Beyond Short Snippets: Deep Networks for Video Classification

Joe Yue-Hei Ng[1]
yhng@umiacs.umd.edu

Matthew Hausknecht[2]
mhauskn@cs.utexas.edu

Sudheendra Vijayanarasimhan[3]
svnaras@google.com

Oriol Vinyals[3]
vinyals@google.com

Rajat Monga[3]
rajatmonga@google.com

George Toderici[3]
gtoderici@google.com

[1]University of Maryland, College Park   [2]University of Texas at Austin   [3]Google, Inc.

Abstract

Convolutional neural networks (CNNs) have been extensively applied for image recognition problems giving state-of-the-art results on recognition, detection, segmentation and retrieval. In this work 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. Our best networks exhibit significant performance improvements over previously published results on the Sports 1 million dataset (73.1% vs. 60.9%) and the UCF-101 datasets with (88.6% vs. 88.0%) and without additional optical flow information (82.6% vs. 73.0%).

1. Introduction

Convolutional Neural Networks have proven highly successful at static image recognition problems such as the MNIST, CIFAR, and ImageNet Large-Scale Visual Recognition Challenge [15, 21, 28]. 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 [2, 13, 14, 19].

Video analysis provides more information to the recognition task by adding a temporal component through which
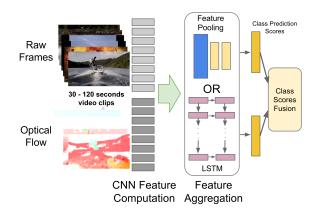

Figure 1: Overview of our approach.

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 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) [11] 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 [14] 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 [14]. 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 [19] 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 constant 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 4.1) and UCF-101 (Section 4.2).

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 [17], in a dense grid [24] or around dense point trajectories [12, 16, 22, 23] 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) [17] or Fisher vector based encodings [23].

However, no previous attempts at CNN-based video recognition use both motion information and a global description of the video: Several approaches [2, 13, 14] 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*. [14] 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*. [19] 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 [3]. In contrast, the Long Short Term Memory (LSTM) [11] 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 [8, 10], speech recognition [9, 7], phoneme detection [5], emotion detection [25], segmentation of meetings and events [18], and evaluating programs [27]. While LSTMs have been applied to action classification in [1], 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*. [1] 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. Approach

Two CNN architectures are used to process individual video frames: AlexNet and GoogLeNet. AlexNet, is a Krizhevsky-style CNN [15] 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 [21], uses a network-in-network approach, stacking Inception modules to form a network 22 layers deep that is substantially different from previous CNNs [15, 28]. 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.
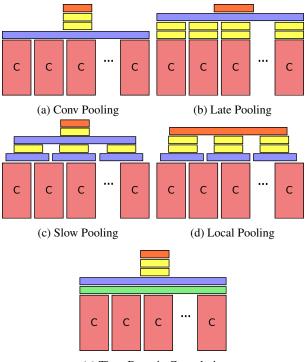
3.1. Feature Pooling Architectures

Temporal feature pooling has been extensively used for video classification [17, 24, 12], 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 [4]. 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 paper 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 2:

(a) Conv Pooling   (b) Late Pooling

(c) Slow Pooling   (d) Local Pooling

(e) Time-Domain Convolution

Figure 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.

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 fea-

tures 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 [21], 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.

3.2. 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:

$$h_t = \mathcal{H}(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \qquad (1)$$
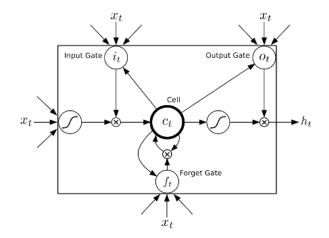$$y_t = W_{ho}h_t + b_o \qquad (2)$$

Figure 3: Each LSTM cell remembers a single floating point value c_t (Eq. 5). This value may be diminished or erased through a multiplicative interaction with the forget gate f_t (Eq. 4) or additively modified by the current input x_t multiplied by the activation of the input gate i_t (Eq. 3). The output gate o_t controls the emission of h_t, the stored memory c_t transformed by the hyperbolic tangent nonlinearity (Eq. 6,7). Image duplicated with permission from Alex Graves.

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 [6] uses memory cells (Figure 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:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \qquad (3)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \qquad (4)$$
$$c_t = f_tc_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \qquad (5)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \qquad (6)$$
$$h_t = o_t \tanh(c_t) \qquad (7)$$

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 [9] (Figure 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

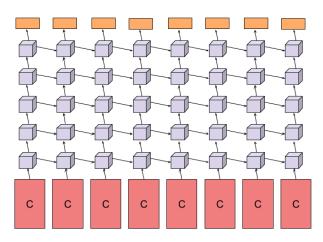with 512 memory cells. Following the LSTM layers, a Softmax classifier makes a prediction at every frame.

Figure 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.

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.

3.4. 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 $1 fps$, 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 [19].

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 [23].

Optical flow is computed from two adjacent frames sampled at $15 fps$ using the approach of [26]. 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.

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.

