Learning and Refining of Privileged Information-based RNNs for Action Recognition from Depth Sequences

Zhiyuan Shi, Tae-Kyun Kim
Department of Electrical and Electronic Engineering,
Imperial College London
{z.shi,tk.kim}@imperial.ac.uk

Abstract

*Existing RNN-based approaches for action recognition from depth sequences require either skeleton joints or hand-crafted depth features as inputs. An end-to-end manner, mapping from raw depth maps to action classes, is non-trivial to design due to the fact that: 1) single channel map lacks texture thus weakens the discriminative power; 2) relatively small set of depth training data. To address these challenges, we propose to learn an RNN driven by privileged information (PI) in three-steps: An encoder is pre-trained to learn a joint embedding of depth appearance and PI (*i.e. *skeleton joints). The learned embedding layers are then tuned in the learning step, aiming to optimize the network by exploiting PI in a form of multi-task loss. However, exploiting PI as a secondary task provides little help to improve the performance of a primary task (*i.e. *classification) due to the gap between them. Finally, a bridging matrix is defined to connect two tasks by discovering latent PI in the refining step. Our PI-based classification loss maintains a consistency between latent PI and predicted distribution. The latent PI and network are iteratively estimated and updated in an expectation-maximization procedure. The proposed learning process provides greater discriminative power to model subtle depth difference, while helping avoid overfitting the scarcer training data. Our experiments show significant performance gains over state-of-the-art methods on three public benchmark datasets and our newly collected Blanket dataset.*

1. Introduction

Action recognition from depth sequences [57, 34, 29, 44, 49] has attracted significant interest recently due to the emergence of low-cost depth sensors. Human action refers to a temporal sequence of primitive movements carried out by a person [55]. Recurrent neural network (RNN) [17] is naturally suited for modeling temporal dynamics of human actions as it can be used to model joint probability distribution over sequences, especially in the case of long short-term memory (LSTM) [18] which is capable of modeling long-term contextual information of complex sequential data.

RNN-based approaches become the dominant solution [61, 42, 9, 27] for action recognition from depth sequence recently. However, these approaches require either skeleton joints [61, 9, 22] or hand-crafted depth features [42] as inputs in both training and testing. Skeleton-based action recognition assumes that a robust tracker can estimate body joints accurately in the testing stage. This often does not hold in practice, especially when a human body is partly in view or the person is not in an upright position. Hand-crafted features with heuristic parameters are designed for task-specific data. This often requires multi-stage processing phases, each of which needs to be carefully designed and tuned.

An end-to-end trainable model from raw video frames [8] is desired to extract spatio-temporal features and model complex sequences in a unified framework. This learning pipeline typically combines a deep convolutional neural network (CNN) [25] as visual feature extractor and an RNN [17] to model and recognize temporal dynamics of sequential data. Unfortunately, these conventional end-to-end manners (CNN+RNN) are difficult to be applied to action recognition from depth sequences due to the fact that: 1) Color and texture are precluded in depth maps, which weaken the discriminative power of the representation captured by the CNN model. 2) Existing depth data of human actions are considered as a small-scale dataset compared to publicly available RGB image dataset. These conventional pipelines are purely data-driven that learn its representation directly from the pixels. Such model is likely at the risk of overfitting when the network is optimized on limited training data.

To address the above-mentioned issues, we propose a privileged information-based recurrent neural network (PRNN) that exploits additional knowledge to obtain a better estimate of network parameters. This additional knowl-

1063-6919/17 $31.00 © 2017 IEEE
DOI 10.1109/CVPR.2017.498

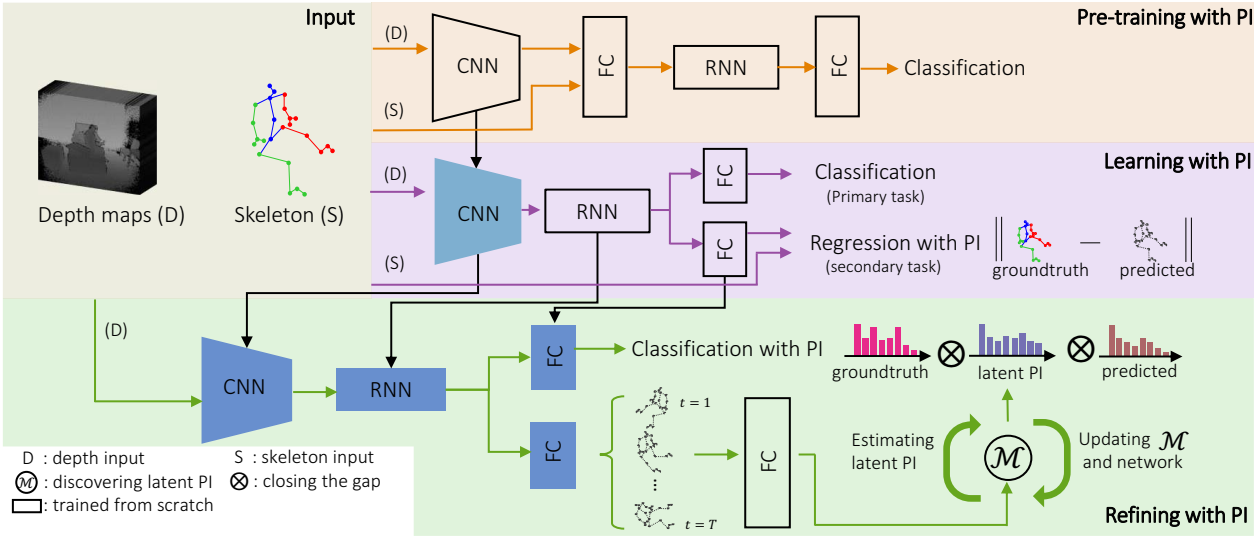

Figure 1: The proposed framework of PI-based RNNs. Our approach consists of three steps: 1) The pre-training step taking both depth maps and skeleton as input. An embedded encoder is trained in a standard CNN-RNN pipeline. 2) The trained encoder is used to initialize the learning step. A multi-task loss is applied to exploit the PI in the regression term as a secondary task. 3) Finally, refining step aims to discover the latent PI by defining a bridging matrix, in order to maximize the effectiveness of the PI. The latent PI is utilized to close the gap between different information. The latent PI, bridging matrix and the network are optimized iteratively in an EM procedure.

