Homework 3: Large-Scale MED

Anusha Prakash
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
`Andrew ID : anushap`

Abstract

The goal of this assignment is to build an event classifier using video and audio features by fusing them together. In other words the task is to perform multimedia event detection (MED) using the video features like : Speeded Up Robust Features (SURF), Improved Dense Trajectories (IDT) and Convolutional Neural Network (CNN) features and audio features like Mel-frequency cepstrum coefficients (MFCCs), ASR Transcriptions, SoundNet features. MED detectors are trained and tested with feature fusion strategy and predictions are made on 3 types of events : P001: assembling_shelter; P002: batting_in_run; P003: making_cake. The evaluation is done by calculating Mean Average Precision (MAP) across the 3 types of events.

1 Multimedia Event Detection

The amount of multimedia data being recorded, stored and shared is increasing daily, and the total sum of data is astronomical. For example, just on the site YouTube, video data is being uploaded at the rate of one hour a second or 31 million hours a year. This yields practical problems when analysts or laypersons want to extract information from this huge data collection. What is needed are tools for automatic tagging, indexing and searching of multimedia data. This is clearly a grand challenge and needs to be divided into manageable sub-problems. One of the interesting sub-problems is automatic event detection in videos.

Multimedia event detection (MED) is the task of detecting given events (e.g. bomb explosion, teaching dance) in a large collection of video clips. Visual features and automatic speech recognition (ASR) typically provide the best features for this task. MED plays an important role in many applications such as video indexing and retrieval. MED is typically formulated as a two-stage process: the first stage generates clip-level feature representations, often by aggregating frame-level features; the second stage performs binary or multi-class classification to decide whether a given event occurs in a video clip. Both stages are usually performed "statically", i.e. using only local temporal information, or Bag-of-Words models.

2 MED Pipeline

The various steps in the MED pipeline are elaborated below :

1. **Feature Extraction**

   Features can be visual or audio Features. Among these two features, the features can be further classified into Low-level, High-level and Text features. Few of the visual Features are SIFT, VGG19, CNN (Low-level), Semantic Concepts (High-level), Optical Character Recognition (Text) etc. Audio Features include MFCC (Low-level), Acoustic

Scene Analysis (High-level), Automatic Speech Recognition (Text) etc.

(a) **AUDIO FEATURES**

- **MFCC Features** : They are Low-level audio features. Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel-scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. In *audio_mfcc_feat_extraction.py*, mono audio tracks (.wav) are extracted from the video files using *ffmpeg*. The *librosa* package is used to extract the MFCC features from the audio tracks.

  In *librosa*, *n_mfcc* - Number of MFCCs to return - 20 by default. Using more MFCC components (n_mfcc) could obtain more fine-grained audio features. Hence experimented with n_mfcc=20 and n_mfcc=40 and choose n_mfcc=40 for the final feature extraction.

- **SoundNet Features** : The SoundNet developed by the MIT CSAIL Lab learns rich natural sound representations by capitalizing on large amounts of unlabeled sound. Raw features from different layers (conv3, conv4, conv6 and conv8) of SoundNet pre-trained model are extracted using an open-source Tensorflow implementation.

  Experimented with different SoundNet Layers' features and it's effect on fusion performance.

Reason for selecting MFCC and SoundNet Features :

ASR Features are obtained either using the Python *speech_recognition* toolkit with *CMU Sphinx* offline engine or the EESEN toolkit, which is an end-to-end speech recognition software using Deep Learning. ASR feature extraction was taking a long time per audio file and the vocabulary size was huge owing to the the number of audios and how long they were. The AWS credits are also limited.

And based on the discussion of HW1, MFCC and ASR had comparable performance and SoundNet performed the best among the three. Extracting MFCC and SoundNet features are relatively easier as explained above. Due to all these reasons, choose MFCC and SoundNet features over ASR features.

(b) **VIDEO FEATURES**

- **CNN Features** : SURF and IDT are hand-crafted features. Convolutional neural networks (CNN) is a learned feature which is expected to perform much better. Feature learning with convolutional neural networks (CNNs) is a hot topic in computer vision and many other fields.

  In *cnn_feat_extraction.py*, the *keras* toolkit is used to extract the image-level CNN features on videos' key-frames. Experimented with *ResNet50* and *VGG19* imagenet pre-trained model for CNN feature extraction. The last layer output which has dimension 1000 is used for the feature extraction from the video frames. Every 10th frame of the video was sampled for the purpose of feature extraction, i.e 10% sampling of the keyframes was done. Initially tried with 2% sampling, but increasing the sampling rate lead to a considerable increase in MAP value and hence 10% sampling was used for final feature extraction.

