Detecting Arbitrary Intermediate Keypoints for Human Pose Estimation with Vision Transformers

Katja Ludwig Philipp Harzig Rainer Lienhart
Machine Learning and Computer Vision Lab, University of Augsburg
{katja.ludwig, philipp.harzig, rainer.lienhart}@uni-a.de

Abstract

Most human pose estimation datasets have a fixed set of keypoints. Hence, trained models are only capable of detecting these defined points. Adding new points to the dataset requires a full retraining of the model. We present a model based on the Vision Transformer architecture that can detect these fixed points and arbitrary intermediate points without any computational overhead during inference time. Furthermore, independently detected intermediate keypoints can improve analyses derived from the keypoints such as the calculation of body angles. Our approach is based on TokenPose [9] and replaces the fixed keypoint tokens with an embedding of human readable keypoint vectors to keypoint tokens. For ski jumpers, who benefit from intermediate detections especially of their skis, this model achieves the same performance as TokenPose on the fixed keypoints and can detect any intermediate keypoint directly.

1. Introduction

In many sports disciplines, video analysis is a popular technique to evaluate and improve the performance of the athletes. This video analysis is often based on the location of certain keypoints of interest in the video frames. Ski jumpers, for example, use the keypoint locations to calculate body angles in order to evaluate their body posture during the flight and achieve long jumping distances. 2D human pose estimation techniques can automate the detection of the keypoint locations, which makes the video analysis less time consuming and available to more athletes. Ski jumpers are mainly interested in angles between body parts or skis, hence intermediate keypoints can help to get a more robust estimation of the angles. With more keypoints, a body part or ski angle is not solely based on the detection of the two end keypoints, but can be calculated as a mean of angles between arbitrary keypoints on the body part or ski. Other analyses that are based on intermediate keypoints or parameters derived from the keypoints can also be improved as the detections of intermediate keypoints can be used in conjunction with the interpolations based on the standard keypoints.

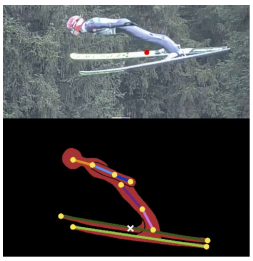

Figure 1. Intermediate keypoint detection. The white cross on the silhouette in the lower image shows the selection of the keypoint. In the upper video frame, the corresponding detected keypoint is displayed with a red circle. The user can move the white cross in the silhouette to change the definition of the keypoint.

In general, 2D human pose estimation is a task of high interest in computer vision research. Most common are architectures that use deep convolutional neural networks because of their high performance in visual tasks. Typically, these models are trained on a dataset with a fixed keypoint definition and the goal is to achieve a localization of these defined keypoints as precisely as possible. The convolutional neural networks learn low and high level features in their backbones combined with a head network that learns to extract detection heatmaps for each defined keypoint based on the backbone features. These head networks are fixed to the defined keypoints. Adding new keypoints requires a full new training of the network head. Recently, Transformer [14] networks have emerged from natural language processing tasks to vision tasks. In language applications, all words are at first embedded to vectors of fixed dimensions. Transformers can handle input sequences of various length, which is useful for sentences. Vision Transformer [4] architectures split images in patches and embed these image patches to vectors like the words. In order to

detect keypoints, TokenPose [9] appends additional learnable vectors to that sequence of embedded image patches. Each of these tokens corresponds to a defined keypoint in the dataset. Our method leverages the variable sequence length as it provides the possibility to append and remove tokens as needed. This is not possible in the TokenPose model because the network relies on the intermediate representations of all tokens to generate suitable predictions. Furthermore, we extend the TokenPose architecture so that we can train on arbitrary intermediate points. We learn a linear transformation of vectors representing the desired keypoints to the embedding space instead of tokens representing the fixed keypoints. Therefore, we can design the keypoint vector during inference time according to the keypoints that we like to detect. Figure 1 shows such an intermediate keypoint detection. [9]

The contributions of this work can be summarized as follows:

- Our proposed training method makes the TokenPose predictions independent of the provided keypoint tokens. Trained with our method, it can detect a subset of the known keypoints without the necessity of all tokens being present.
- We extend the TokenPose model to detect arbitrary intermediate points. The desired points are encoded in human understandable keypoint vectors which are embedded through a learned linear transformation in the token space.
- Experiments show that our method can detect arbitrary intermediate keypoints of ski jumpers while maintaining the detection performance of the standard keypoints of ski jumpers. Furthermore, the method also works on another sports dataset with triple and long jump images and on the COCO [10] dataset.

