Understanding Short Social Video Clips using Visual-Semantic Joint Embedding

Thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science
in
Computer Science by Research

by

ADITYA SINGH
201002055
ADITYA.SINGH@RESEARCH.IIIT.AC.IN

International Institute of Information Technology
Hyderabad - 500 032, INDIA
DECEMBER 2017

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled " Understanding Short Social Video Clips using Visual-Semantic Joint Embedding " by ADITYA SINGH, has been carried out under my supervision and is not submitted elsewhere for a degree.

09/4/2018
Date

Adviser: Prof. P J NARAYANAN

Love You MOM.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. PJ Narayanan for the continuous support of my research, for his patience, motivation and guidance. I would like to thank Saurabh Saini and Rajvi Shah for their support, effort and sleepless nights they have put in helping and shaping my research. Parikshit Saurikar and fellow labmates for their off
on the topic discussions and to one and all, who directly or indirectly, have lent their hand in this venture.

Last but not the least, I would like to thank my family, for their constant support and to my friends without whom I would have finished my work earlier.

Abstract

The amount of videos recorded and shared on the internet has grown massively in the past decade. The most of it is due to the cheap availability of mobile camera phones and easy access to social media websites and their mobile applications. Applications such as Instagram, Vine, Snapchat allows users to record and share their content in matter of seconds. These three are not the only such media sharing platform available but the number of active monthly users are 600 , 200, and 150 million respectively indicate the interest people have in recording, sharing and viewing their content [1, 3]. The number of photos and videos collectively shared on *instagram* alone crosses 40 billion [1]. Vine contain approximately 40 million videos created by the users [3] and on a daily basis played 1 billions times. This cheaply available mode of data can empower many learning tasks which require huge amount of curated data. Also the videos contain novel viewpoints, and reflect real world dynamics. Different from the content available on older established websites such as Youtube, the content shared here is smaller in length (typically few seconds), contains description and associated hash-tags.

Hash-tags can be thought of as keywords assigned by the user to highlight the contextual aspect of the shared media. However, unlike english words these don't have a definite meaning associated to them as the description is heavily reliant on the content, along which the hash-tags are used. To clearly decipher the meaning of the hash-tag one requires the associated media. Hence, Hash-tags are more ambiguous and difficult to categorise than English words.

In this thesis, we attempt to shed some light on applicability and utility of videos shared on a popular social media website `vine.co`. The videos shared here are called *vines* and are typically 6 seconds long. They contain with them description composed of a mixture of english words and hash-tags. We try recognising actions and recommend hash-tags to an unseen vine by utilising the visual and the semantic content and the hash-tags provided by the vines respectively. By this we try to show how this untapped resource of popular social media format can prove beneficial for resource intensive tasks which require huge amount of curated data.

Action recognition deals with categorising the action being performed in a video to one of the seen categories. With the recent developments, considerable precision is achieved on established datasets. However, in an open world scenario, these approaches fail as the conditions are unpredictable. We show how vines are a much difficult categories of videos with respect to the videos currently in circulation for such tasks. To avoid manual annotations for vines we develop a semi-supervised bootstrapping approach. If one is to manually annotate vines this would defeat the purpose of easily available vines.

We iteratively build an efficient classifier which leverages the existing dataset for 7 action categories and also the visual, semantic information present in the vines. The existing dataset forms the source domain and the vines compose the target domain. We utilise semantic *word2vec* space as a common subspace to embed video features from both, labeled source domain and unlabeled target domain. Our method incrementally augments the labeled source with target samples and iteratively modifies the embedding function to bring the source and target distributions together. Additionally, we utilise a multi-modal representation that incorporates noisy semantic information available in form of hash-tags.

Hash-tags form an integral part of vines. Adding more and relevant hash-tags can expand the categories for which a vine can be selected. This enhances the utility of vines by providing missing tags and expanding the scope for the vines. We design a Hash-tag recommendation system to assign tags for an unseen vine from 29 categories. This system uses a vines' visual content only after accumulating knowledge gathered in an unsupervised fashion. We build a *Tag2Vec* space from millions of hash-tags using skip-grams using a corpus of 10 million hash-tags. We then train an embedding function to map video features to the low-dimensional Tag2vec space. We learn this embedding for 29 categories of short video clips with hash-tags. A query video without any tag-information can then be directly mapped to the vector space of tags using the learned embedding and relevant tags can be found by performing a simple nearest-neighbor retrieval in the Tag2Vec space. We validate the relevance of the tags suggested by our system qualitatively and quantitatively with a user study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, video cameras are integrated into almost every other device be it a mobile phone, a pen, a drone etc. Among them cellphones are the primary source for video recording among majority of the people in the world. They offer good resolution, frame rates, usability and hence easily qualify as, everyday pocket camera. Currently there are around 1500 million smartphone users [2]in the world and *quality of camera* ranks third among the factors which a consumer considers while buying a cellphone. As a result of increase in demand for the capturing devices the cost of recording, storing and sharing a video has decreased drastically over the last decade. As a result of this influx of mobile phones large amount of videos has been collected and shared by the general public.

Social networking sites such as *Google+, Facebook, YouTube, Instagram, Vine* allow the users to share their content, image & video, with the world. Their userbase consists of millions of people and the amount of data they share is proportionately large. For example, around 700 uploads are made on *Instagram* per second and around 12 million vines are linked to *Twitter* daily [1, 3]. *Instagram, Vine* are vastly popular with the younger generation as the process of recording to publishing the content online is more-or-less one step. Most of the websites not only allow upload of the visual content but also allow the user to add *hash-tags, description, location, and date of upload*. Other users can also contribute to the uploaded content by adding comments, upvotes, downvotes etc. This additional data bridges the contextual information provided by the video to its visual content.

The bundled information(visual and textual) easily available from the internet can help us solve a variety of problems. These videos being recorded by actual people and under unconstrained environment captures the real representation of the world. A direct use of such a data can be seen to form a dataset to train data hungry deep neural networks. Unlike existing datasets which are factitious, restraint by size, and unable to reproduce the real world, the community collected videos overcome these shortcomings and can prove to be the next benchmark for performance testing in variety of computer applications. One can use the videos available to train an action classifier, understand scenes, improve human recognition, boost object detection etc. The meta-data can be used in query expansion, recommendation systems, tag suggestion/refinement etc. User targeted advertisements and video recommendations can augment this bundled information into existing systems to expand their scope.

This thesis works on utilising cheaply available user shared videos on the internet for the task of action recognition and hash-tag generation. We believe these two areas of work help us understand the vines more as we deal with the visual and semantic content of vines. The premise of using such videos for action recognition is to raise the performance in an unsupervised scenario where manually annotating millions of videos for training is not suitable. This allows us to train classifiers for novel viewpoints which are not seen so far in the older datasets. Action in a video currently is judged by the visual content only, and we explore the possibility of improving the recognition for unseen vines utilising the semantic content of vines. Hash-tags form a crucial part of such videos and efficiently recommending such tags puts these videos ahead of their counterpart in terms of visual-contexual content they provide. For hash-tags we employ an opposite approach to action recognition as we leverage the semantic information gathered without supervision for effectively suggesting hash-tags to new vines. For both of our approaches we extend the use of existing systems which work to merge image and textual features to work for videos and show the advantages of utilising this abundant information has in a semi/un-supervised training environment.

1.1 Problem Overview

The available data we gathered is in form of videos and associated hash-tags collected from *Vine*. Hash-tags are different from tags which are available with images or videos from *Flickr, YouTube* as they are mostly internet slangs popular among the web community. Its important to differentiate between them as existing techniques manipulates tags and construct word hierarchy which is learnt from web documents, news sources etcbut as hash-tags co-exist in majority with uploaded images and videos similar strategy for hash-tags is not possible. Presence of contextual & visual information forces us to utilise them together instead of tackling one at a time.

1.1.1 Action Recognition

Currently the kind of videos handled by the systems are wild but are carefully gathered by the researchers for this work. The noticeable property of standard datasets is that the videos contain minimal camera motion and are shot with above amateurish skill both of which are lacking in the vines. Vines are heavily edited with lots of camera motion, changing perspective etcàll in a length of 6 seconds, the limit of it's length. Moreover, a video hash-tagged with an action label by the user, for example #cycling, doesn't guarantee the presence of the mentioned action. In this way the hash-tags serve as a noisy source of information. 1.1.1 show the mentioned hashtags along with few frame sequences from the vines. Due to the presence of intra-class negatives supervised approaches are left wanting, however, weakly supervised and unsupervised approaches which combine textual & visual modalities are the way to go. We employ a semi-supervised approach where we start with a labeled set of videos obtained from pervious datasets as a training set and videos gathered from vines as our testing set. The testing and

training datasets are very different from each other and hence represent two different domains. The difference in both the domains makes the problem non-trivial as such a difference is known to cause poor performance during testing. Moreover, combing the hash-tags to improve the overall performance is a task at hand. A recent approach for zero-shot learning learns an embedding function for bridging the semantic and image features. How it fares for video features remain untested to our knowledge and hence we adopt it to merge the class-label (the keyword used to download the vines, example cycling) and video representation which is boosted by the hash-tag based weighting function.

1.1.2 Hash-Tag Suggestion

Work has been going on for a long time over developing techniques for automated image, video annotations. These compose of methods to transfer tags from a known sets of tags. All these however mainly focus on some form of neighbourhood voting and/or graph propagation. Tags used in this context are meaningful english words for which many references and occurrences will be available in documents and web pages. Some tags associated with vines can be viewed in 1.1.1. Understanding the meaning of a hash-tag without viewing the associated content proves ambiguous. (d) *#sisterprobs* can mean a lot of different things but in the given context it can be associated to difficulty of kayaking together. Similar inference can be drawn from other examples.This exclusivity of hashtags to social media constraints us to synthesise an effective technique focusing on vines. We aim to solve this problem by constructing a semantic space for hash-tags which has not yet been done for this problem which will enable us to reliably associate tags for seen classes. Learning a [textitTag2Vec space is followed by learning an embedding which maps visual features of a vine to it's semantic features. This mapping then can be used for an unseen vine of the known category to suggest reliable hash-tags.

1.2 Contribution

We utilise and demonstrate the utility of a combined semantic and visual representation for the task for weakly supervised action recognition and tag suggestion activity. Videos are represented using fisher vectors computed from Improved Dense Trajectories. For action recognition we learn an embedding function for videos from a known dataset to map visual representation to a semantic representation. This embedding function is trained for only a seen distribution and performance of classification directly for vines is poor so we provide an iterative strategy to adapt this embedding function to perform well for vines. The words are represented as vectors obtained from word2vec's standard models. This model is constructed from documents, news reports, articles etcfrom the web. For hash-tag suggestions we gather a corpora of hash-tags and learn a *Tag2Vec* model, which to our knowledge has never been constructed before for vines and similar video platform, and represent a hash-tag as a vector in this space. We utilize the embedding function to learn the mapping from visual space to this newly designed tag2vec space and suggest appropriate tags. The advantage of embedding function is that it also helps in mapping

appropriately the classes which have not been seen before. The unseen classes however, shouldn't be entirely different from a seen class, for example, an image of a *cat* will be mapped closer to the word *cat* if training instances of *dogs* has been provided however for *trucks*, the embedding function will perform poorly.
Following are the key contributions of this thesis:

1. We provide a new dataset for video recognition, event detection etcconsisting of 3000 vines. It's one of the largest available dataset and also contains user assigned hash-tags.

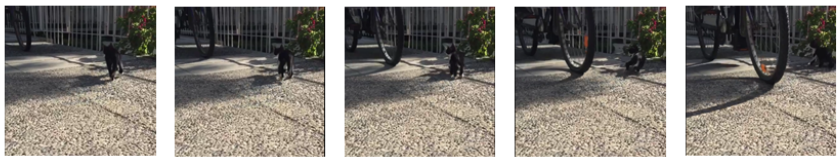
2. We propose a novel weakly supervised iterative domain adaptation strategy to classify actions in vines. To the best of our knowledge, ours is the first work which performs this work utilizing the shallow neural networks for learning the embedding.

3. We also provide the first hash-tag dataset consisting of over a million files gathered using 17000 most occurring english words as the query.

4. We demonstrate the utility of creating a tag2vec space and embedding function as a tool to suggest tags for videos.

1.3 Thesis Organization

Chapter 2 provides overview of prior work in the field of action recognition, domain adaptation, zero-shot learning and video annotations. Chapter 3 discusses the proposed weakly supervised embedding function adaptation method along with the dataset details, experiments. Chapter 4 covers the video annotation method proposed based on our Tag2Vec approach. Chapter 5 concludes this thesis with a discussion on limitation and future work.



