Team SVD: Project 1 Submission

Team Members and Contributions to the project:

- *Dibyendu Mondal* (Team Lead)
  - Task: Implementing the 3 required functions
  - Task: Trying different classifiers and feature vectors

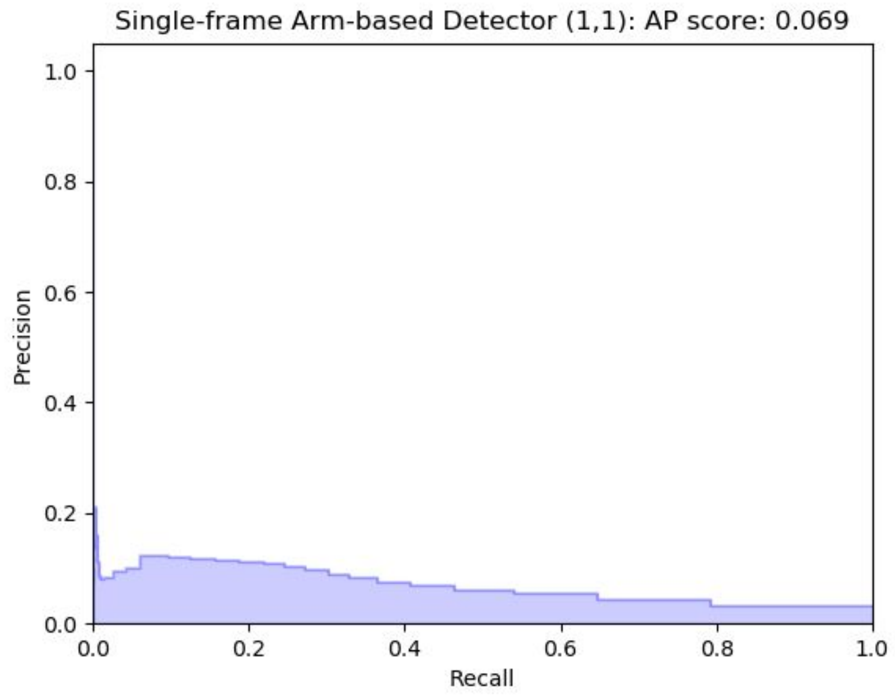
  Task: Implementing temporal smoothing
- *Sarthak Wahal*
  - Task: Implementing Multi-class classifier
  - Task: Analysis of performance
- *Vaibhav Tendulkar*
  - Task: Testing the performance of different ML algorithms
  - Task: Hyper parameter optimization

Part 1: Single Frame Arm-based Detector

Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.069



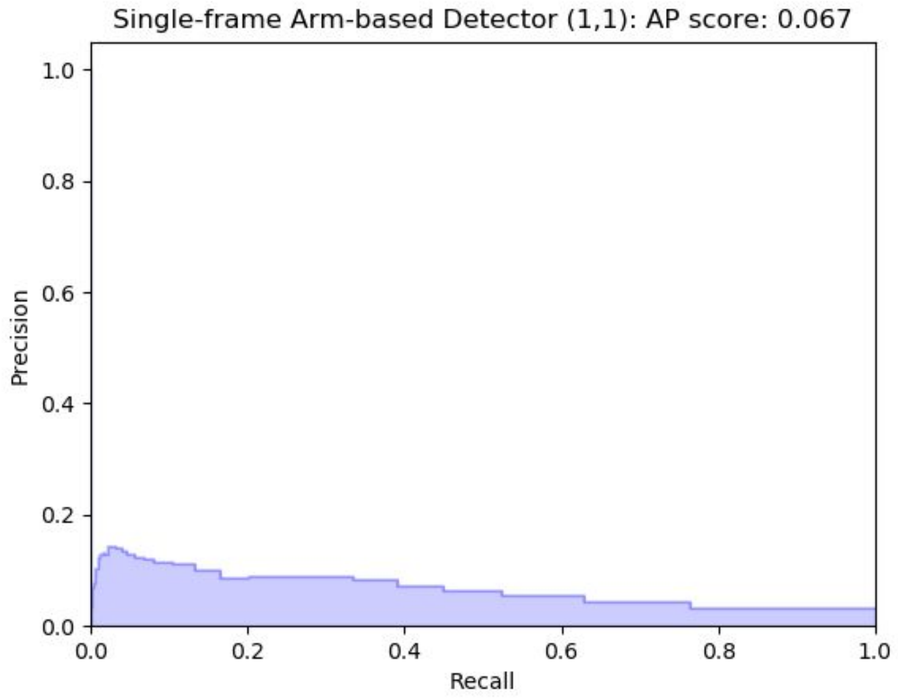
Part 1: Single Frame Arm-based Detector

Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.067

Using confidence as weights for the feature vectors while training.

