BIT SPACE DEVELOPMENT

Northeastern University

CONSTRUCTION SAFETY

Open hole / Unsafe Excavation Detection

Van Kieu Dang

Northeastern University

ALY6890: Capstone Project

CONTENT & DELIVERABLES

OBJECTIVES

Open hole / unsafe excavation detection

This model combined with PPE detection model will be deployed to our Jetson-based edge inference systems for job sites allowing us to monitor and report on unsafe conditions surrounding excavation.

Knowledge acquisition

- Pipeline, workflow
- Object detection
- Automation
- Tools and Platforms

CONTENT

I. Insight Exploration

1. Iterative workflow

2.1 Data collection | 2.2 Annotation | 2.3 Pre-processing

3. Model training

II. Key Results

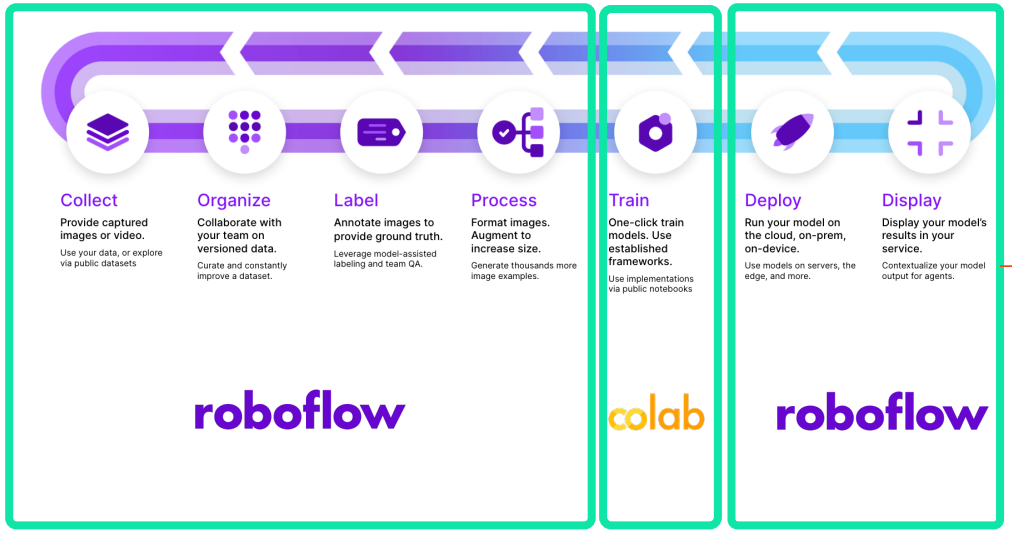
4. Model Performance Comparison

5. Recommendations

III. Challenges and Progress

I. KEY INSIGHTS

1. Iterative Workflow



1. Identify **misclassifications** (FP, FN) and makes necessary improvements.

2. Update **hyper-parameters** used.

```
# Create the Trainer
trainer = Trainer(
    accelerator="gpu",                          # Use GPU acceleration
    max_epochs=200,                              # Train for a maximum of 50 epochs
    precision='32-true',                        # Use 32-bit floating point precision
    callbacks=[early_stopping_callback],        # Early stopping callback
    accumulate_grad_batches=16,                 # Accumulate gradients over 16 batches before
    gradient_clip_val=0.5,                      # Clip gradients if their norm exceeds 0.5
    log_every_n_steps=5,                        # Set frequency of logging training metrics
    enable_model_summary=True                   # Enable model summary
)
```

3. **Adjust** settings, **revisit** data augmentation methods, or **incorporate** additional labeled data.

I. KEY INSIGHTS

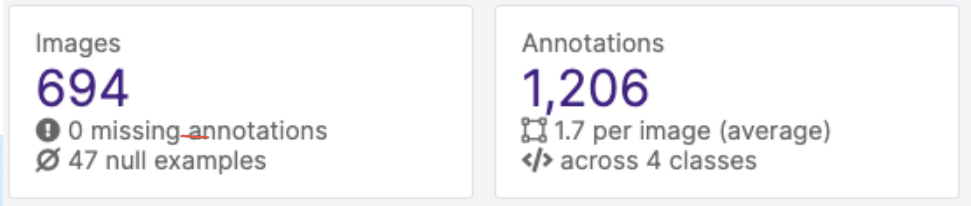
2.1. Data Collection

Various scenarios that models can generalize effectively.

