Feature Seeding for Action Recognition

Pyry Matikainen Rahul Sukthankar* Martial Hebert
pmatikai@cs.cmu.edu rahuls@cs.cmu.edu hebert@ri.cmu.edu
The Robotics Institute, Carnegie Mellon University

Abstract

Progress in action recognition has been in large part due to advances in the features that drive learning-based methods. However, the relative sparsity of training data and the risk of overfitting have made it difficult to directly search for good features. In this paper we suggest using synthetic data to search for robust features that can more easily take advantage of limited data, rather than using the synthetic data directly as a substitute for real data. We demonstrate that the features discovered by our selection method, which we call seeding, improve performance on an action classification task on real data, even though the synthetic data from which the features are seeded differs significantly from the real data, both in terms of appearance and the set of action classes.

1. Introduction

A human researcher who designs a feature has an almost insurmountable advantage over a learning algorithm: they can appeal to an intuition built over thousands of hours of direct experience with the world to decide which parts of the visual experience are important to consider and which are noise.

In contrast, an algorithm that attempts to select or learn features directly from a target dataset risks overfitting, especially if a large number of candidate features are considered.

Intuitively, this problem might be avoided if a large amount of related data were used to learn the features; one promising method to produce such data is synthetic generation using computer graphics techniques. While graphics methods are not quite yet at the point where plausible synthetic images can be economically generated, in the special case of *motion*, the widespread availability of mature motion capture technology has provided a wealth of resources from which synthetic videos of human motion can be produced.

We propose a technique that we refer to as *feature seeding*, in which synthetic data is used to select, or seed, features that are robust against a wide range of tasks and conditions. The actual model is learned entirely on real data; synthetic data has just guided the choice of underlying features. In this case, we only need enough similarity that the same types of features are useful on both synthetic and real data. In video analysis, for motion features we suspect we

Figure 1. Our method uses motion features from synthetic data (left) to seed features that are effective for real data (right), even though the two data sets share no common actions and are very different in terms of appearance.

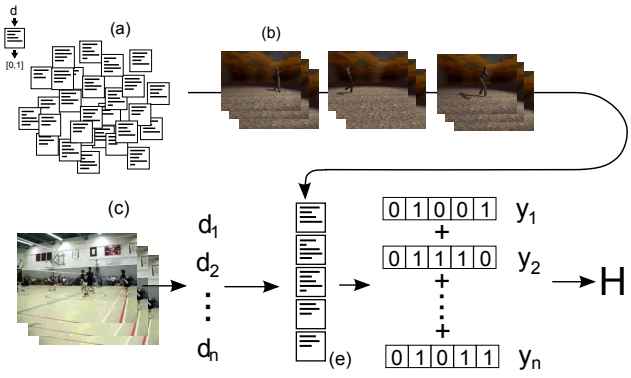

Figure 2. System overview: a pool of randomly generated features (a) is filtered, or seeded, on synthetic data (b) to produce a greatly reduced number of features (e) that are likely to be informative. We extract descriptors (*e.g.* trajectories) on real data (c), and these descriptors are fed through the seeded feature set to produce label vectors q_i, one per descriptor. These label vectors are then accumulated into a histogram H that represents the video clip.

*R. Sukthankar is now at Google Research

can meet that requirement (see Fig. 1).

What we demonstrate is that one can leverage observations of human actions obtained from one source to classify actions observed from another loosely related source, even if the two sets of actions differ. This transfer is possible because the two datasets are correlated — not necessarily in terms of specific actions but because both depict humans performing visually distinctive movements.

In more concrete terms, many popular bag of visual words (BoW) techniques rely on quantizing descriptors computed from video; generally either simple unsupervised techniques such as k-means clustering [11, 15, 20, 24] or hand-crafted quantization strategies [18] are used. Our suggested seeding can be seen as employing synthetic data to drive the selection of the *quantization method* itself.

