On Trajectory based features and video threads for action recognition

Sourabh Daptardar
Stony Brook University
Stony Brook NY 11790
sdaptardar@cs.stonybrook.edu

Abstract

The project aims to study features and learning methods based on trajectory for action recognition. In particular, the question of automatically selecting video threads and its impact on action recognition performance is investigated. This document outlines the experimentations performed and documents the observations.

1. Introduction

Action recognition techniques started with extending the principles of 2D image object recognition to 3D, for example the work of [2]. However, it was found that the properties of features in the spatial and temporal axes of the video are quite different. Object in a 2D video is not a 3D object, but repetition of the 2D object occurs along the temporal axis. Also, motion provides an important cue for action recognition. These properties were exploited in [3] by using four kinds of feature descriptors for a video : Densely sampled trajectories of flow vectors, Histogram of gradients, Histogram of flow vectors, Motion based histograms. It was further improved in [4] by addition of camera motion estimation and human detection. These "trajectory" based features have been comparable to the state-of-the-art approaches for action recognition. However, there still exists scope for improving these features further to improve the recognition performance. The aim of this project is to devise methods for improving trajectory features, learning algorithms and visualizations.

In particular, we study how action classification is impacted by using video "threads". Shots are segments of video, separated by scene change. Threads are obtained by joining relevant shots together [1]. An example of shots and threads of a video clip of Hollywood2 dataset are shown in figure 1. In the study of video threads,[1] came to the following conclusions: knowing relevant threads (threads containing action) can improve action recognition. Percentage of shots containing action in relevant threads is high. Given these results, we investigate the question: can we automatically select all relevant threads (or equivalently drop irrelevant threads) without impacting recognition performance, given that we have labels for video clips and not the threads or shots?

2. Baseline and visualization

The video classification pipeline described in [4] was implemented. For feature vector computation code of [4] was used. The video classification pipeline consisting of feature vector generation, random sampling, GMM codebook generation, Fisher vector encoding, SVM based classification was implemented and run on Hollywood2 dataset. The results obtained were similar to ones reported in [4]. MAP score obtained was 0.635661 with linear SVM and 0.637513 with LSSVM, while the paper reports 0.643. Details of the classwise average precision values are shown in the table. The details are available as a separate pdf or html report.

In addition to the baseline, a html interface for viewing the results was developed. Figure 2 shows a screenshot of the interface. This interface shows videos with the classifier scores and labels. Color coding enables to quickly identify the misclassification cases.

3. Experiments

3.1. Scoring Threads

After baseline implementation, the pipeline for action classification based on trajectory features was modified. Now, we compute fisher vectors for "threads" in the video. The grouping of threads of scene can provide more meaningful representation of the scene. By dropping irrelevant threads we could possibly improve the performance of the classifier. However, the data we have is labeled on video clips and not on threads. Hence we need to devise strategy for dropping threads which contribute negatively to the classification.

The results of the experiments are summarized in the tables 2 to 4. In each case, we do leave one out cross validation of threads to decide which threads to prune. Following three criteria were attempted:

- **C1:** We train model using all threads and compute corresponding mAP. We leave a thread out and calculate the percentage drop from the baseline mAP. Greater, the drop in the score more is its relevance. This provides a ranking of threads and we retain only a fraction these threads to build the model. For test samples we classify each thread, and the score (SVM decision value) of the highest scoring thread amongst all threads of the video is used to assign label to the video.

- **C2:** Here, we do leave one out for each thread as in C1 but we consider video fisher vectors for rest of the videos instead of thread fisher vectors.

- **C3:** We leave one video out, use the threads of the left video as validation set and train on the rest of the videos. A fraction of threads from each video is retained and used to build the improved model for classification. As in the earlier cases, the video is scored for classification by maximum scores (SVM output decision values) of its threads. Here, median was also attempted instead of maximum. Another variation attempted (column 14) was to use the average of the fisher vectors of all threads of the video along with normalization. This showed a rise in the mAP value by about 2%, above the remaining cases which hover around $58 - 59\%$. However this AP value of 61.226 is still below the baseline of 63.56% on video level classification.

