This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Regularizing Long Short Term Memory with 3D Human-Skeleton Sequences for Action Recognition

Behrooz Mahasseni and Sinisa Todorovic
Oregon State University
Corvallis, OR 97331, USA
mahasseb@eecs.oregonstate.edu, sinisa@eecs.oregonstate.edu

Abstract

This paper argues that large-scale action recognition in video can be greatly improved by providing an additional modality in training data – namely, 3D human-skeleton sequences – aimed at complementing poorly represented or missing features of human actions in the training videos. For recognition, we use Long Short Term Memory (LSTM) grounded via a deep Convolutional Neural Network (CNN) onto the video. Training of LSTM is regularized using the output of another encoder LSTM (eLSTM) grounded on 3D human-skeleton training data. For such regularized training of LSTM, we modify the standard backpropagation through time (BPTT) in order to address the well-known issues with gradient descent in constraint optimization. Our evaluation on three benchmark datasets – Sports-1M, HMDB-51, and UCF101 – shows accuracy improvements from 1.7% *up to* 14.8% *relative to the state of the art.*

1. Introduction

This paper is about classifying videos of human actions. We focus on domains that present challenges along two axes. First, we consider a large set of action classes (e.g., the Sports-1M dataset with 487 action classes [15]) which may have very subtle inter-class differences and large intra-class variations (e.g., Sports-1M contains 6 different types of bowling, 7 different types of American football and 23 types of billiards). The actions may be performed by individuals or groups of people (e.g., skateboarding vs. marathon), and may be defined by a particular object of interaction (e.g., bull-riding vs. horseback riding). Second, our videos are captured in uncontrolled environments, where the actions are viewed from various camera viewpoints and distances, and under partial occlusion.

Recent work uses Convolutional Neural Networks (CNNs) to address the above challenges [2, 13, 31, 15, 41, 4, 35, 26]. However, despite the ongoing research efforts to: (a) Increase the amount of training data [15], (b) Fuse hand-designed and deep-convolutional features [35, 26, 41], and (c) Combine CNNs with either graphical models [31, 11, 39], or recurrent neural networks [4, 39] for capturing complex dynamics of human actions, we observe that their classification accuracy is still markedly below the counterpart performance on large-scale image classification. This motivates us to seek a novel deep architecture, leveraging some of the promising directions in (a)–(c).

Our key idea is to augment the training set of videos with additional data coming from another modality. This has the potential to facilitate capturing important features of human actions poorly represented in the training videos, or even provide complementary information missing in the videos. Specifically, in training, we use 3D human-skeleton sequences of *a few* human actions to regularize learning of our deep representation of all action classes. This regularization rests on our hypothesis that since videos and skeleton sequences are about human motions their respective feature representations should be similar. The skeleton sequences, being view-independent and devoid of background clutter, are expected to facilitate capturing important motion patterns of human-body joints in 3D space. This, in turn, is expected to regularize, and thus improve our deep learning from videos.

It is worth noting that the additional modality that we use in training *does not* provide examples of most action classes from our video domain. Rather, available 3D human-skeleton sequences form a small-size training dataset that is insufficient for robust deep learning. Nevertheless, in this paper, we show that accounting for this additional modality greatly improves our performance relative to the case when only videos are used in training.

As illustrated in Fig. 1, for action recognition, we use Long Short Term Memory (LSTM) grounded via a deep Convolutional Neural Network (DCNN) onto the video. LSTM is modified to have an additional representational layer at the top, aimed at extracting deep-learned feature

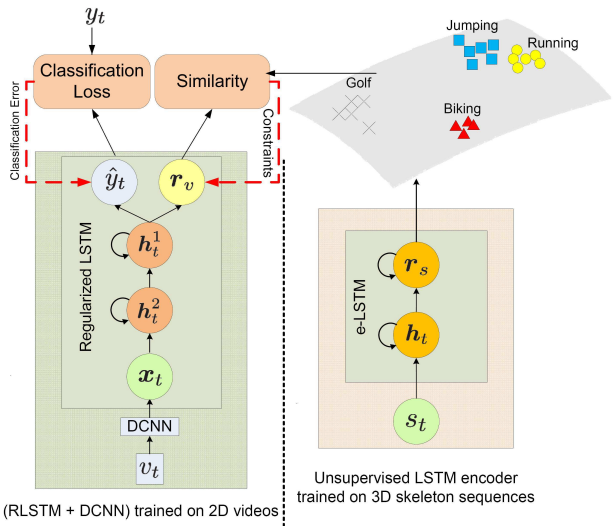

Figure 1: Our novel deep architecture: the LSTM on the left is trained on videos under weak supervision, and the encoder LSTM (eLSTM) on the right is trained in unsupervised manner on 3D human-skeleton sequences. v_t and s_t denote the input video frame and skeleton data at time t. $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ are the output features of LSTM and eLSTM. y and $\hat{y}$ are the ground-truth and predicted class labels. $\boldsymbol{h}_t$, $\boldsymbol{h}_t^1$, and $\boldsymbol{h}_t^2$ are the hidden layers in the two respective LSTMs. $\boldsymbol{x}_t$ is the output feature of DCNN's FC_7 layer. Euclidean distances between corresponding features $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ are jointly used with the prediction loss between y and $\hat{y}$ for a regularized learning of the LSTM on videos.

