ABSTRACT

Title of dissertation: VIDEO UNDERSTANDING WITH DEEP NETWORKS

Joe Yue-Hei Ng
Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

Video understanding is one of the fundamental problems in computer vision. Videos provide more information to the image recognition task by adding a temporal component through which motion and other information can be additionally used. Encouraged by the success of deep convolutional neural networks (CNNs) on image classification, we extend the deep convolutional networks to video understanding by modeling both spatial and temporal information.

To effectively utilize deep networks, we need a comprehensive understanding of convolutional neural networks. We first study the network on the domain of image retrieval. We show that for instance-level image retrieval, lower layers often perform better than the last layers in convolutional neural networks. We present an approach for extracting convolutional features from different layers of the networks and adopt VLAD encoding to encode features into a single vector for each image. Our work provides guidance for transferring deep convolutional networks to other tasks.

We then propose and evaluate several deep neural network architectures to combine image information across a video over longer time periods than previously attempted. We propose two methods capable of handling full length videos. The first method explores various convolutional temporal feature pooling architectures, examining the various design choices which need to be made when adapting a CNN for this task. The second proposed method explicitly models the video as an ordered sequence of frames. For this purpose we employ a recurrent neural network that uses Long Short-Term Memory (LSTM) cells which are connected to the output of the underlying CNN.

Next, we propose a multitask learning model ActionFlowNet to train a single stream network directly from raw pixels to jointly estimate optical flow while recognizing actions with convolutional neural networks, capturing both appearance and motion in a single model. Experiments show that our model effectively learns video representation from motion information on unlabeled videos.

While recent deep models for videos show improvement by incorporating optical flow or aggregating high-level appearance across frames, they focus on modeling either the long-term temporal relations or short-term motion. We propose Temporal Difference Networks (TDN) that model both long-term relations and short-term motion from videos. We leverage a simple but effective motion representation: difference of CNN features in our network and jointly modeling the motion at multiple scales in a single CNN.

VIDEO UNDERSTANDING WITH DEEP NETWORKS

by

Joe Yue-Hei Ng

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor David Jacobs
Professor Hector Corrada Bravo
Professor Ramani Duraiswami

Dedication

To my dearest wife and parents.

Acknowledgments

Firstly, I would like to thank my advisor, Professor Larry Davis for giving me the invaluable opportunity to conduct research on computer vision, one of the most excitingly progressing fields in computer science during my graduate studies. He has given me great freedom to pursue the research direction I am interested in, and been patient in discussing research problems and editing drafts with my limited English skills. I am very fortunate to have him as my advisor. Secondly, I would like to express my gratitude to Professor Rama Chellappa, Professor David Jacobs, Professor Hector Corrada Bravo, and Professor Ramani Duraiswami for serving as my dissertation committee.

I am grateful to be able to work with the amazing colleagues in the computer vision lab: Fan, Ang, Stephen, Yaming, Muzi, Xiyang, Guangxiao, Brandyn, Hyung-tae, Sohil, Bharat, Zhe, Xintong, Zuxuan, Mahyar, Vlad and Zhuolin, all of whom provided invaluable help and feedback on my work. I would also like to acknowledge the support from the staff members of UMD CS, UMIACS, Deepthought2 cluster, and Bluecrab cluster.

I would like to thank my research internship hosts, especially Sudheendra Vijayanarasimhan for his guidance towards my first publication. I've learned a lot from all of my internship mentors: Paul Natsev from Google, Jonghyun Choi and Jan Neumann from Comcast Labs, Koen van de Sande and Cees Snoek from Qualcomm, and Caiming Xiong and Richard Socher from Salesforce.

I have to give my greatest thanks to my family: my mother and father who

raised me and encouraged me to pursue doctoral studies. I thank my wife for her tremendous support during the difficult times in my PhD studies.

It is impossible to remember all those who have helped me, and I apologize to those I've carelessly left out.

Lastly, thank you all and thank you God.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BoW	Bag-of-Word
CNN	Convolution Neural Network
FV	Fisher Vector
HOG	Histogram of Oriented Gradients
HOF	Histogram of Opitcal Flow
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LSTM	Long-short term memory
mAP	Mean Average Precision
MBH	Motion Boundary Histogram
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
ResNet	Residual Network
SSR	Signed Square Root
SIFT	Sclae-invariant Feature Transform
TDN	Temporal Difference Network
TSN	Temporal Segment Network
VLAD	Vector Locally Aggregated Descriptors

Chapter 1: Introduction

1.1 Motivation

Video understanding is one of the fundamental problems in computer vision. Videos provide more information to the image recognition task by adding a temporal component through which motion and other information can be additionally used. Encouraged by the success of deep convolutional neural networks (CNNs) on image classification, in this dissertation we extend the deep convolutional networks to video understanding by modeling both spatial and temporal information.

Traditionally, video action recognition research has been very successful at extracting local features from videos which encode local spatio-temporal patterns. Hand-crafted features such as Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), Motion Boundary Histogram (MBH) and trajectories are extracted from the videos [6, 7]. These local descriptors are then encoded to produce a global video-level feature representation with Bag-of-Word (BoW), VLAD, or Fisher vector encodings. By aggregating spatio-temporal local features to obtain global video representations, these approaches are able to obtain state-of-the-art results in a wide range of video recognition benchmarks.

Deep convolutional networks have shown great success in large scale image

classification [8]. By learning a hierarchy of feature representations through end-to-end optimization, CNNs give superior performance compared to traditional hand-crafted features. It shows great success when transferred to other related tasks such as object detection, semantic segmentation and image retrieval. Different improving network architectures have been proposed like AlexNet [8], VGG [4], Inception [9] and ResNet [10].

There have been several challenges on applying deep networks for video understanding. First, appropriate models are needed to learn spatial appearance and temporal information. Both short term motion and long term context are required to obtain full understanding of videos. Second, having one extra dimension compared to images, processing videos are computationally expensive. Efficient algorithms are needed to process large amounts of data.

In this dissertation, we study the problem of video classification with deep networks. We present multiple approaches for video action recognition which focuses on both aggregating long-term temporal information as well as capturing short-term motion and local apppearance.

1.2 Approaches

To effectively utilize deep networks, we need comprehensive understanding of convolutional neural networks. We first study the network on the domain of image retrieval. We then propose different network architectures for video classification. First, we study network architectures for full-length video classification. Second, we

introduce methods for learning short-term motion representation for video action recognition. Finally, we propose a framework to jointly model low-level motion and high-level temporal relations.

1.2.1 Exploiting Local Features from Deep Networks for Image Retrieval

Deep convolutional neural networks have been successfully applied to image classification tasks. When these same networks have been applied to image retrieval, the assumption has been made that the last layers would give the best performance, as they do in classification. We show that for instance-level image retrieval, lower layers often perform better than the last layers in convolutional neural networks. We present an approach for extracting convolutional features from different layers of the networks, and adopt VLAD encoding to encode features into a single vector for each image. We investigate the effect of different layers and scales of input images on the performance of convolutional features using the recent deep networks VGG-16 and GoogLeNet. Experiments demonstrate that intermediate layers or higher layers with finer scales produce better results for image retrieval, compared to the last layer. Our work provides guidance for transferring deep networks trained on image classification to other tasks.

1.2.2 Deep Networks for Full Length Video Classification

CNNs have been extensively applied for image recognition problems giving state-of-the-art results on recognition, detection, segmentation and retrieval. When extended to video classification, CNNs for video classification are limited to modeling short video clips, instead of reasoning on full length videos.

We propose and evaluate several deep neural network architectures to combine image information across a video over longer time periods than previously attempted. We propose two methods capable of handling full length videos. The first method explores various convolutional temporal feature pooling architectures, examining the various design choices which need to be made when adapting a CNN for this task. The second proposed method explicitly models the video as an ordered sequence of frames. For this purpose we employ a recurrent neural network that uses Long Short-Term Memory (LSTM) cells which are connected to the output of the underlying CNN.

1.2.3 Learning Motion Representation for Action Recognition

For action recognition, in addition to understanding the appearance such as objects and scenes, motion is a key component to fully understand the dynamics in a video. However, even with large amounts of labeled action classification data, convolutional networks are still ineffective in learning motion representation from raw pixels, which suggests that action classification supervision could be insufficient in learning motion in videos. In addition, large-scale labeled video datasets are

often difficult to collect, therefore it is essential that the model could learn motion representation with a small amount of data.

We present a data-efficient representation learning approach to learn video representation with small amounts of labeled data. We propose a multi-task learning model ActionFlowNet to train a single stream network directly from raw pixels to jointly estimate optical flow while recognizing actions with convolutional neural networks, capturing both appearance and motion in a single model. Our model effectively learns video representation from motion information on unlabeled videos to improve action recognition performance.

1.2.4 Temporal Difference Network for Action Recognition

With the help of the strong appearance models learned from large amount of image classification data, deep models significantly improve the performance of video recognition systems. While the deep models for videos show improvement by incorporating optical flow or aggregating high level appearance across frames, they focus on modeling either the long term temporal relations or short term motion. We propose Temporal Difference Networks (TDN) that model both long term relations and short term motion from videos. We leverage a simple but effective motion representation: difference of CNN features in our network and jointly modeling the motion at multiple scales in a single CNN. By taking multiple level of motions on CNN features, our network learns both low-level motion and high-level temporal relations.

1.3 Organization

This dissertation is organized as follows. Chapter 2 introduces the approach of feature extraction from deep convolutional networks for image retrieval. Chapter 3 presents deep network architectures for full length video classification. Chapter 4 proposes ActionFlowNet, a network architecture which jointly learns motion and appearance in a single network through multi-task learning for action recognition. Chapter 5 introduces the Temporal Difference Network (TDN) for learning motion and temporal relations in multiple appearance levels. Finally, in Chapter 6, we conclude and discuss future research directions.

Chapter 2: Exploiting Local Features from Deep Networks for Image Retrieval

2.1 Motivation

Image retrieval has been an active research topic for decades. Most existing approaches adopt low-level visual features, *i.e.*, SIFT descriptors, and encode them using bag-of-words (BoW), vector locally aggregated descriptors (VLAD) or Fisher vectors (FV) and their variants. Since SIFT descriptors capture local characteristics of objects, such as edges and corners, they are particularly suitable for matching local patterns of objects for instance-level image retrieval.

Recently, convolutional neural networks (CNNs) demonstrated excellent performance on image classification problems such as PASCAL VOC and ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [4, 8, 9, 11]. By training multiple layers of convolutional filters, CNNs are capable to automatically learn complex features for object recognition and achieve superior performance compared to handcrafted features. A few works have suggested that CNNs trained for image classification tasks can be adopted to extract generic features for other visual recognition tasks [12–14]. Although several approaches have applied CNNs to extract generic

features for image retrieval tasks and obtained promising results, a few questions still remain unaddressed. First, by default CNNs are trained for classification tasks, where features from the final layer (or higher layers) are usually used for decision because they capture more semantic features for category-level classification. However, local characteristics of objects at the instance level are not well preserved at higher levels. Therefore, it is questionable whether it is best to directly extract features from the final layer or higher layers for instance-level image retrieval, where different objects from the same category need to be separated. Second, most existing work assumes the size of a test image is the same as that of the training images. However, different scales of input images may affect the behavior of convolutional layers as images pass through the network. Only a few recent works attempt to investigate such effects on the performance of CNNs for image retrieval [15, 16].

In view of the power of low-level features (*i.e.*, SIFT) in preserving the local patterns of instances, and the success of CNN features in abstracting categorical information, we process CNN activations from lower to higher layers to construct a new feature for image retrieval by VLAD, although other encoding schemes can be readily applied. Recent deep networks VGG-16 and GoogLeNet pre-trained on ImageNet database are used for evaluation. We find that features from lower layers capture more local patterns of objects, and thus perform better than features from higher layers for instance-level image retrieval, which indicates that it is not the best choice to directly apply the final layer or higher layers that are designed for classification tasks to instance-level image retrieval. In addition, we conduct further experiments by changing the scale of input images and using the same feature ex-

traction and encoding methods. It is surprising that the behavior of filters in each layer changes significantly with respect to the scale of input images. With input images of higher resolution, even the filters at higher layers effectively capture local characteristics of images as well, apart from semantic concepts of objects, thus producing better features and subsequent better retrieval results.

The contributions of this work are three-fold. First, we design and conduct systematic and thorough experiments to investigate the performance of features from different layers and different scales of input test images in instance-level image retrieval. Second, we introduce using VLAD encoding of local convolutional features from CNNs for image retrieval. The new convolutional feature mimics the ability of SIFT descriptors to preserve local characteristics of objects, in addition to the well-known power of CNNs of capturing category-level information. Our framework, based on the new features, outperforms other VLAD and CNN based approaches even with a relatively low-dimensional representation. Finally, we provide insights as to why lower layers should be used for instance-level image retrieval rather than higher layers, while higher layers may achieve better performance for high resolution input images.

2.2 Related Work

Traditional image retrieval approaches rely on hand-crafted features like SIFT descriptors, which are usually encoded into bag-of-words (BoW) histograms [17]. To increase the discriminative ability of SIFT descriptors, RootSIFT [18] was pro-

posed to address the burstiness problem by using the Hellinger kernel on the original SIFT descriptors. Jégou *et al*. [19] proposed the vector locally aggregated descriptor (VLAD) to obtain a compact representation as a replacement for BoW histograms, which achieves good results while requiring less storage. PCA and whitening [20], signed square root (SSR) on VLAD vectors [19] and intra-normalization [21] are later applied to the original VLAD descriptors to reduce noise and further boost performance. Multi-VLAD [21] is based on constructing and matching VLAD features of multiple levels from an image to improve localization accuracy. Other global features such as GIST descriptors and Fisher Vector (FV) [22] have also been evaluated for large-scale image retrieval. Some approaches rely on semantic concepts or attributes to capture mid-level image information [23–25], where attributes are binary values indicating the presence of semantic characteristics. Relative attributes have been widely applied to refine search results. In [26], a set of ranking functions are learned offline to predict the strength of attributes, which are then updated by relative attribute feedback to rerank relevant images from the query stage. Implicit feedback [27] to learn ranking functions using implied user feedback cues and pivot attributes selection [28] to reduce the system's uncertainty have also been proposed to improve reranking performance. [29] learns a generic prediction function and adapts it into a user-specific function using user-labeled samples for personalized image search.

CNNs have led to major improvements in image classification [12–14]. As a universal image representation, CNN features can be applied to other recognition tasks and perform well [11, 12, 14]. Razavian *et al*. [13] first investigated the use of

CNN features, *i.e.*, OverFeat [30], for various computer vision tasks, including image retrieval. However, the performance of CNN feature extracted from the final layer lags behind that of simple SIFT-based methods with BoW and VLAD encoding. Only by additionally incorporating spatial information do they achieve comparable results. In [31], CNN features learned from natural images with various augmentation and pooling schemes are applied to painting retrieval and achieve good results. Gong *et al*. [15] introduce Multi-scale Orderless Pooling (MOP) to aggregate CNN activations from higher layers with VLAD, where these activations are extracted by a sliding window with multiple scales. Experiments on an image retrieval dataset have shown promising results, but choosing which scales and layers to use remains unclear. In [32], a CNN model is retrained on a separate landmark database that is similar to the images at query time. Not surprisingly, features extracted from the retrained CNN model obtain very good performance. Unfortunately, collecting training samples and retraining the entire CNN model requires significant amounts of human and computing resources, making the application of this approach rather limited. [33] conducted a comprehensive study on applying CNN features to real-world image retrieval with model retraining and similarity learning. Encouraging experimental results show that CNN features are effective in bridging the semantic gap between low-level visual features and high-level concepts. Recently, [16] conducted extensive experiments on different instance retrieval dataset and obtained excellent results by using spatial search with CNN features. Our work is inspired by [15] which also employs VLAD on CNN activations on multi-scale setting, but fundamentally different from [15]. They utilize higher layers and multi-scale slid-

ing window to extract CNN features from multiple patches independently, so the network has to be applied multiple times. In contrast, we apply the network only once to the input image, and extract features at each location of the convolutional feature map in each layer. We also explicitly verify the effectiveness of intermediate layers for image retrieval and provide additional analysis on the effect of scale.

[34] introduces latent concept descriptors for video event detection by extracting and encoding features using VLAD at the last convolutional layer with spatial pooling. In contrast, we extend the use of convolutional features to lower layers without additional pooling to preserve local information. We also focus on evaluating performance of different convolutional layers for instance-level image retrieval.

2.3 Approach

We describe our approach of extracting and encoding CNN features for image retrieval in this section. We start by introducing the deep neural networks used in our framework, and then describe the method for extracting features. To encode features for efficient retrieval, we adopt VLAD to compress the CNN features into a compact representations.

2.3.1 Convolutional neural network

Our approach is applicable to various convolutional neural network architectures. We experiment with two variants of recent deep neural networks: VGG-16 [4] and GoogLeNet [9], which ranked top two in ILSVRC 2014. The networks are pre-

trained on ImageNet by Caffe implementation [35] and publicly available on the Caffe model zoo. We adopt the 16 layers VGG-16 trained by [4] as it gives similar performance to the 19 layer version. The network consists of stacked 3×3 convolutional layers and pooling layers, followed by two fully connected layers and takes images of 224×224 pixels as input. We also use a 22-layer deep convolutional network GoogLeNet [9], which gives state-of-the-art results in ImageNet classification tasks. The GoogLeNet takes images of 224×224 pixels as input that is then passed through multiple convolutional layers and stacking "inception" modules. Each inception module is regarded as a convolutional layer containing 1×1, 3×3 and 5×5 convolutions, which are concatenated with an additional 3×3 max pooling, with 1×1 convolutional layers in between for dimensionality reduction. There are totally 9 inception modules sequentially connected, followed by an average pooling and a softmax at the end. Unlike VGG-16, fully connected layers are eliminated which simplifies our experiments, so that we can focus on the convolutional feature maps. Finally, the networks are trained by average-pooled activation followed by softmax. The fully convolutional network GoogLeNet simplifies the extension to applying the network to multiple scales of images, and lets us encode the local convolutional features in the same way for all layers, which allows fair comparisons among layers. Table 2.1 shows the output size of intermediate layers in VGG-16 and GoogLeNet. Since it is time consuming to evaluate the lower layers which have large feature maps, some lower layers are omitted in our evaluation.

2.3.2 Extracting convolutional features

Given a pre-trained network (VGG-16 or GoogLeNet) with L layers, an input image $\mathcal{I}$ is first warped into an $n \times n$ square to fit the size of training images, and then is passed through the network in a forward pass. In the l-th convolutional layer $\mathcal{L}_l$, after applying the filters to the input image $\mathcal{I}$, we obtain an $n^l \times n^l \times d^l$ feature map $\mathcal{M}^l$, where d^l is the number of filters with respect to $\mathcal{L}_l$. For notational simplicity, we denote $n_s^l = n^l \times n^l$. Similar to the strategy in [34], at each location $(i, j), 1 \leq i \leq n^l$ and $1 \leq j \leq n^l$, in the feature map $\mathcal{M}^l$, we obtain a d^l-dimensional vector $\mathbf{f}_{i,j}^l \in \mathbb{R}^{d^l}$ containing activations of all filters, which is considered as our feature vector. In this way, we obtain n_s^l local feature vectors for each input image at the convolutional layer $\mathcal{L}_l$, denoted as $\mathbf{F}^l = \{\mathbf{f}_{1,1}^l, \mathbf{f}_{1,2}^l, \cdots, \mathbf{f}_{n^l,n^l}^l\} \in \mathbb{R}^{d^l \times n_s^l}$. While [34] only extracts features from the last convolutional layer, we extend the feature extraction approach to all convolutional layers. By processing the input image $\mathcal{I}$ throughout the network, we finally obtain a set of feature vectors for each layer, $\{\mathbf{F}^1, \mathbf{F}^2, \cdots, \mathbf{F}^L\}$. The feature extraction procedure is illustrated in Figure 2.1[1].