Reason for selecting CNN Features :

SURF and IDT are hand-crafted features while CNN is a learned feature. IDT feature extraction and encoding takes extremely long time and SURF too is time-consuming. Moreover based on the discussion of HW2, CNN clearly out-performed SURF and IDT by a large margin. This is expected too as it is a learned feature unlike the other two hand-crafted features. Hence, choose CNN features over SURF and IDT features.

2. **Feature Representation**

   In this step, preprocessing is performed on the extracted raw features to pack them into the final representation, so that each video is represented by a single feature vector. Few of the techniques for Feature Encoding include VLAD, Histograms, Spatial Tiling, BoW etc. For video with no audio tracks, exception handling is done by using *all-0 vector* for these kind of videos.

   - **MFCC Features** : For MFCC features, random sampling of 20% MFCCs from each video is done, and this sampled data is used to train a k-means clustering model to cluster the MFCC vectors then represent a video using these cluster centers. This technique will reduce the size of data for k-means clustering and thus speed up the pipeline development. Once clustering is done, each video is represented in a k-dim vector according to the histogram where each dimension indicates the counts of MFCCs to the "closest" cluster center in *mfcc_bag_of_words.py*. Also implemented the VLAD encoding to represent the audio features of each video.

     Experimented with cosine similarity and euclidean distance measures while creating the BoW representations. Found euclidean distance to be more suitable for audio features. Also upon reviewing few state-of-the-art papers, realized that most of them also employ euclidean distance measure. Also experimented with 30% sampling, but this made the clustering process very slow and the performance improvement was not very significant. Hence decided to use 20% sampling itself. For k-means clustering, experimented with cluster size of 200, 300 and 400. 400 clusters gave the best performance, hence 400 clusters is used in the pipeline.

   - **SoundNet Features** : SoundNet features are 4-dimensional arrays in (N, H, W, C) format, whereas N is the batch size of the feature, H and W is the size of feature maps and C is the number of filters. For a particular convolutional layer the 4-d arrays are reshaped to 2-d arrays by keeping the dimension C constant. Once this is done, similar to MFCC feature encoding, Bag-of-Words representation with k-means clustering is implemented in *soundnet_features.py*. Also implemented the VLAD encoding to represent the audio features of each video. But in SounNet, the entire data is used instead of doing a 20% sampling. The value of C is different for different convolutional layers : "conv3" - 64, "conv4" - 128, "conv6" - 512 and "conv8" - 1000.

     Also experimented with MAX, MEAN, MIN features instead of doing k-means clustering and taking a BoW representation.

   - **CNN Features** : The last layer of *VGG19* and *ResNet50* has output dimension of 1000, which can be handled by the classifier directly. Hence in this case, no extra feature encoding was performed. However, the frame-level features were converted with video-level features by performing Pooling in *cnn_feat_extraction.py*. Experimented with Average Pooling and Max Pooling and concluded that Average Pooling gave better results.

3. **Fusion Schemes**

   The final three features experimented with are MFCC, SoundNet (SN) and CNN features. 3 fusion schemes were implemented - Early, Late and Double Fusion. Also trained classifiers with no fusion. The details are explained below :

   (a) **Early Fusion** : In a practical MED system which relies on early fusion for decision, it firstly extracts individual features separately. The extracted features are then combined into a single vector representation for each video. A commonly used feature combination strategy is concatenating vectors from different feature extractors into a long vector. After combination of individual feature vectors for a multimodal representation, the supervised classifiers are employed for classification.

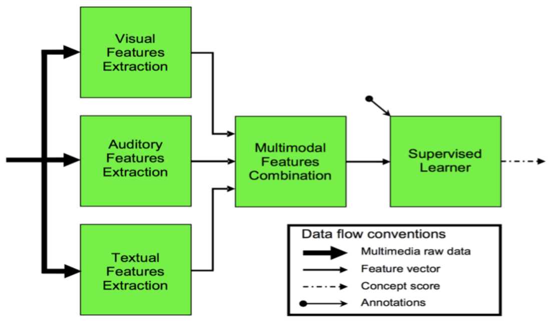
Visual Features Extraction
Auditory Features Extraction
Textual Features Extraction
Multimodal Features Combination
Supervised Learner
Data flow conventions
Multimedia raw data
Feature vector
Concept score
Annotations

Figure 1: Early Fusion

For early fusion, combined the encoded audio and video features into one big feature vector and passed it into a classifier for learning. Tried out 3 combinations : 1) MFCC + CNN, 2) SN + CNN, 3) MFCC + SN + CNN. The results of these three combinations and it's analysis is discussed in the results section.

