LSTM Pose Machines

Yue Luo[1] Jimmy Ren[1] Zhouxia Wang[1] Wenxiu Sun[1] Jinshan Pan[1] Jianbo Liu[1] Jiahao Pang[1] Liang Lin[1,2]

[1]SenseTime Research
[2]Sun Yat-sen University, China
[1]{luoyue, rensijie, wangzhouxia, sunwenxiu, panjinshan, liujianbo, pangjiahao, linliang}@sensetime.com

Abstract

We observed that recent state-of-the-art results on single image human pose estimation were achieved by multi-stage Convolution Neural Networks (CNN). Notwithstanding the superior performance on static images, the application of these models on videos is not only computationally intensive, it also suffers from performance degeneration and flicking. Such suboptimal results are mainly attributed to the inability of imposing sequential geometric consistency, handling severe image quality degradation (e.g. motion blur and occlusion) as well as the inability of capturing the temporal correlation among video frames. In this paper, we proposed a novel recurrent network to tackle these problems. We showed that if we were to impose the weight sharing scheme to the multi-stage CNN, it could be re-written as a Recurrent Neural Network (RNN). This property decouples the relationship among multiple network stages and results in significantly faster speed in invoking the network for videos. It also enables the adoption of Long Short-Term Memory (LSTM) units between video frames. We found such memory augmented RNN is very effective in imposing geometric consistency among frames. It also well handles input quality degradation in videos while successfully stabilizes the sequential outputs. The experiments showed that our approach significantly outperformed current state-of-the-art methods on two large-scale video pose estimation benchmarks. We also explored the memory cells inside the LSTM and provided insights on why such mechanism would benefit the prediction for video-based pose estimations.[1]

1. Introduction

Estimating joint locations of human bodies is a challenging problem in computer vision which finds many real applications in areas including augmented reality, animation and automatic photo editing. Previous methods [2, 6, 38] mainly addressed this problem by well designed graphical models. Newly developed approaches [5, 23, 36] achieved higher performance with deep Convolutional Neural Networks (CNN).

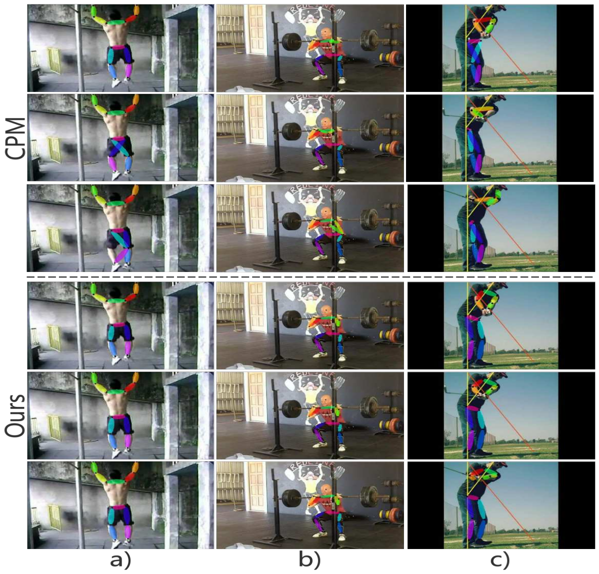

Figure 1. Comparison of results produced by Convolutional Pose Machine (CPM) [36] after setting the video as a series of static images (Up) and our method (Down). Several problems occur during pose estimation on videos: a) Errors and our correct results in estimating symmetric joints. b) Errors and our correct results when joints are occluded. c) Flicking results and our results when the body moves rapidly.

Nevertheless, those state-of-the-art models were trained on still images, limiting their performance on videos. Figure 1 demonstrates some unsatisfactory situations. For instance, the lack of geometric consistency makes the previous methods prone to making obvious errors. Mistakes caused by serious occlusion and large motion are not uncommon as well. In addition, those models usually have a deep architecture and would be computationally very intensive for real-time applications. Therefore, a relatively light-weight model is preferable if we want to deploy it in a

[1] Code is publicly available at `https://github.com/lawy623/LSTM_Pose_Machines`.

real-time video processing system.

An ideal model of such kind must be able to model the geometric consistency as well as the temporal dependency among video frames. One way to address this is to calculate the flow between every two frames and use this additional cue to improve the prediction [26, 32]. This approach is effective when the flow can be accurately calculated. However, this is not always the case because the calculation of optical flow suffers from image quality degradation as well.

In this paper, we adopted a data-driven approach to better tackle this problem. We showed that a multi-stage CNN could be re-written as a Recurrent Neural Network (RNN) if we impose the weight sharing scheme. This new formulation decouples the relationship among multiple network stages and results in significantly faster speed in invoking the network for videos. It also enables the adoption of Long Short-Term Memory (LSTM) units between video frames. By effectively learning the temporal dependency among video frames, this novel architecture well captures the geometric relationships of joints in time and increases the stability of joint predictions on moving bodies. We evaluated our method on two large-scale video pose estimation benchmarks namely, Penn Action [40] and sub-JHMDB [14]. Our method significantly outperformed all previous methods both in performance and speed.

To well justify our findings, we also investigated the internal dynamics of the memory cells inside our LSTM and explained why and how LSTM units would improve the video pose estimation performance. The memory cells were visualized and insights were provided.

The contributions of our work can be summarized as follows.

- First, we built a novel recurrent architecture with LSTM to capture temporal geometric consistency and dependency among video frames for pose estimation. Our method surpassed all the existing approaches on two large-scale benchmarks.
- Second, the new architecture decouples the relationship among network stages and results in much faster inference speed for videos.
- Third, we probed into the LSTM memory cells and visualized how they would help to improve the joint predictions on videos. It provides insights and justifies our findings.

