EdgeFleet.AI Take Home Assignment

Vrishab Kempula

December 2025

1 Introduction

This work focuses on detecting a cricket ball in video frames captured from a fixed camera and a two-stage approach is adopted:

1. Pretrain a YOLOv11 detector on a public cricket ball dataset to learn generic ball features.
2. Perform domain adaptation by fine-tuning the model on a curated subset of manually annotated EdgeFleet test video frames using CVAT, and evaluate on the remaining unseen test videos.

2 Baseline Model on Public Dataset

2.1 Public Cricket Ball Dataset

As a starting point, we use a publicly available "Cricket Ball Dataset for YOLO" from Kaggle, which provides images and YOLO-style annotations for the cricket ball class. The dataset is organised in the standard YOLO folder structure with separate training, validation and test splits.

This dataset offers:

- Diverse backgrounds and camera viewpoints.
- Many examples of the ball at various positions and scales.

Although it does not perfectly match the EdgeFleet's test video camera setup, it is sufficient to learn a strong generic representation of what a cricket ball looks like.

2.2 YOLOv11 Baseline Training

We use Ultralytics YOLOv11 as the base detector, starting from a COCO-pretrained checkpoint. The model is trained on the Kaggle cricket ball dataset for multiple epochs with standard hyperparameters (learning rate, batch size, image size) tuned for small objects.

Motivation. Training from scratch purely on a small in-house dataset would likely overfit and fail to generalise. Pretraining on a larger public dataset allows the model to learn robust low-level and mid-level features for ball-shaped objects (edges, textures, shading), which can later be adapted to the specific appearance and context of the EdgeFleet videos.

Observation. The baseline model performs reasonably well on images similar to the public dataset but struggles on the EdgeFleet videos. This indicates a domain gap: differences in pitch colour, camera angle, ball size in pixels, lighting and compression artefacts cause a noticeable drop in detection quality.

3 Domain Adaptation to EdgeFleet Videos

3.1 EdgeFleet Video Subsets

The EdgeFleet.AI team provided 15 test videos recorded from a static camera. To perform domain adaptation in a controlled way, we split these videos into two subsets:

- **Adaptation subset (labelled):** videos 2, 3, 4, 5, 7, 9, 14.
- **Evaluation subset (unseen):** videos 6, 8, 10, 11, 12, 13, 15.

The adaptation subset is chosen to cover a variety of ball trajectories, bowling actions and lighting conditions. The remaining videos are kept strictly unseen during annotation and fine-tuning, and are used only for final evaluation.

3.2 Frame Extraction and Manual Annotation

Directly annotating long videos is inconvenient and, on free-tier tools, sometimes restricted. To streamline the process, we:

1. **Extract frames** from the selected EdgeFleet videos using a simple frame extraction script (OpenCV-based). Each video is converted into a sequence of individual images.
2. **Annotate frames in CVAT** (Computer Vision Annotation Tool) as image sequences. A single object class `ball` is defined, and tight bounding boxes are drawn around the ball wherever it is visible. CVAT's interpolation and tracking features are leveraged to propagate bounding boxes across neighbouring frames, with manual corrections applied where necessary.

The outcome of this stage is a small but high-quality set of in-domain images where the cricket ball is accurately localised in every frame where it appears.

3.3 Label Cleaning and Quality Control

The exported annotations are nominally in YOLO format (class index and normalised bounding box coordinates). However, different export settings or tool versions can introduce extra columns or minor inconsistencies. Therefore, we perform a light post-processing and quality control step:

- **Format normalisation.** Each annotation file is checked and simplified so that every line contains exactly five values: class index, x_c, y_c, width and height, all normalised to $[0, 1]$.
- **Visual inspection.** A small utility script overlays bounding boxes onto the corresponding images to visually confirm that the boxes are centred on the ball, have reasonable size and are not misaligned.
- **Temporal consistency check.** For quick inspection of annotation quality over time, the annotated frames are recomposed into short videos. Viewing the bounding boxes as the ball moves makes it easy to spot jitter, missed frames or drift.

3.4 Fine-Tuning on Combined Data

Once the in-domain annotations are validated, we combine:

- The original Kaggle cricket ball training images and labels.
- The newly annotated EdgeFleet frames from videos 2, 3, 4, 5, 7, 9 and 14.

The YOLOv11 model pretrained on the Kaggle dataset is then *fine-tuned* on this augmented dataset. Conceptually, this two-stage training can be seen as:

1. Learning generic ball features from a large, diverse, public dataset.
2. Adapting those features to the specific camera, pitch and lighting conditions present in the EdgeFleet videos.

This strategy is particularly effective when the target domain has limited labelled data but is related to a larger source domain.

4 Inference and Qualitative Evaluation

After fine-tuning, the adapted model is evaluated on the held-out EdgeFleet videos (6, 8, 10, 11, 12, 13 and 15), which were not used for annotation or training. For inference, we employ several simple techniques to cope with extremely small balls and to aid qualitative analysis:

- **Region-of-interest (ROI) focus.** Since the camera is static and the action is centred around the pitch, each frame is cropped to a central region before resizing for YOLO. This effectively increases the ball's relative size in pixels.
- **Temporal tracking.** YOLO's built-in tracking functionality is used to link detections across frames. This yields a temporally consistent ball track instead of isolated frame-wise detections.
- **Trajectory visualisation.** For each frame, the detected ball centroid is recorded and a short history of previous centroids is drawn as a trajectory curve. This provides an intuitive visual summary of the ball's motion and helps to spot missed or noisy detections.

Although these tracking and visualisation components are not part of the training loop, they are essential to understanding the behaviour of the final system on realistic video streams.

5 Conclusion

In summary, the proposed pipeline for cricket ball detection on EdgeFleet videos consists of:

1. **Baseline pretraining** of YOLOv11 on a public cricket ball dataset to obtain strong generic features.
2. **Targeted in-domain data creation** by selecting representative EdgeFleet videos, extracting frames and manually annotating the ball positions using CVAT with interpolation.
3. **Annotation cleaning and verification** to enforce a consistent YOLO label format and to visually validate bounding boxes.
4. **Fine-tuning** the pretrained model on the union of public and in-domain data to close the domain gap.
5. **Inference and tracking** on unseen EdgeFleet videos with ROI focus and trajectory visualisation for qualitative assessment.

This staged approach is deliberately chosen to handle the dual challenges of limited labelled in-domain data and a strong domain shift between public datasets and the EdgeFleet camera setup. It balances the benefits of large-scale pretraining with the necessity of targeted domain adaptation, resulting in a practical and robust ball detection pipeline for the given assessment.