2. Related Work

In many sports disciplines, computer vision is a beneficial technique to analyze athletes. Kulkarni et al. [8] use convolutional neural networks to estimate athletes' poses and classify table tennis stroke types. Woinoski et al. [18] detect and track swimmers during races to analyze strokes and detect breaths. Einfalt et al. [5] detect poses of swimmers and improve their estimated poses with using the swimming style as an input to the neural network and a pose refinement over time. Computer vision is also used in team sports, e.g., Bridgeman et al. [2] track athletes in soccer videos and create 3D poses of them and Wei et al.[17] estimate the location of the ball from monocular basketball video footage based on the players' trajectories. Furthermore, human pose and ski estimation is used for different ski disciplines. Wang et al. [15] estimate the poses of freestyle skiers and propose a pose correction and exemplar-based visual suggestions to the athletes. Human and ski pose estimation with robust estimation methods is also used by Ludwig et al. [11] in order to calculate the flight angles of ski jumpers during their flight phase.

In sports, 2D human pose estimation is very common technique among computer vision analysis applications. The approaches with the best scores on leaderboards of common benchmarks like COCO [10] or MPII Human Pose [1] are based on convolutional neural networks [7, 3]. A common backbone for recent human pose estimation approaches which is also used in [7] is the High Resolution Net (HRNet) [16]. It preserves a large resolution throughout the whole network and uses connections between different resolutions instead of an encoder decoder architecture like in [6, 12, 19]. Contrary to the fully convolutional approaches which are most common, TokenPose [9] is a Transformer [14] based approach for human pose estimation. It is usable without any convolutions, but it achieves the best and state-of-the-art results with a part of an HRNet as a feature extractor. The basic Transformer [14] architecture takes sequences of 1D tokens as an input. In order to deal with 2D images or feature maps, Vision Transformer [4] proposes to embed small image patches by a learned linear projection to 1D token vectors. This approach is used by TokenPose. Additionally, learnable keypoint tokens are appended to the image tokens and used as the Transformer input. The output of these keypoint tokens is then transformed through a MLP to heatmaps.

3. Method

Our model is based on TokenPose-Base [9], which is a combined convolutional and Transformer architecture. Basically, the proposed method and architecture are also applicable to all other TokenPose variants.

3.1. Additional Tokens for Intermediate Keypoints

In the standard TokenPose architectures, a token is learned for each defined keypoint. These tokens are appended to the image patches and fed jointly through the Transformer network. In the end, the outputs of the Transformer network that correspond to these tokens are converted to heatmaps with a MLP with shared weights across the keypoints. Hence, the naive approach to add more keypoints is to create a token for each added keypoint and train the network on the larger keypoint set. [9]

3.1.1 Learning Independent Keypoint Tokens

This method has a disadvantage. It always detects all intermediate keypoints at all times, even if only a subset of intermediate points is desired. Removing the unnecessary

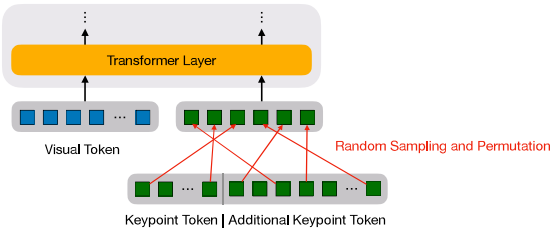

Figure 2. Scematic representation of keypoint sampling and permutation. For each additional keypoint, a token is added to the standard keypoint tokens. During training, a random subset of all keypoint tokens is selected and randomly permuted. Only the selected keypoints are appended to the visual tokens as an input to the Transformer network.

tokens from the Transformer input sequence in order to receive only the necessary detections fails, as shown by evaluations in Section 4. We suppose the reason is that the Transformer network correlates all embeddings of the input with each other, visual tokens as well as keypoint tokens. Therefore, the result for each keypoint is dependent on the intermediate representations of the feature maps corresponding to the keypoint tokens. If these tokens are not given in the input sequence, the Transformer misses necessary information to generate precise predictions. This dependence is a desired effect in TokenPose, as the intermediate results of neighboring keypoints help the model to detect occluded keypoints. This effect is called *contstraint cue* in TokenPose. [9]

We can alter the model so that it can cope with our scenario. In each training step, we randomly select a subset of the keypoints, whereby the number of selected keypoints is random as well, and permute them. As an input to the Transformer model, we use only the tokens that correspond to the selected subset of keypoints, the other tokens are not present in the input sequence. Figure 2 visualizes this technique. Consequently, the loss is calculated based on the sampled keypoints. Solely permuting the keypoint tokens is not sufficient, as Transformer networks are independent from the input sequence order to a certain extent. Permutation and random keypoint sampling is necessary that the Transformer learns to be quite independent from the present keypoint tokens, but evaluations show that there is still a slight performance drop if less keypoint tokens are used (see Section 4).

