Deep Dynamic Neural Networks for Gesture Segmentation and Recognition

Di Wu, Ling Shao

the University of Sheffield
stevenwudi@gmail.com

Abstract. The purpose of this paper is to describe a novel Deep Dynamic Neural Networks *(DDNN)* for the track 3 of the Chalearn Looking at People 2014 challenge [1]. A generalised semi-supervised hierarchical dynamic framework is proposed for simultaneous gesture segmentation and recognition taking both skeleton and depth images as input modules. First, Deep Belief Networks *(DBN)* and 3D Convolutional Neural Networks *(3DCNN)* are adopted for skeletal and depth module accordingly to extract high level spatio-temporal features. Then the learned representations are used for estimating emission probabilities of the Hidden Markov Models to infer action sequence. The framework can be easily extended by including an ergodic state to segment and recognise video sequence in a frame-to-frame mechanism, rendering it possible for online segmentation and recognition for diverse input modules. Some normalisation details pertaining to preprocessing raw features are also discussed. This purely data-driven approach achieves ***0.8162*** score in this gesture spotting challenge. The performance is on par with a variety of the state-of-the-art hand-tuned-feature approaches and other learning-based methods, opening the doors for using deep learning techniques to explore time series multimodal data.

Keywords: Deep Belief Networks, 3D Convolutional Neural Networks, Gesture Recognition, ChaLearn

1 Introduction

In recent years, human action recognition has drawn increasing attention of researchers, primarily due to its growing potential in areas such as video surveillance, robotics, human-computer interaction, user interface design, and multimedia video retrieval.

Previous works on video-based motion recognition focused on adapting hand-crafted features and low-level hand-designed features have been heavily employed with much success. These methods usually have two stages: an optional feature detection stage followed by a feature description stage. Well-known feature detection methods ("interest point detectors") are Harris3D [2], Cuboids [3] and Hessian3D [4]. For descriptors, popular methods are Cuboids [5], HOG/HOF [2], HOG3D [6] and Extended SURF [4]. In a recent work of Wang *et al.* [7], dense

trajectories with improved motion-based descriptors epitomized the pinnacle of handcrafted features and achieved state-of-the-art results on a variety of "in the wild" datasets. Given the current trends, challenges and interests in action recognition, this list would probably continue to spread out extensively.

In the evaluation paper of Wang *et al.* [8], one interesting finding is that there is no universally best hand-engineered feature for all datasets, suggesting that learning features directly from the dataset itself may be more advantageous. Albeit the dominant methodology for visual recognition from images and videos relies on hand-crafted features, there has been a growing interest in methods that learn low-level and mid-level features, either in supervised, unsupervised, or semi-supervised settings [9,10,11].

With the recent resurgence of neural networks invoked by Hinton and others [12], deep neural architectures have been proposed as an effective solution for extracting high level features from data. Deep artificial neural networks (including the family of recurrent neural networks) have won numerous contests in pattern recognition and representation learning. Schmidhuber [13] compiled a historical survey compactly summarising relevant works with more than 850 entries of credited works. Such models have been successfully applied to a plethora of different domains: the GPU-based cuda-convnet [14] classifies 1.2 million high-resolution images into 1000 different classes; multi-column Deep Neural Networks [15] achieve near-human performance on the handwritten digits and traffic signs recognition benchmarks; 3D Convolutional Neural Networks [16,17] recognize human actions in surveillance videos; Deep Belief Networks combining with Hidden Markov Models [18,19] for acoustic and skeletal joints modeling outperform the decade-dominating paradigm of Gaussian Mixture Models+Hidden Markov Models. In these fields, deep architectures have shown great capacity to discover and extract higher level relevant features.

However, direct and unconstrained learning of complex problems is difficult, since (i) the amount of required training data increases steeply with the complexity of the prediction model and (ii) training highly complex models with very general learning algorithms is extremely difficult. It is therefore common practice to restrain the complexity of the model and this is generally done by operating on small patches to reduce the input dimension and diversity [11], or by training the model in an unsupervised manner [10], or by forcing the model parameters to be identical for different input locations (as in convolutional neural networks [14,15,16]).

With the immense popularity of Kinect [20], there has been renewed interest in developing methods for human gesture and action recognition from 3D skeletal data and depth images. A number of new datasets [21,22,23,24] have provided researchers with the opportunity to design novel representations and algorithms and test them on a much larger number of sequences. It may seem that the task of action recognition given 3D joint positions is trivial, but this is not the case, largely due to the high dimensionality of the pose space. Furthermore, to achieve continuous action recognition, the sequence need to be segmented into

contiguous action segments; such segmentation is as important as recognition itself and is often neglected in action recognition research.

In this paper, a data driven framework is proposed, focusing on analysis of acyclic video sequence labeling problems, *i.e.*, video sequences are non-repetitive as opposed to longer repetitive activities, *e.g.*, jogging, walking and running.

2 Experiments and Analysis

2.1 Chalearn LAP Dataset & Evaluation Metrics

This dataset[1] is on "multiple instance, user independent learning and continuous gesture spotting" [21] of gestures. And in the 3 track, there are more than 14,000 gestures are drawn from a vocabulary of 20 Italian cultural/anthropological sign gesture categories with 700 sample sequences for training and validation and 240 sample sequences for testing.

The evaluation criteria for this track is the *Jaccard* index (overlap) on a frame-to-frame basis.

$$J(A,B) = \frac{A \bigcap B}{A \bigcup B} \tag{1}$$

2.2 Model Architecture: Deep Dynamic Neural Networks

Inspired by the framework successfully applied to the speech recognition [18], the proposed model borrows the idea of a data driven learning system, relying on a pure learning approach in which all the knowledge in the model comes from the data without sophisticated pre-processing or dimensionality reduction. The proposed Deep Dynamic Neural Networks*(DDNN)* can be seen as an extension to [19] in that instead of only using the Restricted Boltzmann Machines to model human motion, various connectivity layers, *e.g.*, fully connected layers, convolutional layers, *etc.*, are stacked together to learn higher level features justified by a variational bound [12] from different input modules.