3.2. Scoring Thread Subsets

The experiments 1 to 14 based on criteria C1, C2 or C3 used linear SVM and score threads and use maximum of decision values on all threads of test video. In the next set of experiments, we score every "subset" of threads. The results of these experiments are summarized in tables 5 to 8.

We always start to build our model on video clips for which we have labels. Each video clip is represented by video fisher vector or normalized average of fisher vectors of threads in the video. That forms our training set. We do a leave one out cross-validation, where we "validate" by testing against thread subsets of size n_k, $n_k - 1$, $n_k - 2$, ..., max($n_k - T$,0), and the video fisher vectors themselves, of the left out video. Here n_k is the number of threads for video clip k. In the second phase, we replace each video by the best subset thus obtained and use it for building improved model. Note that this process of substitution is only carried out for the positive samples. The intuition is to drop threads, which do not relate to the positive class. In case of negative samples, we know that none of the threads contain the action. Each thread subset was represented by normalized average of constituent threads. In our test set each video is represented by normalized average of fisher vector of its threads. We use LSSVM instead of linear SVM, as it gives leave one out cross-validation scores along with the training.

Tables 5 to 8, show the average precision values for parameter value T = 0 to T = 3 respectively. As the number of threads can be be exponentially large, we control what subsets should be picked using parameter T. Intuitively, we expect that number of threads inconsistent with the action label would be small. T determines the number of threads that can be dropped to form the thread subset. When T is 0, it means that we do not drop any threads, this is the baseline for comparing results as it gives the classifier where the training and test sets are represented with normalized average of threads. When T is say 2, we consider thread subsets of size n_k, $n_k - 1$, $n_k - 2$, or in other words drop a maximum of 2 threads. In each of the tables 5 to 8, the columns represent a separate experiment. The details of each column is as follows:

- **A:** train set: start with video fisher vectors, validation set: include video fisher vectors along with thread subsets

- **B:** train set: start with video fisher vectors, validation set: do not include video fisher vectors along with subsets

- **C:** train set: start with normalized average of thread fisher vectors, validation set: include video fisher vectors along with subsets

- **D:** train set: start with normalized average of thread fisher vectors, validation set: do not include video fisher vectors along with subsets

- **B1:** simple classification with video fisher vectors

- **B2:** (baseline) simple classification normalized average with video represented by normalized average of thread fisher vectors.

Some observations from the results of tables 5 to 8

- Mean AP values in B2 are of the order of about mere one percent lesser than those in B1. This is very promising as it indicates that normalized average of fisher vector of threads is as good a representation as the fisher vectors of videos themselves.

- Mean AP values in column D, which are obtained through automatic thread subset selection, show improvement over the baseline results in B2

- Mean AP values in column D, are comparable to video fisher vector baseline B1. This shows that despite not knowing the labels of threads we can perform automatic thread subset selection without impacting the quality of video classification pipeline.

4. Conclusion

We started with an action recognition dataset labeled with actions. Threads however, were not labeled as relevant or irrelevant to the action. With our technique based upon leave-one-out cross validation, we can automatically drop irrelevant threads (equivalently retain relevant ones). This can be achieved simultaneously with the use of threads for action recognition and that too without a loss in performance of the action recognition task.