$\boldsymbol{r}_v$ from the entire video. In training, we regularize learning of LSTM such that its output $\boldsymbol{r}_v$ is similar to features $\boldsymbol{r}_s$ produced by another encoder LSTM (eLSTM) grounded onto 3D human-skeleton sequences. The sequences record 3D locations of 18 joints of a human body while the person performs certain actions, such as those in the Carnegie-Mellon Mocap dataset [1] and the HDM05 Mocap dataset [22]. eLSTM is learned in an unsupervised manner by minimizing the data reconstruction error. Note that a hypothetical supervised learning of an LSTM on skeleton data would not be possible in our setting, since we have access to a small dataset representing only a small fraction (or none) of action classes from the video domain. During test time, we do not use any detections of human joints and their trajectories, but classify a new video only based on raw pixels taken as input to the LSTM+DCNN.

Our main contribution represents a novel regularization of LSTM learning. Unlike the standard regularization techniques, such as drop out or weight decay [12, 4, 40], we define a set of constraints aimed at reducing the Euclidean distances between top-layer features of LSTM trained on videos and corresponding output features of eLSTM. We use these constraints to regularize and thus extend the standard backpropagation through time (BPTT) algorithm [4]. BPTT back-propagates a class-prediction loss for updating LSTM parameters via stochastic gradient descent. We additionally back-propagate the above constraints between corresponding features. This requires modifying the standard (unconstrained) gradient descent to an algorithm that accounts for constraints. To this end, we use the hybrid steepest descent [9].

In this paper, we consider several formulations of regularizing LSTM learning corresponding to the cases when ground-truth class labels are available for the skeleton sequences, and when this ground truth is not available.

We present experimental evaluation on three benchmark datasets, including Sports-1M [15], HMDB-51 [19], and UCF101 [28]. We report the performance improvement ranging from 1.7% to 14.8% relative to the state of the art.

In the following, Sec. 2 reviews related work, Sec. 3 specifies LSTM, Sec. 4 formulates our novel deep architecture, and Sec. 5 presents our results.

2. Closely Related Work

This section reviews related work on: combining CNN and LSTM, encoder LSTM, using skeleton data for action classification, and multimodal deep learning in vision.

LSTM+DCNN. Frame-level DCNN features have been used as input to LSTM for action classification [4]. This architecture has been extended with additional layers for convolutional temporal pooling [39]. The main advantages include that these architectures are deeply compositional in both space and time, and that they are capable of directly handling variable-length inputs (e.g., video frames) and capturing long-range, complex temporal dynamics. They are learned with backpropagation through time (BPTT). Our key difference is in the definition of LSTM output layer. Our output layer includes the standard softmax layer for classification and the additional representational layer for mapping input video sequences to a feature space. It is the construction of this new feature space that allows us to regularize LSTM learning. Another difference is that we replace the standard stochastic gradient descent in BPTT, with an algorithm that accounts for constraints in optimization.

Encoder LSTM. LSTM has been used to encode an input data sequence to a fixed length vector, which, in turn, is decoded to predict the next unobserved data [29]. However, this recurrent autoencoder-decoder paradigm has not yet demonstrated competitive performance on action classification in videos. Our main difference is that we use the encoder LSTM to generate a feature manifold for regularizing a supervised learning of another LSTM.

Action classification using skeleton data has a long-track record [10, 20, 33, 34, 37, 38, 5]. However, at test time, these approaches require either 3D locations of human

joints, or detection of human joints in videos. In contrast, at test time, we just use pixels as input, and neither detect human joints nor need their 3D locations.

Multimodal learning. Recent work uses text data as an additional modality to improve image classification [23, 8, 7, 27, 18, 21]. For example, a combination of DCNN and LSTM has been used for multimodal mapping of finer object-level concepts in images to phrases [18]. Closely related work introduces a semi-supervised embedding in deep architectures [36]. Due to the fundamental differences in our problem statements, we cannot use these approaches. More importantly, instead of a heuristic combination of classification loss and the embedding loss, we use a well-defined set of constraints to explicitly exploit the information contained in the embedding space.

