International Journal of Computer Vision manuscript No.

Discriminatively Learned Hierarchical Rank Pooling Networks

Basura Fernando · **Stephen Gould**

Received: date / Accepted: date

Abstract Rank pooling is a temporal encoding method that summarizes the dynamics of a video sequence to a single vector which has shown good results in human action recognition in prior work. In this work, we present novel temporal encoding methods for action and activity classification by extending the unsupervised rank pooling temporal encoding method in two ways.

First, we present *discriminative rank pooling* in which the shared weights of our video representation and the parameters of the action classifiers are estimated jointly for a given training dataset of labelled vector sequences using a bilevel optimization formulation of the learning problem. When the frame level features vectors are obtained from a convolutional neural network (CNN), we rank pool the network activations and jointly estimate all parameters of the model, including CNN filters and fully-connected weights, in an end-to-end manner which we coined as *end-to-end trainable rank pooled CNN*. Importantly, this model can make use of any existing convolutional neural network architecture (e.g., AlexNet or VGG) without modification or introduction of additional parameters.

Then, we extend rank pooling to a high capacity video representation, called *hierarchical rank pooling*. Hierarchical rank pooling consists of a network of rank pooling functions, which encode temporal semantics over arbitrary long video clips based on rich frame level features. By stacking non-linear feature functions and temporal sub-sequence encoders one on top of the other, we build a high capacity encoding network of the dynamic behaviour of the video. The resulting video representation is a fixed-length feature vector describing the entire video clip that can be used as input to standard machine learning classifiers.

We demonstrate our approach on the task of action and activity recognition. We present a detailed analysis of our approach against competing methods and explore variants such as hierarchy depth and choice of non-linear feature function. Obtained results are comparable to state-of-the-art methods on three important activity recognition benchmarks with classification performance of 76.7% mAP on Hollywood2, 69.4% on HMDB51, and 93.6% on UCF101.

Keywords rank pooling · action recognition · activity recognition · convolutional neural networks

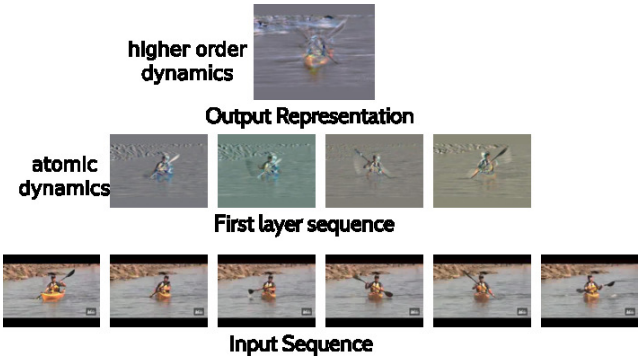


Fig. 1: Illustration of hierarchical rank pooling for encoding the temporal dynamics of a video sequence.

1 Introduction

Representation learning from sequence data has many applications including action and activity recognition from videos [50], gesture recognition [6], music classification from audio clips [46], and gene regulatory network analysis from gene expressions [55]. In this paper we focus on activity and

ACRV, Research School of Engineering
The Australian National University
Canberra, Australia
Tel.: +61-2612-53973
E-mail: basura.fernando@anu.edu.au

action recognition in videos, which is important for many real life applications including human computer interaction, sports analytic, and elderly monitoring and healthcare. Neural network-based supervised learning of representations from sequence data has many advantages compared to hand-crafted feature engineering. However, capturing the discriminative behaviour of sequence data is a very challenging problem; especially when neural network-based supervised learning is used, which can overfit to irrelevant temporal signals. In video sequence classification, and especially in action recognition, a key challenge is to obtain discriminative video representations that generalize beyond the training data. Moreover, a good video representation should be invariant to the speed of the human actions and should be able to capture long term time evolution information, i.e., the temporal dynamics. In action recognition a key challenge is to extract and represent high-level motion patterns, dynamics, and evolution of appearance of videos. One can argue that end-to-end learning of video representations are the key to successful human action recognition. However, it is extremely hard problem due to massive amount of video data that is required to learn such end-to-end video representations. A further challenge is to encode dynamics efficiently and effectively from variable length sequences. This calls for novel spatio-temporal neural network architectures.

Recent success in action and activity recognition has been achieved by modelling evolving temporal dynamics in video sequences [4,20,18,35,61,71]. Some methods use linear ranking machines to capture first order dynamics [20,27]. Other methods encode temporal information using RNN-LSTMs on video sequences [61,71,72], but at the cost of many more model parameters. To further advance activity recognition it is beneficial to exploit temporal information at multiple levels of granularity in a hierarchical manner and thereby capture more complex dynamics of the input sequences [15,41, 59]. As frame based features improve, e.g., from a convolutional neural network (CNN), it is important to exploit information not only in the spatial domain but also in the temporal domain. Several recent methods have obtained significant improvements in image categorisation and object detection using very deep CNN architectures [57]. Motivated by these deep hierarchies [15,41,59,57], we argue that learning a temporal encoding at a single level is not sufficient to interpret and understand video sequences, and that a temporal hierarchy is needed.

In addition, we argue that end-to-end learning of video representations are necessary for reliable human action recognition. In recent years CNNs have become very popular for automatically learning representations from large collections of static images. Many tasks in computer vision, such as image classification, image segmentation and object detection, have benefited from such automatic representation learning [38,23]. However, it is unclear how one may extend these highly successful CNNs to sequence data; especially, when the intended task requires capturing dynamics of video sequences (e.g., action and activity recognition). Indeed, capturing the discriminative dynamics of a video sequence remains an open problem. Some authors have proposed to use recurrent neural networks (RNNs) [15] or extensions, such as long short term memory (LSTM) networks [61], to classify video sequences. However, CNN-RNN/LSTM models introduce a large number of additional parameters to capture sequence information. Consequently, these methods need much more training data. For sequence data such as videos, obtaining labelled training data is significantly more costly than obtaining labels for static images. This is reflected in the size of datasets used in action and activity recognition research today. Even though there are datasets that consist of millions of labelled images (e.g., ImageNet [11]), the largest fully labelled action recognition dataset, UCF101, consists of barely more than 13,000 videos [60]. Some notable efforts to create large action recognition datasets include the Sports-1M [35], the YouTube-8M [1] and the ActivityNet dataset [58]. The limitation of Sports-1M and YouTube-8M is that they are constructed from weakly labelled human annotations and sometimes annotations are very noisy. Furthermore, ActivityNet only consist of 20,000 high quality annotated videos, which is insufficient for learning good video representations. Despite recent efforts in building good action recognition datasets [36], it is highly desirable, therefore, to develop frameworks that can learn discriminative dynamics from video data without the cost of additional training data or model complexity.

Perhaps the most straightforward CNN-based method for encoding video sequence data is to apply temporal max pooling or temporal average pooling over the video frames. However, these methods do not capture any valuable time varying information of the video sequences [35]. In fact, an arbitrary reshuffling of the frames would produce an identical video representation under these pooling schemes. Rank-pooling [20,18], on the other hand, attempts to encode time varying information by learning a linear ranking machine, one for each video, to produce a chronological ordering of the video's frames based on their appearance (i.e., the hand-crafted or CNN features). The parameters of the ranking machine (i.e., fit linear model) are then used as the video representation. However, unlike max and average pooling, it was previously unclear how the CNN parameters can be fine-tuned to give a more discriminative representation when rank-pooling is used since there is no closed-form formula for the rank-pooling operation and the derivative of its input arguments with respect to the rank-pool output not obvious.

The original rank pooling method of Fernando et al. [20, 18] obtained good activity recognition performance using hand-crafted features. Given a sequence of video frames, the rank pooling method returns a vector of parameters encod-

ing the dynamics of that sequence. The vector of parameters is derived from the solution of a linear ranking SVM optimization problem applied to the entire video sequence, i.e., at a single level. We extend that work in two important directions that facilitates the use of richer CNN-based features to describe the input frames and allows the processing of more complex video sequences.

First, we show how to learn discriminative dynamics of video sequences or vector sequences using rank pooling-based temporal pooling. We show how the parameters of the activity classifier, shared parameters of video representations, and the CNN features themselves can all be learned jointly using a principled optimization framework. A key technical challenge, however, is that the optimization problem contains rank pooling as a subproblem—itself a non-trivial optimization problem. This leads to a large-scale bilevel optimization problem [2] with convex inner-problem, which we propose to solve by stochastic gradient descent. The result is a higher capacity model than Fernando et al. [20, 18], which is tuned to produce features that are discriminative for the task at-hand. Concisely, we learn *discriminative dynamics* during learning by propagating back the errors from the final classification layer to learn both video representation and a good classifier.

Second, we propose a hierarchical rank-pooling scheme that encodes a video sequence at multiple levels. The original video sequence is divided into multiple overlapping video segments. At the lowest level, we encode each video segment using rank pooling to produce a sequence of descriptors, one for each segment, which captures the dynamics of the small video segments (see Figure 1). We then take the resulting sequence, divide that into multiple subsequences, and apply rank pooling to each of these next-level subsequences. By recursively applying rank pooling on the obtained segment descriptors from the previous layer, we capture higher-order, non-linear, and more complex dynamics as we move up the levels of the hierarchy. The final representation of the video is obtained by encoding the top-level dynamic sequence using yet one more rank pooling. This strategy allows us to encode more complicated activities thanks to the higher capacity of the model. In summary, our proposed hierarchical rank pooling model consists of a feed forward network starting with a frame-based CNN and followed by a series of point-wise non-linear operations and rank pooling operations over subsequences as illustrated in Figure 3.

Our main contributions are then: (1) a novel discriminative dynamics learning framework in which we learn discriminative frame-based CNN features for the task at-hand in an end-to-end manner or joint learning of parameters of video representation using rank pooled discriminative video representation, and the classifier parameters, (2) a novel temporal encoding method called *hierarchical rank pooling*.

Our proposed method is useful for encoding dynamically evolving frame-based CNN features, and we are able to show significant improvements over other effective temporal encoding methods.

This paper is an extension of our two recent conference papers [19, 21]. In this journal version we provide a broad overview of the action recognition progress and extend the related work section. Here we unify the learning of discriminative rank pooling and full end-to-end parameter learning using the same bilevel optimization framework. Some additional experiments and analysis are also included. The rest of the paper is organised as follows. Related work is discussed in Section 2 followed by a brief background to rank pooling and some preliminaries in Section 3. We present our discriminative networks in Section 4 and discuss how the resulting representation can be used to classify videos. In Section 5 we show how all the parameters of the discriminative networks can be learned. Then in Section 6, we present our hierarchical rank pooling method. In Section 7, we provide extensive experiments evaluating various aspects of our proposed methods. We conclude the paper in Section 8 with a summary of our main contributions and discussion of future directions.

2 Related Work

In the literature, temporal information of video sequences is encoded using different techniques. Fisher encoding [49] of spatial temporal features is commonly used in prior state-of-the-art works [68] while Jain et al. [30] used VLAD encoding [31] for action recognition over motion descriptors. Temporal max pooling and sum pooling are used with bag-of-features [67] as well as CNN features [52]. Temporal fusion methods such as late fusion or early fusion are used in [35] as a temporal encoding method in the context of CNN architectures. In contrast, we rely on principled rank-pooling to encode temporal information inside CNNs and therefore our method is capable capturing dynamics of video sequences.

