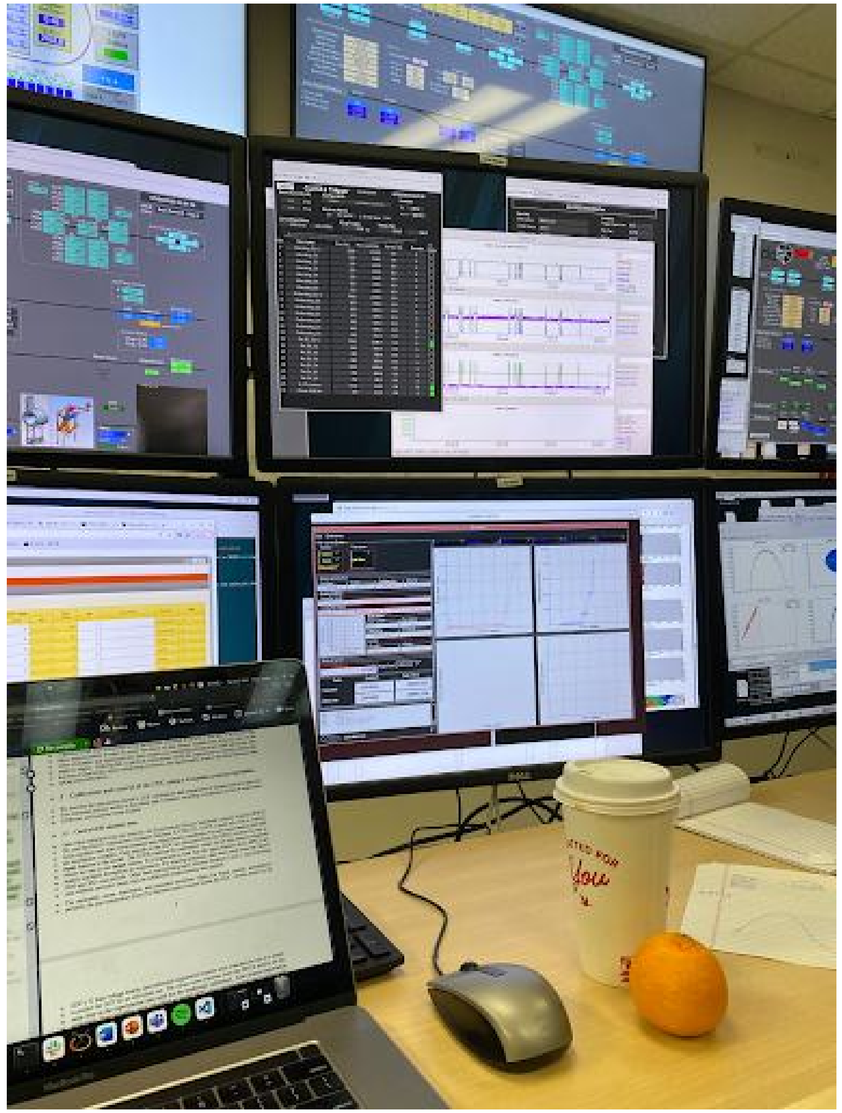
Jefferson Lab

HYDRA

Computer Vision for Data Quality Monitoring

Torri Jeske | roark@jlab.org



EPSCI

Developed and maintained by the Experimental Physics Software and Computing Infrastructure group

Outline

Brief History

Hydra was initially designed for online data quality monitoring incorporating computer vision

Use Cases

We’ve since expanded to visualization and “offline” data quality monitoring

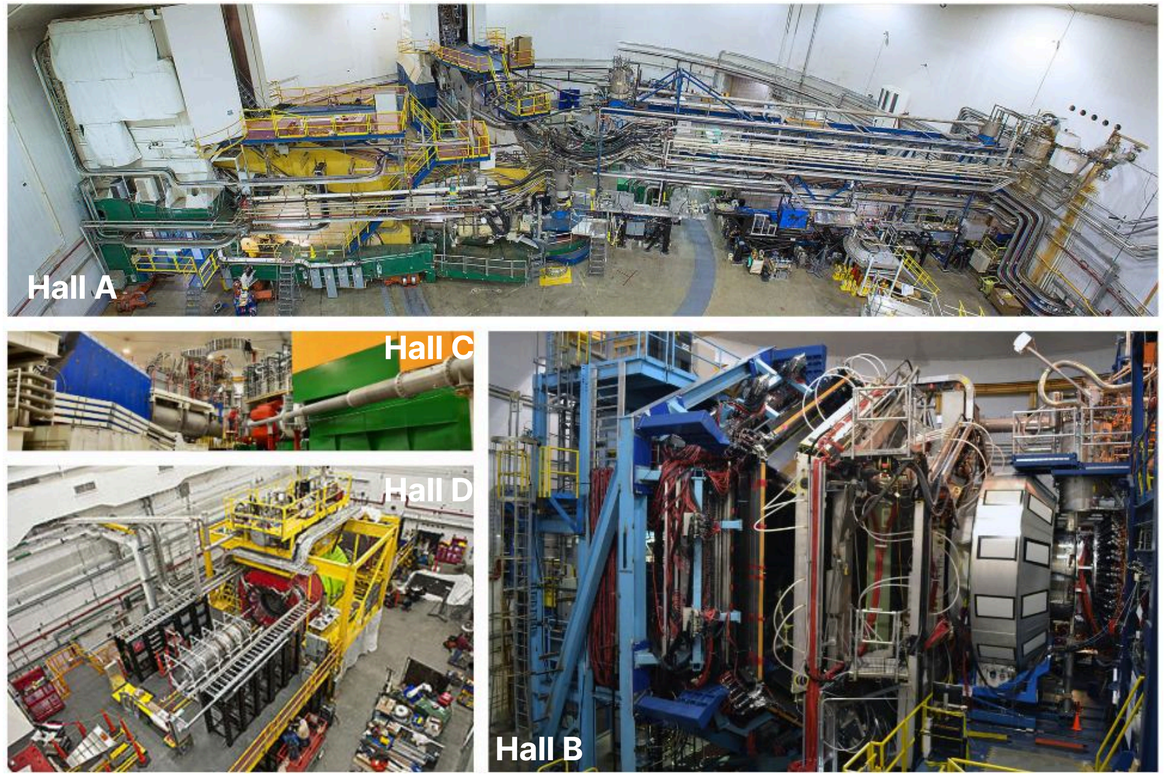
Hydra for ePIC

Hydra can reference and display image-based data for ePIC, used for online DQM in the future