2.3.3 VLAD encoding

Unlike image classification, which is trained with many labeled data for every category, in instance retrieval generally there is no training data available. Therefore, a pre-trained network is likely to fail to produce good holistic representations that are invariant to translation or viewpoint changes while preserving instance level

[1]The k-means clustering figure is from `http://www.vlfeat.org/overview/kmeans.html`

information. In contrast, local features, which focus on smaller parts of images, are easier to represent and generalize to other object categories while capturing invariance.

Since each image contains a set of low-dimensional feature vectors, which has similar structure as dense SIFT, we propose to encode these feature vectors into a single feature vector using standard VLAD encoding. The VLAD encoding is effective for encoding local features into a single descriptor while achieving a favorable trade-off between retrieval accuracy and memory footprint. An overview of our system is illustrated in Figure 2.1.

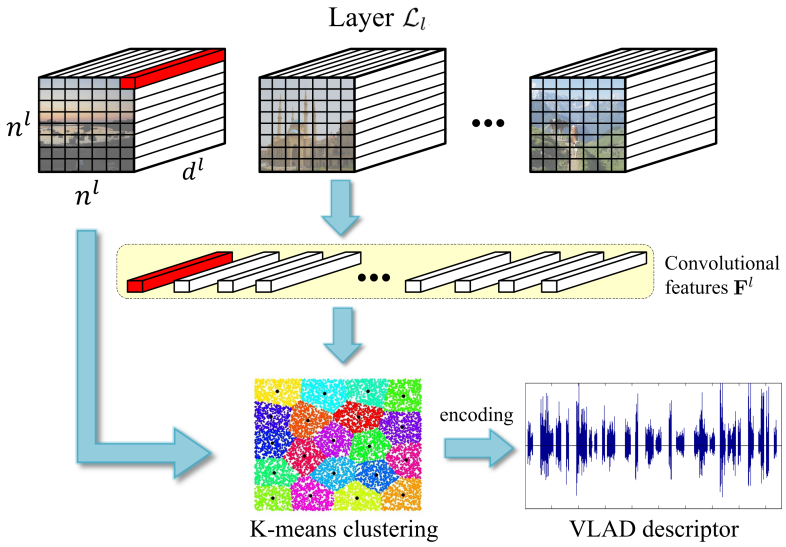


Figure 2.1: Overview of our feature extraction and encoding.

VLAD encoding is similar to constructing BoW histograms. Given a collection of L2-normalized convolutional features from layer $\mathcal{L}_l$, we perform k-means clustering to obtain a vocabulary $\mathbf{c}_1^l, ..., \mathbf{c}_k^l$ of k visual words, where k is relatively small ($k = 100$ in our experiments following [15]), so the vocabulary is coarse. For each image, a

(a) GoogLeNet

Layer (low → high)	Output size ($n^l \times n^l \times d^l$)
`pool1-norm1`	$56 \times 56 \times 64$
`conv2-norm2`	$28 \times 28 \times 192$
`Inception 3a`	$28 \times 28 \times 256$
`Inception 3b`	$28 \times 28 \times 480$
`Inception 4a`	$14 \times 14 \times 512$
`Inception 4b`	$14 \times 14 \times 512$
`Inception 4c`	$14 \times 14 \times 512$
`Inception 4d`	$14 \times 14 \times 528$
`Inception 4e`	$14 \times 14 \times 832$
`Inception 5a`	$7 \times 7 \times 832$
`Inception 5b`	$7 \times 7 \times 1024$

(b) VGG-16

Layer (low → high)	Output size ($n^l \times n^l \times d^l$)
`conv2_1`	$112 \times 112 \times 128$
`conv2_2`	$112 \times 112 \times 128$
`conv2_3`	$112 \times 112 \times 128$
`conv3_1`	$56 \times 56 \times 256$
`conv3_2`	$56 \times 56 \times 256$
`conv3_3`	$56 \times 56 \times 256$
`conv4_1`	$28 \times 28 \times 512$
`conv4_2`	$28 \times 28 \times 512$
`conv4_3`	$28 \times 28 \times 512$
`conv5_1`	$14 \times 14 \times 512$
`conv5_2`	$14 \times 14 \times 512$
`conv5_3`	$14 \times 14 \times 512$

Table 2.1: Size of feature maps

convolutional feature $\mathbf{f}_{i,j}^l$ from layer $\mathcal{L}_l$ is assigned to its nearest visual word $\mathbf{c}_i^l = NN(\mathbf{f}_{i,j}^l)$. For the visual word $\mathbf{c}_i^l$, the vector difference between $\mathbf{c}_i^l$ and the feature $\mathbf{f}_{i,j}^l$ (residual), $\mathbf{f}_{i,j}^l - \mathbf{c}_i^l$, is recorded and accumulated for all features assigned to $\mathbf{c}_i^l$. The VLAD encoding converts the set of convolutional features of an image, $\mathbf{F}^l$, from layer $\mathcal{L}_l$ to a single $d^l \times k$-dimensional vector $\mathbf{v}^l \in \mathbb{R}^{d^l \times k}$, describing the distribution

of feature vectors regarding the visual words. Formally, a VLAD descriptor of an image regarding layer $\mathcal{L}_l$ is represented as

$$\mathbf{v}^l = [\sum_{NN(\mathbf{f}^l_{i,j})=\mathbf{c}^l_1} \mathbf{f}^l_{i,j} - \mathbf{c}^l_1, \cdots, \sum_{NN(\mathbf{f}^l_{i,j})=\mathbf{c}^l_k} \mathbf{f}^l_{i,j} - \mathbf{c}^l_k]. \tag{2.1}$$

Here $\sum_{NN(\mathbf{f}^l_{i,j})=\mathbf{c}^l_k} \mathbf{f}^l_{i,j} - \mathbf{c}^l_k$ is the accumulated residual between the visual word $\mathbf{c}^l_k$ and all convolutional features $\mathbf{f}^l_{i,j}$ that are assigned to $\mathbf{c}^l_k$. The VLAD descriptors are normalized by intra-normalization which has been shown to give superior results than signed square root (SSR) normalization [21]. Since the dimensionality of the original VLAD descriptor is very high, making direct comparison expensive, we further apply PCA to reduce the dimensionality of VLAD descriptors to improve retrieval efficiency and then whitening to increase its robustness against noise.

2.3.4 Image Retrieval

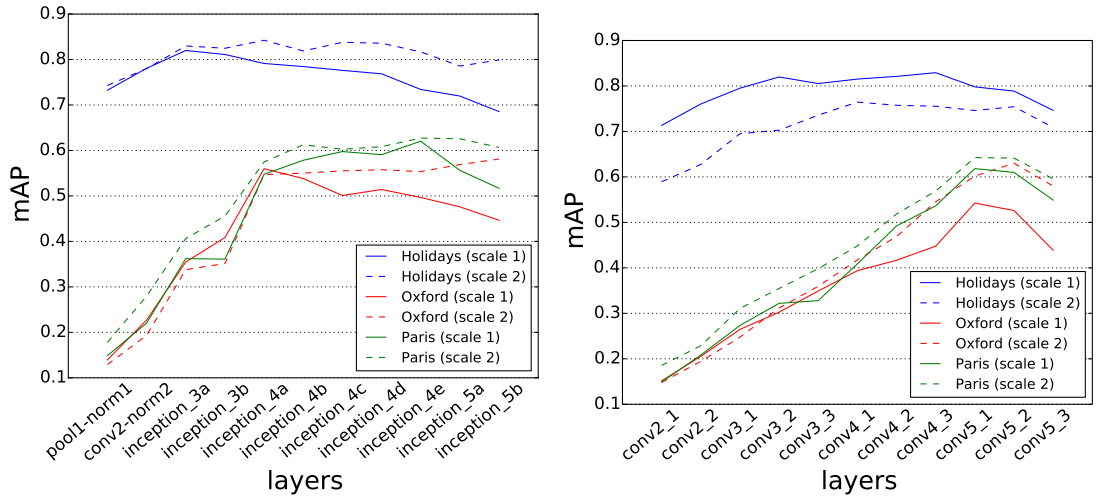
For all database images and a query image, we extract convolutional features and encode them into VLAD descriptors. Image retrieval is done by calculating the L2 distance between the VLAD descriptors of the query image and database images. We use PCA to compress the original VLAD descriptors to relatively low-dimensional vectors (128-D), so that the computation of L2 distance can be done efficiently. We will show in the experiments that the compressed 128-D VLAD vectors achieve excellent results with little loss of performance.

2.4 Experiments

We perform experiments on 3 instance-level image retrieval datasets: Holidays [36], Oxford [37] and Paris [38]. The Holidays dataset includes 1491 images of personal holiday photos from 500 categories, where the first image in each category is used as the query. The Oxford and Paris datasets consist of 5062 images and 6412 images of famous landmarks in Oxford and Paris, respectively. Both datasets have 55 queries with specified rectangular region of interest enclosing the instance to be retrieved, where each landmark has multiple query images. To simplify the experiments, the rectangular regions are ignored and full images are used for retrieval in this work. Following the standard evaluation protocol, we use mean average precision (mAP) to evaluate the performance of our approach.

2.4.1 Comparison of layers

We first study the performance of convolutional features from different layers. We use VLAD to encode convolutional features from each layer and evaluate the mAP with respect to the corresponding layer. Figure 2.2 shows the performance for both VGG-16 and GoogLeNet. There is a clear trend in the results of both networks on the first scale (solid lines in the figure). The mAP first increases as we go deeper into the network because the convolutional features achieve more invariance, until reaching a peak. However, the performance at higher layers gradually drops since the features are becoming too generalized and less discriminative for instance-level retrieval. The best performing layers of GoogLeNet on the Holidays, Oxford and



(a) GoogLeNet (b) VGG-16

Figure 2.2: Performance of different layers on both scales: Solid and dash lines correspond to the original and second scale respectively. Fully-connected layers of VGG-16 are omitted due to incompatible size of the last convolutional layer at scale 2.

Paris datasets are `Inception 3a`, `Inception 4a`, and `Inception 4e` respectively. On the Holidays dataset, the performance of intermediate layers is much better than that of the last layer (82.0% vs 68.5%). In contrast, the best performing layers on the Oxford and Paris datasets are from middle upper layers. Nevertheless, similar trends can still be clearly seen on these two datasets that the intermediate layers perform better than the last layer. We then conduct similar experiment with the 16 layers VGG-16. Although VGG-16 is less deeper than GoogLeNet, we still see this trend. On the Oxford and Paris datasets, the best performing layer is not the last layer, but the intermediate convolutional layers `conv5_1`, showing that increasing generalization at higher layers is not always useful in instance retrieval. This verifies that across different network architectures and datasets, intermediate layers perform the best and should be used for instance-level retrieval.

When convolutional networks grow deeper, which gives an increasing number of choice for layers to transfer, it becomes more important to examine the layers used for image retrieval, since the layers perform very differently in deep networks. Unlike recent work, which suggests only using the last two fully connected layers [13,15,32], or the last convolutional layers [16], our experiments show that higher layers are not always optimal depending on the tasks considered, especially for the very deep networks recently proposed. For instance-level image retrieval, which is very different from classification tasks, lower layers usually perform better than higher layers as features from lower layers preserve more local and instance-level characteristics of objects. We envisage this trend will become more pronounced when networks become deeper in the future.

2.4.2 Scales

Applying a network at multiple scales gives significant improvement over its original scale as shown in previous work [13,15]. In view of this, apart from using the original size of input images (scale 1), we enlarge the size of the input image to $2n \times 2n$ (scale 2) to generate 4 times larger feature maps at each layer, and conduct similar experiments. We evaluate the difference in performance using features extracted from scale 1 and scale 2.

Figure 2.2 shows the performance of different layers at both scales. In general, features from the finer scale, which are obtained from higher resolution images, give better performance than the original scale except VGG-16 on the Holidays dataset.

Interestingly, the relative performance among layers at the higher scale are quite different from the original scale from GoogLeNet. On the Holidays dataset, the performance at scale 2 first increases and then decreases as we go up to higher layers. The trend is similar to scale 1 although the performance difference between layers at scale 2 is smaller. On the Oxford and Paris datasets, we obtain better results using features from higher layers than those from lower layers on the finer scale (scale 2). It is surprising that the networks perform better with larger input images, although by default they should take images of 224×224 pixels that they are trained on as the input [16]. An intuitive explanation for the good performance of the last layer at scale 2 is that the original filters focus more on local details of enlarged images since the size of the filters remains unchanged. Therefore, the convolutional features extracted from the higher layers at a finer scale actually focuses on smaller parts of the images, thus preserving mid-level details of objects to some extent instead of global categorical and abstract information as in the original scale. Our experiments suggest that higher resolution images are preferable even if the network was trained at a coarser level. In contrast, different layers in VGG-16, which was trained in a multi-scale setting, behave similarly for both scales.

2.4.3 Feature visualization

To further understand the features of different layers and scales, we produce visualizations of GoogLeNet features based on the Holidays dataset.

Correspondence visualization. We construct a visualization to observe

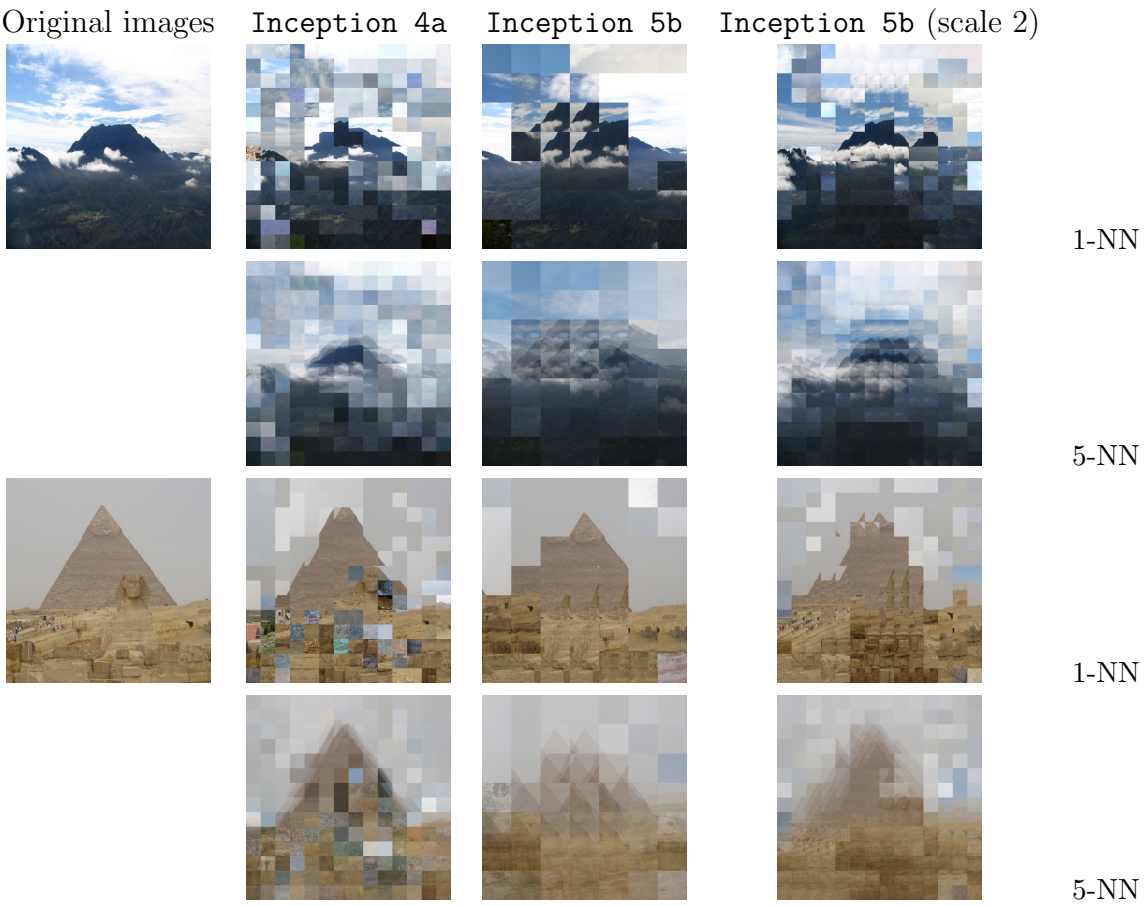


Figure 2.3: Correspondence visualization of images (best viewed electronically).

the correspondence behavior following [39]. To produce the visualization, we first represent each convolutional feature regarding a layer in the database by a square image patch which is obtained from the center of the image region that affects the local feature. Specifically, for an $n \times n$ image with a layer output size $n^l \times n^l$, each local feature will be represented by a square image patch of size $\frac{n}{n^l} \times \frac{n}{n^l}$. For each convolutional feature, the original image patch will be replaced by the average of its k nearest neighbors from all patches extracted in the database. If the local distinction has been abstracted by high level abstraction, locally different image patches will have similar neighbors as these patches may be semantically close; otherwise the

neighbors can be also different since the local distinction is preserved. Note that although the actual image region that affects the local features is much larger than the displayed patch itself due to stacked convolutions, the center patch still preserves localized correspondence [39].

The intermediate convolutional layers of the shallow AlexNet [8] preserve correspondence between different instance objects as well as traditional SIFT descriptor [39]. However, as CNNs become deeper, it is unclear how the intermediate to high level convolutional layers would perform in capturing correspondence information. In addition, we observe the behavior difference between scales of the feature from the visualization. In particular, we would like to understand why the higher layers at finer scale obtain better performance than at lower scale. [39] focuses on part correspondence across different object instances, which is in contrast to our goal of finding correspondence between objects. However, we believe part correspondence is an important step for achieving instance correspondence, and this visualization is also useful in understanding the CNN features in instance correspondence.

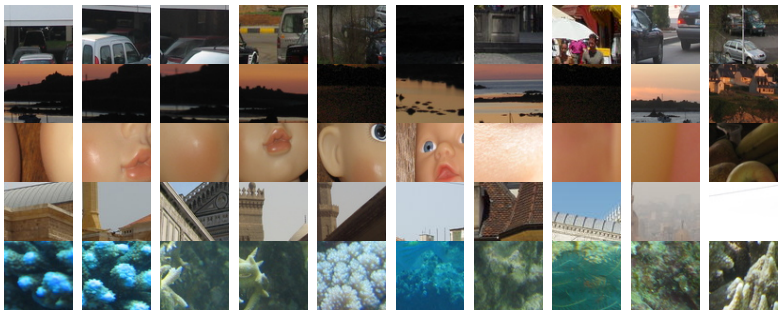
The visualization is presented in Figure 2.3. The size of the convolutional feature map in `Inception 5b` scale 1 is 7×7, which is much smaller than 14×14 in `Inception 4a`'s . Therefore, each patch of `Inception 5b` in the visualization is much larger than `Inception 4a`. From the visualization, it is clear that features from `Inception 5b` do not correspond well compared to those from `Inception 4a`. In `Inception 5b`, we can see many repetitive patterns for both 1-NN and 5-NN cases, which means that local features spatially close to each other are highly similar while the local appearance disparity between them is blurred by convolution

operations. One possible reason is that GoogLeNet is trained with average pooling just before softmax, which encourages the features of the last convolutional layer to be similar. Comparing `Inception 5b` (scale 2) to `Inception 4a`, which have the same feature map sizes, `Inception 5b` retrieves more semantically relevant rather than locally distinct patches. When applied to finer scale (scale 2), `Inception 5b` contains more local appearance details than the original scale, thus producing more diverse patches and roughly preserving the original appearance of the objects. The visualization of `Inception 4a` contains more semantically irrelevant patches, especially in textureless regions, like retrieving grass or sea patches in the pyramid. However, there are less repetitive patterns in the visualization, and the edges in the images are better preserved. This shows that, as an intermediate convolutional layer, `Inception 4a` is more powerful at preserving correspondence of objects and capturing local appearance distinctions.