Temporal information can also be encoded using 3D convolution operators [32, 64] on fixed size temporal segments. However, as recently demonstrated by Tran et al. [64], such approaches rely on very large video collections to learn meaningful 3D-representations. This is due to the massive amount of parameters used in 3D convolutions. Sun et al. [62] propose to factorize 3D convolutions into spatial 2D convolutions followed by 1D temporal convolutions to ease the training. Moreover, it is not clear how these methods can capture long-term dynamics as 3D convolutions are applied only on short video clips. In contrast, our method does not introduce any additional parameters to existing 2D CNN architectures and capable of learning and capturing long term temporal dynamics.

Recently, recurrent neural networks are gaining popularity for sequence encoding, sequence generation and sequence classification [28,63]. Long-short term memory (LSTM) based approaches may use the hidden state of the encoder as a video representation [61]. Derivative of the state of the RNN is modelled in differential RNN (dRNN) to capture the dynamics of video sequences [66]. A CNN feature based LSTM model for action recognition is presented in [71]. Typically, unsupervised recurrent neural networks are trained in a probabilistic manner to maximize the likelihood of generating the next element of the sequence. By construction our hierarchical rank pooling method is unsupervised and does not rely on very large number of training samples as in recurrent neural networks as our method does not have any parameters to learn. Moreover, our hierarchical rank pooling has a clear objective in capturing dynamics of sequences independent of other sequences and has the capacity to capture complex dynamic signals.

Hierarchical methods have also been used in activity recognition [15,44,59]. A CRF-based hierarchical sequence summarization method is presented in [59]; a hierarchical recurrent neural network for skeleton based action recognition is presented in [15]; and a hierarchical action proposal based mid-level representation is presented in [41]. Recently, VLAD for Deep Dynamics (VLAD3), that accounts for different set of video dynamics is presented in [44]. It also captures short-term dynamics with deep convolutional neural net-work features, relying on linear dynamic systems (LDS) to model medium-range dynamics. To account for long-range inhomogeneous dynamics, a VLAD descriptor is derived for the linear dynamic systems and pooled over the whole video, to arrive at the final VLAD3 representation. In contrast to these methods, our method captures different set of mid-level dynamics as well as dynamics of the entire video using rank pooling principle.

Long term temporal dynamics are also modelled using Beta Process Hidden Markov Models (BP-HMM [22]). Using a beta process prior, these approaches discover a set of latent dynamical behaviours that are shared among multiple time series. The size of the set and the sharing pattern are both inferred from data. Some notable extensions of this approach are used in video analysis and action recognition [54, 29]. Compared to these methods, not only is our framework capable of capturing long term dynamics, it is also capable of capturing dynamics at multiple levels of granularity while being able to learn discriminative dynamics.

Recently, two stream models [56] have gained popularity for action recognition. In these methods, a temporal stream is obtained by using optical flow and spatial stream is obtained by RGB frame data and finally the information is fused [17]. Moreover, trajectory-pooled deep-convolutional descriptor (TDD) also uses two stream network architecture where convolutional feature maps are pooled from the local ConvNet responses over the spatio-temporal tubes centered at the improved trajectories [69]. Our method presented in this paper is complimentary to these two stream architectures. For example, our hierarchical temporal encoding as well as the end-to-end trainable rank pooled CNN can be applied over both spatial and temporal streams.

Rank pooling is also used for temporal encoding at representation level [20,18] or at image level leading to dynamic images [4]. However, we are the first to extend rank pooling to a high capacity temporal encoding. Furthermore, we are the first to demonstrate an end-to-end trainable CNN-based rank pool operator.

Our end-to-end learning algorithm introduces a bilevel optimization method for encoding temporal dynamics of video sequences using convolutional neural networks. Bilevel optimization [2,25] is a large and active research field derived from the study of non-cooperative games with much work focusing on efficient techniques for solving non-smooth problems [47] or studying replacement of the lower level problem with necessary conditions for optimality [10]. It has recently gained interest in the machine learning community in the context of hyperparameter learning [37,12] and in the computer vision community in the context of image denoising [13,40]. Unlike these works we take a gradient-based approach, which the structure of our problem admits. We also address the problem of encoding and classification of temporal sequences, in particular action and activity recognition in video.

Recently, several end-to-end video classification and action recognition method were introduced in the literature [32,35,56]. Compare to other end-to-end video representation learning methods our end-to-end learning has two advantages. First, our temporal pooling is based on rank pooling and hence captures the dynamics of long video sequences. Second, it does not introduce any new parameters to existing image classification architectures such as AlexNet [38]. Ji et al. [32] introduces an end-to-end 3D convolution method that can be only applied for a fixed length videos. Karpathy et al. [35] used several fusion architectures. Very large Sports-1M dataset was used for training which consist of more than million YouTube videos of sports activities. Unfortunately, authors found that operating on individual video frames, performs similarly to the networks, whose input is a stack of frames. This indicates that the architectures proposed in [35] are not able to learnt spatio-temporal features or capture dynamics of videos. Simonyan et al. [56] also propose an end-to-end architecture which only operates at frame-level and finally fuse classifier scores per video.

3 Preliminaries

In this section we introduce the notation used in this paper and provide background on the *rank pooling* method [20,

18], which our work extends. Given a training dataset of video-label pairs $\mathscr{D} = \{(X^{(0)}, y)\}$, the goal in action classification is to learn both parameters of the classifier and video representation such that the error on the training set is minimized. Let $X^{(0)} = \langle x_1^{(0)}, \ldots, x_J^{(0)} \rangle$ be the (ordered) sequence of input RGB video frames.

Feature extraction function $\psi_0()$: Let us define a feature extraction function that takes an input frame and returns a fixed-length feature vector by $\psi_0 : x_t^{(0)} \mapsto x_t^{(1)}$. This operation transforms a sequence of RGB frames $X^{(0)}$ into a sequence of feature vectors denoted by $X^{(1)} = \langle x_1^{(1)}, \ldots, x_J^{(1)} \rangle$. Sometimes, to simplify the notation, we denote a sequence of vectors just by $X = \langle x_1, \ldots, x_J \rangle$. Each of the elements $x_t^{(1)}$ in the sequence $X^{(1)}$ is a vector, i.e., $x_t^{(1)} \in \mathbb{R}^D$. For example, the vector $x_t^{(1)}$ can be the activations from the last fully connected layer of a CNN which is obtained from a RGB video sequence at frame t. This frame-based feature extractor can be parametrized $\psi_0(x_t^{(0)}; \theta)$, where for example, θ are the parameters of a trainable CNN.

Non-linear operator $\psi()$: Let us assume that each video is processed by a feature extractor and then a sequence of vectors is obtained by applying a non-linear transformation. Let us denote a point-wise non-linear operator by $\psi()$ and the non-linear transformation is obtained by $v_t = \psi(x_t)$ or a parametrised non-linear transform is obtained by

$$v_t = \psi(W x_t) \tag{1}$$

where $W \in \mathbb{R}^{D \times D}$. Let us denote the obtained sequence of vectors by $V = \langle v_1, \ldots, v_J \rangle$ where each $v_t \in \mathbb{R}^D$.

Temporal encoding function $\phi()$: A compact video representation is needed to classify a variable-length video sequence into one of the activity classes. As such, a temporal encoding function that operates over a sequence of vectors is defined by $\phi(V)$, which maps the video sequence V (or subsequence thereof) into a fixed-length feature vector, $u \in \mathbb{R}^D$. The goal of temporal encoding is to encapsulate valuable dynamic information in V into a single D-dimensional vector $u = \phi(V)$. In general we can write the temporal encoding function as an optimization problem over a sequence V as

$$\phi(V) \in \operatorname*{argmin}_u f(V, u) \tag{2}$$

where $f(\cdot,\cdot)$ is some measure of how well the sequence is described by each representation u and we seek the best representation. Standard supervised machine learning classification techniques learned on the set of training videos can then be applied to these u vectors.

Typical temporal encoding functions include sufficient statistics calculations or simple pooling operations, such as `max` or average (`avg`). For example, `avg.` pooling can be written as the following optimization problem in Equation 3.

$$\texttt{avg}(V) = \operatorname*{argmin}_u \left\{ \frac{1}{2} \sum_{t=1}^{J} \|u - v_t\|^2 \right\} \tag{3}$$

Rank pooling: The `max` and `avg` pooling operators do not capture the dynamic of a video sequence. More sophisticated, temporal encoders such as the `rank-pool` operator, attempts to capture temporal dynamics [20, 18]. The sequence encoder $\phi(\cdot)$ of rank pooling [20, 18] captures time varying information of the entire sequence using a single linear surrogate function ζ parametrised by $u \in \mathbb{R}^D$. The function ζ ranks frames of the video $V = \langle v_1, \ldots, v_J \rangle$ based on the chronology based on their feature representation. Ideally, the ranking function satisfies the constraint

$$\zeta(v_{t_a}) < \zeta(v_{t_b}) \iff t_a < t_b \tag{4}$$

such that the ranking function should learn to order frames chronologically. In the linear case this boils down to finding a parameter vector u such that $\zeta(v; u) = u^T v$ satisfies Equation 4. In rank pooling [20, 18] this is done by training a linear ranking machine such as RankSVM [34] on V. The learned parameters of RankSVM, i.e., u, are then used as the temporal encoding of the video. Since the ranking function encapsulates ordering information and the parameters lie in the same feature space, the ranking function captures the *evolution of information* in the sequence V [20, 18].

Rank pooling can be viewed as a function that estimates the parameters u in a point-wise manner such that it maps feature vectors v_t to time t. Such a mapping clearly satisfies the order constraints of Equation 4. The idea of rank pooling is to parameterize ζ and then find the parameters $u^\star$ that best represents the sequence V. Due to availability of fast implementations, we use Support Vector Regression (SVR) [45] to solve this problem. Given a sequence of length J, the SVR parameters are given by

$$u^\star \in \operatorname*{argmin}_u \left\{ \frac{1}{2}\|u\|^2 + \frac{C}{2} \sum_{t=1}^{J} \left[|t - u^\top v_t| - \varepsilon \right]_{\geq 0}^2 \right\} \tag{5}$$

where $[\cdot]_{\geq 0} = \max\{\cdot, 0\}$ projects onto the positive reals.

The advantage of stability and robustness in modelling dynamics is discussed in [18]. As the SVR objective has some theoretical guarantees on the generalization and stability [5] the obtained temporal representation $u^\star$ is robust to small perturbed versions of the input. Therefore, the above SVR objective is advantageous for modelling dynamics. We use the parameter $u^\star$, returned by SVR, as the temporal encoding vector of the video sequence.

3.1 overview

One of the limitations of rank pooling method presented in [20,18] is that obtained temporal representation u is not discriminative as the classifier and the underlying frame representation is obtained independently. In this work we extend the work of Fernando et al. [20,18]. First, we show a learning framework for discriminative temporal encoding using rank pooling in section 4. Given a collection of labelled videos, we show how to learn frame representation, temporal representation for the video and the classifier jointly. In this case, the temporal representation is obtained by `rank-pool` operator. We also learn a discriminative rank pooling operator when a set of labelled sequences of vectors are provided as the input. In this case, we learn the classifier parameters and the discriminative temporal representation jointly. Parameter learning of these discriminative models is explained in section 5. Second, we show hierarchical rank-pooling, a new hierarchical temporal encoding scheme which extends the `rank-pool` operator in section 6. To learn discriminative hierarchical representation, one can stack discriminative rank pooling network over the hierarchical rank pooling network. In experiments, we demonstrate how to combine hierarchical rank pooling with discriminative learning framework to obtain good results for action recognition (section 7.2).

