IBM Research and Columbia University TRECVID-2012 Multimedia Event Detection (MED), Multimedia Event Recounting (MER), and Semantic Indexing (SIN) Systems

Liangliang Cao,[†] Shih-Fu Chang,[*] Noel Codella,[†] Courtenay Cotton,[*] Dan Ellis,[*]
Leiguang Gong,[†] Matthew Hill,[†] Gang Hua,[†] John Kender,[‡] Michele Merler,[†] Yadong Mu,[*]
John R. Smith,[†] Felix X. Yu[*§]

Abstract

For this year's TRECVID Multimedia Event Detection task, our team studied high-level visual and audio semantic features, mid-level visual attributes, and sophisticated low-level features. In addition, a range of new modeling strategies were studied, including those that take into account temporal dynamics of event semantics, optimize fusion of system components, provide linear approximations of non-linear kernels, and generate synthetic data for the limited exemplar condition.

For the Pre-Specified task, we submitted 4 runs: Run 1 involved the fusion of a broad array of sophisticated low-level features. Run 2 involved the same set of low-level features to model the events under the limited exemplar condition. Run 3 involved the fusion of all our semantic system components. Run 4 was composed of the fusion of all low-level and semantic features used in Runs 1-3, in addition to event models built from techniques for linear approximation of non-linear kernels. For Ad Hoc, we submitted 2 runs: Run 5, which was the fusion of Linear Temporal Pyramids of visual semantics, fused with event models built directly on low-level features. Run 6 was our limited exemplar run, which used both Linear Temporal Pyramids of visual semantics, as well as a method for generating synthetic training data.

Our experiments suggest the following: 1) Semantic modeling improves the event modeling performance of the low-level features they are based on. 2) Mid-level visual attributes contribute complimentary information. 3) Event videos demonstate temporal patterns. 4) Linear approximation methods to non-linear kernels perform similarly to the original non-linear kernels, and hold promise to improve event modeling performance by allowing a scaling up to a broader array of models.

[*]Columbia University, Dept. of Electrical Engineering

[†]IBM T. J. Watson Research Center

[‡]Columbia University, Dept. of Computer Science

[§]Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20070. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

1 System

1.1 Overview

The overall system and process flow is illustrated in Figure 1. As depicted, one process flow extracts and models low-level features using various scene descriptors (visual), SIFT/GIST (local visual features), MFCCs (audio), and STIP features (spatio-temporal). Similarly, a high-level feature process flow extracts and models visual and audio semantics. These provide the basis for the information used to detect events in the video. Subsequent modeling from this information provides prediction of the events. We explored various techniques for combing these predictions in late fusion to make the overall decision about event detection.

1.2 Low-level Feature Extraction

1.2.1 Video processing

We decode the video clip, and uniformly save one frame every two seconds. These frames are then used to extract static visual low-level descriptors. We extract over 100 types of static image features from each of the sampled frames. These features capture a wide range of image information including color, texture, edge, local appearances and scene characteristics. We chose 0.5 fps as a sampling rate based on the data set size in order to yield a number of frames that we could process in a reasonable time.

Our system uses a subset of these low-level features to determine the semantic content of video frames, from which further event modeling is performed. Semantic content is extracted at a slower rate of 0.25 fps due to the added complexity of evaluating the models.

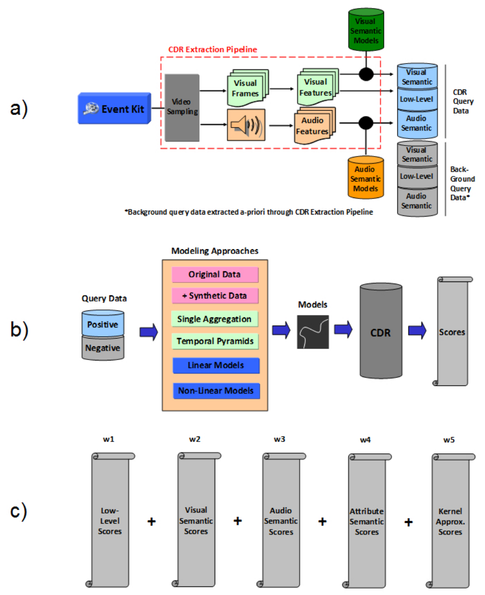


Figure 1: IBM/Columbia mutlimedia event detection system. a) Low-level features extracted from videos. b) Each feature type is used to build an event query model using one of several approaches. c) Score tables are combined by weighted fusion.

1.2.2 Low-Level Descriptors for Semantic Analysis

A combination of local and global descriptors are extracted for the analysis of the visual semantic content of video frames. Features extracted include some standard ones, such as SIFT, LBP, GIST, Color Histogram, etc, as well as a new proprietary feature referred to as a Fourier Polar Pyramid.

Each descriptor is evaluated at various spatial granularities that include global, center, cross, grid, horizontal parts, horizontal center, vertical parts and vertical center – each of which is a fixed division of the image frame into square blocks (numbering from 1 up to 25), and then concatenating the descriptor vectors from each block. Such spatial divisions has been repeatedly shown robust performance in image/video retrieval benchmarks such as TRECVID [12].

SIFT [8] features are extracted with Harris Laplace interest point detection for the sampled frames. Each keypoint is described with a 128-dimensional vector containing oriented gradients. We obtain a “visual keyword” dictionary of size 1000 (by running K-means clustering on a random sample of approximately 300K Interest point features, we then represent each frame with a histogram of visual words. For keyframes we used soft assignment following Van Gemert et al. [15] using $\sigma = 90$.

Local Binary Patterns (LBP) [13] are extracted across two image granularities: global, and a 5x5 grid. The LBP histogram is extracted from the greyscale version of the image as a histogram of 8-bits local binary patterns, each of which is generated by comparing the grayscale value of a pixel with those of its 8 neighbors in circular order, and setting the corresponding bit to 0 or 1 accordingly. A pattern is called uniform if it contains at most two bitwise transitions from 0 to 1. The final histogram for each region in our granularity contains 59 bins, 58 for uniform patterns and 1 for all the non-uniform ones.

In addition to the SIFT bag-of-words and LBP descriptors, we extracted 13 different visual descriptors at 8 granularities and spatial divisions, including Color Histogram, Color Correlogram, Color Moments, Wavelet Texture, Edge Histogram, and GIST. SVMs are trained on each feature and subsequently linearly combined in an ensemble classifier. Details on features and ensemble classifier training can be found in our prior report [1, 2].

We have also developed a proprietary feature referred to as a Fourier Polar Pyramid. It incorporates ideas from both spatial pyramids and from the Curvelets feature transform. The basic idea is to construct a spatial pyramid in Fourier space, under the polar coordinate system, across all 3 color channels red, green, blue, in addition to a grayscale color channel. Pyramid levels in the radial dimension consist of 1, 2, 4, and 8 partitions. For each of these paritions, we also construct a pyramid in the angular dimension, of partitions 1, 2, 4, 8, 16, and 32 segments (see Figure 2). Due to the property of image symmetry in Fourier space, only the top half of the polar Fourier circle is sampled for the feature vector. In addition, we have added a prefiltering step to the original image that multiplies a circular mask to improve the rotational invariance of the discrete Fourier transform. In total, the dimensionality of our new feature vector is 3900 for the global granularity, and 19,500 for the layout granularity. For efficiency purposes, on the MED task, we reduced the complexity of the Fourier Polar Pyramid to radial partitions of 1, 2, 4, and 8. Angular segments were reduced to 1, 2, 4, 8, and 16. This resulted in a feature of 868 dimensions.

1.2.3 Low-Level Descriptors for Direct Event Modeling

Local Descriptors

Bag-of-features (BoF) representation based on various local descriptors is well-known to be especially effective for many tasks in computer vision and multimedia. A standard recipe of generating BoF features is 1) sample sufficient local descriptors and cluster over them to get visual codebook with pre-defined size, 2) quantize the local descriptors in a new image or video

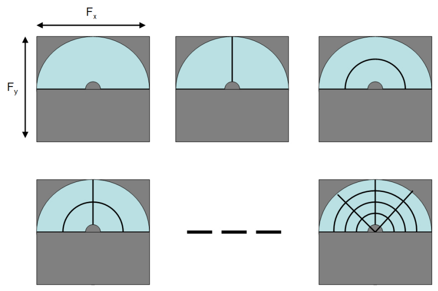


Figure 2: Depiction of the overlapping polar pyramid partitions sampled in Fourier-Mellin space (light blue boxes) used to compute the Fourier Polar Pyramid.

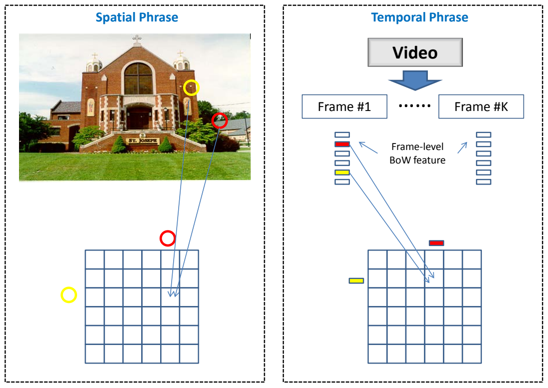


Figure 3: Intuitive understanding of our idea of matrix-based video pooling or high-order feature. Left and right sub-figures correspond to spatial visual phrase and our proposed temporal phrase.

frame. We adopt the following local descriptors due to their notable success in other relevant tasks. Most of them are variants of SIFT features. The major difference lies in the way to detect interest points from images or video frames, around which local descriptors are generated.

