Learning End-to-end Video Classification with Rank-Pooling

Basura Fernando BASURA.FERNANDO@ANU.EDU.AU
Research School of Engineering, The Australian National University, ACT 2601, Australia

Stephen Gould STEPHEN.GOULD@ANU.EDU.AU
Research School of Computer Science, The Australian National University, ACT 2601, Australia

Abstract

We introduce a new model for representation learning and classification of video sequences. Our model is based on a convolutional neural network coupled with a novel temporal pooling layer. The temporal pooling layer relies on an inner-optimization problem to efficiently encode temporal semantics over arbitrarily long video clips into a fixed-length vector representation. Importantly, the representation and classification parameters of our model can be estimated jointly in an end-to-end manner by formulating learning as a bilevel optimization problem. Furthermore, the model can make use of any existing convolutional neural network architecture (e.g., AlexNet or VGG) without modification or introduction of additional parameters. We demonstrate our approach on action and activity recognition tasks.

1. Introduction

Representation learning from sequence data has many applications including, but not limited to, action and activity recognition from video, gesture recognition, music classification, and gene regulatory network analysis. 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. In this paper we present a principled method to jointly learn discriminative dynamic representations and classification model parameters from video data using convolutional neural networks (CNNs).

In recent years CNNs have become very popular for automatically learning representations from large collections of static images. Many application domains, such as image classification, image segmentation and object detection, have benefited from such automatic representation learning (Krizhevsky et al., 2012; Girshick et al., 2014). 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 propose to use recurrent neural networks (RNNs) (Du et al., 2015) or extensions, such as long short term memory (LSTM) networks (Srivastava et al., 2015), 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 data is 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 (Deng et al., 2009)), the largest fully labelled action recognition dataset, UCF101, consists of barely more than 13,000 videos (Soomro et al., 2012). 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 (Karpathy et al., 2014). In fact, an arbitrary reshuffling of the frames would produce an identical video representation under these pooling schemes. Rank-pooling (Fernando et al., 2015), 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 CNN features). The parameters of the ranking machine are then

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

used as the video representation. However, unlike max and average pooling, it is unclear how the CNN parameters can be fine-tuned to give a more discriminative representation when rank-pooling is used.

In this paper, we present a novel approach for learning discriminative representations for videos using rank-pooling over a sequence CNN feature vectors (derived from the video frames). We do this by formulating an optimization problem that jointly learns the video representation and classifier parameters. A key technical challenge, however, is that the optimization problem contains the rank-pooling linear ranking machine as a subproblem—itself a non-trivial optimization problem. This leads to a large-scale bilevel optimization problem (Bard, 1998) with convex inner-problem, which we propose to solve by stochastic gradient descent.

Moreover, because we use support vector regression to solve the inner-optimization problem and obtain our video representation, there are theoretical stability guarantees on the learned temporal representation (Bousquet & Elisseeff, 2002). That is, even if the input sequence is perturbed the output of the temporal encoding layer produces stable video representations leading to a robust model in contrast to CNN-RNN/LSTM models, which can be very sensitive to changes in the input.

Our contributions are two-fold: First, we present an elegant method for encoding temporal information in video sequences from frame-based CNN features. Second, we show that the video representation and classifier parameters can be learned jointly in an end-to-end fashion using a bilevel optimization formulation of the problem. We demonstrate the effectiveness of our method on two challenging video classification datasets.

2. Related work

Temporal information encoding of video sequences using neural networks is an active field of research in both the machine learning and computer vision communities. Recently, several methods have been proposed to tackle the problem of video sequence encoding using neural network architectures (Ji et al., 2013; Donahue et al., 2015; Srivastava et al., 2015; Yue-Hei Ng et al., 2015; Zha et al., 2015; Simonyan & Zisserman, 2014).

Some authors (e.g., (Ji et al., 2013) and (Tran et al., 2014)) propose to use 3D convolutions to incorporate spatial and motion information. However, it is not clear if temporal information of videos can be processed in a similar manner to the spatial information of images. Therefore the use of 3D-convolutions to capture motion may not be the ideal solution. Moreover, these 3D filers are applied over very short video clips capturing only local motion information. Consequently, they are not able to capture long-range dynamics of complex activities. Most of the CNN-based methods use fixed-length short video clips to learn video representations ignoring long-range dynamics (e.g., (Ji et al., 2013; Tran et al., 2014; Simonyan & Zisserman, 2014)). This is not ideal as it is essential use all available temporal information to learn good video representations, especially in tasks such as activity recognition.

