Convolutional Memory Blocks for Depth Data Representation Learning

Keze Wang[1,2], **Liang Lin**[1,3*], **Chuangjie Ren**[1], **Wei Zhang**[3], **Wenxiu Sun**[3]
[1] Sun Yat-sen Univeristy, [2] The Hongkong Polytechnic University, [3] Sensetime Group Limited

Abstract

Compared to natural RGB images, data captured by 3D / depth sensors (e.g., Microsoft Kinect) have different properties, e.g., less discriminable in appearance due to lacking color / texture information. Applying convolutional neural networks (CNNs) on these depth data would lead to unsatisfying learning efficiency, i.e., requiring large amounts of annotated training data for convergence. To address this issue, this paper proposes a novel memory network module, called Convolutional Memory Block (CMB), which empowers CNNs with the memory mechanism on handling depth data. Different from the existing memory networks that store long / short term dependency from sequential data, our proposed CMB focuses on modeling the representative dependency (correlation) among non-sequential samples. Specifically, our CMB consists of one internal memory (i.e., a set of feature maps) and three specific controllers, which enable a powerful yet efficient memory manipulation mechanism. In this way, the internal memory, being implicitly aggregated from all previous inputted samples, can learn to store and utilize representative features among the samples. Furthermore, we employ our CMB to develop a concise framework for predicting articulated pose from still depth images. Comprehensive evaluations on three public benchmarks demonstrate significant superiority (about 6%) of our framework over all the compared methods. More importantly, thanks to the enhanced learning efficiency, our framework can still achieve satisfying results using 50% less training data.

1 Introduction

With the rapid development of inexpensive commodity depth sensors, depth data representation learning is ubiquitous in many applications such as robotic systems [McColl *et al.*, 2011]. Compared to RGB data which provides information about appearance and texture, depth data, reflecting the distance information between the objects and the sensor, are less discriminable. Recently, deep convolutional neural networks (CNNs) have been applied by many methods [Haque *et al.*, 2016; Wang *et al.*, 2016] for depth data analysis. Due to the heavy sensor noise of depth data and the huge parameters of used deep CNNs, these methods require plenty of well annotated samples to train from scratch to achieve the satisfactory performance. However, collecting and annotating on depth data is extremely laborious and time-consuming. It will be beneficial to design a more intelligent network architecture with better learning efficiency on depth data, i.e., to surpass the state-of-the-art performance even with less training data.

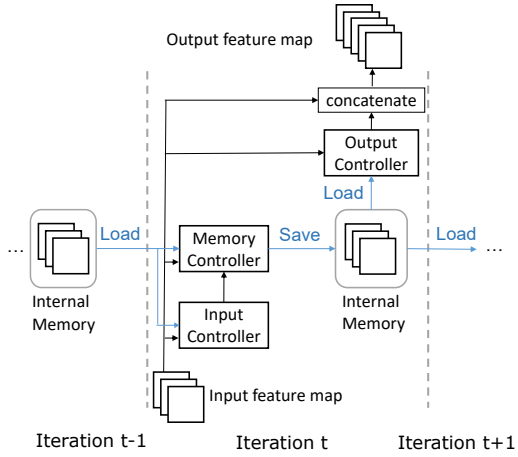

Figure 1: Detailed architecture of our proposed Convolutional Memory Block (CMB). The CMB consists of one internal memory (i.e., a set of feature maps) and three specific convolutional controllers, which can manipulate the internal memory and extract implicit structural representation from the input feature map. Specifically, the old internal memory from the previous training iteration is loaded by the input and memory controller. Then, the memory controller fuses its memory representation and the response of the input controller, and saves the fused representation to be the new internal memory. After that, the output controller loads the new internal memory to generate memory representation, which is further concatenated with the input feature map to be the final output.

In this paper, we propose a novel memory network module, called Convolutional Memory Block (CMB), inspired by the recent work of Neural Turing Machine [Graves *et al.*, 2014]. Considering the special properties of depth data, e.g., low discriminability in appearance due to lacking color / texture information, we leverage the memory mechanism to enhance

*Corresponding author is Liang Lin (Email: linliang@ieee.org).

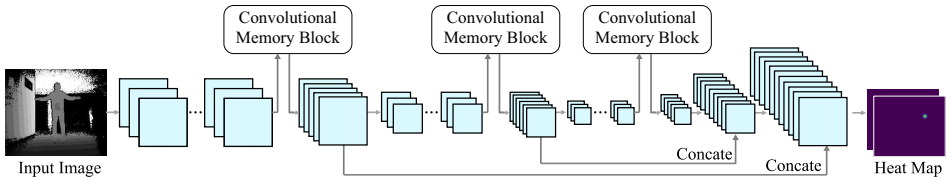

Figure 2: An overview of the proposed convolutional memory block embedded framework for estimating articulated poses.

the pattern abstracting of CNNs by reusing their rich implicit convolutional structures and spatial correlations among the training samples. Specifically, the proposed CMB consists of three specific convolutional controllers and one internal memory (i.e., a set of feature maps) as shown in Fig. 1. The convolutional controller is designed to process the input feature map from the previous layer and manipulate the internal memory. Different from ConvLSTM [Shi *et al.*, 2015] and ConvGRU [Ballas *et al.*, 2016] that require time-series data, the proposed convolutional controller performs convolution in a hierarchical organization via several convolutional layers with batch normalization. This ensures that our proposed CMB is capable of extracting more abstract information from non-sequential training samples to augment image-dependent feature representation. Specifically, our CMB intends to capture and store the representative dependencies or correlations among training samples according to specific learning tasks, and further employ these stored dependencies to enhance the representation of convolutional layers. In this way, our CMB encourages the CNN architecture to be lightweight and require less training data.