edge, also referred to as privileged information (PI) [41], hidden information [50] or side information [54, 19], is only available during training but not available during testing. Our model aims to encode PI into the structure or parameters of networks automatically and effectively during the training stage. In this work, we consider skeleton joints as the PI in the proposed three-step training process (see Fig. 1). A pre-training stage is introduced that taking both depth sequences and skeleton joints as input. The learned embedding layers construct intermediate distributions over the appearance of depth sequences and skeleton joints. As our method aims to utilize only depth sequences as input in testing stage, we then optimize our model by formulating the PI into an multi-task loss in learning step: a standard softmax classification loss as our primary task, and a regression loss as our secondary task, which learn the mapping parameters to predict the skeleton joints from depth appearance. However, We observe empirically that exploiting PI as a secondary task provides little help to improve the performance of primary task due to the gap between them. Finally, a bridging matrix is defined to connect two tasks by discovering latent PI in the refining step. We present a PI-based classification loss serving as a connector to maintain a consistency between latent PI and primary output distribution by penalizing the violation of the loss inequality. We enforces dependencies across regression and classification targets by seeking shared information. The bridging matrix, latent PI and network parameters are iteratively estimated and updated in an expectation-maximization (EM) procedure. This proposed learning process can provide greater discriminative power to model subtle depth difference, while helping avoid overfitting the scarcer training data. As we encode skeleton joints as PI, our model does not require a skeleton tracker in a testing stage, showing its better generalizability in a more challenging scenario, such as when a human body is partly in view or the person is not in an upright position.

We evaluate the proposed PRNN against state-of-the-arts on the task of action recognition from depth sequences. We demonstrate that our approach can achieve higher accuracy on the three public benchmark datasets: MSR Action3D [26], SBU Interaction dataset [59] and Cornell Activity [39]. A larger performance gain can be obtained on our newly collected Blanket dataset, where actions captured from a challenging camera view-point and some actions are partially occluded by a blanket. We also compare with several variants of our model and show that each component consistently contributes to the overall performance.

2. Related Work

Action recognition from depth sequence Human action recognition using depth maps can be classified in local or global methods. The elaborately designed features [26, 47, 34] are typically extracted from spatio-temporal interest points to describe the local appearance in 3D volumes or the area around human joints [16]. On the other

hand, high-level representations [56] aim to globally model the postures and capture the temporal evolution of actions. To model sequential state transitions in a principled way, hidden Markov model (HMM) has attracted a lot of interest [14] in capturing the temporal structure of human action dynamics. These HMM-based methods require that video sequences are precisely cropped and aligned with actions of interest, which itself is a difficult task for real-world videos. RNNs are able to handle both variable-length input and output that become the dominant model [42, 9, 61] recently, achieving superior performance over previous approaches. HBRNN [9] divides human skeleton into five corresponding parts and feed them into five bidirectionally recurrently connected subnets. [61] improve the model of [9] by automatically discovering the inherent correlations among skeleton joints. Instead of assuming skeleton joints are always reliable in testing stages, [42] model the dynamic evolution of actions by measuring the salient motions from the input depth appearance. The depth features are still extracted based on hand-crafted heuristics. In this paper, we provide an end-to-end solution to action recognition from raw depth sequences.

Learning with PI Data-driven approaches leverage large amounts of training data to determine the optimal model parameters in a bottom-up fashion. Purely data-driven methods are often very brittle and prone to fail when learning with limited training data, due to overfitting or an optimization obstacle involved. Learning with additional knowledge is a natural solution to alleviate this issue. This knowledge, also referred to as PI [41], hidden information [50] or side information [54], which can help to provide more explanations in training but will not be available at testing. Learning with PI has been investigated in many existing algorithms. [11] incorporate PI into an objective function of a structural SVM to improve object localization performance. [7] show that the incorporation of additional information can enhance the dependency between output variables and latent variables in a random forest framework. Additional knowledge has also been considered in neural networks. [5] explore the architecture by providing intermediate targets. [10] demonstrate the effectiveness of prior distribution for adjusting the model parameters to improve its generalization. More recently, [30] present a regularized RNNs with additional information for RGB video sequences. However, PI is either pre-trained or fixed in previous methods. In this work, we propose to optimize our end-to-end trainable model with iteratively estimating and updating latent PI for depth-based action recognition.

3. Spatio-Temporal Modeling

We illustrate an overall view of our model in Figure 1. The architecture mainly consists of an encoder, recurrent layers and PI-based learning. The encoder consists of several layers of convolutions which takes as input a collection of videos $\mathcal{V}$, where each video V_j is a sequence of frames $V_j = \{v_t : t = 1, ..., T_j\}$. The encoder produces vector space representations $X_j = \{\boldsymbol{x}_t : t = 1, ..., T_j\}$ for all frames of V_j. The recurrent network is built for integrating over time all the available information from X_j. Finally, PI is incorporated to jointly optimize all the layer parameters in the proposed three-step learning process.

