Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2022

Bc. Jakub Straka

University of West Bohemia

Faculty of Applied Sciences

Department of Cybernetics

MASTER'S THESIS

Automatic Human Pose Estimation using Neural Network

PILSEN, 2022 **Bc. Jakub Straka**

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí

V Plzni dne 23.5.2022 .....

Poděkování

Děkuji vedoucímu diplomové práce Ing. Ivan Gruber, Ph.D. et Ph.D. za odborné vedení, konzultace a cenné rady při vypracování diplomové práce.

Acknowledgement

Thank you to my supervisor, Ing. Ivan Gruber, Ph.D. et Ph.D., for providing professional guidance, consultations and valuable advice while working on this thesis.

Abstrakt

Tato diplomová práce se zabývá odhadem pózy člověka pomocí neuronových sítí. V první části jsou popsány metody založené na konvolučních neuronových sítích a metody založené na architektuře transformeru, které byli úspěšně použity k řešení tohoto problému.

V druhé části je zvolen model, který byl původně navžen pro řešení úlohy odhadu pózy ruky. Dále jsou popsány úpravy tohoto modelu, tak aby byl vhodný pro úlohu odhadu pózy člověka. Výsledkem úprav jsou dvě varianty modelu, které jsou následně trénovány na datasetu COCO. Nakonec jsou výsledky natrénovaných modelů porovnány s některými z modelů představených v první části.

Klíčová slova

počítačové vidění, umělá inteligence, odhad pózy člověka, konvoluční neuronové sítě, transformer architektura

Abstract

This thesis addresses human pose estimation using neural networks. The first part describes methods based on convolutional neural networks and methods based on transformer architecture, which have been successfully used to solve this problem.

In the second part, a model is chosen that was originally designed to solve the problem of hand pose estimation. Next are described modifications of this model so that it is suitable for the task of human pose estimation. The result of the modifications are two variants of the model, which are then trained on the COCO dataset. Finally, the results of the trained models are compared with some of the models presented in the first part.

Keywords

computer vision, artificial intelligence, human pose estimation, convolutional neural networks, transformer architecture

Contents

List of Figures

List of Tables

1 Introduction

The goal of this work is to learn about the human pose estimation (HPE) task and train a neural network that solves this task. In the first part, there will be described methods based on neural networks that are used to solve the HPE problem. The second part will describe the modifications of the Deformable Pose Transformer (DePOTR) [19] model needed for detection on the COCO [16] dataset and the experiments performed during the training of the model. Finally, the results will be compared with selected state-of-the-art models presented in the first part.

Human pose estimation is a computer vision task that aims to estimate significant keypoints on the human body from image or video. Keypoints are often body joints and significant features in the face. Like other computer vision tasks, such as classification, object detection, semantic segmentation, or instance segmentation, HPE methods have improved significantly with the development of deep learning.

With neural networks, large datasets for HPE began to appear. The complexity of the scenes in the individual datasets gradually increased. Earlier datasets contained only limited variation in poses and environments. The pose detection model should be robust and should be able to detect multiple people with different body configurations in the image. HPE is a challenging task mainly because of the flexibility of the human body and the large variability of the environment. Some common challenges in HPE are shown in Figure 1.

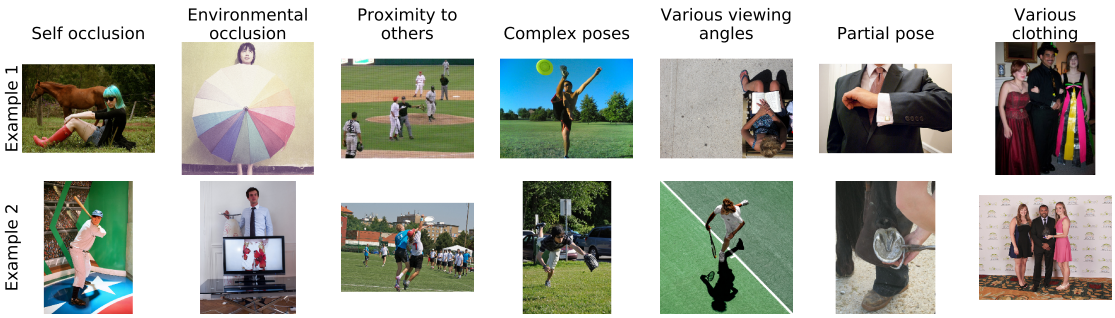


Figure 1: Illustration of common challenges in human pose estimation.

In an ideal situation, the whole pose is visible and no keypoints are occluded or out of the image. However, this case is very rare. Because of interaction with the environment

or with other people, occlusions may occur. If the environment or another part of the body occludes the keypoint, a model must estimate the position based on the body configuration and the position of the other keypoints. A similar problem occurs when only part of the body is in the image. The model must correctly determine which part of the body is in the image and, from this limited information, predict correct positions. In these cases, there may be confusion between the right and left sides as most of the keypoints are in pairs. But even in cases where most of the pose is visible, clothing can cause uncertainty in the position of the joints. Another big challenge is the proximity of other people and crowded areas. In these cases, keypoints may be confused between individuals. Overall, human pose estimation is a challenging task, as illustrated by the above-mentioned situations.

Human pose estimation can be applied in various fields. The large potential for HPE is in sports. An essential component of sport is movement, HPE can help with movement analysis and quantification of movement. In cases where precise movement is needed, such as dance or martial arts, HPE can help with training and increase accuracy and efficiency. Similarly, HPE can be used in healthcare for the diagnosis and quantification of diseases affecting the musculoskeletal system.

Another use is action recognition and action prediction. It is possible from the sequence of movements to estimate what activity is happening and, in some cases, what movements will follow. This can be useful for monitoring people with an increased risk of a fall, such as the elderly, or people with Parkinson's disease. The system could monitor for abnormalities in motion or detect a fall.

There is potential use in the field of motion capture. Motion capture is often used in the entertainment industry to animate characters in movies or games. The motion capture systems used today are accurate but at the same time very expensive and difficult to set up. In some cases, HPE can effectively replace these systems, but improvements in accuracy are still needed. Other cases of use in entertainment and education are virtual reality and augmented reality. By estimating the pose, it is possible to achieve better interaction with virtual objects.

2 Datasets and metrics

2.1 Datasets

HPE consists of a variety of tasks. To train a robust machine learning model, sufficient number of data is necessary. HPE has been studied for a long time and different datasets addressing different variations of this task exist. The first division can be the dimension of data. Datasets with 3D annotations are less common, the majority of datasets provide only 2D data.

The 3D datasets are hard to obtain. Most 3D datasets use marker-based motion capture systems. Therefore, data are captured indoors in a controlled environment. The most used indoor 3D dataset for HPE is Human3.6M [11] introduced in 2014. This dataset contains videos of 11 actors performing 17 different activities. In total dataset include 3.6 million annotated frames. Dataset is captured indoors by a motion capture system. The more recent dataset 3DPW [28] uses Inertial Measurement Units. This system does not need markers and cameras to capture motion and is more suitable to use outdoor. The dataset contains videos of 7 different people performing common activities. Dataset has around 51000 frames and was introduced in 2018.

2D datasets are much easier to obtain as it is possible to annotate images directly without other data. The 2D dataset can differ based on the number of people in a single image. Early datasets contained only single-person images. More recent datasets are focused on the more challenging task of multi-person detection. The domain in which they are taken can also differ. LSP [12] (Leeds Sports Pose) dataset from 2010 captures only 8 different sports. Therefore provides a limited range of poses and limited variation of the background. Dataset has around 2000 images. Today most-used datasets are the MPII [1] dataset and MSCOCO [16]. Both datasets are larger and contain activities from a variety of domains.

Max Planck Institute for Informatics (MPII) dataset [1] was presented in 2014 and aimed to represent common and rare humane poses from everyday activities. Images in the dataset cover 20 categories divided into around 400 activities. The dataset includes over 40 000 annotated human poses in about 25 000 images. Images contain from 1 to 16 people. However, most pictures contain one person. For each person are annotated 3

joints on each leg, 3 joints on each arm, pelvis, thorax, upper neck, and head, a total of 16 keypoints for each person. Annotations can only be partial if the body part is outside the image. The lower part of the limb is most often missing in the images.

Microsoft Common Objects in Context (MSCOCO 2017) dataset [16] is the most used dataset in computer vision as it provides annotations for different tasks as object detection, segmentation and keypoint detection. Dataset is split into training-set with around 120 000 images, validation-set with 5000 images, and test-set with around 15 000 images, together almost 150 000 images. Approximately only half of the images contain at least one person. Some images contain up to 12 annotated poses. Each person has annotated 12 limb joints and 5 keypoints in the face, a total of 17 keypoints. Keypoints are provided in format: $[x_0, y_0, v_0 ... x_k, y_k, v_k]$ where x and y are keypoint location and v is a flag that signalize if keypoint in the image is occluded or visible.

2.2 Metrics

As the dataset developed, the metrics used to evaluate the results also developed. Several metrics have been used over the years. Some metrics depend on the specific annotations of a given dataset, some differ because of different requirements of the task (single/multiple pose estimation). The most common requirements that metrics must include are the position of the predicted keypoints and the size of the human relative to the size of the image.

2.2.1 MPII evaluation

Earlier datasets used the metric Percentage of Correct Parts (PCP). Which evaluates the correct prediction of limbs. If the distance between the ground-truth position and the predicted position is lower than the fraction of the length of the limb, the prediction is considered correct. Threshold 50% of the limb length is often used. The disadvantage is the higher penalization of short limbs. A similar metric is the Percentage of Correct Keypoints (PCK). PCK is defined for keypoint evaluation. A keypoint is considered correctly predicted if the predicted keypoint lies in the $\alpha \cdot \max(w, h)$ distance from the ground-truth keypoint, where w and h are the width and height of the person bounding box. α is often chosen 0.5. As the main metric, in the MPII dataset is used PCKh@0.5

variation of PCK metric. This metric uses the length of the head segment instead of the person bounding box. @0.5 refers to the value of α. The advantage is the independence of the pose configuration.

2.2.2 COCO evaluation

COCO dataset uses for the evaluation average precision (AP) and average recall (AR). This metric was not previously used in HPE, but a similar metric is used in object detection. Precision and recall are defined as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP (true positive) - The distance between the detected keypoint and ground-truth position is less than the selected threshold.

FP (false positive) - The distance between the detected keypoint and ground-truth position is higher than the selected threshold.

FN (false negative) - Annotated keypoint was not predicted.

