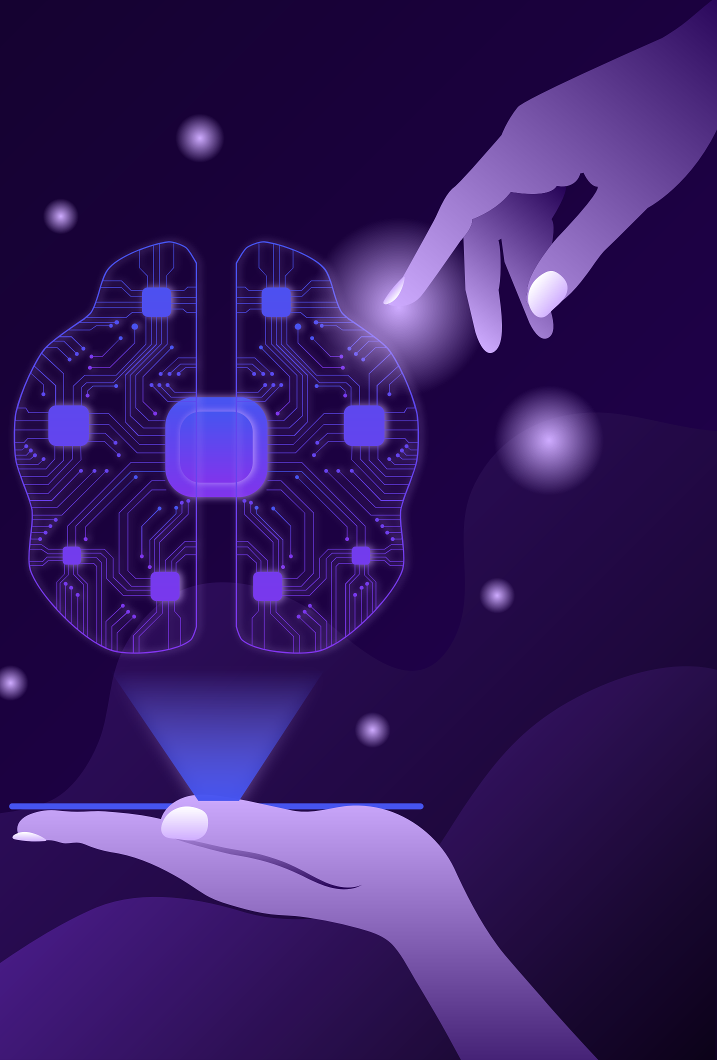
Object Detection 101



Gene, Jose, Ishrak,
and Rachel

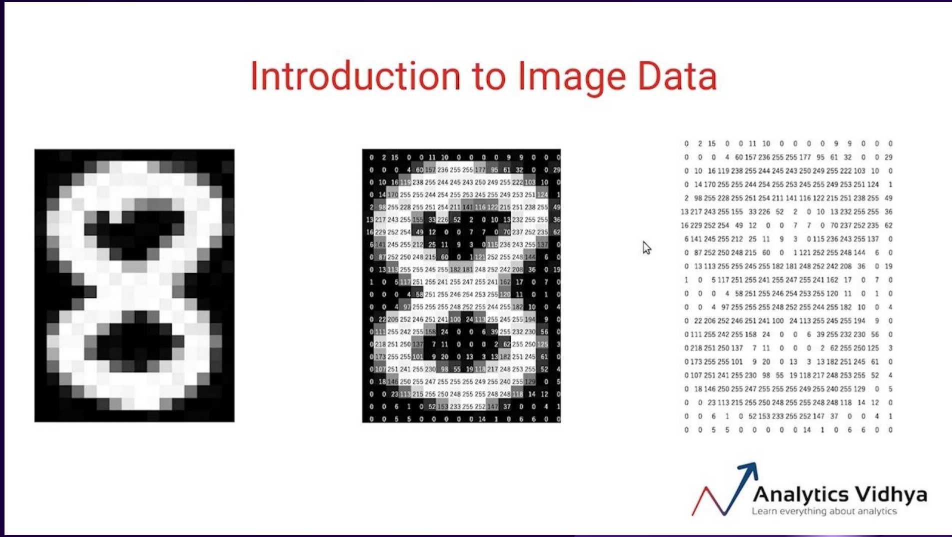
Welcome!

Today, we will show you guys how to use one key AI tools:

- YOLOv8
- YOLOv8-obb

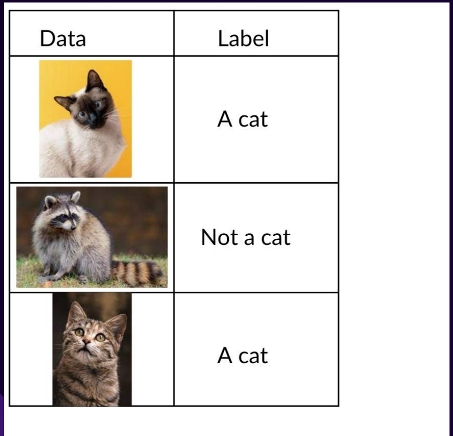
Computer Vision???

- Teaching computers how to see
- Images are not what our eyes see…
- Images are turned into numbers (RGB value)
- Computer reads the number and spots patterns in them.

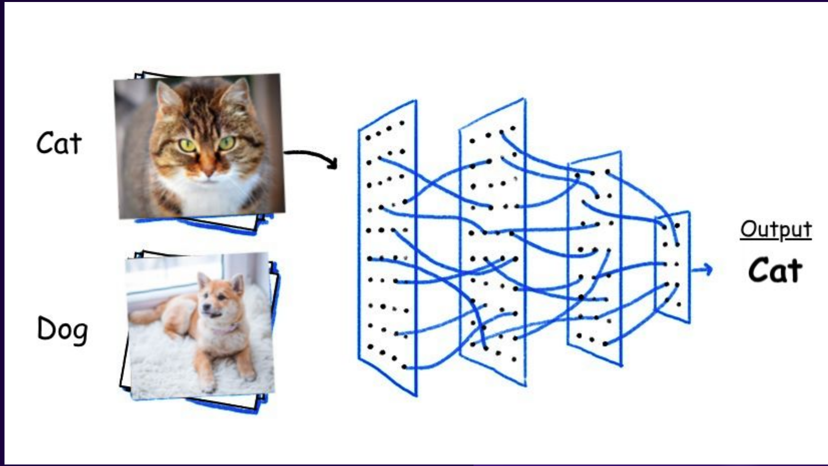


The process

- Two things are given to a computer, an image and a label. The label tells the computer what is in the picture.
- The machine will convert the image to numbers and do some calculations. Based on the result of these calculations, it will guess what the image contains
- If the machine is correct, it is encouraged to keep on doing the same calculations
- If it’s wrong, a slight correction is done.
- This is repeated a loooooot of times, until the machine is corrected and makes accurate guesses