Another difference is that object classes of text data are in one-to-one correspondence with object classes appearing in images (or it is assumed that the image and text domains have a large overlap of object classes). In contrast, our 3D skeleton sequences represent only a few action classes from the video domain. Similar to [8, 27], we do not use the other modality at test time. For multimodal training, these approaches use, for example, the Euclidean distance [27], modified hinge loss [8], parameter transfer and regression loss [7], or pairwise ranking loss [18]. Instead, we minimize the cross entropy loss associated with the softmax layer of LSTM, subject to the feature similarity constraints in our regularization. Importantly, in Sec. 4.3, we provide convergence guarantees for our regularized learning of LSTM.

3. A Brief Review of LSTM

A major building block of our novel architecture is LSTM [12, 29], depicted in Fig. 2. LSTM is a recurrent neural network as briefly reviewed below.

The LSTM's recurrent unit memorizes previous input information in a memory cell, c_t, indexed by time t. The output is hidden variable h_t. Three gates control the updates of c_t and h_t: 'input' i_t, 'output' o_t, and 'forget' f_t. In (1), we summarize their update equations, where symbol $\odot$ denotes the element-wise product, and W are LSTM parameters.

$$\begin{aligned} i_t =& \sigma(W_{xi}\boldsymbol{x}_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \\ o_t =& \sigma(W_{xo}\boldsymbol{x}_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o), \\ f_t =& \sigma(W_{xf}\boldsymbol{x}_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \\ c_t =& f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}\boldsymbol{x}_t + W_{hc}h_{t-1} + b_c), \\ h_t =& o_t \odot \tanh(c_t). \end{aligned} \tag{1}$$

From (1), the input gate, i_t, regulates the updates of c_t, based on inputs $\boldsymbol{x}_t$ and previous values of h and c. The 'output gate', o_t, controls if h_t should be updated given c_t. The forget gate, f_t, resets the memory to its initial value. The LSTM parameters $W_{\text{LSTM}} = \{W_{xi}, W_{xo}, W_{xf}, W_{ci}, W_{co}, W_{cf}, W_{hi}, W_{ho}, W_{hf}\}$ are jointly learned using BPTT.

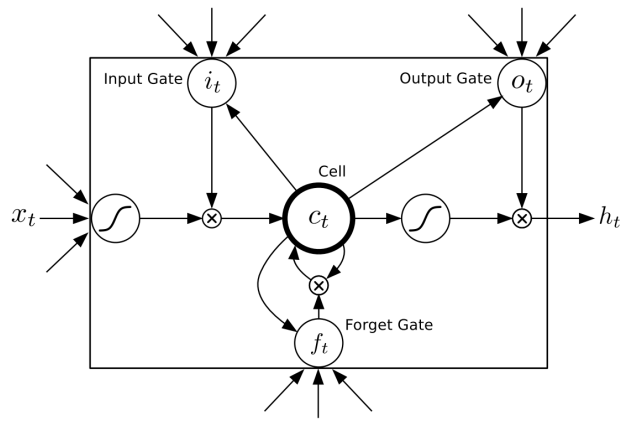

Figure 2: LSTM unit [12, 29].

The LSTM unit preserves error derivatives of the deep unfolded network. This has been shown to avoid the well-known vanishing gradient problem, and allows LSTM to capture long-range dependencies in the input data sequence.

4. Regularizing LSTM for Action Recognition

As mentioned in Sec. 1, our novel architecture consists of eLSTM for learning a feature representation of 3D human-skeleton sequences, and a stacked DCNN+LSTM for classifying videos. Below, we describe these two components.

eLSTM. Fig. 3 shows the time-unfolded encoder and decoder parts of eLSTM. The goal of eLSTM is to encode the input skeleton sequence, $s = \{s_t : t = 1, 2, \dots\}$, consisting of 3D locations of human joints. The sequences may have variable lengths in time. eLSTM observes the entire skeleton sequence s, and encodes it to a feature vector $\boldsymbol{r}_s$. The set of encoded representations $\{\boldsymbol{r}_s\}$ are assumed to form a manifold $\mathcal{M}_s$ of skeleton sequences. To learn the encoder, a decoder LSTM tries to reconstruct the normalized input 3D data in the reverse order. The reconstruction error is then estimated in terms of the mean-squared error in the normalized 3D coordinates, and used to jointly learn both the encoder and decoder LSTMs. The reversed output reconstruction benefits from low range correlations which makes the optimization problem easier. The input to the encoder at each time step t, is the output of the decoder at time step $t-1$, i.e. s_{t-1}. An alternative architecture is to learn an encoder LSTM to predict the next skeleton frame. To avoid over-fitting, we use the standard drop-out regularization for eLSTM [40].