2. Related Works

Early works on single-image pose estimation started from building graphical structures [2, 6, 28, 33, 38] to model the relations between joints. However, those methods rely heavily on hand-crafted features which restrict their generality on varied human poses in reality. The performance of these methods has recently been surpassed by CNN based methods [4, 5, 23, 34, 35, 36]. Those deep models had the capacity to generalize from unseen scenes by learning various spatial relations from data. Recent works [23, 36] employed the strategy of iteratively refining the output of each network stage and achieved state-of-the-art results in many image-based benchmarks. In [3], a recurrent model was proposed to reduce training parameters, but it was designed for images rather than videos.

Directly applying the existing image-based methods on video sequences produces sub-optimal results. There are two major problems. First, these models failed to capture temporal dependency among video frames and they were unable to keep the geometric consistency. It can be shown that the image-based models can easily suffer from motion blur and occlusion and usually generate inconsistent results for neighbouring frames. Second, the image-based models are usually very deep and computationally expensive. It is problematic when adopting them in real-time applications.

A few previous studies integrated temporal cues into pose estimation [8, 12, 19, 24, 26, 27, 32]. Modeep [12] first tried to merge motion features into ConvNet, and Pfister et al. [27] made a creative attempt to insert consecutive frames at different color channels as input. In later works [26, 32], dense optical flow [37] was produced and used to adjust the predicted positions in order to let the movement smooth across frames. Good results were achieved by Thin-Slicing Network [32] which relied on both adjustment from optical flow and a spatial-temporal model. However, this system is computationally very intensive and is slower than the previous image-based method. Our method is similar to the Chained Model [8], which is a simple recurrent architecture that can capture temporal dependencies. Unlike [8], our model better captured temporal dependency by memory augmented RNN (LSTM) and it achieved better performance. LSTM have been widely used in pose-related tasks such as motion tracking and action recognition [7, 13, 20, 22]. RPSM [19] also adopted the LSTM for pose estimation in 3D space, but its LSTM operated in the domain between 2D and 3D conversion and mainly concerned about the quality of such conversion. By employing LSTM in 2D video-based pose estimation, we are able to outperform current state-of-the-art methods while keeping a concise architecture.

Understanding the underlying mechanism behind neural networks is important and of great interests among many researchers. Several works [21, 39] aimed to explain what the convolution models had learned by reconstructing the features into original images. Likewise, [17] studied the long-range interactions captured by recurrent neural network in text processing. And in particular, it interpreted the function of LSTM in text-based works. In this paper, we combined the analysis from these two sides, and visualized how our model learned and helped the work of locating moving joints in videos.

3. Analysis and Our Approach

3.1. Pose Machines: From Image to Video

Pose Machine [29] was first brought up as a method to predict joint locations in a sequentially refined manner. The model was built on the inference machine framework to learn strong interconnections between body parts. Convolutional Pose Machine (CPM) [36] inherited the idea from pose machine with implementing it in a deep architecture. At the same time, it adopted a fully convolutional design by producing predicted heat maps at the end of the system. As a critical strategy exploited in pose machines, passing prior beliefs into next stages and supervising the loss in all stages benefit the training of such a deep ConvNet by addressing the problem of gradient vanishing. Following the descriptions in [36], we can formulate the model mathematically in the following way: Denote $\mathbf{b}_s \in \mathbb{R}^{W \times H \times (P+1)}$ (P joints plus one background channel with size $W \times H$) as the beliefs in stage $s \in \{1, 2, ..., S\}$, they can be calculated iteratively by:

$$\begin{aligned} \mathbf{b}_s &= g_s(X), & s &= 1, \\ \mathbf{b}_s &= g_s(\mathcal{F}_s(X) \oplus \mathbf{b}_{s-1}), & s &= 2, 3, ..., S, \end{aligned} \tag{1}$$

where $X \in \mathbb{R}^{W \times H \times C}$ is the original image sent into every stage. $\mathcal{F}_s(\cdot)$ is a ConvNet used to extract valuable features from input image. Those features will be concatenated (indicated by operation $\oplus$) with prior beliefs (i.e. $\mathbf{b}_{s-1}$) and sent into another ConvNet $g_s(\cdot)$ to produce refined belief maps. It is easy to observe that CPM does a great job on pose estimation because $g_s(\cdot)$ and $\mathcal{F}_s(\cdot)$ are not identical across different stages s even though they share the same architecture (in fact $g_{s=1}(\cdot)$ uses a deeper structure compared with $g_{s>1}(\cdot)$ in order to produce more precise confidence maps for further refinements since its unprocessed input contains only local evidences). It repetitively modifies the confidence maps by adding intermediate supervisions at the end of each stage. However, applying this deep structure for video-based pose estimation is not practical because it does not integrate any temporal information.