The basic organization of our method can be seen in Fig. 2. First, a set of synthetic video clips is generated using motion capture data. These clips are generated in groups, where each group is an independent binary classification problem. Next, raw motion descriptors are extracted from the synthetic data pool in the form of trajectory snippets [18, 19] and histogram of optical flow (HOF) descriptors around space-time interest points (STIP) [15]. Note that we are not proposing a complete system for action recognition; we consider only motion features in a simplified recognition framework in order to isolate the effects of our feature seeding.

Each clip produces many descriptors– trajectory descriptors produce on the order of 300 descriptors per frame of video, while STIP-HOF produces closer to 100 descriptors per frame. These descriptors are sampled to produce a candidate pool of features, where each feature is a radial basis function (RBF) classifier,[1] whose support vectors are randomly drawn from the descriptors. Then the synthetic data is used to rank features based on their aggregate classification performance across many groups of synthetic data. We denote the highly ranked features selected in this way as the *seeded* features. The seeded features can then be applied to real data and used as input to conventional machine learning techniques. For evaluation, we consider the seeded features in a standard bag-of-words framework, using linear SVMs as classifiers.

2. Related work

Our proposed technique is related to both domain adaptation and feature selection, but targets a different level of information transfer than either. Domain adaptation techniques can be powerful across limited and well-characterized domains, such as in [14]. However, the gains are often modest, and as the aptly titled work "frustratingly simple domain adaptation" by Daumé [10] shows, even simple techniques can outperform sophisticated domain adaptation methods. Likewise, transfer learning methods such as transductive SVMs [8] can provide modest benefits, but are often computationally expensive and often restricted to datasets with shared classes.

In terms of feature selection, our method falls firmly into the filtering category of the taxonomy of Guyon and Elissee [13], in which features are selected without knowing the target classifier. The choice of a filtering method rather than a wrapper is motivated by the larger risk of overfitting in wrapper methods [13, 17, 29]. We use a feature ranking technique [13, 29] that is inspired by boosting-based methods [9, 28, 29]. However, since we do not assume that the specific task is known on the target data, we do not rank features by their performance on a single task, but instead on aggregate performance over a basket of independent randomly generated tasks.

Typical domain adaptation techniques assume that the specific task is the same between the source and target domains [2, 10, 14], and this assumption is the most common one in transfer learning as well [8, 9, 12]. That is to say, if the problem were action recognition, then these techniques would need the specific actions of interest to be matched across the domains. For example, Cao *et al*. perform cross-dataset action detection, using one dataset (KTH) [24] to improve performance on a related one (MSR Actions Dataset II) [4]. However, the particular actions that are detected are present in both datasets, and indeed MSR was explicitly constructed to share those actions with KTH. In contrast, our seeding technique requires no forward knowledge of what classes are present in the real data, and does not require any shared action classes at all between the synthetic and target tasks.

A related idea is that of learning from *pseudo-tasks*, as in Ahmed *et al*. [1], where the learning of mid-level features is regularized by penalizing features for poor performance on a set of artificially constructed pseudo-tasks. Our synthetic data can be seen as pseudo tasks, but with the important distinction that our synthetic tasks force features to be good at the specific problem of human action recognition rather than the more general problem of image processing.

We select from a pool of features that are computed from raw descriptors, which may take many forms; we consider trajectory fragments [18] and space-time interest points [15]. These features are slightly modified Gaussian radial basis function (RBF) classifiers. We choose to use features of this form because RBF can approximate arbitrary functions [25], and random classifiers and features have, by themselves, shown benefits over other representations [6, 23]. The choice of each feature as an independent classifier means that, after features have been seeded, only the selected features need to be computed on the test dataset. As our method is capable of drastically reducing

[1] We use the word "feature" in the same way as in boosting approaches [27]: a feature is an individual base classifier.

Figure 3. Example frame from synthetic data. The synthetic video is abstract in appearance, but the movement of the armature man is derived from motion capture data. The texture does not match real data in appearance and serves only to provide descriptor extractors (*e.g*. trajectories derived from optical flow) with stable inputs.

the feature count (*e.g*., from 10,000 to 50 with no loss of accuracy), this results in greatly reduced computational and storage requirements, and simplifies subsequent stages in layered architectures.

