Team : TuringTested

Pedestrian Data Analysis

Programme : Aspiring Researchers
Submission : 22nd June, 2015

Members :

Anshu Aviral (aviral.2815@gmail.com)
Vishal Anand (anand.vishal93@gmail.com)
Shweta Sharma (shweta.the.indian@gmail.com)
Utkarsh Dwivedi (utkarshdvd@gmail.com)

CONTENTS

Abstract

The main aim of our project is to improve the Caltech Pedestrian detection for the purpose of pose estimation in images as well as videos. The output of an algorithm (CPU) is unreliable and not very effective as pedestrians remain unidentified due to unusual pose or the other objects in the scene are interpreted as pedestrian. Human input (HPU) can be pre-owned through the user interface to detect/highlight the pedestrian. Thus the HPU and CPU techniques are compared on the basis of performance and cost (as time taken for the input). Different HPU techniques are also introduced keeping in mind the human behavioral aspect.

PEDESTRIAN DETECTION

We started the project with the aim of detecting 2D pose of a human upper body in a video. The video could be of some movie or TV show or a game sourcing application. The pose can be derived in the form of soft labeling of every pixel as belonging to background or foreground, or a stick-man figure, indicating location, orientation and size of body parts.

We later realized that a good pose estimation algorithm automatically ensues an efficacious localization algorithm. So we diverted our focus towards effectively localizing a person in a video frame or in an image. Foraging a good dataset for this purpose, we stumbled upon this recondite problem of Pedestrian Detection. Hence, the goal of this report is to elucidate our algorithms for pedestrian detection and our findings for the same.

DATASET INFORMATION

We have processed the data from three sources :

1. Fudan Pedestrian Dataset, which comprises of 74 images
2. Penn Pedestrian Dataset, which comprises of 96 images
3. Caltech Pedestrian Dataset, which comprises of 4024 images, wherein there exists 5 sets of data :

a. Set 06 : 1155 images (parts : 61, 61, 59, 59, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61)
b. Set 07 : 746 images (parts : 61, 61, 61, 61, 65, 72, 61, 61, 61, 61, 61, 60)
c. Set 08 : 657 images (parts : 59, 59, 61, 61, 61, 61, 61, 61, 61, 56, 56)
d. Set 09 : 738 images (parts : 62, 62, 61, 61, 59, 66, 61, 61, 61, 61, 61, 62)
e. Set 10 : 728 images (parts : 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 61, 57)

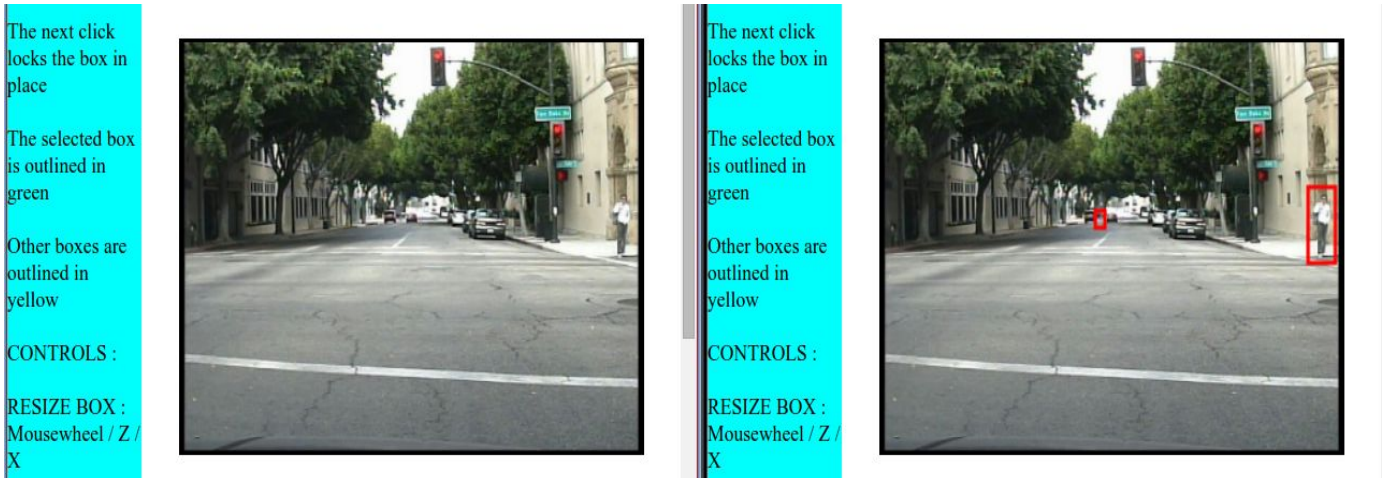
which is a huge number to be processed, and hence, for processing over HPUs, we took the 9th set comprising of 738 images for HPU calculations from this dataset. The Fundan and Penn Pedestrian dataset were processed in the month of April-May so as to validate our methods. We present our main results on Caltech Pedestrian Dataset.

