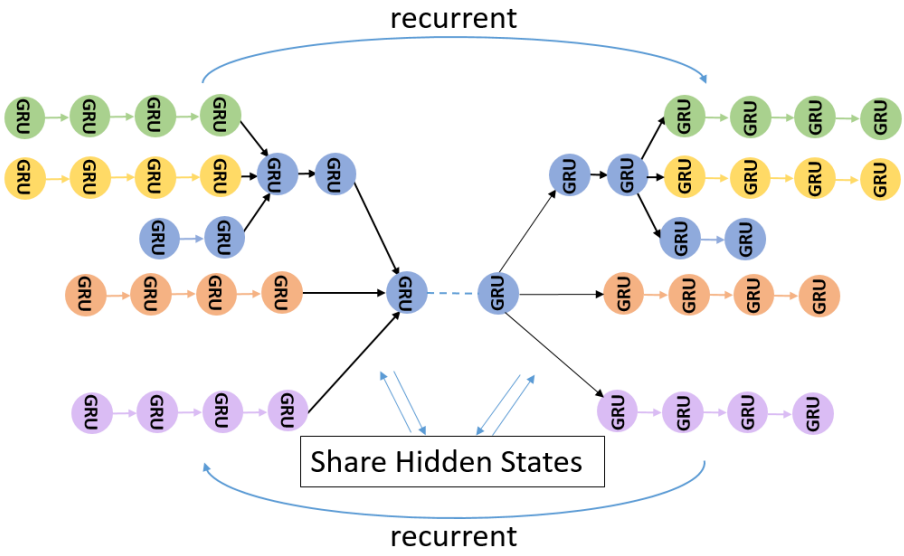
Supplementary Illustrations

A. Details of SeBiReNet



The architecture of the sequential bidirectional recursive network is shown as above figure, which includes two subnets: the left recursive subnet and the right diffuse subnet.

1) Role of each subnet

   The recursive subnet models the dependency from child nodes to parent node, i.e., $p(J_{parent} \mid J_{child})$. Therefore, the information flows from the child nodes to the parent node in recursive subnet. The recursive subnet plays a role of doing inverse kinematics analysis or obtain a semantic summary of a human pose at root node. It can be utilized for joint position estimation or pose/action classification.

   The diffuse subnet models the dependency from the parent node to the child node, i.e., $p(J_{child} \mid J_{parent})$. Therefore, the information flows from the parent node to the child node. The diffuse subnet can be used to imitate the forward kinematics analysis.

   In a word, the two subnets model the skeleton data from converse perspective and interact with each other through the shared hidden states.

2) Nodes

Each node in the SeBiReNet corresponding to a joint in the human skeleton structure. It can be a GRU cell, LSTM cell or some other types of neural cell. In our implementation, we select the GRU cell as we believe the forgetting mechanism of GRU cell can make the network more robust to noisy input.

The number of nodes depends on the joint number in the target human skeleton model. As most skeleton model contains 17 joints, we defined our SeBiReNet with 34 nodes (each subnet contains 17 nodes). The node is easy to add or delete by using the defined adding or deleting operation.

3) Inputs of the network

Inputs of the network is the states of skeleton joints which can be joint 2D/3D positions, velocities, accelerations or any other natural/calculated values according to the application scenario. In our experiments, we only use the joint 3D position, i.e., the (x, y, z) coordinates as input.

In our recursive subnet, the input of each node is the concatenation of the 3D position of corresponding joint and the output states of its child nodes, i.e., input $x_i^r = concat(p_i, h_{children})$

In the diffuse subnet, the input of each node is the concatenation of the 3D position of corresponding joint and the output state of its parent node, i.e., input $x_i^d = concat(p_i, h_{parent})$.

4) Inference process

Inference process of the proposed SeBiReNet is illustrated in detail in our attached video. The recursive subnet and diffuse subnet run alternately. The hidden states are shared between the recursive subnet and the diffuse subnet. In our experiments, we start the inference from recursive subnet.