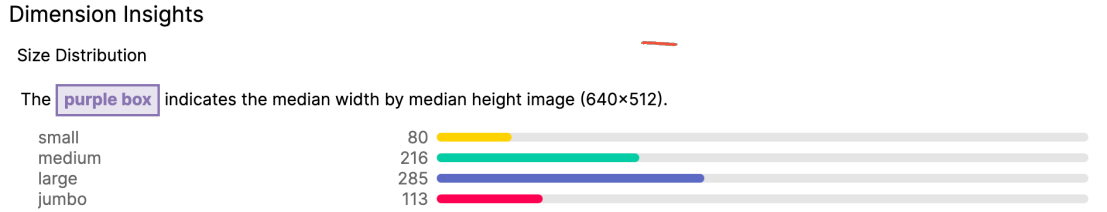
Diverse and representative

Balanced class distribution

Negative mining / augmentation



Leverage Health check - Dimension



I. KEY INSIGHTS

2.2. Data Annotation

Insights to improve Consistency and Data quality

Multi-scale and complex annotation

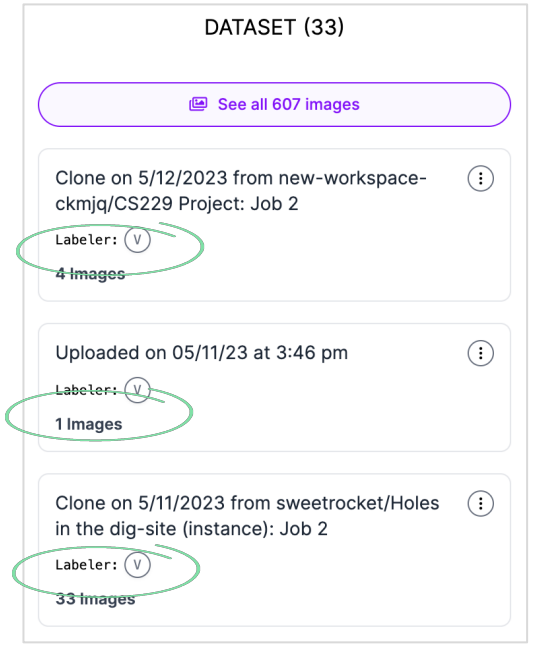
Large

Medium

Small

Annotator guidelines

Only a few members are mapping annotations for consistency of data, strictly following guidelines.



Smart Annotation

Leverage smart annotation from Roboflow to boost data preparation progress.

I. KEY INSIGHTS

2.3. Data Preprocessing

To increase dataset size and prepare data for training

No black and white

No perspective transformation

No random rotati

No edge detection (unnecessary)

I. KEY INSIGHTS

3. Model Training

When it comes to model training, various aspects we can take into account such as hyper-parameters, pre-trained models, optimizers, and so on. In this part, we only highlight some insights from model tuning:

YOLO

- **img-size:** Define input image size, (default value: [640, 640]).
- **workers:** Define maximum number of dataloader workers (default value: 8).
- **batch-size:** Determine batch size.
- **epochs:** Define the number of training epochs.
- **data:** Our dataset locaiton.
- **weights:** Specify a path to weights to start transfer learning from. Here we choose the generic COCO pretrained checkpoint from Yolov7.
- **cache-image:** Cache images for faster training.
- **device:** cuda device, i.e. 0 or 0,1,2,3 or cpu.
- **p (patience):** early stop hyper-parameter (Model will stop after N epochs without improvement). In this case, N =20.

```
# Training the model
%cd /content/yolov7
!python train.py --img-size 640 --batch-size 16 --workers 8 --epochs 200 --device 0 --data {dataset.location}/data.yaml
```

