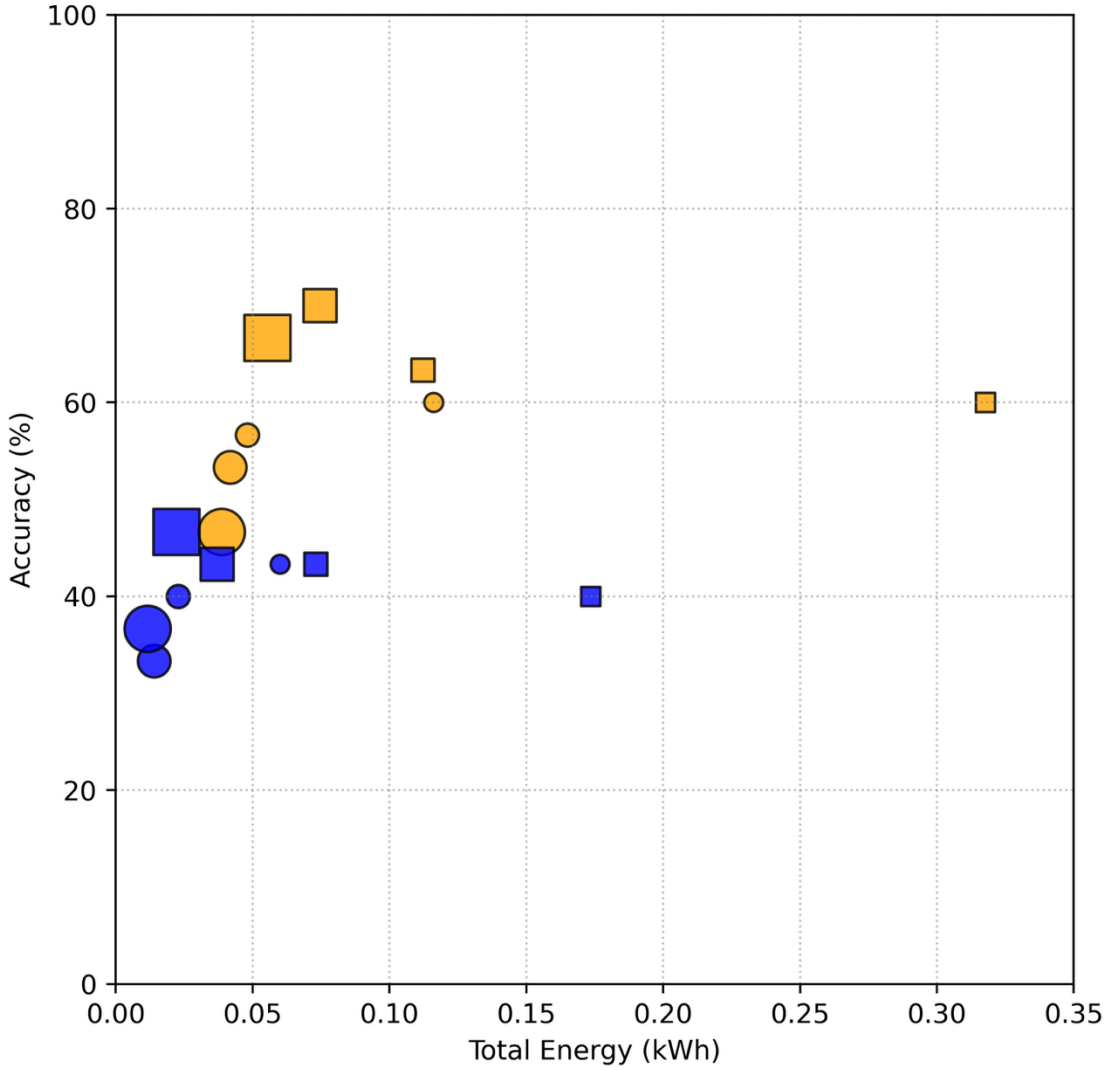
Video Q/A: manual configuration



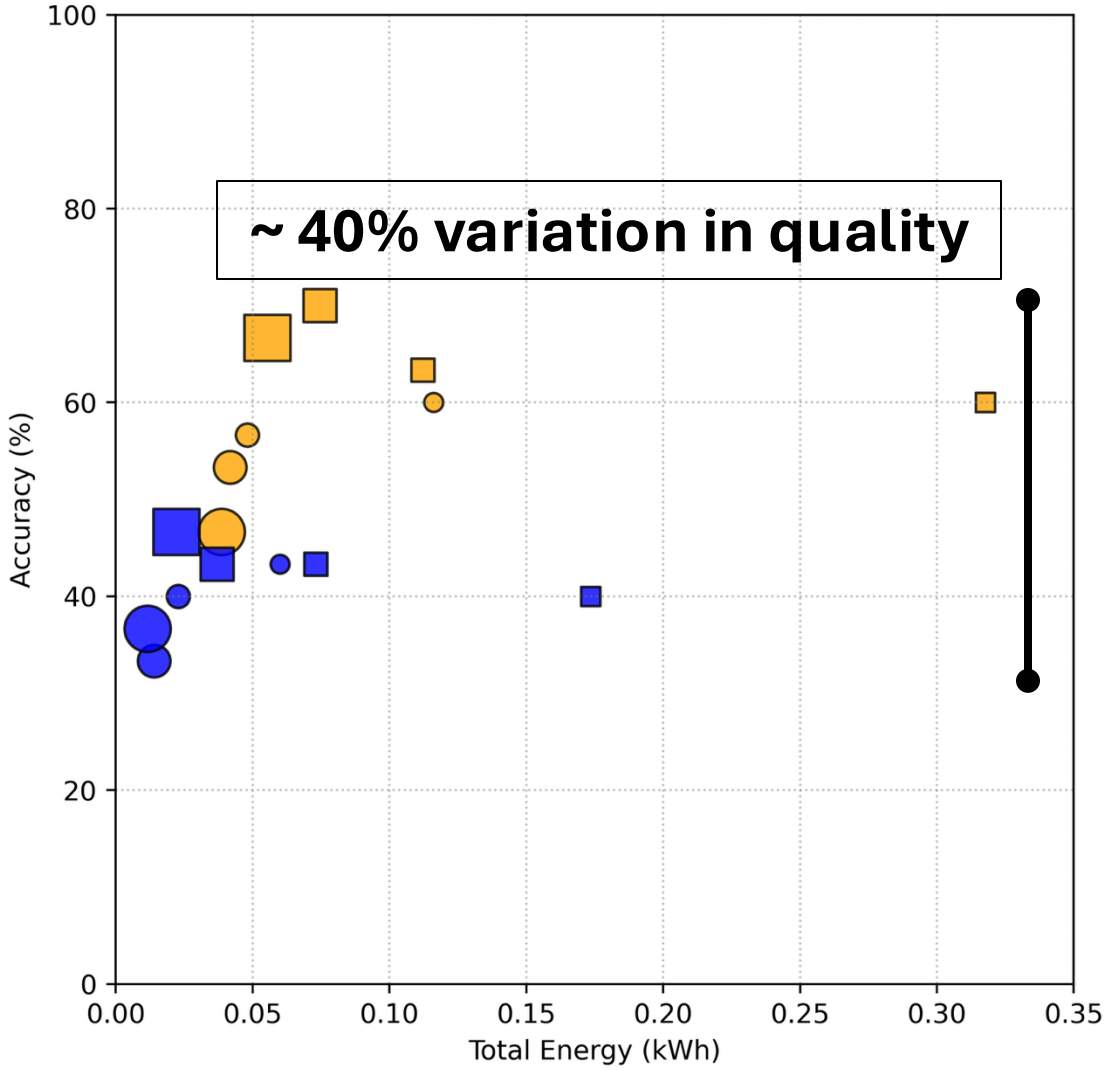
Video Q/A: optimized configuration



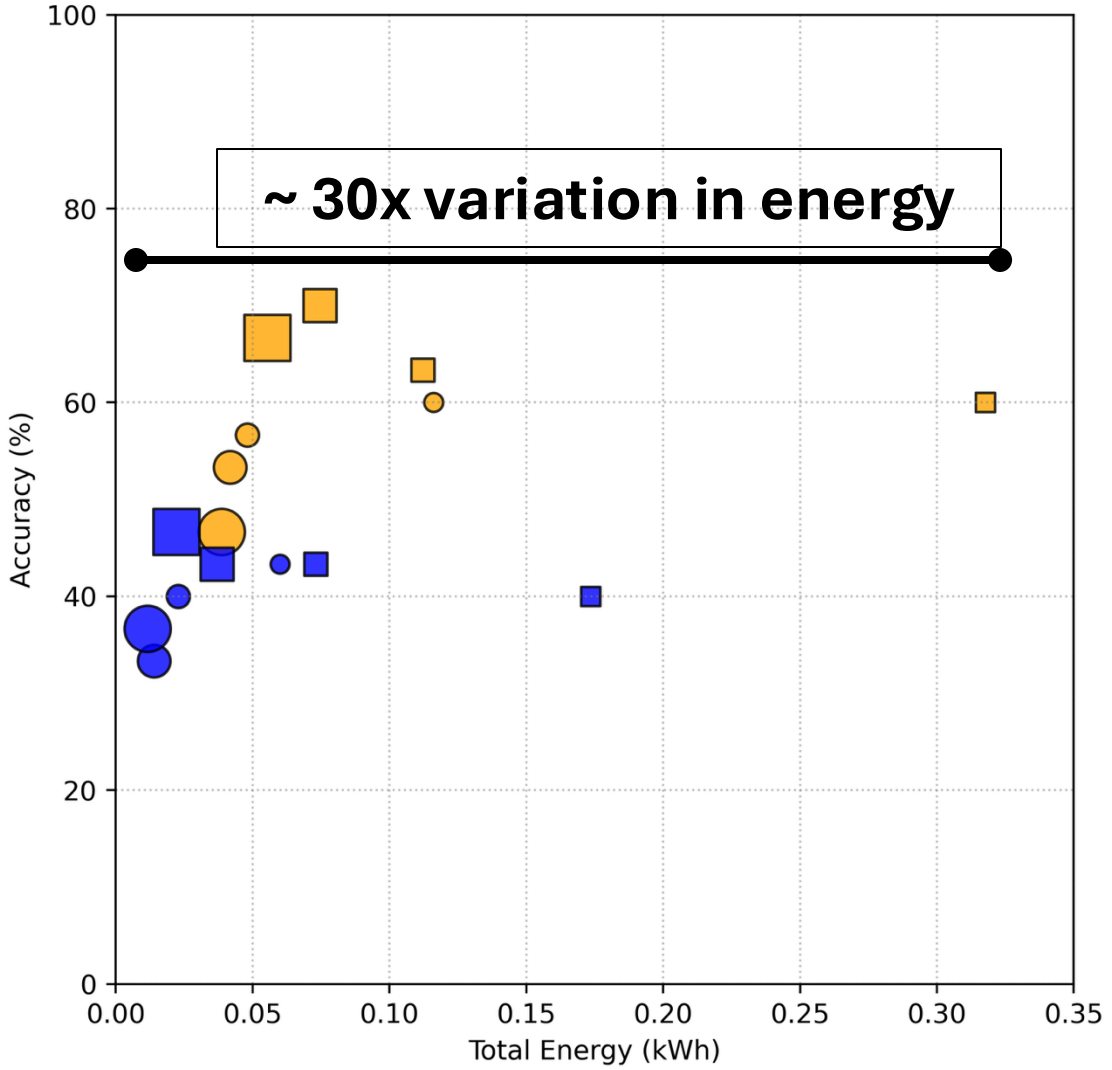
Many configurations with only 2 knobs!



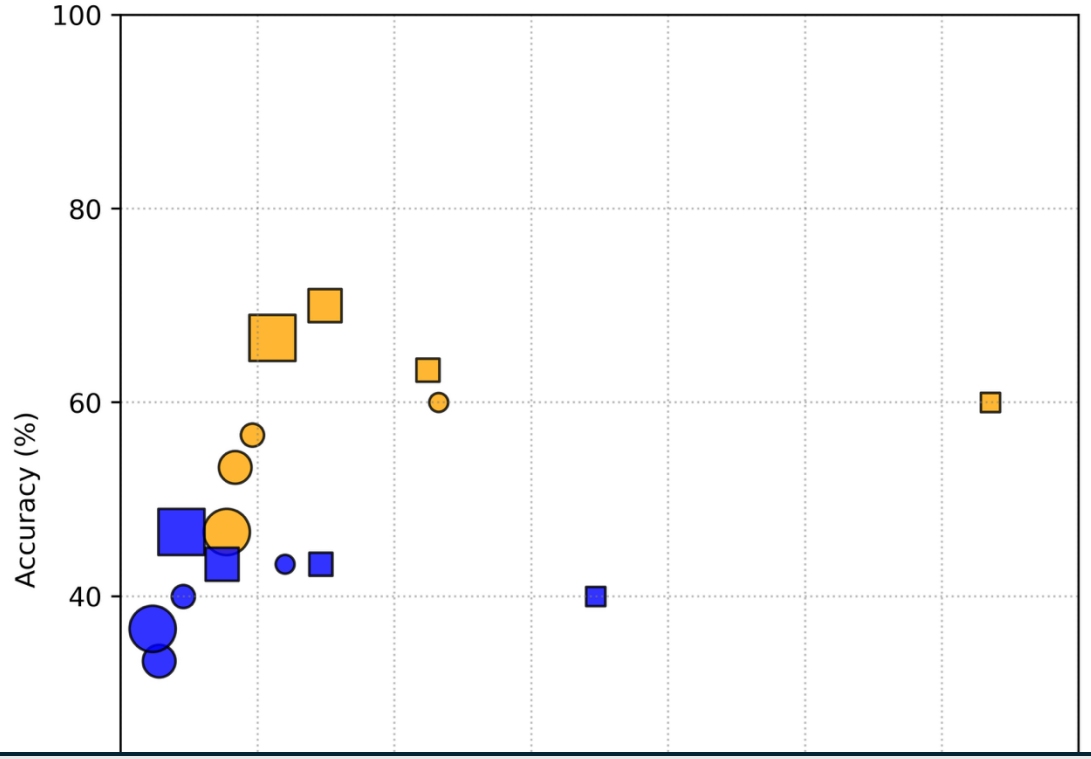
Many configurations with only 2 knobs!



Many configurations with only 2 knobs!



Many configurations with only 2 knobs!



Manual workflow configuration is suboptimal.

Outline

1. Compound AI workflow: Video Q/A
   a. Possible configurations.
   b. Workflow implementation.
   c. Workflow execution.
2. Our vision.
3. Murakkab.

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Models/Tools

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Models/Tools

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Task-specific parameters

Models/Tools

Resources

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Task-specific parameters

Models/Tools

Resources