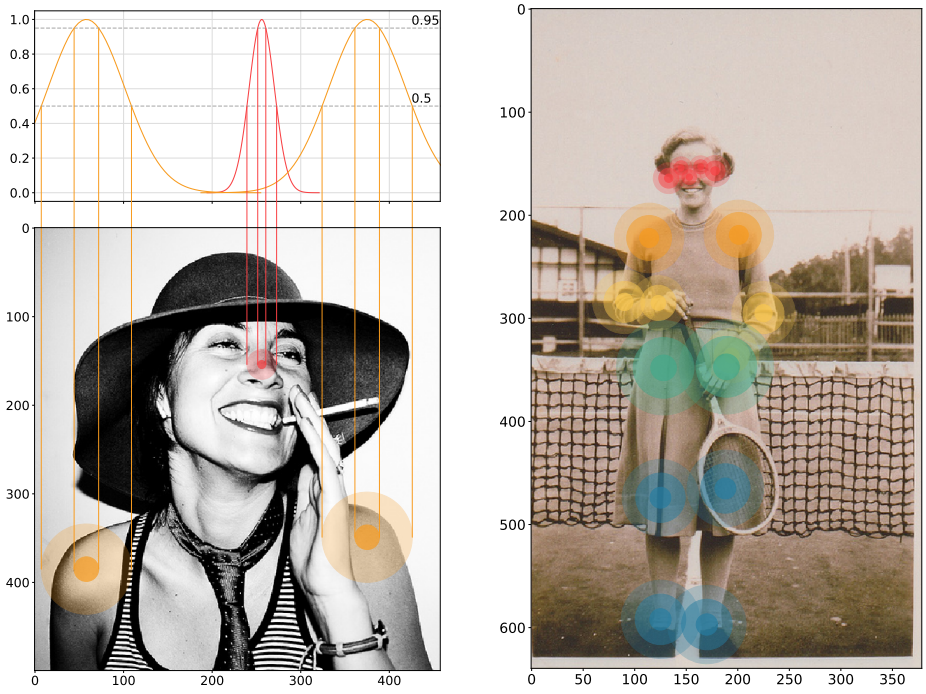
Precision measures what percentage of detections is correct, and recall measures what percentage of ground truth detections were detected. To determine, if the keypoint is correctly detected (true positive or false positive), object keypoint similarity (OKS) is used. OKS serves a similar role as IoU in object detection and is defined as:

$$\text{OKS} = \frac{\sum_i e^{-\frac{d_i^2}{2 \cdot s^2 \cdot \kappa_i^2}} \cdot \delta\left(v_i > 0\right)}{\sum_i \delta\left(v_i > 0\right)} \tag{1}$$

Where d_i is a Euclidean distance between ground truth and detected position, s is scale of object (square root of object area), κ_i is per-keypoint constant and v_i is visibility flag of keypoint. OKS is calculated as the average of all visible keypoints. Visibility v_i is determined by ground truth labels. Values for κ_i are determined from annotated keypoints standard deviation $\kappa_i = 2\sigma_i$. Values for σ_i are: 0.026, 0.025, 0.025, 0.035, 0.035, 0.079, 0.079, 0.072, 0.072, 0.062,0.062, 0.107, 0.107, 0.087, 0.087, 0.089, 0.089 for

the nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle, respectively.

OKS for a single keypoint is defined as unnormalized Gaussian, illustrated in Figure 2a. Figure 2b shows the areas corresponding to three different OKS for all keypoint as concentric circles with a center in the keypoint ground truth position. Circles correspond to OKS = {0.5, 0.75, 0.95}. As the OKS value increases, the radius of the circles decreases. For a predicted keypoint to be recognized as correctly predicted, it must be in the circle given by the selected OKS.



(a) (b)

Figure 2: (a) Illustration of Gaussians corresponding to different keypoints. (b) Illustration of areas corresponding to OKS = {0.5, 0.75, 0.95}. The center of the circles corresponds to the ground truth position of the keypoint.

Because results may differ in accuracy based on object size and level of precision, COCO evaluates [17] 10 different metrics:

Average Precision (AP):	
AP	AP at OKS=.50:.05:.95 (primary challenge metric)
$AP^{OKS=.50}$	AP at OKS=.50 (loose metric)
$AP^{OKS=.75}$	AP at OKS=.75 (strict metric)
AP Across Scales:	
AP^{medium}	AP for medium objects: $32^2 <$ area $< 96^2$
AP^{large}	AP for large objects: area $> 96^2$
Average Recall (AR):	
AR	AR at OKS=.50:.05:.95
$AR^{OKS=.50}$	AR at OKS=.50
$AR^{OKS=.75}$	AR at OKS=.75
AR Across Scales:	
AR^{medium}	AR for medium objects: $32^2 <$ area $< 96^2$
AR^{large}	AR for large objects: area $> 96^2$

Primary challenge matric AP is average over 10 different OKS thresholds starting from OKS=0.5 and ending with OKS=0.95 with step 0.05. Other metrics are only used for the characterization of model performance.

3 Transformers

Transformer architecture proposed in [27] was a novel approach in NLP for sequence transformation. Before transformers, recurrent neural networks (RNN) achieved the best results in tasks like machine translation. The disadvantage of RNNs is their hidden state, which depends on previous inputs. Therefore, parallelization is impossible. Transformer architecture is based on an encoder-decoder structure and does not rely on a hidden state and can process the whole sequence at the same time. Instead of a hidden state, the transformer employs mechanics called self-attention. Model proposed in [27] is in Figure 3.

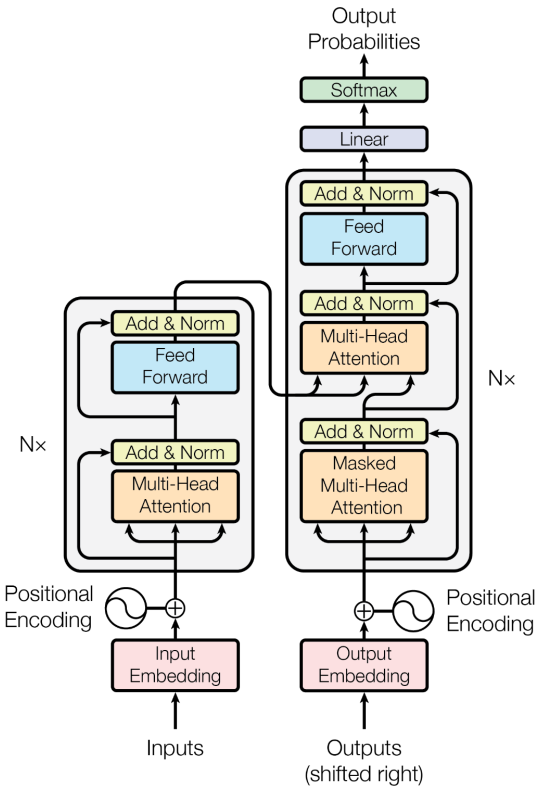


Figure 3: Transformer model [27].

The model is divided into a sequence of N encoder layers and a sequence of N decoder layers. The objective of the encoder is to map input sequence X into sequence Z, then the decoder generates an output sequence Y from the sequence Z. Elements of Y are generated one at a time and the already generated elements are provided as decoder input together with the sequence Z.

The input to the encoder is a sentence. Before passing the sentence into the first layer, each word in the sentence is transformed into a vector representation x with dimension d_m. Because the model is not sequential and all words are processed parallel, the model does not have information about the position of individual words in the sentence. This issue is solved by positional encoding. A vector of the same dimension representing the position in the sentence is added to each vector x. Sine and cosine functions with different frequencies or learnable positional encoding are used for this purpose.

Each encoder layer has a multi-head self-attention mechanism and a fully connected feed-forward network. All input embedding vectors are stacked in one matrix $X \in \mathbb{R}^{n \times d_m}$. Then X is linearly projected into three matrices Q (queries), K (keys), and V (values). The projection is performed by multiplying X and the weight matrices $W^Q, W^K, W^V \in \mathbb{R}^{d_m \times d_k}$ where d_k is a dimension of latent vector representation. Self-attention is then defined as:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

The goal of self-attention is to capture how the individual words in a sentence interact and the relationships between the words. The block is called multi-head because the calculation of self-attention is done in parallel h times. Each attention head has its own weights W^Q, W^K, W^V. The multi-head mechanism allows the model to learn different text properties. Outputs of all attention heads are then concatenated and projected back to latent representation dimension by weights $W^O \in \mathbb{R}^{h \cdot d_k \times d_m}$. The second part of a layer is a small feed-forward network with two layers. Additionally, around both parts of an encoder layer are residual connections followed by layer normalization.

The decoder layer follows a similar structure but with an additional multi-head self-attention block. The first attention block performs attention on the previous outputs of the decoder stack. During training, positions that have not yet been predicted are masked so that the model can not attend to them. In the second attention block, queries are computed from the output of the first attention block and keys and values are computed from the output of the encoder stack. The second block is followed by the feed-forward network. Each of the blocks also has residual connections around them followed by layer normalization. The decoder output is then projected by the linear layer into a vector of a size corresponding to the number of tokens in the model dictionary. The softmax function is then used to obtain the probabilities of individual words.

3.1 Transformers in computer vision

3.1.1 Vision Transformer

After the success of [27] in NLP transformers slowly started to appear in computer vision in combination with CNNs. In 2020 was presented Vision Transformer (ViT) [9] for image classification. The model was based exclusively on a transformer without any convolutional layers. An overview of the model is in Figure 4. The computational complexity of the self-attention layer is quadratic and increases with the size of the image. For the whole image, it would be necessary to calculate the attention between all the pixels in the image. In ViT is this problem solved by dividing an image into patches. Each patch serves the same role as a single token in the NLP task. All patches are flattened and projected into a vector of the fixed size. Also, 1D positional encoding is added to each patch. All vector representations of patches are inserted into the encoder stack. Predictions are made by the MLP head with one hidden layer attached to the encoder stack.

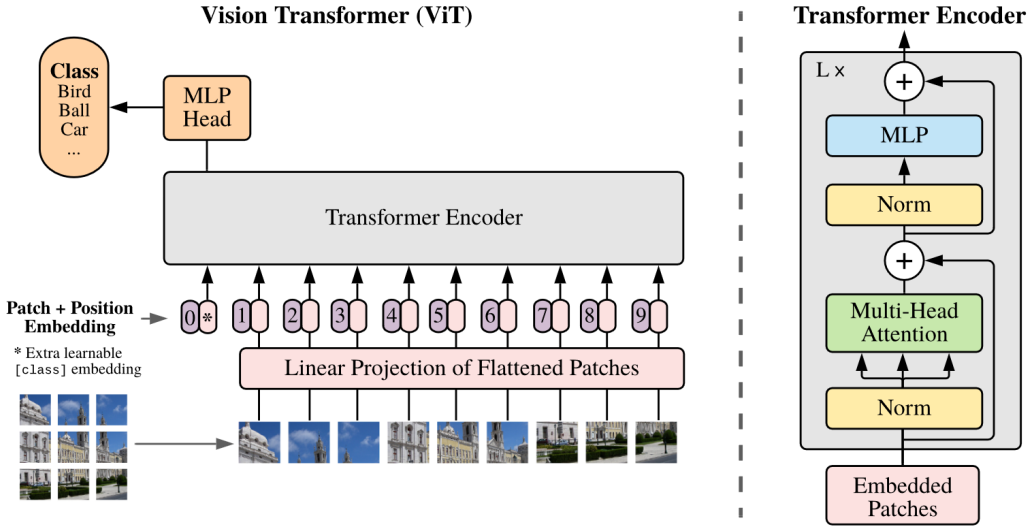


Figure 4: Vision Transformer [9].