Convolutional Neural Network The spatial appearance of action and contextual scenes on an individual frame is captured by our encoder. The architecture of our encoder is illustrated in Figure 2. It is inspired from VGG-VeryDeep [37], which is slightly modified from the 11 weights layer version by considering the depth maps and smaller training data. The network comprises five convolutional layers, five max-pooling layers. The rectified linear unit [25] is adopted as the activation function. Compared to the widely used CNN encoder for RGB data [30, 37], our encoder is more compact and effective for depth sequences. It is used to extract a feature vector from an input frame. Given an input depth frame $\boldsymbol{v}_t \in \mathbb{R}^{224\times224}$, an activation map $f_t^6 \in \mathbb{R}^{7\times7\times512}$ can be obtained from "outMap6" layer. We apply a linear transformation between the activation map and feature vectors by $\boldsymbol{x}_t = \tanh(\boldsymbol{W}^6 f_t^6 + b^6)$. This "map to sequences" operation generates an input vector $\boldsymbol{x}_t \in \mathbb{R}^{1\times1000}$ for recurrent layers in refining step.

Recurrent Neural Network RNNs are neural networks with feedback loops that produce the recurrent connection in the unfolded network [6, 33, 28]. Given an input sequence from the above encoder X_n, the hidden states of a recurrent layer $h_j = (\boldsymbol{h}_t : t = 1, ..., T_j)$ are defined as $\boldsymbol{h}_t = \tanh(\boldsymbol{W}^h \boldsymbol{x}_t + \boldsymbol{U}^h \boldsymbol{h}_{t-1} + b^h)$. Here $\boldsymbol{W}^h$,$\boldsymbol{U}^h$ are parameters of an affine transformation which update the connection weights among input layer, hidden layer. RNNs suffer from the vanishing and the exploding gradient problem [4]. We adopt LSTM [18] to address the problem of learning long-range dependencies, where a memory cell vector $\boldsymbol{c}_t$ is maintained at each time step t. LSTM contains one self-connected memory cell $\boldsymbol{c}$ and three multiplicative units, *i.e*. the input gate $\boldsymbol{i}$, the forget gate $\boldsymbol{f}$ and the output gate $\boldsymbol{o}$, which can store and access the long range contextual information of a temporal sequence. Please refer to [18] for the precise form of the update.

4. PI-Based RNNs

Standard recurrent neural networks do not provide a mechanism to exploit the PI when it is available at training time. We first present a pre-training strategy. The learned encoder is applied to the learning step and tuned together with RNNs by formulating the PI into a multi-task loss. In the final refining step, latent PI is discovered and iteratively updated with network parameters.

4.1. Pre-training with PI

A pre-training strategy is proposed to learn a joint embedding by taking both depth sequences V_j and skeleton joints annotation $\boldsymbol{E} = \{\boldsymbol{e}_1, ..., \boldsymbol{e}_S\}$ as input. Each $\boldsymbol{e}_s \in \mathcal{R}^3$ has 3 coordinates. In this stage, $\boldsymbol{x}_t$ is not directly applied to RNNs. Instead, the additional layer transforms $\boldsymbol{x}_t$ together with $\boldsymbol{E}$ to derive an embedding space :

$$\boldsymbol{x}'_t = \tanh(\boldsymbol{W}^7 \boldsymbol{x}_t + \boldsymbol{W}_e \boldsymbol{E} + b^7) \quad (1)$$

where $\boldsymbol{W}_e$ is the weight matrix connecting the skeleton joints. The resulting $\boldsymbol{x}'_t$ have the same dimensionality (1000) as $\boldsymbol{x}_t$. This is followed by RNNs to model the dynamics of sequential data. Finally, similar to most RNNs for classification task, a softmax layer is adopted to transform the hidden state vector into the probability distribution of action classes.

The key insight of the pre-training stage is to learn a depth encoder that optimizes the embedding over both depth appearance and skeleton joints. The learned encoder serves as an initialization in the next learning stage. This pre-training stage leads to a significant improvement in both efficiency and effectiveness.

4.2. Learning with PI

Multi-task loss. To obtain the class predictions of an input sequence $\boldsymbol{X}_j$, the hidden state can be mapped to an output vector $\boldsymbol{y}_j = (\boldsymbol{y}_t : t = 1, ..., T_j)$. During training, we measure the deviation between groundtruth and last memory cell at the frame T for classification loss, since LSTMs have the ability to memorize the content of an entire sequence. For regression loss, we accumulate the loss of each frame t across the T frame sequence. The final objective function in the learning step is to minimize the cumulative maximum-likelihood loss over all training sequences:

$$\mathcal{L}^{\mathrm{L}}(\boldsymbol{\Omega}) = \sum_{j=1}^{J} \mathcal{L}^c(T, j) + \lambda \sum_{j=1}^{J} \sum_{t=1}^{T} \mathcal{L}^r(t, j) \quad (2)$$

There are J sequences in the training set $\boldsymbol{\Omega}$. The hyper-parameter λ in Eqn. 2 controls the balance between the two losses. The classification loss and regression loss are defined as follows:

Classification loss. $\boldsymbol{y}_t \in \mathbb{R}^K$ represents an 1-of-K encoding of the confidence scores on K classes of actions, which can be derived as $\boldsymbol{y}_t = \tanh(\boldsymbol{W}^y \boldsymbol{h}_t + b^y)$. This output vector can be transformed into a vector of probabilities $p(y_{tk})$ for each class k by softmax function as $p(y_{tk}) = e^{y_{tk}} / \sum_{l=1}^{K} e^{y_{tl}}$. To learn the model parameters of our model, cross entropy loss between the predicted distribution $p(\boldsymbol{y}_t)$ and target class g_t is defined as

$$\mathcal{L}^c(t, j)(= -\sum_{k=1}^{K} \delta(k - g_t) \log p(y_{jtk})$$

for the sample t of the j-th video, where $\delta(\cdot)$ is the Dirac delta function, and g_t denotes the groundtruth label of the sample t.