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Task-specific parameters

Multiple providers

Models/Tools

Resources

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Task-specific parameters

Multiple providers

Application logic

Models/Tools

Resources

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
```

Task-specific parameters

Multiple providers

Application logic

Tightly coupled configuration and application logic.

Outline

1. Compound AI workflow: Video Q/A
    a. Possible configurations.
    b. Workflow implementation.
    c. Workflow execution.
2. Our vision.
3. Murakkab.

Workflow execution today

```
Workflow(
      models=[A,B,C],
      api_keys=[foo,bar],
      stages=[S1,S2,S3],
      knobs={x:5,y:True}
)
```

Explicitly defined workflow logic coupled with configuration details.

Workflow execution today

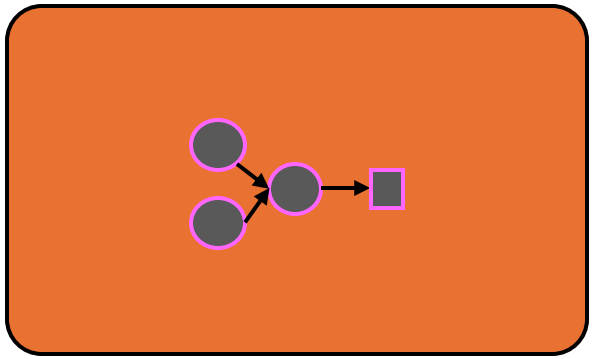
AutoGen

A framework for building AI agents and applications

LangChain

