Researcher: Maes, Brady

Mentor: Uddin, Md Yusuf Sarwar,

Department: School of Science and Engineering, Computer Science

Funding: The Department of the Navy, Office of Naval Research under ONR award number N00014-21-1-2710.

Frame Pruner:
An SVM-based Approach To Improving Detection Model Resource Consumption

Abstract

A major challenge in applying object detection models is their high computational cost, particularly on edge devices within distributed systems. Even lightweight models often struggle to maintain real-time performance on underpowered hardware. This project addresses that challenge by developing a lightweight, hybrid solution that combines both non-AI and AI techniques to filter redundant frames in video streams before running a full detection model.

The approach relies on ORB (Oriented FAST and Rotated BRIEF) feature matching and cosine similarity as non-neural metrics for frame comparison. Ground truth was established using a detection model to label frame pairs as similar or dissimilar based on detected object classes exceeding a confidence threshold. These labels were then used to train two SVMs (Support Vector Machines). Both SVMs utilized ORB data from raw images; however, one used cosine similarity based on raw images, while the other used cosine similarity derived from intermediate feature maps of the detection model's backbone.

Both SVMs were evaluated on the same video, which was not seen during training but was similar in domain to the training footage. On a T4 GPU in Google Colab, the SVM using raw image data for both ORB and cosine similarity achieved a 65.26%. reduction in runtime with only a 1.71% drop in accuracy. The SVM that used feature map-based cosine similarity achieved a 37.78% reduction in runtime with a 1.06% drop in accuracy. These results demonstrate that both models substantially reduce computational overhead while maintaining high detection fidelity when applied to similar-domain videos.

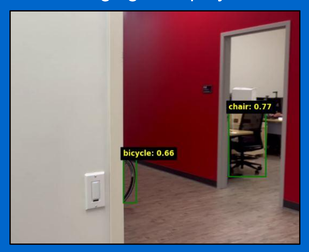
This method is recommended for use only on video footage that is similar to the domain on which the SVM was trained. It is not intended for use with unrelated footage or isolated images lacking temporal continuity.

Keywords: SVM (Support Vector Machine), ORB (Oriented FAST and Rotated BRIEF), cosine similarity, object detection, distributed computing, frame skipping, video processing

Introduction & Key Concepts

Object detection models are machine learning models trained to identify and locate specific objects within an image or video frame. Unlike simple classification models that only predict what's in an image, detection models return both what they see (e.g., a bicycle, a chair, etc.) and where they see it by drawing bounding boxes around the detected objects. These models are widely used in applications such as self-driving cars, video surveillance, and augmented reality. However, they are computationally intensive, especially when run continuously on video streams — which makes them challenging to deploy on resource-constrained devices.

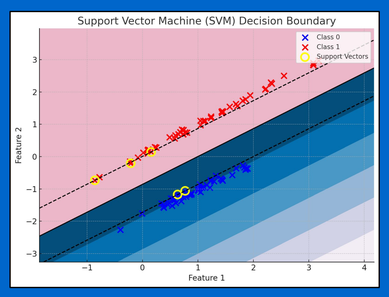
Detection Model Example:



A Support Vector Machine (SVM) is a machine learning algorithm that classifies data by finding the best boundary (or hyperplane) between classes. It only relies on the most important training examples, called support vectors, which makes it efficient, even in high-dimensional spaces (Vapnik, Golowich, & Smola, 1997).

In this project, SVMs are used to decide if a frame is similar to the last processed frame and should be skipped.

Visualization of a 2-dimension SVM:



ORB features use an algorithmic method of finding important features within an image, referred to as keypoints. Most notably, these features are rotation invariant and resistant to white noise (Rublee et. all, 2011). ORB features are particularly useful for tracking objects across frames, which helps establish continuity between video frames. This continuity allows the SVM to learn patterns where similar ORB matches indicate that the same object appears in both frames, even when it changes in size or orientation.

ORB keypoints across 2 frames:

Frame 2090

Frame 2160

Cosine similarity is a simple, well-established technique that measures the cosine of the angle between two non-zero vectors. One key advantage of this method is its insensitivity to magnitude, since it compares direction rather than length. As a result, variations in image brightness — such as shadows — have minimal impact on similarity scores.

Methodology/Pipeline

The overall methodology for this project can be broken down into three core stages: dataset creation, model training, and model evaluation. Each stage builds on the last to develop and assess a lightweight system for intelligently skipping redundant video frames.