3.1.2 Analysis of Keypoint Tokens

Our goal is to detect arbitrary intermediate keypoints directly through the network. A transformation is necessary to create the specific keypoint tokens in order to detect the desired keypoints. A look at the inner product matrix of the learned keypoint tokens, which shows the similarity of the tokens, reveals a problem. In Figure 3, the inner product matrix of left and right ski with nine equally spaced intermediate keypoints is displayed. The matrix shows that the similarity of neighboring keypoint tokens is mostly high, but smaller than the similarity between the corresponding left and right keypoint token. Hence, it is not possible to design the tokens to detect arbitrary intermediate keypoints as there is an interference between tokens corresponding to left and right.

3.2. Intermediate Keypoints Encoded in Vectors

Therefore, we design another architecture, which is visualized in Figure 4. At first, features are extracted from the images with a HRNet [16] backbone. The feature maps are split into feature patches and embedded through a linear projection like proposed by [4]. 2D sine positional encoding is added to the resulting visual tokens. Instead of appending learnable keypoint tokens to the sequence of visual tokens, we use a method similar to the feature patch embedding. We encode the keypoints in keypoint vectors (details follow in Section 3.2.1) and use a linear projection to create embeddings of the keypoints. Hence, the model learns the transformation from keypoint vectors to keypoint tokens. Before appending the keypoint tokens to the input sequence, we randomly sample and permute the tokens, like described in Section 3.1.1. Between the Transformer layers, we always add the positional encoding to the visual tokens, but not to the keypoint tokens as we want them to be independent of

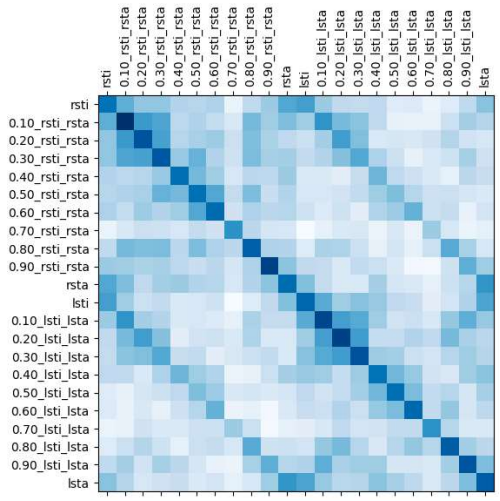

Figure 3. Inner product matrix for left and right ski with intermediate points. *rsti/lsti* stands for right/left ski tip, *rsta/lsta* stands for right/left ski tail. *p_sti_sta* means the keypoint is located on fraction p of the line between *sti* and *sta*. With increasing similarity, the color darkens, indicating a high similarity between corresponding left and right keypoints, which can be seen by the dark colored diagonal squares in the top right and bottom left part of the matrix.

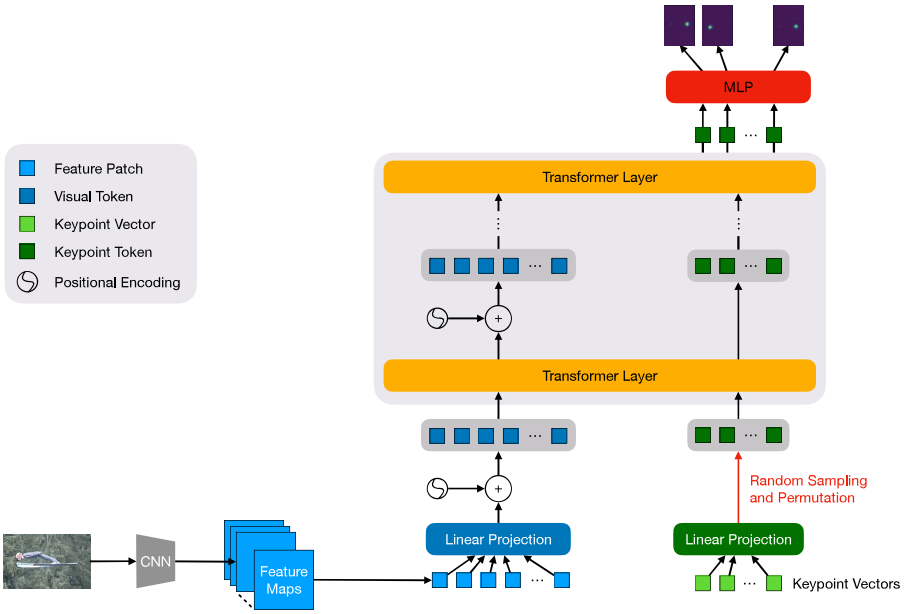

Figure 4. Model architecture with keypoint vectors. Image features are extracted with a convolutional neural network, split into feature patches and transformed to visual tokens using a learned linear projection. Keypoint vectors are treated similarly. They are transformed to keypoint tokens through a learned linear projection. Both linear projections are independent from each other. Positional encoding is added to the visual tokens, but not to the keypoint tokens. Keypoint tokens are randomly sampled and permuted before they are fed through the Transformer network. A MLP is used to transform the resulting keypoint tokens to heatmaps. [9]

the order. Like in TokenPose [9], we keep the outputs of the Transfomer corresponding to the keypoint tokens and use a MLP with shared weights to generate heatmaps for the desired keypoints.

