8-29-2016

Representations and Models for Large-Scale Video Understanding

Du Le Hong Tran
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/dissertations

Part of the Computer Sciences Commons

Recommended Citation
Tran, Du Le Hong, "Representations and Models for Large-Scale Video Understanding" (2016). Dartmouth College Ph.D Dissertations. 53.
https://digitalcommons.dartmouth.edu/dissertations/53

This Thesis (Ph.D.) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Ph.D Dissertations by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.
REPRESENTATIONS AND MODELS FOR LARGE-SCALE VIDEO UNDERSTANDING

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in
Computer Science
by
Du Le Hong Tran
DARTMOUTH COLLEGE
Hanover, New Hampshire
August 29, 2016
Examining Committee:

Lorenzo Torresani, Chair

Chris Bailey-Kellogg

Qiang Liu

Antonio Torralba

F. Jon Kull, Ph.D.
Dean of Graduate and Advanced Studies
Abstract

In this thesis, we investigate different representations and models for large-scale video understanding. These methods include a mid-level representation for action recognition, a deep-learned representation for video analysis, a generic convolutional network architecture for video voxel prediction, and a new high-level task and benchmark of video comprehension.

First, we present EXMOVES, a mid-level representation for scalable action recognition. The entries in EXMOVES representation are the calibrated outputs of a set of movement classifiers over spatial-temporal volumes of the input video. Each movement classifier is a simple exemplar-SVM trained on low-level features. Our EXMOVES requires a minimal amount of supervision while also obtaining good action recognition accuracy. It is approximately 70 times faster than other mid-level video representations.

Second, we propose an effective method for spatiotemporal feature learning using deep 3-dimensional convolutional networks (3D ConvNets) trained on a large-scale video dataset. We show that 3D ConvNets are more suitable for spatiotemporal feature learning compared to 2D ConvNets. Our learned features, C3D, with a simple linear classifier outperform state-of-the-art methods on four different benchmarks and are comparable with current best methods on the other two benchmarks. The features are also very compact, efficient to compute, and easy to use. Third, we develop a generic
3D ConvNet architecture for video voxel prediction. Our preliminary results show that our architecture can be applied for different voxel prediction problems with good results. Finally, we propose a new task, namely Video Comprehension, construct a large-scale benchmark, and develop a set of fundamental baselines as well as conduct a human study on the newly-proposed benchmark.
Acknowledgement

This thesis could not have been completed without support from my advisor, family, and friends. I take this opportunity to appreciate them.

First, I gratefully thank my advisor, Professor Lorenzo Torresani, for giving me valuable guidance, continued support, and positive motivation. I definitely could not complete this work without his support. During the last four years, he has always given me good advice on research directions. He is also very open to my new ideas (which are sometime silly or too ambitious). I am very grateful to him for keeping me motivated when my papers were rejected. To me, he is not only a great advisor, but also a good friend. I have learned a lot from him, not only about research, but also about many other things in life.

I am also thankful to my mentor at Facebook, Manohar Paluri, for continuously giving me support and trust. My thanks are also to Professor Chris Bailey-Kellogg, Professor Qiang Liu, and Professor Antonio Torralba for being on my thesis committee, taking the time to read my thesis, and providing me valuable feedback. I would also thank my labmates at the VLG group at Dartmouth College and my colleagues at Facebook AI Research for helpful research discussions. I am grateful to Dartmouth College, of which I soon become an alumnus, for providing me a good fellowship and beautiful green campus on which to relax on the stressful days, and to Facebook for...
giving me internship opportunities and access to research resources. Lastly, I would like to thank my wife, Vy Pham, and my son, Vi Du, for their unconditional love and unwavering trust and support. I apologize them for not being a good husband and a responsible dad during the deadline time. Their understanding and love helped me to stay strong and overcome many research challenges.
Preface

All of the work presented in this thesis was conducted in the Visual Learning Group (VLG) at Dartmouth College and in the Facebook AI Research (FAIR) group in Menlo Park, CA. A version of Chapter 2 has been published as a conference paper at ICLR14 [104], later extended into a journal article, and published at IJCV16 [105]. A version of Chapter 3 has been published as a conference paper at ICCV15 [100]. A part of Chapter 4 has been published as a workshop paper at CVPR16 Workshop on Deep Learning in Computer Vision [101]. Finally, the work in Chapter 5 is a working paper and a pre-print is available on arXiv [102]. I am the first author of all of the aforementioned papers.
Contents

Abstract ................................................................. iii
Acknowledgement ....................................................... v
Preface ................................................................. vii

1 Introduction .......................................................... 1

2 EXMOVES: a Mid-Level Representation for Scalable Action Analysis .......................................................... 6

2.1 Introduction ........................................................ 7

2.2 Related Work ....................................................... 11

2.2.1 Low-level feature-based approaches ......................... 11

2.2.2 Mid-level feature-based approaches ......................... 12

2.2.3 Top-level action modeling approaches ...................... 13

2.3 Approach Overview ............................................. 13

2.4 Exemplar-Movement SVMs (EXMOVES) ....................... 15

2.4.1 Low-level features used in EXMOVES ...................... 16

2.4.2 Learning EXMOVES ........................................... 17
# 4 Deep End-to-End Voxel-to-Voxel Prediction

- Deep End-to-End Voxel-to-Voxel Prediction ........................................ 74
- 4.1 Introduction .................................................................................... 75
- 4.2 Related Work .................................................................................. 78
- 4.3 Video Voxel Prediction .................................................................... 79
- 4.4 Application I: Video Semantic Segmentation ................................. 82
- 4.5 Application II: Optical Flow Estimation ........................................ 86
- 4.6 Application III: Video Coloring ..................................................... 90
- 4.7 Conclusions .................................................................................... 92

# 5 ViCom: Benchmark and Methods for Video Comprehension

- ViCom: Benchmark and Methods for Video Comprehension ............. 94
- 5.1 Introduction .................................................................................... 95
- 5.2 Related Work .................................................................................. 99
- 5.3 Video Comprehension .................................................................... 100
  - 5.3.1 Problem statement .................................................................... 100
  - 5.3.2 Procedure for Constructing a Video Comprehension Dataset .... 101
  - 5.3.3 The ViCom Dataset .................................................................. 104
- 5.4 Approaches to Video Comprehension .......................................... 107
  - 5.4.1 Regression ................................................................................. 108
  - 5.4.2 Metric learning .......................................................................... 108
  - 5.4.3 Video captioning ....................................................................... 109
- 5.5 Experiments ................................................................................... 110
  - 5.5.1 Experimental setup .................................................................... 110
  - 5.5.2 Experimental results ................................................................. 113
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison of recognition accuracies on four datasets. The classification model is an efficient linear SVM applied to 4 distinct global mid-level descriptors: Action Bank [84], Discriminative Patches [38], Histogram of Space-Time Visual Words (BOW) and our EXMOVES. We consider two different low-level features to build BOW and EXMOVES: HOG-HOF-STIPs and Dense Trajectories. Our EXMOVES achieve the best recognition accuracy on all four datasets using Dense Trajectories, and greatly outperform the BOW descriptor for both our choices of low-level features, HOG-HOF-STIPs and Dense Trajectories.</td>
</tr>
<tr>
<td>2.2</td>
<td>Recognition accuracies of our EXMOVES (applied to Dense Trajectories) compared with those of Action Bank and Discriminative Patches using the same subset of 13 action classes from UCF50 considered in [38].</td>
</tr>
<tr>
<td>2.3</td>
<td>Recognition accuracy on the individual classes of HMDB51 using linear SVMs trained on EXMOVES based on Dense Trajectories. Note that random chance would yield a recognition rate of 1.96%.</td>
</tr>
</tbody>
</table>
2.4 Statistics of time needed to extract the mid-level descriptors Action Bank and EXMOVES. The time needed to extract EXMOVES features for the entire UT-I dataset using a single CPU is only 14 hours; instead, it would take more than 41 days to compute Action Bank descriptors for this dataset.

2.5 Effects of multiple scales and spatio-temporal pyramid levels on EXMOVES. The recognition accuracy on UCF50 and UCF101-part2 are reported for different numbers of scales and levels of spatio-temporal pyramid. The EXMOVE dimensionalities for the different settings are shown in brackets.

2.6 The effects of bounding box annotations on EXMOVES. WS-EXMOVES are trained without manual bounding-box annotations, while EXMOVES are learned using a single manually-annotated bounding box in each video. The action recognition accuracy of WS-EXMOVES is 1-6% lower than that of EXMOVES on the HMDB51, Hollywood-2, UCF50, and UCF101 datasets.

2.7 Action similarity labeling results. Comparisons between EXMOVES and current state-of-the-art features [53] using a binary linear SVM trained on the absolute difference vector, i.e., \(|x_1 - x_2|\) where \(x_1, x_2\) here denote the feature vectors extracted from the two input videos. The numbers are accuracies and area under ROC curve (in parenthesis). EXMOVES outperform other single feature vectors by 3-4%, and combined descriptors by 1%.
3.1 C3D compared to best published results. C3D outperforms all previous best reported methods on a range of benchmarks. On UCF101, we report accuracy for two groups of methods. The first set of methods use only RGB frame inputs while the second set of methods (in parentheses) use all possible features (e.g. optical flow, improved Dense Trajectory).

3.2 Sports-1M classification result. C3D outperforms [46] by 5% on top-5 video-level accuracy and achieves state-of-the-art result on this dataset.

3.3 Action recognition results on UCF101. C3D compared with baselines and current state-of-the-art methods. Top: baseline results; Middle: methods taking only RGB frames as inputs; Bottom: methods using multiple feature combinations.

3.4 Action similarity labeling result on ASLAN. C3D significantly outperforms state-of-the-art method [118] by 9.6% in accuracy and by 11.1% in area under ROC curve.

3.5 Scene recognition accuracy. C3D using a simple linear SVM outperforms state-of-the-art methods on Maryland and YUPENN.

3.6 Runtime analysis on UCF101. C3D is 91x faster than improved dense trajectories [114] and 274x faster than Brox’s optical flow methods, thus relatively more than two orders of magnitude faster than [90].

3.7 Number of parameters and training time comparison of 3D ConvNets with different input resolutions. Note that net-128 is equivalent to the depth-3 net in section 3.3.
4.1 **Semantic segmentation accuracy on GATECH.** V2V consistently outperforms all baselines showing the good benefits of using V2V with 3D convolution/deconvolution compared to 2D convolution/deconvolution or bilinear upsampling.  

4.2 **Runtime comparison.** The first row reports the total runtime (including I/O) to extract optical flow using V2V-Flow and Brox’s method [8] for the entire UCF101 test split 1. V2V-Flow is 70x faster than Brox’s method, besides being slightly more accurate (see Table 4.3).  

4.3 **Optical flow results on Sintel.** V2V denotes our network learned from the UCF101 optical flow computed with Brox’s method. The finetuned-V2V network is obtained by fine-tuning V2V on Sintel (test accuracy is measured in this case using 3-fold cross validation). Both versions of our network perform slightly better than Brox’s algorithm and they allow computation of optical flow with a runtime speedup of 20 times compared to Brox’s software.  

5.1 **Dataset comparison.** Comparison of different video datasets in terms of size, task, and video type. Abbreviations for tasks: cls (classification), det (detection), des (description), qa (question and answer), and vicom (video comprehension). N/A: information is not available in [46, 95].
5.2 **Video comprehension accuracy on ViCom.** Accuracy of different approaches on ViCom compared with random chance and human performance. Methods based on deep metric learning give the best performance (the best gives 55.5%). Random chance is at 20%, while humans achieve an accuracy of 78.8%. . . . . . . . . . . . . . . . . . 114

5.3 **Video comprehension accuracy details.** Performance by topic of different methods compared with human performance. Humans achieve the highest accuracy across all topics except for ‘Technology’ where DML-C3D-FT outperforms humans. LR-C3D performs below random chance on 2 out of 10 topics, and S2VT-Vi has 7 out of 10 topic are below random chance (in underlined text). . . . . . . . . . . . . . . . 115

5.4 **ViCom topics.** Topic modeling of ViCom sentences using LDA [6] with 10 topics. The topic names are picked by authors based on the sharing semantic among of the top words. . . . . . . . . . . . . . . . 118

5.5 **Experiments with Skip-thought Vectors.** Replacing word2vec by skip-thought vectors for the language representation consistently degrades the performance across all different visual embedding on shallow metric learning networks. . . . . . . . . . . . . . . . 119
List of Figures

2.1 Overview of our approach. During an offline stage, a collection of exemplar-movement SVMs (EXMOVES) is learned. Each EXMOVE is trained using a single positive video exemplar and a large number of negative sequences. These classifiers are then used as mid-level feature extractors to produce a semantically-rich representation of videos. 14

2.2 Accuracy on HMDB51 as a function of the number of EXMOVES. We use Recursive Feature Elimination to reduce the number of EXMOVES. The accuracy remains near the state-of-the-art even when using only 100 exemplars. 31
2.3 **Action retrieval on UCF50.** The left column shows the query videos. Next to each query we show the 15-nearest neighbors retrieved using EXMOVES. The incorrectly-retrieved examples (i.e., videos belonging to a class different from that of the query) are marked with a red X. The first two query are from classes that are easy to recognize, the third query is randomly chosen from a class that ranks roughly in the middle in terms of recognition accuracy. The last query is randomly chosen from the most difficult class to recognize (according to the confusion matrix). Best viewed in color.

2.4 **Cross-dataset action retrieval.** The first column shows the query videos from HMDB51. To the right of each query video we show the 5-nearest neighbors retrieved from UCF50. The retrieved examples that have incorrect label are marked with a red X. Best viewed in color.

2.5 **Action similarity labeling.** Visualizations of action similarity pairs. Each row represent a test input pair. The binary classifier using EXMOVES correctly classifies the pairs in the first two rows (true positives). The third row is a false negative, and the last row is a false positive. Note that although the two video clips in the second row have largely different scales and viewpoints, our method is able to correctly label them as containing the same action. Our method fails to label the third pair as “different” because of the different viewpoints, and the fourth pair as “same” because the two videos exhibit similar motions. Best view in color.
3.1 **2D and 3D convolution operations.** a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal.

3.2 **3D ConvNet architecture search.** Action recognition clip accuracy on UCF101 test split-1 of different 3D ConvNet architectures. 2D ConvNet performs worst and 3D ConvNet with $3 \times 3 \times 3$ kernels performs best among the experimented nets.

3.3 **C3D architecture.** C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from pool1 to pool5. All pooling kernels are $2 \times 2 \times 2$, except for pool1 is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.

3.4 **Visualization of C3D model, using the method from [123].** Interestingly, C3D captures appearance for the first few frames but thereafter only attends to salient motion. Best viewed on a color screen.

3.5 **C3D compared with Imagenet and iDT in low dimensions.** C3D, Imagenet, and iDT accuracy on UCF101 using PCA dimensionality reduction. C3D outperforms Imagenet and iDT by 10-20% in low dimensions.
3.6 Feature embedding. Feature embedding visualizations of Imagenet and C3D on UCF101 dataset using t-SNE [107]. C3D features are semantically separable compared to Imagenet suggesting that it is a better feature for videos. Each clip is visualized as a point and clips belonging to the same action have the same color. Best viewed in color.

3.7 Action similarity labeling result. ROC curve of C3D evaluated on ASLAN. C3D achieves 86.5% on AUC and outperforms current state-of-the-art by 11.1%.

3.8 3D ConvNets with different input resolutions. Action recognition clip accuracy on UCF101 test split-1 of 3D ConvNets with different input resolutions.

3.9 Deconvolutions of C3D conv2a feature maps. Each group is a C3D conv2a learned feature map. First two rows: the learned filters detect moving edges and blobs. The last row: the learned filters detect shot changes, edge orientation changes, and color changes. Best viewed in a color screen.

3.11 Deconvolutions of a C3D conv5b learned feature map which detects moving motions of circular objects. In the second last clip, it detects a moving head while in the last clip, it detects the moving hair-curler. Best viewed in a color screen.

3.12 Deconvolutions of a C3D conv5b learned feature map which detects biking-like motions. Note that the last two clips have no biking but their motion patterns are similar to biking motions. Best viewed in a color screen.

3.13 Deconvolutions of a C3D conv5b learned feature map which detects face-related motions: applying eye-makeup, applying lipstick, and brushing tooth. Best viewed in a color screen.

3.14 Deconvolutions of a C3D conv5b learned feature map which detects balance-beam-like motions. In the last clip, it detects hammering which shares similar motion patterns with balance beam. Best viewed in a color screen.

3.15 Deconvolutions of C3D conv5b learned feature maps compared with optical flows. Optical flows fire at all of moving pixels while C3D just pays attention to only salient motions. Best viewed in a color screen.

4.1 **Voxel to voxel prediction:** is a fine-grained video understanding task where the algorithm need to infer a variable for each input voxel. The problem has many potential applications including video semantic segmentation, optical flow prediction, depth estimation, and video coloring.
4.2 **Deconvolutional layers in ConvNets.** a) Visualization of the deconvolutional layer used in [123] where the filter weights are set to be equal to those of the pre-trained convolutional layer. b) Trainable deconvolutional layers [69] learn upsampling. 81

4.3 **V2V Architecture for Voxel Prediction.** The lower part (below dashed line) consists of layers from C3D [100]. Connected to these layers we have three 3D convolution layers: Conv3c, Conv4c, Conv-pre use filters of size $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$. Both Deconv5 and Deconv4 are deconvolutional layers employing kernels of size $4 \times 4 \times 4$ with output stride of $2 \times 2 \times 2$. Deconv3 has kernel size $8 \times 4 \times 4$ and output stride of $4 \times 2 \times 2$. The numbers inside the boxes represent the number of learning filters in that layer, while the numbers near the boxes (above or below) represent the size of output signals produced by that layer. The part inside the thick-dashed box is application-dependent. 83

4.4 **Video Semantic Segmentation Results on GATECH.** The soft-max prediction heat maps produced by V2V for different classes together with input frames. The last two classes are omitted due to their small populations. Best viewed in color. 84
4.5 V2V (top row) compared with upsampling baselines (rows 2-4). V2V consistently outperforms all bi-linear upsampling baselines. Conv5b-Up provides fairly accurate prediction, but over-smoothed due to the high upsampling factor. Conversely, Conv3b-Up yields finer predictions, but more noisy because it uses less deep features. V2V gives by far the best tradeoff as it has access to deep features and it learns the upsampling filters.

4.6 Optical flow estimation on UCF101. The output of V2V is qualitatively compared with Brox’s optical flow for 6 sample clips from the UCF101 test split. For each example we show (from left to right): an input frame, V2V’s predicted optical flow, and Brox’s motion. Note that Brox’s method is used to generate semi-truth data for training V2V. We see that on test videos V2V is able to predict flow of similar quality as that produced by Brox’s algorithm. Best viewed in color.

4.7 Visualizations of optical flow computed by the V2V network (trained on UCF101 without finetuning) for a few sample Sintel clips. For each example we show: input frame, V2V’s predicted optical flow, Brox’s flow, and ground truth. Best viewed in color.
4.8 Visualization of Conv1a filters learned by C3D (top) and V2V (bottom).

Note that C3D is trained to recognize actions (on Sport1M), while V2V is optimized to estimate optical flow (on UCF101). Each set shows the 64 learned filters at the Conv1a layer. Three consecutive square images on each row represent one filter (as kernel size is $3 \times 3 \times 3$). Each square image is upscaled to $30 \times 30$ pixels for better visualization. Best viewed in color. GIF animation of these filters are provided in the supplementary material.

4.9 Examples of video coloring with V2V on the test set of UCF101.

For each example we show (from left to right): a gray-scale input frame, the output frame colored by V2V, and the ground truth color frame. The V2V model is able to predict “common sense” colors such as the color of human skin, sky, woody furniture, river, sea, and mountain. Best viewed in color.

5.1 Video comprehension example. Video comprehension requires choosing one of $k$ possible English sentences as the description for an input video clip. In this example the correct answer is (D).

5.2 Relevant clip detection. The ROC curve of relevant clip detector evaluated on the validation set. At the false positive rate of 0.1, the true positive rate is 0.83.

5.3 ViCom topic distribution. The subject distribution was estimated by training an LDA model with 10 topics on closed caption sentences of ViCom clips.
5.4 **S2VT predictions.** The same clip example used in the introduction with sentences generated by S2VT-MS and S2VT-Vi, and the CC ground truth. S2VT-MS predicts simple sentences. S2VT-Vi is well adapted to news language and predicts longer sentences. Both methods incorrectly predict answer (C) in the test of Figure 1. The correct answer key is (D). ................................................. 110

5.5 **Examples from ViCom.** Some example clips from the ViCom dataset with their true closed caption sentences. It can be noted that they cover a wide range of subjects ranging from environmental events, to science, politics, and accidents. .................................................. 117

5.6 **ViCom topics on different embedding.** Semantic embedding of word2vec, C3D (fc6), and DML-C3D-FT(fc6) on a random subset of 10,000 ViCom test clips. Each dot is a clip representation (language or visual) vector projected into 2 dimension and is labeled (colored) by the topic. .......................................................... 119

5.7 **S2VT predictions.** More prediction samples from S2VT. The sentences generated by S2VT-MS and S2VT-Vi, and the CC ground truth. S2VT-MS predicts simple sentences. S2VT-Vi is well adapted to news language and predicts longer sentences. ................................................. 120
Chapter 1

Introduction

Due to the fast-growing number of Internet applications, there is a large number of videos being shared every minute. For example, 300 hours of video are uploaded to YouTube every minute. This amount of video (not counting other video sharing sites such as Instagram, Facebook, and many others) goes beyond the human capacity to analyze and understand the underlining trends. It is essential for us to have a computer algorithm that can make sense of this data in order to build meaningful and useful applications.

Video understanding is one of the core problems in computer vision which has been studied for decades. However, most recent methods are not well-designed for large-scale applications. Thus, applying the current algorithms to applications where we need to analyze millions of videos daily is difficult.

Let us consider what factors prevent video classification methods from being practical for large-scale applications. The main obstacles of the current methods are their complicated representations and models. Complicated representations are normally

\footnotetext{1YouTube statistics as of December 2014}
hand-designed features [113], interest point detectors [62, 19], or expensive template matching [84]. Expensive classification methods include non-linear kernels [112, 57] and high dimension representations such as the Fisher Vector [77]. We consider these factors carefully when we develop our EXMOVES. In EXMOVES, both the representation and the classification model are designed to be linear, thus EXMOVES is very efficient at run-time. EXMOVES is also scalable because it requires only a minimum number of annotations (e.g., one bounding box annotation per an exemplar).

In recent years, deep learning has shown its great success in many domains and applications [56, 69, 33, 124, 125]. This success is mainly due to the powerful expressiveness of deep networks (highly non-linear with many feature layers) which can be efficiently trained on large-scale datasets (e.g., using back-propagation) by more powerful parallel machines. While deep learning shows good progress in the image domain, there are only a few contributions in regard to videos [46, 90]. These approaches, however, are not well-suited for motion modeling as they use 2D ConvNets which are designed for appearance-based representations. We propose to use 3D ConvNets to learn spatiotemporal features and show that our learned features, C3D, significantly outperform methods that use 2D ConvNet as well as hand-designed features. In addition, our learned features are also fast at run-time, compact to store, and simple to use.

