# Object Detection - YOLOv7

阮柏愷 bkruan.ee11@nycu.edu.tw

黃柏綸 kevin503.ee12@nycu.edu.tw

Special Thanks to Yi-Lun Wu 🤗

# Outline

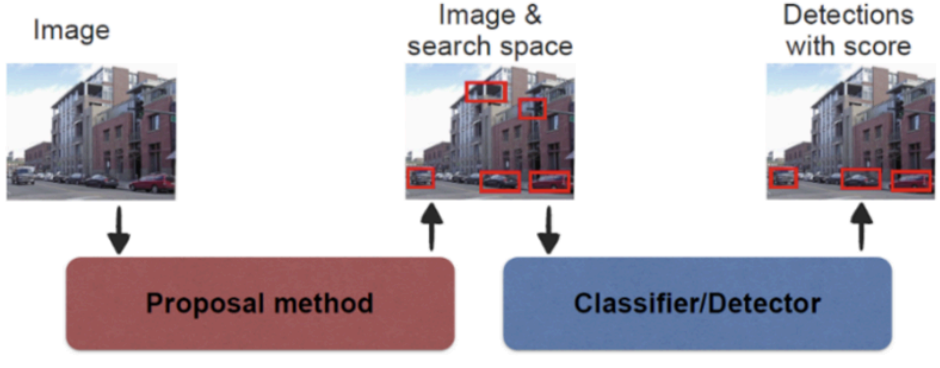
- Traditional Object Detection
- R-CNN, Fast R-CNN, Faster-RCNN
- YOLOv7
- Tasks
    - Create a Virtual Environment
    - Inference
    - Annotation
    - Training

# Object Detection

The algorithm produces a list of object categories present in the image, along with an axis-aligned bounding box that indicates the position and scale of every instance.

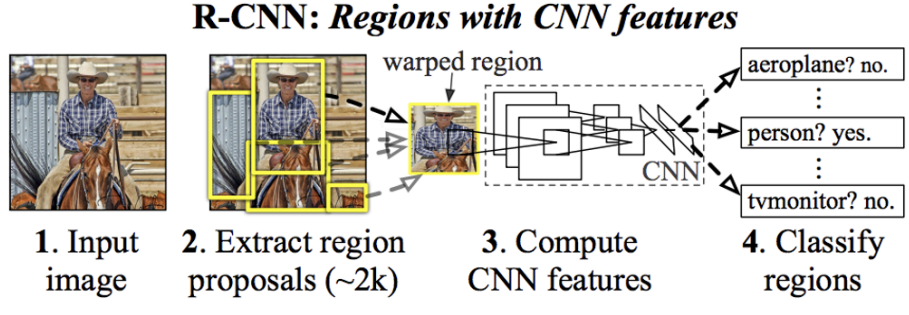
# Traditional Approach

1. Object Localization: Hypothesize bounding boxes, e.g., sliding window.
2. Feature Extraction: Extract features for each box, e.g., SIFT, HOG.
3. Image Classification: Classify each box into an object category, e.g., SVM, KNN.



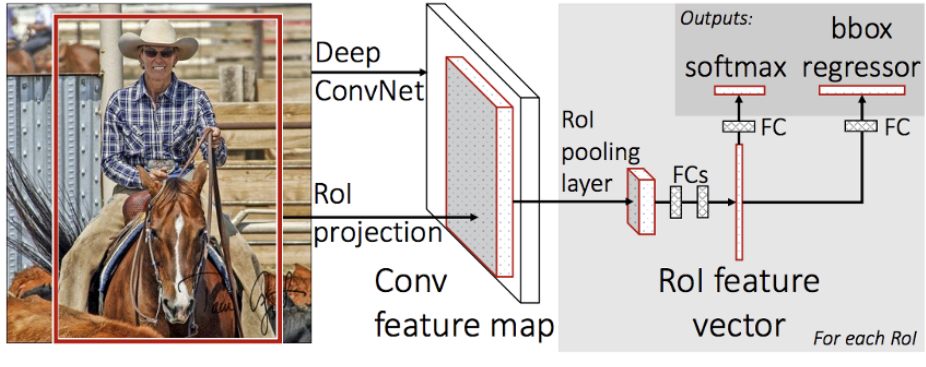
# R-CNN-Based Methods

- Use Selective Search to find region proposals.
- Extract features of each proposal using Convolutional Neural Networks.
- Learn the bounding box through regression and predict the object class with SVM.
- Other variants include Fast R-CNN and Faster-RCNN.



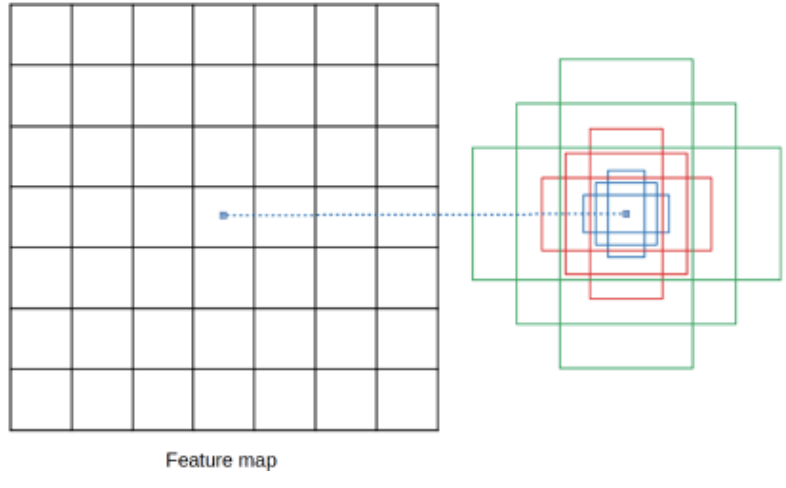
# Fast R-CNN:

- Problem with R-CNN
  - Using a deep CNN to extract all region proposals of an image is very slow.
- Solution:
  - Run the CNN only once to get the features of the full image. Features of region proposals are cropped from the overall features.
  - ROI pooling unifies the dimensions of the cropped features.
  - Replace SVM with Neural Networks.



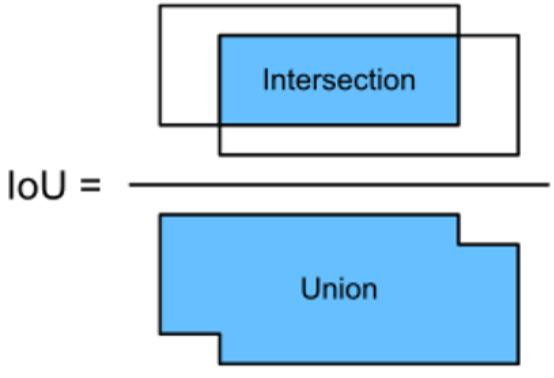
# Faster R-CNN

- Problem with Fast R-CNN:
  - Region proposal is time-consuming.
  - Accuracy is highly affected by region proposals.
- Solution:
  - Learn a region proposal algorithm.
  - Introduce anchor boxes to simplify region proposal.



# Faster R-CNN

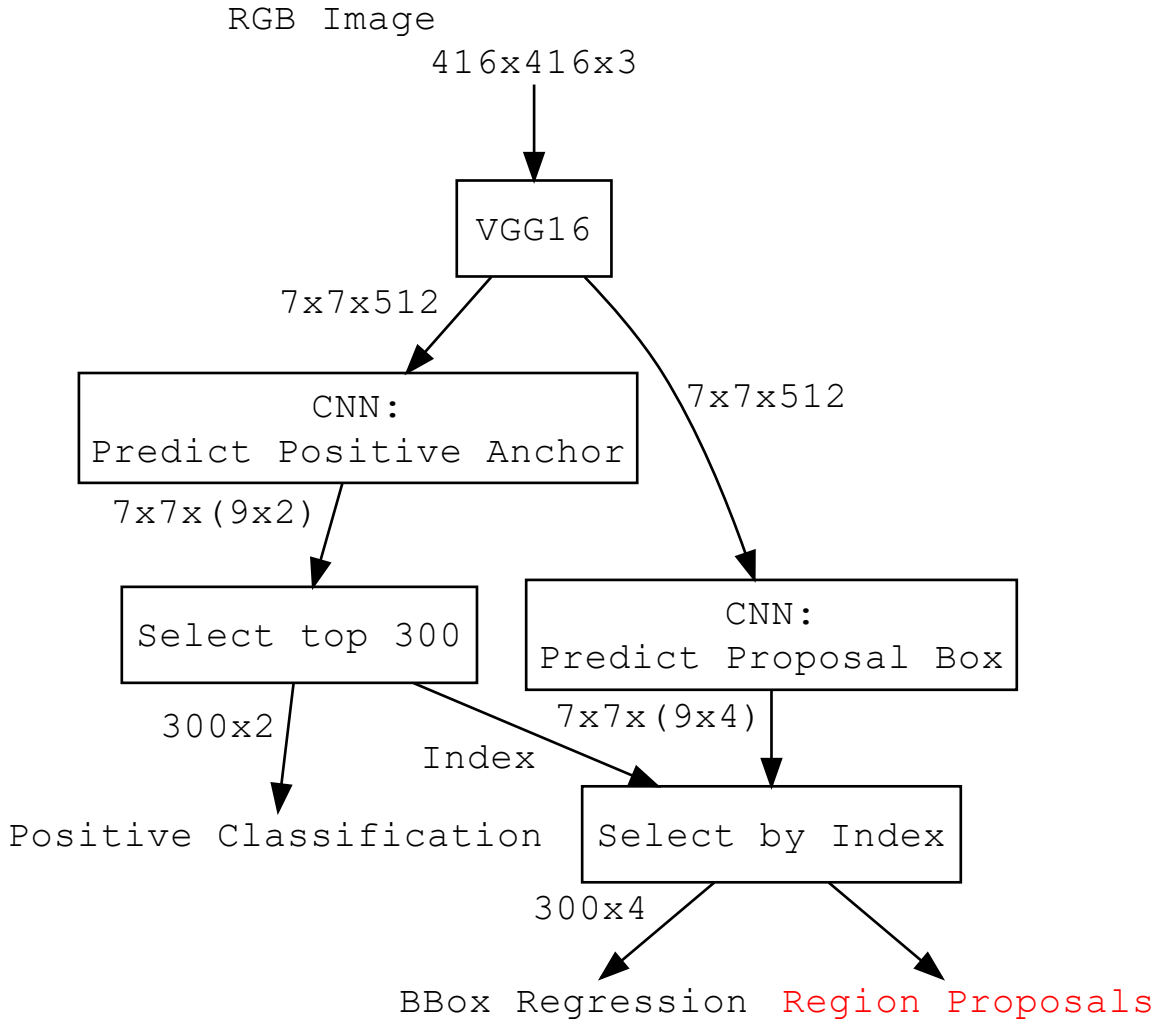
- Intersection over Union (IoU) determines the overlap between a predicted box and a ground-truth box.