A continuous-observation HMM with discrete hidden states is adopted for modelling higher level temporal relationships. At each time step t, we have one random observation variable X_t. Additionally we have an unobserved variable H_t taking values of a finite set $\mathcal{H} = (\bigcup_{a \in \mathcal{A}} \mathcal{H}_a)$,where $\mathcal{H}_a$ is a set of states associated to an individual action $\boldsymbol{a}$ by force-alignment scheme defined in Sec. 2.4. The intuition motivating this construction is that an action is composed of a sequence of poses where the relative duration of each pose may vary. This variance is captured by allowing flexible forward transitions within the chain. With this definitions, the full probability model is now specified as HMM:

$$p(H_{1:T}, X_{1:T}) = p(H_1)p(X_1|H_1)\prod_{t=2}^{T} p(X_t|H_t)p(H_t|H_{t-1}), \tag{2}$$

[1] http://gesture.chalearn.org/homewebsourcereferrals

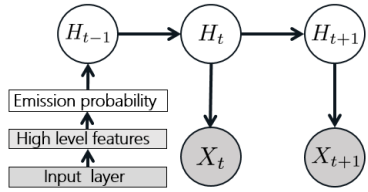


Fig. 1: Per-action model: a forward-linked chain. Inputs (skeletal features or depth image features) are first passed through Deep Neural Nets (Deep Belief Networks for skeletal modality or 3D Convolutional Neural Networks for depth modality) to extract high level features. The outputs are the emission probabilities of the hidden states.

where $p(H_1)$ is the prior on the first hidden state; $p(H_t|H_{t-1})$ is the transition dynamics model and $p(X_t|H_t)$ is the emission probability modelled by the deep neural nets.

The motivation for using deep neural nets to model marginal distribution is that by constructing multi-layer networks, semantically meaningful high level features will be extracted whilst learning the parametric prior of human pose from mass pool of data. In the recent work of [25], a non-parametric bayesian network is adopted for human pose prior estimation, whereas in the proposed framework, the parametric networks are incorporated. The graphical representation of a per-action model is shown as Fig. 1.

2.3 *ES-HMM*: Simultaneous Segmentation and Recognition

The aforementioned framework can be easily adapted for simultaneous action segmentation and recognition by adding an ergodic states-$\mathcal{ES}$ which resembles the silence state for speech recognition. Hence, the unobserved variable H_t takes an extra finite set $\mathcal{H} = (\bigcup_{a\in\mathcal{A}} \mathcal{H}_a) \bigcup \mathcal{ES}$, where $\mathcal{ES}$ is the ergodic state as the resting position between actions and we refer the model as *ES-HMM*.

Since our goal is to capture the variation in speed of performing gestures, we set the transitions in the following way: when being in a particular node n in time t, moving to time $t+1$, we can either stay in the same node (slower performance), move to node $n+1$ (the same speed of performance), or move to node $n+2$ (faster performance). From the $\mathcal{ES}$ we can move to the first three nodes of any gesture class, and from the last three nodes of any gesture class we can move to the $\mathcal{ES}$ as shown in Fig. 2. The *ES-HMM* framework differs from the Firing Hidden Markov Model of [26] in that we strictly follow the temporal independent assumption, forbidding inter-states transverse, preconditioned that a non-repetitive sequence would maintain its unique states throughout its performing cycle.

The emission probability of the trained model is represented as a matrix of size $N_{\mathcal{TC}} \times N_{\mathcal{F}}$ where $N_{\mathcal{F}}$ is the number of frames in a test sequence and output target class $N_{\mathcal{TC}} = N_{\mathcal{A}} \times N_{\mathcal{H}_a} + 1$ where $N_{\mathcal{A}}$ is the number of action class and $N_{\mathcal{H}_a}$ is the number of states associated to an individual action a and

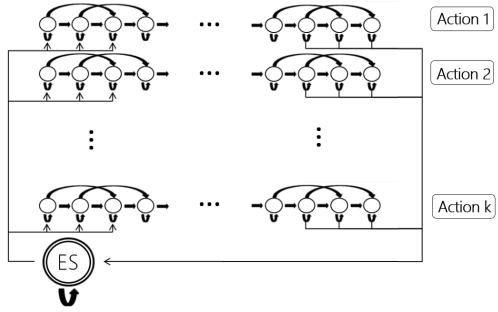


Fig. 2: State diagram of the *ES-HMM* model for low-latency action segmentation and recognition. An ergodic states (ES) shows the resting position between action sequence. Each node represents a single frame and each row represents a single action model. The arrows indicate possible transitions between states.

one $\mathcal{ES}$ state. Once we have the trained model, we can use the normal online or offline smoothing, inferring the hidden marginal distributions $p(H_t|X_t)$ of every node (frame) of the test video. Because the graph for the Hidden Markov Model is a directed tree, this problem can be solved exactly using the max-sum algorithm. The number of possible paths through the lattice grows exponentially with the length of the chain. The Viterbi algorithm searches this space of paths efficiently to find the most probable path with a computational cost that grows only linearly with the length of the chain [27]. We can infer the action presence in a new sequence by Viterbi decoding as:

$$V_{t,\mathcal{H}} = P(H_t|X_t) + \log(\max_{\mathcal{H}\in\mathcal{H}_a}(V_{t-1,\mathcal{H}})) \tag{3}$$

where initial state $V_{1,\mathcal{H}} = \log(P(H_1|X_1))$. From the inference results, we define the probability of an action $a \in \mathcal{A}$ as $p(y_t = a|x_{1:t}) = V_{T,\mathcal{H}}$. Result of the Viterbi algorithm is a path–sequence of nodes which correspond to hidden states of gesture classes. From this path we can infer the class of the gesture (*c.f.* Fig. 10). The overall algorithm for training and testing are presented in Algorithm 1 and 4.