Jefferson Lab

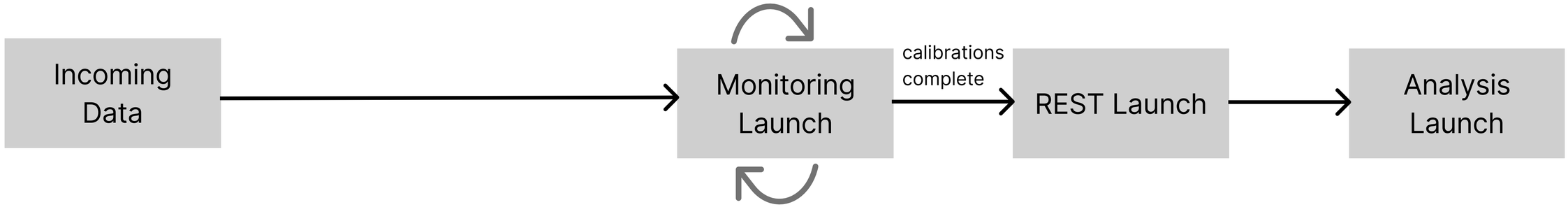
Located in Newport News, Virginia

Home to the Continuous Electron Beam Accelerator Facility (CEBAF) and 4 experimental halls



Monitoring: Data Acquisition to Analysis

A coordinated but *complicated* effort among many people



Minutes/Hours | **Daily** | **Weeks** | **Months** | **Months/Years**

shift crew
detector experts

online monitoring coordinator

detector experts
offline monitoring coordinator
analysis coordinator

offline monitoring coordinator

analysis coordinator

Thomas Britton was responsible for producing a daily brief

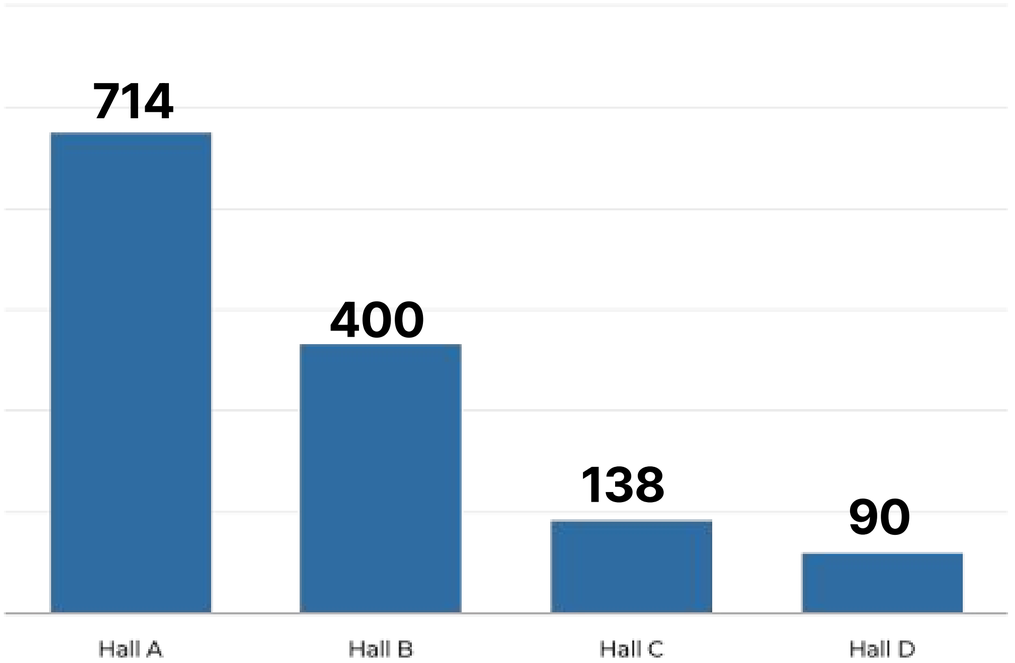
Human-based monitoring does not scale

Fatigue
Continuous monitoring is mentally exhausting.

Inconsistency
Human judgement can vary significantly between individuals.

Slow response time
Humans can only process so much information at a given time.

Scalability
Human-based systems do not scale efficiently as data volumes increase.



Approximate number of individual histograms per experiment per run, monitored by the shift crew