Regression loss. Besides classification output, our model has another sibling output layer as regression term. We define a skeleton regression targets for groundtruth keypoints $\hat{\boldsymbol{E}}_t = \{\hat{\boldsymbol{e}}_{t1}, ..., \hat{\boldsymbol{e}}_{tS}\}$ and predicted locations $\boldsymbol{B}_t = \{\boldsymbol{b}_{t1}, ..., \boldsymbol{b}_{tS}\}$ at each time step t. We select $\hat{\boldsymbol{E}}$ as a subset of the skeleton annotations $\boldsymbol{E}$, because this is secondary target and an accurate estimation of all skeleton joints is not needed in testing. Each instance is accompanied with a set of keypoint $\{\hat{e}^x_{ts}, \hat{e}^y_{ts}\}_{s=1}^{S}$ locations, which are normalized with respect to the center and the width and height of the input region. The loss associated with the task of measuring the skeleton estimation can be expressed as

$$\mathcal{L}^r(t, j) = \frac{1}{S} \sum_{s=1}^{s=S} ((\hat{e}^x_{jts} - b^x_{jts})^2 + (\hat{e}^y_{jts} - b^y_{jts})^2)$$

where we use L_2 distance between the normalized keypoints location to quantify the dissimilarity. This loss function and regression layer only appear in the training stage for optimizing the neural network with additional information.

This extension, known as multi-task learning [32], utilize the task relationships to learn all individual tasks simultaneously, such that information can be shared in the common structure of the model to benefit all tasks. Similar as [12], it will help the classification prediction by considering the regression aspects. During testing, the regression component will be disabled.

4.3. Refining with PI

However, the conventional multi-task loss in the last step does not consider any relationship between two tasks. We observe empirically that purely exploiting PI as a secondary task provides little help to improve the performance of primary task due to the gap between them. To maximize the effectiveness of PI for helping primary task, we propose to discover latent PI from the secondary task in this refining step. The latent PI is utilized in the primary task to optimize the network. The updated network is further used to refine latent PI iteratively in an EM procedure.

Latent PI modeling We define latent PI as a informative distribution which is jointly modeled by secondary task and a bridging matrix. The bridging matrix $\boldsymbol{\mathcal{M}}$ aim to capture the underlying dependencies between primary and secondary task. The log-likelihood of the defined model can be expressed as:

$$Q(\boldsymbol{\Theta}, \boldsymbol{\mathcal{M}}) = \sum_{j=1}^{J} \log(\sum_{k=1}^{K} p(y'|\boldsymbol{X}_j; \boldsymbol{\Theta}) p(g_j|y'; \boldsymbol{\mathcal{M}})), \quad (3)$$

where $\mathbf{\Theta}$ is the set of parameters of the network in refining step. Given $\mathbf{\Theta}$, which initialized by the model from the learning step, we can predict the skeleton joints $\boldsymbol{B}_t$ of a depth frame. We concatenate the predicted skeleton of every frame to a single vector $\boldsymbol{B} = \{\boldsymbol{B}_1, ..., \boldsymbol{B}_{T_n}\}$. y' is then calculated as a fully connected layer: $\boldsymbol{y'} = \boldsymbol{W}^{y'}\boldsymbol{B} + b^{y'}$. $\boldsymbol{W}^{y'}$ and $b^{y'}$ is part of $\mathbf{\Theta}$, but they are trained from scratch. During the training of the refining step, our model aims to maximize the likelihood function by optimizing both the bridging matrix and network parameter iteratively in an EM procedure.

Estimating latent PI The explicit expression of latent PI is as follows:

$$\begin{aligned} u_k &= p(y'_k|\boldsymbol{B}, g; \boldsymbol{W}_{y'}, \boldsymbol{\mathcal{M}}) \\ &= \frac{p(g|y'_k; \boldsymbol{\mathcal{M}})p(y'_k|\boldsymbol{B}; \boldsymbol{W}_{y'})}{\sum_{l=1}^{K} p(g|y'_l; \boldsymbol{\mathcal{M}})p(y'_k|\boldsymbol{B}; \boldsymbol{W}_{y'})} \\ &= \frac{\boldsymbol{\mathcal{M}}_{kg} \exp(\boldsymbol{W}_k^{y'}\boldsymbol{B} + b_{y'})}{\sum_{l=1}^{K} \boldsymbol{\mathcal{M}}_{lg} \exp(\boldsymbol{W}_k^{y'}\boldsymbol{B} + b_{y'})} \end{aligned} \tag{4}$$

$p(y'_k|\boldsymbol{B}; \boldsymbol{W}_{y'})$ is a predicted probability of the class k by observing the predicted skeleton joints $\boldsymbol{B}$ of a input depth sequences. The bridging matrix $\boldsymbol{\mathcal{M}}$ aims to transform the predicted distribution to a latent distribution that can be effectively used in optimizing the network.

Updating model with latent PI The distribution of latent PI $p(\hat{\boldsymbol{u}}_j)$ of an input sequence $\boldsymbol{X}_j$ is defined by $p(\hat{\boldsymbol{u}}_j) = \boldsymbol{u}_j \boldsymbol{z}_t$, where $\boldsymbol{z}_t \in \mathbb{R}^K$ is randomly generated for each frame t from a Multinoulli distribution $\{\hat{g} \sim \mathcal{P}(\alpha), z_{\hat{g}} = 1, z_l = 0, \forall l \neq g\}$, where $\mathcal{P}(\alpha)$ is defined as $p_g = 1 - \frac{K-1}{K}\alpha$ and $p_l = \frac{1}{K}\alpha$, where α is to control how strongly the prior distribution is pushed to classification loss, and g is the groundtruth label. We replace the groundtruth label by the probabilities of latent PI to formulate the PI-based classification loss in refining step:

$$\begin{aligned} \mathcal{L}^{\mathrm{R}} = -\sum_{j=1}^{J} \Bigg(&\sum_{k=1}^{K} p(\hat{u}_{jk}) \log p(y_{jTk}) \\ &-\beta \sum_{k=1}^{K} \delta(k - g_j) \log p(y'_{jk}) \Bigg) \end{aligned} \tag{5}$$

a standard softmax loss is also included in $\mathcal{L}^{\mathrm{R}}$ to update the parameters (*e.g.* $\boldsymbol{W}^{y'}$, $b^{y'}$) from the branch of secondary task. Apart from optimizing network parameters, the bridging matrix of modeling latent PI can be updated iteratively by a closed-form solution in the M-step of EM procedure [31, 36, 3]:

$$\boldsymbol{\mathcal{M}}_{kl}(\Omega) = \frac{\sum_{j=1}^{J} u_{jk}\delta(l - g_j)}{\sum u_{jk}}, \qquad k, l \in \{1, ..., K\} \tag{6}$$

Algorithm 1: PI-based RNNs

Input: A collection of videos $\mathcal{V}$, skeleton joints annotation $\boldsymbol{E}$, subset of skeleton joints $\hat{\boldsymbol{E}}$, groundtruth class label g.
Output: Network parameters, bridging matrix $\boldsymbol{\mathcal{M}}$
Pre-training:
  Eq.1 taking both $\boldsymbol{x}$ of depth sequences V and skeleton joints $\boldsymbol{E}$,
  A encoder is trained by minimize the standard softmax loss.
end
Learning:
  Taking the subset of skeleton joints $\hat{\boldsymbol{E}}$ in the regression term.
  The parameters of network are optimized by minimizing the multi-task loss Eq. 2
end
Refining:
  while *not converge* **do**
    E-step:
      Estimating and updating the latent PI by Eq. 4
    end
    M-step:
      The parameters of network are optimized by PI-based classification loss Eq. 5.
      The bridging matrix $\boldsymbol{\mathcal{M}}$ is updated with Eq. 6
    end
  end
end

Discussion on latent PI Latent PI can be treated as a sufficient information to act as a teacher network [24, 40]. However, our latent PI is obtained in the same framework rather than trained from a separate model. Our model further refines latent PI according to the feedback of the network in each iteration. This updating process us two benefits: (1) The formulation strikes a good balance between the class distributions learned from depth appearance and skeleton information. This is similar in spirit to [35], where a weight distribution is utilized to improve the learning process of random forest. Sun *et al*. [38] also incorporate prior information (*e.g*. human height) to enhance the dependency between output variables and latent variables, where the prior can help to split data effectively. The skeleton and raw depth sequence should share relevant and complementary information. Here, we measure the loss by partially considering the posterior obtained from skeleton joints. We show that this learning process improves the discriminative power of the network. (2) Apart from learning better depth representation, our PI-based classification loss provides an effective way to prevent overfitting. Since the prior label is not per-

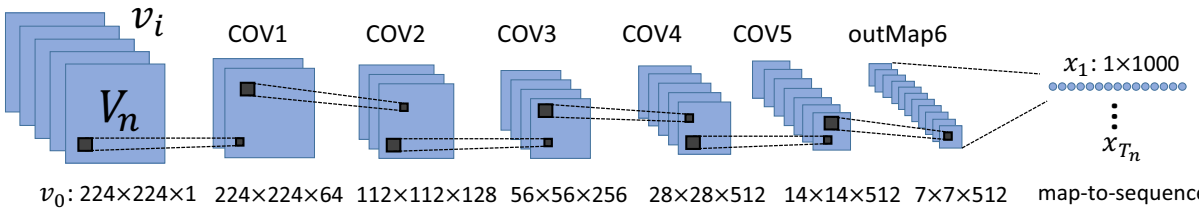

Figure 2: The architecture of the encoder. The convolutional layers (from COV1 to COV5) with kernel size 3×3 and a stride of 1. The padding implements same convolution (and pooling), where the input and output maps have the same spatial extent. max-pooling is performed from COV1 to COV5 over 2×2 spatial windows with stride 2.

fectly trained, the noise is introduced when we switch to the prior label according to α. This term can be treated as a regularizer similar as [53], where they intentionally generate incorrect training labels at the loss layer. Our loss function also seeks to minimize the confusion between the two distributions.

4.4. Model Training

We summarize the whole training process of the proposed PI-based RNNs in Algorithm 1. Note that the learning step and refining step can be potentially preformed alternatively to improve the effectiveness of the trained model. In our experiments, we show that one round of learning and refining step achieves significant improvements. While small improvements can be further obtained with more rounds, which has been verified on SBU dataset, we fix to one round of learning and refining for all experiments with the good trade-off between accuracy and efficiency. In refining step, the EM procedure is still run iteratively until convergence.

For all three steps, the error differentials measured by the last layer of the recurrent neural network will be back-propagated to feature sequences and feed back to the convolutional layers across every frame in the videos. Our approach is an end-to-end trainable network that jointly learns the parameters of the CNN and the RNN. We train each model with stochastic gradient descent on the negative log-likelihood using the Adam optimizer, with a learning rate of 0.001 for MSR Action3D and 0.0001 for the rest. A mini-batch size of 10 is applied to all datasets. We use early stopping when the validation error starts to increase.

5. Experiments

We compare the performance of our model with state-of-the-art methods and baselines on four datasets: MSR Action3D Dataset [26] (Action3D), SBU Interaction dataset [59] (SBU), Cornell Activity Dataset [39](CAD60), and the proposed Blanket dataset (Blanket). We also analyze each component of our model and the computational efficiency.