DETR

```
# Define the EarlyStopping callback
early_stopping_callback = EarlyStopping(
    monitor='validation/loss',
    mode='min',
    patience=20.         # Update the number of epochs for early stop
)
# Set model
model = Detr(lr=1e-4, lr_backbone=1e-5, weight_decay=1e-4)
batch = next(iter(TRAIN_DATALOADER))
outputs = model(pixel_values=batch['pixel_values'], pixel_mask=batch['pixel_mask'])

# Create the Trainer
trainer = Trainer(
    accelerator="gpu",                          # Use GPU acceleration
    max_epochs=200,                             # Train for a maximum of 200 epochs
    precision='32-true',                        # Use 32-bit floating point precision
    callbacks=[early_stopping_callback],        # Early stopping callback
    accumulate_grad_batches=16,                 # Accumulate gradients over 16 batches before optimizer step
    gradient_clip_val=0.5,                      # Clip gradients if their norm exceeds 0.5
    log_every_n_steps=5,                        # Set frequency of logging training metrics
    enable_model_summary=True                   # Enable model summary
)

# Start training
trainer.fit(model)
```

Optimization

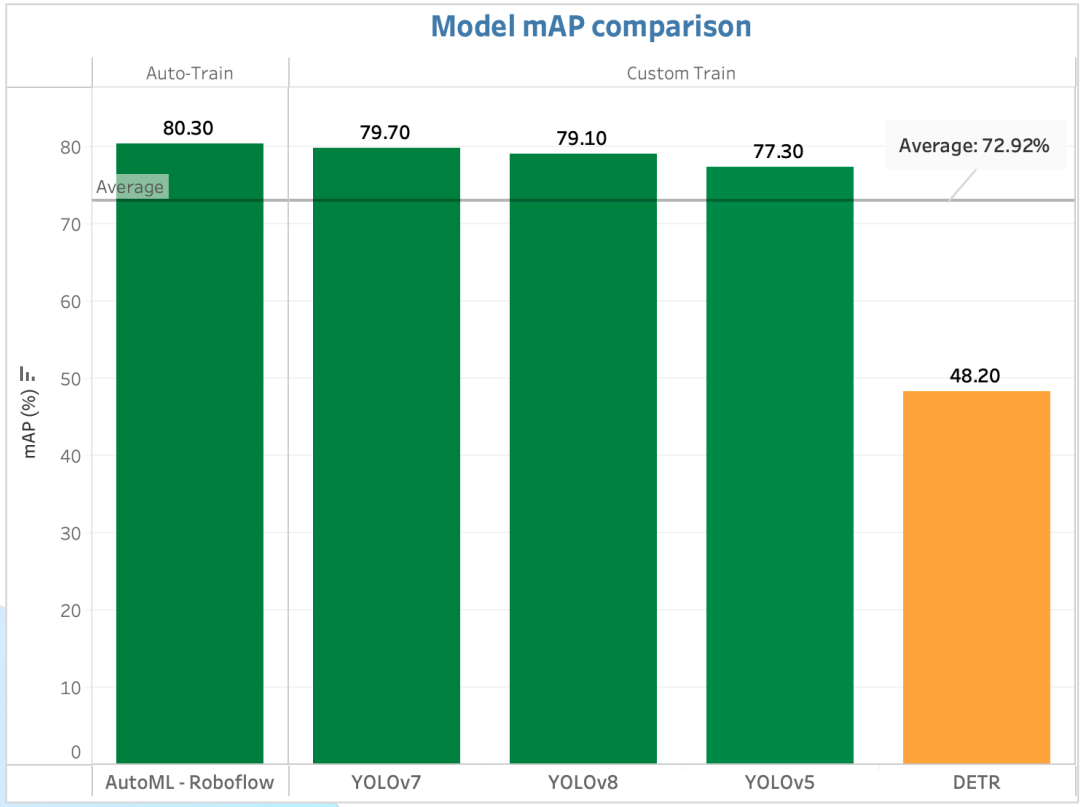
- ❑ Choose a small **learning rate**: 0.001 / 1e-4.
- ❑ More **epochs**.
- ❑ Set **early stop.**
- ❑ Medium **batch size.**
- ❑ Set '**cache**' for faster training.
- ❑ Apply **Weight decay** to prevent overfitting.
- ❑ **Unfreeze** up to n-1 layers to train.
- ❑ Use **Non-max suppression.**
- ❑ …

