2026 ASAS-NANP Workshop

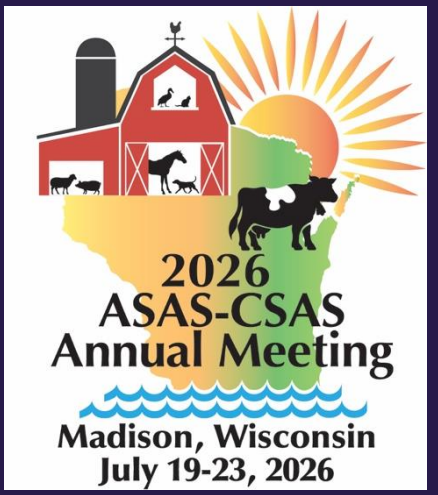
Unlock Computer Vision with AI: Low-Code Solutions for Real-World Precision Livestock Farming Tasks

- Data Collection
- Image Annotation
- Model Training & Deployment

https://github.com/National-Animal-Nutrition-Program/2026ASAS_Xu

Prepared by Dr. Ziteng Xu

Assistant Professor @ Texas A&M University, Department of Poultry Science



Precision Livestock Farming Application

PLF encompasses sensors for the capture of **biological information**, algorithms processing the information, and interfaces that allow for **making use of these data** to **optimize** animal production, health, and welfare (Caria, 2023).

Provide actionable management instructions.

- What are the actionable instructions it can provide to the farmer?
  - Only monitoring the livestock does not enhance efficiency or productivity.
  - Specific assessment leads to specific instructions.
- How much measurable return can it generate for the farmer?
  - Key motivation to adapt the technology.
  - Minimize cost, maximize return.
  - Comprehensive assessment can lead to a higher return.
- How reliable are instructions for farmers to rely on?
  - Maximize accuracy. Minimize false alarms.
  - User-friendly: easy to operate the system and interpret the result.

Data Collection

Sensor Carrier

Stationary

- ✓ Enables 24-7 continuous monitoring.
- ✓ Low infrastructure cost.

- Difficult to trace individual animal.
- Small area coverage

Guided Rail

- ✓ Periodic monitoring.
- ✓ Spatial mapping of the production space.

- High installation cost.
- Low sampling frequency.
- Inflexible path.

Drone

- ✓ Low installation cost.
- ✓ Spatial mapping of production space.
- ✓ Flexible path.

- Noise
- Animal safety
- Limited load capacity.
- Low sampling frequency.

Ground Robot

- ✓ Can carry out certain actions in the field.
- ✓ Spatial mapping of production space.
- ✓ Flexible path.

- Limited power supply and load capacity.
- Low sampling frequency.
- Direct contact with the animals

Data Collection

Sensor Placement/Configuration

› Sensor placement/configuration is one of the most important key design factor for a successful PLF application.

› Some of the key elements for consideration when using imagery sensors:

1. *Angle of view:*
   - *Select a viewing angle that can best capture the selected independent variables.*
   - *Select a view angle that can maximize the sensor's usage.*
2. *Distance:*
   - *Group assessment vs. individual assessment.*
   - *Trade-off between the monitoring area and image details*
3. *Location:*
   - *Places at the area that animals are more likely to visit or stay.*
   - *Unobstructed view*
4. *Monitoring Frequency*
   - *Reduce monitoring frequency -> Increase coverage or reduce computation load*
5. *Consistent placement*
   - *Reusability of previously developed ML models.*

Data Collection

Sensor Placement/Configuration

› PLF for Caged Layer System

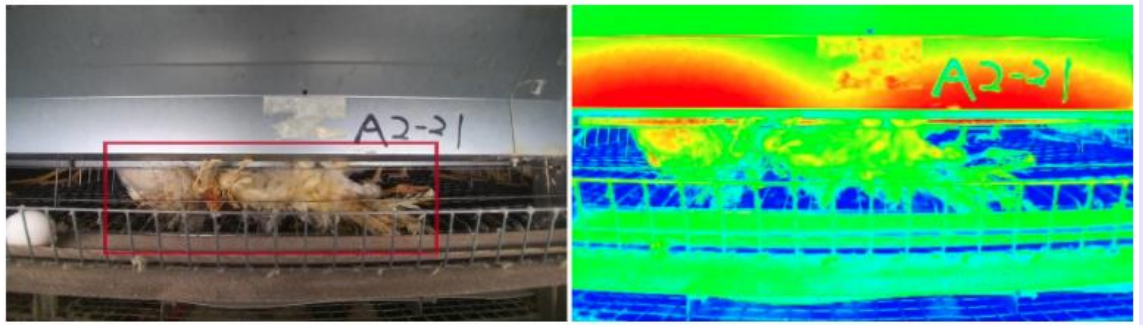
Detection of dead birds via thermal camera.

Pros:

- Labor reduction: farmer only need to remove dead birds in specific cages identified by the system.

Cons:

- Unable to provide timely insight to birds' health status.
- Limited field of view, may not be able to detect dead birds in the back of the cage.

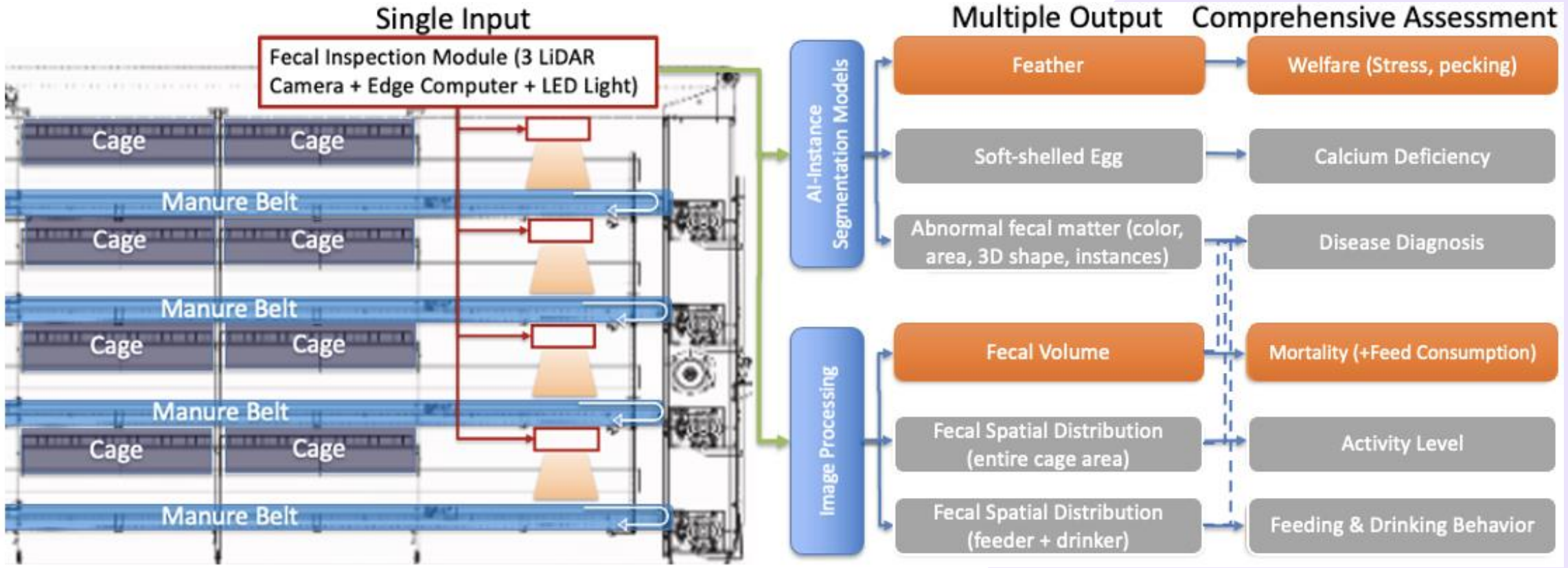


Source Tong et al., 2022

Data Collection

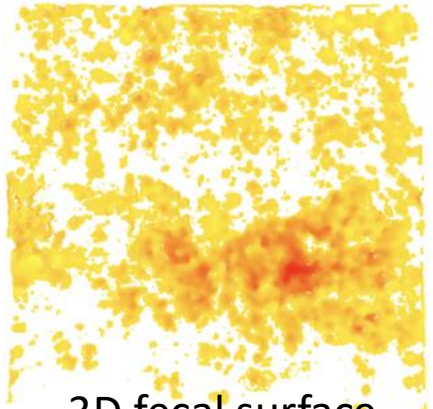
Sensor Placement/Configuration

› PLF for Caged Layer System

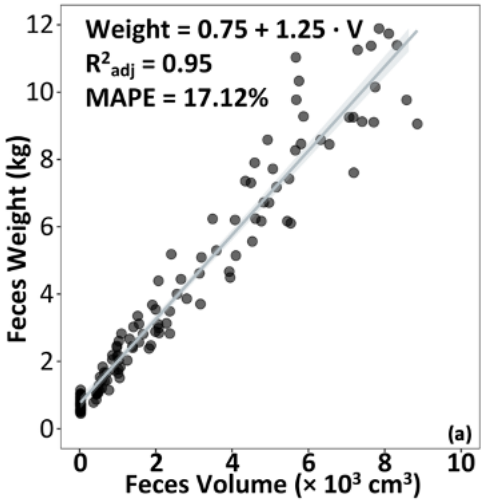


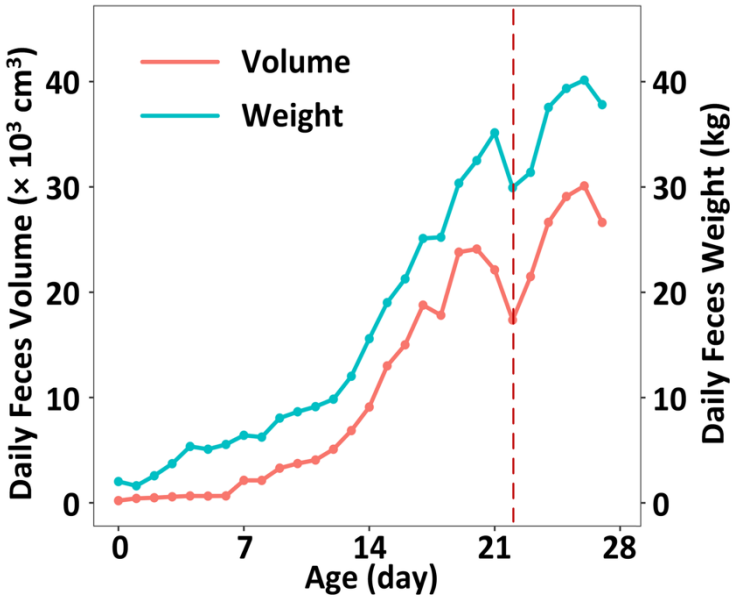
Sensor Placement/Configuration

› PLF for Caged Layer System

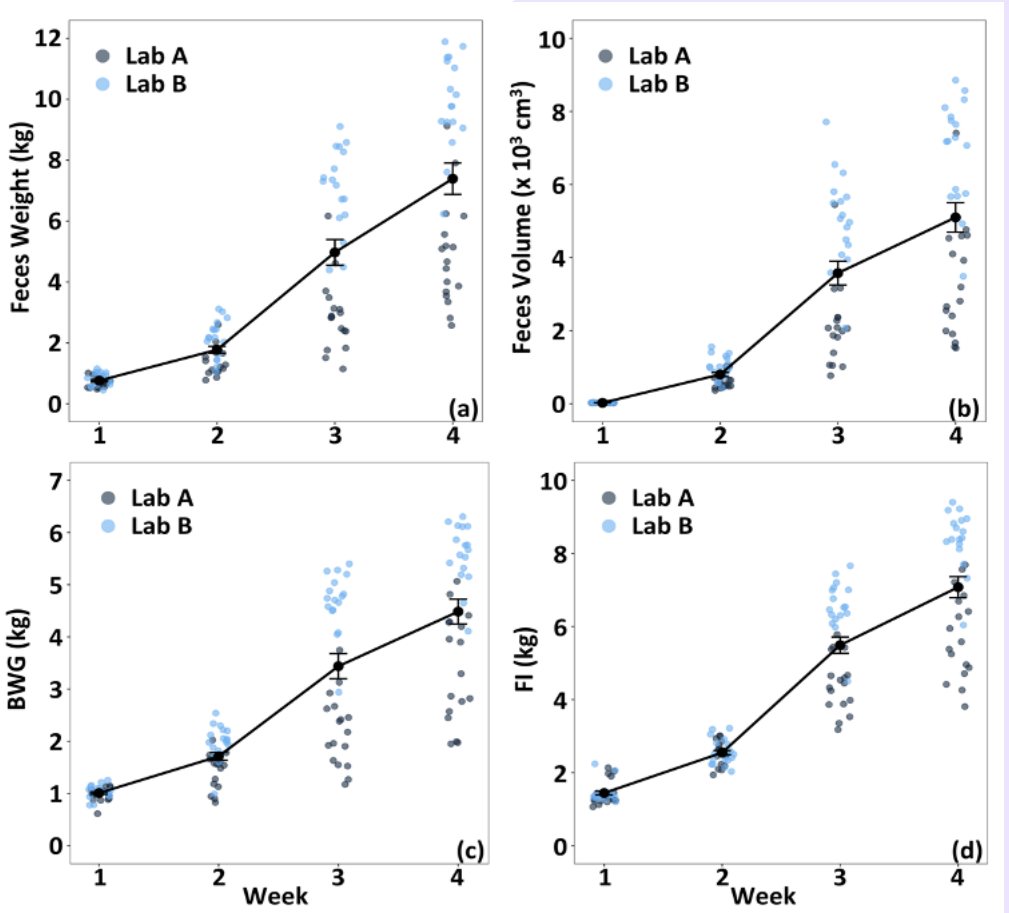


3D fecal surface





Less feces produced when there are mortality present

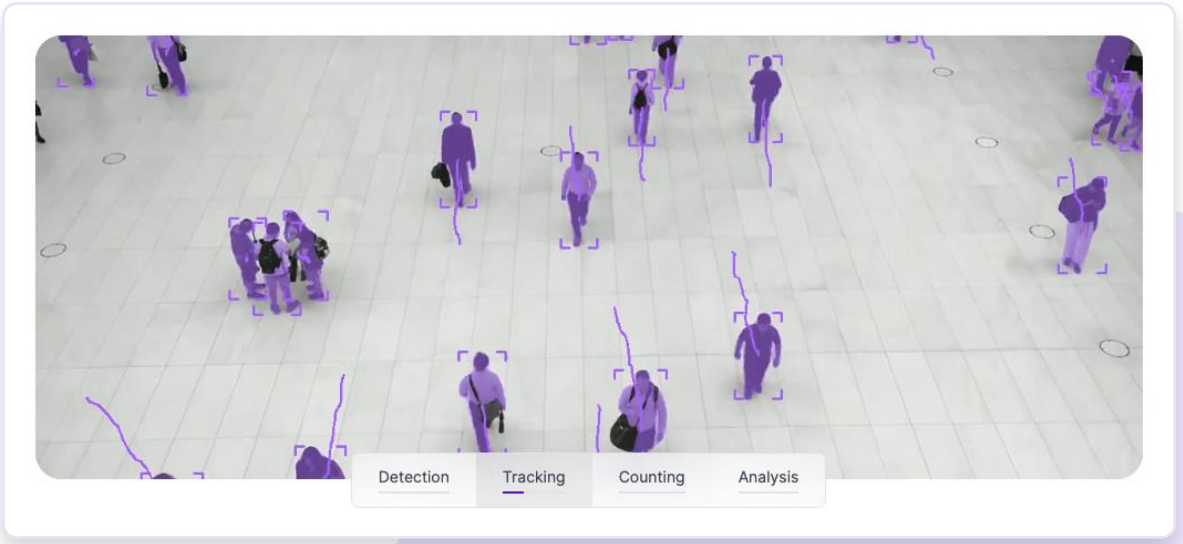


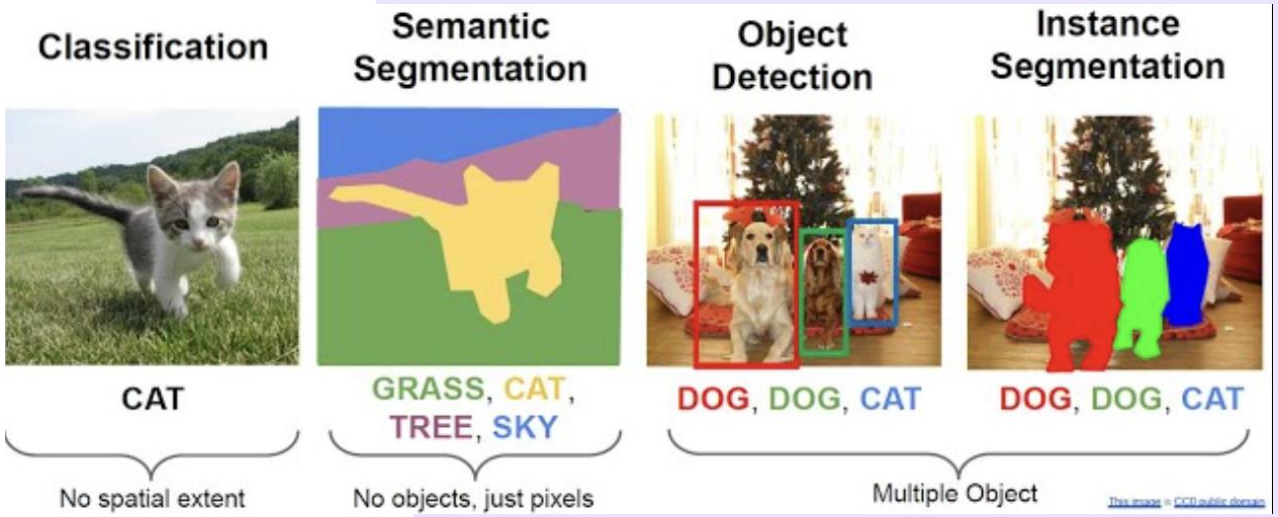
Sick animal (Lab A) consumes less feed and therefore produces less feces

Roboflow

Label, augment & export datasets

roboflow.com





PART 1 · ROBOFLOW

Image Classification

SINGLE TASK

Single label

One output per image — e.g. posture classification.

