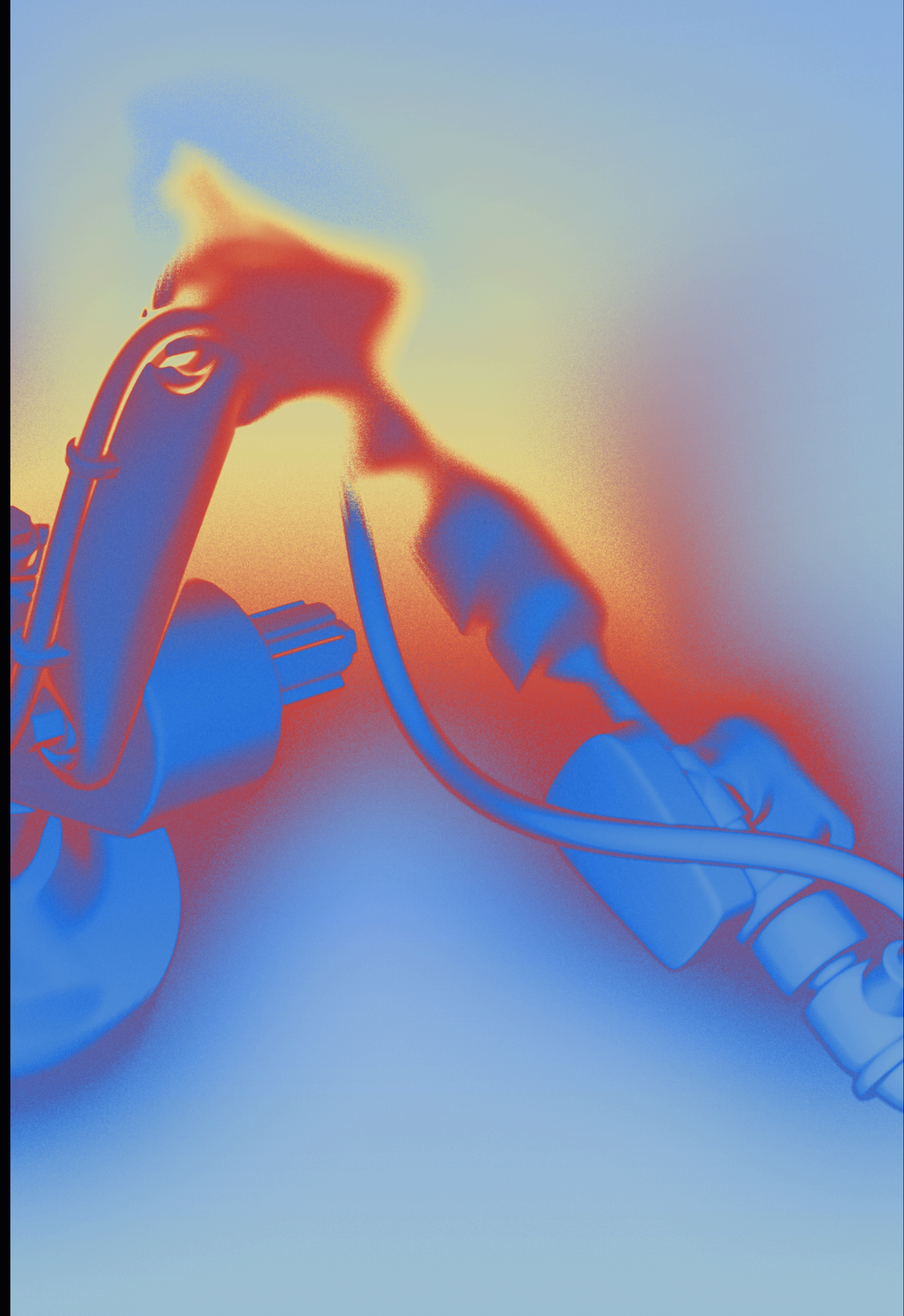
MULTIMODAL DATALOADERS FOR PHYSICALAI

GIJS DE JONG & PETER PRETTENHOFER

PYTORCH CONFERENCE NORTH AMERICA 2026

rerun





Gijs De Jong

ML Eng @ Rerun

Member of humanoid soccer team

WHiRL WIND



Peter Prettenhofer

ML Eng @ Rerun

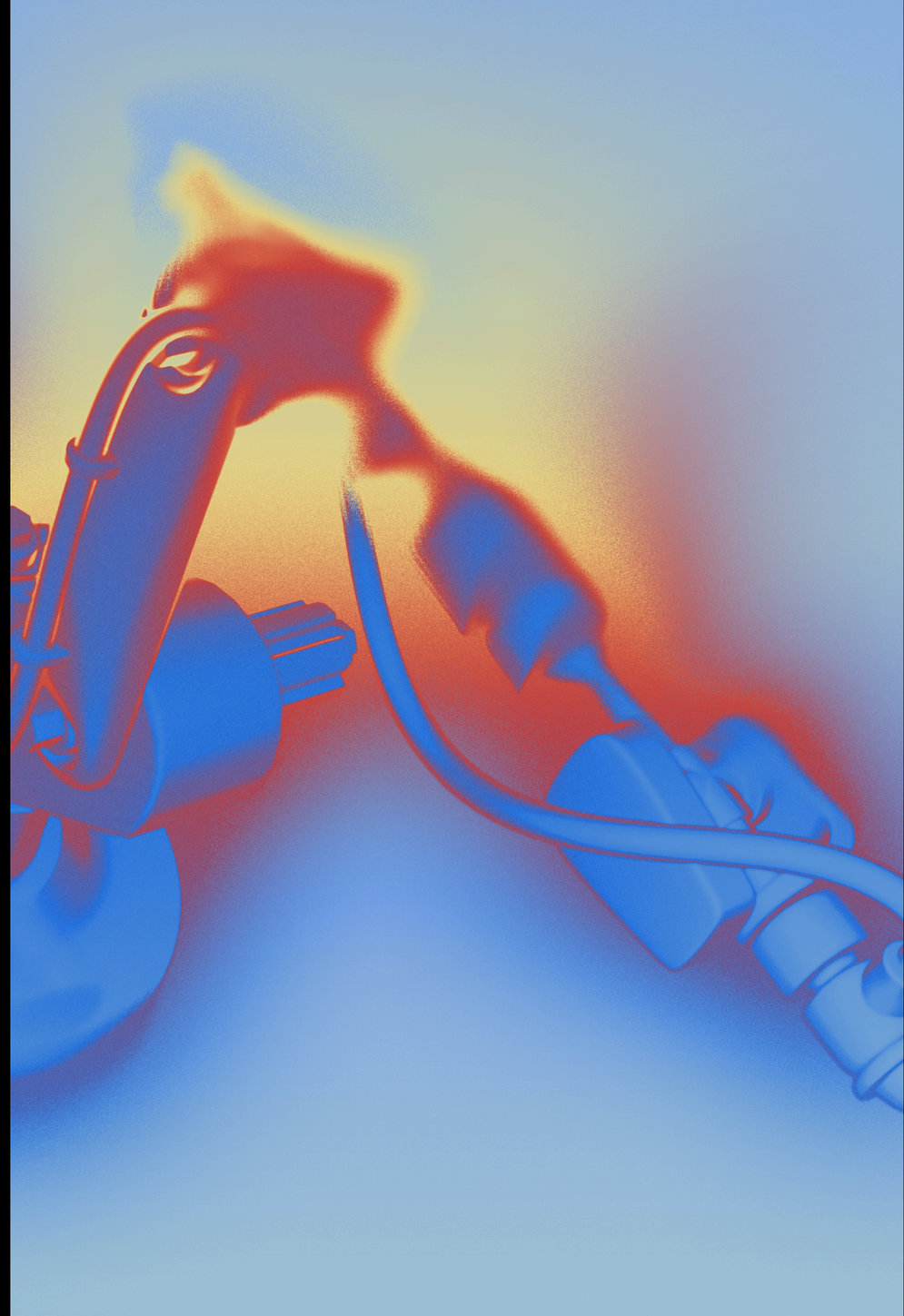
Former, VP Eng @ DataRobot

Core contrib @ scikit-learn

WHAT'S IN FOR YOU

AGENDA

- Motivation
- Data for Robot Learning
- Streaming Dataloader
  for 100+ TB robot datasets



MOTIVATION

VISUO-MOTOR POLICIES

Image observation



Encode

Language instruction