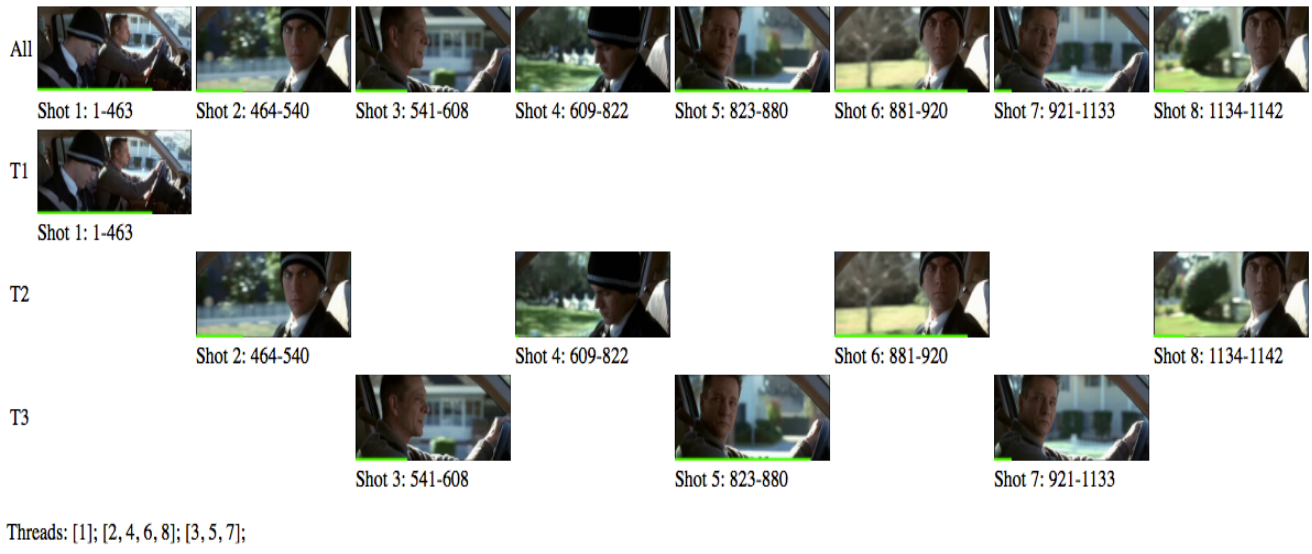

Figure 1. Threads and shots in video

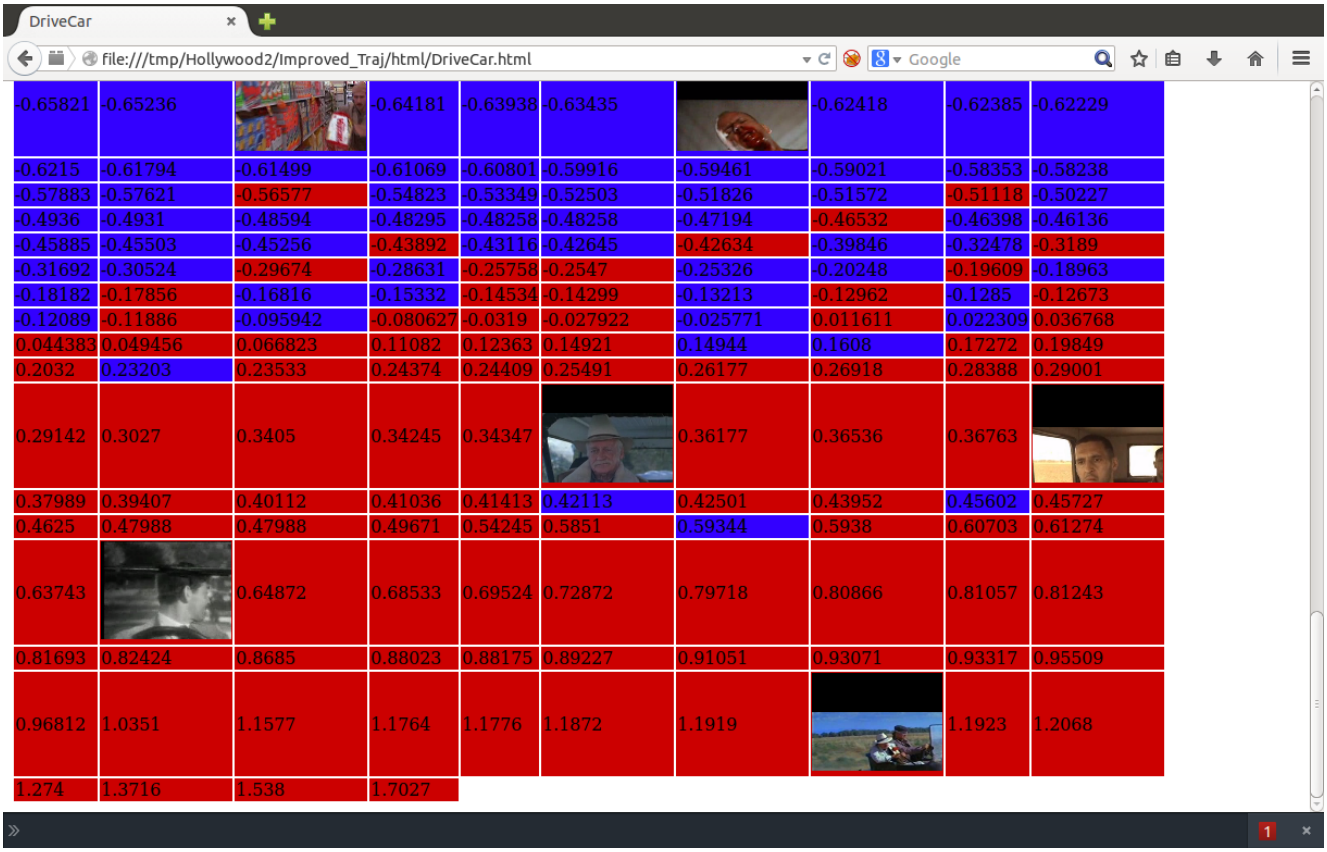

Figure 2. Drive Car classifier: clips ranked by score of the classifier

Classes	AP	CVPR11	PP	P
AnswerPhone	0.333591	0.326000	0	64
DriveCar	0.929323	0.880000	87	102
Eat	0.669142	0.652000	5	33
FightPerson	0.838865	0.814000	50	70
GetOutCar	0.604271	0.527000	5	57
HandShake	0.457570	0.296000	0	45
HugPerson	0.489638	0.542000	3	66
Kiss	0.623824	0.658000	57	103
Run	0.869808	0.821000	123	141
SitDown	0.756986	0.625000	44	108
SitUp	0.281270	0.200000	0	37
StandUp	0.773641	0.652000	76	146

Table 1. Classwise AP values on Hollywood2

Experiment Train on Retain fraction Validation criteria Test (Scoring videos from threads)	Baseline Implementation	CVPR11	1 Threads 0.1 C1 Use max decision value	2 Threads 0.2 C1 Use max decision value	3 Threads 0.25 C1 Use max decision value	4 Threads 0.5 C1 Use max decision value	5 Threads 0.75 C1 Use max decision value