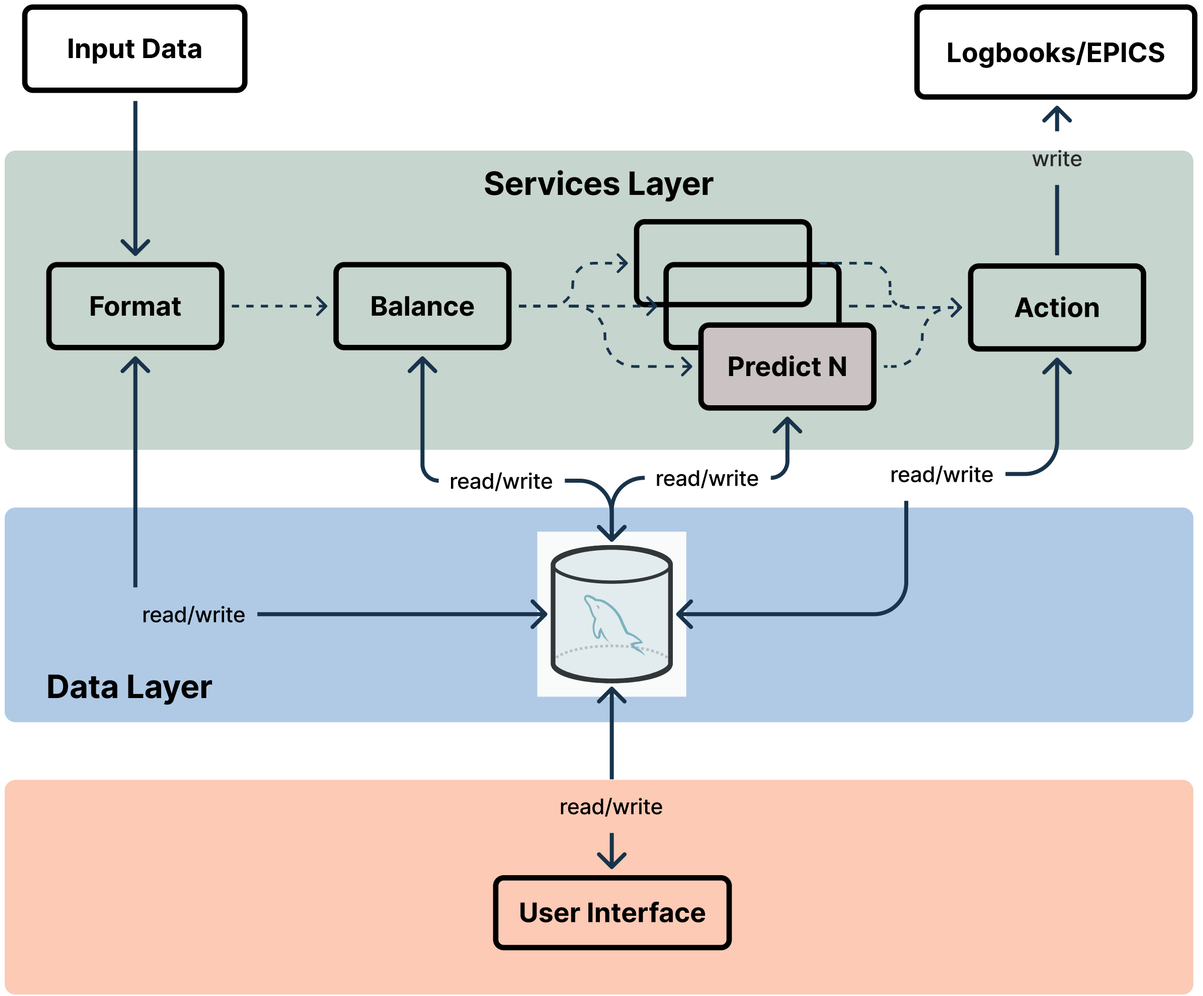
HYDRA

Framework Design

Services Layer (Python/Tensorflow): Convolutional networks (InceptionV3, VGG16, build your own), Siamese networks, Clustering Algorithms, Similarity Algorithms

Data Layer (MySQL): Comprehensive MySQL database, optional Run Conditions Database integration

User Interface (React/Flask): Web application for visualizing data, model classifications, labeling, performance metrics



Summary Statistics

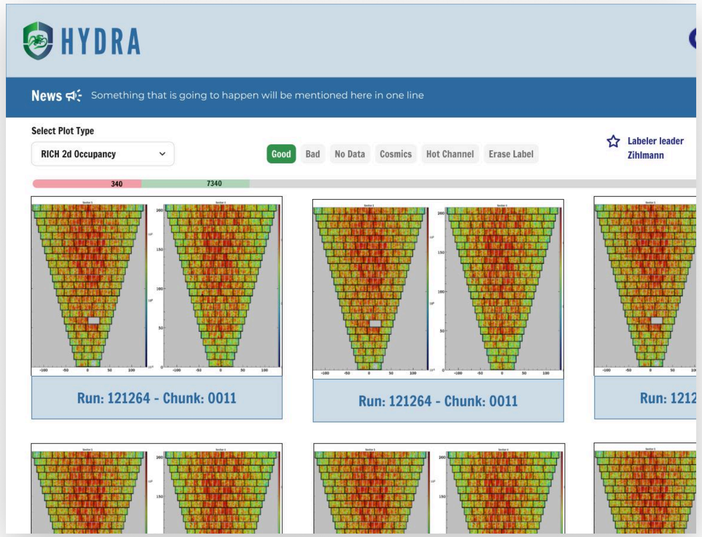
	Hall D	Hall B	Hall A	Hall C	FAIR/GSI	Total
Year deployed	2019	2022	2024	2024	2026	
Labeled Images / Total Images	637,885 / 12,243,320	352,804 / 747,554	5440 / 83,376	7800 / 465,853	2002 / 2002	1,005,949 / 13,542,105
Active Models	19	61	-	6	-	86
Batch / frequency	16 / min	81 / 3-5min	42 / run	92 / run	-	-
Runs with 1 bad image	1,822	1,081	-	33	-	-
Total classified	2,834,043*	2,324,870	-	2475	-	5,161,388

*since record keeping began

Use Cases

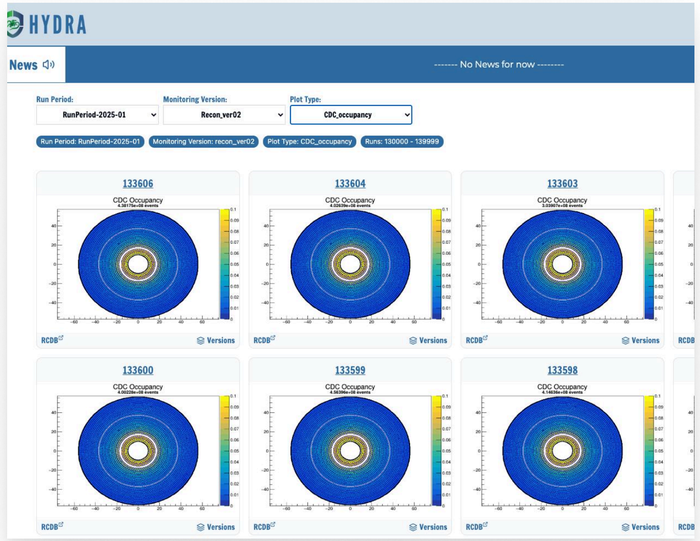
Online Monitoring

Use neural networks to classify images in near real time.