Since articulated (e.g., human body or hand) pose estimation is one of the most dominant applications in depth data representation learning, we develop a simple yet effective articulated pose estimation framework to validate the effectiveness of our CMB by applying it to enhance the convolutional layers. Recently, highly accurate and real-time performance on human pose estimation from depth data has been achieved by deep learning based methods [Haque *et al.*, 2016; Wang *et al.*, 2016]. Nevertheless, all methods borrow CNN architectures from RGB data analysis. None of them considers how to improve the learning efficiency in the perspective of handling depth images. Motivated by the design principles from stacked hourglass [Newell *et al.*, 2016], our developed framework has a lightweight hourglass-like architecture, as illustrated in Fig. 2. Our CMB contributes to collaboratively regress the heat map of each joint by providing the cached representative feature maps, which are aggregated from previous inputted samples.

The **main contributions** of this work are summarized as follows: (i) we propose a novel Convolutional Memory Block (CMB) to promote the representation and learning efficiency of convolutional layers for handling depth data; (ii) we apply our CMB to develop a simple yet effective framework for articulated pose estimation from depth images. Extensive experiments on three public benchmarks not only demonstrate the superiority of our framework over all the compared methods, but also prove our framework can obtain comparable performance with much less training data.

2 Related Work

Neural Networks with Memory. To model the temporal dynamics and dependency, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Unit (GRU) [Cho *et al.*, 2014], have been proposed and achieved the remarkable performance on many vision tasks (e.g., semantic parsing [Liang *et al.*, 2016]). More recently, the Convolutional Long Short-Term Memory model (ConvLSTM) [Shi *et al.*, 2015] and Convolutional Gated Recurrent Unit (ConvGRU) [Ballas *et al.*, 2016] have been proposed to consider the correlations among neighboring pixels in the spatial domain by enabling convolutional operations among its gates, hidden states and memory cells.

Besides above-mentioned models, various memory mechanisms have been proposed to enable neural networks to model sequential data by explicitly remembering variables and data over long timescales. The existing memory network models can be divided into content-based addressing and location-based addressing according to their accessing memory manners. The location-based addressing (e.g., [Graves *et al.*, 2014]) leverages a module so-called controller to receive input data and further store or retrieve valuable information from an external memory via the looked up address, while the content-based addressing (e.g., [Graves *et al.*, 2016]) focuses on reading from or writing to the memory to obtain the relevant memory cells or representations instead. Specifically, Neural Turing Machines [Graves *et al.*, 2014] was first proposed to use an external memory for location addressing to solve some algorithmic problems. [Na *et al.*, 2017] proposed to design the read network and the write network that consist of multiple convolutional layers.

Articulated Pose Estimation from Depth Data. Recently, several works [Shotton *et al.*, 2011; Girshick *et al.*, 2011; Sun *et al.*, 2012; Shotton *et al.*, 2013a; 2013b; Jung *et al.*, 2015] have achieved promising performances on articulated pose estimation such as human and hand pose estimation. [Shotton *et al.*, 2011] designed an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. [Jung *et al.*, 2015] proposed to introduce a regression tree to estimate human poses by applying a supervised gradient descent and MCMC like random sampler in the form of Markov random walks. More recently, improved performances have also been achieved by deep learning based methods [Haque *et al.*, 2016; Wang *et al.*, 2016]. Specifically, [Haque *et al.*, 2016] presented a viewpoint invariant model by combining CNN and RNN with a top-down error feedback mechanism to self-correct previous pose estimates in an end-to-end manner. [Wang *et al.*, 2016] proposed an inference embedded

multi-task learning framework, which is implemented with a deep architecture of neural networks with two cascaded tasks.

The advanced network architectures [Wei *et al.*, 2016; Newell *et al.*, 2016; Grinciunaite *et al.*, 2016] for estimating human poses from RGB images have also been proposed. [Wei *et al.*, 2016] provided a sequential human pose prediction framework for learning rich implicit spatial models. [Newell *et al.*, 2016] proposed to design network architecture by processing features across all scales and consolidate them to best capture the various spatial relationships associated with the body. However, directly applying these architectures to the depth data is unfeasible due to the different challenges and requirements between the RGB data and the depth data. For instance, the depth images usually include heavy sensor noises and preserve coarse appearance details.

3 Convolutional Memory Blocks

As illustrated in Fig. 1, our proposed Convolutional Memory Block (CMB) consists of one internal memory (a set of feature maps) and three convolutional controllers for facilitating the internal memory manipulation. The internal memory is designed to store the learned implicit image-dependent structural features, and is denoted as $M \in \mathbb{R}^{c_m \times h_m \times w_m}$, where c_m, h_m, w_m are the capacity, height, width of the feature map, respectively. Different from the widely used neural controllers [Weston *et al.*, 2015; Park *et al.*, 2017] which use full connections to perform input-to-state and state-to-state transitions, our proposed convolutional controller leverages the convolution operator to process the input feature map from the previous neural layer, and further manipulates the internal memory in both the training and testing phase. Fig. 1 also illustrates three specific convolutional controllers in our proposed CMB, i.e., input controller, memory controller and output controller.

