MixSynthFormer: A Transformer Encoder-like Structure with Mixed Synthetic Self-attention for Efficient Human Pose Estimation

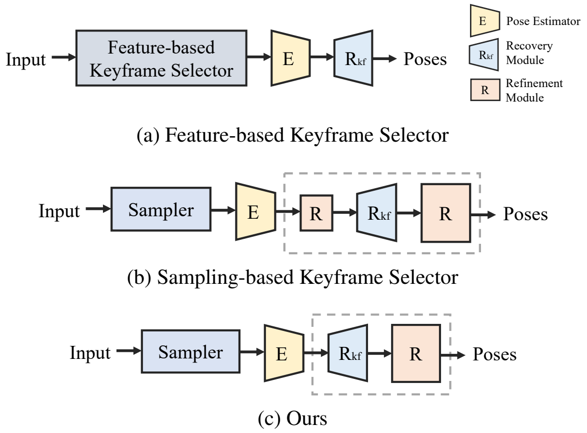
Yuran Sun, Alan William Dougherty, Zhuoying Zhang, Yi King Choi, Chuan Wu
The University of Hong Kong
yuransun@connect.hku.hk,{alanwd,tinawork}@hku.hk,{ykchoi,cwu}@cs.hku.hk

Abstract

Human pose estimation in videos has wide-ranging practical applications across various fields, many of which require fast inference on resource-scarce devices, necessitating the development of efficient and accurate algorithms. Previous works have demonstrated the feasibility of exploiting motion continuity to conduct pose estimation using sparsely sampled frames with transformer-based models. However, these methods only consider the temporal relation while neglecting spatial attention, and the complexity of dot product self-attention calculations in transformers are quadratically proportional to the embedding size. To address these limitations, we propose MixSynthFormer, a transformer encoder-like model with MLP-based mixed synthetic attention. By mixing synthesized spatial and temporal attentions, our model incorporates inter-joint and inter-frame importance and can accurately estimate human poses in an entire video sequence from sparsely sampled frames. Additionally, the flexible design of our model makes it versatile for other motion synthesis tasks. Our extensive experiments on 2D/3D pose estimation, body mesh recovery, and motion prediction validate the effectiveness and efficiency of MixSynthFormer. The code is available at https://github.com/ireneesun/MixSynthFormer.git

1. Introduction

Human pose estimation is a crucial task in computer vision which aims to estimate the keypoint locations of a human body from visual inputs. It has a wide range of applications in virtual/augmented reality, healthcare, and security surveillance [23, 5]. In recent years, there has been a growing demand for *real-time* pose estimation in many of these applications. However, the computational burden of deep learning models makes their real-time estimation on resource-constrained devices (e.g., CPU, mobile devices) challenging. How to estimate human poses in videos efficiently remains an open research problem.



(a) Feature-based Keyframe Selector

(b) Sampling-based Keyframe Selector

(c) Ours

Figure 1: Pipelines of keyframe-based pose estimation frameworks. The recover process is similar to interpolation. (a) is the feature-based pipeline [40, 7], which selects 30% to 40% of frames based on input features and recovers the whole sequence. (b) is the existing sampling-based pipeline [38], which uses a sampler to select frames and conducts keyframe refinement, whole sequence recovery and whole sequence refinement. (c) is our proposed simplified sampling-based pipeline excluding keyframe refinement.

Conventional pose estimators [3, 22, 41, 29, 4, 37, 27] estimate poses frame-by-frame, and the efficiency depends on the model design. Recent methods [40, 7, 38] have proven the viability of estimating poses in sequences based solely on keyframes. By leveraging the temporal redundancy in videos, poses in the remaining frames can be reconstructed from keyframe poses, similar to motion completion, which can significantly reduce the cost of resource-intensive pose estimators.

In current keyframe-based pose estimation frameworks, keyframes are selected using a feature-based selector or a sampler, as illustrated in Figures 1a and 1b. Feature-based keyframe selectors [40, 7] select "good" frames based on features extracted from inputs. The final poses in the whole sequence are recovered from the estimated poses in those frames without further refinements. A more effective approach is to sample a portion of frames directly from inputs

[38]. Without knowing the quality of selected frames in advance, poses detected from them may be noisy. To tackle this issue, the current design employs a refine-recover-refine pipeline to obtain the final pose sequence.

Is there a way to simplify the pipeline and further reduce the computational cost? The two refinement modules for keyframe poses and recovered poses appear repetitive and unnecessary, as the poses in keyframes are refined twice. To address this, we propose a simplified recover-refine pipeline, as shown in Figure 1c. Within our framework, keyframes are selected using a sampler, and the pose sequence is recovered from detected poses in keyframes using an interpolator. The rough sequence is then refined using a single refinement module.

We present a novel framework for sampling-based pose estimation utilizing a lightweight transformer encoder-like structure. The scaled dot product self-attention mechanism in the transformer network [34] is used to determine the relative importance of a token with respect to other tokens. However, the computational cost can be prohibitively high when using a large embedding size. To mitigate this problem, we replace the standard key-query attention with a synthetic self-attention module that generates attention weights using linear layers. To improve the quality of refinements, we synthesize both spatial and temporal attention matrices, which dynamically capture the inter-joint and inter-frame relationships. The features from both dimensions are combined and passed forward for further processing. We name our model *MixSynthFormer*. The main contributions of this work are summarized as follows:

▷ We propose a highly-efficient keyframe-based pose estimation model *MixSynthFormer* following the transformer encoder structure. It can effectively estimate poses from sampled keyframe poses in a recover-refine pipeline.

▷ We design an MLP-based mixed synthetic attention matrix generation module, *MixSynth Attention*, which generates attention matrices spatially and temporally from input representations. This design enables our model to dynamically fuse channel-wise and token-wise features and refine poses effectively. To speed up the computation, we introduce a reduction factor in the attention matrix generation, which further saves computation without compromising performance.