4 Discriminative video representations with rank-pooling networks

In this section, we introduce our proposed trainable rank pooling network based video representation framework. We consider two scenarios to learn discriminative video representations using `rank-pool` operator. In both cases, the temporal encoding of frame level feature vectors is obtained with rank pooling.

1. In the first scenario, the input to our algorithm is a set of labelled row RGB videos $\mathscr{D} = \{(X^{(0)}, y)\}$. Then our aim is to learn parametrized feature extractor (a CNN [38] feature extractor which is denote by $v_t = \psi_0(x_t^{(0)}; \theta)$), the temporal video representation (u) and the action classifiers jointly. In this case θ is the set of parameters in a trainable CNN.
2. In the second scenario, input to our algorithm is a set of labelled sequences of vectors obtained from video sequences. We aim to learn a parameterized non-linear operation denoted by Equation (1) and the classifier parameters jointly. The W matrix is shared across all sequences from all classes.

Next, we provide more details about these two models. First, we discuss our end-to-end video representation and classification model in Section 4.1. Then in Section 4.2, we introduce the discriminative `rank-pool` operator that operates over a sequences of vectors.

4.1 End-to-end trainable rank pooled CNN

In the first scenario, the input to our framework is a sequence of raw RGB videos with action category labels $\mathscr{D} = \{(X^{(0)}, y)\}$. We assume that each video frame in the input sequence is encoded by a CNN network [38] $\psi_0(\cdot;\theta)$ which is parameterized by θ and that the resulting sequence of features $V = \langle \ldots, v_t, \ldots \rangle$ is encoded using rank pooling (the temporal encoder ϕ) by solving the objective function in Equation (5). The model we propose can be summarized by the following network equation:

$$X^{(0)} = \left\langle x_t^{(0)} \right\rangle \overset{\psi_0(\cdot;\theta)}{\longmapsto} \langle v_t \rangle \overset{\phi}{\longmapsto} u \overset{h_\beta}{\longmapsto} \hat{y} \tag{6}$$

where the feed-forward pass of the network go from a video sequence $X^{(0)}$ to predicted label $\hat{y}$. The final layer is our prediction function (a soft-max classifier) h_β parameterized by β. Therefore, the probability of a label y given the input sequence $X^{(0)}$ can be written as

$$P(y \mid X^{(0)}; \beta, \theta) = \frac{\exp(\beta_y^\top u)}{\sum_c \exp(\beta_c^\top u)} \tag{7}$$

where we have used u to denote the final video encoding. Importantly, u is a function of both the input video sequence $X^{(0)}$ and the network parameters θ. Here the predictor function $h_\beta(u)$ takes the highest probability (most likely) y over the discrete set of labels and $\beta = \{\beta_y\}$ are the learned parameters of the model.

The detailed network architecture is shown in Figure 2. We use a CNN architecture similar to CaffeNet [33] with the addition of a temporal pooling layer. In our experiments we use the final activation layers of the CNN as the frame level features and then apply the temporal pooling (`rank-pool` operator) as shown in Figure 2. During training, our objective is to learn the parameters β and θ. During inference we fix θ and β to their learned values; θ is used to obtain the frame representation of the video that is used to obtain u via temporal encoding and which is then classified (using parameters θ) into an estimated action class for the video.

4.2 Discriminative rank pooling

In this section, we discuss the second model where the input to the feature extractor is a sequence of vectors instead of sequence of RGB frames. We present a method to learn dynamics of any vector sequence in a discriminative manner using `rank-pool` operator as the temporal encoder. In this instance, the parameterized non-linear operation as in Equation (1) is applied over the feature vectors of the sequence

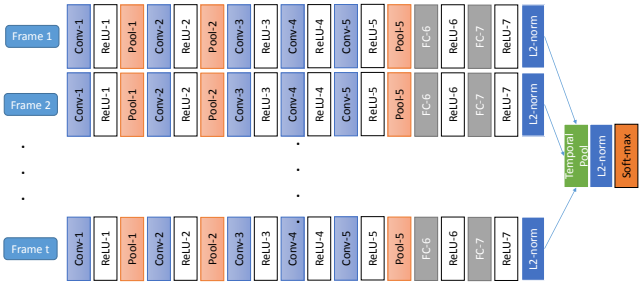


Fig. 2: The CNN network architecture used for learning end-to-end temporal representations of videos. Network takes a sequence of frames from a video as inputs and feed forward till the end of the temporal pooling layer. At the temporal polling layer, the sequence of vectors are encoded by rank-pooling operator to produce fixed length video representation. This fixed length vector is feed to the next layer in the network. Note that this network does not introduce any new parameters to network architectures. During back-propagation, the gradients are feed backwards through the rank-pooling operator to the rest of the CNN network.

$X = \langle x_1, \ldots, x_t, \ldots x_J \rangle$. The function ψ is a non-linear feature function such as ReLU [38]. The discriminative rank pooling network can be summarized as follows:

$$X = \langle x_t \rangle \overset{\psi(\cdot;W)}{\longmapsto} \langle v_t \rangle \overset{\phi}{\longmapsto} u \overset{h_\beta}{\longmapsto} \hat{y} \tag{8}$$

where h_β is the soft-max classifier parameterized by β. Similar to Section 4.1, our aim is to jointly learn the non-linear transformation parameter W of $\psi(\cdot;W)$ along with the classifier parameters denoted by β.

5 Learning the parameters of rank pooling networks

Now we have presented our two video representation models in the previous section, we discuss how to learn the parameters in this section. First, we formulate the overall learning problem in Section 5.1 and then we show how to learn the parameters with stochastic gradient descent in Section 5.2. Then we compute the gradient function of our two models in Section 5.3 and Section 5.4 respectively. Finally, we discuss some optimization difficulties and solutions in Section 5.5.

5.1 Optimization problem

The learning problem can be described as follows. Given a training dataset of video-label pairs $\mathscr{D} = \{(X^{(0)}, y)\}$ (or $\mathscr{D} = \{(X, y)\}$), our goal is to learn both parameters of the classifier and video representation θ (or W) such that the error on the training set is minimized. Let $\Delta(\cdot,\cdot)$ be a loss function. For example, when using the soft-max classifier a typical choice would be the cross-entropy loss

$$\Delta(y, h_\beta(u)) = -\log P(y \mid X^{(0)}) \tag{9}$$

where $P(\cdot \mid \cdot)$ is defined by Equation (7).

We jointly estimate the parameters of the feature extractors (θ or W) and prediction function (β) by minimizing the regularized empirical risk. Formally, our learning problem for end-to-end trainable rank pooled CNN is

$$\begin{array}{ll} \text{minimize}_{\theta,\beta} & \sum_{(X^{(0)},y)\in\mathscr{D}} \Delta\big(y, h_\beta(u_X)\big) + R(\theta,\beta) \\ \text{subject to} & u_X \in \text{argmin}_u f(V,u;\theta) \end{array} \tag{10}$$

where $R(\cdot,\cdot,\cdot)$ is some regularization function, typically the ℓ_2-norm of the parameters, and the function $f(\cdot)$ encapsulates the temporal encoding of the video sequence using rank pooling temporal encoder ϕ by solving (5). The vector u_X then represents the output of the rank pooling operator. It should be noted that the learning problem for discriminative rank pooling of Section 4.2 is similar to the Equation (10).

Equation 10 is an instance of a bilevel optimization problem, which have recently been explored in the context of support vector machine (SVM) hyper-parameter learning [37] but whose history goes back to the 1950s [2]. Here an upper level problem is solved subject to constraints enforced by a lower level problem. A number of solution methods have been proposed for bilevel optimization problems. Given our interest in learning video representations, which is large-scale, gradient-based techniques are most appropriate to learn the parameters.

5.2 Learning with stochastic gradient descent

We are now left with the task of tuning the parameters θ or W to learn a discriminative video representation in order to improve the action recognition performance. One such approach is to learn the classifier parameters and feature encoding parameters jointly via stochastic gradient descent (SGD). However, this requires back propagation of gradients through the network. When the temporal encoding function ϕ can be evaluated in closed-form (e.g., `max` or `avg` pooling) to obtain the temporal encoding vector u_X, we can substitute the constraints in Equation 10 directly into the objective and use (sub-)gradient descent to solve for (locally or globally) optimal parameters. However, when rank pooling is used for temporal encoding the situation is not as simple. Recall that the rank pooling operator is itself an optimization problem, which takes an arbitrary long sequence of feature vectors and returns a fixed-length vector that preserves temporal information. In this instance, the gradient of an argmin function is required. Fortunately, when the lower level objective is twice differentiable we can compute the gradient of the argmin function as other authors have also observed [47, 14, 12]. We repeat the key result here for completeness.

Lemma 1 *[53] Let $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function with first and second derivatives. Let $g(x) = argmin_{y \in \mathbb{R}^n} f(x,y)$. Then*

$$g'(x) = -f_{YY}(x,g(x))^{-1} f_{XY}(x,g(x)).$$

where $f_{YY} \doteq \nabla^2_{yy} f(x,y) \in \mathbb{R}^{n \times n}$ and $f_{XY} \doteq \frac{\partial}{\partial x} \nabla_y f(x,y) \in \mathbb{R}^n$.

Proof We have:

$$f_Y(x,g(x)) \doteq \nabla_Y f(x,y)|_{y=g(x)} = 0 \tag{11}$$

$$\frac{d}{dx} f_Y(x,g(x)) = 0 \tag{12}$$

$$\therefore f_{XY}(x,g(x)) + f_{YY}(x,g(x)) g'(x) = 0 \tag{13}$$

$$\frac{d}{dx} g(x) = -f_{YY}(x,g(x))^{-1} f_{XY}(x,g(x)) \tag{14}$$

□

Interestingly, replacing argmin with argmax in the above lemma yields the same gradient, which follows from the proof that only requires that $g(x)$ be a stationary point. So the result holds for both argmin and argmax optimization problems.

Using Lemma 1 we can compute the gradient of the rank pooling temporal encoding function with respect to a parameterized representation of the feature vectors. We only consider the case of a single scalar parameter θ. The extension to a vector of parameters can be done elementwise.