Datasets: Action3D is an action dataset of depth sequences captured by a depth camera. This dataset consists of 20 actions performed by 10 subjects. Every action was performed by ten subjects three times each. All sequences are captured in 15 FPS, and each frame in a sequence contains 20 skeleton joints. Altogether, the dataset has 557 valid action sequences with 23797 frames of depth maps. **SBU** consists of 282 pre-segmented sequences, which includes 8 classes depicting two-person interaction. Each action is performed by 21 pairs of subjects. **CAD60** consists of 68 video clips captured by Microsoft Kinect device. Each video is of length about 45s. Four different subjects performed 14 different activities in five locations: office, kitchen, bedroom, bathroom and living room. **Blanket** contains 120 depth video clips. There are 12 different action classes performed by 10 subjects. Our dataset contains more static actions (*e.g.* lying and sitting). This dataset is very challenging, as some actions are partially occluded by a blanket. For example, one actor is sitting on the bed while he is covered by a blanket (please refer to our supplementary video for all actions).

Implementation details: We implemented the network using TensorFlow [1]. The architecture of convolutional layers (see Fig. 2) is slightly modified from VGG-VeryDeep [37] (with 11 weight layers) for depth maps. We initialize the weights without pre-training by using the normalized initialization procedure [13]. Unlike images which can be rescaled and randomly cropped to a fixed size, spatio-temporal consistency has to be considered for video sequences. Each input video frame is scaled to 227x227 from the whole frame. We did not perform the operation of randomly cropped and flipped for utilizing PI easily. The depth values are normalized to [-1,1]. Our model has a stack of 2 LSTMs of 1000 hidden units each. To reduce the computation cost, we sample each video of CAD60 with a maximum length of 200 frames. We do not sample frames from MSR Action3D, SBU and Blanket dataset. We unroll the LSTM to a maximum length of 200 time steps for CAD60, 300 time steps for Blanket and 100 time steps for the rest during training, which is a good trade-off between accuracy and complexity.

We mainly consider the skeleton joints as our PI. The prior class distribution is obtained by training DURNN-L [9] with all available skeletons. In our regression loss, we

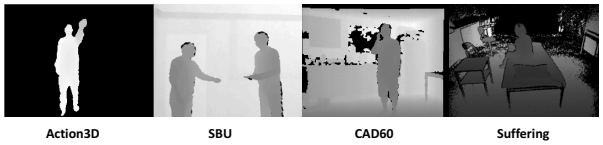

Figure 3: Examples of depth maps on four datasets.

use only six joints (*i.e*. head, hand left, hand right, foot left, foot right, hip center) as this secondary target is formulated for helping classification accuracy. For our Blanket dataset, we annotate the six joints for both pre-training and refining stage because of the special camera view-point. We normalize 3D joint coordinates to a unified coordinate system from the world coordinate system by placing the hip center at the origin [43]. Similar as [9], we apply a simple Savitzky-Golay smoothing filter to smooth the skeleton annotations.

5.1. Comparison to the State-of-the-art

The experimental results are shown in Table 1. Existing state-of-the-art methods can be partitioned into two groups: using 1) only the depth sequences or 2) at least skeleton information in the testing stage.

Results on Action3D : We follow a similar evaluation protocol from [45, 46]. In this setting, the dataset is divided into two sets, where half of the subjects are used for training and the other half are used for testing. Compared to another protocol [9] that splits classes into three subsets, this setting is more challenging as all actions are evaluated together. The average accuracy corresponds to the mean of the confusion matrix diagonal of all classes. Note that 10 skeleton sequences were not used [47] because of missing data. We compare the proposed model PRNN with Xia *et al*. [52], Oreifej *et al*. [34], and Yang *et al*. [56]. All theses methods require only depth maps as input during testing. We can see that our proposed PRNN achieves the best average accuracy (94.9 %) compared with them. For a complete comparison, we also list those skeleton-based approach in the lower part of the Table 1. Skeleton-based approaches demonstrate slightly better performance by assuming a robust skeleton tracker is available in testing. Our method aims to provide a more general framework allowing us to learn the model directly from raw observations of depth videos, rather than explicitly modeling skeletal joints [9] or local appearance [42]. Many of these methods either focus on modeling spatio-temporal structure with a certain assumption [34], or exploit the trajectories of human joints [42, 60] in the testing stage which rely on accurate skeleton joints detection.

SBU : We follow the experimental setting of [59, 61] and use five-fold cross-validation. All action categories are composed of interactions between actors, involving human acting and reacting. This dataset is very challenging, especially in our setting where skeleton information is not available in testing. We summarize the results in Table 1. We can see that our method achieves superior performance to the depth-based approaches and perform close to the skeleton-based approaches.

	Method	Action3D	SBU	CAD60	Blanket
depth	Xia *et al*. [52]	89.3	43.69	-	40.6
	Oreifej *et al*. [34]	88.9	77.0	72.7	42.8
	Yang *et al*. [57]	93.45	-	-	41.2
	PRNN	**94.9**	**89.2**	**87.6**	**53.5**
skeleton	Vemulapalli *et al*. [43]	89.48	-	-	-
	Veeriah *et al*. [42]	92.03	-	-	-
	Hu *et al*. [20]	-	-	**84.1**	-
	Koppula *et al*. [23]	-	-	71.4	-
	Du *et al*. [9]	-	80.35	-	-
	Wang *et al*. [48]	**96.9**	-	-	-
	Wang *et al*. [45]	91.40	-	-	-
	Zhu *et al*. [61]	-	90.41	-	-
	Gori *et al*. [15]	95.38	**93.08**	-	-
	Wang *et al*. [47]	88.2	-	74.7	-

Table 1: Comparison with state-of-the-art methods on four datasets for action recognition. '-' indicates no result was reported and no code is available for implementation.