DCNN+LSTM. As shown in Fig. 1, for classifying human actions in videos we use a stacked architecture of a frame-level DCNN and a two-layer LSTM. DCNNs have been demonstrated as capable of learning to extract good image descriptors for classification [32, 41, 16]. For DCNN,

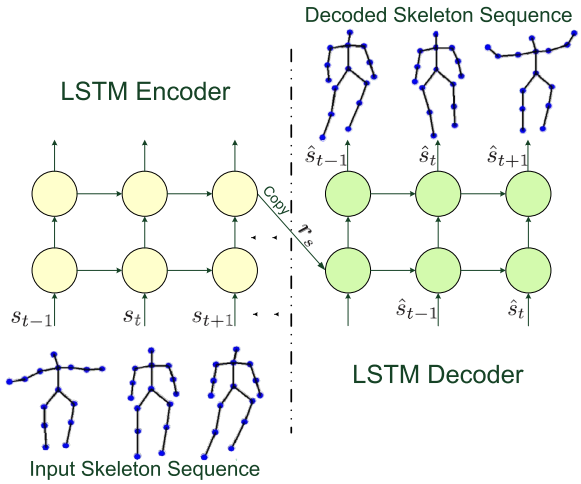

Figure 3: The time-unfolded visualization of the encoder LSTM (left) and decoder LSTM (right) for learning a feature representation from input 3D human-skeleton sequences. The encoder LSTM observes the entire skeleton sequence and encodes it to a fixed-length representation. The decoder LSTM tries to reconstruct 3D locations of human joints in the reverse order of the input sequence. The reconstruction error is then used to jointly learn both the encoder and decoder LSTMs.

we use the same network architecture initially trained on the ImageNet dataset as in [30]. The only difference is in the number of output units of the softmax layer at the top of DCNN, since we address different numbers of action classes. Note that we later fine-tune DCNN parameters together with LSTM ones in our regularized learning.

Our LSTM differs from the model used in [4] in the top output layer. The output layer of our LSTM extends the standard fully connected softmax layer for classification with an additional representation layer. This representation layer is aimed at mapping input video sequences, $v = \{v_t : t = 1, 2, \dots\}$, with variable lengths in time, to fixed-length vectors $\boldsymbol{r}_v$. The size of this representation layer is set to be equal to that of the output layer of eLSTM. Thus, the vectors $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ have the same size.

Our goal of learning DCNN+LSTM parameters, Θ, is to minimize the classification loss, $L(\Theta)$, subject to constraints g between vectors $\boldsymbol{r}_v$ and corresponding vectors $\boldsymbol{r}_s$ in manifold $\mathcal{M}_s$. Thus, for all training videos $v \in D_v$, we formulate the regularized learning of DCNN+LSTM as

$$\begin{aligned} &\min_{\Theta} L(\Theta) \\ &\text{s.t. } \forall v \in D_v \ \ , g(\boldsymbol{r}_v, \mathcal{M}_s) \leq 0, \end{aligned} \tag{2}$$

where $L(\Theta) = \sum_{v \in D_v} l(v, \Theta)$ is the cross entropy loss associated with the softmax layer of LSTM, and g is a constraint based on the distance between $\boldsymbol{r}_v$ and $\mathcal{M}_s$. g can be defined in different ways. We only require that the constraints are differentiable functions. In the following, we define two distinct constraints that give the best results in our experiments.

4.1. Class Independent Regularization

For class independent regularization of learning DCNN+LSTM parameters, the constraint $g = g_1$ in (2) is specified to ensure that, for every video $v \in D_v$, $\boldsymbol{r}_v$ is sufficiently similar to the mapped vectors $\boldsymbol{r}_s \in \mathcal{M}_s$. This is achieved by defining an upper-bound for the average distance between $\boldsymbol{r}_v$ and all $\boldsymbol{r}_s \in \mathcal{M}_s$ as in (3)

$$g_1(\boldsymbol{r}_v, \mathcal{M}_s) = \frac{1}{n} \sum_{\boldsymbol{r}_s \in \mathcal{M}_s} \|\boldsymbol{r}_v - \boldsymbol{r}_s\|_2^2 - \alpha, \tag{3}$$

where $\alpha > 0$ is an input parameter, and n is the number of training skeleton sequences. This type of regularization is suitable for cases when training skeleton sequences do not represent the same action classes as training videos or represent a very small subset of action classes.

4.2. Class Specific Regularization

For class specific regularization of learning DCNN+LSTM parameters, the constraint $g = g_2$ in (2) is specified to take into account action class labels of training skeleton sequences, when available. This type of learning regularization is suitable for cases when some action classes represented by training skeleton sequences do "overlap" with certain action classes in training videos. The definition of class equivalence between the two modalities can be easily provided along with ground-truth annotations.