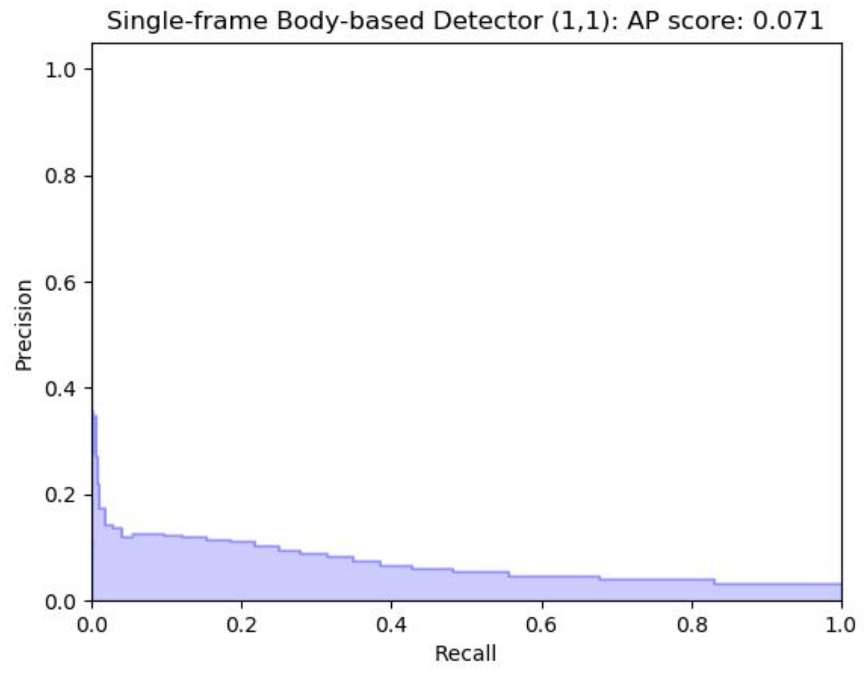


Part 2: Single Frame Body-based Detector

Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.071



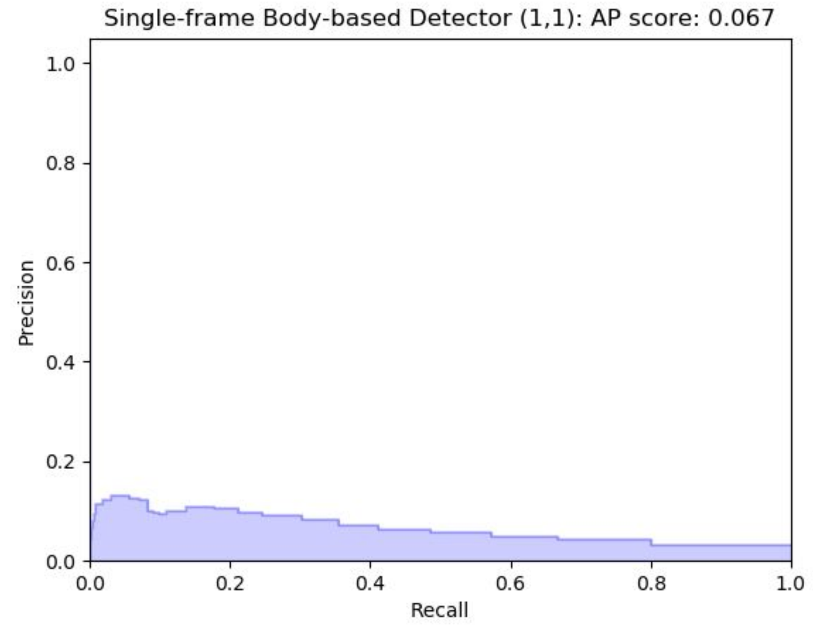
Part 2: Single Frame Body-based Detector

Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.067

Using confidence as weights for the feature vectors while training.

