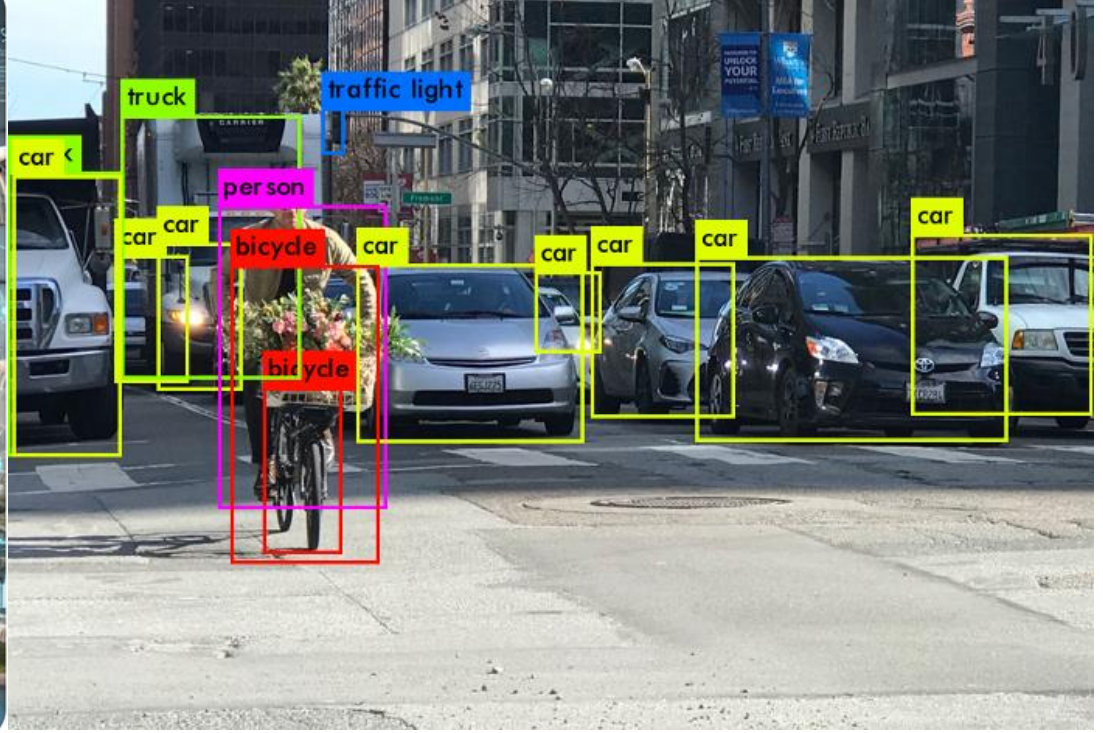
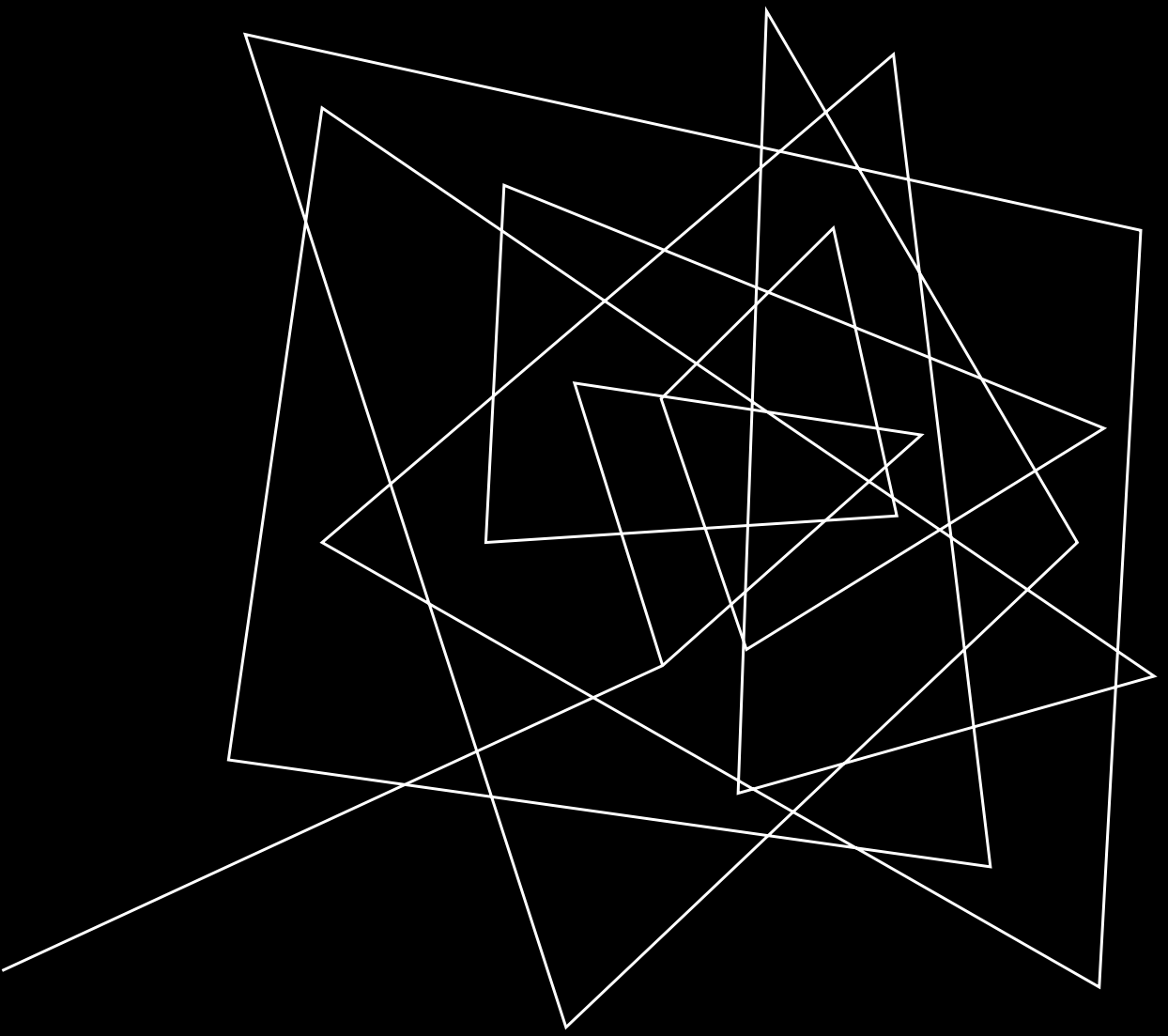