- A positive anchor is either:
    1. The anchor with the highest IoU overlap with a ground-truth box.
    2. An anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

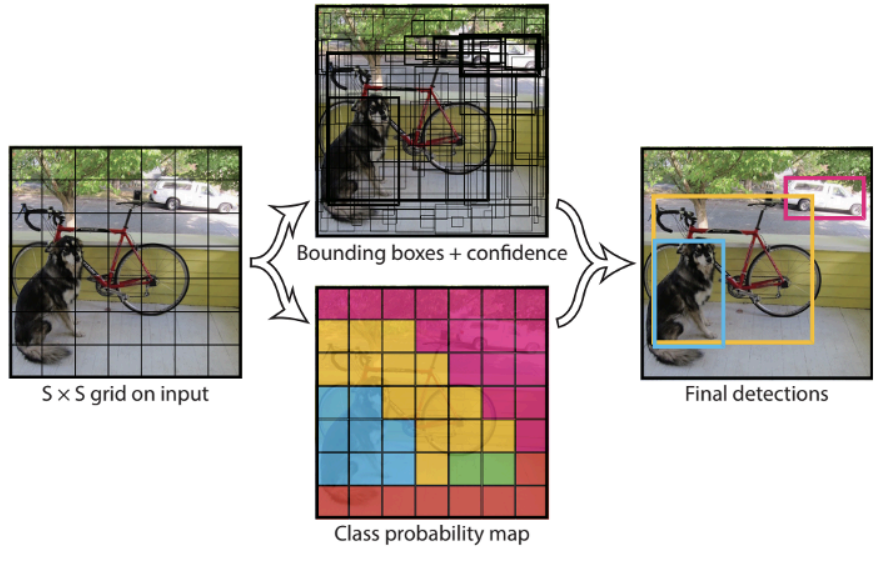
# Faster R-CNN

Learn a region proposal model:

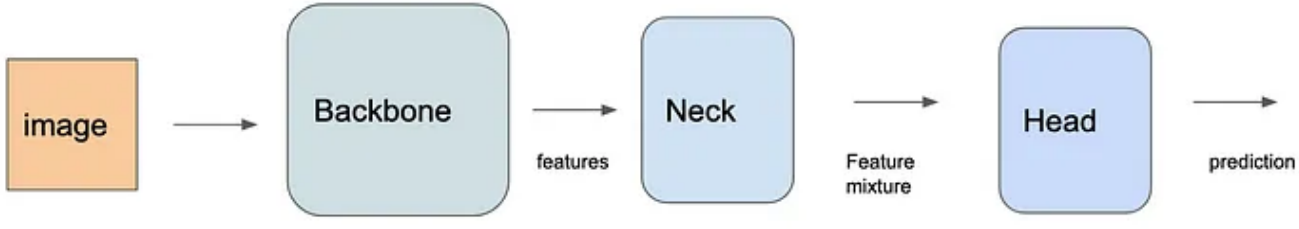


# YOLO

- YOLO Family
  - This includes YOLOv1~8, YOLO-NAS, and others.
  - A single neural network is trained end-to-end to predict bounding boxes and class probabilities.



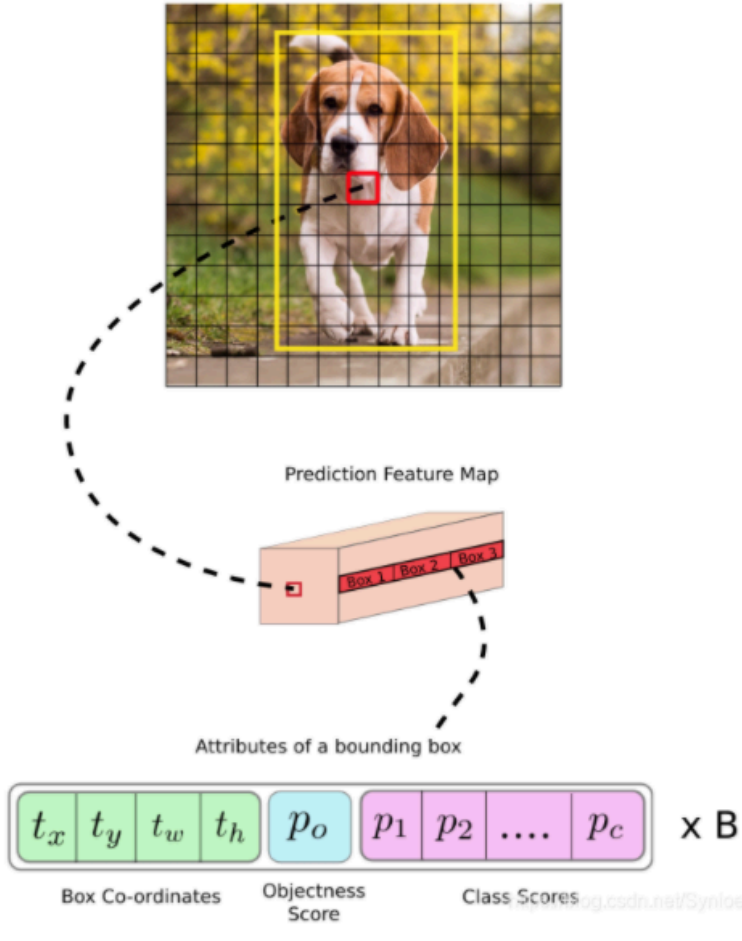
# A general YOLO architecture




https://medium.com/@nahidalam/understanding-yolov7-neural-network-343889e32e4e

- **Backbone**: Feature Extractor.
- **Neck**: Intermediate layers for fusing features from different level or layers,
- **Head**: Predicting the final results (class and bounding boxes),

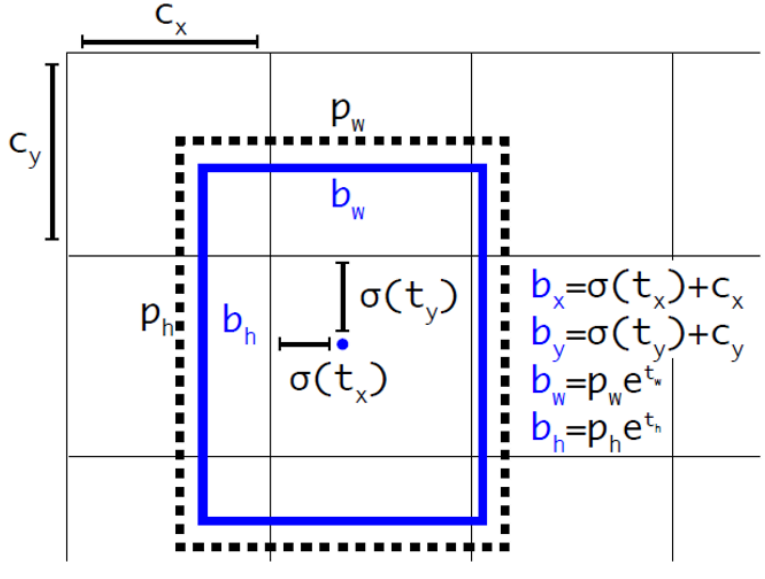
# Each Anchor Box Predicts $5 + n_c$ Values

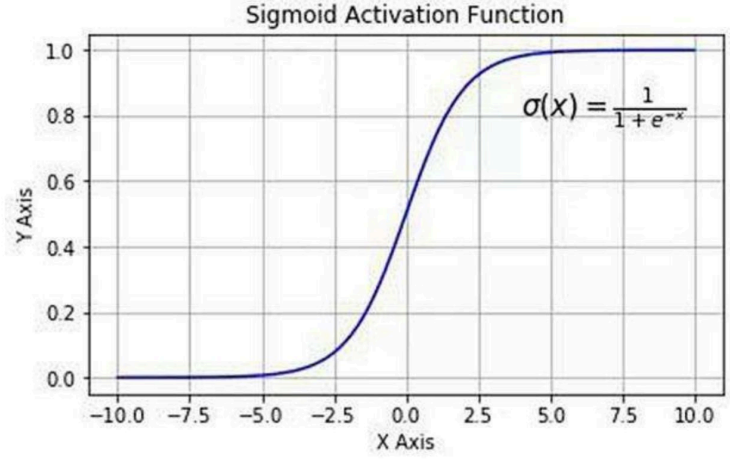



https://b10515007.medium.com/yolov3-%E8%A9%B3%E7%B4%B0%E8%A7%A3%E8%AA%AA-22e3da36260a

# Bounding Box

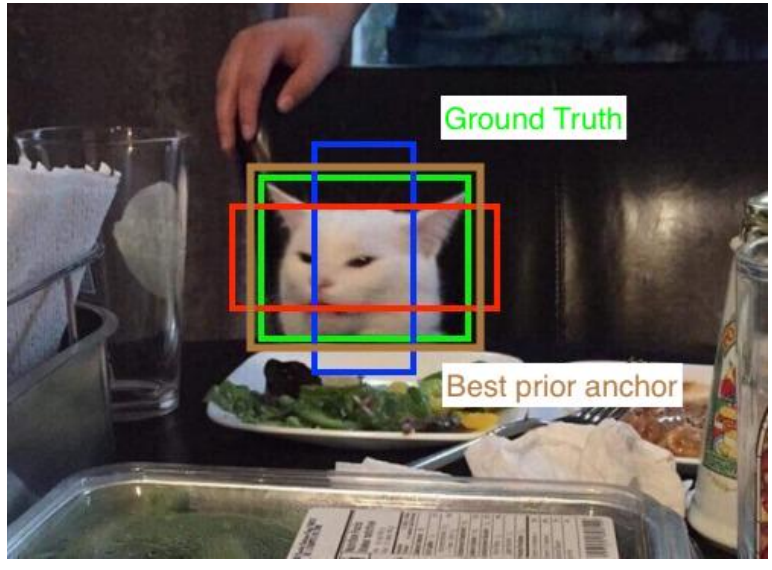
- Predict the distance from the upper-left corner of the grid to the center of the ground truth bounding box. The predicted value is in the range of $[0, 1]$.
- The size of the bounding box is rescaled from the size of the corresponding prior anchor.