```
Workflow(
      models=[A,B,C],
      api_keys=[foo,bar],
      stages=[S1,S2,S3],
      knobs={x:5,y:True}
)
```

Explicitly defined workflow logic coupled with configuration details.

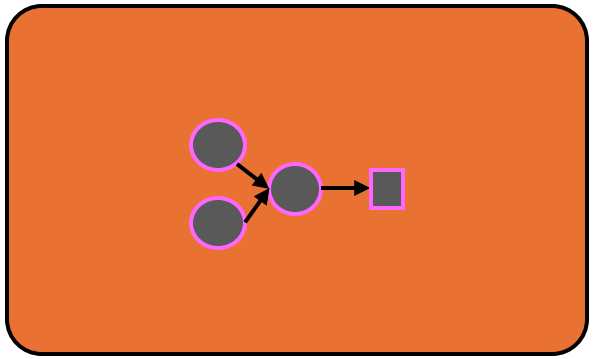


Static DAG with fixed model(s) and other knobs.

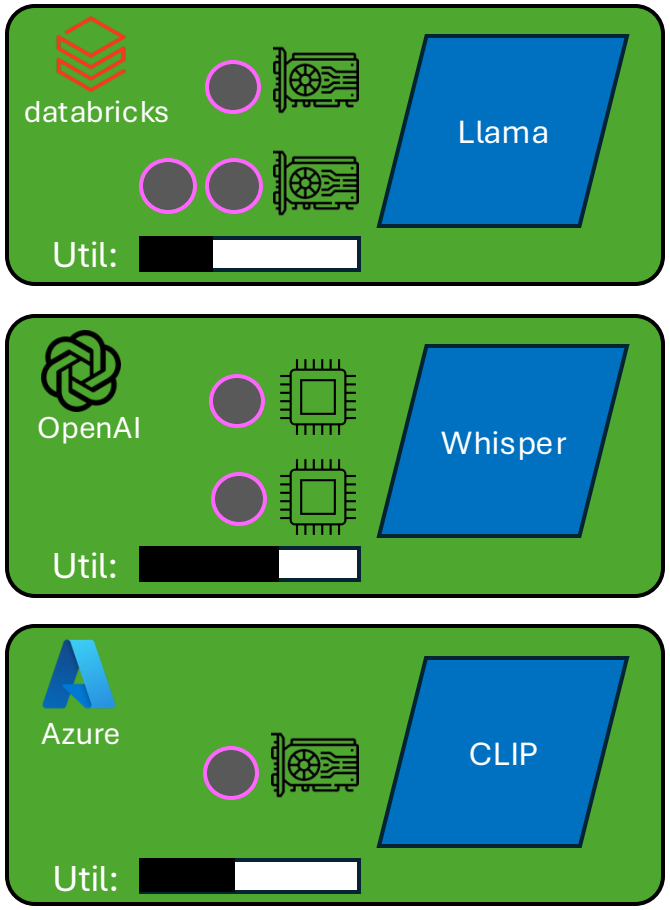
Workflow execution today