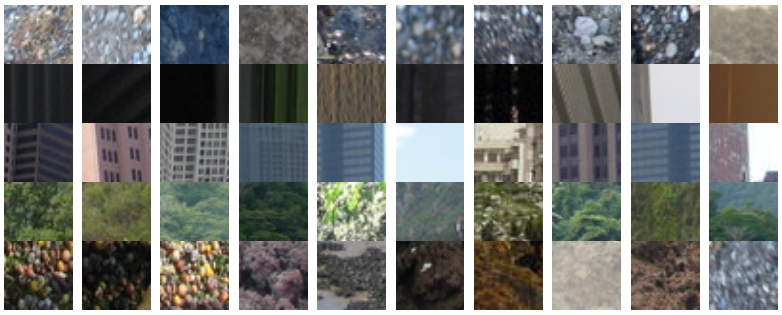
Patch clusters. To better observe the clustering of local CNN features, we sample patches in the dataset and show their nearest neighbors on different layers. Each convolutional feature is represented as a patch in the same way as in the correspondence visualization. Figure 2.4 shows the patch clustering visualization of GoogLeNet layers `Inception 3a`, `Inception 5b` and `Inception 5b` (scale 2). The patch clusters in the lower layer `Inception 3a` are quite similar to SIFT-like low level features, where strong edges, corners and texture are discovered and encoded. For higher layers, such as `Inception 5b`, we can see more generalization of parts with semantic meaning, such as different views of a car or scene, which reflects the tendency of higher layers to capture category-level invariances. However, for the

same layer `Inception 5b` applied to the finer scale, the features focus on smaller parts of the images, thus capturing more local appearance. This confirms that the features behave quite differently when applied to images of different resolutions. Although the higher layers are supposed to encode high level categorical features, more instance-level details are also preserved when they are applied to finer scales, so they are more useful for image retrieval.

(a) `Inception 3a` (scale 1)



(b) `Inception 5b` (scale 1)



(c) `Inception 5b` (scale 2)

Figure 2.4: Visualization of local convolutional features on different layers and scales. Each row represents a cluster of local convolutional features by displaying the corresponding patches. The leftmost column shows the sampled reference patches, and other patches are sorted according to their L2 distance with the reference patches.

Method	Holidays	Oxford	Paris
SIFT-based method			
BoW 200k-D [19]	54.0	36.4	46.0
Improved Fisher [22]	62.6	41.4	-
LCS+RN [40]	65.8	51.7	-
VLADintra+ RootSIFT [21]	65.3	55.8	-
CVLAD [41]	82.7	51.4	-
CNN-based method			
CNNaug-ss [13]	84.3	68.0	79.5
Multi-resolution Spatial Search [16]	**89.7**	**84.4**	**85.3**
Neural codes [32]	79.3	54.5	-
MOP-CNN [15]	80.2	-	-
Ours (VGG-16)	83.8	64.9	69.4
Ours (GoogLeNet)	84.0	58.1	68.8

Table 2.2: Comparison with other methods on image retrieval dataset.

2.4.4 Comparison to state-of-the-art

Since our method only uses simple CNN features and VLAD encoding, we only compare to other recent CNN based approaches and classic SIFT-based representations with BoW and VLAD encoding.

Uncompressed representation. We first compare our approach using uncompressed VLAD representation with other state-of-the-art approaches in Table 2.2. In Figure 2.2, the best performing layers on Holidays, Oxford and Paris datasets are `Inception 3a` on original scale (scale 1), `Inception 5b` and `Inception 4e` on finer scale (scale 2) on GoogLeNet respectively, and `conv4_2`, `conv5_1` and `conv5_2` for Holidays, Oxford and Paris dataset on VGG-16 respectively. The VLAD

descriptors from the two scales on the best performing layer are concatenated as our final multi-scale descriptors. VGG-16, which has much larger convolutional feature maps, performs slightly better than GoogLeNet for image retrieval. Although we do not focus on producing state-of-the-art results on image retrieval but more on investigating the behavior of convolutional features from different layers and the effect of multiple scales, our system gives competitive results compared to state-of-the-art methods. Specifically, our approach significantly outperforms all the classic SIFT-based approaches with BoW and VLAD encoding, which verifies the representative power of the convolutional features compared to traditional SIFT descriptors. Although better results are reported by other SIFT-based approaches using large vocabularies, spatial verification and query expansion, etc., our framework is not limited to the current setting, and can be readily adapted to other encoding schemes (*i.e.*, BoW and FV), and re-ranking techniques (*i.e.*, query expansion). In addition, compared to recent CNN-based approaches, our method still produces better or comparable results. In particular, our approach outperforms its rivals that either use time-consuming multi-scale sliding windows to extract features [15] or retrain the entire network using extra data [32]. It should be noted that including spatial information greatly boosts the performance of CNN-based approaches such as spatial search [13, 16]. Although [13] and [16] produce better results than our method, we believe that our approach of extracting and encoding convolutional features using lower layers and our investigation of how scales affect convolutional features provide a better understanding of why spatial search on multi-scale features from the last layer performs well. Spatial information can be also included in our framework with

few modifications, which will be studied in future work. It would also be interesting to combine multiple layers from the best scales in spatial search to fully utilize the power of deep networks. **Low-dimensional representation.** To trade-off between retrieval accuracy and storage space, most approaches compress the original feature vector to a low-dimensional representation. Therefore, we conduct additional experiments using compressed VLAD descriptors and compare the results with those of other approaches using low-dimensional representations. We use PCA to reduce the dimensionality to 128 and apply whitening to further remove noise.

As shown in Table 2.3, our method obtains state-of-the-art results on two out of three datasets with minimal performance loss. Our method outperforms all SIFT-based approaches by a large margin, which again demonstrates the power of CNNs. Moreover, we obtain better results than [32], even though [32] fine-tunes the pre-trained CNNs using a large amount of additional data. Although adopting similar VLAD encoding scheme, our method still outperforms MOP-CNN [15] which uses a larger 512-D representation, which further verifies that our approach of extracting convolutional features from intermediate layers is more suitable for instance-level image retrieval. The performance of [16] with low-dimensional descriptors drops notably compared to our 128-D representation, showing that elimination of spatial search greatly reduces the power of CNN representation. It is also important to use more sophisticated encoding methods to capture the local information of convolutional features instead of simple max-pooling as in [16]. In contrast, our low-dimensional representation is robust and retains good discriminative power.

Method	dim	Holidays	Oxford	Paris
VLADintra+SIFT [21]	128	62.5	44.8	-
FV+T-embedding [42]	128	61.7	43.3	-
Neural codes [32]	128	78.9	55.7	-
MOP-CNN [15]	512	78.4	-	-
Spatial Pooling [16]	256	74.2	53.3	**67.0**
Ours (VGG-16)	128	81.6	**59.3**	59.0
Ours (GoogLeNet)	128	**83.6**	55.8	58.3

Table 2.3: Comparison of low dimensional descriptors.

2.5 Conclusion

In this work, we systematically experiment with features from different layers of convolutional networks and different scales of input images for instance-level image retrieval, and provide insights into performance through various visualizations. With VLAD encoding on convolutional response, we achieve state-of-the-art retrieval results using low dimensional representations on two of the instance image retrieval datasets.

Chapter 3: Full Length Video Classification

3.1 Motivation

Convolutional Neural Networks have proven highly successful at static image recognition problems such as the MNIST, CIFAR, and ImageNet Large-Scale Visual Recognition Challenge [8,9,11]. By using a hierarchy of trainable filters and feature pooling operations, CNNs are capable of automatically learning complex features required for visual object recognition tasks achieving superior performance to hand-crafted features. Encouraged by these positive results several approaches have been proposed recently to apply CNNs to video and action classification tasks [43–46].

Video analysis provides more information to the recognition task by adding a temporal component through which motion and other information can be additionally used. At the same time, the task is much more computationally demanding even for processing short video clips since each video might contain hundreds to thousands of frames, not all of which are useful. A naïve approach would be to treat video frames as still images and apply CNNs to recognize each frame and average the predictions at the video level. However, since each individual video frame forms only a small part of the video's story, such an approach would be using incomplete information and could therefore easily confuse classes especially if there are

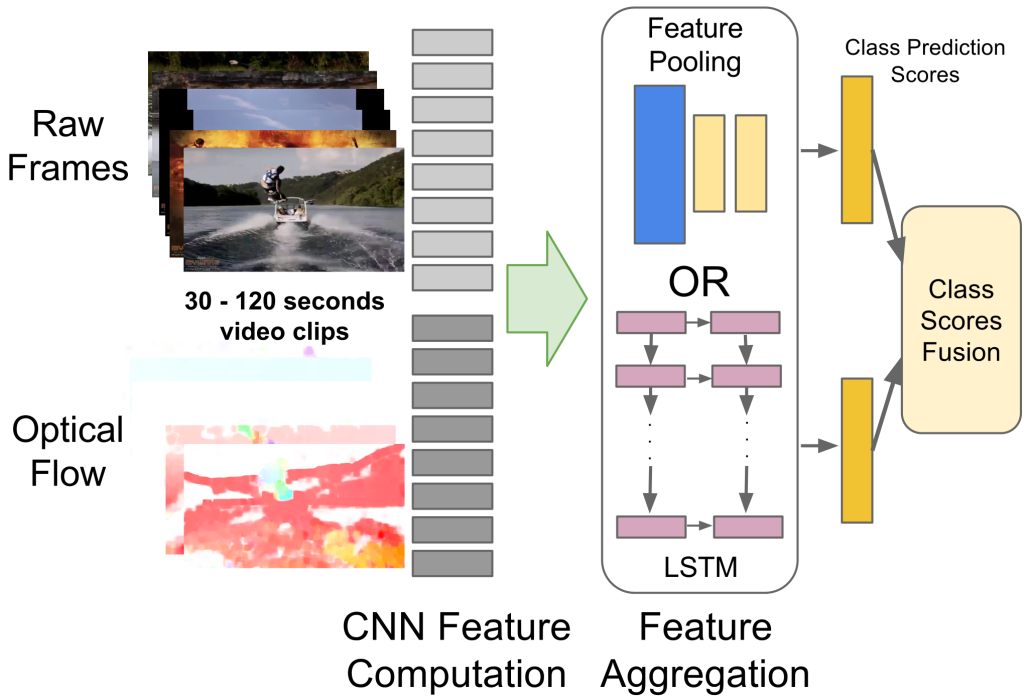


Figure 3.1: Overview of our approach.

fine-grained distinctions or portions of the video irrelevant to the action of interest.

Therefore, we hypothesize that learning a global description of the video's temporal evolution is important for accurate video classification. This is challenging from a modeling perspective as we have to model variable length videos with a fixed number of parameters. We evaluate two approaches capable of meeting this requirement: feature-pooling and recurrent neural networks. The feature pooling networks independently process each frame using a CNN and then combine frame-level information using various pooling layers. The recurrent neural network architecture we employ is derived from Long Short Term Memory (LSTM) [47] units, and uses memory cells to store, modify, and access internal state, allowing it to discover long-range temporal relationships. Like feature-pooling, LSTM networks operate

on frame-level CNN activations, and can learn how to integrate information over time. By sharing parameters through time, both architectures are able to maintain a constant number of parameters while capturing a global description of the video's temporal evolution.

Since we are addressing the problem of video classification, it is natural to attempt to take advantage of motion information in order to have a better performing network. Previous work [45] has attempted to address this issue by using frame stacks as input. However, this type of approach is computationally intensive since it involves thousands of 3D convolutional filters applied over the input volumes. The performance grained by applying such a method is below 2% on the Sports-1M benchmarks [45]. As a result, in this work, we avoid implicit motion feature computation.

In order to learn a global description of the video while maintaining a low computational footprint, we propose processing only one frame per second. At this frame rate, implicit motion information is lost. To compensate, following [46] we incorporate explicit motion information in the form of optical flow images computed over adjacent frames. Thus optical flow allows us to retain the benefits of motion information (typically achieved through high-fps sampling) while still capturing global video information. Our contributions can be summarized as follows:

1. We propose CNN architectures for obtaining global video-level descriptors and demonstrate that using increasing numbers of frames significantly improves classification performance.
2. By sharing parameters through time, the number of parameters remains con-

stant as a function of video length in both the feature pooling and LSTM architectures.

3. We confirm that optical flow images can greatly benefit video classification and present results showing that even if the optical flow images themselves are very noisy (as is the case with the Sports-1M dataset), they can still provide a benefit when coupled with LSTMs.

Leveraging these three principles, we achieve state-of-the-art performance on two different video classification tasks: Sports-1M (Section 3.4.1) and UCF-101 (Section 3.4.2).

3.2 Related Work

Traditional video recognition research has been extremely successful at obtaining global video descriptors that encode both appearance and motion information in order to provide state-of-art results on a large number of video datasets. These approaches are able to aggregate local appearance and motion information using hand-crafted features such as Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), Motion Boundary Histogram (MBH) around spatio-temporal interest points [6], in a dense grid [48] or around dense point trajectories [7, 49–51] obtained through optical flow based tracking. These features are then encoded in order to produce a global video-level descriptor through bag of words (BoW) [6] or Fisher vector based encodings [51].

However, no previous attempts at CNN-based video recognition use both motion information and a global description of the video: Several approaches [43–45]

employ 3D-convolution over short video clips - typically just a few seconds - to learn motion features from raw frames implicitly and then aggregate predictions at the video level. Karpathy *et al.* [45] demonstrate that their network is just marginally better than single frame baseline, which indicates learning motion features is difficult. In view of this, Simonyan *et al.* [46] directly incorporate motion information from optical flow, but only sample up to 10 consecutive frames at inference time. The disadvantage of such local approaches is that each frame/clip may contain only a small part of the full video's information, resulting in a network that performs no better than the naïve approach of classifying individual frames.

Instead of trying to learn spatio-temporal features over small time periods, we consider several different ways to aggregate strong CNN image features over long periods of a video (tens of seconds) including feature pooling and recurrent neural networks. Standard recurrent networks have trouble learning over long sequences due to the problem of vanishing and exploding gradients [52]. In contrast, the Long Short Term Memory (LSTM) [47] uses memory cells to store, modify, and access internal state, allowing it to better discover long-range temporal relationships. For this reason, LSTMs yield state-of-the-art results in handwriting recognition [53,54], speech recognition [5,55], phoneme detection [56], emotion detection [57], segmentation of meetings and events [58], and evaluating programs [59]. While LSTMs have been applied to action classification in [60], the model is learned on top of SIFT features and a BoW representation. In addition, our proposed models allow joint fine tuning of convolutional and recurrent parts of the network, which is not possible to do when using hand-crafted features, as proposed in prior work. Baccouche

et al. [60] learns globally using Long Short-Term Memory (LSTM) networks on the ouput of 3D-convolution applied to 9-frame videos clips, but incorporates no explicit motion information.

3.3 Approach

Two CNN architectures are used to process individual video frames: AlexNet and GoogLeNet. AlexNet, is a Krizhevsky-style CNN [8] which takes a 220 × 220 sized frame as input. This frame is then processed by square convolutional layers of size 11, 9, and 5 each followed by max-pooling and local contrast normalization. Finally, outputs are fed to two fully-connected layers each with 4096 rectified linear units (ReLU). Dropout is applied to each fully-connected layer with a ratio of 0.6 (keeping and scaling 40% of the original outputs).

GoogLeNet [9], uses a network-in-network approach, stacking Inception modules to form a network 22 layers deep that is substantially different from previous CNNs [8, 11]. Like AlexNet, GoogLeNet takes a single image of size 220 × 220 as input. This image is then passed through multiple Inception modules, each of which applies, in parallel, 1×1, 3×3, 5×5 convolution, and max-pooling operations and concatenates the resulting filters. Finally, the activations are average-pooled and output as a 1000-dimensional vector.

In the following sections, we investigate two classes of CNN architectures capable of aggregating video-level information. In the first section, we investigate various feature pooling architectures that are agnostic to temporal order and in the

following section we investigate LSTM networks which are capable of learning from temporally ordered sequences. In order to make learning computationally feasible, in all methods CNN share parameters across frames.

3.3.1 Feature Pooling Architectures

Temporal feature pooling has been extensively used for video classification [6, 48, 49], and has been usually applied to bag-of-words representations. Typically, image-based or motion features are computed at every frame, quantized, then pooled across time. The resulting vector can be used for making video-level predictions. We follow a similar line of reasoning, except that due to the fact that we work with neural networks, the pooling operation can be incorporated directly as a layer. This allows us to experiment with the location of the temporal pooling layer with respect to the network architecture.

We analyze several variations depending on the specific pooling method and the particular layer whose features are aggregated. The pooling operation need not be limited to max-pooling. We considered using both average pooling, and max-pooling which have several desirable properties as shown in [61]. In addition, we attempted to employ a fully connected layer as a "pooling layer". However, we found that both average pooling and a fully connected layer for pooling failed to learn effectively due to the large number of gradients that they generate. Max-pooling generates much sparser updates, and as a result tends to yield networks that learn faster, since the gradient update is generated by a sparse set of features

from each frame. Therefore, in the rest of the chapter we use max-pooling as the main feature aggregation technique.

Unlike traditional bag of words approaches, gradients coming from the top layers help learn useful features from image pixels, while allowing the network to choose which of the input frames are affected by these updates. When used with max-pooling, this is reminiscent of multiple instance learning, where the learner knows that at least one of the inputs is relevant to the target class.

We experimented with several variations of the basic max-pooling architecture as shown in Figure 3.2:

Conv Pooling: The Conv Pooling model performs max-pooling over the final convolutional layer across the video's frames. A key advantage of this network is that the spatial information in the output of the convolutional layer is preserved through a max operation over the time domain.

Late Pooling: The Late Pooling model first passes convolutional features through two fully connected layers before applying the max-pooling layer. The weights of all convolutional layers and fully connected layers are shared. Compared to Conv Pooling, Late Pooling directly combines high-level information across frames.

Slow Pooling: Slow Pooling hierarchically combines frame level information from smaller temporal windows. Slow Pooling uses a two-stage pooling strategy: max-pooling is first applied over 10-frames of convolutional features with stride 5 (e.g. max-pooling may be thought of as a size-10 filter being convolved over a 1-D input with stride 5). Each max-pooling layer is then followed by a fully-connected

layer with shared weights. In the second stage, a single max-pooling layer combines the outputs of all fully-connected layers. In this manner, the Slow Pooling network groups temporally local features before combining high level information from many frames.

Local Pooling: Similar to Slow Pooling, the Local Pooling model combines frame level features locally after the last convolutional layer. Unlike Slow Pooling, Local Pooling only contains a single stage of max-pooling after the convolutional layers. This is followed by two fully connected layers, with shared parameters. Finally a larger softmax layer is connected to all towers. By eliminating the second max-pooling layer, the Local Pooling network avoids a potential loss of temporal information.

Time-Domain Convolution: The Time-Domain Convolution model contains an extra time-domain convolutional layer before feature pooling across frames. Max-pooling is performed on the temporal domain after the time-domain convolutional layer. The convolutional layer consist of 256 kernels of size 3×3 across 10 frames with frame stride 5. This model aims at capturing local relationships between frames within a small temporal window.

GoogLeNet Conv Pooling: We experimented with an architecture based on GoogLeNet [9], in which the max-pooling operation is performed after the dimensionality reduction (average pooling) layer in GoogLeNet. This is the layer which in the original architecture was directly connected to the softmax layer. We enhanced this architecture by adding two fully connected layers of size 4096 with ReLU activations on top of the 1000D output but before softmax. Similar to AlexNet-based

models, the weights of convolutional layers and inception modules are shared across time.

3.3.2 LSTM Architecture

In contrast to max-pooling, which produces representations which are order invariant, we propose using a recurrent neural network to explicitly consider *sequences* of CNN activations. Since videos contain dynamic content, the variations between frames may encode additional information which could be useful in making more accurate predictions.

Given an input sequence $\mathbf{x} = (x_1, \ldots, x_T)$ a standard recurrent neural network computes the hidden vector sequence $\mathbf{h} = (h_1, \ldots, h_T)$ and output vector sequence $\mathbf{y} = (y_1, \ldots, y_T)$ by iterating the following equations from $t = 1$ to T:

$$h_t = \mathcal{H}(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

$$y_t = W_{ho}h_t + b_o \tag{3.2}$$

where the W terms denote weight matrices (e.g. W_{ih} is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is the hidden bias vector) and $\mathcal{H}$ is the hidden layer activation function, typically the logistic sigmoid function.