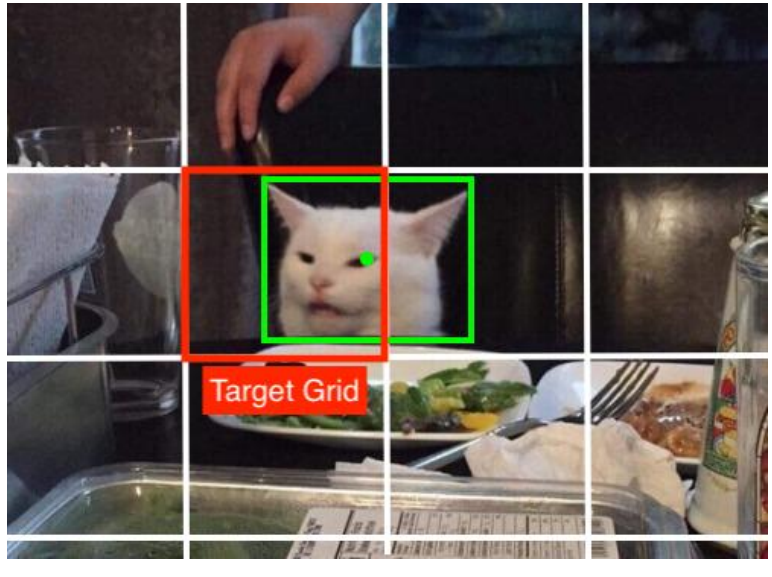


# Bounding Box Loss and Class Loss

- For each ground truth box, center prior anchors and ground truth then select the one with maximum IoU.
- The selected anchor at the grid where the center of the ground truth locates is called the positive anchor, and the label of this anchor is the corresponding ground truth.
- The other anchors (also called negative anchors) do not contribute to the loss calculation in this step.

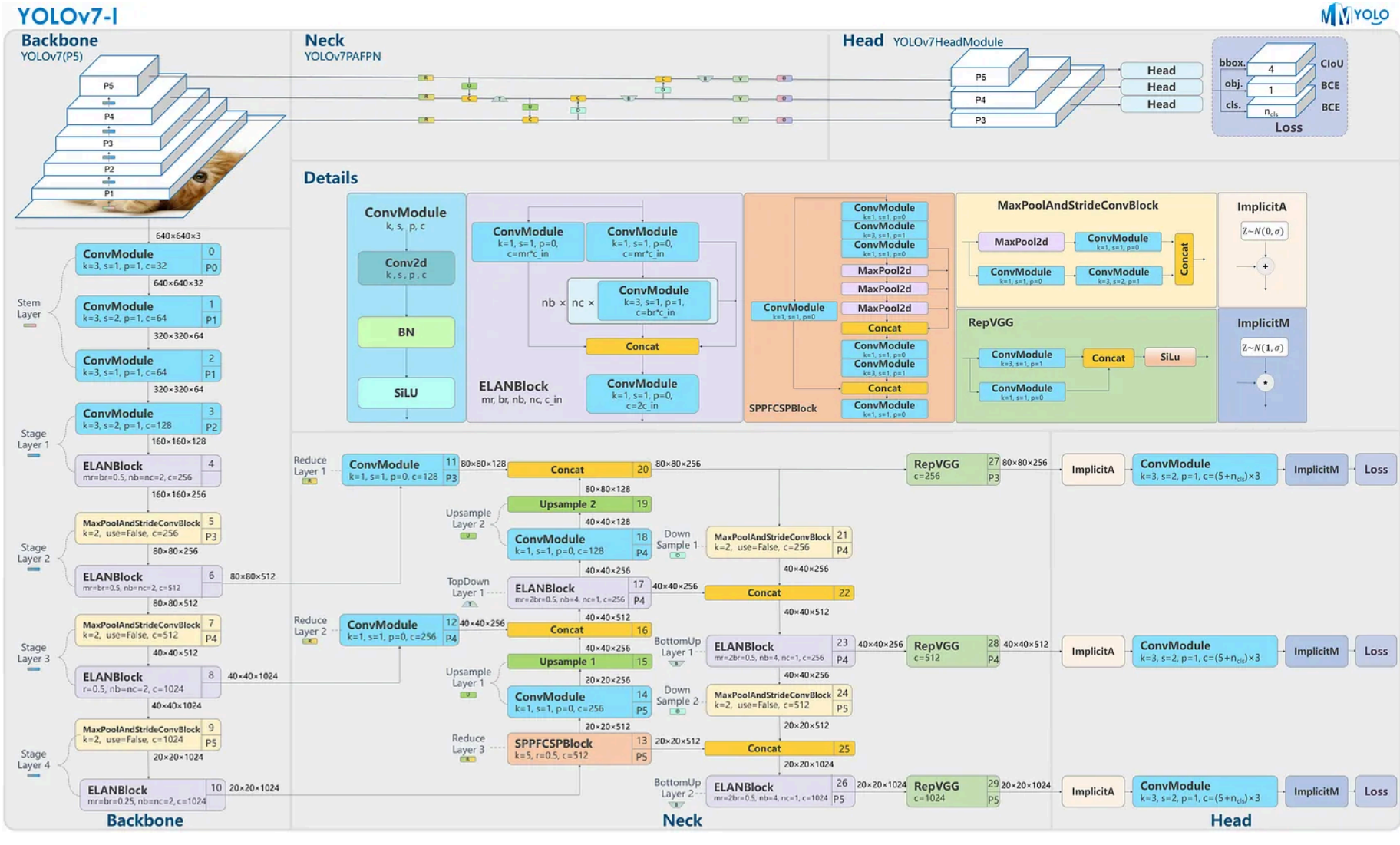

Ground Truth
Best prior anchor


Target Grid

## Objectness Score (Probability $p_o$)

- $p_o = 1$ if it is a positive anchor of some ground truth.
- $p_o = 0$ if the maximum IoU between the predicted bounding box and all ground truth is less than 0.5.
- The other anchors do not contribute to the loss calculation in this step.

# YOLOv7 Model Architecture




https://github.com/open-mmlab/mmyolo

# What's new in YOLOv7?

1. **Extended-Efficient Layer Aggregation Network (only in YOLOv7-E6E)**
   → Solving the deteriorate converge when scaling a model.
2. **Compund model scaling**
   → Using depth and width scaling in different model blocks.
3. **Bag of Freebies (BoF)**
   → BoF refers to the techniques that improve the model without increasing training cost.These techniques include Re-parameterization Convolution, Label Assignment with Varying granularity
4. **and more**

# Loss function in YOLOv7

- Also address the class and the bounding box with some designs derived from: YOLOv4, YOLOv5, SimOTA.
- The prediction of each anchor includes: (x, y, w, h, objectness score, number of classes)

# Source Code

- SSH into an arbitrarily server.
- Clone the source code from Github if you have not already done so:

```
$ git clone https://github.com/Justin900429/hcc-ml
```

  Otherwise, pull the updated code from Github using `git pull`

```
$ cd hcc-ml
$ git pull
```

- Make sure that your working directory is `lab3`.

```
$ cd lab3
```

# Create Virtual Environment

- Create a virtual environment for Lab3.

```
$ python3.12 -m venv {name_of_your_venv}
```

  For example:

```
$ python3.12 -m venv lab3env
```

- Activate your virtual environment.

```
$ source {name_of_your_venv}/bin/activate
# For example
$ source lab3env/bin/activate
```

- Update `pip` and install all required Python packages:

```
(lab3env) $ pip install --upgrade pip
(lab3env) $ pip install -r requirements.txt
```

# Download Pretrained Models

- Run the following commands to download all pretrained weights:

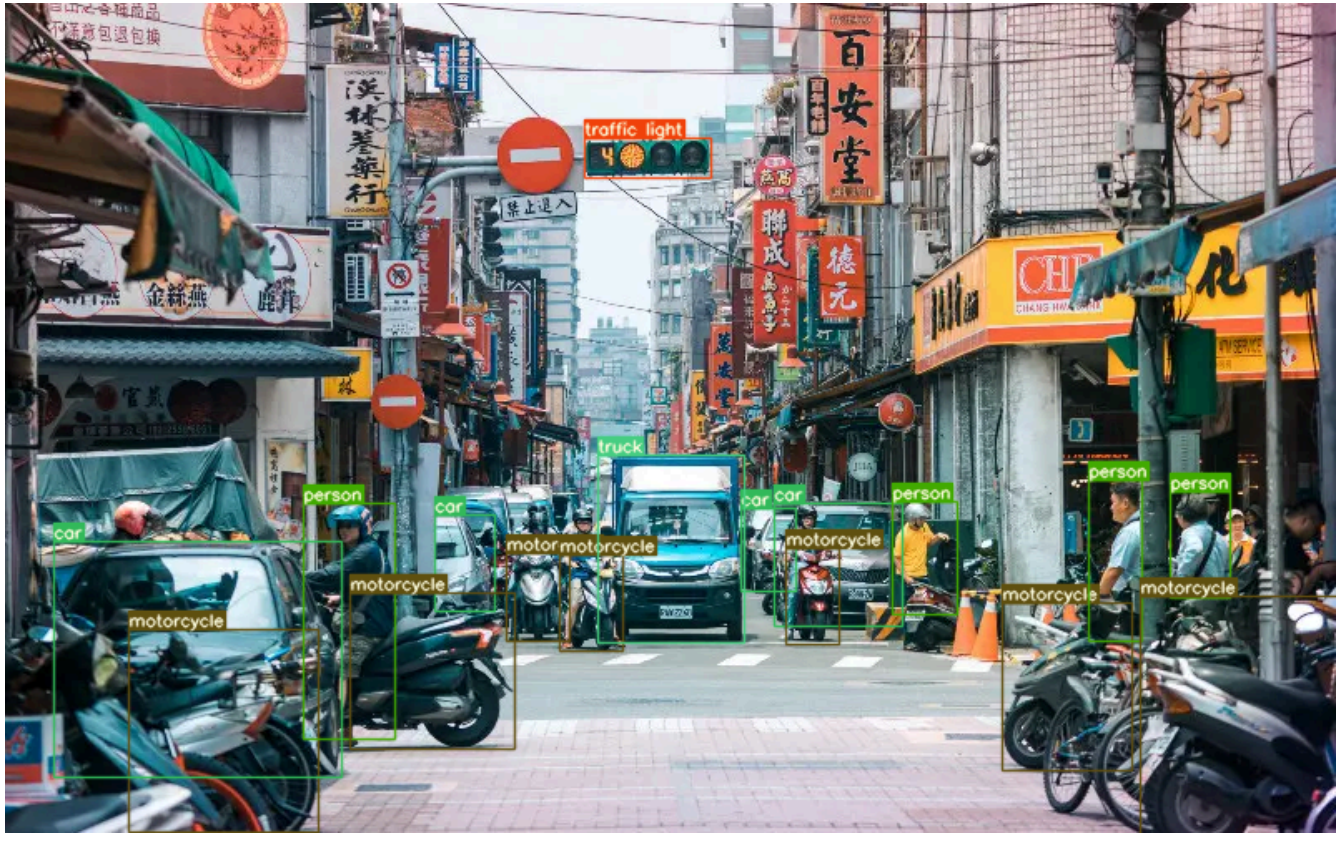
```
$ cd weights
$ ./download_weights.sh
$ cd ..
```