The model performed well when pre-trained on a large dataset and then fine-tuned on specific smaller datasets. Before fine-tuning MLP head is replaced by a linear layer initialized with zeros and size corresponding to a number of classes in a smaller dataset.

3.1.2 Detection Transoformer and Deformable Detection Transformer

In 2020, Detection Transformer (DETR) [4] and Deformable DETR [32] were also published. Both models are designed for object detection. Deformable DETR is an extension of DETR.

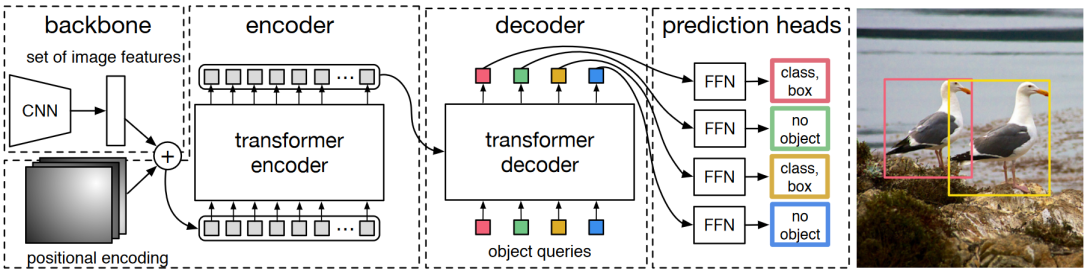


Figure 5: Detection Transformer [4].

DETR follows standard transformer architecture and exploits both encoder and decoder modules. DETR solves the problem of the high complexity of self-attention by adding CNN before the transformer. The output of CNN is a feature map with a significantly smaller size than the input image but with a larger number of channels. Before inserting into the transformer, the feature map is reshaped to a 1D sequence and the number of channels is reduced by 1×1 convolution to a dimension suitable for the transformer. Both encoder and decoder are the same as in a standard transformer. The only difference is in the use of the decoder. In a standard transformer, outputs are generated sequentially. Outputs from the model serve as input for the decoder for the next token. The decoder in DETR processes the whole sequence in parallel at once. As input for decoder serves N learned positional encodings (object queries). The number of object queries determines the number of objects that the model can detect. Final predictions are then made by a small feed-forward network attached to the decoder. The full overview of the model is in Figure 5.

DETR achieved comparable results with Faster-RCNN. However, the model suffers from slow convergence during training and lower accuracy in detecting small objects.

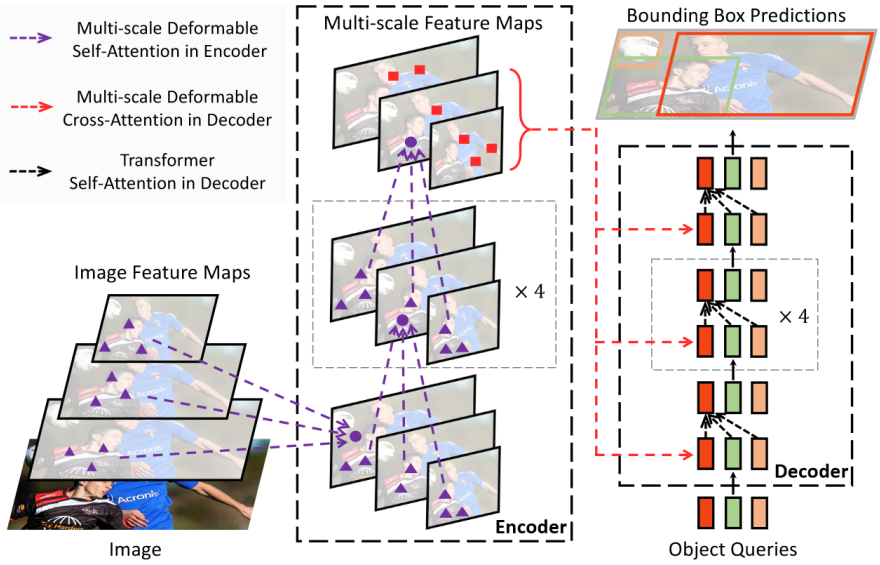


Figure 6: Deformable DETR [32].

To address these issues Deformable DETR was proposed. The low sensitivity of small objects is caused by the insufficient resolution of the final feature map. This problem can be solved by using feature maps from multiple scales. Small objects are detected on high-resolution feature maps and large objects on low-resolution feature maps. Unfortunately, this change increases the sequence size for the transformer and causes an excessive increase in computational and memory complexities. For this reason, the self-attention module in DETR was modified. Inspired by the deformable convolution [7], in the new deformable self-attention module each position in the sequence attends only to a limited number of other positions. Multi-head attention is calculated by:

$$\text{MultiHeadAttn}\left(z_q, x\right) = \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_k} A_{mqk} \cdot W'_m x_k \right] \tag{3}$$

Where x is a feature map, k is an index of key vector, q is an index of query represented by feature vector z_q. M is a number of attention heads, W'_m and W_m are learnable weights, and A_{mqk} are attention weights that are normalized so that their sum is equal to one.

Deformable attention is extension of classical attention mechanism and adds 2D reference points p_q and learnable offsets Δp_{mqk}. For each query is chosen only a small number of

keys K, sampled keys are indexed by k. The reference point determines the vector in the feature map and the offset determines the K positions around the reference point for which the attention will be calculated. Deformable attention is calculated by:

$$\text{DeformAttn}\left(z_q, p_q, x\right) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} \cdot W'_m x\left(p_q + \Delta p_{mqk}\right)\right] \tag{4}$$

The last variation of the attention layer is the extension of the deformable attention by multi-scale feature maps. Multi-scale deformable attention is defined as:

$$\text{MSDeAttn}(z_q, \hat{p}_q, \{x^l\}_{l=1}^{L}) = \sum_{m=1}^{M} W_m \left[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlqk} \cdot W'_m x^l\left(\phi_l\left(\hat{p}_q\right) + \Delta p_{mlqk}\right)\right] \tag{5}$$

As input is used L feature maps on different scale levels $\{x^l\}_{l=1}^{L}$. An index l is added to the offsets of the reference points and attention weights, which denotes the feature map level. Positions of reference points are normalized $\hat{p}_q \in [0,1]^2$. Due to the different sizes of the feature maps, the relative positions of the reference points are transformed by the function ϕ_l into the absolute coordinates of the respective scale level l. In addition to position embeddings, scale-level embeddings e_l are added to feature representation.

Deformable attention proved to be beneficial and achieved better results than DETR with a much smaller number of training epochs.

3.1.3 Deformable Pose Transformer

Deformable Pose Transformer (DePOTR) [19] is a model for 3D hand pose estimation. The model is based on Deformable DETR. The difference compared to DETR is how the query vectors are used. When detecting objects, it is not known in advance how many objects will be in the image. The model must take this fact into account and must be able to predict a variable number of objects. Deformable DETR uses 300 query vectors. The result for each query vector is the predicted bounding box and object class. Bounding boxes that do not contain any object are predicted with a "no object" class. In contrast to object detection, in pose estimation number of keypoints is given

in advance. DePOTR uses a fixed number of query vectors given by the number of keypoints. The resulting predictions indicate only the position of the keypoint and the type of the keypoint (class) is determined by the query vector index. Also, the loss function is different. Because there is no need to match predictions and ground-truth bounding boxes as in Deformable DETR, DePOTR uses a smooth L1 loss to evaluate the results.

4 Human Pose Estimation

Before using deep learning methods, latent tree models [6] were often used. The goal was to create a tree structure that will represent human pose. Histogram of Oriented Gradient [8] (HOG) were frequently used as features. The disadvantage was that the individual keypoints were modeled locally, and each keypoint was modeled separately or in combination with only one neighboring keypoint. This led to low accuracy and models could not generalize enough. With the development of neural networks, extraction of features using convolutional networks has proved to be more efficient than classical methods. In 2012 AlexNet [14] for image classification was proposed, achieving state-of-the-art results on ImageNet ILSVRC-2010 [23]. In 2014 DeepPose [26] model for HPE was presented. The architecture of this model was based on AlexNet and pose estimation was formulated as keypoint regression. The model achieved state-of-the-art performance on four relevant benchmarks at the time. After the DeepPose several models based on convolutional networks were proposed. In recent years, methods based on transformer architecture have begun to emerge.

4.1 Basic division of methods

One of the possible divisions is the number of individuals the HPE model can estimate in the image. Models can be divided into two categories, models for single-person estimation and models for multi-person estimation. The single-person estimation assumes only a single person in the image and predicts only one set of coordinates for body keypoint, on the other hand in the multi-person estimation task, number of people in the image is unknown and the model has to predict keypoints for every individual. Overlapping of body parts of several people can occur. Therefore single-person estimation is significantly easier than multi-person estimation. There are essentially two basic approaches how to solve multi-person estimation. The general procedure for these approaches is shown in Figure 7 and 8.

Top-down: This approach consists of two steps. First, a human detector is used to detect all humans in an image. Then the parts of the image containing humans are cropped. In the second step, all cropped images containing only a single human are processed by a single-pose estimator to predict keypoints. Predicted coordinates are then projected back to the original image. Models adopting this approach often omit the first step and focus only on keypoint estimation. For human detection are used other frameworks such as Faster-RCNN [21]. The disadvantage is the dependence on detections from the first step and slow prediction, especially with a high number of people in the image as every pose is processed separately. Despite these disadvantages, these models are fairly accurate. This approach is used in Simple Baselines [30] and TransPose [31] model which will be discussed later.

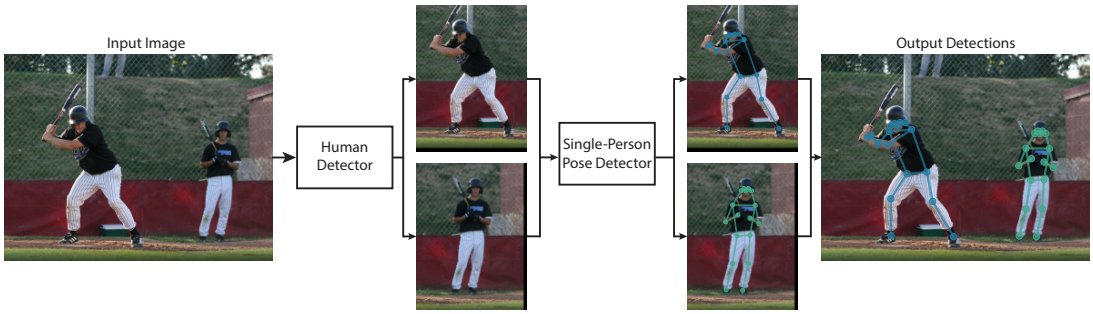


Figure 7: Illustration of top-down approach.

Bottom-up: This approach aims to predict all poses in one pass of the image. The process is divided into two steps. In the first step, all potential keypoints are detected. In the second step, keypoints are grouped into poses and associated with individual people. Grouping keypoints can be computationally difficult and these methods are highly dependent on the accuracy of this step. This approach is used in OpenPose [3]

Figure 8: Illustration of bottom-up approach.