Validation

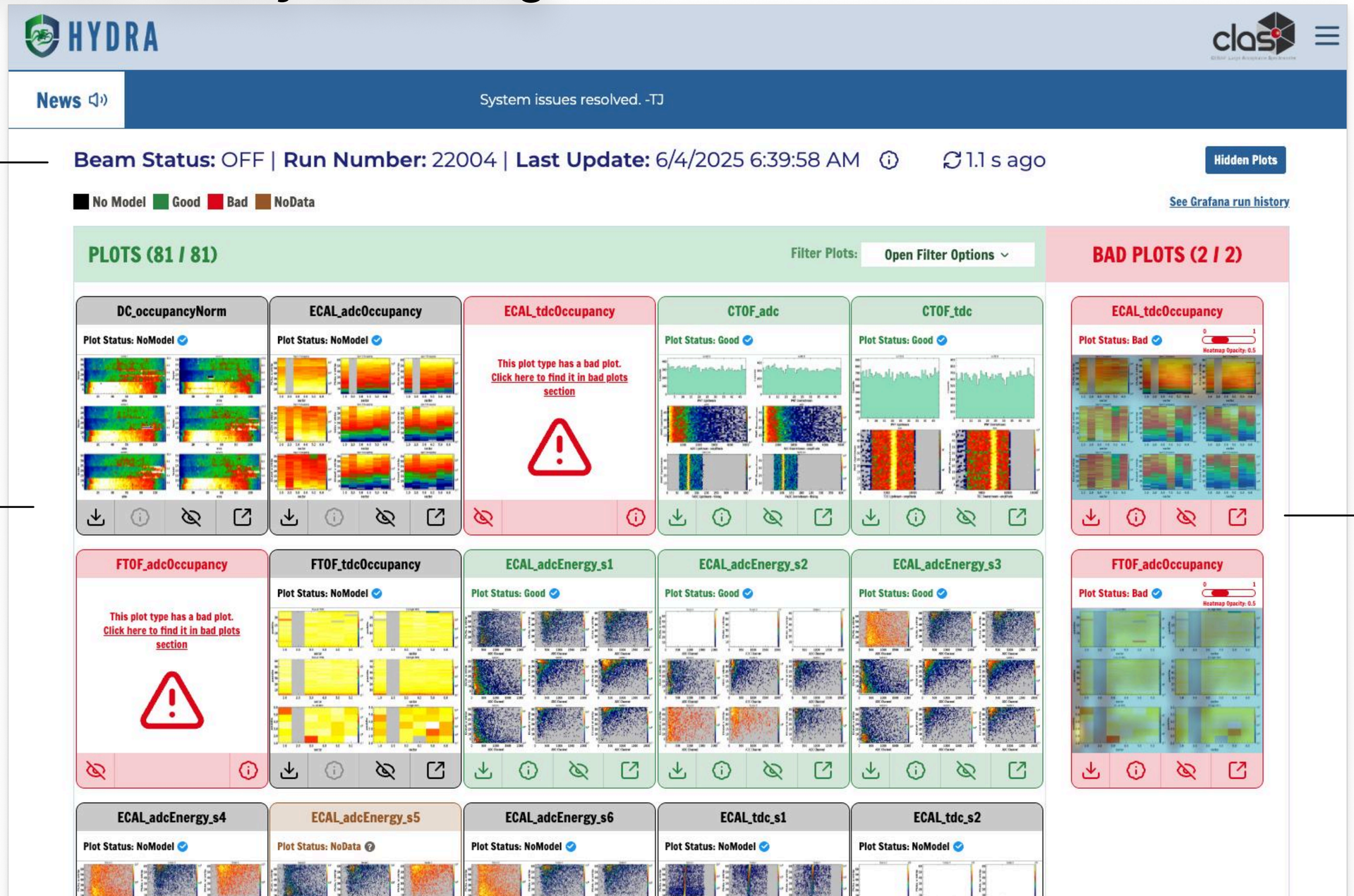
Validate monitoring plots from simulation, calibration changes, etc.



Visualization

View, sort, and filter images.

Online Data Quality Monitoring



status indicators

Main image gallery

reserved for Bad plots

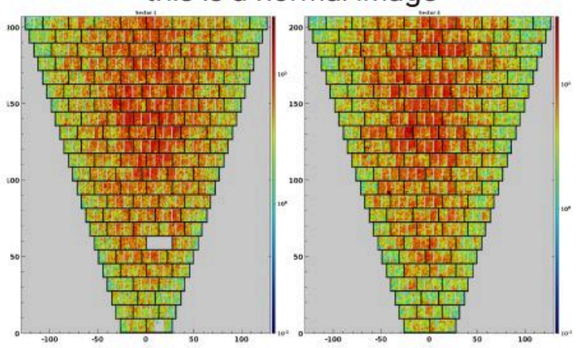
Why is the image bad?

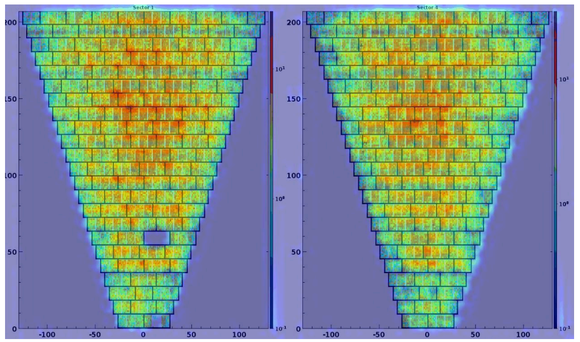
“What about the image does *the model* think is bad?”

Bright spots indicate important regions of the image for the given classification.

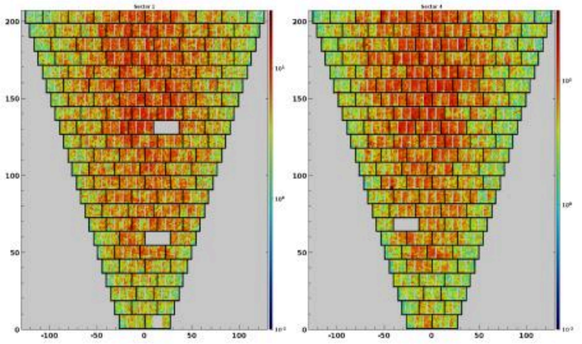
These are very sensitive to how well the model is trained.