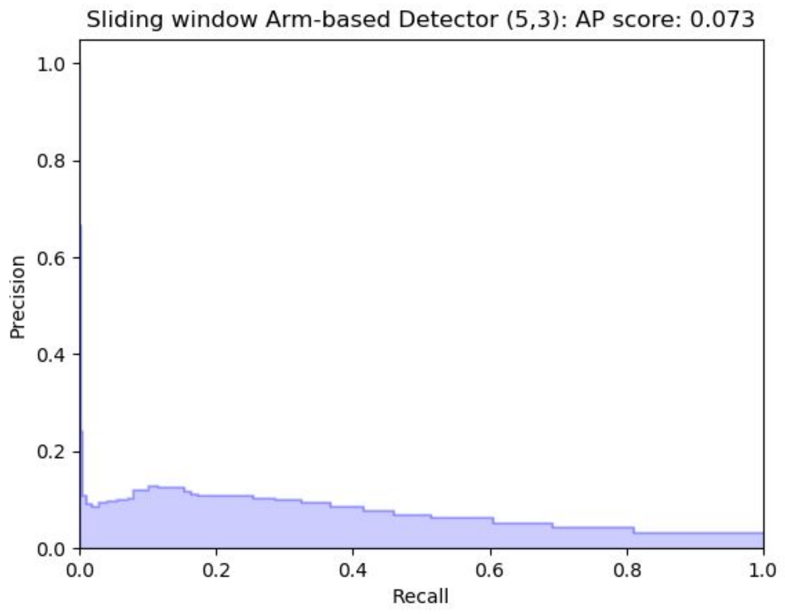


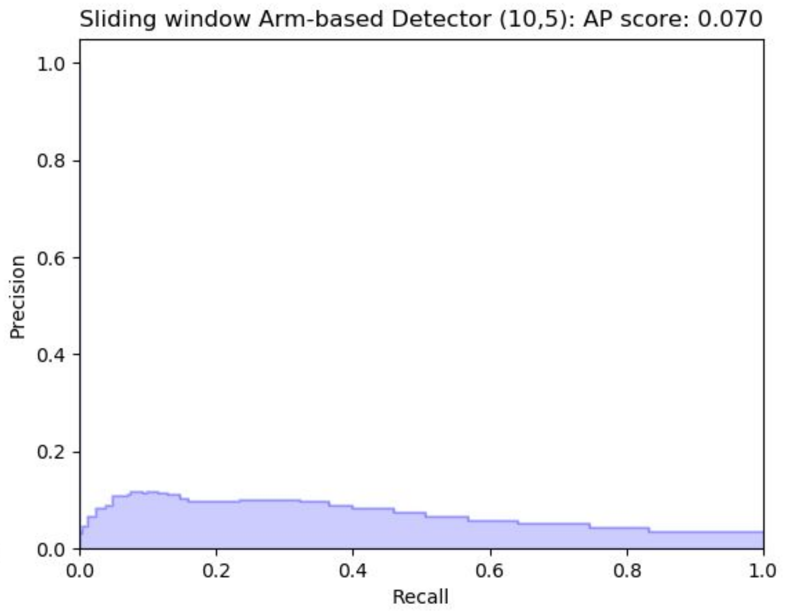
Part 3: Sliding Window Arm-based Detector

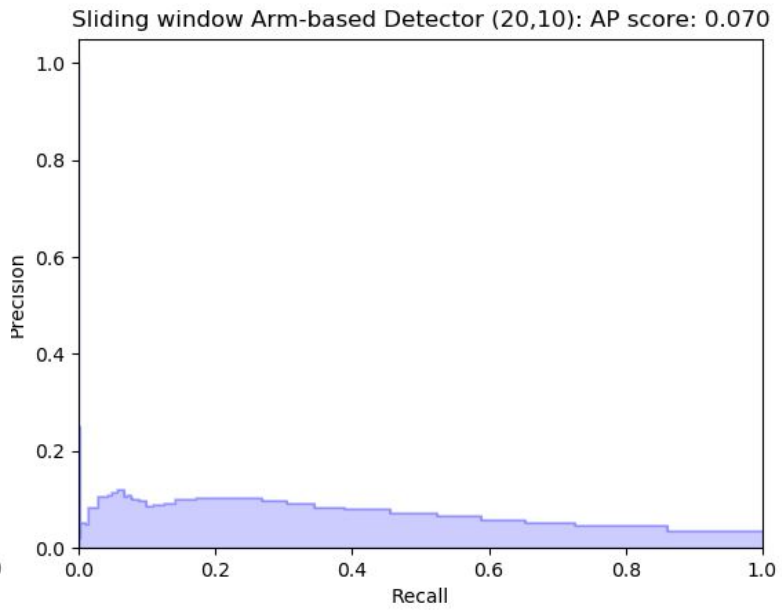
Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.073, 0.070, 0.070