```
Workflow(
      models=[A,B,C],
      api_keys=[foo,bar],
      stages=[S1,S2,S3],
      knobs={x:5,y:True}
)
```

Explicitly defined workflow logic coupled with configuration details.

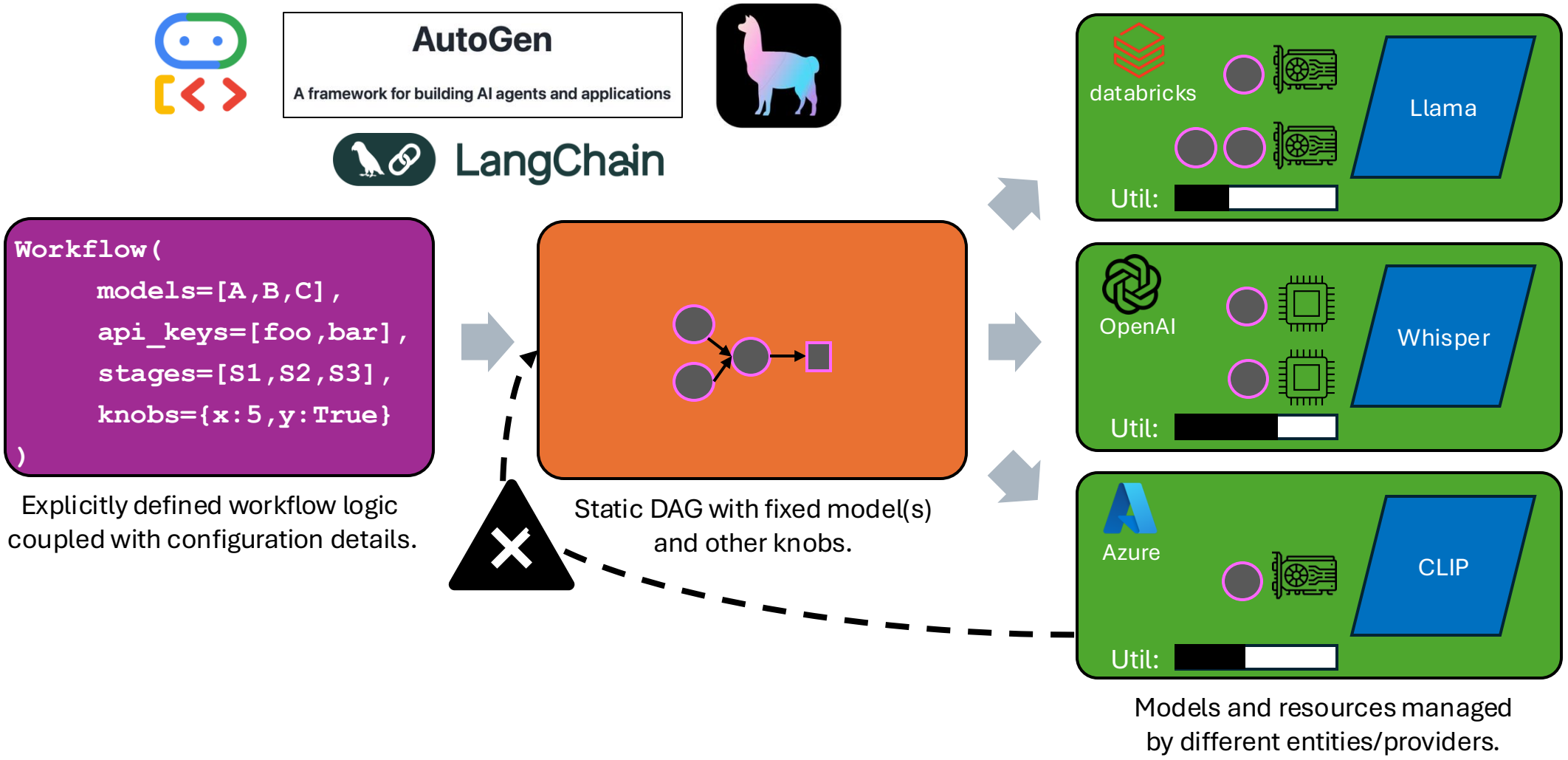


Static DAG with fixed model(s) and other knobs.

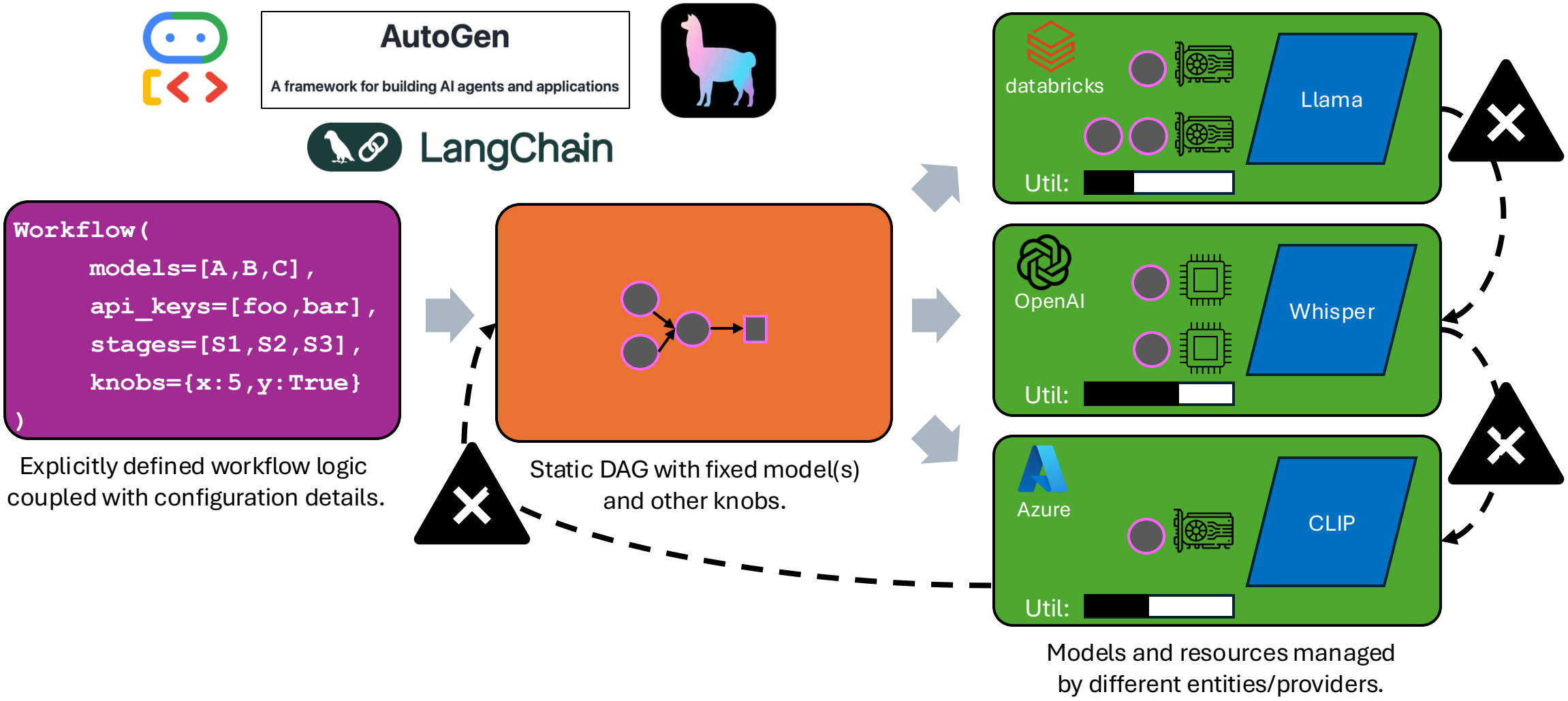


Models and resources managed by different entities/providers.

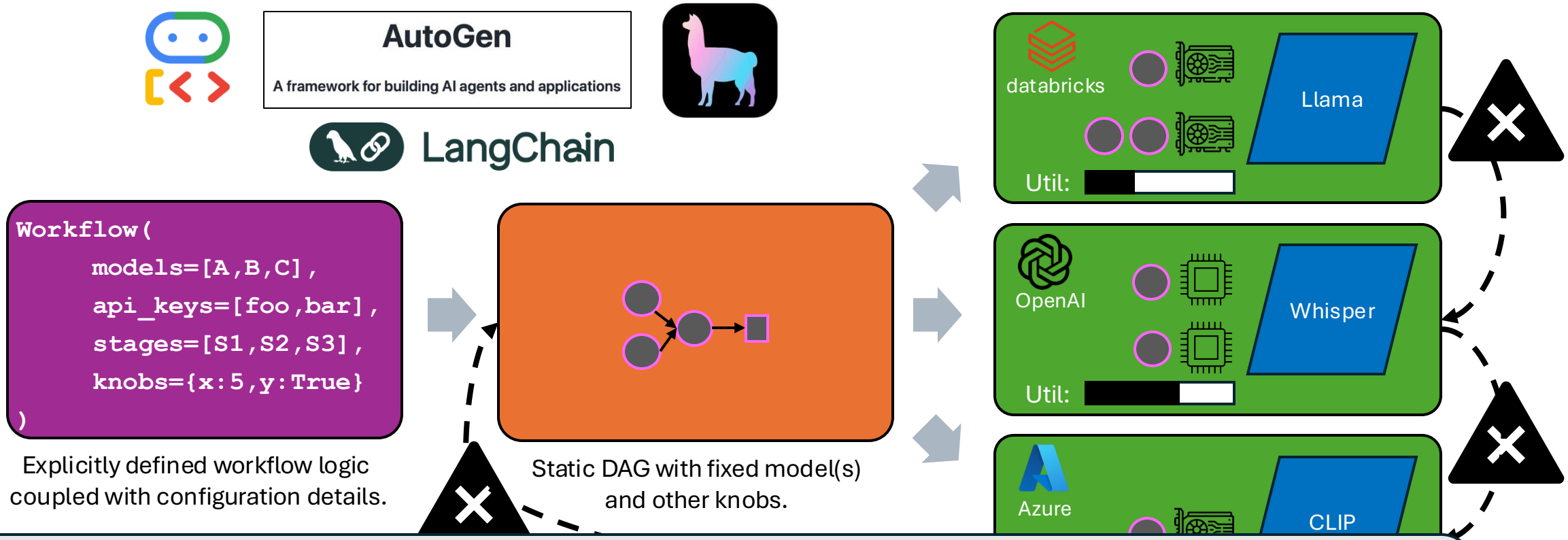
Workflow execution today



Workflow execution today



Workflow execution today

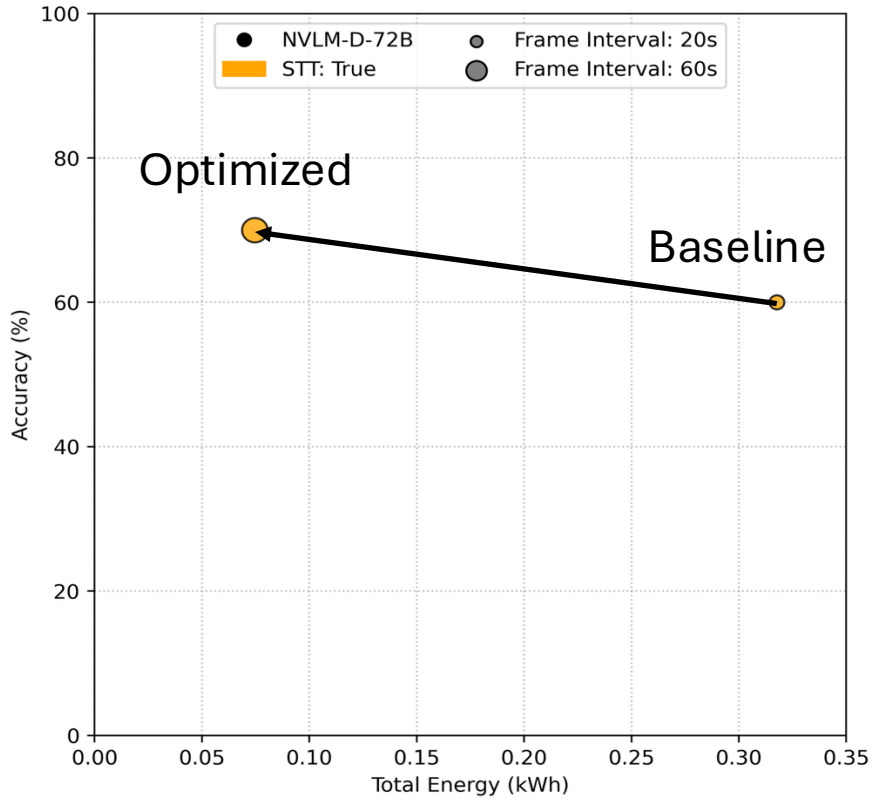


Fragmented workflows spanning different entities with limited inter-layer visibility.

Are these systems efficient?

Are these systems efficient?

NO!



Are these systems efficient?

NO!

Achieving high efficiency while satisfying SLOs is difficult in today's compound AI systems.

Why?

Why?

1. **Rigid workflows:**
   a. Tight coupling between configurations and application logic.

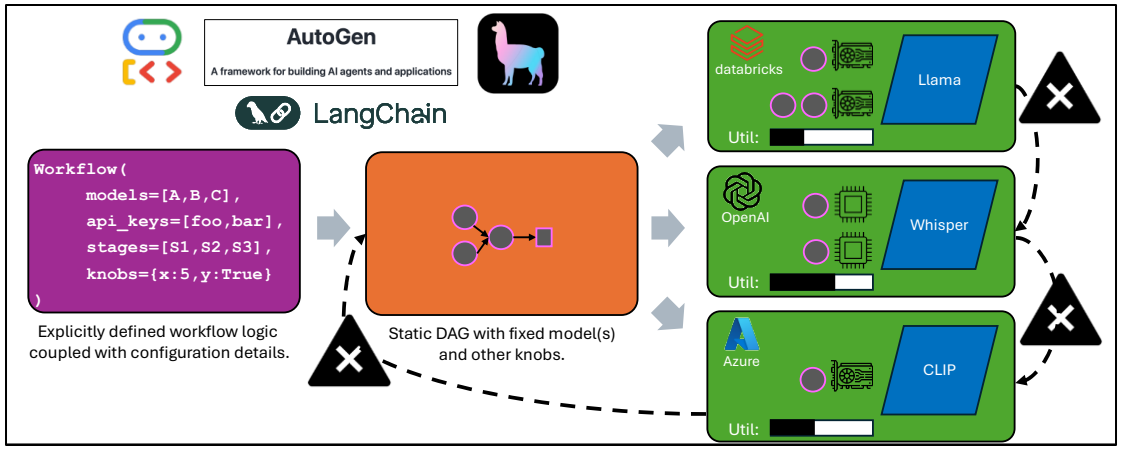
```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

Why?

1. **Rigid workflows:**
   a. Tight coupling between configurations and application logic.

```
/* ===== Nodes of the workflow (simplified) ===== */
scene_detection  = Tool(name="OpenCV", key=AWS_SSH_KEY,
                        params={num_frames: 15},
                        resources={CPUs: 32})

speech_to_text   = MLModel(name="Whisper", key=OPENAI_API_KEY,
                           resources={PTUs: 50})

object_detection = MLModel(name="CLIP", key=AZURE_SSH_KEY,
                           resources={CPUs: 2})

question_answer  = LLM(name="Llama", key=DATABRICKS_API_KEY,
                       params={batch_size: 256},
                       resources={GPUs: 4,GPU_Type: H100},
                       system_prompt="You are an agent that can understand videos.",
                       user_prompt="Answer the given question about the video.")

/* ===== Query and data ===== */
question = "What is the name of the person wearing the red dress?"
videos   = ["road_trip.mp4"]

/* ===== Workflow (ordering of nodes/data flow) ===== */
audio, frames  = scene_detection(videos)
transcript     = speech_to_text(audio)
refined_frames = object_detection(frames)
answer         = question_answer(question, transcript, frames)
```

2. **Limited cross-layer visibility:**
   a. Workflow orchestration and resource management.
   b. Multiple model/tool providers.