this is a normal image





this is a bad image



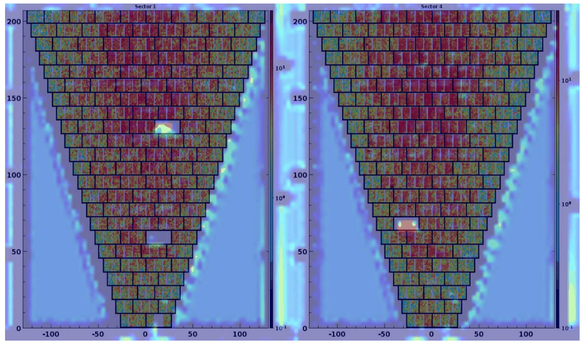
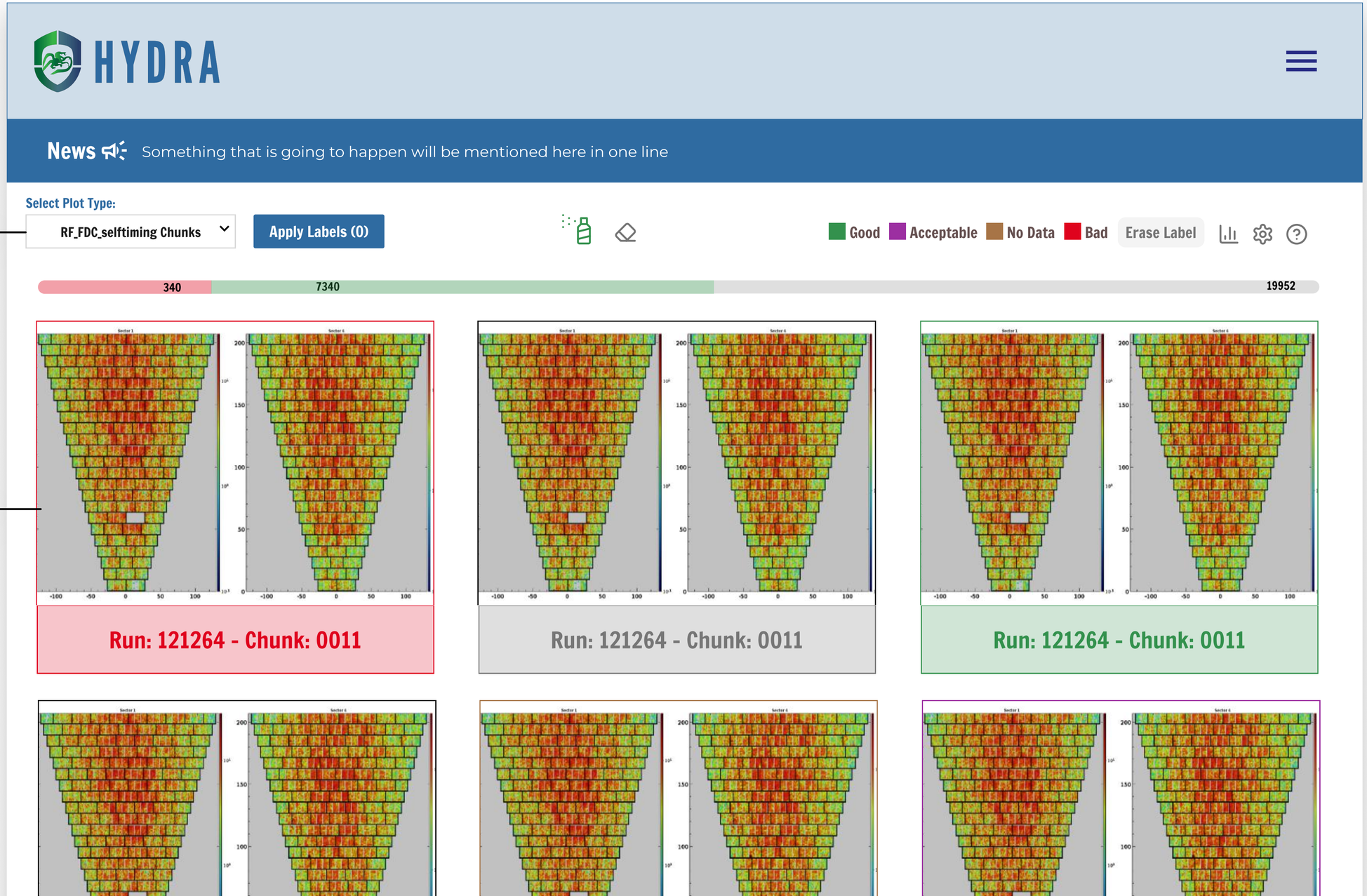


Image Labeler



Plot Type Selector

Images to label

Front-end

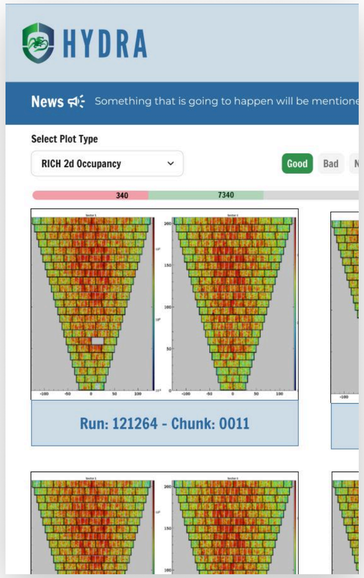
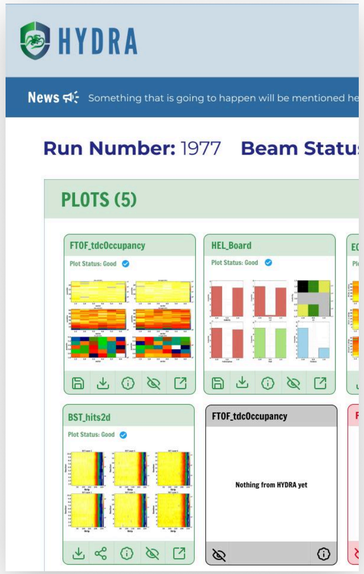


Image Labeler

Efficiently label thousands of images used for training a model.



Run

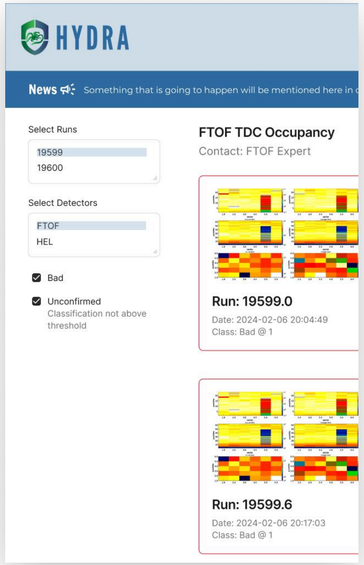
See predictions in near real-time. This page continuously updates with new images during an experiment.

Grafana

Displays all predictions over time. Trend analysis on predictions can indicate when it is time to retrain a model.

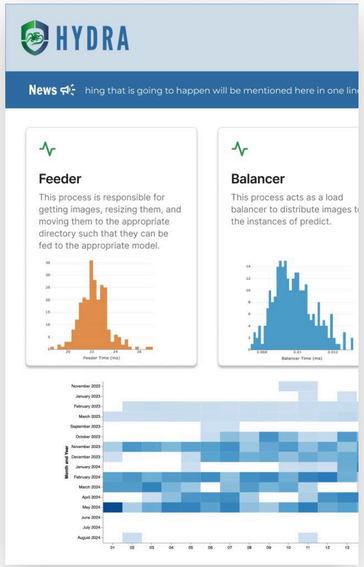
Library

Contains information useful to evaluate a given model's training and performance.



Log

Displays problematic and potentially problematic images from a trailing 24 hour window.