“Pick up the industry object and place it in the yellow bin”

Tokenize



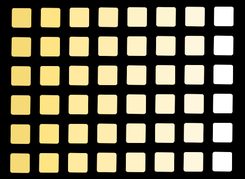
Robot state

- Joint Velocities
- Joint Positions
- Base Position
- EFF Poses

Encode

Model

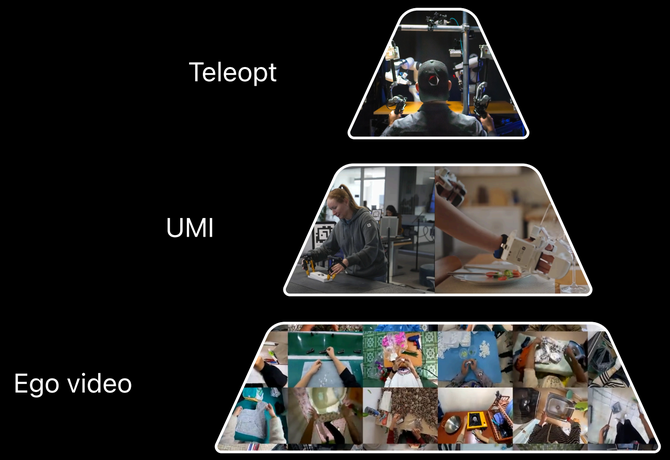
Actions





Drivers of Progress

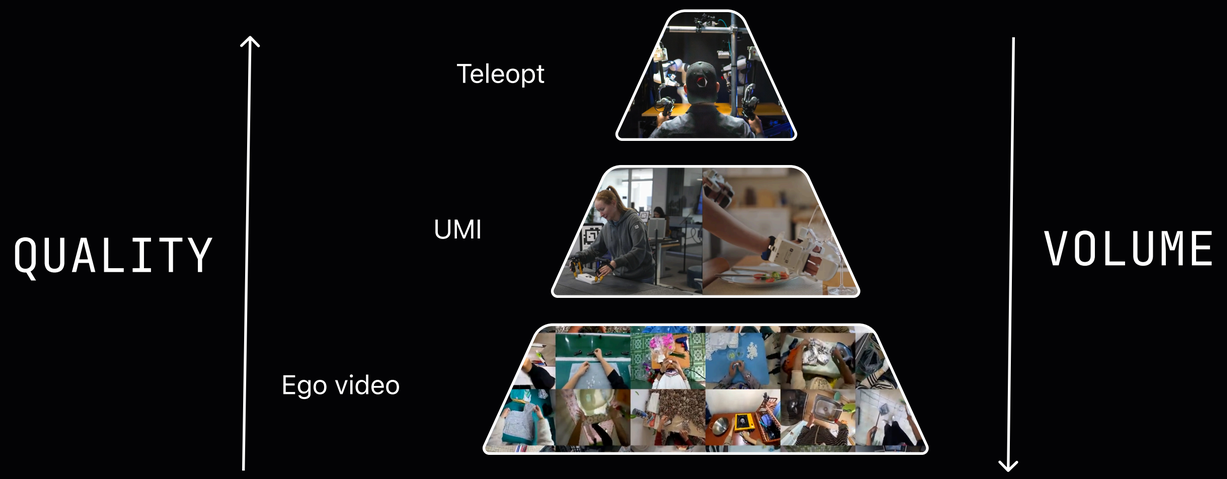
Data



Generative models



THE DATA PYRAMID

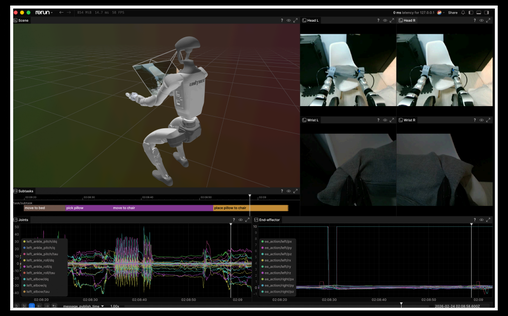


FROM RECORDINGS TO MODELS

ROBOT TRAINING LOOP

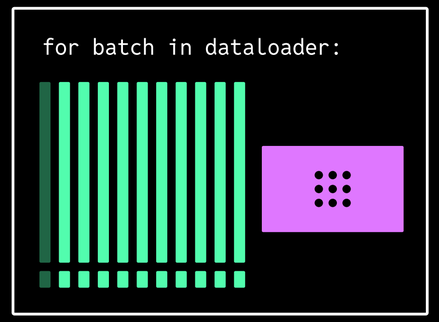
DATA COLLECTION → INGESTION & NORMALIZATION → ANNOTATION & POST-PROCESSING → CURATION → TRAINING → EVALUATION & DEPLOYMENT