CPU ALGORITHM

The first step in the pipeline was to calculate the bounding box using an automated CPU algorithm. We used the Histogram of Oriented Gradients (HOG) algorithm for getting the bounding boxes with confidence values (SVM prediction). We even calculated confidence scores corresponding to every bounding box. Our implementation of the algorithm can be found here: [LINK].

We performed experiments on FunPed and PennPed dataset using this CPU algorithm as baseline. It turned out that our results were not satisfactory, owing to this low baseline. (We describe them in a later section). Hence, we tried to seek help from another team working on the same dataset and having a good CPU algorithm to start with; which was not great either. It turned out that CPU algorithms don't perform up to the mark on Caltech pedestrian dataset. This may be due to an inherent problems we faced with the dataset: it is too accurate. This leads to a high number of false-negatives (FN) during HPU processing.

Consider the following example:



The image on the left shows a sample image. Try to identify the number and locations of the pedestrian in the left image. The image on the right shows the actual number and position of the pedestrians according to the ground truth values (non-occluded). Most of us would miss the farther pedestrian which results in a low HPU performance.

Regardless, we moved ahead, using the CPU results of the our fellow team *Vision93* and performed HPU experiments with it. Next we describe our HPU experiments.

HPU EXPERIMENTS

the higher confidence value of the following images, would ensure the images are correctly processed by the user despite of the decreased interest of the user. Thus, it may lead to better performance metric.

There may occur a heuristic (HPU7) which takes the images of pedestrians and presents the user with alternating values of confidence scores. This goes by the logic that if the user gets a poor pedestrian image and is difficult to identify it; the user would be more certain that the next image would be of a higher quality and thus mistakes in detecting a particular pedestrian would help the user in being apprehending the quality of the next image and be cautious accordingly.

We have tabulated the results and plotted graphs of these HPU approWe ideated seven HPU experiments and their descriptions is as follows: (The code for the same is at the github link)

1. *HPU1*: Draw the bounding boxes without any prior CPU algorithm.
2. *HPU2*: Given a CPU algorithm, Click on all detected bounding boxes that do not contain pedestrians. It aims towards reducing the number of false positives
3. *HPU3*: Given a CPU prediction, Locate and draw bounding boxes around pedestrians that have NOT been detected. By doing this, we can increase the number of true positive and reduce the number of false negative.
4. *HPU4*: Given a CPU algorithm, click on objects not enclosed by bounding box. Then, the CPU chooses the bounding box from the predicted boxes below the threshold to place round it. This method achieves what HPU3 tries to, but in lesser average time.
5. HPU5: Given a CPU algorithm, heuristic to check if showing each pedestrians per image in decreasing order of confidence weeds out false-positives better
6. HPU6: Given a CPU algorithm, heuristic to check if showing each pedestrians per image in increasing order of confidence weeds out false-positives better
7. HPU7: Given a CPU algorithm, heuristic to check if showing each pedestrians per image in the order of alternating sequence of confidence weeds out false-positives better (more confident and lesser confident images spliced).

LOGIC AND REASONING OVER HPUs : 5, 6 and 7

If the user is presented with an image comprising of multiple pedestrians, the user may miss out on the pedestrians of having similar clothes as the colour of the surrounding. But, if the same user is presented with bounding boxes of the pedestrians detected by a given CPU algorithm. The user would more likely to be focussed into detecting the pedestrian or a false-positive as the case maybe.

Thus, we came up with the (HPU5), which presents the user with higher to lower confidence valued images of pedestrians, with the logic that the person would be enthusiastic seeing images with more confidence; but as the confidence values of the ensuing images decrease, the user might be enthusiastic seeing the previous high-confidence images and would continue to give good responses.

However, a reverse heuristic (HPU6) might work better; wherein the user is presented with increasing confidence valued images of pedestrian, with the logic that the person is more attentive at the beginning, but even if the attention of the user starts to dwindle, aches.

N.B.