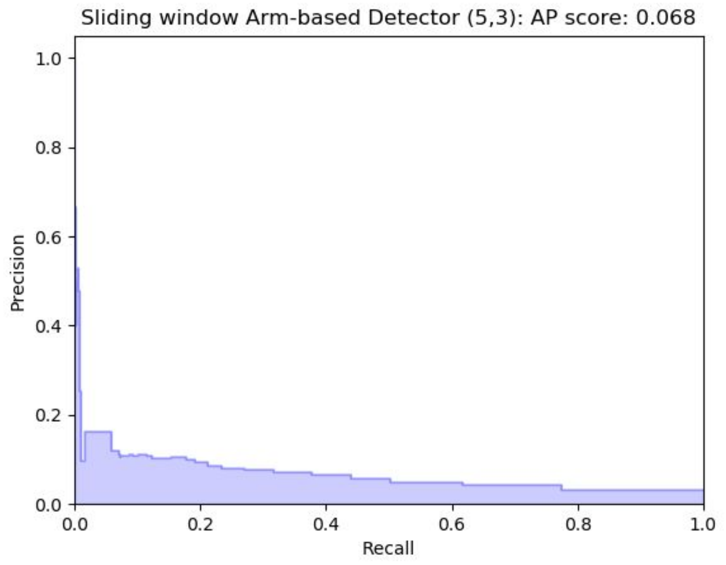
Part 3: Sliding Window Arm-based Detector

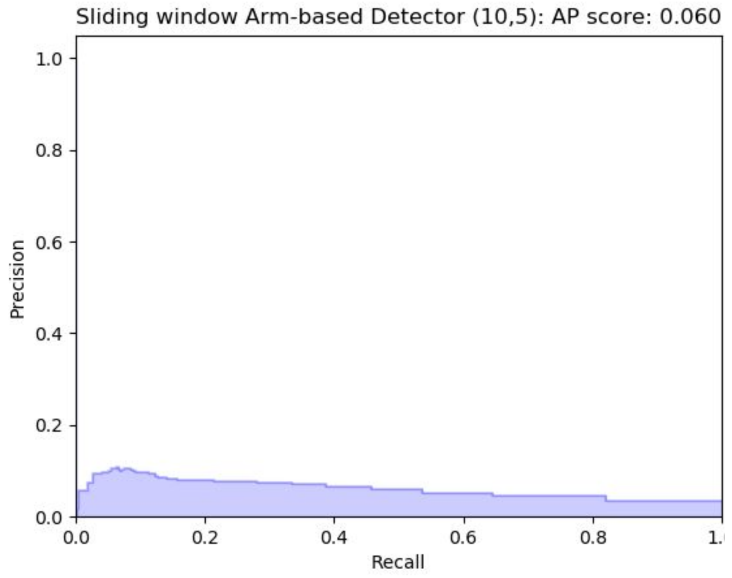
Classifier Choice: Random Forest

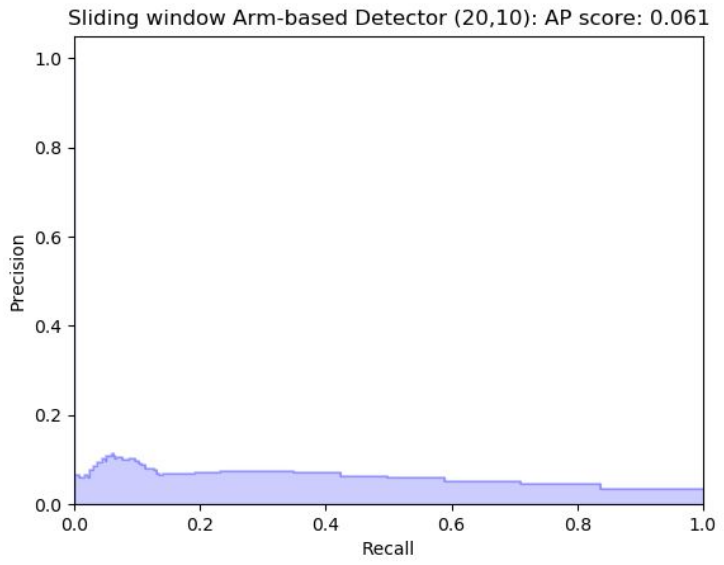
Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.068, 0.060, 0.061

Using confidence as weights for the feature vectors while training.