Data	Label
	A cat
	Not a cat
	A cat



For example,

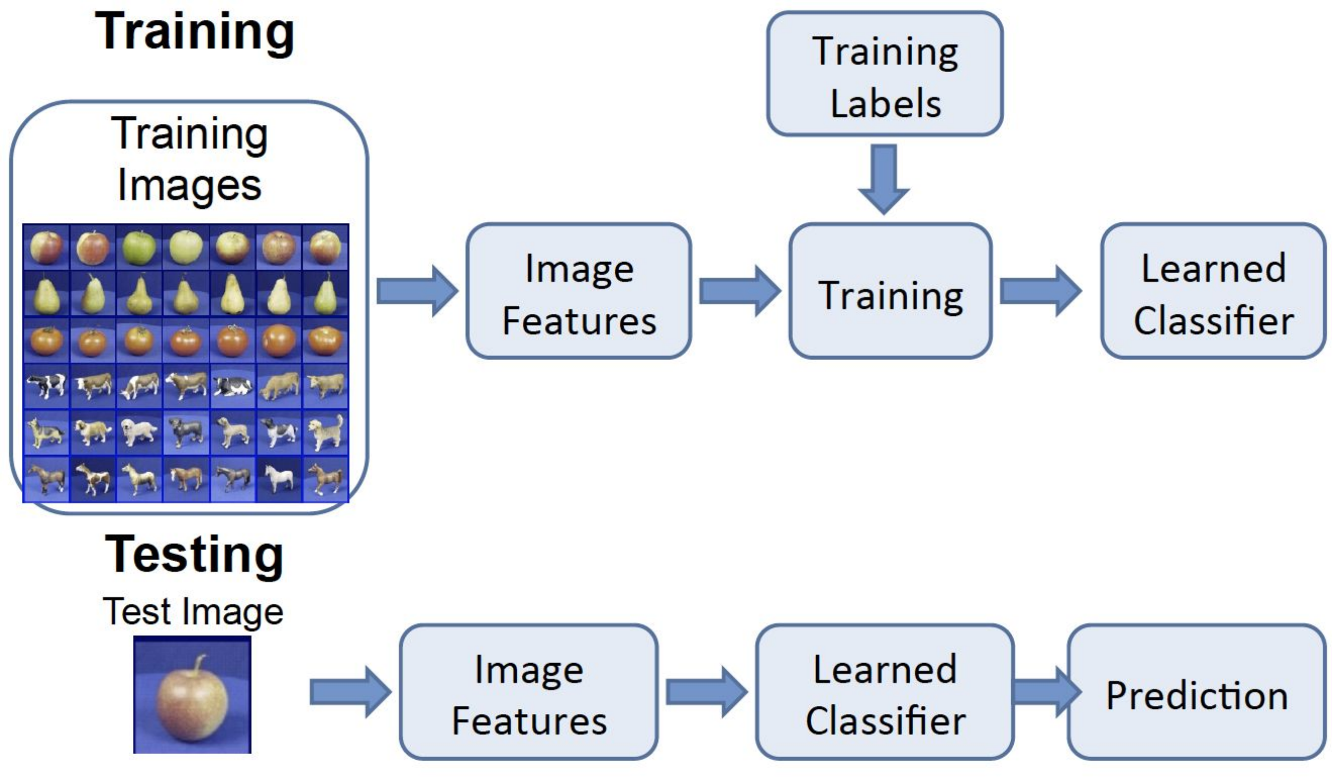


Image features: converting the image to numbers that the computer understands.

Training labels: the indication of the objects in the image

Training: The machine compares prediction with real label and adjusts

Learned classifier: a model that knows how to calculate!

Is this a dog?

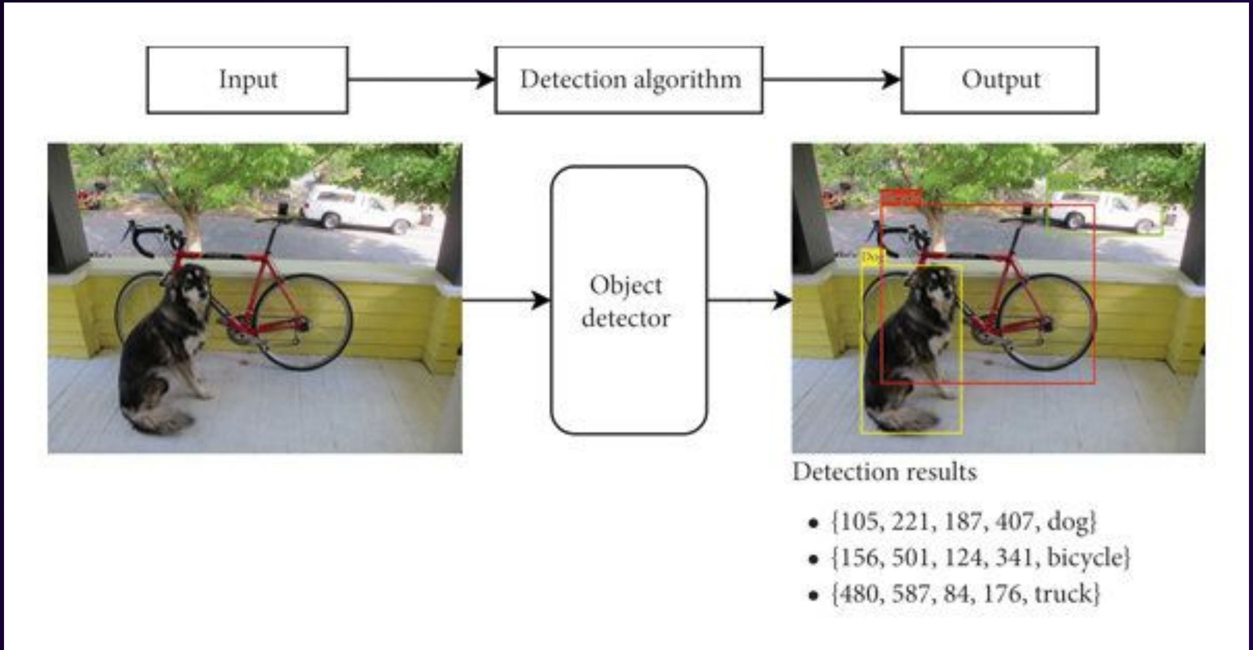
What is there in image and where?