Inspired by the design principles from [Szegedy *et al.*, 2015], our proposed convolutional controllers share the same elaborately crafted structure, which leverages a hierarchical organization of convolutional layers rather than a simple convolutional layer as ConvLSTM [Shi *et al.*, 2015] and ConvGRU [Ballas *et al.*, 2016]. This ensures that the convolutional controller is able to extract rich and high-level implicit structural features. As illustrated in Fig. 3, each controller first employs two 3×3 convolutional kernels to process the input feature map and internal memory, respectively. Then, these two convolution results are added together, and further fed into the followed Batch Normalization (BN) Layer, Rectified Linear Units (ReLU), 1×1 convolutional layer, and another BN layer to obtain the intermediate feature map. For ease of implementation, the channel number and stride of all the convolutional layers are the same. As reported by [Zhang *et al.*, 2018], the 1×1 convolutional layer is imposed to help the information flow across different channels of the feature map. For the convenience of the formulation, we simply denote the operations with the convolutional controller as the function $\phi(\cdot)$. Note that, to simplify the further calculation, the output response of each convolutional kernels all share the same size with the input feature map. For each specific convolution controller, the intermediate feature map is further processed individually. In the following, we will introduce these three specific convolutional controllers in the training and testing phase formally.

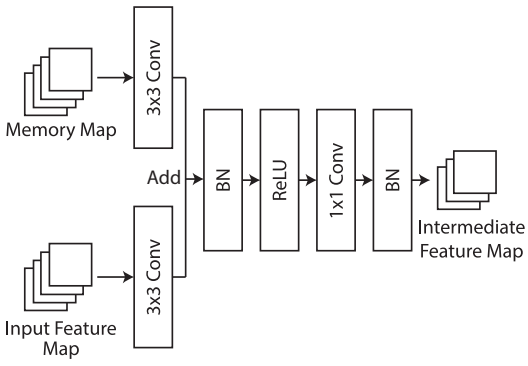

Figure 3: Detailed illustration of our convolutional controller.

3.1 Input Controller

Given the incoming feature map $x \in \mathbb{R}^{c \times h \times w}$, the input controller, denoted as $C_I(\cdot)$, performs three operations in the t-th training mini-batch. Firstly, it converts x to the internal feature representation via its convolutional kernel ω_{ix}; Secondly, it loads the old internal memory M_{t-1} in the previous $(t-1)$-th training iteration via its convolution kernel ω_{im} for the prior guidance; Finally, it adaptively fuses these two feature representations, and further passes the fused result to the memory controller. The whole process can be formulated as follows:

$$C_I(x) = \tanh\left(\phi(\omega_{ix} * x + \gamma_i \cdot \omega_{im} * M_{t-1}; \mathbf{w}_i)\right), \quad (1)$$

where "*" denotes the convolution operator, γ_i is the predefined scalar to control the balance between the input information and old memory. In all our experiments, we empirically set γ_i=1. The function $\phi(\cdot)$ denotes the other operations (i.e., passing through BN, ReLU, 1×1 convolutional layer and BN) illustrated in Fig. 3, where $\mathbf{w}_i$ is the corresponding parameter set for two BN layers and one 1×1 convolutional layer.

3.2 Memory Controller

Motivated by the recent memory network [Santoro *et al.*, 2016], our designed memory controller enables to flexibly write and read complex and abstract information into the internal memory. Specifically, given the incoming feature map x and the output of $C_I(x)$, the memory controller $C_M(\cdot)$ transforms the old memory M_{t-1} into M_t to make it more representative and general for some intended future use in two steps. Firstly, it generate the memory representation $C_M(x)$ for x by using the internal memory M_{t-1} (updated in the previous mini-batch) with the similarity between x and M_{t-1} as weights. Formally, we have:

$$C_M(x) = \sigma\big(\phi(\omega_{mx} * x + \gamma_m \cdot \omega_{mm} * M_{t-1}; \mathbf{w}_m)\big) \circ M_{t-1}, \quad (2)$$

where σ denotes the sigmoid activation function and $\circ$ denotes the Hadamard product. Moreover, ω_{mx} is the weight and bias of the convolutional kernel from C_M for processing input feature map x, while ω_{mm} is the parameter of the convolutional kernel of C_M for the memory-to-memory transition. γ_m is a scalar parameter to balance the response of the new input x and old memory M_{t-1}, and is empirically set as 1 in the experiment. Similar to $\mathbf{w}_i$, $\mathbf{w}_m$ denotes the corresponding parameter set of the memory controller. Once

obtained the memory representation $C_M(x)$ and $C_I(x)$ for the input x, we calculate the new internal memory M_t as:

$$M_t = C_M(x) + \beta \cdot C_I(x), \tag{3}$$

where β is a scalar parameter to interpolate between the currently generated memory representation $C_I(x)$ and the reweighted old memory representation $C_M(x)$. In all our experiments, we empirically set β=1.