▷ We validate our model on 2D and 3D pose estimation and body recovery tasks with four datasets and five estimators. Experimental results demonstrate that our model outperforms all keyframe-based pose estimation frameworks in terms of accuracy and efficiency. Additionally, our model can adapt to other motion synthesis tasks by indicating different keyframes. We conduct experiments on short-term motion prediction as a sub-task, and results show that it is also competitive compared to state-of-the-art motion prediction models.

2. Related Work

2.1. Human Pose Estimation

Pose estimation models can be categorized into two types based on how frames are fed into the model: frame-by-frame and keyframe-based estimators. Common pose estimation frameworks estimate poses in every frame and optimize efficiency through knowledge distillation [22, 27] or model structure design [3, 41, 29, 4, 37]. However, the computational costs are still high due to the frame-by-frame estimation. Besides, these methods are sensitive to occlusion cases in which partial joints are invisible in the camera view due to complex poses or interactions with environments, resulting in incoherent poses regardless of incorporating temporal information. In contrast, keyframe-based pose estimation frameworks exploit human motion continuity and can produce smoother sequences. Only keyframes are used for heavy single-frame estimation, while poses in the rest of the frames are recovered by a lightweight module. Thus, they can boost the efficiency of single-frame pose estimators in different tasks, such as SimplePose [36] for 2D poses, FCN [25] for 3D poses, and SPIN [20], PARE [19] and EFT [17] for body recovery.

Keyframe-based methods can be further divided into two categories depending on how the keyframes are selected: feature-based and sampling-based. Feature-based methods [40, 7] select "good" keyframes relying on intermediate feature representations and recover the whole sequence without refinement. KFPN [40] selects frames that capture global context based on image features extracted from inputs. MAPN [7] uses the internal motion signals and residual errors to determine whether the heavy pose estimation needs to be conducted on a frame. Nevertheless, the cost of feature extraction cannot be ignored.

Sampling-based methods [38] select keyframes using a uniform, random or customized sampling strategy. As selected frame may not be "good", these frameworks add a refinement module to clean noisy poses. Deciwatch [38] uses a standard transformer encoder to refine the uniformly sampled poses and a decoder to refine the recovered pose sequence based on denoised poses. The refinement on sampled keyframe poses is redundant and can be merged with the subsequent refinement module in the decoder. Moreover, Deciwatch [38] focuses on the temporal aspects and does not take spatial dependencies into consideration, which reflect interdependence among joints. Understanding the patterns of how joints are moving together can facilitate the refinement of incorrectly estimated joints.

2.2. Motion Completion and Prediction

Motion completion aims to fill the motion of missing frames with specified keyframe constraints, presenting both past and future frames. Traditional methods utilize lin-

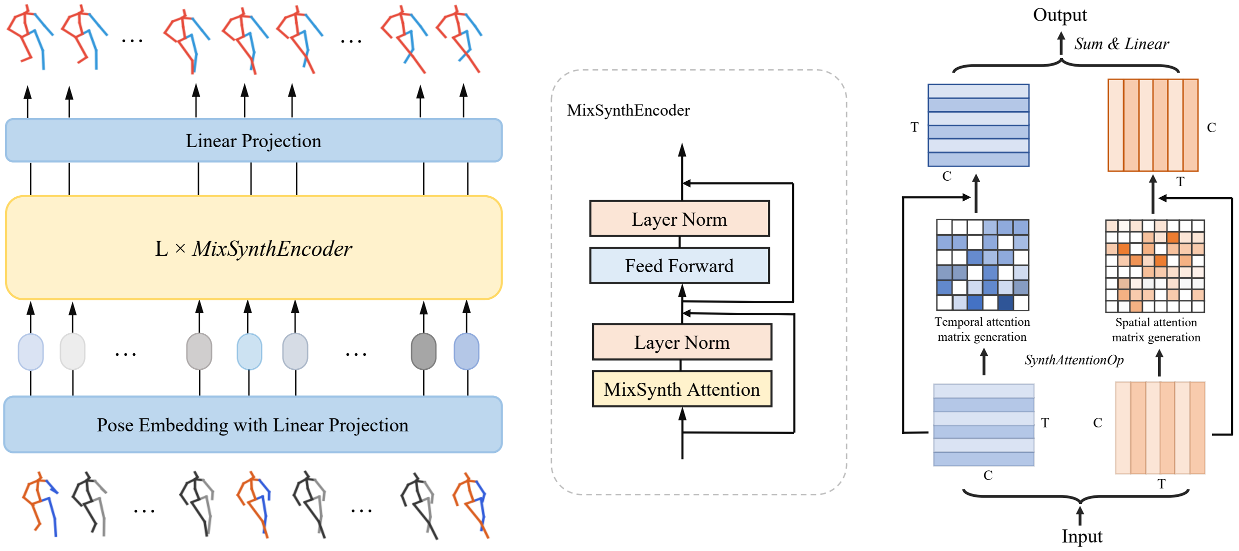


Figure 2: Overview of *MixSynthFormer*. *MixSynthFormer* recovers and refines poses from sparsely sampled keyframes. It consists of L blocks of *MixSynthEncoder* blocks and two linear projection layers. *MixSynthEncoder* follows a standard transformer encoder structure but replaces multi-head self-attention with *MixSynth Attention*. *MixSynth Attention* synthesizes spatial and temporal attention via *SynthAttention* operations (depicted in Figure 3). The output from the two branches are combined and forwarded to a linear layer for feature fusion. (Detected poses in keyframes are colored, and recovered poses by interpolators are in gray.)

ear, cubic-spline, Lagrange, and low-rank matrix completion, producing smooth transitions but lacking details [13]. Early learning-based methods employ RNN [9, 10] and convolutional models [12, 18]. Transformer-based models [6, 28, 30] are also adopted due to their exceptional performance in sequence-to-sequence problems. The aforementioned motion completion models are prevalent in the animation field and utilize ground-truth joint rotations and positions. However, in our case, the inputs are detected poses that can be noisy and aperiodic.