Image Classification

Object Detection

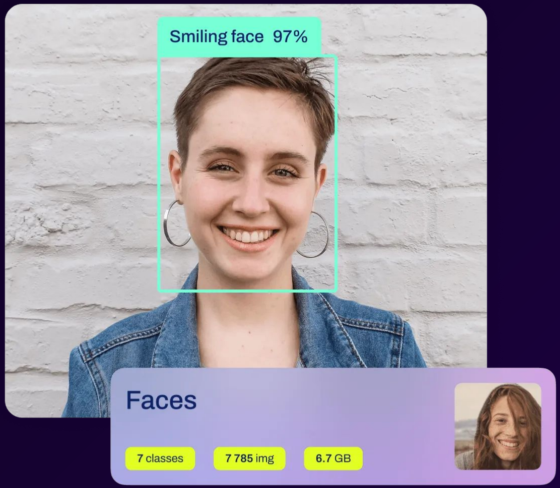
Object Detection…

The training works pretty much the same, but the algorithm has a few more things to calculate:



YOLOv8??

- VERY famous machine learning algorithm
- Intended to be used in REAL-TIME by doing the important calculations in a single step
- Supported by standard hardware (nothing too fancy needed)



How does it work?

NOTE: this is an overly simplified explanation.

The picture is divided into cells. For each cell, YOLOv8 guesses where objects are present (bounding boxes) and also guesses what each object is (class probabilities)

Then, YOLOv8 deletes the overlapping bounding boxes and selects only the most likely one for each object in the picture.

Let us get our hands dirty with Yolov8!😎😎

STEP-BY-STEP SET up Environment

1. Run:

```
git clone https://github.com/HKUGenAI/CV_WorkShop
```

2. Setup Python environment:

```
conda create --name detectionworkshop python=3.12
conda activate detectionworkshop
```

3. In requirements_yolo.txt:

```
pip install -r requirements_yolo
```

4. Our code only works on torch==2.2.1, so go to https://pytorch.org/get-started/previous-versions/ and install v2.2.1 of torch, torchvision, torchaudio

What is this formula?

$$W = mg$$

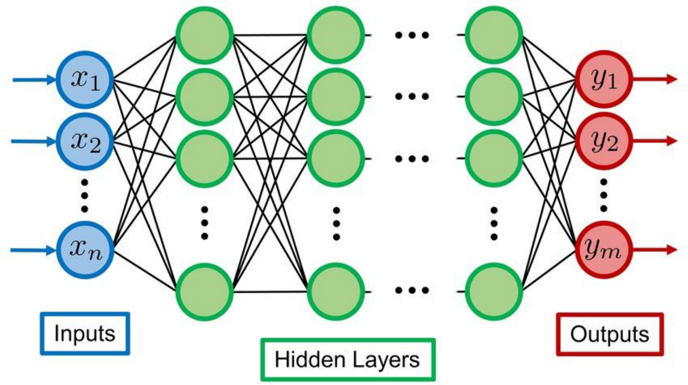
$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2$$