2.4 Experimental Setups

For input sequences, there are three modalities, *i.e.* skeleton, RGB and depth (with user segmentation) provided. However, only skeletal modality and the depth modality are considered(*c.f.* Fig. 3). In the following experiments, the first 650 sample sequences are used for training, 50 for validation and the rest 240 for testing where each sequence contains around 20 gestures with some noisy non-meaningful vocabulary tokens.

Algorithm 1: Multimodal Deep Dynamic Networks – training

Data:
$\mathbf{X^1} = \{\mathbf{x_i^1}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - raw input(skeletal) feature sequence.
$\mathbf{X^2} = \{\mathbf{x_i^2}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - raw input(depth) feature sequence in the form of $M_1 \times M_2 \times T$, where M_1, M_2 are the height and width of the input image and T is the number of contiguous frames of the spatio-temporal cuboid.
Note that the GPU library *cuda-convnet* [14] used requires square size images and T is a multiple of 4.
$\mathbf{Y} = \{\mathbf{y_i}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - frame based local label (achieved by semi-supervised forced-aligment), where $\mathbf{y_i} \in \{C * S + \boldsymbol{1}\}$ with C is the number of class, S is the number of hidden states for each class, $\boldsymbol{1}$ as ergodic state.

for $m \leftarrow 1$ *to* 2 **do**
  if m *is* 1 **then**
    Preprocessing the data $\mathbf{X^1}$ as in Eq.4.
    Normalizing(zero mean, unit variance per dimension) the above features and feed to to Eq.6.
    Pre-training the networks using *Contrastive Divergence*.
    Supervised fine-tuning the Deep Belief Networks using $\mathbf{Y}$ by standard mini-batch *SGD* backpropagation.
  else
    Preprocessing the data $\mathbf{X^2}$ (normalizing, median filtering the depth data) Algo.2 or Algo.3.
    Feeding the above features to Eq.9.
    Supervised fine-tuning the Deep 3D Convolutional Neural Networks using $\mathbf{Y}$ by standard mini-batch *SGD* Backpropagation.

Result:
GDBN - a gaussian bernoulli visible layer Deep Belief Network to generate the emission probabilities for hidden markov model.
3DCNN - a 3D Deep Convolutional Neural Networks to generate the emission probabilities for hidden markov model.
$\mathbf{p(H_1)}$ - prior probability for $\mathbf{Y}$.
$\mathbf{p(H_t|H_{t-1})}$ - transition probability for $\mathbf{Y}$, enforcing the beginning and ending of a sequence can only start from the first or the last state.

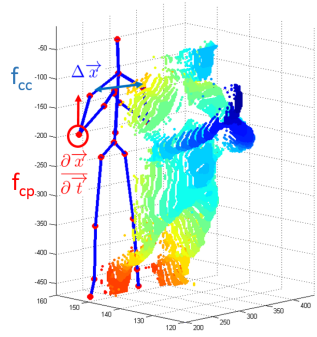


Fig. 3: Point cloud projection of depth image and the 3D positional features.

Hidden states (H_t)**:** Force alignment is used to extract the hidden states, *i.e.*, if a gesture token is 100 frames, the first 10 frames are assigned as hidden state ***1*** and the 10-20 frames are assigned as hidden state ***2*** and so on and so forth.

Ergodic states**:** Neutral frames are extracted as 5 frames before or after a gesture tokens labelled by ground truth.

2.5 Skeleton Module & DBN training

Only upper body joints are relevant to the discriminative gesture recognition tasks. Therefore, only the 11 upper body joints are considered. The 11 upper body joints used are *"ElbowLeft, WristLeft, ShoulderLeft, HandLeft, ElbowRight, WristRight, ShoulderRight, HandRight, Head, Spine, HipCenter"*.

The 3D coordinates of N joints of frame c are given as: $X_c = \{x_1^c, x_2^c, \ldots, x_N^c\}$. 3D positional pairwise differences of joints [19] are deployed for observation domain $\mathcal{X}$. They capture posture features, motion features by direction concatenation: $\mathcal{X} = [f_{cc}, f_{cp}]$ as demonstrated in Eq 4. Note that offset features f_{ci} used in [19] depend on the first frame, if the initialization fails which is a very common scenario, the feature descriptor will be generally very noisy. Hence, the offset features f_{ci} are discarded and only the two more robust features $[f_{cc}, f_{cp}]$ (as shown in Fig. 3) are kept:

$$f_{cc} = \{x_i^c - x_j^c | i, j = 1, 2, \ldots, N; i \neq j\} \tag{4}$$

$$f_{cp} = \{x_i^c - x_j^p | x_i^c \in X_c; x_j^p \in X_p\} \tag{5}$$

This results in a raw dimension of $N_{\mathcal{X}} = N_{joints} * (N_{joints} - 1)/2 + N_{joints}^2) * 3$ where N_{joints} is the number of joints used. Therefore, in the experiment with $N_{joints} = 11, N_{\mathcal{X}} = 528$. Admittedly, we do not completely neglect human prior knowledge about information extraction for relevant static postures, velocity and offset overall dynamics of motion data. Nevertheless, the aforementioned three attributes are all very crude pairwise features without any tweak into

the dataset or handpicking the most relevant pairwise, triple wise, *etc.* , designed features [28,29,26,30]. A similar data driven approach has been adopted in [22] where random forest classifiers were adapted to the problem of recognizing gestures using a bundle of 35 frames. These sets of feature extraction processes resemble the *Mel Frequency Cepstral Coefficients (MFCCs)* for the speech recognition community [18].