- **Sparse-SIFT**: It generates "sparse" local interest points by filtering candidates according to specific criteria. We use two variants: one is based on Difference-of-Gaussian (DoG) detector, and the other is using Harris-affine and Hessian-affine interest point detector[1]. Visual codebooks of size 1,000 are built from K-means clustering. Moreover, the spatial distribution of interest points is known to be useful cue for various tasks. Therefore, we build a two-layer spatial pyramid. The first layer corresponds to the original image, and the second layer splits the image into 2×2 grids. From each grid, all local descriptors drawn from it are aggregated to form a BoW feature vector. It is easily verified that the final dimension of the SIFT BoW feature is 5,000.

- **Dense-SIFT**: Sparse-SIFT is most effective for images with strong scene structure (*e.g.*, urban, indoor etc.). To complement Sparse-SIFT, we also extract a "dense" variant. Each frame is resized to be 240×320 pixels. To capture multi-scale information, a three-layer image pyramid is built, with a scaling factor of 0.5 between adjacent layers. On each layer, 128-d local descriptors are extracted at each location with a stride of 8 pixels.

- **Color-SIFT**: Both of aforementioned SIFT variants are based on greyscale information. However, the color information is supposed to be also useful. We adopt a publicly-available software package[2] to calculate Color-SIFT. The original video frame is transformed into specific color space with three channels. For each channel, we build a 500-word visual codebook. By applying the trick of spatial pyramid, from each channel, a 2,500-d BoW feature is obtained. By concatenating the features extracted from all channels, finally we get a 7,500-d feature for each video frame.

We also extract Classemes feature [14] from each frame. It adopts a multi-kernel method to construct a classifier for each per-defined visual category (the total number of categories is more than 2,600). The major component among the multiple kernels is that built from a 5,000 SIFT BoF feature.

High-Order Feature and Its Compact Representation

We propose a matrix-based approach for the aggregation of frame-level features. For most of the visual appearance oriented features (e.g., SIFT or GIST), a standard processing pipeline is comprised of two steps: feature extraction on frames sampled from the video at specific rate (e.g., 2 frames per second) or content-aware key frames from video shots, and the aggregation operation which summarizes the information from each frame into a single feature vector. The second step, known as video pooling in the computer vision literature, proves to be important for the final performance of multimedia event detection.

Our work is motivated by recent research endeavor on spatial visual phrase. In the problem of image matching, the local features of the query image are matched to those of the images in the database. Due to the adverse effect of varying illumination,

[1]http://www.robots.ox.ac.uk/∼vgg/research/affine/detectors.html

[2]http://staff.science.uva.nl/∼mark/

mutual occlusion between objects, view point variance, or different camera parameter settings, matching between single local features is found to be sensitive and prone to error. Therefore, spatial visual phrase, which refers to a bundle of two (or more) spatially proximal local features, is introduced to improve the matching robustness. Figure 3 shows an example of spatial visual phrase construction.

Similar idea of spatial phrase can be adopted to help build more discriminative video features. Conventionally, each video is represented by a single feature vector, which is computed from many frame-level feature using pooling methods (e.g., average pooling or max pooling). The major down side of those pooling methods is the ignorance of high-order occurrence information among different visual code word; therefore they fail to capture any kind of temporal dynamics of the video.

For the computation of video features, we utilize a matrix data structure to capture the temporal co-occurrence information. A pair of code words corresponds to a unique bin in the matrix structure. A vote count for each bin is calculated from the frequencies of the code words. Moreover, although the example in Figure 3 uses two code words from the same frame, it is beneficial to extend it to several adjacent frames. Although the matrix-based video pooling method captures more discriminative information, it suffers from the high dimensionality of the resultant feature, which requires tremendous storage and computing time. For example, suppose we use a code book of d distinct visual words, the matrix for temporal phrase will be of the dimension $d \times d$. Since a typical value for d is $O(10^3)$, it implies a feature length on the million scale. To obtain tolerable computing and storage overheads, we further propose to build compact "sketch" for the matrix structure. Table 1 shows the mAP scores of our proposed high-order features on our held-out random 20% threshold data partition (not used during event model training). It is observed that the performances of high-order features are comparable to that of original feature, yet the storage overhead is much smaller. More algorithmic details are postponed to our internal technique report.

Feature Name	Dimension	mAP
colorsift	7500	.2200
densesift	5000	.1893
classemes	2659	.1945
colorsift_highorder	1000 (bytes)	.1674
densesift_highorder	1000 (bytes)	.1593
classemes_highorder	1000 (bytes)	.2055
colorsift_classemes_highorder	1000 (bytes)	.2191
densesift_classemes_highorder	1000 (bytes)	.2008

Table 1: Averaged detection performance over all events for original low-level feature and our proposed matrix-based high-order feature. The performances are reported in terms of mean-average-precision (MAP) (0-1). Note that all high-order features are represented by 1000 bytes. The length is much shorter compared with that of original features. For example, dense SIFT feature requires 20000 bytes (assume we use 4 bytes to store the value of each dimension).

1.2.4 Audio Features

We calculated two types of low-level audio features over non-overlapping 2 second windows for each video. The first features are based on conventional MFCCs, and constitute the mean and full covariance matrix of the MFCCs. We use 20 MFCC dimensions in our basic feature (for a finer description of the spectrum than is typically used in speech recognition); we also calculate the deltas and double-deltas. MFCCs are calculated over 32 ms windows every 16 ms, and the delta and double-delta features are calculated over blocks of 9 adjacent feature frames (i.e., around 300 ms). The full representation of each window then consists of 60 mean values for each dimension of the direct, delta, and double-delta features, plus the 1830 $(61 \times 60/2)$ unique elements of the covariance matrix calculated over the 125 frames contained within each 2 sec block. In practice, we used only the 60 means plus the first 3 leading diagonals of the covariance matrix for $60+60+59+58 = 237$ feature dimensions. Each dimension was mean and variance normalized across the training set before creating a Euclidean distance matrix between 2 sec clips to be used as the basis of SVM training.

The second feature type was based on the Auditory Image Model of [9], which captures the fine temporal structure of the audio signal across a set of frequency bands, chops this "auditory image" into a number of different subregions, uses vector quantization to capture the information in each subregion, then performs classification on the concatenated VQ codeword histograms. We replaced the detailed (and rather slow) auditory front-end with a simplified approximation of a linear band-pass filterbank followed by running autocorrelation, and performed VQ on the resulting a PCA reduction of the resulting "correlogram" image in four separate frequency regions (each of 6 bands). With 1000 entries in each codebook, each 2 sec window was represented by a normalized 4000 dimensional histogram. In our experiments, this simplified model performed essentially the same as the full Lyon model, and a little worse than the MFCC features. However, combining MFCC-based and auditory-model-based features gave a substantial improvement of around 15% relative, indicating their complementary information. We used Chi-squared distance to turn this into a distance matrix for use with the SVM.

Both kinds of features can be automatically and exactly aggregated to larger time spans, for instance to calculate the overall features for an entire video. This is implemented within our feature file retrieval routine.

1.3 Semantic Modeling

Traditional methods of modeling video events have been to use machine learning algorithms, such as support vector machines (SVMs), to train models directly from low-level features of video examples that depict the relevant event class as well as unrelated background video examples. However, this approach leaves some key challenges unaddressed. First, the user must formulate a query as a collection of examples of what they are looking for, which may require considerable effort to collect and supply such examples. Second, the patterns learned and used to discriminate between events may significantly deviate from those that a human uses to define those same events in the first place. By providing the system with specific examples, one does not guarantee that the machine learning algorithm will actually learn the concept that the user is searching for, but rather something that may be far too specific, generic, or entirely shifted away from the focus of that concept.

To understand these problems in greater detail, consider the following scenario: perhaps a user would like to retrieve videos of a person changing a vehicle tire. In this case, the user might supply 10 example videos: a few of a person changing a passenger vehicle tire, a few of a truck, and maybe one of a bicyle. With such a limited number example videos, the query is almost guaranteed to represent not only the concept to be searched for, but an additional unwanted bias, i.e., all the supplied examples might demonstrate a person changing a vehicle tire outdoors. Does such a query data distribution imply that the user does not want to search for tires being changed indoors, perhaps in a garage? Should the system return a motorcycle tire being changed, or not? The answer to these questions are ambiguous without additional data supplied by the user. When the user does not supply enough data, the query is ill-posed, and the "correct" retrieval results depends on information that has been essentially "hidden" from the system. Typical methods to deal with "hidden" information make use of prior assumptions in the modeling process that regularize the results, essentially trying to introduce characteristics of the "hidden" information into the model itself, rather than the data. But, in a sense, the algorithm cannot "read the user's mind," especially as these assumptions change from query to query, and thus the system may return results that are indeed incorrect.