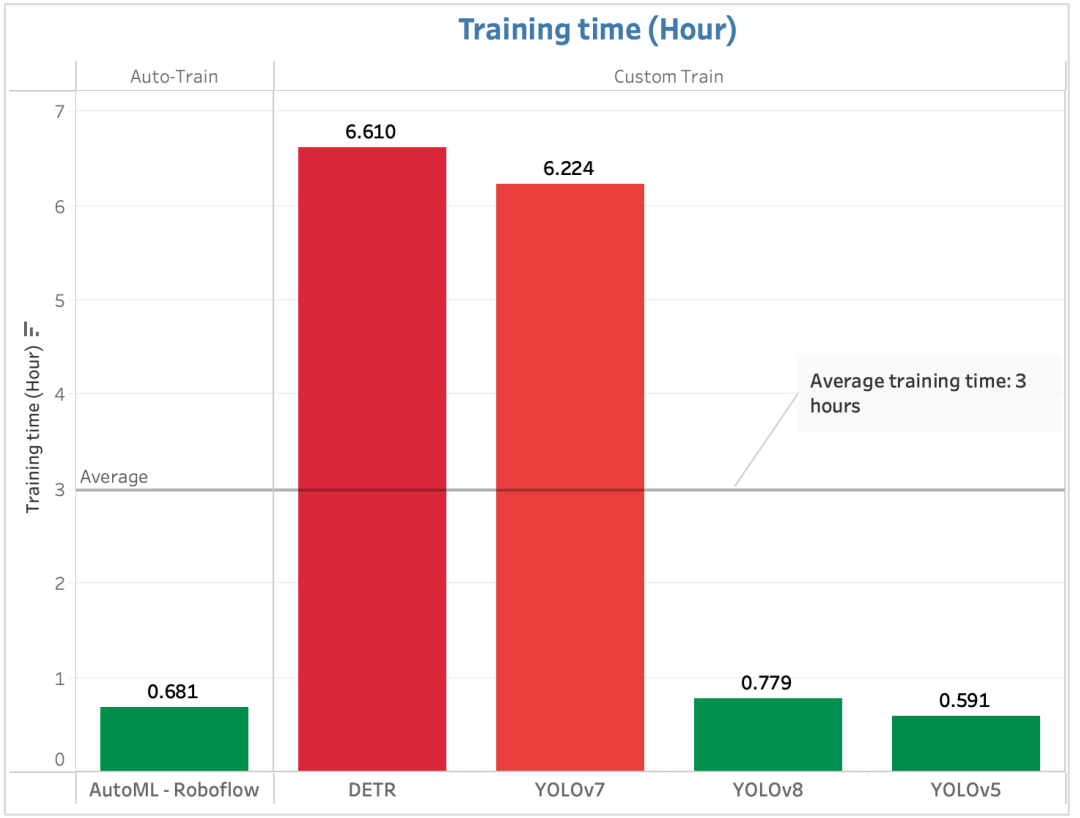
II. KEY RESULTS

5. Comparison

Performance for each model is displayed in detail on respective Tensorboard from our notebooks.

Note: Results are derived from the same dataset, same hardware, and same training set up such as: 200 epochs, patience of 20, batch size of 16,…





II. KEY RESULTS

6. Recommendations

1

No	Category	Model	mAP (%)	Speed / Training time (Hour)	Model size (MB)	Framework Compatibility
1	Custom train	YOLOv5	77.3	0.591	14.5	Darknet, OpenCV, Pytorch, Tensorflow, Keras…
2	Custom train	YOLOv7	79.7	6.224	74.8	Darknet, OpenCV, Pytorch, Tensorflow, Keras…
3	Custom train	YOLOv8	79.1	0.779	22.5	Darknet, OpenCV, Pytorch, Tensorflow, Keras…
4	Custom train	DETR	48.2	6.61	166.6	Pytorch
5	Auto train	AutoML - Roboflow	80.3	0.681	N/A	Variety