LSTM-based methods have also been proposed to learn video representations. For example, unsupervised video representation learning method is presented in Srivastava et al. (2015). In that work temporal information is learned by encoding a video using an LSTM model and by decoding the encoded vector to reconstruct the video in the reverse order. However, it is unclear how to adapt such a strategy to encode discriminative dynamics of a video. At the same time the LSTM model is trained on extracted feature activations from CNNs. However, the LSTM and CNN parameters are not trained jointly, leading to suboptimal parameter settings. The LRCN method proposed by Donahue et al. (2015) has several nice properties such as the ability to train an end-to-end model and handle variable length sequences. However, for inference this method takes the average prediction over all frames, which can destroy valuable dynamical information found by the LSTM component of the LRCN model.

In comparison to these methods, our model is trained in a principled end-to-end fashion using a single convolutional neural network. It captures the time varying information using rank pooling (Fernando et al., 2015; 2016b) and has the ability to handle variable length sequences. Importantly, our method captures video-wide temporal information and does not require the ad hoc assembly of disjoint components typical of other video classification frameworks. A schematic illustration of our model is shown in Figure 1.

Our learning algorithm introduces a bilevel optimization method for encoding temporal dynamics of video sequences using convolutional neural networks. Bilevel optimization (Bard, 1998) 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 (Ochs et al., 2015) or studying replacement of the lower level problem with necessary conditions for optimality (Dempe & Franke, 2015). It has recently gained interest in the machine learning community in the context of hyperparameter learning (Klatzer & Pock, 2015; Do et al., 2007) and in the computer vision community in the context of image denoising (Domke, 2012; Kunisch & Pock, 2013). 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.

Two recent rank-pooling methods with learning have also been proposed (Fernando et al., 2016a; Bilen et al., 2016). Fernando et al. (2016a) propose a discriminative hierarchical rank pooling method on extracted CNN features. However, this method does not learn an end-to-end network. In contrast, Bilen et al. (2016) does learn an end-to-end network, but the rank-pooling operator is simplified for efficiency and only applied to input RGB data.

3. Learning Sequence Classification

In this section we describe our method for sequence classification and the associated end-to-end parameter learning problem. We start by formalising our sequence representation and prediction pipeline. We then present our main contribution—jointly learning the representation of the elements in the sequence and the parameters of the classifier.

3.1. Representation

We consider the problem of classifying a sequence $\vec{\boldsymbol{x}} = \langle \boldsymbol{x}_t \mid t = 1, \dots, T \rangle$ by assigning it a label y from some discrete set of classes $\mathcal{Y}$. For example, the sequence can be a video and the label can be the action occurring in the video. Here $\mathcal{Y}$ is the set of recognizable actions such as "running", "jumping", "skipping", etc. We assume that each element $\boldsymbol{x}_t$ of the sequence is an object from some input domain $\mathcal{X}$ (e.g., a video frame).

Our first task is to transform the arbitrary-length sequence into a form that is amenable to classification. To this end we first map each element of the sequence into a p-dimensional feature vector via a parameterizable feature function $\psi_{\boldsymbol{\theta}}(\cdot)$,

$$\boldsymbol{v}_t = \psi_{\boldsymbol{\theta}}(\boldsymbol{x}_t) \in \mathbb{R}^p. \tag{1}$$

The feature function can be, for example, a convolutional neural network (CNN) applied to a video frame with features extracted from the final activation layers in the network. We introduce the shorthand $\vec{\boldsymbol{v}} = \langle \boldsymbol{v}_1, \dots, \boldsymbol{v}_T \rangle$ to denote the sequence of element feature vectors.

Next we map the sequence of element feature vectors into a single q-dimensional feature vector describing the entire sequence via a temporal encoding function ϕ,

$$\boldsymbol{u} = \phi(\vec{\boldsymbol{v}}) \in \mathbb{R}^q. \tag{2}$$

The vector $\boldsymbol{u}$ is now a fixed-length representation of the sequence, which can be used for classification.