HPE aims to estimate the position of the keypoint from the input image. Methods for single pose estimation performing this task can be split into two groups based on the formulation of the task. The first is direct regression. The model directly maps the input image to the position of keypoints. The advantage of this approach is the potential simplicity and speed of the model. Unfortunately, mapping can be hard due to the high non-linearity of the task. The second group is based on heatmaps. For each keypoint is generated heatmap, pixel values indicate the probability that a keypoint lies at a given position. Heatmaps are generated by 2D Gaussian centered on the position of keypoint. The position of keypoints is obtained from predicted heatmaps as the position of the pixel with maximal value.

4.2 Models

Extensive research was made in HPE in recent years and a large number of models were proposed. These models vary in complexity and approach. It is not possible to say with certainty whether direct keypoint regression or heatmap detection is better, however, heatmap detection was preferred in many works. Since AlexNet [23], convolutional neural networks have been used exclusively for HPE, today the trend is shifting to the transformer architecture first presented for natural language processing (NLP) [27]. In this section few models will be discussed, each of which adopts a different way of solving the problem of HPE.

4.2.1 Simple Baselines for Human Pose Estimation and Tracking

Model [30] was proposed in 2018 in reaction on increasing complexity of HPE models. They argued that HPE models are complex and significantly differ in architecture, but differences in accuracy are small. The main goal was to design a simple baseline models for pose estimation and tracking. At the time the model was published, Hourglass [20] was the leading model on MPII benchmark dataset. The Hourglass model consists of multiple stacked hourglass networks, each network is composed of pooling steps followed by upsampling and skip connections. The idea behind downscaling an image to low resolution and subsequently upscaling it back is to capture information at every scale. Cascade Pyramid Network (CPN) [5] was leading model on COCO dataset. CPN model is based on ResNet [10] backbone. The model is divided into two modules GlobalNet and RefineNet. GlobalNet generates heatmaps from different scales of feature maps by applying convolutional filters. Heatmaps from GlobalNet are upsampled in RefineNet so that all scale levels are the same size. In the end, heatmaps on all scales are concatenated.

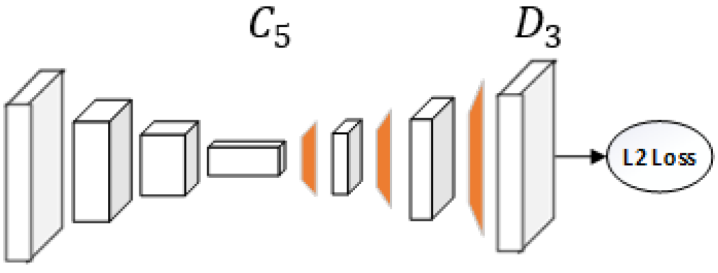


Figure 9: Simple baseline model architecture [30].

The proposed Simple baseline model follows a similar idea of downscaling and upscaling of the input image. The model consists of ResNet [10] backbone followed by a few deconvolutional layers, producing heatmaps as an output. Illustration of model is on Figure 9. Deconvolution proved to be a better alternative to the upsampling used in [20, 5]. Simple baseline achieved better results with simpler architecture.

Deconvolution layers are attached to the last convolution layer called C_5. In the default version, 3 deconvolution layers are attached with a stride 2 and 256 filters with size 4×4. Each deconvolution layer, therefore, increases the size of the feature map $2\times$. Each deconvolution layer is followed by batch normalization and a ReLU activation function. To generate the resulting heatmaps, a convolution layer with k 1×1 kernels is added at the end. Where k indicates the number of keypoints. Mean Squared Error (MSE) is used as the loss between predicted heatmaps and target heatmaps. Target heatmaps are generated as gaussian without normalization with mean on keypoint position $\mu = (x_k, y_k)$ and variance $\sigma^2 = 4$.

Experiments have shown that a larger heatmap size has a positive effect on detection accuracy. By default 64×48 heatmap size is used. As can be expected, greater depth of the backbone network and larger size of the input image have also a positive effect on accuracy. However, computational and memory requirements are also higher. Simple baseline with ResNet-50 as backbone achieved AP 70.4 on COCO val2017 dataset with input size 256×192 and 72.2 AP with input size 384×288. On COCO test-dev dataset with ResNet-152 and input image size 384×288 model achieved 73.7AP.

4.2.2 OpenPose Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

OpenPose [3] was proposed in 2018. The model adopts a bottom-up approach the advantage is a constant prediction time regardless of the number of people in the image. The correct association of detected keypoints with individual people is crucial for this approach. To address this issue, OpenPose uses Part Affinity Fields (PAFs). PAFs are 2D vector fields that encode the position and orientation of the human body between 2 connected keypoints.

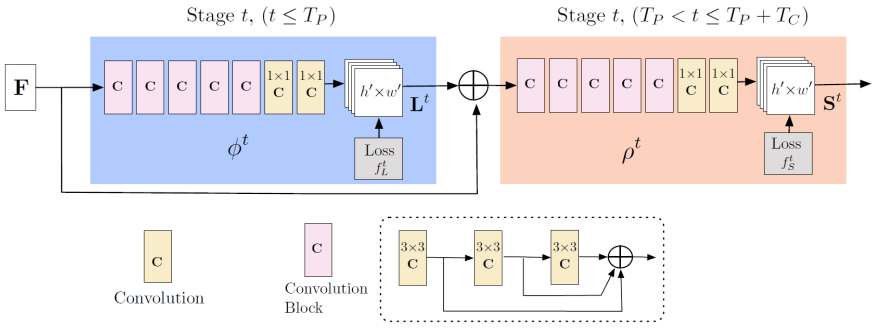


Figure 10: Architecture of OpenPose model [3].

Model architecture is shown in Figure 10. First, the initial 10 layers of VGG-19 are used to obtain feature maps $\boldsymbol{F}$. Then follows CNN for prediction of PAFs, and then CNN for keypoint heatmaps prediction. In Figure 10, there is the CNN for PAFs shown in blue and the CNN for heatmaps shown in orange. Output from the first CNN is set $\boldsymbol{L}$ of PAFs, where $\boldsymbol{L} \in \mathbb{R}^{C \times w \times h \times 2}$, C is the number of PAFs and is determined by the number of keypoint connections, for COCO dataset it is 19. The output of the second CNN is set $\boldsymbol{S}$ of keypoint heatmaps, where $\boldsymbol{S} \in \mathbb{R}^{J \times w \times h}$, J is the number of keypoints, 17 in COCO dataset.

Results from both CNNs are iteratively refined. An input in the first iteration of the first CNN are feature maps only and output are PAFs: $\boldsymbol{L}^1 = \phi^1(\boldsymbol{F})$, ϕ^1 refers to first CNN. In subsequent, iterations the PAFs from previous iteration are concatenated with feature maps and also passed as an input: $\boldsymbol{L}^t = \phi^t(\boldsymbol{F}, \boldsymbol{L}^{t-1})$. Similarly, the feature maps $\boldsymbol{F}$ and the PAFs $\boldsymbol{L}^{T_P}$ are passed to the second CNN as an input for first iteration: $\boldsymbol{S}^{T_P} = \rho^{T_P}(\boldsymbol{F}, \boldsymbol{L}^{T_P})$, ρ^{T_P} refers to the second CNN. In subsequent iterations, previous output is also added to input: $\boldsymbol{S}^t = \rho^t(\boldsymbol{F}, \boldsymbol{L}^{T_P}, \boldsymbol{S}^{t-1})$.

After the prediction stage, it is necessary to make connections between detected keypoints to form poses. Non-maximum suppression is used on heatmaps to get final predictions. It is possible to get more than one prediction for each keypoint as multiple people can occur in the image. For all detected keypoint k_i^n and k_j^m, that are connected, where n and m is number of candidates for keypoint k_i and k_j, bipartite graph is created. Nodes of the graph are keypoint candidates k_i and k_j each keypoint from k_i is connected to each keypoint from k_j by an edge. PAF predictions are used to weigh the edges of the graph. Then, the goal is to choose a subset of edges where no two edges share node and the weight of chosen edges is maximal. To obtain optimal matching they used the Hungarian algorithm [15]. After all keypoint connections are obtained in this way, the whole pose can be assembled for all people in the image.

Figure 11: **Left**: Part Affinity Fields (PAFs). The color encodes orientation. **Right:** Illustration of vector field [3].

Groundtruth heatmaps $\boldsymbol{S}^*$ generated during training must contain positions of keypoints of all people. First, heatmaps for all keypoints for all people are generated separately. Heatmap is generated as gaussian without normalization with center on the position of keypoint. The final heatmap $\boldsymbol{S}_{\boldsymbol{i}}^*$ for keypoint i is obtained as maximum at all position in separate heatmaps of keypoint i. Maximum is taken instead of average, to maintain a precision of nearby peaks. PAFs are 2D vector fields representing the association between two connected keypoints. Similar to the heatmaps, ground truth PAFs $\boldsymbol{L}^*$ must contain information of all people in an image and are also generated separately for individual people. The final PAFs are obtained as an average of PAFs corresponding to individual people. PAFs have a value of zero everywhere except the position between two keypoints, where they are defined as a unit vector with a direction from one keypoint to another. Illustration of PAFs is on Figure 11.

OpenPose did not achieve state-of-the-results on COCO at the time. On COCO test-dev set model achieved 64.2 AP. The main advantage of the model is its high speed.

4.2.3 TransPose Keypoint Localization via Transformer

TransPose [31] was presented in 2020. Model is based on Transformer architecture, adopts a top-down approach, and predicts keypoints using heatmaps. The goal of the work was not only to predict keypoints but also to explain what inputs contributed to the predicted outputs. Explanation of how the model predicts position from the input can give us a better understanding of how the model works and why in some scenarios fails. CNNs can not simply visualize contributions of individual inputs, on the other hand, attention in Transformer architecture can be utilized to achieve this goal. To capture global dependencies, CNNs gradually enlarge their receptive field. Only deep layers work with the context of the whole image. The nature of the self-attention in the transformers makes it possible to obtain a relationship between any positions in the input already in the first layers.

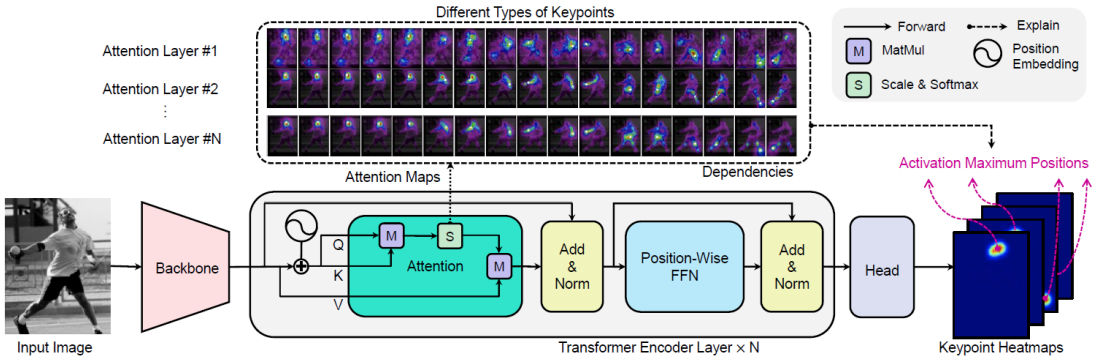


Figure 12: Architecture of TransPose model [31].