Corollary 1 *Let $\theta \in \mathbb{R}$ be a parameter and let $\langle v_t \rangle$ be a sequence where the v_t are functions of θ. Define $f(\theta,u)$ to be the objective of the rank pooling optimization problem Equation 5. That is,*

$$f(\theta,u) = \frac{1}{2}\|u\|^2 + \frac{C}{2} \sum_{t=1}^{J} \left[|t - u^\top v_t| - \varepsilon \right]^2_{\geq 0}.$$

And let $u^\star = argmin_u f(\theta,u)$. Then

$$\frac{du^\star}{d\theta} = \left(I + C \sum_{e_t \neq 0} v_t v_t^\top \right)^{-1} \left(C \sum_{e_t \neq 0} e_t \frac{dv_t}{d\theta} - u^{\star\top} \frac{dv_t}{d\theta} v_t \right)$$

where

$$e_t \doteq \begin{cases} u^{\star\top} v_t - t - \varepsilon, & \text{if } u^{\star\top} v_t - t \geq \varepsilon \\ u^{\star\top} v_t - t + \varepsilon, & \text{if } t - u^{\star\top} v_t \geq \varepsilon \\ 0, & \text{otherwise.} \end{cases}$$

Proof Follows from Lemma 1 with

$$f_U(\theta,u^\star) = u^\star - C \sum_{t=1}^{J} e_t v_t \tag{15}$$

$$f_{UU}(\theta,u^\star) = I + C \sum_{e_t \neq 0} v_t v_t^\top \tag{16}$$

$$f_{\theta U}(\theta,u^\star) = -C \sum_{e_t \neq 0} e_t \frac{dv_t}{d\theta} + u^{\star\top} \frac{dv_t}{d\theta} v_t \tag{17}$$

□

In the subsections below we discuss the specifics of learning the parameters of our two parametric discriminative models (θ and W).

5.3 Learning the parameter of end-to-end trainable rank pooled CNN

Now we present how to learn the parameters of the CNN (θ) and the classifier parameters. Consider again the learning problem defined in Equation 10. The derivative with respect to β, which only appears in the upper-level problem, is straightforward and well known. Using the result of Corollary 1, we compute $\frac{du^{(i)}}{d\theta}$ for each training example and hence the gradient of the objective via the chain rule.We then use stochastic gradient descent (SGD) to learn all parameters jointly.

Consider a single scalar weight update in the CNN. Then, again using Lemma 1 we have

$$\frac{du^{(i)}}{d\theta} = \left(I + C \sum_{e_t \neq 0} v_t v_t^\top \right)^{-1} \left(C \sum_{e_t \neq 0} e_t \psi_0'(x_t;\theta) - u^\top \psi_0'(x_t;\theta) v_t \right) \tag{18}$$

Here $\psi_0'(x_t;\theta)$ is the derivative of the element feature function. In the context of CNN-based features for encoding video frames the derivative can be computed by back-propagation through the network. Note that the `rank-pool` objective function is convex and allows us to solve it efficiently. However, it does include a set of non-differentiable points but we did not find this to cause any practical problems during optimization.

5.4 Learning the parameter W of discriminative rank pooling

Recall, in discriminative rank pooling network model, the sequence of vectors X is processed by optimizing Equation 5 to get u, where $v_t = \psi(W x_t)$. Objective is to learn the classifier parameters and the parameter W jointly. The derivative with respect to classifier parameter β, which only appears in the upper-level problem, is straightforward and well known. However, the partial derivative w.r.t. W is more challenging since u is a complicated function of W defined by Equation 5, which involves solving an argmin optimization problem as before. Thus we have to differentiate *through* the argmin function of the rank pooling problem using Lemma 1.

Recall, we have $v_t = \psi(Wx_t)$ where $\psi(\cdot)$ acts element-wise. From Lemma 1 we have for parameter W_{ij}

$$\frac{\partial u}{\partial W_{ij}} = \left(I + C\sum_{e_t \neq 0} v_t v_t^\top\right)^{-1} \left(C\sum_{e_t \neq 0} e_t \frac{\partial v_t}{\partial W_{ij}} - u^\top \frac{\partial v_t}{\partial W_{ij}} v_t\right) \tag{19}$$

where the k-th element of $\frac{\partial v_t}{\partial W_{ij}}$ is

$$\left(\frac{\partial v_t}{\partial W_{ij}}\right)_k = \begin{cases} \psi'(Wx_t)_{[k]} x_{t[j]}, & \text{if } k = i \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Here the subscript $[i]$ denotes the i-th element of the associated vector.

5.5 Optimization difficulties

One of the main difficulties for learning the parameters of high-dimensional temporal encoding functions (such as those based on CNN features) is that the gradient update in Equation 18 requires the inversion of the Hessian matrix $f_{UU} = (I + C\sum_{e_t \neq 0} v_t v_t^\top)$. One solution is to use a diagonal approximation of the Hessian, which is trivial to invert. For instance let us compute the gradient of discriminative rank pooling model using the diagonal approximation. Considering the derivative of the k-th element of u and approximating the inverse of the first term in Equation 19 by its diagonal, we have

$$\frac{\partial u_k}{\partial W_{ij}} = \left(\frac{1}{1 + C\sum_{e_t \neq 0} v_{t[k]}^2}\right) \times \left(C\sum_{e_t \neq 0} \left(\llbracket i = k \rrbracket e_t - u_i v_{t[k]}\right) \psi_i'(Wx_t) x_{t[j]}\right) \tag{21}$$

Now we have by the chain rule,

$$\frac{\partial}{\partial W_{ij}} \log P(y \mid X) = \left(\nabla_u \log P(y \mid X)\right)^\top \frac{\partial u}{\partial W_{ij}} \tag{22}$$

$$= \left(\beta_y - \sum_c P(c \mid X)\beta_c\right)^\top \frac{\partial u}{\partial W_{ij}} \tag{23}$$

Let $\mathbf{1}$ be the all-ones vector, let $K_t = \frac{\partial v_t}{\partial W}$ where $(K_t)_{[ij]} \doteq \frac{\partial v_{t[i]}}{\partial W_{ij}}$ and let $\hat{\beta}$ denote $\nabla_u \log P(y \mid X)$ scaled by the inverse diagonal hessian, i.e.,

$$\hat{\beta}_{[i]} = \frac{\beta_{y[i]} - \sum_c P(c \mid X)\beta_{c[i]}}{1 + C\sum_{e_t \neq 0} v_{t[i]}^2} \tag{24}$$

Then we can write Equation 23 more compactly as

$$\frac{\partial}{\partial W_{ij}} \log P(y \mid X) = \sum_{e_t \neq 0} \left(\hat{\beta}_{[i]} e_t - u_i \hat{\beta}^\top v_t\right) K_{t[ij]} \tag{25}$$

and the (matrix) gradient with respect to all parameters as

$$\nabla_W \log P(y \mid X) = C\sum_{e_t \neq 0} e_t K_t \odot (\hat{\beta}\mathbf{1}^T) - s_t K_t \odot (u\mathbf{1}^T) \tag{26}$$

where $\odot$ is the Hadamard product and $s_t \doteq \hat{\beta}^\top v_t$.

An alternative, for temporal encoding functions with certain structure like ours, namely where the hessian can be expressed as a diagonal plus the sum of rank-one matrices, the inverse can be computed efficiently using the Sherman-Morrison formula [24],

Lemma 2 *[24] Let $H = I + \sum_{i=1}^n u_i v_i^\top \in \mathbb{R}^{p \times p}$ be invertible. Define $H_0 = I$ and $H_m = H_{m-1} + u_m v_m^\top$ for $m = 1, \ldots, n$. Then*

$$H_m^{-1} = H_{m-1}^{-1} - \frac{H_{m-1}^{-1} u_m v_m^\top H_{m-1}^{-1}}{1 + v_m^\top H_{m-1}^{-1} u_m} \tag{27}$$

whenever $v_m^\top H_{m-1}^{-1} u_m \neq -1$.

Proof Follows from repeated application of the Sherman-Morrison formula.

Since each update in Equation 27 can be performed in $O(p^2)$ the inverse of H can be computed in $O(np^2)$, which is acceptable for many applications. Our experiments include results onbtained by both the diagonal approximation and full inverse.

6 HRP: Hierarchical rank pooling

In this section we present our *hierarchical rank pooling* (HRP) network for video classification. HRP is an unsupervised temporal encoding network which allows us to obtain high capacity temporal encoding.

Even with a rich feature representation of each frame in a video sequence, such as derived from a deep convolutional neural network (CNN) model [38], the shallow rank pooling method [20, 18] may not be able to adequately model the dynamics of complex activities over long sequences. As such, we propose a more powerful yet simple scheme for encoding the dynamics of rich features of complex video sequences. Motivated by the success of hierarchical encoding of deep neural networks [38, 23], we extend rank pooling operator to encode dynamics of a sequence at multiple levels in a hierarchical manner. Moreover, at each stage, we apply a non-linear feature transformation to capture complex dynamical behaviour. We call this method the *hierarchical rank pooling*.

Our main idea is to perform rank pooling on sub-sequences of the video. Each invocation of rank pooling provides a fixed-length feature vector that describes the sub-sequence. Importantly, the feature vectors capture the evolution of frames within each sub-sequence. By construction, the sub-sequences themselves are ordered. As such, we can apply rank pooling

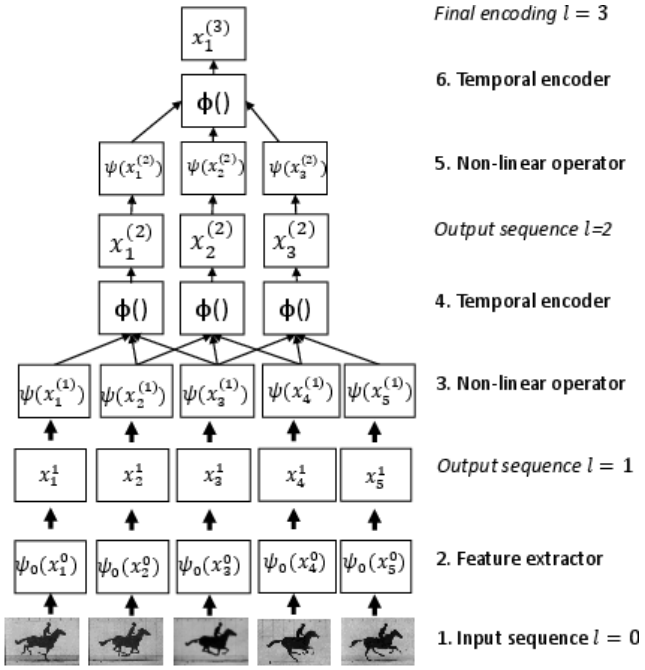


Fig. 3: Two layer network of hierarchical rank pooling with window size three ($\mathscr{M}_\ell = 3$) and stride one ($S_\ell = 1$).

over the generated sequence of feature vectors to obtain a higher-level representation. This process is repeated to obtain dynamic representations at multiple levels for a given video sequence until we obtain a final encoding. To make this hierarchical encoding even more powerful, we apply a point-wise non-linear operation on the input to the rank pooling function. An illustration of the approach is shown in Figure 3.

We assume CNN features are extracted from a fixed CNN. Using a slight change in the notation we denote this by $x_t^{(1)} = \psi_0(x_t^{(0)};\theta)$ where the θ is fixed. In unsupervised hierarchical rank pooling method, we extract feature vectors from each of the frame resulting a sequence of vectors denoted by

$$X^{(1)} = \langle \psi_0(x_1^{(0)}), \ldots, \psi_0(x_J^{(0))}\rangle. \tag{28}$$

We then apply a non-linear transformation ψ to each feature vector to obtain a transformed sequence

$$\widetilde{X}^{(1)} = \langle \psi(x_1^{(1)}), \ldots, \psi(x_J^{(1)})\rangle. \tag{29}$$

Next, applying rank pooling-based temporal encoding ϕ to sub-sequences of $\widetilde{X}^{(1)}$, we obtain a new sequence $X^{(2)}$ of feature vectors describing each video sub-sequence. The process of going from $X^{(1)}$ to $X^{(2)}$ constitutes the first layer of the temporal hierarchy. We now extend the process through additional rank pooling layers, which we formalize by the following definition. Indeed, in our implementation the temporal encoding function ϕ is `rank-pool` operator.