There has been limited work on using synthetic data for action recognition. A number of approaches use synthesized silhouettes from motion capture to match actions (*e.g*. [7]), but these are limited to a constrained domain (where silhouettes can be reliably extracted) and require that the action being searched be present in the data set. Recent work with depth cameras [26] demonstrates the power of this type of approach when the synthetic model of the task is very good. Another line of work by Qureshi and Terzopoulos [22] uses synthetic crowds to tune sensor networks for surveillance. In still image analysis, Pinto *et al*. use a screening approach to select entire models from synthetic data [21]. In contrast, our approach selects generally good features that are useful across models.

We demonstrate that even crude, unmatched synthetic data that makes no attempt to directly mimic the target datasets can be used to improve performance, and furthermore, that this increase can be achieved by straightforward selection mechanisms. Additionally, seeding a quantization scheme from synthetic data outperforms both unsupervised and heuristic quantization schemes.

3. Method

3.1. Descriptor extraction and handling

Our synthetic data can support a wide variety of motion descriptors, and in this paper we consider two different approaches: trajectory based descriptors, and histogram of optical flow (HOF) descriptors as per Laptev *et al*. [15].

For the trajectory descriptors, points are tracked using a KLT tracker [3], with the number of tracked points capped at 300. Each trajectory is segmented into overlapping windows of ten frames, where each window or snippet is expressed as a set of coordinates in the image

$$d_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\}, \quad (1)$$

where $T = 10$ is the number of frames in the overlapping window. This is converted into a relative representation, so that

$$d_i = \{(dx_1, dy_1), (dx_2, dy_2), \ldots, (dx_{T-1}, dy_{T-1})\}, \quad (2)$$

where $dx_t = x_{t+1} - x_t$, and $dy_t = y_{t+1} - y_t$.

This relative representation is the basic trajectory descriptor on which the k-means centers are computed. For input to RBF features we perform an additional normalization step to normalize the length of each link (dx, dy) to 1. This normalization to discard magnitude information is similar to that used in other techniques [18, 19].

For the histogram of optical flow (HOF) based motion descriptor, we use the Space-Time Interest Points of Laptev *et al*. [15] using a HOF descriptor around each point (STIP-HOF). This method finds sparse interest points in space and time, and computes HOF descriptors around each. Each descriptor found in this manner is a 90-dimensional vector.

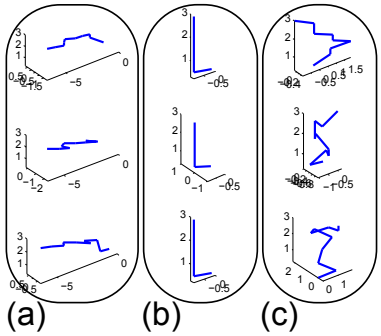

Figure 4. Examples of trajectory descriptors accepted by different classifier features. Some features represent simple concepts, such as leftward movement (a), or a quick jerk (b), while others do not correspond to anything intuitive (c). Given limited labeled data (c) could be indicative of overfitting. Feature seeding allows us to confidently determine that the chosen features generalize well.

3.2. Feature pool

3.2.1 Feature generation and evaluation

We consider a pool of candidate features, where intuitively each feature can be viewed as an RBF classifier. Formally, each feature evaluates a function of the form

$$f_k(d) = \text{clip}(\sum_i w_{k,i} \cdot e^{-\beta_{k,i}||d - v_{k,i}||^2}), \quad (3)$$

where $v_{k,i}$ is one "support vector" for the feature f_k, and $w_{k,i}$ and $\beta_{k,i}$ are the weight and beta for that support vector. The clip(.) function clips the value to the range $[0, 1]$,

so that values less than zero (definite rejections) are thresholded to zero, while values above one (definite accepts) are thresholded to one and all other values are unchanged. Our experiments use RBF classifiers as features due to their generality, but in practice our method can employ any type of classifier.