2 Focus more on Data than Modeling after this phase.

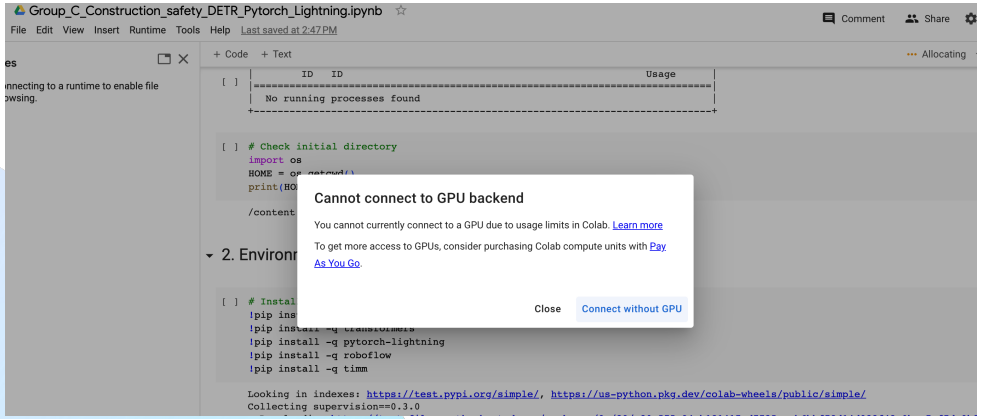
III. SUMMARY

CHALLENGES

- ❑ Open ended project without target images and available dataset.
- ❑ Target objects are hard to detect.

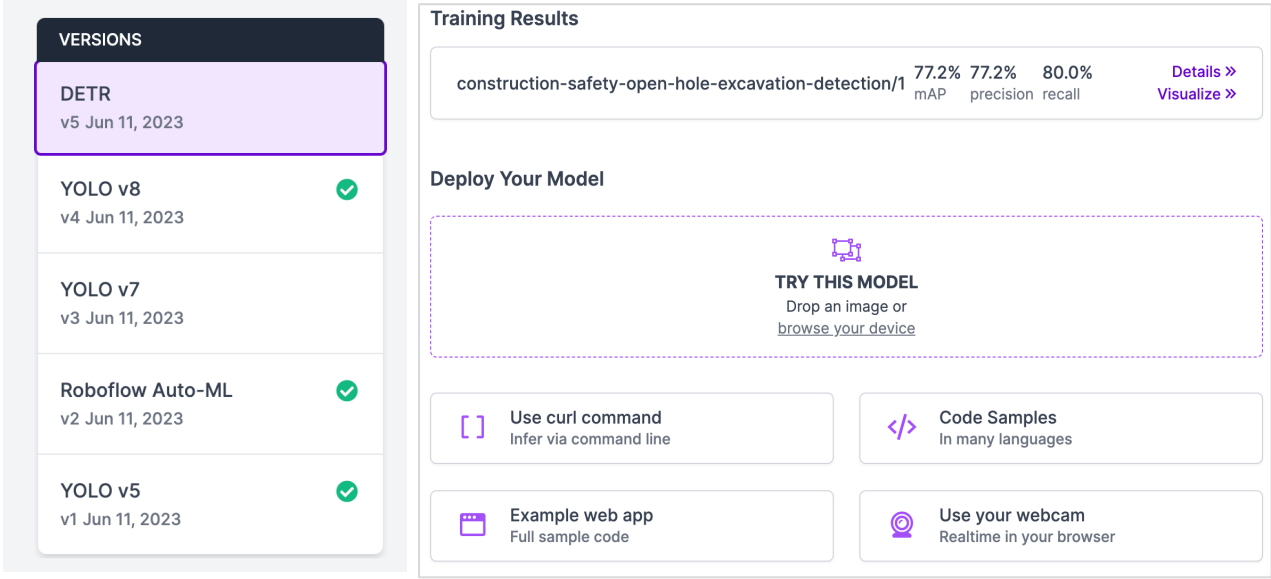
No standards, different size, different shapes,…

- ❑ Limitation of hardware: Google Colab free tier



PROGRESS

- ❑ **Data** seem to be the most critical part of model performance.
- ❑ Acquire **Pipeline, Automation, Tools (Roboflow + Colab) and Object Detection** techniques as well as best practices from the company.
- ❑ Implemented successfully **several models**