Motion prediction generates future motion given only historical frames. RNN [8, 24] and transformer-based [2, 1] methods are proposed to capture temporal dynamics. GCN-based methods, such as STSGCN [31] and STGAGCN [42], further model the spatial relationships among joints, obtaining high performance by closely combining spatio-temporal modeling.

Despite a range of frameworks, none of them treat motion completion and prediction as two mutually dependent tasks. Rather, each approach is specialized for one particular motion synthesis task. Notably, the sole distinction between motion completion and motion prediction pertains to the keyframe constraints. In this work, our model can handle both tasks with a unified structure.

3. Method

Problem Definition. Given an input video of T frames, a sampler samples one frame out of every K frames (T is a multiple of K). The sampled frames are regarded as keyframes where poses will be estimated by a pose estimator. The detected poses from sampled frames may be unreliable due to complex poses or occlusion. We denote poses detected from sampled frames as $\hat{\mathbf{X}}^{sampled}_{noisy} \in \mathbb{R}^{\frac{T}{K}\times P}$ which serves as the input to our framework. P is the number of pose parameters, which can denote joint positions (2D/3D) or rotations (6D) and may vary depending on the dataset and body representation. The objective of sampling-based pose estimation is to recover all the poses $\hat{\mathbf{X}} \in \mathbb{R}^{T\times P}$ in the original sequence from detected poses $\hat{\mathbf{X}}^{sampled}_{noisy}$ in the sampled frames (referred to as sampled detected poses).

Model Overview. Figure 2 provides an overview of our proposed model. The goal of *MixSynthFormer* is to recover and refine the poses to obtain accurate estimations efficiently. Initially, the entire pose sequence is recovered through linear interpolation or a linear layer from the sampled detected poses. Those recovered poses are transformed into pose embeddings using linear projection and are fed into L blocks of *MixSynthEncoder*. Each *MixSynthEncoder* comprises an attention block and a feed-forward network with residual connection, followed by a layer normalization layer. The attention block *MixSynth Attention* synthesizes spatial and temporal attention matrices concurrently. By integrating the spatial and temporal information, our model can refine the entire pose sequence in a simultaneous manner. Finally, the recovered poses are output via another linear layer.

We next present details of *MixSynthFormer*, starting with an overview of the structure and then proceeding to the design of *MixSynth Attention* and the detailed operations within it. Finally, we describe the settings of using our model on motion prediction tasks, showcasing its adaptability to other motion synthesis tasks.

3.1. Design of MixSynthFormer

We adopt a recover-refine pipeline in our framework. The whole pose sequence is preliminarily recovered along the temporal dimension by a traditional or a learned interpolator based on the detected poses $\hat{\mathbf{X}}_{noisy}^{sample}$ in keyframes, denoted as $\hat{\mathbf{X}}_{noisy}^{recover} \in \mathbb{R}^{T\times P}$:

$$\hat{\mathbf{X}}_{noisy}^{recover} = Interp(\hat{\mathbf{X}}_{noisy}^{sampled}) \tag{1}$$

Recovering poses from sparse and noisy inputs results in inaccurate estimates, which necessitates additional processing to obtain a clean output. Moreover, traditional interpolation methods tend to produce overly smooth motion sequences that lack details. Hence, a refinement step is necessary to generate a more precise pose sequence, which can be formulated as a sequence-to-sequence problem. Transformers have shown remarkable success in solving such problems owing to their ability to capture global correlations among tokens. We design *MixSynthFormer* following the workflow of a standard transformer encoder, except for replacing the self-attention module with *MixSynth Attention*. Following are the high-level operations of *MixSynthFormer*.

All poses in $\hat{\mathbf{X}}_{noisy}^{recover}$ are transformed to the pose embeddings $\mathbf{Z}_0 \in \mathbb{R}^{T\times C}$ through a linear layer with weights $\mathbf{W}_0 \in \mathbb{R}^{P\times C}$, where C is the dimension of the embedding space. Attention layers in transformers need position embeddings to specify the locations of input tokens, whereas our MLP-based *MixSynth Attention* is sensitive to the order of tokens in temporal attention synthesis like [33] and does not require additional position embeddings.

$$\mathbf{Z}_0 = \hat{\mathbf{X}}_{noisy}^{recover}\mathbf{W}_0 \tag{2}$$

The pose embeddings are then passed to L blocks of *MixSynthEncoder*. The model conforms to the conventional transformer encoder design, consisting of stacked *MixSynth Attention* module and feed-forward network (*FFN*) which is a two-layer point-wise fully connected layer with GELU activation [11]. Residual connection and layer normalization (*LN*) are applied sequentially after every block.

$$\hat{\mathbf{Z}}_l = LN(MixSynthAtt(\mathbf{Z}_{l-1}) + \mathbf{Z}_{l-1}) \tag{3}$$

$$\mathbf{Z}_l = LN(FFN(\hat{\mathbf{Z}}_l) + \hat{\mathbf{Z}}_l) \tag{4}$$

Here $l \in [1, L]$ denotes the *l-th* block. $\mathbf{Z}_{l-1}$ is the output from the previous encoder block, $\hat{\mathbf{Z}}_l$ is the intermediate representation and $\mathbf{Z}_l$ is the output of the current encoder block. $\mathbf{Z}_l$ and $\hat{\mathbf{Z}}_l$ have the same dimension in $\mathbb{R}^{T\times C}$.