Given this functional form, we generate a pool (in our case, of size 10,000) of features by randomly selecting support vectors from the synthetic dataset's descriptors. The weight associated with each support vector is chosen from a normal distribution $N(0,1)$, and the β associated with each support vector from a uniform distribution over the range $[0,10]$. These parameters were chosen arbitrarily to generate a large range of variation in the classifiers. Example trajectory descriptors that might be accepted by these types of features can be seen in Fig. 4.

The feature can also be seen as computing an intermediate representation $q(d)$ corresponding to a descriptor d, so that

$$q(d) = (f_1(d), f_2(d), \ldots, f_K(d)) \,. \tag{4}$$

When the pool of features is evaluated, the "histogram" bin corresponding to a feature is evaluated according to

$$b_k(D) = \sum_{d \in D} f_k(d), \tag{5}$$

where D is a set of descriptors (*e.g.*, all the descriptors computed from a given video). The entire histogram is expressed as

$$h_D = (b_1, b_2, \ldots, b_K) = \sum_{d \in D} q(d), \tag{6}$$

which is to say that the feature f_k is treated as an indicator function for whether a descriptor belongs to label k, where a descriptor might have multiple labels, and where the labels a descriptor d_i takes on are given in the vector q_i.

3.3. Feature seeding/filtering

Given the pool of features, we select for, or seed, a good set of features from the pool by rating them on a set of synthetic data. In practice, the seeding is similar to a single iteration of boosting, with the important difference that the seeding attempts to find features that work well across many different problems, rather than a single one.

Let P_n and N_n correspond to the sets of descriptor sets (videos) in the positive and negative sample sets, respectively, of a synthetic group n. Then we can express the rating $a_{k,n}$ of a feature k on group n ($n = 1, \ldots, N$) as

$$a_{k,n} = \max_t \frac{\sum_{D \in N_n} I(b_k(D) \leq t) + \sum_{D \in P_n} I(b_k(D) > t)}{||N_n|| + ||P_n||}, \tag{7}$$

where $b_k(D)$ is the result of evaluating feature k on descriptor set (video) D, and $I(.)$ denotes the indicator function.

Note that this is just the accuracy of a decision stump on the $b_k(D)$ values. We have also considered mutual information based rating, but we find that it has slightly worse performance, probably because the stump-classifier rating we use here is a better match for the final SVM classification. However, our method does not depend on any single rating metric, and it is straightforward to swap this metric for another.

Now, we express the aggregate accuracy of a feature over all groups as

$$A_k = g(\{a_{k,n} | n = 1, \ldots, N\}), \tag{8}$$

where $g(.)$ is a function that operates on a set. In our case, we consider three possible aggregation functions g: $g_{\text{min}}(X) = \min(X)$, $g_{\text{max}}(X) = \max(X)$, and $g_{\text{avg}}(X) = \text{mean}(X)$. Intuitively, g_{min} takes the worst-case performance of a feature against a collection of problems, g_{max} takes the best-case performance, and g_{avg} takes the average case. Note that because the evaluation problems are randomly generated from a large motion capture database (see Section 3.4), it is unlikely that they will share any action classes in common with the target task. The goal is to select features that perform well against a variety of action recognition tasks (*i.e.*, that can discriminate between different human actions).

Then we simply rank the features according to their A_k values and select the top s ranked ones. In practice, we use seeding to select the top $s = 50$ features from a pool of 10,000.

Given a set of training and test videos on real data, we compute histograms h_D, where each histogram is computed according to (Eqn. 6) over the reduced set of s features. Then we simply train a linear SVM as the classifier.

3.4. Synthetic data generation

In order to perform our feature seeding, we must be able to generate relatively large amounts of synthetic data. Since it is difficult to produce synthetic data that is comparable to real-world data in terms of raw pixel-level appearance, we concentrate on the simpler task of generating synthetic data that matches real-world data in terms of *motion*. We make no attempt to mimic real-world appearance: the human model in our synthetic data is a abstract armature (Fig. 3). However, in terms of motion it is a reasonable analog, since its motion is derived from human motion capture data (Fig. 1).