We believe event detection may be made more robust, simpler, and more space efficient, if the videos were described by their content in terms of higher level semantics. Some early work done by our group displayed significant performance improvements for event detection on TRECVID MED 2010 data using SVM based event models trained on top of high-level semantics, as opposed to directly from the visual low-level features from which the semantics are defined [10]. More recent experiments agree with this finding (Fig. 20).Such results suggest that semantics may be better suited to represent video content. In addition, a semantic representation of the video may permit the ability to perform semantic based text queries against the data. For example, instead of providing tens or hundreds of examples of a person changing a vehicle tire, the event can be described textually with a query such as "Person handling wheel, presence of a road vehicle." In this scenario, the query is much more specific. The user has intentionally not specified "indoor vs. outdoor," so it may be safely assumed to return both cases. In addition, the user has specified the general term "road vehicle," so the system knows this can be any variety of car, truck, or bicycle.

The bridge between low level features and high level events is referred to as the "Semantic Gap". We propose a technique that fills this gap with an additional semantic layer, connecting low level features to video events through a hierarchy of the visual and audio semantic content of the video. In total, we used 958 visual semantic concepts and 100 audio semantic concepts.

1.3.1 Semantic Taxonomy

For the MED11 event detection task, our team utilized a taxonomy of visual concepts/categories based on the IBM Multimedia Analysis and Retrieval System (IMARS) taxonomy [4]. The IMARS taxonomy is a set of federated multiple facets of concept trees using four conceptual constructs: entity (node), facet (node), is-a (link) and facet-of (link). The top level facets and the number of example images in each is shown in Figure 4. Adopting the facet node type and "facet-of" link type allows greater flexibility in modeling mutually non-exclusive concepts, which represent different view perspective of a same entity (e.g. people - number of people, age of people). Sibling concepts (nodes) with an entity parent node in the taxonomy tree are mutually exclusive. By inferencing the structure and semantic relationships, the taxonomy system can perform efficient labeling of training images by associating images with the each entity node in the hierarchy, and allocates negative training examples accordingly with the recognition of exclusiveness of entity nodes and non-exclusiveness of facet nodes.

One of the limitations with the 2011 version of IMARS system was the difficulty to visualize and comprehend the relations between some concepts due to the nested levels of facets. The other issue was the imbalanced distribution of training samples for concepts, and our experiments have shown the size of training samples has very strong correlation to the performance of concept detectors or classifiers. Figure 5 plots the correlation between training sample size and validation score (AP),

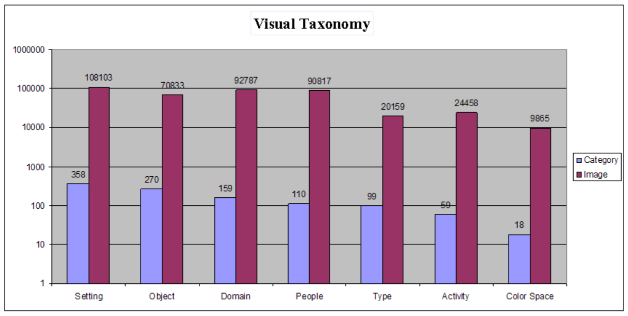


Figure 4: Visual taxonomy: distribution of categries and images across top facets

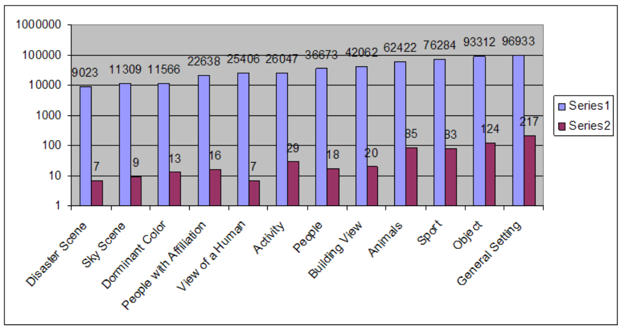


Figure 6: Overview of IMARS Taxonomy

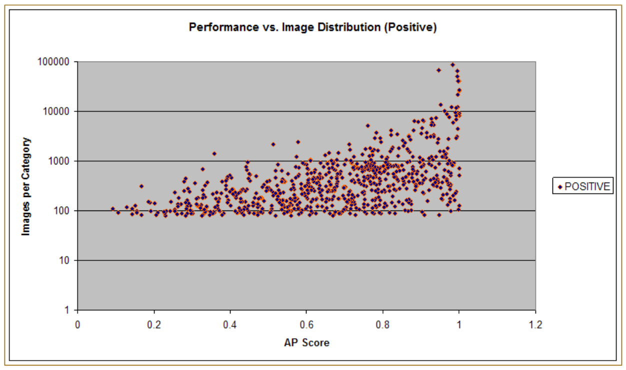


Figure 5: Correlation of sample size and performance

which clearly shows negative performance impact of small sample size. To overcome these limitations, we have implemented: (1) structural transformation of IMARS taxonomy from nested faceted structure to federated faceted structure and (2) expansion of training samples for all the conepts. For the MED12 event detection task, our team utilized this improved taxonomy of 615 visual concepts/categories and over 500K positive training examples.

Figure 6 provides an top-level overview of the current version of IMARS taxonomy includingn 12 facets (setting, sky scene, buidling view, people, object, people of affiliation, view of a human, animals, activity, sport, disaster scene, and dorminant color) with the distributions of categories and image examples among them.

1.3.2 Semantic Indexing (SIN) Task

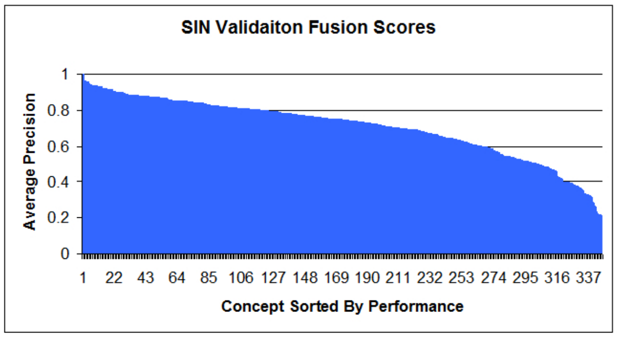
Our team made a submission to the full semantic indexing task. This task consisted of 346 semantic concepts for which positive and negative examples are supplied from TRECVID video dataset frames. We organized the data into training folders which could be fed through our IMARS system to model the semantics (see Section 1.3.3 and Fig. 8). The resultant models were used in conjunction with our own internal semantic taxonomy to extract semantic concepts from the MED video frames.

Given that our team was working under severe time constraints due to an incorrect website link to ground truth data, we sampled the images for each concept by capping the number of positive examples and the number of negative examples each to 2000 per concept, leading to a maximum of 4000 images to represent each concept. Our results this year were negatively affected this, as well as a normalization bug in our scoring system that we discovered after submission (see Section 1.5.3). This bug added noise to the concept scores on the test dataset, but did not affect performance on our own internal validation set used for Unit Model fusion.

The average precision scores on our internal validation dataset used for Unit Model fusion, as well as the SIN test dataset, is shown in Fig. 7. While not achieving the top performance, the utility of our system was demonstrated in terms of the amount of time it took for our modeling pipeline to finish the full SIN task: starting from scratch, processing time was on the order of a couple of days. We are confident that given the opportunity to utilize more data, fix our scoring bug, and implement additional state-of-the-art low-level features, that our system performance will improve significantly.

1.3.3 Visual Semantic Modeling

Visual semantic modeling is carried out by the IBM Multimedia Analytics and Retrieval System (IMARS). IMARS is a machine learning system designed for the extraction of semantic content from images and videos. The system has been in development for a period of over ten years, and is unique in its ability to eval-



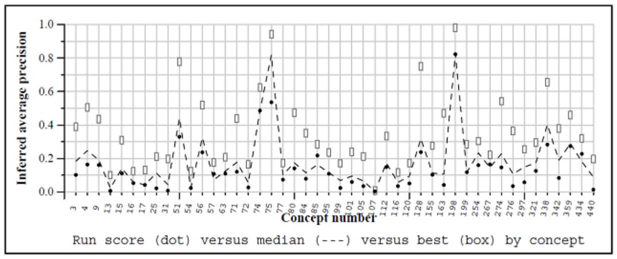


Figure 7: Top: SIN model performance on our internal validation dataset used for Unit Model fusion. Bottom: results on the SIN test dataset. Inferred MAP was 0.142. The scores of our models were affected by a bug in our scoring pipeline, which has since been fixed. We have begun the process of rebuilding and re-evaluating our models, which we expect to improve.