- **PROJECT TITLE:** AI-ENABLED DRONE SWARM NETWORK AND INFORMATION MANAGEMENT
  SUBTITLE: OBJECT DETECTION, DRONE2DRONE INFORMATION EXCHANGE AND COLLABORATIVE AWARENESS



CREATING DATA SETS – ANNOTATION

WHAT YOU NEED TO GET STARTED

- Roboflow account (free or paid)
- Dataset of images
- Understanding of object detection models
- Basic knowledge of bounding boxes and labeling principles

SETTING UP A NEW PROJECT IN ROBOFLOW

1. Log in to Roboflow
2. Click on **Create a New Project**
3. Enter project name, select **Object Detection**
4. Choose a dataset version (e.g., public, private)
5. Click **Create Project**

UPLOADING IMAGES

1. Click **Upload** to add images
2. Drag and drop images or select from local storage
3. Verify images and click **Continue**
4. Choose preprocessing options (optional)

LABELING IMAGES- ANNOTATING OBJECTS IN ROBOFLOW

1. Click on an image to open the annotation tool
2. Select **Bounding Box** tool
3. Draw boxes around objects
4. Assign class labels
5. Save annotations and move to the next image

LABELING BEST PRACTICES

- Ensure bounding boxes are tight around objects
- Avoid overlapping labels when unnecessary
- Use consistent naming conventions
- Label only objects relevant to the detection model

REVIEWING AND MANAGING LABELS – QUALITY CONTROL FOR ANNOTATIONS

- Use the **Review** feature to check labels
- Edit or delete incorrect annotations
- Ensure class balance across dataset
- Collaborate with team members for annotation consistency

AUGMENTING AND EXPORTING DATASET – PREPARING DATA FOR TRAINING

1. Apply data augmentation (flipping, rotation, brightness adjustments)
2. Click **Generate** to create a dataset version
3. Choose export format (YOLOv5, COCO, Pascal VOC, etc.)
4. Download dataset for training

USING THE ANNOTATED DATASET – TRAINING AN OBJECT DETECTION MODEL

- Import dataset into model training frameworks (YOLO, TensorFlow, etc.)
- Train a model using labeled data
- Evaluate performance and refine annotations if necessary

RAINING AN OBJECT DETECTION MODEL WITH YOLOV5

PREREQUISITES

- Basic knowledge of Python and deep learning
- Installed software: Python, pip, Git
- Required libraries: PyTorch, OpenCV, Matplotlib, Pandas, NumPy
- GPU recommended for faster training

SETTING UP YOLOV5 - EXAMPLE

Open a terminal and run:

```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
pip install -r requirements.txt
```

Verify installation:

```
python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source
data/images/S
```

DATASET PREPARATION- EXAMPLE

Use an existing dataset (COCO, Pascal VOC) or create a custom dataset
Label images using Roboflow
Organize dataset into:
    images/train/, images/val/, images/test/
    labels/train/, labels/val/, labels/test/

Create a data.yaml file:
    train: ./data/images/train
    val: ./data/images/val
    nc: 1 # number of classes
    names: ['object']S

TRAINING THE MODEL

Run the training command:

python train.py --img 640 --batch 16 --epochs 50 --data data.yaml --weights yolov5s.pt --device 0

Monitor training progress in runs/train/exp/

Adjust hyperparameters for better accuracy

EVALUATING MODEL PERFORMANCE

Check runs/train/exp/ for metrics like mAP (mean Average Precision)

Use TensorBoard for visual analysis:

```
tensorboard --logdir runs/train
```

Test on new images:

```
python detect.py --weights runs/train/exp/weights/best.pt --img 640 --conf 0.25 --source data/images/test/
```

FINE-TUNING THE MODEL - OPTIMIZING FOR BETTER RESULTS

- Increase dataset size
- Use data augmentation
- Tune hyperparameters: learning rate, batch size, epochs
- Try different YOLO versions (YOLOv5m, YOLOv5l, etc.)

DEPLOYING THE TRAINED MODEL

- Convert model to ONNX or TensorRT for efficiency
- Deploy using Flask or FastAPI for web applications
- Implement on edge devices like Raspberry Pi or Jetson Nano