Definition 2 (Rank Pooling Layer) *Let $X^{(\ell)} = \langle x_1^{(\ell)}, \ldots, x_{J_\ell}^{(\ell)}\rangle$ be a sequence of J_ℓ feature vectors. Let M_ℓ be the window size and S_ℓ be a stride. For $t \in \{1, S_\ell + 1, 2S_\ell + 1, \ldots\}$ define transformed sub-sequences $\widetilde{X}_t^{(\ell)} = \langle \psi_\ell(x_t^{(\ell)}), \ldots, \psi_\ell(x_{t+M_\ell-1}^{(\ell)})\rangle$, where $\psi_\ell(\cdot)$ is a point-wise non-linear transformation. Then the output of the ℓ-th* rank pooling layer *is a sequence $X^{(\ell+1)} = \langle \ldots, x_t^{(\ell+1)}, \ldots\rangle$ where $x_t^{(\ell+1)} = \phi(\widetilde{X}_t^{(\ell)})$ is a temporal encoding of the transformed sub-sequence $\widetilde{X}_t^{(\ell)}$ obtained by `rank-pool` operator.*

Each successive layer in our rank pooling hierarchy captures the dynamics of the previous layer. The entire hierarchy can be viewed as applying a stack of non-linear ranking functions on the input video sequence and shares some conceptual similarities with deep neural networks. A simple illustration of a two-layer hierarchical rank pooling network is shown in Figure 3. By varying the stride and window size for each layer, we control the depth of the rank pooling hierarchy. There is no technical reason to limit the number of layers.

To obtain the final vector representation $x^{(L+1)}$, we construct the sequence for the final layer $X^{(L)}$, and encode the whole sequence $X^{(L)}$ with `rank-pool` operator $\phi(\widetilde{X}^{(L)})$. In other words, the last layer in our hierarchy produces a single temporal encoding of last output sequence $\widetilde{X}^{(L)}$ using `rank-pool` operator. We use this final feature vector $x^{(L+1)}$ of the video as its representation, which is then classified by a SVM classifier.

6.1 Capturing non-linear dynamics with non-linear feature transformations

Usually, video sequence data contains complex dynamic information that cannot be captured simply using linear methods such as linear SVR. We believe that the dynamics captured by standard SVR objective reflects only linear dynamics as the SVR function is linear. To obtain non-linear dynamics, one option is to use non-linear feature maps and transform the input features by a non-linear operation. Here we transform the input vectors x_t by a non-linear operation $\psi(x_t)$ before applying SVR based rank pooling (Equation (5)). In the literature, Signed Square Root (SSR) and Chi-square feature mappings are used to obtain good results. Neural networks employ sigmoid and hyperbolic tangent functions to model non-linearity. The advantage of SSR is exploited by Fisher vector-based object recognition as well as in activity recognition [20,68]. When CNN features are used to represent frames, we suggest to consider positive activations separately from the negative activations. Typically the rectification applied in CNN architectures keeps only the positive activations, i.e., $\psi(x) = \max\{\mathbf{0}, x\}$. However, we argue that negative activations may also contain some useful

information and should be considered. Therefore, we propose to use the following non-linear function on the activations of fully connected layers of the CNN architecture. We call this operation the *sign expansion root (**SER**).*

$$\psi(x) = \left(\sqrt{\max\{0,x\}}, \sqrt{\max\{0,-x\}}\right) \tag{30}$$

This operation doubles the size of the features space allowing us to capture important non-linear information, one for positives and the other for negatives. The square-root operation takes care of projecting features to a some unknown non-linear feature space.

So far in this Section 6, we have described how to represent a video by a fixed-length descriptor using hierarchical rank pooling in an unsupervised manner. These descriptors can be used to learn an SVM classifier for activity recognition. The forward pass algorithm for hierarchical rank pooling is shown in Algorithm 1.

Algorithm 1: Hierarchical Rank Pooling Forward Pass.

1: extract CNN features, $X^{(1)} = \langle x_1^{(1)}, x_2^{(1)}, \ldots, x_J^{(1)} \rangle$
2: **for** each rank pooling layer, $\ell = 1 : L-1$ **do**
3: generate transformed sub-sequences $\widetilde{X}_t^{(\ell)}$ as $\langle \psi(x_t^{(\ell)}) \rangle$
4: rank pool each sub-sequence, $x_t^{(\ell+1)} = \phi(\widetilde{X}_t^{(\ell)})$
5: construct $X^{(\ell+1)}$ as $\langle \ldots, x_t^{(\ell+1)}, \ldots \rangle$
6: **end for**
7: get video representation as $x^{(L+1)} = \phi(\widetilde{X}^{(L)})$

7 Experiments

We evaluate proposed methods using four activity and action recognition datasets. We follow exactly the same experimental settings per dataset, using the same training and test splits as described in the literature. Now we give some details of these datasets (also see Figure 4).

HMDB51 dataset [39] is a generic action classification dataset consists of 6,766 video clips divided into 51 action classes. Videos and actions of this dataset are challenging due to various kinds of camera motions, viewpoints, video quality and occlusions. Following the literature, we use a one-vs-all multi-class classification strategy and report the mean classification accuracy over three standard splits provided by Kuehne et al. [39].

Hollywood2 dataset is created by Laptev et al. [43] using 69 different Hollywood movies that include 12 human action classes. It contains 1,707 video clips in which 823 clips are dedicated for training and 884 clips for testing. The performance is measured by average precision. The mean average precision (mAP) is reported over all classes, as in [43].

UCF101 dataset [60] is an action recognition dataset of realistic action videos, collected from YouTube, consists of 101 action categories. It has 13,320 videos from 101 diverse action categories. The videos of this dataset is challenging which contains large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background and illumination conditions. It is one of the most challenging data set to date. It consist of three splits, in which we report the classification performance over all three splits as done in the literature.

UCF-sports dataset [51] consists of a set of short video clips depicting actions collected from various sports. The clips were typically sourced from footage on broadcast television channels such as the BBC and ESPN. The video collection represents a natural pool of actions featured in a wide range of scenes and viewpoints. The dataset includes a total of 150 sequences of resolution 720×480 pixels. Classification performance is measured using mean per-class accuracy. We use provided train-test splits for training and testing.

The rest of the experimental section is organised as follows. First in Section 7.1 we provide a detailed evaluation of hierarchical rank pooling. Then in Section 7.2, we evaluate the impact of discriminative rank pooling. Section 7.3 is dedicated to provide a detailed evaluation of end-to-end trainable rank pooled CNNs. Finally, we compare with some state-of-the-art action recognition methods and position our contributions in Section 7.4. Implementation of our method is publicly available[1].

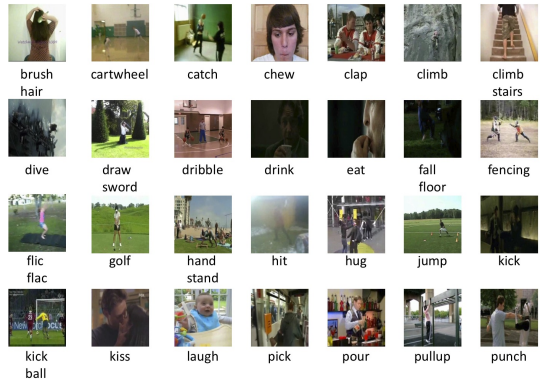
7.1 Evaluating hierarchical rank pooling (HRP)

First, we evaluate activity recognition performance using CNN features and hierarchical rank pooling (HRP) and then provide some detailed analysis.

Experimental details: We utilize pre-trained CNNs without any fine-tuning. Specifically, for each video we extract activations from the VGG-16 [57] network's first fully connected layer (consisting of 4096 values, only from the central patch). We represent each video frame by this 4096 dimensional vector. Note that at this point, we do not use any ReLU [38] non-linearity. As a result the frame representation vector contains both positive and negative components of the activations.

Unless otherwise specified, we use a window size M_ℓ of 20, with a stride S_ℓ of one and a hierarchy depth of two in all our experiments. We use a constant $C = 1$ parameter

[1] `https://bitbucket.org/bfernando/hrp`



(a) HMDB51

(b) Hollywood2

(c) UCF101

(d) UCFSports

Fig. 4: Example frames from (a) HMDB51 (b) Hollywood2 (c) UCF101 and (d) UCF-sports datasets from different action and activity classes.

METHOD	Hollywood2	HMDB51	UCF101
Average pooling	40.9	37.1	69.3
Max pooling	42.4	39.1	72.5
Tempo. pyramid (avg. pool)	46.5	39.1	73.3
Tempo. pyramid (max pool)	48.7	39.8	74.8
LSTM [61]	–	42.8	74.5
Rank pooling	44.2	40.9	72.2
Recursive rank pooling	52.5	45.8	75.6
Hierarchical rank pooling	**56.8**	**47.5**	**78.8**
Improvement	+8.1	+4.7	+4.0

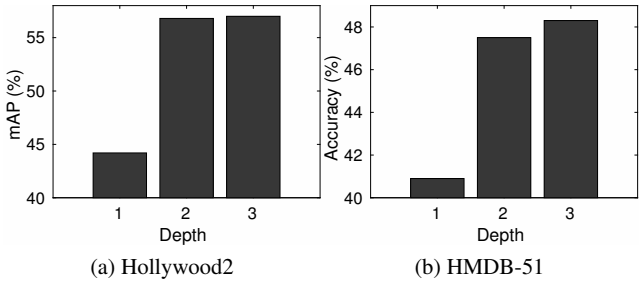
Table 1: Comparing several temporal pooling methods for activity recognition using VGG-16's fc6 features.

for SVR training (Lib-linear [16]) to obtain the `rank-pool`-based temporal encoding as recommended in [18]. We test different non-linear SVM classifiers for the final classification always with $C = 1000$ (LibSVM [8]) as this works well in practice. It should be noted that ideally, the best results can be obtained by cross-validation. However, as commonly done in state-of-the art action recognition methods [68], we use a fixed C for LibSVM training. In the case of multi-class classification, we use a one-against-rest approach and select the class with the highest score. For rank pooling [20,18] and trajectory extraction [68] (in later experiments) we use the publicly available code from the authors.

7.1.1 Comparing temporal pooling methods

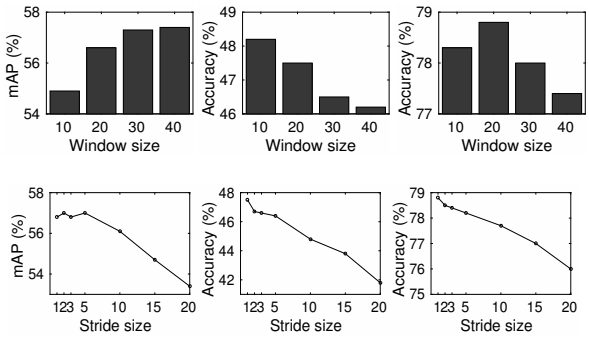
In this section we compare several temporal pooling methods using VGG-16 CNN features. We compare our hierarchical rank pooling with average-pooling, max-pooling, LSTM [61], two level temporal pyramids with mean pooling, two level temporal pyramids with max pooling, and vanilla rank pooling [20,18]. To obtain a representation for average pooling, the average CNN feature activation over all frames of a video was computed. The max-pooled vector is obtain by applying the *max* operation over each dimension of the CNN feature vectors from all frames of a given video. We also compare with a variant of hierarchical rank pooling called *recursive rank pooling*, where the next layer's sequence element at time t denoted by $x_t^{(\ell+1)}$ is obtained by encoding *all frames* of the previous layer sequence up to time t, i.e. $x_t^{(\ell+1)} = \phi(\langle \psi(x_1^{(\ell)}), \ldots, \psi(x_t^{(\ell)}) \rangle)$ for $t = 2, \ldots, J_\ell$.