Chained model [8] provided us a motivation to construct an RNN style model for this problem. And we were also inspired by the design of CPM to reform it into a recurrent one. Referring to Eq. (1), we found that CPM could be easily transformed into a recurrent structure by sharing the weights of those two functions $g_s(\cdot)$ and $\mathcal{F}_s(\cdot)$ across stages. Mathematically, a new Recurrent Pose Machine derived from CPM can be formulated as:

$$\begin{aligned} \mathbf{b}_t &= g_0(X_t), & t &= 1, \\ \mathbf{b}_t &= g(\mathcal{F}(X_t) \oplus \mathbf{b}_{t-1}), & t &= 2, 3, ..., T. \end{aligned} \tag{2}$$

Here, $\mathbf{b}_t$ is no longer the belief maps in a certain stage as described in Eq. (1), but it represents the produced belief maps matched with frame $t \in \{1, 2, ..., T\}$ where T is now the length of frames in this video. The input $X_{t(1 \leqslant t \leqslant T)}$'s are not the same in different stages, but they are consecutive frames from a video sequence. Similarly, $g_0(\cdot)$ at the initial place is still different from $g(\cdot)$, and now all the following stages share an exactly identical function. With this implementation, the model is rebuilt with recurrent design and it can be used to predict joint locations from a variable-length video. Apart from its recurrent property, it also accomplishes another notable achievement which is lessening the parameters for predicting locations from a single frame.

Training of the model described in Eq. (2) can now be proceeded collectively on a set of successive frames. However, this RNN model cannot achieve optimal performance on video-based pose estimation. We found that it was beneficial to include an LSTM unit [10] because of its special gate designs and memory implementation. This modification can be achieved by further adapting Eq. (2). In other words, our new memory-enabled recurrent pose machines become:

$$\begin{aligned} \mathbf{b}_t &= g(\widetilde{\mathcal{L}}(\mathcal{F}^{'}(X_t))), & t &= 1, \\ \mathbf{b}_t &= g(\widetilde{\mathcal{L}}(\mathcal{F}(X_t) \oplus \mathbf{b}_{t-1}))), & t &= 2, 3, ..., T. \end{aligned} \tag{3}$$

$\widetilde{\mathcal{L}}(\cdot)$ is a function controlling memory's inflow and outflow procedures. In Eq. (2), $g_0(\cdot)$ contains two parts, namely a feature encoder and a prediction generator. Since $\widetilde{\mathcal{L}}(\cdot)$ directly receives processed features, we separate these two parts and plug the LSTM between them as shown in Eq. (3). The extractor acts like $\mathcal{F}(\cdot)$ in other stages but it is much deeper, so we denote it as $\mathcal{F}^{'}(\cdot)$. Now we can also see that the generators $g(\cdot)$ are identical across all stages. Since nothing is in LSTM's memory at the first stage, $\widetilde{\mathcal{L}}(\cdot)$ will be a little bit different from that in subsequent stages, but they all perform similar functionality. We will discuss the implementation in detail in later sections, and more importantly, we will explain how the LSTM can robustly boost the performance of our recurrent pose machines.

3.2. LSTM Pose Machines

Details of the Model. Figure 2 illustrates our structure stated in Eq. (3) for pose estimation on video. Consecutive frames in the same video clip will be sent into the network as input in different stages. As shown in the figure, when $t = 1$, $\mathcal{F}^{'}(X_t)$ can be decomposed as $\mathcal{F}_0(X_t) \oplus \mathcal{F}(X_t)$, where $\mathcal{F}_0(\cdot)$ is the *ConvNet1* aiming at processing raw input and $\mathcal{F}(\cdot)$ is the encoder *ConvNet2* consistently used in all stages. $\mathcal{F}_0(\cdot)$ produces preliminary belief maps associated with the first frame. Since the prediction does not have a high confidence level, it will be concatenated with $\mathcal{F}(X_1)$ again to generate a more accurate result. *LSTM* is the most

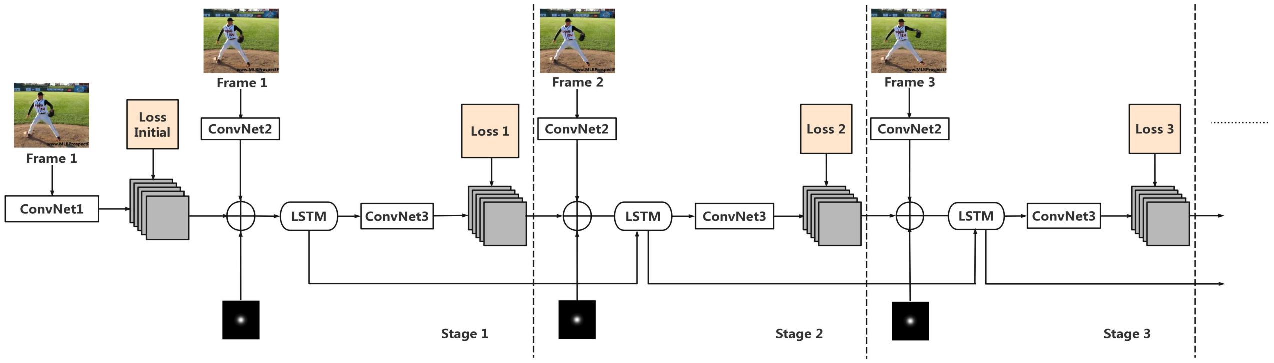

Figure 2. Network architecture for LSTM Pose Machines. This network consists of T stages, where T is the number of frames. In each stage, one frame from a sequence will be sent into the network as input. *ConvNet2* is a multi-layer CNN network for extracting features while an additional *ConvNet1* will be used in the first stage for initialization. Results from the last stage will be concatenated with newly processed inputs plus a central Gaussian map, and they will be sent into the *LSTM* module. Outputs from *LSTM* will pass *ConvNet3* and produce predictions for each frame. The architectures of those *ConvNet*s are the same as the counterparts used in the CPM model [36] but their weights are shared across stages. *LSTM* also enables weight sharing, which reduces the number of parameters in our network.

critical component in this architecture. It can be referred to as the $\widetilde{\mathcal{L}}(\cdot)$ function we mentioned above. In reality, it takes multiple steps to forget the old memory, absorb new information and create the output. *ConvNet3* is the generator $g(\cdot)$ we described in Eq. (3) and it is connected to the output from LSTM. All those ConvNet segments comprise several convolution layers, activation layers and pooling layers. They inherit the design of Convolutional Pose Machines [36], and the architectures of them are the same as the counterparts used in the CPM model. The difference is that our model allows weight sharing for all these components across stages. Following CPM [36], we add an extra slice containing a central Gaussian peak during input concatenation for better performance. Dropout is also included in the last layers of *ConvNet1*.