After L blocks of operations, the features in the embedding space $\mathbf{Z}_L$ are transformed back to the pose dimension via another linear layer with weights $\mathbf{W}_p \in \mathbb{R}^{C\times P}$. The use of the residual connection can expedite model convergence and yield more stable results. The final recovered pose sequence $\hat{\mathbf{X}}$ is thereby the summation of the output from *MixSynthEncoder* blocks and the preliminarily recovered noisy poses $\hat{\mathbf{X}}_{noisy}^{recover}$:

$$\hat{\mathbf{X}} = (LN(\mathbf{Z}_L))\mathbf{W}_p + \hat{\mathbf{X}}_{noisy}^{recover} \tag{5}$$

3.2. MixSynth Attention

Transformers are effective in capturing global correlations among tokens, but they may not be sufficient for refining noisy sequences. Token-wise attention primarily focuses on the temporal aspect, while local attention that captures the correlation among joints is also essential, particularly for recovering complex motions. To learn both global and local features, we propose *MixSynth Attention*, which mixes spatial and temporal synthetic attention matrices. The weights in these matrices reflect the relative inter-joint and inter-frame importance. The spatial matrix reveals the correlation among joints. When performing some motion patterns, joints that move together are highly correlated. It is helpful in locating a wrongly-detected joint from its related joints with accurate detection. The temporal matrix determines the impact of a frame with respect to other frames in the sequence, signifying the extent to which a frame affects the remaining frames.

We explain the *MixSynth Attention* operations in detail. For simplicity, we eliminate the block index l in the following explanation. The synthetic attention operation (*SynthAttenOP*) operates on $\mathbf{Z} \in \mathbb{R}^{T\times C}$ and outputs feature $\mathbf{Y}_i \in \mathbb{R}^{T\times C}$, which can be spatial or temporal, denoted as $\mathbf{Y}_s$ and $\mathbf{Y}_t$, respectively.

$$\mathbf{Y}_i = SynthAttenOP_i(\mathbf{Z}), i \in \{s, t\} \tag{6}$$

The spatially and temporally attended features are generated simultaneously. Inspired by [39], we apply split attention to re-balance the contribution of the two branches. Different motions may have different emphases. Inter-joint relations are more important for certain motions, while others may be more influenced by inter-frame relations. The weights $\mathbf{A}_y \in \mathbb{R}^{2\times C}$ of $\mathbf{Y}_s$ and $\mathbf{Y}_t$ are calculated as

$$\mathbf{A}_y = Softmax(FFN(GAP(\mathbf{Y}_s + \mathbf{Y}_t))) \tag{7}$$

where GAP denotes the global average pooling function, FFN contains two fully connected layers with ReLU activation [26] and $Softmax$ is for normalization. The weighted sum $\mathbf{Y}_{out} \in \mathbb{R}^{T\times C}$ is the summation of the element-wise product of the concatenated representations from two branches with the re-balancing weights $\mathbf{A}_y$:

$$\mathbf{Y}_{out} = Sum(\mathbf{A}_y \circ [\mathbf{Y}_s; \mathbf{Y}_t]) \tag{8}$$

The output of the *MixSynth Attention* block is attained by linearly transforming $\mathbf{Y}_{out}$ with weights $\mathbf{W}_o \in \mathbb{R}^{C \times C}$:

$$MixSynthAtt(\mathbf{Z}) = \mathbf{Y}_{out}\mathbf{W}_o \tag{9}$$

3.3. SynthAttention Operation

The self-attention module in transformers utilizes the pairwise dot product to calculate attention weights. The computation of the query-key matrix multiplication expands quadratically as the embedding size increases. To alleviate the impact of embedding size on computational cost, we propose synthetic attention operation (*SynthAttenOp*), whose computation grows linearly with the increase of embedding size, as a substitute for the standard attention. We discuss this part in detail in supplementary materials. *SynthAttenOP* generates attention matrices using linear layers. Details of its operation are described below.

SynthAttenOp operates spatially and temporally. In spatial attention synthesis, input and output are transposed. Here we use $\mathbf{Z} \in \mathbb{R}^{N \times D}$ to illustrate the general case, where N and D can be substituted with either C or T. The coarse synthetic attention matrix $\mathbf{A}_s \in \mathbb{R}^{N \times N}$ is generated by a linear layer with weights $\mathbf{W}_z \in \mathbb{R}^{D \times N}$:

$$\mathbf{A}_s = \mathbf{Z}\mathbf{W}_z \tag{10}$$

As different frames or joints have different contributions, we apply the Squeeze-and-Excitation layer (*SELayer*) [14] to regulate the influence. It can amplify critical channels while mitigating the effect of less important ones. The fine-grained attention matrix $\mathbf{A}_{synth} \in \mathbb{R}^{N \times N}$ is obtained by

$$SELayer(\mathbf{A}_s) = \mathbf{A}_s \circ \sigma(\mathbf{W}_e \delta(\mathbf{W}_s GAP(\mathbf{A}_s))) \tag{11}$$

$$\mathbf{A}_{synth} = Softmax(SELayer(\mathbf{A}_s)) \tag{12}$$

where GAP denotes global average pooling and $GAP(\mathbf{A}) \in \mathbb{R}^N$, δ and σ are Sigmoid activation and ReLU functions [26]. The reduction ratio (r_{se}) is an empirically found hyperparameter that limits model complexity [14]. $\mathbf{W}_s \in \mathbb{R}^{\frac{N}{r_{se}} \times N}$ is for dimension reduction, and $\mathbf{W}_e \in \mathbb{R}^{N \times \frac{N}{r_{se}}}$ is for expanding to the original dimension. The output of *SELayer* is a re-scaled input after channel-wise multiplication with calculated weights. $Softmax$ is inherited from the standard transformer, which normalizes the attention weights.