We expect that $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ should be more similar if video v and skeleton sequence s share the same class label, $l_v = l_s$, than $\boldsymbol{r}_v$ and $\boldsymbol{r}_{s'}$ of skeleton sequences s' with different class labels, $l_v \neq l_{s'}$. This is defined in (4)

$$g_2(\boldsymbol{r}_v, \mathcal{M}_s) = \frac{1}{n_=} \sum_{\substack{\boldsymbol{r}_s \in \mathcal{M}_s \\ l_v = l_s}} \|\boldsymbol{r}_v - \boldsymbol{r}_s\|_2^2 - \frac{1}{n_\neq} \sum_{\substack{\boldsymbol{r}_{s'} \in \mathcal{M}_s \\ l_v \neq l_{s'}}} \|\boldsymbol{r}_v - \boldsymbol{r}_{s'}\|_2^2. \tag{4}$$

where $n_=$ and $n_\neq$ are the numbers of training skeleton sequences that have $l_v = l_s$ and $l_v \neq l_s$. The constraint g_2 defined in (4) ensures that the average Euclidean distance between $\boldsymbol{r}_v$ and $\boldsymbol{r}_s \in \mathcal{M}_s$ for skeleton sequences s of the same action label is less than the average Euclidean distance between $\boldsymbol{r}_v$ and $\boldsymbol{r}_{s'}$ for skeleton sequences s' of different action labels.

4.3. Hybrid Steepest Descent

We jointly learn parameters, $\Theta = W_{\text{LSTM}} \cup W_{\text{DCNN}}$, by modifying the standard backpropagation and applying a

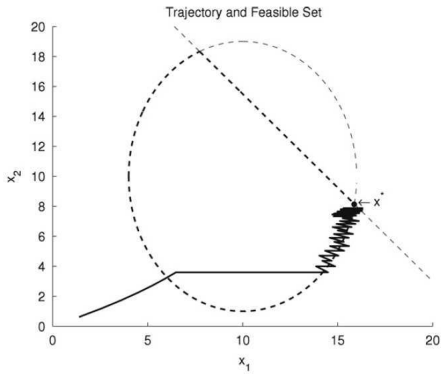

Figure 4: Simulation of the hybrid steepest descent algorithm for a simple optimization problem presented in [9]. In this example $\Theta = x_1, x_2$, $L(\Theta) = -x_1$, $g_1 = \frac{(x_1-10)^2}{36} + \frac{(x_2-10)^2}{81} - 1$, and $g_2 = \frac{10}{8}x_1 + x_2 - 28$. The dark hashed lines show the boundary of the feasible set.

stochastic gradient descent algorithm which accounts for the aforementioned constraints. One standard approach to solve a constrained convex optimization problem is to use Lagrange multipliers and fuse the constraints with the original objective into a new unconstrained optimization problem. Unfortunately, as demonstrated in [24], gradient descent poorly works with Lagrange multipliers, mainly because they introduce saddle points in the new objective. Therefore, we resort to an alternative approach called hybrid steepest descent [9] for solving our constrained convex optimization, as described below.

Fig. 4 shows a simulation of hybrid steepest descent for a simple convex optimization problem. At each iteration, the algorithm checks if the current solution – i.e., the current Θ parameters in our case – satisfies all constraints. If so, Θ is updated according to the gradient of the original objective function – i.e., in our case, the cross entropy loss $L(\Theta)$ – without considering the constraints. If any constraint is violated by the current solution – i.e in our case, $g(\boldsymbol{r}_v, \mathcal{M}_s) \leq 0$ constraint is not satisfied for $\boldsymbol{r}_v$ computed by DCNN+LSTM with the current Θ parameters –, we update Θ according to the gradient of the violated constraints. The proof of correctness, asymptotic stability, and convergence of this algorithm is presented in [9]. Note that there is no guarantee that the final value of the parameters Θ satisfies all constraints. In our implementation we keep track of the top 5 best solutions based which have the minimum number of violated constraints.

We use hybrid steepest descent to modify BPTT for regularized learning of DCNN+LSTM. Note that the updates of the recurrent units only depend on the particular value of the the back-propagated gradient. Once this gradient is computed, we use it in the same way as in the standard BPTT. Thus, as summarized in Alg. 1, our modification of BPTT amounts to alternating the specification of the error gradient at the output layer according to the above rules of hybrid steepest descent. In Alg. 1, we use $g > 0$ to denote a constraint violation, and η_t is the time dependent learning rate.