- ❑ Models under development: **Mask RCNN** and **Unet.**

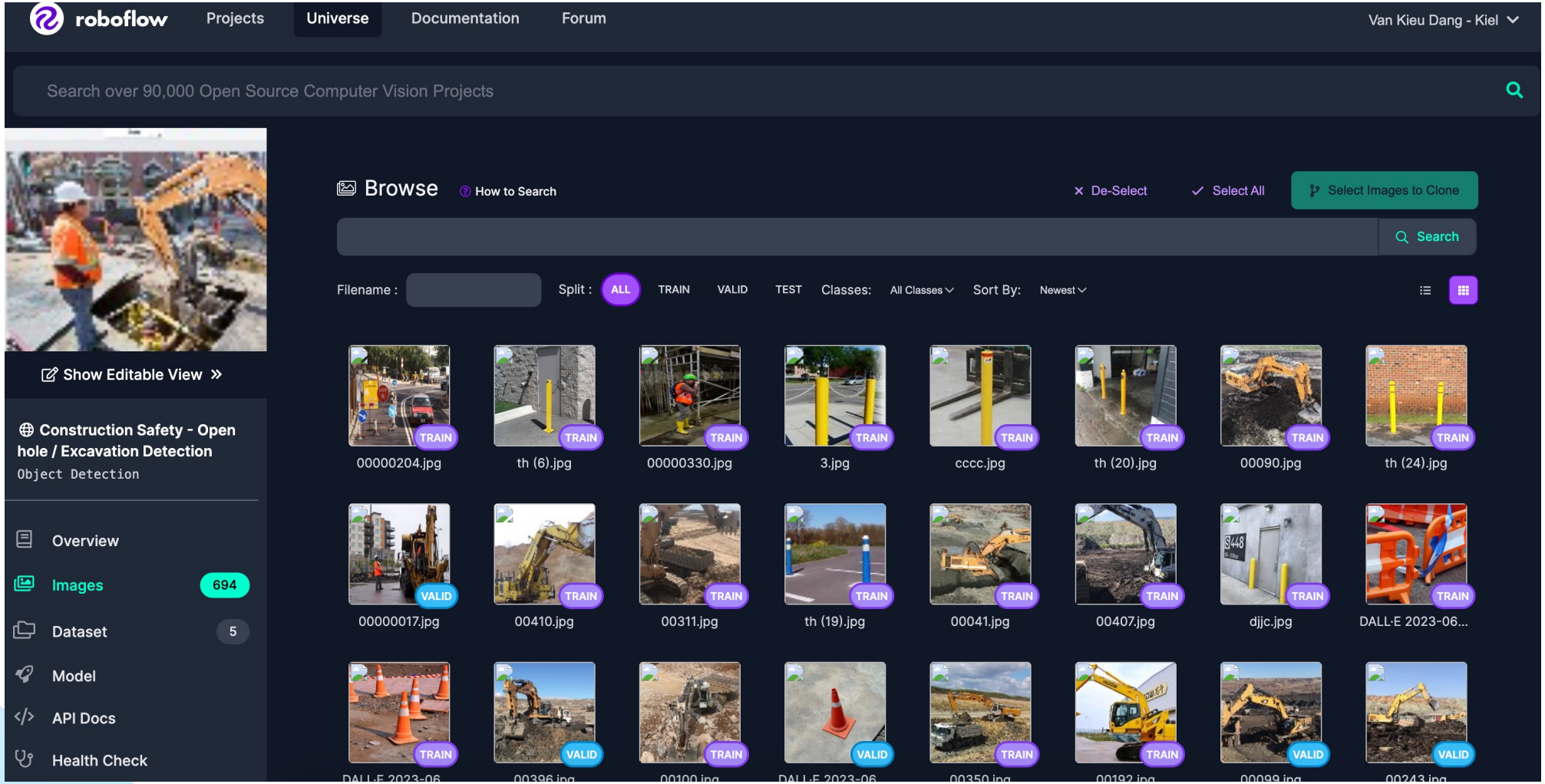
BIT SPACE DEVELOPMENT

CONSTRUCTION SAFETY

THANK YOU

BIT SPACE DEVELOPMENT

Appendix - Dataset



Link to the project: https://universe.roboflow.com/northeastern-4sfxe/construction-safety-open-hole-excavation-detection

BIT SPACE DEVELOPMENT

Appendix – Model Versions

Construction Safety - Open hole / Excavation Detection Image Dataset

Generate New Version

VERSIONS

- DETR
  v5 Jun 11, 2023
- YOLO v8
  v4 Jun 11, 2023
- YOLO v7
  v3 Jun 11, 2023
- Roboflow Auto-ML
  v2 Jun 11, 2023
- YOLO v5
  v1 Jun 11, 2023

YOLO v5

Version 1 Generated Jun 11, 2023

Export Dataset | Edit

ROBOFLOW TRAIN

MODEL TYPE: YOLOV5 MODEL UPLOAD

Training Results

construction-safety-open-hole-excavation-detection/1	77.2% mAP	77.2% precision	80.0% recall	Details » Visualize »