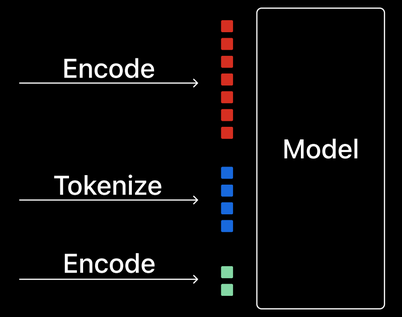








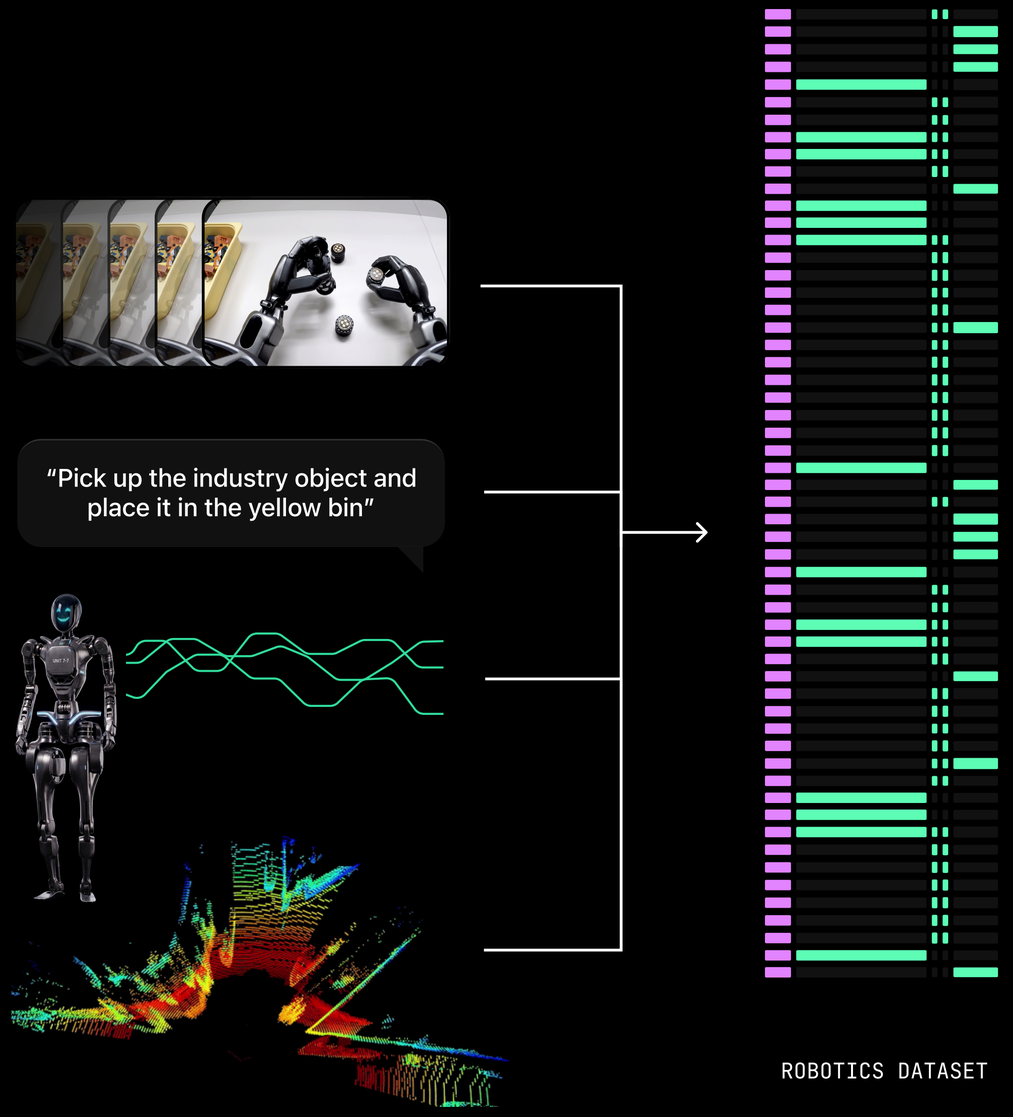




IT’S NOT A TABLE

ROBOTICS DATA

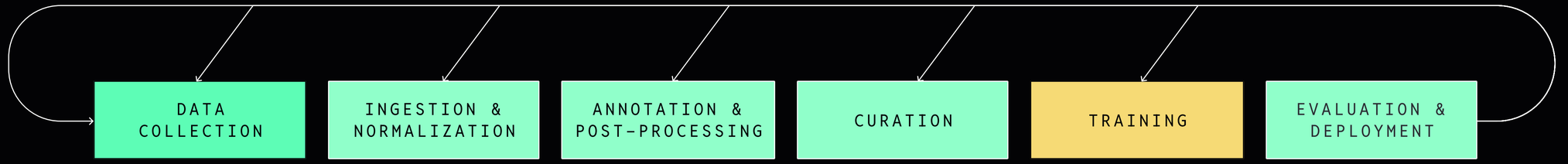
- Multimodal
- Multirate
- Episodic
- High volume / skewed
- Flexible schema



CHALLENGING TO STORE, QUERY, AND PROCESS
IN CLASSICAL TABLE-ORIENTED DATA SYSTEMS

A SOLVED PROBLEM?

OPEN-SOURCE DATA FORMATS



MCAP

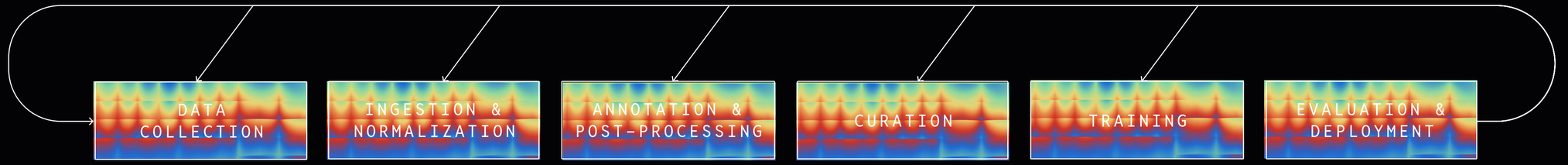
- Write-optimized (record & replay tasks)
- Chunks of heterogenous data, row writes
- Also: default file container for ROS bags
- Not optimized for columnar analytics queries

LEROBOT

- Read-optimized (ML training tasks)
- Time-aligned samples, columnar
- Popular format in physical AI community
- Opinionated, mainly for training tasks

A SOLVED PROBLEM?

OPEN-SOURCE DATA FORMATS



RRD

- Write-optimized (record & replay tasks)
- Columnar chunks
- Footer manifest
- Supports columnar analytics queries

RERUN_SDK

- Open source library and tools for logging, storing, querying, visualizing, and training on multi-rate, multimodal data.
- Implemented in Rust w/ Python bindings