3.2.1 Keypoint Vectors

The keypoint vectors are designed in a way that arbitrary keypoints on lines between the standard keypoints are representable. If a dataset has n annotated keypoints per person, a keypoint vector v for this dataset has length n, hence $v \in \mathbb{R}^n$. If we want to detect a keypoint of these annotated kepoints, e.g., keypoint i, then

$$v_k = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \quad k = 1, ..., n$$

Let the line between keypoints i, j be a body part for the annotated keypoints in the dataset, e.g. forearm, upper arm, thigh, lower leg, neck, etc. If a keypoint should be detected that lies on fraction p on the line between keypoints i and j, v is defined as

$$v_k = \begin{cases} 1-p, & k = i \\ p, & k = j \\ 0, & k \neq i \wedge k \neq j \end{cases} \quad k = 1, ..., n$$

If $p = 1$, the keypoint is located at the end on the line from i to j, hence the desired keypoint equals keypoint j. If $p = 0$, the keypoint definition is equal to keypoint i. For our training, we use all standard keypoints and randomly generate other arbitrary keypoints, sampled from all bodyparts of the dataset with p sampled uniformly from $[0, 1]$. An example for the ski jump dataset is visualized in Figure 5. With this method, it is also possible to generate keypoints that are located between intermediate keypoints. If we take the COCO dataset, for example, and want to generate keypoints on the spine, these keypoints are located on the line between keypoints i and j, whereby keypoint i is located in the middle of the two shoulder keypoints s_1, s_2 and keypoint j in the middle of the left and right hip keypoint h_1, h_2. Hence, if we want to detect a keypoint on fraction p of the line between i and j, our keypoint vector v consists of zeros apart from entries $v_{s_1} = v_{s_2} = (1-p) \cdot 0.5$ and $v_{h_1} = v_{h_2} = p \cdot 0.5$. So, we can design arbitrary intermediate keypoints that lie between annotated keypoints.

3.3. Exponential Moving Average

The validation score has a high fluctuation rate throughout the training, even after convergence. In order to reduce fluctuation, we keep an exponential moving average (EMA) of our model like it is used for mean teacher semi-supervised learning in [13]. Let α be the EMA rate, then all

head	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shoulder	0	1	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
elbow	0	0	1	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
hand	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
hip	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.8
knee	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0.2
ankle	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
right ski tip	0	0	0	0	0	0	0	1	0	0	0	0	0.5	0.2	0	0	0
right ski tail	0	0	0	0	0	0	0	0	1	0	0	0	0.5	0.8	0	0	0
left ski tip	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0.7	0.5	0
left ski tail	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.3	0.5	0

Figure 5. Keypoint vectors for the ski jump dataset. Keypoint vectors are displayed vertically. The first eleven vectors correspond to the standard keypoints. The last six keypoints are generated, whereby the number of generated keypoints is chosen randomly. The first generated keypoint lies on the upper arm, the second and third keypoint on the right ski, keypoint four and five on the left ski and the last keypoint on the thigh. The last keypoint, e.g., is located at 20% of the length on the line between hip and knee, hence it is closer to the hip.

parameters w^t_{EMA} of the EMA model at training iteration t are calculated as

$$w^t_{EMA} = \alpha \cdot w^{(t-1)}_{EMA} + (1 - \alpha) \cdot w^t,$$

whereby w^t are the parameters of the original model at training iteration t and all weights of the EMA model are initialized with the same weights as the original model. The EMA model can be seen as a temporal ensemble of the models from the last iterations, with more recent models having a larger impact after a warm-up phase.

4. Experiments

All experiments are based on the TokenPose-Base architecture [9]. At first, feature maps are extracted using the first three stages of a HRNet-w32 network [16]. The largest feature maps of the HRNet output, which have $\frac{1}{4}$ of the input resolution, are used as an input for the Vision Transformer network. All input images are resized to 256×192 and the resulting feature maps are split into patches of size 4×3. The patches are embedded to vectors of size 192 and we use 12 Transformer layers with 8 heads. We use 2D sine positional encoding, which is added to the visual token and corresponding intermediate representations after each Transformer layer. To obtain the keypoint coordinates from the heatmaps with resolution 64×48, we use the method presented in [20].