Unlike standard RNNs, the Long Short Term Memory (LSTM) architecture [62] uses memory cells (Figure 3.3) to store and output information, allowing it to better discover long-range temporal relationships. The hidden layer $\mathcal{H}$ of the LSTM

is computed as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{3.4}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{3.6}$$

$$h_t = o_t \tanh(c_t) \tag{3.7}$$

where σ is the logistic sigmoid function, and i, f, o, and c are respectively the *input gate*, *forget gate*, *output gate*, and *cell* activation vectors. By default, the value stored in the LSTM cell c is maintained unless it is added to by the input gate i or diminished by the forget gate f. The output gate o controls the emission of the memory value from the LSTM cell.

We use a deep LSTM architecture [5] (Figure 3.4) in which the output from one LSTM layer is input for the next layer. We experimented with various numbers of layers and memory cells, and chose to use five stacked LSTM layers, each with 512 memory cells. Following the LSTM layers, a Softmax classifier makes a prediction at every frame.

3.3.3 Training and Inference

The max-pooling models were optimized on a cluster using Downpour Stochastic Gradient Descent starting with a learning rate of 10^{-5} in conjunction with a momentum of 0.9 and weight decay of 0.0005. For LSTM, we used the same optimization method with a learning rate of $N * 10^{-5}$ where N is number of frames. The learning rate was exponentially decayed over time. Each model had between ten and fifty replicas split across four partitions. To reduce CNN training time, the parameters of AlexNet and GoogLeNet were initialized from a pre-trained ImageNet model and then fine-tuned on Sports-1M videos.

Network Expansion for Max-Pooling Networks: Multi-frame models achieve higher accuracy at the cost of longer training times than single-frame models. Since pooling is performed after CNN towers that share weights, the parameters for a single-frame and multi-frame max-pooling network are very similar. This makes it possible to expand a single-frame model to a multi-frame model. Max-pooling models are first initialized as single-frame networks then expanded to 30-frames and again to 120-frames. While the feature distribution of the max-pooling layer could change dramatically as a result of expanding to a larger number of frames (particularly in the single-frame to 30-frame case), experiments show that transfering the parameters is nonetheless beneficial. By expanding small networks into larger ones and then fine-tuning, we achieve a significant speedup compared to training a large network from scratch.

LSTM Training: We followed the same procedure as training max-pooled

network with two modifications: First, the video's label was backpropagated at each frame rather than once per clip. Second, a gain g was applied to the gradients backpropagated at each frame. g was linearly interpolated from 0...1 over frames $t = 0...T$. g had the desired effect of emphasizing the importance of correct prediction at later frames in which the LSTM's internal state captured more information. Compared empirically against setting $g = 1$ over all time steps or setting $g = 1$ only at the last time step T ($g = 0$ elsewhere), linearly interpolating g resulted in faster learning and higher accuracy. For the final results, during training the gradients are backpropagated through the convolutional layers for fine tuning.

LSTM Inference: In order to combine LSTM frame-level predictions into a single video-level prediction, we tried several approaches: 1) returning the prediction at the last time step T, 2) max-pooling the predictions over time, 3) summing the predictions over time and return the max 4) linearly weighting the predictions over time by g then sum and return the max.

The accuracy for all four approaches was less than 1% different, but weighted predictions usually resulted in the best performance, supporting the idea that the LSTM's hidden state becomes progressively more informed as a function of the number of frames it has seen.

3.3.4 Optical Flow

Optical flow is a crucial component of any video classification approach because it encodes the pattern of apparent motion of objects in a visual scene. Since

our networks process video frames at $1fps$, they do not use any apparent motion information. Therefore, we additionally train both our temporal models on optical flow images and perform late fusion akin to the two-stream hypothesis proposed by [46].

Interestingly, we found that initializing from a model trained on raw image frames can help classify optical flow images by allowing faster convergence than when training from scratch. This is likely due to the fact that features that can describe for raw frames like edges also help in classifying optical flow images. This is related to the effectiveness of Motion Boundary Histogram (MBH), which is analogous to computing Histogram of Oriented Gradients (HOG) on optical flow images, in action recognition [51].

Optical flow is computed from two adjacent frames sampled at $15fps$ using the approach of [63]. To utilize existing implementation and networks trained on raw frames, we store optical flow as images by thresholding at $-40, 40$ and rescaling the horizontal and vertical components of the flow to $[0, 255]$ range. The third dimension is set to zero when feeding to the network so that it gives no effect on learning and inference.

In our investigation, we treat optical flow in the same fashion as image frames to learn global description of videos using both feature pooling and LSTM networks.

3.4 Results

We empirically evaluate the proposed architectures on the Sports-1M and UCF-101 datasets with the goals of investigating the performance of the proposed architectures, quantifying the effect of the number of frames and frame rates on classification performance, and understanding the importance of motion information through optical flow models.

3.4.1 Sports-1M dataset

The Sports-1M dataset [45] consists of roughly 1.2 million YouTube sports videos annotated with 487 classes, and it is representative of videos in the wild. There are 1000-3000 videos per class and approximately 5% of the videos are annotated with more than one class. Unfortunately, since the creation of the dataset, about 7% of the videos have been removed by users. We use the remaining 1.1 million videos for the experiments below.

Although Sports-1M is the largest publicly available video dataset, the annotations that it provides are at video level. No information is given about the location of the class of interest. Moreover, the videos in this dataset are unconstrained. This means that the camera movements are not guaranteed to be well-behaved, which means that unlike UCF-101, where camera motion is constrained, the optical flow quality varies wildly between videos.

Data Extraction: The first 5 minutes of each video are sampled at a frame rate of $1fps$ to obtain 300 frames per video. Frames are repeated from the start

for videos that are shorter than 5 minutes. We learn feature pooling models that process up to 120 frames (2 minutes of video) in a single example.

Data Augmentation: Multiple examples per video are obtained by randomly selecting the position of the first frame and consistent random crops of each frame during both training and testing. It is necessary to ensure that the same transforms are applied to all frames for a given start/end point. We process all images in the chosen interval by first resizing them to 256×256 pixels, then randomly sampling a 220×220 region and randomly flipping the image horizontally with 50% probability. To obtain predictions for a video we randomly sample 240 examples as described above and average all predictions, unless noted otherwise. Since LSTM models trained on a fixed number of frames can generalize to any number of frames, we also report results of using LSTMs without data augmentation.

Video-Level Prediction: Given the nature of the methods presented in this chapter, it is possible to make predictions for the entire video without needing to sample, or aggregate (the networks are designed to work on an unbounded number of frames for prediction). However, for obtaining the highest possible classification rates, we observed that it is best to only do this if resource constrained (i.e., when it is only possible to do a single pass over the video for prediction). Otherwise the data augmentation method proposed above yields between 3-5% improvements in Hit@1 on the Sports-1M dataset.

Evaluation: Following [45], we use Hit@k values, which indicate the fraction of test samples that contain at least one of the ground truth labels in the top k predictions. We provide both video level and clip level Hit@k values in order to

Method	Clip Hit@1	Hit@1	Hit@5
Conv Pooling	**68.7**	**71.1**	**89.3**
Late Pooling	65.1	67.5	87.2
Slow Pooling	67.1	69.7	88.4
Local Pooling	68.1	70.4	88.9
Time-Domain Convolution	64.2	67.2	87.2

Table 3.1: Conv-Pooling outperforms all other feature-pooling architectures (Figure 3.2) on Sports-1M using a 120-frame AlexNet model.

compare with previous results where clip hit is the hit on a single video clip (30-120 frames) and video hit is obtained by averaging over multiple clips.

Comparison of Feature-Pooling Architectures: Table 3.1 shows the results obtained using the different feature pooling architectures on the Sports-1M dataset when using a 120 frame AlexNet model. We find that max-pooling over the outputs of the last convolutional layer provides the best clip-level and video-level hit rates. Late Pooling, which max-pools after the fully connected layers, performs worse than all other methods, indicating that preserving the spatial information while performing the pooling operation across the time domain is important. Time-Domain Convolution gives inferior results compared to max-pooling models. This suggests that a single time-domain convolutional layer is not effective in learning temporal relations on high level features, which motivates us to explore more sophisticated network architectures like LSTM which learns from temporal sequences.

Comparison of CNN Architectures: AlexNet and GoogLeNet single-frame CNNs (Section 3.3) were trained from scratch on single-frames selected at

Method	Hit@1	Hit@5
AlexNet single frame	63.6	84.7
GoogLeNet single frame	**64.9**	**86.6**
LSTM + AlexNet (fc)	62.7	83.6
LSTM + GoogLeNet (fc)	**67.5**	**87.1**
Conv pooling + AlexNet	70.4	89.0
Conv pooling + GoogLeNet	**71.7**	**90.4**

Table 3.2: GoogLeNet outperforms AlexNet alone and when paired with both Conv-Pooling and LSTM. Experiments performed on Sports-1M using 30-frame Conv-Pooling and LSTM models. Note that the (fc) models updated only the final layers while training and did not use data augmentation.

random from Sports-1M videos. Results (Table 3.2) show that both CNNs outperform Karpathy *et al.*'s prior single-frame models [45] by a margin of 4.3-5.6%. The increased accuracy is likely due to advances in CNN architectures and sampling more frames per video when training (300 instead of 50).

Comparing AlexNet to the more recent GoogLeNet yields a 1.9% increase in Hit@5 for the max-pooling architecture, and an increase of 4.8% for the LSTM. This is roughly comparable to a 4.5% decrease in top-5 error moving from the Krizhevsky-style CNNs that won ILSVRC-13 to GoogLeNet in ILSVRC-14. For the max-pool architecture, this smaller gap between architectures is likely caused by the increased number of noisy images in Sports-1M compared to ImageNet.

Fine Tuning: When initializing from a pre-trained network, it is not always clear whether fine-tuning should be performed. In our experiments, fine tuning was crucial in achieving high performance. For example, in Table 3.2 we show that a

Method	Frames	Clip Hit@1	Hit@1	Hit@5
LSTM	30	N/A	72.1	90.4
Conv pooling	30	66.0	71.7	90.4
	120	70.8	72.3	90.8

Table 3.3: Effect of the number of frames in the model. Both LSTM and Conv-Pooling models use GoogLeNet CNN.

Method	Hit@1	Hit@5
LSTM on Optical Flow	59.7	81.4
LSTM on Raw Frames	72.1	90.6
LSTM on Raw Frames + LSTM on Optical Flow	73.1	90.5
30 frame Optical Flow	44.5	70.4
Conv Pooling on Raw Frames	71.7	90.4
Conv Pooling on Raw Frames + Conv Pooling on Optical Flow	71.8	90.4

Table 3.4: Optical flow is noisy on Sports-1M and if used alone, results in lower performance than equivalent image-models. However, if used in conjunction with raw image features, optical flow benefits LSTM. Experiments performed on 30-frame models using GoogLeNet CNNs.

LSTM network paired with GoogLeNet, running on 30 frames of the video achieves a Hit@1 rate of 67.5. However, the same network with fine tuning achieves 69.5 Hit@1. Note that these results do not use data augmentation and classify the entire 300 seconds of a video.

Effect of Number of Frames: Table 3.3 compares Conv-Pooling and LSTM models as a function of the number of frames aggregated. In terms of clip hit, the 120 frame model performs significantly better than the 30 frame model. Also our best clip hit of 70.8 represents a 70% improvement over the Slow Fusion approach of [45] which uses clips of few seconds length. This confirms our initial hypothesis that we need to consider the entire video in order to benefit more thoroughly from

Category	Method	Frames	Clip Hit@1	Hit@1	Hit@5
Prior Results [45]	Single Frame	1	41.1	59.3	77.7
	Slow Fusion	15	41.9	60.9	80.2
Conv Pooling	Image and Optical Flow	120	**70.8**	72.4	**90.8**
LSTM	Image and Optical Flow	30	N/A	**73.1**	90.5

Table 3.5: Leveraging global video-level descriptors, LSTM and Conv-Pooling achieve a 20% increase in Hit@1 compared to prior work on the in Sports-1M dataset. Hit@1, and Hit@5 are computed at video level.

its content.

Optical Flow: Table 3.4 shows the results of fusion with the optical flow model. The optical flow model on its own has a much lower accuracy (59.7%) than the image-based model (72.1%) which is to be expected given that the Sports dataset consists of YouTube videos which are usually of lower quality and more natural than hand-crafted datasets such as UCF-101. In the case of Conv Pooling networks the fusion with optical flow has no significant improvement in the accuracy. However, for LSTMs the optical flow model is able to improve the overall accuracy to 73.1%.

Overall Performance: Finally, we compare the results of our best models against the previous state-of-art on the Sports-1M dataset at the time of submission. Table 3.5 reports the results of the best model from [45] which performs several layers of 3D convolutions on short video clips against ours. The max-pool method shows an increase of 18.7% in video Hit@1, whereas the LSTM approach yields a relative increase of 20%. The difference between the max-pool and LSTM method is explained by the fact that the LSTM model can use optical flow in a manner which lends itself to late model fusion, which was not possible for the max-pool model.

3.4.2 UCF-101 Dataset

The UCF-101 [64] contains 13,320 videos with 101 action classes covering a broad set of activities such as sports, musical instruments, and human-object interaction. We follow the suggested evaluation protocol and report the average accuracy over the given three training and testing partitions. It is difficult to train a deep network with such a small amount of data. Therefore, we test how well our models that are trained in Sports-1M dataset perform in UCF-101.

Comparison of Frame Rates: Since UCF-101 contains short videos, 10-15 seconds on average, it is possible to extract frames at higher frame rates such as $6fps$ while still capturing context from the full video. We compare 30-frame models trained at three different frame-rates: $30fps$ (1 second of video) and $6fps$ (5 seconds). Table 3.6 shows that lowering the frame rate from $30fps$ to $6fps$ yields slightly better performance since the model obtains more context from longer input clips. We observed no further improvements when decreasing the frame rate to $1fps$. Thus, as long as the network sees enough context from each video, the effects of lower frames rate are marginal. The LSTM model, on the other hand can take full advantage of the fact that the videos can be processed at 30 frames per second.

Overall Performance: Our models achieve state-of-the-art performance on UCF-101 (Table 3.7), slightly outperforming approaches that use hand-crafted features and CNN-based approaches that use optical flow. As before, the performance edge of our method results from using increased numbers of frames to capture more of the video.

Method	Frame Rate	3-fold Accuracy (%)
Single Frame Model	N/A	73.3
Conv Pooling (30 frames)	30 fps 6 fps	80.8 82.0
Conv Pooling (120 frames)	30 fps 6 fps	82.6 82.6

Table 3.6: Lower frame rates produce higher UCF-101 accuracy for 30-frame Conv-Pooling models.

Our 120 frames model improves upon previous work [46] (82.6% vs 73.0%) when considering models that learn directly from raw frames without optical flow information. This is a direct result of considering larger context within a video, even when the frames within a short clip are highly similar to each other.

Compared to Sports-1M, optical flow in UCF-101 provides a much larger improvement in accuracy (82.6% vs. 88.2% for max-pool). This results from UCF-101 videos being better centered, less shaky, and better trimmed to the action in question than the average YouTube video.

High Quality Data: The UCF-101 dataset contains short, well-segmented videos of concepts that can typically be identified in a single frame. This is evidenced by the high performance of single-frame networks (See Table 3.7). In contrast, videos in the wild often feature spurious frames containing text or shot transitions, hand-held video shot in either first person or third person, and non-topical segments such as commentators talking about a game.

Method	3-fold Accuracy (%)
Improved Dense Trajectories (IDTF)s [51]	87.9
Slow Fusion CNN [45]	65.4
Single Frame CNN Model (Images) [46]	73.0
Single Frame CNN Model (Optical Flow) [46]	73.9
Two-Stream CNN (Optical Flow + Image Frames, Averaging) [46]	86.9
Two-Stream CNN (Optical Flow + Image Frames, SVM Fusion) [46]	88.0
Our Single Frame Model	73.3
Conv Pooling of Image Frames + Optical Flow (30 Frames) Conv Pooling of Image Frames + Optical Flow (120 Frames)	87.6 **88.2**
LSTM with 30 Frame Unroll (Optical Flow + Image Frames)	**88.6**

Table 3.7: UCF-101 results. The bold-face numbers represent results that are higher than previously reported results.

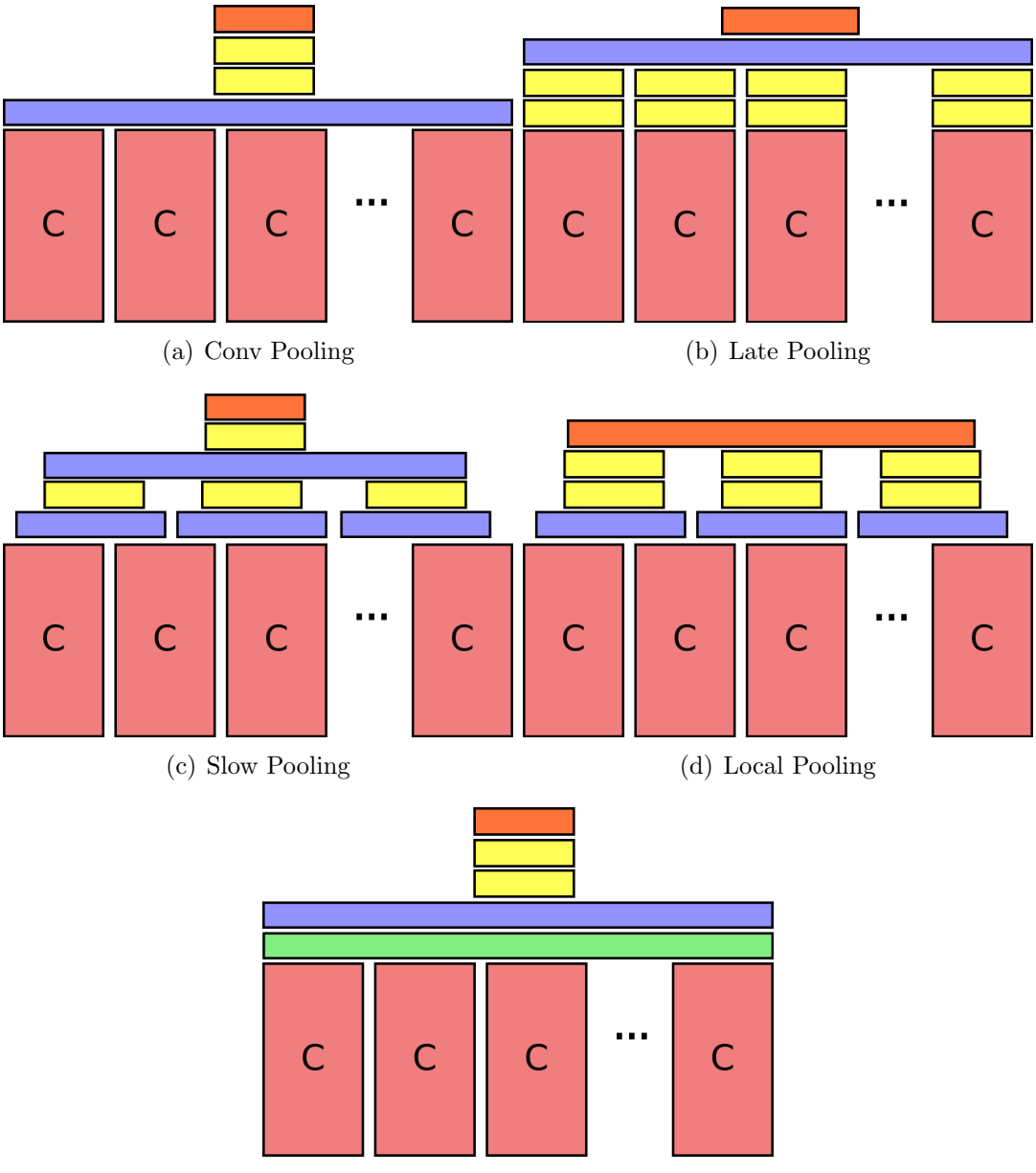
3.5 Conclusion

We presented two video-classification methods capable of aggregating frame-level CNN outputs into video-level predictions: Feature Pooling methods which max-pool local information through time and LSTM whose hidden state evolves with each subsequent frame. Both methods are motivated by the idea that incorporating information across longer video sequences will enable better video classification. Unlike previous work which trained on seconds of video, our networks utilize up to two minutes of video (120 frames) for optimal classification performance. If speed is of concern, our methods can process an entire video in one shot. Training is possible by expanding smaller networks into progressively larger ones and fine-tuning. The resulting networks achieve state-of-the-art performance on both the Sports-1M and

UCF-101 benchmarks, supporting the idea that learning should take place over the entire video rather than short clips.