The proposed model can be split into three parts, architecture is in Figure 12. Input is an image of a single person as the model adopts a top-down approach and output is a heatmap for every keypoint. The first part is backbone CNN for low-level feature extraction. They used two different backbones ResNet [23] and HRNet [29]. Only the first few layers of networks are used. The second part is Transformer. The authors followed the standard Transformer architecture [27], but only the encoder part is used. N encoders are stacked in series, followed by the feed-forward network. Output from backbone is image feature map $\boldsymbol{X_f} \in \mathbb{R}^{d \times H \times W}$, feature map is than flattened into sequence $\boldsymbol{X} \in \mathbb{R}^{L \times d}$, where L is number of d-dimensional vectors $L = H \times W$. $\boldsymbol{X}$ serves as the transformer input. The last part is a head module. Output from Transformer $\boldsymbol{E} \in \mathbb{R}^{L \times d}$ is reshaped back to original shape $\mathbb{R}^{d \times H \times W}$. Than 1×1 convolution is used to

get final heatmaps $H \in \mathbb{R}^{K \times H \times W}$, where K is a number of keypoints. Final predictions are obtained from heatmaps as positions with maximal value. Target heatmaps are generated in the same way as in SimpleBaselines [30].

Due to the self-attention layer, it is possible to capture what locations in the image contribute to keypoint prediction the most. In Figure 13 is illustrated dependency of keypoints on other positions in image. It is noteworthy, that the occluded keypoints partly depend on the position of nearby keypoints.

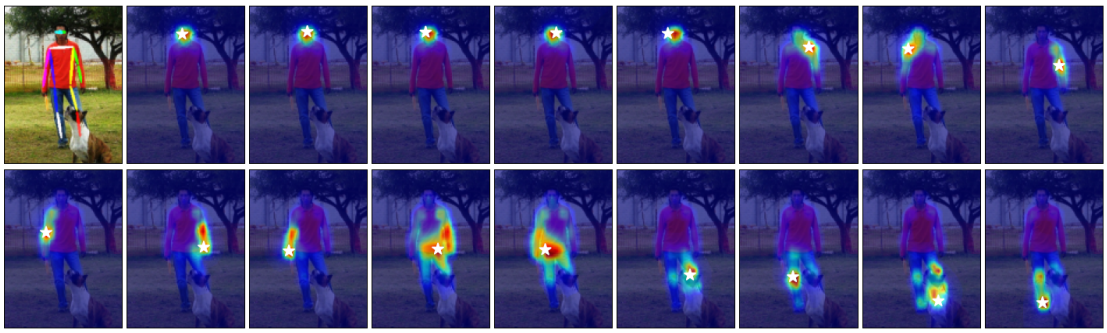


Figure 13: Visualization of predicted keypoints and dependency areas [31].

Model was trained with two diferent backbones [10, 29]. Both variations achieved comparable results to SimpleBaselines [30] and HRNet [29] with less model parameters and higher inference speed. On COCO validation dataset with input image size 256×192 and ResNet backbone model achievd 72.6AP with HRNet backbone 75.8AP. Results on COCO test-dev dataset with same image size was 75.0AP for model with HRNet backbone.

HPE methods are getting more accurate, but improving efficiency remains a challenge. Most methods still can not be used for real-time tasks and maintain accuracy. Methods based on CNNs achieved great results, but have limited ability to capture the global context in the initial layers. Downsampling image and increasing receptive field in deeper layers can capture global context but also increase memory and computational complexity. Despite that CNNs are used as a powerful tool for image feature extraction. On the other hand, Transformers are very effective in capturing the interaction between any two positions in an image. Due to the high complexity of the self-attention layer, it is first necessary to reduce the input image. For this task are often used CNNs as they can reduce image size and extract low-level features. The results on COCO dataset of all the methods presented above are in Table 1.

Method	**Input Size**	**AP**	**Dataset**
SimpleBaseline-50	256×192	70.4	val2017
SimpleBaseline-50	384×288	72.2	val2017
TransPose-ResNet	256×192	72.6	val2017
TransPose-HRNet	256×192	75.8	val2017
SimpleBaseline-152	384×288	73.7	test-dev
OpenPose	256×192	64.2	test-dev
TransPose-HRNet	384×288	75.0	test-dev

Table 1: Results on COCO validation set and test set.

5 Model and data

This section will describe the modifications of the DePOTR model and the preprocessing and postprocessing of the data. Two different derivations of the original model will be made. The first variant predicts keypoints positions directly as the original DePOTR model. The second variant uses heatmaps to make predictions. Data processing is done so that the models can be trained on the COCO dataset keypoint detection task.

5.1 Regression based model

Deformable POTR was proposed for 3D hand pose estimation. Because 3D hand pose estimation and 2D human pose estimation are similar tasks, there was no need to make major model changes. An obvious change was in the output dimension of the output feed-forward network. Because the positions of the keypoints in the original model were predicted in 3D, the output dimension had to be reduced by one.

Humans can be in image only partly, but the output of the model predicts all keypoints every time. According to the COCO protocol for results evaluation, it is unnecessary to provide whether a keypoint is in the image. Keypoint visibility information is taken from ground truth data. Therefore, contribution to the loss from keypoints that were predicted but are out of the image should not be taken into account during training. In the original DePOTR, smooth L1 loss is used. To address the issue of keypoints outside of the image, a binary mask is added to the loss:

$$L = \frac{1}{N_j} \sum_{i}^{N_j} \sum_{j}^{2} W(i) \cdot L1_{smooth}(y_j^i, \hat{y}_j^i) \tag{6}$$

$$L1_{smooth}(x, y) = \begin{cases} 0.5 \cdot (x - y)^2, & \text{if } |x - y| < 1 \\ |x - y| - 0.5, & \text{otherwise} \end{cases} \tag{7}$$

Where y_j^i and $\hat{y}_j^i$ are j^{th} components of the i^{th} predicted and target keypoint, W is binary mask with $W(i) = 0$ when i^{th} keypoint is outside of the image and N_j is the total number of keypoints.

5.2 Heatmap based model

Diagram of the modified model is in Figure 14. This model will be referred to as DePOTR-HM. At the output of the DePOTR is a feed-forward network. Its inputs are the vectors produced by the decoder and its outputs are the coordinates of K keypoints. In DePOTR-HM, the feed-forward network outputs are instead vectors for individual keypoints of dimension 192. Vectors are reshaped to $\mathbb{R}^{K\times16\times12}$. The spatial dimension is then increased by two transposed convolutions with stride 2 and 3×3 kernel. Transposed convolution is followed by one convolution layer which generates final heatmaps with size $\mathbb{R}^{K\times64\times48}$. The output dimension of the feed-forward network should be adjusted according to the selected output size of the heatmap. In this case, heatmap size was chosen based on image size 256×192 which is often used in HPE. As the loss is used, Mean Squared Error (MSE). As in previous modification of the model, keypoints that are not in the image do not contribute to the loss.

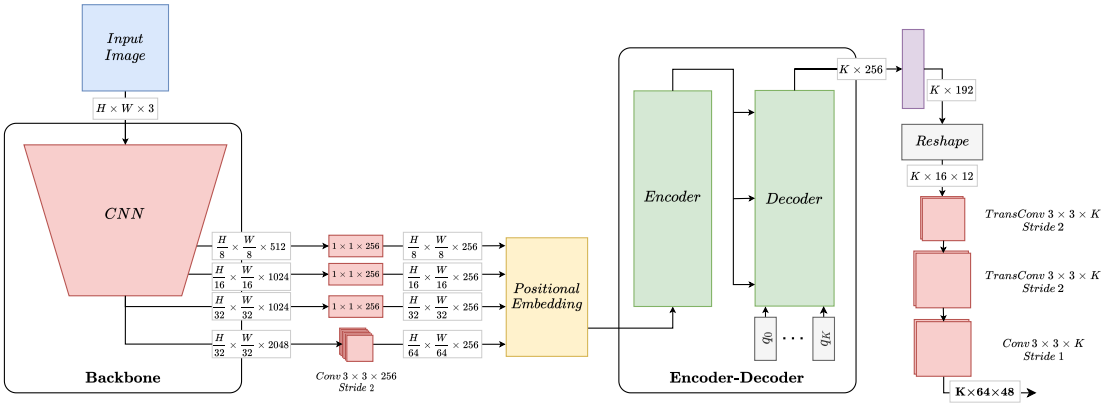


Figure 14: Diagram of modified DePOTR model.

5.3 Data processing

Before inserting data into the model, the data must first be preprocessed. It is necessary to ensure that there is only one person in the image. However, because there can be multiple people in an image, people are always cropped from the image and the cropped images are resized to the same size. To crop people from the image, it is needed to know their exact position (bounding box) in the image. For training and validation sets annotations contain bounding boxes for individual people. After cropping the image, it is

also necessary to transform the coordinates of the keypoints from the coordinates of the original image to the coordinates of the new cropped image. Finally, the new coordinates are converted from absolute to relative values. Augmentations may be performed before cropping the image. The inverse process is performed with the model output. The coordinates are first converted from relative to absolute values. The coordinates are then transformed back into the coordinates of the input image.

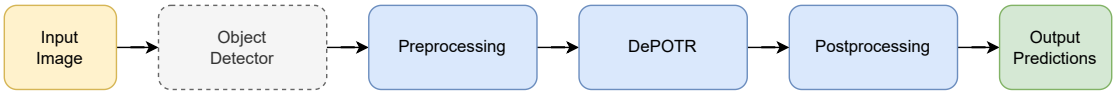


Figure 15: A schemantic diagram for inference.

In the test phase, when the positions of the keypoints are not known and we want to predict them, the procedure is almost identical. The schematic diagram of inference is in Figure 15. The position of the people must be provided with the image. This is achieved either by adding a human detector or providing a file with pre-detected human positions. Then the images are processed in the same way as during training. The image is cropped at the locations specified by the file or detector. Cropped images are one by one inserted into the model. The resulting keypoints are converted to absolute coordinates and transformed into the coordinate system of the original image.

The data processing is almost identical for the DePOTR-HM. When generating target data, the process is the same until the coordinates are converted to relative values. Instead, a heatmap is created for each keypoint. Heatmaps are created as a non-normalized 2D gaussian with the center at the position of a given keypoint. Processing of the output of the model also differs only slightly. The output keypoints positions are determined as the position of the pixel with the maximum value in the heatmap. Then the coordinates are transformed back to the coordinate system of the original image.

5.4 Augmentations

Data are necessary for neural network training. Data should sufficiently represent the solved problem. Usually, the more data the better. The goal of training is to teach the model to generalize. The model should learn the general properties of training data and should be able to apply this learned information to unseen data. With a low number of

training images, overfitting may occur. Overfitting is a process when the model does not generalize but memorizes correct outputs for input data in the training set. Overfitted model performs very well on training data but poorly on validation data.