4.1. Ski Jump

Dataset. The ski jump training dataset contains 11,381 annotated images from 354 jump videos. The videos were recorded at different ski jumping hills, during multiple events and with different athletes, so their statures and dressings vary. The videos have different resolutions, most of them 720×576 pixels, but there are also a lot of HD and full HD videos included in the dataset. The footage covers a wide variety of weather and light conditions, e.g. snow, rain, fog, summer, winter, day and night. Only few images from every video are annotated, usually 2 - 4 frames per camera view, whereby each video consists of 4 - 8 camera views. Annotated keypoints are both ski tips and tails, head, shoulder, elbow, hand, hip, knee and ankle. The annotations of the joints are only available of one side of the body (the one facing the camera). The dataset contains images of the flight phase as well as images of the athlete during in-run, where the skis are not visible and not annotated. The best models are chosen according to validation on a seperate validation set with 200 images. The final scores are collected on the test set which contains 3,783 images from 121 videos.

Evaluation Metric. We use the Percentage of Correct Keypoints (PCK) for evaluation purposes. The PCK metric considers a keypoint as correct at a certain threshold t if the distance of the detected keypoint to the ground truth keypoint is less or equal than t times the torso size, which is the distance between shoulder and hip joint in this case. The recall at a certain PCK threshold tells the percentage of keypoints that is considered correct at that threshold. We use $t = 0.1$, which corresponds to approx. 5 cm in this dataset.

Results. For all ski jump experiments, we use pretrained weights from the COCO dataset. At first, we train a standard TokenPose-Base model. With an overall PCK of 80.7%, the Transformer model achieves a similar performance in comparison to a pure HRNet-w32 model, which scores 80.8% PCK. If we use the EMA model like described in Section 3.3, the total PCK rises to 81.9% and exceeds the HRNet score. This shows that the temporal ensemble included in the EMA model improves the model performance. In the next experiments, we will stick to the EMA model for our results. See Table 1, model A for further details.

In order to detect continuous keypoints on the body parts of the ski jumpers and the skis, we add 36 intermediate keypoints to the model. Three equally spaced intermediate points are added to the neck, upper arm, forearm, thigh, lower leg and torso, as well as nine equally spaced intermediate points to left and right ski. A training on these keypoints shrinks the detection score by 0.8%, as the model is now trained on a larger problem and can not focus on the standard keypoints. But regarding the performance on all keypoints, including the added keypoints, the performance rises by 4.1%. These scores are only achieved if we use all 47 learned keypoint tokens. Using only the keypoint tokens corresponding to the 11 standard keypoints instead of the 47 used during training, the score drops to only 14.4% PCK. It is remarkable that the performances vary between the keypoints. The elbow score is still high, while the score

Mod.	Input	K	Head	Shou.	Elb.	Hand	Hip	Knee	Ank.	rsti	rsta	lsti	lsta	Avg	Full
A'	std		98.3	91.3	73.2	65.1	85.0	80.1	83.7	73.9	81.9	67.4	81.2	80.7	
A	std		98.9	93.2	75.6	65.9	87.0	81.6	83.8	74.6	82.2	68.9	81.6	81.9	
B	all		98.7	93.5	74.5	64.8	86.6	81.3	81.8	74.7	80.8	68.3	80.0	81.1	86.0
B	std		0.9	52.0	72.5	0.5	11.1	0.0	1.1	0.2	0.0	0.3	0.0	14.4	
C	all		98.3	91.2	71.6	63.5	83.5	78.6	75.8	69.5	77.4	62.8	76.0	77.9	83.2
C	std		10.8	84.7	8.7	0.2	33.3	0.1	6.1	21.9	37.2	21.0	37.8	23.0	
D	all		98.9	92.9	74.1	62.9	85.8	80.2	81.1	73.3	79.8	64.9	78.4	80.1	85.3
D	std		98.8	92.8	73.4	62.6	85.6	79.7	81.0	72.7	79.0	65.1	77.6	79.7	
E	all	✓	98.2	91.1	72.4	65.0	83.7	79.8	78.6	72.4	80.8	65.6	79.3	79.4	82.1
E	std	✓	97.2	85.1	71.0	58.5	81.8	76.7	76.8	20.6	78.8	20.2	77.5	70.3	
F	all	✓	98.9	93.5	75.0	65.8	87.4	82.0	83.1	75.7	81.9	68.7	81.0	81.8	84.8
F	std	✓	98.9	93.4	74.8	65.6	87.3	82.2	83.2	75.2	82.0	68.9	81.1	81.8	
F	sgl	✓	98.8	93.2	75.0	65.6	87.3	82.2	82.9	68.7	81.5	63.4	80.8	80.9	