We depict the inference process using the following algorithm chart:

Inference process of the SeBiReNet: Initialize all nodes state h_i^r and h_i^d with zero states. Therefore, the initial shared node states h_i is zero. $W_{xi}^r, W_{hi}^r, b_i^r, W_o^r, b_o^r$ are weights of the recursive subnet. $W_{xi}^d, W_{hi}^d, b_i^d, W_o^d, b_o^d$ are weights of the diffuse subnet.

- Sample minibatch of m examples $\left\{(\tilde{p}^{(1)}, p^{(1)}), (\tilde{p}^{(2)}, p^{(2)}), \cdots, (\tilde{p}^{(m)}, p^{(m)})\right\}$ from dataset

  for number of recurrent iterations **do**

  - In recursive subnet, input the corrupted human pose $\tilde{p}$

    From leaf nodes to root node, calculate

    $$x_i^r \leftarrow concat(p_i, h_{children})$$
    $$h_i^r \leftarrow \varphi(W_{xi}^r x_i^r + W_{hi}^r h_i + b_i^r)$$
    $$\mathrm{O}_i^r \leftarrow \mathrm{O}(W_o^r h_i^r + b_o^r)$$

    Update the shared hidden states and network outputs

    $$h_i \leftarrow h_i^r$$
    $$\mathrm{O}_i \leftarrow \mathrm{O}_i^r$$

  - In diffuse subnet, input the corrupted human pose $\tilde{p}$

    From root node to child nodes, calculate

    $$x_i^d \leftarrow concat(p_i, h_{parent})$$
    $$h_i^d \leftarrow \varphi(W_{xi}^d x_i^d + W_{hi}^d h_i + b_i^d)$$
    $$\mathrm{O}_i^d \leftarrow \mathrm{O}(W_o^d h_i^d + b_o^d)$$

    Update the shared hidden states and network outputs

    $$h_i \leftarrow h_i^d$$
    $$\mathrm{O}_i \leftarrow \mathrm{O}_i^d$$

  end for

B. Implementation Details

1. Hidden unit number of GRU cell

   The hidden unit number of GRU cell in SeBiReNet is 32 in our experiments. Actually, we also do experiments with 64 and 128 units version. Increasing the number of units doesn't bring much improvement of the performance but make the size of the network much larger. Therefore, we choose the 32 version as a trade-off selection.

2. Training time

   Our experiment is implemented with TensorFlow on a computer with 4 NVIDIA Titan Xp graphics cards, 64 GB RAM and an Intel Xeon(R) processor E5-2640. The batch size is 64. About 12.6 seconds need for each 100 steps. That is to say, only 126 ms is needed for each step. Totally, we trained 30 epochs in our experiments and about one and half an hour is needed for each training.

3. Corrupted skeletons

   In our experiments, we at most move 5 randomly selected joints of each skeleton to invalid positions. We believe that, if more joints are destroyed, it doesn't do any help for the network to learn intrinsic human skeleton feature. When randomly selected joints locate adjacent to each other, it will make the result even worse since even our human beings cannot tell the position of a missing arm or leg. Therefore, the network will try to remember the training samples and lead to overfitting.

   The invalid pose is obtained by moving the selected joints with a proper large displacement with respect to the largest bone length. As we don't know which joint will be selected, the large displacement is fixed for all joints to guarantee the achieved pose is illegal and for convenience. Though some joints only need a small displacement to move to illegal position, a large displacement will always lead to a violation of both bone length and joint motion limits.

4. Max frames in action recognition

   Northwestern-UCLA dataset: As the max frame number in this dataset is 201, we set the max time steps of LSTM to 201. For those samples with less frames, zero padding is utilized.

   NTU RGB+D dataset: We made a statistic analysis on this dataset and found that about 98% action samples have less than 200 frames but the max frame number is over 350. Hence, in this dataset we set the max time step to 200. Zero padding is used for those samples with less frames. The frames that over 200 are discarded.