Apart from accuracy and scalability, video understanding methods also need to be able to provide high-quality predictions. This property is an indication of the extent to which and in how much detail the algorithms understand the videos. Current methods mainly tackle the classification problem in which they provide very simple predictions: a class label per one input video [60, 76, 43, 84, 74]. On the other hand, some video
captioning (or video description) approaches [34, 109] can provide high semantic-level video predictions such as a sentence describing the input video. This type of prediction is also lacking in detail. We propose a generic deep ConvNet architecture for video voxel prediction. Our method can provide much more detailed information about videos, such as one prediction per input voxel. We show that our approach is useful for various video analysis applications.

Finally, as an attempt to make a long-term impact on video understanding, we propose a new challenging task called Video Comprehension, along with a large-scale benchmark for this task. In video comprehension, computers take a video clip as well as a multiple-choice question (a set of $K$ English sentences) as an input and then predict which English sentence best describes the content of the input clip. Although this task shares some similarities with other video understanding tasks such as video description [109] and visual question-and-answer (VQA) [1], it has some important differences, making it a more well-posed problem and easier to quantitatively evaluate. In fact, video description is an ill-posed problem and challenging to quantitatively evaluate because of many possible correct descriptions and linguistic ambiguity. For example, given an input video, there are many ways to describe it, including indentifying the weather, scenes, subjects, actions, or even the feeling of the subject appearing in the video. This makes it very difficult to quantitatively compare the effectiveness of different methods on this task. On the other hand, VQA is a less ill-posed problem compared to video captioning. However, multiple true answers are still possible, such as in terms of different linguistic ways to express an answer. In addition, scaling up the dataset for VQA is difficult because of its complicated and expensive annotations of questions and answers.
Introduction

We believe that video comprehension is more well-posed and easier to quantitatively evaluate than visual captioning and VQA. We propose a semi-automatic framework for constructing a large-scale benchmark for video comprehension. The Imagenet Challenge [82] taught us that if we can define a good and well-posed task, provide a large-scale and unbiased benchmark, and motivate the research community to solve it together, good research impacts are possible when the problem is solved (or partly solved). It potentially leads to a better understanding about the problem as well as related problems. When Imagenet classification challenge is (partly) solved, the other related problems (e.g., detection, segmentation, optical flow) are also improved.

In summary, the main contributions of this thesis are the following:

- A mid-level representation for scalable video classification that is mainly designed for large-scale applications. We show that on a budget annotation cost, limited training data, and moderate machine power, scalable video classification is feasible.

- A novel deep representation for videos with desired properties for large-scale applications. We show that with large-scale training data, we can learn good spatiotemporal features that can run at super-realtime while being highly accurate and compact.

- A novel approach for video voxel prediction that is generic enough to be applied for different voxel labeling tasks. We show that video voxel prediction is feasible provided that voxel ground truth annotations are available.

- A new challenging video understanding task, Video Comprehension, and a large-scale benchmark for this task. The new task is more well-posed and easier for
Introduction

quantitative evaluations. We also conduct a varied set of baselines as well as a human study to gain a better understanding of the newly-proposed task.
Chapter 2

EXMOVES: a Mid-Level Representation for Scalable Action Analysis

Abstract

In this chapter we present EXMOVES – learned exemplar-based features for efficient recognition and analysis of actions in videos. The entries in our descriptor are produced by evaluating a set of movement classifiers over spatial-temporal volumes of the input video sequences. Each movement classifier is a simple exemplar-SVM trained on low-level features, i.e., an SVM learned using a single annotated positive space-time volume and a large number of unannotated videos.

Our representation offers several advantages. First, since our mid-level features are learned from individual video exemplars, they require minimal amount of supervision. Second, we show that simple linear classification models trained on our global video
descriptor yield action recognition accuracy approaching the state-of-the-art but at orders of magnitude lower cost, since at test-time no sliding window is necessary and linear models are efficient to train and test. This enables scalable action recognition, i.e., efficient classification of a large number of actions even in massive video databases. Third, we show the generality of our approach by training our mid-level descriptors from different low-level features and testing them on two distinct video analysis tasks: human activity recognition as well as action similarity labeling. Experiments on large-scale benchmarks demonstrate the accuracy and efficiency of our proposed method on both these tasks.

2.1 Introduction

Human action recognition and matching are important but still largely-unsolved computer vision problems motivated by many useful applications, including content-based video retrieval, automatic surveillance, and human-computer interaction. The difficulty of the task stems from the large intra-class variations in terms of subject and scene appearance, motion, viewing positions, as well as action duration.

We argue that most of the existing action recognition methods are not designed to handle such heterogeneity. Typically, these approaches are evaluated only on simple datasets involving a small number of action classes and videos recorded in lab-controlled environments \cite{5,108}. Furthermore, in the design of the action recognizer very little consideration is usually given to the computational cost which, as a result, is often very high.

We believe that modern applications of action recognition demand scalable systems that can operate efficiently on large databases of unconstrained image sequences,
2.1 Introduction

such as YouTube videos. For this purpose, we identify three key-requirements to address: 1) the action recognition system must be able to handle the substantial variations of motion and appearance exhibited by realistic videos; 2) the training of each action classifier must have low-computational complexity and require little human intervention in order to be able to learn models for a large number of human actions; 3) the testing of the action classifier must be efficient so as to enable recognition in large repositories, such as video-sharing websites.

This work addresses these requirements by proposing a global video descriptor that yields state-of-the-art action recognition accuracy even with simple linear classification models. The feature entries of our descriptor are obtained by evaluating a set of movement classifiers over the video. Each of these classifiers is an exemplar-SVM \cite{71} trained on low-level features \cite{61,113} and optimized to separate a single positive video exemplar from an army of “background” negative videos. Because only one annotated video is needed to train an exemplar-SVM, our features can be learned with very little human supervision. The intuition behind our proposed descriptor is that it provides a semantically-rich description of a video by measuring the presence/absence of movements similar to those in the exemplars. Thus, a linear classifier trained on this representation will express a new action-class as a linear combination of the exemplar movements (which we abbreviate as EXMOVES). We demonstrate that these simple linear classification models produce surprisingly good results on challenging action datasets. In addition to yielding high-accuracy, these linear models are obviously very efficient to train and test, thus enabling scalable action recognition, i.e., efficient recognition of many actions in large databases.

Our approach can be viewed as extending to videos the idea of classifier-based
2.1 Introduction

image descriptors [111, 99, 67, 16] which describe a photo in terms of its relation to a set of predefined object classes. To represent videos, instead of using object classes, we adopt a set of movement exemplars. In the domain of action recognition, our approach is most closely related to the work of Sadanand and Corso [84], who have been the first to describe videos in terms of a set of actions, which they call the Action Bank. The individual features in Action Bank are computed by convolving the video with a set of predefined action templates. This representation achieves high accuracy on several benchmarks. However, the template-matching step to extract these mid-level features is very computationally expensive. As reported in [84], extracting mid-level features from a single video of UCF50 [92] takes a minimum of 0.4 hours up to a maximum of 34 hours. This computational bottleneck effectively limits the number of basis templates that can be used for the representation and constrains the applicability of the approach to small datasets.

Our first contribution is to replace this prohibitively expensive procedure with a technique that is almost two orders of magnitude faster. This makes our descriptor applicable to action recognition in large video databases, where the Action Bank framework is simply too costly to be used. The second advantage of our approach is that our mid-level representation can be built on top of any arbitrary spatial-temporal low-level features, such as appearance-based descriptors computed at interest points or over temporal trajectories. This allows us to leverage the recent advances in design of low-level features: for example, we show that when we use dense trajectories [113] as low-level features, a simple linear classifier trained on the HMDB51 dataset using our mid-level representation yields a 41.6% relative improvement in accuracy over the Action Bank built from the same set of video exemplars. Furthermore, we demonstrate
that our representation is general in the sense that it can be applied to different low-level features and it can be used for several video analysis tasks, such as action recognition and action similarity labeling. Finally, the experiments reported in this chapter show that a linear classifier applied to our mid-level representation produces consistently much higher accuracy than the same linear model directly trained on the low-level features used by our descriptor.

Our EXMOVES are also related to Discriminative Patches [38], which are spatial-temporal volumes selected from a large collection of random video patches by optimizing a discriminative criterion. The selected patches are then used as a mid-level vocabulary for action recognition. Our approach differs from this prior work in several ways. As discussed in 2.4.4, each EXMOVE feature can be computed from simple summations over individual voxels. This model enables the use of Integral Videos [47], which reduce dramatically the time needed to extract our features. Discriminative Patches cannot take advantage of the Integral Video speedup and thus they are much more computationally expensive to compute. This prevents their application in large-scale scenarios. On the other hand, Discriminative Patches offer the advantage that they are automatically mined, without any human intervention. EXMOVES require some amount of human supervision, although minimal (just one hand-selected volume per exemplar). In practice such annotations are inexpensive to obtain. In our experiments we show that EXMOVES learned from only 188 volumes greatly outperform Discriminative Patches using 4000 volumes.
2.2 Related Work

Human action recognition and analysis have a long history in the computer vision literature. The previous approaches can be roughly classified into low-level feature-based, mid-level feature-based, and top-level action modeling approaches.

2.2.1 Low-level feature-based approaches

Low-level feature-based approaches represent videos by low-level feature primitives. These features can be either sparsely or densely sampled from the videos. Spatio-temporal interest points can also be applied for sparse features. Efros et al. used optical flows to represent and classify actions [22]. Laptev and Lindeberg extended the Harris corner detector to 3D in order to detect spatio-temporal interest points (STIPs) [62, 61]. Dollár et al. used a 1D Gabor filter and a 2D Gaussian smoothing kernel to detect Cuboids for behavior recognition [19]. The Cuboids interest point detector is denser compared to STIPs and allows the users to adjust the desired level of sparsity. Gorelick et al. proposed Space-Time Shapes for modeling actions [5] by solving a Poisson equation. Derpanis et al. used 3D Gabor filters to extract “Space-time orientation” for action recognition [18]. Motivated by the success of image-based features such as HOG [14] and SIFT [70], HOG3D [86] and SIFT3D [50] were also proposed for modeling video features. Ke et al. used boosting to learn volumetric features for event detection [48]. Quoc et al. demonstrated that spatio-temporal features can be learned under unsupervised setting using stacked ISA with strong performance [65]. Recently, Wang et al. proposed Dense Trajectories [116] and its improved version, namely improved Dense Trajectories [113] which is widely considered the current state-of-the-art in video features for human action recognition, achieving
2.2 Related Work

the top performance on various benchmarks.

2.2.2 Mid-level feature-based approaches

Mid-level feature-based approaches represent videos using a set of mid-level features, which are usually classifiers trained on low-level representations. Fathi and Mori used Adaboost to train a set of mid-level weak classifiers for human action recognition [26] using optical flow as low-level features. Similarly, Ke et al. used a Boosting method to learn volumetric features for action detection [47], but directly on raw video voxels. Along the line of visual attributes for static images [30, 59, 25], Liu et al. proposed to represent human actions by data-driven attributes and used them for action recognition [68]. Inspired by the success of ObjectBank [67], Sadanand and Corso proposed to represent videos as a set of video templates called Action-Bank [84]. Despite its promising discriminative power, the high computational cost prevents this method to be applicable in large-scale scenarios. Jian et al. used Discriminative Patches to represent videos for action classification [38]. The main benefit of this method is being trained in an unsupervised manner. However, due to the unsupervised nature of the training, the method does need to have a large number of mid-level features in order to attain a reasonable discriminative capacity (see results in our experimental section). Our EXMOVES are closely related to Action Bank and Discriminative Patches as these are all forms of mid-level representation. Compared to Action Bank, our mid-level classifiers are linear SVMs while Action Bank builds on template matching, which is much more computationally expensive. On the other hand, while Discriminative Patches are trained without supervision, our EXMOVES are weakly supervised. In our experiments we show that EXMOVES
achieve better discriminative power while requiring a minimal amount of annotation, i.e., one annotated bounding box per mid-level classifier.

2.3 Approach Overview

We explain the approach at a high level using the schematic illustration in Figure 2.1. During an offline stage, our method learns $N_a$ exemplar-movement SVMs (EXMOVES), shown on the left side of the figure. Each EXMOVE is a binary classifier optimized...
2.3 Approach Overview

**Figure 2.1:** Overview of our approach. During an offline stage, a collection of exemplar-movement SVMs (EXMOVES) is learned. Each EXMOVE is trained using a single positive video exemplar and a large number of negative sequences. These classifiers are then used as mid-level feature extractors to produce a semantically-rich representation of videos.

to recognize a specific action exemplar (e.g., an instance of “biking”) and it uses histograms of quantized space-time low-level features for the classification. Note that in order to capture different forms of each activity, we use multiple exemplars per activity (e.g., multiple instances of “biking”), each contributing a separate EXMOVE. The set of learned EXMOVES are then used as mid-level feature extractors to produce an intermediate representation for any new input video: we evaluate each EXMOVE on subvolumes of the input video in order to compute the probability of the action at different space-time positions in the sequence. Specifically, we slide the subvolume of each EXMOVE exemplar at $N_s$ different scales over the input video. As discussed in section 2.4.4, this evaluation can be performed efficiently by using Integral Videos [17]. Finally, for each EXMOVE, we perform max-pooling of the classifier scores within $N_p$
2.4 Exemplar-Movement SVMs (EXMOVES)

spatial-temporal pyramid volumes. Thus, for any input video this procedure produces a feature vector with \( N_a \times N_s \times N_p \) dimensions. Because the EXMOVE features provide a semantically-rich representation of the video, even simple linear classification models trained on our descriptor achieve good action categorization accuracy.

2.4 Exemplar-Movement SVMs (EXMOVES)

Our EXMOVE classifiers are linear SVMs applied to histograms of quantized space-time low-level features calculated from subvolumes of the video. In section 2.4.1 we describe the two space-time low-level descriptors used in our experiments, but any quantize-able appearance or motion features can be employed in our approach.

In principle, to train each SVM classifier we need a reasonable number of both positive and negative examples so as to produce good generalization. Unfortunately, we do not have many positive examples due to the high human cost of annotating videos. Thus, we resort to training each SVM using only one positive example, by extending to videos the exemplar-SVM model first introduced by Malisiewicz et al. for the case of still images [71]. Specifically, for each positive exemplar, we manually specify a space-time volume enclosing the action of interest and excluding the irrelevant portions of the video. The histogram of quantized low-level space-time features contained in this volume becomes the representation used to describe the positive exemplar. Then, our objective is to learn a linear SVM that separates the positive exemplar from the histograms computed from all possible subvolumes of the same size in negative videos.

It may appear that training a movement classifier from a single example will lead to severe overfitting. However, as already noted in [71], exemplar-SVMs actually have good generalization as their decision boundary is tightly constrained by the millions of
2.4 Exemplar-Movement SVMs (EXMOVES)

negative examples that the classifier must distinguish from the positive one. In a sense, the classifier is given access to an incredible amount of training examples to learn what the positive class is not. Furthermore, we use the exemplar-SVMs simply as mid-level feature extractors to find movements similar to the positive exemplar. Thus, their individual categorization accuracy is secondary. In other words, rather than applying the individual exemplar-SVMs as action recognizers, we use them collectively as building blocks to define our action categorization model, in a role similar to the weak-learners of boosting techniques [110].

2.4.1 Low-level features used in EXMOVES

Although any arbitrary low-level description of space-time points or trajectories can be used in our framework, here we experiment with the two following representations:

- **HOG-HOF-STIPs.** Given the input video, we first extract spatial-temporal interest points (STIPs) [61]. At each STIP we compute a Histogram of Oriented Gradients (HOG) and a Histogram of Flows (HOF) [15] using the implementation in [63]. We concatenate the HOG and the HOF descriptor to form a 162-dimensional vector representing the STIP. Finally, we run \( k \)-means on these vectors to learn a codebook of \( D = 5,000 \) cluster centroids. Given the codebook, any space-time volume in a video is represented in terms of the histogram of codewords occurring within that volume. We normalize the final histogram using the L1 norm.

- **Dense Trajectories.** These are the low-level motion and appearance descriptors obtained from dense trajectories according to the algorithm described in [113]. The trajectories are computed for non-stationary points using a median-filtered
2.4 Exemplar-Movement SVMs (EXMOVES)

optical flow method and are truncated every 15 frames. Each trajectory is then described in terms of its shape (point coordinate features, 30 dimensions), appearance (HOG features, 96 dimensions), optical flow (HOF features, 108 dimensions) and boundary motion (MBHx and MBHy features, 96 dimensions each). As in [113], we learn a separate dictionary for each of these 5 descriptors. We use a codebook of \( d = 5,000 \) cluster centroids for each descriptor. Thus, each space-time volume in a video is then represented as a vector of \( D = 25,000 \) dimensions obtained by concatenating the 5 histograms of trajectories occurring within that volume. We L1-normalize the final histogram.

2.4.2 Learning EXMOVES

The input for learning an EXMOVE consists of a positive video \( V^+ \) containing a manually-annotated space-time 3D box bounding the action of interest \( x_E \), and thousands of negative videos \( V^-_{1..N} \) without action volume annotations. The only requirement on the negative videos is that they must represent action classes different from the category of the positive exemplar (e.g., if the exemplar contains the action dancing, we exclude dancing videos from the negative set). But this constraint can be simply enforced given action class labels for the videos, without the need to know the space-time volumes of these negative actions. For example, tagged Internet videos (e.g., YouTube sequences) could be used as negative videos, by choosing action tags different from the activity of the positive exemplar.

It is worth noting that different movement exemplars will have different 3D box shapes. For example, we expect a walking action to require a tall volume while swimming may have a volume more horizontally elongated. As further discussed
below, we maintain the original shape-ratio of the exemplar volume in both training and testing. This means that we look for only tall volumes when detecting walking, and short-and-wide volumes when searching for the swimming action.

Let \( x_E \) be the manually-specified volume in the positive sequence \( V^+ \).

Let us denote with \( \phi(x) \) the L1-normalized histogram of codewords (computed from either HOG-HOF-STIPs or Dense Trajectories) within a video volume \( x \), i.e.,

\[
\phi(x) = \frac{1}{c(x)} [c_1(x), \ldots, c_D(x)]^T,
\]

where \( c_i(x) \) is the number of codeword \( i \) occurring in volume \( x \), and \( c(x) \) is the total number of codewords in \( x \). Note that in the case of Dense Trajectories, each trajectory contributes 5 codewords into the histogram since it is quantized according to the 5 separate dictionaries.

Adopting the exemplar-SVM method in [71], our exemplar-SVM training procedure learns a linear classifier \( f(x) = w^T \phi(x) + b \), by minimizing the following objective function:

\[
\min_{w,b} \|w\|^2 + C_1 \sum_{x \in V^+ \text{ s.t. } \frac{|x \cap x_E|}{|x_E|} \geq 0.5} h(w^T \phi(x) + b) + C_2 \sum_{i=1}^N \sum_{x \in V_i^-} h(-w^T \phi(x) - b) 
\]

where \( h(s) = \max(0, 1 - s) \) is the hinge loss, while \( C_1 \) and \( C_2 \) are pre-defined parameters that we set so as to equalize the unbalanced proportion of positive and negative examples. Note that the first summation in the objective involves subvolumes whose spatial overlap with \( x_E \) is greater than 50% and thus are expected to yield a positive score, while the second summation is over all negative subvolumes. Unfortunately, direct minimization of the objective in Eq.2.1 is not feasible since it requires optimizing
2.4 Exemplar-Movement SVMs (EXMOVES)

the SVM parameters on a gigantic number of subvolumes. Thus, we resort to an alternation scheme similar to that used in [71] and [29]: we iterate between 1) learning the parameters ($w, b$) given an active set $S$ of negative volumes and 2) mining new negative volumes with the current SVM parameters.

We first initialize the parameters of the classifier by traditional SVM training using the manually-selected volume $x_E$ as positive example and a randomly selected subvolumes from each of the other videos as negative example. At each iteration the current SVM is evaluated exhaustively on every negative video to find violating subvolumes, i.e., subvolumes yielding an SVM score below exceeding $-1$. These subvolumes are added as negative examples to the active set $S$ to be used in the successive iterations of SVM learning. Furthermore, our training procedure adds as positive examples the subvolumes of $V^+$ that have spatial overlap with $x_E$ greater than 50% and SVM score below 1. We stop the iterative alternation between these two steps when either no new subvolumes are added to the active set or a maximum number of iterations $M$ is reached. In our implementation we use $M = 10$, but we find that in more than 85% of the cases, the learning procedure converges before reaching this maximum number of iterations.

The pseudocode of our learning procedure is given in Algorithm 1. Lines 1 – 3 initialize the active set. The function `svm_training` in line 5 learns a traditional binary linear SVM using the labeled examples in the active set. Note that we found that at each iteration we typically have millions of subvolumes violating the constraints (lines 7-11). In order to maintain the learning of the SVM feasible, in practice we add to the active set only the volumes that yield the largest violations in each video, for a maximum of $k^- = 3$ per negative video and $k^+ = 10$ for the positive video.
2.4 Exemplar-Movement SVMs (EXMOVES)

Algorithm 1 EXMOVE training

Require: A set of negative videos \( \{\mathcal{V}_1^-, \ldots, \mathcal{V}_N^-\} \) and a manually-selected volume \( \mathbf{x}_E \) in exemplar video \( \mathcal{V}^+ \).

Ensure: Parameters \((w, b)\) of exemplar-SVM.

\[
S \leftarrow \{(\mathbf{x}_E, +1)\}
\]

for \( i = 1 \) to \( N \) do

\[
S \leftarrow S \cup \{(\mathbf{x}_i, -1)\} \text{ with } \mathbf{x}_i \text{ randomly chosen from } \mathcal{V}_i^-
\]

end for

for \( \text{iter} = 1 \) to \( M \) do

\[
(w, b) \leftarrow \text{svm\_training}(S) \quad S_{old} \leftarrow S
\]

for all \( \mathbf{x} \) in \( \mathcal{V}^+ \) s.t. \( w^T \mathbf{x} + b < 1 \) & \( \frac{|\mathbf{x}^T \mathbf{x}_E|}{|\mathbf{x}_E|} > 0.5 \) do

\[
S \leftarrow S \cup \{(\mathbf{x}, +1)\} \text{ //false negative}
\]

end for

for \( i = 1 \) to \( N \) do

for all \( \mathbf{x} \) in \( \mathcal{V}_i^- \) s.t. \( w^T \mathbf{x} + b > -1 \) do

\[
S \leftarrow S \cup \{(\mathbf{x}, -1)\} \text{ //false positive}
\]

end for

end for

if \( S_{old} = S \) then

break

end if

end for

2.4.3 Calibrating the ensemble of EXMOVES

The learning procedure described above is applied to each positive exemplar independently to produce a collection of EXMOVES. However, because the exemplar classifiers are trained dis-jointly, their score ranges and distributions may vary considerably. A standard solution to this problem is to calibrate the outputs by learning for each classifier a function that converts the raw SVM score into a proper posterior probability compatible across different classes. To achieve this goal we use the procedure proposed by Platt in [78]: for each exemplar-SVM \((w_E, b_E)\) we learn parameters \((\alpha_E, \beta_E)\) to produce calibrated probabilities through the sigmoid function
2.4 Exemplar-Movement SVMs (EXMOVES)

\[ g(x; w_E, b_E, \alpha_E, \beta_E) = \frac{1}{1 + \exp(\alpha_E(w_E^T x + b_E) + \beta_E)} \]. The fitting of parameters \((\alpha_E, \beta_E)\) is performed according to the iterative optimization described in \cite{78} using as labeled examples the positive/negative volumes that are in the active set at the completion of the EXMOVE training procedure. As already noted in \cite{71}, we also found that this calibration procedure yields a significant improvement in accuracy since it makes the range of scores more homogeneous and diminishes the effect of outlier values.