3.4.1 Synthetic data organization

The synthetic data is organized into groups of clips. Each group consists of a number of *positive* samples all gener-

ated from a single motion capture sequence, and a number of *negative* samples randomly drawn from the entire motion capture dataset. In this way, each task is an independent binary classification problem where the goal is to decide which clips belong to the action vs. a background of all other actions. We reiterate that the actions used in the synthetic data do *not* correspond to the actions used in the final classification task on real data. Since the synthetic actions are randomly chosen out of motion capture sequences, they may not correspond to easily named actions at all. The two sets of tasks are unmatched so that the seeded features can be used in any future classification task. Each clip is 90 frames long, and each group has 100 clips, corresponding to 50 positive samples and 50 negative samples. In this paper we use 20 groups, for a total of 2000 clips.

A clip is produced by moving a simple articulated human model according to the motion capture sequence, with some added distortions. The synthetic data is rendered at a resolution of 320×240 and a nominal framerate of 30fps in order to match the MSR and UCF-YT datasets (see Sec. 4.1).

3.4.2 Motion generation

The motion of the human model in the synthetic videos is produced by taking motion capture sequences from the CMU motion capture database [5] and adding temporal distortions and time varying noise to the joint angles.

For each clip a motion capture file is chosen from the 2500 clips in the CMU motion capture database. If the clip is meant to be a positive example, then the motion capture file and approximate location within that file is given, and the starting frame is perturbed by approximately $\pm$1s. If the clip is meant to be a negative example, a motion capture file is randomly chosen from the entire database, and a starting position within that file is randomly chosen.

Next, temporal distortion is added by introducing a temporal scaling factor (*e.g.*, if the factor is 2.0, then the motion is sped up by a factor of two). Non-integral scaling factors are implemented by interpolating between frames of the motion capture file. Then, a random piece-wise linear function is used to dynamically adjust the temporal scaling factor of the rendered clip. In practice, we limit the random scaling factor to drift between a value of 0.1 and 2.0. Consequently, the timing of a rendered clip differs from that of the base motion capture file in a complicated and nonlinear fashion.

A similar approach is used to add time-varying distortion to the joint angles. A random piece-wise linear function is generated for every degree of freedom for every joint in the armature, and this function is simply added to the joint angles obtained from the motion capture sequence. The magnitude of this distortion is ± 0.3 radians.

We add several other distortions and randomizations to the synthetic data. The viewing angle is randomly chosen for each clip, as is the viewing distance. Additionally, the position of the actor/armature is randomized for each clip. The lighting is also randomized between clips, because the effects and positions of shadows can have a significant effect on the extraction of feature trajectories.

4. Results

4.1. Datasets

We evaluate our method on two standard datasets: the UCF YouTube "Actions in the Wild" dataset (UCF-YT) [16] and the Microsoft Research Action Dataset (MSR) [30]. The UCF-YT dataset is a straightforward forced-choice classification problem between 11 action classes (mostly various sports). The dataset contains 1600 videos of approximately 150 frames each, and we divide these videos into a training set of 1222 videos and a testing set of 378 videos. The UCF-YT dataset is further sub-divided into subsets of related videos (*e.g.*, all from the same sports match, or sharing the same background); in order to avoid training/testing contamination from closelyrelated videos, we employ a stratified training/testing split that places each subset either entirely in the training or the testing set.

The MSR Action Dataset consists of of sixteen relatively long (approximately 1000 frames per video) videos in crowded environments. The videos are taken from relatively stationary cameras (there is some camera shake). The dataset only has three actions — clap, wave, and box, with each action occurring from 15 to 25 times across all videos. The actions may overlap. For evaluation we consider MSR to be three separate binary classification problems, i.e., clap vs. all, wave vs. all, and box vs. all, rather than a three-way forced choice because the actions overlap in several parts. Each problem has an equal number of negative samples drawn by randomly selecting segments that do not feature the action in question, so for example, the wave vs. all problem is a binary classification between the 24 positive examples of the wave action and 24 negative examples randomly drawn from the videos. Due to the limited amount of data in this set, evaluation is by leave-one-out cross validation.

As described earlier, for feature seeding we use a synthetic dataset, which consists of 2000 short videos; the "actions" in this dataset do not necessarily correspond to any of the action classes in either the UCF-YT or MSR datasets.