Algorithm 1: Regularized Learning of LSTM

Input: Training videos D_v and $\mathcal{M}_s$
Output: LSTM parameters Θ

% note that during training we assume sequences in D_v are of the same length
repeat
  for *every* $v \in D_v$ **do**
    forward-pass v through unfolded LSTM
    if *all constraints satisfied* **then**
      $\Theta_t \xleftarrow{BPTT} \Theta_{t-1} - \eta_t \nabla L(\Theta)$;
      % This updates Θ by back-propagating the gradient of the cross entropy loss.
    end
    else
      $\Theta_t \xleftarrow{BPTT} \Theta_{t-1} - \eta_t \sum_{g>0} \nabla g(\Theta)$;
      % This updates Θ by back-propagating the sum of gradients of the violated constraints.
    end
  end
until *until convergence*;

5. Results

For evaluation, we use the Sports-1M [15], HMDB-51 [19], and UCF-101 [28] datasets. Sports-1M consists of more than 1 million videos from Youtube, annotated with 487 action classes. There are on average 3000 videos for each action class, where the average video length is 5 minutes 36 seconds. Considering the large number of action classes, long duration of the videos, and large variety of camera motions, Sports-1M is currently acknowledged as one of the most challenging benchmarks for action recognition in the wild. HMDB-51 consists of 6849 videos with 51 action labels. Each action class has at least 100 videos with an average video length of 3.1 seconds. UCF-101 consists of 13,320 videos of 101 action classes with average video length of 7.2 seconds. We follow [39], and report our average accuracy on the given three dataset splits.

We use human skeleton sequences of the Carnegie-Mellon Mocap [1] and HDM05 [22] datasets to train eLSTM. HDM05 consists of 2337 sequences with 130 action classes performed by 5 subjects. These sequences record the 3D locations of 31 human joints at each frame. Carnegie-Mellon Mocap consists of 2605 sequences with 23 action classes. These sequences record the 3D locations of 41 human joints at each frame. For consistency, we use

only 18 human-body joints (head, lower back, upper back, upper neck, right/left clavicle, hand, humerus, radius, femur, tibia, foot) of the skeleton sequences, and resolve name conflicts and duplicates in these two datasets.

eLSTM: An LSTM with two hidden layers is used for the encoder and decoder. Input and output of the encoder and decoder LSTMs are $54 = 18 \times 3$ dimensional vectors of 3D human body joint positions. We normalize input data in the range of [0, 1]. We empirically verified that eLSTM with 512 and 1024 hidden units in the first and second recurrent layer of the encoder LSTM, and the same number of hidden units in a reverse order for the decoder LSTM results in the smallest reconstruction error. The model is trained on 16 frame sequences. Since the training is unsupervised, both Carnegie-Mellon Mocap and HDM05 datasets are used in training phase. It takes 16-19 hours to converge for about 3160 minutes of skeleton sequences. For defining g_2 constraints, we use the class labels defined in HDM05.

DCNN: We use GoogLeNet [30] trained on ImageNet [25] as DCNN in our approach. This DCNN is fine-tuned on a set of randomly sampled video frames. The output layer is modified for the fine-tuning based on the number of action classes. On average (150-200) frames are sampled from each video. The second to the last fully connected layer (FC_7) is used as the frame descriptor input to LSTM. Note that, later, DCNN parameters are fine-tuned again jointly with LSTM training.

Our Regularized LSTM (RLSTM): Our RLSTM for action recognition contains two hidden layers of 2048 and 1024 hidden units, respectively. The number of output units for classification is 487 for Sports-1M, 51 for HMDB-51, and 101 for UCF101. The number of output units for representation is 512, which is equal to the number of hidden units in the second recurrent layer of eLSTM. Similar specifications are used in [4, 29]. Fixed length sequences of 16 frames are used in training. We find that a random initialization of RLSTM converges to a poor local optimum. To overcome this, we train a non regularized LSTM using one-tenth of the training instances in Sports-1M. We use weights of this learned model to initialize weights of RLSTM. Weights of the representation layer are initialized randomly between [-0.1, 0.1]. Similar to [4, 29], we estimate an average prediction of 16 block frames with a stride of 8 in inference. The linear weighting presented in [39] is used to combine predictions from each block for the entire video.

Implementation: We use Caffe [14] and a modified version of the RNN library in Torch [3] in our experiments. All experiments are performed on an Intel quad core-i7 CPU and 16GB RAM PC with two Tesla-K80 Nvidia cards.

Baselines: We conduct a comparison with several baselines in order to evaluate effectiveness of different constraints in our regularized learning of RLSTM. These baselines include the following: 1) **DCNN**: This is a 'single-frame' based action recognition evaluated in [15] – a video is represented by a single frame and classification is perform only based on the content in that frame, 2) **LSTM**: This is a (DCNN+LSTM) learned without any regularization and constraints, similar to the approach of [4] with only difference in the number of hidden units (due to a different number of classes considered).