2.4.4 Efficient computation of EXMOVE scores

Although replacing the template matching procedure of Action Bank with linear SVMs applied to histograms of space-time features yields a good computational saving, this by itself is still not fast enough to be used in large-scale datasets due to the exhaustive sliding volume scheme. In fact, the sliding volume scheme is used in both training and testing. In training, we need to slide the current SVM over negative videos to find volumes violating the classification constraint. In testing, we need to slide the entire set of EXMOVE classifiers over the input video in order to extract the mid-level features for the subsequent recognition. Below, we describe a solution to speed up the sliding volume evaluation of the SVMs.

Let \(\mathcal{V}\) be an input video of size \(R \times C \times T\). Given an EXMOVE with parameters \((w_E, b_E)\), we need to efficiently evaluate it over all subvolumes of \(\mathcal{V}\) having size equal to the positive exemplar subvolume \(x_E\) (in practice, we slide the subvolume at \(N_s\) different scales but for simplicity we illustrate the procedure assuming we use the original scale).

It is worth noting that the branch-and-bound method of Lampert \textit{et al.} \cite{58} cannot be applied to our problem because it can only find the subwindow maximizing the
2.4 Exemplar-Movement SVMs (EXMOVES)

classification score while we need the scores of all subvolumes; moreover it requires unnormalized histograms.

Instead, we use integral videos [47] to efficiently compute the EXMOVE score for each subvolume. An integral video is a volumetric data-structure having size equal to the input sequence (in this case $R \times C \times T$). It is useful to speed up the computation of functions defined over subvolumes and expressed as cumulative sums over voxels, i.e., functions of the form $H(\mathbf{x}) = \sum_{(r,c,t) \in \mathbf{x}} h(r,c,t)$, where $(r,c,t)$ denotes a space-time point in volume $\mathbf{x}$ and $h$ is a function over individual space-time voxels. The integral video for $h$ at point $(r,c,t)$ is simply an accumulation buffer $B$ storing the sum of $h$ over all voxels at locations less than or equal to $(r,c,t)$, i.e., $B(r,c,t) = \sum_{r' \leq r} \sum_{c' \leq c} \sum_{t' \leq t} h(r',c',t')$. This buffer can be built with complexity linear in the video size. Once built, it can be used to compute $H(\mathbf{x})$ for any subvolume $\mathbf{x}$ via a handful of additions/subtractions of the values in $B$.

In our case, the use of integral video is enabled by the fact that the classifier score can be expressed in terms of cumulative sums of individual point contributions, as we illustrate next. For simplicity we describe the procedure assuming that $\phi(\mathbf{x})$ consists of a single histogram (as is the case for HOG-HOF-STIPs) but the method is straightforward to adapt for the scenario where $\phi(\mathbf{x})$ is the concatenation of multiple histograms (e.g., the 5 histograms of Dense Trajectories). Let us indicate with $P(\mathbf{x})$ the set of quantized low-level features (either STIPs or Dense Trajectories) included in subvolume $\mathbf{x}$ of video $\mathcal{V}$ and let $i_p$ be the codeword index of a point $p \in P(\mathbf{x})$. Then we can rewrite the classification score of exemplar-SVM $(\mathbf{w}, b)$ on a subvolume $\mathbf{x}$ as follows (we omit the constant bias term $b$ for brevity):
2.5 Implementation Details

\[ w^T \phi(x) = \frac{1}{c(x)} \sum_{i=1}^{D} w_i c_i(x) = \frac{\sum_{p \in P(x)} w_p}{\sum_{p \in P(x)} 1}. \]  

Equation 2.2 shows that the classifier score is expressed as a ratio where both the numerator and the denominator are computed as sums over individual voxels. Thus, the classifier score for any \( x \) can be efficiently calculated using two integral videos (one for the numerator, one for the denominator), without ever explicitly computing the histogram \( \phi(x) \) or the inner product between \( w \) and \( \phi(x) \). In the case where \( \phi(x) \) contains the concatenation of multiple histograms, then we would need an integral video for each of the histograms (thus 5 for Dense Trajectories), in addition to the common integral video for the denominator.

2.5 Implementation Details

Training data for EXMOVES. Since our approach shares many similarities with Action Bank, we adopt training and design settings similar to those used in [84] so as to facilitate the comparison between these two methods. Specifically, our EXMOVES are learned from the same set of UCF50 [92] videos used to build the Action Bank templates. This set consists of 188 sequences spanning a total of 50 actions. Since the Action Bank volume annotations are not publicly available, we manually selected the action volume \( x_E \) on each of these exemplar sequences to obtain \( N_a = 188 \) exemplars. As negative set of videos we use the remaining 6492 sequences in the UCF50 dataset: for these videos no manual labeling of the action volume is available nor it is needed by our method. Action Bank also includes 6 templates taken from other sources but
these videos have not been made publicly available; it also uses 10 templates taken from the KTH dataset. However, as the KTH videos are lower-resolution and contain much simpler actions compared to those in UCF50, we have not used them to build our EXMOVES. In the experiments we show that, while our descriptor is defined by a smaller number of movement classifiers (188 instead of 205), the recognition performance obtained with our mid-level features is consistently on par with or better than Action Bank.

**Parameters of EXMOVE features.** In order to compute the EXMOVE features from a new video, we perform max-pooling of the EXMOVE scores using a space-time pyramid based on the same settings as those of Action Bank, i.e., $N_s = 3$ scaled versions of the exemplar volume $x_E$ (the scales are 1, 0.75, 0.5), and $N_p = 73$ space-time volumes obtained by recursive octree subdivision of the entire video using 3 levels (this yields 1 volume at level 1, 8 subvolumes at level 2, 64 subvolumes at level 3). Thus, the final dimensionality of our EXMOVE descriptor is $N_a \times N_s \times N_p = 41,172$.

## 2.6 Experiments

### 2.6.1 Action Recognition

**Action classification model.** All our action recognition experiments are performed by training a one-vs-the-rest linear SVM on the EXMOVES extracted from a set of training videos. We opted for this classifier as it is very efficient to train and test, and thus it is a suitable choice for the scenario of large-scale action recognition that
2.6 Experiments

we are interested in addressing. The hyperparameter $C$ of the SVM is tuned via cross-validation for all baselines, Action Bank, and our EXMOVES.

**Test datasets.** We test our approach on the following large-scale action recognition datasets:

(a) HMDB51 [57]: It consists of 6849 image sequences collected from movies as well as YouTube and Google videos. They represent 51 action categories. The results for this dataset are presented using 3-fold cross validation on the 3 publicly available training/testing splits, each containing 3580 training videos and 1540 test examples.

(b) Hollywood-2 [72]: This dataset includes over 20 hours of video, subdivided in 3669 sequences, spanning 12 action classes. We use the publicly available split of training and testing examples.

(c) UCF50: This dataset contains 6676 videos taken from YouTube for a total of 50 action categories. This dataset was used in [84] and [38] to train and evaluate Action Bank and Discriminative Patches.

(d) UCF101 [92] (part 2): UCF101 is a superset of UCF50. For this test we only use videos from action classes 51 to 101 (from now on denoted as part 2), thus omitting the above-mentioned classes and videos of UCF50. This leaves a total of 6851 videos and 51 action classes. We report the accuracy of 25-fold cross validation using the publicly available training/testing splits.

**Comparison of recognition accuracies.** We now present the classification performance obtained with our features on the four benchmarks described above. We
2.6 Experiments

<table>
<thead>
<tr>
<th>Low-level features</th>
<th>Mid-level descriptor</th>
<th>Descriptor dimensionality</th>
<th>HMDB51</th>
<th>Hollywood-2</th>
<th>UCF50</th>
<th>UCF101 (part 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Gaussians</td>
<td>Action Bank</td>
<td>44,895</td>
<td>26.9</td>
<td>n/a</td>
<td>57.9</td>
<td>n/a</td>
</tr>
<tr>
<td>HOG3D</td>
<td>Discriminative Patches</td>
<td>9,360</td>
<td>n/a</td>
<td>n/a</td>
<td>61.2</td>
<td>n/a</td>
</tr>
<tr>
<td>HOG-HOF-STIPs</td>
<td>BOW</td>
<td>5,000</td>
<td>20.0</td>
<td>32.6</td>
<td>52.8</td>
<td>49.1</td>
</tr>
<tr>
<td></td>
<td>EXMOVES</td>
<td>41,172</td>
<td>27.7</td>
<td>44.7</td>
<td>63.4</td>
<td>57.2</td>
</tr>
<tr>
<td>Dense Trajectories</td>
<td>BOW</td>
<td>25,000</td>
<td>34.4</td>
<td>43.7</td>
<td>81.8</td>
<td>60.9</td>
</tr>
<tr>
<td></td>
<td>EXMOVES</td>
<td>41,172</td>
<td>41.9</td>
<td>56.6</td>
<td>82.8</td>
<td>71.6</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of recognition accuracies on four datasets. The classification model is an efficient linear SVM applied to 4 distinct global mid-level descriptors: Action Bank [84], Discriminative Patches [38], Histogram of Space-Time Visual Words (BOW) and our EXMOVES. We consider two different low-level features to build BOW and EXMOVES: HOG-HOF-STIPs and Dense Trajectories. Our EXMOVES achieve the best recognition accuracy on all four datasets using Dense Trajectories, and greatly outperform the BOW descriptor for both our choices of low-level features, HOG-HOF-STIPs and Dense Trajectories.

consider in our comparison three other mid-level video descriptors that can be used for action recognition with linear SVMs: Action Bank [84], Discriminative Patches [38] as well as histograms of visual words (BOW) built for the two types of low-level features that we use in EXMOVES, i.e., HOG-HOF-STIPs and Dense Trajectories. As in [113], we use a dictionary of 25,000 visual words for Dense Trajectories and 5,000 visual words for HOG-HOF-STIPs. Due to the high computational complexity of the extraction of Action Bank features, however, we were unable to test this descriptor on the large-scale datasets of Hollywood-2 and UCF101. For Discriminative Patches, we can only report accuracy on UCF50 since this is the only large-scale dataset on which they were tested in [38] and no software to compute these features is available.

The accuracies achieved by the different descriptors are summarized in Table 2.1. From these results we see that our EXMOVE descriptor built from Dense Trajectories yields consistently the best results across all four datasets. Furthermore, EXMOVES gives always higher accuracy than BOW built from the same low-level features, for both
2.6 Experiments

<table>
<thead>
<tr>
<th>Action Class</th>
<th>Action Bank</th>
<th>Discriminative Patches</th>
<th>EXMOVES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketball</td>
<td>53.84</td>
<td>50.00</td>
<td>56.93</td>
</tr>
<tr>
<td>Clean and Jerk</td>
<td>85.00</td>
<td><strong>95.65</strong></td>
<td>91.07</td>
</tr>
<tr>
<td>Diving</td>
<td>78.79</td>
<td>61.29</td>
<td><strong>96.08</strong></td>
</tr>
<tr>
<td>Golf Swing</td>
<td><strong>90.32</strong></td>
<td>75.86</td>
<td>90.14</td>
</tr>
<tr>
<td>High Jump</td>
<td>38.46</td>
<td>55.56</td>
<td><strong>81.30</strong></td>
</tr>
<tr>
<td>Javeline Throw</td>
<td>45.83</td>
<td>50.00</td>
<td><strong>73.50</strong></td>
</tr>
<tr>
<td>Mixing</td>
<td>42.85</td>
<td>55.56</td>
<td><strong>97.16</strong></td>
</tr>
<tr>
<td>PoleVault</td>
<td>60.60</td>
<td>84.37</td>
<td><strong>94.38</strong></td>
</tr>
<tr>
<td>Pull Up</td>
<td>91.67</td>
<td>75.00</td>
<td><strong>96.00</strong></td>
</tr>
<tr>
<td>Push Ups</td>
<td>85.00</td>
<td>86.36</td>
<td><strong>91.18</strong></td>
</tr>
<tr>
<td>Tennis Swing</td>
<td>44.12</td>
<td>48.48</td>
<td><strong>85.03</strong></td>
</tr>
<tr>
<td>Throw Discus</td>
<td>75.00</td>
<td>87.10</td>
<td><strong>93.13</strong></td>
</tr>
<tr>
<td>Volleyball Spiking</td>
<td>43.48</td>
<td><strong>90.90</strong></td>
<td>89.66</td>
</tr>
<tr>
<td>Mean Classification</td>
<td>64.23</td>
<td>70.47</td>
<td><strong>87.35</strong></td>
</tr>
</tbody>
</table>

Table 2.2: Recognition accuracies of our EXMOVES (applied to Dense Trajectories) compared with those of Action Bank and Discriminative Patches using the same subset of 13 action classes from UCF50 considered in [38].

HOG-HOF-STIPs and Dense Trajectories. The gap is particularly large on challenging datasets such as Hollywood-2 and HMDB51. This underscores the advantageous effect of the movement exemplars to which we compare the input video in order to produce the EXMOVE features.

Table 2.2 lists the individual action recognition accuracies for the same subset of 13 UCF50 classes analyzed in [38]. We see that EXMOVES give the highest accuracy on 10 out of these 13 action categories.

In Table 2.3 we present the recognition accuracy for the individual classes of HMDB51 using a linear SVM trained on our EXMOVES with Dense Trajectories. The best recognition performance is achieved for “golfing” (accuracy is 96.7%), while the worst prediction is for the class “waving” (accuracy is 5.6%).
2.6 Experiments

Computational cost of mid-level feature extraction. We want to emphasize that although our EXMOVES are based on a subset of the exemplars used to build Action Bank, they always generate equal or higher accuracy. Furthermore, our approach does so with a speedup of almost two-orders of magnitude in feature extraction: Table 2.4 reports the statistics of the runtime needed to extract EXMOVES and Action Bank. We used the software provided by the authors of [84] to extract Action Bank features from input videos. Due to large cost of Action Bank extraction, we collected our runtime statistics on the smaller-scale UT-I [83] dataset, involving only 120 videos. Runtimes were measured on a single-core Linux machine with a CPU @ 2.66GHz. The table reports the complete time from the input of the video to the output of the descriptor, inclusive of the time needed to compute low-level features. The extraction of EXMOVES is on average over 70 times faster than for Action Bank when using HOG-HOF-STIPs and 11 times faster when using Dense Trajectories. We can process the entire UT-Interaction dataset with HOG-HOF-STIPs using a single CPU in 14 hours; extracting the Action Bank features on the same dataset would take 41 days.

We were unable to collect runtime statistics for Discriminative Patches due to the unavailability of the software. However, we want to point out that this descriptor uses many more patches than EXMOVES (1040 instead of 188) and it cannot use the Integral Video speed-up.

Computational cost of action recognition. Finally, we would like to point out that as shown in Table 2.1 the accuracies achieved by an efficient linear SVM trained on EXMOVES are very close to the best published results of [113], which instead
2.6 Experiments

<table>
<thead>
<tr>
<th>Action</th>
<th>SVM (%)</th>
<th>Action</th>
<th>SVM (%)</th>
<th>Action</th>
<th>SVM (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>golf</td>
<td>96.7</td>
<td>laugh</td>
<td>48.9</td>
<td>smoke</td>
<td>31.1</td>
</tr>
<tr>
<td>pullup</td>
<td>87.8</td>
<td>ride bike</td>
<td>47.8</td>
<td>stand</td>
<td>31.1</td>
</tr>
<tr>
<td>pushup</td>
<td>76.7</td>
<td>turn</td>
<td>47.8</td>
<td>kick</td>
<td>27.8</td>
</tr>
<tr>
<td>brush hair</td>
<td>75.6</td>
<td>shoot bow</td>
<td>46.7</td>
<td>kick ball</td>
<td>26.7</td>
</tr>
<tr>
<td>situp</td>
<td>71.1</td>
<td>sit</td>
<td>46.7</td>
<td>walk</td>
<td>26.7</td>
</tr>
<tr>
<td>kiss</td>
<td>68.9</td>
<td>drink</td>
<td>45.6</td>
<td>sword</td>
<td>25.6</td>
</tr>
<tr>
<td>catch</td>
<td>65.6</td>
<td>hit</td>
<td>44.4</td>
<td>cartwheel</td>
<td>20.0</td>
</tr>
<tr>
<td>shake hands</td>
<td>65.6</td>
<td>push</td>
<td>43.3</td>
<td>run</td>
<td>20.0</td>
</tr>
<tr>
<td>hug</td>
<td>62.2</td>
<td>fall floor</td>
<td>42.2</td>
<td>sword exercise</td>
<td>18.9</td>
</tr>
<tr>
<td>dribble</td>
<td>61.1</td>
<td>somersault</td>
<td>41.1</td>
<td>dive</td>
<td>17.8</td>
</tr>
<tr>
<td>pour</td>
<td>61.1</td>
<td>shoot ball</td>
<td>38.9</td>
<td>eat</td>
<td>17.8</td>
</tr>
<tr>
<td>climb</td>
<td>58.9</td>
<td>talk</td>
<td>37.8</td>
<td>shoot gun</td>
<td>16.7</td>
</tr>
<tr>
<td>ride horse</td>
<td>56.7</td>
<td>jump</td>
<td>36.7</td>
<td>pick</td>
<td>14.4</td>
</tr>
<tr>
<td>flic flac</td>
<td>53.3</td>
<td>climb stairs</td>
<td>35.6</td>
<td>punch</td>
<td>11.1</td>
</tr>
<tr>
<td>chew</td>
<td>48.9</td>
<td>draw sword</td>
<td>35.6</td>
<td>throw</td>
<td>7.8</td>
</tr>
<tr>
<td>clap</td>
<td>48.9</td>
<td>smile</td>
<td>33.3</td>
<td>swing baseball</td>
<td>5.6</td>
</tr>
<tr>
<td>fencing</td>
<td>48.9</td>
<td>handstand</td>
<td>32.2</td>
<td>wave</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Table 2.3: Recognition accuracy on the individual classes of HMDB51 using linear SVMs trained on EXMOVES based on Dense Trajectories. Note that random chance would yield a recognition rate of 1.96%

were obtained with a much more computationally expensive model, not suitable for scalable action recognition: they report a top-performance of 46.6% and 58.2% on HMDB51 and Hollywood-2, respectively, using an expensive non-linear SVM with an RBF-$\chi^2$ kernel applied to BOW of Dense Trajectories. In our experiments we found that training a linear SVM on EXMOVES for one of the HMDB51 classes takes only 6.2 seconds but learning a kernel-SVM on BOW of Dense Trajectories requires 25 minutes (thus overhead is 250X); the testing of our linear SVM on a video takes only 7 milliseconds, while the nonlinear SVM is on average more than two orders of magnitude slower. Its cost depends on the on the number of support vectors, which varies from a few hundreds to several thousands. Nonlinear SVMs also need more memory to store the support vectors.
2.6 Experiments

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Extraction time per video (minutes)</th>
<th># frames per second</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>max</td>
</tr>
<tr>
<td>Action Bank</td>
<td>495</td>
<td>1199</td>
</tr>
<tr>
<td>EXMOVES w/ HOG-HOF-STIPs</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>EXMOVES w/ Dense Trajectory</td>
<td>43</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 2.4: Statistics of time needed to extract the mid-level descriptors Action Bank and EXMOVES. The time needed to extract EXMOVES features for the entire UT-I dataset using a single CPU is only 14 hours; instead, it would take more than 41 days to compute Action Bank descriptors for this dataset.

Varying the number of exemplars. In this experiment we study how the accuracy of our method changes as a function of the number of EXMOVES used in the descriptor. Starting from our complete feature vector defined by $N_a = 188$ exemplars and having dimensionality $N_a \times N_s \times N_p = 41,172$, we recursively apply a feature selection procedure that eliminates at each iteration one of the EXMOVE exemplars and removes its associated $N_s \times N_p$ features from the descriptor. We apply a variant of multi-class Recursive Feature Elimination [11] to determine the EXMOVE to eliminate at each iteration. This procedure operates as follows: given a labeled training set of video examples for $K$ classes, at each iteration we retrain the one-vs-the-rest linear SVMs for all $K$ classes using the current version of our feature vector and then we remove from the descriptor the EXMOVE that is overall “least used” by the $K$ linear classifiers by looking at the average magnitude of the SVM parameter vector $w$ for the different EXMOVE sub-blocks.

We perform this analysis on the HDMB51 dataset using both HOG-HOF-STIPs and Dense Trajectories as low-level features for EXMOVES. Figure 2.2 reports the
2.6 Experiments

Figure 2.2: Accuracy on HMDB51 as a function of the number of EXMOVES. We use Recursive Feature Elimination to reduce the number of EXMOVES. The accuracy remains near the state-of-the-art even when using only 100 exemplars.

3-fold cross-validation error as a function of the number of EXMOVES used in our descriptor. Interestingly, we see that the accuracy remains close to the top-performance even when we reduce the number of exemplars to only 100. This suggests a certain redundancy in the set of movement exemplars. The accuracy begins to drop much more rapidly when fewer than 50 exemplars are used.

The effects of multiple scales and spatio-temporal pyramid levels. We study the effects of varying the number of scales and the number of spatio-temporal pyramid levels on EXMOVES. The version of EXMOVES having highest dimensionality involves three different scales (1, 0.75, 5) and three different spatio-temporal pyramid levels (1 × 1 × 1, 2 × 2 × 2, and 3 × 3 × 3), as already discussed in the previous section. Here we vary the number of scales from only 1 scale (1), 2 scales (1, 0.75), or all 3 scales.
2.6 Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of scales</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 pyr. level</td>
<td>68.3 (188)</td>
<td>68.4 (376)</td>
<td>68.4 (564)</td>
</tr>
<tr>
<td></td>
<td>2 pyr. levels</td>
<td>74.8 (1,692)</td>
<td>77.1 (3,384)</td>
<td>78.0 (5,076)</td>
</tr>
<tr>
<td></td>
<td>3 pyr. levels</td>
<td>77.3 (13,724)</td>
<td>80.1 (27,448)</td>
<td><strong>82.8</strong> (41,172)</td>
</tr>
<tr>
<td>UCF101-part2</td>
<td>1 pyr. level</td>
<td>51.9 (188)</td>
<td>54.3 (376)</td>
<td>53.7 (564)</td>
</tr>
<tr>
<td></td>
<td>2 pyr. levels</td>
<td>62.0 (1,692)</td>
<td>64.7 (3,384)</td>
<td>66.2 (5,076)</td>
</tr>
<tr>
<td></td>
<td>3 pyr. levels</td>
<td>65.4 (13,724)</td>
<td>69.1 (27,448)</td>
<td><strong>71.6</strong> (41,172)</td>
</tr>
</tbody>
</table>

Table 2.5: **Effects of multiple scales and spatio-temporal pyramid levels on EXMOVES.** The recognition accuracy on UCF50 and UCF101-part2 are reported for different numbers of scales and levels of spatio-temporal pyramid. The EXMOVE dimensionalities for the different settings are shown in brackets.