# Checkpoint 1

- Run `demo.py` to test the envrionment and the pretrained weight.

```
$ python demo.py \
    --image ./data/street.jpg \
    --weights ./weights/yolov7.pt
```

- Show `demo.png`



# Checkpoint 2

- Use the flag `--help` to see more options of `demo.py`:

```
$ python demo.py --help
usage: demo.py [-h] [--image IMAGE] [--weights WEIGHTS [WEIGHTS ...]
    [--img_size IMG_SIZE] [--conf_threshold CONF_THRESHOLD]
    [--nms_threshold NMS_THRESHOLD] [--device DEVICE]

options:
  -h, --help            show this help message and exit
  --image IMAGE         source
  --weights WEIGHTS [WEIGHTS ...]
                        model.pt path(s)
  --img_size IMG_SIZE   inference size (pixels)
  --conf_threshold CONF_THRESHOLD
                        object confidence threshold
  --nms_threshold NMS_THRESHOLD
                        IOU threshold for NMS
  --device DEVICE       cuda device, i.e. 0 or 0,1,2,3 or cpu
```

# Checkpoint 2

- Confidence (`--conf_threshold`):
  The confidence of a bbox is the objectness score $\times$ class probability. This is used to filter out the bboxes before Non-Maximum Suppression.
- Non-Maximum Suppression (`--nms_threshold`):
  1. Let $A$ be the set of all candidate bboxes and $B =$ be the initial result.
  2. Repeat steps 3 and 4 until $A$ is empty.
  3. Select bbox $b$ in $A$ with the highest confidence. If the maximum IoU between $b$ and all bboxes in $B$ is less than nms_threshold, add $b$ to set $B$.
  4. Remove $b$ from $A$.

# Checkpoint 2-1

- Change the confidence threshold to detect the traffic light on the right-hand side of the image.

```
$ python demo.py \
    --image ./data/street.jpg \
    --weights ./weights/yolov7.pt \
    --conf_threshold ??
```

# Checkpoint 2-1

# Checkpoint 2-2

- Change the confidence threshold and NMS threshold to detect the motorcycles on the rightmost side of the image (there should be at least three motorcycles).

```
$ python demo.py \
    --image ./data/street.jpg \
    --weights ./weights/yolov7.pt \
    --conf_threshold ?? \
    --nms_threshold ??
```

# Checkpoint 2-2

# Label Studio

# Step 1

The following steps should be executed on your personal computer:

- Install Label Studio by running the following commands:

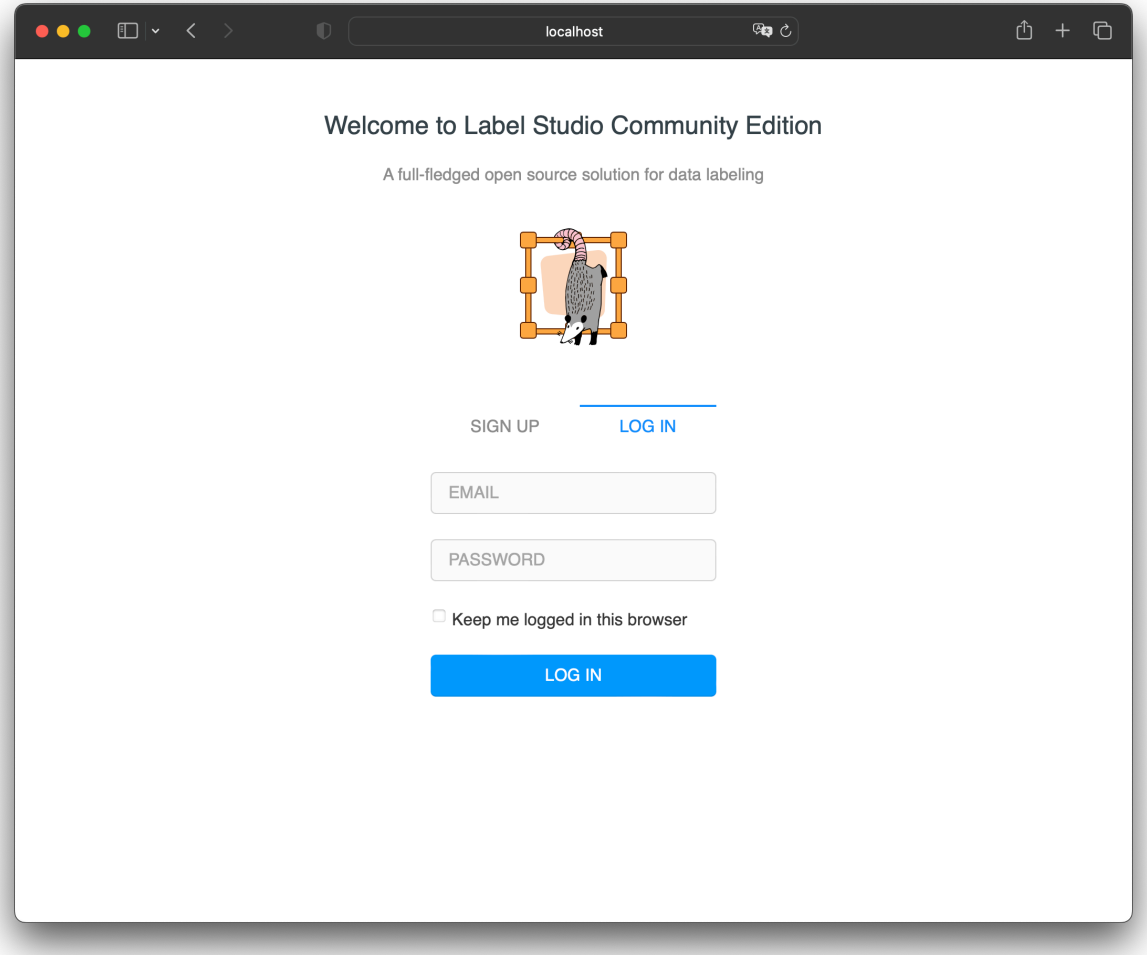
```
$ pip install label-studio
```

- Start Label Studio by entering the following command:

```
$ label-studio
=> Database and media directory: /Users/yilun/Library/Application Support/labe
=> Static URL is set to: /static/
...
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
...
```

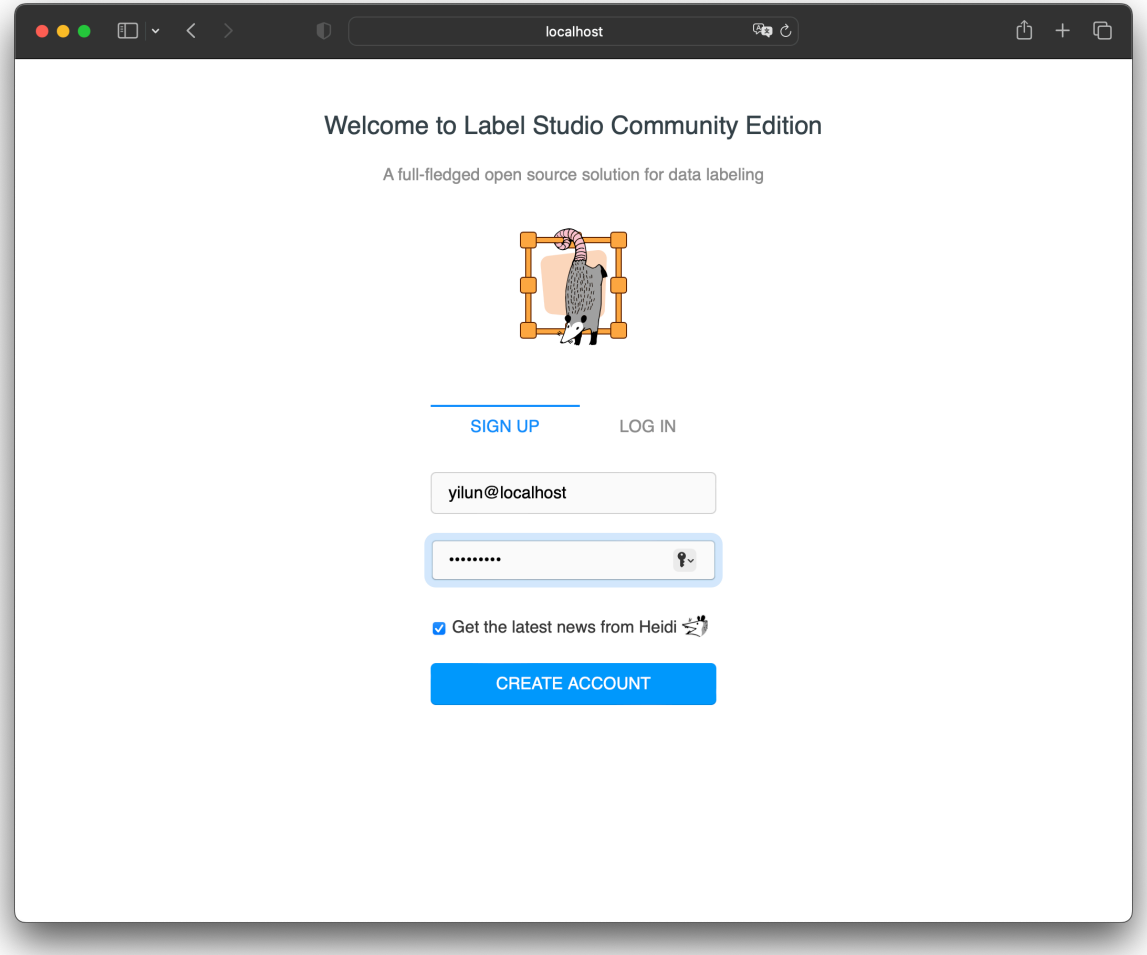
# Step 2

- Visit localhost:8080 in your browser.