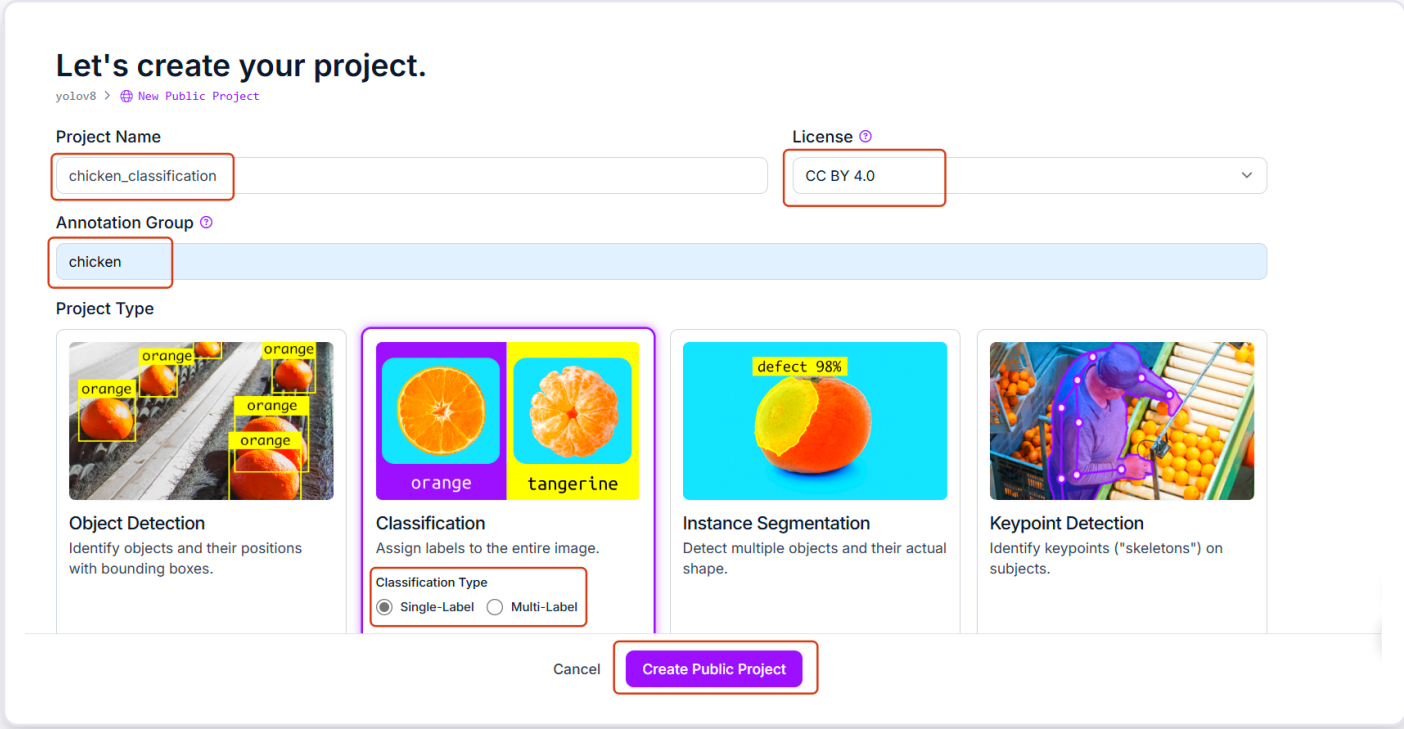
MULTI TASK

Multiple labels

Several outputs at once — e.g. “Pig standing” + “Piglet present.”

1 **Create Project**

Choose Classification as the project type, then name the project and set the annotation group.

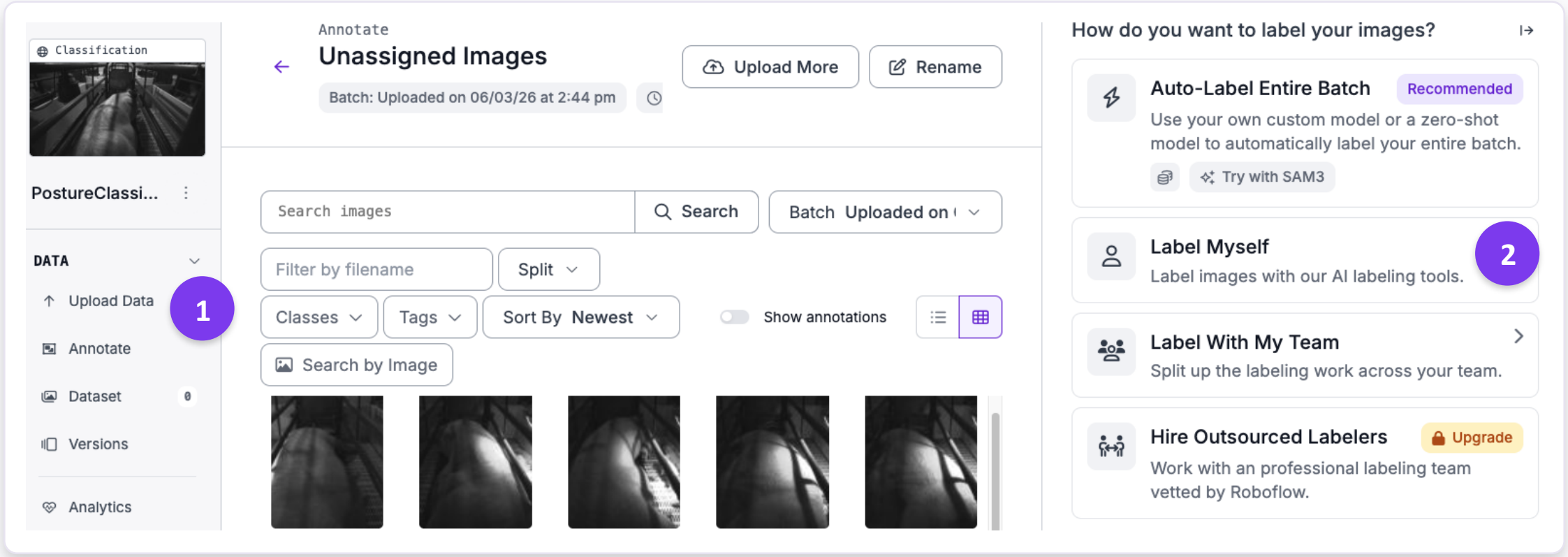


CLASSIFICATION · STEP 2

Upload Images

1 **Open the Upload tab**
Drag in images or a folder of files.

2 **Process the batch**
Roboflow queues every image for annotation.



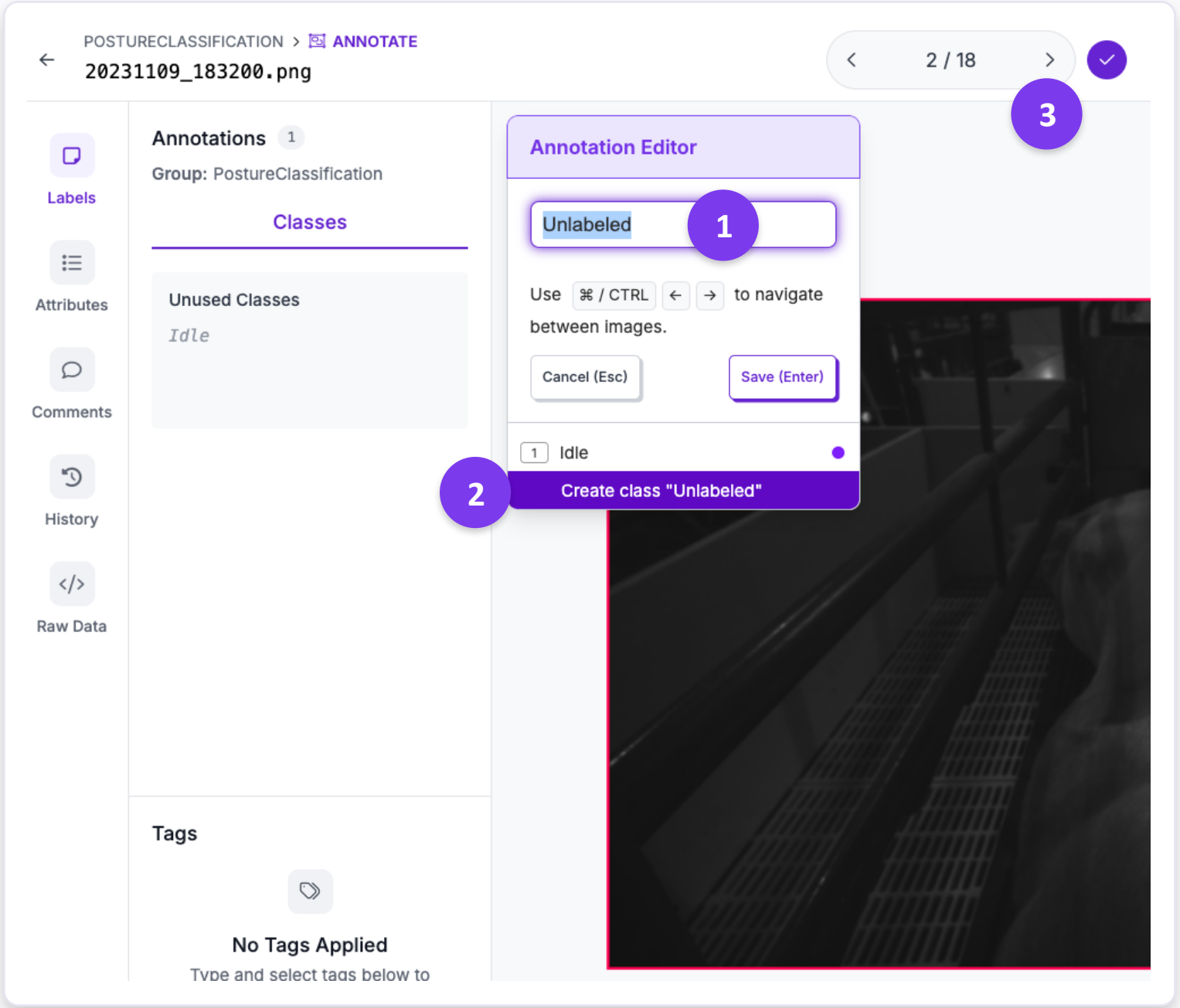
CLASSIFICATION · STEP 3

Image Annotation

1 Open the editor
Select an unassigned image to begin.

2 Assign a class
Pick the label that describes the image.

3 Save & continue
Move to the next image and repeat.



CLASSIFICATION · STEP 4

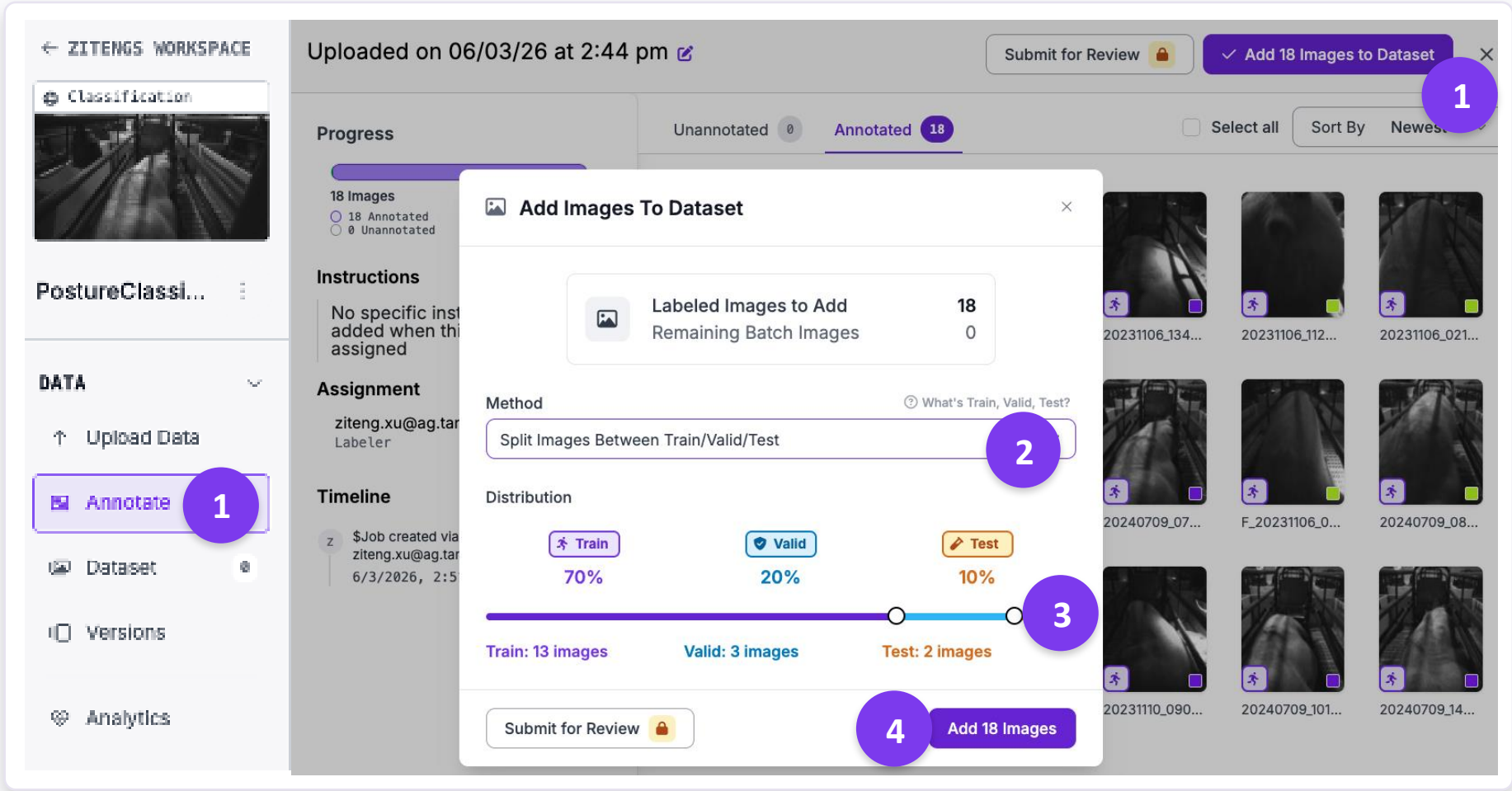
Split the Dataset

1. **Add images to dataset**
   Send labeled images into the version.

2. **Train / Valid / Test**
   Roboflow proposes a balanced split.

3. **Adjust ratios**
   Tune the slider to your needs.

4. **Confirm**
   Lock the split for this version.

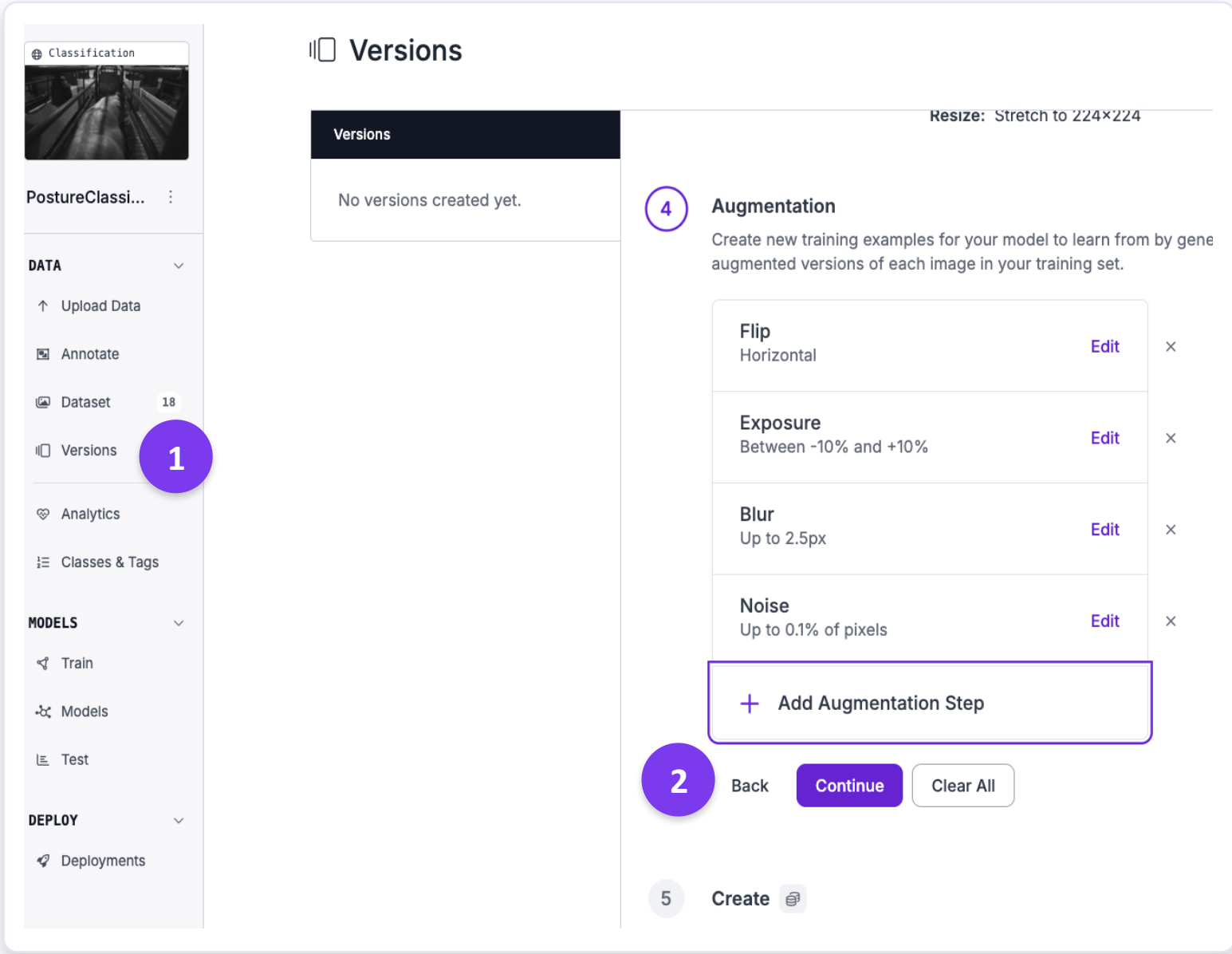


CLASSIFICATION · STEP 5

Apply Augmentation

1 **Add augmentation step**
Expand the training set with transforms.

2 **Choose transforms**
Flip, rotate, blur, brightness and more.