Convolutional LSTM Module. The structure and functionality of LSTM have been discussed in many prior works [10, 9, 31]. A *vanilla* LSTM is defined in [9] and it is the most commonly used LSTM implementation. In [9], Greff et al. conducted a comprehensive study on the components of LSTM, and they found out that this *vanilla* LSTM with forget gate, input gate and output gate already outperformed other variants of LSTM. Eq. (4) illustrates the operations inside a *vanilla* LSTM unit that we used in our recurrent model:

$$\begin{aligned} g_t &= \varphi(\mathbf{W}_{xg} * X_t + \mathbf{W}_{hg} * h_{t-1} + \epsilon_g), \\ i_t &= \sigma(\mathbf{W}_{xi} * X_t + \mathbf{W}_{hi} * h_{t-1} + \epsilon_i), \\ f_t &= \sigma(\mathbf{W}_{xf} * X_t + \mathbf{W}_{hf} * h_{t-1} + \epsilon_f), \\ o_t &= \sigma(\mathbf{W}_{xo} * X_t + \mathbf{W}_{ho} * h_{t-1} + \epsilon_o), \\ C_t &= f_t \odot C_{t-1} + i_t \odot g_t, \\ h_t &= o_t \odot \varphi(C_t) \end{aligned} \tag{4}$$

Unlike traditional LSTM, '*' here does not refer to a matrix multiplication but to a convolution operation similar as that in [31] and [18]. As a result, all the '+' in Eq. (4) represent the element-wise addition. The ϵ's here denote the bias terms. These settings result in our convolutional LSTM design. $i_t(\cdot)$, $f_t(\cdot)$, $o_t(\cdot)$ are the input gate, forget gate and output gate at time t respectively. They are controlled by new input X_t and hidden state from last stage h_{t-1} mutually. Note that X_t here is not the same as that in Eq. (3). Here it is already the concatenated inputs (*i.e.* $\mathcal{F}(X_t) \oplus \mathbf{b}_{t-1}$ in Eq. (3)). Convolutional design of the gates focuses more on regional context rather than global information, and it pays more attention to the changes of joints in smaller local areas. One convolution layer with 3 × 3 kernel is found to be best for performance. C_t is the memory cell which preserves knowledges in a long range by forgetting old memory and taking in new information continuously. Hidden state h_t will be outputted from the newly formed memory and it will be used to generate current beliefs via the generator $g(\cdot)$. The first memory cell C_1 is calculated by $i_1 \odot g_1$ only since forget operation is unavailable.

Training of the Model. Our LSTM Pose Machine is implemented in *Caffe* [15], and functions in LSTM are simply implemented by convolutions and element-wise operations. Labels in Cartesian coordinates are transformed into heat maps with Gaussian peaks centred at the joint positions. The network has T stages, where T is the number of consecutive frames in the training sequence. Loss will be added at the end of each stage to supervise the learning periodically. Training aims to reduce the total l_2 distance between prediction and ground truth for all joints and all frames jointly. Loss function is defined as:

$$F = \sum_{t=1}^{T} \sum_{p=1}^{P+1} \|b_t(p) - g.t._t(p)\|^2, \tag{5}$$

where $b_t(p)$ is the produced belief and $g.t._t(p)$ is the ground truth heat map for part p in stage t.

4. Experiments and Evaluations

In this section, we present our experiments and quantitative results on two widely used datasets. Our method achieved state-of-the-art results in both of them. Qualitative results will be also provided in this part. At last, we will explore and visualize the dynamics inside LSTM units.

4.1. Datasets

Penn Action Dataset. Penn Action Dataset [40] is a large dataset containing in total 2326 video clips, with 1258 clips for training and 1068 clips for testing. On average each clip contains 70 frames, but the number in fact varies a lot for different cases. 13 joints including *head*, *shoulders*, *elbows*, *wrists*, *hips*, *knees* and *ankles* are annotated in all the frames. An additional label indicates whether a joint is visible or not in a single image. Following previous works, evaluation will be only conducted on visible joints.

Sub-JHMDB Dataset. JHMDB [14] is another video-based dataset for pose estimation. For comparison purpose, we only conduct our experiment on a subset of JHMDB called sub-JHMDB dataset to maintain consistency with previous works. This subset contains only complete bodies and no invisible joint is annotated. Sub-JHMDB has 3 different split schemes, so we trained our model separately and reported the average result over these three splits. This subset has 316 clips with all 11200 frames in the same size. Split results in a train/test ratio which is roughly equal to 3.

4.2. Implementation Details

Data Augmentation is randomly performed to increase variation of input. Since a set of frames will be sent into the network at the same time, the transformation will be consistent within a patch. Images will be randomly scaled by a factor. For Penn this factor is between 0.8 to 1.4 while for sub-JHMDB it is between 1.2 to 1.8 since the bodies are originally smaller. Images will then be rotated with degree $[-40°,40°]$ and flipped with randomness. At last, all the images will be cropped to a fixed size (368×368) with bodies set at center.