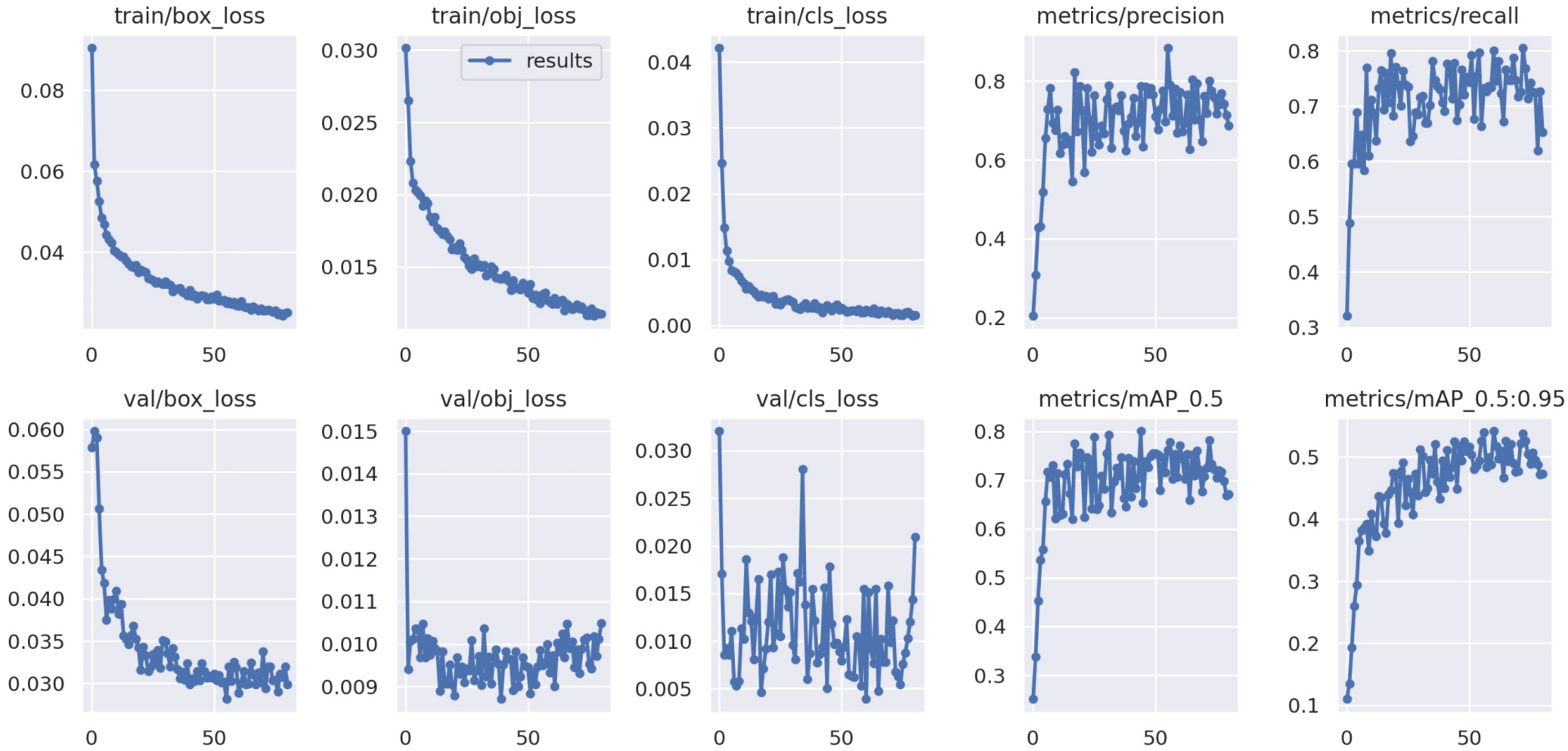
Deploy Your Model

TRY THIS MODEL

Drop an image or
browse your device

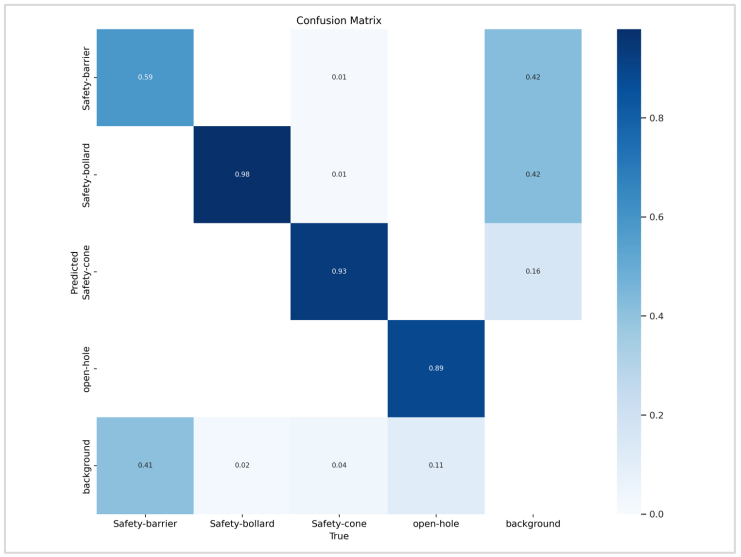
BIT SPACE DEVELOPMENT