Gaussian Bernoulli Restricted Boltzmann machines Because input skeletal features(*a.k.a.*observation domain $\mathcal{X}$) are continuous instead of binomial features, we use the Gaussian RBM ($GRBM$) to model the energy term of first visible layer:

$$E(v,h;\theta) = -\sum_{i=1}^{D}\frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}h_j\frac{v_i}{\sigma_i} - \sum_{j=1}^{F} a_j h_j \tag{6}$$

The conditional distributions needed for inference and generation are given by:

$$P(h_{j=1}|\mathbf{v}) = g(\sum_i W_{ij}v_i + a_j)); \tag{7}$$

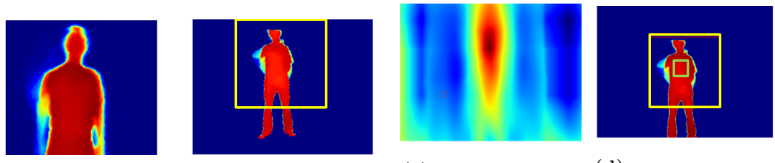
$$P(v_{i=1}|\mathbf{h}) = \mathcal{N}(v_i|\mu_i, \sigma_i^2). \tag{8}$$

where $\mu_i = b_i + \sigma_i^2\sum_j W_{ij}h_j$ and $\mathcal{N}$ is normal distribution. In general, we normalize the data (mean substraction and standard deviation division) in the preprocessing phase. Hence, in practice, instead of learning σ_i^2, one would typically use a fixed, predetermined unit value $\boldsymbol{1}$ for σ_i^2.

For high level skeleton feature extraction, two network architectures, *i.e.*, a smaller one and a larger one were experimented: $[N_{\mathcal{X}}, 1000, 1000, 500, N_{\mathcal{TC}}]$ and $[N_{\mathcal{X}}, 2000, 2000, 1000, N_{\mathcal{TC}}]$, where $N_{\mathcal{X}} = 528$ is the observation domain dimension; $N_{\mathcal{TC}}$ is the output target class. Because in all our experiments the number of states associated to an individual action $N_{\mathcal{H}_a}$ is chosen as 10 for modeling the states of an action class, therefore $N_{\mathcal{TC}} = 20 + 1 = 201$.

In the training set, there are in total $400,117$ frames. During the training of DBN, 90% is used for training, 8% for validation (for the purpose of early stopping) 2% is used for test evaluation. The feed forward networks are pretrained with a fixed recipe using stochastic gradient decent with a mini-batch size of 200 training cases. Unsupervised initializations tend to avoid local minima and increase the networks performance stability and we have run 100 epochs for unsupervised pre-training. For Gaussian-binary RBMs, learning rate is fixed at 0.001 while for binary-binary RBMs as 0.01 (note in generally training $GRBM$ requires smaller learning rate). For fine-tuning, the learning rate starts at 1 with 0.99999 mini-batch scaling. Maximum number of fine-tuning epoch is 500 with early stopping strategy and during the experiments, early stopping occurs around 440 epoch. Optimization complete with best validation score (the frame based prediction error rate) of 38.19%, with test performance 38.11%.

Though we believe further carefully choosing network architecture would lead to more competitive results, in order not to "creeping overfitting", as algorithms



(a) template image (b) test image (c) template response (d) shift-resize image

Fig. 4: Illustration of normalization scheme 1: template matching.

over time become too adapted to the dataset, essentially memorizing all its idiosyncrasies, and losing ability to generalize [31], we would like to treat the model as the aforementioned more generic approach. Since a completely new approach will initially have a hard time competing against established, carefully fine-tuned methods. More fundamentally, it may be that the right way to treat dataset performance numbers is not as a competition for the top place. This way, fundamentally new approaches will not be forced to compete for top performance right away, but will have a chance to develop and mature.

The performance of skeleton module is shown in Tab 1. And it can be seen that larger net (Net2) will generally perform better than smaller net (Net1), averaging multi-column nets almost will certainly further improve the performance [15]. Hence, in the following experiments, only the multi-column averaging results are reported.

2.6 Depth 3D Module

Preprocessing & Normalizing: shifting, scaling and resizing. Working directly with raw input Kinect recorded data frames, which are 480×640 pixel images, can be computationally demanding. Deepmind technology [32] presented the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using deep reinforcement learning. Similarly, basic preprocessing steps are adopted aimed at reducing the input dimensionality from the original 480×640 pixels to 90×90 pixels. The square-sized of the final image is required because the used GPU implementation from [14] expects square inputs and the input channel should be in the set of $[1, 3, 4x]$. There are two normalization schemes implemented as 2 using depth template matching method and 3 using skeletal joins as assistant normalization. Note that the scheme 2 depends heavily on the provided maximum depth from the recording scene and scheme 3 depends on the accurate detection of skeleton joins, and both scheme require the performer remains a roughly static position (though the max pooling scheme in 3DCNN to some extend overcome the problem of position shifting). Generally, scheme 3 is more robust than scheme 2 because the provided maximum depth can sometimes be very noisy, *e.g.*, Sample0671, Sample0692, Sample0699, *etc.* After the normalization and resizing, a cuboid of 4 frames, hence, size $90 \times 90 \times 4$, is extracted as a spatio-temporal unit.

Algorithm 2: Normalization scheme 1: template matching

Data:
$\mathbf{T}$ - exemplary template with original scale of size 320×320, (Sample0003 is chosen as the exemplary template, shown in 4a).
$\mathbf{R_{depth}}$ - reference depth, fixed to ***1941*** (acquired from the above exemplary template $\mathbf{T}$).
$\hat{\mathbf{T}}$ - test image, as shown in 4b.
$\mathbf{M}$ - user foreground segmented mask.