Parameter settings. Since we directly revised the architecture of Convolutional Pose Machines [36], we can easily initialize the weights based on the pre-trained CPM model. Instead of directly copying weights from it, we first built a single image model which used the same structure as our model trained on video sequences. The difference is that we set $T = 6$ for this single image model and the inputs are identical in all stages. We only copied the weights in the first two stages of CPM model since weights in our model are shareable across stages. This model was fine-tuned for several epochs on the combination of LSP [16] and MPII [1] datasets, which is the same data source for training the CPM model from scratch.

Our models for training on Penn and sub-JHMDB started by copying the weights from our single image models described above. During training, length of our recurrent model is set to be 5 (*i.e.* T=5), which is large enough to observe sufficient changes from a video sequence. Stochastic gradient descent with momentum of 0.9 and weight decay of 0.0005 is used to optimize the learning process. Batch size is selected to be 4. The initial learning rate is set to be 8×10^{-5} and it will drop by multiplying a factor of 0.333 every 40k iterations. Gradient clipping is used and set as 100 to prevent gradient explosion. Dropout ratio is 0.5 in the first stage.

4.3. Evaluation on Pose Estimation Results

Similar to many prior works, beliefs for joints are produced at the end of each stage. Positions in x,y coordinates can then be interpolated from finding the maximum confidence. During testing, we first rescaled the input into different sizes, and averaged the outputs to produce a more reliable belief. In our experiments, we rescaled the images into 7 scales and the scaling factors are within the corresponding regions that we used for augmentation during training. To evaluate the results, we adopt the PCK metric introduced in [38]. An estimation is considered correct if it lies within $\alpha \cdot max(h, w)$ from the true position, where h and w are the height and width of the bounding box. In order to consistently compare with other methods, α is chosen to be 0.2 for evaluation on both datasets. Penn already annotates the bounding box within each image, but the bounding boxes for sub-JHMDB are deduced from the puppet masks used for segmentation.

Method	Head	Sho	Elb	Wri	Hip	Knee	Ank	Mean
[25]	62.8	52.0	32.3	23.3	53.3	50.2	43.0	45.3
[24]	64.2	55.4	33.8	24.4	56.4	54.1	48.0	48.0
[11]	89.1	86.4	73.9	73.0	85.3	79.9	80.3	81.1
[8]	95.6	93.8	90.4	90.7	91.8	90.8	91.5	91.8
[32]	98.0	97.3	95.1	94.7	97.1	97.1	96.9	96.5
CPM [36]	98.6	97.9	95.9	95.8	98.1	97.3	96.6	97.1
RPM	98.5	98.2	95.6	95.1	97.4	97.5	96.8	97.0
LSTM PM	**98.9**	**98.6**	**96.6**	**96.6**	**98.2**	**98.2**	**97.5**	**97.7**

Table 1. Comparisons of results on Penn dataset using PCK@0.2. RPM here simply removes the LSTM module from LSTM PM. Notice that [25] is N-Best, [8] is Chained Model, and [32] is Thin-Slicing Net. The best results are highlighted in Bold.

4.4. Analysis of Results

Results on Penn and sub-JHMDB. Table 1 and table 2 show the performance of our models and previous works on Penn dataset as well as sub-JHMDB dataset. Apart from

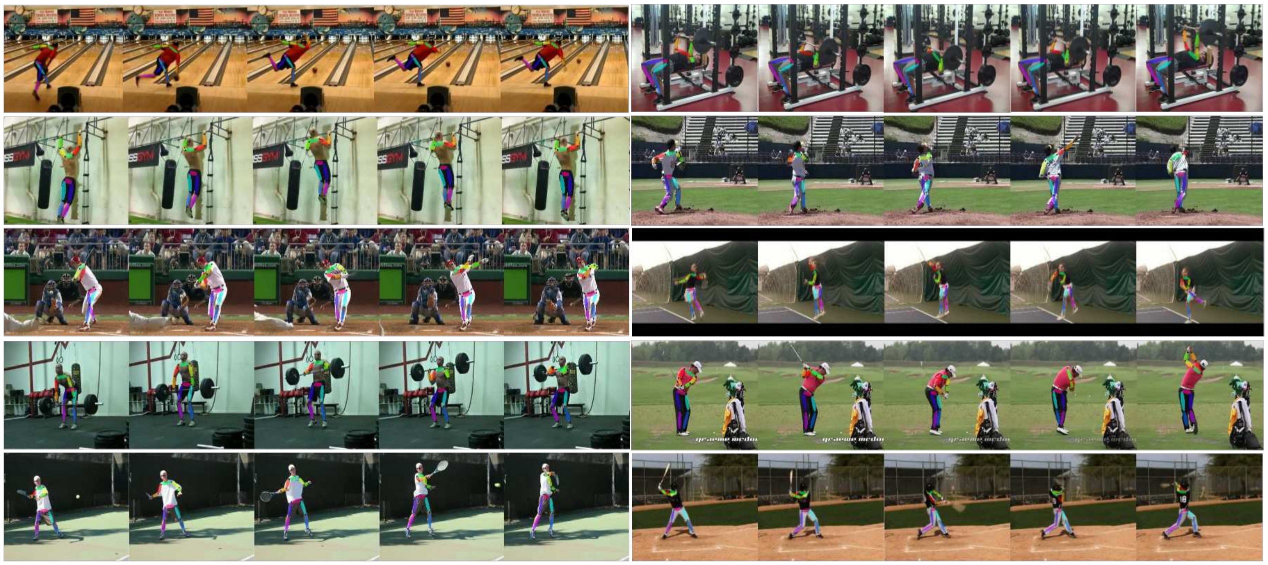

Figure 3. Qualitative results of pose estimations on Penn and sub-JHMDB datasets using our LSTM Pose Machines.