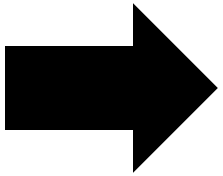


Outline

1. Compound AI workflow: Video Q/A
    a. Possible configurations.
    b. Workflow implementation.
    c. Workflow execution.
2. Our vision.
3. Murakkab.

Rigid Workflows

Limited Inter-Layer Visibility



Fungible Workflows

Unified Compound AI Systems

Poor efficiency!

High efficiency!

Idea 1: fungible workflows

Idea 1: fungible workflows

- High level descriptions (in natural language):

```
job_description = "What is the name of the person wearing the red dress?"
inputs          = ["road_trip.mp4"]


result          = run(job_description, inputs,     )
```

Idea 1: fungible workflows

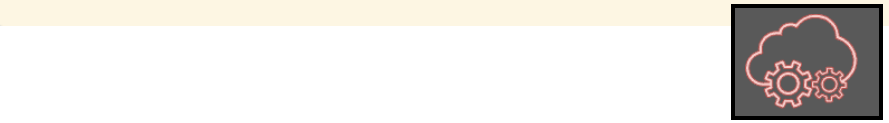
- High level descriptions (in natural language) and SLOs:

```
job_description = "What is the name of the person wearing the red dress?"
inputs          = ["road_trip.mp4"]
SLOs            = {cost: "<$2"}

result          = run(job_description, inputs, SLOs)
```

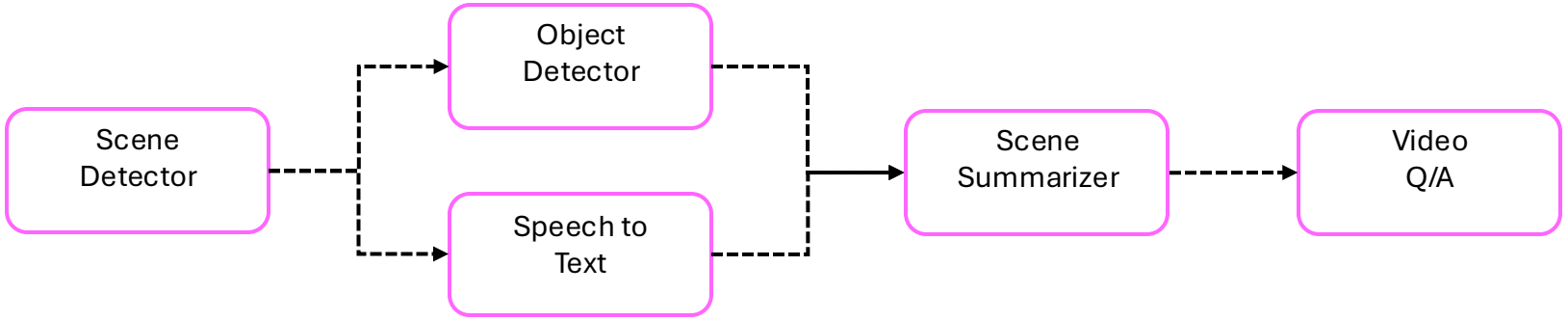
Idea 1: fungible workflows

- High level descriptions (in natural language) and SLOs:

```
job_description = "What is the name of the person wearing the red dress?"
inputs          = ["road_trip.mp4"]
SLOs            = {cost: "<$2"}

result          = run(job_description, inputs, SLOs)
```



LLM Planner



Idea 1: fungible workflows

- High level descriptions (in natural language) and SLOs:

```
job_description = "What is the name of the person wearing the red dress?"
inputs          = ["road_trip.mp4"]
SLOs            = {cost: "<$2"}

result          = run(job_description, inputs, SLOs)
```

LLM Planner

Object

Decouple configuration from application logic.