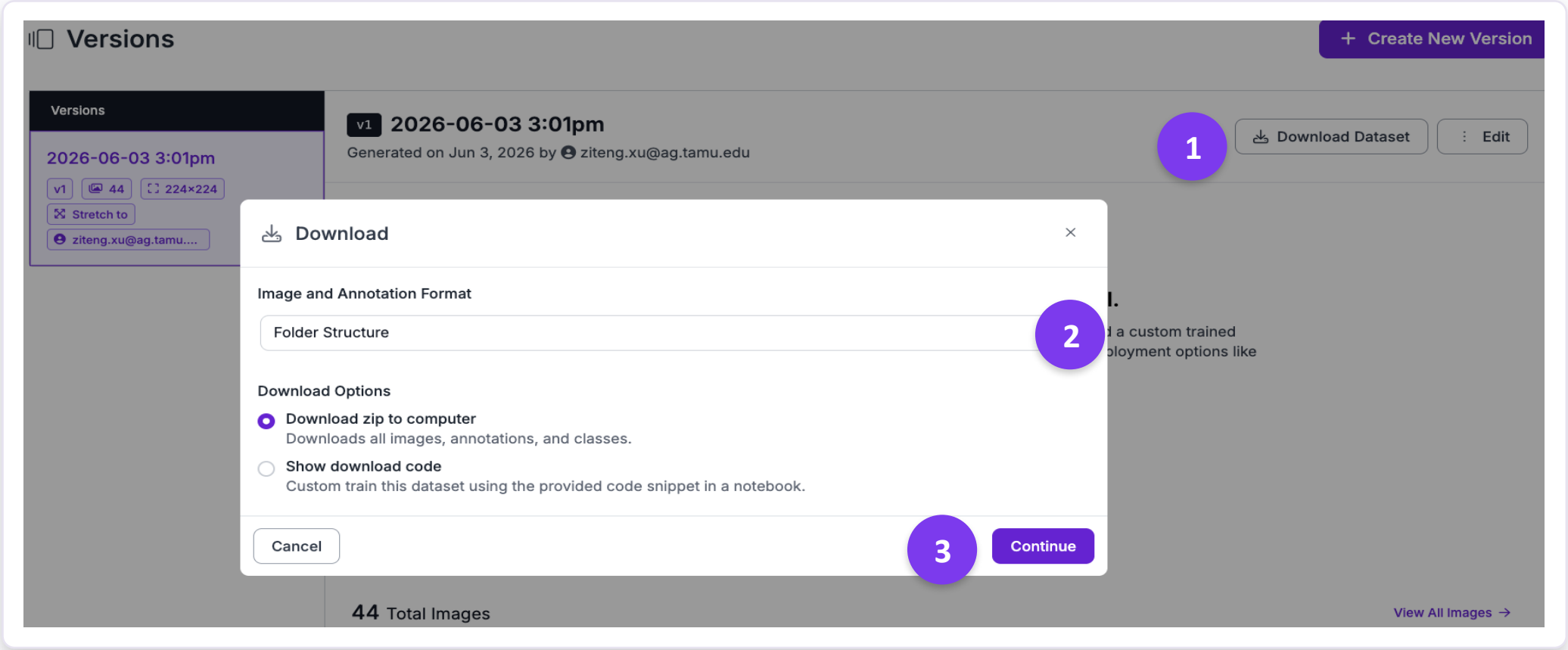
CLASSIFICATION · STEP 6

Export the Dataset

1 **Generate a version**
Bundle images, splits and augmentations.

2 **Pick a format**
Choose the export format for your model.

3 **Download or link**
Grab a zip or a code snippet.

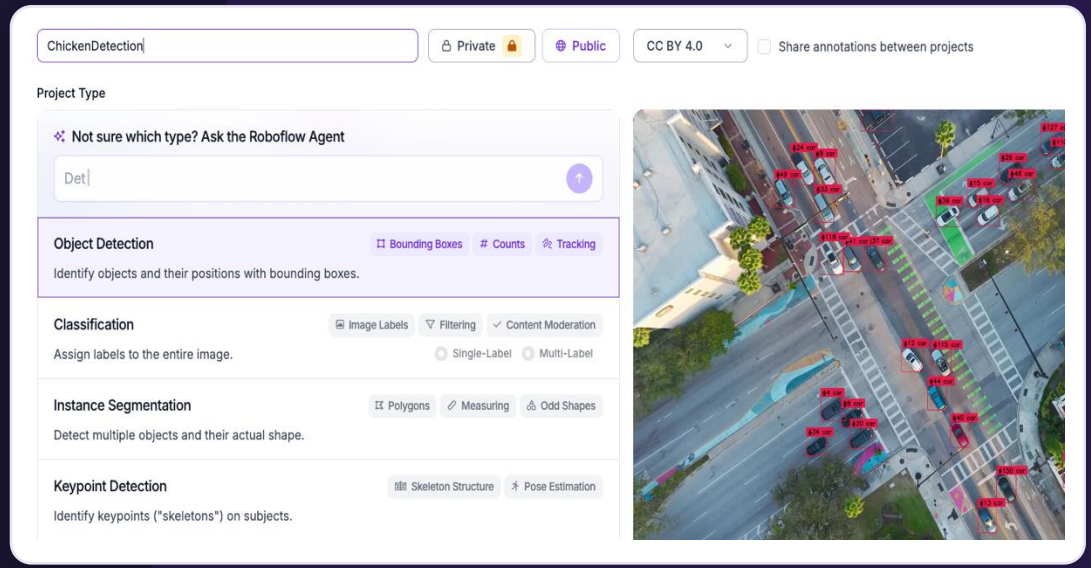


2

ROBOFLOW

Object Detection

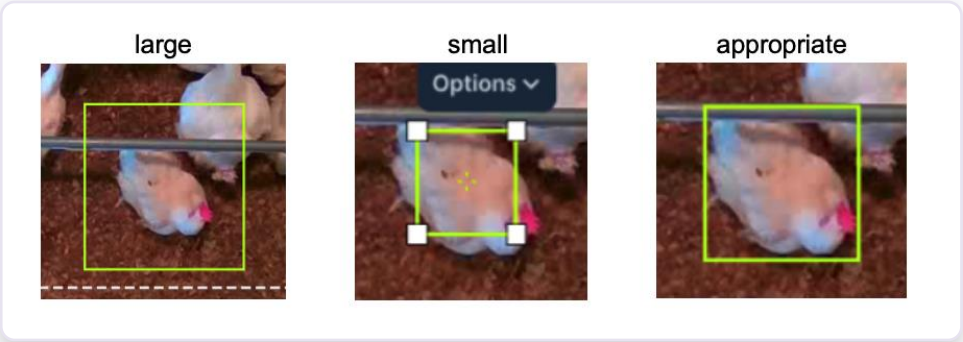
Draw bounding boxes around every object of interest — the foundation of detection datasets.

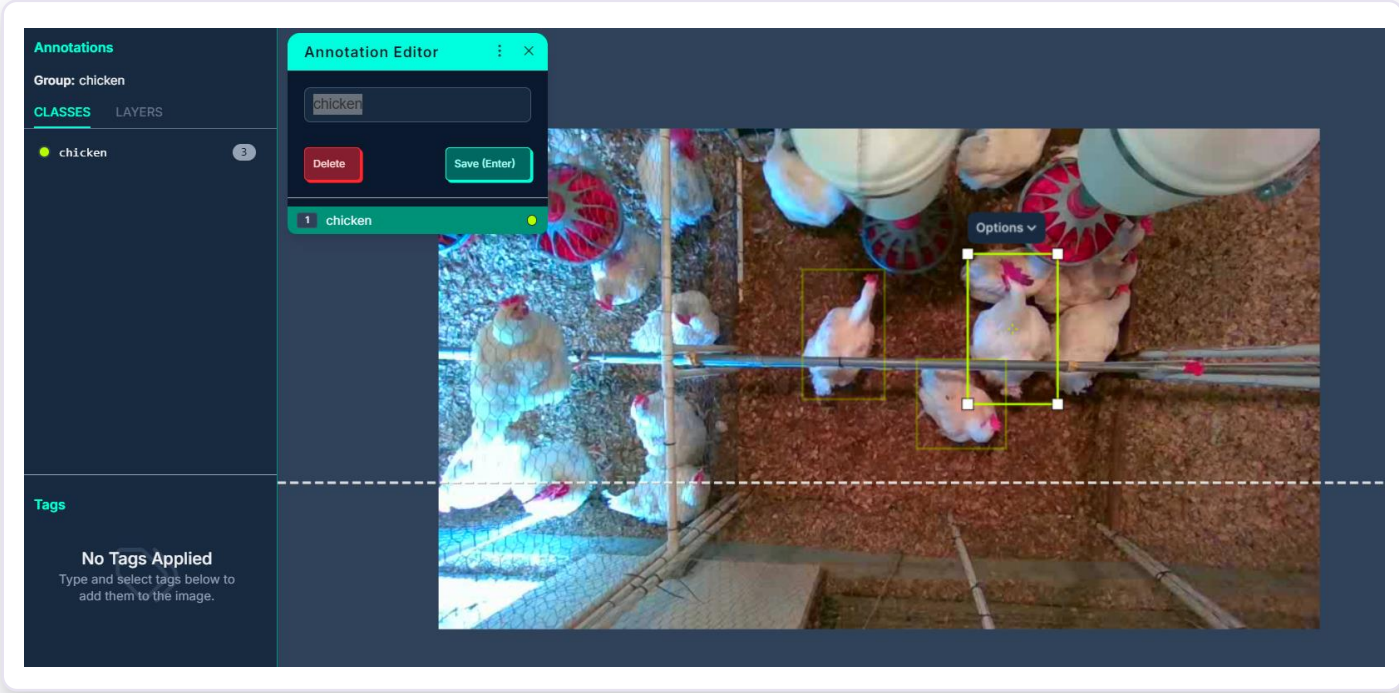


OBJECT DETECTION · STEP 1

Drawing Bounding Boxes

Press **B** to start a box. Draw a tight rectangle around one object and name it, then repeat for every object in the region of interest.

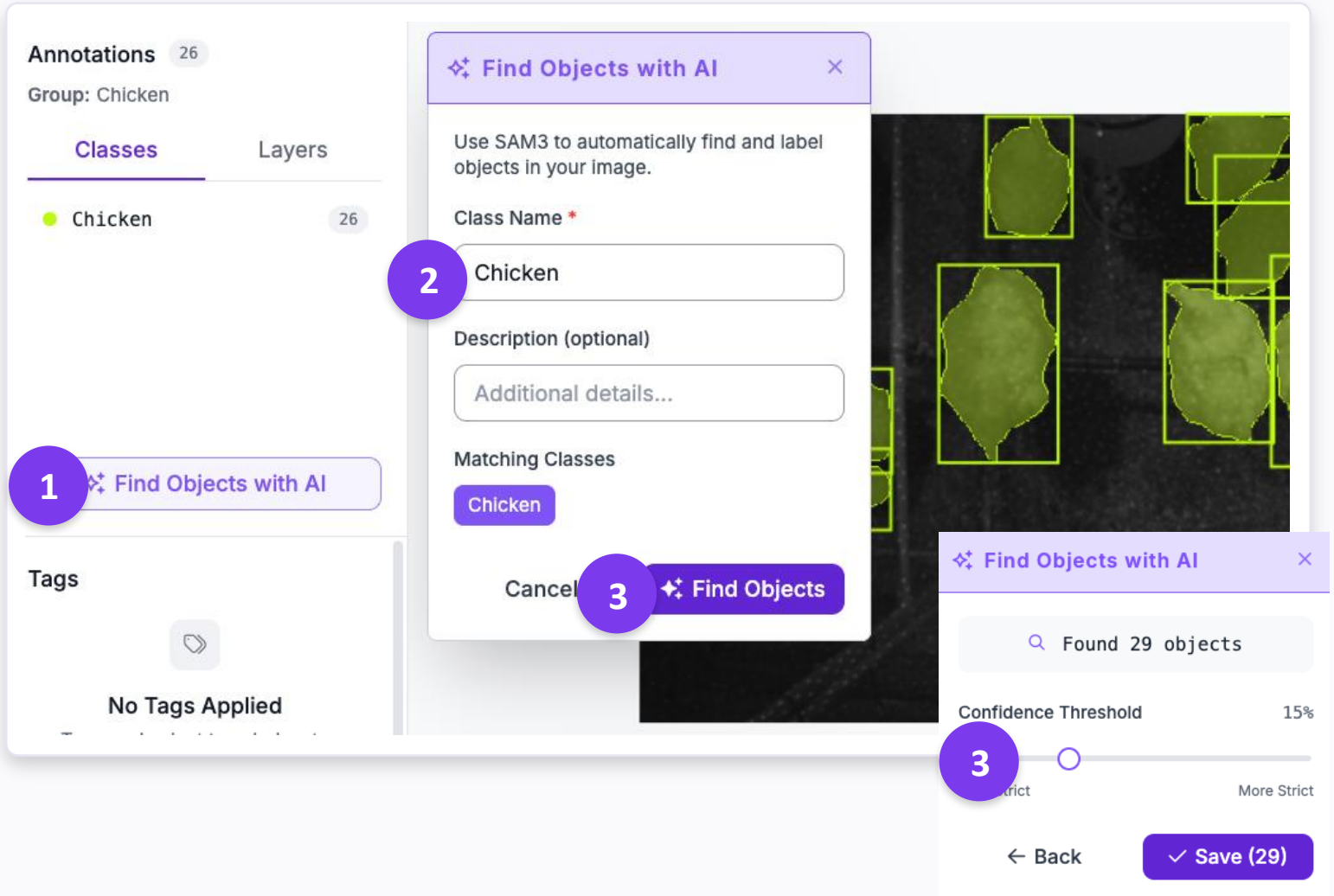




OBJECT DETECTION · AUTO-LABEL

Find Objects with AI

1. **Find Objects with AI**
   Let the model propose detections.
2. **Enter a class name**
   Tell it what to look for.
3. **Tune the threshold**
   Lower it to catch all candidates.
4. **Clean up**
   Delete misclassifications in the image or Layers tab.



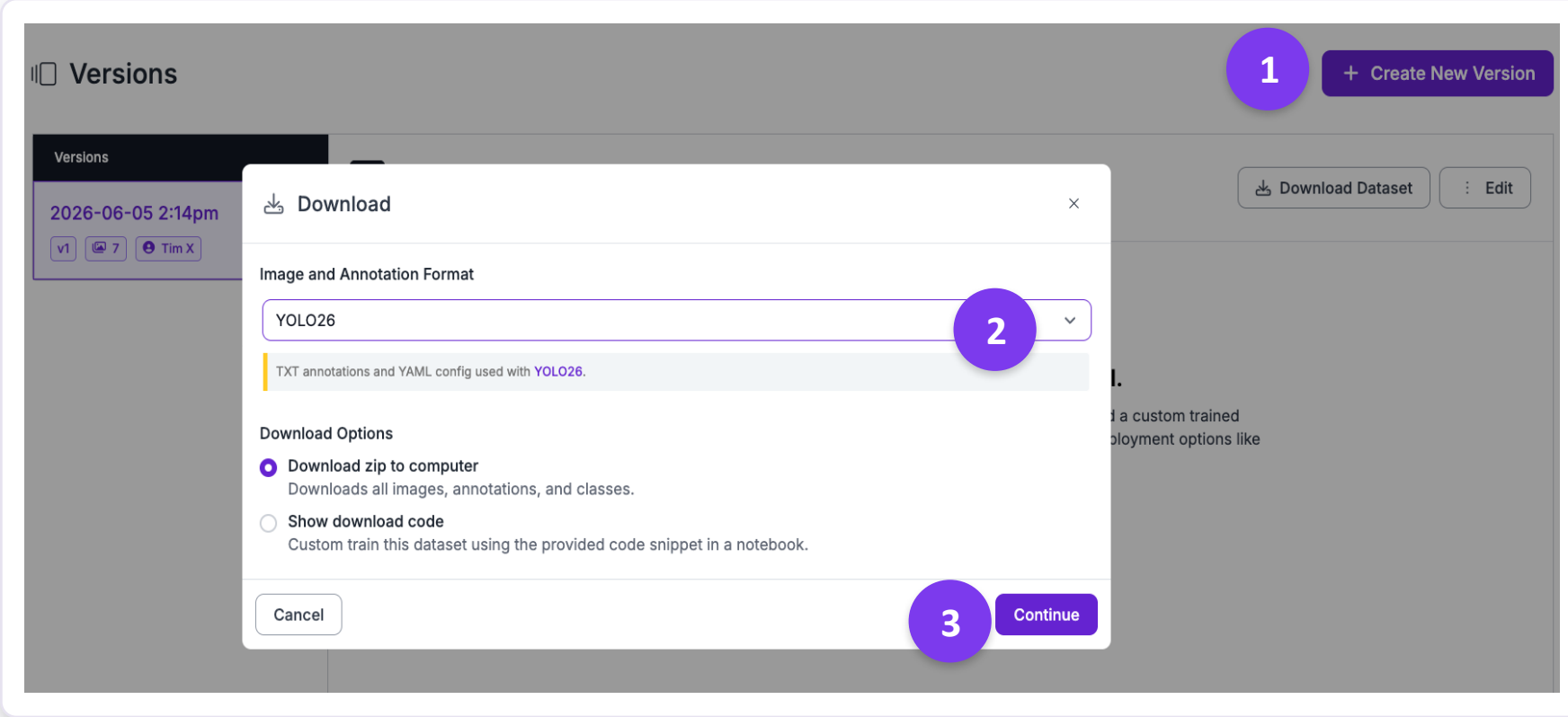
OBJECT DETECTION · STEP 2

Generate a Version

› Bundle your labeled images into a dataset version.

› Re-use the same split → augment → export flow from Part 1.

› Each version is reproducible and shareable.