1. Dataset Creation

We begin by converting a video into individual frames at a given rate (30FPS). Each frame is saved and passed through a pre-trained detection model (EfficientDetD7) to extract:

- Detected object classes (above a set confidence threshold)
- Intermediate feature maps from the backbone (EfficientNetB2, block 5)

These detections are logged per frame in a CSV file.
A new column labeled "BinaryResult" is added to the CSV by applying the following logic:

- Let classes1 represent the detected classes in the first frame, and classes2 in the second frame.
- Label = 0 (Skip) when:
  - Both frames contain no detected classes
  - classes1 == classes2
  - classes2 is a subset of classes1 (i.e., only objects leave the frame)
- Label = 1 (Process) when:
  - classes2 contains new object classes not in classes1
  - The same classes are present, but one or more appears more frequently in classes2
    (*e.g.,* classes1 = [chair, person], classes2 = [chair, person, person])

Where "BinaryResult" is 1 if two frames are similar and 0 otherwise.

ORB is then applied to all frames, and descriptors from each frame pair are matched using Brute-Force Matching with Hamming distance. The ratio of matched descriptors is used as the ORB similarity score between the frames. Similarly, the

Cosine similarity is computed for each frame pair by first flattening the image arrays and dividing them into four quadrants. The cosine of the angle between corresponding quadrant vectors is calculated, producing four similarity scores (cos1–cos4) that reflect localized visual similarity between frames. The same process is repeated but on the feature maps for the feature map-based cosine similarity dataset (for training the second SVM).

The ORB and cosine similarity scores are then appended to the corresponding image pairs in the BinaryResult CSV file.

2. Model Training

A Support Vector Machine (SVM) is used to classify frame pairs as either "process" (1) or "skip" (0) based on ORB and cosine similarity features. Since the data is **not linearly separable**, an RBF kernel is used to enable flexible, non-linear decision boundaries.

Because most frames are redundant and labeled as 0, the dataset is heavily imbalanced toward class 0. To address this, the following techniques were applied:

- **Class weighting** to give more weight to class 1
- **Confidence threshold adjustment** to bias toward predicting class 1
- **SMOTE** (Synthetic Minority Over-sampling Technique) to generate synthetic samples for the underrepresented class

3. Model Evaluation

Since frame skipping is not intended to improve the detection model's accuracy, we assume the original detection model (with no frame skipping) achieves 100% detection of the target object above the confidence threshold. This assumption provides a reference point for evaluating how frame skipping affects performance.

To assess whether the skipped-frame version still detects the target object, we compare the index (i.e., frame number) at which the object is first detected:

Effective Accuracy (A'):

$$A' = \frac{original\ frame\ detected\ index}{new\ detected\ frame\ index}$$

- A' represents the relative accuracy after frame skipping.
- A value closer to 1 means the target was detected with little or no delay.

To evaluate computational efficiency, we compare the runtime with and without frame skipping:

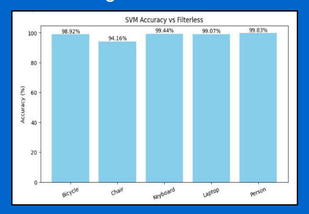
$$Time\ Saved = (1 - \frac{new\ run\ time}{original\ run\ time}) \times 100$$

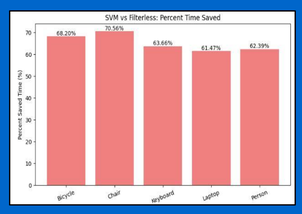
Where the new run time is with our SVM filtering and the original run time uses no filtering technique

Results & Conclusion

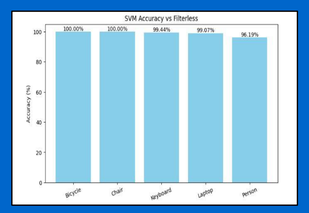
The trained SVMs were evaluated on a video that was similar in domain but distinct from the training video. The results below reflect performance on five target object classes using a 60% detection confidence threshold.

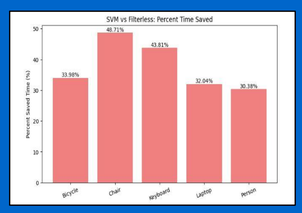
Raw Cosine SVM:





Feature map-based SVM:





The SVM using raw image data for both ORB and cosine similarity reduced runtime by 65.26% with only a 1.71% drop in accuracy. The model using feature map-based cosine similarity achieved a 37.78% reduction in runtime with just a 1.06% accuracy loss. These results show that both approaches offer significant computational savings while preserving strong detection performance when applied to videos within the same domain as the training data.

References

1. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2011). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

2. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011). ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2564–2571). https://doi.org/10.1109/ICCV.2011.6126544

3. Vapnik, V., Golowich, S. E., & Smola, A. J. (1997). Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), Advances in neural information processing systems (Vol. 9, pp. 281–287). MIT Press.