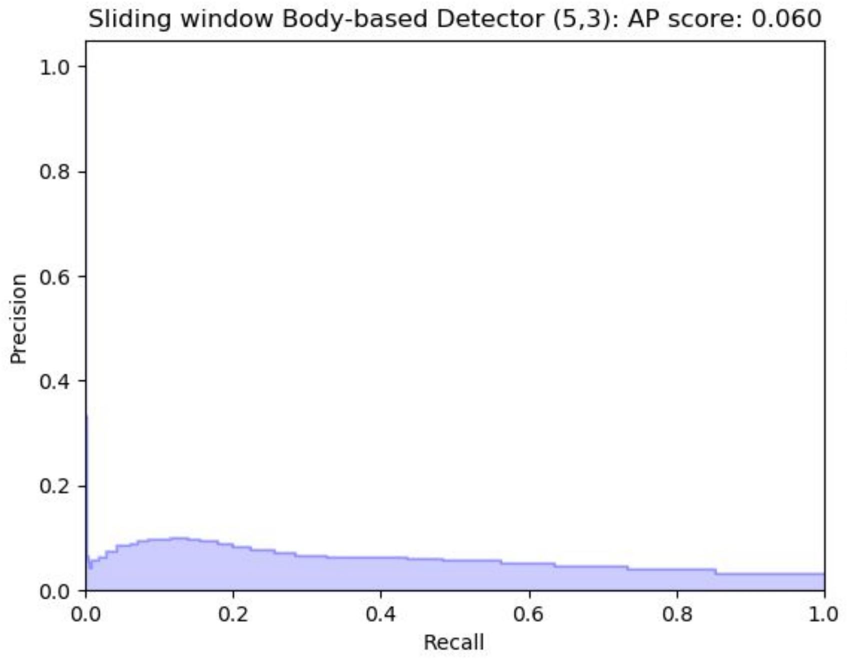


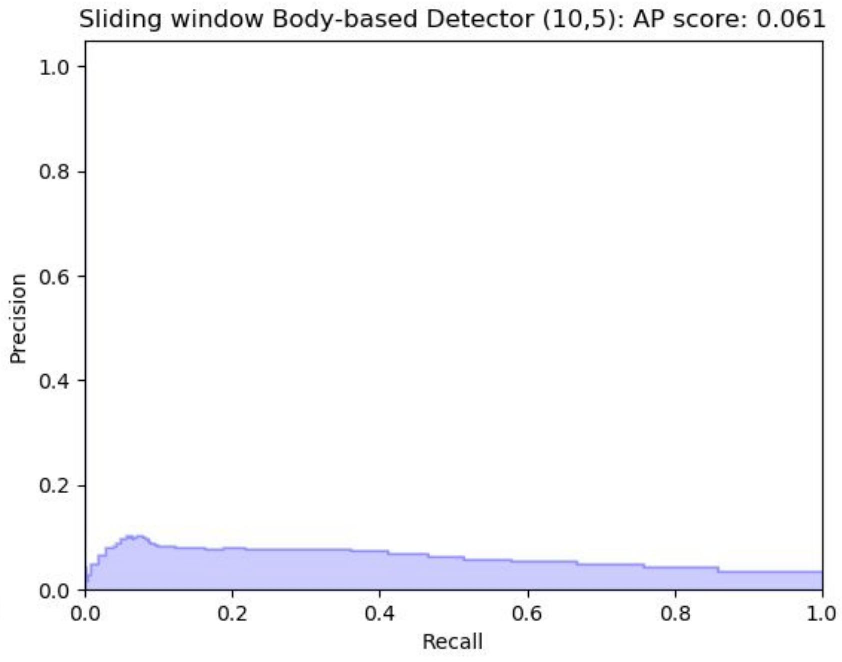
Part 4: Sliding Window Body-based Detector

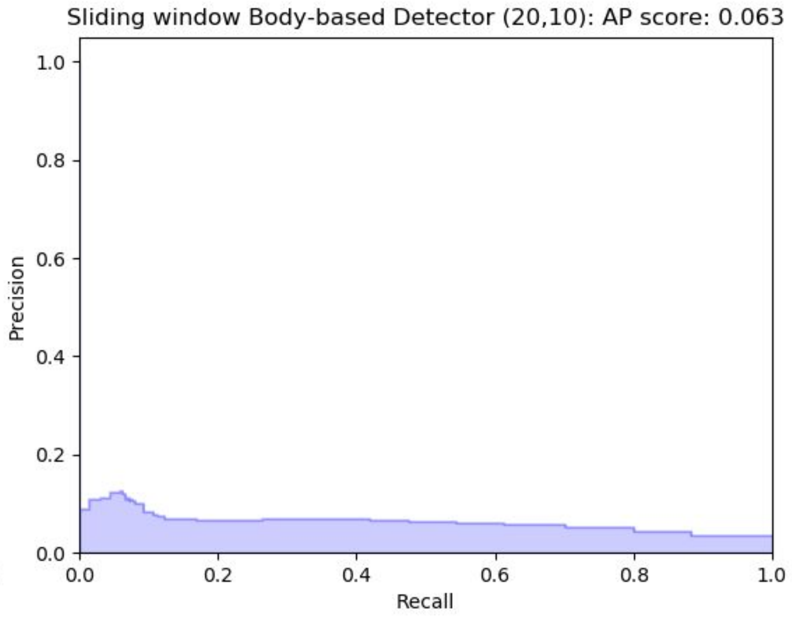
Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.060, 0.061, 0.063