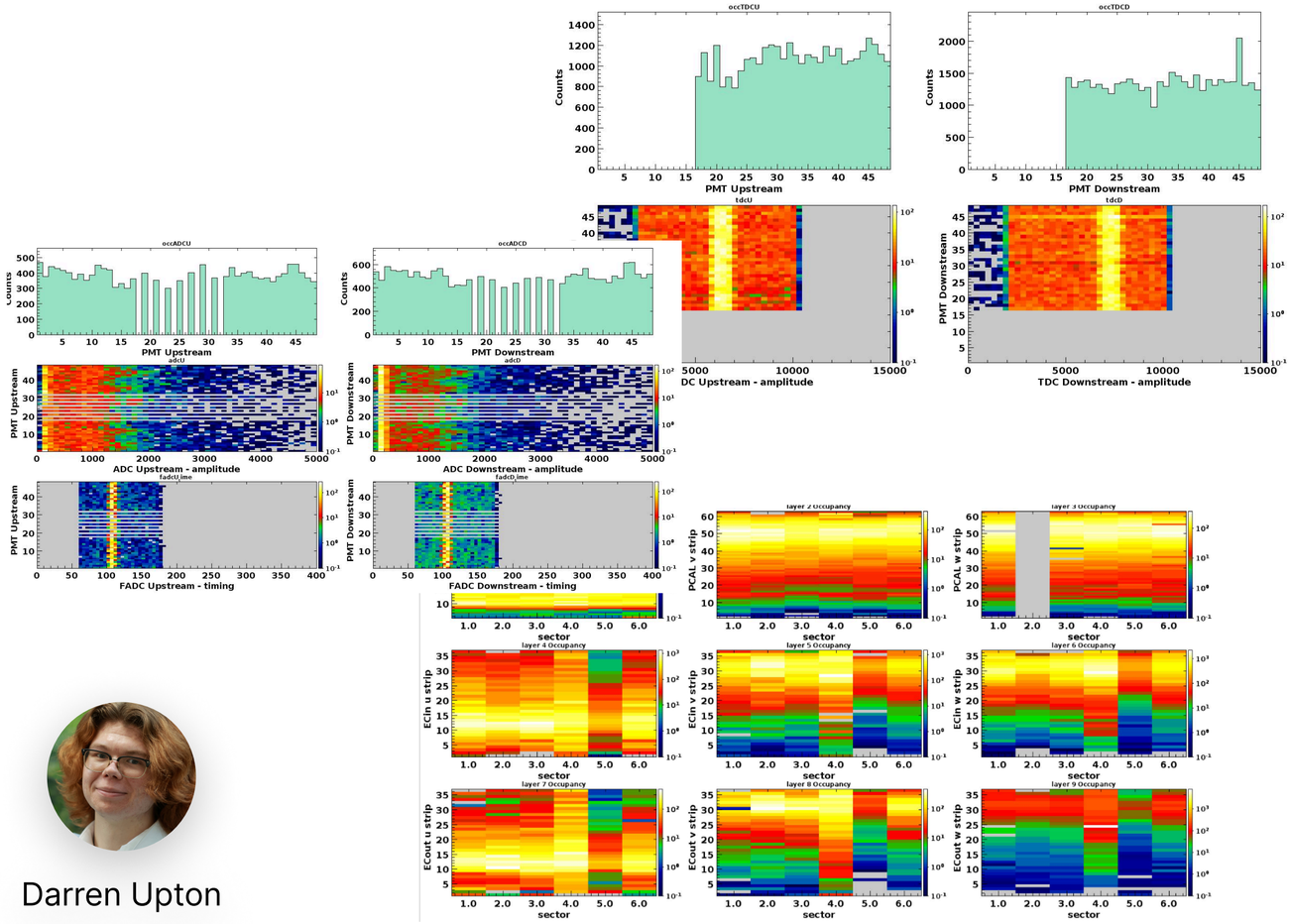


Status

Primarily used by administrators to monitor system performance.

What if we don’t have any “Bad” images?

Our graduate students have developed to software to emulate various failure modes in real data by modifying databases and regenerating the images.

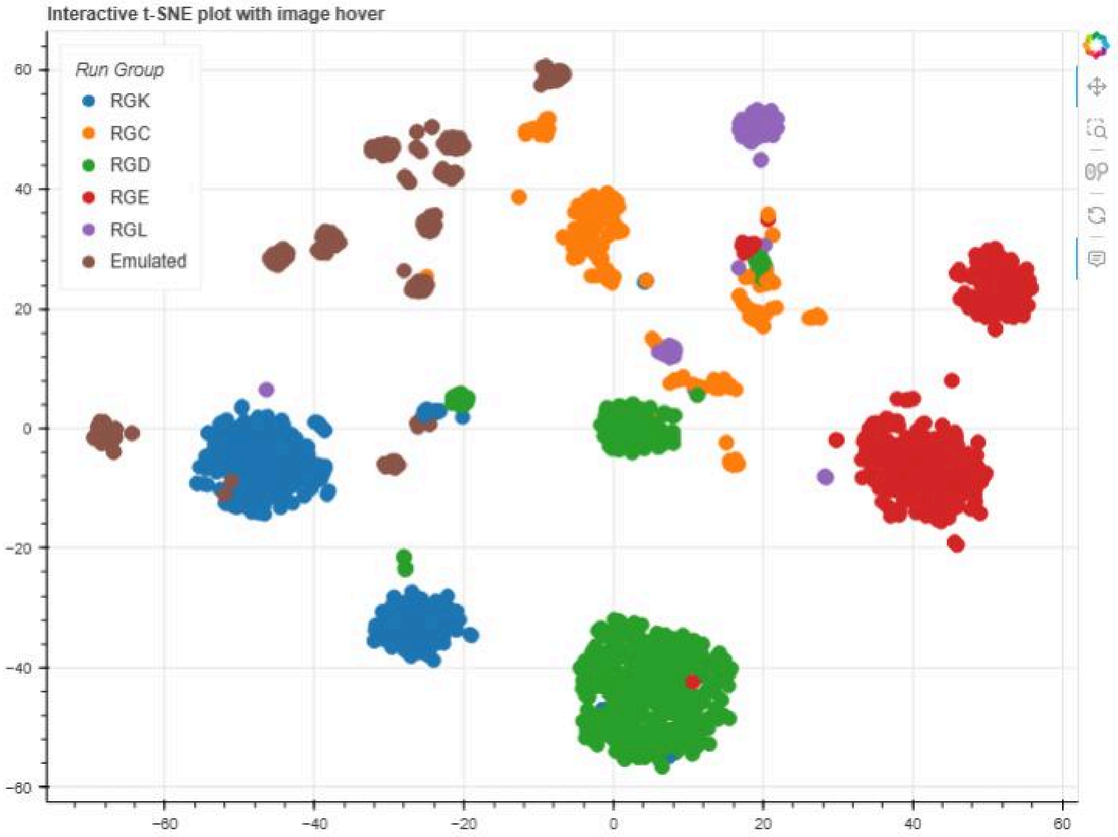


Darren Upton

We can simulate bad plots to use in training.