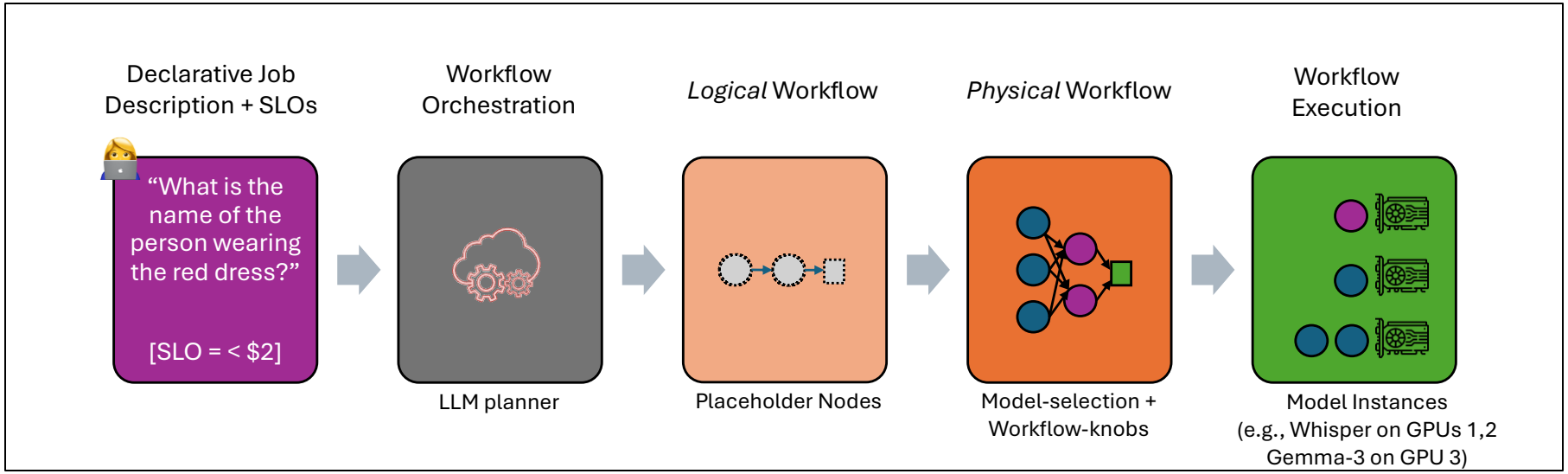
Idea 2: unified compound AI system

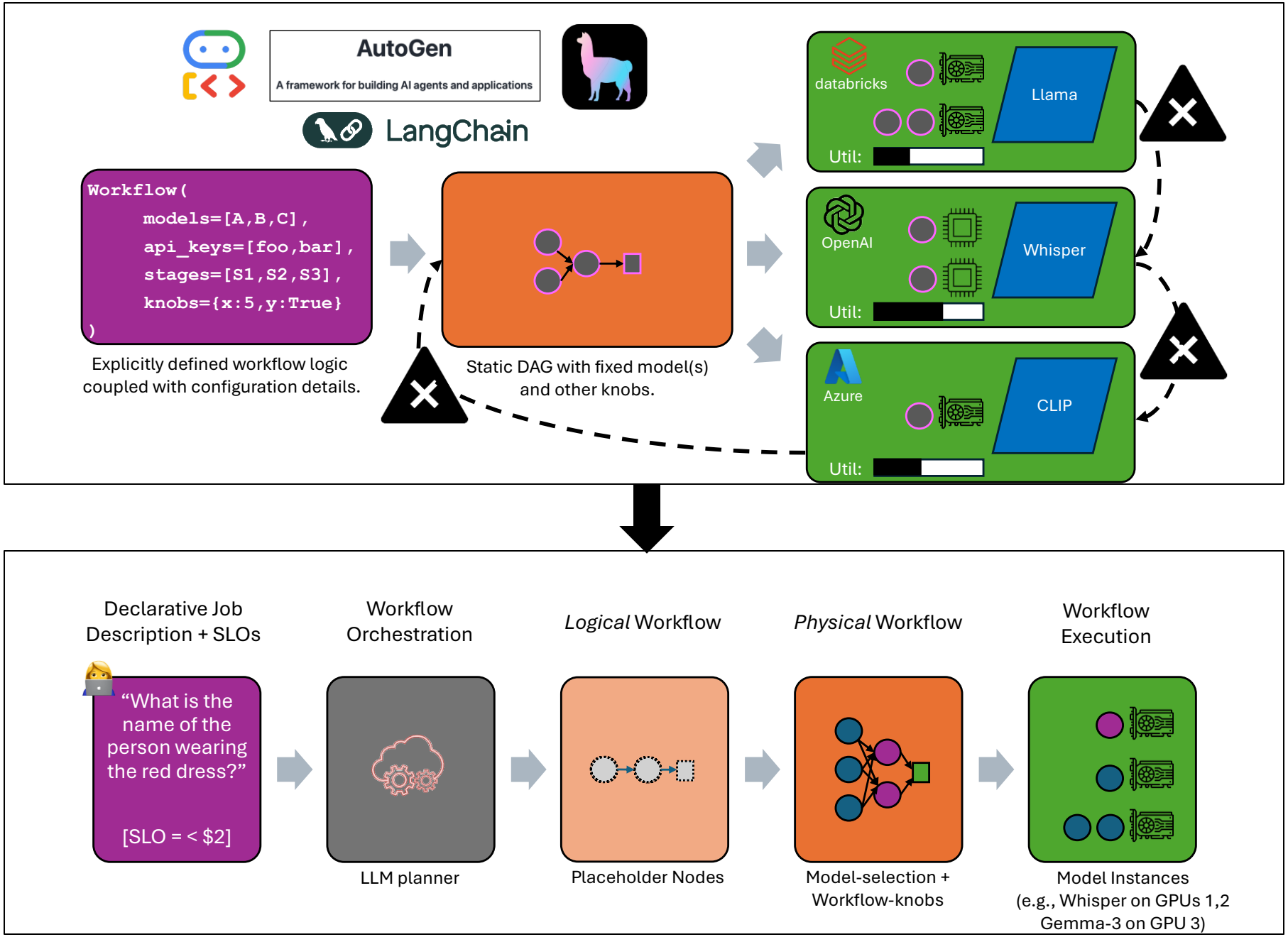
Idea 2: unified compound AI system

- Single entity manages:
  - Workflow orchestration
  - Models/tools and resources

Idea 2: unified compound AI system

- Single entity manages:
  - Workflow orchestration
  - Models/tools and resources




AutoGen
A framework for building AI agents and applications
LangChain
Workflow(
models=[A,B,C],
api_keys=[foo,bar],
stages=[S1,S2,S3],
knobs={x:5,y:True}
)
Explicitly defined workflow logic coupled with configuration details.
Static DAG with fixed model(s) and other knobs.
databricks
Llama
Util:
OpenAI
Whisper
Util:
Azure
CLIP
Util:
Declarative Job Description + SLOs
"What is the name of the person wearing the red dress?"
[SLO = < $2]
Workflow Orchestration
LLM planner
Logical Workflow
Placeholder Nodes
Physical Workflow
Model-selection + Workflow-knobs
Workflow Execution
Model Instances
(e.g., Whisper on GPUs 1,2
Gemma-3 on GPU 3)

Outline

1. Compound AI workflow: Video Q/A
   a. Possible configurations.
   b. Workflow implementation.
   c. Workflow execution.
2. Our vision.
3. Murakkab.

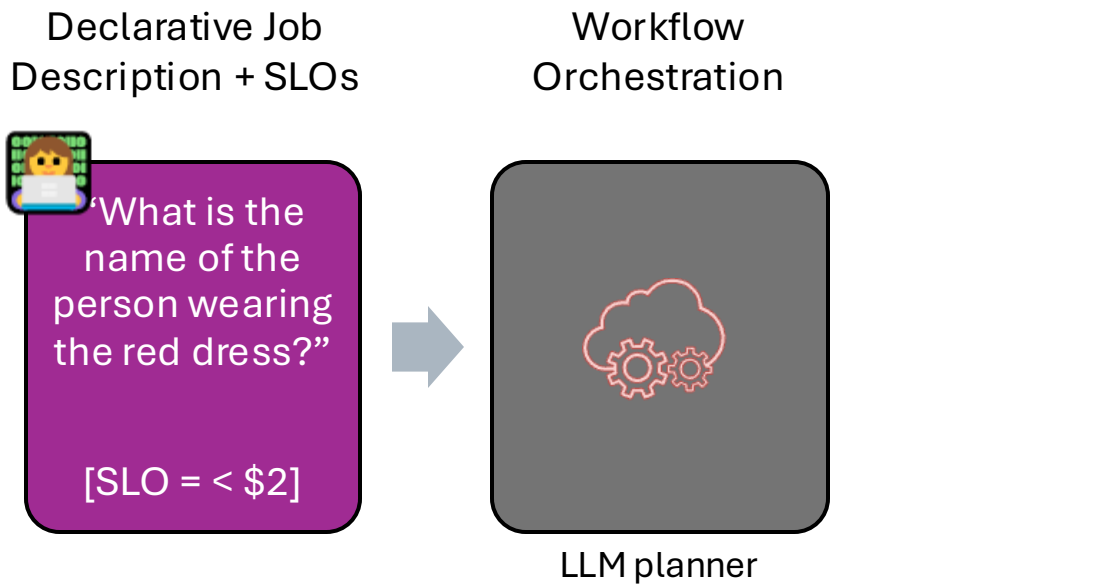
Murakkab

Declarative Job Description + SLOs

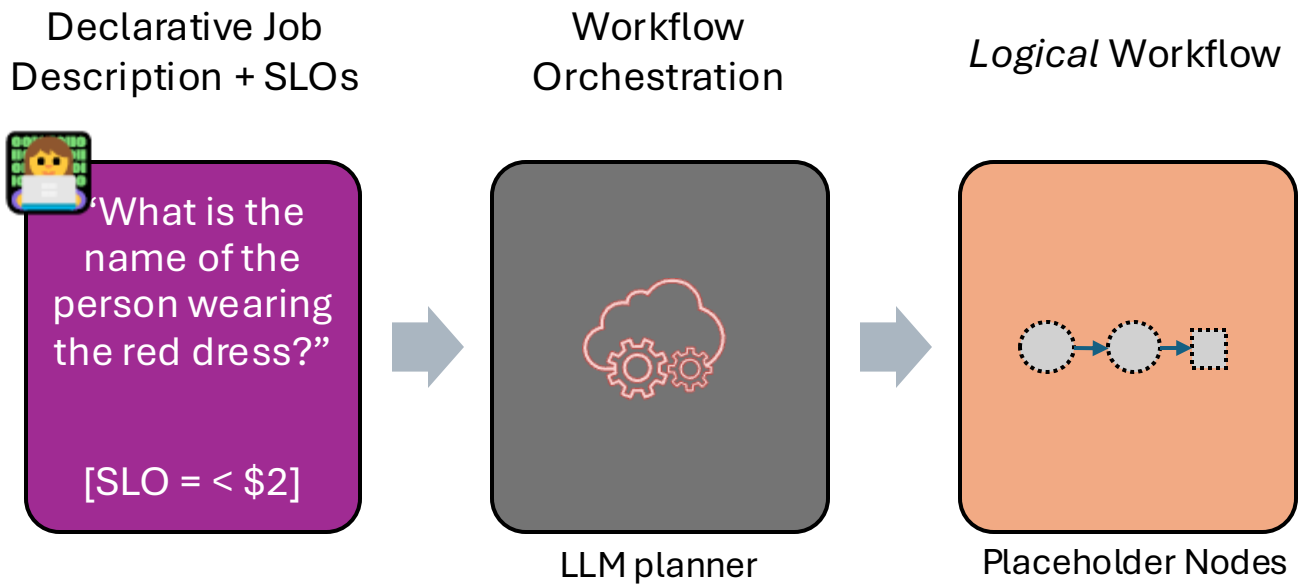
"What is the name of the person wearing the red dress?"