#cycling #cats #kittensofvine #cute #funny #izmir

(a)

#cycling #london #fun

(b)

#kayaking #fail #carnage

(c)

#kayaking #irony #sisterprobs

(d)

Chapter 2

Motivation & Related Work

2.1 Motivation

Our motivation stems from the need to harness the profusion of data uploaded on the social media websites. The data available is unlabeled, unconstrained and captures wider spectrum of an action class. The advantage such a rich resource provides us is a possibility of creating a much more generalized datasets which reflect the real world appropriately as opposed to the manually annotated and curated videos available. Leveraging the available video collections to train a classifier for the wild videos which eventually can serve as a data gathering technique is what we explore. This following section provides a brief introduction to the related work which helped us to guide our system of unsupervised classification for vines.

2.2 Related Work

Videos on the web provide a rich source for training generalized classifiers. However, web-data is unlabeled and categorizing such data using traditionally trained systems has many limitations owing to dataset bias intrinsic to the collection process, unaccounted assumptions and domain differences. This chapter gives an introduction to the prior work on which this thesis is built on, covering computer vision algorithms for domain adaptation, zero-shot learning, action recognition and tag/multi-label anotation. Related work to classify actions for vines can be loosely categorized into **video datasets**, **video features**, **transfer learning**. Multi-label annotation can be segregated into **model based** and **data driven** approaches.

2.2.1 Video Datasets

Datasets form an itegral part to any classification, detection, recognition task. It helps to analyze pros and cons of the propose method and also a direct comparision with existing methods. The implication of performing well on datasets is that a given method will also fare well out in the world. Hence, a dataset

comprises of samples which represent a part of the real world. For action recognition specifically, [10], [17], [39], [28] etc. show chronologically increasing complexity of the datasets. [10] collected 90 low-resolution video sequences expanding over 10 classes of actions. The videos contain a static background, a single performer, are without any camera motion, and good lighting condition which causes videos to be heavily constrained and the actions being performed not a true representative of their real world counterpart. However, a simpler representation of the real world actions helped in containing the unconstrained problem and paved the way for improvements and development in the area. Fig 2.1 shows video sequences for 3 classes and their extracted foreground using simple algorithm of change detection on a static video.

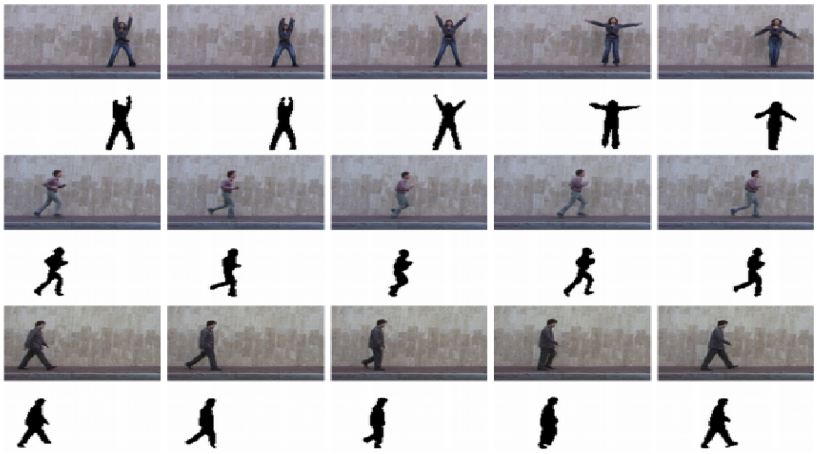


Figure 2.1: Video sequences and their respective foreground segmentation for wave-two-hands, run, and walk classes.

In 2011, the need for a much more real dataset was required due to the very high performances of new methods on the existing datasets as mentioned by [17] . They introduced the HMDB51 database, a collection of approximately 7000 videos ranging over 51 classes collected from movies and YouTube. This introduces high intra-class variability (see fig 2.2). They broadly categorize these 51 actions into general facial expression (laugh, chew), facial expressions with object interation (smoke, eat), genral body movement (clap, dive), body movements with object interaction (catch, golf), and body movements with human interactions (hug, kiss). As the source for the videos varies, they normalize the size and remove significant camera motion via. image stitching. They also contain meta labels describing the property of the clip which are (i) visible body parts (ii) camera motion (iii) camera viewpoint (iv) number of people involved in the action (v) video quality

A significantly larger dataset was introduced by [39] in 2012. The UCF101 dataset containing 13320 videos gathered from youtube for 101 action classes. These videos contain actions performed by single or multiple users under varying lighting condition with camera motion and occlusion. Similar to HMDB51 these videos of 101 classes are categorized into human-object interaction, body-motion, human-human interaction, playing musical instrument, and sports. However, unlike HMDB51 they

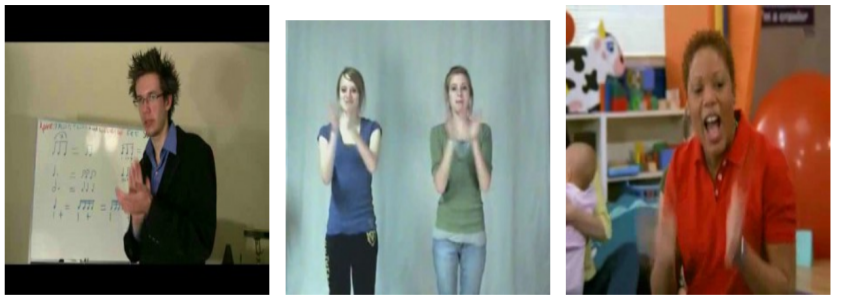


Figure 2.2: Variability in *clapping* action class.

don't provide meta-labels for a video. Fig 2.3 shows some classes for the database. TRECVID [28]

Figure 2.3: Sample scenes from 6 action classes from the UCF101 dataset.

has multimedia event detection (MED) as one of the components to promote content based analysis and retrieval of digital videos. As of 2016, TRECVID-MED composed of $700,000$ videos from Flickr and 9300 hr of footage gathered by Linguistic Data Consotrium and National Institute of Standard and Technology.

2.2.2 Video Features

We saw in the previous section how the complexities of the datasets over the years evolved. As the datasets became more complex the features required to describe them reveal the same trend.Early approaches in human action recognition focused on feature tailoring and spatio-temporal data [10, 27, 49], appropriate human body modeling [12, 13] and extension of popular image features to videos [34, 15, 46]. [10] treats action as shapes in space-time (see fig 2.4). These shapes contain human pose as well as motion information. They extract space-time features such as local space-time saliency, action dynamics, shape structure and dynamics. They extract these global features over a temporal range. To filter only the actions of the primary subject it is required that the background is known beforehand. They perform 1-NN with euclidean distance from a video to classify action of a test sample. Fig 2.5 shows the video sequence and corresponding features obtained. The algorithm is claimed to be robust to view-point change and partial occlusion but the dataset used to test was highly constrained and primal.

Figure 2.4: Space-time shapes of *jumping-jack*, *walk*, and *run* action classes.

Space-time interest points are the positions in space and time where sudden changes due to movement

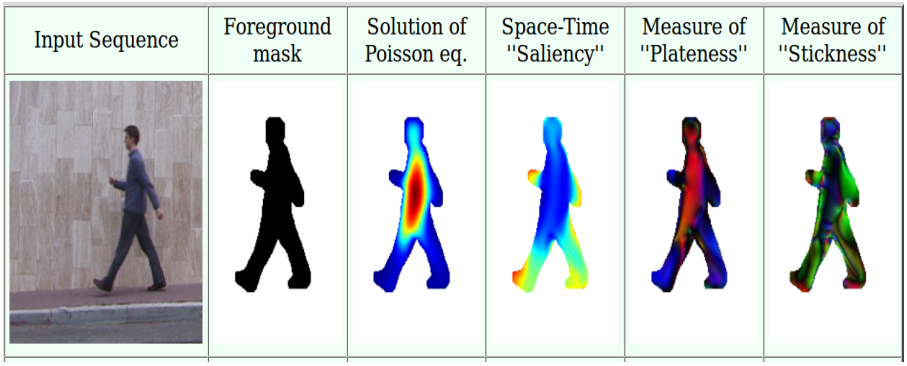


Figure 2.5: For a *walking* sequence the generated features are illustrated above.

occurs in a given video. [27] perform unsupervised action recognition using bag of spatio-temporal word representation. Unlike the previous approach which used global descriptors for videos, they rely on local descriptors. Video is representated as a collection of spatio-temporal words by extracted space-time interest points. A codebook is generated to describe a video as a histogram of these spatio-temporal words. The algorithm learns the probability distributions of spatio-temporal words by applying one of the two latent topic models of probabilistic Latent Semantic Analysis and Latent Dirichlet Allocation. They demonstrated the advantage of their approach for learning and predicting an action by maximizing the posterior probability of an action category (topic) distributions. They also achieved localization and categorization of multiple actions in one video. Fig 2.6 shows the workflow of their method. Recent approaches present increasingly complex features from large-scale dictionaries, fisher vectors, Convolutional Neural Networks etc. [44] rely on low level cues sampled from the entire videos. Large amount of interest points are tracked across frames, along these tracked points HOG, HOF, MBH are computed which describe shape and motion features around it. They remove motion induced due to camera by computing homography between consecutive frames. This is done by finding correspondences using SURF and (shi and tomasi,94) and using RANSAC to estimate the parameters of the model. It is made sure that the correspondences are not selected within a huan bounding box if available. Fig 2.7 shows the refined trajectories and optical flow obtained. Fisher vectors are used as a clustering method for the

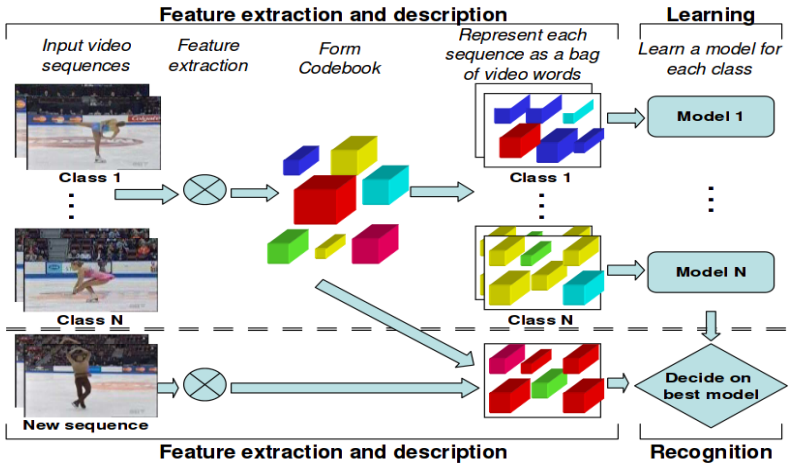


Figure 2.6: The local space-time regions are extracted using space-time interest points followed by clustering of these regions into a codebook. pLSA or LDA helps in learning probability distributions and intermediate topics which are further used for action recognition.

low-level features to represent a video. Integration of motion(Trajectory, HOF, MBH) with background information (HOG) allows them to achieve state of the art performances for majority of the datasets.

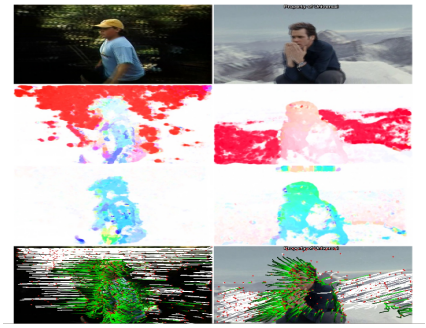


Figure 2.7: The camera motion (red) is removed from the optical flow which refines the tracking accuracy which in turns makes the method robust for a general setting.

Shifting away from handcrafted features (3D Convolutional Neural Networks for Human Action Recognition) proposed a framework to predict actions in human surveillance videos by training a convolutional neural network which extracts features from spatial and temporal dimensions 2.8. But it is highly expensive (if not utterly impossible) to build a generalized annotated datasets and features which are true to the real world class distribution. Although CNNs have shown promise in this regard, they require resource intensive supervised training and also implementation/extension of such methods is non-trivial.