We also vary the number of pyramid levels: only 1 level (level 1), 2 levels (1 and 2), or all 3 levels. At the lowest dimensionality, we only use 1 scale and 1 level of pyramid, which gives rises a 188-dimensional feature vector. At the highest dimensionality, with 3 scales and 3 pyramid levels, EXMOVES become 41,172-dimensional feature vectors. Table 2.5 presents the action recognition accuracy of EXMOVES varying the number of scales and pyramid levels on UCF50 and UCF101-part 2. The empirical results show that EXMOVES do not benefit much from multiple scales, but their performance is significantly boosted by the use of multiple spatio-temporal pyramid levels. Reducing from 3 scales to 2 scales causes only a 1-2% in accuracy, while going from 2 scales to 1 scale causes a degradation in accuracy of 2.5-3.5% on both datasets. Instead, moving from 3 to 2 pyramid levels the accuracy drops by 4-5%, and reducing the number of pyramid levels from 2 to 1 degrades the accuracy by 10-12%. Interestingly, EXMOVES built with 1-scale and 1-pyramid-level are 188-dimensional and they achieve an accuracy of 68.3% on UCF50. This is considerably higher than the accuracies of Action Bank [84] and Discriminative Patches [38] (57.9% and 61.2%, respectively), which have much higher dimensionality.
2.6 Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>HMDB51</th>
<th>Hollywood-2</th>
<th>UCF50</th>
<th>UCF101-part2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS-EXMOVES</td>
<td>35.2</td>
<td>56.0</td>
<td>78.0</td>
<td>69.3</td>
</tr>
<tr>
<td>EXMOVES</td>
<td>41.9</td>
<td>56.6</td>
<td>82.8</td>
<td>71.6</td>
</tr>
</tbody>
</table>

Table 2.6: The effects of bounding box annotations on EXMOVES. WS-EXMOVES are trained without manual bounding-box annotations, while EXMOVES are learned using a single manually-annotated bounding box in each video. The action recognition accuracy of WS-EXMOVES is 1-6% lower than that of EXMOVES on the HMDB51, Hollywood-2, UCF50, and UCF101 datasets.

The effects of bounding box annotations. We study the effects of annotations on our EXMOVES. In this experiment, we train our EXMOVES without using any bounding box annotations. We call these features WS-EXMOVES (weakly-supervised EXMOVES). We note that, as before, we are still using one positive example and many negative examples to train our mid-level descriptor, except for not using bounding box annotations. To train each WS-EXMOVE, we randomly generate $k^+ = 10$ subvolumes from the positive video and $k^- = 3$ subvolumes from each negative video. These subvolumes are used as positive and negative training examples to train a linear SVM. Each linear SVM is then calibrated by the same algorithm as before [78]. Table 2.6 compares the accuracies of WS-EXMOVES and EXMOVES on four different datasets. On Hollywood-2, the difference is small – only 0.6%, due to the small dataset and the reduced number of classes to discriminate (12-categories). On UCF50 and UCF101-part 2, the difference is about 2.3%-4.8%, while on the more challenging HMDB51 dataset the gap is 6.7%. As the accuracy drop for not having bounding box annotations is small, one can even afford to increase the number of exemplars to improve the discriminative power of the descriptor at very little annotation cost.
2.6 Experiments

2.6.2 Qualitative Results on Action Retrieval

We also qualitatively evaluate our EXMOVES on the task of action retrieval. In this experiment, given a query video, we perform simple Top-K retrieval using the Euclidean distance. Figure 2.3 shows the 15-nearest neighbors for 4 different queries using EXMOVES on UCF50. We intentionally chose queries from classes that can be recognized reliably (“Pommel Horse” and “Punch”), but also from the hardest category to recognize (“Basketball” which is often confused with “Volleyball Spiking” and “Pizza Tossing”). “Clean and Jerk” ranks roughly in the middle among all categories in terms of recognition accuracy with EXMOVES. As shown in Figure 2.3 for the two queries in the categories “Pommel Horse” and “Punch”, all 15-nearest belong to the same class as the query. The top-15 results retrieved for the query “Clean and Jerk” include one incorrect example (belonging to the class “Bench Press”). The retrieval results for the query belonging to the most challenging class (“Basketball”) include many more mistakes (10 in the top-15).

We also qualitatively evaluate our EXMOVES on the task of action retrieval across different datasets. Figure 2.4 shows the top-5 retrieval results when the query videos are from HMDB51 and the retrieval database is UCF50. We randomly selected queries from the two classes having the highest recognition accuracy (the first and the second query), the two classes having the worst recognition accuracy (the fifth and the sixth query), and from two classes with intermediate accuracy (the third and the fourth query). For the “ride horse” query, the top-5 results are all correct. The “pull up” retrieval results include an error (a “Swing” video, which exhibits appearance and motions similar to those of “pull up”). Note how for the query “riding bike”, the top-5 results include one incorrect video of “riding horse”. Finally, the “push up” query yields
2.6 Experiments

three incorrect examples of “bench press”, which are similar in pose and motions to “push up”.

2.6.3 Action Similarity Labeling

We now show that our EXMOVES can be applied to tasks beyond action recognition by presenting results on the problem of action similarity labeling [53].

Dataset. We use the ASLAN challenge dataset [53] for action similarity labeling. The dataset consists of 3697 video clips of 432 action categories. Given a pair of video clips as input, the objective is to determine whether they contain the “same” action or “different” actions. Thus, this can be viewed as a binary classification problem. In [53], the authors define 10 splits of the dataset. Each split contains 300 pairs of videos with same actions and 300 video pairs with different actions. The dataset is difficult because the number of action categories is large and the action classes are fine-grained. For example, there are 29 variants of jumping, and 10 distinct categories of “sitting-up”.

Binary classification model for action similarity labeling. In [53] the authors report the performance of several features (HOG, HOF, HNF [61], and their combination) with 12 different distance metrics used as kernels for binary classification of video pairs. In order to maintain our approach scalable and efficient, we train a binary linear SVM on the absolute difference of the two EXMOVE descriptors extracted from the input pair. In other words, the SVM is trained on the absolute difference vector to predict whether the two videos contain the same action or not. Note that the other distances used in [53], such as the $\chi^2$ or other non-linear kernels,
2.6 Experiments

Figure 2.3: **Action retrieval on UCF50.** The left column shows the query videos. Next to each query we show the 15-nearest neighbors retrieved using EXMOVES. The incorrectly-retrieved examples (i.e., videos belonging to a class different from that of the query) are marked with a red X. The first two query are from classes that are easy to recognize, the third query is randomly chosen from a class that ranks roughly in the middle in terms of recognition accuracy. The last query is randomly chosen from the most difficult class to recognize (according to the confusion matrix). Best viewed in color.
2.6 Experiments

Figure 2.4: **Cross-dataset action retrieval.** The first column shows the query videos from HMDB51. To the right of each query video we show the 5-nearest neighbors retrieved from UCF50. The retrieved examples that have incorrect label are marked with a red X. Best viewed in color.
2.6 Experiments

are much more costly to compute. We report the labeling accuracy as well as the area under ROC curve using 10-fold cross validation as used in [53].

**Comparison of features for action similarity labeling.** Table 2.7 presents the accuracy of our EXMOVES on the similarity labeling challenge. We include comparative results obtained with current state-of-the-art features using the same binary classification model, i.e., a binary SVM trained on the absolute difference vector. Our EXMOVES outperform all single feature descriptors (HOG, HOF, HNF) by 2-3% on accuracy and 3-4% on AUC. Our EXMOVES are even better than the combination of these three feature vectors, providing an improvement of 0.5% and 1% on accuracy and AUC, respectively.

Figure 2.5 shows qualitative results of action similarity labeling using EXMOVES for 4 test pairs of videos. Each row shows an input test pair of video clips (we present three frame of each video clip). The ground-truth action labels are marked in blue in the right bottom corner of each image sequence. The first two test pairs are true positives, i.e., the linear SVM using EXMOVES correctly labels these pairs as “same”. It is worth noting that the second test example is quite difficult as the same actions appearing in different scales, view, and lighting condition. The third pair causes a false negative prediction: the SVM using EXMOVES fails to label this pair as “same,” probably due to the largely different viewpoints of the two video clips. The last row shows a false positive case. Our system fails to label the two videos as “different” because of the similar patterns of motions and poses.
### 2.7 Conclusions

We have presented an approach for efficient large-scale analysis of human actions. It centers around the learning of a mid-level video representation that enables state-of-the-art accuracy with efficient linear classification models. The benefits of our features are threefold. First, building our representation requires very little human intervention, as only one positive manual annotation is required for each feature entry. Second, our approach is easy to scale to large datasets thanks to low computational cost of EXMOVE extraction and the good accuracy obtainable with linear classifiers, which are fast to train and test. Last but not least, our approach is quite general, as it provides good accuracy with different types of low-level features and different problems of human action analysis. Experiments on large-scale benchmarks of action recognition and action similarity labeling show the accuracy and efficiency of our approach. To our best knowledge, this work is the first one experimented on all known large-scale benchmarks for human action analysis.

Our mid-level features are produced by evaluating a set of movement classifiers over the input video. An important question we plan to address in future work is: how many mid-level classifiers do we need to train before accuracy levels off? Also, what kind of movement classes are particularly useful as mid-level features? Currently, we are restricted in the ability to answer these questions by the scarceness of labeled

<table>
<thead>
<tr>
<th>Feature</th>
<th>HOG</th>
<th>HOF</th>
<th>HNF</th>
<th>HOG-HOF-HNF</th>
<th>EXMOVES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc (AUC)</td>
<td>52.23 (54.41)</td>
<td>53.53 (55.59)</td>
<td>53.75 (55.90)</td>
<td>54.80 (57.01)</td>
<td>55.32 (58.06)</td>
</tr>
</tbody>
</table>

Table 2.7: Action similarity labeling results. Comparisons between EXMOVES and current state-of-the-art features [53] using a binary linear SVM trained on the absolute difference vector, i.e., \(|x_1 - x_2|\) where \(x_1, x_2\) here denote the feature vectors extracted from the two input videos. The numbers are accuracies and area under ROC curve (in parenthesis). EXMOVES outperform other single feature vectors by 3-4%, and combined descriptors by 1%.
Figure 2.5: **Action similarity labeling.** Visualizations of action similarity pairs. Each row represent a test input pair. The binary classifier using EXMOVES correctly classifies the pairs in the first two rows (true positives). The third row is a false negative, and the last row is a false positive. Note that although the two video clips in the second row have largely different scales and viewpoints, our method is able to correctly label them as containing the same action. Our method fails to label the third pair as “different” because of the different viewpoints, and the fourth pair as “same” because the two videos exhibit similar motions. Best view in color.

Data available, in terms of both number of video examples but also number of action classes. An exciting avenue to resolve these issues is the design of methods that can learn robust mid-level classifiers from weakly-labelled data, such as YouTube videos.

Additional material including software to extract EXMOVES from videos is available at [http://vlg.cs.dartmouth.edu/exmoves](http://vlg.cs.dartmouth.edu/exmoves).
Chapter 3

Learning Spatiotemporal Features with 3D Convolutional Networks

Abstract

In this chapter, we propose a simple, yet effective approach for spatiotemporal feature learning using deep 3-dimensional convolutional networks (3D ConvNets) trained on a large scale supervised video dataset. Our findings are three-fold: 1) 3D ConvNets are more suitable for spatiotemporal feature learning compared to 2D ConvNets; 2) A homogeneous architecture with small $3 \times 3 \times 3$ convolution kernels in all layers is among the best performing architectures for 3D ConvNets; and 3) Our learned features, namely C3D (Convolutional 3D), significantly outperform state-of-the-art methods on 4 different video analysis tasks and 6 different benchmarks with a simple linear SVM. In addition, the features are compact: achieving 52.8% accuracy on UCF101 dataset with only 10 dimensions and also very efficient to compute: 91 times faster than the current best hand-crafted features and approximately 2 orders of magnitude faster.
### 3.1 Introduction

Multimedia on the Internet is growing rapidly resulting in an increasing number of videos being shared every minute. To combat the information explosion it is essential to understand and analyze these videos for various purposes like search, recommendation, ranking etc. The computer vision community has been working on video analysis for decades and tackled different problems such as action recognition [62, 5], abnormal event detection [7], and activity understanding [55]. Considerable progress has been made in these individual problems by employing different specific solutions. However, there is still a growing need for a generic video descriptor that helps in solving large-scale video tasks in a homogeneous way.

There are four properties for an effective video descriptor: (i) it needs to be **generic**, so that it can represent different types of videos well while being discriminative. For example, Internet videos can be of landscapes, natural scenes, sports, TV shows,
movies, pets, food and so on; (ii) the descriptor needs to be compact: as we are working with millions of videos, a compact descriptor makes processing, storing, and retrieving tasks much more scalable; (iii) it needs to be efficient to compute, as thousands of videos are expected to be processed every minute in real world systems; and (iv) it must be simple to implement. Instead of using complicated feature encoding methods and classifiers, a good descriptor should work well even with a simple model (e.g. linear classifier).

Inspired by the deep learning breakthroughs in the image domain [56] where rapid progress has been made in the past few years in feature learning, various pre-trained convolutional network (ConvNet) models [44] are made available for extracting image features. These features are the activations of the network’s last few fully-connected layers which perform well on transfer learning tasks [123, 125]. However, such image based deep features are not directly suitable for videos due to lack of motion modeling (as shown in our experiments in sections 3.4, 3.5, 3.6). In this chapter we propose to learn spatio-temporal features using deep 3D ConvNet. We also show these features significantly outperform image based deep features and hand-crafted features on various video benchmarks by a good margin both qualitatively and quantitatively. Although 3D ConvNets were proposed before [43, 46], to our knowledge this work exploits 3D ConvNets in the context of large-scale supervised training datasets and modern deep architectures to achieve the best performance on different types of video analysis tasks. The features from these 3D ConvNets encapsulate information related to objects, scenes and actions in a video, making them useful for various tasks without requiring to finetune the model for each task. C3D has the properties that a good descriptor should have: it is generic, compact, simple and efficient. To summarize,
3.2 Related Work

our contributions in this chapter are:

- We experimentally show 3D convolutional deep networks are good feature learning machines that model appearance and motion simultaneously.

- We analyze different 3D ConvNet architectures empirically and find that the architectures with $3 \times 3 \times 3$ convolution kernels for all layers achieve the best accuracy.

- The proposed features with a simple linear model significantly outperform best published results on 4 different tasks and 6 different benchmarks (see Table 3.1). They are also compact and efficient to compute.

3.2 Related Work

Videos have been studied by the computer vision community for decades. Over the years various problems like action recognition [62], anomaly detection [7], video retrieval [3], event and action detection [76, 45], and many more have been proposed. Considerable portion of these works are about video representations. Laptev and Lindeberg [62] proposed spatio-temporal interest points (STIPs) by extending Harris corner detectors to 3D. SIFT and HOG are also extended into SIFT-3D [86] and HOG3D [51] for action recognition. Dollar et al. proposed Cuboids features for behavior recognition [19]. Sadanand and Corso built ActionBank for action recognition [84]. Recently, Wang et al. proposed improved Dense Trajectories (iDT) [114] which is currently the state-of-the-art hand-crafted feature. The iDT descriptor is an interesting example showing that temporal signals could be handled differently from that of spatial signal. Instead of extending Harris corner detector into 3D, it
3.2 Related Work

Figure 3.1: **2D and 3D convolution operations.** a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal.

starts with 2D Harris corners in video frames and uses optical flow to track them. For each tracker corner different hand-crafted features are extracted along the trajectory. Despite its good performance, this method is computationally intensive and becomes intractable on large-scale datasets.

With recent availability of powerful parallel machines (GPUs, CPU clusters), together with large amounts of training data, convolutional neural networks (ConvNets) [66] have made a come back providing breakthroughs on visual recognition [33, 56]. ConvNets have also been applied to the problem of human pose estimation in both images [39] and videos [40]. More interestingly these deep networks are used for image feature learning [21]. Similarly, Zhou et al. and perform well on transferred learning tasks. Deep learning has also been applied to video feature learning in an unsupervised setting [65]. In Le et al. [65], the authors use stacked ISA to learn spatio-temporal features for videos. Although this method showed good results on action recognition, it is still computationally intensive at training and hard to scale up for testing on large datasets. 3D ConvNets were proposed for human action recognition [43, 96] and for medical image segmentation [111, 106]. Recently, Karpathy et al. [46] trained deep networks on a large video dataset for video classification. Simonyan and Zisserman [90] used two stream networks to achieve best results on action recognition.
3.3 Learning Features with 3D ConvNets

Among these approaches, the 3D ConvNets approach in [43] is most closely related to us. This method used a human detector and head tracking to segment human subjects in videos. The segmented video volumes are used as inputs for a 3-convolution-layer 3D ConvNet to classify actions. In contrast, our method takes full video frames as inputs and does not rely on any preprocessing, thus easily scaling to large datasets. We also share some similarities with Karpathy et al. [46] and Simonyan and Zisserman [90] in terms of using full frames for training the ConvNet. However, these methods are built on using only 2D convolution and 2D pooling operations (except for the Slow Fusion model in [46]) whereas our model performs 3D convolutions and 3D pooling propagating temporal information across all the layers in the network (further detailed in section 3.3). We also show that gradually pooling space and time information and building deeper networks achieves best results and we discuss more about the architecture search in section 3.3.2.

3.3 Learning Features with 3D ConvNets

In this section we explain in detail the basic operations of 3D ConvNets, analyze different architectures for 3D ConvNets empirically, and elaborate how to train them on large-scale datasets for feature learning.

3.3.1 3D convolution and pooling

We believe that 3D ConvNet is well-suited for spatiotemporal feature learning. Compared to 2D ConvNet, 3D ConvNet has the ability to model temporal information better owing to 3D convolution and 3D pooling operations. In 3D ConvNets, convolu-
3.3 Learning Features with 3D ConvNets

tion and pooling operations are performed spatio-temporally while in 2D ConvNets they are done only spatially. Figure 3.1 illustrates the difference, 2D convolution applied on an image will output an image, 2D convolution applied on multiple images (treating them as different channels [90]) also results in an image. Hence, 2D ConvNets lose temporal information of the input signal right after every convolution operation. Only 3D convolution preserves the temporal information of the input signals resulting in an output volume. The same phenomena is applicable for 2D and 3D polling. In [90], although the temporal stream network takes multiple frames as input, because of the 2D convolutions, after the first convolution layer, temporal information is collapsed completely. Similarly, fusion models in [46] used 2D convolutions, most of the networks lose their input’s temporal signal after the first convolution layer. Only the Slow Fusion model in [46] uses 3D convolutions and averaging pooling in its first 3 convolution layers. We believe this is the key reason why it performs best among all networks studied in [46]. However, it still loses all temporal information after the third convolution layer.

In this section, we empirically try to identify a good architecture for 3D ConvNets. Because training deep networks on large-scale video datasets is very time-consuming, we first experiment with UCF101, a medium-scale dataset, to search for the best architecture. We verify the findings on a large scale dataset with a smaller number of network experiments. According to the findings in 2D ConvNet [91], small receptive fields of \( 3 \times 3 \) convolution kernels with deeper architectures yield best results. Hence, for our architecture search study we fix the spatial receptive field to \( 3 \times 3 \) and vary only the temporal depth of the 3D convolution kernels.

**Notations:** For simplicity, from now on we refer video clips with a size of \( c \times l \times h \times w \)
3.3 Learning Features with 3D ConvNets

where \( c \) is the number of channels, \( l \) is length in number of frames, \( h \) and \( w \) are the height and width of the frame, respectively. We also refer 3D convolution and pooling kernel size by \( d \times k \times k \), where \( d \) is kernel temporal depth and \( k \) is kernel spatial size.

**Common network settings:** In this section we describe the network settings that are common to all the networks we trained. The networks are set up to take video clips as inputs and predict the class labels which belong to 101 different actions. All video frames are resized into \( 128 \times 171 \). This is roughly half resolution of the UCF101 frames. Videos are split into non-overlapped 16-frame clips which are then used as input to the networks. The input dimensions are \( 3 \times 16 \times 128 \times 171 \). We also use jittering by using random crops with a size of \( 3 \times 16 \times 112 \times 112 \) of the input clips during training. The networks have 5 convolution layers and 5 pooling layers (each convolution layer is immediately followed by a pooling layer), 2 fully-connected layers and a softmax loss layer to predict action labels. The number of filters for 5 convolution layers from 1 to 5 are 64, 128, 256, 256, 256, respectively. All convolution kernels have a size of \( d \) where \( d \) is the kernel temporal depth (we will later vary the value \( d \) of these layers to search for a good 3D architecture). All of these convolution layers are applied with appropriate padding (both spatial and temporal) and stride 1, thus there is no change in term of size from the input to the output of these convolution layers. All pooling layers are max pooling with kernel size \( 2 \times 2 \times 2 \) (except for the first layer) with stride 1 which means the size of output signal is reduced by a factor of 8 compared with the input signal. The first pooling layer has kernel size \( 1 \times 2 \times 2 \) with the intention of not to merge the temporal signal too early and also to satisfy the clip length of 16 frames (e.g. we can temporally pool with factor 2 at most 4 times before completely collapsing the temporal signal). The two fully connected layers have
2048 outputs. We train the networks from scratch using mini-batches of 30 clips, with initial learning rate of 0.003. The learning rate is divided by 10 after every 4 epochs. The training is stopped after 16 epochs.

**Varying network architectures:** For the purposes of this study we are mainly interested in how to aggregate temporal information through the deep networks. To search for a good 3D ConvNet architecture, we only vary kernel temporal depth $d_i$ of the convolution layers while keeping all other common settings fixed as stated above. We experiment with two types of architectures: 1) homogeneous temporal depth: all convolution layers have the same kernel temporal depth; and 2) varying temporal depth: kernel temporal depth is changing across the layers. For homogeneous setting, we experiment with 4 networks having kernel temporal depth of $d$ equal to 1, 3, 5, and 7. We name these networks as depth-$d$, where $d$ is their homogeneous temporal depth. Note that depth-1 net is equivalent to applying 2D convolutions on separate frames. For the varying temporal depth setting, we experiment two networks with temporal depth **increasing:** 3-3-5-5-7 and **decreasing:** 7-5-5-3-3 from the first to the fifth convolution layer respectively. We note that all of these networks have the same size of the output signal at the last pooling layer, thus they have the same number of parameters for fully connected layers. Their number of parameters is only different at convolution layers due to different kernel temporal depth. These differences are quite minute compared to millions of parameters in the fully connected layers. For example, any two of the above nets with temporal depth difference of 2, only has 17K parameters fewer or more from each other. The biggest difference in number of parameters is between depth-1 net and depth-7 net where depth-7 net has 51K more parameters which is less than 0.3% of the total of 17.5 millions parameters of each
3.3 Learning Features with 3D ConvNets

![Figure 3.2: 3D ConvNet architecture search. Action recognition clip accuracy on UCF101 test split-1 of different 3D ConvNet architectures. 2D ConvNet performs worst and 3D ConvNet with $3 \times 3 \times 3$ kernels performs best among the experimented nets.]