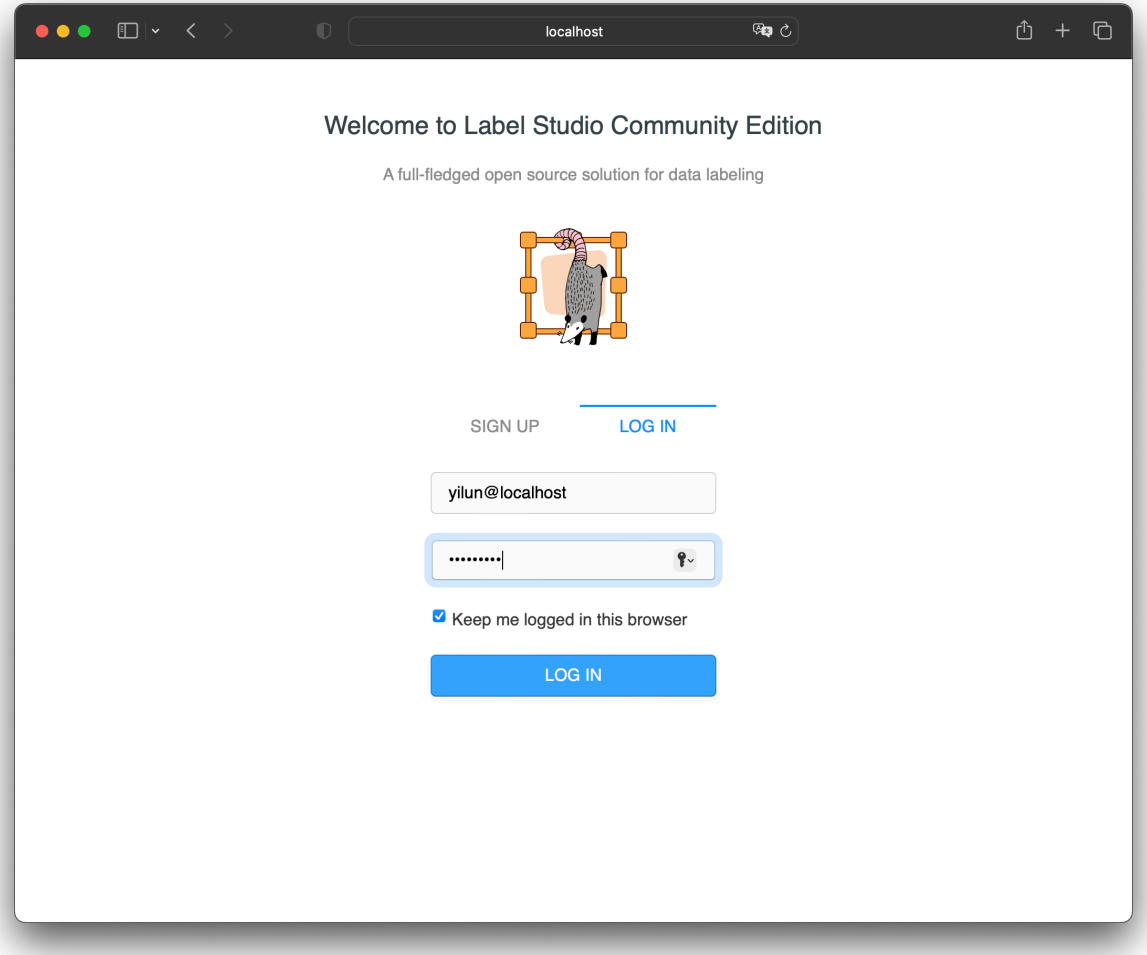
# Step 3

- Sign up with an arbitrary email



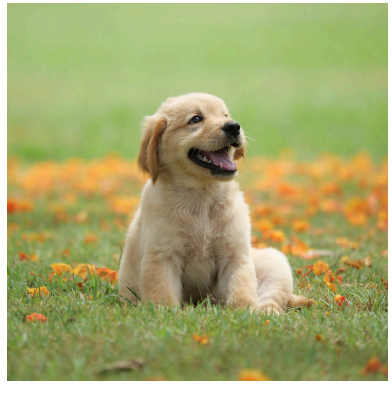
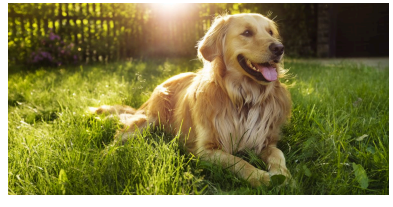
# Step 4

- Log in


localhost
Welcome to Label Studio Community Edition
A full-fledged open source solution for data labeling
SIGN UP
LOG IN
yilun@localhost
Keep me logged in this browser
LOG IN

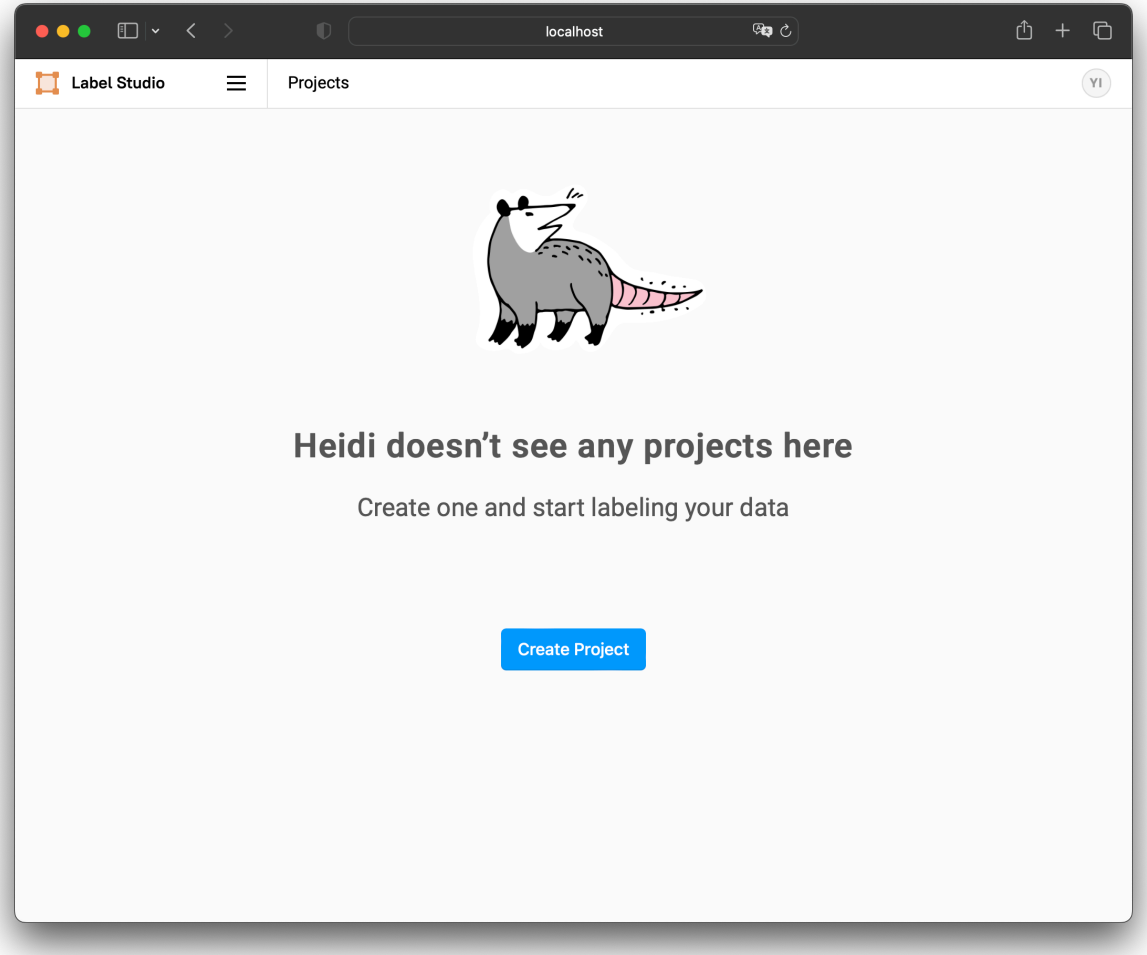
# Step 5

- Collect 6 random dog and cat images from the internet.