Typical temporal encoding functions include sufficient statistics calculations or simple pooling operations, such as `max` or `avg`. However, the temporal encoding function can be much more sophisticated, such as the recently introduced `rank-pool` operator (Fernando et al., 2015). Unlike `max` and `avg` pooling operators, which can be expressed in closed-form, `rank-pool` requires an optimization problem to be solved in order to determine the representation. Mathematically, we have

$$\boldsymbol{u} \in \operatorname*{argmin}_{\boldsymbol{u}'} f(\vec{\boldsymbol{v}}, \boldsymbol{u}') \tag{3}$$

where $f(\cdot, \cdot)$ is some measure of how well a sequence is described by each representation and we seek the best representation. It is this type of temporal encoding function that we are interested in this paper.

Note also, that the optimization problem is very general and can include constraints on the solution in addition to just minimizing the objective f. Moreover, many standard pooling operations can be formulated in this way. For example, `avg` pooling can be written (somewhat offensively) as

$$\texttt{avg}(\vec{\boldsymbol{v}}) = \operatorname*{argmin}_{\boldsymbol{u}} \left\{ \frac{1}{2} \sum_{t=1}^{T} \|\boldsymbol{u} - \boldsymbol{v}_t\|^2 \right\}. \tag{4}$$

Importantly, the `rank-pool` operator encodes temporal dynamics of the sequence, which `max` and `avg` pooling operators do not. Specifically, the `rank-pool` operator attempts to capture the order of elements in the sequence by finding a vector $\boldsymbol{u}$ such that $\boldsymbol{u}^\mathsf{T}\boldsymbol{v}_a < \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_b$ for all $a < b$, i.e., the function $\boldsymbol{v} \mapsto \boldsymbol{u}^T\boldsymbol{v}$ honors the relative order of the elements in the sequence. This is achieved by regressing the element feature vector onto its index in the sequence and solved using regularized support vector regression (SVR) to give a point-wise ranking function (Liu, 2009). Concretely, we define $\texttt{rank-pool}(\vec{\boldsymbol{v}})$ as

$$\operatorname*{argmin}_{\boldsymbol{u}} \left\{ \frac{1}{2}\|\boldsymbol{u}\|^2 + \frac{C}{2} \sum_{t=1}^{T} \left[|t - \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t| - \epsilon \right]_{\geq 0}^2 \right\} \tag{5}$$

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

3.2. Prediction

With a fixed-length sequence descriptor $\boldsymbol{u} \in \mathbb{R}^q$ in hand the prediction task is to map $\boldsymbol{u}$ to one of the discrete class labels. Let $h_{\boldsymbol{\beta}}$ be a predictor parameterized by $\boldsymbol{\beta}$. We can summarize our classification pipeline of an arbitrary-length sequence $\vec{\boldsymbol{x}}$ to a label $y \in \mathcal{Y}$ as:

$$\vec{\boldsymbol{x}} = \langle \boldsymbol{x}_t \rangle \overset{\psi_{\boldsymbol{\theta}}}{\longmapsto} \langle \boldsymbol{v}_t \rangle \overset{\phi}{\longmapsto} \boldsymbol{u} \overset{h_{\boldsymbol{\beta}}}{\longmapsto} y \tag{6}$$

Typical predictors include (linear) support vector machines (SVM) and soft-max classifiers. For the latter—which we use in this work—the probability of a label y given the sequence $\vec{\boldsymbol{x}}$ can be written as

$$P(y \mid \vec{\boldsymbol{x}}) = \frac{\exp(\beta_y^\mathsf{T}\boldsymbol{u})}{\sum_{y'} \exp(\beta_{y'}^\mathsf{T}\boldsymbol{u})}. \tag{7}$$