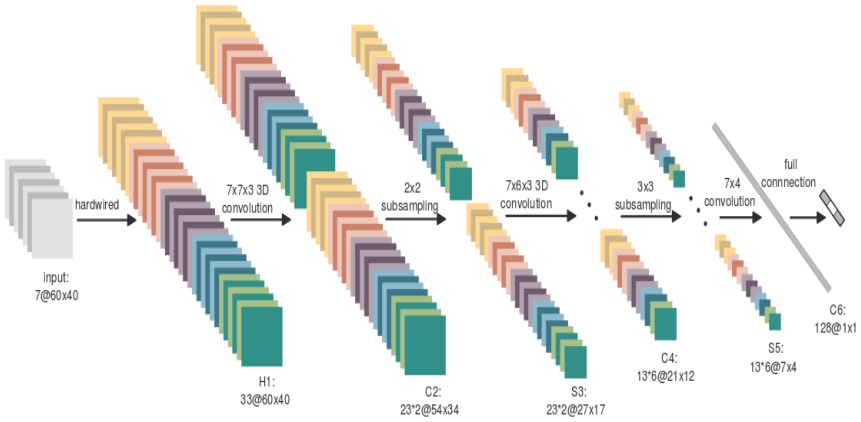


Figure 2.8: 3D CNN architecture for action recognition.

2.2.3 Transfer learning

Transfer learning aims at solving a given task using knowledge gained from solving similar/related tasks. For the task of recognition on a new unknown set, older much simpler datasets can be used. But using them directly is not feasible as samples of similar classes vary from one set to another. Different datasets for same classes form different distributions eventhough they represent similar concepts. This reflects biasness in the datasets. Transfer learning is applied to used to learn the task for a new distribution given some known distribution. The video datasets are biased even though they contain videos of similar content with respect to its direction, camera motion, actors etc. This means that a particular classifier trained on samples from one dataset will perform poorly if directly used on some second dataset. The approach is to analyze the bias/shift between the source and target dataset, and exploit it to appropriately transfer this information while constructing the target system. However, videos are not only full of visual, auditory data they also contain textual information provided by the uploader inform of tags, comments, description, location etc. These form the context of a given video or an image. We go through uni-modal appraoches of domain adaptation after which we look at works done for a multi-modal scenario.

2.2.3.1 Uni-Modal Approaches

[42, 32] discuss the problem of differences arising from bias in the dataset. The intra & inter class variability differ from one dataset to another. The work aimed specifically for object recognition in images it can easily be extended for action recognition in videos. They raise an important point of having methods perform well for an established datasets but remain still wanting in a real world scenario. Heterogeneous/multi-domain transfer [11, 9, 51, 29] etc. methods have shown promise in building more generalizable systems. However, most of these methods work on image/object category problems or text-data problems and their application to web-scale wild video distributions remain untested. [18]

learns a non-linear transformation between domains (distributions) in the kernelized space. However, this method requires labels from source and target domain. The supervised samples from both the domains form class based constraints to learn an efficient transformation between them. [14] proposes a max-margin framework which learns set of biasness-weights for each dataset. They theorize that it by removing the biasness effect from each dataset a common universal set can be obtianed which inturn can solve the problem of poor performance across datasets. This they achieve by undoing the bias present in each dataset utilizing the weights learnt for individual datasets and obtain visual world weights. This method is also supervised as it requires labels for source and target samples. Both the above mentioned work tried to find a relation in the feature vector space of the datasets. [40] worked on encoding such features which can account for the biasness present in the datasets. They show that one can achieve reasonable action classification accuracies on current datasets by training a classifier just from the background data, which for an action recognition task shouldn't be the case. Also they show the performance drop in recognition if the training and testing samples come from different datasets. Working on the premise that background should have minimal influence over an action recognition task and relevant features from the the foreground should have relatively higher weightage they propose foreground-weighted representation of features which performs well in recognizing actions across datasets. However, to obtain foreground-background separation is a hard problem in itself.

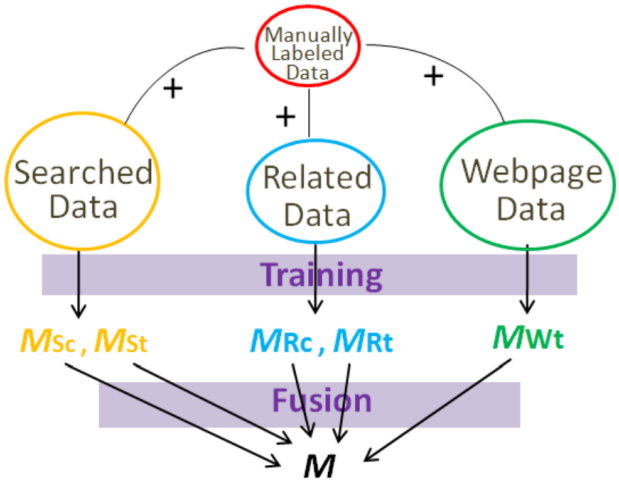
2.2.3.2 Multi-Modal Approaches

[23] presented with a theoretical analysis of the problem of domain-adaptation with multiple sources. They showed that, remarkably, for any fixed target function, there exists a distribution weighted combining rule that has a loss of at most ϵ with respect to any target mixture of the source distributions. They demonstrate the eficacy of their results on a sentiment analysis dataset which consisted of reviews in a textual form and ratings. The work of [48], which came out in 2010, uses hierarchical category taxonomy tree, designed by professional linguists, to categorize large scale YouTube videos. They provide a fusion framework where a small set of labeled videos are extended using information such as related videos, searched videos and text-based webpages 2.9.An important point to note is the small set of labeled videos are from the youtube itself and not from another source. They experiment on about 80K videos from 29 most frequent categories. However, this approach falls short in the generalization and extensibility for the task of action classification. It is difficult to handcraft a category taxonomy specific to human actions.

Zero shot learning targets solving a task for which it wasn't trained. [38] perform image feature embedding into the semantic word space for novelty detection. A new unseen image is classified as a novel class if it doesn't lie on the manifold of seen images.

A recent related work [16], formulate zero-shot learning as an unsupervised domain adaptation problem. They propose a regularized sparse coding based framework for improving the zero-shot performance for object and action recognition. This assumes disjoint source and target labels and restricts embedding function to be a linear transformation. [50] utilize data augmentation and self-training to

(a)



(b)

Figure 2.9: Samples of youtube videos for two categories used in [48] along with an overview of the method proposed.

improve visual to address the problem of projection complexity and generalization across domain-shift. They utilize semantic embedding representation and learn the mapping for zero-shot action recognition making use of regressors and improve on the mapping uby adding target training data with auxiliary training data.

2.2.4 Multi Label Video Annotation

Video annotation and tagging is tackled by the following two approaches:

2.2.4.1 Model Based

These methods mainly apply several concept based detectors, pre-trained with low-level video features, and use the resulting concept labels for effective tagging.
In 2007, [33] proposed integrated multi-label annotation approach. They simultaneously model both the individual concepts and their interactions in a single formulation, by doing this they minimize the error propagation of individual binary concept detectors. They focus on multi-label results and use Gibbs random field based mathematical model.
[19] construct a joint probability of visual region-based words with text annotations, incorporating co-occurrent visual features, and co-occurrent annotations to demonstrate that statistical methods can be used to retrieve videos by content.
[26] proposed a probabilistic Bayesian Multinet approach to explicitly model the relationship between the multiple concepts through a factor graph which is built upon the underlying video ontology semantics. However, the main drawback of model-based approaches is the limit on detectors that can be trained. As there are thousands of concepts for which it is difficult to gather large training data for reliable learning. Due to this, data-driven similarity based methods are preferred.

2.2.4.2 Data Driven

These methods utilize the abundance of videos shared by users and transfer tags based on similarity measures.
[5] proposed a video retagging approach based on visual and semantic consistency. This approach however only acknowledges tags which are nouns in the WordNet lexicon. Often the hash-tags are informal internet slang-words which rarely occur in proper language documents and hence do not have semantic consistency.
In their approach each video is segmented into shots, using a fast and simple algorithm that analyzes frame luminance and uses a global threshold to detect transitions and large content changes. Relevance of the video tags is computed for each shot, possibly eliminating tags that are not relevant, then new tags are added to each shot. The tags associated to a video are used to select and download from Flickr a set of images, which are clustered using k-means. The keyframes from the shots are assigned to the nearest centre and based on some comparisons a tag from the images is assigned to the keyframe. The overview of their proposed system can be referred from the Figure 2.10

[53] tackle the problem of *cross-source tagging* which uses existing tags in one search engine to tag videos available in other search engines, & *cross-time tagging* which uses existing tags collected over years to tag recently uploaded videos. In their approach the videos are crawled offline and a corpus is constructed. After pre-processing these videos key frames are selected on top of which clustering is performed to obtain bag of words for these keyframes. Inverted file indexing plus Hamming embedding [7] is employed to support scalable keyframe retrieval with fast similarity evaluation. For queries similar approach is used. For tagging, the collection of candidate tags is pooled from the set of nearest

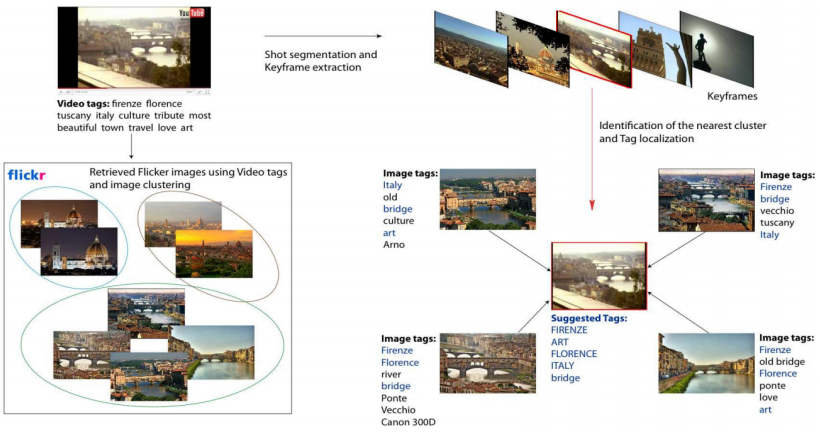


Figure 2.10: The video annotation system based on semantic and visual consistency.

matching retrieved videos. An effective measure which considers tag frequency, the number of tags, and the similarity weight of videos is proposed to rank the tags according to their relevance.

[25] devices a graph reinforcement framework to propagate tags developed by crawling tags of similar videos for annotation by using text and visual features. The graph supplies possible annotations of a different modality (e.g., text) that can be mined for annotations of the target. Their system first collects similar videos and generates small graphs for different visual features. A stable graph is found by iterative diffusion using similarity matrix over the connected nodes and then annotations are mined from the weighted annotations of each stable graph. A zipf-based cutoff is used to determine the relevant annotations for a particular video. The overview of the system is depicted in 2.11

[47] computed the similarity between two samples along with the difference in their surrounding neigh-

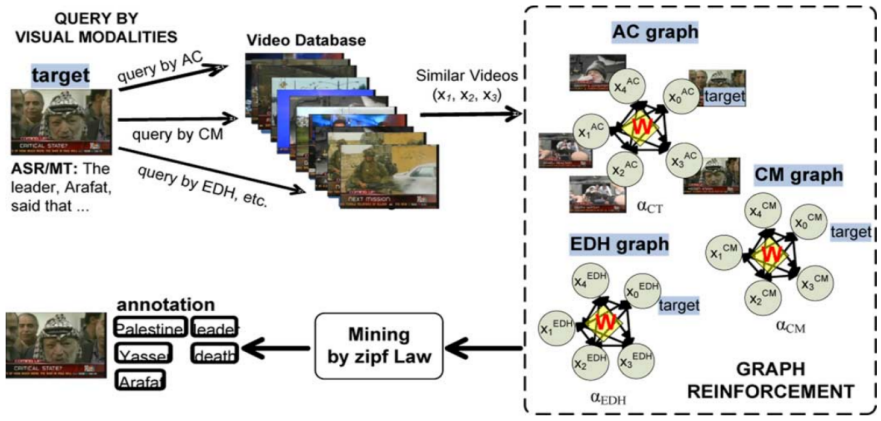


Figure 2.11: Overview of the graph reinforcement approach.

bourhood sample & label distribution. The neighbourhood sample similarity is computed using KL

divergence and label similarity is based on difference of label histograms of the two samples. Their neighborhood similarity between two samples simultaneously takes into account the following three characteristics:

- Their distance
- The distribution difference of the surrounding samples
- The distribution difference of surrounding labels

[52], in a much recent work in 2013, utilized the user click-through data along with the similarity based measures to tackle the problem of video tagging. Similarity is computed by measuring the click-through data and video document features. Visual question answering (VQA) systems [4] employ deep learning to train an answering system for natural language queries. Though deep features have shown promise in image based captioning and VQA systems, one either needs a huge amount of data to train deep networks or needs to fine-tune a pre-trained network. For videos such pre-trained networks are not readily available yet and though we have collected a dataset of nearly 3000 short videos, it is not sufficient to train a deep network. Hence, we use the state-of-the-art hand-crafted features for our application. On a related note, recently [41] released an interesting video based question answering dataset by aligning book descriptions to movie scenes. Our work however has a different focus in its application to hash-tags and wild social video clips.