DATA ACCESS PATTERN

TIME-ALIGNED



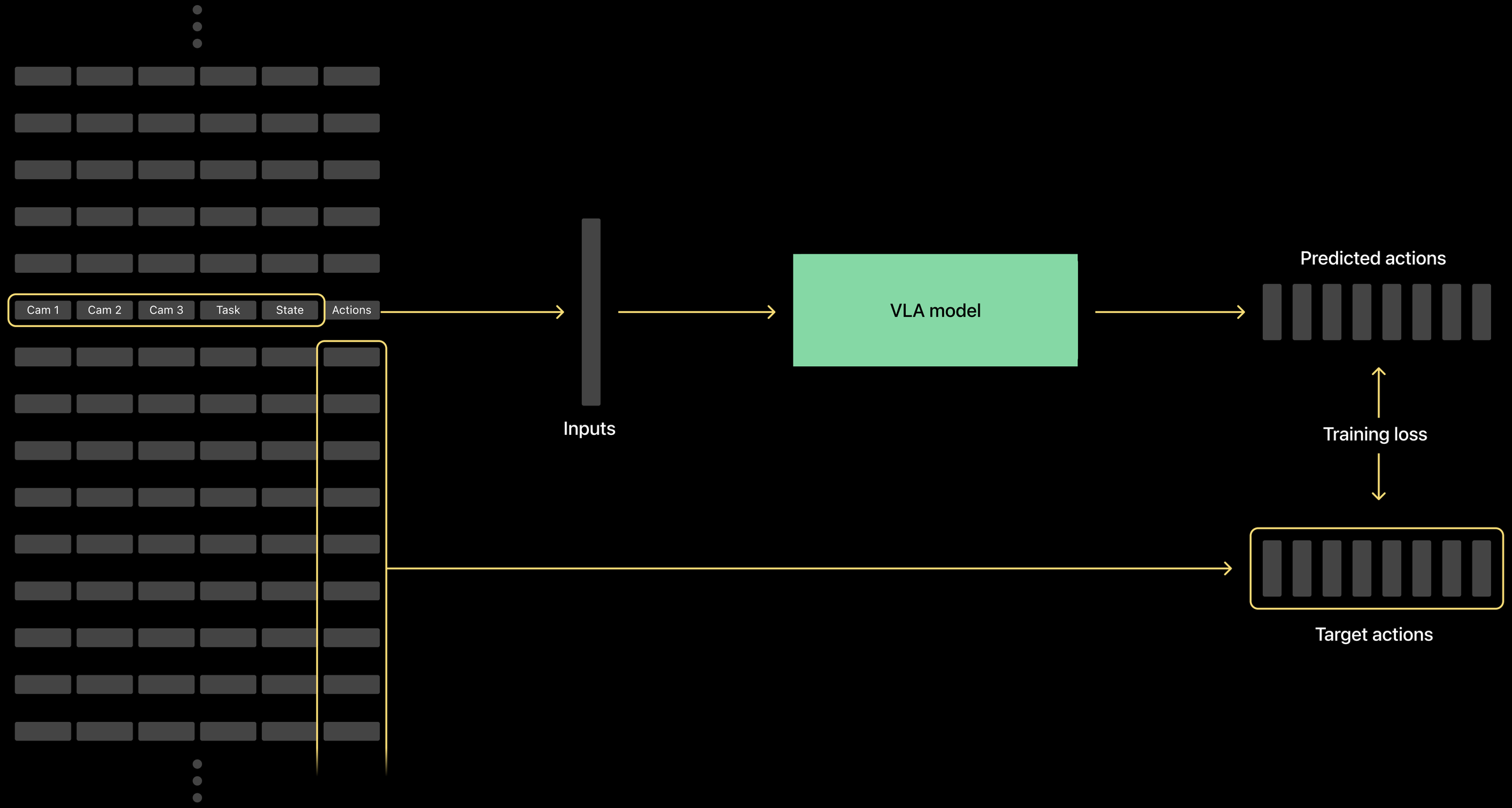
CONTEXT WINDOW



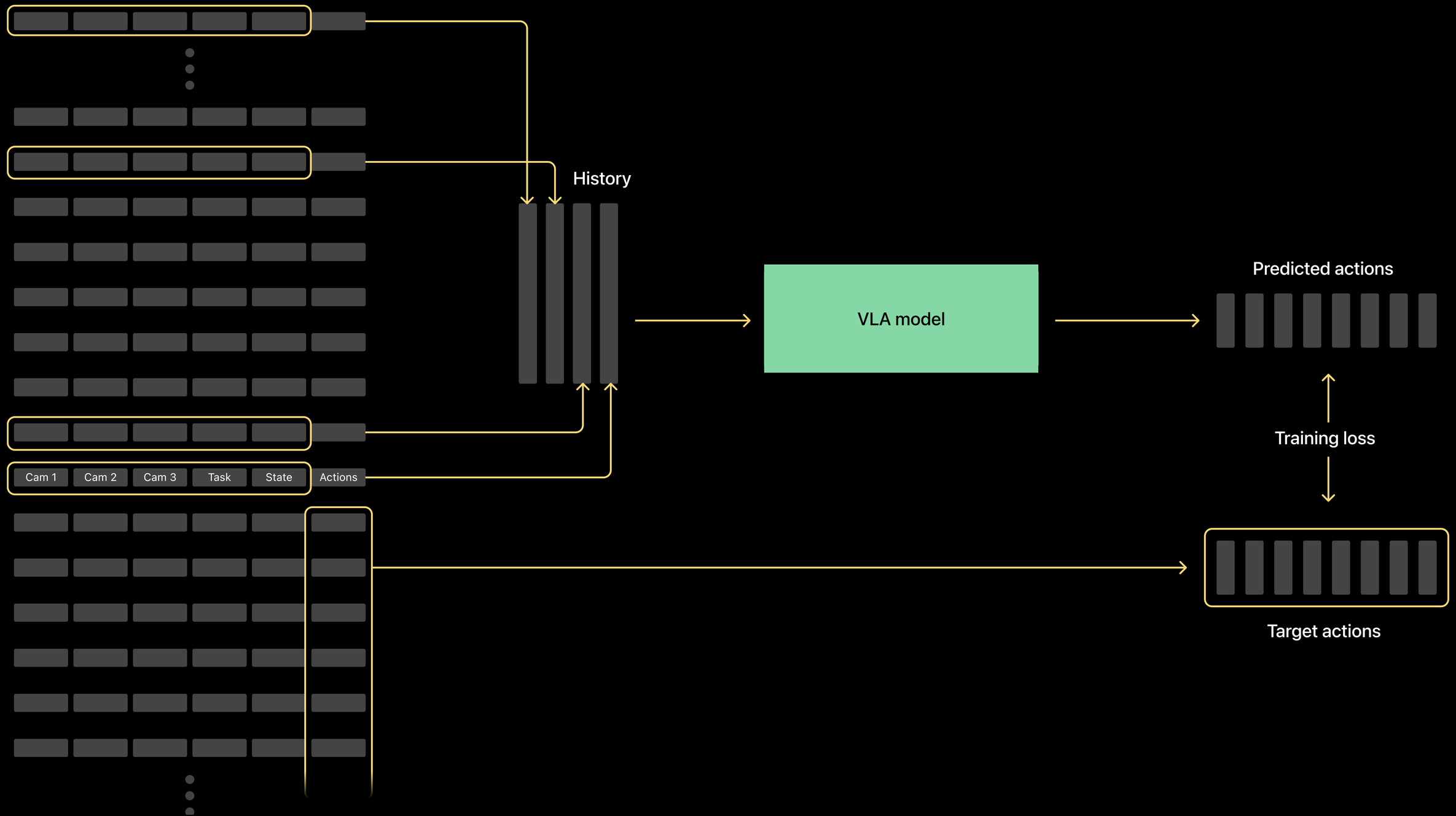
MULTI-MODAL



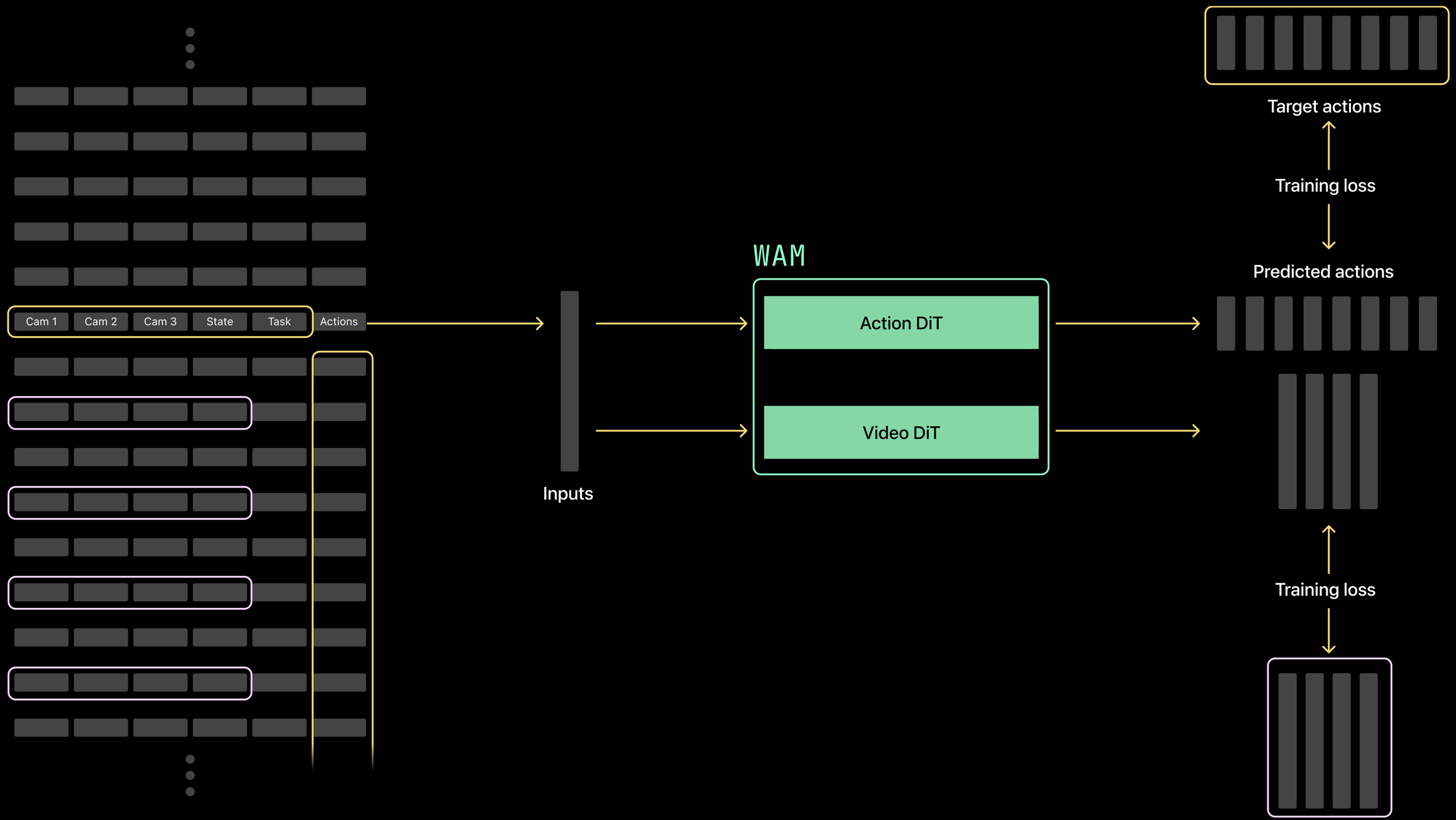
Basic sample construction for a VLA model



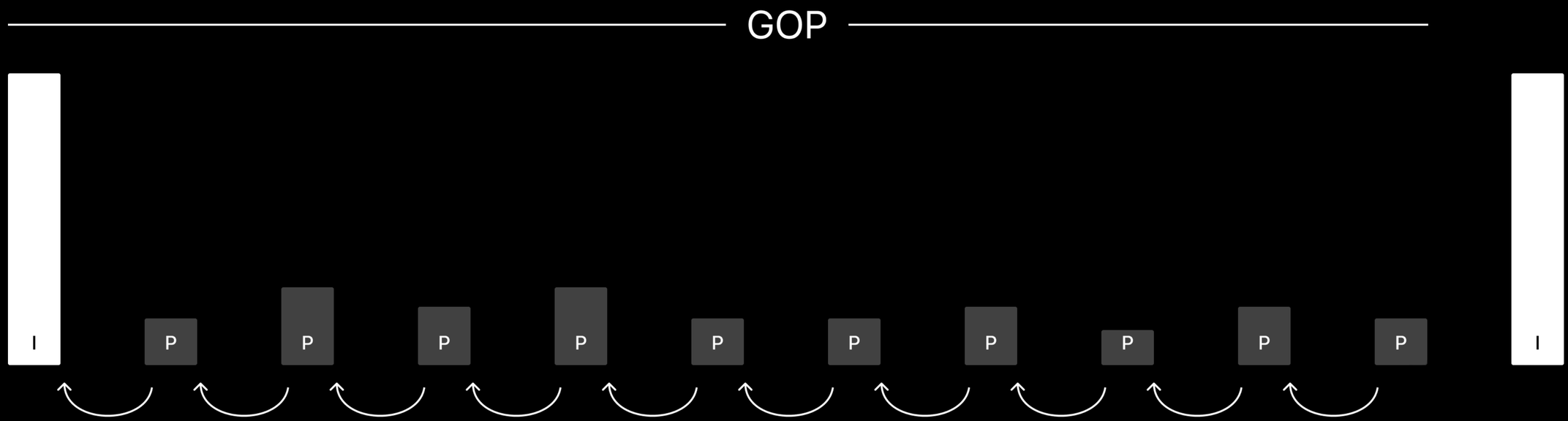
Sample construction with history



Sample construction for a World-Action-Model



Video compression and GOPs



Video is compressed in Groups of Pictures (GOP) containing keyframes and delta-frames

Video often 90% + of dataset size

Need to look back in time to decode a given video packet to an image



STREAMING DATALOADER

Challenges & design decisions for 100+ TB robot datasets

CHALLENGES

DATA VOLUME

Data residing on object stores favors few contiguous reads

AMPLIFICATION

Data formats and GOP contribute to redundant data transfer on scattered reads

DECODE HEAVY

Efficient decoding of target frames, avoid redundant frame decoding

PROPOSED SOLUTION

BLOCK SHUFFLE

Trades shuffle-quality for contiguous reads

Outline:

- Group samples into **contiguous blocks**
- **Permute blocks** and assign them to workers
- Workers fetch blocks, decode samples and put them into a node-local **shuffle buffer**