Image Clustering

How similar are our images within a given run group and across run groups? Can we identify “Bad” or “dissimilar” images without using labels?



t-distributed stochastic neighborhood embedding: similar objects are modeled by nearby points and dissimilar are modeled by distant points. This link, https://distill.pub/2016/misread-tsne/, contains fun visualizations and explanations.

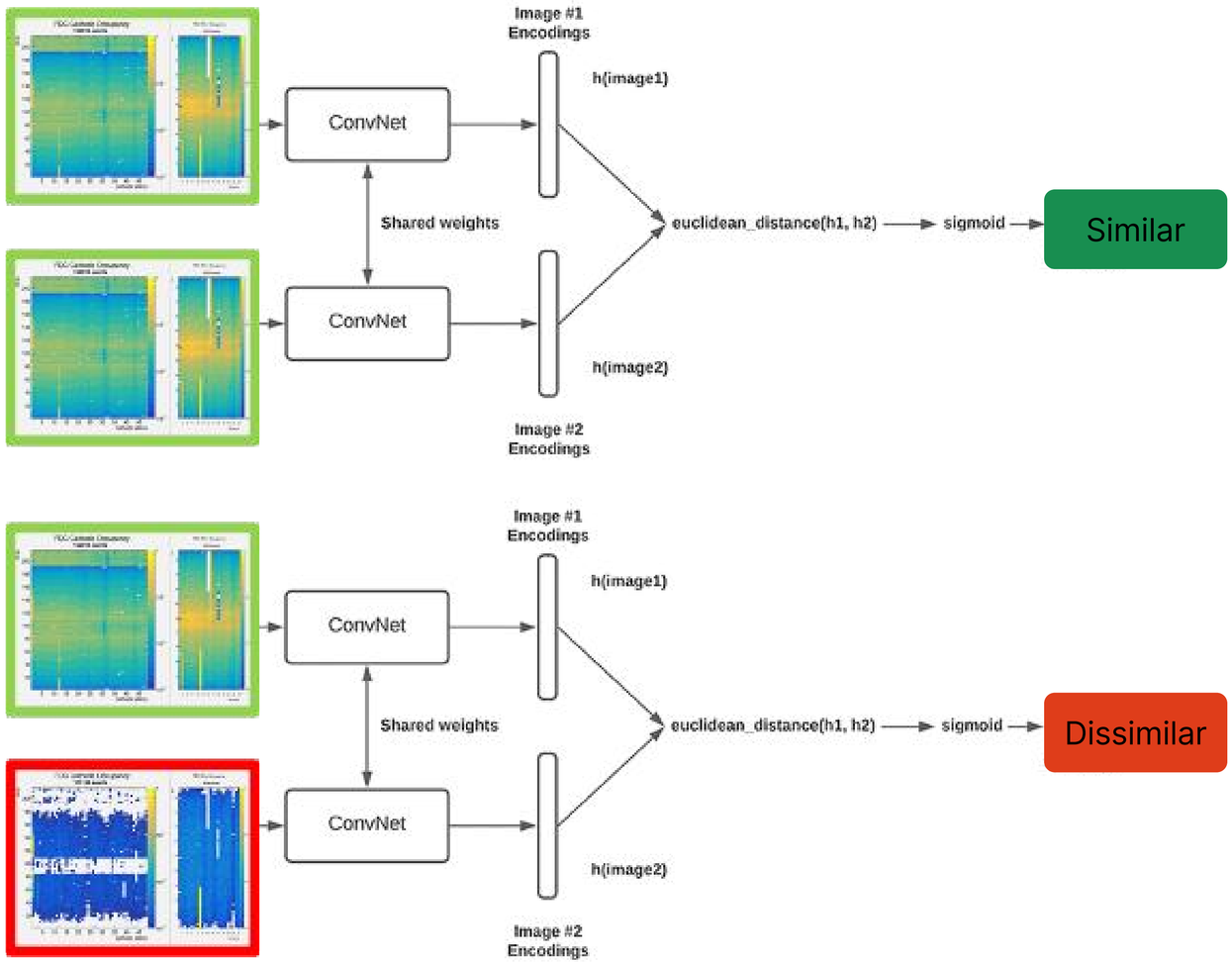
Siamese Models

Siamese models learn to compare two inputs (in our case images) and determine how *similar* they are.

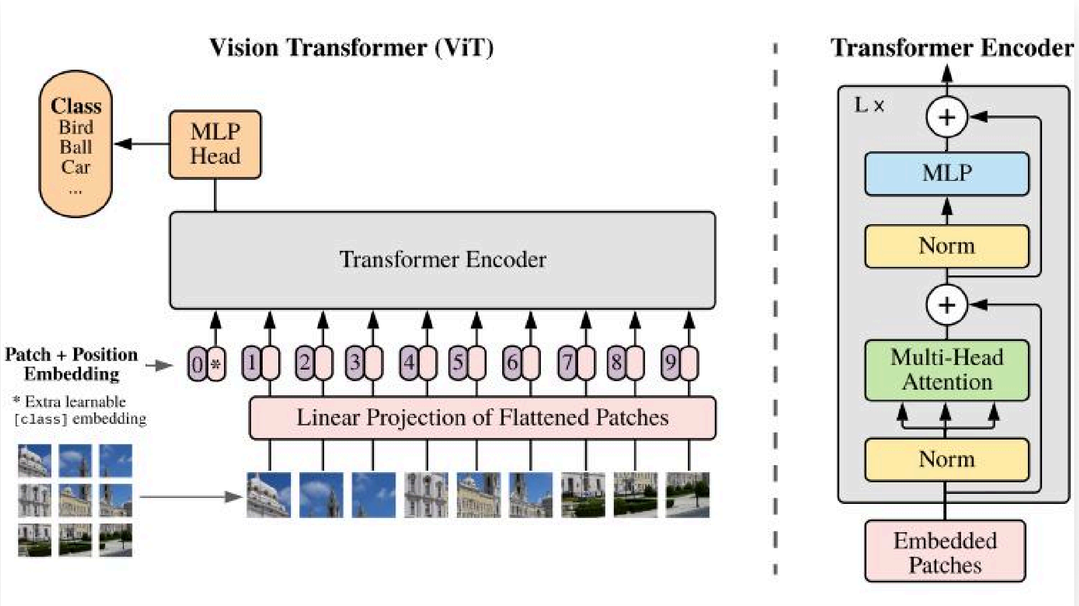
Think of this as Hydra's equivalent to the shift crew comparing images to their reference.