1 Apply a 5×5 aperture median filter to test depth frame $\hat{\mathbf{T}}$ as in [33] to reduce the salt and pepper noise.
2 Multiply test depth frame $\hat{\mathbf{T}}$ with the user segmented mask $\mathbf{M}$: $\hat{\mathbf{T}} = \hat{\mathbf{T}} \times \mathbf{M}$.
3 Template matching test image $\hat{\mathbf{T}}$ with $\mathbf{T}$ using normalized cross-correlation [34], the response score $\mathbf{R}$ is shown in 4c.
4 Shift the image according to the maximum response $\mathbf{R}$ to its centre applying affine transformation [35].
5 Scale the image according to reference depth $\mathbf{R_{depth}}$ and the median depth of a bounding box in the centre of the image with 25×25 size as shown as the green boundingp box in 4d.
6 Resize the image from 320×320 to 90×90.

Result:
$\tilde{\mathbf{T}}$ - Resize-normalized image shown in the yellow bounding box of 4d.

3DCNN Architecture & Details of Learning. The 3D convolution is achieved by convolving a 3D kernel to the cuboid formed by stacking multiple contiguous frames together. We follow the nomenclature as in [17]. However, instead of using *tanh* unit as in [17], the Rectified Linear Units (*ReLUs*) [14] were adopted where trainings are several times faster than their equivalents with *tanh* units. Formally, the value of a unit at position (x, y, z) (z here corresponds the time-axis) in the jth feature map in the ith layer, denoted as v_{ij}^{xyz}, is given by:

$$v_{ij}^{xyz} = max(0, (b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(t+r)})) \tag{9}$$

The 3DCNN architecture is depicted as Fig. 5: the input contextual frames are stacked as size $90 \times 90 \times 4$ substracting the mean activity over the training set from each pixel, the first layer contains 16 maps of $7 \times 7 \times 4$ 3D kernel followed by local response normalization layer [14] and stride 2 max pooling; the second convolutional layers has 32 maps of 5×5 kernel followed by local response normalization layer and stride 2 max pooling; the third convolution layer is composed of 32 maps of 6×6 kernel followed by max pooling; then we have one fully connected layer of size 1000; the output layer $N_{\mathcal{TC}}$ is of size $201 = 10 \times 20 + 1$ (number of hidden states for each class$\times$ number of classes plus one ergodic state).

Algorithm 3: Normalization scheme 2: skeleton normalization

Data:
$\mathbf{S_{spine}}$ - Skeleton Spine joints pixel coordinates.
$\mathbf{S_{shoulder}}$ - Skeleton Shoulder joints pixel coordinates.
$\hat{\mathbf{T}}$ - test image.
$\mathbf{M}$ - user foreground segmented mask.
$\mathbf{R_{length}}$ - reference length of shoulder to spine, fixed to ***100*** (1 meter).

1 Apply a 5×5 aperture median filter to test depth frame $\hat{\mathbf{T}}$.
2 Multiply test depth frame $\hat{\mathbf{T}}$ with the user segmented mask $\mathbf{M}$.
3 Shift the image according to the centroid of Spine joint $\mathbf{S_{spine}}$.
4 Scale the image according to the $\mathbf{R_{length}}/(\mathbf{S_{spine}} - \mathbf{S_{shoulder}})$.

Result:
$\tilde{\mathbf{T}}$ - Resize the shifted-scaledp image to 90×90 .

The training set is roughly of 400,000 frames and is divided into 33 mini-batches with first 30 batches for training and the rest 3 batches for validation. Standard *SGD* is run for the first 100 epochs with learning rate of 0.1 and the weight learning rate as 0.001 and weight bias learning rate 0.002 both momentum are fixed as 0.9, weight decay is fixed to 0.0005, the next 100 epochs with $0.1\times$ learning rate. Another network trained by randomly cropping 82×82 pixels on the flight as [14] is also implemented to enhance the model's robustness. During the test time, the centre part and other 4 corner parts are averaged to obtain the final score, *c.f.* Fig 6c. Due to the time constraint, only 150 epochs are trained with the learning rate reduced to one tenth at the 92th epoch. The training frame based classification error for the aforementioned two networks are shown in 6a and 6b. One interesting observation is that for the network with uncropped input, reducing the learning rate at 100 epoch, the frame-based classification rate reduces drastically whereas for the network with cropped input, reducing the learning rate results in a spike increase of frame-based classification error rate. The reason for this discrepancy worths further investigation. The performance of depth module is shown in Tab 1.

Looking into the networks-visualization of filter banks. The weight filters of the first *conv1* layer are illustrated in Fig 7 and it can be seen that both shape pattern filters and motion filters are learnt effectively and the filters/weights of the cropped input trained networks are smoother then the uncropped one. Interestingly, the 3DCNN is able to learn the most informative motion part of the body effectively (highest response parts are the arms/hands areas), albeit no signal was explicitly given during training instructing which body parts the gesture recognition tasks should focus on.