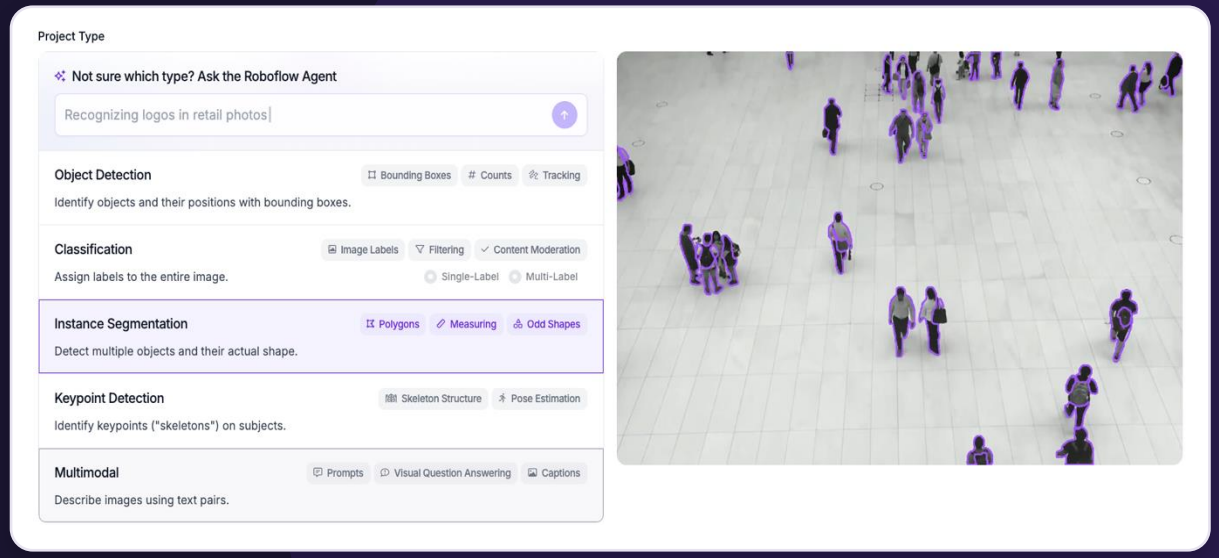


3

ROBOFLOW

Instance Segmentation

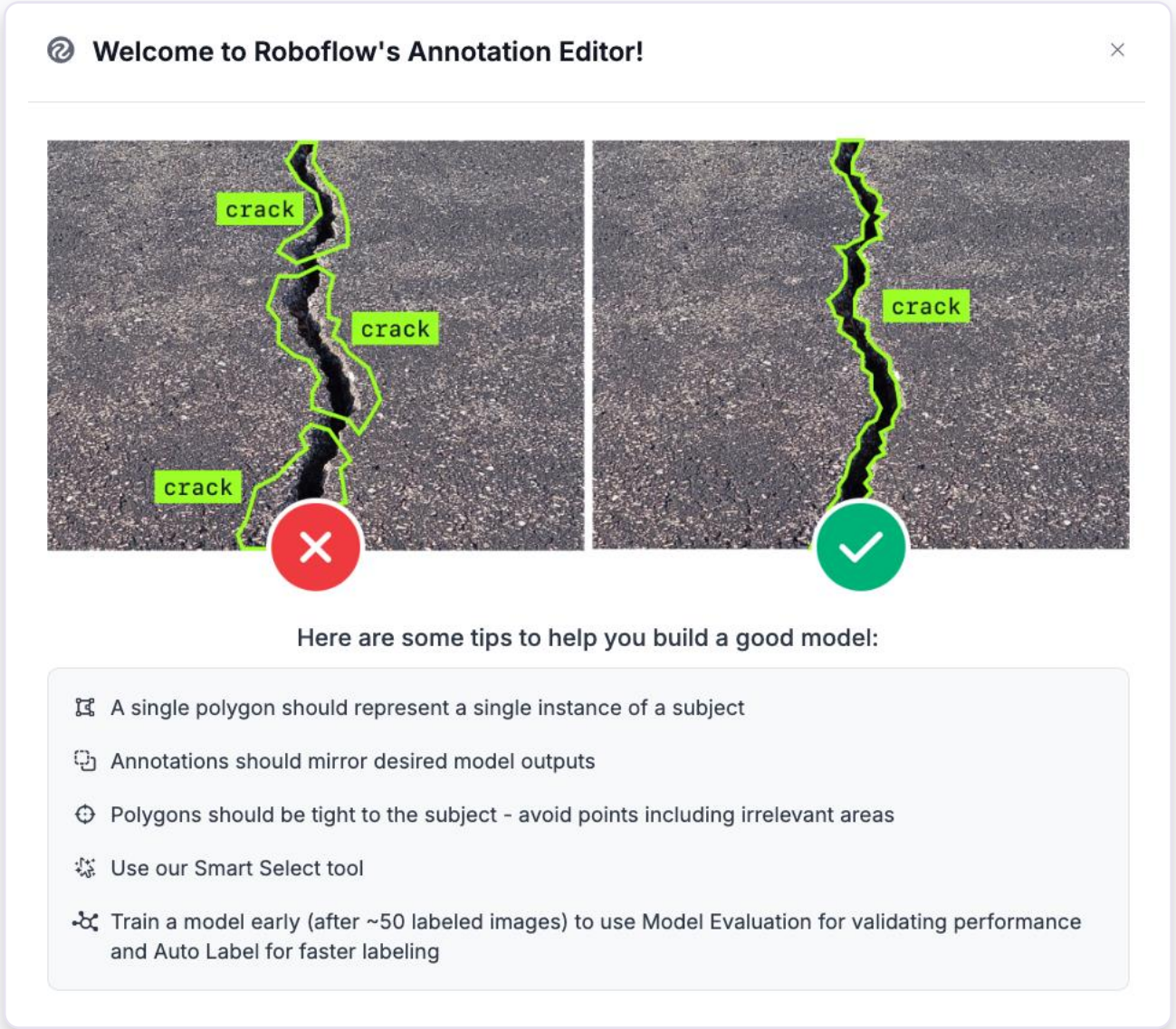
Trace each object with a polygon mask — one polygon per instance, tight to the subject.



INSTANCE SEGMENTATION

Labeling Best Practices

- One polygon = one instance of a subject.
- Annotations should mirror your desired model outputs.
- Keep polygons tight — avoid points over irrelevant areas.
- Use the Smart Select tool to speed up tracing.
- Train early (~50 images) to unlock Model Evaluation & Auto Label.



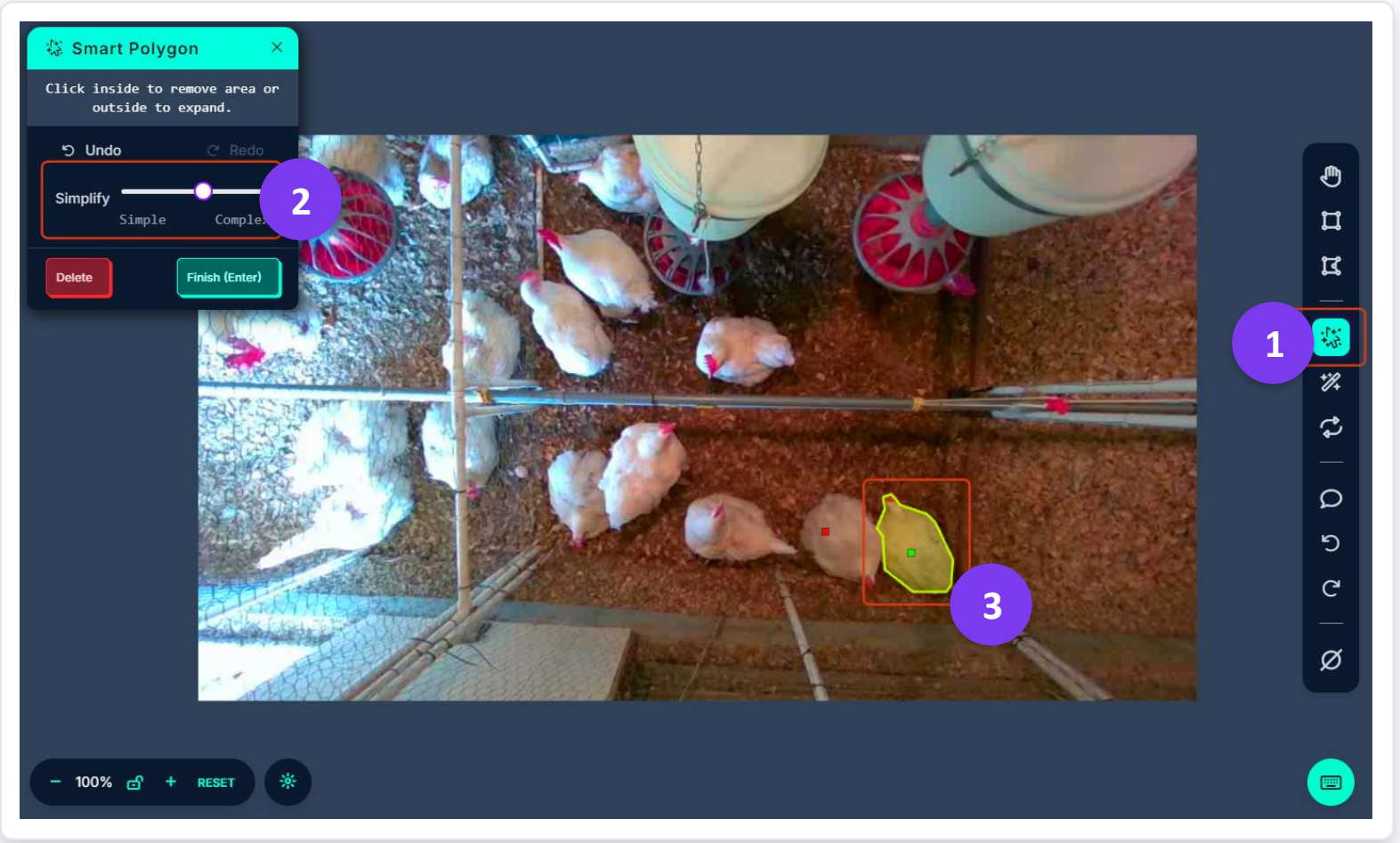
INSTANCE SEGMENTATION · WORKFLOW

Smart Polygon Tool

1 **Pick Smart Polygon**
Then click the object in the image.

2 **Refine the mask**
Click to add area, click again to subtract.

3 **Split instances**
Click outside to extend, inside to remove.

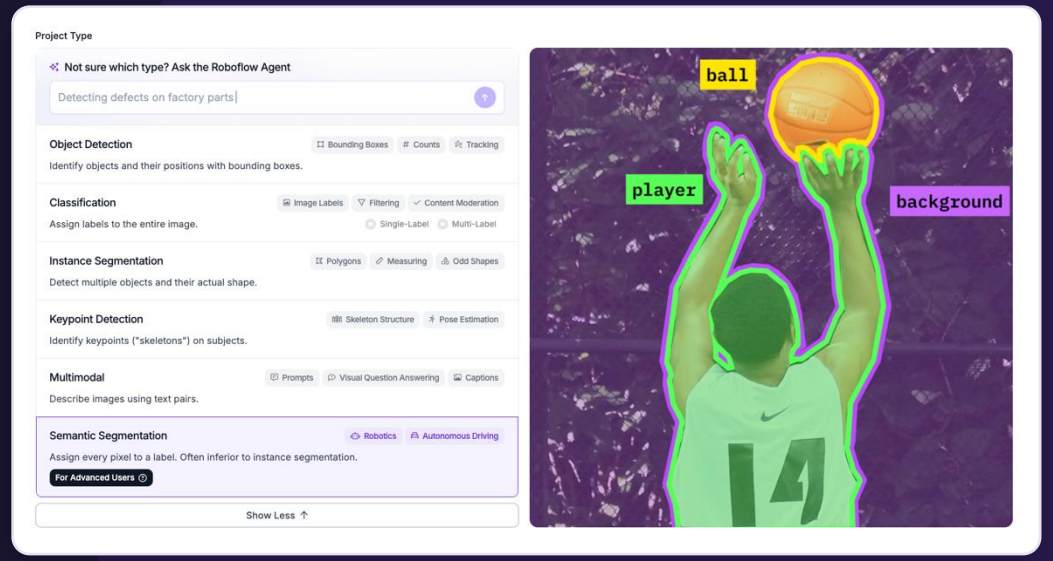


4

ROBOFLOW

Semantic Segmentation

Label every pixel by class — no separation between individual instances (Same procedure as instance segmentation).

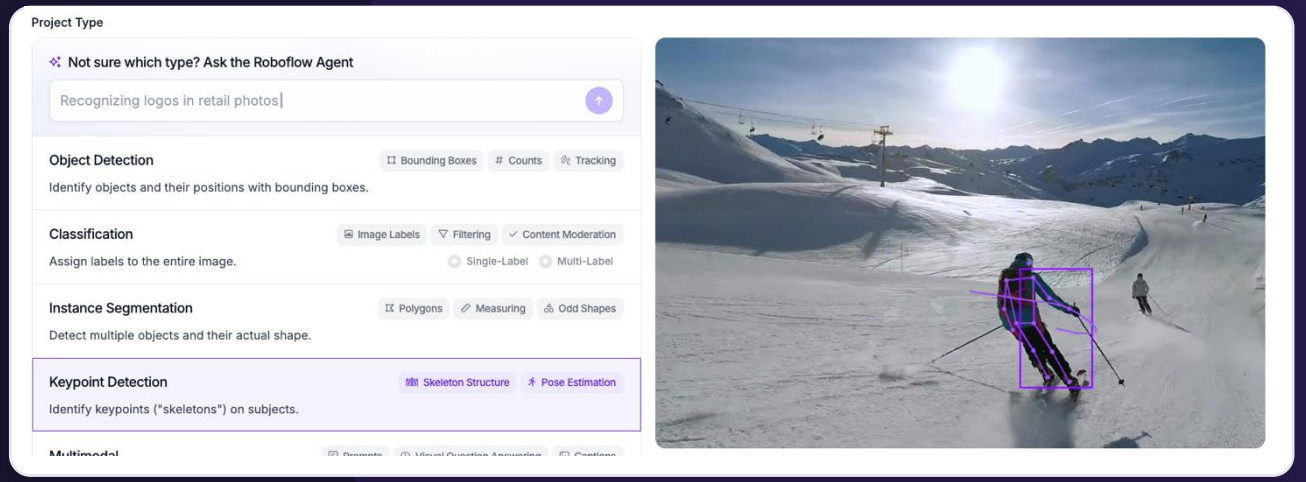


5

ROBOFLOW

Keypoint Detection

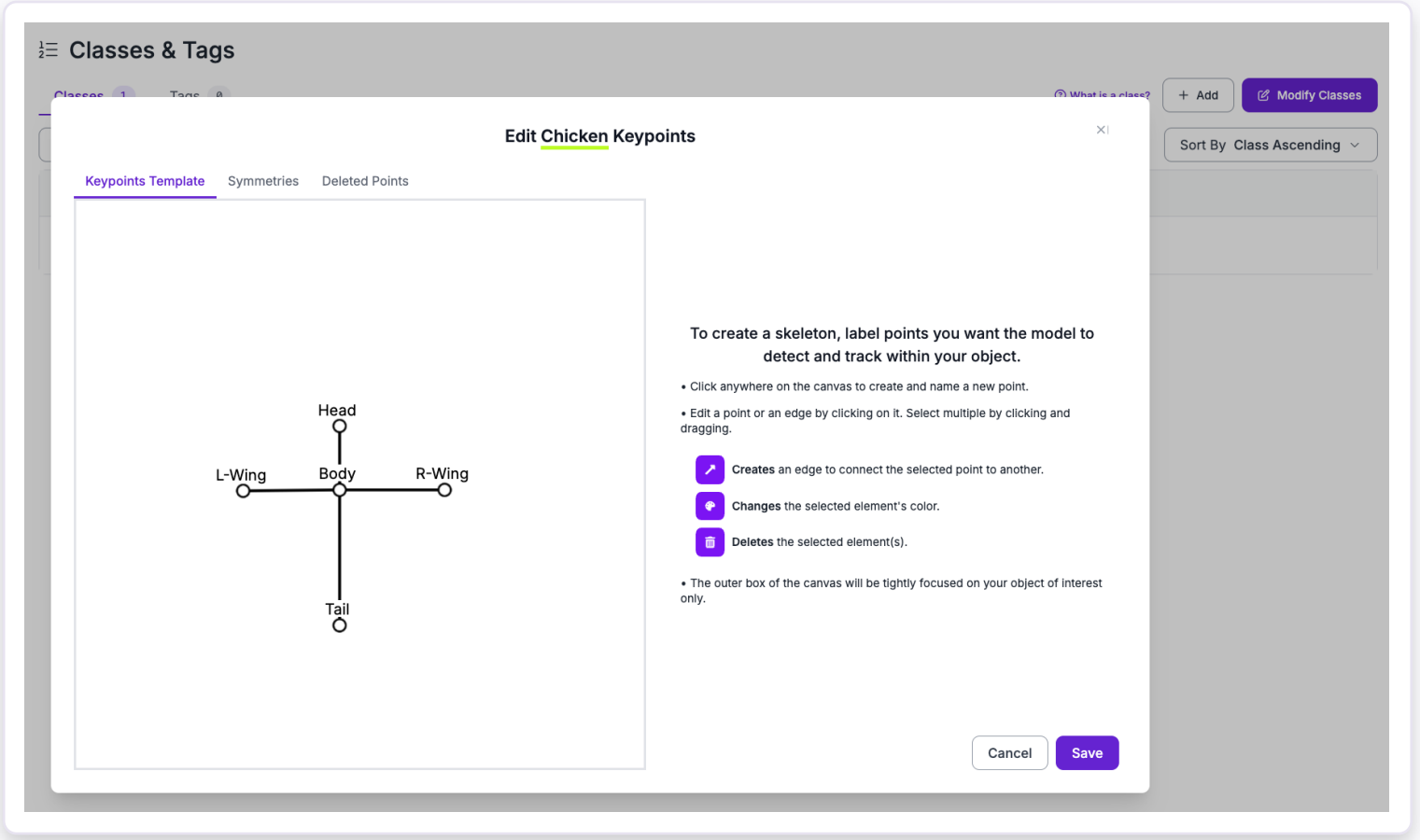
Define a skeleton of keypoints inside a bounding box — for pose and landmark tasks.



KEYPOINT DETECTION · SETUP

Define the Skeleton

- Create a skeleton class and edit its keypoints.
- Name each keypoint and connect them into a graph.
- This template is reused for every annotated object.