We compare these base temporal encoding methods on three datasets and report results in Table 1. Results show that the rank pooling method is only slightly better than max pooling or mean pooling when used with VGG16 features. We believe this is due to the limited capacity of rank pooling [20,18]. Moreover, temporal pyramid seems to outperform rank pooling except for HMDB51 dataset. Moreover, as shown in Table 1, when we extend the rank pooling to *recursive rank pooling*, we notice a jump in performance from 44.2% to 52.5% for Hollywood2 dataset and 40.9%



(a) Hollywood2 (b) HMDB-51

Fig. 5: Activity recognition performance versus hierarchy depth on Hollywood2 and HMDB-51.



(a) Hollywood2 (b) HMDB-51 (c) UCF101

Fig. 6: Activity recognition performance versus window size (top) and stride (bottom).

to 45.8% for HMDB51 dataset. We also see a noticeable improvement in UCF101 dataset. Hierarchical rank pooling improves over rank pooling by a significant margin. The results suggest that it is important to exploit dynamic information in a hierarchical manner as it allows complicated sequence dynamics of videos to be expressed. To verify this, we also performed an experiment by varying the depth of the hierarchical rank pooling and reported results for one to three layers. Results are shown in Figure 5.

As expected the improvement from depth of one to two is significant. Interestingly, as we increase the depth of the hierarchy to three, the improvement is marginal. Perhaps with only two levels, one can obtain a high capacity dynamic encoding.

7.1.2 Evaluating the parameters of HRP

Hierarchical rank pooling consists of two more hyper-parameters: (1) **window size** (M_ℓ), i.e., the size of the video sub-sequences and (2) **stride** (S_ℓ) of the video sampling. These two parameters control how many sub-sequences can be generated at each layer. In the next experiment we evaluate how performance varies with **window size** and **stride**. Results are reported in Figure 6(top). The window size does not seem to make a big impact on the results (1–2%) for some datasets. However, we experimentally verified that a window size of 20 frames seems to be a reasonable compromise for all activity recognition tasks. The trend in Figure 6(bottom) for the stride is interesting. It shows that the best results are always obtained by using a small stride. Small strides generate more encoded sub-sequences capturing more statistical information.

7.1.3 The effect of non-linear feature maps on HRP

Non-linear feature maps are important for modeling complex dynamics of an input video sequence. In this section we compare Sign Expansion Root (SER) feature map introduced in Section 6.1 with the Signed Square Root (SSR) method, which is commonly used in the literature [49]. Results are reported in Table 2. As evident in the table, SER feature map is useful not only for hierarchical rank pooling, which gives an improvement of 6.3% over SSR, but also for baseline rank pooling method, which gives an improvement of 6.8%. This seems to suggest that there is valuable information in both positive and negative activations of fully connected layers. Furthermore, this experiment suggests that it is important to consider positive and activations separately for activity recognition.

METHOD	Rank pooling	Hierarchical rank pooling
Signed square root (SSR)	44.2	50.5
Sign expansion root (SER)	51.0	**56.8**

Table 2: Effect non-linear feature maps during the training of rank pooling methods using Hollywood2 dataset.

7.1.4 The effect of non-linear kernel SVM on HRP

In this experiment we evaluate several non-linear kernels that exist in literature and compare their effect when used with Hierarchical Rank Pooling method. We compare classification performance using different kernels (1) linear, (2) linear kernel with SSR, (3) Chi-square kernel, (4) Kernelized SER (5) combination of Chi-square kernel with SER. Results are reported in Table 3. On all three datasets we see a common trend. First, the SSR kernel is more effective than not utilizing any kernel or feature map. Interestingly, on deep CNN features, Chi-square Kernel is more effective than SSR. Perhaps this is because the Chi-square kernel utilizes both negative and positive activations in a separate manner to some extent. The SER method seems to be the most effective kernel. Interestingly, applying SER feature map over Chi-square kernel seems to improve results further. We conclude that SER non-linear feature map is effective not only during the training of rank pooling techniques, but also for action classification specially when used with CNN activa-

KERNEL TYPE	Hollywood2 (mAP %)	HMDB51 (%)	UCF101 (%)
Linear	45.1	40.0	66.7
Signed square root (SSR)	48.6	42.8	72.0
Chi-square kernel	50.6	44.2	73.8
Sign expansion root (SER)	54.0	46.0	76.6
Chi-square + SER	**56.8**	**47.5**	**78.8**

Table 3: Effect of non-linear SVM kernels on action classification with hierarchical rank pooling representation.

KERNEL TYPE	Avg. pool	Max pool	Rank pool	Ours
Linear	38.1	39.6	33.3	45.1
Signed square root (SSR)	38.6	38.4	35.3	48.6
Chi-square kernel	39.9	41.1	40.8	50.6
Sign expansion root (SER)	39.4	41.0	37.4	54.0
Chi-square + SER	**40.9**	**42.4**	**44.2**	**56.8**

Table 4: Effect of non-linear kernels on other pooling methods using Hollywood2 dataset (mAP %).

tion features. Next we also evaluate the effect of non-linear kernels on final video representations when used with other pooling methods such as rank pooling, average pooling and max pooling. Results are reported in Table 4 on Hollywood2 dataset. A similar trend as in the previous table can be observed here. We conclude that our kernalized SER is useful not only for our hierarchical rank pooling method, but also for the other considered temporal pooling techniques.

7.1.5 Combining hierarchical rank pooled CNN features with improved trajectory features

In this experiment we combine hierarchical rank pooled CNN features with the Improved Dense Trajectory (IDT) features (MBH, HOG, HOF) [68]. The objective of this experiment is to show the complimentary nature of IDT and hierarchical rank pooled CNN features. IDT are encoded with Fisher vectors [49] at the frame level and then temporally encoded with rank pooling. Due to the very high dimensional nature of Fisher vectors, it is not practical to use hierarchical rank pooling over Fisher vectors. We utilize a Gaussian mixture model of 256 components to create the Fisher vectors. To keep the dimensionality manageable, we halve the size of each descriptor using PCA. This is exactly the same setup used by Fernando et al. [20, 18]. For each dataset we report results on HOG, HOF and MBH features obtained with the publicly available code of rank pooling [20, 18]. We construct a kernel gram matrix for each feature type (HOG, HOF, MBH, and CNN) and take the averaging of the kernels to fuse features. Results are shown in Table 5. Hierarchical rank pooled (CNN) outperforms trajectory based HOG features on all three datasets. Furthermore, on UCF101 dataset, Hierarchical rank pooled (CNN) outperforms rank pooled HOF features. Nevertheless, trajectory based MBH features

METHOD	Hollywood2	HMDB51	UCF101
RP. (HOG)	53.4	44.1	72.8
RP. (HOF)	64.0	53.7	78.3
RP. (MBH)	65.8	53.9	82.6
RP. (ALL)	68.5	60.0	86.5
RP. (CNN)	44.2	40.9	72.2
RP. (ALL+CNN)	71.4	63.0	88.1
HRP. (CNN)	56.8	47.5	78.8
RP. (ALL)+ HRP (CNN)	**74.1**	**65.0**	**90.7**

Table 5: Combining CNN-based Hierarchical Rank Pooling (HRP) with improved trajectory features encoded with Fisher vectors and Rank Pooling(RP).

still dominate the best results for an individual feature. The combination of rank pooled trajectory features (HOG + HOF + MBH) with hierarchically rank pooled CNN features gives a significant improvement. It is interesting to see that the biggest improvement is obtained in Hollywood2 dataset. On UCF-101 dataset the combination brings us an improvement of 4.2% over rank pooled trajectory features. We conclude that our hierarchical rank pool features are complimentary to trajectory-based rank pooling.

7.1.6 Combining with trajectory features

We also apply hierarchical rank pooling over improved dense trajectories which are encoded with the bag-of-words. For this experiment, we use MBH features and use a dictionary of size 4096 which is constructed with K-means. Results are reported in Table 6. As before, both average pooling and

Method	UCF101 Acc. (%)	HMDB51 Acc. (%)
Average pooling	72.3	45.0
Max pooling	71.5	43.1
Rank pooling	77.5	48.1
Hierarchical rank pooling	82.1	54.2

Table 6: Action classification performance using IDT (MBH) features encode with BOW (4096 dictionary) using different temporal pooling methods and SVM classifiers.

max pooling perform worst than the rank pooling method. Hierarchical rank pooling obtains large improvement over rank pooling. On HMDB51 dataset, the improvement over rank pooling is about 6%. HRP obtains an improvement of 4.6% on UCF101 over rank pooling. It is interesting to see the impact of hierarchical rank pooling over deep features as well as traditional hand-crafted features such as dense trajectory features and bag-of-words encoding. We conclude that the hierarchical rank pooling is effective not only on recent deep features, but also with more traditional IDT-based bag-of-words features.

Method	UCF101 Acc. (%)	HMDB51 Acc. (%)
Average pooling	76.5	48.3
Max pooling	78.8	50.2
Rank pooling	81.0	54.7
Hierarchical rank pooling	**84.0**	**57.3**

Table 7: Action classification performance using non-fine-tuned ResNet features [26] using different temporal pooling methods and SVM classifiers.

7.1.7 The impact of residual network features on HRP

In this experiment, we evaluate the impact of Residual Network Features [26] on action recognition using UCF101 and HMDB51 datasets. Results for max pooling, average pooling, Rank pooling, and Hierarchical Rank pooling with ResNet features are shown in Table 7 for UCF101 and HMDB51 datasets. For this analysis, we extract frame level ResNet features from the output of final pooling layer which has a dimensionality of 2048. We compare our hierarchical rank pooling with max pooling method. For rank pooling we obtain classification accuracy of 84.0% only using frame-level ResNet features on UCF101. This is an improvement of 5.3% over VGG-16 features. Similarly, for max pooling we obtain 78.8 % which is an improvement of 6.3 % over VGG-16. Similar trends can be observed for HMDB51 dataset. In fact, for HMDB51, it seems the improvement from VGG-16 to ResNet features is significant (11.2 % for average pooling, 11.1 % for max pooling, 13.8 % for rank pooling and 9.8 % for hierarchical rank pooling).

In another experiment, we also used publicly available ResNet-152 networks that are finetuned for RGB stream [17]. Then only using the center crop of UCF101 frames, we extract 2048 dimensional features per frame and experiment with several baseline methods. For RNN and LSTM baselines, we use Keras [9] with hidden size of 256. We report results in Table 8. Interestingly, simple RNN and LSTM methods does not outperform max pooling or the average pooling results. Rank pooling is better than max pooling while hierarchical rank pooling is significantly better than rank pooling.