4.2. Feature statistics

A natural question to consider is how informative these RBF features are; that is, how likely is our seeding method to find useful features. Because the features are evaluated by treating them as stump classifiers, the worst an individual feature could do is 0.5 accuracy; any lower, and the classi-

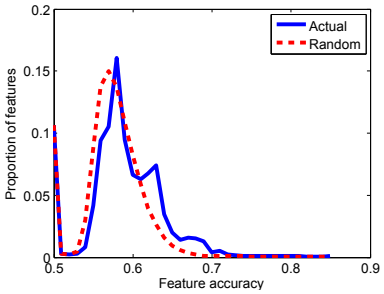

Figure 5. Accuracy distribution of RBF classifier features on synthetic data, compared with the expected number of false positives. Above accuracy 0.61, the majority of features are true positives. The difference between these two distributions is statistically significant to $p < 0.001$ according to the Kolmogorov-Smirnov test.

fier simply flips direction. Since there is noise in the data, a classifier that is uncorrelated with video content can still vary in value across videos, and this means that it is possible for it to obtain an accuracy better than 0.5 on the limited data simply by chance. If we were considering a single feature, then we could ignore this unlikely possibility, but with a pool of 10,000, statistically we can expect to see several such false positives.

It is easy to empirically estimate the false positive distribution by simply randomly permuting the labels of all of the test videos; in this way, a classifier cannot be legitimately correlated with the video labels, and the resulting distribution must be entirely due to false positives.

As can be seen in Fig. 5, the accuracy distribution of the real features is quite different from the false positive distribution. In particular, the real feature distribution is shifted to the right of the random distribution, indicating that there are more high-accuracy features than would be expected by chance, even in the worst-case scenario that the vast majority of the features are uninformative. Note that the false positive distribution takes on a log-normal type distribution, albeit with a spike at 0.5 corresponding to zero-variance features. The same test performed with the aggregation techniques produces similar results, indicating that the aggregation techniques also reveal informative features.

4.3. Comparison with other quantization methods

Since the goal of the proposed technique is to improve on the early quantization and accumulation steps of the bag-of-words model, a natural baseline against which to compare is the standard bag-of-words model consisting of k-means clustering followed by nearest neighbor quantization and histogram accumulation. Additionally, for the UCF-YT dataset we compare against a somewhat more sophisticated quantization technique, trajectons [18].

Our results on the UCF-YT dataset are shown in Table 1. Here the feature shows large gains over both k-means and random feature subsets, at 46.0% to 37.0% and 40.2% respectively. Additionally, our feature selection improves upon the trajectons quantization technique, which obtains an accuracy of 42.2%. The power of our technique is further emphasized by the fact that our technique uses only 50 features compared to the 216 of trajectons. Even with a pairwise spatial relationship coding scheme, their technique achieves 47.7%, which is only slightly better than the performance of our independent features without any spatial information.

Note that our performance with 50 seeded features matches that of running the entire candidate set of 10,000 features. Beyond the obvious computational and storage benefits of processing only 50 features instead of 10,000, methods that build on top of these quantized features will likely benefit from the reduced dimensionality (*e.g.*, if pairwise relationships are considered, it is better to consider 50×50 rather than 10000×10000). While the "kitchen sink" approach of feeding all 10,000 classifiers into an SVM worked in this case (likely due to the resilience of linear SVMs against overfitting), other classifiers may not be as robust.

The results of this comparison on the MSR dataset are shown in Table 3. Overall, the feature selection posts relatively large gains in the g_{max} and g_{avg} selection methods, while g_{min} remains largely the same as for k-means. For the individual classes, the selection method improves performance on the clap and box categories, while performance on wave is largely similar.

It is interesting that the selection techniques that perform

Table 1. Results on UCF YouTube dataset (motion features only).