Additionally, we explore the necessity of motion information, and confirm that for the UCF-101 benchmark, in order to obtain state-of-the-art results, it is necessary to use optical flow. However, we also show that using optical flow is not always helpful, especially if the videos are taken from the wild as is the case in the Sports-1M dataset. In order to take advantage of optical flow in this case, it is necessary to employ a more sophisticated sequence processing architecture such as LSTM. Moreover, using LSTMs on both image frames, and optical flow yields the highest published performance measure for the Sports-1M benchmark.

In the current models, backpropagation of gradients proceeds down all layers and backwards through time in the top layers, but not backwards through time in the lower (CNN) layers. In the future, it would be interesting to consider a deeper integration of the temporal sequence information into the CNNs themselves. For instance, a Recurrent Convolutional Neural Network may be able to generate better features by utilizing its own activations in the last frame in conjunction with the image from the current frame.



(a) Conv Pooling (b) Late Pooling

(c) Slow Pooling (d) Local Pooling

(e) Time-Domain Convolution

Figure 3.2: Different Feature-Pooling Architectures: The stacked convolutional layers are denoted by "C". Blue, green, yellow and orange rectangles represent max-pooling, time-domain convolutional, fully-connected and softmax layers respectively.

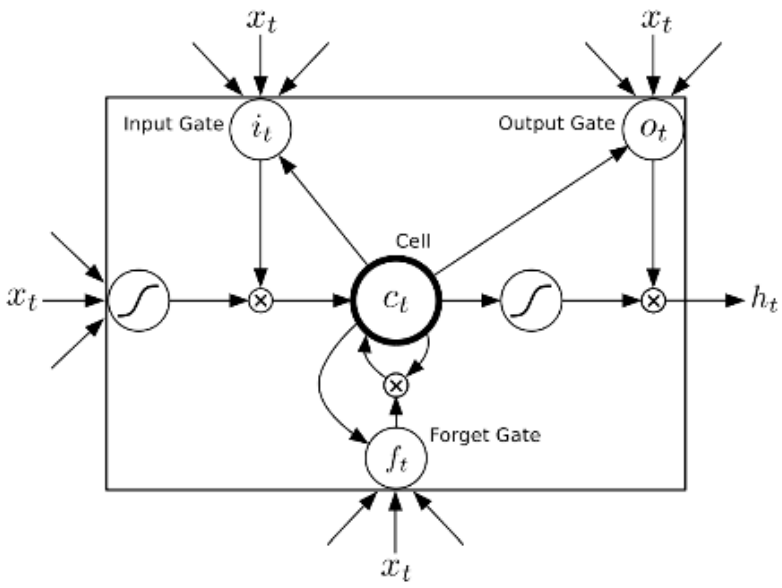


Figure 3.3: Each LSTM cell remembers a single floating point value c_t (Eq. 3.5). This value may be diminished or erased through a multiplicative interaction with the forget gate f_t (Eq. 3.4) or additively modified by the current input x_t multiplied by the activation of the input gate i_t (Eq. 3.3). The output gate o_t controls the emission of h_t, the stored memory c_t transformed by the hyperbolic tangent nonlinearity (Eq. 3.6,3.7). Image duplicated from [5].

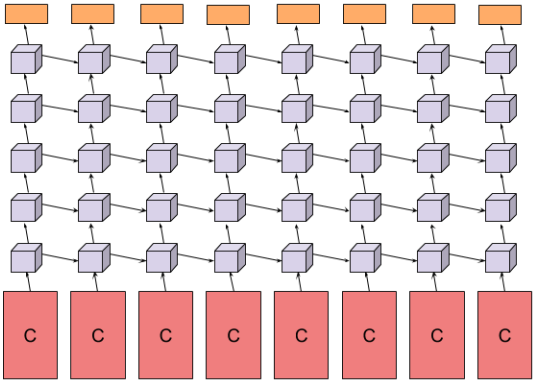


Figure 3.4: Deep Video LSTM takes input the output from the final CNN layer at each consecutive video frame. CNN outputs are processed forward through time and upwards through five layers of stacked LSTMs. A softmax layer predicts the class at each time step. The parameters of the convolutional networks (pink) and softmax classifier (orange) are shared across time steps.

Chapter 4: ActionFlowNet: Learning Motion Representation for Action Recognition

4.1 Motivation

Convolutional Neural Networks have demonstrated great success to multiple visual recognition tasks. With the help of large amount of annotated data like ImageNet, the network learns multiple layers of complex visual features directly from raw pixels in an end-to-end manner without relying on hand-crafted features. Unlike image labeling, manual video annotation often involves frame-by-frame inspection and temporal trimming of videos that are expensive and time consuming. This prohibits the technique to be applied to other problem domains like medical imaging where data collection is difficult.

We focus on effectively learning video motion representation for action recognition without large amount of external annotated video data. Following previous work [65–67] that leverages spatio-temporal structure in videos for unsupervised or self-supervised representation learning, we are interested in learning video representation from motion information encoded in videos in addition to semantic labels.

Learning motion representation on videos from raw pixels is challenging. With

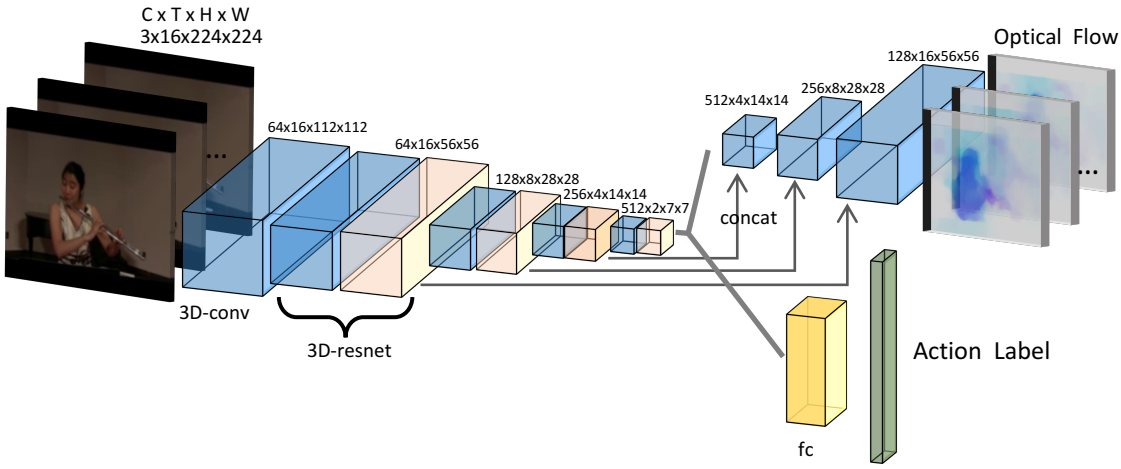


Figure 4.1: ActionFlowNet for jointly estaimting optical flow and recognizing actions. Orange and blue blocks represent ResNet modules, where blue blocks represents strided convolution. Channel dimension is not shown in the figure.

large scale datasets such as Sports-1M [45] and Kinetics [68], one could train a high capacity classifier to learn complex motion signatures for action recognition by extending image based CNN architectures with 3D convolutions for video action recognition [2,45,69]. However, while classification loss is an excellent generic appearance learner for image classification, it is not necessarily the most effective supervision for learning motion features for action recognition. As shown in [2], even with large amount of labeled video data, the model still benefits from additional optical flow input stream. This suggests that the model is ineffective in learning motion representation for action recognition from video frames, and thus alternative approach should be explored for learning video representation.

Two-stream convolutional neural networks, which separately learn appearance and motion by two convolutional networks on static images and optical flow respectively, show impressive results on action recognition [46]. The separation, however, fails to learn the interaction between the motion and the appearance of objects, and

introduces additional complexity of computing the flow to the classification pipeline. In addition, human visual system does not take optical flow as front end input signals but infer the motion from raw intensities internally. Therefore, we focus to learn both motion features and appearance directly from raw pixels without hand-crafted flow input.

Encouraged by the success on estimating optical flow with convolutional neural networks [70], we train a single stream feed-forward convolutional neural network - *ActionFlowNet* - for jointly recognizing actions and estimating optical flow. Specifically, we formulate the learning problem as multitask learning, which enables the network to learn both appearance and motion in a single network from raw pixels. The proposed architecture is illustrated in Figure 4.1. With the auxiliary task of optical flow learning, the network effectively learns useful representations from motion modeling without a large amount of human annotation. Based on the already learned motion modeling, the model then only requires action annotations as supervision to learn action class specific details, which results in requiring less annotation to perform well for action recognition.

Our experiments and analyses show that our model successfully learns motion features for action recognition and provide insights on how the learned optical flow quality affects action classification. We demonstrate the effectiveness of our learned motion representation on two standard action recognition benchmarks - UCF101 and HMDB51. Without providing external training data or fine-tuning from already well-trained models with millions of samples, we show that jointly learning action and optical flow significantly boosts action recognition accuracy compared to state-

of-the-art representation learning methods trained without external labeled data. Remarkably, our model outperforms the models trained with large datasets Sports-1M pretrained C3D by 1.6% on UCF101 dataset, showing the importance of feature learning algorithms.

4.2 Related Work

Over the past few years, action recognition accuracy has been greatly improved by learned features and various learning models utilizing deep networks. Two-stream network architecture was proposed to recognize action using both appearance and motions separately [46]. A number of follow up methods have been proposed based on two-stream networks that further improved action recognition accuracies [3, 71–74]. Our work is motivated by their success in incorporating optical flow for action recognition, but we focus on learning from raw pixels instead of relying on hand-crafted representations.

Optical flow encodes motion between frames and is highly related to action recognition. Our model is motivated by the success of FlowNet [70] and 3D convolutions for optical flow estimation in videos [1], but emphasizes on improving action recognition.

Pre-training the network with a large dataset helps to learn appearance signatures for action recognition. Karpathy *et al*. proposed a "Slow Fusion" network for large scale video classification [45]. Tran *et al*. trained a 3D convolutional neural network (C3D) with a large amount of data and showed the learned features are generic

for different tasks [69]. Recently, Carreira and Zisserman trained I3D models [2] on the Kinetics dataset [68] and achieved strong action recognition performance. In contrast, since training networks on such large scale datasets is extremely computationally expensive, we focus on learning from small amounts of labeled data. With only small amount of labeled data, we show that our model performs competitive to models trained with large datasets.

Leveraging videos as a source for unsupervised learning has been suggested to learn video representations without large labeled data. Different surrogate tasks have been proposed to learn visual representations from videos without any labels. Wang *et al*. trained a network to learn visual similarity for patches obtained from visual tracking in videos [75]. Misra *et al*. trained a network to differentiate the temporal order of different frames from a video [65]. Jacob *et al*. learned apperance features by predicting the future trajectories in videos [76]. Fernando *et al*. proposed Odd-One-Out networks (O3N) to identify video sequences that are out of order for self-supervised learning [67]. Our work, similarly, uses video as an additional source for learning visual representation. However, in contrast to previous work which focused on learning visual representations for a single image, we learn motion representations for videos which models more than a single frame. Vondrick *et al*. used a Generatie Adversarial Network to learn a generative model for video [66]. We focus on learning motion representations but not video generation.

Independent to our work, Diba *et al*. trained a two stream network with flow estimation [77]. They based their network on C3D with a two-stream architecture. Our work employs a single stream network to learn both appearance and motion.

While we both estimate motion and recognize actions in the same model, we focus on learning motion representations without pretraining on large labeled datasets and provide more analysis to learn flow representations for action recognition.

4.3 Approach

We propose a single end-to-end model to learn both motions and action classes simultaneously. Our primary goal is to improve action classification accuracy with the help of motion information; we use optical flow as a motion signature. Unlike previous methods that utilize externally computed optical flow as the input to their models, we only use the video frames for input and simultaneously learn the flow and class labels.

4.3.1 Multi-frame Optical Flow with 3D-ResNet

Fischer *et al*. proposed FlowNet [70] that is based on convolutional neural networks to estimate high quality optical flow. Tran *et al*. proposed to use 3D convolution and deconvolution layers to learn multi-frame optical flow from videos [1]. In addition, He *et al*. introduced residual networks (ResNet) to train a deeper convolutional neural network model by adding shortcut connections [10].

In addition to the benefit of easy training, ResNet is fully convolutional, so is easily applied to pixel-wise prediction of optical flow, unlike many architectures with fully connected layers including AlexNet [8] and VGG-16 [4]. In contrast to other classification architectures like AlexNet and VGG-16, which contains multiple

max pooling layers that may harm optical flow estimation, the ResNet architecture only contains one pooling layer right after conv1. We believe the reduced number of pooling layers makes ResNet more suitable for optical flow estimation where spatial details need to be preserved. Specifically, we use an 18 layers ResNet, which is computationally efficient with good classification performance [10].

Taking advantage of ResNet for flow estimation, we extend ResNet-18 to *3D-ResNet-18* for multi-frame optical flow estimation by replacing all $k \times k$ 2D convolutional kernels with extra temporal dimension $k \times k \times 3$, inspired by [1]. The deconvolution layers in the decoder are extended similarly. Skip connections from encoder to decoder are retained as in [70] to obtain higher resolution information in the decoder. Unlike [70], we only use the loss on the highest resolution to avoid downsampling in the temporal dimension. We do not apply temporal max pooling suggested in [1, 69], but use only strided convolutions to preserve temporal details. After the third residual block, the temporal resolution is reduced by half when the spatial resolution is reduced.

Future Prediction. In addition to computing the optical flow between the T input frames, we train the model to predict the optical flow on the last frame, which is the optical flow between the T^{th} and $(T+1)^{st}$ frames. There are two benefits of training the model to predict the optical flow of the last frame: 1) It is practically easier to implement a model with the same input and output sizes, since the output sizes of deconvolution layers are usually multiples of the inputs; and 2) Semantic reasoning is required for the model to extrapolate the future optical flow given the previous frames. This possibly trains the model to learn better motion

features for action recognition, as also suggested by previous work [76], which learned appearance feature by predicting the future.

Following [70], the network is optimized over the end-point error (EPE), which is the sum of L_2 distance between the ground truth optical flow and the obtained flow over all pixels. The total loss for the multiple frame optical flow model is the EPE of T output optical flow frames:

$$\sum_{t=1}^{T} \sum_{p} \|\mathbf{o}_{j,t,p} - \widehat{\mathbf{o}_{j,t,p}}\|_2, \tag{4.1}$$

where $\mathbf{o}_{j,t,p}$ is 2-dimensional optical flow vector of the t^{th} and the $(t+1)^{st}$ frame in the j^{th} video at pixel p.

Note that the T^{th} optical flow frame $\mathbf{o}_{j,t}$ is the future optical flow for the T^{th} and $(T+1)^{st}$ input frames, where the $(T+1)^{st}$ frame is not given to the model.

4.3.2 ActionFlowNet

Knowledge Transfer by Finetuning. Finetuning a pretrained network is a common practice to transfer knowledge from different datasets and tasks. Unlike previous work, where knowledge transfer has been accomplished between very similar tasks (image classification and detection or semantic segmentation), knowledge transfer in our model is challenging since the goals of pixel-wise optical flow and action classification are not obviously compatible. We transfer the learned motion by initializing the classification network using a network trained for optical flow estimation. Since the network was trained to predict optical flow, it should encode

motion information in intermediate levels which support action classification. However, finetuning a pretrained network is known to have the problem of catastrophic forgetting. Specifically, when training the network for action recognition, the originally initialized flow information could be destroyed when the network adapts the appearance information. We prevent catastrophic forgetting by using the multitask learning framework.

ActionFlowNet. To force the model to learn motion features while training for action recognition, we propose a multitask model *ActionFlowNet*, which simultaneously learns to estimate optical flow, together with predicting the future optical flow of the last frame, and action classification to avoid catastrophic forgetting. With optical flow as supervision, the model can effectively learn motion features while not relying on explicit optical flow computation.

In our implementation, we take 16 consecutive frames as input to our model. In the last layer of the encoder, global average pooling across the spatial-temporal feature map, with size $512 \times 2 \times 7 \times 7$, is employed to obtain a single 512 dimensional feature vector, followed by a linear softmax classifier for action recognition. The architecture is illustrated in Figure 4.1. The multitask loss is given as follows:

$$\text{MT-Loss}_j = \underbrace{-\mathbb{1}(y_j = \widehat{y_j}) \log p(\widehat{y_j})}_{\text{Classification Loss}} + \underbrace{\lambda \sum_{t=1}^{T} \sum_{p} \|\mathbf{o}_{j,t,p} - \widehat{\mathbf{o}_{j,t,p}}\|_2}_{\text{Flow Loss}}, \quad (4.2)$$

where $\mathbb{1}(\cdot)$ is a indicator function, y_j and $\widehat{y_j}$ are the groundtruth and predicted action labels respectively of the j^{th} video. λ is a hyper-parameter balancing the

classification loss and the flow loss, where optical flow estimation can be seen as a regularizer for the model to learn motion feature for classification.

Although previous work on multitask learning [78] suggests that sharing parameters of two different tasks may hurt performance, this architecture performs well since optical flow is known empirically to improve video action recognition significantly. In addition, our architecture contains multiple skip connections from lower convolutional layers to decoder. This allows higher layers in the encoder to focus on learning more abstract and high level features, without constraining them to remembering all spatial details for predicting optical flow, which is beneficial for action recognition. This idea is central to Ladder Networks [79] which introduced lateral connections to learn denoising functions and significantly improved classification performance.

It is worth noting that this is a very general architecture and requires minimal architectural engineering. Thus, it can be trivially extended to learn more tasks jointly to adapt knowledge from different domains.

ActionFlowNet Inference. During inference for action classification, optical flow estimation is not required since the motion information is already learned in the encoder. Therefore, the decoder can be removed and only the forward pass of the encoder and the classifier are computed. If the same backbone architecture is used, our model runs at the same speed as a single-stream RGB network without extra computational overhead. Since the optical flow estimation and flow-stream CNN are not needed, it is more efficient than two-stream counterparts.

4.3.3 Two-Frame Based Models

In this section, we propose various models that take two consecutive input frames. Experimenting with two-frame models has three benefits. First, when there are multiple frames in the input, it is difficult to determine whether the performance improvement comes from motion modeling or aggregating long term appearance information. Thus for better analysis, it is desirable to use the two frame input. Second, training two-frame models is computationally much more efficient than multi-frame models which take N video frames and output $N-1$ optical flow images. Third, we can measure the effectiveness of external large scale optical flow datasets, such as the FlyingChairs dataset [70], which provide ground-truth flow on only two consecutive frames, for action recognition.

Learning Optical Flow with ResNet. Similarly, we use ResNet-18 as our backbone architecture and learn optical flow. Like FlowNet-S [70], we concatenate two consecutive frames to produce a 6(ch) $\times$ 224(w) $\times$ 224(h) input for our two frames model. At the decoder, there are four outputs with different resolutions. The total optical flow loss is the weighted sum of end-point error at multiple resolutions per the following equation:

$$\sum_{r=1}^{4} \alpha_r \sum_{p} \|\mathbf{o}_{j,t,p}^{(r)} - \widehat{\mathbf{o}_{j,t,p}^{(r)}}\|_2, \tag{4.3}$$

where $\mathbf{o}_{j,t,p}^{(r)}$ is the optical flow vector of the r^{th} layer output and α_r is the weighting coefficient of the r^{th} optical flow output. We refer to this pre-trained optical flow estimation network as *FlowNet*.