We conclude that ResNet feature [26] are useful for action and activity recognition and our proposed hierarchical rank pooling method is complimentary to both VGG-16 features [57] as well as ResNet features.

7.1.8 Confusions with the use of residual network features and HRP

We also analyse the confusions made by ResNets when pooled using max operator and hierarchical rank pooling (see Figure 7). The most confusing category for max pooling is *Swing* for *Tennis swing* (44 times) and *Basketball* for *Basketball-Dunk* (37 times) (–see Figure 8 left). The most confusing

Method	UCF101
Simple RNN	74.8
Simple LSTM	75.9
Stacking of two LSTMs	75.3
Average pooling	79.1
Max pooling	81.3
Rank pooling	82.1
Hierarchical rank pooling	**85.6**

Table 8: Action classification performance using fine-tuned ResNet-152 features [17] using spatial stream. We use different temporal pooling methods and compare results on UCF101 dataset.

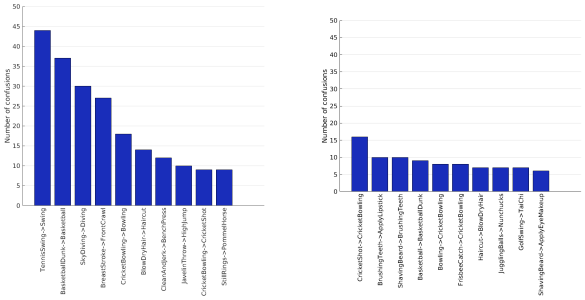


Fig. 7: The most confused classes for max pooling (left) and hierarchical rank pooling in the right. Max pooling makes most confusions.

for hierarchical rank pooling is *Cricket-Bowling* for *Cricket-Shot* which happens only 16 times (–see Figure 8 right). Generally, from the dynamics point of view, it is very hard to distinguish Cricket bowling from Cricket-Shot as indeed the Cricket-Shot just follows after Cricket-bowling. In particular, in many cases Cricket-bowling can be observed for Cricket-Shot video clips in UCF101 dataset.

7.1.9 Impact of mid-level pooled features

In this experiment, we evaluate the impact of low-level, learned mid-level, and higher level features of the hierarchical rank pooling. We use non-fine-tuned ResNet-150 features [26] as the frame representation. As before we use a window size of 20 and stride of 1. After applying three layered hierarchical rank pooling, we use the first/second layer mid-level features as the mid-level sequence representations. We randomly pick a mid-level feature vector to represent the entire video sequence. To compare, we also pick a single frame feature to represent a video. Furthermore, we randomly select 39 frames from each video and apply temporal max pooling and temporal average pooling as baselines. We evaluate the impact of each mid-level feature and position the results with respect to the highest level hierarchical rank-pooled feature. We repeat each experiment 10 times and report the mean and standard deviation in Table 9. Clearly,

Fig. 8: Some of the most confusing classes for max pooling (left) and hierarchical rank pooling (right) respectively using ResNet features.

Level	HMDB51 Acc. (%)
0 - frame level	38.8 ± 1.1
temporal max pooling (39 frames)	49.2 ± 1.6
temporal avg. pooling (39 frames)	46.9 ± 0.5
1^{st} layer mid-level feature	41.9 ± 1.4
2^{nd} layer mid-level feature	47.5 ± 0.6
3^{rd} layer Hierarchical rank pooling	57.4

Table 9: The impact of low-level, mid-level, and highest level sequence representation using hierarchical rank pooling method.

frame level feature performs the worst. This is expected. Interestingly, using just a single random frame, we are able to obtain a mean classification accuracy of 38.8 %. First layer mid-level feature is better than frame level representation which obtains 41.9 %. The second layer mid-level feature is even better which obtains 47.5 %. This is an indication of the impact of mid-level dynamics. Note that the temporal resolution of the first layer feature is 20 frames while the second layer mid-level feature has a resolution of 39 frames. Most interestingly, the highest level features obtain 57.4 % which is significant. However, the highest level feature has the full temporal resolution. These results suggest that indeed, the hierarchical rank pooling is capable of capturing low-level, mid-level and higher level dynamics. The highest level temporal dynamics captured by HRP improves the activity recognition performance significantly.

7.2 The effect of discriminative rank pooling

In this experiment, we use discriminative rank pooling in the final layer of the hierarchical rank pooling network. In this case we first construct the sequence for the final layer $X^{(L)}$ and apply SSR feature map. Then we feed forward this sequence through the parameterized non-linear transform $\psi(Wx_t^{(L)})$, temporal encoder $\phi(\widetilde{X}^{(L)})$, and apply the classifier to get a classification score. During training we propagate errors back to the parametric non-linear transformation layer $\psi(\cdot)$ and perform a parameter update. We implement this optimization in a GPU.

METHOD	HMDB51	UCF101
Rank pooling	40.9	72.2
Hierarchical rank pooling	47.5	78.8
Discriminative hierarchical rank pooling	$\mathbf{49.9 \pm 0.08}$	$\mathbf{81.4 \pm 0.04}$

Table 10: Effect of learning discriminative dynamics for hierarchical rank pooling on the HMDB51 and UCF101 dataset.

We use MatConvNet [65] with stochastic gradient descent with variable learning rate starting at 10^{-3} and decreased to 10^{-5} in a logarithmic manner over epochs. We also use a momentum term of 0.9 and a weight decay of 0.0005. Our layer is implemented in matlab with GPU support. We evaluate the effect of this method only on the largest datasets, the HMDB51 and UCF101. We first construct the first layer sequence using hierarchical rank pooling. Then we learn the parameters W using the labelled video data while keeping the CNN parameters fixed. We initialize the W matrix to the identity and the classifier parameters to those obtained from the linear SVM classifier. Results are reported in Table 10. We improve results by 2.4% and 2.6% over hierarchical rank pooling and a significant improvement of 9.0% and 9.2% over rank pooling using HMDB51 and UCF101 datasets respectively. During test time, we process a video at 120 frames per second.

7.3 Comparing the effect of end-to-end trainable rank pooled CNN.

In this section we evaluate the effectiveness of end-to-end video representation learning with rank-pooling introduced in section 4.1. Due to the computational complexity, we only use moderate (Hollywood2) and small scale (UCF sports) action recognition dataset for evaluation. We compare our end-to-end training of the rank-pooling network against the following baseline methods.

Method	Acc. (%)
Average pooling + svm	67.1
Max pooling + svm	66.0
Rank pooling + svm	66.4
Average pooled-cnn-end-to-end	70.4
Max pooled-cnn-end-to-end	71.2
Frame-level fine-tuning	69.8
Frame-level fine-tuning + Rank pooling	72.9
Rank-pooled-cnn-end-to-end	**87.1**

Table 11: Classification accuracies for action recognition on the ten-class UCF-sports dataset [51] using end-to-end video representation learning with rank pooling.

avg pooling + svm: We extract FC7 feature activations from the pre-trained Caffe reference model [33] using MatConvNet [65] for each frame of the video. Then we apply temporal average pooling to obtain a fixed-length feature vector per video (4096 dimensional). Afterwards, we use a linear SVM classifier (LibSVM) to train and test action and activity categories.

max pooling + svm: Similar to the above baseline, we extract FC7 feature activations for each frame of the video and then apply temporal max pooling to obtain a fixed-length feature vector per video. Again we use a linear SVM classifier to predict action and activity categories.

rank pooling + svm: We extract FC7 feature activations for each frame of the video. We then apply time varying mean vectors to smooth the signal as recommended by [20], and L2-normalize all frame features. Next, we apply the rank-pooling operator to obtain a video representation using publicly available code [20]. We use a linear SVM classifier applied on the L2-normalized representation to classify each video.

frame-level fine-tuning (fn): We fine-tune the Caffe reference model on the frame data considering each frame as an instance from the respective action category. Then we sum the classifier scores from each frame belonging to a video to obtain the final prediction.

frame-level fine-tuning + rank-pooling (fn+rankpool): We use the pre-trained model as before and fine-tune the Caffe reference model on the frame data considering each frame as an instance from the respective action category. Afterwards, we extract FC7 features from each video (frames). Then we encode temporal information of fine-tuned FC7 video data using rank-pooling. Afterwards, we use soft-max classifier to classify videos.

end-to-end baselines: We also compare our method with end-to-end trained max and average pooling variants. Here the pre-trained CNN parameters were fine-tuned using the classification loss.

The first five baselines can all be viewed as variants of the CNN-base temporal pooling architecture of Figure 2. The differences being the pooling operation and whether end-to-end training is applied.

We compare the baseline methods against our rank-pooled CNN-based temporal architecture where training is done end-to-end. We do not sub-sample videos to generate fixed-length clips as typically done in the literature (e.g., [56,64]). Instead, we consider the entire video during training as well as testing. We use stochastic gradient descent method without batch updates (i.e., each batch consists of a single video). We initialize the network with the Caffe reference model and use a variable learning rate starting from 0.01 down to 0.0001 over 60 epochs. We also use a weight decay of 0.0005 on an L2-regularizer over the model parameters. We explore two variants of the learning algorithm. In the first variant we use the diagonal approximation to the rank-pool gradient during the back-propagation. In the second variant we use the full gradient update, which requires computing the inverse of matrices per video (see Section 5.5). For the UCF-sports dataset we use the cross-entropy loss for all CNN-based methods (including the baselines). Whereas for the Hollywood2 dataset, where performance is measured by mAP (as is common practice for this dataset), we use the hinge-loss.

Results for experiments on the UCF-sports dataset are reported in Table 11. Let us make several observations. First, the performance of max, average and rank-pooling are similar when CNN activation features are used without end-to-end learning. Perhaps increasing the capacity of the model to better capture video dynamics (say, using a non-linear SVM) may improve results perhaps a future work. Second, end-to-end training helps all three pooling methods. However, the improvement obtained by end-to-end training of rank-pooling is about **21%**, significantly higher than the other two pooling approaches. Moreover, the performance using the diagonal approximation is 87.0% which is very close to the full gradient based approach. This suggests that the diagonal approximation is driving the parameters in a desirable direction and may be sufficient for a stochastic gradient-based method. Last, and perhaps most interesting, is that using state-of-the-art improved trajectory [68] features (MBH, HOG, HOG) and Fisher vectors [49] with rank-pooling [20] obtains 87.2% on this dataset. This result is comparable with the results obtained with our method using end-to-end feature learning. Note, however, that the dimensionality of the feature vectors for the state-of-the-art method are extremely high (over 50,000 dimensional) compared to our 4,096 dimensional feature representation.