In alignment with the standard transformer encoder, the output is the product of the attention matrix and the value:

$$\mathbf{Y} = \mathbf{A}_{synth}\mathbf{V} \tag{13}$$

where $\mathbf{V} = \mathbf{Z} \in \mathbb{R}^{N \times D}$ and $\mathbf{Y} \in \mathbb{R}^{N \times D}$.

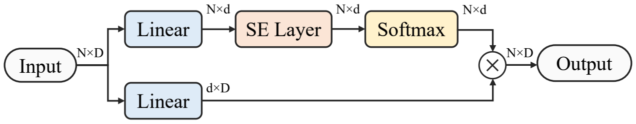


Figure 3: The procedure of reduced *SynthAttenOp*

The method described above is applicable for cases when N is relatively small. When N is large, the computation cost grows quadratically owing to the matrix multiplication in Eqn. 13. We apply a reduction factor r to the last dimension D, saving the computation in Eqn. 13 by r^2 times. We denote the reduced dimension as $d = \lfloor N/r \rfloor$. Figure 3 shows the process of reduced attention matrix generation and feature fusion. The reduced coarse attention matrix $\mathbf{A}_s^r \in \mathbb{R}^{N \times d}$ and reduced value $\mathbf{V}^r \in \mathbb{R}^{d \times D}$ are obtained through

$$\mathbf{A}_s^r = \mathbf{Z}\mathbf{W}_z^r, \mathbf{V}^r = \mathbf{W}_v\mathbf{Z} \tag{14}$$

where weights for reduced attention matrix generation and for reducing the value are $\mathbf{W}_z^r \in \mathbb{R}^{D \times d}$ and $\mathbf{W}_v \in \mathbb{R}^{d \times N}$, respectively.

3.4. Efficiency Calculation

The overall cost of the model per frame combines the cost of a pose estimator and the cost of *MixSynthFormer*:

$$FLOPs = \frac{1}{K} * f(Est.) + f(MSF) \tag{15}$$

$f(\cdot)$ evaluates the FLOPs of a model per frame. As one frame out of K frames is sampled for estimation, the per-frame cost of the estimator is $\frac{1}{K} * f(Est.)$. *MixSynthFormer* operates on every frame, recovering and refining poses on the whole sequence, and its FLOPs per frame is $f(MSF)$. In particular, $f(MSF)$ is at least 10^4 times smaller than $f(Est.)$ and is negligible compared to the heavy pose estimator. With 10% sampling ratio ($K = 10$), our framework can speed up the computation by ten times.

3.5. Loss Function

Following [38], we apply L1 loss to minimize the error between the finely recovered pose sequence $\hat{\mathbf{X}}$ and the ground-truth pose sequence $\mathbf{X}$:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} |\hat{\mathbf{X}}^t - \mathbf{X}^t| \tag{16}$$

3.6. Motion Prediction Setting

Motion prediction and motion completion are two subfields in conditional motion generation with the only difference lying in the keyframe constraints. We include short-term motion prediction as a sub-task and demonstrate the versatility of our model on this task.

The input of motion prediction to our model is a historical ground-truth motion sequence $\mathbf{X}_{1:T_h} \in \mathbb{R}^{T_h \times P}$, where

T_h is the number of consecutive historical frames and P is the size of pose parameters (i.e., 3D positions of human body joints). The goal is to predict the future sequence $\mathbf{X}_{T_h+1:T_h+T_f} \in \mathbb{R}^{T_f \times P}$ given $\mathbf{X}_{1:T_h}$, where T_f is the number of frames to be predicted. Similar to the pose estimation settings, the future poses are coarsely generated by interpolation, and then *MixSynthFormer* refines and outputs the future poses.

We adopt the L2-norm between the predicted sequence and the ground-truth sequence calculated only on future frames as the loss function, which is equivalent to the Mean Per Joint Position Error (MPJPE) in [31, 42]:

$$\mathcal{L}_f = \frac{1}{T_f} \sum_{t=T_h+1}^{T_h+T_f} ||\hat{\mathbf{X}}^t - \mathbf{X}^t||_2 \tag{17}$$

We will test the above model in experiments, which showcases our model's adaptability to other synthesis tasks.

4. Experiments

Datasets. We evaluate *MixSynthFormer* on four datasets for three tasks. For 2D pose estimation (to locate the 2D coordinates of joints from images), we use the Sub-JHMDB dataset [16] that contains daily and sports actions. For 3D pose estimation (to extend 2D pose data with depth information prediction), we use the widely-used Human3.6M indoor dataset [15]. For 3D body recovery (to estimate human shape and joint angles), we use the in-the-wild dataset 3DPW which includes videos taken by a moving phone camera [35] and AIST++ [21] with diverse dancing actions shot indoors.

Evaluation Metrics. We adopt the same evaluation metrics as in [38]. For 2D pose estimation, Percentage of the Correct Keypoints (PCK) is used, and three thresholds (20%, 10% and 5% of the bounding box size under the pixel level) are set. For 3D tasks, Mean Per Joint Position Error (MPJPE) is utilized to report distance error and the mean Acceleration error (Accel) is adopted to reflect the smoothness of the estimated pose sequence. Mean Floating Point Operations (FLOPs (G)) per frame are also reported to show the computation costs.

Implementation Details. We use the same single-frame pose estimators and sampling strategy as in [38]. The uniform sampling ratio is set to 10% ($K = 10$) in all experiments. The presence of the first and last frames is vital in recovery. In practice, we sample Q frames with index $K \times q, q \in [0, \ldots, Q-1]$ from the input of length $K \times Q+1$ and additionally include the last frame of the input sequence as keyframes. The number of keyframes Q, the embedding dimension C and the number of *MixSynthEncoder* blocks L vary among datasets and tasks. We train the model on a single A100 GPU with testing performed on an 8-core CPU Apple M1 Pro chip to obtain the inference time. Due to page limit, other detailed training settings and qualitative results are provided in supplementary materials.