Method	UCF101	HMDB-51
Single-Frame [15]	64.9	41.2
LSTM	75.2	43.1
RLSTM-$g1$	78.3	49.3
RLSTM-$g2$	81.5	51.4
RLSTM-$g3$	86.9	55.3

Table 1: Average classification accuracy of regularized LSTM models and baselines on UCF101 and HMDB-51. Our approach outperform the LSTM baseline by $11.7\% - 12.2\%$

Variations of our approach: Based on the constraints, defined in Sec. 4.1 and 4.2, for regularizing learning of RLSTM, we define the following three variations of our approach: 1) **RLSTM**-$g1$: uses class independent constraints $g1$ to regularize the learning, 2) **RLSTM-**$g2$: uses class specific constraints $g2$ to regularize the learning , 3) **RLSTM-**$g3$: uses $g1 \cup g2$ to regularize the learning.

Table 1 shows our average classification accuracy on HMDB-51 and UCF101. All variations of our method improved the accuracy of the baseline LSTM (3.1% to 12.2%). RLSTM-$g2$ achieves a better accuracy compared to RLSTM-$g1$. This strongly supports the hypothesis that deep-learned features of vides and skeleton sequences should be similar, and that our regularization should improve action recognition. Because the 3D human skeleton datasets and UCF101/HMDB-51 share a few common action classes, RLSTM-$g3$ which combines the class independent and class specific constraints outperforms RLSTM-$g2$.

Comparison of our method with the state of the art deep learning based approches is presented in Tab. 2. We can see that RLSTM-$g3$ outperforms variations of [15, 29, 4, 32, 26] which only use raw frames by $1.7\% - 22\%$. Higher accuracy is reported in [41, 39, 26, 35] on UCF101 for (raw pixels + optical flow) input. In comparison with their accuracy of $88.6\% - 90.3\%$, we achieve a comparable performance of 86.9% accuracy on UCF101 *by using only pixels of video frames*.

Hit@k values are a standard evaluation for large-scale datasets. A test instance is considered correctly labeled if the ground truth label is among the top k predictions. Similar to [39, 16] we use Hit@1 and Hit@5 values to report accuracy on Sports-1M. Table 3 shows the classification ac-

Method	UCF101	HMDB-51
[15]	65.4	-
[29]	75.8	44.1
[4]	71.12	-
[26]	72.8	40.5
[41]	79.34	-
[32]	85.2	-
RLSTM-$g3$	86.9	55.3

Table 2: Average classification accuracy of RLSTM-$g3$ and the state of the art on UCF101 and HMDB-51. Our approach outperform the best result in state of the art by $1.7 - 11.2\%$

Method	Hit@1	Hit@5
Single-Frame [15]	59.3	77.7
LSTM	71.3	89.9
[15]	60.9	80.2
[39]	72.1	90.6
[32]	61.1	85.2
RLSTM-$g1$	73.4	91.3
RLSTM-$g2$	62.2	85.3
RLSTM-$g3$	75.9	91.7

Table 3: Average classification accuracy of regularized LSTM models, baselines, and the state of the art on Sports-1M. Our approach outperform the best result in state of the art by 2.5% and the LSTM baseline by 4.6%.

curacy of the baselines, different RLSTM models, and the state of the art on Sports-1M. One interesting result is that RLSTM-g_1 outperforms RLSTM-g_2. We believe that this is because of a poor overlap of the large number of action classes in Sports-1M with the set of action classes in the skeleton data obtained from HDM05 and CMU Mocap. Our best approach improves the classification accuracy on Hit@1 by $2.5\% - 16.6\%$. Also the baseline non-regularized LSTM yields better accuracy than the temporal pooling approaches of [15, 32]. We believe that this is mainly because the baseline LSTM has access to the longer video frames.

To verify the effectiveness of hybrid-gradient descent used in our regularized learning, we also train our DCNN+LSTM using AdaGrad [6] and Adam [17]. These two alternative algorithms are aimed at minimizing the standard weighted sum of the classification and representation loss. The comparison is shown in Table 4, where RLSTM denotes our approach to regularized training with hybrid-gradient descent, and LSTM($\cdot$) denotes our approach to regularized training with AdaGrad or Adam.

Method	UCF101	HMDB-51
LSTM(adm)-$g1$	75.7	44.6
LSTM(adm)-$g2$	79.20	49.8
LSTM(adm)-$g3$	81.8	51.3
LSTM(adg)-$g1$	74.9	44.3
LSTM(adg)-$g2$	81.6	49.5
LSTM(adg)-$g3$	80.7	48.6
RLSTM-$g1$	78.3	49.3
RLSTM-$g2$	81.5	51.4
RLSTM-$g3$	86.9	55.3

Table 4: Average classification accuracy of our approach on UCF101 and HMDB-51, when the regularized training is conducted with AdaGrad [6] or Adam [17] – denoted as LSTM(adg) and LSTM(adm) – or hybrid-gradient descent – denoted as RLSTM.