We train these networks on the train split 1 of UCF101. Figure 3.2 presents clip accuracy of different architectures on UCF101 test split 1. The left plot shows results of nets with homogeneous temporal depth and the right plot presents results of nets that changing kernel temporal depth. **Depth-3** performs best among the homogeneous nets. Note that **depth-1** is significantly worse than the other nets which we believe is due to lack of motion modeling. Compared to the varying temporal depth nets, **depth-3** is the best performer, but the gap is smaller. We also experiment with bigger spatial receptive field (e.g. $5 \times 5$) and/or full input resolution ($240 \times 320$ frame inputs) and still observe similar behavior. This suggests $3 \times 3 \times 3$ is the best kernel choice for 3D ConvNets (according to our subset of experiments) and 3D ConvNets are
3.3 Learning Features with 3D ConvNets

![Diagram of C3D architecture](image)

**Figure 3.3:** **C3D architecture.** C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from $\text{pool1}$ to $\text{pool5}$. All pooling kernels are $2 \times 2 \times 2$, except for $\text{pool1}$ is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.

<table>
<thead>
<tr>
<th>Method</th>
<th># of Nets</th>
<th>Clip@1</th>
<th>Video@1</th>
<th>Video@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Frame + Multires of [46]</td>
<td>3 nets</td>
<td>42.4</td>
<td>60.0</td>
<td>78.5</td>
</tr>
<tr>
<td>Slow Fusion of [46]</td>
<td>1 net</td>
<td>41.9</td>
<td>60.9</td>
<td>80.2</td>
</tr>
<tr>
<td>C3D (trained from scratch)</td>
<td>1 net</td>
<td>44.9</td>
<td>60.0</td>
<td>84.4</td>
</tr>
<tr>
<td>C3D (fine-tuned)</td>
<td>1 net</td>
<td><strong>46.1</strong></td>
<td><strong>61.1</strong></td>
<td><strong>85.2</strong></td>
</tr>
</tbody>
</table>

**Table 3.2:** **Sports-1M classification result.** C3D outperforms [46] by 5% on top-5 video-level accuracy and achieves state-of-the-art result on this dataset.

Consistently better than 2D ConvNets for video classification.

**Verify on large-scale dataset:** Previous experiments give us a lot of insights about good architectures for 3D ConvNet, however, they are still conducted on a medium-size dataset. To further verify if 3D ConvNet performs better than 2D ConvNet, we collect a large-scale dataset, namely I380K, consisting of 380K Instagram videos of 382 video concepts. The dataset is split into a train/test split for training and evaluation. We train two 2D ConvNets and a 3D ConvNet on I380K from scratch and evaluate their accuracy to verify if 3D ConvNet outperforms 2D ConvNets in large-scale setting. The two 2D ConvNets are AlexNet [50] and VGG NetA [91]. The 3D ConvNet has an architecture similar to VGG NetA except we replace 2D convolution and pooling by 3D operations. We find that our 3D ConvNet outperforms AlexNet and VGG NetA by 4.5% and 2% respectively (random chance is 0.26%).
3.3 Learning Features with 3D ConvNets

3.3.3 Spatiotemporal feature learning

**Network architecture**: Our findings in the previous section indicate that homoge-
neous setting with convolution kernels of $3 \times 3 \times 3$ is the best option for 3D ConvNets.
This finding is also consistent with a similar finding in 2D ConvNets [91]. With a
large-scale dataset, one can train a 3D ConvNet with $3 \times 3 \times 3$ kernel as deep as
possible subject to the machine memory limit and computation affordability. With
current GPU memory, we design our 3D ConvNet to have 8 convolution layers, 5
pooling layers, followed by two fully connected layers, and a softmax output layer.
The network architecture is presented in figure 3.3. For simplicity, we call this net
C3D from now on. All of 3D convolution filters are $3 \times 3 \times 3$ with stride (1 in both
space and time). All 3D pooling layers are $2 \times 2 \times 2$ (except for pool1) with stride 1.
Only pool1 is of $1 \times 2 \times 2$ with the intention of preserving the temporal information
in the early phase. Each fully connected layer has 4096 output units.

**Dataset.** To learn spatiotemporal features, we train our C3D on Sports-1M
dataset [46] which is currently the largest video classification benchmark. The dataset
consists of 1.1 million sports videos. Each video belongs to one of 487 sports categories.
Compared with UCF101, Sports-1M has 5 times the number of categories and 100
times the number of videos.

**Training:** Training is done on the Sports-1M train split. As Sports-1M has many
long videos, we randomly extract five 2-second long clips from every training video.
Clips are resized to have a frame size of $128 \times 171$. On training, we randomly crops
input clips into $16 \times 112 \times 112$ crops for spatial and temporal jittering. We also
horizontally flip them with 50% probability. Training is done by SGD with mini-batch
size of 30 examples. Initial learning rate is 0.003, and is divided by 2 every 150K
3.3 Learning Features with 3D ConvNets

iterations. The optimization is stopped at 1.9M iterations (about 13 epochs). Beside the C3D net trained from scratch, we also experiment with C3D net fine-tuned from the model pre-trained on I380K.

**Sports-1M classification results:** Table 3.2 presents the results of our C3D networks compared with DeepVideo [46]. Both C3D networks outperform DeepVideo’s networks and achieve state-of-the-art accuracy. The C3D network trained from scratch obtains 84.4% and the one fine-tuned from I380K pre-trained model yields 85.5% at video top-5 accuracy. This result outperforms DeepVideo’s networks [46], which is currently the best published result, by 5% on Sports-1M the current largest video classification benchmark.

**C3D video descriptor:** After training, C3D can be used as a feature extractor for other video analysis tasks. To extract C3D feature, a video is split into 16 frame long clips with a 8-frame overlap between two consecutive clips. These clips are passed to the C3D network to extract \( \text{fc6} \) activations. These clip \( \text{fc6} \) activations are averaged to form a 4096-dim video descriptor which is then followed by an L2-normalization. We refer to this representation as C3D video descriptor/feature in all experiments, unless we clearly specify the difference.

**What does C3D learn?** We use the deconvolution method explained in [123] to understand what C3D is learning internally. We observe that C3D starts by focusing on appearance in the first few frames and tracks the salient motion in the subsequent frames. Figure 3.4 visualizes deconvolution of two C3D \( \text{conv5b} \) feature maps with highest activations projected back to the image space. In the first example, the feature focuses on the whole person and then tracks the motion of the pole vault performance over the rest of the frames. Similarly in the second example it first focuses on the eyes.
3.4 Action recognition

Figure 3.4: Visualization of C3D model, using the method from [123]. Interestingly, C3D captures appearance for the first few frames but thereafter only attends to salient motion. Best viewed on a color screen.

and then tracks the motion happening around the eyes while applying the makeup. Thus C3D differs from standard 2D ConvNets in that it selectively attends to both motion and appearance. We provide more visualizations in the appendix to give a better insight about the learned feature.

3.4 Action recognition

Dataset: We evaluate C3D features on UCF101 dataset [92]. The dataset consists of 13,320 videos of 101 human action categories. We use the three split setting provided with this dataset.

Classification model: We extract C3D features and input them to a multi-class linear SVM for training models. We experiment with C3D descriptor using 3 different nets: C3D trained on I380K, C3D trained on Sports-1M, and C3D trained on I380K and fine-tuned on Sports-1M. In the multiple nets setting, we concatenate the L2-normalized C3D descriptors of these nets.

Baselines: We compare C3D feature with a few baselines: the current best hand-crafted features, namely improved dense trajectories (iDT) [114] and the popular-used deep image features, namely Imagenet [44], using Caffe’s Imagenet pre-train model.
### 3.4 Action recognition

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [46]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [90]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [20]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [93]</td>
<td>75.8</td>
</tr>
<tr>
<td>C3D (1 net)</td>
<td>82.3</td>
</tr>
<tr>
<td>C3D (3 nets)</td>
<td><strong>85.2</strong></td>
</tr>
<tr>
<td>iDT with Fisher vector [77]</td>
<td>87.9</td>
</tr>
<tr>
<td>Temporal stream network [90]</td>
<td>83.7</td>
</tr>
<tr>
<td>Two-stream networks [90]</td>
<td>88.0</td>
</tr>
<tr>
<td>LRCN [20]</td>
<td>82.9</td>
</tr>
<tr>
<td>LSTM composite model [93]</td>
<td>84.3</td>
</tr>
<tr>
<td>Multi-skip feature stacking [60]</td>
<td>89.1</td>
</tr>
<tr>
<td><strong>C3D (3 nets) + iDT</strong></td>
<td><strong>90.4</strong></td>
</tr>
</tbody>
</table>

Table 3.3: **Action recognition results on UCF101.** C3D compared with baselines and current state-of-the-art methods. Top: baseline results; Middle: methods taking only RGB frames as inputs; Bottom: methods using multiple feature combinations.

For iDT, we use the bag-of-word representation with a codebook size of 5000 for each feature channel of iDT which are trajectories, HOG, HOF, MBHx, and MBHy. We normalize histogram of each channel separately using L1-norm and concatenate these normalized histograms to form a 25K feature vector for a video. For Imagenet baseline, similar to C3D, we extract Imagenet fc6 feature for each frame, average these frame features to make video descriptor. A multi-class linear SVM is also used for these two baselines for a fair comparison.

**Results:** Table 3.3 presents action recognition accuracy of C3D compared with the two baselines and current best methods. The upper part shows results of the two baselines. The middle part presents methods that use only RGB frames as inputs. And the lower part reports all current best methods using all possible feature combinations (e.g. optical flows, iDT).
3.4 Action recognition

Figure 3.5: **C3D compared with Imagenet and iDT in low dimensions.** C3D, Imagenet, and iDT accuracy on UCF101 using PCA dimensionality reduction. C3D outperforms Imagenet and iDT by 10-20% in low dimensions.

C3D fine-tuned net performs best among three C3D nets described previously. The performance gap between these three nets, however, is small (1%). From now on, we refer to the fine-tuned net as C3D, unless otherwise stated. C3D using one net which has only 4,096 dimensions obtains an accuracy of 82.3%. C3D with 3 nets boosts the accuracy to 85.2% with the dimension is increased to 12,288. C3D when combined with iDT further improves the accuracy to 90.4%, while when it is combined with Imagenet, we observe only 0.6% improvement. This indicates C3D can well capture both appearance and motion information, thus there is no benefit to combining with Imagenet which is an appearance based deep feature. On the other hand, it is benefit to combine C3D with iDT as they are highly complementary to each other. In fact, iDT is hand-crafted features based on optical flow tracking and histograms of low-level gradients while C3D captures high level abstract/semantic information.

Compared with the baselines, C3D with 3 nets achieves 85.2% which is 9% and 16.4% better than iDT and Imagenet, respectively. On the only RGB input setting,
3.4 Action recognition

Figure 3.6: **Feature embedding.** Feature embedding visualizations of Imagenet and C3D on UCF101 dataset using t-SNE \[107\]. C3D features are semantically separable compared to Imagenet suggesting that it is a better feature for videos. Each clip is visualized as a point and clips belonging to the same action have the same color. Best viewed in color.

compared with CNN-based approaches, Our C3D outperforms deep networks \[46\] and spatial stream network in \[90\] by 19.8% and 12.6%, respectively. Both deep networks \[46\] and spatial stream network in \[90\] use AlexNet architecture. While in \[46\], the net is fine-tuned from their model pre-trained on Sports-1M, spatial stream network in \[90\] is fine-tuned from Imagenet pre-trained model. Our C3D is different from these CNN-base methods in term of network architecture and basic operations. In addition, C3D is trained on Sports-1M and used as is without any finetuning. Compared with Recurrent Neural Networks (RNN) based methods, C3D outperforms Long-term Recurrent Convolutional Networks (LRCN) \[20\] and LSTM composite model \[93\] by 14.1% and 9.4%, respectively. C3D with only RGB input still outperforms these two RNN-based methods when they used both optical flows and RGB as well as the temporal stream network in \[90\]. However, C3D needs to
3.4 Action recognition

be combined with iDT to outperform two-stream networks \[90\] and other iDT-based methods \[77, 60\]. Apart from the promising numbers, C3D also has the advantage of simplicity compared to the other methods.

**C3D is compact:** In order to evaluate the compactness of C3D features we use PCA to project the features into lower dimensions and report the classification accuracy of the projected features on UCF101 \[92\]. We apply the same process with iDT \[114\] as well as Imagenet features \[21\] and compare the results in Figure 3.5. At the extreme setting with only 10 dimensions, C3D accuracy is 52.8% which is more than 20% better than the accuracy of Imagenet and iDT which are about 32%. At 50 and 100 dim, C3D obtains an accuracy of 72.6% and 75.6% which are about 10-12% better than Imagenet and iDT. Finally, with 500 dimensions, C3D is able to achieve 79.4% accuracy which is 6% better than iDT and 11% better than Imagenet. This indicates that our features are both compact and discriminative. This is very helpful for large-scale retrieval applications where low storage cost and fast retrieval are crucial.

We qualitatively evaluate our learned C3D features to verify if it is a good generic feature for video by visualizing the learned feature embedding on another dataset. We randomly select 100K clips from UCF101, then extract fc6 features for those clips using for features from Imagenet and C3D. These features are then projected to 2-dimensional space using t-SNE \[107\]. Figure 3.6 visualizes the feature embedding of the features from Imagenet and our C3D on UCF101. It is worth noting that we did not do any fine-tuning as we wanted to verify if the features show good generalization capability across datasets. We quantitatively observe that C3D is better than Imagenet.
3.5 Action Similarity Labeling

Dataset: The ASLAN dataset consists of 3,631 videos from 432 action classes. The task is to predict if a given pair of videos belong to the same or different action. We use the prescribed 10-fold cross validation with the splits provided with the dataset. This problem is different from action recognition, as the task focuses on predicting action similarity not the actual action label. The task is quite challenging because the test set contains videos of “never-seen-before” actions.

Features: We split videos into 16-frame clips with an overlap of 8 frames. We extract C3D features: prob, fc7, fc6, pool5 for each clip. The features for videos are computed by averaging the clip features separately for each type of feature, followed by an L2 normalization.

Classification model: We follow the same setup used in [53]. Given a pair of videos, we compute the 12 different distances provided in [53]. With 4 types of features,
3.5 Action Similarity Labeling

<table>
<thead>
<tr>
<th>Method</th>
<th>Features</th>
<th>Model</th>
<th>Acc.</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[53]</td>
<td>STIP</td>
<td>linear</td>
<td>60.9</td>
<td>65.3</td>
</tr>
<tr>
<td>[54]</td>
<td>STIP</td>
<td>metric</td>
<td>64.3</td>
<td>69.1</td>
</tr>
<tr>
<td>[52]</td>
<td>MIP</td>
<td>metric</td>
<td>65.5</td>
<td>71.9</td>
</tr>
<tr>
<td>[35]</td>
<td>MIP+STIP+MBH</td>
<td>metric</td>
<td>66.1</td>
<td>73.2</td>
</tr>
<tr>
<td>[118]</td>
<td>iDT+FV</td>
<td>metric</td>
<td>68.7</td>
<td>75.4</td>
</tr>
<tr>
<td>Baseline</td>
<td>Imagenet</td>
<td>linear</td>
<td>67.5</td>
<td>73.8</td>
</tr>
<tr>
<td>Ours</td>
<td>C3D</td>
<td>linear</td>
<td><strong>78.3</strong></td>
<td><strong>86.5</strong></td>
</tr>
</tbody>
</table>

Table 3.4: Action similarity labeling result on ASLAN. C3D significantly outperforms state-of-the-art method [118] by 9.6% in accuracy and by 11.1% in area under ROC curve.

we obtain 48-dimensional \((12 \times 4 = 48)\) feature vector for each video pair. As these 48 distances are not comparable to each other, we normalize them independently such that each dimension has zero mean and unit variance. Finally, a linear SVM is trained to classify video pairs into same or different on these 48-dim feature vectors. Beside comparing with current methods, we also compare C3D with a strong baseline using deep image-based features. The baseline has the same setting as our C3D and we replace C3D features with Imagenet features.

**Results:** We report the result of C3D and compare with state-of-the-art methods in table 3.4. While most current methods use multiple hand-crafted features, strong encoding methods (VLAD, Fisher Vector), and complex learning models, our method uses a simple averaging of C3D features over the video and a linear SVM. C3D significantly outperforms state-of-the-art method [118] by 9.6% on accuracy and 11.1% on area under ROC curve (AUC). Imagenet baseline performs reasonably well which is just 1.2% below state-of-the-art method [118], but 10.8% worse than C3D due to lack of motion modeling. Figure 3.7 plots the ROC curves of C3D compared with current methods and human performance. C3D has clearly made a significant improvement which is a halfway from current state-of-the-art method to human...
### 3.6 Scene and Object Recognition

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Imagenet</th>
<th>C3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maryland</td>
<td>43.1</td>
<td>87.7</td>
</tr>
<tr>
<td>YUPENN</td>
<td>80.7</td>
<td>98.1</td>
</tr>
</tbody>
</table>

Table 3.5: **Scene recognition accuracy.** C3D using a simple linear SVM outperforms state-of-the-art methods on Maryland and YUPENN.

3.6 Scene and Object Recognition

**Datasets:** For dynamic scene recognition, we evaluate C3D on two benchmarks: YUPENN [17] and Maryland [89]. YUPENN consists of 420 videos of 14 scene categories and Maryland has 130 videos of 13 scene categories. For object recognition, we test C3D on egocentric dataset [80] which consists 42 types of everyday objects. A point to note, this dataset is egocentric and all videos are recorded in a first person view which have quite different appearance and motion characteristics than any of the videos we have in the training dataset.

**Classification model:** For both datasets, we use the same setup of feature extraction and linear SVM for classification and follow the same leave-one-out evaluation protocol as described by the authors of these datasets. For object dataset, the standard evaluation is based on frames. However, C3D takes a video clip of length 16 frames to extract the feature. We slide a window of 16 frames over all videos to extract C3D features. We choose the ground truth label for each clip to be the most frequently occurring label of the clip. If the most frequent label in a clip occurs fewer than 8 frames, we consider it as negative clip with no object and discard it in both training and testing. We train and test C3D features using linear SVM and report the object performance (98.9%).
3.7 Runtime Analysis

We perform a runtime analysis of C3D and compare it with iDT [114] (best handcrafted features) and the Temporal stream network [90] (best deep learning based approach) for action recognition. For iDT, we use the code kindly provided by the authors [114]. For [90], there is no public model available to evaluate. However, this method uses Brox’s optical flows [8] as low level input signals. We manage to evaluate
### 3.8 Effects of Input Resolution

<table>
<thead>
<tr>
<th>Method</th>
<th>iDT</th>
<th>Brox’s</th>
<th>Brox’s</th>
<th>C3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>CPU</td>
<td>CPU</td>
<td>GPU</td>
<td>GPU</td>
</tr>
<tr>
<td>RT (in hours)</td>
<td>202.2</td>
<td>2513.9</td>
<td>607.8</td>
<td>2.2</td>
</tr>
<tr>
<td>FPS</td>
<td>3.5</td>
<td>0.3</td>
<td>1.2</td>
<td>313.9</td>
</tr>
<tr>
<td>x Slower</td>
<td>91.4</td>
<td>1135.9</td>
<td>274.6</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.6: **Runtime analysis on UCF101.** C3D is 91x faster than improved dense trajectories [114] and 274x faster than Brox’s optical flow methods, thus relatively more than two orders of magnitude faster than [90].

runtime of Brox’s method using two different versions: CPU implementation provided by the authors [8] and the GPU implementation provided in OpenCV. We note that, the runtime of Simonyan and Zisserman [90] is greater than that of [8].

We report runtime of the three above-mentioned methods to extract features for the whole UCF101 dataset in table 3.6 using a single CPU or a single K40 Tesla GPU. Note that this is not a fair comparison for iDT as it uses only CPU. We cannot find any GPU implementation of this method and it is not trivial to implement a parallel version of this algorithm on GPU. Note that C3D is much faster than real-time, processing at **313 fps** while the other two methods have a processing speed of less than 4 fps.

### 3.8 Effects of Input Resolution

As part of the architecture study, we examine the effects of input resolution on 3D ConvNets. We use the same common network setting described in section 3.3. We fix all convolution kernels to $3 \times 3 \times 3$ and vary the input resolutions to study the effects. We experiment with 3 different nets with input resolutions of $64 \times 64$, $128 \times 128$, and $256 \times 256$, namely **net-64**, **net-128**, and **net-256**, respectively. Note that **net-128** is equivalent to the **depth-3** net in section 3.3.2. Because of the difference in input

63
3.8 Effects of Input Resolution

Figure 3.8: 3D ConvNets with different input resolutions. Action recognition clip accuracy on UCF101 test split-1 of 3D ConvNets with different input resolutions.

resolutions, these nets have different output size at the last pooling layer, thus leading to a significant difference in terms of number of parameters. Table 3.7 reports the numbers of parameters and the training time of these nets. Figure 3.8 presents the clip accuracy of these nets on UCF101 test split-1. *Net-128* outperforms *net-64* by 3.1% and attains a comparable accuracy with *net-256*. This indicates that *net-128* provides a good trade-off between training time, accuracy, and memory consumption. We note that with the current GPU memory limit, one has to use model parallelism to train C3D with $256 \times 256$ input resolution.

<table>
<thead>
<tr>
<th>Net</th>
<th>net-64</th>
<th>net-128</th>
<th>net-256</th>
</tr>
</thead>
<tbody>
<tr>
<td># of params (M)</td>
<td>11.1</td>
<td>17.5</td>
<td>34.8</td>
</tr>
<tr>
<td>Train time (mins/epoch)</td>
<td>92</td>
<td>270</td>
<td>1186</td>
</tr>
</tbody>
</table>

Table 3.7: Number of parameters and training time comparison of 3D ConvNets with different input resolutions. Note that net-128 is equivalent to the depth-3 net in section 3.3.
3.9 Visualization of C3D Learned Features

For a better understanding of what C3D learned internally, we provide additional visualizations using deconvolution.

**Decovolutions of C3D:** We randomly select 20K clips from UCF101. We group clips that fire strongly for the same feature map at a pre-selected convolution layer. We use deconvolution \cite{123} to project the top activations of these clips back into image space. We visualize the gradients causing the activation together with the corresponding cropped image sequences. Note that we did not do any fine-tuning of C3D model on UCF101.