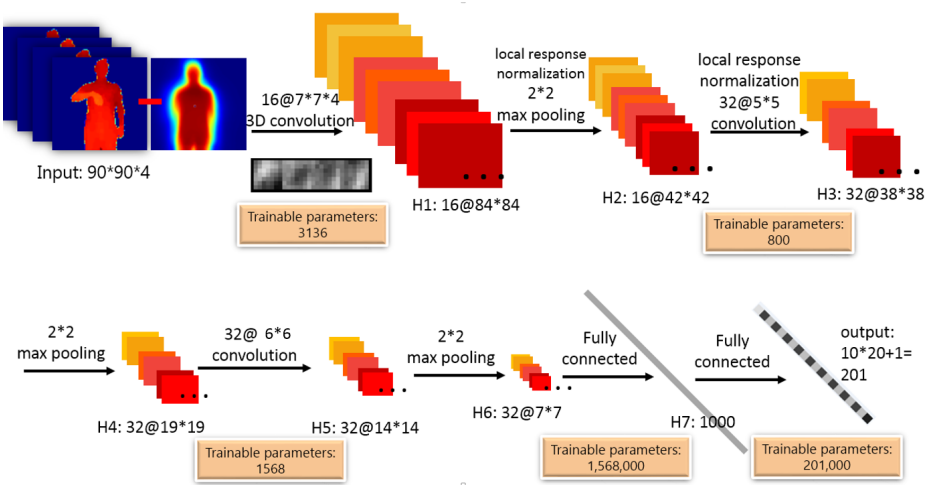
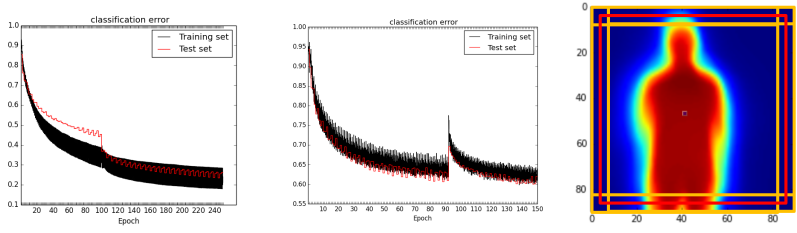


Fig. 5: An illustration of the architecture of the 3DCNN architecture.



(a) Frame based classification error with uncropped input.

(b) Frame based classification error with cropped input.

(c) Cropped images to enhance model's robustness.

Fig. 6: Visualization of the first filters and training statistics for 3DCNN.

2.7 Post-Processing

The predicted token less than 20 frames are discarded as noisy tokens. Note that there are many noisy gesture tokens predicted by viterbi decoding. One way to sift through the noisy tokens is to discard the token path log probability small than certain threshold. However, because the metric of this challenge: *Jaccard index* strongly penalizes false negatives, experiments show that it's better to have more false positives than to miss true positives. Effective ways to detect false positives should be an interesting aspect of future works.

2.8 Score Fusion

To fuse the dual model prediction, the strategy shown as Fig 8 is adopted. The complementary properties of both modules can be seen from the Viterbi

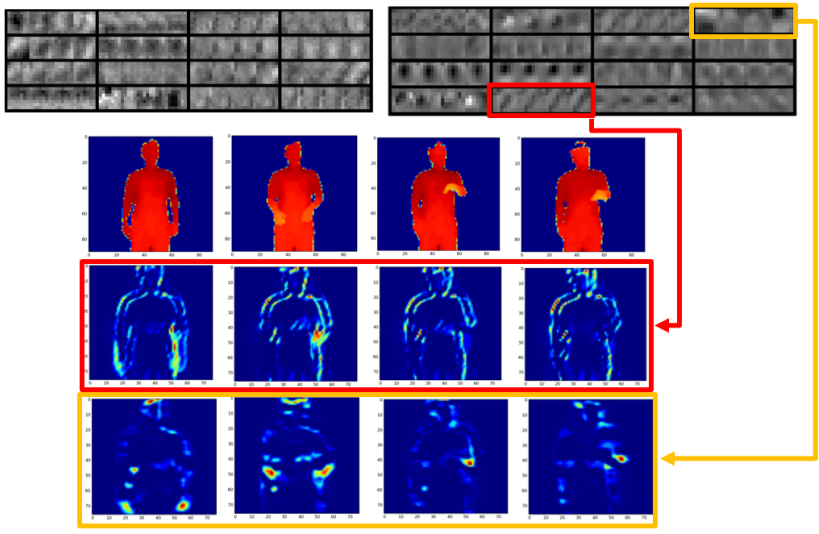


Fig. 7: Top left: the *conv1* weights of the 3DCNN learnt with uncropped input; top right: the *conv1* weights of the 3DCNN learnt with cropped input. It can be seen that filters/weights of the cropped input trained networks are smoother. Bottom: visualization of sample frames after *conv1* layer (Sample0654, 264-296 frames, sampled every 8 frames). It can be seen that the filters of the first convolutional layer are able to learn both shape pattern(red bounding box) and motion(yellow bounding box). Also note that the high response maps correspond to the most informative part of the body, even though during the training process, all local patches are learned indiscriminately regardless of its location.

path decoding plot in Fig 10. Note that the skeleton module generally performs better than the depth module, one reason could be that the skeleton joints learnt from [20] lie in success of utilizing huge and highly varied training data: from both realistic and synthetic depth images, a total number of 1 million images were used to train the deep randomized decision forest classifier in order to avoid overfitting. Hence skeleton data are more robust.

2.9 Computational complexity

Though learning the Deep Neural Networks using stochastic gradient descent is tediously lengthy, once the model finishes training, with a low inference cost, our framework can perform real-time video sequence labeling. Specifically, a single feed forward neural network incurs trivial computational time ($\mathcal{O}(T)$) and is fast because it requires only matrix products and convolution operations. The complexity of Viterbi algorithm is $\mathcal{O}(T * |S|^2)$ with number of frames T and state number S.

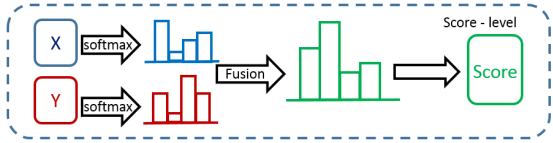


Fig. 8: Illustration of descriptor fusion.

Module \ Evaluation Set	Validation	Test
Skeleton–DBDN Net1	0.7468	-
Skeleton–DBDN Net2	0.8017	-
Skeleton–DBDN MultiNet	0.8236	0.7873
Depth–3DCNN Norm1 2	0.6378	-
Depth–3DCNN Norm2 3	0.6924	0.6371
Score Fusion	0.8045	***0.8162***

Table 1: Comparison of results in terms of Jaccard index between different network structures and various modules. DBDN Net1 corresponds to network structure of $[528, 1000, 1000, 500, 201]$ and DBDN Net2 $[528, 2000, 2000, 1000, 201]$, DBDN MultiNet is the average of 3 Nets (2 Net1 and 1 Net2 with different initializations). It can be seen that larger net has better performance and multi-column net will further improve the classification rate. Norm1 corresponds to the normalization scheme 2 and Norm2 corresponds to the scheme 3.