4.1. Comparison on 2D Pose Estimation

Existing keyframe-based pose estimation methods use SimplePose [36] as the pose estimator. We follow the same experiment settings as in [40, 7, 38]. Table 1 presents the detailed PCK@0.2 results of each joint, the average performance of all three metrics and computation costs.

Compared with feature-based methods, KFP [40] and MAPN [7], our method improves the accuracy by over 4.6% in PCK@0.2. The improvements on fast-moving joints, such as elbows, wrists and ankles, are significant with a maximum increase of 14.8%. Those joints often cause image blur, leading to wrong detection results. Our sampling-based method incorporates a refinement module and can mitigate the effects of incorrect detection.

Compared to the sampling-based model DeciWatch [38], our model features a lighter design with 0.4M FLOPs computation, yet achieving a superior accuracy of 99.3% under PCK@0.2. Notably, our model achieves almost 100% in predicting the head and torso positions, which can be attributed to the integration of spatial dependencies. While the attention mechanism in [38] only considers inter-frame relations, our method also leverages inter-joint relations, which is particularly useful for limb position estimation. The accuracy of ankle increases from 96.5% to 97.8%. By fusing information from highly-correlated joints, our model also outperforms DeciWatch in strict metrics PCK@0.1 by 1% and PCK@0.05 by 1.3%.

4.2. Comparison on 3D Pose Estimation

We next compare our method on 3D keypoint estimation and body recovery tasks against single-frame estimators [25, 20, 19, 17] and the sampling-based model DeciWatch [38]. Results on localization error and acceleration error are in Table 2.

Compared with DeciWatch [38], our model is significantly smaller with no performance degradation, thanks to *SynthAttention*. Our model achieves a 1-2mm reduction in MPJPE on Human3.6M and 3DPW without compromising smoothness. Notably, the FLOPs of the model trained on 3DPW are 0.03M, which is more than ten times smaller than DeciWatch. The inference time is less than 0.1 ms per frame on CPU. There are two main reasons for this: first, we only do refinements once; second, the attention matrix in our model is generated by the cost-efficient *SynthAttenOp*.

On AIST++, our model reduces the acceleration error by 33% compared to DeciWatch and 86% compared to the single-frame estimator SPIN. We attribute the acceleration reduction to the integration of spatial and temporal attention. AIST++ contains poses with fast movements and diverse dancing actions. Due to the complexity of these poses,

Table 1: Comparison with keyframe-based pose estimation methods on Sub-JHMDB dataset [16] for 2D pose estimation. R means ResNet [39]. Ratio represents the sampling ratio. *Sho., Elb., Wri., Ank., Avg.* stand for shoulder, elbow, wrist, ankle and average PCK, respectively.

Sub-JHMDB dataset - 2D Pose Estimation PCK@0.2											PCK@ 0.1	PCK@ 0.05
Methods	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Avg.	FLOPs(G)	Ratio		
KFP(R18) [40]	94.7	96.3	95.2	90.2	96.4	95.5	93.2	94.5	7.19	40.8%	-	-
MAPN(R18) [7]	98.2	97.4	91.7	85.2	99.2	96.7	92.2	94.7	2.70	35.2%	-	-
SimplePose [36]	98.6	97.3	96.5	98.6	90.3	95.4	86.5	94.0	11.96	100%	81.6	57.3
DeciWatch [38]	99.8	99.5	99.7	99.7	98.7	99.4	96.5	98.8	1.196+0.0005	**10.0%**	94.1	79.4
Ours	**99.9**	**99.9**	**99.8**	**100.0**	**99.2**	**99.9**	**97.8**	**99.3**	**1.196+0.0004**	**10.0%**	**95.1**	**80.7**

Table 2: Comparison for 3D pose estimation and body recovery on Human3.6M [15], 3DPW [35] and AIST++ [21]. The estimators in keyframe-based frameworks are the same as the corresponding single-frame pose models.

Methods	MPJPE	Accel	FLOPs(G)
Human3.6M - 3D Pose Estimation			
FCN [25]	54.6	19.2	6.21
DeciWatch [38]	52.8	**1.5**	0.621+0.0007
Ours	**50.9**	**1.5**	**0.621+0.0001**
3DPW - 3D Body Recovery			
SPIN [20]	96.6	34.7	4.13
DeciWatch [38]	93.3	7.1	0.413+0.0004
Ours	**91.2**	**6.8**	**0.413+0.00003**
EFT [17]	90.3	32.8	4.13
DeciWatch [38]	89.0	6.8	0.413+0.0004
Ours	**88.1**	**6.3**	**0.413+0.00003**
PARE [19]	78.9	25.7	15.51
DeciWatch [38]	77.2	6.9	1.551+0.0004
Ours	**76.5**	**6.7**	**1.551+0.00003**
AIST++ - 3D Body Recovery			
SPIN [20]	107.7	33.8	4.13
DeciWatch [38]	71.3	7.1	0.413+0.0007
Ours	**71.2**	**4.7**	**0.413+0.0005**

original detections may contain erroneous estimations. By mixing spatial and temporal attention, we can effectively refine incorrect joints while maintaining the smoothness.

4.3. Comparison on Motion Completion

Our model recovers and refines poses in a completion-like manner. We compare our model with traditional interpolation methods and a transformer-encoder based Motion Completion Model (MCT) [6] from the animation field. MCT can handle different motion completion tasks, and uses the concatenation of positional and rotational data as input. To be consistent with the pose estimation task, we re-implement the MCT model and tune it on our datasets. Our recover-refine process is similar to the in-filling task in MCT. Both are to recover missing poses from uniformly sampled keyframes.

Table 3: Comparison with motion completion methods on Human3.6M [15], 3DPW [35] and AIST++ [21] with estimators stated inside brackets. Nearest, linear, quadratic, cubic-spline methods are used for traditional interpolation. MPJPE (Accel) of different recovery methods are reported. The results from traditional interpolators with the lowest MPJPE are underlined.