Data augmentation is one of the approaches to prevent overfitting on small datasets. Large datasets often do not overfit on training data, but adding appropriate augmentations increases the robustness of the model. Augmentation is the process by which data is modified so that important information is preserved and information that is not important to the model is changed. Augmented images should correspond to real possibilities. This makes it easier for the model to learn what information to focus on. For example, brightness can vary greatly between images but is not important for most computer vision tasks, and changing it should not change the result. Therefore, increasing the brightness variability in the training data may improve the results on the validation and test data.

The COCO dataset used for DePOTR training is quite large. But augmentations still proved to be beneficial. Augmentations were performed using a library Albumentations [2]. Because the dataset contains images from diverse scenes, the selected augmentations modify the image only slightly. The selected augmentations are horizontal flip, rotation, Gaussian blur, color shift, and random adjustment of brightness and contrast. Horizontal flip is important because the direction in which the pose is facing should not be important for estimation. Most of the people in the dataset stand upright, but even so, not all poses are directly perpendicular to the image frame, due to the movement of the people or the rotation of the camera. Therefore, it was appropriate to use rotation with a small angle. Color shift and brightness and contrast adjustment are used to simulate different types of scene lighting. Lastly, Gaussian blur is used. Blurring may occur in the image for multiple reasons therefore, it is reasonable to simulate it. One of the reasons is when a person is at a greater distance in the image. Even though the image may have a high resolution, the people in the distance are blurred when cropped and resized. Also, images with lower resolution are not uncommon. Examples of individual augmentations are shown in Figure 16. The resulting augmentations used during training are shown in Figure 17.

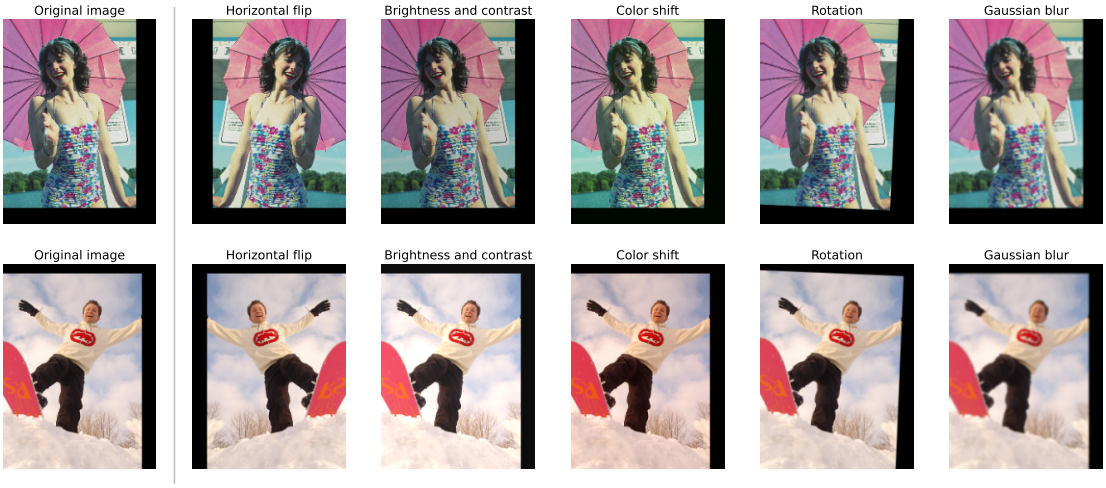


Figure 16: Examples of individual augmentations.

Figure 17: Examples of augmentations.

6 Experiments and results

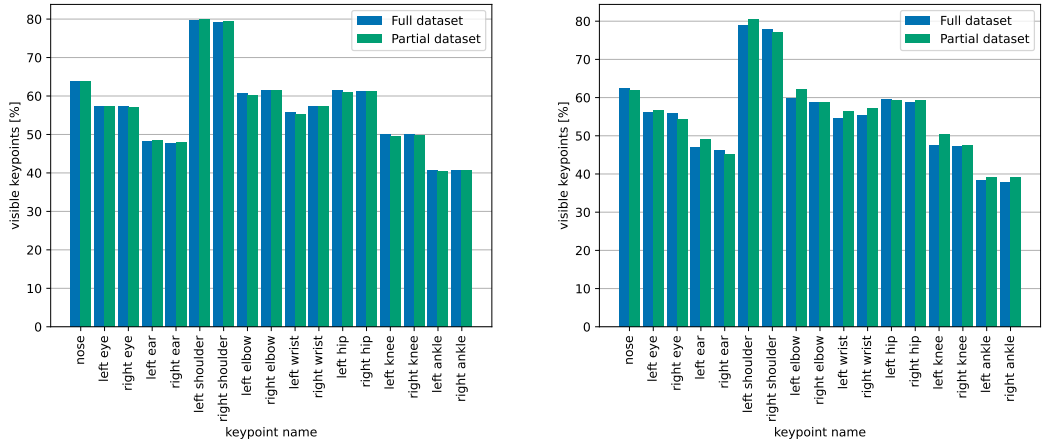
In this section, the various settings of the models during training will be described and the achieved results will be analyzed. Finally, the results will be compared with the results of the SimpleBaseline and TransPose models.

To improve results, Transpose and SimpleBaseline used a procedure called flip test. In this procedure, a prediction is made for the image and its vertically flipped variant. The result of the flipped image is flipped back and the average of both predictions is used as the final result. This precedent was also performed with the results of the DePOTR. This resulted in an increase of 1-3% AP.

6.1 Experiments setting

The models were trained on the COCO dataset. The dataset is quite large and training takes a long time. Therefore, the DePOTR was first trained on a subset of the COCO dataset, which accounted for approximately 10% of the total number of images. Experiments with this subset should serve as a guide to make it easier to choose the correct settings for training with the full dataset later.

For the results, to match the results that would be achieved when training with the full dataset, the distribution of data in the subset must be as similar as possible to the full dataset. As the indicators for the splitting, were chosen the total number of individual keypoint occurrences in all images and what portion of the keypoints is occluded. Both criteria can affect how well the model learns to detect individual keypoints. Figure 18 compares the percentage of individual keypoints in the full dataset and in the selected subset. In all cases, approximately 55% of all possible keypoints are visible, 8% is in the image but occluded and 37% is out of the image.



(a) Train set

(b) Validation set

Figure 18: Comparison of percentage of keypoints in the full dataset and subset.

During training were used official annotation data for the training and validation set. Files with pre-predicted human positions (bounding boxes) were used for the final detections and calculation of AP on the validation set and the test set. Data were produced by the detector with an AP of 56.4 on the validation set and 60.9 AP on the test set. The same files were used for the SimpleBaseline and TransPose models.

6.2 Experiments with the subset

All experiments are performed with a transformer with 6 encoder, 6 decoder layers and each attention layer has 8 attention heads. The embedding dimension is 256, the dimension of the feed-forward layer in a transformer is 1024, and sine positional embedding is used. The experiments are performed with the DePOTR, which is based on direct regression. Later, the findings from training on the subset are also applied to DePOTR-HM when training on the full dataset.

When training a neural network, the choice of optimizer can greatly affect the training speed and the resulting loss. The basic but often used is SGD with momentum [25]. Another common option is the Adam [13] optimizer, which belongs to the group of adaptive optimizers. Figure 19 compares the validation loss during training with the SGD and AdamW [18] optimizers. When training with AdamW, the model achieved significantly better results. The loss function of the model with SGD suggests that there

would be further improvement with longer training, but it is unlikely that the model would achieve better results than the model with AdamW.

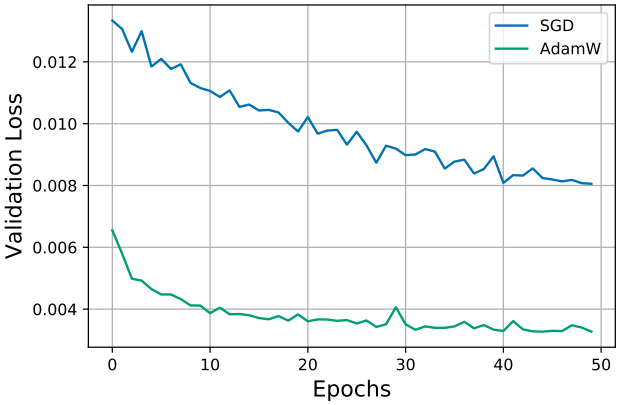
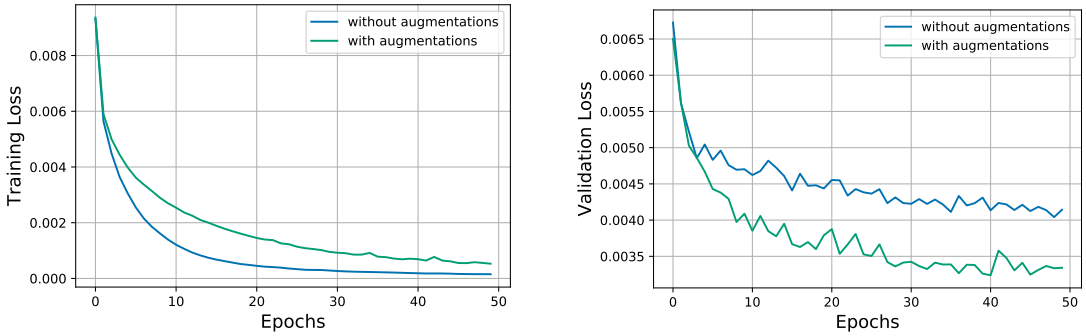


Figure 19: Comparison of different optimizers.

As already mentioned, appropriately selected augmentations can be beneficial. This statement is supported by Figure 20, which shows a comparison of validation loss during training with and without augmentations. The training loss is lower for the model without augmentations, because the model gets the same images each iteration, making learning easier. The model with augmentations is forced to learn more general information about poses resulting in lower validation loss.



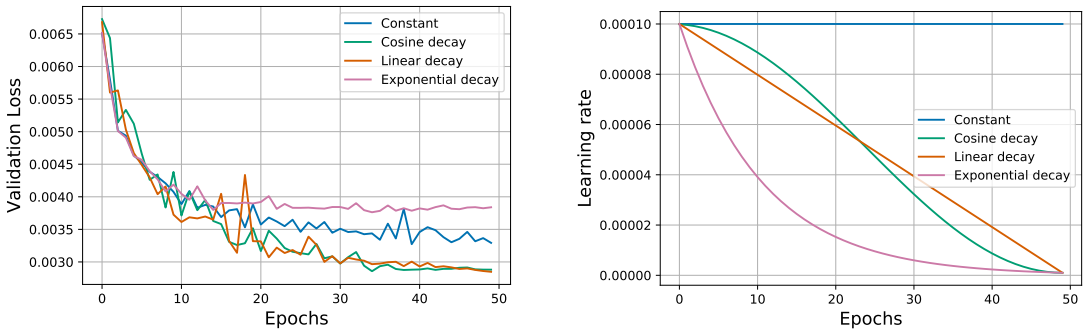
(a) Training loss.

(b) Validation loss.

Figure 20: Comparison of training with and without augmentations.