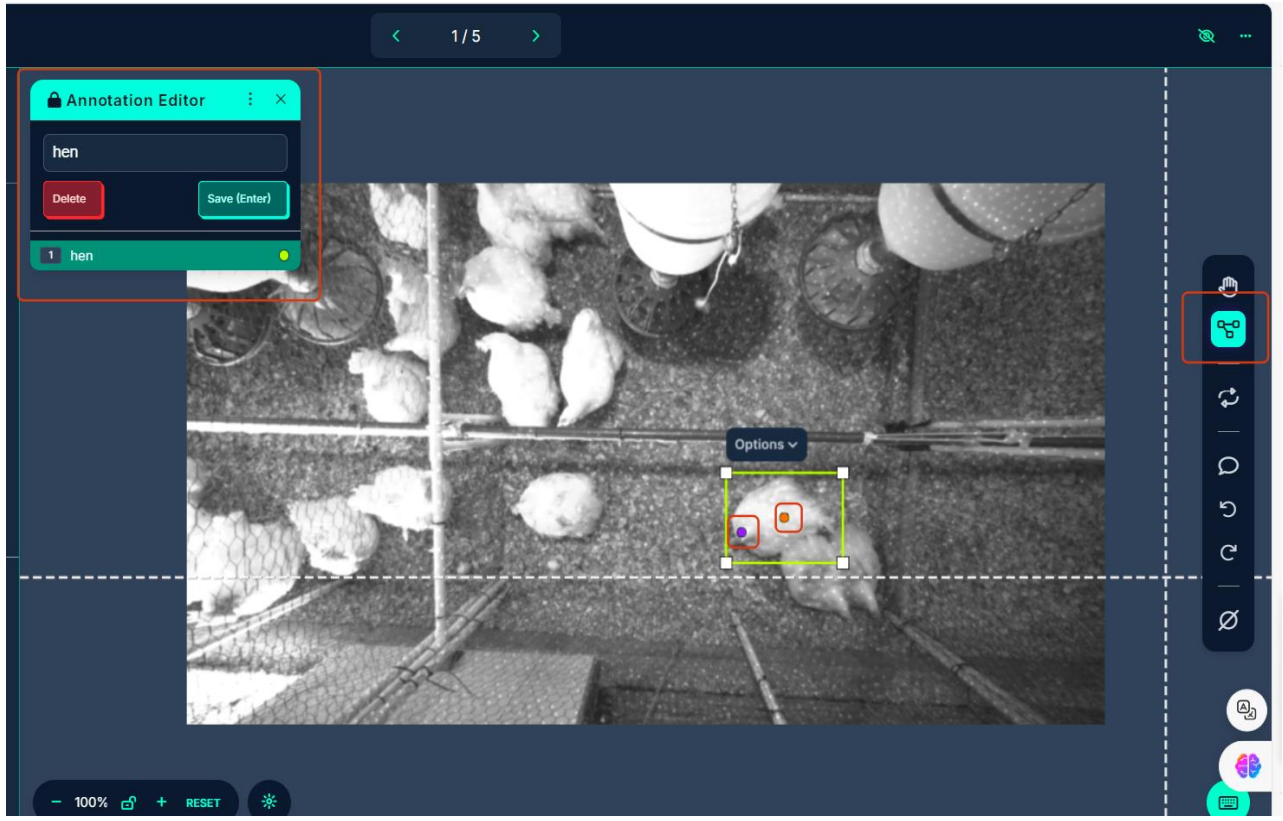
KEYPOINT DETECTION · WORKFLOW

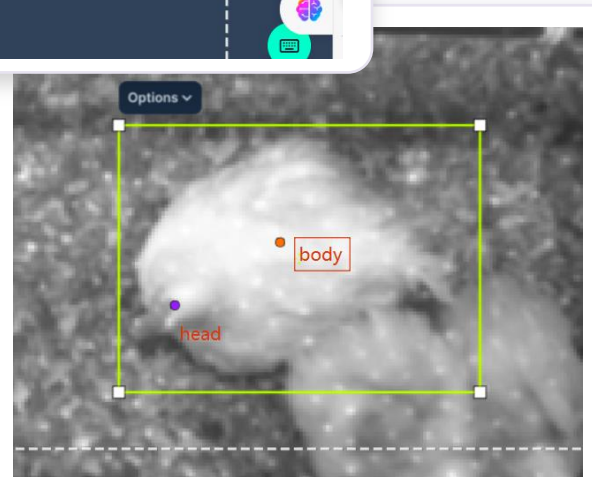
Placing Keypoints

1 **Box the object**
Add a bounding box first.

2 **Position keypoints**
Drag each point to its landmark.

3 **Stay inside the box**
Never place a keypoint outside the box.





6

TRAINING

Ultralytics HUB

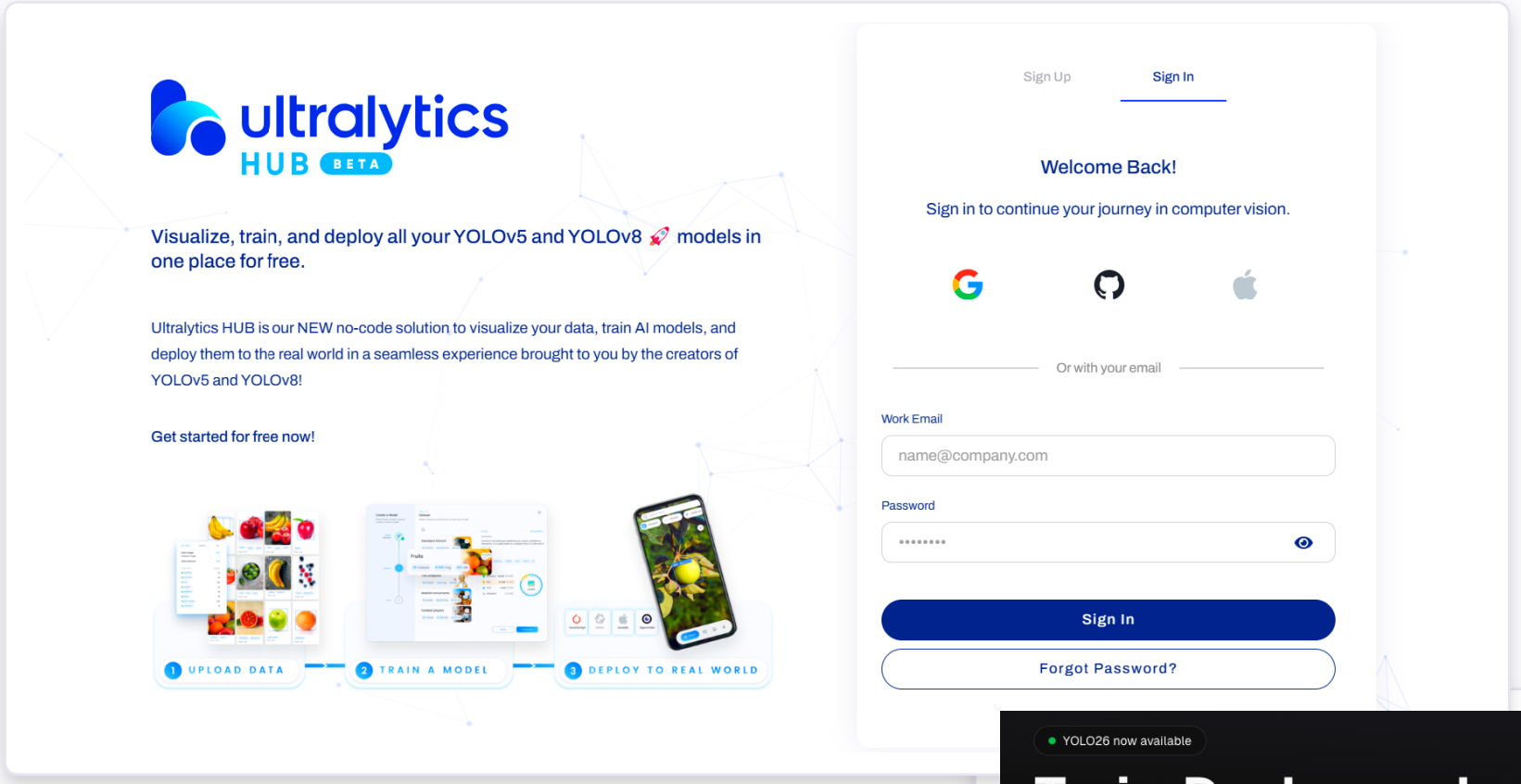
Take your labeled dataset and train, deploy and evaluate YOLO models in the cloud or locally.

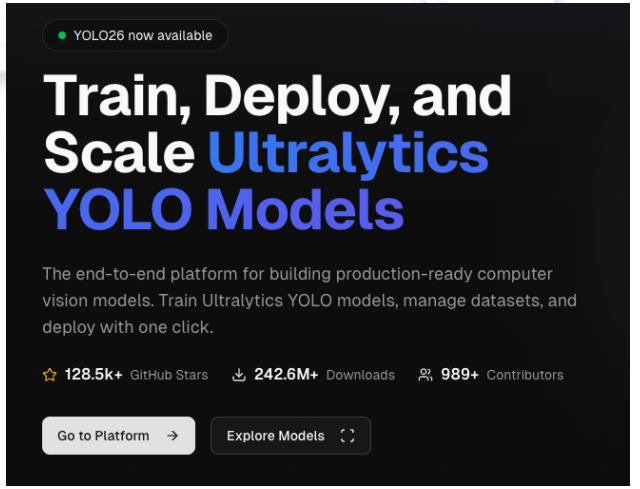
Registration

1. **Visit Ultralytics**
   hub.ultralytics.com/ or platform.ultralytics.com/

2. **Create an account**
   Fast, intuitive onboarding.

3. **Ready to train**
   Upload datasets & train YOLO models.





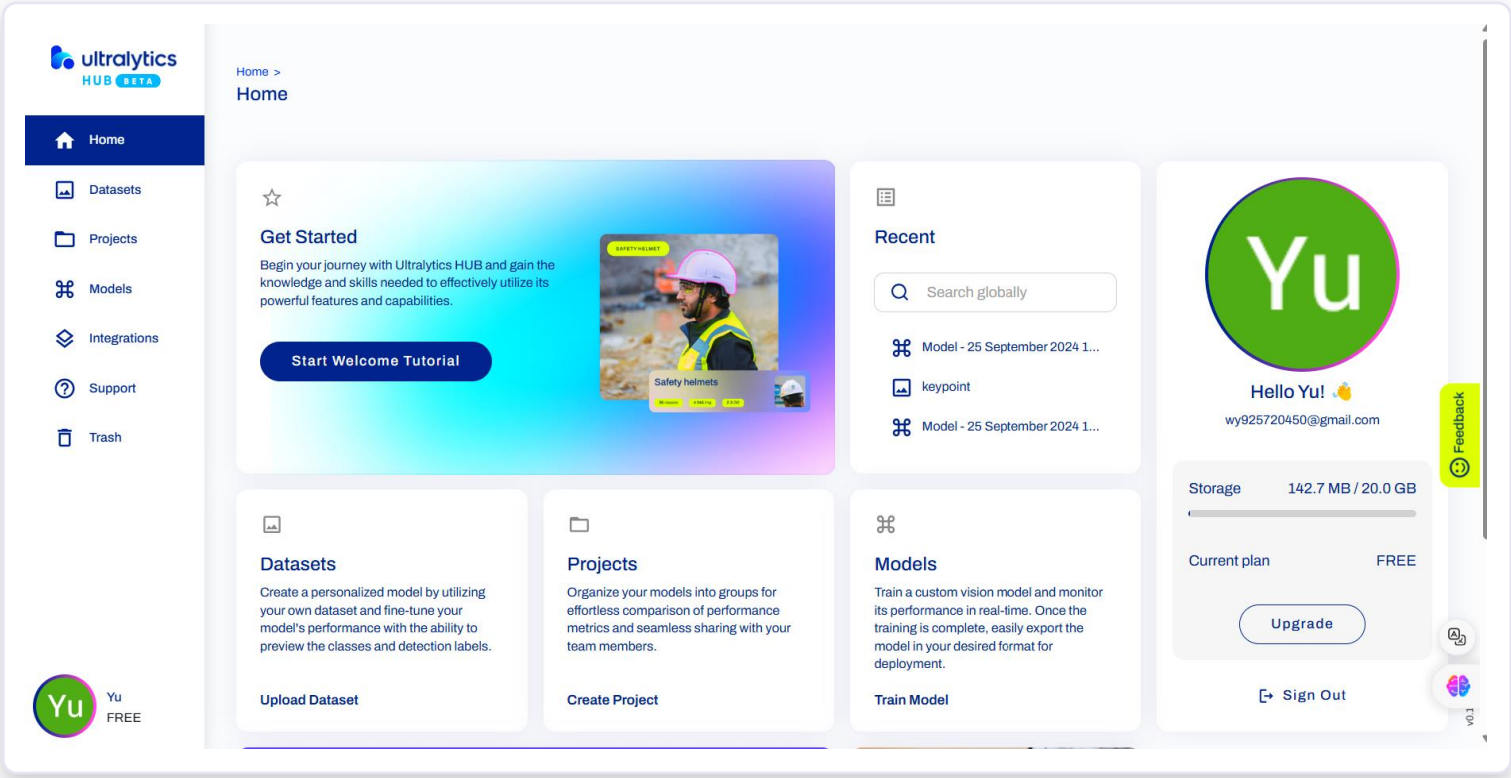
ULTRALYTICS · 1.1

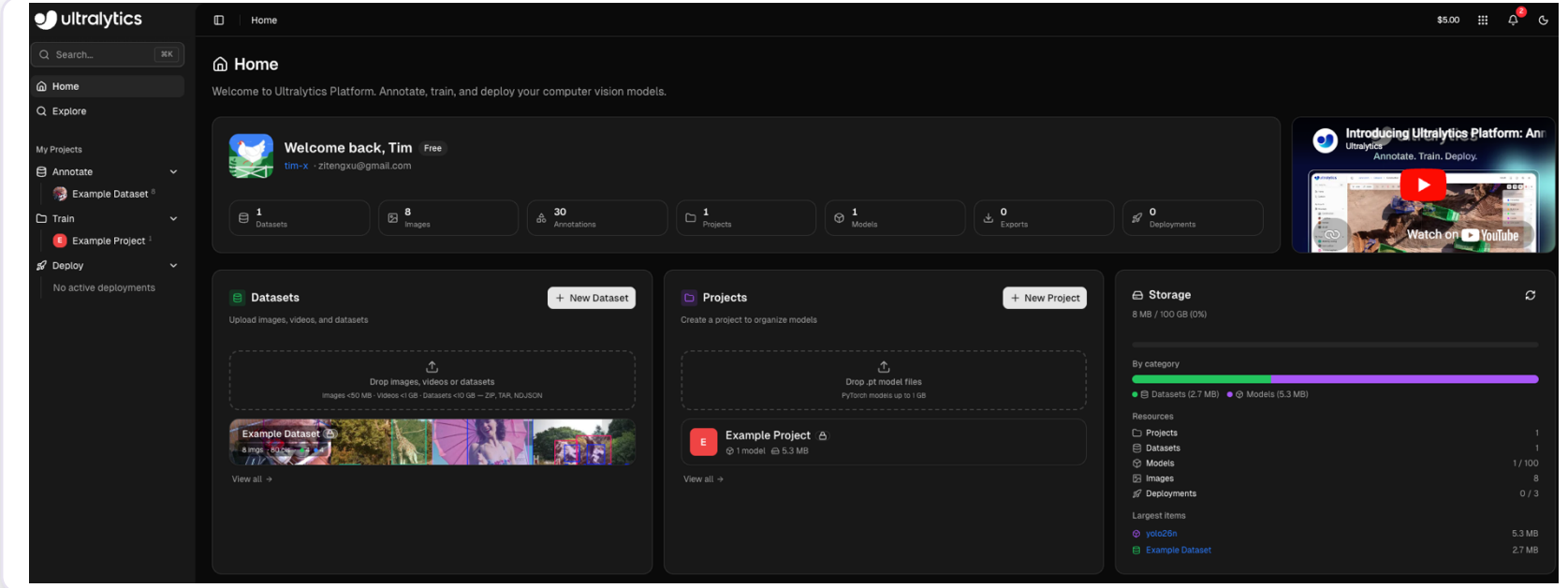
Your Workspace

› After sign-in you land in the dashboard.

› Manage datasets, projects and trained models in one place.

› Everything syncs to the cloud for collaboration.



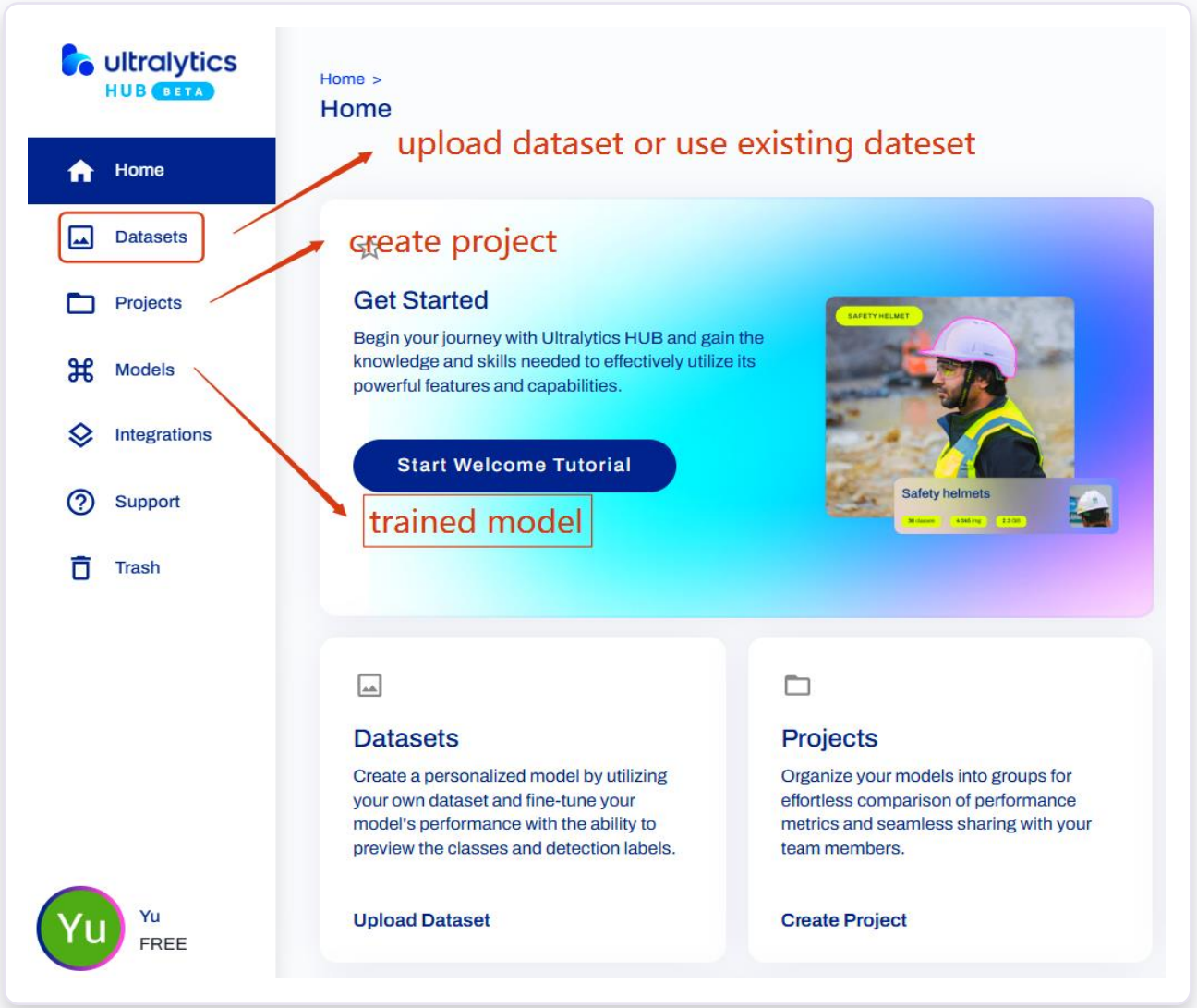


Create a Project

1. **New project**
   Group related datasets and models.