4.1. Sports-1M dataset

The Sports-1M dataset [14] 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 $1 fps$ 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 paper, 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

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 1: Conv-Pooling outperforms all other feature-pooling architectures (Figure 2) on Sports-1M using a 120-frame AlexNet model.

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 [14], 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 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 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) were trained from scratch on single-frames selected at random from Sports-1M videos. Results (Table 2) show that both CNNs outperform Karpathy *et al.*'s prior single-frame models [14] 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

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 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.

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: 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 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.

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 2 we show that a 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 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 [14] 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 its content.

Optical Flow: Table 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 5 reports the results of the best model from [14] 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.

4.2. UCF-101 Dataset

The UCF-101 [20] 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 $6 fps$ while still capturing context from the full video. We compare 30-frame models trained at three different frame-rates: $30 fps$ (1 second of video) and $6 fps$ (5 seconds). Table 6 shows that lowering the frame rate from $30 fps$ to $6 fps$ 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 $1 fps$. 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.

Category	Method	Frames	Clip Hit@1	Hit@1	Hit@5
Prior	Single Frame	1	41.1	59.3	77.7
Results [14]	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 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.

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

Table 6: Lower frame rates produce higher UCF-101 accuracy for 30-frame Conv-Pooling models.

Overall Performance: Our models achieve state-of-the-art performance on UCF-101 (Table 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.

Our 120 frames model improves upon previous work [19] (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 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.

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

Method	3-fold Accuracy (%)
Improved Dense Trajectories (IDTF)s [23]	87.9
Slow Fusion CNN [14]	65.4
Single Frame CNN Model (Images) [19]	73.0
Single Frame CNN Model (Optical Flow) [19]	73.9
Two-Stream CNN (Optical Flow + Image Frames, Averaging) [19]	86.9
Two-Stream CNN (Optical Flow + Image Frames, SVM Fusion) [19]	88.0
Our Single Frame Model	73.3
Conv Pooling of Image Frames + Optical Flow (30 Frames)	87.6
Conv Pooling of Image Frames + Optical Flow (120 Frames)	**88.2**
LSTM with 30 Frame Unroll (Optical Flow + Image Frames)	**88.6**

Table 7: UCF-101 results. The bold-face numbers represent results that are higher than previously reported results.

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.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. In *Proc. ICANN*, pages 154–159, Thessaloniki, Greece, 2010. 2

[2] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential Deep Learning for Human Action Recognition. In *2nd International Workshop on Human Behavior Understanding (HBU)*, pages 29–39, Nov. 2011. 1, 2

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

[4] Y.-L. Boureau, J. Ponce, and Y. Lecun. A theoretical analysis of feature pooling in visual recognition. In *Proc. ICML*, pages 111–118, Haifa, Israel, 2010. 3

[5] S. Fernández, A. Graves, and J. Schmidhuber. Phoneme recognition in TIMIT with BLSTM-CTC. *CoRR*, abs/0804.3269, 2008. 2

[6] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *JMLR*, 3:115–143, 2002. 4

[7] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proc. ICML*, pages 1764–1772, Beijing, China, 2014. 2

[8] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. PAMI*, 31(5):855–868, 2009. 2

[9] A. Graves, A.-R. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. 2, 4

[10] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proc. NIPS*, pages 545–552, Vancouver, B.C., Canada, 2008. 2

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computing*, 9(8):1735–1780, Nov. 1997. 2

[12] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *Proc. CVPR*, pages 2555–2562, Portland, Oregon, USA, 2013. 2, 3

[13] S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *IEEE Trans. PAMI*, 35(1):221–231, Jan. 2013. 1, 2

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, pages 1725–1732, Columbus, Ohio, USA, 2014. 1, 2, 6, 7, 8

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, Lake Tahoe, Nevada, USA, 2012. 1, 2, 3

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proc. ICCV*, pages 2556–2563, Barcelona, Spain, 2011. 2

[17] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, pages 1–8, Anchorage, Alaska, USA, 2008. 2, 3

[18] S. Reiter, B. Schuller, and G. Rigoll. A combined LSTM-RNN - HMM - approach for meeting event segmentation and recognition. In *Proc. ICASSP*, pages 393–396, Toulouse, France, 2006. 2

[19] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. NIPS*, pages 568–576, Montreal, Canada, 2014. 1, 2, 5, 8

[20] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 7

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1, 3, 4

[22] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Proc. CVPR*, pages 3169–3176, Washington, DC, USA, 2011. 2

[23] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *Proc. ICCV*, pages 3551–3558, Sydney, Australia, 2013. 2, 5, 8

[24] H. Wang, M. M. Ullah, A. Klser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proc. BMVC*, pages 1–11, 2009. 2, 3

[25] M. Wllmer, M. Kaiser, F. Eyben, B. Schuller, and G. Rigoll. LSTM-modeling of continuous emotions in an audiovisual affect recognition framework. *Image Vision Computing*, 31(2):153–163, 2013. 2

[26] 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. 5

[27] W. Zaremba and I. Sutskever. Learning to execute. *CoRR*, abs/1410.4615, 2014. 2

[28] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, pages 818–833, Zurich, Switzerland, 2014. 1, 3