Crucial for training is the correct choice of learning rate. When chosen too high model will not learn, and when chosen too low, learning will be slow. Also, the learning rate should gradually decrease with the increasing number of epochs. It is possible to use

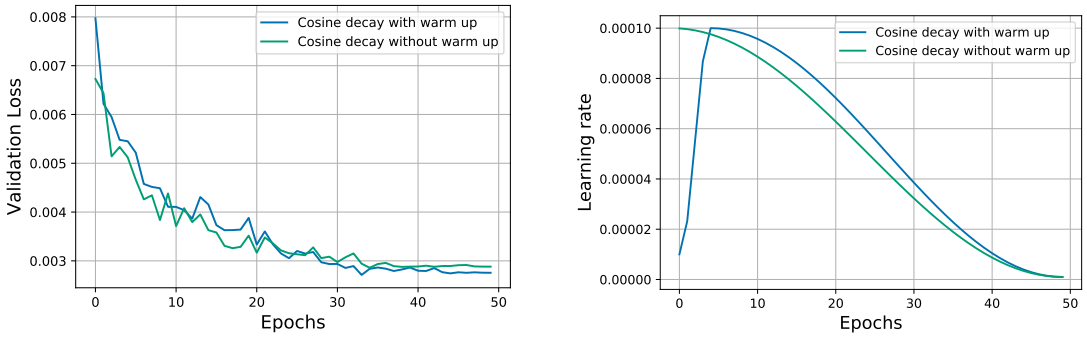
learning rate schedulers to change learning rate during training. Figure 21 shows some possible learning rate schedulers and validation losses for models trained with these schedulers. The best results were achieved by schedulers with linear and cosine decay [24]. The worst results were achieved by the scheduler with exponential decay because the learning rate decreased too soon.



(a) Validation loss. (b) Learning rate.

Figure 21: Comparison of different learning rate schedulers.

Another variation of the scheduler may be the addition of a warm-up phase. During the warm-up phase, the model starts at a low value of the learning rate, and during several iterations the learning rate increases. After this phase, the process continues as in the case without the warm-up phase. The goal of this phase is to allow adaptive optimizers to calculate the initial gradient statistics correctly. Figure 22 includes results of training with and without the warm-up phase. The model with a scheduler with warm-up achieved slightly better results.



(a) Validation loss. (b) Learning rate.

Figure 22: Comparison of schedulers with and without warm-up phase.

Images are first processed by the backbone, and feature maps on different levels are then passed to the transformer. The backbone network used affects the resulting feature maps. Deeper networks can better capture semantic information in the image. Figure 23 shows a comparison of ResNet networks with different numbers of layers. The deepest network with 152 layers achieved the best results.

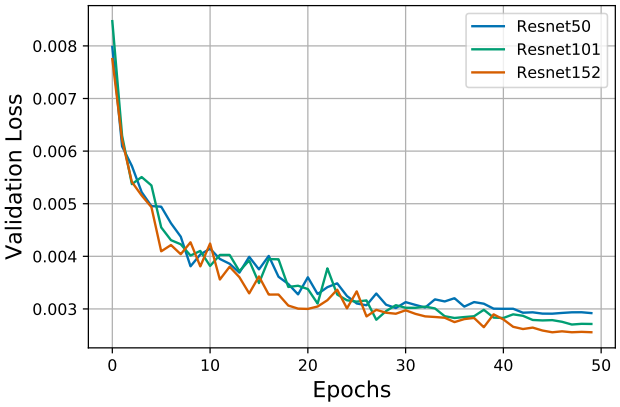


Figure 23: Backbone depth comparison.

Another important aspect is the size of the input image. If the image is not large enough, details may be lost, resulting in lower accuracy. Results of two models trained with different input image sizes in Figure 24 suggest that in this case increase in size did not greatly increase accuracy. When using a larger input image, the computational complexity also increases, in this case, the training time for the model with a larger input image was almost 2 times longer.

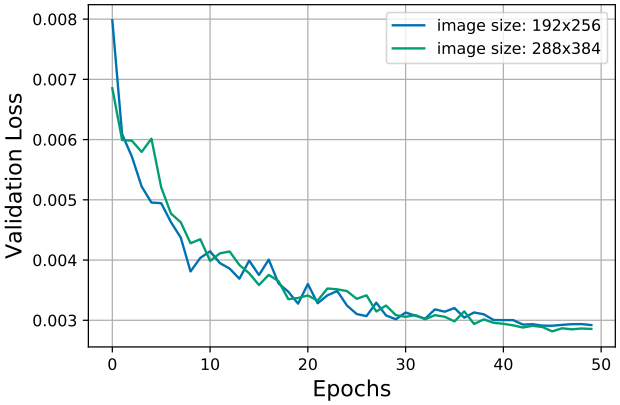


Figure 24: Comparison of models trained with different input image sizes.

The last comparison is made with models with a different number of training epochs. Validation losses are on Figure 25. Both models were trained with a scheduler with cosine decay and warm-up phase. The longer trained model achieved better results. There was an improvement in AP of approximately 2.2%.

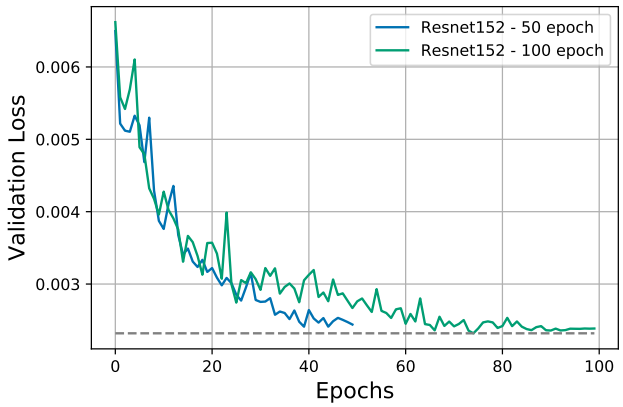


Figure 25: Comparison of different lengths of training.

6.3 Experiments with the full dataset

6.3.1 DePOTR

Backbone	Epoch	Image size	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
ResNet50	50	256 × 192	51.3	78.6	56.5	49.9	56.4	63.9	87.8	69.9	60.1	69.2
ResNet50	50	384 × 288	52.7	**80.3**	58.3	50.8	58.2	64.6	**88.2**	71.7	60.4	70.6
ResNet50	100	256 × 192	**53.8**	79.1	**59.2**	**52.4**	**58.9**	**66.1**	87.8	**72.3**	**62.2**	**71.7**
ResNet50	100	384 × 288	52.7	79.8	59.0	50.9	58.0	64.7	88.1	72.2	60.6	70.6
ResNet152	50	256 × 192	52.0	**79.4**	57.2	51.2	56.7	64.7	88.2	71.4	61.1	69.9
ResNet152	50	384 × 288	51.4	79.2	56.6	49.9	56.4	63.5	87.4	70.2	59.6	69.0
ResNet152	100	256 × 192	**52.7**	79.0	**58.2**	**51.4**	**57.8**	**65.1**	**87.7**	**71.7**	**61.3**	**70.6**
ResNet152	100	384 × 288	51.6	78.4	56.9	50.4	56.4	63.8	87.2	70.2	59.8	69.4

Table 2: Results on the validation set of DePOTR model trained on the full dataset.

In Table 2 are summarized results from training with full dataset. Experiments with a subset indicated that the most suitable optimizer is AdamW and a scheduler with a warm-up phase and linear decay. This setting was used to train all models. Previously presented augmentations were also used. Experiments were performed with two different backbones, two different input image sizes, and two different training lengths.

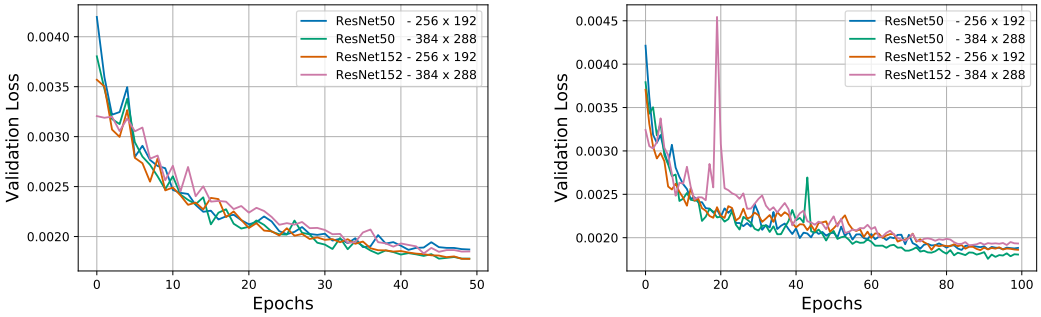


Figure 26: The resulting validation loss of models trained on the full dataset.

The results of all models are very similar. The resulting AP shows that the models with ResNet50 performed better in three out of four cases than the models with ResNet152. In all cases, longer training time increased AP. Increasing the size of the input image from 256×192 to 384×288 improved the result in only one case.

6.3.2 DePOTR-HM

Backbone	Epoch	Image size	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
ResNet50	50	256×192	58.8	83.7	64.9	56.9	63.8	65.4	88.9	71.3	62.2	70.1
ResNet50	50	384×288	**63.3**	**85.5**	**70.3**	**61.0**	**68.8**	**69.6**	**90.4**	**76.2**	**66.0**	**74.7**
ResNet152	50	256×192	61.3	84.2	68.5	60.0	66.3	68.6	90.1	75.3	65.4	73.2
ResNet152	50	384×288	**64.9**	**86.5**	**72.7**	**62.3**	**70.9**	**71.5**	**91.3**	**78.5**	**67.6**	**76.9**

Table 3: Results on the validation set of DePOTR-HM model trained on the full dataset.

In Table 3 are the results of the DePOTR-HM trained on full COCO dataset. The training was performed with the same optimizer and scheduler that were used in the DePOTR training. Two different backbones and two image sizes were used. Heatmaps output size was in all cases 64×48.

An increase in input image size improved the results of both backbones. A deeper backbone also improved the results in both case. Contrary to the previous variant of DePOTR, the best result was achieved with ResNet152. In comparison, the best DePOTR-HM model achieved 11.1% AP better results than the best DePOTR model. All models were trained for 50 epochs, a longer training scheme could improve the results. Further improvement could be potentially achieved by enlarging the output heatmaps when training with larger images.

6.4 Results

6.4.1 DePOTR best result

The model that achieved the best results was trained with ResNet50 backbone, transformer with 6 encoder and decoder layers each with 8 attention heads. As optimizer was used AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For learning rate was used scheduler with 5 epoch long warm-up phase followed by linear decay. After warm-up learning rate was $1e-4$ for transformer and $1e-5$ for backbone. The final learning rates were 100 times smaller. The batch size was 32 and the input images had a resolution of 192×256. The model achieved 53.8% AP on the validation set and 55.7 % AP on the test set.