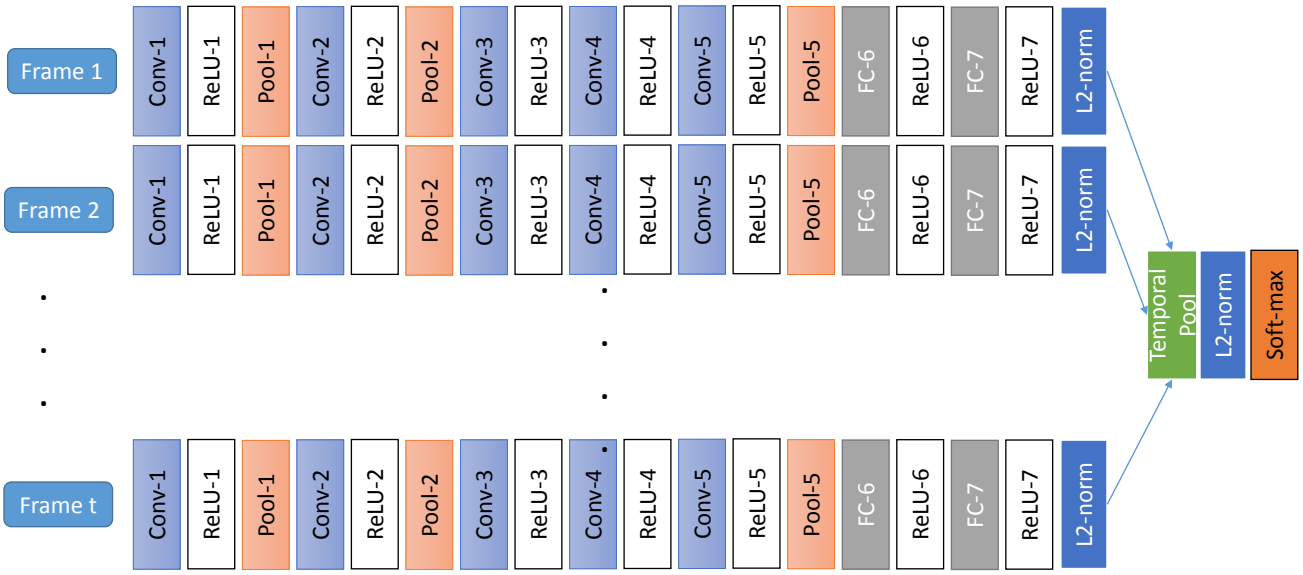

Figure 1. The CNN network architecture used in this paper for learning end-to-end temporal representations of videos. Network takes a sequence of frames from a video as inputs and feed forward till the end of the temporal pooling layer. At the temporal polling layer, the sequence of vectors are encoded 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 we do not introduce any new parameters to network architectures such as AlexNet or Caffe reference model. During back-propagation, the gradients are feed backwards through the rank-pooling operator to the rest of the CNN network.

Here $h_{\boldsymbol{\beta}}(\boldsymbol{u})$ represents the (discrete) probability distribution over all labels and $\boldsymbol{\beta} = \{\beta_y\}$ are the learned parameters of the model.

3.3. Learning

Given a dataset of sequence-label pairs, $\{(\vec{\boldsymbol{x}}^{(i)}, y^{(i)})\}_{i=1}^{n}$, our goal is to learn both the parameters of the classifier and representation of the elements in the sequence. 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

$$\Delta(y, h_{\boldsymbol{\beta}}(\boldsymbol{u})) = -\log P(y \mid \vec{\boldsymbol{x}}). \tag{8}$$

We jointly estimate the parameters of the feature function and prediction function by minimizing the regularized empirical risk. Our learning problem is

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{\theta},\boldsymbol{\beta}} & \sum_{i=1}^{n} \Delta\big(y^{(i)},\, h_{\boldsymbol{\beta}}(\boldsymbol{u}^{(i)})\big) + R(\boldsymbol{\theta},\boldsymbol{\beta}) \\ \text{subject to} & \boldsymbol{u}^{(i)} \in \text{argmin}_{\boldsymbol{u}}\, f(\vec{\boldsymbol{v}}^{(i)}, \boldsymbol{u}) \end{array} \tag{9}$$

where $R(\cdot,\cdot)$ is some regularization function, typically the ℓ_2-norm of the parameters, and $\boldsymbol{\theta}$ also appears in the definition of the $\vec{\boldsymbol{v}}^{(i)}$ by Eq. 1.

Eq. 9 is an instance of a bilevel optimization problem, which have recently been explored in the context of support vector machine (SVM) hyper-parameter learning (Klatzer & Pock, 2015). 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 from powerful CNN features, gradient-based techniques are most appropriate in allowing the fine-tuning of the CNN parameters.