2. **Upload your dataset**
   Bring in the data you labeled in Roboflow.

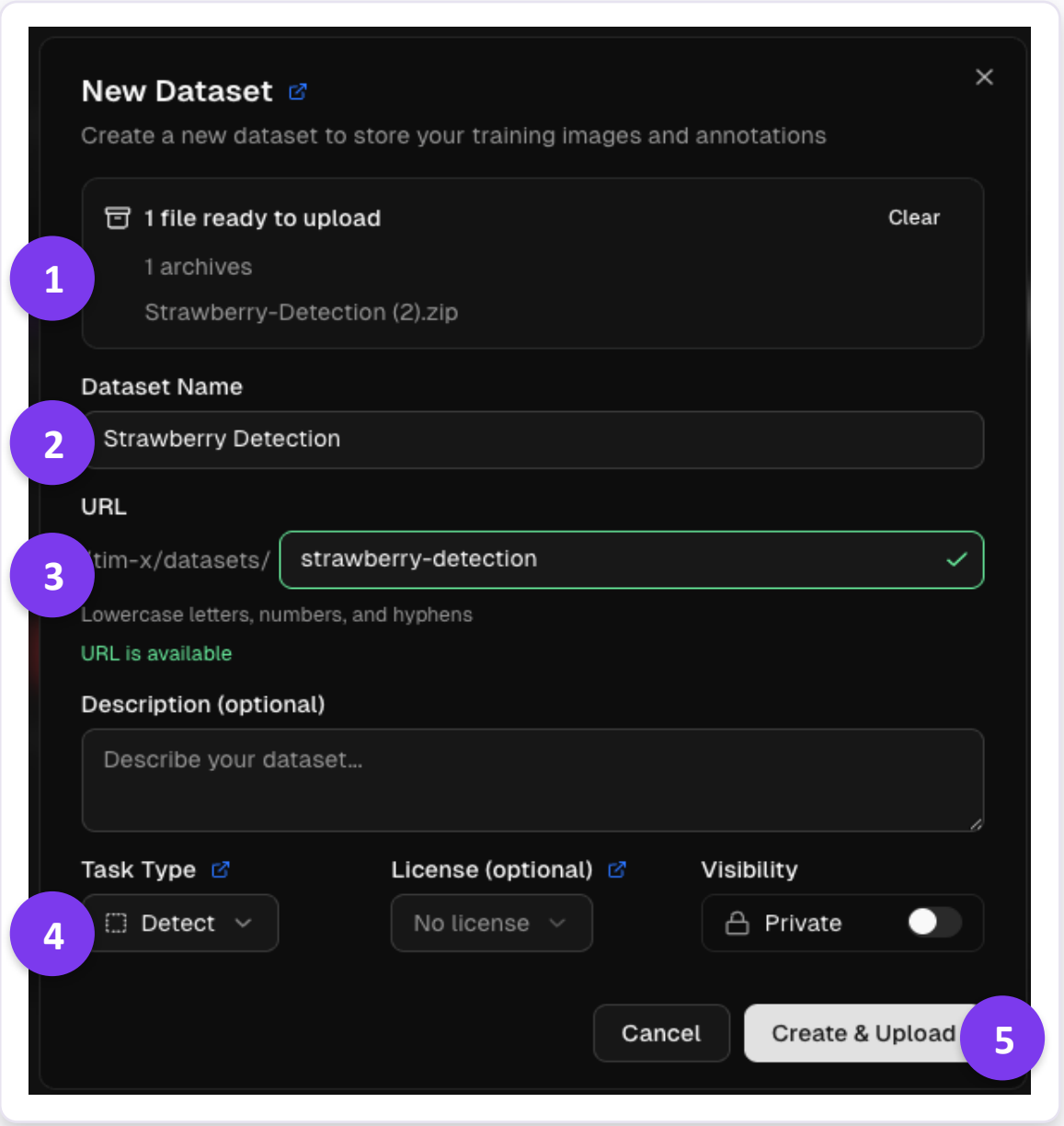
3. **Train inside it**
   Run experiments scoped to the project.



ULTRALYTICS · DATA

Upload a Dataset

1. **Start upload**
   Open the dataset dialog.
2. **Name it**
   Give the dataset a clear name.
3. **Confirm name**
   Verify before continuing.
4. **Match task type**
   Must match your labeled task. Set to private if needed.
5. **Create & upload**
   Push the dataset to the HUB.



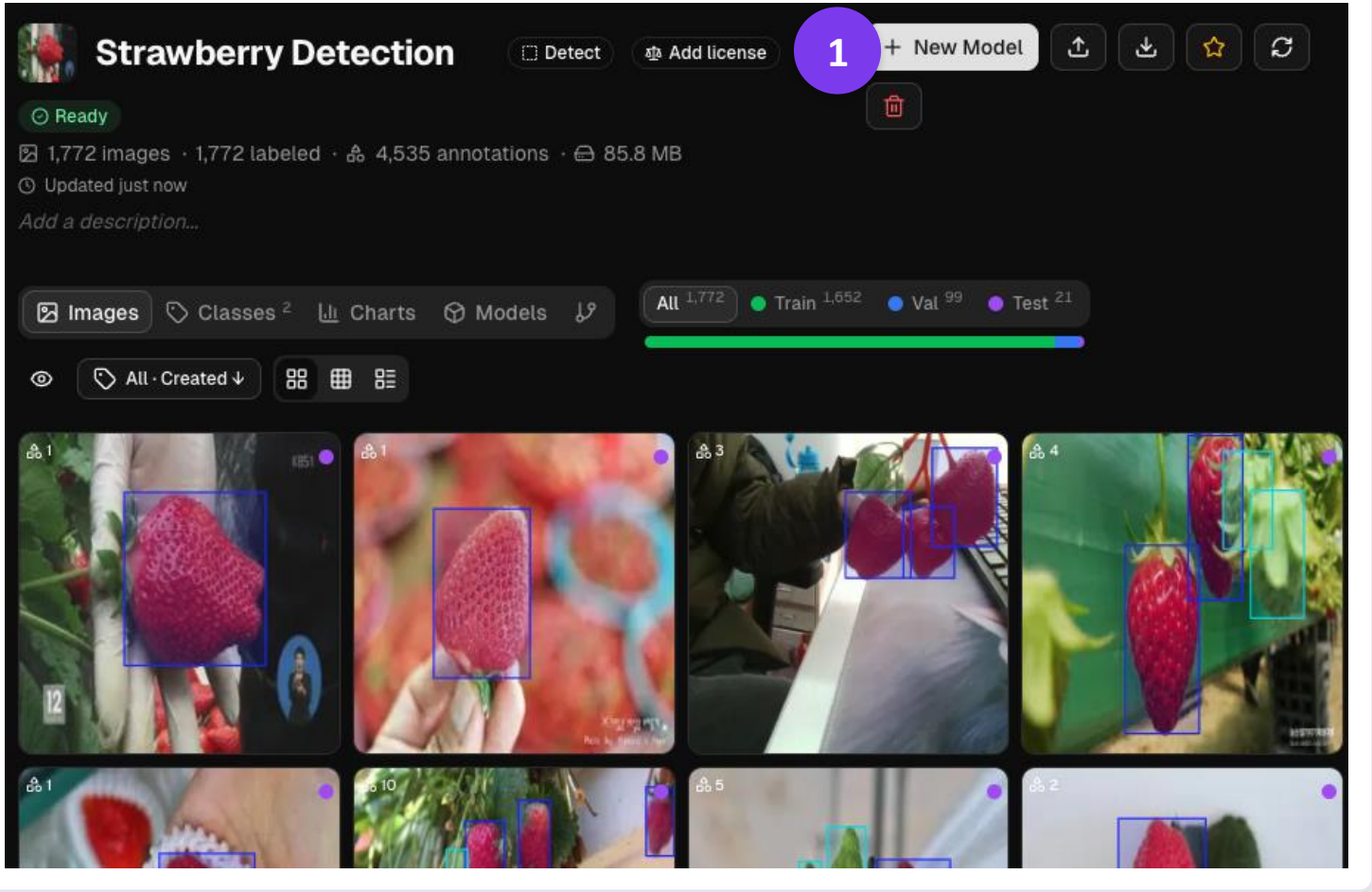
ULTRALYTICS · TRAINING

Start a Training Run

› Open your dataset and choose Train Model.

› Ultralytics walks you through model and compute choices.

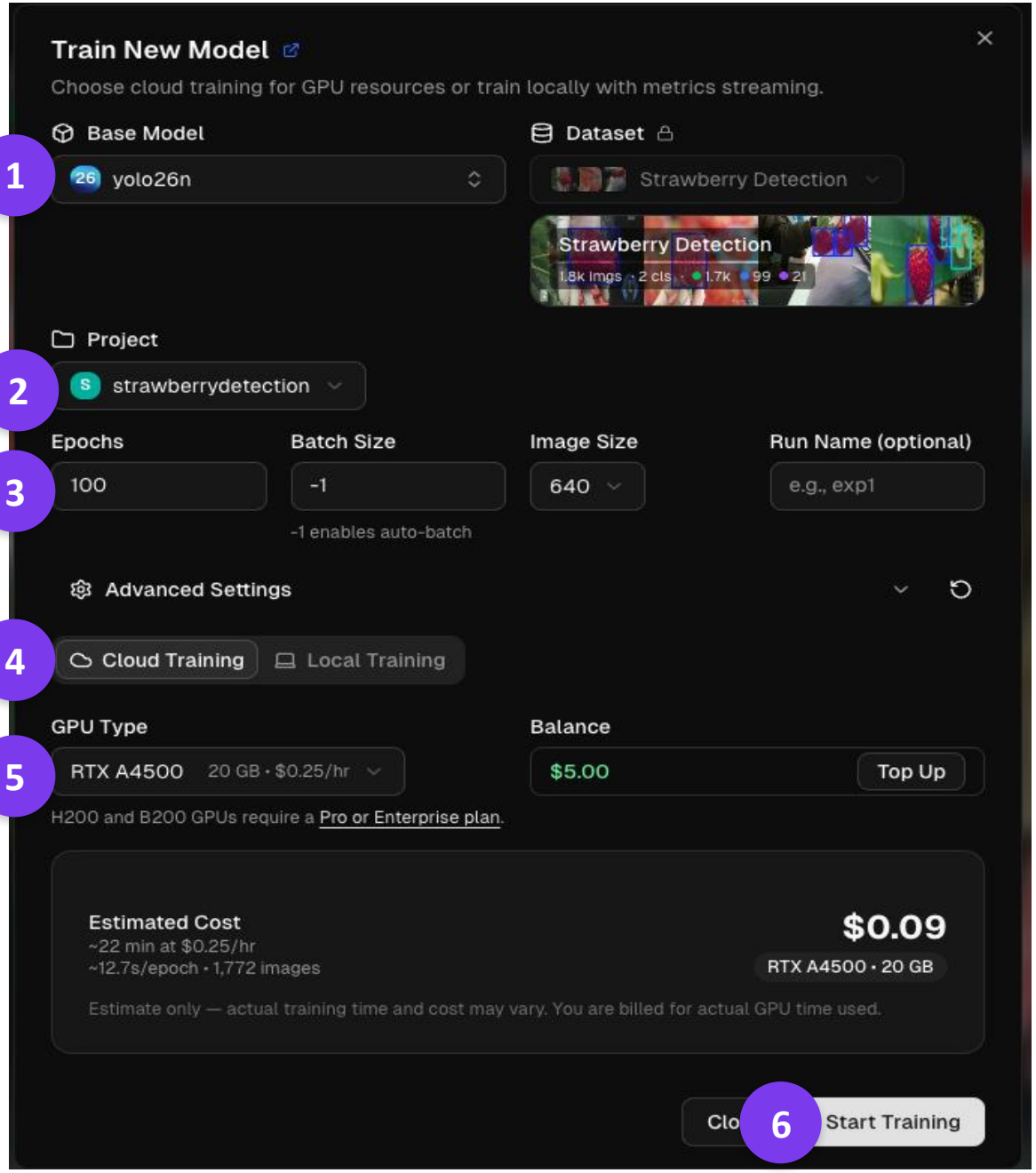
› The next slides break down each setting.



ULTRALYTICS · TRAINING

Configure the Run

1. **Select a base model**
   Pick the YOLO variant to fine-tune.
2. **Choose the project**
   Associate it with your dataset.
3. **Set hyperparameters**
   Epochs, batch size, image size.
4. **Cloud or local**
   Both are free to run.
5. **Pick a GPU**
   No high-end GPU needed for small data.
6. **Begin training**
   Kick off the run.



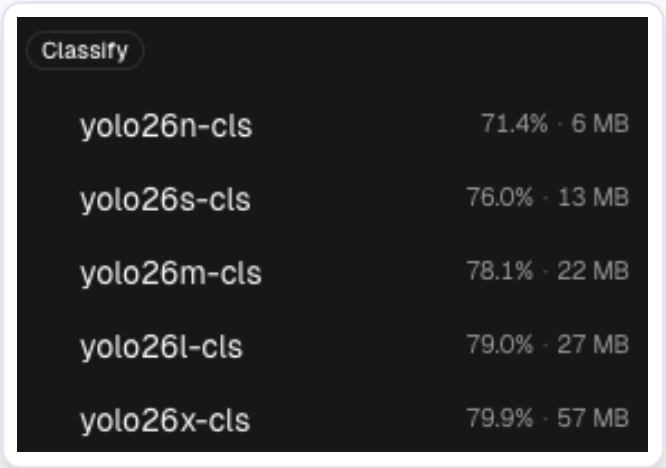
ULTRALYTICS · TRAINING

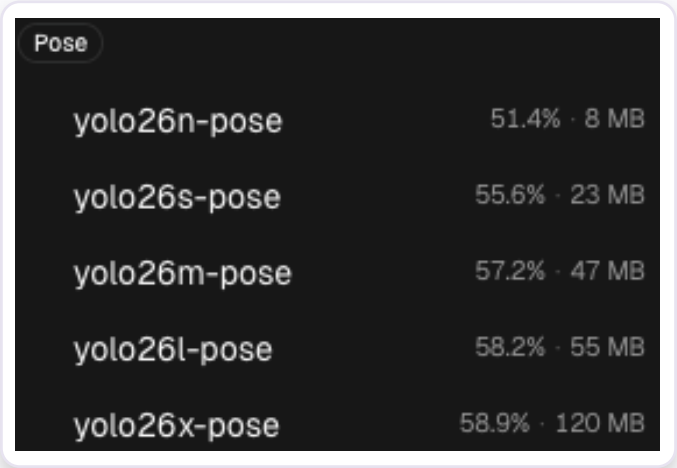
Choosing a Model

› Pick the model family for your task — detection, classification, segmentation or pose.

› Bigger models (YOLO26-X vs 26-N) are more accurate…

› …but cost more training time and slower inference.





Accuracy ↔ Speed. Start small (N/S) to iterate fast, then scale up (L/X) once your pipeline works end-to-end.

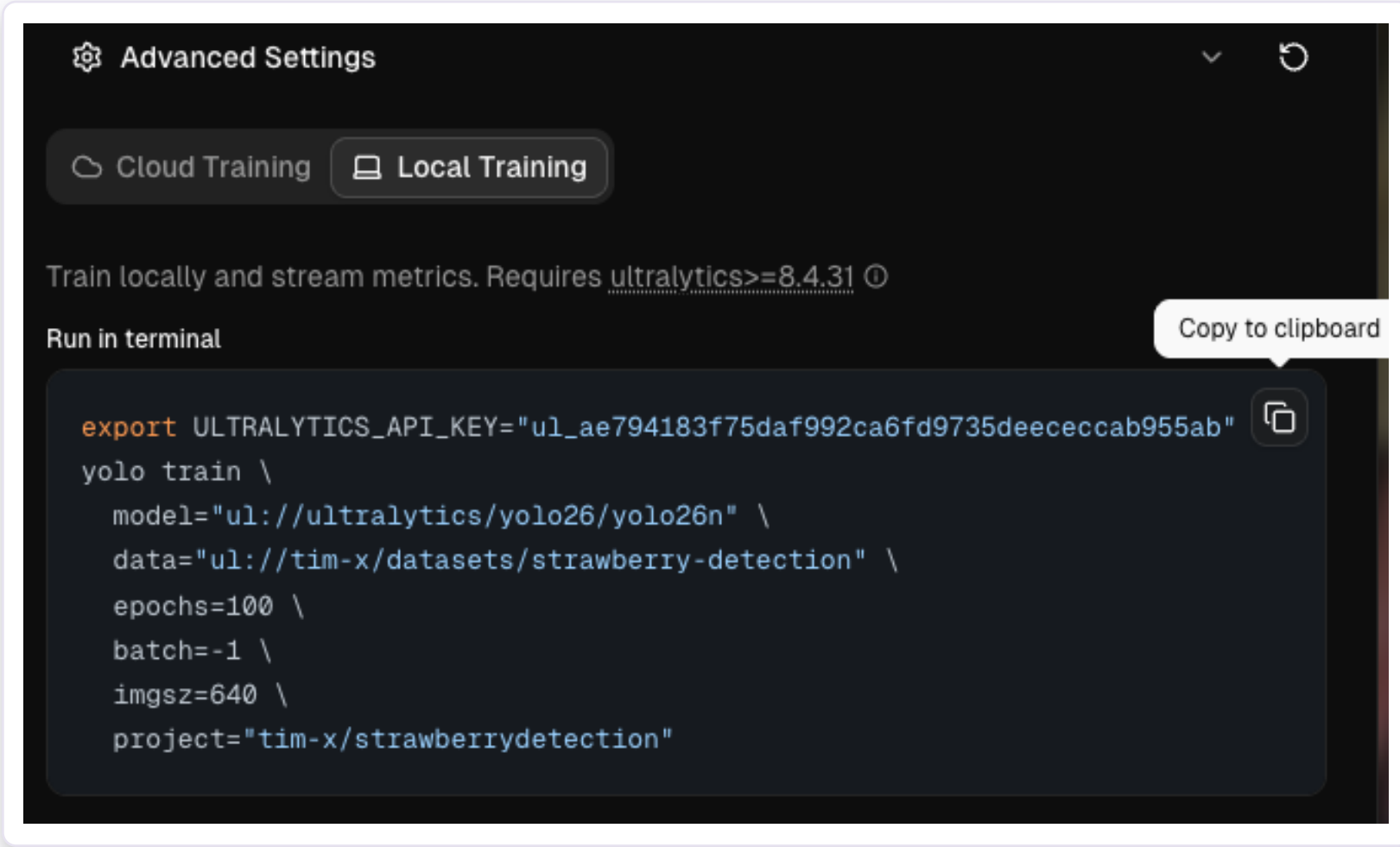
ULTRALYTICS · LOCAL

Train Locally

1. **Open a terminal**
   Work on your own machine.