2.2.5 Embedding Learning

2.2.6 Word Embedding

Common approaches for word embedding is through neural networks [37], dimensionality reduction of co-occurrence matrix. [20, 21, 22], explicitly constructed probabilistic models [8] etc. Amongst the early approaches, [7] reduces the high dimensionality of words representations in contexts by learning a distributed representation for words. They use a feed forward neural network with a linear projection layer and a non-linear hidden layer which jointly learns a word vector representation and a statistical language model. Currently, widely popular technique of [24] proposes method for learning word vectors from a large amount of unstructured data. They also show that the learned space is a metric space and meaningful algebraic operations can be performed on the word vectors. [6] proposes a bayesian skip-gram method which maps words to densities in a latent space rather than word vectors which results in less effort in hyperparameter tuning.

2.2.7 Summary

In this chapter we presented an overview of related work in the field of action recognition and domain adaptation. We analysed some of the milestone datasets and how each one of them differs from the

other. We saw the shift from artifically generated video clips to community submitted videos and as a result the datasets evolved to be representative of real world actions. We understood few feature extraction strategies proposed for videos, both hand tailored and neural network generated along with their shortcomings. We lastly delved into transfer learning approaches proposed for uni-modal and muti-modal data. As the field of action recognition tied with zero-shot learning is fairly new, the work in this thesis can be extended and enriched in countless ways.

Chapter 3

Action Classification: Embedding Space Adaptation for Vines

As we have discussed, current methods treat actions and tags as separate entities which restricts the knowledge of the domain. We propose an embedding space adaptation method which transfers the knowledge from a known source (UCF) to an unknown source (Vine) while making use of the hash-tags. This chapter covers the details of the dataset, video representation, algorithm and experiments.

3.1 Dataset

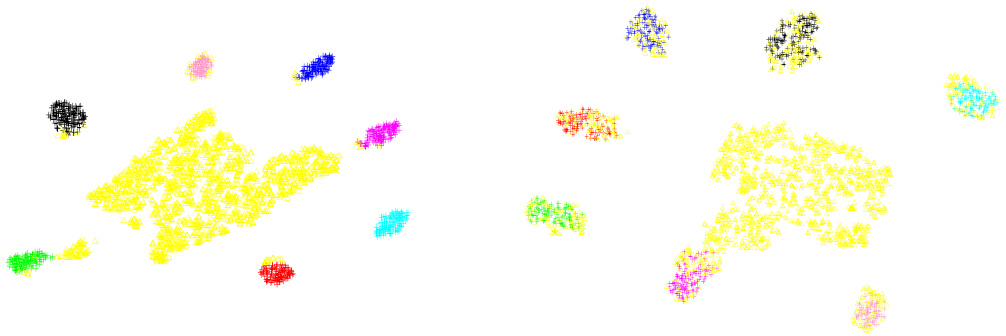
3.1.1 Source Dataset (UCF50)

We work on seven of the 50 classes of UCF50 based on the general availability. The classes are, Biking, Billiards, Diving, Golf, Horseriding, Kayaking, Pushups. Each class contains approximately 120 videos. These videos form our auxiliary or source dataset.

3.1.2 Target Dataset (Vines)

Vines are 6 second clips uploaded by the users from their cameras to a popular media sharing website *vine.co*. No two vines capturing the same concept are identical. They are heavily edited (contain texts, stop motion, changing perspective), amateurish etc. They also come along with hash-tags which users consider appropriate fro their content. People are free to create their own hash-tags. We have downloaded 450 vines for 7 different classes using Billiards, Cycling, Diving, Golf, Horseriding, kayaking, and Pushups as the keyword. The retrieval of vines is based on the occurrence of query word in either the corresponding 'hash-tag' or description. We discard the vines which do not contain action category as the hash-tag. Thus, we have a total of 2357 vines and associated hash-tags. These vines are split into testing and training set. The vines in the testing set are annotated by human operators.

Table 3.1 shows the distribution of samples across classes and auxiliary, target train, and test sets. Since, the hash-tags are noisy, many vines in the target set do not have the respective action (false positives). The test set is pruned to remove all such false positives. However, since our training is unsupervised, we do not alter the target train set. The first two rows in Table 3.1 shows the total



(a) (b)

Figure 3.1: t-SNE Visualization of semantic embedding of UCF and Vines before and after iterative training. The crosses represent auxiliary (UCF) samples and are color-coded according to their class labels. The yellow triangles represent unlabeled target training samples (vines). After iterations, many more vines merge with the clusters formed by the auxiliary samples. The leftover vines possibly belong to none of the action classes and hence do not merge into any cluster. Please see this visualization in color.

samples in the target train set and the number of true positives for each class. We provide this statistic to demonstrate the fact that hash-tags are extremely noisy labels. This fact can also be observed from leftover vines in Figure 3.1.

Action Class		Billiards	Cycling	Diving	Golfswing	Horseride	Kayaking	Push up
Target Domain (vines)	Train (total)	267	280	258	268	233	284	286
	Train (true +ves)	100	92	133	151	106	73	178
	Test	24	27	39	34	29	16	40
Auxiliary Domain (UCF)		129	119	124	120	169	129	90

Table 3.1: Number of samples in auxiliary and target sets across classes. For some classes the true positives are less than 25% of the total samples in the target train set. This imbalance indicates the fact that 'hash-tags' can only provide noisy labels and other modalities need to be utilized for effectively labeling target training vines.

3.2 Feature Representation

Many feature representations based on spatio-temporal constructs [10, 27, 49], appropriate human body modeling [12, 13], successful image features [34, 15, 46], etc. are proposed in action recognition literature. Our approach leverages multi-modal feature representation to reliably augment the auxiliary set with target samples. We use motion features, semantic embedding features, and tag-distribution features in our method. We explain these features and the related terminology in this section. More details on parameters and code are given in item 3.5.

3.2.0.0.1 Motion features: Motion encoding is the most preferred feature representation for action recognition training. We compute the fisher vector encoded improved dense trajectories (IDT) [44] for samples of both auxiliary and target sets. IDT features include histogram of oriented gradients (HoG), histogram of optical flow (HoF), and motion boundary histogram (MBH) descriptors across frames. To classify these features, we use linear support vector machines (SVM).

3.2.0.0.2 Semantic features: [24] provide a mechanism to represent a word as a vector in a 300-dimensional vector space, commonly known as *word2vec* space. [38] proposed a neural network based supervised method to learn a non-linear function that embeds visual (image) features into the word2vec space based on the corresponding object category words. We use this framework and learn a semantic embedding function that projects motion features (fisher vectors) into the word2vec space corresponding to the action word. We call the resulting 300-dimensional representation, embedded semantic features, or simply semantic features.

3.2.0.0.3 Tag Features: Hash-tags can be seen as noisy semantic labels of a video provided by the users. We assume that similar videos will have similar tags. Tagged words are usually slangs which are used to describe a video in an informal manner and hence do not strictly adhere to the word2vec representation of word space. Hence, we utilize tag features in a separate framework. First, we collect all tags associated with the vines in our target dataset and perform stemming to obtain cleaner, non-redundant set of tag-words. We then create a histogram of all tag-words and form a tag dictionary by removing all singleton words. A tag feature for a vine is simply a binary vector of the dimension of the dictionary (1048 in our experiment), such that the value in i^{th} position indicates whether the i^{th} tag in the dictionary is associated with the vine or not.

3.3 Prelim. Experiment

We conduct a small experiment to see if actually addition of training samples will help in classifying the vines. The embedding space is modified slowly by the addition of labeled positive samples from the training vine set and distance of test samples from class labels is measured. 10% of labeled positives are added in every iterations. The average distance of labeled positve are computed at each iteration. The table 3.3 shows how the distance over iterations decreases for the test samples. This however is when we are training using the labeled positives. As in our approach we will not be using the labels for training our aim will be to obtain a modified embedding function which performs on a similar line.

3.4 Method

Figure 3.2 shows a block diagram of the proposed iterative training method. Each step of this method is explained in detail here. We first describe the notations used, then discuss the strategy for initializa-

Table 3.2: Nonlinear Model Results

Samples Added	Cycling	Diving	Golf	Horseriding	Billiards	Kayaking	Pushups
10%	0.6960	0.6972	0.6211	0.6115	0.6830	0.6092	0.5493
20%	0.6521	0.6264	0.5557	0.5396	0.5962	0.5270	0.4871
30%	0.5862	0.6099	0.5119	0.4910	0.5551	0.4719	0.4212
40%	0.5368	0.5689	0.4765	0.4854	0.5320	0.4697	0.3930
50%	0.5090	0.5480	0.4601	0.4845	0.5075	0.4726	0.3654

Table 3.3: The decreasing distance over addition of samples is indicative of the fact that a method which can reliably add vine positives should be able to reproduce similar results.

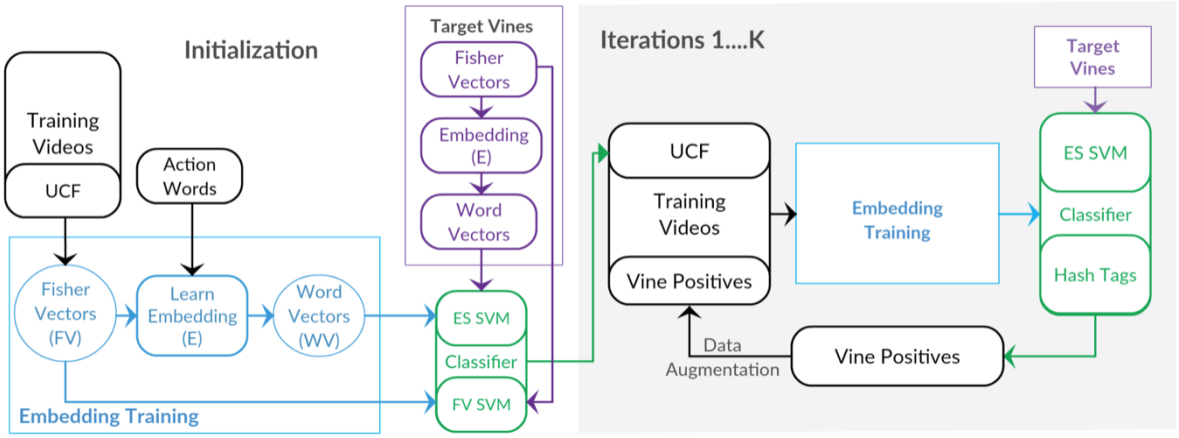


Figure 3.2: Block representation of our iterative training approach. Left block shows the specifics of the initialisation step and the right block shows the process for iterative updates. The embedding training sub-block (blue) is the same for both initialization and iteration steps (depicted in detail for only initialisation step).

tion and incremental updation of the training set, followed by explanation of these updation rules, and sampling choices. The outline of the algorithm is provided 1 for better understanding of the described method.

3.4.0.0.1 Notation: We denote the set of action categories as $\mathbb{C} = \{c_i \ | i \in [1, 7]\}$, where $c_1 - c_7$ represent the seven action categories 'billiards', 'cycling', 'diving', 'golf', 'horseriding', 'kayaking', and 'pushups'. The negative label corresponding to an action category c_i is represented as $\tilde{c_i}$. The SVM classifiers for fisher vectors and embedded semantic features are denoted respectively by $\mathrm{H_{FV}}$ and $\mathrm{H_{WV}}$. The auxiliary set features are represented as $\mathbb{A}$. The training set at k^{th} iteration for learning the classifiers for category c_i is denoted by $\mathbb{T}_k^{c_i}$. The sets of positive and negative examples in $\mathbb{T}_k^{c_i}$ are denoted respectively as $\mathbb{P}_k^{c_i}$ and $\mathbb{N}_k^{c_i} = \bigcup_{j \neq i} P_k^{c_j}$. The target set of unlabeled vines with hash-tag c_i are represented as $\mathbb{U}^{c_i}$. At k^{th} iteration, the set of remaining unlabeled vines is $\mathbb{U}_k^{c_i}$, $\mathbb{U}_k^{c_i} \subset \mathbb{U}^{c_i}$.