THE CORE BUILDING BLOCK

BLOCK SHUFFLE

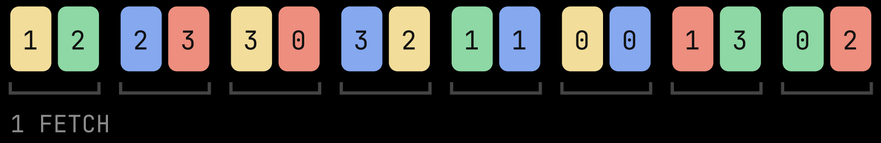
DATASET

4 segments, natural order

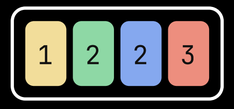


SAMPLE SHUFFLE

every fetch scatters across segments



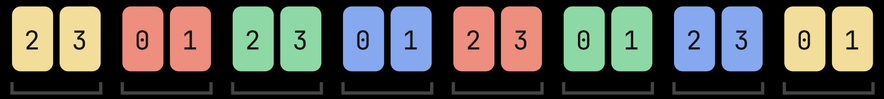
BATCH



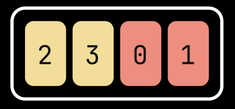
✓ 4 segments
well mixed, but slow fetches

BLOCK SHUFFLE

every fetch reads one contiguous span

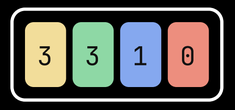


BATCH · NO BUFFER



✕ 2 segments
consecutive frames, correlated

BATCH · SHUFFLE BUFFER



✓ 4 segments
well mixed again, fast fetches

WHY IT WORKS

BLOCK SHUFFLE

Block shuffle preserves **fetch** and **decoder locality**.

Large shuffle buffer decorrelates nearby frames and mixes segment-level conditions.

Decorrelated samples reduce gradient variance and thus **increase effective batch size**.

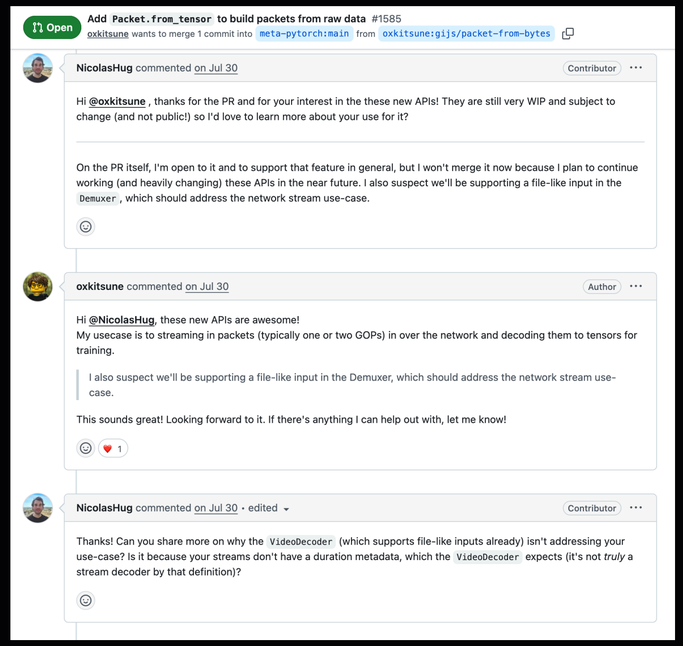
VIDEO DECODING DETAILS

BLOCK SHUFFLE

If you fetch encoded packets you either feed them to

- A **packet-level decoder API** (pyAV), or
- **Remux** them into a temporary MP4 for an API that requires an MP4 index (e.g. torchcodec)

We are excited about ongoing work in torchcodec to allow the above more easily.



PR#1585

IN ACTION

BLOCK SHUFFLE

RerunIterableDataset

Block shuffle w/ small blocks
Per-process shuffle buffer

Pros

- High shuffle quality

Cons

- High memory req

StreamingLeRobotDataset (<=0.6)

Block shuffle w/ large shards
Per-process shuffle buffer

Pros

Cons

- High memory req

StreamingLeRobotDataset (>0.6)

Resident episode buffer
Random access in resident episodes

Pros

- Low memory req
- High throughput

Cons

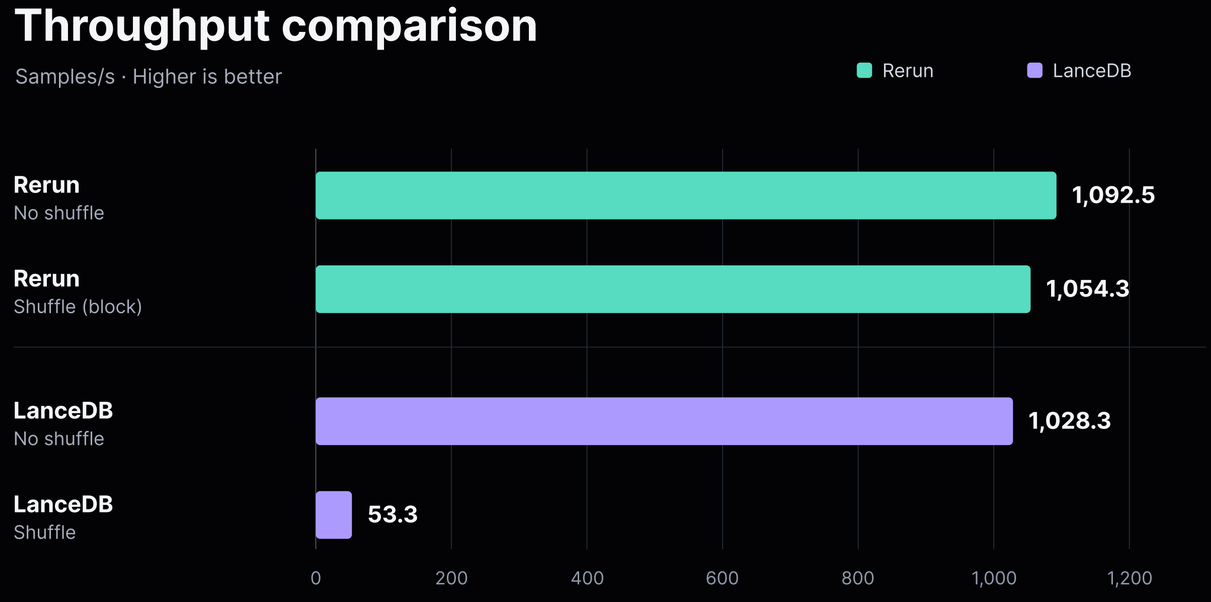
- Low episode-level diversity (see PR#4874)