CAD60 : We follow the same experimental setting as in [47, 20] by adopting the leave-one-person-out cross validation. *i.e*. the model was trained on three of the four people, and tested on the fourth. Table 1 compares the results on CAD60. We can see that the proposed PRNN achieves the 87.6% accuracy with only seeing the depth maps, comparing against previous works which utilize multiple cues (*i.e*. RGB frames, depth maps and the tracked skeleton joint positions) in testing. Some different human actions of CAD60 share similar body motions such as "chopping" and "stirring". Our model takes advantage of the PI-based learning process, which allows to distinguish the subtle motions from depth maps [21].

Blanket: Similar to CAD60, we follow the protocol as [47] and perform cross-validation on our proposed dataset. We compare our model with three baseline methods: Xia *et al*. [52], Oreifej *et al*. [34], Yang *et al*. [57]. We use their publicly available codes and train their model with varying their parameters, so as to report the best results for fair comparison. The experimental results are shown in Table 1. The proposed PRNN obtains the state-of-the-art accuracy of 53.5%. Our collected data is more difficult to learn than the existing dataset. Although each basic action is simple like "sitting" and "lying down", the actor (*i.e*. patient) is either partially occluded by a blanket or in a suffering status when he performs these actions. It introduces severe noise (*e.g*. shaking his body, trembling) to the basic actions. Moreover, this special camera view-point (see Figure 3) and the occlusion by a blanket will cause difficulties for skeletal estimation. As expected, a larger performance gap is seen between our model and other approaches. This demonstrates the potential of our model in representing and modeling the dynamics of actions directly from depth maps.

In brief, we show the competitive performance of the

Method	Action3D	SBU	CAD60	Blanket
CNN-RNN (vanilla)	87.3	79.2	81.5	37.8
PRNN-NoPreTrain	89.2	85.6	78.6	47.8
PRNN-NoRefine	83.4	71.6	70.5	40.3
PRNN	**94.9**	**89.2**	**87.6**	**53.5**

Table 2: Contribution of each model component

proposed PRNN on four human action datasets. Our model provides an effective end-to-end solution for modeling temporal dynamics in action sequences by exploiting the PI in training time. Unlike most of the previous works that are based on a certain assumption about the structure of the depth maps or the availability of a robust skeleton tracker, our model automatically learns features from raw depth maps irrespective of any assumptions [58, 51] on the structure of video sequences .

5.2. Model Analysis

Evaluation of individual components To verify the effect of individual components in our framework and demonstrate that if each of them contributes to the performance boost, we evaluate three variants of our approach: (1) PRNN-NoPreTrain discards the pre-training strategy as shown in Sec. 4.1. Instead, the CNN encoder is trained from the scratch in the learning stage. (2) PRNN-NoRefine ignores the last refining step as described in Sec. 4.3. The final model is trained by pre-train and learning steps. Note that the learning step in Sec. 4.2 can not be removed individually, because the latent PI is obtained based on the regression term of the learning step. We report the performance of a vanilla CNN-RNN pipeline. This is similar to our model in pre-training step, except that skeleton is not a part of the input during training. Note that our pre-training stage (taking both depth and skeleton as input) is specifically designed for our learning stage (with classification and regression loss). We tried to initialize vanilla CNN-RNN (depth input with classification loss) with our pretrained model. It performs much worse than learning from scratch.

We show the average accuracy of all stripped-down versions of our model in Table 2. Overall, our method consistently achieves better performance with integrating each individual component, suggesting that each one of them contributes to the final performance. Without exploiting PI in the pre-training step, our model performs poorly due to the ineffective initialization. The vanilla CNN-RNN also suffers from the relatively small number of training data, and thus cannot take full advantage of the end-to-end manner. By considering the latent PI information in the refining step, this overfitting problem can be greatly alleviated from CNN-RNN and PRNN-NoRefine. It is clear that the performance has been substantially improved (PRNN) when combining these steps together.

Qualitative analysis We compare our approach with three variants in Figure 4, which illustrates the real-time prediction of an example sequence on every 15 time steps. The groundtruth action label is "falling from the bed". All methods give a low confidence to the correct action class at the beginning. As time evolves, we find that our approach first correctly predict the action labels. We attribute this faster learning ability to the mechanism of encoding PI [2], which allows us to distinguish the subtle depth difference across successive frames.

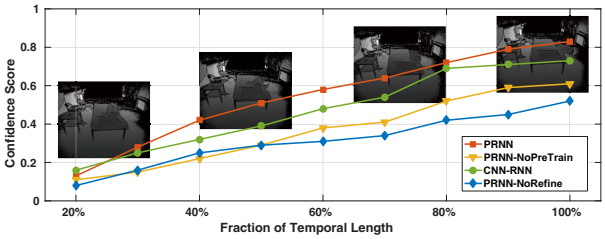

Figure 4: Qualitative comparison of real-time prediction as time evolves for the action "falling from the bed"

Computational efficiency We take the Action3D as an example to discuss the efficiency of our approach. With Python and C++ implementation on a NVIDIA Titan X GPU, our three-steps learning process takes about 11 hours to converge after continuously decreasing over 200k SGD iterations. Gradients are averaged over each minibatch in every training iteration. During testing, it can achieve real-time performance ($\approx$ 38 FPS). Compared with multi-stage models, the efficiency of our approach is mainly attributed to its end-to-end property without preprocessing step. Please refer to our supplementary video for real-time testing performance.

6. Conclusion and Future Work

In this paper, we propose to learn a recurrent neural network with PI. The presented learning process provides threefold benefits: 1) The pre-training stage provides a mid-level embeddings which can be effectively tuned in the further stage. 2) In learning stage, a multi-task loss is formulated to exploit PI as a secondary task. 3) The learned information is further modeled to a latent PI, which is defined to close the gap between two tasks. The latent PI is used to enhance the discriminative power of the learned representation by closing two distributions. The latent PI is also updated iteratively in an EM fashion. In addition, the randomly sampled classification loss operates as a regularizer to reduce the tendency for overfitting. We apply our model to the problem of action recognition from depth sequences, and achieve better performance on three publicly available datasets and our newly collected dataset. In the future, we will consider to investigate more different types of PI and seek to model this information in the intermediate level of neural network [5].