3.3 Output Controller

Given the new internal memory M_t, the output controller, denoted as $C_O(\cdot)$, outputs the new memory representation for the input feature map via its two types of convolutional kernels with the parameters $\{\omega_{ox}, \omega_{om}\}$:

$$C_O(x) = \sigma\big(\phi(\omega_{ox} * x + \gamma_o \cdot \omega_{om} * M_t; \mathbf{w}_o)\big) \circ M_t, \tag{4}$$

where σ denotes the sigmoid activation function, $\circ$ denotes the Hadamard product, and γ_o (being empirically set as 1 in the experiment) is a scalar parameter to balance the response of the new input x and new memory M_t. The $\mathbf{w}_o$ is the parameter set for the other operations inside the output controller. Finally, the memory representation C_O is concatenated with the input feature map x and passed to the next neural layer, while the updated internal memory M_t is ready to be loaded for the next training/testing iteration.

3.4 Training and Testing Details

Since all above-mentioned formulations are differentiable, we can directly employ the standard back propagation algorithm [LeCun *et al.*, 1990] to fine-tune the CMB parameters in the training phase. Note that, the internal memory map is randomly initialized at the start of the training and then is uninterruptedly updated without resetting to zero.

In the testing phase, we fix all the parameters inside our CMB except for the internal memory, which has been well optimized during the training and will be further updated by handling the testing samples. Note that, since we expect our CMB to learn to store the informative/representative patterns among samples, the data for both training and testing are randomly shuffled to encourage the internal memory to cache task-specific features via a fully data-driven manner.

3.5 Comparing CMB to ConvLSTM / ConvGRU

Our CMB is entirely different from the ConvLSTM and ConvGRU, although it seems similar to them. The detailed dissimilarities are listed as follows: (i) *Memory Mechanism.* Similar to the LSTM, the ConvLSTM employs the memory cells and hidden states to describe the hidden representation for general-purpose sequence modeling. Therefore, the ConvLSTM requires sequential inputs to obtain a reliable hidden representation during the training and testing phase. The ConvGRU also faces the same issue. Whereas, the internal memory of our proposed CMB adapts to no-sequential inputs by directly fusing with the feature representation of the under-processing sample. Thus, our CMB can be used to store the representative spatial correlations among samples for both training and testing; (ii) *Output Generation.* Unlike the ConvLSTM and ConvGRU that consider the updated hidden states as the final output, our CMB directly leverages the newly updated internal memory to process the input feature map to generate outputs. Moreover, our CMB naturally concatenates the output representation and the input feature map to form a residual representation as [He *et al.*, 2016]. This ensures that all the information can be passed directly through the CMB, i.e., our CMB can provide additional feature map enhancement without corrupting the input feature map.

Table 1: Details of the proposed articulated pose estimation framework. The scalar K is set according to the number of body joints.

Layer Index	**1**	**2**	**3**	**4**	**5**
Layer Name	conv1_1	conv1_2	max1	CMB1	conv2_1
Channel(kernel-stride)	32(3-1)	32(3-1)	32(2-2)	32	64(3-1)
Layer Index	**6**	**7**	**8**	**9**	**10**
Layer Name	max2	CMB2	conv3_1	conv3_2	max3
Channel(kernel-stride)	64(2-2)	64	128(3-1)	128(3-1)	128(2-2)
Layer Index	**11**	**12**	**13**	**14**	**15**
Layer Name	CMB3	conv4_1	conv4_2	neareast upsampling	concatenate conv3_2
Channel(kernel-stride)	128	256(3-1)	256(3-1)	256	384
Layer Index	**16**	**17**	**18**	**19**	**20**
Layer Name	conv5	neareast upsampling	concatenate conv2_1	conv6	conv7
Channel(kernel-stride)	128(3-1)	128	196	64(3-1)	K(1-1)

4 CNN with Convolutional Memory Blocks

To clarify the effectiveness of the proposed CMB, we have developed a concise yet powerful framework for articulated pose estimation by embedding the CMB into an hourglass-shape network (see Fig. 2), which is inspired by designing principles of the architecture in [Newell *et al.*, 2016]. Regarding the articulated pose estimation as a problem of deep regression as [Tompson *et al.*, 2014], our developed framework takes a depth image as input, and outputs a dense heat map for each joint (e.g., the body part of human). Note that, the heat map denotes a per-pixel likelihood of being the joint. The coordinates of maximum value of the predicted heat map will be treated as the center position of the target body joint.

As illustrated in Table 1, our framework is stacked by ten convolutional layers, three max-pooling layers and two concatenate layers to form an hourglass shape as [Newell *et al.*, 2016]. The kernel size of all the convolutional layers are set to 3×3 with a stride of 1, and three max-pooling layer are set to have 2×2 with a stride of 2. The detailed parameters for our framework can be found in Table 1. It is obvious that our framework contains two downsampling-upsampling steps (resulting in three different scales of feature maps, which are denoted in different colors) to capture the various spatial relationships associated with the pose from the depth image. Therefore, We impose three individual CMBs (i.e., no parameters are shared) to enhance these three kinds of feature maps.

5 Experiments

Dataset Description. We have evaluated the estimation performance of our framework on the newly created Kinect2 Human Pose Dataset (K2HPD) [Wang *et al.*, 2016], which includes about 100K clean depth images with various human poses under three challenging scenarios. We have also used the Invariant-Top View Dataset (ITOP) dataset [Haque *et al.*, 2016], which contains large amount of real-world depth images from two different camera viewpoints by 20 actors performing 15 sequences each. Moreover, we have conducted

Table 2: Detailed comparison of the estimation accuracy on the K2HPD benchmark using the PDJ metric.