Figure 3.9 and 3.10 visualize deconvolutions of C3D learned feature maps at the layers \texttt{conv2a} and \texttt{conv3b}. Visualizations of the same feature map are grouped together. For figures 3.11, 3.12, 3.13, and 3.14, each figure presents the deconvolutions of one learned feature map of the \texttt{conv5b} layer. Finally, figure 3.15 compares the deconvolutions of several C3D \texttt{conv5b} feature maps with optical flows. As showed in the visualizations, at early convolution layer \texttt{conv2a}, C3D learns low-level motion patterns such as moving edges, blobs, short changes, edge orientation changes, or color changes. At a higher layer of \texttt{conv3b}, C3D learns bigger moving patterns of corners, textures, body parts, and trajectories. Finally, at the deepest convolution layer, \texttt{conv5b}, C3D learns more complicated motion patterns such as moving circular objects, biking-like motions.
3.9 Visualization of C3D Learned Features

Figure 3.9: Deconvolutions of C3D conv2a feature maps. Each group is a C3D conv2a learned feature map. First two rows: the learned filters detect moving edges and blobs. The last row: the learned filters detect shot changes, edge orientation changes, and color changes. Best viewed in a color screen.
3.9 Visualization of C3D Learned Features

Figure 3.10: Deconvolutions of C3D conv3b feature maps. Each group is a C3D conv3b learned feature map. Upper: feature maps detect moving corners and moving textures. Middle: feature maps detect moving body parts. Lower: feature maps detect object trajectories and circular objects. Best viewed in a color screen.
Figure 3.11: Deconvolutions of a C3D conv5b learned feature map which detects moving motions of circular objects. In the second last clip, it detects a moving head while in the last clip, it detects the moving hair-curler. Best viewed in a color screen.
3.9 Visualization of C3D Learned Features

Figure 3.12: Deconvolutions of a C3D conv5b learned feature map which detects biking-like motions. Note that the last two clips have no biking but their motion patterns are similar to biking motions. Best viewed in a color screen.
3.9 Visualization of C3D Learned Features

Figure 3.13: Deconvolutions of a C3D conv5b learned feature map which detects face-related motions: applying eye-makeup, applying lipstick, and brushing tooth. Best viewed in a color screen.
Figure 3.14: Deconvolutions of a C3D conv5b learned feature map which detects balance-beam-like motions. In the last clip, it detects hammering which shares similar motion patterns with balance beam. Best viewed in a color screen.
3.9 Visualization of C3D Learned Features

Figure 3.15: Deconvolutions of C3D conv5b learned feature maps compared with optical flows. Optical flows fire at all of moving pixels while C3D just pays attention to only salient motions. Best viewed in a color screen.
3.10 Conclusions

In this work we try to address the problem of learning spatiotemporal features for videos using 3D ConvNets trained on large-scale video datasets. We conducted a systematically study to find the best architecture for 3D ConvNets. We showed that C3D can model appearance and motion information simultaneously and outperforms the 2D ConvNet features on various video analysis tasks. We demonstrated that C3D feature achieves state-of-the-art results on 4 different video analysis tasks and 6 different benchmarks. Last but not least, the proposed C3D feature is efficient, compact, and extremely simple to use.

C3D source code and pre-trained model are available at [http://vlg.cs.dartmouth.edu/c3d](http://vlg.cs.dartmouth.edu/c3d).
Chapter 4

Deep End-to-End Voxel-to-Voxel Prediction

abstract

Over the last few years deep learning methods have emerged as one of the most prominent approaches for video analysis. However, so far their most successful applications have been in the area of video classification and detection, i.e., problems involving the prediction of a single class label or a handful of output variables per video. Furthermore, while deep networks are commonly recognized as the best models to use in these domains, there is a widespread perception that in order to yield successful results they often require time-consuming architecture search, manual tweaking of parameters and computationally intensive pre-processing or post-processing methods.

In this chapter we challenge these views by presenting a deep 3D convolutional architecture trained end to end to perform voxel-level prediction, i.e., to output a variable at every voxel of the video. Most importantly, we show that the same exact
Voxel to voxel prediction: is a fine-grained video understanding task where the algorithm needs to infer a variable for each input voxel. The problem has many potential applications including video semantic segmentation, optical flow prediction, depth estimation, and video coloring.

The proposed architecture can be used to achieve competitive results on three widely different voxel-prediction tasks: video semantic segmentation, optical flow estimation, and video coloring. The three networks learned on these problems are trained from raw video without any form of preprocessing and their outputs do not require post-processing to achieve outstanding performance. Thus, they offer an efficient alternative to traditional and much more computationally expensive methods in these video domains.

4.1 Introduction

During the last decade we have witnessed a tremendous growth in the number of videos created and shared on the Internet thanks to the advances in network bandwidth and computation. In turn this has lead to a strong effort toward the creation of better
tools and apps to search, browse and navigate this large and continuously expanding video collections. This poses new challenges for the computer vision community and gives new motivations to build better, faster and more generally applicable video analysis methods.

In the still-image domain deep learning has revolutionized the traditional computer vision pipeline, which typically consisted of: pre-processing, hand-construction of visual features, training of a learning model, and post-processing. Instead, the successful introduction of deep convolutional neural network [56, 33, 88, 91] has shown that much better results can be obtained through end to end learning on very large collections of image examples, where the network is trained on raw image input and it directly predicts the target output. Besides the demonstrated advantages in improved accuracy, these end to end learned models have also been shown to be often more computationally efficient than traditional hand-designed approaches because they eliminate the need for computationally expensive pre-processing and post-processing steps and because convolution can run very fast, particularly on GPUs.

The video domain is also harnessing the benefits of this revolution but it is still lagging compared to the image setting [21, 125, 94]. In particular, most of the end to end learning approaches for video analysis have been introduced in the area of classification and detection [46, 90, 113, 100] and involve predicting a single label or few output variables per video. However, there are many computer vision problems that require labeling every single voxel of a video. Examples include optical flow computation, video semantic segmentation, depth estimation and video coloring. There have been only a few attempts at approaching these pixel-labeling problems with deep learning [69, 31, 23] for images. One of the reasons is that deep networks
4.1 Introduction

typically involve a large set of pooling layers which significantly lower the spatial resolution of the output. In order to output pixel labels at the original resolution, several “unpooling” strategies have been proposed, including simple upsampling, and multi-scale approaches. One of the most promising solution in this genre is learning convolution filters that upsample the signal. The primary benefit of convolutional upsampling is that it only requires learning a small number of location-agnostic filters and thus it can be carried out with limited training data.

The objective of our work is to demonstrate that 3D convolutional networks (3D ConvNets) with upsampling layers enable highly effective end to end learning of voxel to voxel prediction models on various video analysis problems. Instead of building a highly specialized network for each problem, our goal is to show that the same 3D ConvNet architecture trained on three distinct application domains (optical flow prediction, semantic segmentation, video coloring) can produce competitive results on each of them. Although a thorough architecture search is likely to yield improved results, we find it useful to employ a single network model for the three distinct tasks to convey the message that deep learning methods do not necessarily require to be highly specialized for the task at hand in order to produce good results. For the same reason, we do not employ any pre-processing or post-processing of the data. Because our model is fully convolutional, it involves a small number of learning parameters which can be optimized with limited amount of supervised data. Furthermore, the elimination of computationally expensive pre-processing and post-processing methods (such as CRF optimization or variational inference) and the exclusive reliance on efficient convolution implies that our learned models run very fast and can be used in real-time video-processing applications such as those arising in big-data domains.
4.2 Related Work

In summary, our work provides the following findings:

(a) Fully convolutional 3D ConvNets enable end to end learning of voxel to voxel prediction models with limited training data.

(b) The same exact architecture can be employed to obtain competitive results on three different voxel-labeling applications: optical flow estimation, semantic segmentation of image sequences, and video coloring.

(c) In domains where supervised training data is scarce (such as in the case of optical flow), we can train our end to end learning model on the output of an existing hand-designed algorithm. We show that this results in a 3D ConvNet that achieves slightly better accuracy than the complex hand-tuned vision method but, most importantly, it is significantly more efficient.

(d) While fine-tuning a pre-trained model helps in most cases, it actually hurts when the new domain requires visual features that are quite distinct from those of the pre-learned model, such as in the case of fine-tuning an action recognition network for optical flow estimation.

4.2 Related Work

Video analysis has been studied by the computer vision community for decades. Different approaches were proposed for action recognition including: tracking-based methods [22], bag-of-visual words [75], biologically-inspired models [42], space-time shapes [5], HMMs [37], and template-based Action-Bank [84]. Different spatio-temporal features were also introduced for video and action classification: Spatio-Temporal
4.3 Video Voxel Prediction

Interest Points [62], improved Dense Trajectories [113]. Various methods were used for action and video event detection [87, 10, 121]. Although these methods showed to work reasonably well, they are not scalable because most of them require computational intensive steps during preprocessing (e.g. tracking, background subtraction, or feature extraction) or post-processing (CRF, variational inference).

Deep learning methods have recently shown good on different computer vision problems [94, 88, 74, 33, 4]. Thanks to their large learning capacity and the ability to optimize all parameters end to end, these methods achieved good performance on classification [56] and feature learning [94, 100] provided that there is sufficient supervised training data. Among the deep learning approaches, our proposed method is most closely related to the depth estimation method described in [23], the Fully Convolutional Network (FCN) [69], and FlowNet [31]. Our method shares with these approaches the property of making pixel-level predictions. However, all these prior methods are designed for still image problems, while our method operates on videos. To the best of our knowledge, our method is the first one addressing end-to-end training of video voxel prediction.

4.3 Video Voxel Prediction

Problem statement. The input to our system is video with size $C \times L \times H \times W$, where $C$ is the number of color channels, $L$ is its temporal length (in number of frames), and $H, W$ are the frame height and width. Then, a voxel prediction problem requires producing a target output of size $K \times L \times H \times W$, where $K$ is an application-dependent integer denoting the number of output variables that need to be predicted per voxel. It is worth nothing that the size of the input video and the output prediction are
4.3 Video Voxel Prediction

the same, except only for the number of input channels $C$ and the number of output channels $K$ are different. Normally, $C = 3$ for the case of color video inputs and $C = 1$ for gray-scale inputs. For the three voxel-prediction applications considered in this chapter, $K$ will have the following values: $K = 2$ for optical flow estimation (the horizontal and vertical motion displacement for each voxel), $K = 3$ for video coloring (the three color channels) and $K$ will be equal to the number of semantic classes in the case of video semantic segmentation.

**Proposed approach.** We propose a novel and unified approach for video voxel prediction based on a 3D ConvNet architecture with 3D deconvolution layers. We show the generality of the model by demonstrating that a simple unified architecture can work reasonably well across different tasks without any engineering efforts in architecture search. Since our method uses 3D deconvolution layers, we will start by briefly explaining the idea of 2D deconvolution \[123, 69\] and then present our architecture based on 3D deconvolution for voxel prediction.

**Deconvolution.** The concept of deconvolution was introduced by Zeiler and Fergus \[123\] to visualize the internal-layer filters of a 2D ConvNet. Because the objective of this prior work was merely filter visualization, there was no learning involved in the deconvolution layers and the weights were simply set to be equal to the transpose of the corresponding pre-trained convolution layers. Instead, Long *et al.* \[69\] introduced the idea of deconvolution as a trainable layer in 2D ConvNets with applications to image semantic segmentation. As shown in Figure 4.2, a filter of a trainable deconvolution layer acts as a learnable local upsampling unit. In convolution, input signals are convolved by the kernel filter and one value is placed on the output plane. Conversely, deconvolution takes one value from the input, multiples the value
4.3 Video Voxel Prediction

Figure 4.2: **Deconvolutional layers in ConvNets.** a) Visualization of the deconvolutional layer used in [123] where the filter weights are set to be equal to those of the pre-trained convolutional layer. b) Trainable deconvolutional layers [69] learn upsampling.

by the weights in the filter, and place the result in the output channel. Thus, if the 2D filter has size $s \times s$, it generates a $s \times s$ output matrix for each pixel input. The output matrices can be stored either overlapping or not overlapping in the output channel. If not overlapping, then deconvolution with a $s \times s$ filter would upsample the input by a factor $s$ in both dimensions. When the output matrices overlap, their contributions in the overlap are summed up. The amount of output overlap depends on the output *stride*. If the output stride is bigger than 1, then the deconvolution layer produces an output with size larger than the input, thus acts as an upsampler.

In our architecture, we use 3D deconvolutional layers, instead of 2D deconvolutional layers. This means that the filters are deconvolved spatio-temporally, instead of only spatially as in 2D ConvNets.

**Architecture for voxel prediction.** Our architecture (which we name V2V, for voxel-to-voxel) is adapted from the C3D network described in [100], which has shown good performance for different video recognition tasks. In order to apply it to voxel-prediction problems, we simply add 3D deconvolutional layers to the C3D network. Note that C3D operates by splitting the input video into clips of 16 frames.
each and perform prediction separately for each clip. Thus, our V2V model also takes as input a clip of 16 frames and then outputs voxel labels for the 16 input frames. Figure 4.3 illustrates our V2V architecture for voxel prediction. The lower part contains layers from C3D, while the upper part has three 3D convolutional layers, three 3D deconvolutional layers, two concatenation layers, and one loss layer. All three convolutional layers (Conv3c, Conv4c, and Conv-pre) use filters of size $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$ and padding $1 \times 1 \times 1$. Conv3c and Conv4c act as feature-map reducers, while Conv-pre acts as a prediction layer. Deconv5 and Deconv4 use filters of size $4 \times 4 \times 4$ with output stride $2 \times 2 \times 2$ and padding $1 \times 1 \times 1$. The Deconv3 layer uses kernels of size $8 \times 4 \times 4$, an output stride of $4 \times 2 \times 2$, and padding $2 \times 1 \times 1$. Note that the number written inside the box of each layer in the Figure indicates the number of filters (e.g., 64 for Deconv3). The voxel-wise loss layer and Conv-pre layer are application-dependent and will be described separately for each of the applications considered in this chapter. Since V2V shares the bottom layers with C3D, we have the option to either fine-tuning these layers starting from the C3D weights, or learning the weights from scratch. We will report results for both options in our experiments.

### 4.4 Application I: Video Semantic Segmentation

**Dataset.** Our experiments for video semantic segmentation are carried out on the GATECH dataset [79], which comes with a public training/test split. The training set contains 63 videos while the test set has 38 sequences. There are 8 semantic classes: sky, ground, solid (mainly buildings), porous (mainly trees), cars, humans, vertical mix, and main mix.

**Training.** Similarly to C3D, we down-scale the video frames to size $128 \times 171$. 

82
4.4 Application I: Video Semantic Segmentation

Figure 4.3: V2V Architecture for Voxel Prediction. The lower part (below dashed line) consists of layers from C3D [100]. Connected to these layers we have three 3D convolution layers: Conv3c, Conv4c, Conv-pre use filters of size $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$. Both Deconv5 and Deconv4 are deconvolutional layers employing kernels of size $4 \times 4 \times 4$ with output stride of $2 \times 2 \times 2$. Deconv3 has kernel size $8 \times 4 \times 4$ and output stride of $4 \times 2 \times 2$. The numbers inside the boxes represent the number of learning filters in that layer, while the numbers near the boxes (above or below) represent the size of output signals produced by that layer. The part inside the thick-dashed box is application-dependent.

Because the dataset is quite small, we split each training video into all possible clips of length 16 (thus, we take overlapping clips with stride 1). For testing, we perform prediction on all non-overlapping clips of the video (stride equal to 16). We use the V2V architecture described in section 4.3 with $K = 8$ prediction channels, corresponding to the 8 semantic classes. We use a voxel-wise softmax for the loss layer. We fine-tune the full V2V network initialized from C3D, using randomly initialized weights for the new layers. The learning rate is set initially to $10^{-4}$, and it is divided by 10 every 30K iterations. The size of each mini-batch is 1. Fine-tuning is stopped at 100K iterations, approximately 9 epochs.

Baselines. We compare our V2V model with several baselines to gain better insights about our method. The first set of baselines are based on bilinear upsampling. The purpose of these baselines is to understand the benefits of our 3D deconvolution
4.4 Application I: Video Semantic Segmentation

Figure 4.4: **Video Semantic Segmentation Results on GATECH.** The softmax prediction heat maps produced by V2V for different classes together with input frames. The last two classes are omitted due to their small populations. Best viewed in color.

layers compared to simple upsampling. Instead of using V2V with deconvolution layers, we use only C3D up to Conv5b, we then add a prediction layer (analogous to Conv-pret). Because the prediction made at Conv5b has size $2 \times 7 \times 7$, we apply a bilinear upsampling to produce a prediction of the same size as the input. We call this baseline Conv5b-up. We include two other baselines, namely, Conv4b-up and Conv3b-up, corresponding to adding a prediction layer and an upsampling layer at Conv4b and Conv3b, respectively. Besides these upsampling baselines, we also compare our fine-tuned V2V model with the V2V architecture trained from scratch on GATECH, which we call V2V-0. We also trained a 2D version of V2V, namely 2D-V2V. The model 2D-V2V has the same architecture as V2V except that all 3D convolutional layers, 3D pooling layers, and 3D deconvolutional layers are replaced with 2D convolutional layers, 2D pooling layers, and 2D deconvolutional layers, respectively. As we do not have a pre-trained model of 2D-V2V, we train 2D-V2V from scratch on GATECH.
4.4 Application I: Video Semantic Segmentation

<table>
<thead>
<tr>
<th>Method</th>
<th>Train</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D-V2V</td>
<td>from scratch</td>
<td>55.7</td>
</tr>
<tr>
<td>V2V-0</td>
<td>from scratch</td>
<td>66.7</td>
</tr>
<tr>
<td>Conv3b+Up</td>
<td>fine-tune</td>
<td>69.7</td>
</tr>
<tr>
<td>Conv4b+Up</td>
<td>fine-tune</td>
<td>72.7</td>
</tr>
<tr>
<td>Conv5b+Up</td>
<td>fine-tune</td>
<td>72.1</td>
</tr>
<tr>
<td>V2V</td>
<td>fine-tune</td>
<td><strong>76.0</strong></td>
</tr>
</tbody>
</table>

Table 4.1: Semantic segmentation accuracy on GATECH. V2V consistently outperforms all baselines showing the good benefits of using V2V with 3D convolution/deconvolution compared to 2D convolution/deconvolution or bilinear upsampling.

Results. Figure 4.4 visualizes some qualitative results of semantic segmentation using V2V on GATECH. Table 4.1 presents the semantic segmentation accuracy on GATECH of V2V compared with all of the baselines. 2D-V2V, trained from scratch on GATECH, obtains 55.7% which is 11% below V2V-0. This result underscores the advantages of 3D convolution and 3D deconvolution over their 2D counterparts. Note also that V2V-0 is 9.3% below V2V. This predictably confirms the benefit of large-scale pre-training before fine-tuning. Finally, V2V also outperforms all bilinear upsampling baselines showing the advantages of using deconvolution over traditional upsampling. More qualitative comparisons of V2V with upsampling baselines are presented in Figure 4.5. Here we can see that Conv5b-Up yields fairly accurate predictions but over-smoothed due to its big upsampling rate. On the other extreme, Conv3b-up produces finer predictions thanks to the lower upsampling rate, but its segments are noisy and fragmented because it relies on feature maps at layer 3, thus less deep and less complex than those used by Conv5b-Up.
4.5 Application II: Optical Flow Estimation

![Diagram of optical flow estimation](image)

Figure 4.5: **V2V (top row) compared with upsampling baselines (rows 2-4).** V2V consistently outperforms all bi-linear upsampling baselines. Conv5b-Up provides fairly accurate prediction, but over-smoothed due to the high upsampling factor. Conversely, Conv3b-Up yields finer predictions, but more noisy because it uses less deep features. V2V gives by far the best tradeoff as it has access to deep features and it learns the upsampling filters.

### 4.5 Application II: Optical Flow Estimation

**Dataset.** Since there is no large-scale video dataset available with optical flow ground truth, we fabricate our training data by applying an existing optical flow method on unlabeled video. Specifically, we use the OpenCV GPU implementation of Brox’s method [8] to generate semi-truth data on both UCF101 [92] (public test split 1) and MPI-Sintel [9] (training set).

**Training.** We use the same V2V architecture with the number of channels at prediction layer set to $K = 2$. On both horizontal and vertical motion components, we use the Huber loss for regression as it works well with noisy data and outliers.
<table>
<thead>
<tr>
<th>Method</th>
<th>Brox</th>
<th>V2V-Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run-time (hours)</td>
<td>202.6</td>
<td>2.8</td>
</tr>
<tr>
<td>FPS</td>
<td>1.3</td>
<td>91.6</td>
</tr>
<tr>
<td>x Slower</td>
<td>70.5</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: **Runtime comparison**. The first row reports the total runtime (including I/O) to extract optical flow using V2V-Flow and Brox’s method [8] for the entire UCF101 test split 1. V2V-Flow is 70x faster than Brox’s method, besides being slightly more accurate (see Table 4.3).

Formally, this is given by

\[
H(x) = \begin{cases} 
\frac{1}{2}x^2, & |x| \leq 1 \\
|x|, & \text{otherwise}
\end{cases} \tag{4.1}
\]

To avoid numerical issues, the optical flow values are divided by a constant (\(\alpha = 15\)) so that most values fall in the range of \([-1, 1]\). We note that larger optical flows are still handled by the Huber loss. The V2V network takes as input clips of size \(3 \times 16 \times 122 \times 112\) and produces clip outputs of size \(2 \times 16 \times 112 \times 112\). The network is trained from scratch on UCF101 (using non-overlapping clips from each video) with a mini-batch size of 1. The initial learning rate is set to \(10^{-8}\) and it is divided by 10 every 200K iterations (about 2 epochs). Training is stopped at 800K iterations. We note that, at inference time, we need to scale the predictions by \(\alpha = 15\) to convert them back into the correct optical flow range.

**Results.** Figure [4.6](#) visualizes optical flow predicted by our V2V method and compares it with that computed by Brox’s method for a few sample clips taken from the test split of UCF101. The V2V end point error (EPE) on the UCF101 test split 1 (treating Brox’s optical flow as ground truth) is only 1.24. To better understand the performance of the learned V2V network, we further evaluate its performance on
4.5 Application II: Optical Flow Estimation

Figure 4.6: **Optical flow estimation on UCF101.** The output of V2V is qualitatively compared with Brox’s optical flow for 6 sample clips from the UCF101 test split. For each example we show (from left to right): an input frame, V2V’s predicted optical flow, and Brox’s motion. Note that Brox’s method is used to generate semi-truth data for training V2V. We see that on test videos V2V is able to predict flow of similar quality as that produced by Brox’s algorithm. Best viewed in color.

the training set of the MPI-Sintel dataset [9], which comes with ground truth data. This ground truth data is unbiased and allows us to assess performance independently from the accuracy of Brox’s flow. Table 4.3 shows the EPE error obtained with two variants of our model: V2V stands for our network learned on the UCF101 Brox’s flow, while finetuned-V2V denotes our model after fine-tuning V2V on Sintel ground truth data using 3-fold cross validation. The table also contains the best method on Sintel which is better than V2V by a good margin. Even though V2V is not state of the art, the results are very interesting: both V2V and finetuned-V2V perform better than their “teacher”, the optical flow method that is used to generate the semi-truth training data. While the improvement is slim, it is important to highlight that V2V is **much faster** than Brox’s algorithm (70x faster, see Table 4.2). Thus, this experiment shows that the V2V network can be employed to learn efficient implementations of
Table 4.3: Optical flow results on Sintel. V2V denotes our network learned from the UCF101 optical flow computed with Brox’s method. The finetuned-V2V network is obtained by fine-tuning V2V on Sintel (test accuracy is measured in this case using 3-fold cross validation). Both versions of our network perform slightly better than Brox’s algorithm and they allow computation of optical flow with a runtime speedup of 20 times compared to Brox’s software.

Table 4.2 presents the detailed runtime comparison between V2V-Flow and Brox’s method [8]. We use the GPU implementation of Brox’s method provided in OpenCV. Table 4.2 reports the runtime (including I/O) to extract optical flow for the whole UCF101 test split 1 by the two methods using a NVIDIA Tesla K40. V2V-Flow is 70x faster than Brox’s method. It can run at 91 fps while Brox’s method operates at less than 2 fps.