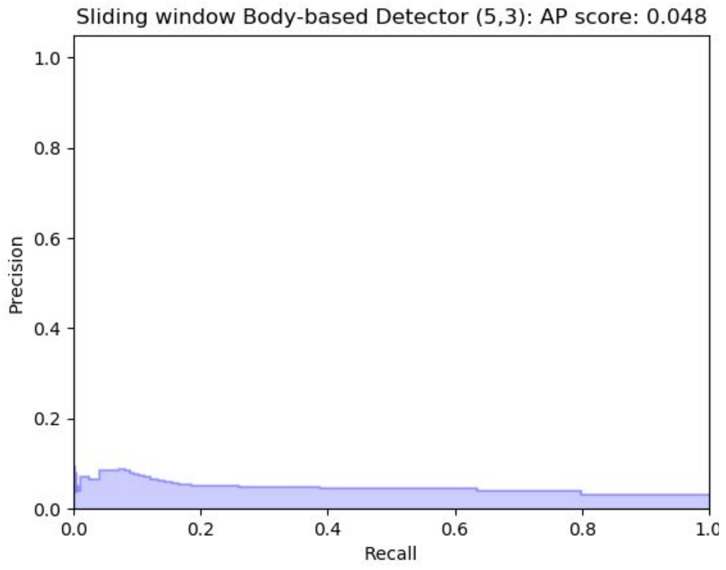
Part 4: Sliding Window Body-based Detector

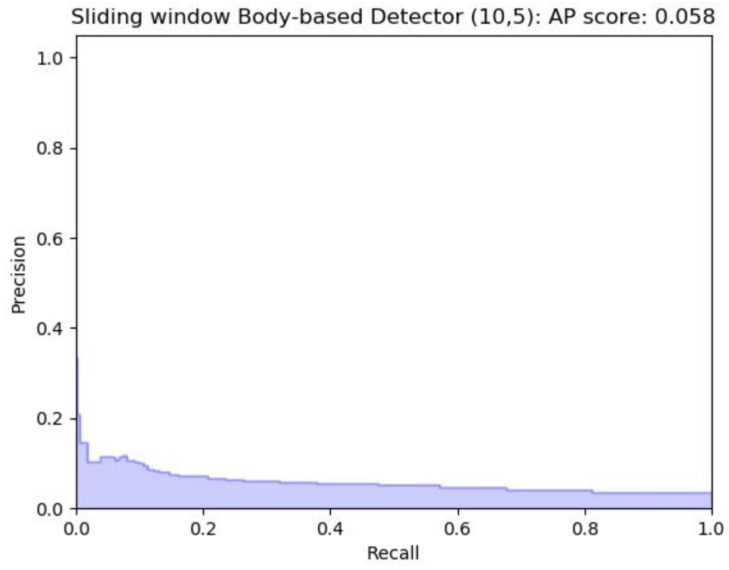
Classifier Choice: Random Forest

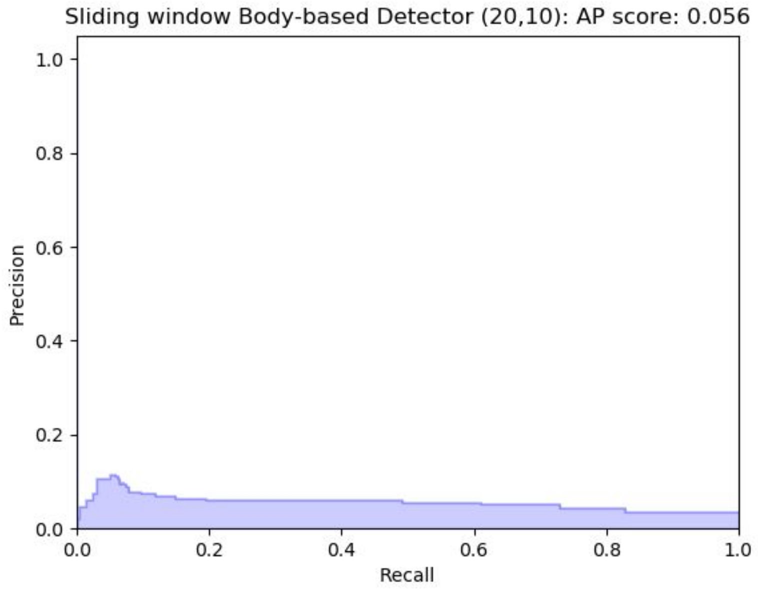
Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.048, 0.058, 0.056

Using confidence as weights for the feature vectors while training.







Analysis of Performance

How did the arm-based and body-based detectors compare in their performance? Which one was better and why?

Ans. Body-based detectors are performing better than arm-based detectors. This is because relative positioning of the body and the arm helps differentiate between pointing and non-pointing gestures. Even if OpenPose fails to detect keypoints for arms, the body key points can help the classifier get an idea as to whether the child is pointing or not.