AnswerPhone	0.333591	0.326	0.24883	0.213421	0.245579	0.221443	0.2457
DriveCar	0.929323	0.88	0.877488	0.875922	0.878593	0.877354	0.886206
Eat	0.669142	0.652	0.654786	0.659412	0.656772	0.65885	0.65743
FightPerson	0.838865	0.814	0.693374	0.747628	0.737169	0.738467	0.736127
GetOutCar	0.604271	0.527	0.620498	0.609957	0.613193	0.617877	0.61778
HandShake	0.45757	0.296	0.406341	0.363197	0.376363	0.378495	0.377816
HugPerson	0.489638	0.542	0.375903	0.376777	0.38199	0.392253	0.39286
Kiss	0.623824	0.658	0.595982	0.594928	0.602041	0.598455	0.598954
Run	0.869808	0.821	0.840032	0.83352	0.835776	0.831931	0.838769
SitDown	0.756986	0.625	0.739918	0.739841	0.75125	0.743823	0.746883
SitUp	0.28127	0.2	0.291904	0.277735	0.287831	0.27887	0.289873
StandUp	0.773641	0.652	0.697075	0.700999	0.707007	0.697649	0.716733
Mean	0.63566075	0.58275	0.586844	0.582778	0.589464	0.586289	0.592094