When the temporal encoding function ϕ can be evaluated in closed-form (e.g., `max` or `avg`) we can substitute the constraints in Eq. 9 directly into the objective and use (sub-)gradient descent to solve for (locally or globally) optimal parameters.

Fortunately, when the lower level objective is twice differentiable we can compute the gradient of the argmin function as other authors have also observed (Ochs et al., 2015; Domke, 2012; Do et al., 2007) and the following lemmas show. It is then simply a matter of applying the chain rule to obtain the derivative of the loss function with respect to any parameter in the model. We begin by considering the scalar case and then extend to the vector case.

Lemma 1. : Let $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ be a continuous function with first and second derivatives. Let $g(x) =$

$\operatorname{argmin}_y f(x,y)$. Then

$$\frac{dg(x)}{dx} = -\frac{f_{XY}(x,g(x))}{f_{YY}(x,g(x))}$$

where $f_{XY} \doteq \frac{\partial^2 f}{\partial x \partial y}$ and $f_{YY} \doteq \frac{\partial^2 f}{\partial y^2}$.

Proof. Differentiating f with respect to y we have

$$\left.\frac{\partial f(x,y)}{\partial y}\right|_{y=g(x)} = 0 \quad (\text{since } g(x) = \operatorname*{argmin}_y f(x,y)) \tag{10}$$

$$\therefore \frac{d}{dx}\frac{\partial f(x,g(x))}{\partial y} = 0 \qquad (\text{differentiating lhs and rhs}) \tag{11}$$

But

$$\frac{d}{dx}\frac{\partial f(x,g(x))}{\partial y} = \frac{\partial^2 f(x,g(x))}{\partial x \partial y} + \frac{\partial^2 f(x,g(x))}{\partial y^2}\frac{dg(x)}{dx} \tag{12}$$

Equating to zero and rearranging gives the desired result

$$\frac{dg(x)}{dx} = -\left(\frac{\partial^2 f(x,g(x))}{\partial y^2}\right)^{-1}\frac{\partial^2 f(x,g(x))}{\partial x \partial y} \tag{13}$$

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

□

Lemma 2.: Let $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ be a continuous function with first and second derivatives. Let $\boldsymbol{g}(x) = \operatorname{argmin}_{\boldsymbol{y}\in\mathbb{R}^n} f(x,\boldsymbol{y})$. Then

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

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

Proof. Similar to Lemma 1, we have:

$$f_Y(x,\boldsymbol{g}(x)) \doteq \nabla_Y f(x,\boldsymbol{y})|_{\boldsymbol{y}=\boldsymbol{g}(x)} = 0 \tag{15}$$

$$\frac{d}{dx} f_Y(x,\boldsymbol{g}(x)) = 0 \tag{16}$$

$$\therefore f_{XY}(x,\boldsymbol{g}(x)) + f_{YY}(x,\boldsymbol{g}(x))\boldsymbol{g}'(x) = 0 \tag{17}$$

$$\frac{d}{dx}\boldsymbol{g}(x) = -f_{YY}(x,\boldsymbol{g}(x))^{-1} f_{XY}(x,\boldsymbol{g}(x)) \tag{18}$$

□

Interestingly, replacing argmin with argmax yields the same gradient, which follows from the proof above that only requires that $\boldsymbol{g}(x)$ be a stationary point.

Consider again the learning problem defined in Eq. 9. Using the result of Lemma 2 we can compute $\frac{d\boldsymbol{u}^{(i)}}{d\theta}$ for each training example and hence the gradient of the objective via the chain rule.[1] We then use stochastic gradient descent (SGD) to learn all parameters jointly.