5. Training protocols in action recognition

   Northwestern-UCLA dataset: It was recorded in a real lab scenario with a clutter background. The action categories performed in this dataset include: pick up with one hand, pick up with two hands, drop trash, walk around, sit down, stand up, donning, doffing, throw, carry. Each action is performed by 10 actors and captured from 3 different views. Totally, it contains 1494 motion sequences. The skeleton collected in this dataset is noisy because of frequent self-occlusion and thus quite challenging for skeleton-based action recognition. *Following previous works, we adopt the cross-view training protocol to train our action classifier on the first two views and test it on the data from the third view*.

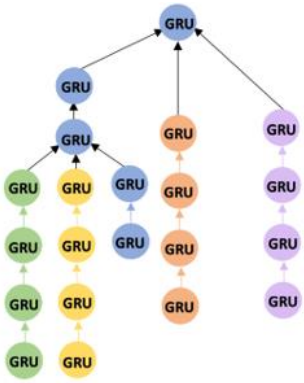
   NTU RGB+D dataset: It contains 60 kinds of actions including drink water, eat meal/snack, brushing teeth, brushing hair, and so on. These actions are performed by 40 subjects and generate more than 56,000 motion samples from various views. Actions in this dataset have a high inter-class similarity. Besides actions performed by single subjects, ten kinds of interaction between two persons, such as kick/push other people, are also included. *According to the author's instruction, 18960 action samples from camera 1 are used for testing and 37920 action sequences from camera 2 and 3 are selected for training*.

   As these two datasets are collected by Kinect sensor, many noisy skeletons exist. They are not used as a training set in the representation learning phase, though they are much large than the Cambridge-Imperial APE dataset which is collected with motion capture device.

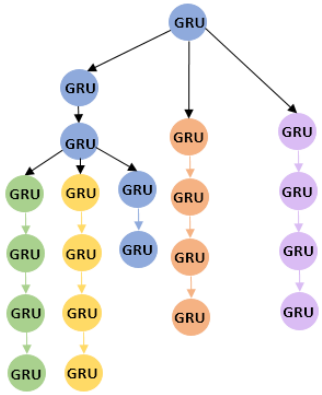
C. Some Network Architectures

1. Network architectures in Sec. 4.2 of the paper

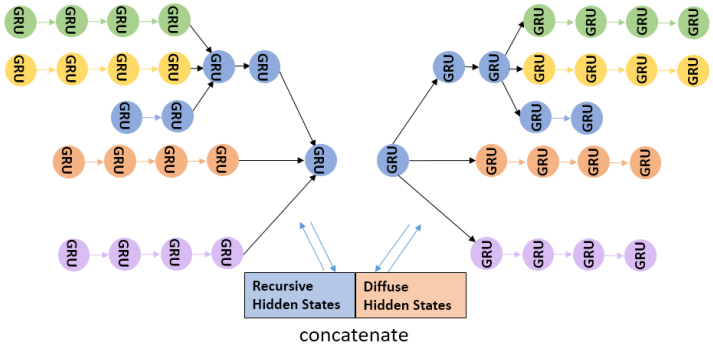
   In Sec. 4.2, we compared our architecture with several conventional structure designs. Here we show those structures in detail as below figures.



(a) conventional tree (only with recursive subnet)



(b) diffuse subnet



(c) concatenate recursive and diffuse subnet

To compare these structures, we only used these structures in the encoding part and the decoder is a 3 layer MLP. Therefore, the pose recovery performance only depends on the capability of these structures in encoder regardless the coupling effect between encoder and decoder. The learning architecture used in Table 1 is shown as below figure.