Additionally, the extra features may be good to split on, which improves results, or may not be good to split on, which does not spoil the result. The only downside of additional features is the extra time taken.

Analysis of Performance

How did the single frame and window-based detectors compare? Which one was better and why?

Ans. Window-based detectors are performing slightly better than single frame detectors. Looking at neighbouring frames gives the classifier a better idea as to whether the current frame is pointing or not. For example, if OpenPose outputs zeros if it is not confident about a keypoint. In this case, looking at neighbouring frames in the window can guide the classifier as to whether the child is pointing or not.

However, the improvement is not statistically significant, as the comparison in decision trees is feature to feature, while what is important here is the relative change among frames

Analysis of Performance (continued)

Take your best detector from Parts 1-4 (your baseline detector) and look at the mistakes that it made

What were some reasons why you had *false negatives*?

Ans. The dataset is highly skewed i.e. the number of training samples with Y-label 0 are far higher than the number of samples with Y-label 1. Due to an imbalanced dataset, the classifier did not have enough positive data points to train. Hence, it classified some pointing frames as non-pointing (false negatives).

Analysis of Performance (continued)

What were some reasons why you had *false positives*?

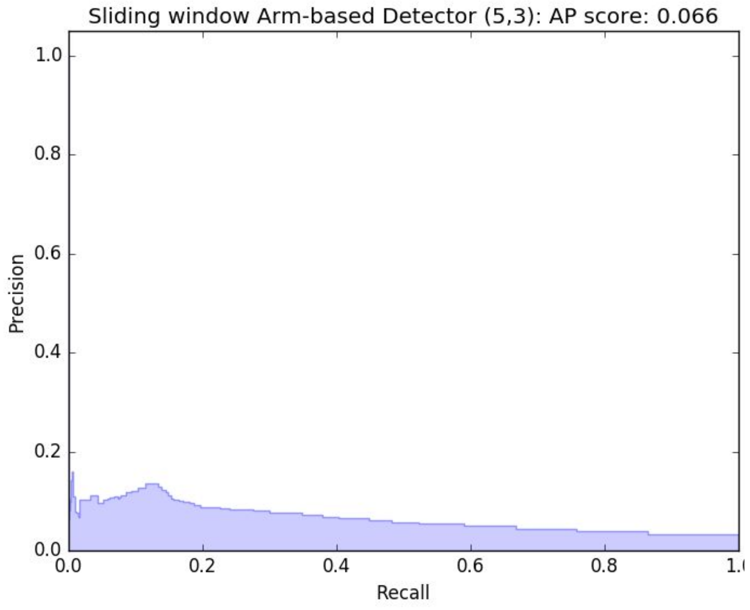
Ans. Upon examining the data frame by frame, we observed that the child was exhibiting other gestures like pat, tap, push and reach. But as these gestures were very similar to the pointing gesture, the classifier classified some of these as pointing when in fact they were not (false positives).

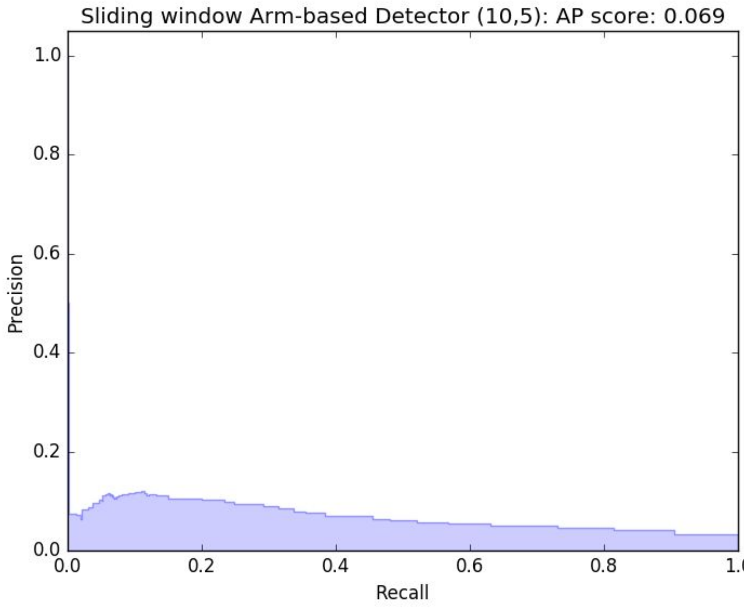
Part 5: Sliding Window Arm-based Detector