Table 2. Classwise AP values on Hollywood2

Experiment Train on Retain fraction Validation criteria Test (Scoring videos from threads)	Baseline Implementation	CVPR11	6 Threads 1 C1 Use max decision value	7 Threads 1 C2 Use max decision value	8 Threads 0.75 C2 Use max decision value	9 Threads 0.25 C3 Use max decision value	10 Threads 0.2 C3 Use max decision value
AnswerPhone	0.333591	0.326	0.254605	0.245356	0.280336	0.254839	0.254839
DriveCar	0.929323	0.88	0.890472	0.879696	0.878263	0.901175	0.90779
Eat	0.669142	0.652	0.658967	0.653397	0.655547	0.62911	0.62911
FightPerson	0.838865	0.814	0.7458	0.727614	0.739076	0.79839	0.801217
GetOutCar	0.604271	0.527	0.611473	0.598469	0.580679	0.602964	0.602964
HandShake	0.45757	0.296	0.3766	0.377073	0.351254	0.377315	0.377315
HugPerson	0.489638	0.542	0.394058	0.380167	0.372552	0.428628	0.428628
Kiss	0.623824	0.658	0.596313	0.597554	0.590507	0.588624	0.588624
Run	0.869808	0.821	0.840638	0.83677	0.826216	0.837409	0.833311
SitDown	0.756986	0.625	0.747175	0.75184	0.723537	0.757847	0.757847
SitUp	0.28127	0.2	0.287802	0.28665	0.265413	0.268109	0.268109
StandUp	0.773641	0.652	0.699898	0.701513	0.686556	0.731066	0.733786
Mean	0.63566075	0.58275	0.591983	0.586342	0.579161	0.597956	0.598628

Table 3. Classwise AP values on Hollywood2

Experiment			11	12	13	14
Train on						Threads
Retain fraction			0.75	1	1	1
Validation criteria			C3	C3	C3	C3
Test (Scoring videos from threads)	Baseline Implementation	CVPR11	Use max decision value	Use max decision value	Use median decision value	Average Thread Fisher Vectors
AnswerPhone	0.333591	0.326	0.243117	0.245579	0.338449	0.317951
DriveCar	0.929323	0.88	0.886817	0.878593	0.889595	0.928893
Eat	0.669142	0.652	0.663166	0.656772	0.567332	0.666668
FightPerson	0.838865	0.814	0.789077	0.737169	0.577458	0.623793
GetOutCar	0.604271	0.527	0.625795	0.613193	0.581832	0.61097
HandShake	0.45757	0.296	0.391267	0.376363	0.393231	0.446889
HugPerson	0.489638	0.542	0.390983	0.382005	0.317678	0.417411
Kiss	0.623824	0.658	0.598168	0.602041	0.623665	0.629112
Run	0.869808	0.821	0.839893	0.835713	0.834627	0.847685
SitDown	0.756986	0.625	0.749721	0.75125	0.723493	0.745446
SitUp	0.28127	0.2	0.287792	0.287792	0.317415	0.323656
StandUp	0.773641	0.652	0.728948	0.706999	0.785486	0.788642
Mean	0.63566075	0.58275	0.599562	0.589456	0.579188	0.61226

Table 4. Classwise AP values on Hollywood2

Class	B1	B2	A	B	C	D
AnswerPhone	0.352318	0.356223	0.377555	0.356783	0.297425	0.361389
DriveCar	0.937761	0.939435	0.939541	0.937681	0.937325	0.937306
Eat	0.665600	0.657716	0.674482	0.660988	0.662708	0.662708
FightPerson	0.855741	0.812424	0.814248	0.809589	0.813784	0.813224
GetOutCar	0.606999	0.628346	0.617854	0.610082	0.634892	0.629263
HandShake	0.450137	0.419162	0.459539	0.447437	0.429893	0.442413
HugPerson	0.486628	0.431455	0.446085	0.429357	0.436370	0.436816
Kiss	0.613252	0.614991	0.621873	0.620196	0.624699	0.626282
Run	0.858116	0.845533	0.860147	0.852214	0.867071	0.859477
SitDown	0.744193	0.728496	0.738354	0.735339	0.729714	0.738783
SitUp	0.305450	0.298705	0.303194	0.305071	0.307577	0.295033
StandUp	0.773961	0.788922	0.798053	0.794347	0.781939	0.797476
Mean AP	0.637513	0.626784	0.637577	0.629924	0.626950	0.633348

Table 5. AP on Hollywood2: Scoring of thread subsets (T = 0)