2. **Install the package**
   pip install ultralytics

3. **Run the snippet**
   Paste the generated command and go.



ULTRALYTICS · COLAB

Train on Google Colab

1. **Upload data to Drive**
   Put the dataset folder in Google Drive.
2. **Open a Colab notebook**
   Set GPU runtime in the browser.
3. **Install & mount**
   Install packages, connect to Drive.
4. **Select model type**
   Choose your YOLO variant.
5. **Train**
   Set data + output paths, then run.

```
%pip install ultralytics  # install
from ultralytics import YOLO, checks, hub

checks()  # checks
import os
import cv2
import random
import shutil
from google.colab import drive
from ultralytics import YOLO
from google.colab.patches import cv2_imshow
drive.mount('/content/drive')
```

```
model = YOLO("yolov8n-cls.pt")#classification model

model.train(
    data="/content/drive/MyDrive/TAMU_Farrow_LabeledData",
    epochs=10,
    imgsz=224,
    batch=32,
    device=0,
    project="/content/drive/MyDrive/YOLO_runs",
    name="Swine_Farrow_Posture"
)
```

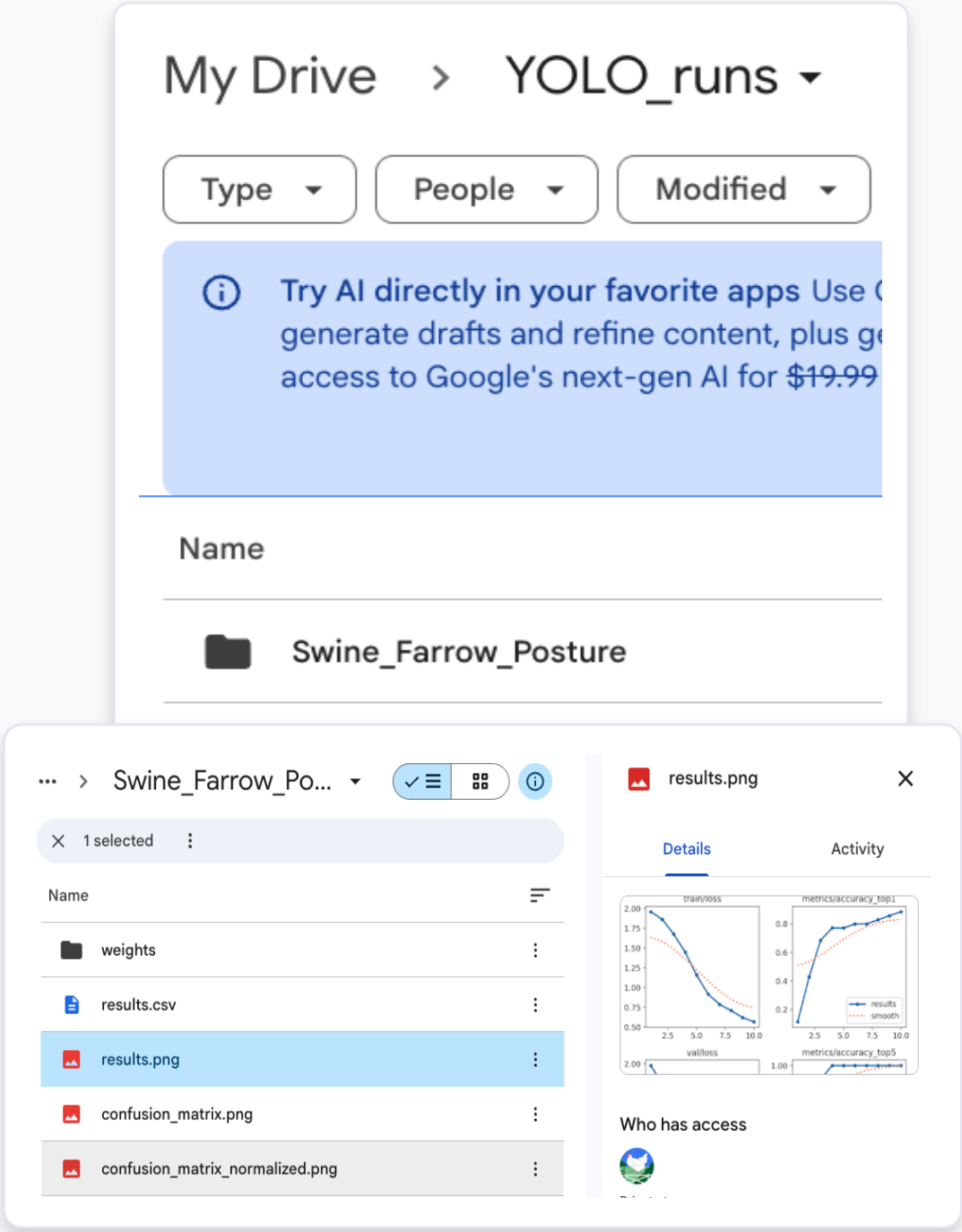
ULTRALYTICS · EVALUATION

Inspect Training Outputs

› Find all artifacts in your designated Google Drive folder.

› Weights, training history and the confusion matrix.

› Use them to validate and compare runs.



ULTRALYTICS · EVALUATION

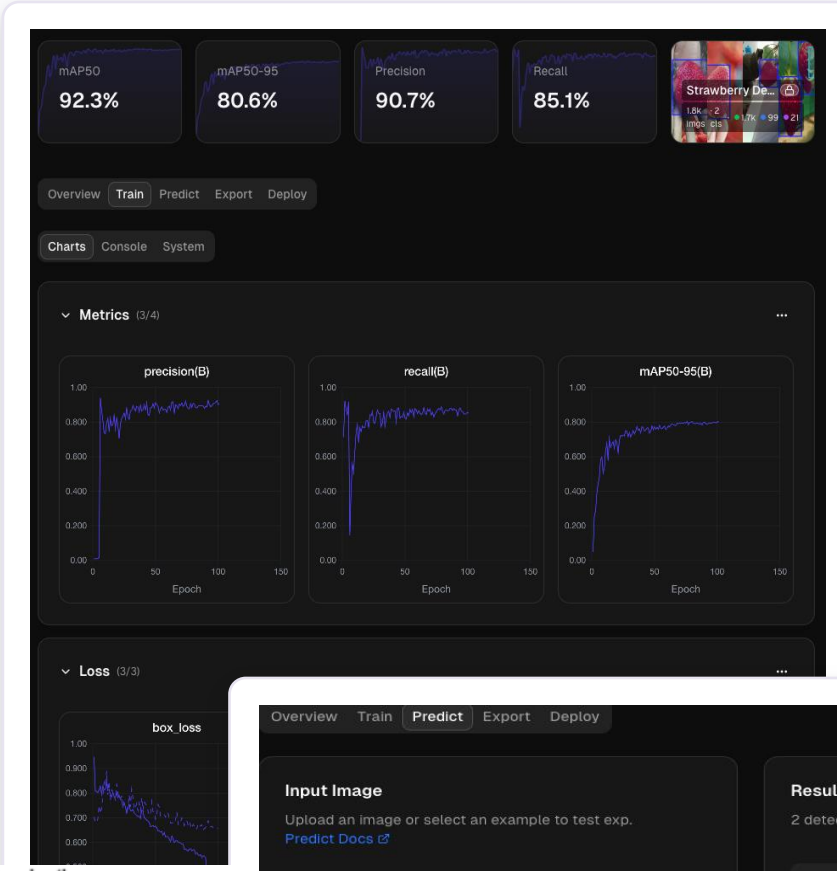
Visualize on the Platform

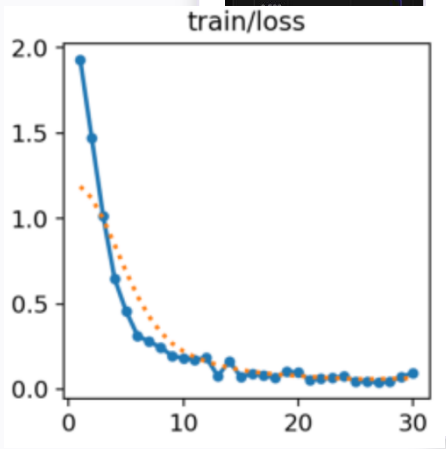
› View training curves and metrics in the HUB.

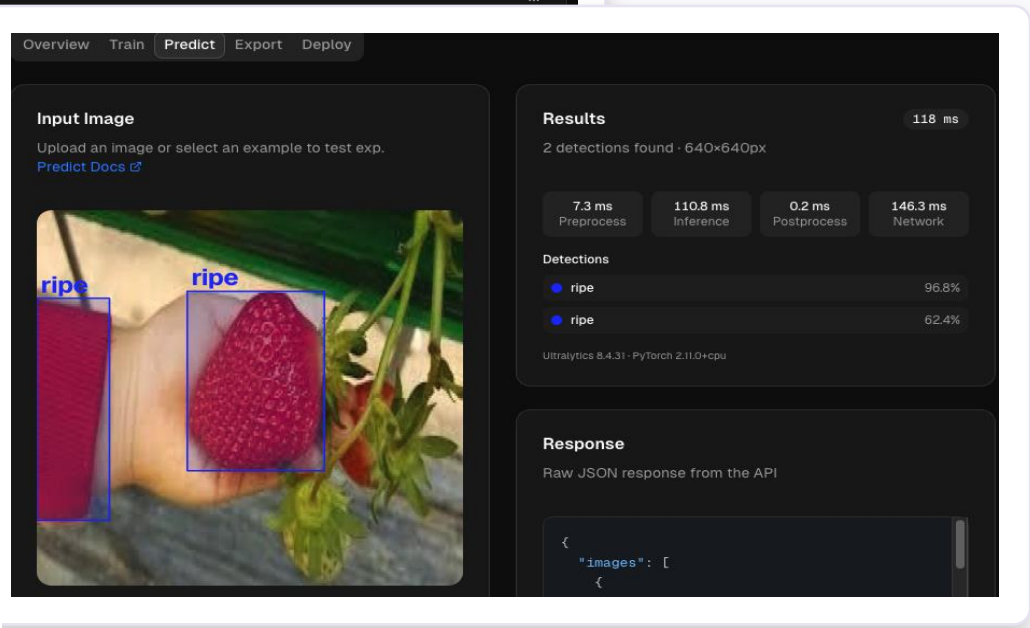
› Preview model predictions on sample images.

› Share results and deploy when you're satisfied.

› Training and validation loss should converge (increase batch size and training epoch).







YOU NOW HAVE THE FULL PIPELINE

Label → Train → Evaluate

1 Roboflow

Label, augment & export datasets for any vision task.

2 Ultralytics

Train YOLO models in the cloud, on Colab, or locally.

3 Evaluate

Review metrics, predictions & weights — then deploy.

Happy building.

Background Removal & Scale Normalization

Model Generalization Challenge

Diverse bedding materials

Different housing structures

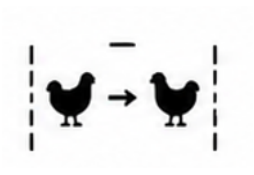
Different floor types

Variation in housing environment and sensor placement can reduce model's performance in a different farm

Existing solutions

1 Data Augmentation

Rotate



Flip

Brightness

Hue

Limited improvement in cross-farm generalization.

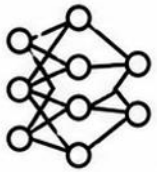
2 More Training Data

More Samples

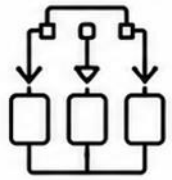
Multiple Farms

Requires larger and more diverse labeled datasets.

3 Larger / Complex Models



Deep Networks

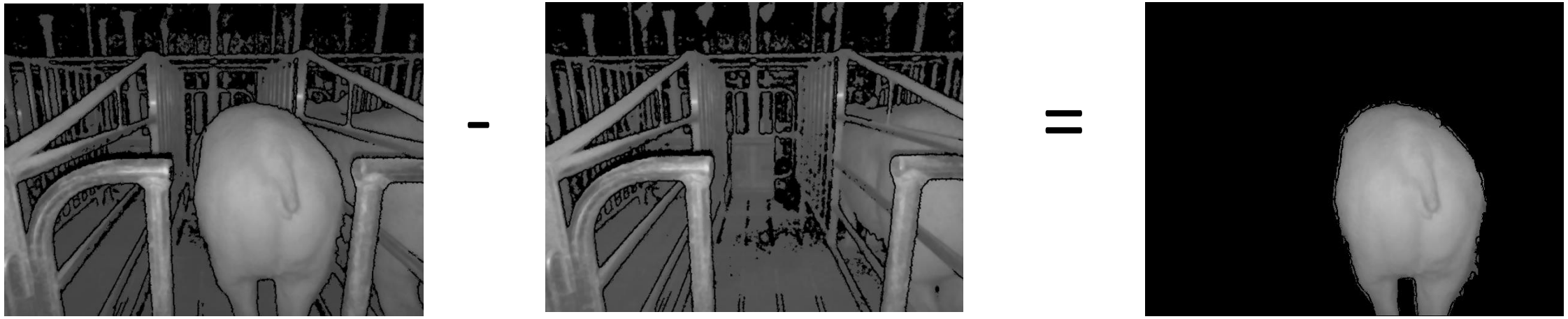


Transformers

Higher computational cost and deployment burden

Rationale

- The reduced performance is due to variation in the background and the sensor placement.
- We can remove background and have the model to focus on the animal itself.
- **For stationary monitoring system setup, background can be removed through frame subtraction (Frame with animal – Frame without animals).**
- **For variation introduced by distance between animal and camera, we can focus on the animal by removing the empty region in the image.**



Example

Dataset Sources

Poultry Farms

Broiler Pens

- **Housing:** Litter-floor system
- **Environment:** Wood shavings, different feeder types, equipment layout and bird density

Layer Pens

- **Housing:** Litter-floor system
- **Environment:** Wood shavings, different feeder types, equipment layout and bird density

Male Broiler Pens

- **Housing:** Groove-floor system
- **Environment:** Different floor texture, bird density and housing layout

Swine Farms

1st Swine Farm

- **Housing:** Swine farrowing stall
- **Environment:** slatted flooring and different stall

2nd Swine Farm

- **Housing:** Sow estrus stall
- **Environment:** Different pen layout, floor/background, and scene structure

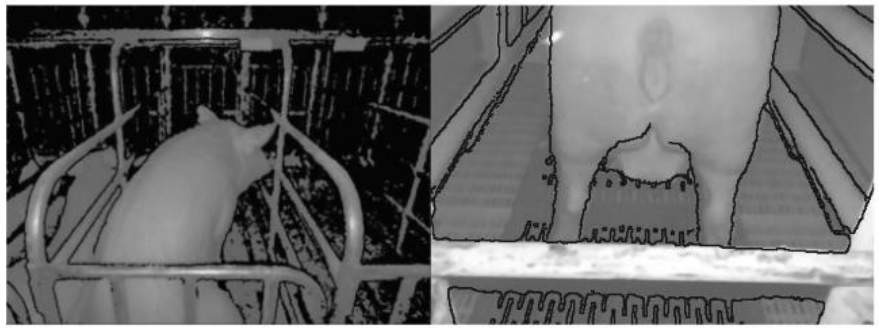
Example

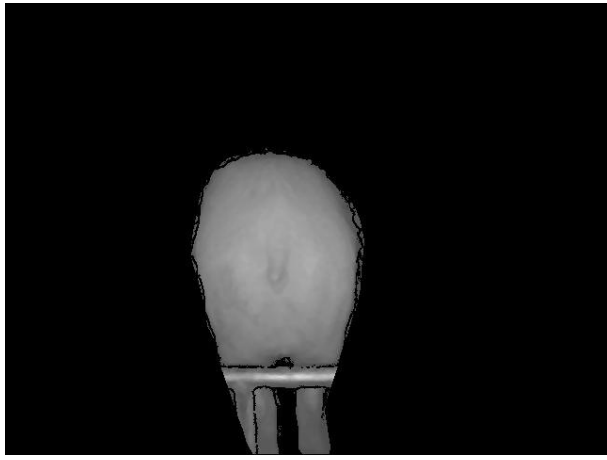
Poultry dataset:

- Imaging device: Intel RealSense D405 & D456 Camera
- Image type: RGB images
- Image size: 848 × 480 pixels
- Camera angle: Vertical top-down view
- Camera height: Layer Pens: ~2.2 m, Broiler Pens: ~1.9 m, Male Broiler Pens: ~1.8 m.