Observation. Unlike the case of video semantic segmentation application where V2V could be effectively fine-tuned from the initial C3D network, we empirically discovered that fine-tuning from C3D does not work for the case of optical flow estimation as in this case the training consistently converges to a bad local minimum.
4.6 Application III: Video Coloring

![Conv1a filters learned by C3D (top) and V2V (bottom).](image)

Figure 4.8: Visualization of Conv1a filters learned by C3D (top) and V2V (bottom). Note that C3D is trained to recognize actions (on Sport1M), while V2V is optimized to estimate optical flow (on UCF101). Each set shows the 64 learned filters at the Conv1a layer. Three consecutive square images on each row represent one filter (as kernel size is $3 \times 3 \times 3$). Each square image is upscaled to $30 \times 30$ pixels for better visualization. Best viewed in color. GIF animation of these filters are provided in the supplementary material.

We further investigated this phenomenon by visualizing the learned filters of the first few convolutional layers for both the original C3D as well as the V2V learned from scratch on Brox’s flow. The results are visible in Fig. 4.8. We see that the filters of the two networks look completely different. This is understandable, as C3D is trained to complete a high-level vision task, e.g. classifying sports. Thus the network learns a set of discriminative filters at the early layers. Some of these filters capture texture, some focus on discriminative motion patterns, while others respond to particular appearance or color cues. Instead, V2V is trained to perform a low-level vision task, e.g. predict motion directions. The Figure shows that the V2V filters are insensitive to color and texture as they focus exclusively on motion estimation. This explains why the pre-trained C3D model is a bad initialization to learn V2V for optical flow, but it is instead a good initialization for training V2V on semantic segmentation.

4.6 Application III: Video Coloring

**Setup and Training.** In this experiment we use UCF101 again in order to learn to color videos. We use the public training/test split 1 for the training and testing of...
4.6 Application III: Video Coloring

Figure 4.9: Examples of video coloring with V2V on the test set of UCF101. For each example we show (from left to right): a gray-scale input frame, the output frame colored by V2V, and the ground truth color frame. The V2V model is able to predict “common sense” colors such as the color of human skin, sky, woody furniture, river, sea, and mountain. Best viewed in color.

our model. In this study we generate training data by converting the color videos to grayscale. V2V is fed with $C = 1$ input grayscale channel and it is optimized to predict the $K = 3$ ground truth original color channels. For this application we use the L2 regression loss as colors have no outliers. We use mini-batches of size 1. The learning rate is set initially to $10^{-8}$ and it is divided by 10 every 200K iterations. The training is stopped at 600K iterations. Similarly to the case of semantic segmentation, we compare our V2V with its 2D version baseline, 2D-V2V, both optimized on the same training set. Both models were learned from scratch.

We note that video coloring is challenging and ill-posed because there are some
4.7 Conclusions

objects (e.g., clothes) that can be colored with any valid color. A reasonable expectation is that the coloring algorithm should learn to color correctly objects that typically occur only in one color. For example, the sky is usually blue (not always but often) and the grass is typically green. Thus, the model should learn to predict well the colors of such objects.

**Results.** To assess performance, we use as metric the average Euclidean distance between the predicted color and the true color. Here each voxel color is represented in $(r, g, b)$ and $r, g, b \in [0, 1]$. V2V has an average distance error (ADE) of 0.1375 whereas the 2D baseline has an ADE of 0.1495. Figure 4.9 presents some qualitative results of V2V on predicting voxel colors. It is interesting to see that the algorithm learns “common sense” colors such as the color of skin, sky, trees, river, sea, mountains, wood furniture, and the billiard table. For objects whose color is ambiguous, V2V applies very little coloring, leaving them almost in the original grayscale form. One can imagine extending V2V to have sparse inputs of color to make the problem well-posed for objects that can occur in various colors.

4.7 Conclusions

We have presented V2V, a novel architecture for voxel to voxel prediction using 3D convolutional networks. The proposed approach can be trained end to end from raw video input to predict target voxel labels without the need to preprocess or post-process the data. We have shown that the same architecture trained on three distinct application domains delivers competitive results on each of them. In the course of our experiments we have discovered that fine-tuning pre-trained models does not always help: for the case of optical flow estimation, learning from scratch is beneficial over
4.7 Conclusions

fine-tuning from an action recognition model. We have also demonstrated that in absence of large-scale supervised data, V2V can be trained to reproduce the output of an existing hand-constructed voxel prediction model. Quite surprisingly, in our study the resulting learned model has accuracy superior (albeit only slightly) to its “teacher” method. We believe that bootstrapping the learning from an existing model can be an interesting avenue for future work and can be a successful strategy to learn efficient implementation of computationally expensive algorithm, such as in our case where V2V predicts optical flow with a 70x speedup over the original optical flow method that was used to generate training data. While we purposely avoided specializing the network to each task in order to emphasize the general applicability of the approach, we believe that further improvements can be obtained from more thorough architecture search.
Chapter 5

ViCom: Benchmark and Methods for Video Comprehension

Abstract

There is a widespread agreement that future technology for organizing, browsing and searching videos hinges on the development of methods for high-level semantic understanding of video. But, so far the community has not reached to a consensus on the best way to train and assess models for this task. Casting video understanding as a form of action or event categorization is unsatisfying as it is not clear what the semantic classes or abstractions of this domain should be. Language has been exploited to sidestep the problem of defining abstract video categories by formulating video understanding as the task of captioning or description. However, language is highly complex, redundant and sometimes ambiguous. Many different captions may express the same semantic concept. To account for this ambiguity, quantitative evaluation of video description requires sophisticated metrics, whose performance
scores are typically hard to interpret by humans.

This chapter provides four contributions on this problem. First, we formulate \textit{Video Comprehension} as a new well-defined task with an easy-to-interpret performance measure. Second, we describe a general semi-automatic procedure to create benchmarks for this task. Third, we publicly release a large-scale video benchmark created with an implementation of this procedure and we include a human study that assesses human performance on our dataset. Finally, we propose and test a varied collection of approaches on this benchmark for the purpose of gaining a better understanding of the new challenges posed by video comprehension.

5.1 Introduction

Over the last few years deep learning has revolutionized the field of still-image analysis by producing breakthrough results in several domains including object categorization \cite{82, 56}, detection \cite{32}, scene classification \cite{125}, and semantic segmentation \cite{69}. While there has been widespread expectation that these performance improvements will naturally extend to the video domain, the results so far have been lagging compared to the image setting.

We argue that progress in this field has been held back primarily by the small-scale of existing labeled video datasets and by the low-quality of the annotations available to train machine learning models. Deep architectures have extensive learning capacity but require large-scale training sets (containing millions of examples) in order to learn effectively. Conversely, they are extremely prone to overfitting and poor performance when trained on small datasets. Unfortunately, today even the largest video analysis benchmarks, such as UCF101 \cite{92} and Sports-1M \cite{46}, are too small in size to enable
5.1 Introduction

effective learning of deep models. Furthermore, the labels manually collected on these
datasets merely specify the class of the action in each video (e.g., walking or sitting) but do not indicate where the action is performed. Thus, the learning algorithm is left with the burden of discovering on its own the portion of the video that is truly representative of the action. Moreover, these datasets are quite limited in semantic scope, as they include only a small number of action categories (e.g., 101 classes for UCF101 and just sport activities for the case of Sports-1M). Therefore, features learned from such datasets are unlikely to perform well on videos containing more general, everyday actions. Finally, one can argue that understanding a video is much more than recognizing the actions contained therein: it also requires understanding the scene context (office vs home), the objects (book vs laptop), the interactions between the subjects (discussing vs arguing) and much more.

For these reasons, several authors have proposed to abandon the classic view of video understanding as a form of action categorization and they suggested to reformulate the task as a description or captioning problem, where the objective is to generate a sentence summarizing the input video. On one hand, this makes the output directly readable by human subjects, which is desirable for application scenarios such as video browsing or searching. The downside however is that the outputs produced by such methods are hard to evaluate quantitatively. This happens because for each input there is not a single correct output, as many different sentences can reasonably describe a given video. To address this ambiguity, one can resort to comparative evaluation by human judges \[13\]. But this would require a huge crowdsourcing effort for every new algorithm to assess. Another approach is to design sophisticated metrics (e.g., METEOR or BLUE) that can capture the similarities of
5.1 Introduction

captions expressing the same semantic concept. However, it is hard for humans to interpret the meaning of these scores, e.g., is a METEOR score of 28% representing an acceptable captioning performance, or how big a difference in these scores would lead to a noticeable difference in the predictions?

In this chapter we propose to cast video understanding in the form of multiple choice tests that assess the ability of the algorithm to comprehend the semantics of the video. Figure 5.1 illustrates an example of video comprehension. The algorithm is presented with an input video and \( k \) possible descriptions. Only one of them represents the correct caption for the video. The task is well-posed as a traditional classification problem, with performance numbers easy to interpret (e.g., random chance produces an accuracy of \( 1/k \)). Yet, our classification task does not entail the definition of arbitrary video classes or action categories. Furthermore, we describe a procedure to construct multiple-choice tests for video comprehension with very little human intervention. This makes it possible to generate large-scale benchmarks for training and testing deep models on this task. Using this procedure we built a dataset that we will release to the research community. Although this is only the first version of our benchmark, it has already size comparable with the largest existing datasets for video analysis. Furthermore, in the supplementary material we discuss our plan to scale up considerably the size of this benchmark within the next year since it involves very little human intervention. Finally, in this chapter we also present preliminary results achieved with three distinct approaches to video comprehension, respectively based on the methodologies of regression, metric learning and video captioning. Perhaps surprisingly, video captioning yields the worst results, which suggests that models trained to generate description are not effective for video comprehension. Conversely,
5.1 Introduction

we demonstrate that metric learning methods produce by far the best accuracy on this task. In summary, the contributions of our work are four-fold:

• We propose a new high-level video understanding task which is well-posed and easy to evaluate.

• We describe a general semi-automatic procedure to construct benchmarks for video comprehension (section 5.3.2). The procedure requires a limited set of manual annotations, independent of the size of the dataset. This renders our approach applicable to build benchmarks of unprecedented scale.

• We present an implementation of this procedure, which we used to create a video comprehension benchmark of size comparable to the biggest existing datasets (section 5.3.3).

• We introduce and assess a varied set of baselines and methods to tackle the problem of video comprehension on our benchmark (section 5.4).

Figure 5.1: Video comprehension example. Video comprehension requires choosing one of \( k \) possible English sentences as the description for an input video clip. In this example the correct answer is (D).
5.2 Related Work

Video understanding has been studied for many years. Early approaches focused on action recognition [22, 5, 92], event detection [48], irregularity detection [7], action similarity labeling [53]. Most of these methods rely on hand-crafted features [62, 113] and train machine learning models on top of these representations. Recent advances in deep learning have opened up the possibility of learning models from raw videos. Simonyan and Zisserman introduced a two-stream network that achieved strong results on action recognition [90]. Tran et. al. proposed to use 3D ConvNets to learn spatiotemporal features from a large-scale dataset [100]. Despite their good performance on action categorization, these approaches are by design limited to predict a single label per video and thus are not really addressing a semantic understanding of the video.

Inspired by recent promising results on image captioning [13], different approaches have been proposed for video description [109, 98]. These methods are based on recurrent neural networks (e.g., LSTM) and are trained to predict a single sentence to describe the input video. This area shows good promise for developing algorithms that can understand and describe videos in a human-readable language. However, captioning is very hard to assess. This limitation makes it hard to compare competing algorithms, even when resorting to human judges. Visual question and answer (QA) was also recently introduced for both images [1] and videos [95]. Compared to captioning, Visual QA is better-posed, as the problem is conditioned on the question being asked. However, in the free-form QA setting, there are still multiple correct answers that can be correctly applied to a single question. Moreover, collecting ground truth annotations for QA is very expensive. This makes it hard to build large-scale
5.3 Video Comprehension

datasets on this task. Our video comprehension problem shares similarities with video
captioning and QA, as our task also assesses algorithms on their ability to understand
the semantics of the video. However, our task is different in its formulation: it entails
selecting one of the $k$ possible sentences from a multiple choice test, rather than asking
to describe or to answer a particular question. This renders the quantitative evaluation
easy to carry out and makes performance scores very intuitive. Our work relates also
to [46, 24] in terms of the aim at building large-scale video datasets. However, our
dataset is purposely constructed for video comprehension while the aforementioned
ones are for classification and detection.

5.3 Video Comprehension

5.3.1 Problem statement

Given an input video clip $V$ and a set of $k$ English sentences $s_1, s_2, \ldots, s_k$, the problem
of video comprehension is to predict which of these $k$ choices best describes the visual
content of the input clip. Note that readable text in the frames is automatically
blurred and audio is removed. A concrete example of video comprehension is provided
in Figure 5.1. We argue that for a system to do well on this task it must be able to
infer the true semantics of the video, including context, the nature of the interactions
among the subjects, and the objects appearing in the scene. Thus, we believe this to
be a more fitting assessment of video comprehension by machines than the tasks of
action recognition or video captioning, explored in prior benchmarks.
5.3 Video Comprehension

5.3.2 Procedure for Constructing a Video Comprehension Dataset

In order to assess and compare methods on the task of video comprehension, a dataset must be constructed to enable the training and testing of models on this problem. Here we review the desiderata that inspired the construction design of our benchmark. Ideally, the dataset must be:

(a) Large-scale. In the still-image domain we have witnessed a dramatic revolution in methodology and breakthrough results with the introduction of a large-scale dataset. In fact, recent research has shown that the problem of overfitting and difficult optimization with deep models are vastly reduced when leveraging large datasets for training. Thus, our desired benchmark for video comprehension must be large enough to enable the training of these powerful models.

(b) Semi-automatic. The process of dataset construction must be semi-automatic and must require little human intervention. This is a fundamental requirement in order to be build a massive collection of examples. We note that the limited scale of prior datasets in the video domain is a direct consequence of the high human cost and time consumption needed to label video clips.

(c) Semantically diverse. As our objective is to train universal models that can comprehend video of arbitrary nature, the training and testing sets must contain a wide representation of subjects, including politics, sports, science, technology, arts, and travel.

To meet this criteria, we design a procedure that generates semi-automatically video comprehension tests (as shown in Figure 5.1) by leveraging an existing gigantic
5.3 Video Comprehension

repository of TV news programs – the TV News Archive. We note that access to the Archive’s Collections is granted at no cost for scholarship and research purposes. Thus it represents a fitting platform for the construction of video benchmarks. Furthermore, as TV news cover all social, cultural, and natural aspects of modern life, the collection is inherently semantically diverse. Finally, the videos have accurate associated time-synchronized English captions providing a well-aligned textual transcription of the audio (the TV News Archive uses sphinx and phonemes to align the timing of the broadcasted CC with video). We utilize these closed captions (CCs) to automatically generate the textual descriptions corresponding to the ground truth labels of the videos. This source of information allows our procedure to generate a massive collection of comprehension tests with ground truth labels almost fully automatically (as further explained below, a small set of initial human annotations are needed to bootstrap the process).

In this section we discuss in detail the construction of the dataset. Each video downloaded from the Archive is a complete TV news show from a particular channel recorded and broadcasted on a specific day (e.g. ABC News Good Morning America on August 27, 2011 from 8am to 9am) with lengths varying from 30 minutes to 2 hours. Our procedure then performs a sequence of steps aimed at generating a set of video comprehension tests from each program. The steps include clip segmentation, clip elimination, and multiple-choice test generation.

**Clip segmentation.** Each TV news video is segmented into short clips, corresponding to individual sentences (terminated by a period) of the closed captions. For each such sentence, using the time stamps of its beginning and ending, we segment the corresponding clip from the video. This process yields a massive number of clips.
from each video. In order to build a dataset of clips having fairly homogeneous length and to have enough temporal context in each clip, we eliminate clips that are shorter than 2 seconds or longer than 5 seconds. Similarly, we discard clips corresponding to sentences that are either too short (5 words or fewer) or too long (more than 60 words).

Clip elimination. This step is carried out to remove clips whose visual content is not informative. Examples include advertisement, static scenes, segments showing anchors speaking, sections inside the news studio, such as the weather forecast portion of the news program. Such clips are not useful for training general computer vision models. In order to make our dataset construction scalable, we develop a detector to automatically discard irrelevant clips. The detector is trained on a small collection of clips manually labeled as either irrelevant (e.g. studio, advertisement, weather-forecast clips) or relevant (out-of-studio footage, such as dynamic scenes where human subjects or the camera are moving). The detector is trained on the visual component of each clip (thus, without considering its CC). The details of our detector are discussed in section 5.3.3.

Multiple-choice test generation. Given a video clip, we form a multiple-choice test of $k$ potential textual descriptions by including $k−1$ distractors and the true associated CC sentence. The distractors can be selected in different ways. The simplest solution is to randomly sample the $k−1$ distractors from the entire set of CC sentences. In order to make the test more challenging, one may want to select distractors that are not too distant from the correct response, according to a semantic metric over text descriptions, such as the Euclidean distance of word2vec vectors representing sentences [73].
5.3 Video Comprehension

![ROC curve for relevant clip detection](image)

Figure 5.2: **Relevant clip detection.** The ROC curve of relevant clip detector evaluated on the validation set. At the false positive rate of 0.1, the true positive rate is 0.83.

5.3.3 The ViCom Dataset

In this section we discuss a specific implementation of the general procedure outlined above. This implementation was used to construct a dataset of 310,216 multiple-choice video comprehension tests, which we will make publicly available to the research community. The benchmark is split into 218,331 training examples and 91,885 test examples. We name our dataset **ViCom.**

The dataset is constructed from 4,990 news videos from the TV News Archive. These videos were obtained by considering 77 distinct TV news shows (BBC World News, MSNBC News Live, PBS News Hour, etc.). In order to yield a dataset with heterogenous news content, we sampled (roughly uniformly) each of this daily news shows in the period from January 1, 2009 to December 31, 2014.

We use a subset of 20 news videos (randomly selected from our original 4,990 TV videos) as a training set exclusively for the development of our relevant/irrelevant clip detector. Note that we remove these 20 videos from the collection used for dataset construction. We manually labeled all clips segmented from this set as either irrelevant.
5.3 Video Comprehension

![Figure 5.3: ViCom topic distribution. The subject distribution was estimated by training an LDA [6] model with 10 topics on closed caption sentences of ViCom clips.](image)

(e.g. studio, advertisement, weather-forecast clips) or relevant. We represent each clip using the C3D spatiotemporal features [100], which are activations of a convolutional neural network (ConvNet) optimized for action classification. We use the activations from layer \(fc6\). We opted for this descriptor as it has been shown by the authors to yield good performance on a variety of tasks involving semantic analysis of video. We train a simple a linear SVM on this representation to classify whether a clip is relevant or not. We evaluate this detector on our training set of 20 videos using 20-fold cross validation. The resulting ROC curve is shown in Figure 5.2. The detector achieves an area under the curve (AUC) of 0.94. We use this ROC curve to choose the cutoff threshold to reject irrelevant clips. We chose the threshold corresponding to a false positive rate of 0.1, which yields a true positive rate rate of 0.83. This represents a good trade-off in terms of recall vs error (in other words, to retrieve 83% of the relevant clips we must cope with only 10% of irrelevant clips). Further filtering of these clips could be performed via crowdsourcing at a fairly limited financial cost. However, our experiments suggest that a 10% of irrelevant clips (i.e., clips whose visual content is not strongly correlated their associated CC sentence) in our dataset
5.3 Video Comprehension

does not prevent the training of effective models for video comprehension but it offers the big benefit of a semi-automatic solution.

Applying this detector to the remaining 4970 videos yields a total of 310K clips deemed relevant for the purpose of video comprehension. We partition this dataset into training and testing splits (using a ratio of 7:3), with the additional constraint that all clips from a video are inserted in the same split (either training or testing). This is done as clips from the same video are often strongly correlated and this would bias the statistical assessment.

Table 5.1 compares ViCom to existing video datasets in terms of size, task, as well as video content. ViCom is the second biggest dataset in term of both number of total hours and number of clips. However, while in Sports-1M and ActivityNet (the largest datasets in this comparison) each clip is labeled with an action class, each clip of ViCom is labeled with a textual description that typically provides a semantically richer annotation than an action tag. Some examples of CC sentences associated to video clips are shown in the supplementary material. Furthermore, we point out that this only represents the first version of our benchmark. Our plan is to scale the dataset to much larger sizes in the near future. The little reliance on human intervention renders this plan easy to implement.

In order to understand the distribution of subject matters represented in ViCom clips, we trained an LDA [6] model on the CC sentences of our entire training set using 10 topics. We visually inspected the most frequent words of each topic in order to manually assign a subject tag to each topic (“politics”, “economics”, “technology”, etc). The most frequent words are listed in a table included in the supplementary material. Figure 5.3 shows the subject distribution computed on the 310K clips of
5.4 Approaches to Video Comprehension

<table>
<thead>
<tr>
<th>Dataset</th>
<th>UCF101</th>
<th>Sports1M</th>
<th>ActivityNet</th>
<th>MSVD</th>
<th>MPII-MD</th>
<th>M-VAD</th>
<th>MovieQA</th>
<th>ViCom</th>
</tr>
</thead>
<tbody>
<tr>
<td># clips</td>
<td>13K</td>
<td>1.1M</td>
<td>28K</td>
<td>2K</td>
<td>68K</td>
<td>49K</td>
<td>400</td>
<td>310K</td>
</tr>
<tr>
<td># hours</td>
<td>24</td>
<td>N/A</td>
<td>648</td>
<td>5</td>
<td>74</td>
<td>85</td>
<td>N/A</td>
<td>628</td>
</tr>
<tr>
<td>Task</td>
<td>cls</td>
<td>cls</td>
<td>cls/det</td>
<td>des</td>
<td>des</td>
<td>des</td>
<td>qa</td>
<td>vicom</td>
</tr>
<tr>
<td>Type</td>
<td>action</td>
<td>sport</td>
<td>action</td>
<td>movie</td>
<td>movie</td>
<td>movie</td>
<td>movie</td>
<td>news</td>
</tr>
</tbody>
</table>

Table 5.1: **Dataset comparison.** Comparison of different video datasets in term of size, task, and video type. Abbreviations for tasks: **cls** (classification), **det** (detection), **des** (description), **qa** (question and answer), and **vicom** (video comprehension). N/A: information is not available in [46, 95].

ViCom. It can be seen that the breadth of topics covered in ViCom distinguishes our dataset from prior video collections, which are much more focused in content (e.g., videos depicting only sports or movies). This makes ViCom particularly fitting for the training of models for general and comprehensive video understanding.

For each clip (in both the training and the testing split) we formed a multiple-choice test by randomly selecting 4 distractor sentences from our entire pool of CC sentences, in addition to the correct answer (the true CC). Thus, each test includes 5 sentences from which the correct one must be chosen.

5.4 Approaches to Video Comprehension

In this section, we consider different approaches to tackle the task of video comprehension on our ViCom dataset. For clarity, we first introduce our notation. Let us denote the training set with \{x^i, y_1^i, y_2^i, \ldots, y_k^i, t^i\}_{1..n}, where \(x^i\) is the \(i\)-th video clip, \(y_1^i, y_2^i, \ldots, y_k^i\) are the \(k\) sentences defining the multiple choice test, and \(t^i \in [1..k]\) is the answer key, i.e., the index to the correct answer. Let \(\phi_v(x)\) be a visual embedding (i.e., a feature representation computed from pixel values) of video clip \(x\) and \(\phi_l(y)\) the language embedding of the text sentence \(y\). Examples of possible choices for
5.4 Approaches to Video Comprehension

the visual embedding include aggregations of deep image features computed from individual frames of the clip (e.g., average pooling of VGG activations [91]) or deep video clip descriptors (e.g., C3D fc6 activations [100]). The language embedding can be produced by averaging the word2vec representation [73] of all words in the sentence.