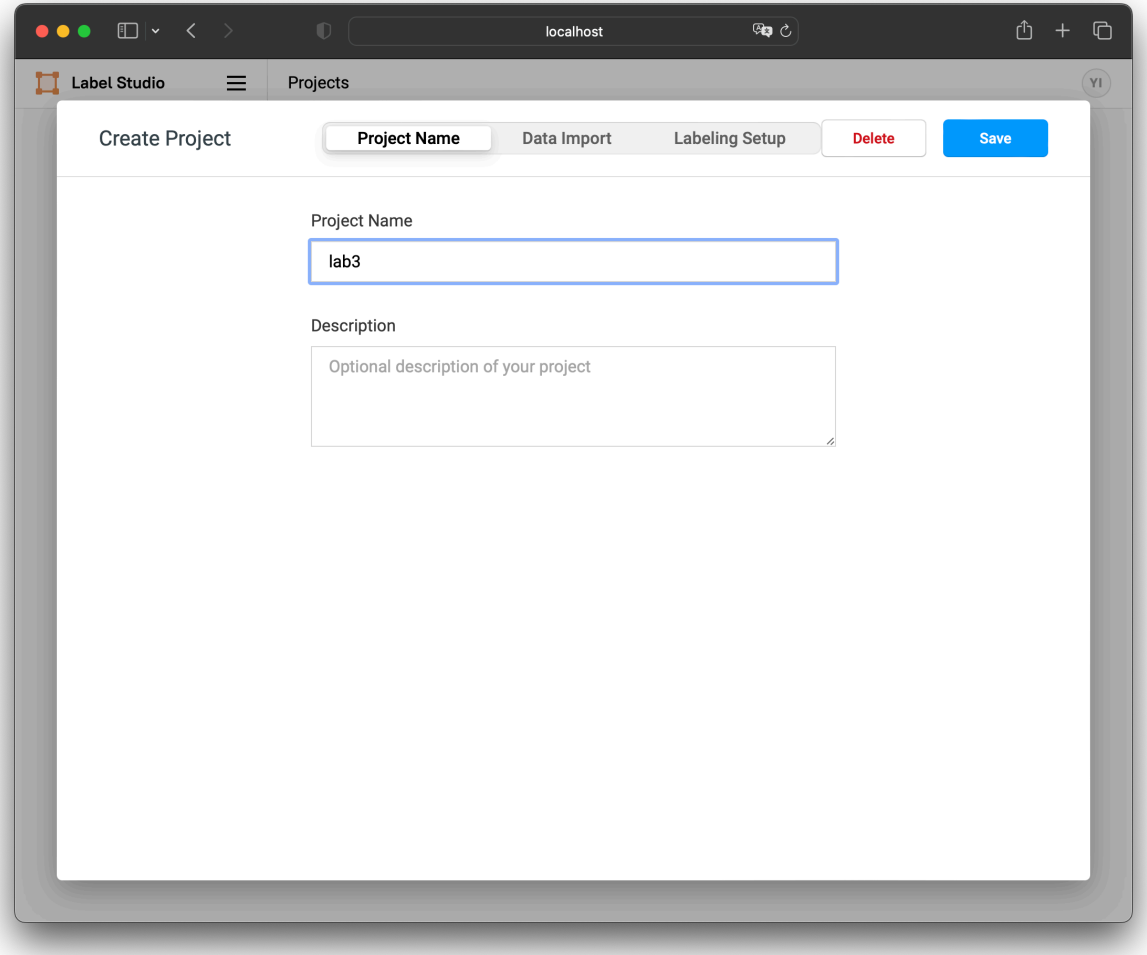

# Step 6

- Create a project



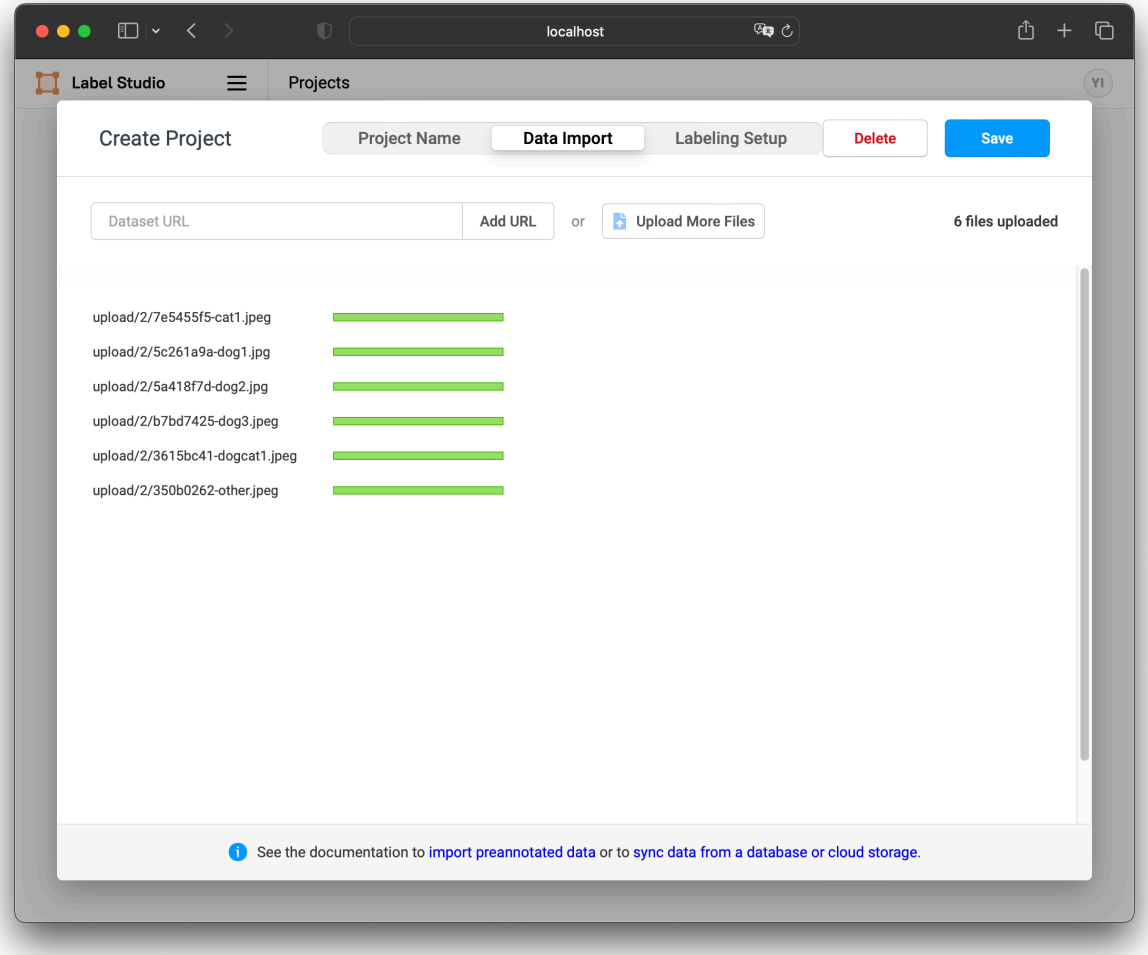
# Step 7

- Set the project name



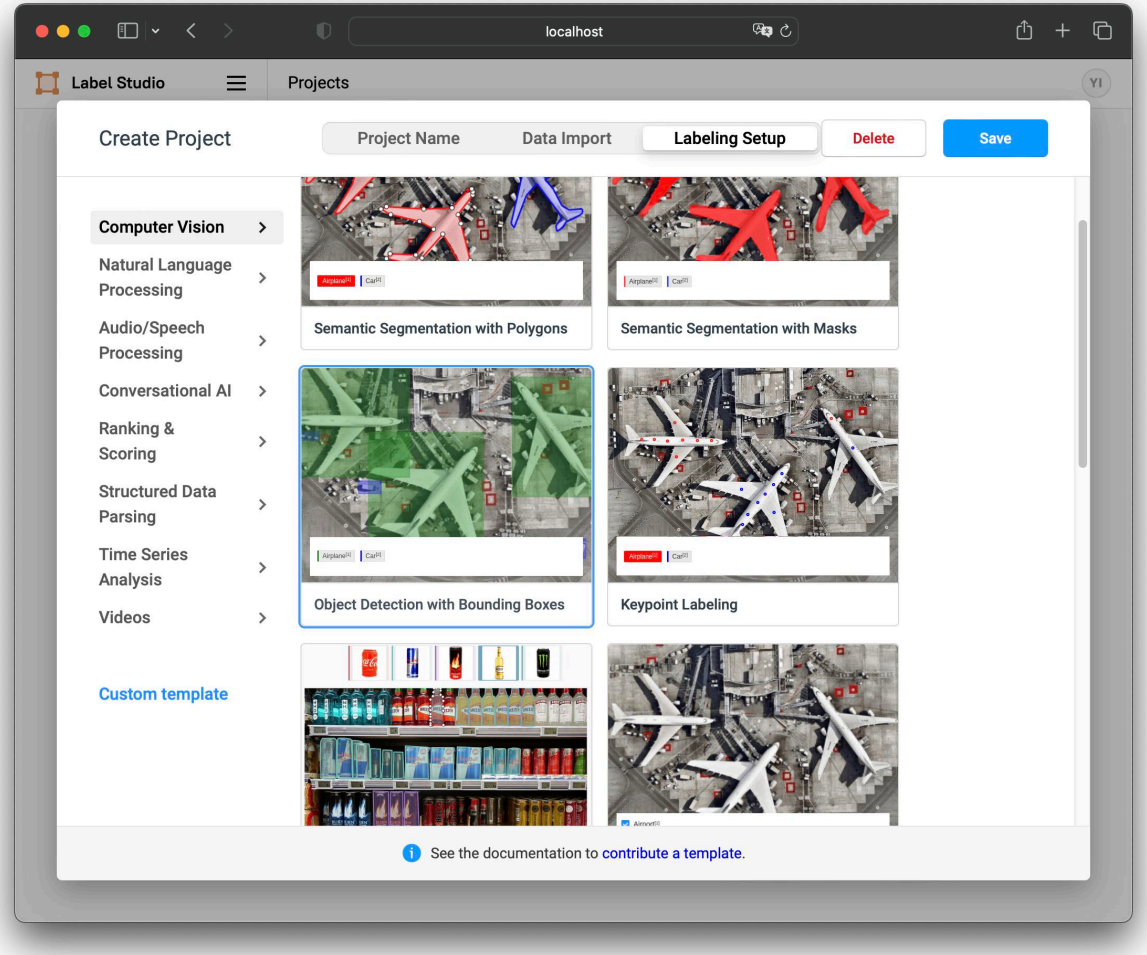
# Step 8

- Use the `Upload more files` button to upload all dog and cat images.