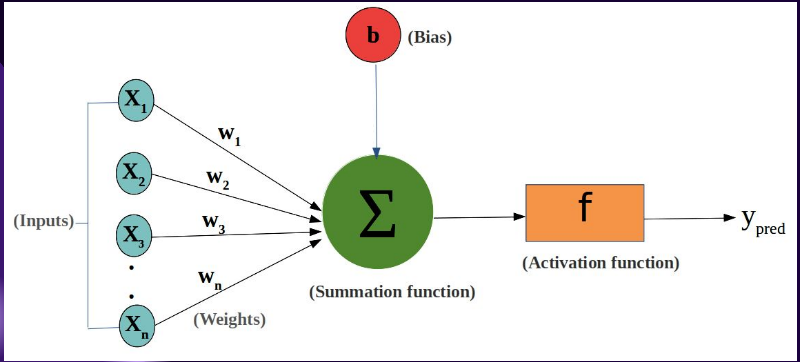
$$\begin{gathered}
\mathcal{L}_{SM} = -\tfrac{1}{2}\partial_\nu g^a_\mu \partial_\nu g^a_\mu - g_s f^{abc}\partial_\mu g^a_\nu g^b_\mu g^c_\nu - \tfrac{1}{4}g_s^2 f^{abc} f^{ade} g^b_\mu g^c_\nu g^d_\mu g^e_\nu - \partial_\nu W^+_\mu \partial_\nu W^-_\mu - \\
M^2 W^+_\mu W^-_\mu - \tfrac{1}{2}\partial_\nu Z^0_\mu \partial_\nu Z^0_\mu - \tfrac{1}{2c_w^2} M^2 Z^0_\mu Z^0_\mu - \tfrac{1}{2}\partial_\mu A_\nu \partial_\mu A_\nu - igc_w(\partial_\nu Z^0_\mu (W^+_\mu W^-_\nu - \\
W^+_\nu W^-_\mu) - Z^0_\nu (W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + Z^0_\mu (W^+_\nu \partial_\nu W^-_\mu - W^-_\nu \partial_\nu W^+_\mu)) - \\
igs_w(\partial_\nu A_\mu (W^+_\mu W^-_\nu - W^+_\nu W^-_\mu) - A_\nu (W^+_\mu \partial_\nu W^-_\mu - W^-_\mu \partial_\nu W^+_\mu) + A_\mu (W^+_\nu \partial_\nu W^-_\mu - \\
W^-_\nu \partial_\nu W^+_\mu)) - \tfrac{1}{2}g^2 W^+_\mu W^-_\mu W^+_\nu W^-_\nu + \tfrac{1}{2}g^2 W^+_\mu W^-_\nu W^+_\mu W^-_\nu + g^2 c_w^2 (Z^0_\mu W^+_\mu Z^0_\nu W^-_\nu - \\
Z^0_\mu Z^0_\mu W^+_\nu W^-_\nu) + g^2 s_w^2 (A_\mu W^+_\mu A_\nu W^-_\nu - A_\mu A_\mu W^+_\nu W^-_\nu) + g^2 s_w c_w (A_\mu Z^0_\nu (W^+_\mu W^-_\nu - \\
W^+_\nu W^-_\mu) - 2A_\mu Z^0_\mu W^+_\nu W^-_\nu) - \tfrac{1}{2}\partial_\mu H \partial_\mu H - 2M^2 \alpha_h H^2 - \partial_\mu \phi^+ \partial_\mu \phi^- - \tfrac{1}{2}\partial_\mu \phi^0 \partial_\mu \phi^0 - \\
\beta_h \left(\tfrac{2M^2}{g^2} + \tfrac{2M}{g}H + \tfrac{1}{2}(H^2 + \phi^0\phi^0 + 2\phi^+\phi^-)\right) + \tfrac{2M^4}{g^2}\alpha_h - \\
g\alpha_h M \left(H^3 + H\phi^0\phi^0 + 2H\phi^+\phi^-\right) - \\
\tfrac{1}{8}g^2\alpha_h \left(H^4 + (\phi^0)^4 + 4(\phi^+\phi^-)^2 + 4(\phi^0)^2\phi^+\phi^- + 4H^2\phi^+\phi^- + 2(\phi^0)^2 H^2\right) - \\
gMW^+_\mu W^-_\mu H - \tfrac{1}{2}g\tfrac{M}{c_w^2} Z^0_\mu Z^0_\mu H - \\
\tfrac{1}{2}ig \left(W^+_\mu (\phi^0 \partial_\mu \phi^- - \phi^- \partial_\mu \phi^0) - W^-_\mu (\phi^0 \partial_\mu \phi^+ - \phi^+ \partial_\mu \phi^0)\right) + \\
\tfrac{1}{2}g \left(W^+_\mu (H\partial_\mu \phi^- - \phi^- \partial_\mu H) + W^-_\mu (H\partial_\mu \phi^+ - \phi^+ \partial_\mu H)\right) + \tfrac{1}{2}g\tfrac{1}{c_w}(Z^0_\mu (H\partial_\mu \phi^0 - \phi^0 \partial_\mu H) + \\
M \left(\tfrac{1}{c_w} Z^0_\mu \partial_\mu \phi^0 + W^+_\mu \partial_\mu \phi^- + W^-_\mu \partial_\mu \phi^+\right) - ig\tfrac{s_w^2}{c_w} M Z^0_\mu (W^+_\mu \phi^- - W^-_\mu \phi^+) + igs_w M A_\mu (W^+_\mu \phi^- - \\
W^-_\mu \phi^+) - ig\tfrac{1-2c_w^2}{2c_w} Z^0_\mu (\phi^+ \partial_\mu \phi^- - \phi^- \partial_\mu \phi^+) + igs_w A_\mu (\phi^+ \partial_\mu \phi^- - \phi^- \partial_\mu \phi^+) - \\
\tfrac{1}{4}g^2 W^+_\mu W^-_\mu \left(H^2 + (\phi^0)^2 + 2\phi^+\phi^-\right) - \tfrac{1}{8}g^2 \tfrac{1}{c_w^2} Z^0_\mu Z^0_\mu \left(H^2 + (\phi^0)^2 + 2(2s_w^2 - 1)^2 \phi^+\phi^-\right) - \\
\tfrac{1}{2}g^2 \tfrac{s_w^2}{c_w} Z^0_\mu \phi^0 (W^+_\mu \phi^- + W^-_\mu \phi^+) - \tfrac{1}{2}ig^2 \tfrac{s_w^2}{c_w} Z^0_\mu H (W^+_\mu \phi^- - W^-_\mu \phi^+) + \tfrac{1}{2}g^2 s_w A_\mu \phi^0 (W^+_\mu \phi^- + \\
W^-_\mu \phi^+) + \tfrac{1}{2}ig^2 s_w A_\mu H (W^+_\mu \phi^- - W^-_\mu \phi^+) - g^2 \tfrac{s_w}{c_w}(2c_w^2 - 1) Z^0_\mu A_\mu \phi^+\phi^- - \\
g^2 s_w^2 A_\mu A_\mu \phi^+\phi^- + \tfrac{1}{2}ig_s \lambda^a_{ij} (\bar{q}^\sigma_i \gamma^\mu q^\sigma_j) g^a_\mu - \bar{e}^\lambda (\gamma\partial + m^\lambda_e) e^\lambda - \bar{\nu}^\lambda (\gamma\partial + m^\lambda_\nu) \nu^\lambda - \bar{u}^\lambda_j (\gamma\partial + \\
m^\lambda_u) u^\lambda_j - \bar{d}^\lambda_j (\gamma\partial + m^\lambda_d) d^\lambda_j + igs_w A_\mu \left(-(\bar{e}^\lambda \gamma^\mu e^\lambda) + \tfrac{2}{3}(\bar{u}^\lambda_j \gamma^\mu u^\lambda_j) - \tfrac{1}{3}(\bar{d}^\lambda_j \gamma^\mu d^\lambda_j)\right) + \\
\tfrac{ig}{4c_w} Z^0_\mu \{(\bar{\nu}^\lambda \gamma^\mu (1 + \gamma^5) \nu^\lambda) + (\bar{e}^\lambda \gamma^\mu (4s_w^2 - 1 - \gamma^5) e^\lambda) + (\bar{d}^\lambda_j \gamma^\mu (\tfrac{4}{3}s_w^2 - 1 - \gamma^5) d^\lambda_j) + \\
(\bar{u}^\lambda_j \gamma^\mu (1 - \tfrac{8}{3}s_w^2 + \gamma^5) u^\lambda_j)\} + \tfrac{ig}{2\sqrt{2}} W^+_\mu \left((\bar{\nu}^\lambda \gamma^\mu (1 + \gamma^5) U^{lep}{}_{\lambda\kappa} e^\kappa) + (\bar{u}^\lambda_j \gamma^\mu (1 + \gamma^5) C_{\lambda\kappa} d^\kappa_j)\right) + \\
\tfrac{ig}{2\sqrt{2}} W^-_\mu \left((\bar{e}^\kappa U^{lep\dagger}{}_{\kappa\lambda} \gamma^\mu (1 + \gamma^5) \nu^\lambda) + (\bar{d}^\kappa_j C^\dagger_{\kappa\lambda} \gamma^\mu (1 + \gamma^5) u^\lambda_j)\right) + \\
\tfrac{ig}{2M\sqrt{2}} \phi^+ \left(-m^\kappa_e (\bar{\nu}^\lambda U^{lep}{}_{\lambda\kappa} (1 - \gamma^5) e^\kappa) + m^\lambda_\nu (\bar{\nu}^\lambda U^{lep}{}_{\lambda\kappa} (1 + \gamma^5) e^\kappa\right) + \\
\tfrac{ig}{2M\sqrt{2}} \phi^- \left(m^\lambda_e (\bar{e}^\lambda U^{lep\dagger}{}_{\lambda\kappa} (1 + \gamma^5) \nu^\kappa) - m^\kappa_\nu (\bar{e}^\lambda U^{lep\dagger}{}_{\lambda\kappa} (1 - \gamma^5) \nu^\kappa\right) - \tfrac{g}{2}\tfrac{m^\lambda_\nu}{M} H (\bar{\nu}^\lambda \nu^\lambda) - \\
\tfrac{g}{2}\tfrac{m^\lambda_e}{M} H (\bar{e}^\lambda e^\lambda) + \tfrac{ig}{2}\tfrac{m^\lambda_\nu}{M} \phi^0 (\bar{\nu}^\lambda \gamma^5 \nu^\lambda) - \tfrac{ig}{2}\tfrac{m^\lambda_e}{M} \phi^0 (\bar{e}^\lambda \gamma^5 e^\lambda) - \tfrac{1}{4}\bar{\nu}_\lambda M^R_{\lambda\kappa} (1 - \gamma_5) \hat{\nu}_\kappa - \\
\tfrac{1}{4}\bar{\nu}_\lambda M^R_{\lambda\kappa} (1 - \gamma_5) \hat{\nu}_\kappa + \tfrac{ig}{2M\sqrt{2}} \phi^+ \left(-m^\kappa_d (\bar{u}^\lambda_j C_{\lambda\kappa} (1 - \gamma^5) d^\kappa_j) + m^\lambda_u (\bar{u}^\lambda_j C_{\lambda\kappa} (1 + \gamma^5) d^\kappa_j\right) + \\
\tfrac{ig}{2M\sqrt{2}} \phi^- \left(m^\lambda_d (\bar{d}^\lambda_j C^\dagger_{\lambda\kappa} (1 + \gamma^5) u^\kappa_j) - m^\kappa_u (\bar{d}^\lambda_j C^\dagger_{\lambda\kappa} (1 - \gamma^5) u^\kappa_j\right) - \tfrac{g}{2}\tfrac{m^\lambda_u}{M} H (\bar{u}^\lambda_j u^\lambda_j) - \\
\tfrac{g}{2}\tfrac{m^\lambda_d}{M} H (\bar{d}^\lambda_j d^\lambda_j) + \tfrac{ig}{2}\tfrac{m^\lambda_u}{M} \phi^0 (\bar{u}^\lambda_j \gamma^5 u^\lambda_j) - \tfrac{ig}{2}\tfrac{m^\lambda_d}{M} \phi^0 (\bar{d}^\lambda_j \gamma^5 d^\lambda_j) + \bar{G}^a \partial^2 G^a + g_s f^{abc} \partial_\mu \bar{G}^a G^b g^c_\mu + \\
\bar{X}^+ (\partial^2 - M^2) X^+ + \bar{X}^- (\partial^2 - M^2) X^- + \bar{X}^0 (\partial^2 - \tfrac{M^2}{c_w^2}) X^0 + \bar{Y} \partial^2 Y + igc_w W^+_\mu (\partial_\mu \bar{X}^0 X^- - \\
\partial_\mu \bar{X}^+ X^0) + igs_w W^+_\mu (\partial_\mu \bar{Y} X^- - \partial_\mu \bar{X}^+ Y) + igc_w W^-_\mu (\partial_\mu \bar{X}^- X^0 - \\
\partial_\mu \bar{X}^0 X^+) + igs_w W^-_\mu (\partial_\mu \bar{X}^- Y - \partial_\mu \bar{Y} X^+) + igc_w Z^0_\mu (\partial_\mu \bar{X}^+ X^+ - \\
\partial_\mu \bar{X}^- X^-) + igs_w A_\mu (\partial_\mu \bar{X}^+ X^+ - \\
\partial_\mu \bar{X}^- X^-) - \tfrac{1}{2}gM \left(\bar{X}^+ X^+ H + \bar{X}^- X^- H + \tfrac{1}{c_w^2} \bar{X}^0 X^0 H\right) + \tfrac{1-2c_w^2}{2c_w} igM \left(\bar{X}^+ X^0 \phi^+ - \bar{X}^- X^0 \phi^-\right) + \\
\tfrac{1}{2c_w} igM \left(\bar{X}^0 X^- \phi^+ - \bar{X}^0 X^+ \phi^-\right) + igMs_w \left(\bar{X}^0 X^- \phi^+ - \bar{X}^0 X^+ \phi^-\right) + \\
\tfrac{1}{2} igM \left(\bar{X}^+ X^+ \phi^0 - \bar{X}^- X^- \phi^0\right).
\end{gathered}$$