1. The Fudan and Penn Pedestrian Dataset have been processed by HPU1, HPU2, HPU3, and HPU4.
2. The Caltech Pedestrian Dataset has been processed by HPU1, HPU2, HPU3, HPU5, HPU6 and HPU7

We next present our results in tabulated form. Special attention must be paid to the change in the number of True Positives (TP), False Positives (FP) and False Negatives (FN) for every HPU and how it varies from one HPU to another. Before moving to the actual results, we have described the metric used for the assertions.

RESULTS

The steps involved in calculating the final stats are as follows:

1. The *Intersection Over Union* (IOU) was calculated for every box in the experimental data. A bounding box in the ground-truth data was compared with all other boxes in output data to get the IOU value.
2. Hence we get the following three cases:
    a. If the ground-truth box couldn't find any match, it was declared as False Negative (FN)
    b. If the IOU value for a box in output data is found to greater than the accepting ratio, we declared it as True Positive (TP)
    c. If the IOU value for a box in output data is found to less than the accepting ratio, we declared it as False Positive (FP)
3. We calculated the total number of TP, FP and FN each for CPU, HPU1, HPU2, HPU3 and HPU4.

Please find linked our code for the calculation of IOU and other statistical parameters here: [LINK]

From these values, we finally calculate Precision, Recall and F-measure for each experiments.

	Ground Truth BB	
HPU/CPU value BB	Match	No Match
Detection	True Positive (TP)	False Positive (FP)
No Detection	False Negative (FN)	True Negative (TN)

Note on average time calculation: The average time for CPU algorithm was assumed to be zero. The input bounding boxes for HPU3 and HPU4 were taken to be the output of HPU2 so that the false positive could be removed before improving true positive.

The tabulated form of the results are presented here:

	#TP	#FP	#FN	Recall	Precision	F-measure	Avg. Time Taken
CPU CPU	22	121	136	15.38	13.92	14.61	0 sec
HPU1	110	13	48	89.43	69.62	78.29	6.16 sec
HPU2	22	95	136	18.80	13.92	16.00	1.80 sec
HPU3	45	122	113	26.94	28.48	27.69	3.17 sec
HPU4	35	107	123	24.64	22.15	23.33	3.02 sec

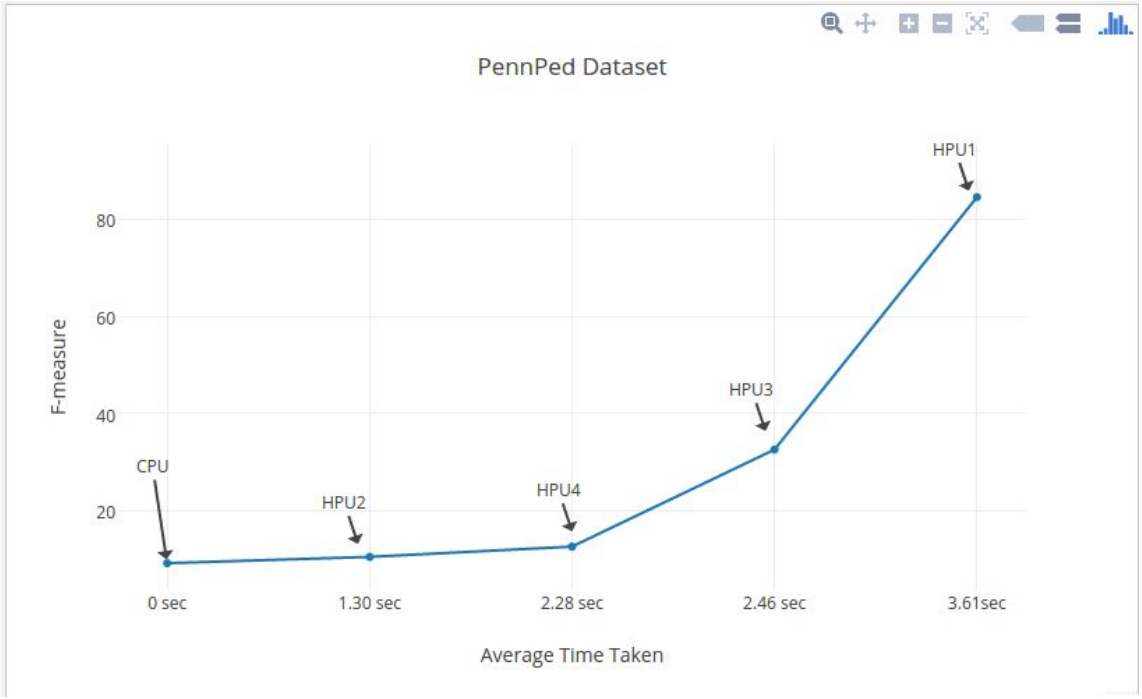
TABLE 1: Fundan Dataset

	#TP	#FP	#FN	Recall	Precision	F-measure	Avg. Time Taken
CPU CPU	22	209	229	9.52	8.76	9.12	0 sec
HPU1	194	2	69	98.97	73.76	84.53	3.61sec
HPU2	22	157	221	12.29	9.05	10.42	1.30 sec
HPU3	82	159	181	34.02	31.17	32.53	2.46 sec
HPU4	32	215	231	12.95	12.16	12.54	2.28 sec