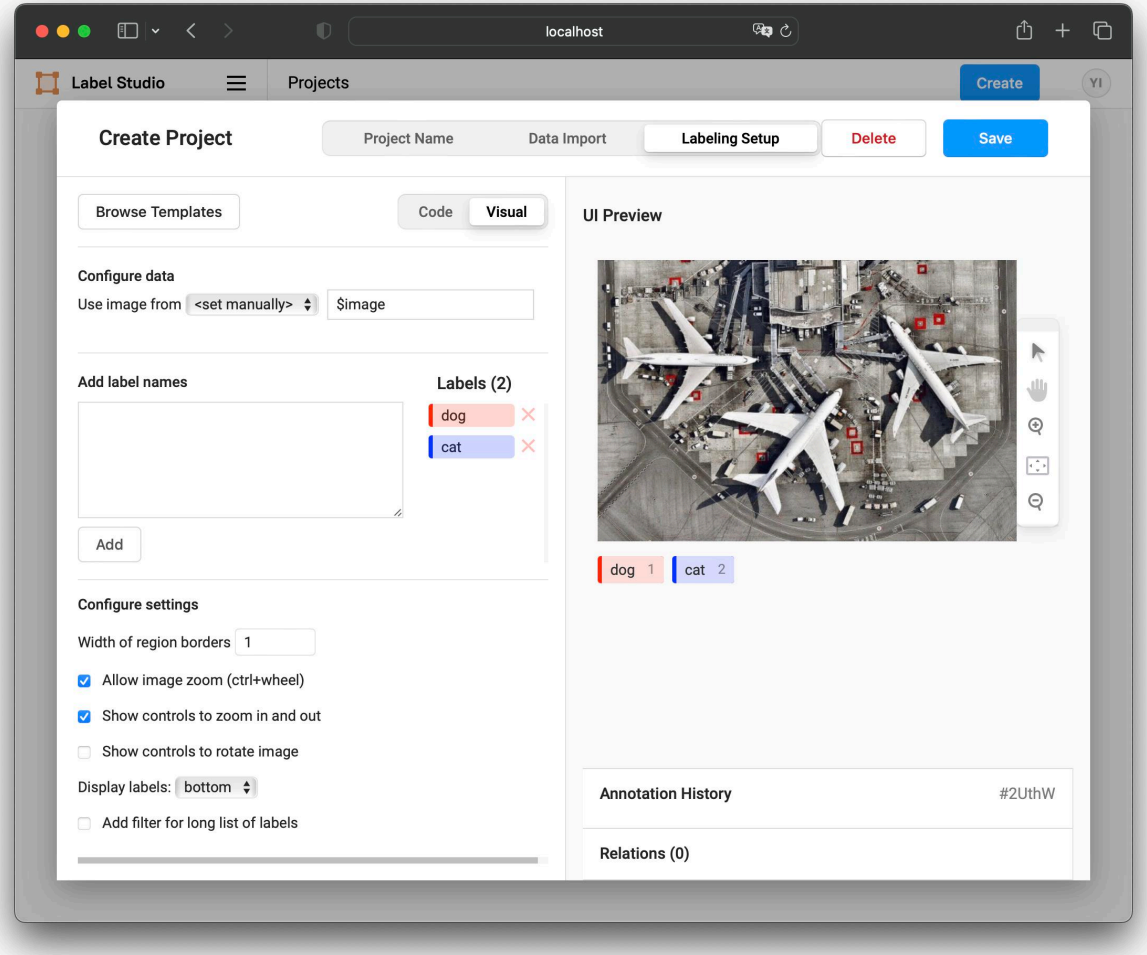
# Step 9

- Select `Object Detection with Bounding Boxes`



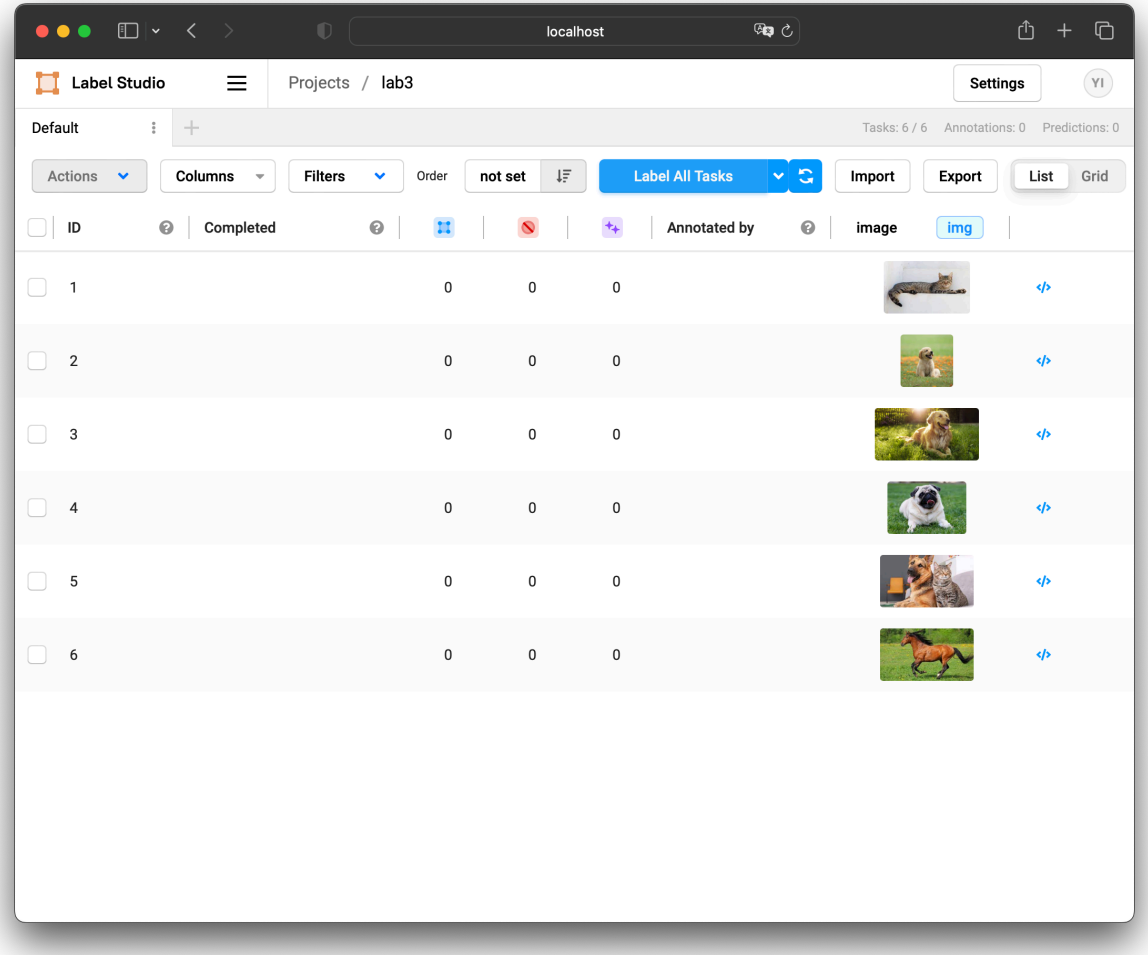
# Step 10

- Add dog and cat to label names. Note that the colors can be customized.
- Save.



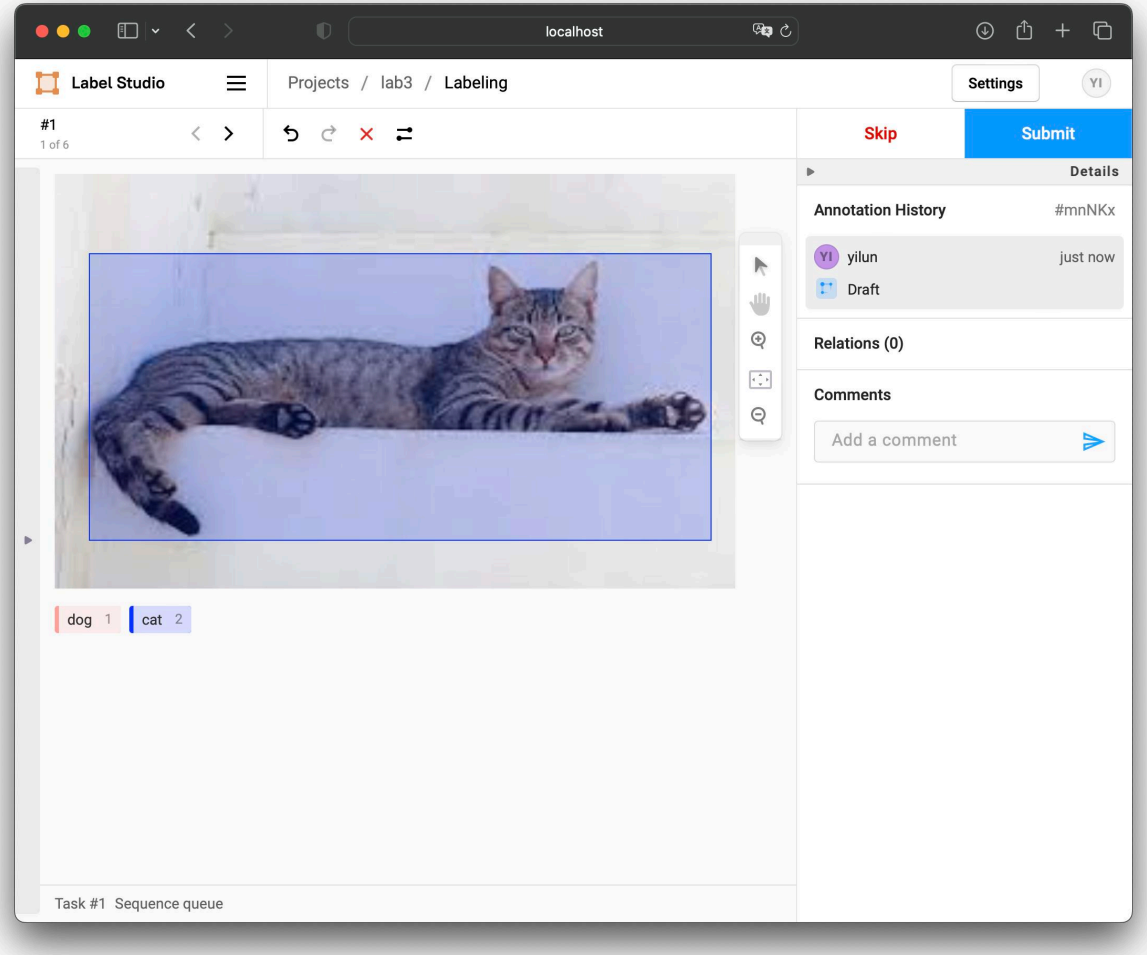
# Step 11

- Use the `Label All Tasks` button to start/resume labeling.