Acknowledgement: This work was supported by the Omron Corporation.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 2015.

[2] S. Baek, K. I. Kim, and T.-K. Kim. Real-time online action detection forests using spatio-temporal contexts. In *WACV*, 2016.

[3] A. J. Bekker and J. Goldberger. Training deep neural-networks based on unreliable labels. In *ICASSP*, 2016.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *TNN*, 1994.

[5] Çağlar Gülçehre and Y. Bengio. Knowledge matters: Importance of prior information for optimization. *JMLR*, 2016.

[6] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *NIPS*, 2015.

[7] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool. Real-time Facial Feature Detection using Conditional Regression Forests . In *CVPR*, 2012.

[8] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[9] Y. Du, W. Wang, L. Wang, and . Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015.

[10] S. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

[11] J. Feyereisl, S. Kwak, J. Son, and B. Han. Object localization based on structural svm using privileged information. In *NIPS*. 2014.

[12] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[14] D. Gong, G. Medioni, and X. Zhao. Structured time series analysis for human action segmentation and recognition. *TPAMI*, 2014.

[15] I. Gori, J. K. Aggarwal, L. Matthies, and M. S. Ryoo. Multitype activity recognition in robot-centric scenarios. *IEEE Robotics and Automation Letters*, 2016.

[16] M. A. Gowayyed, M. Torki, M. E. Hussein, and M. El-Saban. Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition. In *IJCAI*, 2013.

[17] A. Graves. *Supervised sequence labelling with recurrent neural networks*. Springer, 2012.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[19] J. Hoffman, S. Gupta, and T. Darrell. Learning with side information through modality hallucination. In *CVPR*, 2016.

[20] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *CVPR*, 2015.

[21] C. Jia, G. Zhong, and Y. Fu. Low-rank tensor learning with discriminant analysis for action classification and image recovery. 2014.

[22] P. Koniusz, A. Cherian, and F. Porikli. Tensor representations via kernel linearization for action recognition from 3d skeletons. In *ECCV*, 2016.

[23] H. S. Koppula, R. Gupta, A. Saxena, and . Learning human activities and object affordances from rgb-d videos. *IJRR*, 2013.

[24] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling. Bayesian dark knowledge. In *NIPS*, 2015.

[25] A. Krizhevsky, Sutskever, Ilya, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[26] W. Li, Z. Zhang, and Z. Liu. Action Recognition Based on A Bag of 3D Points. In *CVPRW*, 2010.

[27] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*, 2016.

[28] Q. Liu, S. Wu, L. Wang, and T. Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. 2016.

[29] J. Luo, W. Wang, and H. Qi. Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps. In *ICCV*, 2013.

[30] B. Mahasseni and S. Todorovic. Regularizing long short term memory with 3d human-skeleton sequences for action recognition. In *CVPR*, June 2016.

[31] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[32] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[33] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, 2016.

[34] O. Oreifej and Z. Liu. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *CVPR*, 2013.

[35] S. Schulter, P. Wohlhart, C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. Alternating decision forests. In *CVPR*, 2013.

[36] Z. Shi, T. M. Hospedales, and T. Xiang. Bayesian joint modelling for object localisation in weakly labelled images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[38] M. Sun, P. Kohli, and J. Shotton. Conditional Regression Forests for Human Pose Estimation. In *CVPR*, 2012.

[39] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured Human Activity Detection from RGBD Images. In *ICRA*, 2012.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints*, 2015.

[41] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *NN*, 2009.

[42] V. Veeriah, N. Zhuang, G.-J. Qi, and . Differential recurrent neural networks for action recognition. In *ICCV*, 2015.

[43] R. Vemulapalli, F. Arrate, R. Chellappa, and . Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*, 2014.

[44] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. Campos. STOP: Space-Time Occupancy Patterns for 3D Action Recognition from Depth Map Sequences. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2012.

[45] C. Wang, J. Flynn, Y. Wang, and A. Yuille. Recognizing actions in 3d using action-snippets and activated simplices. In *AAAI*, 2016.

[46] C. Wang, Y. Wang, and A. L. Yuille. Mining 3d key-pose-motifs for action recognition. In *CVPR*, 2016.

[47] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Learning Actionlet Ensemble for 3D Human Action Recognition. *TPAMI*, 2014.

[48] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li. Beyond covariance: Feature representation with nonlinear kernel matrices. In *ICCV*, 2015.

[49] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. Ogunbona. Action Recognition from Depth Maps Using Deep Convolutional Neural Networks. In *IEEE Transactions on Human Machine Systems*, 2015.

[50] Z. Wang and Q. Ji. Classifier learning with hidden information. In *CVPR*, June 2015.

[51] S. F. Wong, T.-K. Kim, and R. Cipolla. Learning motion categories using both semantic and structural information. In *CVPR*, 2007.

[52] L. Xia and J. K. Aggarwal. Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera. In *CVPR*, 2013.

[53] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian. DisturbLabel: Regularizing CNN on the Loss Layer. In *CVPR*, 2016.

[54] M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *NIPS*. 2013.

[55] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992.

[56] X. Yang and Y. Tian. Super Normal Vector for Activity Recognition Using Depth Sequences. In *CVPR*, 2014.

[57] X. Yang and Y. Tian. Super normal vector for human activity recognition with depth cameras. *TPAMI*, 2016.

[58] T.-H. Yu, T.-K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forest. In *BMVC*, 2010.

[59] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *CVPRW*, 2012.

[60] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In *ICCV*, 2013.

[61] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie. Co-occurrence Feature Learning for Skeleton based Action Recognition using Regularized Deep LSTM Networks. In *AAAI*, 2016.