

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: <cl ass- i d> <x\_cent er > <y\_cent er > <wi dt h> <hei ght > (normalized to image dimensions)12.
* 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 34.
  + Resize: Standardize image dimensions to



pixels for YOLO input requirements. This ensures uniformity without distortion 53.

1. Incorporating Base Sample into DOFBOT Robotic Arm with Camera Steps to Generate Base Sample Dataset

Capture Images: Use the DOFBOT robotic arm's camera to capture images of a Rubik's

Cube.

Labels and Classes: Define four initial color labels (e.g., red, blue, green, yellow) for

simplicity.

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



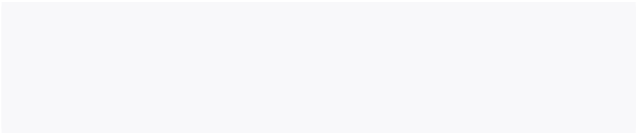
pixels to match YOLO model requirements.

* Auto-Orient: Correct image orientation based on metadata to ensure consistent alignment during training and inference 34.

1. Creating Model Training Code and Integration Training Code

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

! pyt hon t r ai n. py \



* - dat a dat a. yaml \
* - cf g yol ov4. cf g \
* - wei ght s yol ov4. conv. 137 \
* - epochs 100 \
* - i mg- si ze 640
* dat a. yaml : Specifies paths to training and validation datasets.
* cf g: Configuration file for YOLOv4.
* wei ght s: Pretrained weights for transfer learning.

Integration with DOFBOT Camera

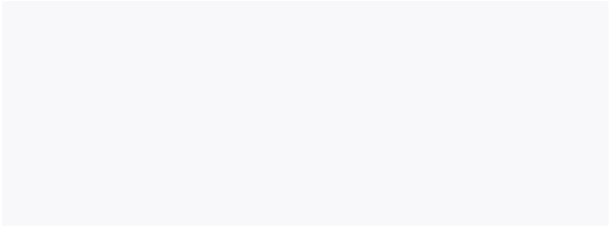
Install necessary libraries:

pi p i nst al l opencv- pyt hon r obof l ow



Modify the DOFBOT camera script to feed live video frames into the trained YOLO model:

i mpor t cv2



f r om yol ov4 i mpor t Det ect or

det ect or = Det ect or ( wei ght s="best . pt ", conf i g="yol ov4. cf g", cl asses="cl asses. t xt ")

cap = cv2. Vi deoCapt ur e( 0)

whi l e Tr ue:

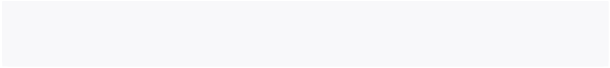
r et , f r ame = cap. r ead( )

det ect i ons = det ect or . det ect ( f r ame)

# Di spl ay det ect i ons on f r ame

cv2. i mshow( "Rubi k' s Cube Det ect i on", f r ame) i f cv2. wai t Key( 1) & 0xFF == or d( ' q' ) :

br eak



cap. r el ease( )

cv2. dest r oyAl l Wi ndows( )

1. 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.

1. Next Steps

Dataset Expansion:

* Collect more diverse images of Rubik's Cubes under different lighting conditions and angles.
  + Add annotations for all six colors.

Model Optimization:

* Experiment with smaller models like YOLOv4Tiny for faster inference on edge devices like Raspberry Pi89.
  + Fine-tune hyperparameters such as learning rate and batch size.

Deployment:

Deploy the trained model on the Raspberry Pi-powered DOFBOT robotic arm using OpenCV or Roboflow Inference API1011.



Real-Time Applications:

Integrate with robotic arm control algorithms to manipulate the Rubik's Cube based on detected colors.



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.

⁂

https://www.basic.ai/blog-post/data-annotation-for-yolo-model-training-techniques-and-best-practice

s-1

https://docs.ultralytics.com/yolov5/tutorials/train\_custom\_data/ https://thelinuxexperiment.com/resizing-and-correctly-orienting-images-with-mogrify/ https://www.alibabacloud.com/help/en/oss/user-guide/auto-rotate-4

https://pplx-res.cloudinary.com/image/upload/v1741369884/user\_uploads/GwVxDwWXoPngFSS/Scree

nshot-20250307-at-11.48.13.jpg

https://blog.roboflow.com/getting-started-with-roboflow/

https://stackoverflow.com/questions/71265841/how-to-validate-my-yolo-model-trained-on-custom-dat

a-set

https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/1998337/bf4a7845-a09f-471d-814980

1536f7723b/5.Model-training.pdf

https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection\_676e852f-69434444-a023

28f1e49351dd/cdd0823107394996-b6e0-cc5427a206a9/5.Model-training.pdf

https://roboflow.com/model/yolo-world https://thinkrobotics.com/products/dofbot-ai-vision-robotic-arm-with-ros-for-raspberry-pi-4b-online https://www.youtube.com/watch?v=KYPFQ0erNjw