3 Conclusion and Discussion

Hand-engineered, task-specific features are often less adaptive and time-consuming to design. This difficulty is more pronounced with multimodal data as the features have to relate multiple data sources. In this paper, we presented a novel Deep Dynamic Neural Networks(DDNN) framework that utilizes Deep Belief Networks and 3D Convolutional Neural Networks for learning contextual frame-level representations and modeling emission probabilities for Markov Field. The heterogeneous inputs from skeletal joints and depth images require different feature learning methods and the late fusion scheme is adopted at the score level. The experimental results on bi-modal time series data show that the multimodal DDNN framework can learn a good model of the joint space of multiple sensory inputs, and is consistently as good as/better than the unimodal input, opening the door for exploring the complementary representation among multimodal inputs. It also suggests that learning features directly from data is a very important research direction and with more and more data and flops-free computational power, the learning-based methods are not only more generalizable to many domains, but also are powerful in combining with other well-studied probabilistic graphical models for modeling and reasoning dynamic sequences. Future works include learning the share representation amongst the heterogeneous inputs at

the penultimate layer and backpropagating the gradient in the share space in a unified representation.

References

1. Escalera, S., Bar, X., Gonzlez, J., Bautista, M., Madadi, M., Reyes, M., Ponce, V., Escalante, H., Shotton, J., Guyon, I.: Chalearn looking at people challenge 2014: Dataset and results. In: ECCV workshop. (2014)
2. Laptev, I.: On space-time interest points. International Journal of Computer Vision (2005)
3. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, IEEE (2005)
4. Willems, G., Tuytelaars, T., Gool, L.V.: An efficient dense and scale-invariant spatio-temporal interest point detector. In: European Conference on Computer Vision, Springer (2008)
5. Scovanner, P., Ali, S., Shah, M.: A 3-dimensional sift descriptor and its application to action recognition. In: International Conference on Multimedia, ACM (2007)
6. Klaser, A., Marszalek, M., Schmid, C.: A Spatio-Temporal Descriptor Based on 3D-Gradients. In: British Machine Vision Conference. (2008)
7. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision (2013)
8. Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C., et al.: Evaluation of local spatio-temporal features for action recognition. In: British Machine Vision Conference. (2009)
9. Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C.: Convolutional learning of spatio-temporal features. In: European Conference on Computer Vision. Springer (2010)
10. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: IEEE Conference on Computer Vision and Pattern Recognition. (2011)
11. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Spatio-temporal convolutional sparse auto-encoder for sequence classification. In: British Machine Vision Conference. (2012)
12. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation (2006)
13. Schmidhuber, J.: Deep learning in neural networks: An overview. arXiv preprint arXiv:1404.7828 (2014)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems. (2012)
15. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition. (2012)
16. Shuiwang Ji, Wei Xu, M.Y., Yu, K.: 3d convolutional neural networks for human action recognition. In: International Conference on Machine Learning, IEEE (2010)
17. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on (2013)

18. Mohamed, A., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. Audio, Speech, and Language Processing, IEEE Transactions on (2012)
19. Wu, D., Shao, L.: Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. (2014)
20. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR. (2011)
21. Escalera, S., Gonzlez, J., Bar, X., Reyes, M., Lops, O., Guyon, I., Athitsos, V., Escalante, H.J.: Multi-modal gesture recognition challenge 2013: Dataset and results. In: ACM ChaLearn Multi-Modal Gesture Recognition Grand Challenge and Workshop. (2013)
22. Fothergill, S., Mentis, H.M., Kohli, P., Nowozin, S.: Instructing people for training gestural interactive systems. In: ACM CHI. (2012)
23. Guyon, I., Athitsos, V., Jangyodsuk, P., Hamner, B., Escalante, H.J.: Chalearn gesture challenge: Design and first results. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2012)
24. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. In: IEEE Conference on Computer Vision and Pattern Recognition. (2012)
25. Lehrmann, A., Gehler, P., Nowozin, S.: A non-parametric bayesian network prior of human pose. In: International Conference on Computer Vision. (2013)
26. Nowozin, S., Shotton, J.: Action points: A representation for low-latency online human action recognition. Technical report (2012)
27. Bishop, C.: Pattern recognition and machine learning. Springer (2006)
28. Chaudhry, R., Ofli, F., Kurillo, G., Bajcsy, R., Vidal, R.: Bio-inspired dynamic 3d discriminative skeletal features for human action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2013)
29. Müller, M., Röder, T.: Motion templates for automatic classification and retrieval of motion capture data. In: SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association (2006)
30. Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., Bajcsy, R.: Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. Journal of Visual Communication and Image Representation (2013)
31. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: IEEE Conference on Computer Vision and Pattern Recognition. (2011)
32. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
33. Wu, D., Zhu, F., Shao, L.: One shot learning gesture recognition from rgbd images. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, IEEE (2012) 7–12
34. Lewis, J.: Fast normalized cross-correlation. In: Vision interface. Volume 10. (1995) 120–123
35. Bradski, G. Dr. Dobb's Journal of Software Tools
36. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy). (June 2010) Oral Presentation.

4 Supplementary materials

4.1 Deep Learning Library: Theano & cuda-convnet

Theano. The Deep Belief Network library used in this section is ***Theano*** [36] [2] which is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

cuda-convnet. The GPU enabled blazing fast Convolutional Neural Network library used in this section is ***cuda-convnet*** [14] [3] which is a fast C++/CUDA implementation of convolutional (or more generally, feed-forward) neural networks. It can model arbitrary layer connectivity and network depth. Any directed acyclic graph of layers will do. Training is done using the backpropagation algorithm.