Table 1. Recall values in % at PCK threshold 0.1 for the ski jump datset of head, shoulder, elbow, hand, hip, knee, ankle, right ski tip, right ski tail, left ski tip, left ski tail and the average PCK over all 11 keypoints (second last column). If more than these 11 standard keypoints are used during training, the PCK score including the generated points is given in the last column. If the keypoint vector model is used, this is indicated in the third column. The qualifiers *std*, *all* and *sgl* in the input column refer only to the used inference protocol and not to the training procedure. *std* means that only the keypoint tokens/vectors that correspond to the standard joints are used as an input during inference, *all* means that the full input as during training is used (either 47 keypoints or keypoint vectors with generated keypoints) and *sgl* stands for single evaluation, meaning that a keypoint vector representing a single keypoint is passed to the model during inference and all keypoints are obtained separately. Model A is the pure TokenPose-Base implementation trained on the standard keypoints. Model A' is the non-EMA model, all other results are from the EMA models. Model B is the TokenPose implementation with 36 intermediate keypoints added. Model C is trained with token permutation, but not with token sampling. Model D is trained with token permutation and sampling. Model E uses the keypoint vector model introduced in this work, trained with permutation and sampling of the standard keypoints. Model F uses the keypoint vector model as well, trained with permutation and sampling of all keypoints.

for the knee and the ski tails drops to zero. Permuting the keypoint tokens results in a further performance drop for the evaluation with all tokens present. Using the standard keypoint tokens only, the detection score rises a little to 23.0% (see model C), but this is too low for a usable model. Including the random sampling and permutation technique changes this behavior. We randomly choose at minimum five keypoints and maximum all 47 keypoints. This method achieves a PCK of 80.1% evaluated on all keypoint tokens, which is little lower than without this method. But its performance drops only slightly if we evaluate with the standard keypoints only, it achieves a PCK of 79.9%.

For training the model that is based on keypoint vectors, we generate 1 - 30 additional keypoints on the neck, upper arm, forearm, thigh, lower leg, torso, left and right ski. The fraction p that defines the location on the bodypart is uniformly sampled between 0 and 1. Hence, we generate arbitrary keypoints during training. We permute and randomly sample at minimum five and at maximum all keypoints (standard and generated keypoints). However, randomly sampling only the standard keypoints results in a inferior score (see model E). This model achieves a PCK score of 81.8%, independent of the keypoint vectors in the input, which is a very similar performance as training solely on the standard keypoints, but it is capable of directly detecting arbitrary intermediate keypoints, which is proven by the higher full PCK score of 84.8%. If we evaluate the model only on generated keypoints (randomly between 1 - 30, randomly chosen arbitrary keypoints), we get a PCK score of even 86.3% (not in table). This proves that the model can really detect arbitrary points. The model achieves also good results if only a single keypoint vector is used as an input, the average PCK is 80.9% in this case. Hence, the dependence on the other keypoints is reduced to a minimum and the model is really flexible for the sole detection of the desired keypoints (see model F). Figure 6 shows some examples for intermediate detections. The model is lightweight so that these visualizations are executable in nearly realtime on a notebook CPU (see supplementary video).

4.2. Triple and Long Jump

Dataset. The triple and long jump dataset contains 4,522 labeled images from 186 video sequences, whereby 3,154 images from 122 videos are used for training, 200 images from 18 videos for validation and 1,062 images from 46 videos as the test set. The footage belongs to competition

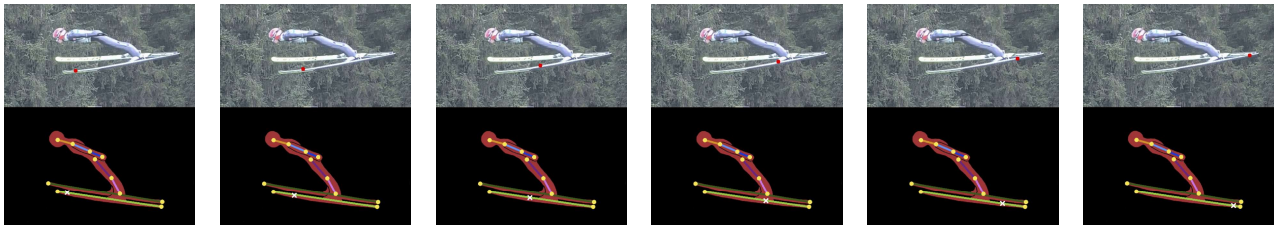

Figure 6. Examples for intermediate detections on the left ski. The white cross on the silhouette of the ski jumper in the lower image shows the selection of the intermediate keypoint. In the upper image, the corresponding keypoint is detected and displayed with a red circle.

Model	Avg PCK All Input	Full PCK All Input	Avg PCK Std Input	Avg PCK Sgl Input
A			91.3	
B	91.1	93.2	35.1	
C	91.1	93.2	91.1	
D	91.3	92.5	90.4	
E	91.7	92.9	91.7	91.5

Table 2. Recall values in % at PCK threshold 0.1 for the triple and long jump datset. The first column displays the average PCK of the standard keypoints with full input (all keypoint tokens/vectors) as used during training. The average PCK score including the generated points is given in the second column. The third column shows the average PCK of the standard keypoints with only the keypoint tokens/vectors of the standard keypoints present in the Transformer input and the last column the evaluation with keypoint vectors representing only single joints. Model A is TokenPose trained on the standard keypoins. Model B is TokenPose with 50 intermediate keypoints added. Model C is trained with token permutation and sampling. Model D is the keypoint vector model, trained with random samling of the standard keypoints and model E with random sampling of all keypoints.

and training scenarios and shows various sports sites and athletes. All videos are annotated with 20 keypoints (head, neck, r./l. shoulder, r./l. elbow, r./l.wrist, r./l. hip, r./l. knee, r./l. ankle, r./l. big toe, r./l. small toe, r./l. heel).