Jordan O'Kronley

UT Physics Graduate Student
Image clustering and siamese models



AI Developments in Progress (aka.. fun side projects)



Vision Transformers

Paper presented at ICLR: https://arxiv.org/pdf/2010.11929
Transformers for Image Recognition at Scale

Autoencoders

Recently approached by medical physicists to use autoencoders for medical image analysis

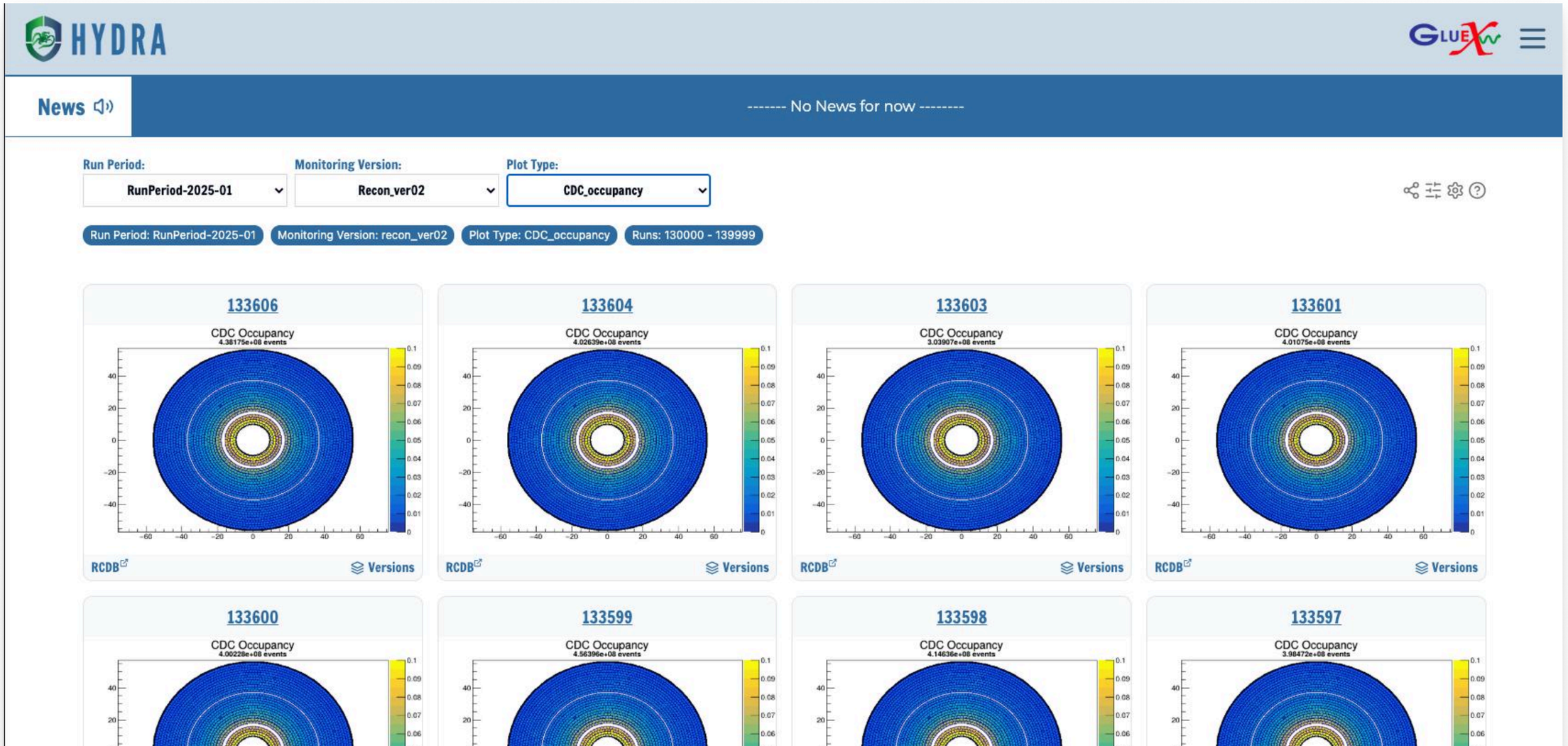
A natural language interface to Hydra

Inspired by n8n's "chat with your database" workflow

"Hey Hydra, tell me what detectors had potential issues within the last 24 hours"

Plot Browser

View, filter, and sort plots based on run numbers, monitoring versions, detectors, simulation campaigns, etc
This is the most used consistently used interface across all of our deployments.



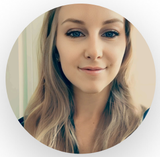
Hydra team

David Lawrence

EPSCI
Physics, CS, AI/ML

Thomas Britton

EPSCI
Physics, CS, AI/ML



Torri Jeske

EPSCI
Operations, Full Stack Development

Nataliia Matsiuk

Information + Records
Containerization, QA

Raiqa Rasool

EPSCI
CS, Full Stack Development

Darren Upton

ODU Physics Grad Student
MLOps, Simulating Bad Images

Jordan O'Kronley

UT Physics Graduate Student
Siamese Models

Conclusions

Organize, store, and Visualize image data

Web accessibility ensures everyone can access the same data from anywhere (with appropriate access controls)

Online Data Quality Monitoring

Jefferson Lab’s experimental halls use Hydra to perform more frequent, accurate, and consistent data quality monitoring at scale

Code is hosted on GitLab

code.jlab.org/epsci/hydra
hydrateam@jlab.org

extras

Development Timeline

2019

Deployed Hydra Run in Hall-D, using InceptionV3 to classify images

2022

Initial refactor to account for different experimental halls

Deployed to Hall-B, scaling from 6 to 61 active models

2024

Deployed to Halls A/C

2025

Consolidation of image-based visualization and analysis tools from Hall-D into Hydra

Implementation of unsupervised techniques

Complete rewrite using React and Flask

2026

Deployed to FAIR/GSI

Implementation of Siamese model architectures