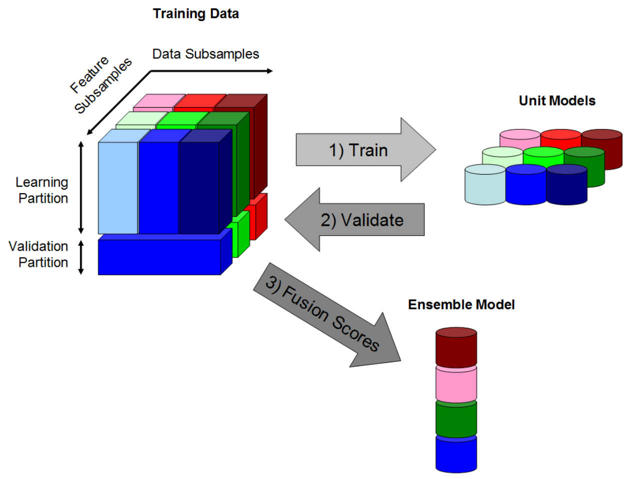


Figure 8: IBM Multimedia Analytics and Retrieval System (IMARS) learning approach. Training data is partitioned into Learning and Validation sets, which are further divided by features and data samples, referred to as "bags." Unit Models are trained for each bag, and an ensemble fusion approach using forward model selection determines the best combination of unit models to discriminate for the given concept.

uate a multitude of modeling strategies to determine the best approach for each semantic concept. In addition, the structure of the system gives it the ability to arbitrarily scale to large learning problems.

Instead of training very large models by concatenating all features in early fusion, and training models from all available data, our system trains smaller Unit Models [18]. For each concept, Unit Models are trained on a single feature, a single image granularity (such as whole image, which results in a descriptor matching the dimensionality of the feature, or a 5x5 grid resulting in a descriptor 25 times the dimensionality of the original feature), and a random subsample of data, referred to as a bag (Fig. 8). For each Unit Model, the system tries a variety of modeling strategies and kernel parameters (29 or more), selecting the most effective approach via n-fold cross-validation. The most discriminative Unit Models for each concept are selected to be fused into an ensemble classifier based on their performance on a held-out validation set.

Building models in this manner yields several desirable properties:

1. The first is that data imbalance is markedly reduced. When a Unit Model is sub-sampled, a maximum data imbalance threshold is enforced. The whole of the majority class is covered by the generation of many Unit Models, each with a different sampling of examples.

2. The second is that the learning problem is much more efficient when training many smaller models, instead of one large model, since the computational complexity of training a model is polynomial in nature

$$O\left(k \cdot \left(\frac{n}{k}\right)^c\right) << O\left(n^c\right) \qquad (1)$$

, especially for large k and large c.

3. The third is that since each Unit Model is an independent training task, we can easily parallelize to arbitrary scale.

4. The fourth is that since each Unit Model is trained over an individual feature, the process of forward model selection is also implicitly performing feature selection, as it determines the optimal combination of models to combine for each concept.

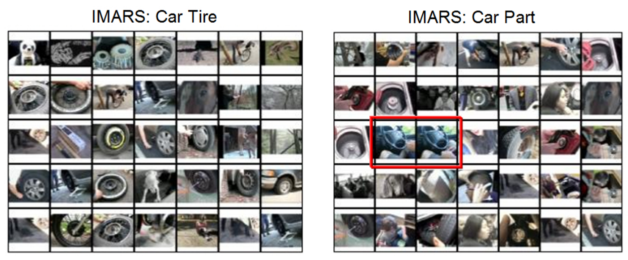


Figure 9: Example top scoring semantic retrieval results on a random sampling of 80,000 frames from the MED training data. Note how the semantic classifiers can extract some subtle differences between concepts, such as that a steering wheel shown in the red box is a "Car Part" yet not a "Car Tire" even though it has a round shape.

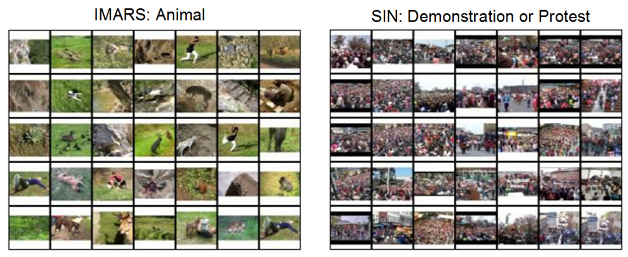


Figure 10: Example top scoring semantic retrieval results on a random sampling of 80,000 frames from the MED training data.

Some example retrieval results using the trained semantic models are shown in Figures 9 and 10. Note how our classifiers tend to be able to differentiate some subtle characteristics, such as a steering wheel being a car part, and not a care tire, even though both objects are round in shape.

1.3.4 Multimedia Event Recounting: Thematic roles

The current IBM-Columbia MER system is based on the close connection between our semantic classifier ontology and the theory of natural language thematic roles. Since both are grounded in a similar representation, the outputs of our visual and aural concept detectors map directly into the input slots of a template-based grammar. Each temporal segment of a video with high classifier responses therefore can generate a single sentence for the segment. A typical output, for example, is: "A Single_Person does Violent_Action to a Home_Appliance using a Knife at a Kitchen or at a Laundry_Room, with sounds of Vocals." In fact, segments are defined not by signal-level activity, but by temporal clusters of those semantics that have been prespecified as being relevant to a particular event.

Thus, our MER output is very lightweight, taking as input a matrix of classifier scores versus key frames; typically this is only about 1000 rows by about 100 columns (one frame every two seconds). During in-house user study experiments, this approach has led to near-perfect recognition of event type from the short streams of sentences generated for each video, and above-chance video identification.

1.3.5 Visual Attribute Modeling

Our semantic features are learned from the taxonomy, which is manually defined based on expert knowledge. Although semantically plausible, the taxonomy may not be consistent to the visual feature distributions. Consequently, some nodes are difficult to be modeled. In this section, we propose attributes as a feature-consistent way of modeling semantics. Specifically, we define attributes as multiple partitions of the concept pool (518 leaf nodes of the taxonomy). The partitions are optimized such that they are discriminative in terms of separating the known concepts, and consistent in terms of visual similarities based on the low-level feature. Weighted SVM classifiers trained by such partitions are then used as attribute feature extractors for event modeling.

To test the performance of the proposed approach, we have trained and extracted 2,500 dimensional attribute feature for the pre-specified task. Figure 11 shows the performance of the low-level feature and the proposed attribute feature, using linear SVMs for modeling, and frames downsampled to 1 frame every 8 seconds. Impressively, attributes have achieved relative performance gain over 60%, improving the MAP from 0.075 to 0.123. Note that identical low-level features are used for direct event modeling, training semantic features and training attributes.

1.3.6 Audio Semantic Modeling

We trained a total of 100 semantic audio models on our low-level audio features. The particular models were defined primarily by the availability of suitable training data. 55 of the models were the same as 2011, and were based on earlier clip-level labeling efforts based on YouTube data selected to consist of unedited consumer videos [7, 5], and on the MED2010 data segmented in to 10 sec chunks. The remaining 45 models were based on the 60 CD BBC Sound Effects Library, which consists of 2238 sound files covering a wide range of conditions, each lasting anywhere from a second to several minutes. Each track in the BBC collection comes with a one-line description including several keywords describing the sound; we generated a list of candidate semantic classes by choosing the 100 most

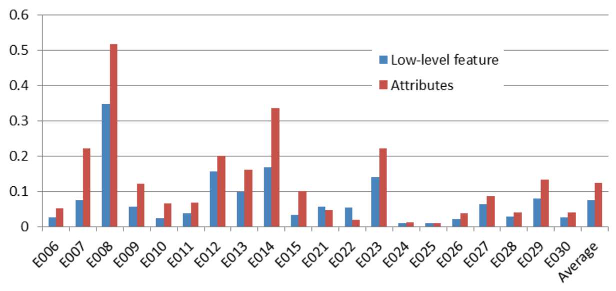


Figure 11: Average Precision results base on low-level feature and attributes for full exemplar. The results are evaluated on the internal threshold split containing 20% of the training data. Linear SVMs are used for event modeling.

Table 2: *The 45 semantic terms extracted from the brief annotations of the BBC Sound Effects library.*

stairs	ambience	speech
pavement	babies	cars
walking	sports	car
women	steam	boat
running	cat	transport
warfare	vocals	urban
wooden	siren	train
footsteps	animals	crowd
country	rhythms	electric
wood	animal	traffic
men	birds	street
horses	rural	children
emergency	electronic	household
construction	horse	crowds
futuristic	aircraft	voices

common words in these descriptions, and training preliminary classifiers for each one. We then sorted the words by the accuracy of these preliminary classifiers, and further filtered it down to 45 by choosing concepts that were semantically meaningful, reasonably successfully detected, and not redundant with other selected terms. The full list of 45 terms is in table 2.

Since our semantic classification features are to be provided at 2 sec resolution, we need to train on frames of this size. The training labels we have, however, are at the level of video or sound clip – typically much longer. For instance, a video tagged as containing "Music Performance" may include several windows of non-music sound prior to the performance beginning. Training a "Music Performance" classifier on these frames of generic background noise might hurt its discrimination.

To address this, we developed a Multiple-Instance Learning (MIL) procedure. MIL refers to the scenario in which data points belong to "bags", with labels that indicate whether a particular bag contains *any* items of that class. Thus a bag with a negative label will consist only of negative examples, but a bag with a positive label will in general contain a mixture of both positive and negative examples (like the frames in our Music Performance video). Our procedure first trains classifiers assuming all frames in the positive bags are true positives, and attempts to discriminate them from the frames in the negative bags. Every item in each positive bag is then submitted to a classifier (with cross-validation, so a classifier is never applied to frames used in its training), and any frames that fall below some threshold in classifier score are relabeled as negative, with the constraint that at least one frame in the bag (and any other frames whose classifier scores are very close to this "best" frame) must retain positive labels. Ideally, this will remove negative examples from the positive pool; classifiers are then retrained, and the process repeats until no further increases are observed on held-out development data (where a simple combination rule is used to produce a clip-level label from individual frames).