Gradient of the `rank-pool` function: For completeness let us now derive the gradient of the `rank-pool` function. Assume a scalar parameter θ for the element feature function ψ (the extension to a vector of parameters can be derived elementwise). Let

$$f(\theta,\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|^2 + \frac{C}{2}\sum_{t=1}^{T}\Big[|t - \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t| - \epsilon\Big]^2_{\geq 0} \tag{19}$$

where $\boldsymbol{v}_t = \psi_\theta(\boldsymbol{x}_t)$. Let

$$e_t \doteq \begin{cases} \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t - t + \epsilon, & \text{if } t - \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t \geq \epsilon \\ \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t - t - \epsilon, & \text{if } \boldsymbol{u}^\mathsf{T}\boldsymbol{v}_t - t \geq \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Then by Lemma 2 we have

$$\frac{d}{d\theta}\operatorname*{argmin}_{\boldsymbol{u}} f(\theta,\boldsymbol{u}) = \left(I + C\sum_{e_t\neq 0}\boldsymbol{v}_t\boldsymbol{v}_t^\mathsf{T}\right)^{-1} \left(C\sum_{e_t\neq 0} e_t\psi'_\theta(\boldsymbol{x}_t) - \boldsymbol{u}^\mathsf{T}\psi'_\theta(\boldsymbol{x}_t)\boldsymbol{v}_t\right) \tag{21}$$

where with slight abuse of notation the $\boldsymbol{u}$ on the right-hand side is the optimal $\boldsymbol{u}$. Here $\psi'_\theta(\boldsymbol{x}_t)$ 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 but includes a zero-measure set of non-differentiable points. However, this does not cause any practical problems during optimization in our experience.

3.4. 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 Eq. 21 requires the inversion of the hessian f_{YY}. One solution is to use a diagonal approximation of the hessian, which is trivial to invert. Fortunately, 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 (Golub & Loan, 1996),

[1] The derivative with respect to $\boldsymbol{\beta}$, which only appears in the upper-level problem, is straightforward.

Table 1. Classification accuracies for action recognition on the ten-class UCF-sports dataset (Rodriguez et al., 2008).

METHOD	ACC. (%)
AVG-POOL + SVM	67
MAX-POOL + SVM	66
RANK-POOL + SVM	66
FRAME-LEVEL FINE-TUNING	70
FRAME-LEVEL FINE-TUNING + RANK-POOLING	73
AVG-POOL-CNN-END-TO-END	70
MAX-POOL-CNN-END-TO-END	71
RANK-POOL-CNN-END-TO-END (DIAG.)	**87**
RANK-POOL-CNN-END-TO-END	**87**
IMPROVED-TRAJECTORY +FISHER VECTORS+RANK-POOLING	87.2

Lemma 3.: Let $H = I + \sum_{i=1}^{n} u_i v_i^\mathsf{T} \in \mathbb{R}^{p\times p}$ be invertible. Define $H_0 = I$ and $H_m = H_{m-1} + u_m v_m^\mathsf{T}$ for $m = 1, \ldots, n$. Then

$$H_m^{-1} = H_{m-1}^{-1} - \frac{H_{m-1}^{-1} u_m v_m^\mathsf{T} H_{m-1}^{-1}}{1 + v_m^\mathsf{T} H_{m-1}^{-1} u_m} \tag{22}$$

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

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

Since each update in Eq. 22 can be performed in $O(p^2)$ the inverse of H can be computed in $O(np^2)$, which is acceptable for many applications. In the Section 4 we present experimental results and discuss the overall running time of our algorithm.

4. Experiments

We conduct experiments on action and activity recognition tasks in video using two real-world datasets, and compare our approach against some strong baseline methods.

4.1. Datasets and tasks

First, we use UCF-sports dataset (Rodriguez et al., 2008) for the task of action classification. The dataset 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. Video clips are grouped into ten action categories as shown in Figure 2. Classification performance is measured using mean per-class accuracy. We use provided train-test splits for training and testing.

Figure 2. Example frames from UCF-sports (top) and Hollywood2 (bottom) dataset from different action and activity classes.

Second, we use the Hollywood2 dataset (Laptev et al., 2008) for the task of activity recognition. The dataset has been constructed from 69 different Hollywood movies and includes 12 activity classes. It has 1,707 videos in total with a pre-defined split of 823 training videos and 884 test videos. Training and test videos are selected from different movies. The dataset represents a very challenging video collection. The length of the video clips varies from hundreds to several thousand frames. As is standard on this dataset, performance is measured by mean average precision (mAP) over all classes. Different activity classes are shown in Figure 2.