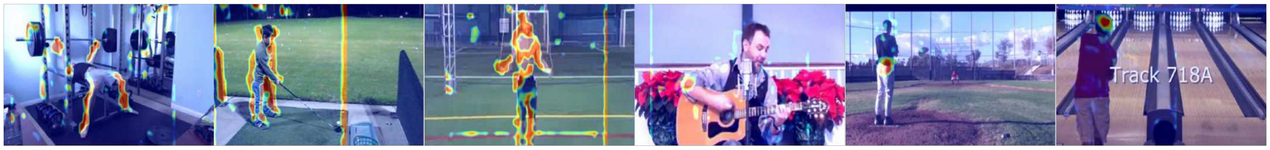

Figure 4. attention from different memory channels. The first three focus on trunks or edges while the other three focus on a particular joint.

LSTM Pose Machines (LSTM PM) stated in Eq. (3), we also present a simplified Recurrent Pose Machine model (RPM) as described in Eq. (2). It simply takes off the LSTM modules and it was trained using the same parameters in order to study the contribution of LSTM component. By considering long-term temporal information in our models, we achieved improved results in both benchmarks. Comparing our state-of-the-art LSTM Pose Machines with previous video-based pose estimation methods such as Thin-Slicing Net [32], we observe an overall improvement of 1.2% which is evenly distributed in all body parts in the case of Penn benchmark. Among all those parts, we find that the greatest boost of 1.9% increase comes from the *wrist*. Similarly, for sub-JHMDB dataset, we achieved improvements in almost all the joints. It is worth noticing that the biggest increases come from *elbow* and *wrist*. This is a significant result since we have robustly improved the predictive accuracy of the joints that are subject to drastic movements and occlusion. In our experiments, we trained a CPM model [36] on these two datasets with the same training scheme as well. We can see that it has already surpassed all existing methods on both benchmarks but it still can not compete with us. Qualitative results are presented in figure 3. We can see that our method is especially suitable to cope with big changes across frames through its strong predictive power. Even though the body is in motion or it suffers from an occlusion in the middle of the video, positions can be inferred from their past trajectories smoothly.

Method	Head	Sho	Elb	Wri	Hip	Knee	Ank	Mean
[25]	79.0	60.3	28.7	16.0	74.8	59.2	49.3	52.5
[24]	80.3	63.5	32.5	21.6	76.3	62.7	53.1	55.7
[11]	90.3	76.9	59.3	55.0	85.9	76.4	73.0	73.8
[32]	97.1	95.7	87.5	81.6	98.0	92.7	89.8	92.1
CPM [36]	**98.4**	94.7	85.5	81.7	97.9	94.9	90.3	91.9
RPM	98.0	95.5	86.9	82.9	97.9	94.9	89.7	92.2
LSTM PM	98.2	**96.5**	**89.6**	**86.0**	**98.7**	**95.6**	**90.9**	**93.6**

Table 2. Comparisons of results on sub-JHMDB dataset using PCK@0.2. RPM here simply removes the LSTM module from LSTM PM. Notice that [25] is N-Best and [32] is Thin-Slicing Ne. The best results are highlighted in Bold.

Contribution of LSTM Module. From table 1 and table 2, we can see that our recurrent models without LSTM module (RPM) also provided improved results comparing to all previous video-based methods. CPM is a strong baseline on image-based pose estimation and it uses multi-stage refinements to get inference of joint locations. RPM utilizes temporal information which is found essential in video-based tasks while it uses a shorter structure. Experiments show that RPM does not strictly beat CPM since RPM does not utilize temporal correlations in an optimal way. Our memory augmented recurrent model better captures temporal information and surpasses both of them. Comparing with RPM, our LSTM model achieves an average increment of 0.7% in PENN and 1.4% in sub-JHMDB. For those easy parts such as head, shoulder and hip, RPM is already able to perform well. But for those joints that are easily subject to occlusion or motion, the memory cells help to robustly promote the estimation accuracy of them by better utilizing their historical locations. With the help of our LSTM module, we can conclude that our approach increased overall stability in predicting joints from moving frames.

T	Head	Sho	Elb	Wri	Hip	Knee	Ank	Mean
1	97.0	95.0	85.9	81.8	98.4	92.6	87.0	91.1
2	98.1	96.2	88.6	84.4	**98.7**	95.5	90.7	93.2
5	98.2	**96.5**	89.6	**86.0**	**98.7**	**95.6**	**90.9**	**93.6**
10	**98.5**	**96.5**	**89.7**	**86.0**	98.5	94.9	90.1	93.5

Table 3. Comparisons the results of different iterations of LSTM on sub-JHMDB dataset using PCK@0.2. The best results are highlighted in Bold.

Analysis of increasing the iterations of LSTM In this part, we explore the effect of using different iterations T. We train our model with different number of stages, i.e., T=1, 2, 5, 10, on the sub-JHMDB dataset, and report the experimental results in Table 3. When there is just one iteration in the LSTM, the performance drops a lot, even worse than CPM, since there is no temporal information or refine operations like CPM model. When iterations increase to 2, the performance has a notable improvement, since current frame would keep information about the joints which are nearly static compared to the last frame from the last stage, and just learn the joints which move a litter faster. It makes the preference more stable among video frames. What's more, the performance still increases when we add iterations from 2 to 5, which means long-term temporal information is good for video pose estimation. However, it doesn't mean the more iterations, the higher performance. The experiment in T=10 tells us that the information of the frames which are very long before current frame is helpless. In order to balance the performance and training computation consumption, we set T=5.