Our approach to choosing the threshold was to create histograms of the classifier scores from the negative and positive frames, calculate the cumulative distribution functions in opposite senses (i.e., $P(\text{score} < \theta)$ for the negative frames, and $P(\text{score} > \theta)$ for the positive frames), then choose the threshold θ where they intersect. By scaling one function prior to finding the intersect, the threshold can be made to remove frames from the positive bags more or less aggressively. Table 3 gives an example of the changes in test set mean Average Precision for iterative relabeling of the five frames in each 10 sec clip of our segmented MED2010 set; performance at the clip level improves for 4 epochs, then gets worse, so the labels for the 4th epoch are used as the basis for the final 2 sec-resolution semantic classifier. Note that we do not have any ground-truth labels at 2 sec resolution, so we cannot directly measure the frame-level classifier performance.

Audio classifiers for the 100 semantic classes were trained on both types of raw audio features, MFCC statistics and Auditory Model histograms (Section 1.2.4). Because a single video may contain hundreds of 2 sec windows, and because SVM distance matrix calculation is $O(n^2)$ in the number of frames, this training was far more computationally expensive than training whole-video classifiers. Further, because the histogram features relied on a chi-squared distance measure, which is around 10 times slower to compute than the euclidean distance used for the statistics, it was not possible to complete labeling of the MED12 development data in the available time; at 2 sec level, our 100 audio semantic features are based only on MFCC statistic features.

Table 3: *Mean Average Precision for labels aggregated to the clip level from the iterative relabeling of 2 sec audio clips, for the 10 sec segmented MED2010 corpus.*

Epoch	mAP
1	0.523
2	0.537
3	0.559
4	0.562
5	0.559

At whole-video level, however, we were able to train separate sets of semantic classifiers on both low-level audio features. The outputs of these classifiers – SVM distance-to-boundary scores, which have been found to be largely comparable between different classifiers – were then combined by simple summing to create a set of fused audio semantic features.

To evaluate these semantic features, and to compare them to the raw features, we used a task based on the first release of the MED2011 DEV data and the example videos for events 001 to 015. This gives a combined pool of 6354 videos, which were broken into 5 cuts, with classifiers trained on 4/5ths of the data and tested on the remaining 1/5th. For this test, all features were at the whole-video level. 15 independent per-event classifiers were trained using the different feature sets. The results are shown in Figure 12. We see that the raw MFCC and Auditory Model (sbpca) features have different strengths, with the Auditory Model features doing particularly well for E004 Wedding Ceremony, E008 Flash Mob Gathering, and E012 Parade. The semantic features broadly reflect the raw feature performance, but do better in some cases, such as E001 Attempting a Board Trick and E009 Getting a Vehicle Unstuck (at least for MFCCs). Finally, the fused semantic features (sema100sum) are generally successful in capturing or improving on the best individual feature in each category, delivering a 15% relative improvement in mean Average Precision, from 0.19 for the raw MFCC features to 0.22 for the fused semantic features.

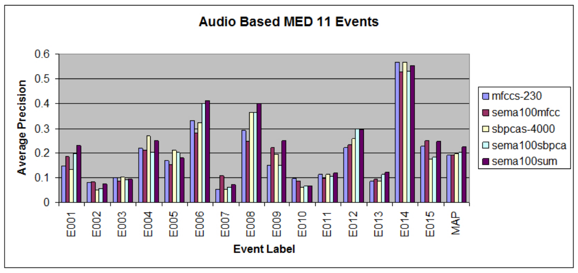


Figure 12: Audio-based Average Precision results for MED2011 event classifiers (6354 video set), comparing classifiers based on raw features (MFCCs and Auditory Model (sbpca)), and on 100-class semantic audio features based on each set of raw features (sema100mfcc, sema100sbpca), and fused by summing (sema100sum).

1.4 Event Modeling

1.4.1 Temporal Modeling & Feature Aggregation

When training an event model, a single descriptor must be generated for each video to represent its entirety. This is done typically by aggregating low or high level features from individual frames using average or maximum values. One major drawback of this method is that temporal information is ignored. This year we wanted to employ a method that could model the temporal progressions of semantics in event kit videos. Our team has previously published studies that investigate the effects of various temporal matching strategies on event detection performance [17, 3], and found generally that such methods improve system performance by signficant amonts (25% boost to MAP on MED11 events as measured on MED11TEST). This year, we chose to apply temporal pyramid models on top of the 958-dimensional visual semantics for the Ad Hoc event kits. We quantify the changes in event modeling performance on our internal held-out validation set used for model fusion, consisting of a random 20% sampling of the training data.

The temporal pyramid structure is depicted in Fig. 14. The pyramid is broken into individual components, each defined by the number of temporal segments contained (or the level of the pyramid), the aggregation style used, and whether fixed or dynamic temporal segment boundaries were employed. Dynamic boundaries are a new approach, compared to our previous studies: temporal segment boundaries are determined as those frames where the derivative of the semantic features with respect to time is greatest. The n-1 top values of this derivative define the boundaries for the n-th pyramid level.

A sampling of data (a "bag") from each component then is used to train a Unit Model, which is fed into the IMARS training platform. We use 30 bags of size 1000 to ensure complete coverage of negative examples. For each Unit Model, the system iterates over several modeling strategies that include both static temporal order matching kernels, in addition to dynamic order matching kernels (Fig. 13).

The comparison between temporal pyramids and single aggregation event models is shown in Figure 15. As can be seen in the figure, temporal pyramids improve overall event recognition performance (in terms of MAP) from the baseline single aggre-

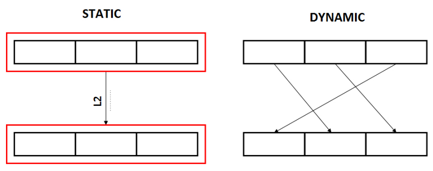


Figure 13: The IMARS system evaluates both static order and dynamic order temporal matching kernels during Unit Model training of Linear Temporal Pyramids, and selects the best method for each Unit Model.

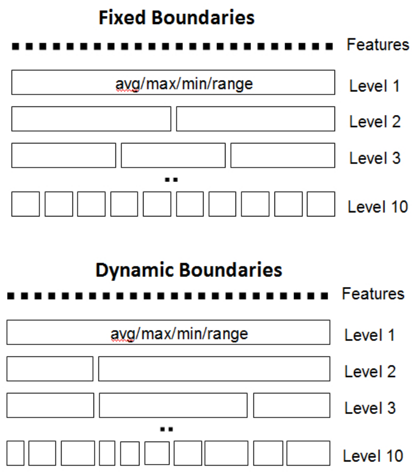


Figure 14: Visual depiction of constructing temporal pyramid aggregations over video frames.

gation by about 30% once data saturation has been reached. This performance boost is in agreement with results seen on MED11 data in our prior publication [3]. All events saw significant performance improvements (over 10%).

The performance comparison between two single unit models is shown in Figure 16: pyramid level 1 (a single average aggregation) is compared to Pyramid level 10 (also an average aggregation). It is important to note that at this stage, a unit model has been trained on only 80% of the available training data, and no model selection has been performed based on the held-out 20%. As can be seen in this figure, the increased temporal granularity improves performance substantially in two categories, as compared to single aggregation, suggesting that the ability to account for temporal order improves event recognition accuracy.

We believe that the events, while they may not contain any explicit temporal information in their textual event description

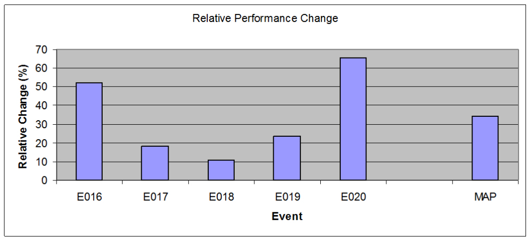


Figure 15: Performance of event models that use temporal pyramid matching, as compared to single level aggregations. Significant improvements in performance are seen across all categories.

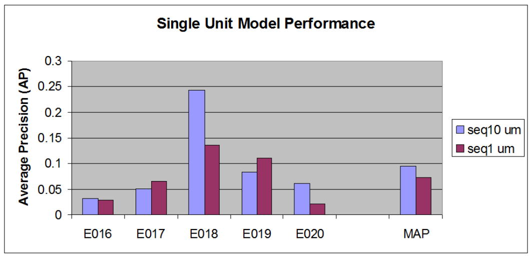


Figure 16: Performance of individual unit models across events. Those shown are single level average aggregations, and a 10 segment average aggregation using fixed partitions.

(i.e. step 1, step 2, step 3, etc), may have a tendancy toward temporal trends, as in the case of Fig. 17. In other words, the temporal order is not necessary for the event to be defined, but is common. Due to this, when temporal pyramids are applied, our forward model selection algorithm is able to choose which temporal granularity, which partitioning (fixed or dynamic temporal boundaries), which pooling method (avg, max, min, or range), and which kernel method (fixed or dynamic temporal matching) fits the data in the best possible manner. This rich spectrum of modeling strategies in combination with a robust selection algorithm has demonstrated systematic improvements in performance across categories.