In machine learning, we assume that every pattern in data can be approximated by mathematical functions

$$\hat{y} = f(w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_{10{,}000} x_{10{,}000} + b)$$

What’s going on inside the machine?



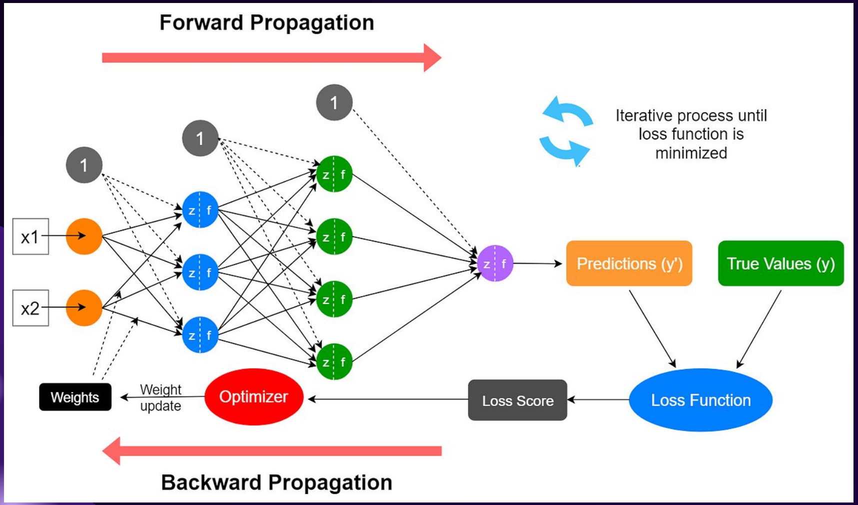


- We train the machine with thousands of samples so that the machine can recognize in the data.
- During training, the machine adjusts weights and biases to improve accuracy.
- In YOLOv8, the image is passed as an input through layers of the network that do some calculation to help determine the image and come up with final prediction.

How does the network get better?

Prediction (Forward Pass) → **Compare to real answer** (Loss Function) → **Learn** (Backpropagation)

https://xnought.github.io/backprop-explainer/



- **Loss Function** tells how wrong the prediction (output) is.
- **Backpropagation** (Backward Pass) calculates how much each weight and bias contributes to final error.
- Then, those parameters are updated by optimizer.
- We train the machine until the loss reaching a desirable low value.