Method	Total accuracy (%)
Seeded RBF [STIP] (g_{max})	34.4
k-means [STIP]	36.6
k-means [Traj] (MSR centers)	36.6
k-means [Traj] (synth centers)	37.0
Seeded RBF [STIP] (g_{min})	38.6
Seeded RBF [Traj] (g_{max})	38.9
Seeded RBF [Traj] (g_{avg})	39.4
Unseeded RBF [Traj]	40.2, $\sigma = 1.9$
Trajectons [18]	42.2
All 10,000 RBF [Traj]	46.0
Seeded RBF [Traj] (g_{min})	46.0

Table 2. Comparison of seeding source on UCF YouTube.

Method / Source	UCF-YT	Synthetic
Seeded RBF [Traj] (g_{max})	38.6	38.9
Seeded RBF [Traj] (g_{avg})	34.7	39.4
Seeded RBF [Traj] (g_{min})	41.0	46.0

Table 3. Seeding outperforms k-means, unseeded RBF, and boosting on MSR.

Method	Clap	Wave	Box	Total
Boosting (50/500)	65.0	60.0	57.0	60.7
Unseeded RBF	60.0	61.0	61.7	61.0
Seeded RBF (g_{min})	60.7	62.5	60.4	61.2
k-means (MSR centers)	53.6	62.5	68.7	61.6
k-means (synth centers)	53.6	62.5	68.7	61.6
Boosting (50/10000)	75.0	64.6	52.0	63.9
Seeded RBF (g_{avg})	71.4	62.5	66.7	66.9
Seeded RBF (g_{max})	75.0	58.3	70.8	68.0

well are exactly inverted between MSR and UCF-YT, with g_{max} and g_{avg} performing well on MSR, while g_{min} performs well on UCF-YT. In practice, g_{avg} works like a weaker g_{max}, so it is unsurprising that its performance is similar to that of g_{max} on both datasets. Between g_{min} and g_{max}, however, we suspect the difference is due to how similar the datasets are to the synthetic data that was used for feature selection. The MSR dataset is much more similar to the synthetic data than the UCF-YT dataset, which may explain why the more aggressive g_{max} selection performs better on the former while the more robust g_{min} selection performs best on the latter. More specifically, the MSR dataset has a fixed camera and simple human motions, which matches the cinematography of the synthetic data (albeit varying in the specific actions). By contrast, UCF-YT exhibits highly variable cinematography and includes non-human actions (*e.g.*, horses and dogs) as well as actions with props (*e.g.*, basketballs and bicycles).

4.4. Comparison of base descriptors

The results of a comparison of base descriptors (trajectories vs. STIP-HOF) is shown in Table 1. Overall, the performance of STIP-HOF features is worse than that of trajectory-based ones. However, note that the best selection method (g_{min}) outperforms k-means for both STIP and trajectory features, and that g_{min} outperforms the other two methods for both features.

4.5. Comparison to unseeded RBF features

As an additional baseline we compare the performance of the features seeded from the synthetic data to that of random sets of features. The purpose of this baseline is to establish whether the gains seen with the classifier sets over k-means are due to the selection process, or whether the classifier-based features are inherently more informative than k-means histogram counts. As can be seen in Tables 1 and 3, the performance of random feature sets is very similar to that of codebooks produced by k-means, indicating that random classifier sets are by themselves about only as powerful as k-means codebooks. It is only after selection (either on the data itself, if there is enough, or on synthetic data) that significant gains are seen over the k-means baseline.

4.6. Comparison with feature selection on real data

We perform experiments using AdaBoost for feature selection on the MSR dataset (see Table 3). While boosting on the data itself improves performance on the clap action, the overall performance increase is modest, suggesting that when features are selected from the entire pool of 10,000 classifiers, boosting overfits. When the features are boosted from smaller subsets chosen at random, the overall performance is closer to that of unseeded features. However, the average performance of boosting on the real data is not much better than that of random subsets, and lower than that of seeded features.