[SLO = < $2]

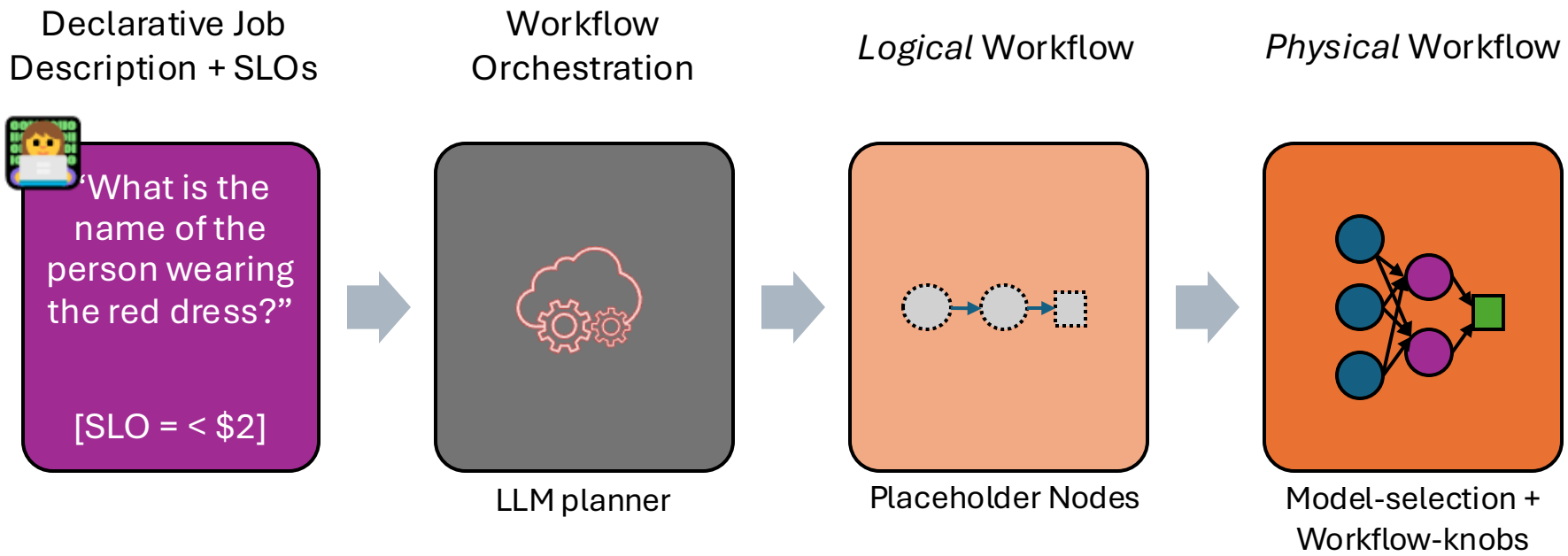
Murakkab


Declarative Job Description + SLOs
Workflow Orchestration
"What is the name of the person wearing the red dress?"
[SLO = < $2]
LLM planner
Agents and Tools Library
(LLMs, ML models, tools)
[e.g., DeepSeek-R1, Whisper, Web Browser API]

Murakkab


Declarative Job Description + SLOs
Workflow Orchestration
Logical Workflow
"What is the name of the person wearing the red dress?"
[SLO = < $2]
LLM planner
Placeholder Nodes
Agents and Tools Library
(LLMs, ML models, tools)
[e.g., DeepSeek-R1, Whisper, Web Browser API]

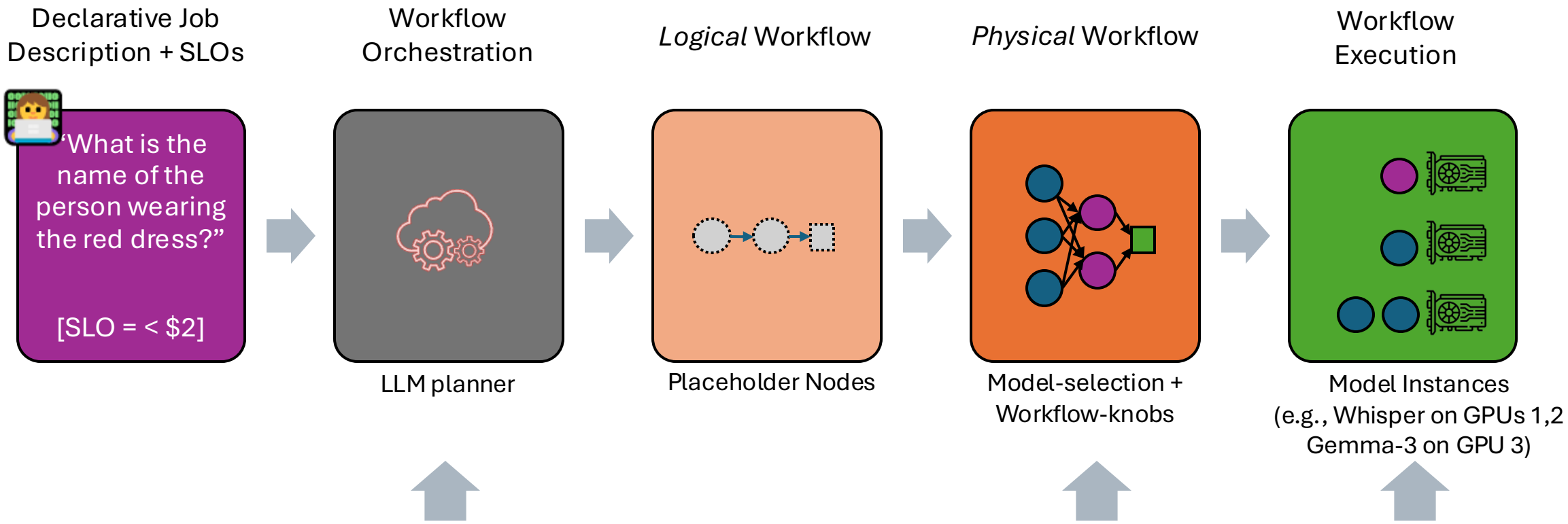
Murakkab



Agents and Tools Library
(LLMs, ML models, tools)
[e.g., DeepSeek-R1, Whisper, Web Browser API]

Runtime System + Infrastructure
(Resource monitoring, allocation, scheduling, auto-scaling)

Murakkab


Declarative Job Description + SLOs
Workflow Orchestration
Logical Workflow
Physical Workflow
Workflow Execution
"What is the name of the person wearing the red dress?"
[SLO = < $2]
LLM planner
Placeholder Nodes
Model-selection + Workflow-knobs
Model Instances (e.g., Whisper on GPUs 1,2 Gemma-3 on GPU 3)
Agents and Tools Library
(LLMs, ML models, tools)
[e.g., DeepSeek-R1, Whisper, Web Browser API]
Runtime System + Infrastructure
(Resource monitoring, allocation, scheduling, auto-scaling)

Murakkab

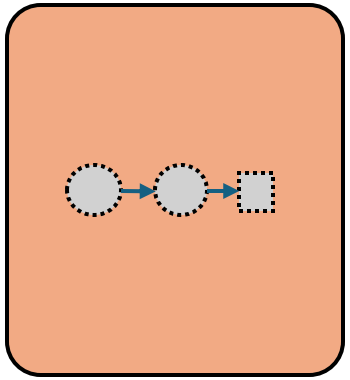
SLOs:
1. Price ($)
2. Quality (% or tier)
3. Latency (s)

Efficiency:
1. Cost ($)
2. Energy (kWh)
3. Resources (e.g., # GPUs)

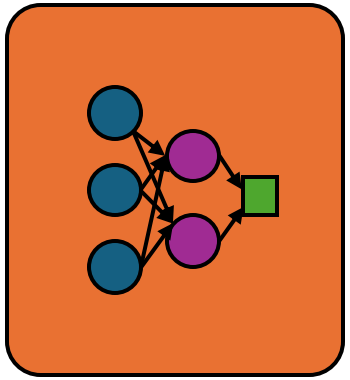
"What is the name of the person wearing the red dress?"

[SLO = < $2]

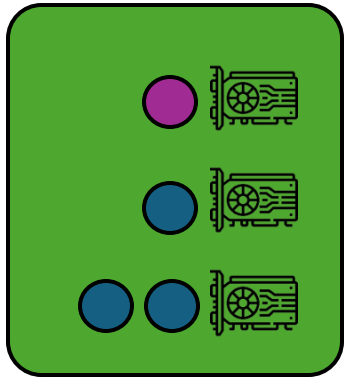
LLM planner



Placeholder Nodes



Model-selection + Workflow-knobs



Model Instances (e.g., Whisper on GPUs 1,2 Gemma-3 on GPU 3)

Agents and Tools Library
(LLMs, ML models, tools)
[e.g., DeepSeek-R1, Whisper, Web Browser API]

Runtime System + Infrastructure
(Resource monitoring, allocation, scheduling, auto-scaling)

Optimizing for efficiency

Efficiency:

Quality:

Optimizing for efficiency

Efficiency:

- Reallocate models and resources with changing workflows/load.

Quality:

Optimizing for efficiency

Efficiency:

- Reallocate models and resources with changing workflows/load.
- Collocation and model sharing:
  - More batching reduces marginal cost of serving new workflows.

Quality:

Optimizing for efficiency

Efficiency:

- Reallocate models and resources with changing workflows/load.
- Collocation and model sharing:
  - More batching reduces marginal cost of serving new workflows.

Quality:

- Open-source datasets/benchmarks as a proxy for new queries.