We first propose an architecture to classify actions on top of the optical flow estimation network, which we call the *Stacked Model*. Then, we present the two-frame version of ActionFlowNet to classify the actions and estimate the optical flow, which we call the *ActionFlowNet-2F*.

4.3.3.1 Stacked Model

A straightforward way to use the trained parameters from FlowNet is to take the output of FlowNet and learn a CNN on top of the output, as shown in Figure 4.2. This is reminiscence of the temporal stream in [46] which learns a CNN on precomputed optical flow. If the learned optical flow has high quality, it should give similar performance to learning a network on optical flow.

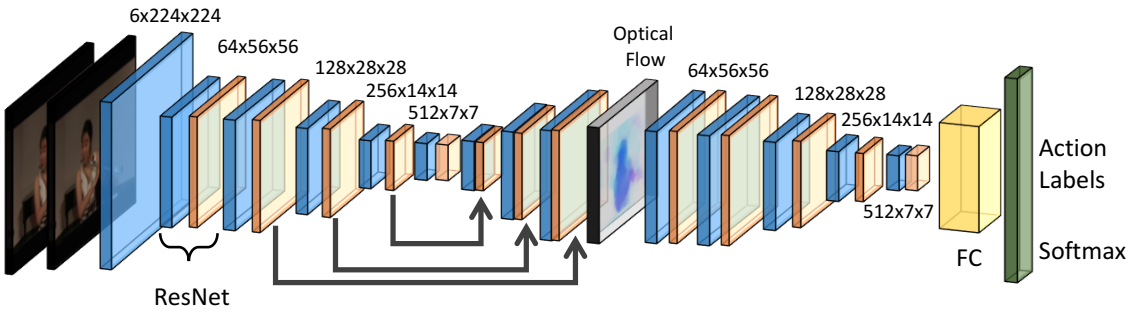


Figure 4.2: Network structure of the 'Stacked Model'.

Since the output of FlowNet has 4 times lower resolution than the original image, we remove the first two layers of the CNN (conv1 and pool1) and stack the network on top of it. We also tried to upsample the flow to the original resolution and use the original architecture including conv1 and pool1, but this produces slightly worse results and is computationally more expensive.

The stacked model introduces about 2x number of parameters compared to

the original ResNet, and is also 2x more expensive for inference. It learns motion features by explicitly including optical flow as an intermediate representation, but cannot model appearance and motion simultaneously, similar to learning a CNN on precomputed optical flow.

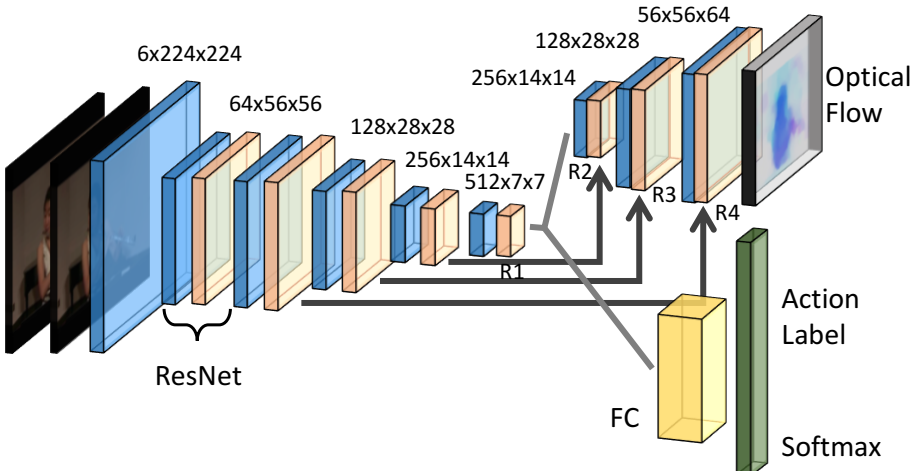


Figure 4.3: Network structure of the ActionFlowNet-2F

4.3.3.2 ActionFlowNet-2F

The multitask ActionFlowNet-2F architecture, as illustrated in Figure 4.3, is based on the two-frame FlowNet with additional classifier. Similar to ActionFlowNet, classification is performed by average pooling the last convolutional layer in the encoder followed by a linear classsifier.

Just as with the stacked model, the loss function is defined for each frame. For the t^{th} frame in the j^{th} video the loss is defined as a weighted sum of classification

loss and optical flow loss:

$$\text{MT-Loss}_{j,t} = \underbrace{-\mathbb{1}(y_j = \widehat{y_j}) \log p(\widehat{y_j})}_{\text{Classification Loss}} + \underbrace{\lambda \sum_{r=1}^{4} \alpha_r \sum_{p} \|\mathbf{o}_{j,t,p}^{(r)} - \widehat{\mathbf{o}_{j,t,p}^{(r)}}\|_2}_{\text{Flow Loss}}, \quad (4.4)$$

4.4 Experiments

4.4.1 Datasets

We use two publicly available datasets, UCF101 and HMDB51, to evaluate action classification accuracy. The UCF101 dataset contains 13,320 videos with 101 action classes [64]. The HMDB51 contains 6,766 videos with 51 action categories [50]. As the number of training videos in HMDB51 is small, we initialized our models trained on UCF101 and fine-tuned for HMDB51 similar to [46]

The UCF101 and HMDB51 do not have groundtruth optical flow annotation. Similar to [1], we use EpicFlow [80] as a psuedo-groundtruth optical flow to train the motion part of the network.

To experiment models with better learned the motion signature, we also use FlyingChairs dataset [70] as it has groundtruth optical flow since it is a synthetic dataset. The FlyingChairs dataset contains 22,872 image pairs and ground truth flow from synthetically generated chairs on real images. We use the Sintel dataset [81], which provides dense groundtruth optical flow, to validate the quality of optical flow models.

4.4.2 Experimental Setup

Overfitting Prevention. We use different data augmentations on different datasets and tasks. On the FlyingChairs dataset for optical flow estimation, we augment the data using multi-scale cropping, horizontal flipping, translation and rotation following [70]. On the UCF101 dataset for optical flow estimation, we use multi-scale cropping and horizontal flipping, but do not use translation and rotation in order to maintain the original optical flow distribution in the data. On UCF101 dataset for action recognition, we use color jittering [9], multi-scale cropping and horizontal flipping. Dropout is applied to the output of the average pooling layer before the linear classifier with probability 0.5.

Optimization and Evaluation. The models are trained using Adam [82] for 40,000 iterations with batch size 128 and learning rate 1×10^{-4}. For evaluation, we sample 25 random video segments from a video and run a forward pass to the network on the 10-crops (4 corners + center with their horizontal reflections) and average the prediction scores.

4.4.3 Improving Action Recognition

We first evaluate the action recognition accuracy by the various proposed two-frame models described in Section 4.3.3, and then the multi-frame models in Section 4.3.2, on both UCF101 and HMDB51 datasets. All models take RGB inputs only without external optical flow inputs. The recognition accuracies are summarized in Table 4.1.

Method	UCF101	HMDB51
Two-frame Models		
Scratch	51.3	23.9
FlowNet fine-tune	66.0	29.1
Stacked	69.6	42.4
ActionFlowNet-2F (UCF101)	70.0	42.4
ActionFlowNet-2F (FlCh+UCF101)	**71.0**	**42.6**
ImageNet pretrained ResNet-18	80.7	47.1
Multi-frame Models		
Multi-frame FlowNet fine-tune	80.8	50.6
ActionFlowNet (UCF101)	**83.9**	**56.4**
Sports-1M pretrained C3D [69]	82.3	53.5
Kinetics pretrained I3D [2]	95.6	74.8

Table 4.1: Action recognition accuracies of our models on UCF101 and HMDB51 datasets (split 1). FlCh denotes FlyingChairs dataset. "ActionFlowNet-2F (UCF101)" denotes its FlowNet part is pretrained on UCF101, and "ActionFlowNet-2F (FlCh+UCF101)" denotes its FlowNet part is pretrained on FlyingChairs dataset. All ActionFlowNets are then learned on UCF101 dataset for action and flow. For reference, we additionally show the results trained with large scale datasets [1, 2], but it is not directly comparable since our models are trained with significantly less annotation.

Two-frame Models. 'Scratch' is a ResNet-18 model that is trained from scratch (random initialization) using UCF101 without any extra supervision, which represents the baseline performance without motion modeling. 'FlowNet fine-tune' is a model that is pretrained from UCF101 for optical flow only, and then fine-tuned with action classification, which captures motion information by initialized FlowNet. 'Stacked' is a stacked classification model on top of optical flow output depicted in Figure 4.2. Its underlying FlowNet is trained with UCF101 and is fixed to predict optical flow, so only the CNN classifier on top is learned. 'ActionFlowNet-2F' is

the multitask model depicted in Figure 4.3, which is trained for action recognition and optical flow estimation to learn both motion and appearance. We trained two versions of ActionFlowNet-2F: one with FlowNet pretrained on UCF101 and one on FlyingChairs dataset.

As shown in the table, all proposed models - 'FlowNet fine-tune', 'Stacked' and 'ActionFlowNet-2F' significantly outperform 'Scratch' . This implies that our models can take advantage of the learned motion for action recognition, which is difficult to learn implicitly from action labels.

Both the Stacked model and two ActionFlowNet-2Fs outperform the finetuning models by a large margin (up to 5.0% in UCF101 and up to 13.5% in HMDB51). As all models are pretrained from the high quality optical flow model, the results show that the knowledge learned from previous task is prone to be forgotten when learning new task without multitask learning. With extra supervision from optical flow estimation, multitask models regularize the action recognition with the effort of learning the motion features.

While the Stacked model performs similarly to ActionFlowNet-2F when trained only on UCF101, ActionFlowNet-2F is much more compact than the Stacked model, containing only approximately half the number of parameters of the Stacked model. When ActionFlowNet-2F is first pretrained with FlyingChairs, which predicts better quality optical flow in EPE, and finetuned with the UCF101 dataset, it further improves accuracy by 1%. This implies that our multitask model is capable of transferring general motion information from other datasets to improve recognition accuracy further.

Our ActionFlowNet-2F still performs inferior compared to ResNet pretrained on ImageNet, especially in UCF101 (71.0% vs 80.7%) because of the rich background context appearance in the dataset. When evaluated on HMDB51, where the backgrounds are less discriminative, our ActionFlowNet-2F is only slightly behind the ImageNet pretrained model (42.6% vs 47.1%), indicating that our model learns strong motion features for action recognition.

Multi-frame Models. We train 16-frame ActionFlowNet on UCF101. The results are shown in the lower part of Table 4.1. By taking more frames per model, our multi-frame models significantly improve two-frame models (83.9% vs 70.0%). This confirms previous work [45, 83] that taking more input frames in the model is important.

Remarkably, without pretraining on large amounts of labeled data, our ActionFlowNet outperforms the ImageNet pretrained single frame model and Sports-1M pretrained C3D. Our ActionFlowNet gives 1.6% and 2.9% improvements over C3D on UCF101 and HMDB51 repsectively. The recently published I3D models [2] achieved strong performance by training on the newly released Kinetics dataset [68] with large amount of clean and trimmed labeled video data and performing 3D convolutions on 64 input frames instead of 16 frames. Although the I3D model achieved better results compared to previous work, their RGB model could still benefit from optical flow inputs, which indicates that even with large amount of labeled data the I3D model does not learn motion features effectively.

It should be noted that there is prior work that gives better results with the use of large scale datasets like ImageNet and Kinetics dataset [2], or with the help

of external optical flow input [46]. Those results are not directly comparable to us because we are using a significantly smaller amount of labeled data - only UCF101 and HMDB51. Nevertheless, our method shows promising results for learning motion representations from videos. Even with only a small amount of labeled data, our action recognition network outperforms methods trained with a large amount of labeled data with the exception of the recently trained I3D models [2] which used ImageNet and Kinetics dataset [68]. We envision the performance of ActionFlowNet would further improve when trained on larger datasets like Kinetics and taking more input frames in the model.

Method	UCF101 Accuracy
ResNet-18 Scratch	51.3
VGG-M-2048 Scratch [46]	52.9
Sequential Verification [65]	50.9
VGAN [66]	52.1
O3N [67]	60.3
OPN [84]	59.8
FlowNet fine-tuned (ours)	66.0
ActionFlowNet-2F (ours)	70.0
ActionFlowNet (ours)	**83.9**

Table 4.2: Results on UCF101 (split 1) from single stream networks with raw pixel input and without pretraining on large labeled dataset.

Comparison to state-of-the-arts. We compare our approach to previous work that does not perform pretraining with external large labeled datasets in Table 4.2 on UCF101. All models are trained only with UCF101 labels with different unsupervised learning methods. Our models significantly outperform previous work that use videos for unsupervised feature learning [65–67, 84]. Specifically, even with

(a) Image (b) Output flow from ActionFlowNet-2F (c) ImageNet Model: Appearance Only (d) ActionFlowNet-2F: Motion and Appearance

Figure 4.4: Visualization of important regions for action recognition. Our ActionFlowNet-2F discovers the regions where the motions are happening to be important while 'Appearance Only' captures discriminative regions based on the appearance.

only our two-frame fine-tuned model on UCF101, the model obtain more than 5.9% improvement compared to Sequential Verification, VGAN and O3N, indicating the importance of motion in learning video representations. When combined with multitask learning, the performance improves to 70.0%. Finally, when extending our model to 16 frames by 3D convolutions, the performance of ActionFlowNet further boost to 83.9%, giving a *23.6% improvement* over the best previous work. This shows that explicitly learning motion information is important for learning video representations.

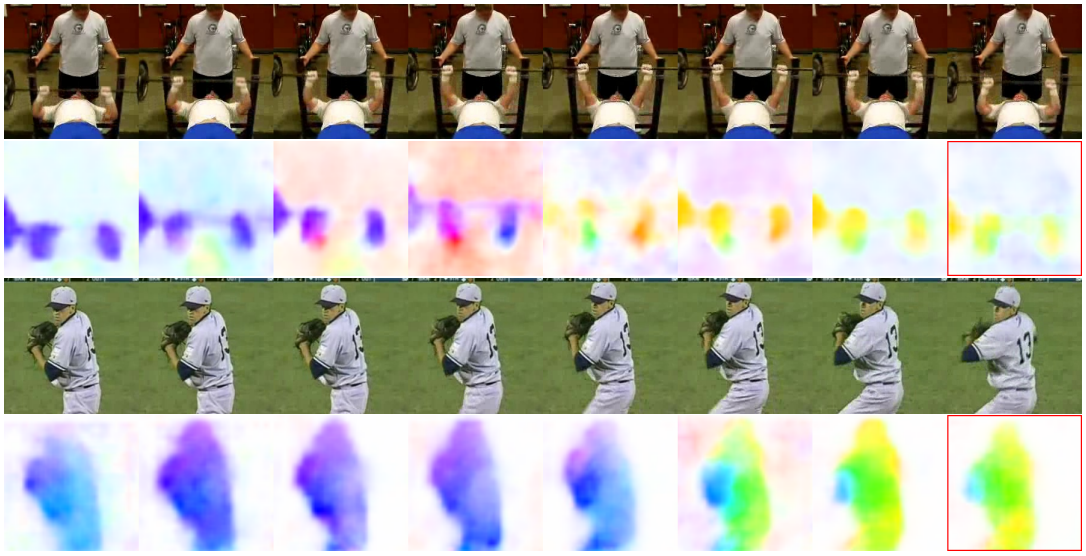


Figure 4.5: Optical flow and future prediction outputs from our multi-frame model. The 1st and 3rd row shows an example of input videos, and the 2nd and 4th row shows the corresponding optical flow outputs. The last optical flow output frames (in red border) are extrapolated rather than computed within input frames. Only last 8 frames are shown per sample due to space limit.

4.4.3.1 Learning Motions for Discriminative Regions

We visualize what is learned from the multitask network by using the method from [11] by using a black square to occlude the frames at different spatial locations and compute the relative difference between classification confidence before and after occlusion. We visualize the two-frame based ActionFlowNet-2F for more straightforward visualization.

We compare the discriminative regions discovered by our multitask network with ones by the ImageNet pretrained ResNet-18, which only models the discriminative appearances without motion. Figure 4.4 shows example results. The visualization reveals that our model focuses more on motion, while the ImageNet pretrained network relies more on background appearance, which may not directly relate to the

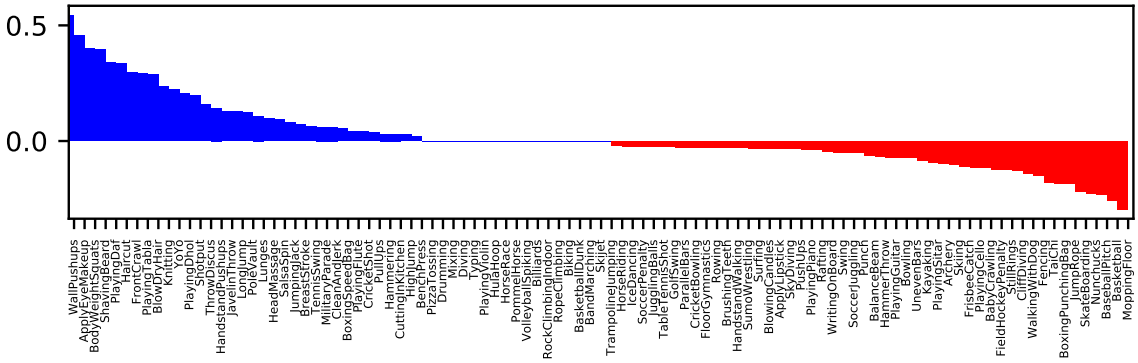
action itself. However, when appearance is discriminative - for example the writing on the board in the last example - our model can also focus on appearance, which is not possible for models that learn from optical flow only.

4.4.3.2 Optical Flow and Future Prediction

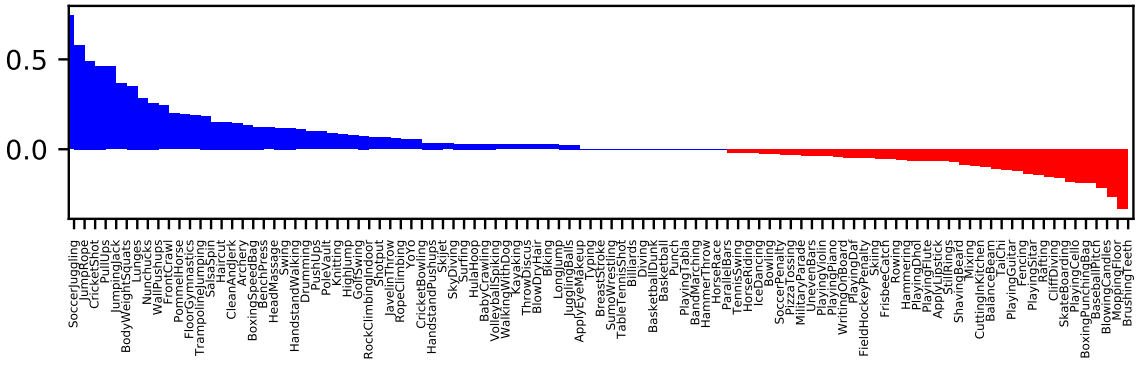
Figure 4.5 shows the optical flow estimation and prediction results from our multi-frame model. Although the model does not have accurate optical flow groundtruth for training, the optical flow quality is fairly good. The model predicts reasonable future optical flow, which shows semantic understanding from the model to the frames in addition to simply performing matching between input frames.

4.4.3.3 Classes Improved By Learning Motions

We compare the per class accuracy for ActionFlowNet, ImageNet pretrained model and C3D. Not all action classes are motion-centric - objects and their contextual (background) appearances provide more discriminative information for some classes [85], which can greatly benefit from large amounts of labeled data. As shown in Figure 4.6, our model better recognizes action classes with simple and discriminative motion like WallPushups and ApplyEyeMakeup, while C3D and ImageNet models perform better on classes with complex appearance like MoppingFloor and BaseballPitch.