1.4.2 Kernel Approximation Modeling

To explore different aspects of visual phenomenon, we need a good fusion strategy for multiple features. However, some complicated feature fusion strategy are not applicable for the large scale training dataset. Traditional nonlinear Supporting Vector Machine (SVM), builds classification model with many support vectors. The number of support vectors grows with the the size of training samples, which will make both the training and test-

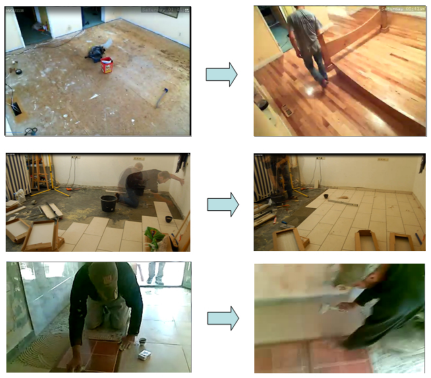


Figure 17: Example of temporal trends in event E019, ”Installing Flooring.” Many videos start off with a bare floor followed by a floor that has been further finished. However, some do not have a clear direction from the camera angle given.

ing stage slow. When the number of images grows, or when the feature dimension increases, traditional SVM solvers will be efficient enough for large scale problems. Among all the kernels in practice, the exponential Chi-square kernel often yields very good performance compared with the others. Moreover, a large amount of our features, including LBP histogram, edge histogram, color histogram, and SIFT histogram, are in the form of histogram features. Exponential Chi-square kernel is arguable regarded as the first choice for histogram form features. In our work, we focus on how to efficiently solve exponential Chi-square kernel only.

We consider the Chi-square kernel in the form of

$$K(x, y) = \sum_d \frac{2x_d y_d}{x_d + y_d}, \tag{2}$$

where $x = [x_1, x_2, \cdots, x_d, \cdots, y = [y_1, y_2, \cdots, y_d, \cdots$.

It is easy to see that Eq.(2) is defined as the additive sum of different dimensions. Such a kernel is referred to as an additive kernel. As suggested by [16], such a group of kernels can be approximated by mapping the feature into a high dimensional space. By the representer theorem [6], the solution of classification model can be written in the form of

$$f(x) = \sum_{i=1}^{N} K(x, x_i),$$

where i denotes the index of training samples. For any positive definite kernel, there exists a mapping $x \to \phi(x)$ so that the final classification model becomes

$$f(x) = w_\phi^T \phi(x) + b$$

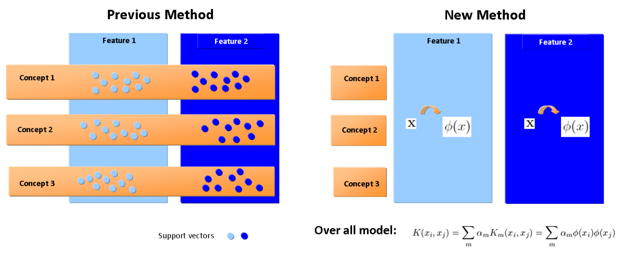


Figure 18: Comparing the new fusion method with traditional approach.

where w_ϕ denotes the weights of the linear model in the mapped space. In this work, we will use Nystrom's approximation to construct the mapping function explicitly.

To make the representation simply we let

$$k(x, y) = \frac{2x_d y_d}{x_d + y_d},$$

then we can see the kernel is

$$K(x, y) = \sum_d k(x_d, y_d). \tag{3}$$

Next we will discuss how to approximate $k(x, y)$, which is a function on 1D space.To approximate $k(x, y)$, we employ explicitly kernel mappling [16].

Figure 18 compares the fusion method with traditional method. It is easy to see the new method reduce the computational complexity. Suppose we have m features, $1 \leq m \leq M$, and each feature is of dimension d_m. The number of training samples is N, and size of testing set is T. For each d_m features, we map it to the space of dimension $7d_m$. Note that the evaluation of kernel SVM depends on the number of support vectors, which in practice is proportional to the number of training examples. Also our linear approximation requires the extra cost of feature mapping, which is $7dN$ for training and $7dT$ for testing. Table 4 compares the computational complexity of the two methods. It is easy to see our linear approximation is much more efficient in both training and testing stage. Our linear approximation is even plausible for the scenario with a lot of features.

1.4.3 Limited Exemplar Modeling

Pre-Specified

For the Pre-Specified limited exemplar, we tried to generate an 80% learning and 20% fusion split to assess performance of each component method; however, under these conditions no

Table 4: Comparing the computational cost of kernelized SVM and linear approximation.

	Training	Testing
Kernel SVM	$O(N^2 \sum_m d_m)$	$O(TN \sum_m d_m)$
Linear approx	$O(\alpha N \sum_m d_m) + O(N \sum_m d_m)$	$O(\alpha T \sum_m d_m) + O(T \sum_m d_m)$

component produced any appreciable performance. As a result, we opted to train the low-level component run on all 10 training examples, fuse different features using manually-determined weights (the fusion weights are nearly identical for different features), score the training dataset using these generated event models, and set the threshold as the minimum score produced by the 10 positive examples. Therefore, we have no performance plots to report for this case.

Ad Hoc

We conjectured that our difficulties with limited exemplar may have been in part the result of severe data imbalance and poor representation of the actual event distributions in each space. In order to address this challenge, we made the assumption that in the semantic space, the 10 examples might form, roughly speaking, a convex hull, or subspace, of the event distribution (Fig. 19). This assumption is similar to the assumption taken when training a linear SVM on top of the data. Previously, we have shown that linear SVMs perform better over high-level semantics than from low-level features on which the semantics are trained (Fig. 20); therefore, we believe this to be reasonable.

Under these assumptions, we create synthetic training data by taking random subspace projections (RSS), or random linear combinations, of the given examples, and then retrain our models on the synthetic data. While the new distribution might not exactly represent the event distribution, the intent is to help the learned decision boundary better generalize.

Our method loosely resembles the SMOTE technique [11], however, in our case, k=7 since we train on 8 examples and create synthetic data from random linear combinations of all datapoints. In addition, synethic points are allowed to exist anywhere in the subspace of points used to generate them, instead of on the line connecting a pair of neighboring points.

Our submission for limited exemplar query of the Ad Hoc task was thus to combine the temporal pyramids of visual semantics, using RSS to generate additional synthetic training data: the 8 examples used for model training were expanded to 208 examples. The 2 for model fusion were independently expanded to 202. The resultant models were then scored against the original 20% held-out data consisting of 2 positive examples per event.

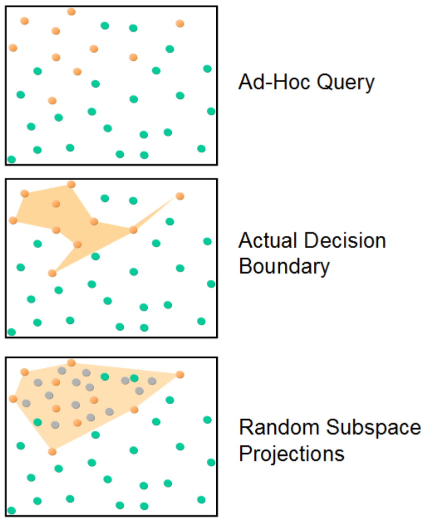


Figure 19: Illustration of Random Subspace Projection method for limited exemplar conditions. Top: Limited exemplar case, where only 10 examples are given. Middle: Actual decision boundary. Bottom: Under the assumption that in the realm of semantic space, the 10 examples form a convex hull, or subspace, of the event distribution, synthetic training examples can be produced by taking random normalized linear combinations of the 10 positive examples to fill in the subspace.

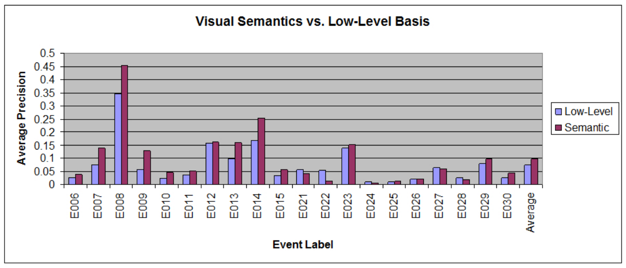


Figure 20: Event modeling results comparing high-level semantic features to the low-level features used to build the semantic models. Experiment was performed using linear SVM on a set of average value aggregated frames that were downsampled to 1 per 8 seconds.

1.4.4 Fusion Modeling

Fusion of Low-Level Features

From the diverse feature set that we define for the MED task, a large number of components can be accordingly generated, characterized by the feature type, kernel choice (e.g., RBF kernel or histogram intersection kernel) and parameter selection. Optimally fusing all these information is a critical operation for the final performance. Two most popular strategies for feature fusion is known as early fusion (i.e., fuse features before sending them to the event models) and late fusion (i.e., fuse the decision scores generated by different event models). In practice we have evaluated the performance under both settings. However, we find that early fusion will significantly uplift the feature dimension and increase the computational overhead. Therefore, we mainly capitalize on the idea of late fusion to boost the event detection accuracies.