BENCHMARKS

BLOCK SHUFFLE

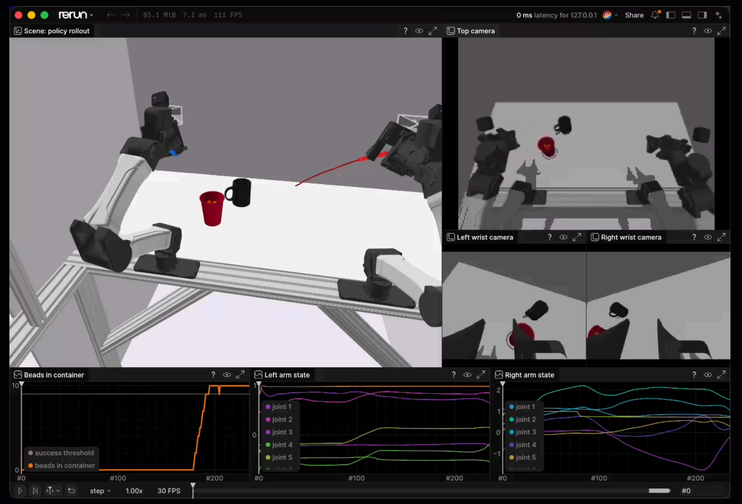
stable-worldmodels

- Benchmark on simulated 2d push-t task
- Compared against LanceDB
- Data in AWS S3



BENCHMARKS

BLOCK SHUFFLE



abc-sim

- Task: pour beads
- ABC-DiT finetune on 2× H100
- Compared against abcdl (local)

	ABCDL (LOCAL)	RERUN (S3)
Training throughput	2.70 it/s	**2.61 it/s** (97%)
GPU time spent waiting on data	—	**0 ms** median per step (<0.2%)
Validation error @ 25k	0.0544	**0.0481** (12% lower)
Success Rate (200 rollouts)	131 / 200	**126 / 200** (on par)

SINCERELY

THANK YOU

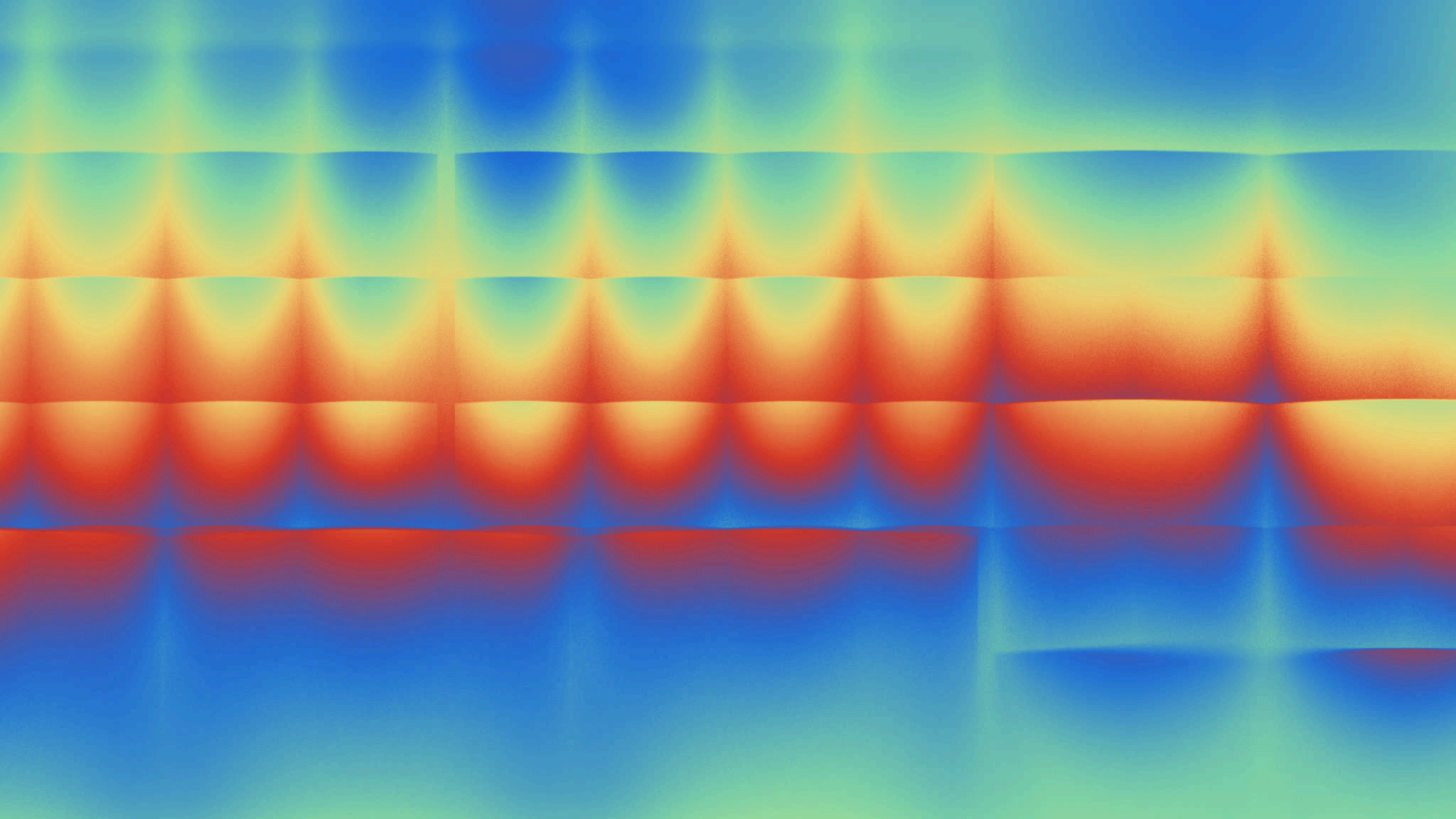
Stay in touch!

Join the Discord

Contributions welcome



Feedback



rerun

Building the data layer for physical AI:

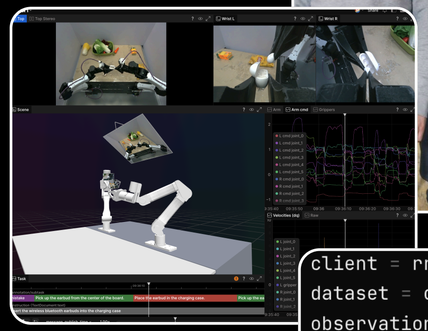
→ SDK, Viewer & Infra to log, query, view & train

Widely used by some of the best teams in the space.

- Founded in Stockholm
- International remote team with regular offsites
- Tech stack built on Rust, wgpu, egui, ...
- Fully open-source viewer & SDK, commercial Hub
- Strong OSS backgrounds & contributors







```
client = rr.catalog.CatalogClient(CATALOG_URL)
dataset = client.get_dataset(name="sample_dataset")
observations = dataset.filter_contents(["/observation/**"]).reader(
    index="real_time"
)
```













WE ❤️ OPEN SOURCE

 `pip install rerun-sdk`   `cargo add rerun`  github.com/rerun-io/rerun

Open source Free forever

→ Ephemeral
→ Local storage
→ Local access
→ Built for workstation-scale

Rerun Hub Commercial

→ Identical APIs
→ Persistent and managed
→ Indexed across your object stores
→ Secure team access
→ Built for petabyte scale