(a) ActionFlowNet vs C3D



(b) ActionFlowNet vs ImageNet pretrained ResNet-18

Figure 4.6: Classwise accuracy improvement by ActionFlowNet over pretrained models. The blue bars show positive improvements and the red ones show otherwise.

4.4.4 Recognition and Optical Flow Quality

In this section, we study the effects of different optical flow models for action recognition based on the two-frame models. We train our optical flow models on FlyingChairs or UCF101 and evaluate their accuracies on the Sintel dataset (similar to [70] that trains the model on FlyingChairs but tests on other datasets).

We investigate how the quality of the learned optical flow affects action recognition. Since optical flow in the multitask model is collaboratively learned with the recognition task, the quality of optical flow in the multitask model does not

directly affect recognition accuracy. Thus, we use our Stacked model learned with different datasets, fix the optical flow part and train the classification part in the network shown in Figure 4.2. We compare the end-point-error of different optical flow learners and the corresponding classification accuracy in Table 4.3.

Method	EPE on Sintel	Classification Accuracy (%)
Stacked on FlyingChairs	**9.12**	51.7
Stacked on UCF101	11.84	**69.6**
ResNet on EpicFlow	6.29	77.7

Table 4.3: Comparison between End-Point-Error (EPE, lower is better) and the classification accuracy. Interestingly, better optical flow does not always result in better action recognition accuracy. Refer to the text for discussion.

(a) Frame with movements highlighted with red

(b) EpicFlow

(c) FlowNet on FlyingChairs

(d) FlowNet on UCF101

Figure 4.7: Qualitative comparison of flow outputs. It shows an example of small motion, where the maximum magnitude of displacement estimated from EpicFlow is only about 1.6px. FlowNet trained on FlyingChairs dataset fails to estimate small motion, since the FlyingChairs dataset consists of large displacement flow.

Action Recognition with Learned Flow. Surprisingly, even with lower end-point-error the Stacked model pretrained on FlyingChairs performs significantly worse than the one pretrained on UCF101 dataset (51.7% vs 69.6%), as shown in Table 4.3. Compared to the model directly taking high quality optical flow as input

(77.7%), our models are still not as good as training directly on optical flow. We believe this is because the quality of learned optical flow is not high enough.

To understand how the learned optical flow affects action recognition, we qualitatively observe the optical flow performance in Figure 4.7. Even though the endpoint error on Sintel of the FlowNet pretrained on FlyingChairs is low, the estimated optical flow has lots of artifacts in the background and the recognition accuracy on top of that is correspondingly low. We believe the reason is that the FlyingChairs dataset mostly consists of large displacement flow, and therefore the model performs badly on estimating small optical flow, which contributes less in the EPE metric when averaged over the whole dataset. This is in contrast to traditional optimization based optical flow algorithms that can predict small displacements well but have difficulties for large displacements.

In addition, traditional optical flow algorithms such as TV-L1 and EpicFlow explicitly enforce smoothness and constancy. They are able to preserve object shape information when the flow displacements are small, which is important for action recognition. While our models perform comparably to traditional optical flow algorithms in terms of endpoint error, our model is not optimized for preserving flow smoothness. This shows that end-point-error of optical flow in public dataset may not be a good indicator of action classification performance, since shape preservation is not accounted for in the metric.

4.5 Conclusion

We presented a multitask framework for learning action with motion flow, named *ActionFlowNet*. By using optical flow as supervision for classification, our model captures motion information while not requiring explicit optical flow computation as input. Our model significantly outperforms previous feature learning methods trained without external large scale data and additional optical flow input.

Chapter 5: Temporal Difference Networks for Video Action Recognition

5.1 Motivation

Video action recognition is one of the fundamental problems in computer vision. One of the main challenges is to model the complex temporal relations in addition to image appearance. Unlike images, which have a fixed size input, videos and their corresponding actions are of arbitrary length. While convolutional neural networks (CNNs) have been very sucessful in image based recognition tasks [8–11], it is still unclear how to model the temporal evolution of videos effectively by deep networks. In recent work, there are two main approaches to model the temporal dimension of videos - model the short term motion and model the longer term temporal relations.

Motion modeling usually focuses on video clips that span less than one second. Optical flow has been used extensively [3, 46, 71–73, 83] as a motion representation and shows improvement in conjunction with RGB image inputs for action recognition. While it is an effective hand-crafted feature as input for CNNs to recognize action, it is still unclear how to learn motion features directly from raw pixels with-

out hand-crafted inputs. Furthermore, optical flow, which is defined at the pixel level, can only model short term motion effectively, but does not capture longer range and high level temporal dependency.

Another line of work models the high level representation of frames from the output of deep CNNs by models such as Long-short-term memory (LSTM). These approaches rely on the strong high level CNN features but do not consider the low level correspondence between frames. While they can sucessfully combine semantic information over long videos, we believe that CNN features capture high level abstract concepts and are unsuitable for learning motion, as the apperance of consecutive frames in a video might be very similar and the precise spatial details may be lost after multiple pooling layers in the network.

We believe successful video action recognition models require temporal reasoning on multiple levels of appearance. Current approaches, which focus on modeling either the long term temporal relations or short term motion, are insufficient as they model on a fixed level of appearance. In this work, we propose a novel deep network architecture - *Temporal Difference Network (TDN)* - to model temporal relations in videos. Instead of leveraging independent techniques for modeling short term motion [46] and high level temporal relations [3, 72, 83], our framework unifies these two strategies and learns video motion representations from multiple levels of apperance abstraction, leveraging low level to high level image features.

While pixel level motion can be represented using optical flow, the motion of mid-level concepts is not well-defined. We consider an alternative representation of motion - *Eulerian motion* - which is defined in terms of image differences. In

addition to differences of raw input images, we consider the Eulerian motion of mid-level to high-level image features, and then combine the multiple layers of motion information in a single CNN. By forcing the network to model the motion directly instead of implicitly learn from class label supervision, our network effectively models the temporal relations between frames rather than just aggregates apperance information over a video.

We test our model on three public video classification benchmarks and achieve state-of-the-art results. Remarkably, the improvement in the RGB stream is significant, demonstrating the effectiveness of modeling temporal relations in our network from raw pixels.

5.2 Related Work

Video action recognition has been studied extensively in computer vision. Please refer to the survey by Poppe [86] and Wu *et al*. [87] for complete background. Recent work on action recognition falls into two main categories: 1) long range temporal relations modeling and 2) short term motion representation.

Long range temporal relations modeling. Recurrent neural networks have been used to model the temporal relations in video sequences [83, 88–91]. Ng *et al*. learned long range temporal information in videos by LSTMs for long video classification [83]. Donahue *et al*. similarly learns LSTM models for action recognition. Mahasseni and Todorovic regularized the LSTM model with 3D human-skeleton sequences [89]. Srivastava *et al*. learns LSTM in an unsupervised manner

for action recognition. Another strategy to aggregate information across frames is pooling. Ng *et al*. applied max pooling at the last convolutional layer for long video classification [83]. Wang *et al*. proposed temporal segment networks to combine the final categorical classification scores from multiple frames by average pooling [72]. Wang *et al*. represent actions by a transformation from the initial state to final state of videos and treat the temporal location of the states as latent variables [3]. Rank pooling has been proposed to capture temporal evolution of videos by considering the temporal ordering of video frames [92–94].

All these methods learn temporal relations only based on high level CNN features, i.e. the last convolutional layer or fully connected layers, which lose detailed spatial information and do not effecitvely learn small motions. In contrast, our work operates over multiple level of appearance within a single network, and learns both small motions and high level temporal relations.

Ballas *et al*. exploit multiple layers of image features to train GRU-RNN for video representations [91]. Instead of using recurrent networks, we train a feed-forward network to directly model the motion between frames.

Many previous research leverages the temporal structure in videos for action recognition [95–99]. Niebles *et al*. uses latent SVM to discover the temporal structure of videos [95]. Tang *et al*. model the temporal structure using a varient of HMM [96]. Pirsiavash and Ramanan represent actions by segmental grammars [97]. However, their approaches are not end-to-end learnable for temporal structure modeling and thus cannot fully utilize the advantage of deep networks.

Short term motion representation. Karpathy *et al*. trained a "Slow Fu-

sion" network to learn motion from large number of labeled videos [45]. Ji *et al*. and Tran *et al*. use multiple 3D convolutional layers for learning motion features from raw pixels [44, 69]. Varol *et al*. train 3D convolutional networks for longer clips and show improvements in recognition. Carreira and Zisserman recently trained 3D convolutional networks [2] on newly released Kinetics dataset [68]. However, training these models requires large labeled video datasets and is extremely computationally expensive, which limits the size of the network that could be trained. Our work reuses the pretrained static image appearance model and learns motion based on that, which could easily adapt to very deep image based convolutional networks.

Bilen *et al*. computed dynamic image by rank pooling as motion representation as input to the network [100]. Our Temporal Difference Networks learn motion from multiple levels and is an end-to-end model.

Simonyan and Zisserman feed stacked optical flow frames as input to the network and showed that combining optical flow network and ImageNet pretrained network significantly improves action recognition performance over the single frame appearance model [46]. We also train networks for both RGB and optical flow frames as input, but in addition to modeling short term motion, our network learns higher level temporal relations as well. Feichtenhofer *et al*. further improves the original two stream networks by combining two input modalities into a single network [71, 73]. Our network architecture is similar to [73], but we exploit the difference of image features to learn temporal relations instead of using optical flow as input for motion.

Image difference, also known as the Eulerian motion, has been used to represent motion of images. Sun *et al*. and Wang *et al*. exploited image differences as

inputs to the network [72, 101] to complement RGB inputs. Xue *et al*. represented motion by Eulerican motion to synthesize future video frames from a single image [102]. Villegas *et al*. decompose videos into motion and content by representing motion as image differences for future video sequence prediction [103]. Wu *et al*. magnify the Eulerian motion of videos to visualize subtle change in videos [104]. Extending these previous work, we compute the Eulerian motion not only of the input frames, but also the intermediate CNN features to capture high level motion.

5.3 Approach

5.3.1 Eulerian Motion of Features

The image difference, also known as the Eulerian motion, of two images is defined as:

$$v = I_2 - I_1$$

where I_1 and I_2 are consecutive frames in a video. While image differences capture some short term motion information, they do not effectively model longer range temporal relation in videos.

Instead, we encode motion additionally by differences of image features, which can be regarded as the Eulerian motion of image features:

$$v^{(\ell)} = f^{(\ell)}(I_2) - f^{(\ell)}(I_1)$$

where $f^{(\ell)}(I)$ is an appearance feature extractor for image I at layer ℓ in a CNN.

Compared to raw images, the image features are more robust to translation and appearance changes, and thus their differences are more suitable for capturing higher level temporal relations across longer periods of time. The difference of features can also be seen as a special case of rank pooling [92], where only two frames are considered instead of multiple frames in the video. Similarly, image difference can be seen as a special case of dynamic image [100] from two frames.

In a convolutional neural network (CNN), different layers capture different levels of appearance abstraction [11]. For shorter time periods, the difference of lower level features should be more informative as small motion can be captured better in lower layers; and for longer time periods, higher level features should be more useful to model the temporal relations between frames. While the best level of abstraction is unclear and situation dependent, we use the differences over multiple layers in the CNN features to capture temporal relations over all scales.

5.3.2 Temporal Difference Network

In this section, we describe the architecture of our proposed Temporal Difference Network (TDN).

Jointly learning motion and appearance on video action recognition datasets is challenging, since image appearance provides abundant information to the model to overfit the dataset, while ignoring motion which corresponds to the actual action. While separating the motion into an independent optical flow CNN shows great improvement to the RGB inputs [46], motions, however, clearly depend on appear-

anace which suggests appearance model should help learning motion. Therefore, in our proposed Temporal Difference Network, we leverage the image appearance models to learn video motions, and force the network to learn the motion by explicitly model the Eulerian motion of image features as inputs. In addition to modeling motion in a fixed layer, we aggregate the multi-level feature differences in one single network.

We build the Temporal Difference Network on a well trained image appearance model. We use a 50 layers Residual Network (ResNet-50) as our base model [10], which provides a good trade-off between accuracy and training time. Our approach is flexible to adopt other architectures and further recognition improvement is possible with deeper and more accurate pretrained networks. The TDN consists of two subnetworks: the image subnetwork and the difference subnetwork. The architecture is illustrated in Figure 5.1.

Image subnetwork. The image subnetwork is a standard residual network which takes a single frame (or stacked consecutive frames) as input. At the end of the network, the prediction scores from different frames are averaged before softmax similar to [72]. The parameters of the image subnetwork of different input frames are shared. This is essential to force the network to learn from the feature differences instead of using the appearance from one of the image subnetworks.

Difference subnetwork. The difference subnetwork has the exact same size as the image subnetwork. At the layers right before reducing the spatial resolution, we compute the Eulerian motion of the features from the image subnetwork, which is the difference between two feature maps. The differences are then combined with

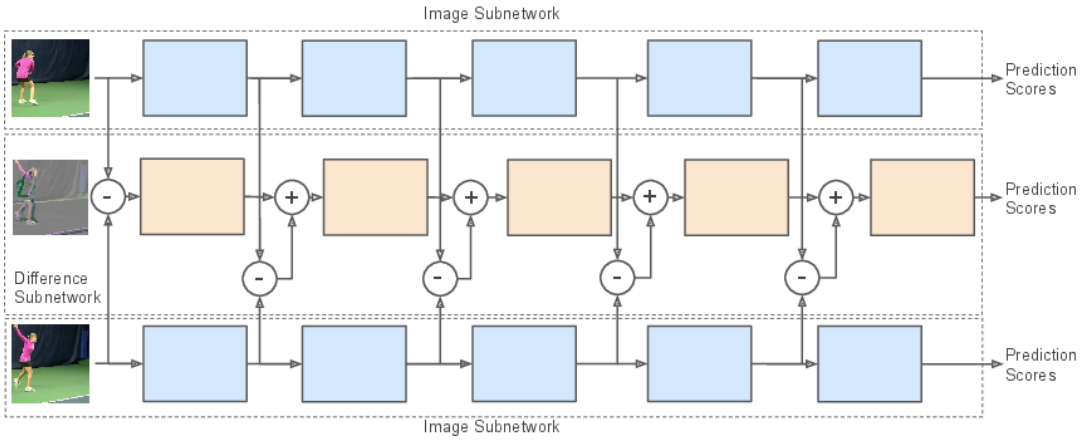


Figure 5.1: The figure shows our Temporal Difference Network architecture. Each rectangle in the figure represents the convolutional layers with max pooling or stacked residual modules. The blue blocks represent layers in the image subnetwork and the orange blocks represent layers in the difference subnetwork. The circles represent element-wise add or subtraction operation. At the layers before reducing the spatial resolution, the difference of the convolutional feature maps from the image subnetwork is computed and then added into the difference subnetwork. The class prediction scores are obtained at the end of each subnetwork.

the bottom up activations from the difference subnetwork by element-wise addition, inspired by [73], as the difference of features and the CNN features have the exact same dimensions. Specifically, in our implementation the difference features have spatial resolution of 224×224 (input images), 56×56, 28×28 and 14×14. More layers in the network can be used for computing the image feature differences. The difference in the last convolutional layer is not used because it would be immediately fed into the final classifier without additional transformation, which makes it unlikely to help classification.

While the difference in the features is basically a linear operation and could be learned by 3D convolutions, which has been explored previously in [69, 83], it is not an effective method to model the motion between frames since the frame based image

appearance already provides extremely strong cues to the classification task, and the model can simply learn to aggregate the appearance context for classification instead of learning the action itself. By explicitly taking the difference of the features, the network is forced to focus on the motion and temporal evolution of the videos, instead of relying on strong appearance cues for classification.

We employ a simple summation to combine the bottom up activations of the difference subnetwork and the differences of image features. We have additionally experimented with adding a 1×1 convolutional layer to the difference features before the addition, or dynamicly computing the weights of two inputs by a gating mechanism, but observed no significant difference in performance.

By taking the difference of higher level features, we are able to not only model the motion between consecutive frames, but also over longer time periods. Since the mid-level features are more robust to translation and view point changes, the model can then focus on the difference in mid-level concepts like pose. Therefore, we sparsely sample frames throughout the whole video as input, instead of only considering consecutive frames.

Final prediction. There are many ways to combine the final classification outputs of the image subnetwork and difference subnetwork during testing. We found that simple averaging works very well and this is used in all experiments.

TDN with TSN. Our TDN can be trained with more than two frames as a temporal segment network (TSN) [72]. As a TSN with s input snippets, our TDN produces s outputs from image subnetworks for each frame, and $s-1$ outputs from difference subnetworks for each pair of adjacent frames. Following [72], we

use average pooling as the segmental consensus function to combine the prediction scores of each network during training. Figure 5.2 illustrates an example with three input frames ($s = 3$). By taking more frames, the network captures more context in the video during training before classification. This is important especially for training the difference subnetwork, since the difference of only two video frames may not have enough information for recognizing actions where TSN reduce the noise in training by considering more frames at once.

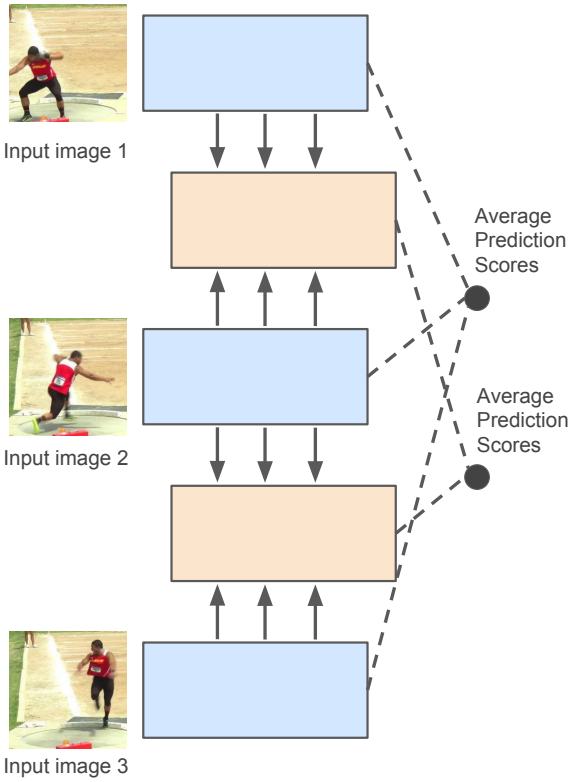


Figure 5.2: Temporal Difference Network as temporal segment network with $s = 3$ input snippets. The blue and orange blocks represent the image subnetworks and difference subnetworks respectively. The class prediction scores of image subnetworks and difference subnetworks are averaged across frames independently.

5.3.3 Fusion from Multiple Modalities

Optical flow has shown improvements in conjunction with RGB inputs for video action recognition performance. Our model can be applied to different modalities including RGB and optical flow, although we expect more improvements from the RGB network as optical flow already encodes motion. Long term temporal relations, which are not encoded in optical flow, can be learned through our framework. Following previous work [72], we separately train two networks for RGB inputs as spatial stream and stacked optical flow inputs as temporal stream. At test time, we compute the weighted average of prediction scores with RGB:Flow as 1:1.5. We use the confidence scores before softmax for fusion after ℓ_1-normalization.

5.3.4 Training

During training, we randomly sample frames throughout the videos as in temporal segment network [72]. The input videos are split into s approximately equally sized segments, and k consecutive frames ($k = 1$ for RGB input and $k = 5$ for optical flow) are randomly picked from each segments as an input snippet.

Cross Modality Pretraining. Following [72, 83], which suggests initializing the optical flow network with image pretrained model, we initialize the weight of convolutional layers in the difference subnetwork with ImageNet pretrained model. Since the inputs to the difference subnetwork, which are the differences of image features, contain similar spatial structure to the image features, ImageNet pretrained models should possess useful representation for modeling the motion and thus help

the training by initialization.