3.4.0.0.2 Initialization: The initial training of classifiers for class c_i is performed using the auxiliary set (UCF50 examples) as the training set. At this point, all vines in the target set are unlabeled. Hence, for the 0^{th} iteration,

$$\mathbb{T}_0 = \mathbb{A},\ \mathbb{P}_0^{c_i} = \mathbb{A}^{c_i},\ \mathbb{N}_0^{c_i} = \mathbb{A}^{\tilde{c_i}},\ \mathbb{U}_0^{c_i} = \mathbb{U}^{c_i}$$

For each class, we train the SVM classifiers $\mathrm{H_{FV}}$ and $\mathrm{H_{WV}}$ using samples from the initial training set $\mathbb{T}_0$. The SVM classifier for each class return a confidence score $\in [0, 1]$ for each target sample. The two SVM scores are multiplied to yield a combined confidence score for every vine sample in the unlabeled target set $\mathbb{U}_0^{c_i}$. The multiplicative scoring function penalizes the overall score when any of the two scores is low and helps to ensure that only the samples with highest confidence are labeled. We pick the top-K scoring vines as potential positive samples, where K is emperically selected to be 10% of the auxiliary positive set ($|\mathbb{P}_0^{c_i}|$) at every iteration. We update the negative set for a class c_i by adding the newly labeled positives of other classes as labeled negatives for c_i. The auxiliary set size can be fixed or modified incrementally as explained later. The positives and negatives in iteration k for the class c_i are denoted as $\mathbb{L}_k^{c_i}$ and $\mathbb{L}_k^{\tilde{c_i}} = \bigcup_{j \neq i} \mathbb{L}_k^{c_j}$. Note, we don't use tag based scoring in the initialization due to unavailability of tags for the initial auxiliary set.

3.4.0.0.3 Iterative training and update: The training set for class c_i at iteration $k > 0$ is formed by augmenting the newly labeled vines to the auxiliary set.

$$\mathbb{T}_k^{c_i} = \mathbb{T}_{k-1}^{c_i} \cup \mathbb{L}_{k-1}^{c_i} \cup \mathbb{L}_{k-1}^{\tilde{c_i}}, \qquad \mathbb{P}_k^{c_i} = \mathbb{P}_{k-1}^{c_i} \cup \mathbb{L}_{k-1}^{c_i}, \qquad \mathbb{N}_k^{c_i} = \mathbb{N}_{k-1}^{c_i} \cup \mathbb{L}_{k-1}^{\tilde{c_i}}$$

The parameters of the embedding function are re-estimated using the augmented training set. Our hypothesis is that by incrementally adding more vines to the training set, in each iteration, we slowly modify the initial embedding that worked well for the auxiliary set to adapt for the target set (vines). As we are not altering the fisher vector space, we drop the motion feature classifier ($\mathrm{H_{FV}}$) after the initial iteration. The tag score for a target vine in iteration k is the average number of co-occurring tags between given vine and positively labeled vines in the previous iterations. The tag-score s_t is computed as follows,

$$s_t(x_t^v) = \frac{1}{|\mathbb{L}^{c_i}|} \frac{1}{\bar{n_t}} \sum_{x_p \in \mathbb{L}^{c_i}} \sum_{i=1}^{N_D} (x_t(i) * x_p(i)), \quad \text{where,} \quad \mathbb{L}^{c_i} = \mathbb{L}_{k-1}^{c_i} \cup \mathbb{L}_{k-2}^{c_i} \cup ... \mathbb{L}_1^{c_i}$$

Here, N_D is the size of the tag dictionary and $\bar{n_t}$ is the average number of tags per vine in the target set (15 in our experiment). The combined score of a target training vine is computed by multiplying the semantic space SVM confidence scores and the tag-score. The tag-score boosts the overall score of the test vines that have many co-occurring tags with the previously labeled positive vines. The tags help in distinguishing samples of different classes retrieved as a result of the hash-tag. For example, Apart from 'diving', 'sky-diving' and 'diving in a pool' will have different accompanying tags which will match accordingly to the currently classified/labeled vines. We stop the iterations when we have labeled approximately 50% of the auxiliary positives, i.e. $P_0^{c_i}$.

3.4.0.0.4 Sampling choice for auxiliary set augmentation: In addition to augmenting the auxiliary set, we also perform an experiment where we gradually replace the auxiliary samples by target samples. This approach allows us to diminish the influence of auxiliary samples and provide more priority to target samples. We evaluate the performance of our method for both sampling choices, with and without replacement in section 3.6.

Algorithm 1 Our Approach

$\theta = 0.1 * |\mathcal{P}_0^{c_i}|$
procedure INITIALIZATION ▷ Initializes positve and negative sets for training
  for `each class` c_i **do**
    $\mathcal{P}_0^{c_i} = \{(x_f^a, x_s^a, l^a) \,|\, l^a = c_i\}$
    $\mathcal{N}_0^{c_i} = \{(x_f^a, x_s^a, l^a) \,|\, l^a \neq c_i\}$
    $\mathcal{U}^{c_i} = \{(x_f^v, x_s^v, x_t^v) \,|\, l^v = c_i\}$
    $\mathcal{S}(\mathcal{U}^{c_i}) = h_f(\mathcal{U}^{c_i}) * h_s(\mathcal{U}^{c_i})$
    $\mathcal{L}_k^{c_i} = \arg\max_\theta S$
    $\mathcal{L}_k^{\tilde{c_i}} = \bigcup_{j \neq i} \mathcal{L}_k^{c_j}$
    $\mathcal{U}^{c_i} = \mathcal{U}^{c_i} \setminus \mathcal{L}_k^{c_i}$
  end for
end procedure

procedure ITERATIONS ▷ Adding of new samples based on semantics and hash-tags
  for `iterations` **do**
    for `each class` c_i **do**
      $\mathcal{T}_k^{c_i} = \mathcal{T}_{k-1}^{c_i} \cup \mathcal{L}_{k-1}^{c_i} \cup \mathcal{L}_{k-1}^{\tilde{c_i}}$
      $\mathrm{P}_k^{c_i} = \mathrm{P}_{k-1}^{c_i} \cup \mathcal{L}_{k-1}^{c_i}$
      $\mathrm{N}_k^{c_i} = \mathrm{N}_{k-1}^{c_i} \cup \mathcal{L}_{k-1}^{\tilde{c_i}}$
      $\mathcal{S}(\mathcal{U}^{c_i}) = s_t(\mathcal{U}^{c_i}) * h_s(\mathcal{U}^{c_i})$
      $\mathcal{L}_k^{c_i} = \arg\max_\theta S$
      $\mathcal{L}_k^{\tilde{c_i}} = \bigcup_{j \neq i} \mathcal{L}_k^{c_j}$
      $\mathcal{U}^{c_i} = \mathcal{U}^{c_i} \setminus \mathcal{L}_k^{c_i}$
    end for
  end for
end procedure

3.5 Experiments, Results and Analysis

In this section, we explain the experimental setups, establish the baseline performance, and finally report the results of our method and present our interpretations. The list of experiments conducted are as follows:

1. Action recognition performance on Vine test
2. Semantic space distance

	Iterative Training(Ours)						Baseline Methods					
	With-replacement			Without-replacement			FV_svm			ES_knn		
Class	prec.	rec.	F-score	prec.	rec.	F-score	prec.	rec.	F-score	prec.	rec.	F-score
Billiards	.750	.875	.807	73.3	.916	**.814**	1.00	.166	.285	1.00	.250	.400
Cycling	.956	.814	**.880**	.884	.851	.867	.585	.888	.705	.621	.851	.718
Diving	.750	.538	.626	.814	.564	**.666**	.645	.512	.571	.620	.461	.529
Golf-Swing	.909	.588	.714	1.00	.588	**.740**	.641	.735	.684	.638	.676	.657
Horseriding	.634	.896	.742	70.2	.896	.787	.857	.827	**.842**	.857	.827	**.842**
Kayaking	.388	.875	**.538**	.361	.812	.500	.407	.687	.511	.333	.750	.461
Pushups	.967	.750	.845	.969	.800	**.876**	.846	.825	.835	.891	.825	.857

Table 3.4: Comparison table for our methods with the baselines.

3.6 Experiment: Action Recognition for Vine test

Here, we explain the code setup and parameters used for feature computation and classifier training. To extract the motion features, we compute the improved dense trajectory descriptors [44, 45] for all videos using the code[1] by the authors. For fisher vector computation, the experimental parameters are same as [45] and the GMM paramters are estimated over UCF samples of 50 action classes. The fisher vectors thus computed are of $101,376$ dimensions. For computing the semantic features, we embed the fisher vectors into the 300 dimensional word2vec space. For learning the embedding function, we use publicly available implementations of word2vec[2] and zero-shot learning [3]. For initializing the embedding function we use 400 hidden nodes and limit the maximum iterations to 1000. For computing the dictionary of tag-words, we first perform stemming – a commonly used trick in NLP applications to reduce the related forms of a word to its root form. From the remaining words we remove the singletons. The tag feature for a vine is a binary vector of the size of the tag dictionary such that each 0/1 element indicates whether that tag in the dictionary occurs with this vine or not. We use linear SVMs for fisher vector and semantic features classification in our iterative training. For SVMs we fix $C = 1$ and the weight for the positive class to be 7 times more than the negatives to compensate for fewer positive samples as compared to the negatives being added in each iteration.

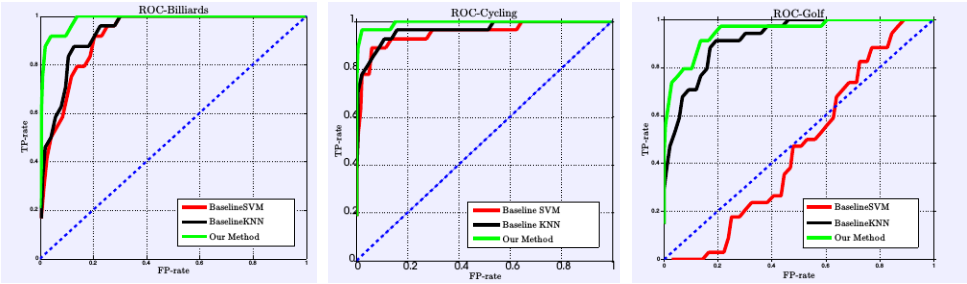
3.6.1 Performance Evaluation

We evaluate the performance of our approach by classifying a test set using the semantic embedding learned in the final iteration. We compare the performance of our method with the baseline classifiers trained on the auxiliary dataset (UCF50) for semantic and motion features. We report precision, recall, and F-score for classification of the test dataset using all methods in Table 3.4 and the ROC-curves for

[1] https://lear.inrialpes.fr/people/wang/improved_trajectories
[2] https://code.google.com/p/word2vec/
[3] https://github.com/mganjoo/zslearning

three classes are shown in Figure 3.3. The two baseline classifiers trained on the auxiliary dataset are represented as, (i) Motion-only (FV+SVM), and, (ii) Semantic-only (ES+NN).

For Motion-only baseline, we train 7 one-vs-rest linear SVM classifiers and assign the labels based on the highest decision value of the corresponding classifiers. For Semantic-only baseline, we train an embedding function using the auxiliary set and use it to project the test samples to the semantic space. We classify the samples to the nearest class in the word2vec space using L_2 distance. The semantic embedding function learnt using our iterative approach is also evaluated in a similar fashion (ES+NN). We evaluate our final semantic embedding for both sampling choices , when (i) the auxiliary set videos are replaced by the newly labeled vines (with replacement), (ii) the auxiliary set is only augmented by the newly labeled vines (without replacement).



(a) ROC - Billiards (b) ROC - Cycling (c) ROC - Golf

Figure 3.3: ROC comparison of our iterative training method (without replacement) with baseline FV+SVM and ES+NN.

3.6.2 Analysis

Iterative relearning of embedding function In Figure 3.4, we show the effect of relearning the embedding function by t-SNE visualization of the test samples. In 0^{th} iteration, we see UCF samples forming separate clusters but vines forming an inseparable distribution. This shows that the embedding function learnt solely using auxiliary set is not adaptable on the test set. However, after iterations we see that the embedding projects most of the test samples around their class labels and close to corresponding UCF samples. This observation supports our hypothesis of incrementally modifying the embedding functin using augmented training set. If this hypothesis was faulty then the embedding function would degrade performance of the test classification after iterations (see Figure 3.4, Table 3.4).

Comparision of baseline classifiers Table 3.4 shows that the baseline methods, FV+SVM and ES+NN, have comparable performances. However, ES+NN operates in a significantly lower dimensional space (101376 vs. 300) and has much lower time complexity serving as a better alternative.

Figure 3.4: t-SNE Visualization of semantic embedding of UCF and Vines before and after iterative training.

Without-replacement vs. baseline classifiers Our method outperforms both the baseline methods for all the classes except 'Horseriding' & 'Kayaking'. For Horseriding we obtain a higher recall, however due to fall in precision the overall performance decreases. Though our method performs better than ES+NN but falls short on precision against FV+SVM. Kayaking contains the least ratio of true positives ($< 25\%$) in the target set (Table 3.1). Incremental addition of target samples still helps in improving precision and recall as compared to ES+NN baseline for 'kayaking'.