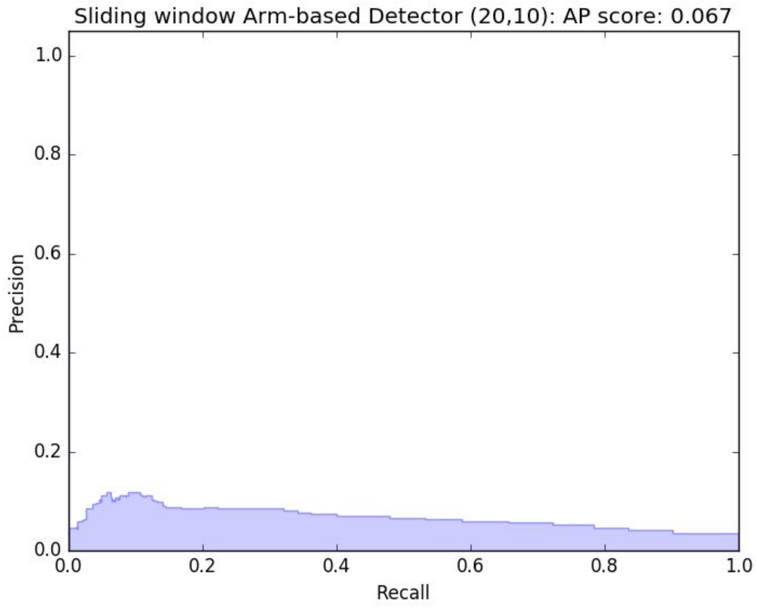
Classifier Choice: Random Forest

Classifier Implementation: scikit-learn

Average Precision (and PR curve): 0.066, 0.069, 0.067







Part 5

AP of your baseline detector: 0.073

Strategies we used to try to improve the baseline:

1. Multi-class classifier for pat, reach, push, tap and point
2. Temporal smoothing
3. Normalization
4. Using confidence as weights for feature vectors while training
5. Hyper parameter tuning

Final best AP that you obtained from modifications: 0.069

Part 5

Strategy 1. Multi-class classifier for pat, reach, push, tap and point

Reason for use and why was it successful (or unsuccessful)?

The classifier now classified pat, reach, push and tap as well which reduced the number of false positives (samples that were actually “reach” but were being classified as “pointing” in the earlier version). But the classifier also reduced the number of true positives (samples that are actually “pointing” but are now being classified as “reach” / “pat” etc). So the contribution by decrease in false positives is offset by the decrease in true positives and the AP score did not improve. (Refer the multiclass PR curves which accompany the parts 1-4).

Part 5

Strategy 2. Temporal smoothing

Reason for use?

If OpenPose is not confident about a keypoint, it outputs zeros for its coordinates. We added temporal smoothing using rolling_mean() function which handled the missing zeros in the data. We expected temporal smoothing to improve the score of the classifier but it failed to do so.

Part 5

Strategy 3. Normalization

Reason for use and why was it successful (or unsuccessful)?

The coordinates have a greater magnitude while the confidence values lie between 0 and 1. So a classifier would assign greater weight to coordinates and lesser to confidence values. Normalization is needed for getting all the data on the same scale. For Random Forest, since one feature is never compared in magnitude to the other features, the ranges do not matter. It is only the range of one feature that is split at each stage. Since our baseline model is Random Forest, the AP score did not improve after normalization.

Part 5

Strategy 4. Using confidence as weights for feature vectors while training

Reason for use and why was it successful (or unsuccessful)?

We also removed the confidence values from our feature vector and trained the classifier using the confidence values for each frame as weights for the feature vectors. That is, confidence changed from being a set of features to a single value which decided the weight of the feature vector. The idea was that higher confidence samples should contribute more in training. But it may be that the random forests' method to do so is better than ours. We thought this would perform well but actually gave worse results in most of the cases.

Part 5

Strategy 5. Hyper parameter tuning

We tried different parameters for Random Forest like setting max_features to None, adding n_estimators, setting oob_score to true. Setting max_features to None decreased our AP score but the other parameters increased it at the cost of a few more minutes in the runtime.

Part 5

Other techniques tried :

We also tried other classifiers like Multilayer Perceptron and Support Vector Machine. MLP was performing a lot worse than Random Forest. SVM with RBF kernel was taking a hours to train even for the first fold, so we did not continue with that.

We also tried Adaboost, but it was performing way worse than others. We got an AP score of 0.030 for similar setup as P3.

Final Comments

Even after attempting many ideas and techniques, we were unable to obtain improvements. The most probable reason would be the large number of features, which should not be seen in isolation but relative to each other. The highly unbalanced dataset is also a problem. With about 3.4% of the samples of ‘pointing’ class, the AP score still reveals it to be a weak Learner, and thus better than random. But improving the result by single modifications was highly challenging.