Method	PHR	CPM	SH	IEML	Ours
PDJ (0.05)	26.8	30.0	41.0	43.2	**52.5**
PDJ (0.10)	70.3	58.5	73.7	64.1	**84.2**
PDJ (0.15)	84.7	87.8	84.6	88.1	**91.7**
PDJ (0.20)	91.3	93.6	89.0	91.0	**95.1**
Average	68.3	67.5	72.1	71.6	**80.9**

Table 3: Detailed comparison of the estimation accuracy on the K2HPD benchmark using the PCKh@0.5 metric.

Method	3DCNN	PHR	CPM	SH	Ours
Head	48.6	83.2	69.9	81.6	**93.2**
Neck	50.9	83.3	71.8	87.3	**97.1**
Shoulders	52.6	82.4	71.1	86.2	**95.0**
Elbows	45.5	71.2	65.7	77.9	**88.4**
Hands	42.1	65.7	65.9	73.4	**84.8**
Writs	42.6	63.4	63.3	72.5	**84.3**
Torso	54.6	80.8	70.4	85.5	**95.6**
Hips	55.0	73.9	66.8	81.2	**91.0**
Knees	50.1	82.7	74.8	86.9	**88.0**
Feet	45.4	81.4	72.7	**91.3**	90.8
Upper Body	47.0	74.2	67.8	79.7	**90.3**
Lower Body	51.5	79.3	70.9	86.0	**91.1**
Full Body	48.9	76.3	69.1	82.3	**90.6**

the experiment on the hand-depth image dataset [Xu and Cheng, 2013] named ASTAR, which consists of 870 depth images of captured by a time-of-flight camera with a data-glove. For a fair comparison on these benchmarks, we follow the same training and testing setting as their officially defined.

Compared Methods. For human pose estimation on the ITOP benchmark, we have compared our framework with Random Forest (RF) [Shotton *et al.*, 2013b], Random Tree Walk (RTW) [Jung *et al.*, 2015], Iterative Error Feedback (IEF) [Carreira *et al.*, 2016], and Viewpoint Invariant (VI) [Haque *et al.*, 2016]. On the K2HPD benchmark, we can compare our framework with Inference Embedded Multi-task Learning (IEML) [Wang *et al.*, 2016], whose results are obtained directly from its paper. In order to justify the advancement of our framework on depth-based human pose estimation, we have also considered the RGB-based human pose estimation approaches. We have made a quantitative comparison with five RGB-based state-of-the-art methods, i.e., 3DCNN [Grinciunaite *et al.*, 2016], Part Heat-map Regression (PHR) [Bulat and Tzimiropoulos, 2016], Convolutional Pose Machines (CPM) [Wei *et al.*, 2016], and Stacked Hourglass (SH) [Newell *et al.*, 2016]. To directly apply the state-of-the-art RGB-based network architectures to handle the depth data (having a single channel), we need to reduce the input channel of these networks from 3 to 1 and train them from scratch on the above-mentioned benchmarks.

Evaluation Metric. To measure the accuracy of predicting human body joints, we employ the popular Percent of Detected Joints (PDJ) metric [Toshev and Szegedy, 2014], Percentage of Correct Key points (PCKh@0.5), and 10cm-rule [Haque *et al.*, 2016] as the evaluation criteria. Specifically, the PDJ metric considers a body joint is correctly estimated only if the distance between the predicted and the true joint position is within a certain fraction of the torso diameter. The PCKh@0.5 metric recognizes the estimated joint position as correct if its distance to the ground truth joint in the image space is within 50% of the head segment length (i.e., the distance between the head and neck joints). The 10cm-rule metric identifies the correct prediction when the distance between the predict joint and the ground truth joint is less than 10cm in the 3D world coordinate defined by Kinect.

Table 4: Detailed comparison of the estimation accuracy on the ITOP (front-view) using the 10cm-rule metric.

Method	RF	RTW	IEF	VI	Ours
Head	63.8	97.8	96.2	**98.1**	97.7
Neck	86.4	95.8	85.2	97.5	**98.5**
Shoulders	83.3	94.1	77.2	**96.5**	75.9
Elbows	73.2	**77.9**	45.4	73.3	62.7
Hands	51.3	70.5	30.9	68.7	**84.4**
Torso	65.0	93.8	84.7	85.6	**96.0**
Hips	50.8	80.3	83.5	72.0	**87.9**
Knees	65.7	68.8	81.8	69.0	**84.4**
Feet	61.3	68.4	80.9	60.8	**83.8**
Upper Body	70.7	**84.8**	61.0	84.0	80.3
Lower Body	59.3	72.5	82.1	67.3	**86.9**
Full Body	65.8	80.5	71.0	77.4	**83.3**

Table 5: Comparison of the average running time (milliseconds per image), model size (MB) and model complexity (GFLOPs) on the K2HPD benchmark. Note that, '-' denotes the result is not available.