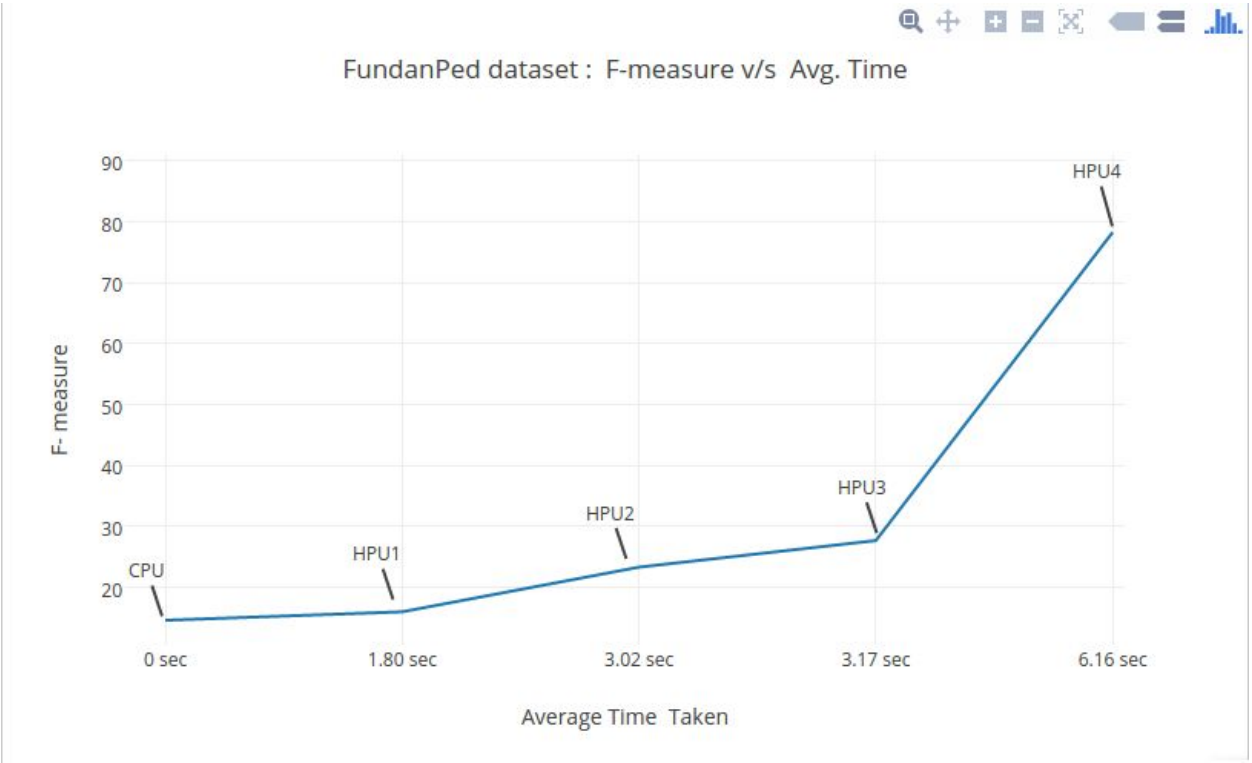
TABLE 2: PennPed Dataset

The output is graphically represented at the following links ():

1. PennPed dataset : F-measure v/s Avg. Time Graph



2. FundanPed dataset : F-measure v/s Avg. Time Graph



OBSERVATIONS

The pattern observed in the result is close to what was expected. The low value of F-measure can be attributed to poor performance of the CPU algo. Nevertheless, we make the following observations:

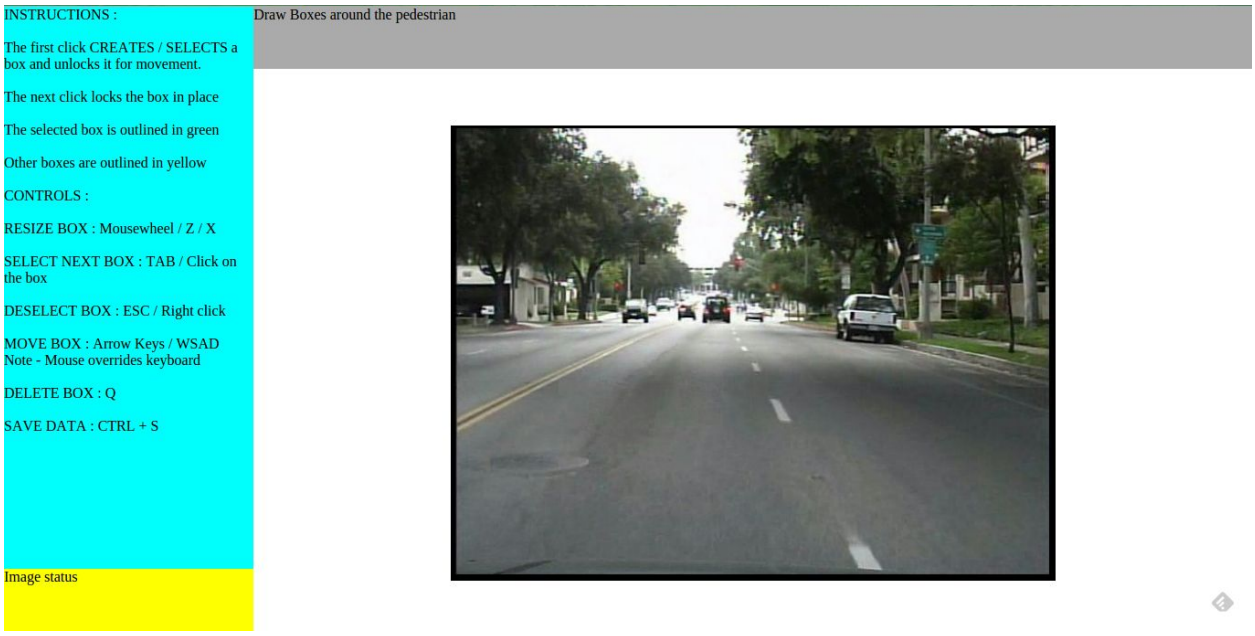
1. The CPU algorithm gives a very low accuracy on both the datasets.
2. The HPU1 proves to be the best platform for getting the highest precision but its too costly as can be inferred by the highest average time taken.
3. HPU2 was performed to reduce the number of false positives and we can observe the same in both the tables. In fact, the F-measure was improved by **9.5%** for Fundan and **14.2%** for PennPed dataset.
4. The HPU3 was performed on the bounding box output of HPU2. Increase in number of true positive and fall in number of false negative can be seen. A 89.52% improvement for Fundan and **256%** improvement for PennPed dataset was observed for F-measure.
5. Finally, HPU4 was aimed at giving the same performance as HPU3 while reducing the average time required. While it did reduce the time taken but we observe a decrease in performance as well. May be, its because of the poor calibration of the size of box in the HPU algorithm.

Note: This work was done earlier in the project. The feedback and comments from Prof. Davis were incorporated in the Caltech Pedestrian Dataset. The results presented above are presented here as it is, without improving upon it.

CALTECH PEDESTRIAN DATASET

- IOU acceptance threshold is one parameter which is varied and analyzed. It is the minimum amount of intersection under union required for acceptance of a bounding box.
- Average Time is the amount of time dedicated by the user per image in order to perform that particular experiment.

HPU Setup screenshot



This screenshot depicts the view of our setup. Detailed instructions on the left(blue), general instructions at the top(grey), image at the center(white, bounded by a black box), image status in bottom left(yellow).

HPU 1:

The user in this experiment, as already described in previous sections, starts from scratch and marks all the pedestrians in the frame. The expectation is to get a high precision, recall and hence high f-measure as the user is all free to mark as many pedestrians as possible.

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	392 17 0 28	406 3 0 11	407 2 0 11
Total Detection	409	409	409
Precision	0.95	0.993	0.995
Recall	0.93	0.974	0.973
F-Measure	0.94	0.983	0.984
Ground Total TP + FP + FN	417 437	417 420	417 420
Avg. time: 5.21284972678 sec			

Table depicting calculations on HPU1

ANALYSIS:

1. F-measure for this experiment is exceptionally high which is as expected.
2. But there certainly exists a trade off between the f-measure and time taken as can be concluded from the table above. Higher accuracy is achieved by devoting longer time per image which we would like to reduce. Hence, we use a hybrid CPU+HPU approach.