4.2 Details of the Code

Deep Belief Dynamic Networks The python project for "Leveraging Hierarchical Parametric Network for Skeletal Joints Action Segmentation and Recognition" can be found at:
`https://github.com/stevenwudi/CVPR_2014_code`

Deep 3D Convolutional Dynamic Networks The python project, C++/CUDA backend for Deep 3D Convolutional Dynamic Network can be found at:
`https://github.com/stevenwudi/3DCNN_HMM`

4.3 Extra Figures for Illustration

[2] http://deeplearning.net/software/theano/
[3] https://code.google.com/p/cuda-convnet/

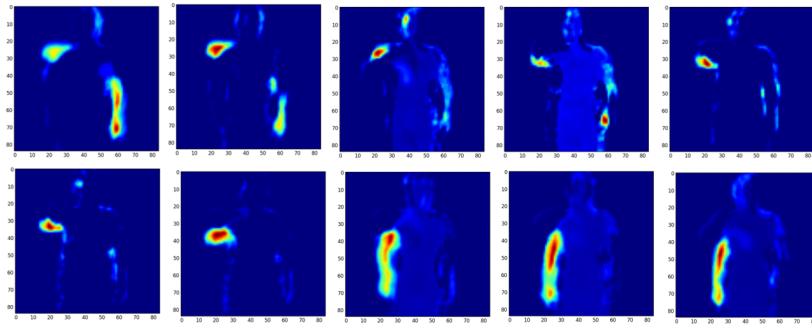


Fig. 9: More illustrations of the middle level features from the activation images after first convolutional layer. High response arms and hands areas are learnt automatically without explicit learning signal in term of location information.

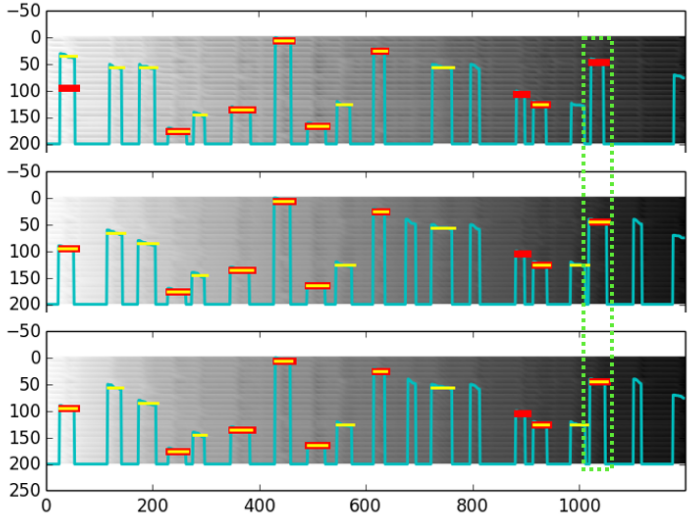


Fig. 10: Viterbi decoding of two modules and their fusion result of sample sequence 704. Top to bottom: skeleton, depth, score fusion with x-axis representing the time and y-axis representing the hidden states of all the classes with the ergodic state at the bottom. Red lines are the ground truth label, cyan lines are the viterbi shortest path and yellow lines are the predicted label. There are some complementary information of the two modules and generally skeletal module outperforms the depth module. The fusion of the two could exploit the uncertainty, *e.g.* light green dashed box indicates that depth module makes the correct prediction whereas the skeletal module fails, the combined module is still making the correct prediction.

Algorithm 4: Multimodal Deep Dynamic Networks – testing

Data:
$\mathbf{X^1} = \{\mathbf{x_i^1}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - raw input(skeletal) feature sequence.
$\mathbf{X^2} = \{\mathbf{x_i^2}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - raw input(depth) feature sequence in the form of $M \times M \times T$.
GDBN - a gaussian bernoulli visible layer Deep Belief Network to generate the emission probabilities for hidden markov model.
3DCNN - the trained 3D Deep Convolutional Neural Networks to generate the emission probabilities for hidden markov model.
$\mathbf{p(H_1)}$ - prior probability for $\mathbf{Y}$.
$\mathbf{p(H_t|H_{t-1})}$ - transition probability for $\mathbf{Y}$.

for $m \leftarrow 1$ *to* 2 **do**
  if m *is* 1 **then**
    Preprocessing and normalizing the data $\mathbf{X^1}$ as in Eq.4.
    Feedforwarding network **GDBN** to generate the emission probability $\mathbf{p(X_t|H_t)}$ in Eq.2.
    Generating the score probability matrix $\mathbf{S^1} = \mathbf{p(H_{1:T}, X_{1:T})}$.
  else
    Preprocessing the data $\mathbf{X^2}$ (normalizing, median filtering the depth data) Algo.2 or Algo.3.
    Feedforwarding **3DCNN** to generate the emission probability $\mathbf{S^2} = \mathbf{p(X_t|H_t)}$ in Eq.2.
    Generating the score probability matrix $\mathbf{S^2} = \mathbf{p(H_{1:T}, X_{1:T})}$.
Fusing the score matrix $\mathbf{S} = \mathbf{S^1} + \mathbf{S^2}$.
Finding the best path $\mathbf{V_{t,\mathcal{H}}}$ using $\mathbf{S}$ by Viterbi decoding as in Eq.3.

Result:
$\mathbf{Y} = \{\mathbf{y_i}\}_{\mathbf{i}\in[\mathbf{1}\ldots\mathbf{t}]}$ - frame based local label where $\mathbf{y_i} \in \{C * S + \boldsymbol{1}\}$ with C is the number of class, S is the number of hidden states for each class, $\boldsymbol{1}$ as ergodic state.
$\mathbf{C}$ - global label, the anchor point is chosen as the middle state frame.