Method	PHR	CPM	SH	IEML	Ours
Time	62	72	56	40	14
Model Size	570	525	418	-	58
Complexity	63.1	85.0	30.7	-	5.4

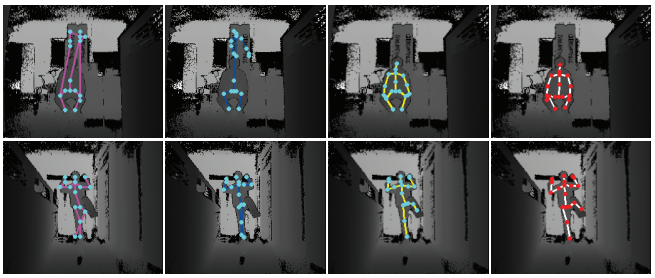

Figure 4: Qualitative comparison between our framework and the compared state-of-the-art methods on the K2HPD benchmark. The estimated joints are directly shown in the images. It is obvious that our framework can obtain much more accurate estimations than the compared methods. Best viewed in color.

Implement Details. All our experiments are carried out on a desktop with Intel 3.4 GHz CPU and NVIDIA GTX-980Ti GPU. In order to reduce overfitting, we employ the image horizontal-flipping and rotating strategy to augment the training data. As for the training process, we train our model from scratch by Adam optimizer [Kendall and Cipolla, 2016] with the batch size of 16 and the initial learning rate of 0.00025, β_1=0.9, β_2=0.999. An exponential learning rate decay is applied with a decay rate of 0.95 every 1000 training iterations.

5.1 Results and Comparisons

Table 2 demonstrates the comparison results under both the PDJ and PCKh@0.5 on the K2HPD benchmark. As one can see from Table 2, our framework significantly outperforms all the compared state-of-the-art methods under every normalized precision threshold. This validates that our framework is more suitable for estimating human poses from depth data, compared with those state-of-the-art methods for RGB data. The similar significant performance gain can also be observed in Table 3. Specifically, our framework has obtained the highest accuracy on all types of body joints. Some visual

comparison results are shown in Fig. 4.

The comparison results on the ITOP dataset (front-view) are illustrated in Table 4. As shown, it is obvious that our framework dramatically surpasses all the compared methods on the estimation accuracy of full body by a clear margin, and is significantly superior to others. Note that, since the prediction of our framework is 2D, we build upon the heat map layer by two fully connected layers with 1024 neurons to regress 3D predictions.

To further evaluate our framework on the small-scale data, we have conducted the 3D hand pose estimation experiment on the ASTAR benchmark, which only provides 435 images for training. The median/mean joint error is regarded as evaluation metric, and obtained after submitting the estimated hand pose coordinates to the official evaluation server. The median/mean joint errors of our framework is 10.02/11.42mm, and is nearly 100% better than the current state-of-the-art method [Xu *et al.*, 2015], which only achieves 21.1/22.7 mm. This validates that our framework is general to the small-scale hand pose estimation task.

To verify the contribution of our CMB to make network lightweight, we further quantitatively perform real-time analysis of our framework and the compared approaches (except for 3DCNN, which requires sequential data) on the K2HPD dataset. As demonstrated in Table 5, our method runs about 70fps, and performs about 4 times faster than the best of the compared approaches. The reason is that the model size of our method is only 58MB with the complexity 5.4GFLOPS, nearly 10, 6 and 5 times smaller than the compared PHR, CPM and SH, respectively. This demonstrates that the superior performance of our framework, thanks to the employed lightweight architecture with moderate parameters.

5.2 Component Analysis

To perform the detailed component analysis of our framework, we have conducted the following experiment on the K2HPD benchmark to validate the contributions of the introduced CMB. Specifically, we have discarded all the CMBs inside our proposed framework, and directly employ the convolutional layers to estimate human poses. This variant version of our framework reflects the pure performance of the fully convolutional networks, and is denoted as "Ours w/o CMB". We have also conducted two variants of our framework by replacing the convolutional memory block with ConvLSTM [Shi *et al.*, 2015] and ConvGRU [Ballas *et al.*, 2016], and denote them as "Ours w/ ConvLSTM" and "Ours w/ ConvGRU", respectively. Note that, these two variants require sequential data for training and testing, i.e., the action sequence for each subject are sequentially fed into the network without shuffling. Given the same sequential data, we have also trained and tested our framework (denoting as "Ours w/ Sequence". To further analyze the performance of the internal memory capacity of the CMB, we have modified the feature map number of the internal memory inside the CMBs. Specifically, the "Ours w/ Half" denotes the variant of Ours that the feature map number is decreased to half of its original, while the "Ours w/ Double" denotes the channel number is doubled. The original memory capacity of CMB is the same as the channel of the input feature map from the previous layer.

Table 6: Detailed comparison of the estimation accuracy for component analysis on the K2HPD benchmark using the PCKh@0.5 metric. The entries with the best values for each row are bold-faced.

Method	Ours w/o CMB	Ours w/ ConvGRU	Ours w/ ConvLSTM	Ours w/ Sequence	Ours w/ Half	Ours w/ Double	Ours
Head	91.9	94.3	91.2	**95.5**	93.5	94.0	93.2
Neck	95.4	92.7	95.3	96.5	96.5	96.9	**97.1**
Shoulders	92.7	91.1	92.8	94.1	94.1	**95.1**	95.0
Elbows	84.6	85.6	86.3	89.2	88.5	**89.7**	88.4
Hands	81.5	80.1	82.3	85.5	82.3	**85.9**	84.8
Writs	79.9	80.6	81.8	84.1	81.9	**85.3**	84.3
Torso	89.2	92.3	92.8	94.9	94.9	**96.0**	95.6
Hips	74.3	86.3	85.2	89.2	90.2	**91.7**	91.0
Knees	80.8	78.4	85.2	88.0	89.2	**89.8**	88.0
Feet	90.2	87.0	90.9	88.1	91.0	**91.8**	90.8
Upper Body	87.4	87.0	88.1	90.5	89.1	**91.0**	90.3
Lower Body	82.7	85.6	88.0	89.7	91.0	**92.1**	91.1
Full Body	85.4	86.4	88.1	90.2	89.9	**91.4**	90.6