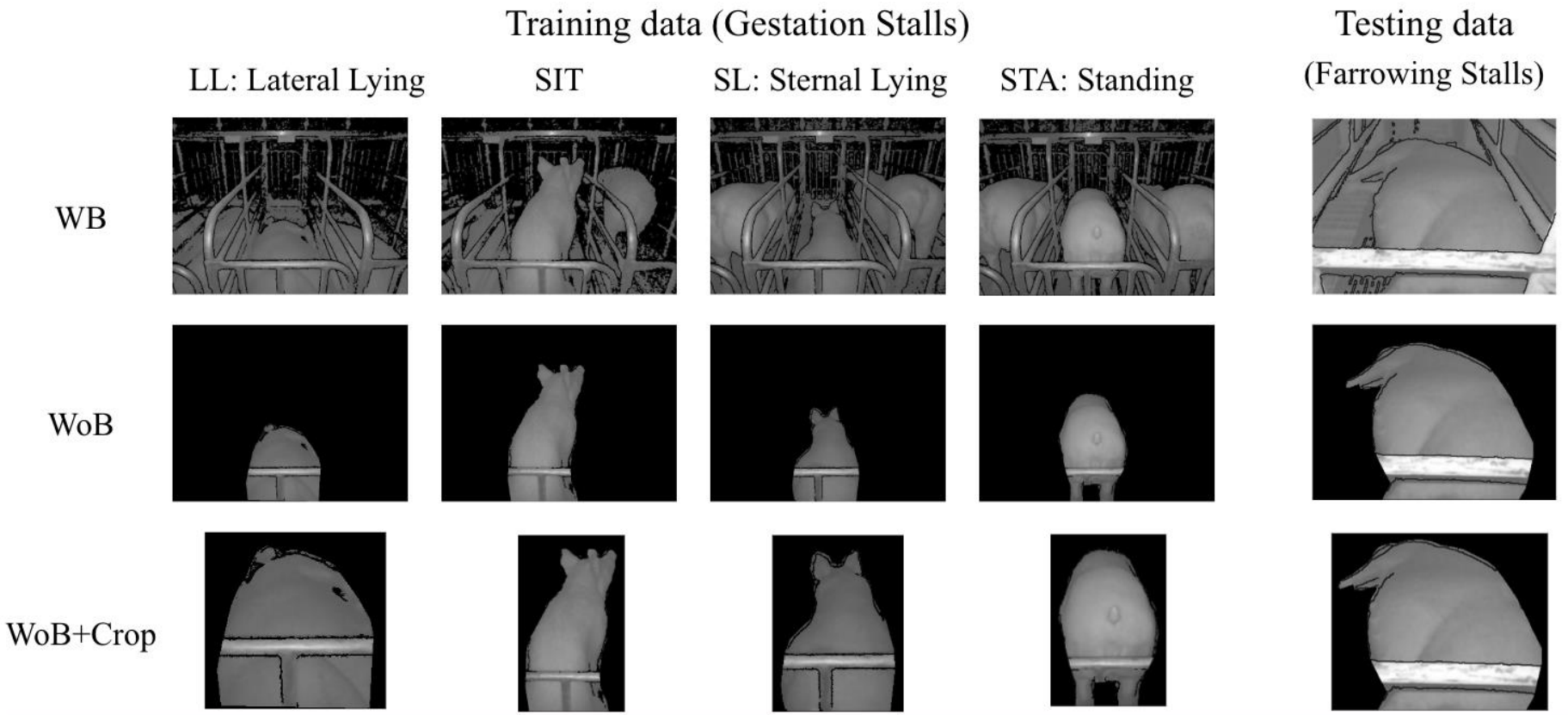
Swine dataset:

- Imaging device: Intel RealSense L515 Camera
- Image type: IR-Depth fused images
- Image size: 640 × 480 pixels
- Camera angle: 1st swine farm : ~5° downward, 2nd farm : ~30° downward





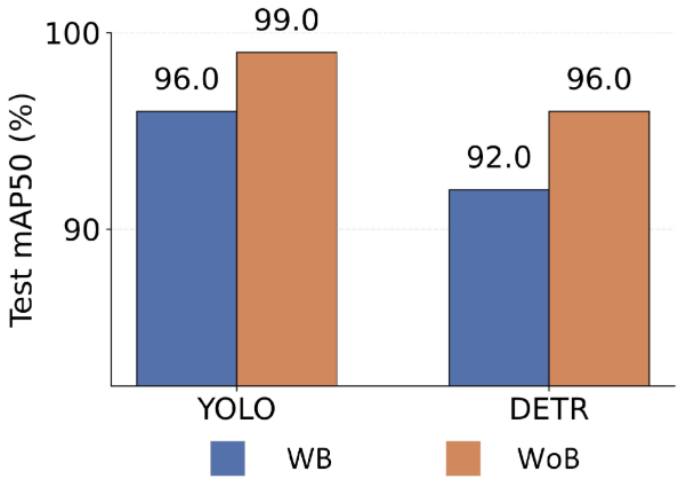
Example



Cropping was applied only to the swine dataset after background removal to achieve scale normalization.

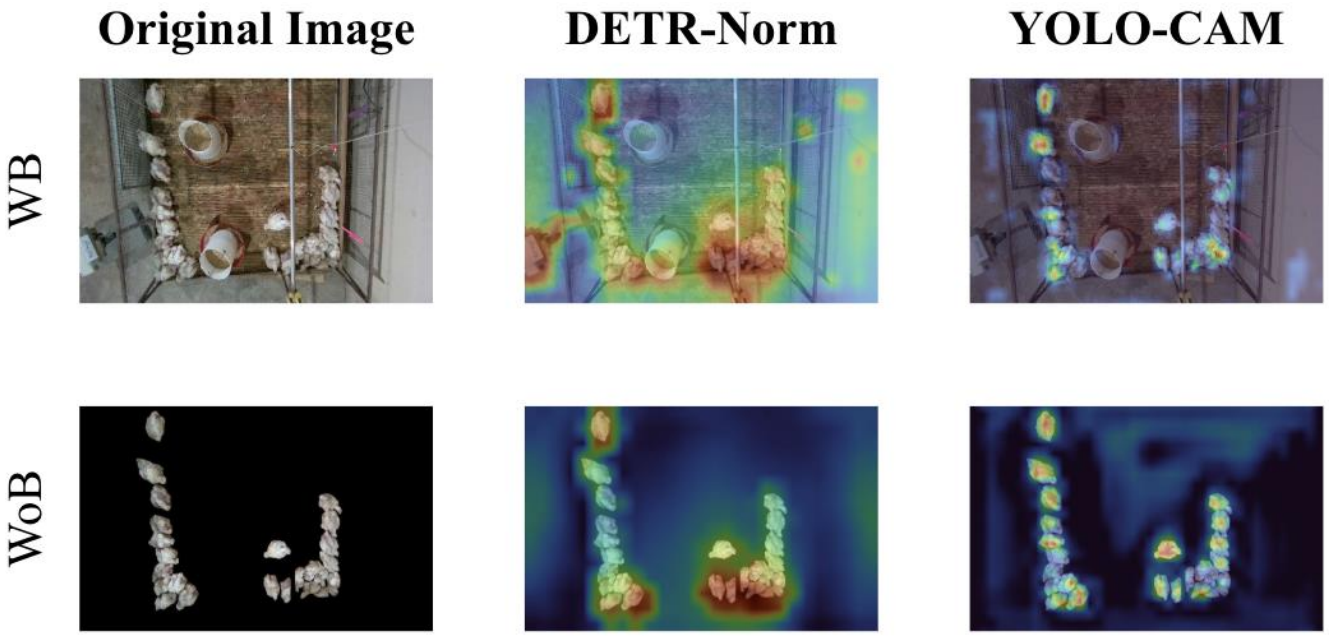
Background Removal & Scale Normalization

Example



Cross-Farm Broilers Detection

Model	Input	Precision	Recall	F1	mAP@50	mAP@50-95
YOLO	WB	0.94	0.90	0.92	0.96	0.72
	WoB	**0.98**	**0.95**	**0.97**	**0.99**	**0.77**
DETR	WB	0.72	0.93	0.82	0.92	0.60
	WoB	0.87	0.96	0.91	0.96	0.65

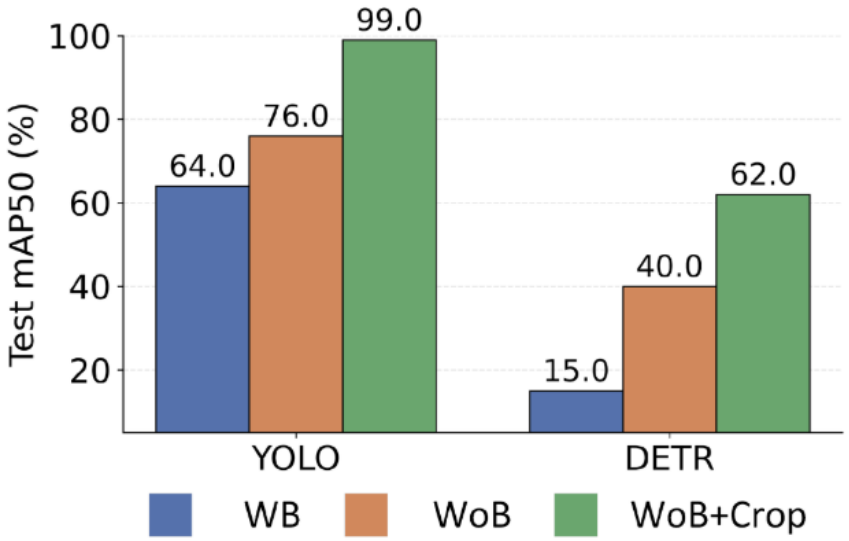


(1) DETR token responses became more concentrated on broiler regions after background removal.

(2) YOLO CAM highlighted more discriminative broiler regions after background removal.

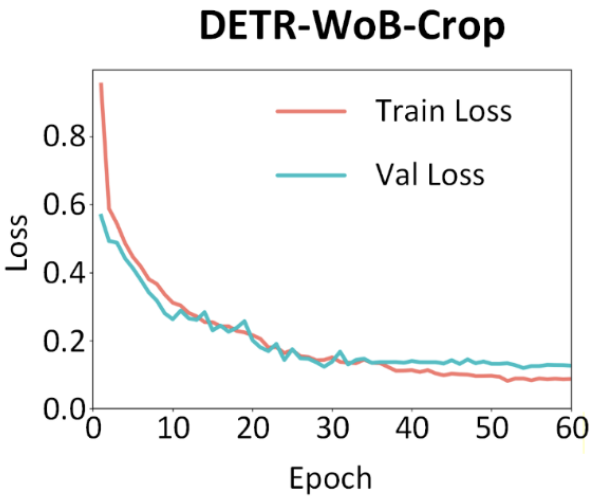
Example

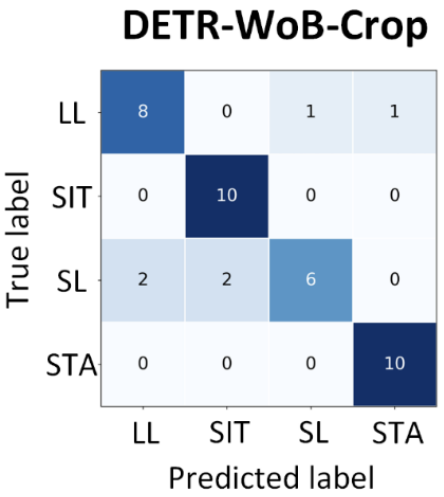
Cross-Farm Swine Posture Classification



Model	Input	Precision	Recall	F1	mAP@50	mAP@50-95
YOLO	WB	0.74	0.37	0.39	0.64	0.37
	WoB	0.55	0.90	0.59	0.76	0.57
	WoB+Crop	**0.96**	**0.97**	**0.96**	**0.99**	**0.94**
DETR	WB	0.12	0.05	0.07	0.15	0.12
	WoB	0.42	0.50	0.45	0.40	0.38
	WoB+Crop	0.57	0.55	0.56	0.62	0.41

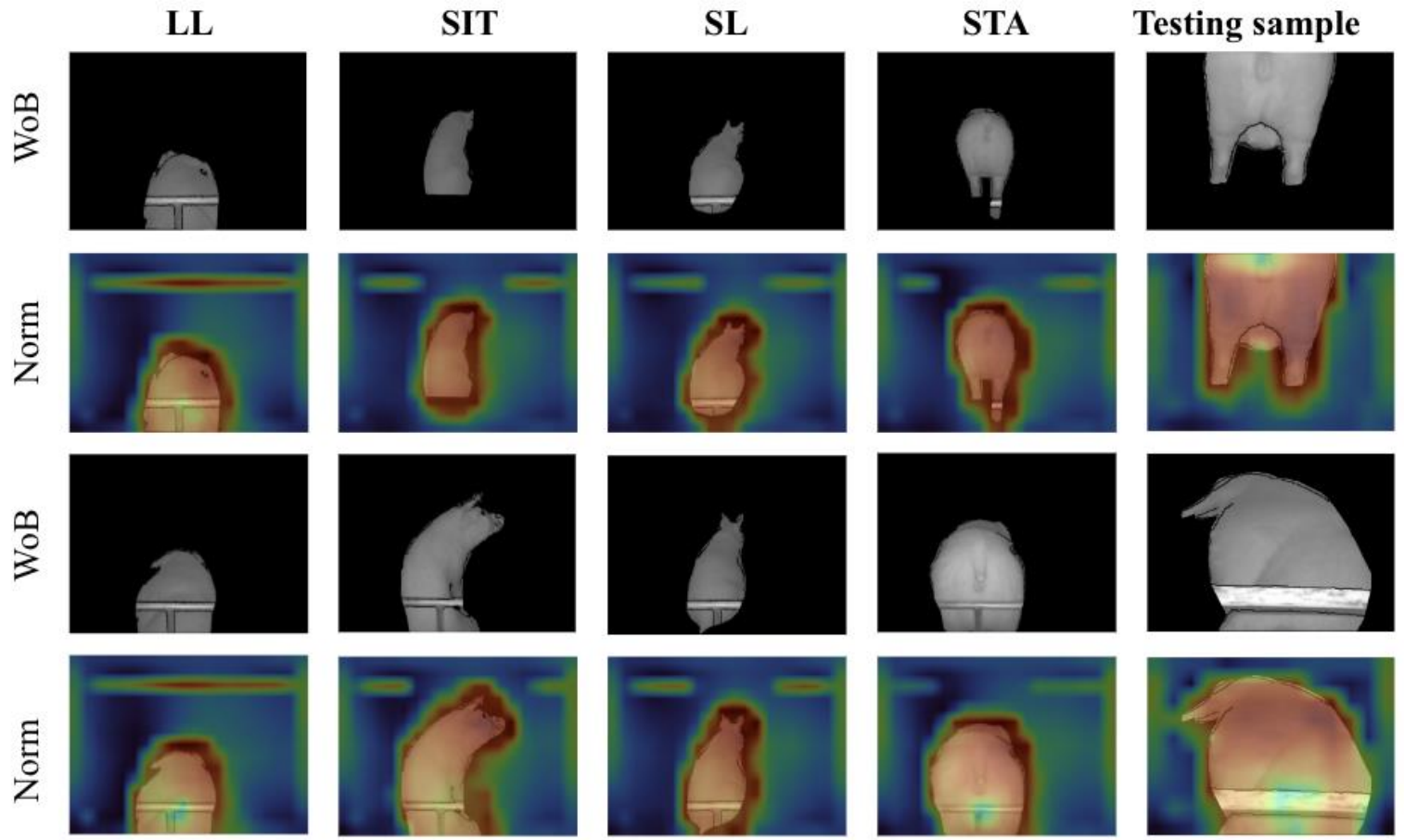
Background removal and scale normalization (Crop) substantially improved cross-farm swine detection for both YOLO and DETR.





Example

DETR-Norm

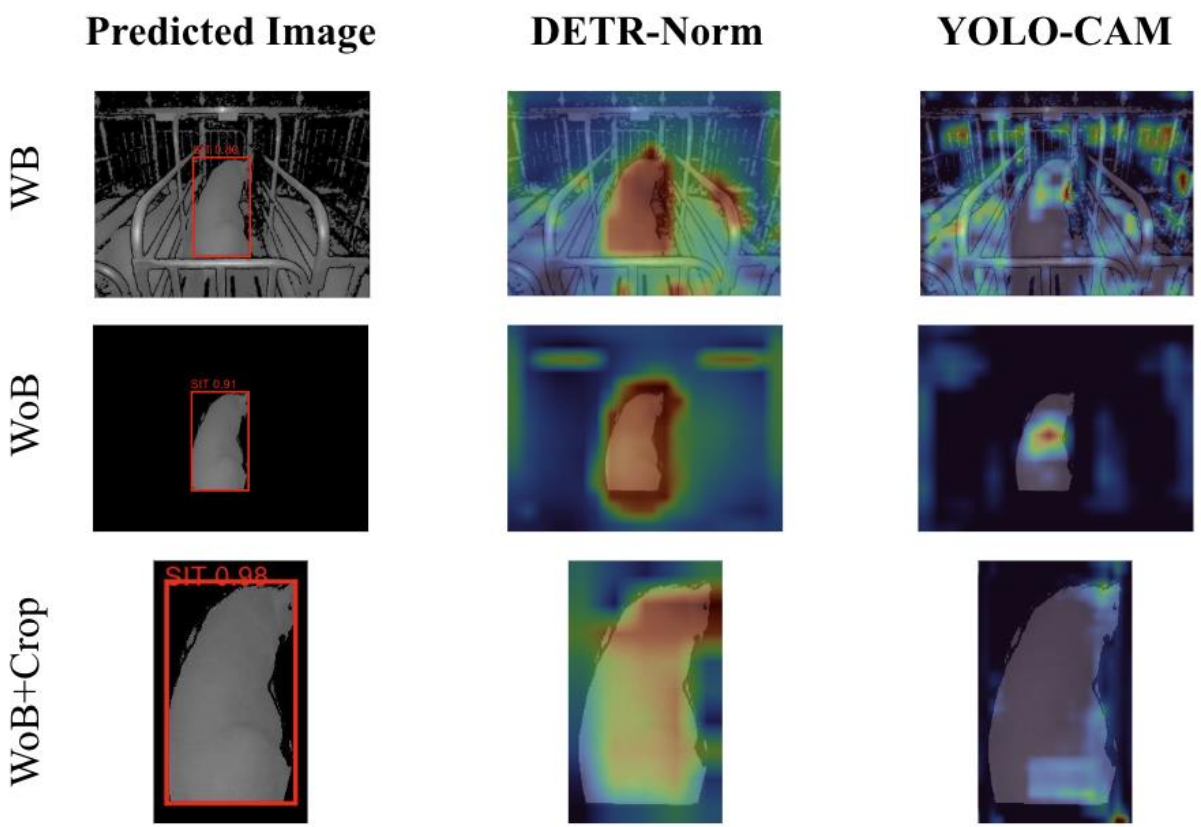


(1) Similar residual responses were observed across different swine postures.

(2) These residual responses were observed only in the training images but disappeared in the testing images.

(3) This suggests that DETR developed a shortcut based on training-specific spatial cues, which failed to generalize to the testing set.

Example



(1) Both models used background structures, particularly the stalls, in WB images.

(2) DETR and YOLO responses became more concentrated on the swine after background removal (WoB and WoB+Crop).

(3) However, DETR still exhibited residual responses at fixed image locations (WoB).

Example

1. Background removal improved cross-farm detection performance and encouraged both models to focus more on animal-specific visual cues.

2. For more complicated task (i.e., object classification), when camera angle and distance differs substantially, additional scale normalization further improved DETR generalization by reducing reliance on training-specific spatial cues.

3. Background removal combined with scale normalization provides a practical strategy for developing more robust deep learning models for livestock detection across different farming environments.