5.4.1 Regression

A simple strategy to video comprehension is to train a regression model \( \mathcal{R}(x; W) \) parameterized by weights \( W \) to map from the the visual embedding to the language embedding, i.e., such that \( \mathcal{R}(x^i; W) \approx \phi_l(y^i_t) \). A simple instantiation of this method would consists in learning a linear transformation of \( \phi_v(x) \), i.e., \( \mathcal{R}(x^i; W) = W \phi_v(x') \), where the parameter matrix \( W \) can be estimated via least-square regression. Predictions can then be made by choosing the sentence whose language descriptor is closest to the transformed visual descriptor, i.e., \( t^* = \arg\min_{t \in [1..k]} \| W \phi_v(x) - \phi_l(y_t) \|_2^2 \).

This proposed strategy can be made more powerful by replacing the linear regression model with a deep convolutional network \( \mathcal{R}(x; W) \) (here \( W \) denotes weights) that is trained directly on the raw video input \( x^i \) to regress the associated CC language embedding vector \( \phi_l(y^i_t) \). After training, we make predictions by choosing \( t^* = \arg\min_{t \in [1..k]} \| \mathcal{R}(x; W) - \phi_l(y_t) \|_2^2 \). We explore both of these proposed models in our experiments.

5.4.2 Metric learning

It can be argued that the regression strategy outlined above is overly aggressive as it forces the visual vectors to be mapped into their language counterparts. This objective is difficult to realize. We can relax this desideratum by stating that the visual vector
5.4 Approaches to Video Comprehension

mapped to the language embedding should merely be closer to the correct answer than to any of the distractors. This can be achieved by learning a mapping $\mathcal{M}$ that projects a raw video $x$ to the language embedding space $\phi_l(y)$ by minimizing the triplet metric learning loss used in [85], i.e.,:

$$W^* = \arg\min_W \sum_i^n \sum_{t \neq t^i} \left[ \| \mathcal{M}(x^i; W) - \phi_l(y^i_t) \|_2^2 - \| \mathcal{M}(x^i; W) - \phi_l(y^t_t) \|_2^2 + \alpha \right]_+.$$  

(5.1)

$\mathcal{M}(x; W)$ is a mapping with parameters $W$. $\mathcal{M}(x; W)$ can be a deep ConvNet trained on raw input video $x$. In a simpler case, it simply takes a predefined $\phi_v(x)$ as input and learns a simple linear projection. $[.]_+$ is the hinge function. Finally, $\alpha$ is a hyper-parameter to control the margin between the distance to the true sentence $y^i_t$ and the distances to the wrong sentences $y^t_t$ (with $t \neq t^i$). After training, this approach also makes predictions by searching the nearest neighbor in the language embedding space $t^* = \arg\min_{t \in [1..k]} \| \mathcal{M}(x; W^*) - \phi_l(y_t) \|_2^2$.

5.4.3 Video captioning

An alternative approach consists in generating a textual description $y = \mathcal{C}(x)$ by running an existing video captioning model $\mathcal{C}(x)$ (e.g. S2VT [109]) on the input video clip. Effectively this strategy maps the input clip $x$ into an English sentence. The resulting textual description is then embedded into the language space to identify the closest answer: $t^* = \arg\min_{t \in [1..k]} \| \phi_l(\mathcal{C}(x)) - \phi_l(y_t) \|_2^2.$
5.5 Experiments

5.5.1 Experimental setup

**Language embedding**: In all of our experiments, we use \texttt{word2vec} \cite{mikolov2013efficient} as the language embedding $\phi_l(y)$. \texttt{word2vec} is a shallow neural network trained on a large-corpus to reconstruct the linguistic context of the words. It is used as a word embedding which maps words that share similar contexts into vectors that are close (in distance). We use the \texttt{word2vec} model provided by \cite{mikolov2013efficient} which is pre-trained on the Google News dataset. This gives a 300-dimensional vector representation for each word. To represent $\phi_l(y)$, we extract \texttt{word2vec} vectors for all words in $y$, average these vectors, then L2-normalize the averaged vector to build a language representation for the sentence $y$.

**Visual embedding**: We use different visual representations for $\phi_v(x)$ in different experiments. These representations are computed from different ConvNet architectures pre-trained on different datasets. We input our frames (or the entire clip in the case of C3D) into these pre-trained ConvNets to extract activations of a particular layer and use them as representations. We specify a visual representation by a pair of an architecture name and a layer name. We use the AlexNet \cite{krizhevsky2012imagenet} implemented in \cite{szegedy2015going}, the
5.5 Experiments

VGG architecture \[91\], and C3D \[100\]. The pre-trained models are provided by the authors of \[44, 91, 100\]. For simplicity, from now on we denote these representations as AlexNet, VGG, and C3D, respectively. It is worth noting that most example clips in ViCom have varying length (a few dozens to few hundreds frames), while AlexNet and VGG operates on frames, and C3D uses short clips of 16 frames. We average the frame (or 16-frame clip) features, and then L2-normalize the averaged vector to make visual representations for the long clips.

**Regression models:** We experimented with linear regression applied to C3D-\(fc6\) and AlexNet-\(fc6\). We call these two approaches LR-C3D and LR-Alex, respectively. We also use a 3D ConvNet as a deep regressor. We choose to use an architecture similar to that of C3D. The network layers are identical to those of C3D up to \(fc6\). We then add a fully-connected layer with linear activation and 300 output units (effectively a linear projection). We optimize this model with a regression loss. Because this architecture is similar to C3D, we have the option either to train from scratch or to initialize the bottom layers from C3D. We name these two methods DR-C3D-0 and DR-C3D-FT.

**Captioning models:** We use the S2VT pre-trained model provided by \[109\] which is trained on MSVD \[12\]. S2VT is a 2-hidden-layer LSTM which takes a sequence of VGG-\(fc7\) frame features as input and predicts a sentence. We name this approach S2VT-MS. We note that the language used in TV news is very different from the sentences in MSVD. In order to allow the captioning method to adapt to the news language, we also trained S2VT on ViCom. We name this method S2VT-Vi.

**Metric learning models:** We experiment with two different sets of architectures: shallow and deep networks. In the shallow network setting, we assume that we have a
5.5 Experiments

reasonable good visual representation, and we just learn a single fully-connected layer without nonlinear unit. We optimize this model by the triplet loss (as described in section 5.4.2). We test the shallow metric learning with three different representations: Alex-fc6, VGG-fc6, and C3D-fc6. We name these approaches SML-Alex, SML-VGG, and SML-C3D, respectively. We also test this shallow net applied to a combined representation of Alex-fc6, VGG-fc6, and C3D-fc6 (a simple concatenation). We name this approach SML-Com. For the deep network setting, we use again an architecture similar to C3D. We use all layers identical to those of C3D up tp fc6. We then add a linear fully-connected layer with 300 output units. We can either train this network from scratch or finetune it from C3D. We name these approaches DML-C3D-0 and DML-C3D-FT, respectively.

Training settings: Both shallow and deep networks are trained using SGD with a momentum of 0.9. For shallow networks, we use a mini-batch size of 128. The initial learning rate is 0.01 and it is reduced by a factor of 0.1 every 10K iterations. Training is stopped at 60K iterations. For deep networks, we use a mini-batch size of 30. The initial learning rates are $3 \times 10^{-4}$ and $3 \times 10^{-5}$ for DML-C3D-0 and DML-C3D-FT, respectively. They are reduced by 0.1 for every 100K iterations and the training is stopped at 600K iterations. Since training deep networks is time-consuming, we choose $\alpha$ by cross validation on shallow networks. Our experiments show that using $\alpha = 0.1$ gives the best results among the tested margins of 0.01, 0.1, and 1 for all visual features features. Thus, we use $\alpha = 0.1$ in all deep metric learning networks.
5.5 Experiments

5.5.2 Experimental results

**Video comprehension**: Table 5.2 presents the accuracy of different approaches on ViCom. Among the regression methods, LR-C3D performs the best, but is just 5.3% better than random chance. Deep regression approaches perform poorly because training regression in high-dimensional space is difficult. S2VT-MS gives a very low accuracy which is only slightly above random chance. We observe that this method generates very simple sentences because it was trained on simple description sentences of MSVD. S2VT-Vi, trained on ViCom, generates sentences more similar to those in the news reports. Figure 5.4 shows some generated captions of these two methods compared to the ground truth. However, the performance of S2VT-Vi is still low which indicates that video comprehension is a different task compared to video description. Shallow metric learning methods perform reasonably well on various visual representations, and SML-C3D gives the best performance (53.5%) among the shallow networks with a single representation. Combining the three visual representations boosts the accuracy to 54.4%. The deep network DML-C3D-FT gives the highest accuracy (55.5%).

**Human study**: To better understand the challenges posed by the task, we also conducted a human study. For this study, a subset of 200 clips were randomly drawn from the test split. Each clip and its associated multiple-choice test were shown (without audio) to 5 human annotators. We asked them to select the sentence best describing the video clip out of the $k = 5$ choices. Because TV news often contain tickers with informative text, we trained a HOG-based SVM text detector to detect and blur the text tickers. We performed two human study experiments: the first one is applied on the original 200 clips while the second one is applied on the same
### 5.5 Experiments

<table>
<thead>
<tr>
<th>Approach</th>
<th>Name</th>
<th>Model</th>
<th>Features</th>
<th>Training scheme</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>Regression</td>
<td>LR-Alex</td>
<td>LR</td>
<td>Alex−fc6</td>
<td>-</td>
<td>22.9</td>
</tr>
<tr>
<td></td>
<td>LR-C3D</td>
<td>LR</td>
<td>C3D−fc6</td>
<td>-</td>
<td>25.3</td>
</tr>
<tr>
<td></td>
<td>DR-C3D-0</td>
<td>DR</td>
<td>-</td>
<td>from scratch</td>
<td>21.9</td>
</tr>
<tr>
<td></td>
<td>DR-C3D-FT</td>
<td>DR</td>
<td>-</td>
<td>finetuned</td>
<td>21.6</td>
</tr>
<tr>
<td>Description</td>
<td>S2VT-MS</td>
<td>S2VT</td>
<td>VGG−fc7</td>
<td>train on MSVD</td>
<td>20.9</td>
</tr>
<tr>
<td></td>
<td>S2VT-Vi</td>
<td>S2VT</td>
<td>VGG−fc7</td>
<td>train on ViCom</td>
<td>21.2</td>
</tr>
<tr>
<td>Metric learning</td>
<td>SML-Alex</td>
<td>SML</td>
<td>Alex−fc6</td>
<td>-</td>
<td>39.8</td>
</tr>
<tr>
<td></td>
<td>SML-VGG</td>
<td>SML</td>
<td>VGG−fc6</td>
<td>-</td>
<td>52.7</td>
</tr>
<tr>
<td></td>
<td>SML-C3D</td>
<td>SML</td>
<td>C3D−fc6</td>
<td>-</td>
<td>53.5</td>
</tr>
<tr>
<td></td>
<td>SML-Com</td>
<td>SML</td>
<td>Combined−fc6</td>
<td>-</td>
<td>54.5</td>
</tr>
<tr>
<td></td>
<td>DML-C3D-0</td>
<td>DML</td>
<td>-</td>
<td>from scratch</td>
<td>34.4</td>
</tr>
<tr>
<td></td>
<td>DML-C3D-FT</td>
<td>DML</td>
<td>-</td>
<td>finetuned</td>
<td>55.5</td>
</tr>
<tr>
<td>Human-0-ticker</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>66.5</td>
</tr>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>78.8</td>
</tr>
</tbody>
</table>

Table 5.2: **Video comprehension accuracy on ViCom**. Accuracy of different approaches on ViCom compared with random chance and human performance. Methods based on deep metric learning give the best performance (the best gives 55.5%). Random chance is at 20%, while humans achieve an accuracy of 78.8%.

200 clips but with tickers detected and blurred. We denote these two experiments as **Human** and **Human-0-ticker**, respectively. We note that the 5 annotators involved in the first experiment are completely different from the 5 annotators in the second one. The first experiment answers how well humans can comprehend videos provided all information (e.g. text and visual) while the second one tells us the human-level performance on video comprehension given only visual inputs. Results from these experiments show that humans achieve an accuracy of 78.8% ± 1.35 when provided the original clips and 66.5% ± 5.6 when provided only visual inputs. This indicates that video comprehension is a hard task and that our best approach, DML-C3D-FT, is still 23.3% below human-level video comprehension.

We present in Table 5.3 comprehension accuracy by topic for different approaches...
5.6 Conclusions

and humans. Humans outperform machines across all topics except for “Technology” where DML-C3D-FT does better than humans. DML-C3D-FT performs reasonably well across all topics while LR-C3D and S2VT-Vi perform worse than random chance on 2 and 7 topics, respectively, out of a total of 10 topics.

5.6 Conclusions

In this chapter we introduced a new high-level video understanding task, we presented a general procedure to construct semi-automatically benchmarks for this task, we created a dataset (ViCom) that we plan to release to the community and we evaluated a series of approaches on it. ViCom fulfills the following desirable properties: 1) it defines a well-posed task with a good quantitative evaluation metric; 2) it assesses the ability to semantically comprehend video; 3) it is large-scale, thus enabling effective training of deep models. We hope that this new task and our benchmark will become important stepping stones to fundamentally transform video analysis into higher-level video understanding. We have seen this happening in the image domain where a new large-scale benchmark [82] married with a powerful machine learning model [56] gave rise to a new generation of computer vision algorithms. We also expect that our

<table>
<thead>
<tr>
<th>Topic</th>
<th>Politics</th>
<th>Climate</th>
<th>Election</th>
<th>Time</th>
<th>Misc</th>
<th>Tech</th>
<th>Legal</th>
<th>Economics</th>
<th>Crime</th>
<th>Emotion</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR-C3D</td>
<td>30.3</td>
<td>27.4</td>
<td>27.5</td>
<td>19.8</td>
<td>27.3</td>
<td>23.0</td>
<td>26.7</td>
<td>27.5</td>
<td>27.4</td>
<td>18.3</td>
<td>25.3</td>
</tr>
<tr>
<td>S2VT-Vi</td>
<td>18.9</td>
<td>17.0</td>
<td>20.8</td>
<td>18.8</td>
<td>29.7</td>
<td>19.3</td>
<td>18.3</td>
<td>18.4</td>
<td>17.7</td>
<td>24.6</td>
<td>21.2</td>
</tr>
<tr>
<td>DML-C3D-FT</td>
<td>53.6</td>
<td>65.6</td>
<td>59.9</td>
<td>42.7</td>
<td>48.9</td>
<td>63.3</td>
<td>46.1</td>
<td>54.5</td>
<td>65.9</td>
<td>48.3</td>
<td>55.5</td>
</tr>
<tr>
<td>Human-0-ticker</td>
<td>60.0</td>
<td>79.2</td>
<td>78.9</td>
<td>68.0</td>
<td>66.7</td>
<td>30.0</td>
<td>64.0</td>
<td>66.0</td>
<td>72.0</td>
<td>53.3</td>
<td>66.5</td>
</tr>
<tr>
<td>Human</td>
<td>87.7</td>
<td>95.0</td>
<td>88.9</td>
<td>78.0</td>
<td>73.9</td>
<td>40.0</td>
<td>85.0</td>
<td>76.0</td>
<td>80.0</td>
<td>61.3</td>
<td>78.8</td>
</tr>
</tbody>
</table>

Table 5.3: Video comprehension accuracy details. Performance by topic of different methods compared with human performance. Humans achieve the highest accuracy across all topics except for ‘Technology’ where DML-C3D-FT outperforms humans. LR-C3D performs below random chance on 2 out of 10 topics, and S2VT-Vi has 7 out of 10 topic are below random chance (in underlined text).
5.7 Additional Experiments and Details

benchmark will spur active research at the intersection between video understanding and natural language processing.

Most of our future work will be devoted to scaling up the size of ViCom. As our dataset construction is semi-automatic we believe that it will be possible to scale up ViCom quickly to a much larger benchmark with little human, computational and financial cost. We expect to increase the dataset by an order of magnitude within the next year. In order to stimulate steady progress in this area, we plan to organize a series of grand challenges built around our benchmark. We will release the ViCom dataset, all implementations and models upon publication of this article.

5.7 Additional Experiments and Details

5.7.1 ViCom topics and examples

In the previous section we presented an experiment where we use LDA [6] to model topics of ViCom sentences using 10 topics. The top words for each topic are presented in Table 5.4.

Figure 5.5 presents some examples from ViCom dataset with their corresponding ground truth sentences.

5.7.2 t-SNE embedding of ViCom using different features

We compare the semantic embedding learned by DML-C3D-FT with that of C3D with respect to ViCom topics. We randomly select 10,000 clips from our ViCom test split. For visualization purpose, we only pick 5 topics. We use t-SNE [107] to project the two comparing embedding: C3D (fc6) and DML-C3D-FT (fc6) into 2-dimensional spaces.
5.7 Additional Experiments and Details

Bee sting therapy is gaining popularity in parts of the world.

Pope Francis has held private talks at the Vatican with Russian President Vladimir Putin.

600 flights canceled, as much as three feet of snow with five-foot drifts.

And it kept spinning into the distance, completely out of control.

He spent two days drifting before Japanese troops were able to rescue him yesterday.

Figure 5.5: **Examples from ViCom.** Some example clips from the ViCom dataset with their true closed caption sentences. It can be noted that they cover a wide range of subjects ranging from environmental events, to science, politics, and accidents.
5.7 Additional Experiments and Details

<table>
<thead>
<tr>
<th>Topic</th>
<th>Top words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Politics</td>
<td>government, problem, country, american, war, military, protest, attack,</td>
</tr>
<tr>
<td></td>
<td>unit, security, question, force, leader, nation, call</td>
</tr>
<tr>
<td>Climate</td>
<td>area, city, storm, fire, hour, water, north, across, air, mile, center,</td>
</tr>
<tr>
<td></td>
<td>snow, force, south, weather, power, rain, thousand, hit, through</td>
</tr>
<tr>
<td>Election</td>
<td>house, republican, big, obama, romney, white, senate, democrat, vote, last,</td>
</tr>
<tr>
<td></td>
<td>campaign, election, party, game, mitt, governor, win, night, race, poll</td>
</tr>
<tr>
<td>Time</td>
<td>next, tonight, story, world, hour, weekend, around, few, numberth, week,</td>
</tr>
<tr>
<td></td>
<td>ahead, show, begin, america, chuck, start, stay, york, daily</td>
</tr>
<tr>
<td>Technology</td>
<td>san, kill, old, man, francisco, west, future, police, cover, bloomberg,</td>
</tr>
<tr>
<td></td>
<td>shot, business, technology, pier, welcome, men, hospital</td>
</tr>
<tr>
<td>Legal</td>
<td>case, court, call, charge, investigate, police, law, response, release,</td>
</tr>
<tr>
<td></td>
<td>decision, office, official, former, action, death, against, depart, defense</td>
</tr>
<tr>
<td>Economics</td>
<td>dollar, million, care, job, health, cut, tax, plan, than, paid, money,</td>
</tr>
<tr>
<td></td>
<td>program, billion, government, american, company, announcement, raise,</td>
</tr>
<tr>
<td></td>
<td>develop</td>
</tr>
<tr>
<td>Crime</td>
<td>close, car, police, street, off, school, fire, video, inside, scene,</td>
</tr>
<tr>
<td></td>
<td>show, build, last, crash, worker, open, wall, park, home, office</td>
</tr>
<tr>
<td>Emotion</td>
<td>thank, much, little, let, join, learn, way, washington, stephanie, love,</td>
</tr>
<tr>
<td></td>
<td>read, early</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>thing, them, lot, put, because, really, work, got, happen, did, keep,</td>
</tr>
<tr>
<td></td>
<td>these, very, something, way, try, need, well, any</td>
</tr>
</tbody>
</table>

Table 5.4: ViCom topics. Topic modeling of ViCom sentences using LDA [6] with 10 topics. The topic names are picked by authors based on the sharing semantic among of the top words.

Figure 5.6 visualizes these results embedding. In the figure, each dot is a clip projected in 2-dimensional space and colored according to its topic label. We quantitatively verify that DML-C3D-FT (fc6) are more semantically clustered according ViCom topic labels. This indicates that DML-C3D-FT learns a meaningful metric and also explains why it performs much better than LR-C3D. For a complete view, we also present t-SNE embedding of word2vec with respect to ViCom topics.

5.7.3 Long sentence modeling

We study if modeling sentences in a holistic manner improves video comprehension performance. For this purpose, we use skip-thought vectors [49] as an alternative to our language embedding. skip-thought can be considered as a whole sentence-to-vector embedding which maps a sentence to a vector of 4,800 dimensions. We use the shallow metric learning networks (SML-Alex, SML-VGG, SML-C3D, and SML-
5.7 Additional Experiments and Details

Figure 5.6: **ViCom topics on different embedding.** Semantic embedding of `word2vec`, `C3D (fc6)`, and `DML-C3D-FT(fc6)` on a random subset of 10,000 ViCom test clips. Each dot is a clip representation (language or visual) vector projected into 2 dimension and is labeled (colored) by the topic.

<table>
<thead>
<tr>
<th>$\phi_l$ option</th>
<th>SML-Alex</th>
<th>SML-VGG</th>
<th>SML-C3D</th>
<th>SML-Com</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>word2vec</code></td>
<td>39.8</td>
<td>52.7</td>
<td>53.5</td>
<td>54.5</td>
</tr>
<tr>
<td><code>skip-thought</code></td>
<td>37.3</td>
<td>47.2</td>
<td>47.6</td>
<td>48.4</td>
</tr>
</tbody>
</table>

Table 5.5: **Experiments with Skip-thought Vectors.** Replacing `word2vec` by `skip-thought` vectors for the language representation consistently degrades the performance across all different visual embedding on shallow metric learning networks.

Com as described previously) with the only difference in which we replace `word2vec` by `skip-thought`. Table 5.5 shows the performance of our shallow metric learning networks on ViCom using `skip-thought` vectors compared with their corresponding networks using `word2vec`. This replacement consistently degrades the accuracy about 2-6% across different feature representations.

5.7.4 S2VT predictions

Additional sample predictions from S2VT-MS and S2VT-Vi are shown in Figure 5.4.
5.7 Additional Experiments and Details

Figure 5.7: **S2VT predictions.** More prediction samples from S2VT. The sentences generated by S2VT-MS and S2VT-Vi, and the CC ground truth. S2VT-MS predicts simple sentences. S2VT-Vi is well adapted to news language and predicts longer sentences.
Chapter 6

Conclusions

In this thesis, we have proposed two approaches for video representations: EXMOVES and C3D. EXMOVES is a mid-level video representation specially designed for large-scale applications. EXMOVES can be useful when training data is limited. When large-scale training data is available, C3D can be applied to learn generic spatiotemporal features for videos. C3D is accurate, compact, and efficient to compute, thus well-suited for large-scale applications. We have also presented a method for dense video voxel labeling. Our results show that dense video voxel labeling is feasible and the same architecture can be applied for different labeling tasks. Finally, we have proposed a new task and benchmark for video comprehension and provided a fundamental set of baselines and a human study for video comprehension. We hope that the new task and benchmark will help to foster further research in video understanding.
Bibliography


BIBLIOGRAPHY