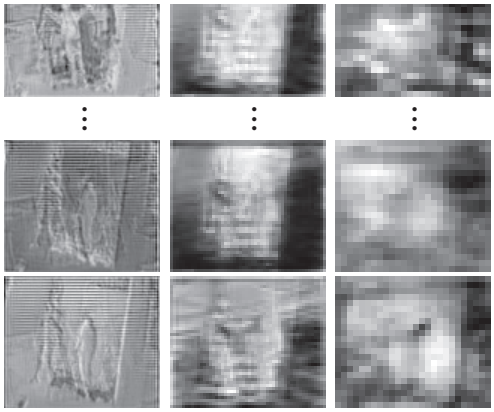

Figure 5: The visualization of some learned memory maps from (a) the first CMB, (b) the second CMB, (c) the third CMB.

Table 6 demonstrates the PCKh@0.5 comparison results. As one can see from Table 6, there lies a significant performance gap (about 5.2%) between Ours and Ours w/o CMB. This highlights the superiority of our proposed CMB. Thanks to the proposed CMB, the representation of the convolutional layers inside our framework has been significantly enhanced. Moreover, without requiring sequential training data, our framework even performs about 4% and 2% better than the Ours w/ convGRU and Ours w/ ConvLSTM, respectively. More importantly, our framework achieves similar performance as the Ours w/ Sequence. This proves the superior performance of our CMB in storing the representative features for human pose estimation from non-sequential depth data. As one can see from Table 6, the estimation accuracy of our framework slightly increase from 89.9% to 91.4% as the memory capacity inside the CMB grows from Ours w/ Half to Ours w/ Double. The reason may be that larger memory capacity can store richer implicit structural features for convolutional layer augmentation. However, larger memory capacity can also bring much more computational cost (nearly double). Therefore, we employ the current capacity to achieve a trade-off between the accuracy and efficiency. Updating the internal memory during the testing can obtain about 1.0 higher accuracy than the fixed version on the K2HPD benchmark using PCK@0.5 metric. The reason is that the internal

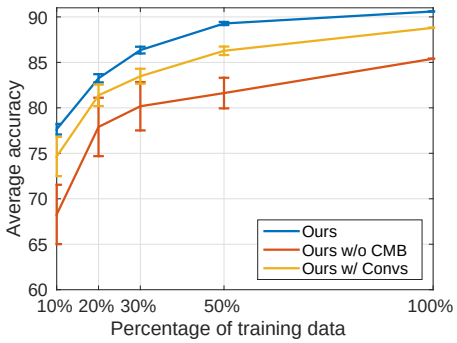

Figure 6: Experimental study on the average estimation accuracy with standard deviation under various percentages of training data from K2HPD using PCK@0.5 metric.

memory can store some relevant features/patterns to benefit the current 3D pose estimation under this circumstance.

Moreover, since the memory map contributes to store the abstraction patterns for reusing the rich implicit convolutional structures of CNNs and spatial correlations among the training samples, we demonstrate the visualization of learned memory maps in Fig. 5. As shown, the memory maps have high responses along the human body structures. This validates the effectiveness of our CMB.

5.3 Evaluation with Insufficient Training Data

To further explore the effectiveness of our framework under the insufficient training data setting, we have fine-tuned our framework with different percentages of training data on the K2HPD benchmark using the PCK@0.5 metric. We have also compared our framework with Ours w/o CMB to validate the contribution of the proposed CMB. For a fair comparison, we have evaluated a variant of our method (denoted as "Ours w/ Convs") by replacing CMB with convolutional layers in a fair number of parameters.

The detailed estimation accuracy with standard deviation of 5 repeated trails using the PCK@0.5 metric is listed in Fig. 6. As the percentage of training data increases, the increased estimation accuracy can be gradually obtained. However, the accuracy of our framework obtains a steady growth with small deviation from 10% to 100% of training data, while that of Ours w/o CMB increases sharply with large deviation. Especially, under the 10% training data setting, our framework performs nearly 16% better than Ours w/o CMB. Moreover, Ours w/ Convs performs significantly better than Ours w/o CMB due to the additional introduced convolutional parameters. However, our framework still consistently outperforms Ours w/ Convs by clear margins as the percentage of training data increases. This comprehensively validates the significant contribution of the CMB on improving learning efficiency when given insufficient training data.