Specifically, we adopt the ridge regression method for late fusion due to its simplicity and robustness (we have empirically compare ridge regression with other methods, e.g. linear regression with sparsity regularization, and observe comparable performances for all methods). Suppose we have K components, generating scores $X^1, \ldots, X^K \in R^n$, where n is the number of training data. Denote the ground truth event labels as vector $y \in \{0,1\}^n$, and the regression coefficients as $\alpha \in R^K$. The optimal solution of ridge regression is obtained by solving the following problem:

$$\min_{\alpha} \quad \|\mathbf{X}\alpha - y\|^2 + \lambda\|\alpha\|^2, \tag{4}$$

where $\mathbf{X} = [X^1, \ldots, X^K]$ is the concatenation of all components. It is well known that the above problem has close-form solution $\alpha^* = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T y$, where I is the identity matrix.

Fusion of Component Runs

Each major component of our system was scored against the progress set, and our internal held-out set of 20%, independently. This included Low-Level, Visual Semantics, Audio Semantics, Attributes, and Kernel Approximation. Fusion between runs was performed as a weighted sum of normalized scores.

Components are fused together one at a time. Fused components are considered a single component during the fusion of the next component. Weights were initialized to the average precision scores of each component on the threshold set, and updated iteratively as follows: if after fusion, the combined performance (in terms of average precision) is greater, the weight is kept. If the combined performance is lower than the best single component, the weights are re-adjusted in favor of the best single component. This process is iterated until at least the performance of the single best component is matched.

1.5 Scaling

We used computing clusters for two aspects of our system: learning model classifiers from a training set and applying those models to evaluation sets to produce semantic model vectors. We also call this process of applying the models "scoring". We made use of two platforms: Apache Hadoop, and IBM InfoSphere Streams, which are both part of IBM BigInsights[3].

1.5.1 Learning Visual Semantic Models

Scalability is achieved by implementing the IMARS learning system in the Map-Reduce framework (Open-Source Hadoop, or BigInsights as offered by IBM), where modeling is performed in two stages: a Map stage, where typically millions of Unit Models are constructed. Each of these models is an independent learning job, which makes them suitable as a Map task; however, several thousand such models typically correspond to a single concept being learned. This makes it possible to pass the resultant Unit Models to the second stage, or Reduce stage, where for each concept, the set of all Unit Models for that concept are subject to a process of forward model selection that determines the most effective combination of these Unit Models. Our system was implmeneted on dual-rack Apache Hadoop 0.20.2 system, with 224 CPU cores, 448 GB of total RAM, and 14 TB of HDFS storage.

Similar to the learning framework, our image scoring framework is also suitable for arbitrary scale, built around the Map-Reduce paradigm. During a scoring job, each Unit Model is

[3]http://www.ibm.com/software/data/infosphere/biginsights

applied to the data in the Map step, and their outputs for each concept are combined in a Reduce step.

1.5.2 Scoring Visual Semantic Models

The process of scoring the previously learned models on a test set of frames is computationally intensive. For example, scoring our 958 semantic models takes over 100 times longer than low-level visual feature extraction. The memory requirements are not exceptionally large, at about 90GB total for our set of semantic models. It is still somewhat uncommon for single machines to have that much memory, for instance our typical servers have 16 or 32GB. By spreading the models across a set of machines, we both parallelize the computations and sidestep the large memory requirement.

InfoSphere Streams is a scalable IBM software platform which fits video processing tasks well. It is designed for data in motion, such as frame by frame processing, and defines a set of stream operators that make it easy to filter and process streaming data in a cluster. We had temporary access to a cluster of 36 nodes containing a total of 336 virtual CPU cores. The nodes each had 4,8, or 16 cores, some employing hyper-threading. Each node has between 8 and 132GB of RAM and access to shared disk storage where the learned models were stored, which were about 90GB on disk. Streams allows us to assign sub-tasks to nodes based on their capability in order to maximize throughput.

Furthermore, some of the 958 models had many more support vectors than others, which was directly correlates with the relative amount of time required to evaluate them, as noted in Yan et al.[18]

In order to balance the processing time across nodes so that no CPU was starved for tasks, we used a bin-packing heuristic algorithm to allocate the 958 models into groups. A node would be assigned a number of model groups equal to the number of cores on the node. We wanted the model data to be in RAM at all times, so he target group size (in RAM space required) was selected so that the number of cores on a node times the size was less than 80% of the RAM on the node. So for a subtask on a set of 5 nodes, 16 cores each, with 32GB RAM per node, we had 80 total cores, and 80 model groups. The total of the models was about 90GB, so each model group was about 1125MB, requiring 18GB total RAM per node. With more nodes, we could scale by either making more model groups, or by replicating the task, splitting the input image frames into distinct sets and running them in parallel. If, for example, we had 1 core per model, we found we could score models at about 12 frames per second, either by model or frame based parallelism. We found that the Streams platform added minimal overhead to the processing.

1.5.3 Note About Distributed Software Bug

Both of our distributed systems, Hadoop and Streams, were used to extract semantic information from video frames of the progress set through their respective scoring pipelines. Each system was responsible for one half of the data. Additionally, our Hadoop distributed system was used extract semantics from training dataset frames, as well as to apply semantic event models generated from the training dataset to the entire progress set after aggregation to video level semantic features.

Midway between our Pre-Specified submissions and our Ad-Hoc submission, we discovered a bug in our Hadoop scoring pipeline. The Hadoop scoring bug had two significant consequences in our Pre-Specified submission: 1) The two halves of the progress set have different semantic statistics. This leads to semantic event model scores on one half of the progress to contain additional noise and a constant offset. 2) The semantic event models themselves produce additional noise on all data scored, since the same Hadoop pipeline was also used for this purpose. This would lead to a drop in event modeling performance. Event model scoring on the internal threshold data was not effected, since this process is actually embedded in our learning framework, which in part explains why we saw better performance internally than was produced on the progress set.

For our Ad Hoc submission, we attempted to mitigate the effects of the bug as follows:

The bug was fixed to prevent it from adding noise to the scores of any subsequent event models. However, this does not address the problem that our semantic models have different statistics on the two halves of the progress set. In order to reduce the effects of this problem, rank normalization was applied to the event model scores of each half of the progress set independently, and the data was recombined into one dataset. This mitigates the effects of any constant offsets in our event model scores, enforcing consistency in the distributions of scores between the two halves.

As can be seen from our results, we believe that the Hadoop scoring bug had a significant impact on the performance of our semantic models; however, our mitigation strategies helped reduce the affect of this bug in our later Ad Hoc submission. We expect that on a re-run of our system, we should see increases in performance for all of our semantic runs, both on internal and external data.

1.6 Score Calibration & Threshold Selection

Pre-Specified

To facilitate the selection of the optimal threshold for event recognition, the prediction scores from the event models need to be normalized. In our experiments, we normalize the prediction scores from the event models using a Sigmoid function, i.e.,

$$p(s) = \frac{1}{1 + e^{-as}} \tag{5}$$

where s is score from the event model, $p(s)$ is the normalized score between 0 and 1, a is a scaling factor learned from the collection statistics on the internal 20% threshold dataset.

Full Exemplar Threshold Selection
To pick up the optimal threshold for event prediction, we first obtain the ROC curve of the event recognition results on the internal threshold dataset. Then we pick up the threshold corresponding to the best operating point on the ROC curve.

Limited Exemplar Threshold Selection
For this case, we used all the training data to build event models. We then score the training data, and used the lowest score on the positive examples as the threshold.

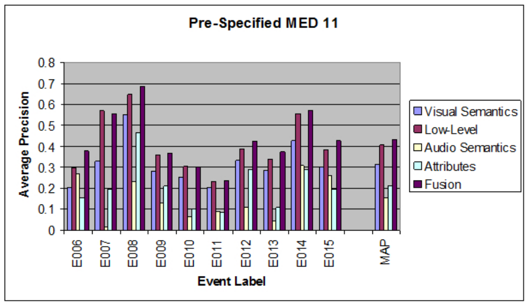
Ad-Hoc
After discovering the bug in our Hadoop scoring pipeline (Section 1.5.3), for our Ad-Hoc submission, we switched to applying rank normalization to each half of the progress set indepently (since semantics were extracted from one half in a buggy Hadoop system, and the other half in a bug-free Streams environment), and then recombining the scores into a single set. This approach was taken to compensate for event model score shifts between the two halves as a result of the bug. Rank normalization was applied to all component runs prior to fusion.

Full & Limited Exemplar Threshold Selection
To pick up the optimal threshold for event prediction, we first obtain the ROC curve of the event recognition results on the internal threshold dataset. Then we pick up the threshold corresponding to the best operating point on the ROC curve.

2 Experimental Results

Pre-Specified
Figure 21 shows the results of the various modeling strategies we applied to the pre-specified event task using the full training dataset. Single aggregations ignoring temporal information were applied to all components. Low-level features achieved top performance for any single approach.