Evaluation Metric. Like in Section 4.1, we use the PCK metric relative to the torso size, which is defined as the distance between left shoulder and right hip in this case. Like before, we use $t = 0.1$, which corresponds to approx. 6 cm in this dataset.

Results. We depict the evaluation results in Table 2. The TokenPose model with standard keypoints achieves a PCK score of 91.3%. We add 50 points on the neck, upper arm, forearm, thigh, lower leg, shoulder axis, hip axis, spine and the lines between neck and left/right hip as well as left/right shoulder and neck. The performance on the standard joints is similar, but the performance with solely the standard input keypoint tokens drops to 35.1%. Uniformly sampling between five and all keypoints rises the detection score of the standard keypoints to 91.1% independent of the number of tokens in the input sequence. The keypoint vector model is capable of increasing the performance further, even surpassing the standard model's performance slightly with 91.7% PCK independent of the keypoint vector input. For that model, we generate 5 - 50 keypoints uniformly distributed between all bodyparts during training. If we use keypoint vectors that represent only a single joint and evaluate these results, the model still achieves an accuracy of 91.5% PCK, which is still better than the standard model.

4.3. COCO

Dataset. The COCO [10] dataset contains over 200,000 images with 250,000 labeled person instances. For training, we use the train2017 split consisting of 57,000 images, our results are reported on the val2017 split. We train every model for 1.2 million steps on a single GPU, therefore, the results are not identical to the results reported in [9]. This dataset has 17 annotated keypoints. Some keypoints might not be visible in the images, which is different from the other datasets. The annotated keypoints are nose, l./r. eye, l./r. ear, l./r. shoulder, l./r. elbow, l./r. wrist, l./r. hip, l./r. knee, l./r. ankle.

Evaluation Metric. We use the average precision (AP) based on Object Keypoint Similarity (OKS) as the primary metric for our evaluation, as this is the standard metric for the COCO dataset. OKS is calculated as

$$OKS = \frac{\sum_i exp(-d_i^2/2s^2k_i^2)\sigma(v_i > 0))}{\sum_i \sigma(v_i > 0)},$$

whereby d_i is the euclidean distance between corresponding ground truth and detected keypoint, v_i is the ground truth visibility flag, s is the object scale and k_i per-keypoint specific constants. Additionally, we use the PCK at threshold 0.1 as we can not measure the performance of arbitrary points with OKS.

Results. We display the results in Table 3. Our TokenPose training on COCO with the standard keypoints achieves an AP of 74.8% without the EMA and an AP of 75.4% with the EMA. We add 53 points on the head, neck, upper arm, forearm, thigh, lower leg, shoulder axis, hip axis, spine and the lines between neck and left/right hip as well as left /right shoulder and neck. The performance drop

Model	Input	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR	Avg PCK	Full PCK
TokenPose std joints non EMA	std	74.8	92.4	81.5	71.9	79.3	77.8	81.4	
TokenPose std joints	std	75.4	92.5	82.5	72.5	79.9	78.3	81.6	
TokenPose added joints	all	73.7	91.5	80.7	71.1	77.8	76.6	81.3	82.6
TokenPose added joints	std	56.5	87.4	65.0	54.6	59.8	60.0	68.1	
TokenPose added joints & sampling	all	69.4	89.5	77.5	66.9	73.6	72.8	78.7	80.3
TokenPose added joints & sampling	std	69.5	89.5	77.4	66.9	73.5	72.7	78.6	
Keypoint Vector	all	73.7	91.5	80.6	71.2	78.0	76.6	81.0	81.8
Keypoint Vector	std	73.6	91.5	80.6	71.2	77.9	76.6	81.0	
Keypoint Vector	sgl	73.5	91.5	80.6	70.9	77.8	76.5	80.9	

Table 3. OKS results and average recall values at PCK threshold 0.1 in % on the COCO dataset. If more than the standard keypoints are used, the PCK score including the generated points is given in the last column. The qualifiers *std*, *all* and *sgl* in the input column refer only to the used inference protocol and not to the training procedure. *std* means that only the keypoint tokens/vectors that correspond to the standard joints are used as an input during inference, *all* means that the full input as during training is used (either 70 keypoints or keypoint vectors with generated keypoints) and *sgl* stands for single evaluation, meaning that a keypoint vector representing a single keypoint is passed to the model during inference and all keypoints are obtained separately.

using only the standard keypoint input tokens is less on the COCO dataset, but still significant. Using all tokens, the model achieves an AP of 73.7%, with the standard tokens, the AP drops to 56.5%. With the keypoint vector model, the detection performance is nearly equal for full, standard and single input. As the full PCK is higher than the standard PCK, this proves that the model is capable of precisely detecting arbitrary generated keypoints. The PCK in general is lower than the PCK of the TokenPose model trained on the standard keypoints, but this is caused by the training duration. We stop after 1.2 million steps, where the keypoint vector model did not fully converge as it learns slower due to the significantly lower propagated signal caused by the random sampling.