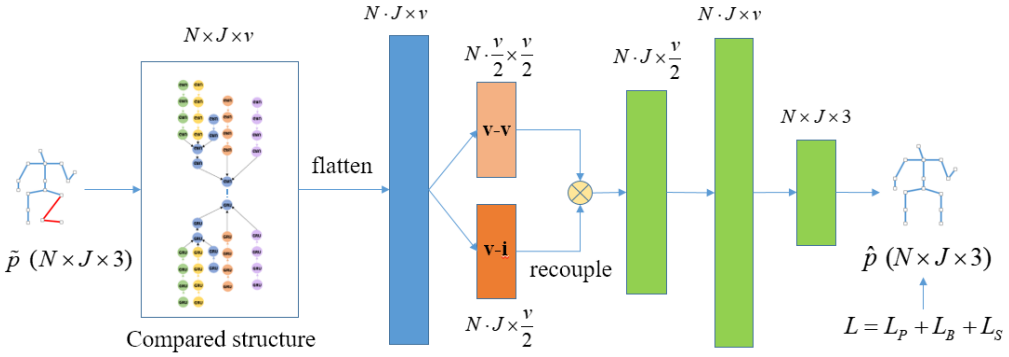


Fig. 1 Learning architecture used in the experiment of Table 1. The above number denotes the feature dimension output from each layer, where N is the batch size and J is the joint number. v-v represents the view-dependent feature and v-i represents the pose-dependent feature.

The proposed full architecture shown in Fig. 3 adds the SeBiReNet in the decoder. One of the stream is shown as below Fig. 2. Compared to structures as shown in Fig. 1, utilizing SeBiReNet both in the encoder and decoder achieves the best pose recovery performance (MPJPE 33.39mm). This result also demonstrates that the proposed SeBiReNet performs better than the MLP structure in processing the human skeleton data.

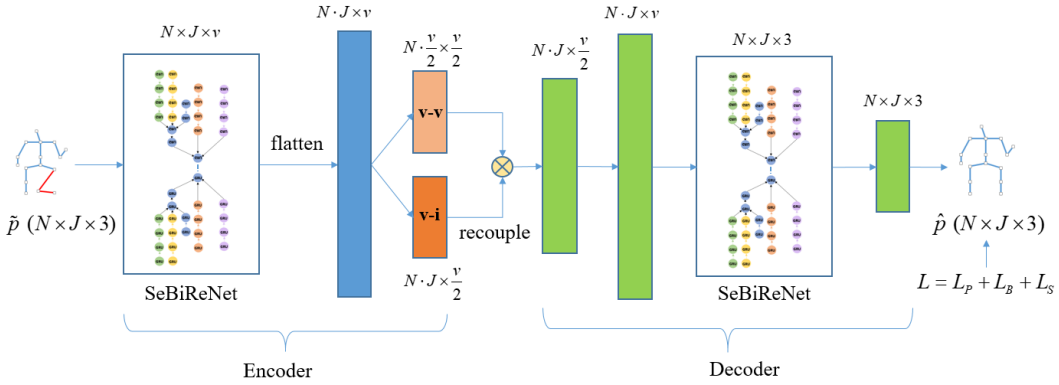
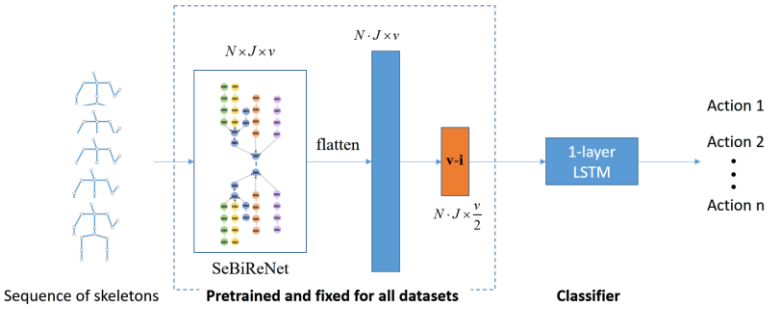


Fig. 2 One stream of the proposed architecture (which shown in Fig.3 in the manuscript): SeBiReNet is utilized both in the encoder and decoder

2. Network architecture used in the unsupervised cross-view action recognition task



The designed encoder is pretrained and fixed for all action datasets in unsupervised action recognition task. That is to say, our encoder is not fine-tuned in any action recognition dataset after training on the Cambridge APE dataset. It only used as a feature extractor in this task. Therefore, the performance of our method in action recognition and denoising unseen poses from action datasets shows the transferability of the learned representation across different datasets and different tasks (pose recovery and action recognition).