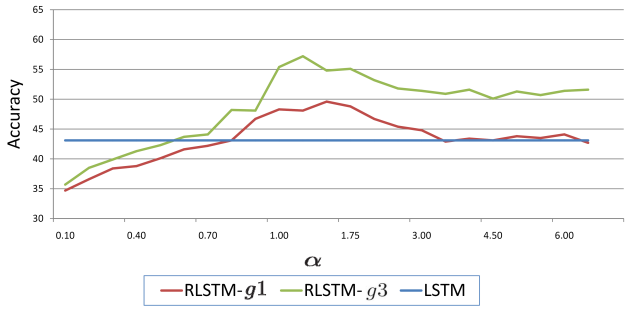

Figure 5: Average classification accuracy of RLSTM-$g1$ and RLSTM-$g3$ on HMDB-51 for different α values, where α is the input parameter that controls the regularized learning of RLSTM-$g1$ and RLSTM-$g3$. Small values of α enforce stronger regularization that the feature outputs of RLSTM and eLSTM are highly similar.

The regularized learning of RLSTM-$g1$ and RLSTM-$g3$ is controlled by the α parameter, specified in (3). Fig. 5 shows how our accuracy changes for different values of α on HMDB-51. We can see that for small values of α the accuracy is very low. We observe that for these values the learning algorithm does not converge. The algorithm alternates gradient updates with respect to classification error and violated constraints. This is because the optimization problem becomes almost infeasible for small values of α. The accuracy gradually increases for larger values of α, but again decreases when α becomes sufficiently large. The accuracy of RLSTM-$g1$ reaches that of the standard LSTM for large values of α, and remains nearly the same. This is because sufficiently large values of α yield an unconstrained optimization, i.e., non-regularized learning of LSTM.

6. Conclusion

We have proposed a novel deep architecture for large-scale action recognition. The main contribution of our work is to use 3D human-skeleton sequences to regularize the learning of LSTM, which is grounded via DCNN onto the video for action recognition. We have modified the backpropagation through time algorithm in order to account for constraints in our regularized joint learning of LSTM+DCNN. Our experimental results demonstrate that the skeleton sequences could successfully constrain the learning of LSTM+DCNN leading to an improved performance relative to the case when LSTM is trained only on videos. Specifically, on Sports-1M, UCF101, and HMDB-51 our accuracy improves by $2.5\% - 16.6\%$, $1.7 - 21.5\%$, and $11.2 - 14.8\%$, respectively.

Acknowledgement

This work was supported in part by grant NSF RI 1302700.

References

[1] Carnegie-Mellon Mocap. http://mocap.cs.cmu.edu. Accessed: 2003. 2, 5

[2] C. Bo. Deep learning of invariant spatio-temporal features from video. 2010. 1

[3] R. Collobert, S. Bengio, and J. Marithoz. Torch: A modular machine learning software library, 2002. 6

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015. 1, 2, 4, 6, 7

[5] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, pages 1110–1118, 2015. 2

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011. 7

[7] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, pages 2584–2591. IEEE, 2013. 3

[8] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 3

[9] M. Gerard, B. De Schutter, and M. Verhaegen. A hybrid steepest descent method for constrained convex optimization. *Automatica*, 45(2):525–531, 2009. 2, 5

[10] D. Gong and G. Medioni. Dynamic manifold warping for view invariant action recognition. In *ICCV*, pages 571–578. IEEE, 2011. 2

[11] H. Hajimirsadeghi and G. Mori. Learning ensembles of potential functions for structured prediction with latent variables. In *ICCV*, 2015. 1

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 3

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

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1, 2, 5, 6, 7

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732. IEEE, 2014. 3, 6

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[18] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 3

[19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2, 5

[20] J. Luo, W. Wang, and H. Qi. Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. In *ICCV*, pages 1809–1816. IEEE, 2013. 2

[21] J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014. 3

[22] M. Mller, T. Rder, M. Clausen, B. Eberhardt, B. Krger, and A. Weber. Documentation mocap database hdm05, 2007. 2, 5

[23] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013. 3

[24] J. C. Platt and A. H. Barr. Constrained differential optimization for neural networks. 1988. 5

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, April 2015. 6

[26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 6, 7

[27] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013. 3

[28] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 5

[29] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015. 2, 3, 6, 7

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.

Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 4, 6

[31] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pages 140–153. Springer, 2010. 1

[32] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014. 3, 6, 7

[33] C. Wang, Y. Wang, and A. L. Yuille. An approach to pose-based action recognition. In *CVPR*, pages 915–922. IEEE, 2013. 2

[34] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, pages 1290–1297. IEEE, 2012. 2

[35] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015. 1, 6

[36] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 3

[37] D. Wu and L. Shao. Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *CVPR*, pages 724–731. IEEE, 2014. 2

[38] L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPR Workshop*, pages 20–27. IEEE, 2012. 2

[39] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015. 1, 2, 5, 6, 7

[40] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014. 2, 3

[41] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015. 1, 3, 6, 7