CPU:

This is a completely computer generated output with no help from any *worker* whatsoever. We use this as a baseline and work upon it with *human processors* in order to increase the precision, recall and f-measure.

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	204 162 0 255	205 161 0 252	207 159 0 248
Total Detection	366	366	366
Precision	0.557	0.560	0.565
Recall	0.444	0.449	0.455
F-Measure	0.495	0.498	0.504
Ground Total TP + FP + FN	417 621	417 618	417 614

Table depicting calculations on CPU algorithm

ANALYSIS:

1. This CPU algorithm is evidently not up to the mark. The f-measure is as low as 0.50 even for iou threshold of 0.
2. This needs to be improved by reducing False Positives and False Negatives and increasing number of True Positives.
3. These demands are met by HPU2 and HPU3 respectively.
4. Also, as expected the f-measure decreases by increasing the iou threshold.

HPU2:

The sole function of HPU2 is to reduce the number of False Positives and it does it really well. The user is presented with a series of boxes, as generated by the CPU algorithm, which may or may not contain a pedestrian. The user simply needs to press *Enter* or *Space* to separate the two. The user may, in some cases mark a pedestrian to be negative, but that's not an issue here. The user needs to correctly mark out the non pedestrians from the images.

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	174 18 0 268	189 9 0 257	190 8 0 253
Total Detection	192	198	198
Precision	0.906	0.955	0.96
Recall	0.394	0.424	0.429
F-Measure	0.549	0.587	0.593
Ground Total TP + FP + FN	417 460	417 455	417 451
Avg time: 1.028 sec			

Table depicting calculations on HPU2

ANALYSIS:

1. As expected, the number of False Positives plummets from 161 to 8 for the case of *iou* being 0. This vindicates the purpose of the experiment.
2. Though the number of false negatives seem a bit off right now, it is nothing to be worried about as the next experiment takes care of it.
3. High number of False Negatives results in low f-measure but it's higher than that of only CPU case.
4. Also, the low Average time per image is a positive sign.

HPU3:

This HPU specializes in increasing TP and reducing FN. This is accomplished by providing the output of HPU2 (False Positives removed) to the user, who then needs to mark undetected pedestrian.

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	411 23 0 41	421 13 0 28	423 11 0 24
Total Detection	434	434	434
Precision	0.947	0.97	0.975
Recall	0.909	0.938	0.946
F-Measure	0.928	0.954	0.960
Ground Total TP + FP + FN	417 475	417 462	417 458
Avg time: 3.697 sec			

ANSLYSIS:

1. The reduction of False negatives as compared to HPU2 has improved the f-measure making it at par with that of HPU1 experiment
2. The average time for HPU1 exceeds the sum of average time of HPU2 and HPU3 which means our proposal of dividing the problem and focusing on it by parts, worked.

We propose three more HPU experiments whose purpose and mechanism are already described in previous sections. We now tabulate their results.

HPU5:

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	112 11 0 311	117 6 0 302	117 6 0 302
Total Detection	123	123	123
Precision	0.911	0.951	0.951
Recall	0.265	0.279	0.279
F-Measure	0.41	0.431	0.432
Ground Total TP + FP + FN	417 434	417 425	417 425
Avg. time: 1.073 sec			

HPU 6:

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	129 2 0 291	127 4 0 295	125 6 0 297
Total Detection	131	131	131
Precision	0.985	0.969	0.954
Recall	0.307	0.3	0.296
F-Measure	0.468	0.459	0.452
Ground Total TP + FP + FN	417 422	417 426	417 428
Avg. Time: 1.102 sec			