Class	B1	B2	A	B	C	D
AnswerPhone	0.352318	0.356223	0.380090	0.344431	0.323755	0.366270
DriveCar	0.937761	0.939435	0.938973	0.936326	0.937917	0.937486
Eat	0.665600	0.657716	0.674461	0.664558	0.663154	0.663154
FightPerson	0.855741	0.812424	0.814889	0.811911	0.815784	0.816340
GetOutCar	0.606999	0.628346	0.621509	0.613855	0.633165	0.630156
HandShake	0.450137	0.419162	0.472444	0.474026	0.452034	0.458637
HugPerson	0.486628	0.431455	0.437729	0.432313	0.423529	0.422981
Kiss	0.613252	0.614991	0.621404	0.622932	0.623124	0.626094
Run	0.858116	0.845533	0.859479	0.855855	0.865767	0.862390
SitDown	0.744193	0.728496	0.738044	0.735358	0.735612	0.740264
SitUp	0.305450	0.298705	0.303904	0.299306	0.306690	0.283643
StandUp	0.773961	0.788922	0.797678	0.793960	0.780524	0.797247
Mean AP	0.637513	0.626784	0.638384	0.632069	0.630088	0.633722

Table 6. AP on Hollywood2: Scoring of thread subsets (T = 1)

Class	B1	B2	A	B	C	D
AnswerPhone	0.352318	0.356223	0.385747	0.362363	0.317132	0.366163
DriveCar	0.937761	0.939435	0.939090	0.938102	0.939516	0.939286
Eat	0.665600	0.657716	0.674382	0.664242	0.661866	0.661866
FightPerson	0.855741	0.812424	0.814957	0.814240	0.814010	0.814457
GetOutCar	0.606999	0.628346	0.621754	0.613788	0.631331	0.631354
HandShake	0.450137	0.419162	0.471409	0.468916	0.455251	0.458004
HugPerson	0.486628	0.431455	0.440131	0.439790	0.418741	0.419413
Kiss	0.613252	0.614991	0.621685	0.622128	0.622700	0.624843
Run	0.858116	0.845533	0.859734	0.856402	0.866771	0.862468
SitDown	0.744193	0.728496	0.736630	0.735240	0.735559	0.741422
SitUp	0.305450	0.298705	0.303904	0.299306	0.306690	0.283643
StandUp	0.773961	0.788922	0.797370	0.793349	0.779161	0.795913
Mean AP	0.637513	0.626784	0.638899	0.633989	0.629061	0.633236

Table 7. AP on Hollywood2: Scoring of thread subsets (T = 2)

Class	B1	B2	A	B	C	D
AnswerPhone	0.352318	0.356223	0.385747	0.362363	0.317132	0.366163
DriveCar	0.937761	0.939435	0.939292	0.938221	0.939112	0.938959
Eat	0.665600	0.657716	0.674382	0.664242	0.661866	0.661866
FightPerson	0.855741	0.812424	0.815251	0.813101	0.813402	0.813647
GetOutCar	0.606999	0.628346	0.621754	0.613788	0.631331	0.631354
HandShake	0.450137	0.419162	0.471409	0.468916	0.455251	0.458004
HugPerson	0.486628	0.431455	0.440131	0.439790	0.418741	0.419413
Kiss	0.613252	0.614991	0.621685	0.622128	0.622700	0.624843
Run	0.858116	0.845533	0.859610	0.856382	0.866658	0.862735
SitDown	0.744193	0.728496	0.736705	0.734804	0.734958	0.740605
SitUp	0.305450	0.298705	0.303904	0.299306	0.306690	0.283643
StandUp	0.773961	0.788922	0.796936	0.793858	0.779161	0.795448
Mean AP	0.637513	0.626784	0.638900	0.633908	0.628917	0.633057

Table 8. Classwise AP values on Hollywood2: Scoring of thread subsets (T = 3)

References

[1] M. Hoai and A. Zisserman. Thread-safe: Towards recognizing human actions across shot boundaries. In *Asian Conference on Computer Vision*, 2014.

[2] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[4] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. IEEE, 2013.