11-775 HW2 Pipelined MED with Video Features

Xin Qian
xinq@cs.cmu.edu

Abstract

In this report, we conduct experiment and performance analysis on the performance of two Multimedia Event Detection (MED) tasks using video features. Results show that feature learning with CNN are significantly stronger than imtraj or SIFT features. In terms of the two approaches to compact the representation of frames into one single vector, BOW is more suitable for dense frame feature vectors while simple average/summation is more suitable for sparse vectors. The three given event classes also have different classification difficulty, P002 is the easiest while P003 is the hardest. Performance does not largely improve when we increase the clustering hyperparameter k from 200 to 1000. It could be because we are using small k for images. Images have more micro features than text, therefore we could hardly port the successful practice from HW1 to HW2.

For grading purpose, the running t2.medium instance ID i-045800720604ac8ab has an associated Elastic IP of 35.164.172.162.

1 Implementation Choice

In this assignment, we implemented a general MED pipeline with three video frame feature setup. As the first component of the pipeline, we use ffmpeg as HW1 to cut the video to retain only the first 30 secs to reduce the amount of data used to represent a video. The 30 secs is a suggested value without losing classification accuracy. However, with unlimited disk space, longer sample will be more desirable.

One of the three setup is using Imtraj pre-extracted features. In the Imtraj/ directory, each video is represented as a single sparse vector. No representation compaction needed. This representation saves up space. During SVM training step, we convert the sparse vector into dense vector again (this is for code simplicity, although leaving it sparse for sklearn SVC is also supported.)

The second implementation choice is SIFT features where we select a number of scale-invariant keypoints to represent each frame. In this way, each video is represented by varies number of vectors from varies number of frames. Each vector has a dimension of 128. We utilized Python version of OpenCV to invoke SIFT feature extractors. The third setup is CNN. The suggested framework is Caffe with its wide selection of pre-trained model zoo. Since our assignment has limited time allowance, utilizing pre-trained ImageNet-sized dataset models will greatly save time without losing effectiveness (since training from scratch is slow, expensive and tricky.) After preliminary exploration, we found that Keras is efficient in deploying with sufficient handy pre-trained model. My final choice is vgg16 model. The input layer for vgg16 require size to be 224*224*3.

During the SIFT and CNN feature extraction, each video has a list of N-dimension vectors. Each audio file contributes a mixed bag of vocal words where we want to cluster k centroids to represent them well. To improve clustering efficiency while retaining clustering quality, I randomly sample

Table 1: 3 fold cross validation results using IMTRAJ features

Event	MAP	Accuracy	TPR	TNR
P001	0.559877	0.9603	0.1349	0.9983
P002	0.658223	0.9628	0.0374	1
P003	0.484921	0.9523	0	1
Average	**0.5677**	0.9585	0.0574	0.9994

Table 2: 3 fold cross validation results using SIFT features with KMeans cluster number as 200

Event	MAP	Accuracy	TPR	TNR
P001	0.455672	0.9587	0.0594	1
P002	0.592748	0.9620	0.0303	1
P003	0.158581	0.9523	0	1
Average	**0.402318**	0.9577	0.0299	1

10% of vectors from all audio files. 10% is a heuristic number worth further examining, however, image tends to contain more features than text in HW1, 10% is the comparative alternative choice from the 20% from HW1. Yet we argue this is a legitimate threshold as it yields around 100 MB aggregated vectors instead of GBs tiny vectors. These sampled vectors were fed into a skilearn KMeans++ function to learn cluster centroids. KMeans++ is a bit more robust than KMeans by overcoming the randomness in clustering seed initialization. Our default cluster number choice is 200, adopted from HW1. We further examined different number of clusters.