With-replacement vs. baseline classifiers Our method performs significantly better for all classes except 'Horseriding'. For 'Kayaking' our method performs marginally better and as mentioned earlier the improvement is limited due to the lack of positives in the target dataset. For 'Billiards', we achieve an acceptable precision with significantly higher recall against both the baselines which show a highly skewed performance.

With-replacement vs. without-replacement We experimented with these two sampling choices to see whether diminishing the influence of auxiliary domain will hamper or aid the process of learning the embedding function. It is evident from Table 3.4, for 5 of the 7 classes without-replacement performs better and for the other two classes the difference is marginal. This result suggests that augmenting the auxiliary domain by target domain without replacement has a positive influence on iterative learning.

3.7 Experiment: Semantic Distance Convergence

We show how our unsupervised training compares with that of a supervised one. We compute the distance of test samples using the embedding function obtained after the termination of our method. We can see from 3.6 that the distance over iterations indeed decreases and confirms to our initial experimental. These results however, are not directly comparable to the ones we've observed in 3.3 as the number of samples added at each iteration varies. In our approach we add a % of source dataset and during the prelim. experiment we added % of labeled vine positives.

3.7.1 Analysis

We can observe from the visual representation of the semantic space before and after the completiton of the method as how the samples seem to segregate. The initial vine cluster looks randomly spread around organized UCF training set and after the completition of the iterations the class-labels for ma separate cluster in the semantic space. Figure 3.4 shows the similar lableled samples to be forming a separate cluster after final iteration as the distance from the class labels diminishes.

3.8 Summary

In this chapter we introduced our vine dataset along with the method to construct an embedding space for action classification in vines. We utilize the embedding space learnt from source dataset as a

Table 3.5: Nonlinear Model Results

Iterations	Cycling	Diving	Golf	Horseriding	Billiards	Kayaking	Pushups
1	0.5245	0.6253	0.6199	0.4753	0.5880	0.4111	0.5657
2	0.5253	0.5959	0.6049	0.4595	0.5495	0.4221	0.5457
3	0.5056	0.5940	0.5893	0.4312	0.5270	0.4182	0.5410
4	0.4986	0.5818	0.5971	0.4059	0.4825	0.4358	0.5231
5	0.4796	0.5683	0.5861	0.3768	0.4914	0.4234	0.5303

Table 3.6: Similar to the treand observer in 3.3 the average distance of test samples decreases over iterations.

base which we modify over iterations. The two sampling strategies further help in analysing the positive/negative impact of biasness inherent in a dataset. To establish the advantage of our method we conduct two experiments firstly to check the action recognition performance and secondly to demonstrate the manoeuvrability of final embedding function. In the next chapter we utilize the vine videos to construct a hash-tag space which will enable us to assign hash-tags to any new test vine.

Chapter 4

Tag2Vec: Hash-Tags for Vines

We saw in the previous chapter how we can use visual and semantic features to effectively categorise vine videos. Word2vec helped us to obtain semantic aspect of the vines. The 300-dimensional space captured relations between words with nearby words signifying high correlation. However, for the purpose of suggesting hash-tags to a video this space is no good. Hash-tags can be a made up word which don't even exist in english vocabulary and will be hard to decipher if one doesn't understand the context in which it's used, for example, #SorryNotSorry is a complete phrase representing some aspect of the content a vine consists. We need to build a space which contains the relations between various hash-tags and then learn to efficiently suggest most confident ones to a vine. This chapter covers in detail the methodology, dataset details and experiments.

4.1 Method

The proposed system is trained using a large number of videos and associated hash-tags scraped from social media platform `vine.co`. Videos shared on this platform (commonly known as vines) are six seconds long, often captured by hand-held or wearable devices, with cuts and edits, and present a significantly wilder and more challenging distribution than traditional videos. For each uploaded video, the original poster also provides hash-tags. Unlike tag words typically used as meta-data, hash-tags serve more of a social purpose to improve content visibility and to associate content with social trends. Many hash-tags do not adhere to the commonly understood semantics of the natural language. Due to this reason, we learn a new tag space representation *Tag2Vec* instead of directly using semantically structured Word2Vec space of [24]. For example, `#FitFluential` and `#Mr315` are non-word hash-tags but understandable social media jargons given the content.

As mentioned previously, our end-to-end training for *video-to-tag* mapping consists of two stages, of learning the Tag2Vec representation, and of learning the video features to tag vector space embedding. Finally, given a query video, the learned embedding projects it to the tag space and nearby hash-tags are retrieved as suggestion. This process is outlined in Figure 4.1. In the following subsections, we explain

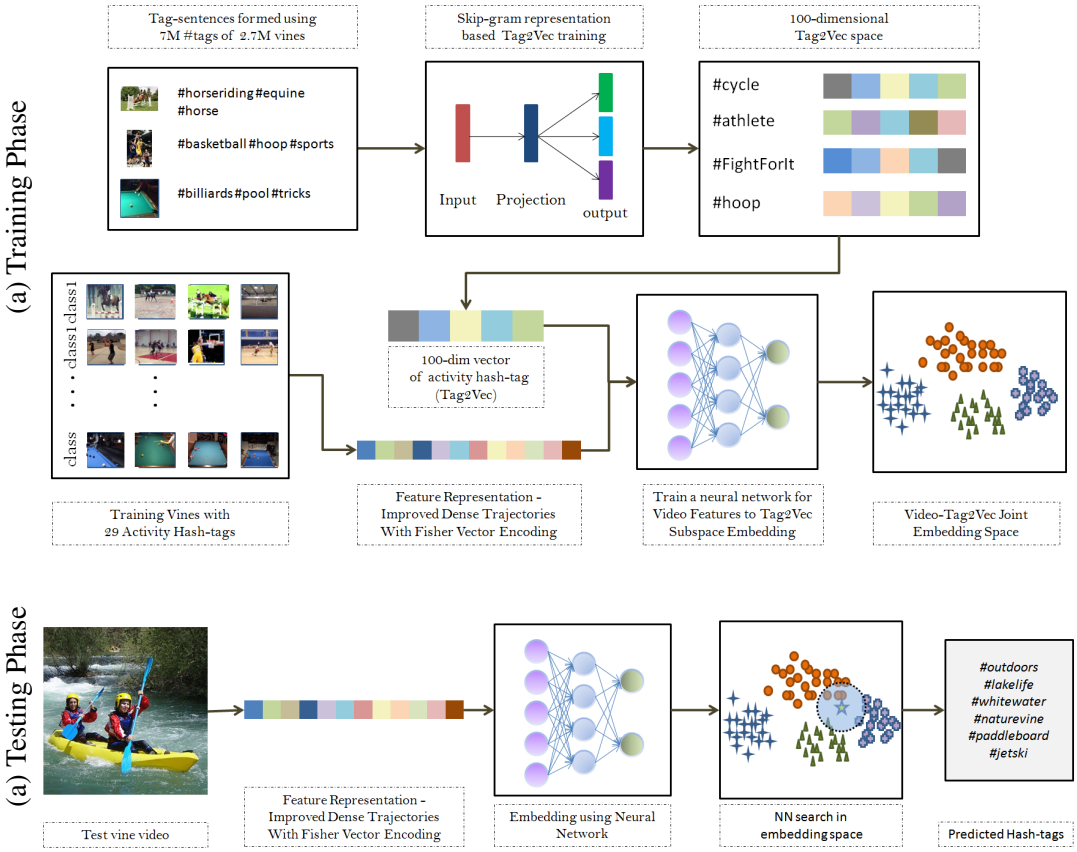


Figure 4.1: Pictorial representation of our tag recommendation system

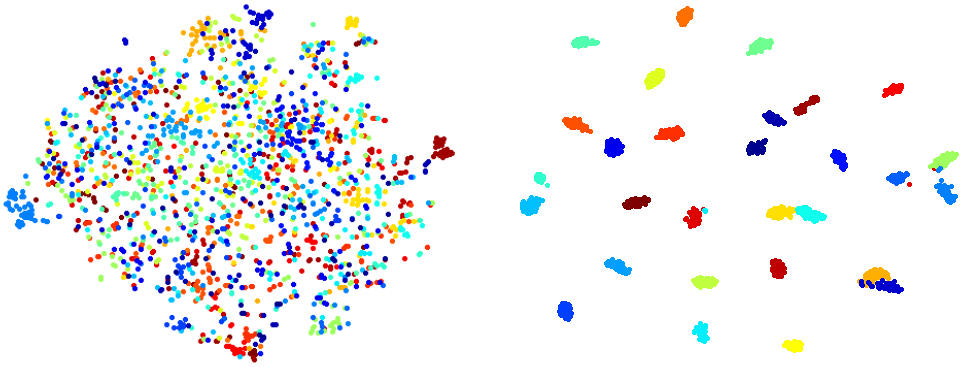
(i) the hash-tag data and learning of Tag2Vec space, (ii) the video data and learning of visual to Tag2Vec space embedding, and finally (iii) retrieval for tag recommendation.

4.1.1 Hash-tag Data and Pre-processing

To gather hash-tags data, we first use the 17,000 most common English words (as determined by n-gram frequency analysis of the Google's Trillion Word Corpus [1]) as queries and retrieve a total of about 2.7 million videos ($\sim$ 150 videos per query). We scrape the hash-tags corresponding to each retrieved video, remove all special characters, and perform *stemming* on the hash-tags.

Stemming is a popular technique in natural language processing (NLP) community for reducing words to a root form such that multiple inflections of a word reduce to the same root e.g. 'fish', 'fished', 'fishing', and 'fish-like' reduce to the stem 'fish'. The stemmed hash-tag words for each video form a *hash-tag sentence* leading to a total of 2.7 million sentences. These 2.7 million hash-tag sentences together form a text-corpus that we use to learn the Tag2Vec representation.

[1] https://github.com/first20hours/google-10000-english



(a) Before training the fisher vector space (b) After training the embedding space

Figure 4.2: The fisher vector and embedding spaces

4.1.2 Learning Tag2Vec Representation

[24] proposed efficient unsupervised neural network based methods to learn embeddings of semantic words in vector space using a large corpus of text data (web-based) consisting of 1.6 billion words. These methods either use continuous Bag of Words representation or skip-gram representation of text sequences to train a two-layer neural network. The resulting word embedding assigns every word in the corpus to a vector in a 300-dimensional space. This word embedding is known as Word2Vec and the final vector space is popularly referred to as Word2Vec space. The main advantage of this representation is that vector operations can be performed on words. This property is extremely useful, e.g. to compute similarity between words using vector space distances.

Similar to this, we also train a neural network to learn hash-tag embeddings in a vector space and call it *Tag2Vec*. Since, we work with much smaller data ($\sim$ 7 million unique tags and 2.7 million sentences), we use skip-gram representation which is more effective on small data and we also restrict the resulting space to be 100-dimensional. We use publicly available code[2] for learning the Tag2Vec embeddings. The training converges quickly ($\sim$ 10 minutes) as we have a relatively small corpus of hash-tags. The resulting vector space enables us to perform vector operations on hash-tags. Figure 4.4 shows a word cloud representation of 30 nearest neighbours in Tag2Vec space for query vectors corresponding to stemmed tag words, 'basketball', 'billiards', 'pushup', 'baseball', 'kayaking', and 'breast stroke'. The word sizes are inversely related to the distance from the respectively query words, the closer the words in vector space, the larger the font size. It can be observed that 'basketball' tag has many tags with high similarity whereas 'billiards' has fewer. It is also worth noting how contextual similarity is also well captured, for example 'kayaking' tag has strong neighbours such as 'state park' and 'summer adventure'.