Evaluation metrics

- When the prediction and actual label do not match, it is FALSE, else TRUE
- When the class is predicted, it is POSITIVE, else NEGATIVE

		Predicted Label	
		Positive	Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

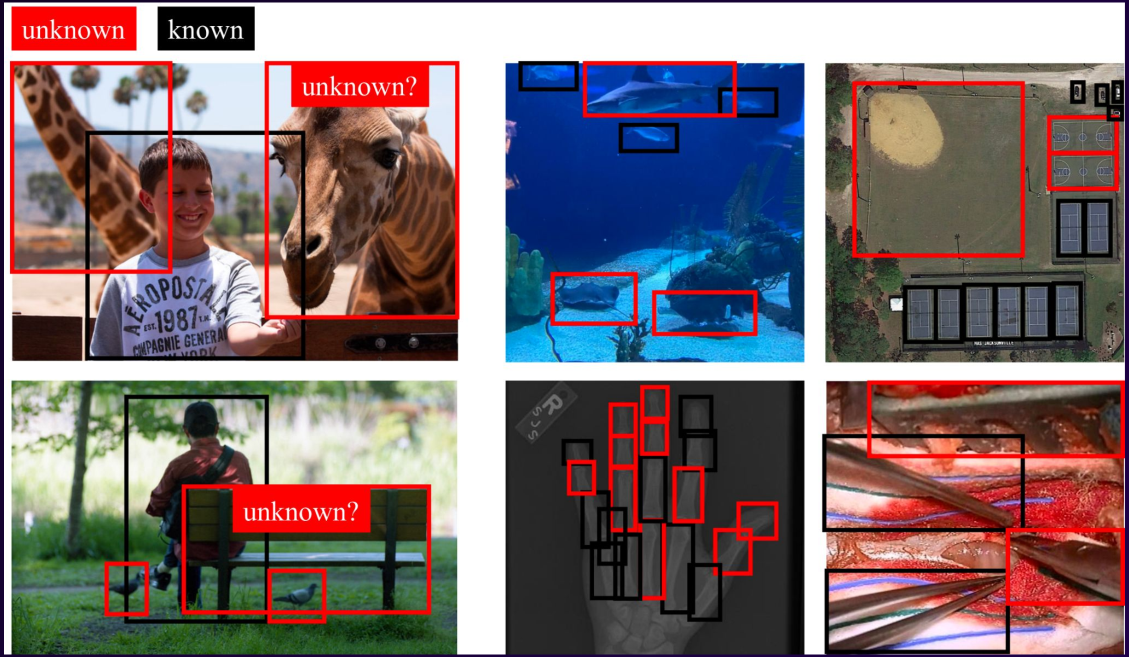
F1 Score

- Precision = TP / (TP + FP)
  - Focus on predicted classes being correct.
  - **Correctness - do not make wrong prediction**.
- Recall = TP / (TP + FN)
  - Focus on actual classes being predicted.
  - **Completeness - do not miss out**.
- F1 score = (2 x Precision x Recall) / (Precision + Recall)
  - Balance between precision and recall

		Predicted Label	
		Positive	Negative
Actual Label	Positive	TRUE POSITIVE TP	FALSE NEGATIVE FN
	Negative	FALSE POSITIVE FP	TRUE NEGATIVE TN

The only things Yolo can detect:

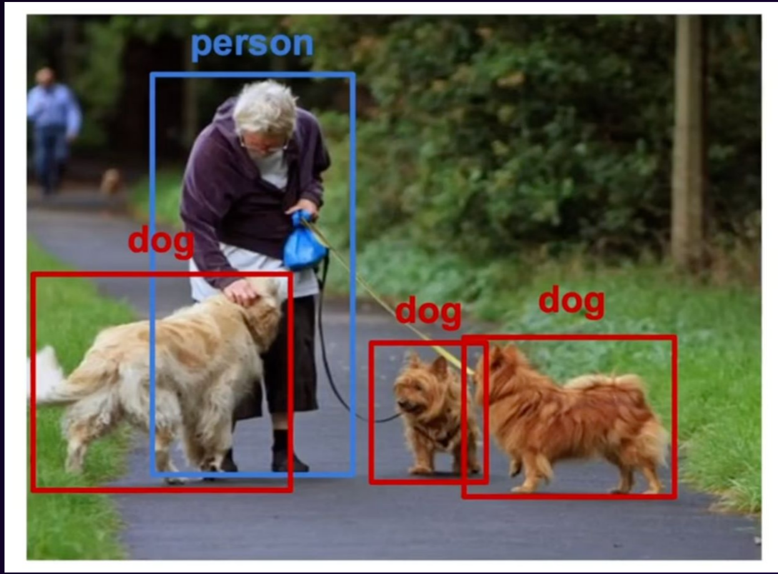
{0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21: 'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26: 'handbag', 27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat', 35: 'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis racket', 39: 'bottle', 40: 'wine glass', 41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange', 50: 'broccoli', 51: 'carrot', 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier', 79: 'toothbrush'}

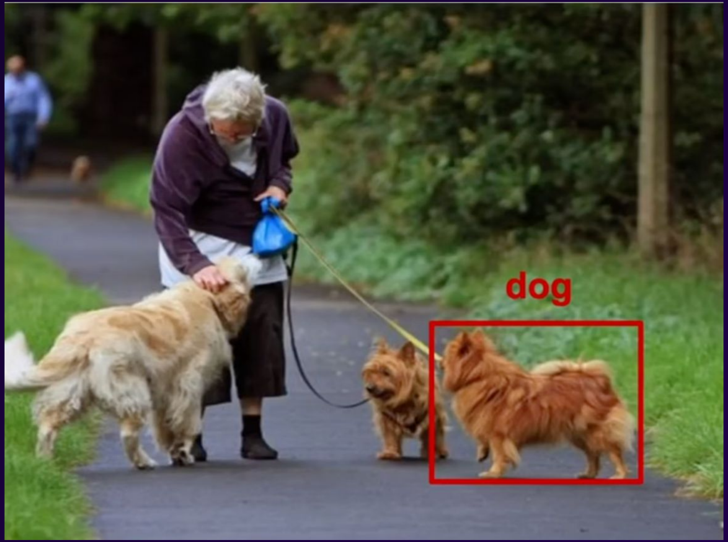


If you want to detect something else, **YOU MUST TRAIN THE MODEL!!!**

This is called **"closed-set object detection"**...