(b) **Late Fusion** : A MED system which uses late fusion for classification also starts with extracting different feature descriptors. In contrast to early fusion, where features are then combined into a multimodal representation, approaches for late fusion firstly learn separate supervised classifiers directly from unimodal features. In the test phase, the prediction scores from different models are then combined to yield a final score. In general, late fusion schemes combine learned unimodal scores into a multimodal representation. Compared to early fusion, late fusion focuses on the individual strength of modalities.

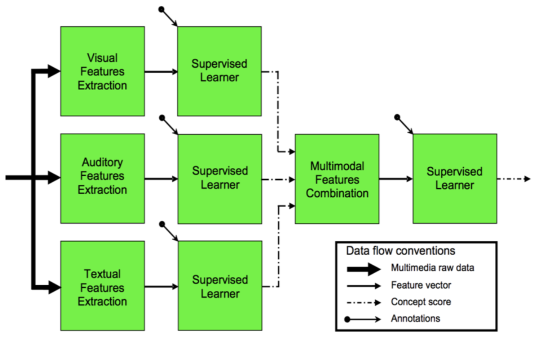

Figure 2: Late Fusion

There are many approaches for combining the scores. One approach is to learn a final score for the concept after fusion. Another approach is to do a average or weighted average of the scores. Implemented the latter averaging approaches. Again experimented with 3 combinations : 1) MFCC + CNN, 2) SN + CNN, 3) MFCC + SN + CNN. The results of these three combinations and it's analysis is discussed in the results section.

(c) **Double Fusion** : In double fusion, we first perform early fusion to generate different combinations of features from subsets on the single features pool. After that, we train classifiers on each feature or feature combination and carry out late fusion on the output of these classifiers.

Here again, made use of three features MFCC, SN and CNN. Experimented with 3 combinations : 1) MFCC, CNN, MFCC+CNN 2) SN, CNN, SN+CNN 3) MFCC, SN, CNN, MFCC+CNN, SN+CNN. The results of these three combinations and it's analysis is discussed in the results section.

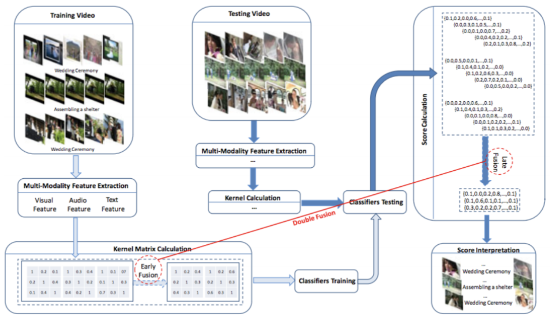

Figure 3: DoubleFusion

(d) **No Fusion** : In no fusion scheme, individual classifiers are trained on each of the 3 features separately, without any fusion. This is done to compare the individual performance with the fusion performance and also to see if the features can actually perform better or worse without any fusion. The 3 features used here are MFCC, SN and CNN. The results are discussed in the later section.

4. **Event Prediction and Evaluation** :

   Once the classifier is trained, in the testing phase, for each video, a decision score indicating how likely this video belongs to the event P00X is obtained. In *evaluator.py*, Average Precision (AP) and Mean Average Precision (MAP) is calculated based on the prediction scores for the 3 events.

3 Results and Analysis

The Mean Average Precision (MAP) is calculated on the self-defined validation set which contains 412 videos. The ratio of videos belonging to each category (P001, P002, P003,NULL) is maintained same within the train and valid set. Results for each type of fusion scheme is documented below :

The first table in each section documents the performance in terms of AP and MAP for each technique and each combination of features. The second table in each section documents the classifier and the penalty parameter used to obtain that performance.

Classifiers : The various classifiers experimented with to improve the MAP are : LogisticRegression, SVC, LinearSVC and AdaBoostClassifier. Slack parameter hyper tuning was done using GridSearch to obtain the best MAP value.

Parameters : The parameters used for testing on Validation Set are listed below. The various approaches / parameters tried has already been explained in the previous sections.