4.2. Baseline methods and implementation details

We compare our end-to-end training of the rank-pooling network against the following baseline methods.

avg pooling + svm: We extract FC7 feature activations from the pre-trained Caffe reference model (Jia et al., 2014) using MatConvNet (Vedaldi & Lenc, 2015) 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 (Fernando et al., 2015), and L2-normalize all frame features. Next, we apply the rank-pooling operator to obtain a video representation using publicly available code (Fernando et al., 2015). We use a linear SVM classifier applied

Table 2. Classification performance in average precision for activity recognition on the Hollywood2 dataset (Laptev et al., 2008).

CLASS	AVG+SVM	MAX+SVM	RANKPOOL+SVM	AVG+CNN	MAX+CNN	RANKPOOL+CNN
ANSWERPHONE	23.6	19.5	**35.3**	29.9	28.0	25.0
DRIVECAR	**60.9**	50.8	40.6	55.6	48.6	56.9
EAT	19.7	22.0	16.7	**27.8**	22.0	24.2
FIGHTPERSON	**45.6**	28.3	28.1	26.6	17.6	30.4
GETOUTCAR	39.5	29.2	28.1	48.9	43.8	**55.5**
HANDSHAKE	28.3	24.4	34.2	38.4	**40.0**	32.0
HUGPERSON	30.2	23.9	22.1	25.9	26.6	**33.2**
KISS	38.2	27.5	36.8	50.6	45.7	**54.2**
RUN	55.2	53.0	39.4	59.6	52.5	**61.0**
SITDOWN	30.0	28.8	32.1	30.6	30.0	**39.6**
SITUP	23.0	20.2	18.7	23.8	**26.4**	25.4
STANDUP	34.6	32.4	39.9	37.4	34.8	**49.9**
MAP	35.7	30.0	31.0	37.9	34.7	**40.6**

on the L2-normalized representation to classify each video.

frame-level fine-tuning: 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: 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.

state-of-the-art: Last, we benchmark our approach against combined state-of-the-art improved trajectory (Wang & Schmid, 2013) features (MBH, HOG, HOG) and Fisher vectors (Perronnin et al., 2010) with rank-pooling for temporal encoding (Fernando et al., 2015).

The first five baselines can all be viewed as variants of the CNN-base temporal pooling architecture of Figure 1. The differences being the pooling operation and whether end-to-end training is applied. The last baseline represents a state-of-the-art video classification pipeline.

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., (Simonyan & Zisserman, 2014; Tran et al., 2014)). 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 3). 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, we use the hinge-loss (as is common practice for this dataset).

4.3. Experimental results

Results for experiments on the UCF-sports dataset are reported in Table 1. 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 but that is beyond the scope of our study in this paper. 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 the same as when full gradient is used. This suggests that the diagnoal approximation is driving the parameters in a desireable 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 (Wang & Schmid, 2013) features (MBH, HOG, HOG) and Fisher vectors (Perronnin et al., 2010) with rank-pooling (Fernando et al., 2015) 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 2 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 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. To answer this question we ran two additional baselines. In the first we simply fine-tuned the CNN parameters to classify each video frame with the ground-truth activity and average the frame-level predictions at test time. In the second we apply rank-pooling to the fine-tuned frame features. On the test set we get 34.1 mAP and 36.3 mAP, respectively. Thus we observe gradual improvement from frame-level fine-tuning to fine-tuning with rank-pooling to end-to-end training (40.6 mAP).

For this dataset, one can obtain much higher accuracy using the state-of-the-art improved trajectory features (MBH, HOG, HOG) and Fisher vectors with rank-pooling (Fernando et al., 2015). Here Fernando et al. (2015) used several kinds of data augmentations (forward reverse rank pooling and mirrored videos data) to get to 70.0 mAP after combining all features. Individually, HOG, HOF, MBH,and TRJ features obtains 45.3, 59.8, 60.5, and 49.8 mAP, respectively. We improve CNN feature performance from 31.0 (vanilla rank pooling) to 40.6 mAP using end-to-end training, and note that here our objective is not to obtain state-of-the art but to show that rank-pooling operator of (Fernando et al., 2015) can be improved in the context of CNN-based video classification.

That said, our end-to-end trained CNN features can be combined with HOG+HOF+MBH features to boost performance. Here, without any data augmentation, we obtain 73.4 mAP whereas combining the vanilla CNN features combined with HOG+HOF+MBH we only get 71.4 mAP. These results indicate that our end-to-end training is useful even when combined with hand-crafted HOG, HOF and MBH features.