Closed-Set Object Detection VS Opened-Set Object Detection





Find me the objects that you **HAVE BEEN TRAINED**

Find me the rightmost dog **(just prompt keywords)**

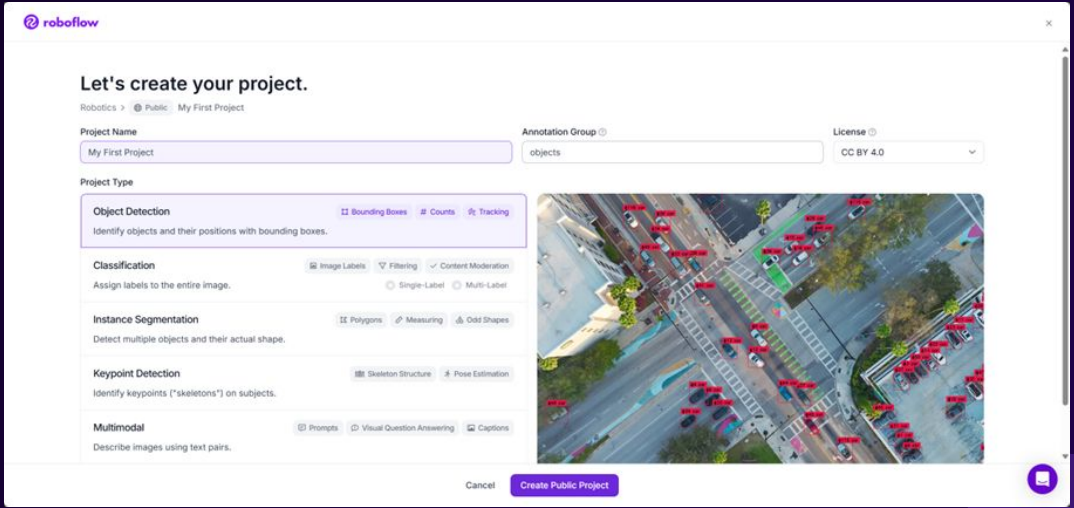

CLOSED-SET
etc.
OPEN-SET

Custom Your Own Dataset

- Closed-Set for specific task!
- Create your Own Label for objects

Custom Your Own Dataset

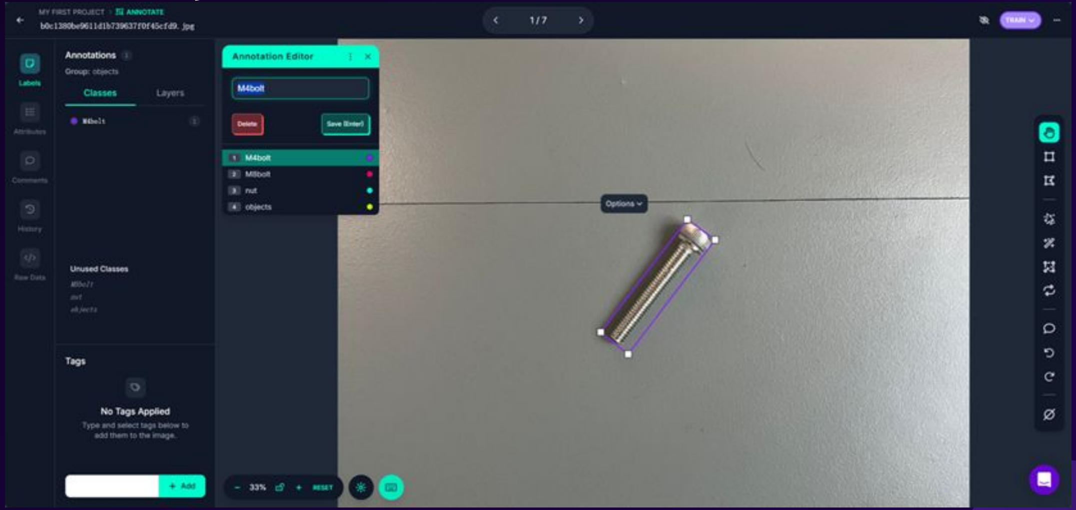
- Platform – Roboflow
- Procedure - Startup