4.5. Inference Speed

Inference time is critical for real-time applications. Previous methods are relatively time-consuming in producing the results because they need to go through many stages for a single frame. Our method only needs to go through a single stage for every video frame thus performs significantly faster than the previous multi-stage CNN based methods. Note that for the first frame, our method needs to go through a longer stage to get started. For fair comparison, we randomly pick a video clip with 100 frames and send them into the CPM model and our model for testing separately. The experiment result shows that the CPM model needs 48.4ms per-frame, but we only need 25.6ms per-frame which means that our model runs about 2x faster than the CPM model. Comparing to the flow based methods such as Thin-Slicing Net [32], which is based on CPM and needs to generate flow map, our model has greater advantages in speed. Thus our model is especially preferable for real-time video-based pose estimation applications.

4.6. Exploring and Visualizing LSTM

In order to better understand the mechanism behind LSTM, exploring the content of memory supplies substantial cues. Sharma et al. [30] and Li et al. [18] have made an attempt on relevant issues recently. In their works, they focused more on the static attention in each stage, but we are going to address the transition of memory content resulted from the changing positions.

Figure 4 displays the results of our exploration. We first up-sampled the channels in memory and mapped them back to original image space. Following our setup, there are 48 channels in each memory cell and we only selected some representative ones here for visualization. From the figure, we can see that memories in different channels are the attention on distinct parts. Some of them are the global views on trunks or edges (the first three samples), and some just focus on a particular joint (the other three show the memory attention on *elbow*, *hip* and *head*). Remember that those memories will be selectively outputted and processed by a network for estimation. Therefore, the memory cell containing both global and local information helps the prediction of spatially correlated joints on a single frame.

A more important property of LSTM is that it maintains its memory by using both useful prior information and new knowledge. As described in Eq. (4), LSTM goes through the process of forgetting and remembering during each iteration. In each row of Figure 5 illustrates different phases of the memory cell within one iteration. It captures the evolution of our LSTM inside the iteration (only represented by one selected channel). Each column represents a single phase according to the figure's description. We can observe from the first sample that the forget operation selectively retains useful information for the prediction in next stage, such as *wrists* and *head*, which are nearly static in the three consecutive frames (col. 3), while new input of this stage brings more emphasis on the regions containing latest appearance of joints, such as *knees*, which have movement in the three consecutive frames (col. 4). These two parts are combined to be a new memory and the new memory produces the predictions on a new frame with high confidence (col. 5). That is why our model can capture temporal geometric consistency and prevent the mistakes in videos as illustrated in Figure 1. For the second sample, in the first