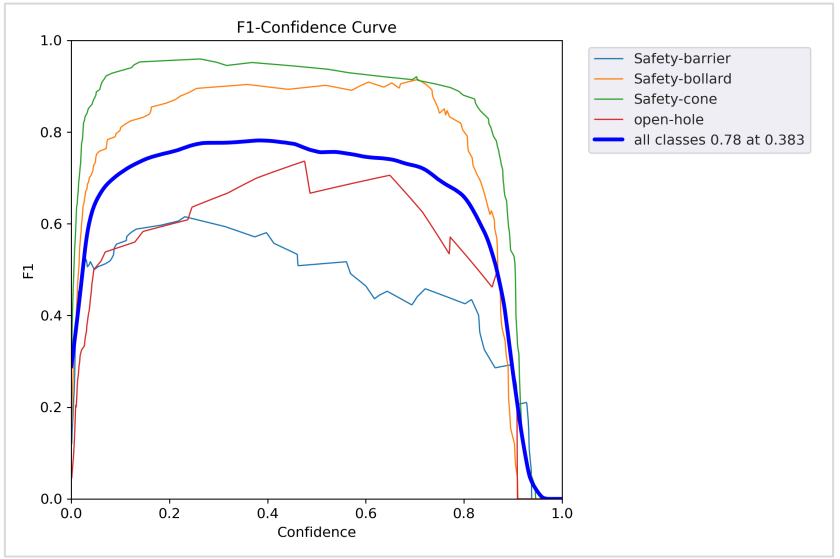
Appendix – YOLO v5 Performance Details

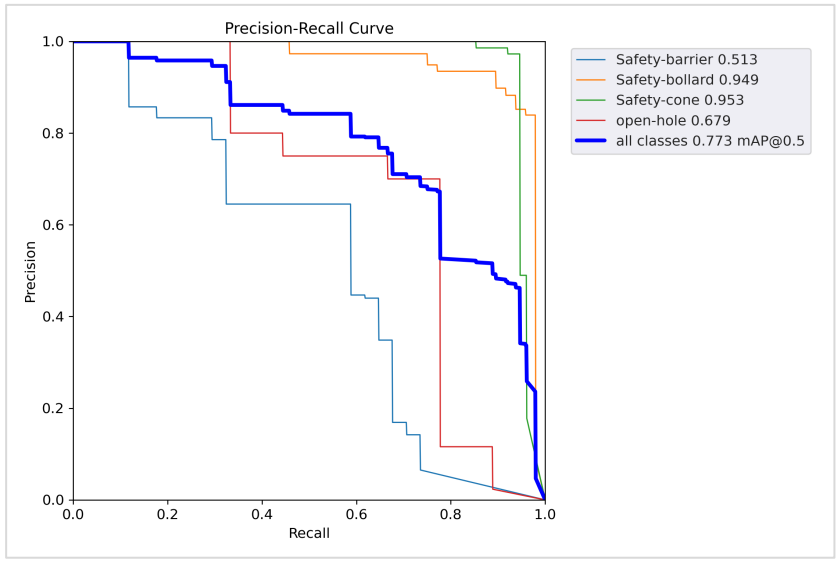


BIT SPACE DEVELOPMENT

Appendix – YOLO v5 Performance Details







Ground truth

Prediction