For the Pre-Specified limited exemplar, we tried to generate an 80% learning and 20% fusion split to assess performance of each component method; however, under these conditions no component produced any appreciable performance. As a result, we opted to train the low-level component run on all 10 training examples, score the training dataset using these generated event



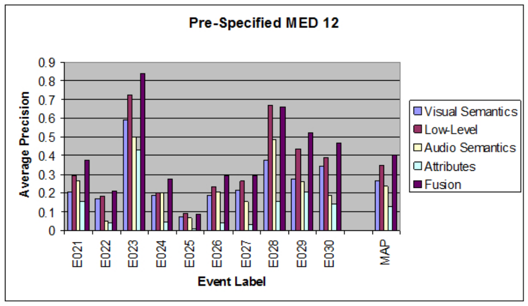


Figure 21: Results of various modeling strategies on the pre-specified events using full training data. Average precision scores were measured on the 20% threshold partition used for model fusion and threshold selection.

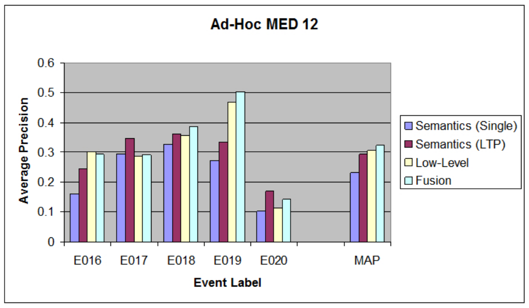


Figure 22: Ad Hoc event modeling performance on the full training data task. Average precision scores were measured on the 20% threshold partition used for model fusion and threshold selection. "Semantics (Single)" refers to video level aggregations of visual semantics ignoring temporal information. "Semantics (LTP)" refers to video level aggregations of visual semantics using the Linear Temporal Pyramid (LTP).

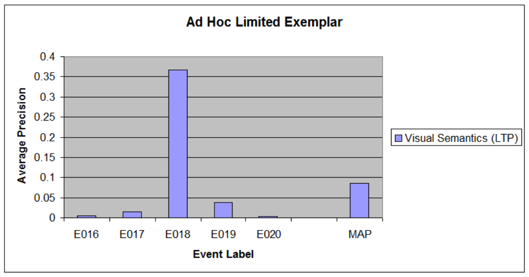


Figure 23: Ad Hoc event modeling performance on the limited exemplar training data task. Average precision scores were measured on the 20% threshold partition (2 positive examples per event) used for model fusion and threshold selection. The only component to produce appreciable results was Visual Semantics with Linear Temporal Pyramids and Random Subspace Projections for generation of synthetic training data.

models, and set the threshold as the minimum score produced by the 10 positive examples. Therefore, we have no performance plots to report for this case.

The following summarizes the components included in each run our team submitted:

- **Run 1:** Low-Level event models only.
- **Run 2:** Limited Exemplar. Low-Level event models only.
- **Run 3:** Fusion of Visual and Audio Semantics, as well as Visual Attributes.
- **Run 4:** Fusion of Visual and Audio Semantics, Visual Attributes, Low-Level, and Kernel Approximation methods.

Ad Hoc

Figure 22 summarizes the results of Ad Hoc full-exemplar experiments. Visual semantics using single aggregations and linear temporal pyramids are shown as the first two components, followed by low-level event models, and the fusion between semantics with temporal pyramids and low-level.

These experiments demonstrate an important observation: linear temporal pyramids of semantics built on top of mostly simple global features virutally matches the performance of event models build directly on top of the more sophisticated low-level features fused via ridge regression. This demonstrates that semantic and temporal modeling on top of low-level features holds great promise for significantly boosting the performance of direct event modeling on top of low-level features.

For the Ad-Hoc limited exemplar case, we again attempted to generate an 80% learning and 20% fusion split to assess performance of each component method. This time we used temporal pyramids on top of visual semantics, and implemented Random Subspace Projections to generate synthetic training data. With RSS, 8 positive examples used for model learning were expanded to 208. 2 positive examples used for Unit Model fusion were expanded to 202 examples. Once models were trained using these two sets, and the subsequently produced models were scored on the threshold dataset containing only the original 2 positive examples. The results of this approach are shown in Figure 23, and was the only one to produce appreciable results; therefore, it was selected for our submission.

The following summarizes the components included in each run our team submitted:

- **Run 5:** Fusion of Visual Semantics using Linear Temporal Pyramids and Low-Level event models.
- **Run 6:** Limited Exemplar. Visual Semantics using Linear Temporal Pyramids (levels 1-5) in conjunction with random subspace projections to synthesize 200 additional training data.

3 Conclusion

This year our team focused its efforts on pushing the boundaries of novelty and originality to examine a variety of approaches that, while alone may not achieve top performance, provide unique and complimentary strengths in comparison to other systems. We performed a broad range of experiments that have contributed useful information to each part of the event modeling pipeline. Our results suggest the following:

1. Methods that map low-level features to high-level semantics improve event modeling performance, in comparison to modeling directly on those same low-level features, in both visual and audio domains.
2. Event kits data contain temporal patterns that when properly modeled, hold potential to signficantly improve performance.
3. Mid-level visual attributes provide complimentary information to both high-level semantics and low-level visual features.
4. Kernel approximation methods greatly improving modeling and scoring efficiency, and show the potential to vastly expand the array of techniques that are used in parallel to model events on large datasets, which may lead to further improvements in performance.

5. There is no ”magic bullet” method to model events based on examples. The best approach is to use a broad variety of techniques, and dynamically determine which methods best model the unique characteristics of each event.

While our semantic runs were negatively effected by a bug in our scoring pipeline, we intend to correct the problem and rerun our internal experiments before the submission of the final version of this manuscript.

References

[1] Murray Campbell, Alexander Haubold, Ming Liu, Apostol Natsev, John R. Smith, Jelena Tesic, Lexing Xie, Rong Yan, and Jun Yang. Ibm research trecvid-2007 video retrieval system. *Proc. NIST TRECVID Workshop*, 2007.

[2] Liangliang Cao, Shih-Fu Chang, Noel Codella, Courtenay Cotton, Dan Ellis, Leiguang Gong, Matthew Hill, Gang Hua, John Kender, Michele Merler, Yadong Mu, Apostol Natsev, and John R. Smith. Ibm research and columbia university trecvid-2011 multimedia event detection system. *Proc. NIST TRECVID Workshop*, 2011.

[3] Noel C. F. Codella, Apostol Natsev, Gang Hua, Matthew Hill, Liangliang Cao, Leiguang Gong, and John R. Smith. Video event detection using temporal pyramids of visual semantics with kernel optimization and model subspace boosting. In *IEEE International Conference on Multimedia and Expo*, pages 747–752, 2012.

[4] A. Haubold and A. Natsev. Web-based information content and its application to concept-based video retrieval. In *ACM International Conference on Image and Video Retrieval (ACM CIVR)*, 2008.

[5] Yu-Gang Jiang, Guangnan Ye, Shih-Fu Chang, Daniel P. W. Ellis, and Alexander C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ACM International Conference on Multimedia Retrieval (ICMR), Trento, Italy*, Trento, Apr 2011.

[6] George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. In *Journal of Mathematical Analysis and Applications*, volume 33, 1971.

[7] Keansub Lee and D. P. W. Ellis. Audio-based semantic concept classification for consumer video. *IEEE Tr. Audio, Speech, Lang. Proc.*, 18(6):1406–1416, Aug 2010.

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[9] R.F. Lyon, M. Rehn, S. Bengio, T.C. Walters, and G. Chechik. Sound retrieval and ranking using sparse auditory representations. *Neural Computation*, 22(9), Sept. 2010.

[10] Lexing Xie Gang Hua Michele Merler, Bert Huang and Apostol Natsev. Semantic model vectors for complex video event recognition. In *IEEE Transactions on Multimedia, Special issue on Object and Event Classification in Large-Scale Video Collections*, 2012.

[11] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer. Smote: Synthetic minority oversampling technique. In *Journal of Artificial Intelligence Research*, volume 16, pages 321–357, 2002.

[12] Alan F. Smeaton, Paul Over, and Wessel Kraaij. High level feature detection from video in trecvid: a 5-year retrospective of achievements. *Multimedia Content Analysis*, pages 151–174, 2009.

[13] Abdenour Hadid Timo Ahonen and Matti Pietikainen. Face recognition with local binary patterns. In *European Conference on Computer Vision (ECCV)*, pages 469–481, 2004.

[14] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient object category recognition using classemes. In *European Conference on Computer Vision*, 2010.

[15] Jan C. van Gemert, Cees G. M. Snoek, Cor J. Veenman, Arnold W. M. Smeulders, and Jan-Mark Geusebroek. Comparing compact codebooks for visual categorization. *Journal of Computer Vision and Image Understanding*, 2010.

[16] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 3, 2012.

[17] Dong Xu and Shih-Fu Chang. Video event recognition using kernel methods with multilevel temporal alignment. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 30:1985 – 1997, 2008.

[18] R. Yan, M. Fleury, M. Merler, A. Natsev, and J. R. Smith. Large-scale multimedia semantic concept modeling using robust subspace bagging and mapreduce. In *ACM Multimedia Workshop on Large-Scale Multimedia Retrieval and Mining (LS-MMRM)*, Oct. 2009.