Optimizing for efficiency

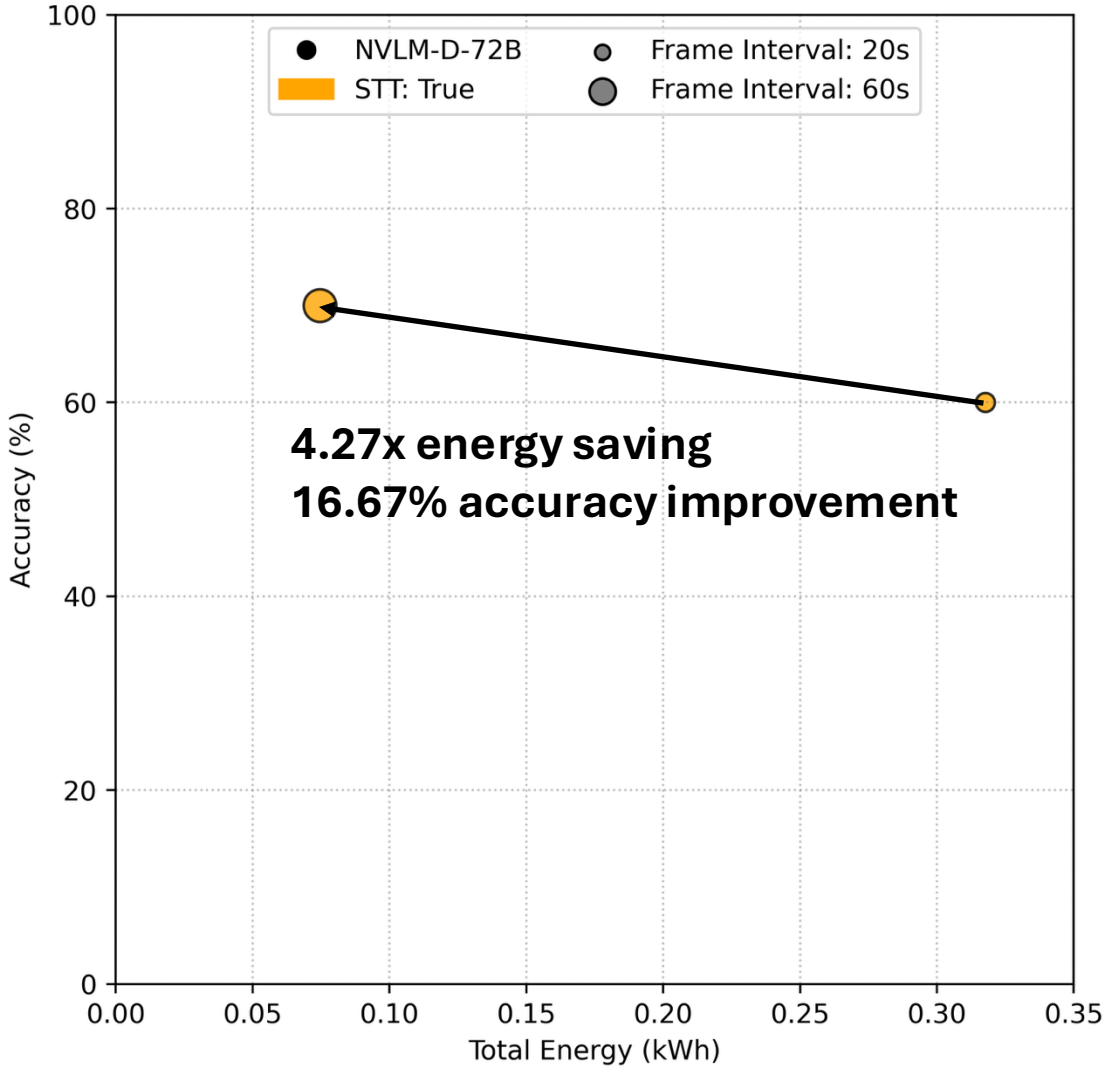
Efficiency:

- Reallocate models and resources with changing workflows/load.
- Collocation and model sharing:
  - More batching reduces marginal cost of serving new workflows.

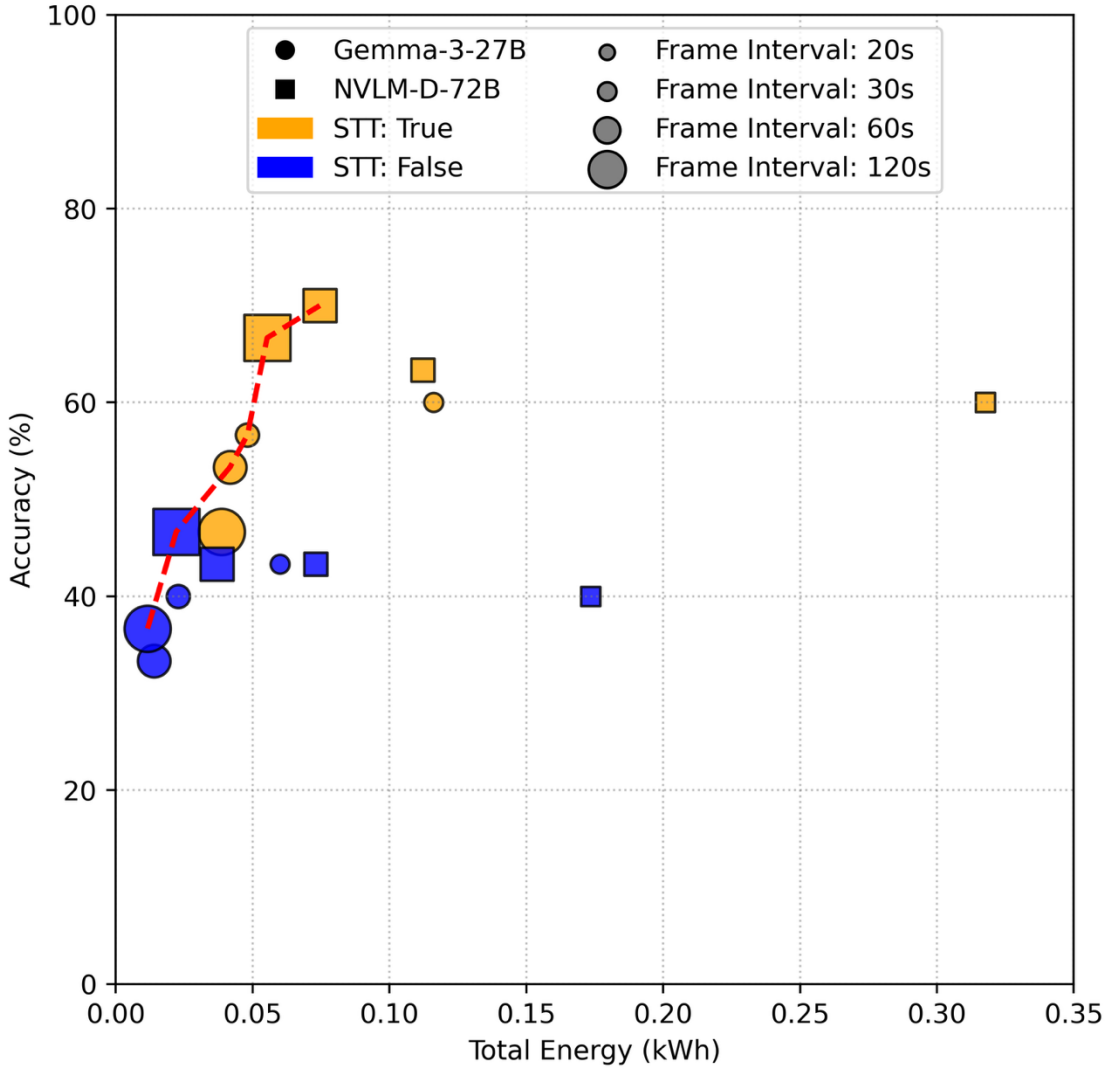
Quality:

- Open-source datasets/benchmarks as a proxy for new queries.
- Periodic feedback from users.

Video Q/A: optimized configuration



Dynamic workflow optimization



Are these systems efficient?

Are these systems efficient?

NO! But they can be...

Are these systems efficient?

NO! But they can be…

1. **Decoupling app. logic from configuration**
2. **Unified compound AI system**

What’s next?

Efficiency:

Quality:

What’s next?

Efficiency:

- Different hardware configurations.
- Multi-workflow joint optimization.

Quality:

What’s next?

Efficiency:

- Different hardware configurations.
- Multi-workflow joint optimization.

Quality:

- Per-task quality vs end-to-end quality.
- Closed source models (e.g., GPT-4o).

What’s next?

Efficiency:

- Different hardware configurations.
- Multi-workflow joint optimization.

Quality:

- Per-task quality vs end-to-end quality.
- Closed source models (e.g., GPT-4o).

End-to-end automatic profiling and optimization framework for multiple workflows on shared infrastructure.

Outline

1. Compound AI workflow: Video Q/A
   a. Possible configurations.
   b. Workflow implementation.
   c. Workflow execution.
2. Our vision.
3. Murakkab.

girfan@mit.edu