We use 3-fold cross validation techniques to examine algorithm robustness. For each event, there contains three rounds, each with 824 training videos and 412 validation videos. Our final test result are trained by the 1236 training videos. Sklearn's SVM function was invoked as the final stage in our pipeline, with the flexibility of altering series of training options, pipelining pre-processing such as standard normalization and training, etc. In short, this section briefly introduces the implementation choice that I made during the pipeline designing process. Next section shows detailed control experiment result where we alter one variable in the pipeline to monitor metric value change.

2 Experiment Result

Below is a list of experiment that explores several minor pipeline variation. All other factores are held equal in each table. Our best performing clssifer for all three classes are from CNN feature, with AGG16 pre-trained Keras model. Imtraj features achieves comparable performance for each of the three classes. SIFT features has distinct performance different for each of the three classes, this could be due to the coarse-grained frame extraction component setup.

2.1 Comparison between Imtraj, SIFT and CNN

Table 1, 2 and 7 demonstrates the comparative performance for Imtraj, SIFT and CNN. CNN has 73.869% while SIFT has an average of 40.23% and Imtraj has an average of 56.77%. Results show that CNN achieves the best performance regardless of the event type. CNN outperforms the other two with a list of factors including analogy over human brains with biological plausibility and the layered hierarchy from similar to complex. Since our task is a classification task instead of an identification task which SIFT tends to win over, CNN wins without much surprise to us.

2.2 SIFT: number of k-means cluster

Table 2, 3 and 4 shows the trend when we increase the number of k-means cluster. K=200 achieves the best performance of 40.23%. Choosing the optimal K value is tricky, unless with cross-validation. Due to time limit, we cross validated K=200, 500 and 1000. A better parameter sweep could be validating around big values first (say up to 10000) and then scale down to reach on best MAP. Our optimal value of 200 could be justified the Elken method from VLFeat, who chosed K=50 for KMeans clustering for SIFT features.

Table 3: 3 fold cross validation results using SIFT features with KMeans cluster number as 500

Event	MAP	Accuracy	TPR	TNR
P001	0.405497	0.9587	0.0594	1
P002	0.601456	0.9620	0.0303	1
P003	0.148947	0.9523	0	1
Average	**0.38501**	0.9577	0.0299	1

Table 4: 3 fold cross validation results using SIFT features with KMeans cluster number as 1000

Event	MAP	Accuracy	TPR	TNR
P001	0.379345	0.9596	0.3730	1
P002	0.644177	0.9628	0.0455	1
P003	0.148517	0.9526	0	1
Average	**0.39068**	0.9583	0.1395	1

Table 5: 3 fold cross validation results using SIFT features with average aggregation

Event	MAP	Accuracy	TPR	TNR
P001	0.273995	0.9563	0	1
P002	0.508633	0.9620	0.0303	1
P003	0.320732	0.9523	0	1
Average	**0.377886**	0.9569	0.0101	1

Table 6: 3 fold cross validation results using SIFT features with summation aggregation

Event	MAP	Accuracy	TPR	TNR
P001	0.2393	0.9563	0	1
P002	0.341398	0.9611	0	1
P003	0.102906	0.9523	0	1
Average	**0.227968**	0.9566	0	1

Table 7: 3 fold cross validation results using CNN Keras VGG16 pre-trained model features with AVG over vectors and rbf kernel

Event	MAP	Accuracy	TPR	TNR
P001	0.743173	0.9620	0.146822	1
P002	0.927116	0.9903	0.0455	1
P003	0.545770	0.9523	0	1
Average	**0.73869**	0.9682	0.1395	1

Table 8: 3 fold cross validation results using CNN Keras VGG16 pre-trained model features with AVG over vectors and sigmoid

Event	MAP	Accuracy	TPR	TNR
P001	0.571548	0.9612	0.1190	1
P002	0.87596	0.9911	0.7980	0.9983
P003	0.40789	0.9523	0	1
Average	**0.633734**	0.9682	0.3057	0.9994

Table 9: 3 fold cross validation results using CNN Keras VGG16 pre-trained model features with AVG over vectors and linear kernel