Next, we evaluate the contribution of the synthetic data itself, in order to rule out the possibility that it is only the seeding technique (*i.e.*, randomly partitioning the data into groups and then evaluating aggregate performance) that produces performance gains. We perform our feature seeding using the real training data as the seeding source. In order to mimic the structure of the synthetic data groups (one action class vs. everything else), we partition the UCF-YT training data into groups, where each consists of one action class vs. the remaining 10. We further randomly partition each group into five, for a total of 55 groups. We then perform the feature seeding. These results are shown in Table 2. Note that for every selection method (*e.g.* g_{min}), the seeding from synthetic data outperforms the seeding from the real data. Additionally, the selection method g_{min} is the best regardless of the seeding source. Thus, the synthetic data itself plays an important role.

5. Conclusion

In this paper we propose feature seeding, a novel approach for using synthetic data to improve action recognition. Since the synthetic data (1) does not match the appearance of real world video and (2) is not guaranteed to contain the same actions as the test datasets, it is difficult to apply traditional domain adaptation, feature selection, or transfer learning approaches. Nevertheless, we demonstrate that seeding, which is a feature ranking selection technique on appropriately organized data, significantly improves performance on real world data. Seeding outperforms both the popular k-means quantization method and a more sophisticated engineered quantization method, demonstrating that even in very different action datasets there are deep commonalities that can be exploited.

Tellingly, features seeded from synthetic data have better performance than those seeded from the real data, despite the similar sizes of the datasets, indicating that the synthetic data itself contributes to the success of the tech-

nique. This highlights the potential benefits of appropriately constructed synthetic data (*i.e.*, where low-level descriptors are similar to real data and high levels of variation can be generated).

We believe that this general approach, in which synthetic data is used to select for robust algorithms, is an especially important avenue of exploration given the increasing demands placed on learning-based techniques and the sparsity of appropriately annotated data. Although the experiments presented here focus on the video action recognition domain, the proposed approach is broadly applicable to many learning-based vision tasks.

Acknowledgments

This research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0061. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV*, 2008. 2

[2] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *NIPS*, 2007. 2

[3] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, 2007. 3

[4] L. Cao, Z. Liu, and T. Huang. Cross-dataset action detection. In *CVPR*, 2010. 2

[5] Carnegie Mellon University Graphics Lab. CMU graphics lab motion capture database, 2001. 5

[6] G. Carneiro. The automatic design of feature spaces for local image descriptors using an ensemble of non-linear feature extractors. In *CVPR*, 2010. 2

[7] Y. Chen, R. Parent, R. Machiraju, and J. Davis. Human activity recognition for synthesis. In *IEEE CVPR Workshop on Learning, Representation and Context for Human Sensing in Video*, 2006. 3

[8] R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7, 2006. 2

[9] W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *ICML*, 2007. 2

[10] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of Association of Computational Linguistics*, 2007. 2

[11] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005. 2

[12] L. Duan, D. Xu, I. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. In *CVPR*, 2010. 2

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 2

[14] K. Lai and D. Fox. Object recognition in 3D point clouds using web data and domain adaptation. *International Journal of Robotics Research*, 29:1019–1037, 2010. 2

[15] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2, 3

[16] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, 2009. 5

[17] J. Loughrey and P. Cunningham. Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In *Research and Development in Intelligent Systems XXI*, pages 33–43. Springer, 2005. 2

[18] P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. In *ECCV*, 2010. 2, 3, 6

[19] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009. 2, 3

[20] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 79(3), 2008. 2

[21] N. Pinto, D. Doukhan, J. DiCarlo, and D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Computational Biology*, 5(11), 2009. 3

[22] F. Qureshi and D. Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. In *CVPR*, 2007. 3

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. 2

[24] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *Proceedings of International Conference on Pattern Recognition*, 2004. 2

[25] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. 2

[26] J. Shotton, A. Fitzgibbon, M. Cook, and A. Blake. Real-time human pose recognition in parts from single depth images. In *In CVPR*, 2011. 3

[27] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2

[28] X. Wang, C. Zhang, and Z. Zhang. Boosted multi-task learning for face verification with applications to web image and video search. In *CVPR*, 2009. 2

[29] T. Windeatt and K. Dias. Feature ranking ensembles for facial action unit classification. In *Proceedings Artificial Neural Networks in Pattern Recognition*, 2008. 2

[30] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009. 5