4.4. Diagonal approximation vs. full gradient

As we have seen, optimization using the diagonal approximation of the gradient obtains results that are on par with full gradient. Using the full gradient optimization is ten times slower than the approximate method, resulting in processing videos at 5 frames per second versus 50 frames per second (for the approximate method) during training on a Titan-X GPU. Even with the diagonal approximation end-to-end training is currently prohibitively slow on very large video collections such as the UCF101 dataset. Here we estimate training to take over 340 hours for 60 epochs over the 1M frames in the dataset. However, we are hopeful the next-generation GPUs, promised to be ten times faster than the Titan-X, will make our method tractable on very large video collections.

5. Conclusions

We propose an effective, clean, and principled temporal encoding method for convolutional neural network-based video sequence classification task. 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.

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. Last, our ability to update model parameters at 50 frames per second suggests that an agent, provided with appropriate supervision, could learn to recognize activities in real-time.

Acknowledgements

This research was supported by the Australian Research Council (ARC) through the Centre of Excellence for Robotic Vision (CE140100016).

References

Bard, Jonathan F. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Press, 1998.

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

Bousquet, Olivier and Elisseeff, André. Stability and generalization. *JMLR*, 2:499–526, 2002.

Dempe, S and Franke, S. On the solution of convex bilevel

optimization problems. *Computational Optimization and Applications*, pp. 1–19, 2015.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Do, Chuong B., Foo, Chuan-Sheng, and Ng, Andrew Y. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, 2007.

Domke, Justin. Generic methods for optimization-based modeling. In *AISTATS*, 2012.

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

Du, Yong, Wang, Wei, and Wang, Liang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015.

Fernando, Basura, Gavves, Efstratios, Oramas, Jose, Ghodrati, Amir, and Tuytelaars, Tinne. Modeling video evolution for action recognition. In *CVPR*, 2015.

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

Fernando, Basura, Gavves, Efstratios, Oramas, Jose, Ghodrati, Amir, and Tuytelaars, Tinne. Rank pooling for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016b.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jagannath. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Golub, Gene H. and Loan, Charles F. Van. *Matrix Computations*. Johns Hopkins University Press, 3 edition, 1996.

Ji, Shuiwang, Xu, Wei, Yang, Ming, and Yu, Kai. 3d convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pp. 675–678. ACM, 2014.

Karpathy, Andrej, Toderici, George, Shetty, Sanketh, Leung, Thomas, Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Klatzer, Teresa and Pock, Thomas. Continuous hyperparameter learning for support vector machines. In *Computer Vision Winter Workshop (CVWW)*, 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

Kunisch, Karl and Pock, Thomas. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983, 2013.

Laptev, Ivan, Marszalek, Marcin, Schmid, Cordelia, and Rozenfeld, Benjamin. Learning realistic human actions from movies. In *CVPR*, 2008.

Liu, Tie-Yan. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–331, 2009.

Ochs, P., Ranftl, R., Brox, T., and Pock, T. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, 2015.

Perronnin, Florent, Liu, Yan, Sánchez, Jorge, and Poirier, Hervé. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

Rodriguez, Mikel D, Ahmed, Javed, and Shah, Mubarak. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.

Simonyan, Karen and Zisserman, Andrew. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pp. 568–576, 2014.

Soomro, Khurram, Zamir, Amir Roshan, and Shah, Mubarak. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

Tran, Du, Bourdev, Lubomir, Fergus, Rob, Torresani, Lorenzo, and Paluri, Manohar. Learning spatiotemporal features with 3d convolutional networks. *arXiv preprint arXiv:1412.0767*, 2014.

Vedaldi, A. and Lenc, K. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

Wang, Heng and Schmid, Cordelia. Action recognition with improved trajectories. In *ICCV*, 2013.

Yue-Hei Ng, Joe, Hausknecht, Matthew, Vijayanarasimhan, Sudheendra, Vinyals, Oriol, Monga, Rajat, and Toderici, George. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

Zha, Shengxin, Luisier, Florian, Andrews, Walter, Srivastava, Nitish, and Salakhutdinov, Ruslan. Exploiting image-trained CNN architectures for unconstrained video classification. In *BMVC*, 2015.