Event	MAP	Accuracy	TPR	TNR
P001	0.504142	0.9563	0	1
P002	0.873448	0.9684	0.2707	0.9992
P003	0.414954	0.9523	0	1
Average	**0.607514**	0.9590	0.0902	0.9997

Table 10: 3 fold cross validation results using CNN Keras VGG16 pre-trained model features with AVG over vectors and poly kernel

Event	MAP	Accuracy	TPR	TNR
P001	0.619324	0.9595	0.0754	1
P002	0.877428	0.9862	0.6767	0.9992
P003	0.404450	0.9523	0	1
Average	**0.633734**	0.966	0.2507	0.9997

2.3 Feature representation compaction

Table 2, 5 and 6 compares different options to compact the representation of the video into a single vector. It shows that SIFT works best fro BOW with KMeans. It could be because SIFT is rather dense while for sparse vectors like CNN features, simple average over all vectors over all vectors extracted from the video will be sufficiently effective.

2.4 CNN with Keras: kernel function choice

Up to this point, our experiment relies on the default setting of sklearn default SVC parameter setting. Our final experiment was based on the Kernel functions of the SVM model on CNN Keras features. In table 7, 8, 9 and 10, we see that rbf kernel wins over all events with the highest MAP value over 70%. However, a more justified way of doing so is to run a grid search by specifying a list of parameters to be tuned and using sklearn's GridSearchCV method to tune them.

3 Evaluation

3.1 Metrics

In this experiment, we used MAP, Accuracy, TPR and TNR to evaluate model effectiveness. Accuracy, also named as Rand Index (RI),which measures the percentage of decisions that are correct, which is simply accuracy:

$$RI/Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

From above tables, we could see MAP is more indicative than Accuracy in our dataset. This reveals a shortback of Accuracy. When the class distribution is highly skewed, Accuracy could be high even it cannot classify one single correct positive instance. This is because the majority of training instances are negative cases, which makes TP (True Positive) case extremely rare and TN case dominant in both training and test set. It makes no surprise that Accuracy stabilize at over 90% for all methods since all the methods have little confidence in classify the positive case right.

On the other hand, MAP treats MED as a search task, where a list of possible relevant document are returned with relevancy probability (in our case, it's the probability of classify the case as positive). In all above tables, MAP values are indicative in terms of classification/retrieval effectiveness.

The introduction of TPR and TNR in this assignment mitigates the problem. TPR and TNR serves as supplementary metrics for us to diagnose. Since our ultimate goal is to classify positve class correct, it would be better if one method has comparatively higher TPR with slightly lower MAP. However, in most senario as show in the tabels above, the correlations between TPR and MAP are high.

4 Future work

4.1 Hierarchical feature vector aggregation

In our implementation, we concatenate feature vectors from within one frame to the feature vectors from other frames, however, if we For feature vectors within the same frame, how do we handle them with vectors from other frames. There's a hierarchy between these vectors. A solution is to average vectors from the same frame, run a Kmeans clustering and assign a BOW for each video.

4.2 Video scale-invariant sampling

As our pipeline basic ballpark is 60% for CNN, we wonder if increasing the frame size and the length of video snippet will increase the performance. In the video of making a cake, the first 30s could be tedious introduction instead of showing key instructions, which is noisy. Instead of extracting a constant video length of 30s, choosing the middle 40%-60% content could be more robust. Depending on the event's nature, video can have distinct length distribution. For example, assembling a shelter could be lengthy while batting in a run could last only several seconds.

4.3 Event nature and classification accuracy upperbound

It is yet unknown to me how to model a classification accuracy upper bound given the nature of each event. In our dataset, the three event differs in classification difficulty. This difficulty is technique specifict. The easiest one in video extraction and image classification is P002. Most likely because the scenary nature of video images during a batting in a run event. However, the easiest one in ASR classification was P003, making a cake, which involves a oral tutorial full of semantic meaning (baking instructions) while the hardest one bing P001 most likely because assembling a shelter involves less oral interaction and more environmental noise.

References