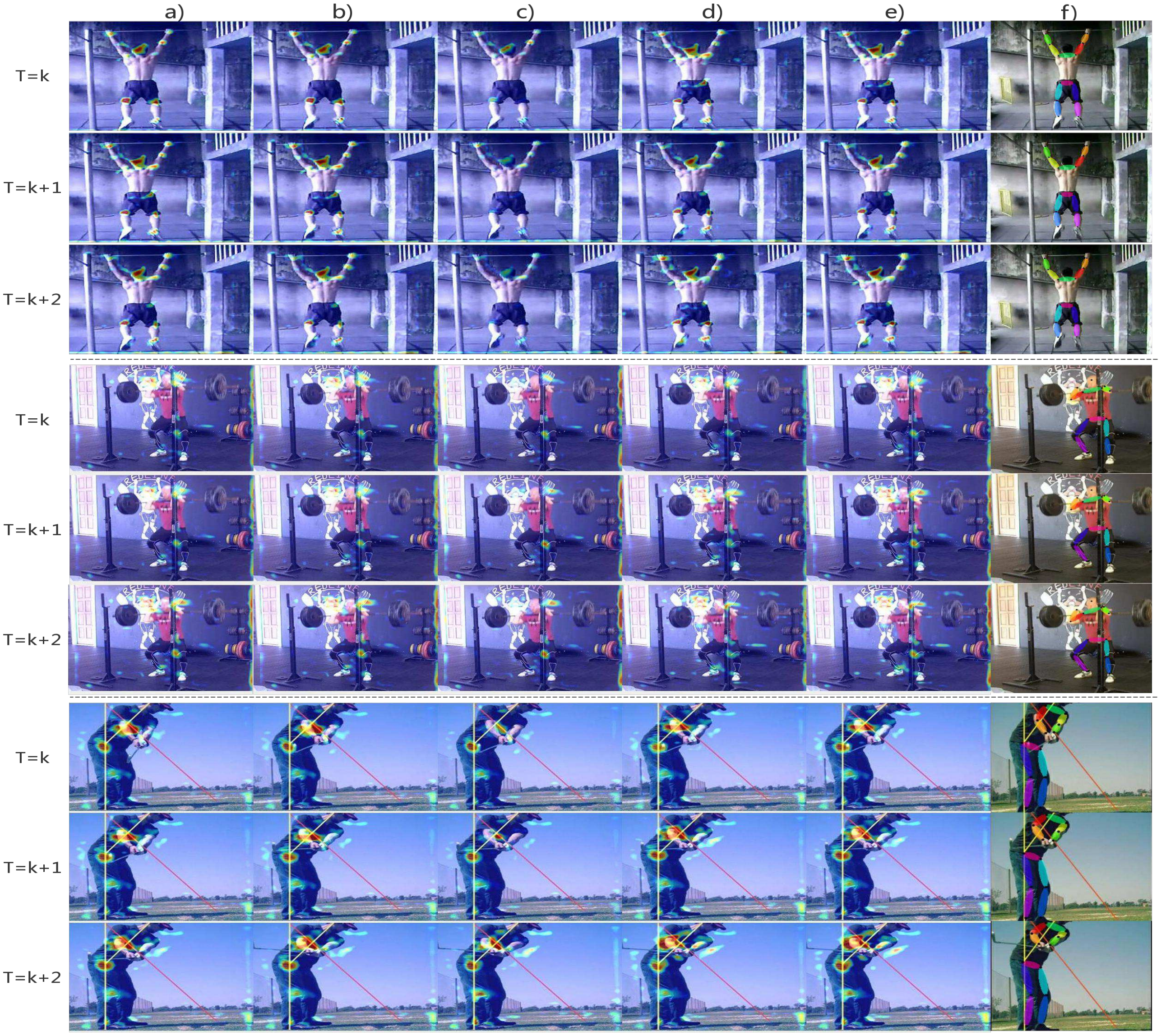

Figure 5. Exploration of LSTM's Memory. a)memory from last stage (*i.e.* C_{t-1}) on last frame X_{t-1}, b)memory from last stage (*i.e.* C_{t-1}) on new frame X_t, c)memory after forget operation (*i.e.* $f_t \odot C_{t-1}$) on new frame X_t , d)newly selected input(*i.e.* $i_t \odot g_t$) on new frame X_t, e)newly formed memory (*i.e.* C_t) on new frame X_t, which is the element-wise sum of c) and d), and f)the predicted results on new frame X_t. For each samples we pick three consecutive frames.

frames, the left *wrist* still can be seen, but it is occluded in the next two frames. In our model, since the left *wrist* has been recognized in the first frame, the following frames can infer the location of it by the memory cell of the last stage though it has been occluded. What's more, the movement of elbows in the third sample is flicking, but our model can keep the static joints (e.g. *hip* and *keen*), and quickly track the new information of rapidly moving joints (e.g. *elbows*) by memory cells and new inputs.

In conclusion, those mechanisms can help to make the predictions more accurate and robust for pose estimation on video.

5. Conclusions

In this paper, we presented a novel recurrent CNN model with LSTM for video pose estimation. We achieved significant improvement in terms of both accuracy and efficiency. We did observe some erroneous predictions when the joint is not visible for a long time, but we still found that the LSTM module indeed contributed to the better utilization of temporal information and it made stable and accurate predictions across the video. In the end, we explored and visualized the memory cells inside the LSTM and explained the underlying dynamics of the memory during pose estimation on changing frames.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 5

[2] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: people detection and articulated pose estimation. In *CVPR*, 2009. 1, 2

[3] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In *International Conference on Automatic Face and Gesture Recognition*, 2017. 2

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 2

[5] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang. Multi-context attention for human pose estimation. *CVPR*, 2017. 1, 2

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005. 1, 2

[7] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015. 2

[8] G. Gkioxari, A. Toshev, and N. Jaitly. Chained predictions using convolutional neural networks. In *ECCV*, 2016. 2, 3, 5

[9] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. In *arxiv. 1503.04069*, 2015. 4

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 3, 4

[11] U. Iqbal, M. Garbade, and J. Gall. Pose for action-action for pose. In *arxiv. 1603.04037*, 2016. 5, 6

[12] A. Jain, J. Tompson, Y. LeCun, and C. Bregler. Modeep: A deep learning framework using motion features for human pose estimation. In *ACCV*, 2014. 2

[13] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016. 2

[14] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 2, 5

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *arxiv. 1408.5093*, 2014. 4

[16] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR*, 2011. 5

[17] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. In *arxiv. 1506.02078*, 2015. 2

[18] Z. Li, E. Gavves, M. Jain, and C. G. M. Snoek. Videolstm convolves, attends and flows for action recognition. In *arxiv. 1607.01794*, 2016. 4, 7

[19] M. Lin, L. Lin, X. Liang, K. Wang, and H. Cheng. Recurrent 3d pose sequence machines. *CVPR*, 2017. 2

[20] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*, 2016. 2

[21] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 2

[22] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017. 2

[23] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 1, 2

[24] B. X. Nie, C. Xiong, and S.-C. Zhu. Joint action recognition and pose estimation from video. In *CVPR*, 2015. 2, 5, 6

[25] D. Park and D. Ramanan. N-best maximal decoders for part models. In *ICCV*, 2011. 5, 6

[26] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *ICCV*, 2015. 2

[27] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *ACCV*, 2014. 2

[28] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet conditioned pictorial structures. In *CVPR*, 2013. 2

[29] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh. Pose machines: Articulated pose estimation via inference machines. In *ECCV*, 2014. 3

[30] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *ICLR workshop*, 2016. 7

[31] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015. 4

[32] J. Song, L. Wang, L. Van Gool, and O. Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *CVPR*, 2017. 2, 5, 6, 7

[33] Y. Tian, C. L. Zitnick, and S. G. Narasimhan. Exploring the spatial hierarchy of mixture models for human pose estimation. In *ECCV*, 2012. 2

[34] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 2

[35] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 2

[36] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1, 2, 3, 4, 5, 6

[37] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 2

[38] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *PAMI*, 35(12):2878–2890, 2013. 1, 2, 5

[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2

[40] W. Zhang, M. Zhu, and K. G. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, 2013. 2, 5