To demonstrate that the learned Tag2Vec space is different from Word2Vec space, we list the top-10 near-neighbours for four action word queries in both spaces (see Table 4.1). It can be clearly seen

[2] `https://code.google.com/p/word2vec/`

Query Words	Vector Space	top-10 retrieval results									
Pushups	Word2Vec	jumping jacks	pushup	situps	calisthenics	abdominal crunches	pushups situps	burpees	pullups	ab crunches	squat thrusts
	Tag2Vec	workout	gym	burpees	gymflow	gymday	exercise	muscleup	calisthenics	superset	gymrat
Polevault	Word2Vec	Ivan Ukhov	Gulfiya Khanafeyeva	Yaroslav Rybakov	Tatyana Lysenko	Anna Chicherova	Andrey Silnov	champion Tatyana	Croatia Blanka Vlasic	Svetlana Feofanova	Olga Kuzenkova
	Tag2Vec	USATF	athlete	tracknation	trackandfield	discusthrow	tooathlete	highjump	maxvelocity	blockstart	longjump
Kayaking	Word2Vec	canoeing	Kayaking	kayak	paddling	sea kayaking	rafting	whitewater kayaking	kayaking canoeing	rafting kayaking	canoing
	Tag2Vec	statepark	lake	whitewater	paddleboard	outdoorsfinland	lagoon	lakelife	outdooraction	emeraldbay	boat
Basketball	Word2Vec	baskeball	volleyball	basketbal	basektball	hoops	soccer	softball	football	bas ketball	roundball
	Tag2Vec	dunk	ballislife	hoopmixtape	basketballvine	NBA	basketballneverstops	streetball	basketballhighlight	bball	dunkcity

Table 4.1: Comparision of Tag2Vec and Word2Vec spaces. Top-10 nearest neighbour results shown for four query words. It can be seen that the tag words retrieved from Tag2Vec space are more diverse and socially relevant.

that the tags retrieved using Tag2Vec space are more diverse and socially relevant. See particularly the results for 'Polevault' and 'Basketball' queries. We also measure similarity between pairs of tag vectors and see that the Tag2Vec space models social jargons well. For example, we noticed that tags like #lol and #laugh have high similarity, #lol is also has high similarity with #fail owing to users tagging funny videos showing people failing at doing something, #fight is closer to both #win and #fail. This shows that our tag2vec space is able to capture meaningful tag relationships and similarity.

4.1.3 Video Data and Feature Representation

Our video data consists of vine video clips (vines) spanning 29 categories. These categories are listed in Table 4.3. As mentioned before, vines are short but wild and complex amateur video clips with heavy camera motion, cuts, and edits. Hence, they have high intra-class variance and direct similarity based approaches don't work well. We obtain 3000 useful videos with hash-tags after removing duplicates. We split these into training and testing sets of 2740 and 260 videos respectively. For both sets of videos, we compute visual and motion features.

In particular, we compute Improved Dense Trajectory (IDT) [44] features which consist of HoG (histogram of oriented gradients), HoF (histogram of optical flow), and MBH (motion boundary histograms) features.

These low level features capture global scene, motion and rate of motion information respectively. We encode the low-level IDT features using 1003676 dimensional fisher vectors for better generalization [30, 31]. Fisher encoding relies on gaussian mixture model (GMM) computed over a large vocabulary of low-level features. We use UCF51 action recognition dataset [39] to compute generalized vocabulary for GMM estimation. For IDT extraction, we use the code[3] made available by the authors. For computing GMM parameters and fisher vectors, we use the VLFeat Computer Vision library [43].

[3] https://lear.inrialpes.fr/people/wang/improved_trajectories

	exercis	lazy	sleep	sad	laugh	fight	runnin	swimmin	win	fail	lol
exercis	-	0.744	0.738	0.713	0.754	0.753	0.708	0.831	0.739	0.755	0.727
lazy	0.744	-	0.908	0.753	0.814	0.740	0.802	0.733	0.750	0.781	0.875
sleep	0.738	0.908	-	0.793	0.809	0.744	0.700	0.702	0.681	0.748	0.859
sad	0.713	0.753	0.793	-	0.733	0.762	0.699	0.688	0.727	0.746	0.755
laugh	0.754	0.814	0.809	0.733	-	0.788	0.681	0.704	0.780	1.032	1.761
fight	0.753	0.740	0.744	0.762	0.788	-	0.656	0.777	0.856	0.832	0.779
runnin	0.708	0.802	0.700	0.699	0.681	0.656	-	0.678	0.719	0.680	0.689
swimmin	0.831	0.733	0.702	0.688	0.704	0.777	0.678	-	0.796	0.782	0.678
win	0.739	0.750	0.681	0.727	0.780	0.856	0.719	0.796	-	0.903	0.784
fail	0.755	0.781	0.748	0.746	1.032	0.832	0.680	0.782	0.903	-	1.111
lol	0.727	0.875	0.859	0.755	1.761	0.779	0.689	0.678	0.784	1.111	-

Table 4.2: Similarity between 11 (stemmed) tag pairs in Tag2Vec space. Similarity computed as inverse of L_2 distance. Note how 'laugh' has maximum similarity with 'lol'. Similarly 'fight' has high similarity with 'fail' and 'win'; 'lazy' with 'sleep'; 'swimmin' with 'exercis' etc. This illustrates the semantics captured by our Tag2Vec space.

Baseball	Basketball	Benchpress	Biking	Billiards
Boxing	Breaststroke	Diving	Drumming	Fencing
Golf	HighJump	Horseriding	HulaHoop	Juggling
Kayaking	Lunges	Nunchucks	Piano	PoleVault
Pushups	Yoyo	Salsa	Skateboarding	Skiing
Soccer	Swing	Tennis	Volleyball	

Table 4.3: 29 categories used for training

4.1.4 Learning Video to Hash-tag Embedding

In the first step, a 100-dimensional vector space, representative of the hash-tag distribution has been learned. Next we need to learn a mapping function that can project the 1003676 dimensional video features (fisher encoded IDTs) to the 100 dimensional tag vector space.

[38] proposed a method to learn a mapping from visual features (images) to word vectors (word2vec space) for detecting objects in a cross-modal zero-shot framework. We adopt this cross-modal learning approach to learn a mapping from fisher vectors to tag vectors. Similar to [38], we train a neural network with `(fisher vector, tag word)` pairs for each of the 2770 training videos to learn a non-linear embedding function from video features too Tag2Vec space. The tag word is the same as the category/class label of the training video. For learning this embedding function, we use the publicly

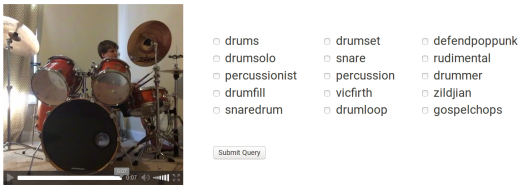


Figure 4.3: A screen shot of our user annotation (survey) platform.

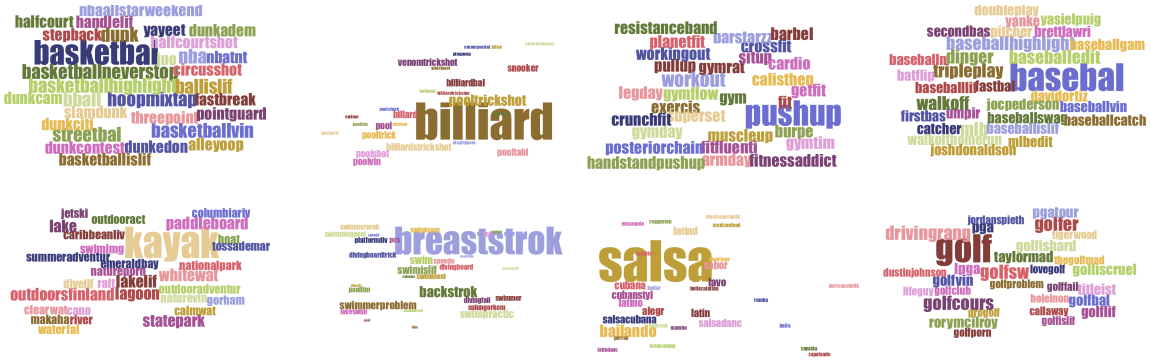


Figure 4.4: Illustration of effectiveness of Tag2Vec representation. The word clouds represent 29 nearest neighbours in Tag2Vec space for the following queries : 'basketball', 'billiards', 'pushup', 'baseball', 'kayaking', 'breast stroke', salsa, and golf. The word sizes are proportional to similarity (inversely proportional of L_2 distances), query word being the largest due to 100% self-similarity. Note that tag words are stemmed, not spelled incorrectly.

available code for zero-shot learning [4]. The neural network is set up to have 600 hidden nodes and maximum iterations are set to 1000 as we have more categories. Training this network with our data took approximately 2 hours.

Figure 4.2 shows the t-SNE (t-Stochastic Neighborhood Embedding) visualization of training features in fisher vector space and Tag2Vec space. It can be clearly seen that after embedding the the training vectors form distinct clusters around their category words. Once the embedding function is learned, a query videos (belonging to these 29 categories) can be directly mapped to the tag space and relevant hash-tags can be recommended. In the next section, we the hash-tag recommendation mechanism.

4.1.5 Tag Suggestion Metrics

Given a query video, we first compute its fisher vector representation. We then use the learned embedding function to project the query fisher vector in the learned Tag2Vec space. We utilize a simple nearest neighbour approach based on L_2 distance to retrieve potentially relevant hash-tags for a given query video. It can be seen that we do not directly compare the test/query vector to any of the training

[4] https://code.google.com/p/word2vec/

video vectors, neither in fisher vector space, not after the embedding. This is advantageous in terms of retrieval time and memory because, (i) there is no need to store the training set but only the Tag2Vec model and the fisher vectors to Tag2Vec space embedding function; and (ii) redundant comparisons are avoided. The tag words retrieved from the Tag2Vec space are stem words. Since we cannot suggest stems as hash-tags we need to convert a stemmed tag to its proper form. However, each stem corresponds to multiple tags, e.g. 'beauty', 'beautiful', 'beautifully' would all map to stem word 'beauti'. In our system, for a particular stemmed tag, we pick the most commonly used word in our hash-tag corpus from among all corresponding inflections of that stem. This de-stemming approach is a bit limiting as many variations of the same stem words would always be rejected. A better approach based on parts of speech (PoS) tagging and edit distance can replace this.

The time complexity of tag suggestion depends on the dimensionality of the tag space (d) and number of tags (N), $O(N * d)$. Our MATLAB implementation takes around 1 second for video-to-tag space embedding and less than a second for near-neighbour tag retrieval. For large-scale application, the simple nearest neighbour approach can be replaced by better retrieval mechanisms for efficiency and robustness.

4.2 Experiments

We conduct two sets of experiments, (i) to see the impact of embedding function if trained in batches or separately. (ii) To see how good this embedding function and Tag2Vec space performs on test vines.

4.2.1 Batch Embedding Learning

For this experiment we learn embedding function for 7 classes separately and together. The impact of learning is measured by comparing the average distance of the training vines from their respective class label for both the strategies. We find that learning in batch or separately doesn't have much impact on the learnt mapping. The ratio of batch distance to individual distance is close to 1.0 for all classes indicating not much difference in the learnt mapping. However, for computational efficiency we prefer the batch learning.

4.2.2 Tags Suggestion

We conduct this experiment to both qualitatively and quantitatively validate the tags suggested by our approach. A user study is conducted where in every task 15 recommended tags are presented along with their corresponding vines. The testing set contains 270 vines with approximately 9 vines per class. Figure 4.3 shows how each vine is shown to the user for selection of relevant tags from among those suggested by our system. Users, familiar with social networking jargon, are required to select the tags they consider can be used along with the vine. We perform the experiment with 14 users where every user marks each vine once. As vines vary drastically from one another we try to compute the average

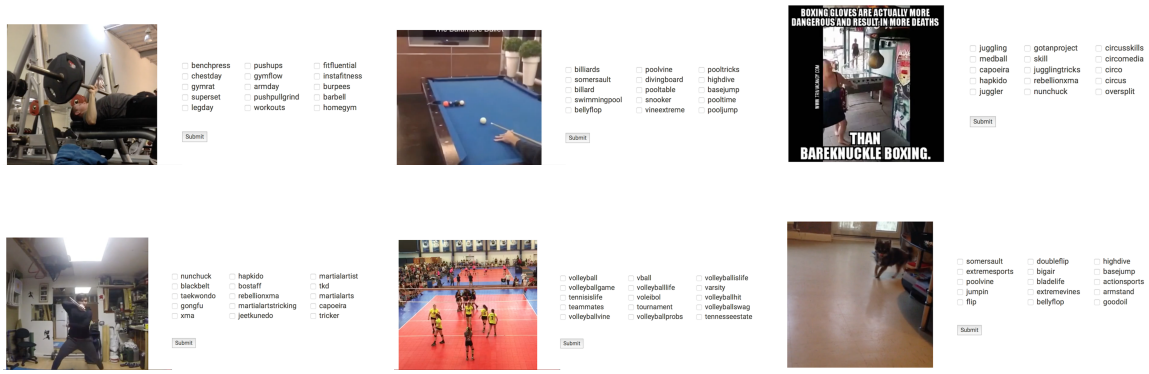


Figure 4.5: Hash tags suggested by our framework for the given video clip. First two columns show relevant hash-tag suggestions as predicted by our proposed model (like armday, benchpress, instafitness etc. for top left image). Last column shows two failure cases.