5. Conclusion

At first, this paper proposes a training routine that makes it possible to train a TokenPose [9] model with additional kepoints, but independent of these additional keypoints during inference. In the standard TokenPose model, all keypoint tokens need to be present in the Transformer input, as the model learns the location of the keypoints not only from the image but also in dependence of the detections of the other keypoints. Random sampling and permuting the keypoint tokens in the input forces the model to learn the keypoint locations independent of the other keypoints.

This model has the disadvantage that only the intermediate keypoints that are trained can be detected afterwards. Designing the tokens such that the model finds arbitrary keypoints is not possible as the coherence of left and right tokens is larger than the coherence of neighboring tokens. Hence, this paper proposes a novel architecture that uses keypoint vectors instead of keypoint tokens as an input. Keypoint vectors have the same length as the number of annotated keypoints in the dataset. They sum up to 1 and represent an arbitrary intermediate keypoint as a mixture of the standard keypoints. Keypoint vectors are embedded like the visual patches with a linear transformation in the embedding space. Instead of learning fixed keypoint tokens, our model learns this linear embedding.

Evaluations show that our keypoint vector model in combination with the random sampling strategy works as desired. The PCK on the standard joints is similiar or even slightly better for all three experiments with different datasets. The PCK including arbitrary intermediate keypoints is even higher, hence, the model can really detect any desired intermediate keypoint. The detection of single keypoints is also nearly completely independent from other keypoints, which is proven by an evaluation with only single keypoint vectors as an input to the Transformer.

In the future, we would like to extend our model to detect not only arbitrary intermediate keypoints, but arbitrary keypoints in general. In order to achieve that, segmentation masks have to be leveraged and included in the model to define the desired keypoints. With segmentation masks, it is possible to know how far intermediate keypoints can be located away from the direct line between two keypoints.

6. Acknowledgements

This work was funded by the Federal Institute for Sports Science (BISp) based on a resolution of the German Bundestag. We would like to express our gratitude to the Institute for Applied Training Science (IAT) Leipzig for the collection and provision of the ski jump data. Furthermore, we would like to thank the Olympic Training Center Hessen for collecting and providing the triple and long jump data.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] Lewis Bridgeman, Marco Volino, Jean-Yves Guillemaut, and Adrian Hilton. Multi-person 3d pose estimation and tracking in sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[3] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Toward fast and accurate human pose estimation via soft-gated skip connections. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 8–15. IEEE, 2020.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Moritz Einfalt, Dan Zecha, and Rainer Lienhart. Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 446–455. IEEE, 2018.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[7] Junjie Huang, Zengguang Shan, Yuanhao Cai, Feng Guo, Yun Ye, Xinze Chen, Zheng Zhu, Guan Huang, Jiwen Lu, and Dalong Du. Joint coco and lvis workshop at eccv 2020: Coco keypoint challenge track technical report: Udp++. 2020.

[8] Kaustubh Milind Kulkarni and Sucheth Shenoy. Table tennis stroke recognition using two-dimensional human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4576–4584, 2021.

[9] Yanjie Li, Shoukui Zhang, Zhicheng Wang, Sen Yang, Wankou Yang, Shu-Tao Xia, and Erjin Zhou. Tokenpose: Learning keypoint tokens for human pose estimation. *arXiv preprint arXiv:2104.03516*, 2021.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[11] Katja Ludwig, Moritz Einfalt, and Rainer Lienhart. Robust estimation of flight parameters for ski jumpers. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2020.

[12] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[13] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[15] Jianbo Wang, Kai Qiu, Houwen Peng, Jianlong Fu, and Jianke Zhu. Ai coach: Deep human pose estimation and analysis for personalized athletic training assistance. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 374–382, 2019.

[16] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[17] Xinyu Wei, Long Sha, Patrick Lucey, Peter Carr, Sridha Sridharan, and Iain Matthews. Predicting ball ownership in basketball from a monocular view using only player trajectories. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 63–70, 2015.

[18] Timothy Woinoski and Ivan V Bajić. Swimmer stroke rate estimation from overhead race video. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2021.

[19] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.

[20] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7093–7102, 2020.