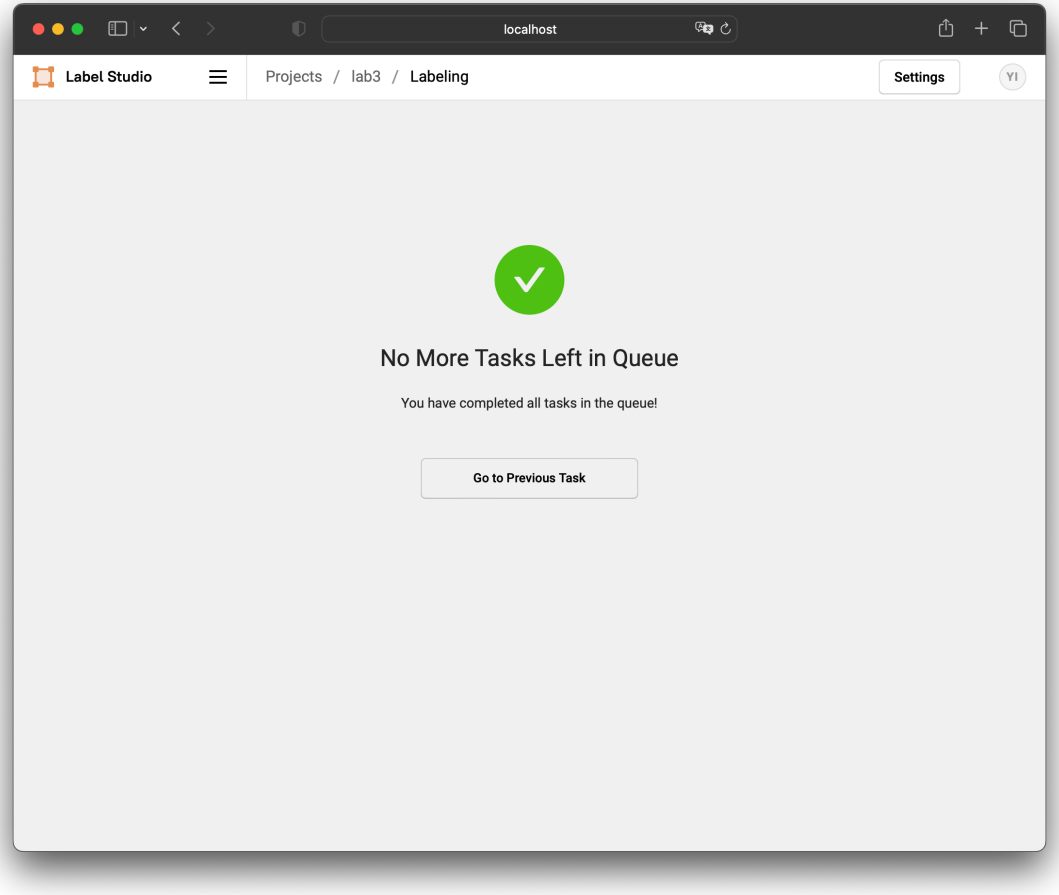
# Step 12

- Select a label and drag the bounding box
- Using `Submit` to save the labels



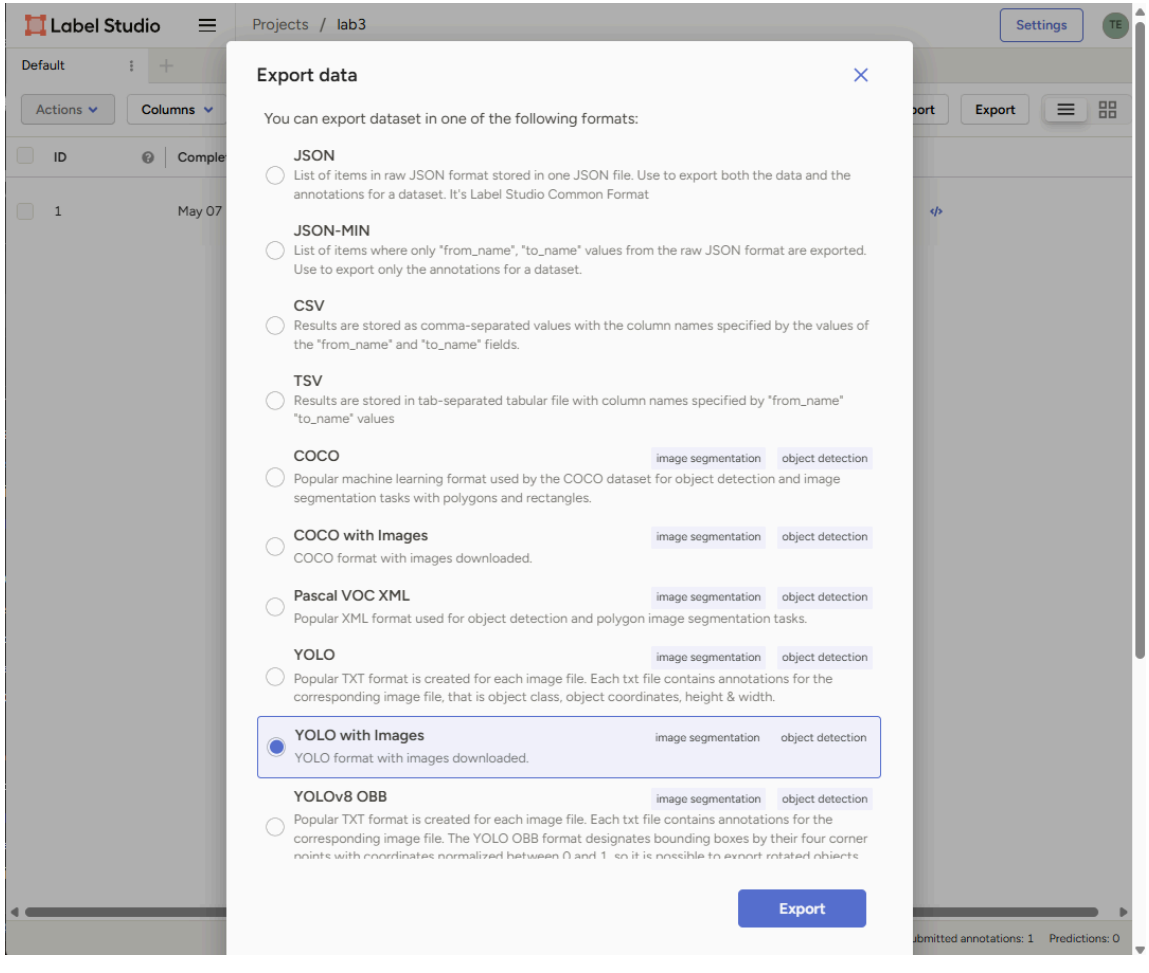
# Step 13

- Finished!



# Step 14

- Use the `Export` button to download labels and images.
- Select the **YOLO with Images** format, then export.



# Transmit Dataset To Server

- On your computer, use `scp` to copy the data via the `ssh` protocol to the server.

```
$ scp -P 10122 \
    {path/to/file.zip} \
    {student_id}@140.113.160.149:~/hcc-ml/lab3
```

  For example,

```
$ scp -P 10122 \
    ~/Downloads/project-4-at-2023-03-23-15-21-316bdd45.zip \
    123456@140.113.160.149:~/hcc-ml/lab3
```

# Unzip the Dataset on the Server

- Unzip the dataset in the folder `hcc-ml/lab-3` on the server:

```
$ unzip {path/to/file.zip} -d dogcat
```

- For example,

```
$ unzip project-4-at-2023-03-23-15-21-316bdd45.zip -d dogcat
```

# Create a customized labeled file

- Create a YAML file called `dogcat.yaml` under the `data` folder:

```yaml
# image root
train: ./dogcat/images
val: ./dogcat/images

# number of classes
nc: 2

# class names
names: ['cat', 'dog']
```

- Important: The order of `names` should follow the `dogcat/classes.txt`.

# Create a customized model file

- Copy the file `cfg/training/yolov7-w6.yaml` to `cfg/training/dogcat-custom.yaml`

```
cp cfg/training/yolov7-w6.yaml cfg/training/dogcat-custom.yaml
```

- Modify the number of classes in `cfg/training/dogcat-custom.yaml`

```
nc: 80 -> nc: 2
```

# Friendly Tips

- Use the `tmux` tool to start a new session window.

```
$ tmux new
```

- Run any long-running command in that session.
- Leave the session without interrupting the program by using the shortcut key:

```
push CTRL -> push B -> release CTRL and B -> push D -> release D
```

- Attach back to previously created session by using command

```
$ tmux attach
```

- Leave and delete the session by using the shortcut key:

```
push CTRL -> push D
```

- Cheat sheet

# Start Training

- To start training, run the following command:

```
python train.py \
    --project dogcat \
    --epochs 50 \
    --device 0 \
    --batch-size 2 \
    --img 640 640 \
    --data data/dogcat.yaml \
    --cfg cfg/training/dogcat-custom.yaml \
    --weights weights/yolov7-w6.pt \
    --hyp data/hyp.scratch.custom.yaml
```

- Users can find weights and training info under `dogcat/exp`

# Model Zoo

| Model | Test Size | $AP^{test}$ | $AP^{test}_{0.5}$ | $AP^{test}_{0.75}$ | batch 1 fps | batch 32 average time |
|---|---|---|---|---|---|---|
| YOLOv7-W6 | 1280 | **54.9%** | **72.6%** | **60.1%** | 84 *fps* | 7.6 *ms* |
| YOLOv7-E6 | 1280 | **56.0%** | **73.5%** | **61.2%** | 56 *fps* | 12.3 *ms* |
| YOLOv7-D6 | 1280 | **56.6%** | **74.0%** | **61.8%** | 44 *fps* | 15.0 *ms* |
| YOLOv7-E6E | 1280 | **56.8%** | **74.4%** | **62.1%** | 36 *fps* | 18.7 *ms* |

# Checkpoint 3.

- After finishing the training, test the trained model with your images using the following command:

```
python demo.py \
    --image dogcat/images/73770b2b-img4.jpg  \
    --weights dogcat/exp/weights/best.pt \
    --conf_threshold 0.1
```

  Replace `73770b2b-img4` with your own filename and select a proper threshold.
- Show `demo.png` which should contain some reasonable bounding boxes.
-

# Friendly Tips Again

- Use nvtop to inspect the GPU and GPU memory usage of a server node.
- Use htop to inspect the CPU and system memory usage of a server node.
- When you log in, the login banner will show all GPU usage of this cluster (updated every 30 seconds).
- Each team can only use one GPU at a time.
- The user of GPU can be found in the login banner or using the command nvtop.
- If you find any team using more than one GPU, please report it to me (bkruan.ee11@nycu.edu.tw), and I will terminate their all process immediately.

# The End