5.4 Sensitivity Analysis

We have conducted sensitivity analysis for the hyper-parameter γ_i, γ_o, γ_m and β. Considering share the similar controller architecture, we let $\gamma_i, \gamma_o, \gamma_m$ all equal to a variable γ_0 to analyze them together since they all work as a balance between the representation from input feature map and

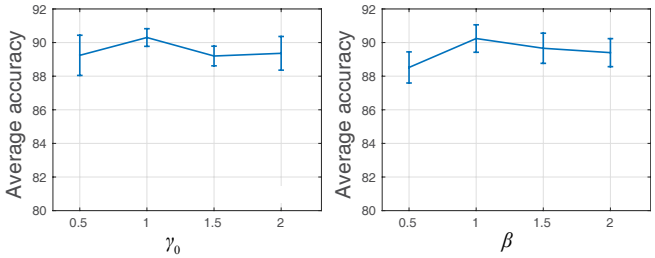

Figure 7: Experimental study on the average estimation accuracy with standard deviation under various hyper-parameters on K2HPD using PCK@0.5 metric.

the representation from previous memory map. Specifically, we fix $\beta = 1$ and perform five trials with γ_0 ranging from 0.5 to 2 on the K2HPD benchmark using the PCKh@0.5 metric. Similarly, we fix $\gamma_i = \gamma_o = \gamma_m = 1$ and perform five trials with β ranging from 0.5 to 2. The average accuracy with standard deviations is illustrated in Fig. 7. As shown, our method is relatively insensitive to the hyper-parameters from a global perspective. There is a higher average accuracy with a smaller deviation when γ_0 and β equal 1. Too large or too small γ_0 results in a smaller average accuracy with a larger deviation.

6 Conclusion

This paper presented a novel memory network module called Convolutional Memory Block (CMB) for improving the learning efficiency and representation of CNNs on still depth images, which lack of color / texture information. The CMB is designed to enable the memory mechanism of convolutional layers by leveraging an internal memory with three specific controllers to store the rich representative structural features and spatial correlations among training samples. Based on the proposed CMB, we have developed a concise yet powerful articulated pose estimation framework, which employs three CMBs to enhance its three different scales of convolutional feature maps. Extensive experiments validated the effectiveness and efficiency of our CMB and framework. In the future, we will extend our CMB to support depth video data for other tasks, e.g., human action/activity recognition.

Acknowledgment

This work was supported in part by the Hong Kong Polytechnic University's Joint Supervision Scheme with the Chinese Mainland, Taiwan and Macao Universities (Grant no. G-SB20). This work was also supported in part by National Natural Science Foundation of China (NSFC) under Grant U1611461 and Grant 61702565, and in part by Science and Technology Planning Project of Guangdong Province of No.2017B010116001.

References

[Ballas *et al.*, 2016] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[Bulat and Tzimiropoulos, 2016] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *ECCV*, 2016.

[Carreira *et al.*, 2016] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feed- back. In *CVPR*, page 4733– 4742, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[Girshick *et al.*, 2011] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, pages 415–422, 2011.

[Graves *et al.*, 2014] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. In *arXiv:1410.5401*, 2014.

[Graves *et al.*, 2016] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, and et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

[Grinciunaite *et al.*, 2016] Agne Grinciunaite, Amogh Gudi, Emrah Tasli, and Marten den Uyl. Human pose estimation in space and time using 3d cnn. In Gang Hua and Hervé Jégou, editors, *ECCV Workshops*, 2016.

[Haque *et al.*, 2016] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *ECCV*, October 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

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

[Jung *et al.*, 2015] Ho Yub Jung, Soochahn Lee, Yong Seok Heo, and Il Dong Yun. Random tree walk toward instantaneous 3d human pose estimation. In *CVPR*, pages 2467–2474, 2015.

[Kendall and Cipolla, 2016] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *ICRA*, 2016.

[LeCun *et al.*, 1990] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[Liang *et al.*, 2016] Xiaodan Liang, Xiaohui Shen, Donglai Xiang, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with local-global long short-term memory. In *CVPR*, 2016.

[McColl *et al.*, 2011] Derek McColl, Zhe Zhang, and Goldie Nejat. Human body pose interpretation and classification for social human-robot interaction. *International Journal of Social Robotics*, 3(3):313, Jun 2011.

[Na *et al.*, 2017] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *ICCV*, Oct 2017.

[Newell *et al.*, 2016] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[Park *et al.*, 2017] Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. Attend to you: Personalized image captioning with context sequence memory networks. In *CVPR*, 2017.

[Santoro *et al.*, 2016] A Santoro, S Bartunov, M Botvinick, D Wierstra, and T Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016.

[Shi *et al.*, 2015] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015.

[Shotton *et al.*, 2011] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011.

[Shotton *et al.*, 2013a] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *TPAMI*, 35(12):2821–2840, 2013.

[Shotton *et al.*, 2013b] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, 2013.

[Sun *et al.*, 2012] M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. In *CVPR*, pages 3394–3401, 2012.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, June 2015.

[Tompson *et al.*, 2014] Jonathan Tompson, Arjun Jain, Yann Lecun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1799–1807, 2014.

[Toshev and Szegedy, 2014] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, pages 1653–1660, 2014.

[Wang *et al.*, 2016] Keze Wang, Shengfu Zhai, Hui Cheng, Xiaodan Liang, and Liang Lin. Human pose estimation from depth images via inference embedded multi-task learning. In *ACM MM*, pages 1227–1236, 2016.

[Wei *et al.*, 2016] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, June 2016.

[Weston *et al.*, 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

[Xu and Cheng, 2013] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *ICCV*, 2013.

[Xu *et al.*, 2015] Chi Xu, Ashwin Nanjappa, Xiaowei Zhang, and Li Cheng. Estimate hand poses efficiently from single depth images. *International Journal of Computer Vision (IJCV)*, 2015.

[Zhang *et al.*, 2018] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.