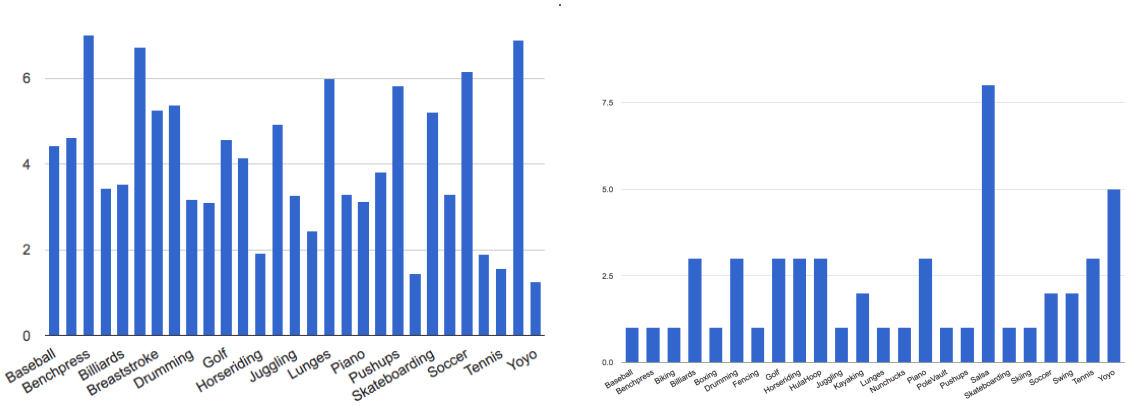


Figure 4.6: Left image shows the average number of relevant tags marked by the users for each class out of 15 suggest tags. Right image shows average number of videos across all users per class for which there were no relevant tags found out of entire test dataset of 50 videos per class

relevance score per class in order to contain much of this randomness and for better understanding of case where we might perform better or worse. As the vines are wild and unconstrained there will be cases where we are unable to suggest any relevant tags and to see if this happens in only specific classes or for some classes its more than the others we compute the number of files per class which didn't get even a single relevant tag.

The plot in Figure 4.6 (left) shows the average number of relevant tags suggested for each class. The plot in Figure 4.6 (right) shows the class-wise distribution of vines with no relevant hashtags. Benchpress contains the highest number of tags, 7, on an average followed by Volleyball at 6.88. Yoyo & Salsa are the worst performers with 1.25 & 1.45 tags respectively on an average. This can be attributed to multiple performers in the scene for Salsa and occluded performer for the Yoyo category, and as the presence of the performer is required for building the visual features and mapping, the overall accuracy of the system fails. In total 52 vines out of the 270 didn't contain a single relevant hashtag. Upon

viewing these vines, we noticed that these vines in majority were the ones which visually and textually contained no information pertaining to the action tag. For example, a person talking about how good boxing is might contain relevant hashtags as assigned by the uploader but as we don't process auditory modality and training of the embedding function relies only on visual features, the system is unavailable to correctly map such vines to the relevant concepts. Figure 4.5 shows some examples of success and failure cases for qualitative evaluation.

There were also cases where tags of different categories were confused, for example, #pool can be present with #swimmingpool and #biliiards, and as a result Tag2Vec will consist of these two classes quite close to each other resulting in mixing of hash-tags.

The overall number of relevant tags suggested for a vine is 4.03 out of 15 which is 27% of the tags suggested for each vine. Based on the hash-tag statistics collected by scraping the vine platform, we observed that 4.79 is the average number of hashtags associated with a typical vine (based on 2.5 million entries). One thing to note is that not all the tags in the ground-truth data are relevant hence this number is likely to come down. By suggesting 15 tags we are able to reproduce a similar number where an uploader finds 4 tags relevant to the vine which suggests that our system performs well even for such unconstrained videos.

4.3 Summary

This chapter presents a method to automatically suggest hash-tags for short social video clips. Hash-tags are nosy and have ambiguous semantics. We learn a vector space which we show is able to capture these semantics. We call this vector space Tag2Vec. Also for automatically annotating hash-tags for a given video we learn a neural network based embedding function. We work on a self gathered dataset of wild short video clips of 29 categories. The embedding function embeds any query video to our Tag2Vec space from which we propose hash-tags using a simple nearest neighbour based classification. We show that our Tag2Vec space has desired semantic structure and we are able to suggest relevant hash-tags for the query videos. In the future we would like to explore the semantics of the Tag2Vec space further by taking into account the type of web content with which tags are associated and then suggest hash-tags based on the current trending of the hash-tag. Also it would be interesting to explore the possibility of devising an evolving semantic space according to changing web trends and addition of new hash-tags.

Chapter 5

Conclusions

5.1 Conclusions

Social media aided the outburst in the number of videos freely available on the internet and it's this data which can leverage us in various computer applications if used correctly. Vines are short video clips uploaded by users which are upto 6 seconds in length and are present in abundance. The drawback of vines is that they are quite random and require significant manual labour to prune out the good examples to be put for use. In this thesis we focused on one such method which can potentially help us in filtering usable vines from the perspective of action recognition. For action recognition we discussed the shortcomings of videos currently present in the traditional datasets and how utilising vines can help us move towards achieving better performance in the wild.

Vines along with consisting visual content are composed of hash-tags. Hash-tags can be understood as keywords assigned by the video uploader capturing some context of the visual content. We saw how we can enhance the embedding learning used to classify actions to suggest hash-tags for vines. Such a system is modular and can be used as hash-tag recommendation for new vines.

To summarise, in this thesis we explored the problem of action classification and hash-tag recommendation and suggested appropriate approaches to solve them. The common theme to both the approaches was visual-semantic embedding space. For action classification of vines we proposed an iterative method which in every iteration uses highly confident vine samples to expand the learning of the action criteria. The embedding mapping is learnt from the visual space to the testitWord2Vec space. We showed how in every iteration the distance of samples decreased from the class label hence indicating of learning an improved mapping over time. In 4 we created an improved semantic space from millions of hash-tags which are an improvement over textitWord2Vec for relating vines to hash-tags. *Tag2Vec* inherits relations between hash-tags where the distance is inversely proportional to similarity between two hash-tags. The corresponding mapping then can be directly used for predicting tags which we evaluated through our user survey. This fairly simple and fast approach provides decent results over unconstrained and wild collection of vines.

These two applications are two of the many for which vines can be put to use and for these two we utilised the visual and semantic modalities. As we have audio, visual and semantic information with the vines one can utilise all these modalities and not just two to further improve the system performance. One can use our approach of classifying actions [35] as a pre-processing stage to gather abundance of filtered data from the internet and save valuable time of manually pruning out the unwanted vines. As the stopping criteria is currently heuristic, stronger methods which can act as an indicator can be worth looking into. It would also be interesting how a never ending learner built on top the Tag2Vec space performs over time, as it is bound to collect more information hence a richer semantic space. Currently, *Tag2Vec* mapping [36] once learnt can be used directly across platforms to suggest relevant tags to the user. This modularity if preserved can prove beneficial for video hosting sites to improve their retrieval, suggestions and recommendation systems.

———————————————————

——————————————————— Recommended 'Related Publication'

Chapter 6

Related Publications

1. Aditya Singh, Saurabh Saini, Rajvi Shah, and P J Narayanan. From traditional to modern: Domain adaptation for action classification in short social video clips. In *Proceedings of German Conference on Pattern Recognition (GCPR)*, pages 245-257, 2016.

2. Aditya Singh, Saurabh Saini, Rajvi Shah, and P J Narayanan. Learning to hash-tag videos with tag2vec. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 94:1-94:8, 2016

Bibliography

[1] Instagram statistics. http://expandedramblings.com/index.php/important-instagram-stats/. Accessed: 2016-12-25.

[2] Number of smartphone users worldwide from 2014 to 2020 (in billions). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide. Accessed: 2016-09-30.

[3] Vine statistics. http://expandedramblings.com/index.php/vine-statistics/. Accessed: 2016-12-25.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[5] Lamberto Ballan, Marco Bertini, Alberto Del Bimbo, Marco Meoni, and Giuseppe Serra. Tag suggestion and localization in user-generated videos based on social knowledge. In *Proceedings of Second ACM SIGMM Workshop on Social Media*, WSM '10, pages 3–8. ACM, 2010.

[6] Oren Barkan. Bayesian neural word embedding. *CoRR*, abs/1603.06571, 2016. URL http://arxiv.org/abs/1603.06571.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435.

[8] Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean embedding of co-occurrence data. In *Advances in Neural Information Processing Systems*, pages 497–504, 2005.

[9] Boqing Gong, Yuan Shi, Fei Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073, 2012.

[10] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[11] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, 2007.

[12] Nazl Ikizler and Pnar Duygulu. Histogram of oriented rectangles: A new pose descriptor for human action recognition. *Image and Vision Computing*, 27(10):1515 – 1526, 2009.

[13] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3192–3199, Sydney, Australia, 2013.

[14] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision (ECCV)*, 2012.

[15] Alexander Klser, Marcin Marszaek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *In BMVC08*.

[16] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[17] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[18] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Computer Vision and Pattern Recognition , 2011 IEEE Conference on*, pages 1785–1792, 2011.

[19] V. Lavrenko, S. L. Feng, and R. Manmatha. Statistical models for automatic video annotation and retrieval. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 3, pages iii–1044–7 vol.3, 2004.

[20] Rémi Lebret and Ronan Collobert. Word embeddings through hellinger pca. In *14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

[21] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. 2014.

[22] Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3650–3656, 2015.

[23] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1041–1048. Curran Associates, Inc., 2009.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[25] E. Moxley, T. Mei, and B. S. Manjunath. Video annotation through search and graph reinforcement mining. *IEEE Transactions on Multimedia*, 12(3):184–193, 2010.

[26] Milind R Naphade, Igor Kozintsev, and Thomas S Huang. Probabilistic semantic video indexing. In *NIPS*, pages 967–973, 2000.

[27] JuanCarlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[28] Paul Over, George Awad, Martial Michel, Jonathan Fiscus, Wessel Kraaij, Alan F. Smeaton, Georges Quenot, and Roeland Ordelman. Trecvid 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2015*.

[29] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[30] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[31] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 143–156, 2010.

[32] J. Ponce, T.L. Berg, M. Everingham, D.A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B.C. Russell, A. Torralba, C.K.I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In *Toward Category-Level Object Recognition*, volume 4170, pages 29–48. 2006.

[33] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 17–26. ACM, 2007.

[34] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th International Conference on Multimedia*, pages 357–360, 2007.

[35] Aditya Singh, Saurabh Saini, Rajvi Shah, and P J Narayanan. From traditional to modern: Domain adaptation for action classification in short social video clips. In *Proceedings of German Conference on Pattern Recognition (GCPR)*, pages 245–257, 2016.

[36] Aditya Singh, Saurabh Saini, Rajvi Shah, and P J Narayanan. Learning to hash-tag videos with tag2vec. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 94:1–94:8, 2016.

[37] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 935–943. 2013.

[38] Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D. Manning, and Andrew Y. Ng. Zero-shot learning through cross-modal transfer. *CoRR*, abs/1301.3666, 2013.

[39] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[40] W. Sultani and I. Saleemi. Human action recognition across datasets by foreground-weighted histogram decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 764–771, 2014.

[41] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] A. Torralba and A.A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528, 2011.

[43] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1469–1472, 2010.

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

[45] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, pages 1–20, 2015.

[46] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition , 2012 IEEE Conference on*, pages 1290–1297, 2012.

[47] M. Wang, X. S. Hua, J. Tang, and R. Hong. Beyond distance measurement: Constructing neighborhood similarity for video annotation. *IEEE Transactions on Multimedia*, 11(3):465–476, 2009.

[48] Zheshen Wang, Ming Zhao, Yang Song, Sanjiv Kumar, and Baoxin Li. Youtubecat: Learning to categorize wild web videos. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pages 879–886, 2010.

[49] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2), 2006.

[50] X. Xu, T. Hospedales, and S. Gong. Semantic embedding space for zero-shot action recognition. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 63–67, Sept 2015. doi: 10.1109/ICIP.2015.7350760.

[51] Jun Yang, Rong Yan, and Alexander G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th International Conference on Multimedia*, pages 188–197, 2007.

[52] Ting Yao, Tao Mei, Chong-Wah Ngo, and Shipeng Li. Annotation for free: Video tagging by mining user search behavior. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 977–986. ACM, 2013.

[53] W. L. Zhao, X. Wu, and C. W. Ngo. On the annotation of web videos by efficient near-duplicate search. *IEEE Transactions on Multimedia*, 12(5):448–461, 2010.