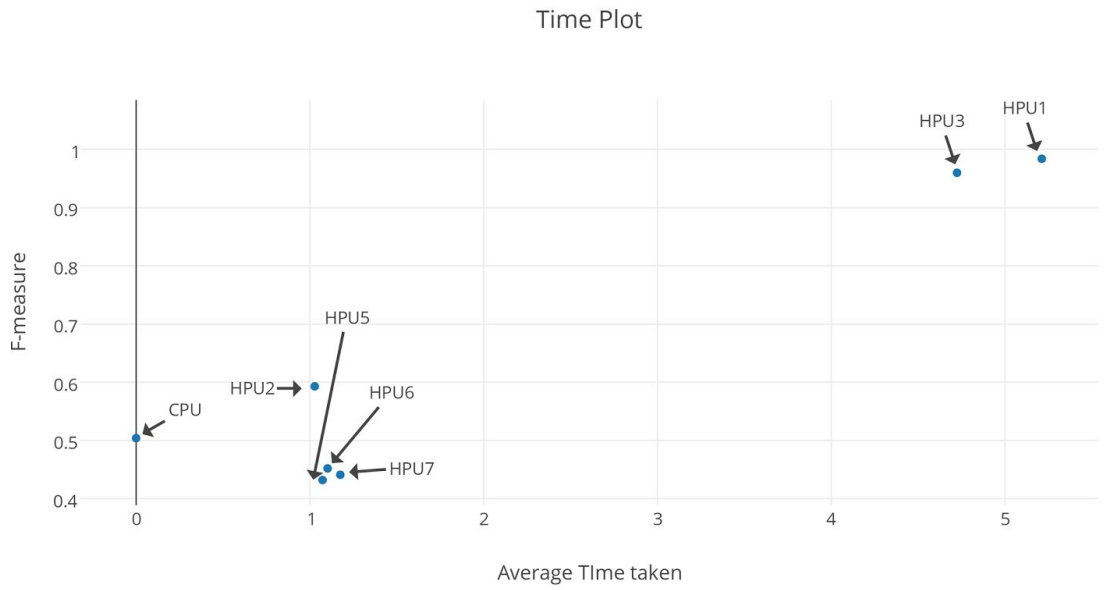
HPU 7:

iou_acceptance	0.2	0.1	0.0
TP FP TN FN	117 7 0 305	118 6 0 303	120 4 0 299
Total Detection	124	124	124
Precision	0.943	0.951	0.967
Recall	0.277	0.28	0.286
F-Measure	0.428	0.433	0.441
Ground Total TP + FP + FN	417 429	417 427	417 423
Avg. Time: 1.175 sec			

The data of the respective HPUs which generated the above table can be found here: Caltech Results