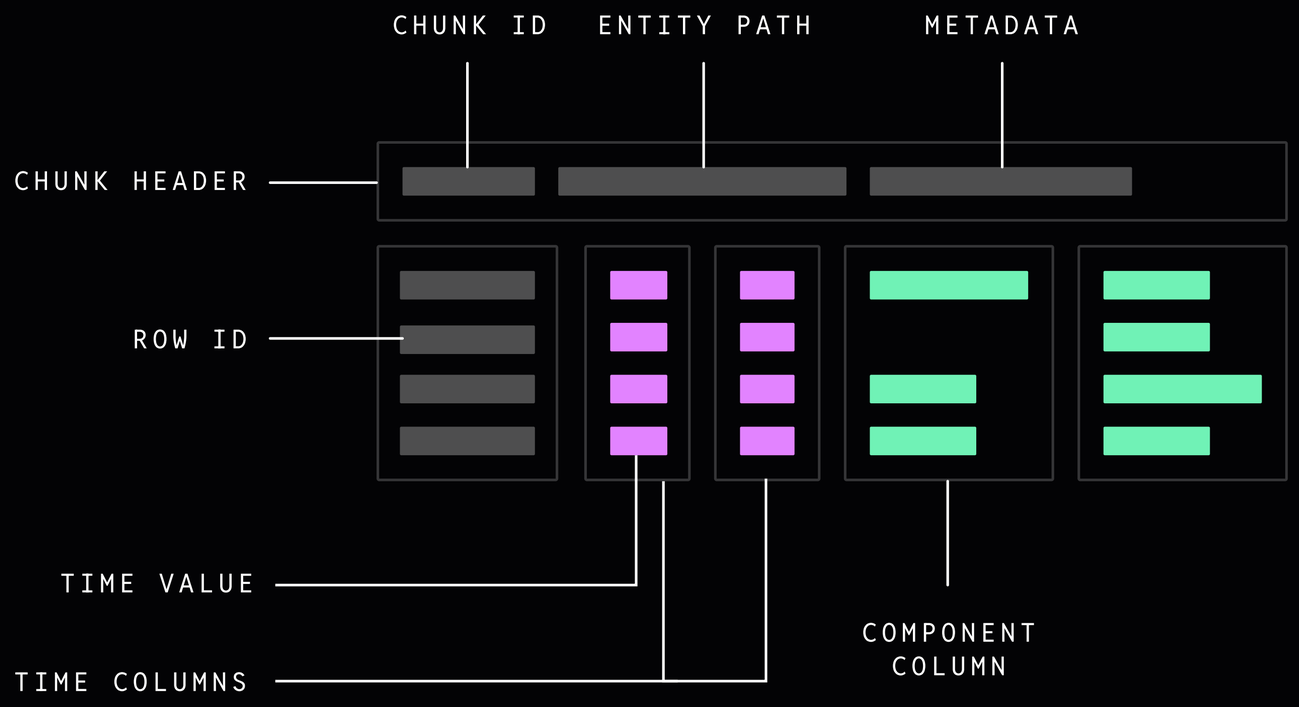
THE CORE BUILDING BLOCK

COLUMNAR CHUNKS

APACHE ARROW

Per-entity data lives in Arrow RecordBatch:

- Arrow metadata (schema description & versioning)
- One contiguous table
  - Index / time columns
  - Component columns (e.g. points, colors, labels, …)



OPEN STANDARDS

WHY ARROW?

→ Open-source industry-standard

→ Used by Parquet, Polars, (Pandas), Spark...

→ Zero-copy IPC between client libraries

→ Built for query performance



Apache Datafusion builds query engine on Arrow

→ SQL & Dataframe queries

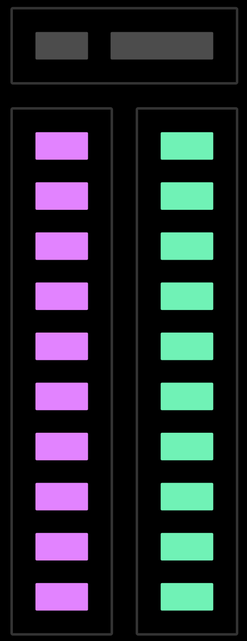
→ Extensible with custom table providers



HETEROGENEOUS CHUNK STORAGE

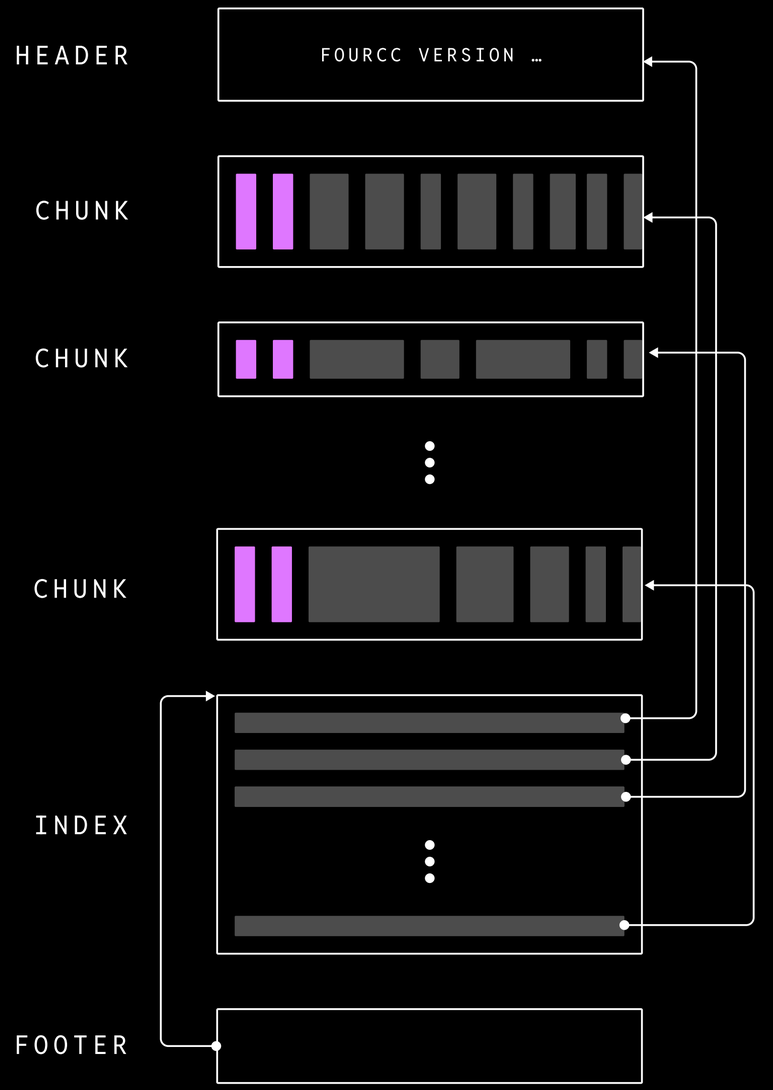
BATCHING & INDEXING

CHUNK WITH SMALL HIGH-FREQUENCY COMPONENT



CHUNK WITH SINGLE LARGE COMPONENT





RECAP

RRD

RRD is the core data model that powers Rerun

→ Built on indexed chunks of columnar Arrow record batches

→ Efficient random-access for view, query and training tasks

→ Append-friendly & modular augmentation via layers

The open-source Rerun SDK provides APIs and tools for all stages of the training loop