perplexity

Rubik's Cube Color Detection Model Training Documentation

This document outlines the process of training a YOLO-based object detection model to identify the colors of squares on a Rubik's Cube. It includes steps for dataset preparation, model training, integration with a Raspberry Pi-powered DOFBOT robotic arm, and evaluation.

1. Working with Sample Datasets on Roboflow

Dataset Preparation

- **Dataset Collection**: Gather images of Rubik's Cubes in various orientations and lighting conditions. Ensure that all six colors (red, blue, yellow, green, orange, white) are represented.
- **Annotation**: Use Roboflow's annotation tool to label each square with its corresponding color. YOLO annotations require bounding boxes around objects and labels in the format: `<class-id> <x_center> <y_center> <width> <height>` (normalized to image dimensions)[1] [2].
- **Exporting Annotations**: Export the dataset in YOLO format for compatibility with YOLO training frameworks.

Preprocessing

- **Auto-Orient**: Automatically adjusts image orientation based on metadata to ensure consistency during training[3] [4].
- **Resize**: Standardize image dimensions to
$$640 \times 640$$
pixels for YOLO input requirements. This ensures uniformity without distortion[5] [3].

2. Incorporating Base Sample into DOFBOT Robotic Arm with Camera

Steps to Generate Base Sample Dataset

1. **Capture Images**: Use the DOFBOT robotic arm's camera to capture images of a Rubik's Cube.
2. **Labels and Classes**: Define four initial color labels (e.g., red, blue, green, yellow) for simplicity.
3. **Prepare Dataset**:

- Annotate images using Roboflow or LabelImg.
- Split the dataset into training (70%), validation (20%), and testing (10%) subsets[6].

Data Transformations

- **Resizing Images**: Adjust all images to 640×640 pixels to match YOLO model requirements.
  - **Auto-Orient**: Correct image orientation based on metadata to ensure consistent alignment during training and inference[3] [4].

3. Creating Model Training Code and Integration

Training Code

The following Python code snippet demonstrates how to train a YOLOv4 model using the annotated dataset:

```
!python train.py \
--data data.yaml \
--cfg yolov4.cfg \
--weights yolov4.conv.137 \
--epochs 100 \
--img-size 640
```

- `data.yaml`: Specifies paths to training and validation datasets.
- `cfg`: Configuration file for YOLOv4.
- `weights`: Pretrained weights for transfer learning.

Integration with DOFBOT Camera

1. Install necessary libraries:
   ```
   pip install opencv-python roboflow
   ```
2. Modify the DOFBOT camera script to feed live video frames into the trained YOLO model:
   ```
   import cv2
   from yolov4 import Detector

   detector = Detector(weights="best.pt", config="yolov4.cfg", classes="classes.txt")

   cap = cv2.VideoCapture(0)
   while True:
       ret, frame = cap.read()
       detections = detector.detect(frame)
       # Display detections on frame
       cv2.imshow("Rubik's Cube Detection", frame)
       if cv2.waitKey(1) & 0xFF == ord('q'):
   ```

```
        break
cap.release()
cv2.destroyAllWindows()
```

4. Evaluating the Model

Before Adding Custom Data

- Evaluate performance using mAP (mean Average Precision), precision, and recall metrics[7].
- Use test data from the initial dataset split to validate model accuracy.

After Adding Custom Data

- Retrain the model with additional images captured by the DOFBOT camera.
- Compare mAP scores before and after retraining to assess improvement.

5. Next Steps

1. **Dataset Expansion**:
   - Collect more diverse images of Rubik's Cubes under different lighting conditions and angles.
   - Add annotations for all six colors.
2. **Model Optimization**:
   - Experiment with smaller models like YOLOv4-Tiny for faster inference on edge devices like Raspberry Pi[8] [9].
   - Fine-tune hyperparameters such as learning rate and batch size.
3. **Deployment**:
   - Deploy the trained model on the Raspberry Pi-powered DOFBOT robotic arm using OpenCV or Roboflow Inference API[10] [11].
4. **Real-Time Applications**:
   - Integrate with robotic arm control algorithms to manipulate the Rubik's Cube based on detected colors.
5. **Advanced Features**:
   - Explore depth sensing or 3D vision for more complex tasks like cube-solving automation[12].

By following these steps, you can successfully train a YOLO-based object detection model for Rubik's Cube color detection and integrate it into a robotic system for real-time applications.

⁂

1. https://www.basic.ai/blog-post/data-annotation-for-yolo-model-training-techniques-and-best-practices-1
2. https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/
3. https://thelinuxexperiment.com/resizing-and-correctly-orienting-images-with-mogrify/
4. https://www.alibabacloud.com/help/en/oss/user-guide/auto-rotate-4
5. https://pplx-res.cloudinary.com/image/upload/v1741369884/user_uploads/GwVxDwWXoPngFSS/Screenshot-2025-03-07-at-11.48.13.jpg
6. https://blog.roboflow.com/getting-started-with-roboflow/
7. https://stackoverflow.com/questions/71265841/how-to-validate-my-yolo-model-trained-on-custom-data-set
8. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/1998337/bf4a7845-a09f-471d-8149-801536f7723b/5.Model-training.pdf
9. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_676e852f-6943-4444-a023-28f1e49351dd/cdd08231-0739-4996-b6e0-cc5427a206a9/5.Model-training.pdf
10. https://roboflow.com/model/yolo-world
11. https://thinkrobotics.com/products/dofbot-ai-vision-robotic-arm-with-ros-for-raspberry-pi-4b-online
12. https://www.youtube.com/watch?v=KYPFQ0erNjw