- MFCC feature extraction, *n_mfcc* = 40
- MFCC - 20% random sampling of the data and k = 400 clusters for BoW and VLAD
- MFCC Distance measure used - Euclidean distance using *numpy.linalg.norm* for BOW
- SoundNet - Entire data (No random sampling) and k = 200 clusters for BoW and VLAD - conv4
- SoundNet - Entire data (No random sampling) and k = 400 clusters for BoW - conv3
- SoundNet Distance measure used - Euclidean distance using *numpy.linalg.norm* for BOW

- CNN feature extraction - *VGG19* network and *ResNet50*, *last layer* - 1000 dimension
- CNN feature extraction - 10% key-frames of each video sampled and used (every 10th frame), Average Pooling

1. **SELECTED FINAL RESULT** :

Performance on Validation Set - Final Feature CNN				
Features	**MAP**	**Events - Avg Precision**		
		P001	P002	P003
CNN (RESNET50)	0.73	0.73	0.90	0.56

Table 1: Performance on Validation Set - Final Feature CNN

SVM Classifiers - Final Feature CNN			
Features	**Events**		
	P001	P002	P003
CNN (RESNET50)	SVC(kernel='linear',C=0.01)	AdaBoostClassifier()	AdaBoostClassifier()

Table 2: Classifier with Kernel and Penalty (C) Parameters - Final Feature CNN

Finally the predictions submitted is CNN **ResNet50** feature prediction. A weighted average of CNN, MFCC and SN was done where in MFCC and SN were assigned weight 0 in Late Fusion Technique.

Based on the ablation studies, which are reported below, either of the fusions only decreased the MAP value, where as using CNN feature alone gave the highest MAP. Due to this reason, CNN was chosen as the primary feature.

In CNN too, after experimenting with *VGG19* and *ResNet50*, it was observed that **ResNet50** gave a slightly better performance, as a result, **ResNet50** was used as the final feature.

2. **No Fusion Results** :

Performance on Validation Set - No Fusion				
Features	**MAP**	**Events - Avg Precision**		
		P001	P002	P003
MFCC (BOW)	**0.32**	**0.29**	**0.58**	**0.10**
MFCC (VLAD)	0.34	0.25	0.70	0.06
SN (CONV3 + BOW)	0.33	0.32	0.49	0.17
SN (CONV4 + BOW)	**0.35**	**0.33**	**0.51**	**0.21**
SN (CONV4 + VLAD)	0.32	0.25	0.60	0.11
CNN (VGG19)	**0.71**	**0.82**	**0.80**	**0.52**
CNN (RESNET50)	0.73	0.73	0.90	0.56

Table 3: Performance on Validation Set - No Fusion

- **MFCC** : Experimented with BoW and VLAD encoding. BoW gave a more balanced MAP value across the events than VLAD for MFCC and also took much lesser time to classifiy and hence was chosen.

- **SoundNet** : Experimented with convolutional layer 3 and 4 features for SoundNet. Conv4 performed better in terms of the overall MAP and the and the individual APs. For conv4, experimented with BoW and VLAD encoding. BoW in general gave a better performance than VLAD for SoundNet and hence was chosen.

SVM Classifiers - No Fusion			
Features	**Events**		
	P001	P002	P003
MFCC (BOW)	**AdaBoostClassifier()**	**LinearSVC(C=0.000001)**	**SVC(kernel='linear')**
MFCC (VLAD)	AdaBoostClassifier()	LinearSVC(C=0.000001)	SVC(kernel='linear')
SN (CONV3 + BOW)	AdaBoostClassifier()	AdaBoostClassifier()	AdaBoostClassifier()
SN (CONV4 + BOW)	**SVC(kernel='rbf',C=100)**	**AdaBoostClassifier()**	**AdaBoostClassifier()**
SN (CONV4 + VALD)	SVC(kernel='rbf',C=100)	AdaBoostClassifier()	AdaBoostClassifier()
CNN (VGG19)	**SVC(kernel='linear')**	**SVC(kernel='linear',C=100)**	**LinearSVC()**
CNN (RESNET50)	SVC(kernel='linear',C=0.01)	AdaBoostClassifier()	AdaBoostClassifier()

Table 4: Classifier with Kernel and Penalty (C) Parameters - No Fusion

- **CNN** : Experimented with last layers of *VGG19* and *ResNet50*. Initially only tried *VGG19* and hence all the fusion is performed only using *VGG19*. Once it was clear that CNN outperforms any fusion technique, also tried out *ResNet50*.

The final features chosen for other fusion techniques are bolded in the table

CNN gives the best performance, followed by SN and then by MFCC. This is expected as CNN is a learnt video feature which is very rich, and SN is an audio features extracted from a pre-trained neural network, where as MFCC is a low-level audio feature.