Custom Your Own Dataset

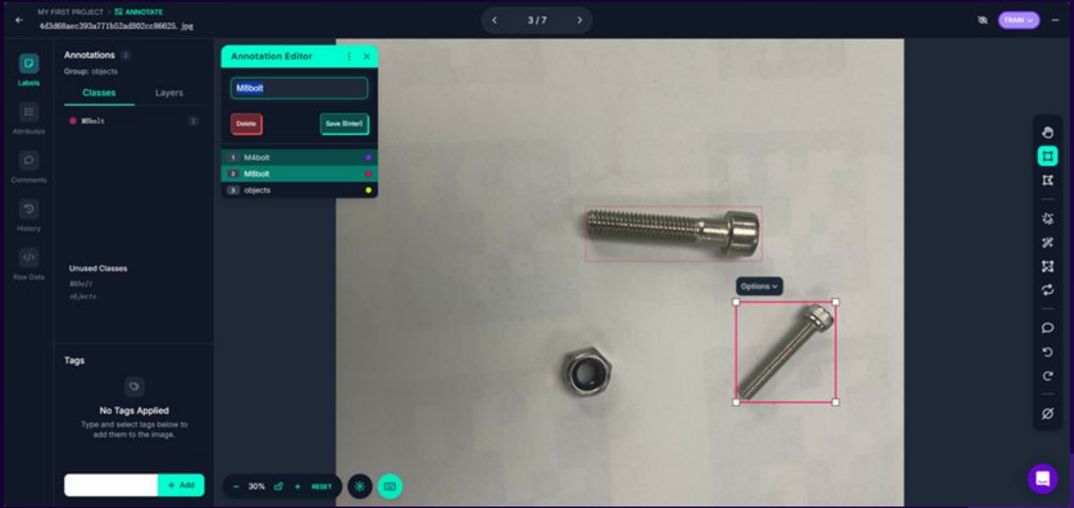
- Procedure – Label

- Single / Multi objects



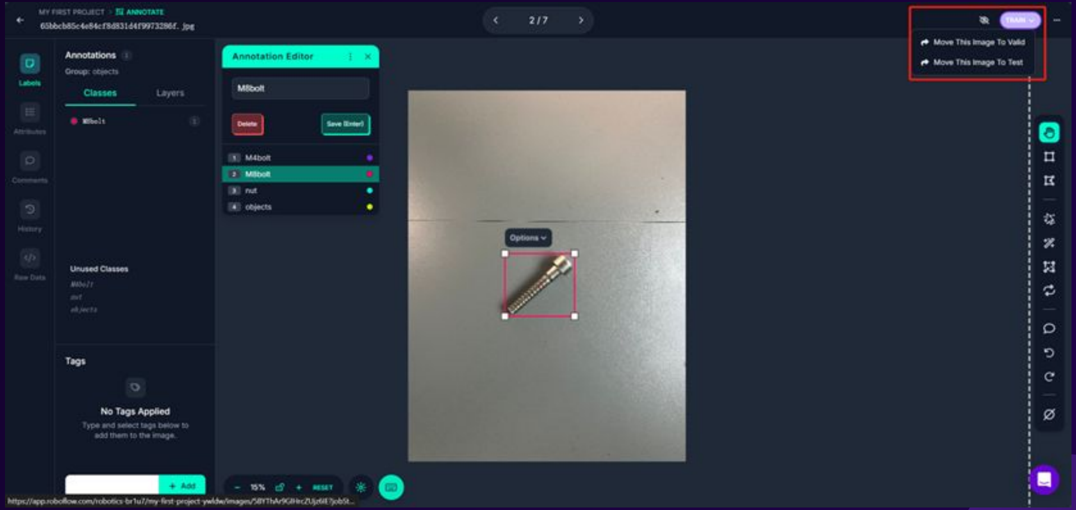
Custom Your Own Dataset

- Procedure – Label
- Bounding Box Shape



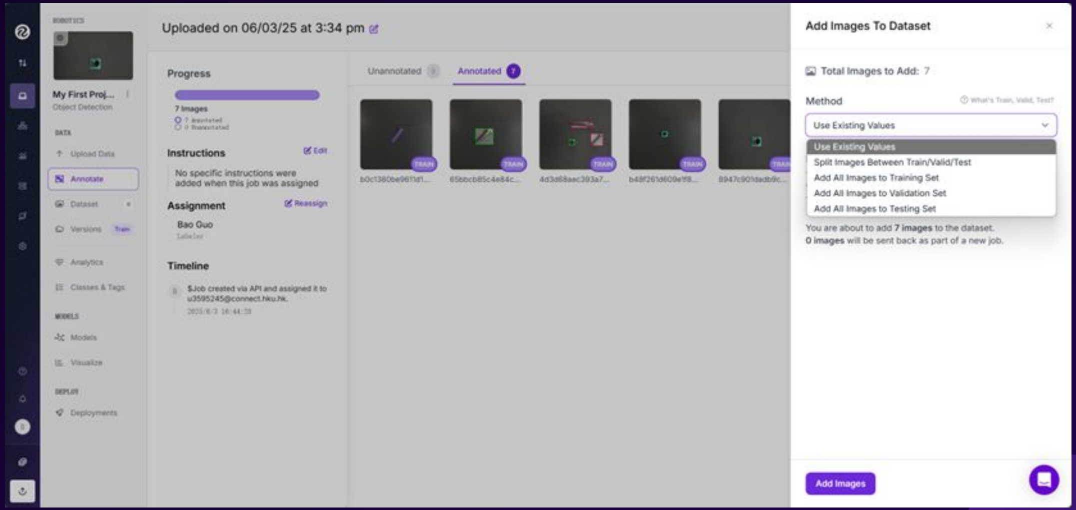
Custom Your Own Dataset

- Procedure – Label
- Data for Purpose



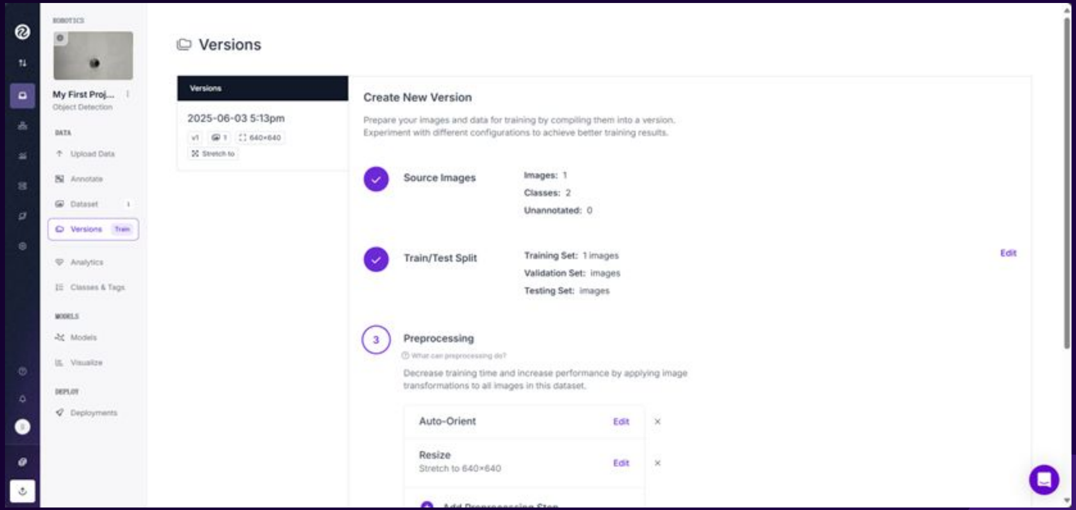
Custom Your Own Dataset

- Procedure – Add to Dataset
- Data for Purpose



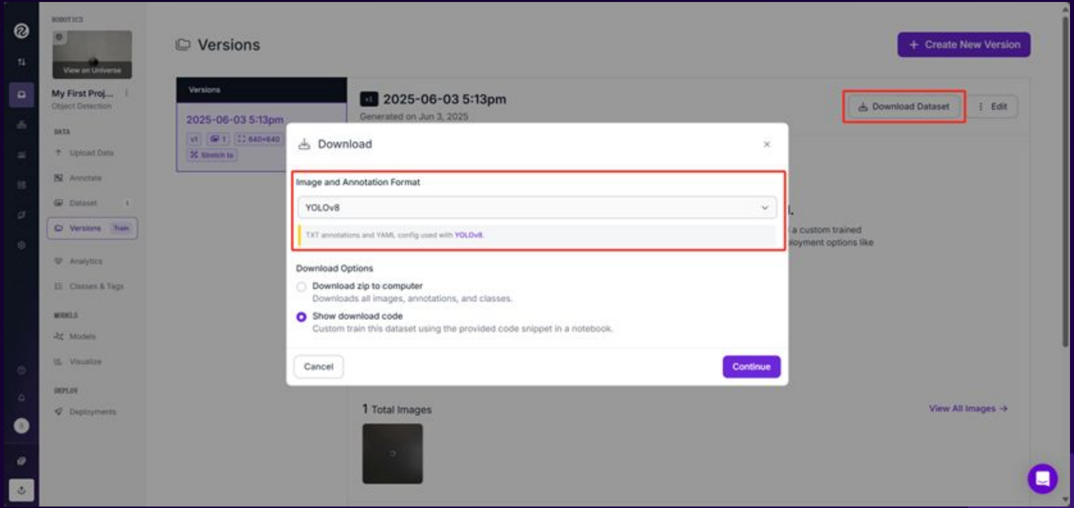
Custom Your Own Dataset

- Procedure – Processing
- Preprocess & Augmentation



Custom Your Own Dataset

- Almost Done!
- Select and download your dataset



Custom Your Own Dataset

- Almost Done!

- Folder Structure

```
May not be exactly the same!

root\
|
|__ data.yaml
|__ README.dataset.txt
|__ README.roboflow.txt
|__ train\
|    |__ images\
|         |__ xxx.jpeg (.png, .jpg)
|         |__ ...
|         |__ xxx.jpeg
|    |__ labels\
|         |__ xxx.jpeg (.png, .jpg)
|         |__ ...
|         |__ xxx.jpeg
|__ val\
|    |__ ...
|__ test\
     |__ ...
```

Custom Your Own Dataset

- Reference: Notion Page

Thanks!!

Any questions?