Two phase Training. To improve training stability, we employ a two phase training strategy. We first fix the image subnetwork and only train the classifier in the image subnetwork and the entire difference subnetwork. After convergence, we jointly finetune both image subnetwork and difference subnetwork.

When using optical flow inputs, we initialize the "image" subnetwork with a pretrained temporal segment network on optical flow inputs, and the difference subnetwork is still initialized with ImageNet pretrained weights. We similarly fix the already trained "image" subnetwork and train the difference subnetwork first, and then finetune the entire network.

5.4 Experiments

5.4.1 Datasets

We test our Temporal Difference Networks on three video datasets: HMDB51, ACT and FCVID.

The HMDB51 dataset [50] contains 6,766 video clips taken from mostly movies with 51 action classes. The standard three splits averaged accuracy is reported. This is a challenging action recognition dataset since many action classes, for example "turn" and "kiss", cannot be recognized from a static frame or backgroud context.

The ACT dataset [3] contains 11,234 video clips with 43 action classes. The videos are donwloaded from the web and then labeled by a commercial crowdsourcing organization. We evalute our model on the first task for the dataset, which is the

standard action classification proposed in [3]. We follow the train and test split by the dataset authors with 7,260 training videos and 3,974 for testing.

The Fudan-Columbia Video Dataset (FCVID) [105] contains 91,223 web videos with 239 categories including social events, procedural events, objects and scenes. The average video length is 167 seconds. It is computationally very expensive to process such a large dataset, so the frames are sampled every three seconds and resized to 256×256. We evaluate our models using the same train and test split as [105], which contains approximately half of the videos for training and half for testing, and compute the mean average precision (mAP) across categories.

We do not test on the UCF101 dataset [64] as it heavily relies on appearance and context information, and the performance has already been saturated with more than 94% accuracy by [72, 73] and 98% with pretraining on the Kinetics dataset [2, 68].

5.4.2 Implementation

We implement our network with Torch7 [106] using multiple GPUs for data-parallelism. SGD is used for training with mini-batch size 128 and momentum set to 0.9. To regularize the network on small datasets, we follow the good practice from [72] and use high dropout rate (drop probability 0.8), corner cropping, scale jittering and partial-BN, which fixes the mean and variance in batch normalization layers except the first one, to train the model on the HMDB51 and ACT datasets. We also apply weight decay with rate 0.0005 and color jittering [9] for data aug-

mentation.

We roughly follow [72] for numbers of training iterations and learning rate decay schedules to train the network for the HMDB51 dataset. As the size of ACT is similar to UCF101, we adopt their settings for training on ACT. For FCVID, we first train the TSN for 30,000 iterations with initial learning rate 0.01, and divide the learning rate by a factor of 10 every 10,000 iterations. The TDN is trained with the same settings as TSN, and we jointly fine-tune the whole network for another 10,000 iterations. We use $s = 3$ input snippets for ACT and FCVID, and $s = 2$ for HMDB51.

For the temporal stream networks, we compute the optical flow with the TV-L1 algorithm [107] using the OpenCV implementation with CUDA and save the flow as images after discretized into $[0, 255]$ range following [46]. We sample 5 consecutive optical flow frames as input to the flow network following [72].

Druing testing, we randomly sample 25 clips and perform 10 crop data augmentation, which crop the 4 corners and 1 center with their horizontal reflections, and compute the average of the prediction scores as final prediction.

5.4.3 Results

We present the results of our Temporal Difference Networks on various datasets. For fair comparisons, we train ResNet-50 temporal segmental networks [72] in all three evaluation datasets as our baselines.

5.4.3.1 HMDB51 Dataset

We compare our TDN with previous two-stream based networks including [3, 46, 71–73, 101, 108–110] that uses ImageNet as the only external dataset. The 3 splits averaged accuracy of the models are shown in Table 5.1. We exclude the fusion results with extra input modality like warped optical flow fields or hand-crafted features like improved dense trajectories for fair comparison. We compare to methods that only uses ImageNet as external datasets. Since different based networks are used in previous work including VGG, BN-Inception and ResNet-50, we additionally trained temporal segmental networks with ResNet-50 as baseline for comparison. We would also want to note that the recent work ST-ResNet [73] is also based on ResNet-50 and therefore can be directly compared.

Our implementation of TSN with ResNet-50 is better than the previous state-of-the-art in [72], which verifies the strength of our baseline. Our Temporal Difference Networks further improve over TSNs on both RGB and optical flow streams. Remarkably, our model significantly improves the RGB network by 4.4%. The improvement on the temporal stream with optical flow inputs is marginal, which is reasonable since the stacked optical flow already encodes motion information. Overall, our TDN achieves better accuracy than previous work with improvement of 1.9% (70.4% vs 68.5%).

Recently, Carreira and Zisserman trained I3D models [2] on newly released Kinetics dataset [68]. While they obtained strong recognition performance, their models have access to external video data thus could not be directly compared.

Since training on Kinetics dataset is extermely computationally expensive, we leave the experiments with Kinetics dataset as future work.

Method	RGB	Flow	Fusion
Two Stream [46]	40.5	54.6	59.4
Two Stream (VGG) [3, 4]	42.2	55.0	58.5
$F_{ST}CN$ (SCI fusion) [101]	-	-	59.1
Actions $\sim$ Transformation [3]	44.1	57.1	62.0
TDD + FV [108]	50.0	54.9	63.2
Key Volumne Mining [109]	-	-	63.3
LTC [110]	-	59.0	64.8
Two-stream Fusion [71]	-	-	65.4
ST-ResNet [73]	-	-	66.4
BN-Inception TSN [72]	51.0	64.2	68.5
ResNet-50 TSN	51.1	64.6	69.6
ResNet-50 TDN (ours)	**55.5**	**64.8**	**70.4**

Table 5.1: Classification accuracies on HMDB51 (3 splits average). Our model achieves state-of-the-art accuracy on the HMDB51 dataset. In particular, the RGB model significantly improves the TSN baseline.

5.4.3.2 ACT Dataset

We test our models on ACT and observe similar improvement. The evaluation results are shown in Table 5.2. Our ResNet-50 TSN baseline is the better than previous two-stream based networks with precondition and effect modeling by [3], and our TDNs again outperform the TSN baseline. The improvement in RGB stream is especially significant (75.9% vs 72.0%), showing that our model is capable of learning motion from the difference subnetwork. Overall, our TDNs improve on both RGB and optical flow inputs, and give slight improvement to TSN after fusion. Our model performs significantly better than previous work by a large margin (85.1% vs 80.6%).

Method	RGB	Flow	Fusion
Two Stream	66.8	71.4	78.7
LSTM + Two Stream	68.7	72.1	78.6
Actions $\sim$ Transformation [3]	69.5	73.7	80.6
ResNet-50 TSN	72.0	76.1	84.3
ResNet-50 TDN (ours)	**75.9**	**77.0**	**85.1**

Table 5.2: Performance comparison for the first task on ACT dataset. All baselines are trained by [3] with VGG-16 [4]. Our models significantly outperform previous methods.

5.4.3.3 FCVID

We compare our models to previous results reported by [105]. They provide strong baselines by combining static CNN features, improved dense trajectories and audio features with various fusion techniques. In particular, they proposed rDNN to exploit features and class relationships with deep networks to combine the predictions from multiple modalities.

As the dataset is very large and computing optical flow for the whole dataset is very time consuming, we only train our network for raw image inputs. We lower the weight decay rate to 0.0001 and do not use regularization techniques like dropout, corner cropping and partial-BN suggested in [72] since the dataset is already very large.

The recognition results are summarized in Table 5.3. The TSN baseline alone is already much better than the previous state-of-the-art rDNN from [105], which

fused multiple features, by 5.8% in mAP. We believe the difference in performance should be due to: 1) the advancement in CNN architecture (ResNet vs AlexNet); 2) end-to-end finetuning rather than feature extraction in [105]; and 3) training as TSN rather than single frame network.

Our TDN further improves the TSN model significantly by 1.9%, which demonstrates that our network is able to learn temporal relationships effectively even on large scale settings with unconstrained and noisy videos.

Method	mAP (%)
Static CNN [105]	63.8
rDNN (Static CNN + Motion + Audio) [105]	76.0
ResNet-50 TSN	81.8
ResNet-50 TDN (ours)	**83.7**

Table 5.3: Performance comparison on the FCVID. Our TDN substantially improves on the strong TSN baseline, and significantly outperforms previous work.

We compare the class-wise average precision of TSN and TDN. The top 5 improving classes from our TDN are paperCutting (+14%), dumbbellWorkout (+13%), makingIceCream (+10%), pushUps (+10%) and makingHotdog (+10%). We can clearly observe that all of them are actions instead of scenes or objects, showing that our model improves action recognition by modeling the temporal relations in videos.

(a) Input image 1 (b) Input image 2 (c) CAM of image subnetwork on image 1 (d) CAM of image subnetwork on image 2 (e) CAM of difference subnetwork (overlaid on the average of two input images)

Figure 5.3: Class Activation Mapping (CAM) for the image subnetwork and difference subnetwork. The action classes from top to bottom are: turn, climbing-chairs, hit, lifting-benchpress and kiss. Our difference subnetwork can capture the motion regions effectively while the image subnetwork focuses on the apppearance and background context. Note the camera motion and large movement between frames may make optical flow between two frames ineffective, but our TDN sucessfully learns from the large difference on multiple CNN layers.

5.4.4 Influence of Multiple Layers

One natural question to the TDN architecture is whether the multiple layers of feature differences are really helpful in action recognition in addition to image difference. To answer this question, we train TDNs with different settings on the FCVID dataset. We train multiple models by incrementally adding layers of feature differences into the network. As shown in Table 5.4, incorporating multiple layers of feature differences gradually improves the performance. This shows that our network benefits from the differences in higher level features in addition to image difference, and incorporating motions in multiple layers are indeed important to achieve good classification performance.

TDN Layers	mAP (%)
`input`	82.7
`input` – `conv1`	82.9
`input` – `res2`	83.3
`input` – `res3`	**83.8**
`input` – `res4`	83.7

Table 5.4: Effects of incorporating multiple layers of motion. The "`input`" row represents the network only taking difference of the RGB frames, and "`input` – `conv1`" represents the network taking difference of RGB frames and `conv1` outputs into the network and so on. Adding more layers improves recognition performance.

5.4.5 Visualization

We visualize the network outputs to understand what is learned in the Temporal Difference Network. As our base network is a residual network which includes

a global average pooling layer before the final linear classifier, we can compute the Class Activation Mapping (CAM) [111] of the image subnetwork and difference subnetwork respectively, by removing the average pooling layer and applying the linear classifier in all spatial locations. We then use bilinear upsampling to enlarge the heatmaps back to the input size 224×224 and overlaid with the input images.

The visualizations are shown in Figure 5.3. Although the output heatmaps only have 7×7 resolution restricted by the size of the last convolutional layer outputs, we can clearly see what is salient to the network with respect to the action classes. The image subnetwork focuses more on the appearance and the background context, and the difference subnetwork focuses on the motions and actions. This shows that the image subnetwork and the difference subnetwork learn complementary features that help action classification when combined.

5.5 Conclusion

We present a novel network architecture - Temporal Difference Network - for learning temporal relations from videos for action recognition. Instead of learning temporal relation at a fixed level, we capture the Eulerian motions of image features at multiple levels and combine with a single CNN to jointly model motions in multiple scales. We obtain state-of-the-art performance on three public video action recognition benchmarks, demonstrating the effectiveness of our approach.

Chapter 6: Conclusion

In this dissertation, we study the problem of video understanding with deep networks. We first study the deep networks through the problem of image retrieval. We then introduce several deep neural network architectures to combine image information across a video over long time periods. Next, we present a multitask learning framework ActionFlowNet to train a network directly from raw pixels to learn motion representation for video action recognition. Finally, we propose Temporal Difference Networks (TDN) that model both long term relations and short term motion from videos.

While the performance of video action recognition system has been significantly improved, there are still many remaining challenges for video understanding. Future research directions include better network architecture for video recognition, studies of the effectivness and transferrability of video action data, and explicit higher level reasoning with scenes, poses and objects.

Bibliography

[1] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Deep End2End Voxel2Voxel Prediction. In *CVPRW DeepVision Workshop*, 2016.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[3] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions ˜ transformations. In *CVPR*, 2016.

[4] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2014.

[5] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[6] Ivan Laptev, Marcin Marszaek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, pages 1–8, Anchorage, Alaska, USA, 2008.

[7] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proc. CVPR*, pages 3169–3176, Washington, DC, USA, 2011.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, Lake Tahoe, Nevada, USA, 2012.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

[11] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV*, 2014.

[12] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[13] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, pages 512–519, 2014.

[14] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724, 2014.

[15] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, pages 392–407. 2014.

[16] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Visual instance retrieval with deep convolutional networks. *CoRR*, abs/1412.6574, 2014.

[17] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[18] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, pages 2911–2918, 2012.

[19] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, 2012.

[20] Hervé Jégou and Ondrej Chum. Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. In *ECCV*, pages 774–787, 2012.

[21] Relja Arandjelovic and Andrew Zisserman. All about VLAD. In *CVPR*, pages 1578–1585, 2013.

[22] Florent Perronnin, Yan Liu, Jorge Sánchez, and Herve Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, pages 3384–3391, 2010.

[23] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *CVPR*, pages 745–752, 2011.

[24] Behjat Siddiquie, Rogério Schmidt Feris, and Larry S. Davis. Image ranking and retrieval based on multi-attribute queries. In *CVPR*, pages 801–808, 2011.

[25] Mohammad Rastegari, Ali Diba, Devi Parikh, and Ali Farhadi. Multi-attribute queries: To merge or not to merge? In *CVPR*, pages 3310–3317, 2013.

[26] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image search with relative attribute feedback. In *CVPR*, pages 2973–2980, 2012.

[27] Devi Parikh and Kristen Grauman. Implied feedback: Learning nuances of user behavior in image search. In *ICCV*, pages 745–752, 2013.

[28] Adriana Kovashka and Kristen Grauman. Attribute pivots for guiding relevance feedback in image search. In *ICCV*, pages 297–304, 2013.

[29] Adriana Kovashka and Kristen Grauman. Attribute adaptation for personalized image search. In *ICCV*, pages 3432–3439, 2013.

[30] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[31] E. J. Crowley and A. Zisserman. In search of art. In *Workshop on Computer Vision for Art Analysis, ECCV*, 2014.

[32] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, pages 584–599. 2014.

[33] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *ACM Multimedia*, pages 157–166, 2014.

[34] Zhongwen Xu, Yi Yang, and Alexander G. Hauptmann. A discriminative CNN video representation for event detection. *arXiv preprint arXiv:1411.4006*, 2014.

[35] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008.

[37] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, pages 1–8, 2007.

[38] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[39] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *NIPS*, pages 1601–1609, 2014.

[40] Jonathan Delhumeau, Philippe Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the VLAD image representation. In *ACM Multimedia*, pages 653–656, 2013.

[41] Wan-Lei Zhao, Guillaume Gravier, and Hervé Jégou. Oriented pooling for dense and non-dense rotation-invariant features. In *BMVC*, 2013.

[42] Hervé Jégou and Andrew Zisserman. Triangulation embedding and democratic aggregation for image search. In *CVPR*, pages 3310–3317, 2014.

[43] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential Deep Learning for Human Action Recognition. In *2nd International Workshop on Human Behavior Understanding (HBU)*, pages 29–39, November 2011.

[44] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE Trans. PAMI*, 35(1):221–231, January 2013.

[45] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks. In *CVPR*, 2014.

[46] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. NIPS*, pages 568–576, Montreal, Canada, 2014.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computing*, 9(8):1735–1780, November 1997.

[48] Heng Wang, Muhammad Muneeb Ullah, Alexander Klser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proc. BMVC*, pages 1–11, 2009.

[49] Mihir Jain, Hervé Jégou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *Proc. CVPR*, pages 2555–2562, Portland, Oregon, USA, 2013.

[50] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011.

[51] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *Proc. ICCV*, pages 3551–3558, Sydney, Australia, 2013.

[52] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, 5(2):157–166, 1994.

[53] Alex Graves, Marcus Liwicki, S. Fernandez, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. PAMI*, 31(5):855–868, 2009.

[54] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proc. NIPS*, pages 545–552, Vancouver, B.C., Canada, 2008.

[55] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proc. ICML*, pages 1764–1772, Beijing, China, 2014.

[56] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. Phoneme recognition in TIMIT with BLSTM-CTC. *CoRR*, abs/0804.3269, 2008.

[57] Martin Wllmer, Moritz Kaiser, Florian Eyben, Bjrn Schuller, and Gerhard Rigoll. LSTM-modeling of continuous emotions in an audiovisual affect recognition framework. *Image Vision Computing*, 31(2):153–163, 2013.

[58] Stephan Reiter, Bjrn Schuller, and Gerhard Rigoll. A combined LSTM-RNN - HMM - approach for meeting event segmentation and recognition. In *Proc. ICASSP*, pages 393–396, Toulouse, France, 2006.

[59] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *CoRR*, abs/1410.4615, 2014.

[60] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. In *Proc. ICANN*, pages 154–159, Thessaloniki, Greece, 2010.

[61] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *Proc. ICML*, pages 111–118, Haifa, Israel, 2010.

[62] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *JMLR*, 3:115–143, 2002.

[63] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM Conference on Pattern Recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag.

[64] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. In *CRCV-TR-12-01*, 2012.

[65] Ishan Misra, Lawrence C Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[66] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016.

[67] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017.

[68] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[69] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*, 2015.

[70] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Phillip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *ICCV*, 2015.

[71] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[72] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*, pages 20–36. Springer, 2016.

[73] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

[74] Joe Yue-Hei Ng and Larry S. Davis. Temporal difference networks for video action recognition. In *WACV*, 2018.

[75] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[76] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, pages 835–851. Springer, 2016.

[77] Ali Diba, Ali Mohammad Pazandeh, and Luc Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. *arXiv preprint arXiv:1608.08851*, 2016.

[78] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch Networks for Multi-task Learning. In *CVPR*, 2016.

[79] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-Supervised Learning with Ladder Networks. In *NIPS*, 2015.

[80] E. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*, 2015.

[81] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[82] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[83] Joe Y.-H. Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond Short Snippets: Deep Networks for Video Classification. In *CVPR*, 2015.

[84] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017.

[85] Mihir Jain, Jan van Gemert, and Cees Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR*, 2015.

[86] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[87] Zuxuan Wu, Ting Yao, Yanwei Fu, and Yu-Gang Jiang. Deep learning for video classification and captioning. *arXiv preprint arXiv:1609.06782*, 2016.

[88] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[89] Behrooz Mahasseni and Sinisa Todorovic. Regularizing long short term memory with 3d human-skeleton sequences for action recognition. In *CVPR*, 2016.

[90] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[91] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *ICLR*, 2016.

[92] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *IEEE TPAMI*, 2016.

[93] Basura Fernando and Stephen Gould. Learning end-to-end video classification with rank-pooling. In *ICML*, 2016.

[94] Basura Fernando, Peter Anderson, Marcus Hutter, and Stephen Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016.

[95] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[96] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

[97] Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014.

[98] Chen Sun and Ram Nevatia. Active: Activity concept transitions in video event classification. In *CVPR*, 2013.

[99] Hamid Izadinia and Mubarak Shah. Recognizing complex events using large margin joint low-level event model. In *ECCV*, 2012.

[100] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[101] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[102] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

[103] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017.

[104] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. *ACM Transactions on Graphics (TOG)*, 31(4):65, 2012.

[105] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *arXiv preprint arXiv:1502.07209*, 2015.

[106] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[107] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *Pattern Recognition*. 2007.

[108] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[109] Wangjiang Zhu, Jie Hu, Gang Sun, Xudong Cao, and Yu Qiao. A key volume mining deep framework for action recognition. In *CVPR*, 2016.

[110] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016.

[111] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.