We now evaluate activity recognition performance on the Hollywood2 dataset. Results are reported in Table 12 as average precision performance for each class and we take the mean average precision (mAP) to compare methods. As before, for this task, the best results are obtained by end-to-end training using rank-pooling for temporal encoding. The

class	avg+svm	max+svm	rankpool+svm	avg+cnn	max+cnn	fn	fn+rankpool	rankpool+cnn
AnswerPhone	23.6	19.5	**35.3**	29.9	28.0	27.4	34.3	25.0
DriveCar	**60.9**	50.8	40.6	55.6	48.6	48.1	50.4	56.9
Eat	19.7	22.0	16.7	**27.8**	22.0	21.1	23.1	24.2
FightPerson	**45.6**	28.3	28.1	26.6	17.6	18.4	20.4	30.4
GetOutCar	39.5	29.2	28.1	48.9	43.8	43.1	45.3	**55.5**
HandShake	28.3	24.4	34.2	38.4	**40.0**	39.4	39.5	32.0
HugPerson	30.2	23.9	22.1	25.9	26.6	26.1	30.3	**33.2**
Kiss	38.2	27.5	36.8	50.6	45.7	44.9	45.6	**54.2**
Run	55.2	53.0	39.4	59.6	52.5	52.4	52.9	**61.0**
SitDown	30.0	28.8	32.1	30.6	30.0	29.7	34.4	**39.6**
SitUp	23.0	20.2	18.7	23.8	**26.4**	24.1	25.1	25.4
StandUp	34.6	32.4	39.9	37.4	34.8	34.4	34.8	**49.9**
mAP	35.7	30.0	31.0	37.9	34.7	34.1	36.3	**40.6**

Table 12: Classification performance in average precision for activity recognition on the Hollywood2 dataset [43] using end-to-end video representation learning with rank pooling.

improvement over non-end-to-end rank pooling is **9.6** mAP. One may ask whether this performance could be achieved without end-to-end training but just fine-tuning the frame-level features. Simple frame-level fine-tuning obtains only 34.1 mAP (see Table 12 with the column denoted by fn) while frame-level fine-tuning + rank-pooling obtains 36.3 mAP (see Table 12 with the column denoted by fn+rankpool). Our end-to-end method obtains better results (40.6 mAP) compared to frame-level fine-tuning and fine-tuning with rank-pooling.

7.4 Comparing to the state-of-the-art

In this section we position our paper with respect to the current state-of-the-art performance in action recognition using standard datasets. We perform a series of experiments using hierarchical rank pooled deep cnn features for UCF101 and HMDB51 datasets. We use two types of cnn features, one extracted from VGG-16-CNN architecture and the other extracted from ResNet architecture. We also experimented with discriminative hierarchical rank pooling. To further improve results, we use rank pooled [18] improved dense trajectory features (IDT) [68] and optical-flow-based [7] deep features for UCF101 and HMDB51 datasets. It should be emphasized, we choose parameters for hierarchical rank pooling based on the prior experimental results reported in Figures 5 and 6 for each dataset, i.e., *without* use of any grid search. As in [20,27] we use data augmentation only for Hollywood2 and HMDB51. Results are reported in the following Table 13.

When ResNet (RGB) features are combined with IDT, our hrp-based method obtains staggering 93.1% on UCF101. Furthermore, if we add optical flow-based features (similar to RGB-based hierarchical rank pooling), we obtain 93.6% classification performance on UCF101 dataset. Only using ResNet-based RGB data and Optical Flow data, hierarchical rank pooling with default settings obtains 90.6% on UCF101. Similarly, on HMDB51 dataset, hierarchical rank pooled ResNet (RGB + Optical-flow) features obtains 63.1%. When we combine that with IDT features, for HMDB51 dataset we obtain 69.4 % which is par with the state-of-the art for this dataset. On Hollywood2 dataset, hierarchical rank pooled VGG-16 features are combined with IDT to obtain state-of-the art 76.7 mAP. This is a significant improvement over rank pooling [20] method.

Because, different methods used different information such as optical flow features, different motion representations, different object models and trajectory-based features, it is difficult to compare methods in a purely fair manner using the published results alone. However, from these results obtained in Table 13, we conclude that our sequence encoding method and end-to-end learning method are complimentary to existing techniques and video data and features.

8 Conclusion

In this paper we extend the rank pooling method in two ways. First, we introduce an effective, clean, and principled temporal encoding method based on the discriminative rank pooling framework which can be applied over vector sequences or convolutional neural network-based video sequences for action classification tasks. Our temporal pooling layer can sit above any CNN architecture and through a bilevel optimization formulation admits end-to-end learning of all model parameters. We demonstrated that this end-to-end learning significantly improves performance over a traditional rank-pooling approach by 21% on the UCF-sports dataset and 9.6 mAP on the Hollywood2 dataset.

Secondly, we presented a novel temporal encoding method called *hierarchical rank pooling* which consists of a network of non-linear operations and rank pooling layers. The obtained video representation has high capacity and capability of capturing informative dynamics of rich frame-based feature representations. We also presented a principled way to

Method	Feature	Holly.2	HMDB51	UCF101
hrp	ResNet (RGB+Opt.Flow) + IDT	–	**69.4**	**93.6**
hrp	ResNet (RGB) + IDT	–	68.9	93.1
dhrp	VGG-16 (RGB) + IDT	–	68.1	91.4
hrp	VGG-16 (RGB) + IDT	**76.7**	66.9	91.2
hrp	ResNet(RGB+Opt.Flow)	–	63.1	90.6
Zha et al. [72]	VGG-19 (RGB)+IDT			89.6
Ng et al. [71]	GoogLeNet (RGB + Opt.FLow)			88.6
Simonyan et al. [56]	CNN-M-2048 (RGB + Opt.FLow)		59.4	88.0
Wang et al. [69]	CNN-M-2048 (RGB + Opt.FLow) + IDT		65.9	91.5
Feichtenhofer et al. [17]	VGG-16 (RGB+Opt.Flow) + IDT		69.2	93.5
Methods without CNN features				
Lan et al. [42]	IDT	68.0	65.4	89.1
Fernando et al. [20]	IDT	73.7	63.7	
Hoai et al. [27]	IDT	73.6	60.8	
Peng et al. [48]	IDT		66.8	
Wu et al. [70]	IDT		56.4	84.2
Wang et al. [68]	IDT	64.3	57.2	

Table 13: Comparison with the state-of-the-art methods.

learn non-linear dynamics using a stack consisting of parametric non-linear activation layers, rank pooling layers, discriminative rank pooling layer and, a soft-max classifier which we coined *discriminative hierarchical rank pooling*. We demonstrated substantial performance improvement over other temporal encoding and pooling methods such as max pooling, rank pooling, temporal pyramids, and LSTMs. Combining our method with features from the literature, we obtained good results on the Hollywood2, HMDB51 and UCF101 datasets.

One of the limitations of our rank pooling-based end-to-end learning is the computational complexity. Especially, the gradient computation of the rank-pooling operator is computationally expensive which limits applicability of end-to-end learning on very large datasets. One solution is to simplify the gradient computation or relax the constraints of the learning objective function as shown in prior work [4, 3]. If one wants to use discriminative rank pooling inside hierarchical rank pooling networks, then perhaps one can find a strategy to reuse the gradient computation of the neighbouring subsequences. These are possible solutions to make the back-propagation faster in our proposed framework.

We believe that the framework proposed in this paper will open the way for embedding other traditional optimization methods as subroutines inside CNN architectures. Our work also suggests a number of interesting future research directions. First, it would be interesting to explore more expressive variants of rank-pooling such as through kernalization. Second, our framework could be adapted to other sequence classification tasks (e.g., speech recognition) and we conjecture that as for video classification there may be accuracy gains for these other tasks too.

Acknowledgements This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

1. Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.
2. Jonathan F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Press, 1998.
3. Hakan Bilen, Basura Fernando, Efstratios Gavves, and Andrea Vedaldi. Action recognition with dynamic image networks. *arXiv preprint arXiv:1612.00738*, 2016.
4. Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.
5. Olivier Bousquet and André Elisseeff. Stability and generalization. *JMLR*, 2:499–526, 2002.
6. Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *CVPR*, pages 568–574. IEEE, 1997.
7. Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.
8. Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
9. François Chollet. Keras, 2015.
10. S Dempe and S Franke. On the solution of convex bilevel optimization problems. *Computational Optimization and Applications*, 63(3):685–703, 2016.
11. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.
12. Chuong B. Do, Chuan-Sheng Foo, and Andrew Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, 2007.
13. Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012.
14. Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012.
15. Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015.

16. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.
17. Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, June 2016.
18. B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *TPAMI*, PP(99):1–1, 2016.
19. Basura Fernando, Peter Anderson, Marcus Hutter, and Stephen Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016.
20. Basura Fernando, Efstratios Gavves, Jose Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015.
21. Basura Fernando and Stephen Gould. Learning end-to-end video classification with rank-pooling. In *ICML*, 2016.
22. Emily Fox, Michael I Jordan, Erik B Sudderth, and Alan S Willsky. Sharing features among dynamical systems with beta processes. In *NIPS*, pages 549–557, 2009.
23. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
24. Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3 edition, 1996.
25. Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 1(1):1, July 2016.
26. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.
27. Minh Hoai and Andrew Zisserman. Improving human action recognition using score distribution and ranking. In *ACCV*, 2014.
28. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
29. M. C. Hughes and E. B. Sudderth. Nonparametric discovery of activity patterns from video collections. In *CVPR Workshops*, pages 25–32, June 2012.
30. Mihir Jain, Hervé Jégou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.
31. Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311. IEEE, 2010.
32. Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013.
33. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.
34. Thorsten Joachims. Training linear svms in linear time. In *ICKDD*, 2006.
35. 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.
36. 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.
37. Teresa Klatzer and Thomas Pock. Continuous hyper-parameter learning for support vector machines. In *Computer Vision Winter Workshop (CVWW)*, 2015.
38. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
39. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.
40. Karl Kunisch and Thomas Pock. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983, 2013.
41. Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015.
42. Zhengzhong Lan, Ming Lin, Xuanchong Li, Alex G. Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015.
43. Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
44. Yingwei Li, Weixin Li, Vijay Mahadevan, and Nuno Vasconcelos. Vlad3: Encoding dynamics of deep features for action recognition. In *CVPR*, 2016.
45. Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
46. Lie Lu, Hong-Jiang Zhang, and Hao Jiang. Content analysis for audio classification and segmentation. *IEEE Transactions on speech and audio processing*, 10(7):504–516, 2002.
47. P. Ochs, R. Ranftl, T. Brox, and T. Pock. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 654–665, 2015.
48. X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014.
49. Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.
50. Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.
51. Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.
52. Michael S. Ryoo, Brandon Rothrock, and Larry Matthies. Pooled motion features for first-person videos. In *CVPR*, June 2015.
53. Kegan G. G. Samuel and Marshall F. Tappen. Learning optimized MAP estimates in continuously-valued MRF models. In *CVPR*, 2009.
54. Ozan Sener, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *ICCV*, pages 4480–4488, 2015.
55. Kazuo Shinozaki, Kazuko Yamaguchi-Shinozaki, and Motoaki Seki. Regulatory network of gene expression in the drought and cold stress responses. *Current opinion in plant biology*, 6(5):410–417, 2003.
56. Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.
57. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 1(1):1, 2014.
58. Cees Snoek, Bernard Ghanem, and Juan Carlos Niebles. The activitynet large scale activity recognition challenge, 2016.
59. Yale Song, Louis-Philippe Morency, and Randall Davis. Action recognition by hierarchical sequence summarization. In *CVPR*, 2013.
60. Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 1(1):1, 2012.
61. Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 1(1):1, 2015.

62. Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
63. Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.
64. Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.
65. A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.
66. Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential recurrent neural networks for action recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
67. Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103:60–79, 2013.
68. Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
69. Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015.
70. Jianxin Wu, Yu Zhang, and Weiyao Lin. Towards good practices for action video encoding. In *CVPR*, 2014.
71. Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.
72. Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. In *BMVC*, 2015.