Data	Nea.	Lin.	Qua.	Cub.	MCT	Ours
H36M	57.0	54.8	54.6	54.6	54.5	**50.9**
(FCN)	(10.2)	(1.9)	(1.6)	(1.5)	(2.1)	**(1.5)**
3DPW	100.5	97.2	97.5	97.7	91.9	**91.2**
(SPIN)	(19.9)	(6.7)	(6.2)	(6.3)	(8.0)	**(6.8)**
AIST	114.7	105.9	107.1	107.6	72.0	**71.2**
(SPIN)	(28.5)	(5.7)	(4.6)	(4.6)	(6.9)	**(4.7)**

The results are presented in Table 3. Traditional interpolators tend to smooth the sequence but lack the ability to correct noisy poses, resulting in low acceleration errors and high position errors. MCT can reduce position errors at the expense of reduced smoothness. Our model surpasses all other methods. It can produce smooth sequences like interpolation but with more detailed and accurate motions. Specifically, when applied to Human3.6M, our model achieves the same smoothness as cubic-spline interpolation and decrease MPJPE by 7%. When evaluated on AIST++, our model reduces MPJPE by over 30%. The significant improvement in position errors without tradeoffs in smoothness shows the superiority of *MixSynthFormer* in recovering noisy sequences.

4.4. Comparison with Motion Prediction Methods

We evaluate our model on the sub-task short-term motion prediction (Sec. 3.6) under four testing intervals ranging from 80 ms to 400 ms, using 400 ms input sequences. Table 4 gives the results.

Our model outperforms STSGCN [31] and STGAGCN [42]. Taking advantage of motion continuity, our model can predict short-term poses that conform to the motion pattern of the historical input poses. It reduces error by 17% in an 80 ms interval while the improvement is less than 3% in a 400 ms interval. Limited by refinement on the overly-smoothed interpolation results, our model cannot learn spe-

Table 4: Comparison for short-term motion prediction. Average MPJPE across frames on Human3.6M [15] are reported.

Methods	80	160	320	400
STSGCN [31]	10.1	17.1	33.1	38.3
STGAGCN [42]	10.1	16.9	32.5	38.5
Ours	**8.4**	**15.8**	**30.5**	**37.4**

Table 5: Comparison with different attention matrix generation methods on 3DPW [35]

Methods	MPJPE	Accel	#Param	FLOPs
Vanilla	92.6	6.8	0.14M	0.12M
Random	92.6	6.9	0.12M	0.09M
Dense	92.5	6.8	0.13M	0.11M
Ours	**92.1**	**6.8**	**0.10M**	**0.09M**

cific periodic motion patterns and high-frequency motions. The uncertainty of human motions grows as the prediction interval increases, leading to a downgraded performance in longer intervals. Nevertheless, it yields improvements on complex actions or actions with occlusion like sitting and taking photos. Action-wise results are reported in supplementary materials.

4.5. Ablation Study

We conduct the ablation study on attention matrix generation, reduction factor r and model structure. The input sequences are linearly interpolated from keyframe poses.

Attention Matrix Generation. The attention matrix in our model is generated by a linear layer followed by a SELayer. It can also be generated by a FFN or is a learnable matrix as proposed in [32], referred to as Dense and Random. We compare our generation method with these two methods, together with the vanilla self-attention in transformers. To ensure a fair evaluation, we exclusively focus on the temporal attention matrix. Table 5 presents a comparative analysis of various attention methods, among which our generation approach stands out with the lowest number of parameters (0.1M) and minimum FLOPs (0.09M), owing to the reduction factor. Importantly, our method achieves the lowest MPJPE of 92.1 mm.

Effect of Reduction Factor. We examine the impact of the reduction factor r which is introduced for computation saving in *SynthAttention*. Specifically, we investigate how r in spatial (r_s) and temporal (r_t) dimensions affects the model size and its performance. As shown in Table 6, overall, applying r on one dimension saves 8% of the computation and up to 25% of parameters. When r is applied in temporal attention synthesis, we observe a smoother motion sequence with a decrease in MPJPE by 0.8 mm. Applying reduction on both dimensions further reduces the model parameters by 30% compared to a non-reduced one at the expense of slightly increased MPJPE.

Table 6: Effects of the reduction factor r on Human3.6M [15]

r_t	r_s	MPJPE	Accel	#Param	FLOPs
1	1	52.2	1.7	0.26M	0.13M
4	1	**51.4**	**1.6**	0.21M	0.12M
1	4	51.8	1.7	0.23M	0.12M
4	4	51.8	**1.6**	**0.18M**	**0.11M**

Table 7: Effects of components in *MixSynth Attention* block on Sub-JHMDB [16]. w/o means that the component is removed.

	PCK@0.2	PCK@0.1
Ours	**99.3**	**95.1**
w/o SE Layer	99.0	94.2
w/o weighted sum	98.9	94.7
w/o temporal attention	98.8	94.4
w/o spatial attention	98.6	92.9

Components in MixSynth Attention Block. *MixSynth Attention* blocks mix synthesized spatial and temporal attention. SELayer is incorporated to augment feature efficacy by emphasizing salient features while suppressing uninformative ones during attention matrix generation. The final output is the weighted sum of spatial and temporal representations. To evaluate the contribution of each component, we conduct ablation experiments by removing one of the components.

Results under different settings are summarized in Table 7. Removing any component in *MixSynth Attention* leads to worse performance. When SELayer is removed, PCK@0.1 decreases by 0.9%, indicating its crucial role in re-calibrating attention weights. The most significant performance decrease is observed when the spatial attention component is excluded: PCK@0.1 drops from 95.1% to 92.9%. This finding highlights the usefulness of the inter-joint relation from spatial attention generation in refinement. Moreover, combining spatial and temporal attention through a weighted sum further improves the performance by 0.4%. These results demonstrate the effectiveness of all components in *MixSynth Attention*.