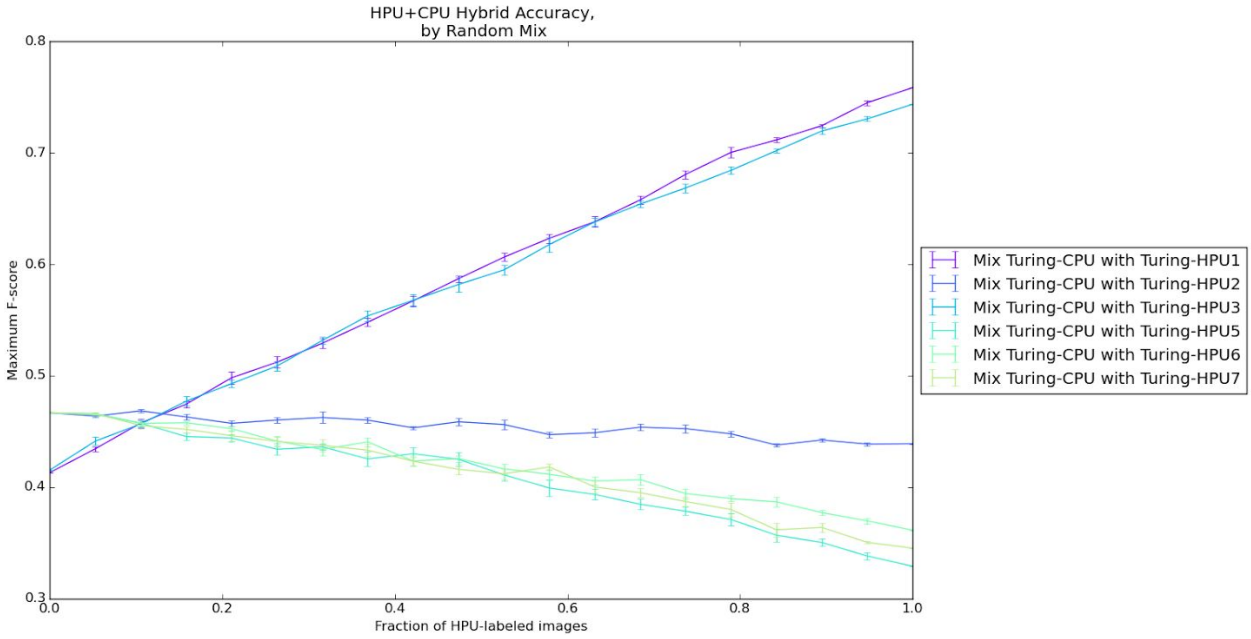
Graphs

HPU time plot for Caltech data

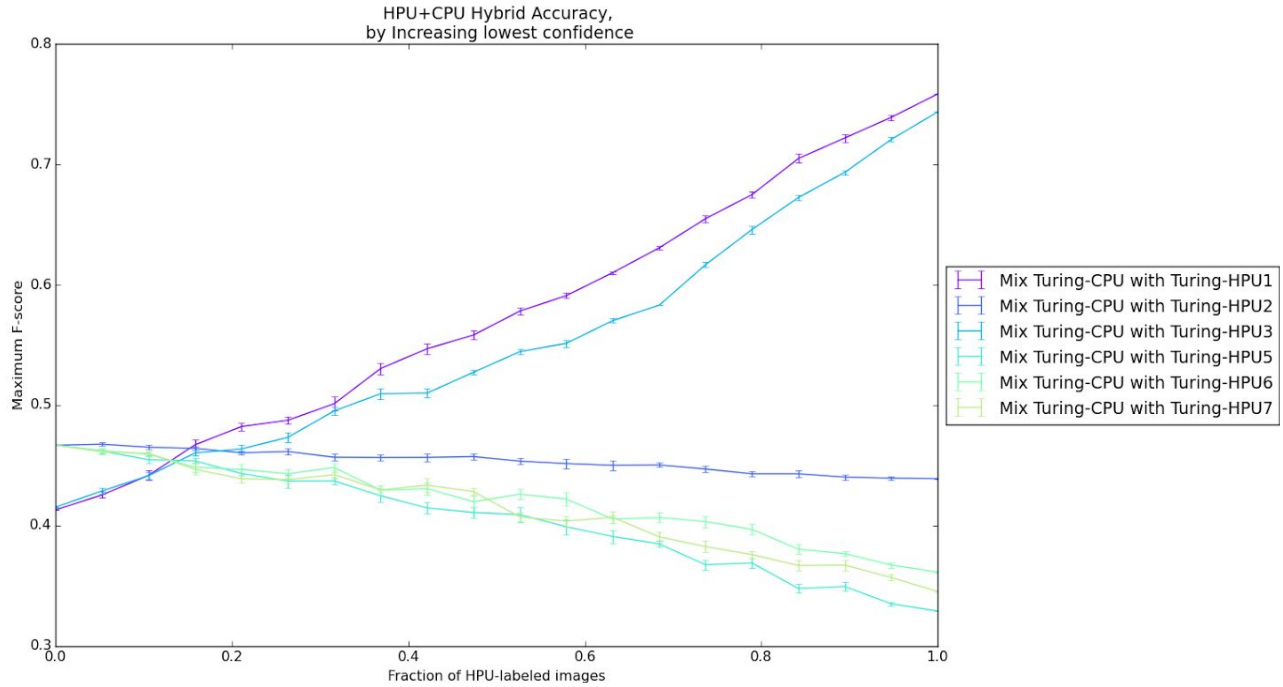


These graphs are plotted using arc-evaluater code. We used a negative confidence threshold.

1. Plot depicting various HPU vs CPU with a random strategy mix



2. With increasing lowest confidence.



In both graphs, HPU1 has stayed above HPU2 as fraction of images has increased, they have both stayed above other units, that too increasing in nature.

COMMENTS on HPU Processing of CALTECH PEDESTAIN DATASET

1. *On IOU ratio for image match* : The way we have calculated the matching of images, is by considering the intersection of the HPU is atleast a fraction iou_acceptance as compared to the Ground Truth values. One may consider the fraction 0.5 to be a nice metric. But considering the smallness of the bounding boxes, the fraction even if it is non-zero, would should also be permissible since the click of a mouse could very well be the reason of the HPU bounding box being some pixels away from the pedestrian (and the IOU calculation would waver a lot in that tiny pixel change of the HPU). So we calculated the metrics considering a fraction of 0.0, 0.1 and 0.2 as the thresholds for bounding box matches.
2. *On the high FN in Caltech Dataset* : Since the pedestrians in the Caltech Dataset are mostly very tiny, the user may miss out on some pedestrian, and thus a pedestrian who might be hidden with the background colours, may miss out the user; which would cause the FN to increase. Hence, if the pedestrians are larger in size, then user would easily mark the bounding boxes correctly and thus, the FN would decrease to a much lower value.

APPENDIX

HOG Prediction for Images
HOG Prediction for Video
Fundan and PennPed Dataset
Intersection Over Union code
HPU code
HPU Portal
Caltech Results