3. **Early Fusion Results** :

Performance on Validation Set - Early Fusion				
Features	**MAP**	**Events - Avg Precision**		
		P001	P002	P003
MFCC + CNN	**0.62**	**0.56**	**0.81**	**0.47**
SN + CNN	0.59	0.46	0.72	0.59
MFCC + SN + CNN	**0.63**	**0.64**	**0.86**	**0.40**

Table 5: Performance on Validation Set - Early Fusion

SVM Classifiers - Early Fusion			
Features	**Events**		
	P001	P002	P003
MFCC + CNN	**AdaBoostClassifier()**	**AdaBoostClassifier()**	**AdaBoostClassifier()**
SN + CNN	AdaBoostClassifier()	AdaBoostClassifier()	AdaBoostClassifier()
MFCC + SN + CNN	**AdaBoostClassifier()**	**AdaBoostClassifier()**	**AdaBoostClassifier()**

Table 6: Classifier with Kernel and Penalty (C) Parameters - Early Fusion

The results are summarized in the tables. Based on the experiments, the following things can be concluded :

- When MFCC is combined with CNN, a better performance is obtained than when MFCC is used alone without fusion. CNN is a stronger feature and hence leads to a better MAP when MFCC is combined with it.

- When SN is combined with CNN, a better performance is obtained than when SN is used alone without fusion. CNN is a stronger feature and hence leads to a better MAP when SN is combined with it.

- CNN being a learned feature is by itself a strong feature. By combining it with MFCC or SN or both, the performance goes down. Although combining it with MFCC gives a better performance than when combined with SN, for early fusion and combining with both gives the best performance amongst the combinations.

4. **Late Fusion Results** :

Performance on Validation Set - Late Fusion (Avg)				
Features	**MAP**	**Events - Avg Precision**		
		P001	P002	P003
MFCC + CNN	0.55	0.71	0.83	0.11
SN + CNN	0.65	0.739	0.83	0.39
MFCC + SN + CNN	0.53	0.65	0.84	0.11

Table 7: Performance on Validation Set - Late Fusion (Avg)

The results are summarized in the tables. The averaging technique is employed in late fusion instead of learning the final scores. Equal weightage is given to all the features. Based on the experiments, the following things can be concluded :

- When MFCC is combined with CNN, a better performance is obtained than when MFCC is used alone without fusion. CNN is a stronger feature and hence leads to a better MAP when MFCC is combined with it.

- When SN is combined with CNN, a better performance is obtained than when SN is used alone without fusion. CNN is a stronger feature and hence leads to a better MAP when SN is combined with it.

- CNN being a learned feature is by itself a strong feature. By combining it with MFCC or SN or both, the performance goes down. Although combining it with SN gives a better performance than when combined with MFCC, for late fusion and that is the best performance amongst the combinations, followed by MFCC+CNN and by MFCC+SN+CNN

- MFCC+CNN and MFCC+SN+CNN performs better in the case of Early than Late fusion, while SN+CNN performs better in the case of Late than Early fusion. This may be because MFCC features are better captured when combined through early fusion while SN features are better captured through late fusion.

5. **Double Fusion Results** :

Performance on Validation Set - Double Fusion (Avg)				
Features	**MAP**	**Events - Avg Precision**		
		P001	P002	P003
MFCC, CNN, MFCC+CNN	0.56	0.70	0.82	0.15
SN, CNN, SN+CNN	**0.68**	**0.68**	**0.82**	**0.54**
MFCC, SN, CNN, MFCC+CNN, SN+CNN	0.56	0.67	0.83	0.19

Table 8: Performance on Validation Set - Double Fusion (Avg)

The results are summarized in the tables. The averaging technique is employed in late fusion part of double fusion instead of learning the final scores. Equal weightage is given to all the features. Based on the experiments, the following things can be concluded :

- In the case of MFCC and CNN combination, Early Fusion performs best, followed by Double and Late fusion. This is definitely better than using MFCC feature alone.

- In the case of SN and CNN combination, Double Fusion performs best, followed by Late and Early fusion. This is definitely better than using SN feature alone.

- CNN being a learned feature is by itself a strong feature. By combining it with MFCC or SN or both, the performance goes down.

- Although combining it with SN gives a better performance than when combined with MFCC, for double fusion and that is the best performance amongst the combinations, followed by MFCC,CNN,MFCC+CNN and MFCC,SN,CNN,MFCC+CNN,SN+CNN, which are both equal.

- Early fusion performs best by combining all 3 features - MFCC, SN, CNN, followed by double and then by late fusion.

Best performing fusion scheme in each combination is highlighted in the tables

- **CPU Time :** It took the entire duration of the assignment time to extract all the features and get the prediction results out.

- **Credit left on AWS account :** No credits left