5. Conclusion

In this paper, we propose *MixSynthFormer*, a transformer encoder-like structure with mixed synthetic attention for efficient human pose estimation in videos. The MLP-based synthetic attention module generates spatial and temporal matrices simultaneously, allowing fast and accurate recovery and refinement. The flexible model design also enables it to easily adapt to other motion synthesis tasks such as short-term motion prediction. Comprehensive experiments demonstrate the superior performance of our model in 2D/3D pose estimation, body mesh recovery, and motion prediction tasks, making it a promising solution for practical use cases.

Acknowledgement. This work was supported in part by Hong Kong Innovation and Technology Support Programme Platform Research Project fund (ITS/332/21FP).

References

[1] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction. In *2021 International Conference on 3D Vision (3DV)*, pages 565–574. IEEE, 2021. 3

[2] Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, et al. Learning progressive joint propagation for human motion prediction. In *Proceedings of the European Conference on Computer Vision*, pages 226–242. Springer, 2020. 3

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017. 1, 2

[4] Sangbum Choi, Seokeon Choi, and Changick Kim. Mobilehumanpose: Toward real-time 3d human pose estimation in mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2328–2338, 2021. 1, 2

[5] Yann Desmarais, Denis Mottet, Pierre Slangen, and Philippe Montesinos. A review of 3d human pose estimation algorithms for markerless motion capture. *Computer Vision and Image Understanding*, 212:103275, 2021. 1

[6] Yinglin Duan, Yue Lin, Zhengxia Zou, Yi Yuan, Zhehui Qian, and Bohan Zhang. A unified framework for real time motion completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4459–4467, 2022. 3, 7

[7] Zhipeng Fan, Jun Liu, and Yao Wang. Motion adaptive pose estimation from compressed videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11699–11708, 2021. 1, 2, 6, 7

[8] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 3

[9] Félix G Harvey and Christopher Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, pages 1–4. 2018. 3

[10] Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60–1, 2020. 3

[11] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[12] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016. 3

[13] Samuel J. Howarth and Jack P. Callaghan. Quantitative assessment of the accuracy for three interpolation techniques in kinematic analysis of human movement. *Computer Methods in Biomechanics and Biomedical Engineering*, 13(6):847–855, 2010. PMID: 21153975. 3

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 5

[15] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2013. 6, 7, 8

[16] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3192–3199, 2013. 6, 7, 8

[17] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human model fitting towards in-the-wild 3d human pose estimation. In *International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2021. 2, 6, 7

[18] Manuel Kaufmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges. Convolutional autoencoders for human motion infilling. In *International Conference on 3D Vision (3DV)*, pages 918–927. IEEE, 2020. 3

[19] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. Pare: Part attention regressor for 3d human body estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11127–11137, 2021. 2, 6, 7

[20] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2252–2261, 2019. 2, 6, 7

[21] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++, 2021. 6, 7

[22] Zheng Li, Jingwen Ye, Mingli Song, Ying Huang, and Zhigeng Pan. Online knowledge distillation for efficient pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11740–11750, 2021. 1, 2

[23] Wu Liu, Qian Bao, Yu Sun, and Tao Mei. Recent advances of monocular 2d and 3d human pose estimation: a deep learning perspective. *ACM Computing Surveys*, 55(4):1–41, 2022. 1

[24] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017. 3

[25] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017. 2, 6, 7

[26] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, 2010. 4, 5

[27] Xuecheng Nie, Yuncheng Li, Linjie Luo, Ning Zhang, and Jiashi Feng. Dynamic kernel distillation for efficient pose estimation in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6942–6950, 2019. 1, 2

[28] Boris N. Oreshkin, Antonios Valkanas, Félix G. Harvey, Louis-Simon Ménard, Florent Bocquelet, and Mark J. Coates. Motion inbetweening via deep δ-interpolator, 2022. 3

[29] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019. 1, 2

[30] Jia Qin, Youyi Zheng, and Kun Zhou. Motion in-betweening via two-stage transformers. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022. 3

[31] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. Space-time-separable graph convolutional network for pose forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11209–11218, 2021. 3, 6, 7, 8

[32] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pages 10183–10192. PMLR, 2021. 8

[33] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24261–24272. Curran Associates, Inc., 2021. 4

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 2

[35] Timo von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision*, September 2018. 6, 7, 8

[36] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision*, pages 466–481, 2018. 2, 6, 7

[37] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10440–10450, 2021. 1, 2

[38] Ailing Zeng, Xuan Ju, Lei Yang, Ruiyuan Gao, Xizhou Zhu, Bo Dai, and Qiang Xu. Deciwatch: A simple baseline for 10x efficient 2d and 3d pose estimation. In *Proceedings of the European Conference on Computer Vision*. Springer, 2022. 1, 2, 5, 6, 7

[39] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2736–2746, 2022. 4, 7

[40] Yuexi Zhang, Yin Wang, Octavia Camps, and Mario Sznaier. Key Frame Proposal Network for Efficient Pose Estimation in Videos. In *Proceedings of the European Conference on Computer Vision*, August 2020. 1, 2, 6, 7

[41] Lin Zhao, Nannan Wang, Chen Gong, Jian Yang, and Xinbo Gao. Estimating human pose efficiently by parallel pyramid networks. *IEEE Transactions on Image Processing*, 30:6785–6800, 2021. 1, 2

[42] Chongyang Zhong, Lei Hu, Zihao Zhang, Yongjing Ye, and Shihong Xia. Spatio-temporal gating-adjacency gcn for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6447–6456, 2022. 3, 6, 7, 8