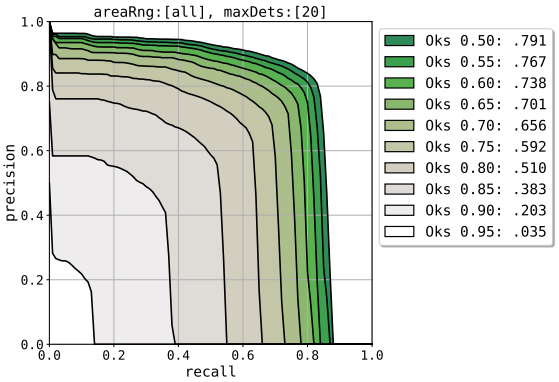


Figure 27: Results of DePOTR for different levels of OKS on validation set.

In addition to evaluation, COCO provides code for analysis of results [22]. Code generates precision-recall (PR) curves for different OKS levels and PR curves for different types of errors. RC curve shows precision for different recall values. High recall means that the model has a low number of false negatives and high precision indicates how accurately these results were predicted. The goal is to maximize the area under the PR curve. In Figure 27 are PR curves for different OKS thresholds. The OKS value indicates how far the predicted keypoint can be from the ground truth position to still be considered correctly detected. Therefore, high values of OKS will have a higher rate of false negatives. This can also be seen in the results of the DePOTR model. The predicted keypoints are not accurate enough, resulting in a low recall at high OKS levels. On the other hand, the results for low OKS levels are significantly better and even with increasing recall, accuracy remains high.

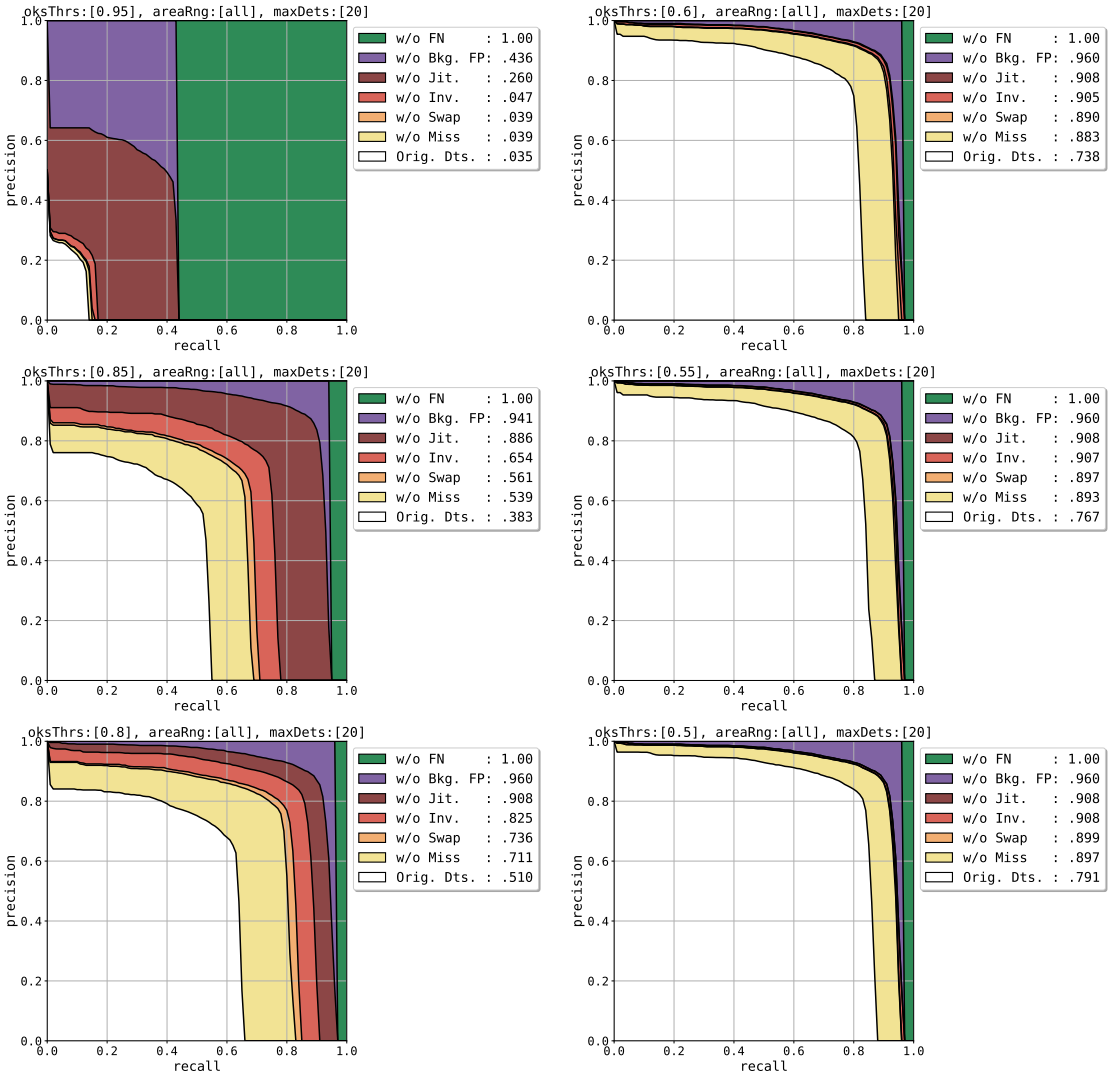


Figure 28: Impact of different errors on results for 6 different OKS thresholds.

In Figure 28 are graphs also generated by [22]. The graphs show what the area under the PR curve would be without individual errors for different OKS levels. Six different types of errors are evaluated.

- Miss: The detected keypoint is not near the ground truth position.
- Swap: The Keypoint is detected semantically correctly, but the position is confused between the two people.
- Inv. (Inversion): Two pair keypoints (right, left ankle) of one person are confused.

- Jit. (Jitter): Keypoint is near the ground truth position, but not within the OKS region.
- Bkg. FP (False Positive): Detected keypoint was not matched with ground truth annotation.
- FN (False Negative): Keypoint was not detected (annotation was not matched with any detected keypoint).

The error for results with a high OKS level is largely due to jitter error. Jitter can easily be caused by keypoints that are occluded and their position needs to be determined from the surrounding environment. As the OKS threshold increases, jitter error decreases. The error for low OKS results is mostly due to miss errors. The lower rate of inversion errors in low OKS results compared to high OKS results indicates that the error occurs at keypoints that are close to each other and potentially overlap, an example might be a person standing sideways.

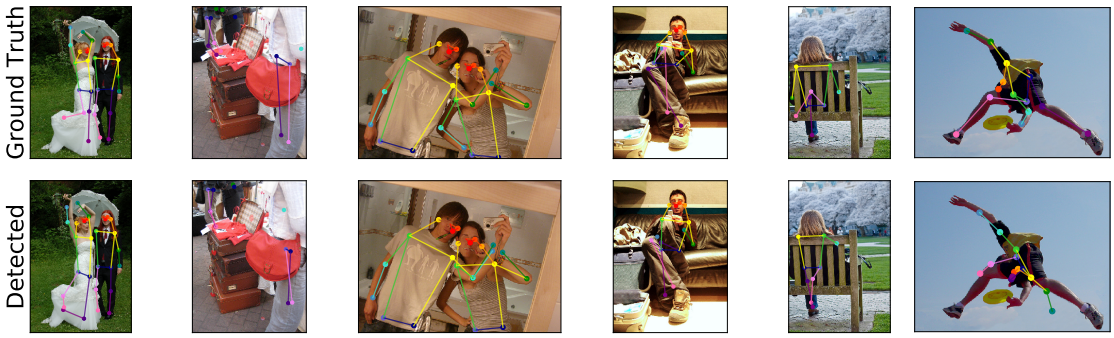


Figure 29: Examples of common detection failures.

Figure 30: Examples of correct detections.

Figure 29 shows common examples of situations in which a model often fails. Scenes where the body is largely occluded or only a small part of the body is in the image are often problematic. Also, scenes in which the pose is upside down or in another complex setting tend to be poorly predicted. On the other hand, in cases where a person is largely visible, predictions tend to be accurate. Even in cases where people are in close proximity. Examples of some correct predictions are in Figure 30.

6.4.2 DePOTR-HM best result

The model used almost the same settings as the DePOTR. The only difference was the larger size of the input image, and the deeper backbone. On the validation set, the model achieved 64.9% AP, and on the test set achieved 65.0%. In Figure 31 are PR curves for different OKS levels. In comparison with the DePOTR model, DePOTR-HM achieved more precise results for low recall values on high OKS levels.

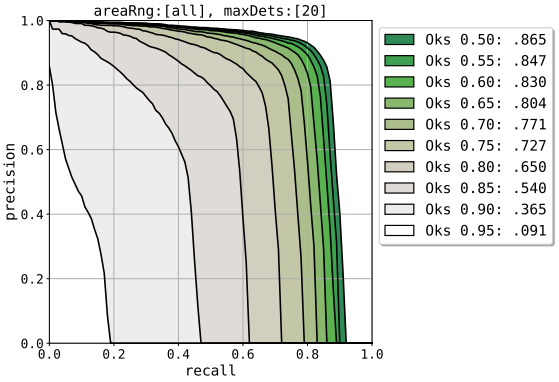


Figure 31: Results of DePOTR-HM for different levels of OKS on validation set.

6.5 Comparison with other models

Method	Backbone	Image size	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
SimpleBaseline	ResNet50	384 × 288	71.5	91.1	78.7	67.8	78.0	76.9	**94.6**	83.5	72.3	**83.2**
TransPose	ResNet50	256 × 192	**72.1**	**91.3**	**79.9**	**68.5**	**78.2**	**77.3**	**94.6**	**84.3**	**73.0**	**83.2**
DePOTR	ResNet50	256 × 192	55.7	83.8	61.4	54.1	60.5	66.5	90.2	72.8	62.8	71.6
DePOTR-HM	ResNet50	384 × 288	63.4	87.8	70.6	61.1	68.1	69.2	91.8	75.9	65.8	73.9
SimpleBaseline	ResNet152	384 × 288	73.7	91.9	81.1	70.3	80.0	79.0	95.2	85.6	74.8	84.9
TransPose	HRNet	256 × 192	**75.0**	**92.2**	**82.3**	**71.3**	**81.1**	**80.1**	**95.4**	**86.7**	**75.9**	**85.9**
DePOTR	ResNet152	256 × 192	54.5	83.2	60.1	53.1	59.1	65.4	89.7	71.9	62.0	70.2
DePOTR-HM	ResNet152	384 × 288	65.0	88.6	72.4	62.5	70.0	71.0	92.5	78.0	67.3	76.1

Table 4: Comparison of DePOTR and DePOTR-HM with SimpleBaseline and TransPose on COCO test set.

Table 4 compares the results of the DePOTR and DePOTR-HM with SimpleBaseline and TransPose on the COCO test set. The results for SimpleBaseline and TransPose were generated using official implementations and provided pre-trained models.

DePOTR and DePOTR-HM did not outperform state-of-the-art methods. When trained with ResNet50 SimpleBaseline outperformed DePOTR by 15.8% AP and DePOTR-HM by 8.1 AP. TransPose outperformed DePOTR by 16.4% AP and DePOTR-HM by 8.7% AP. Expect DePOTR all models achieved better results with deeper backbones. SimpleBaseline improved by 2.2% AP, TransPose by 2.9% AP, and DePOTR-HM by 1.6% AP. In the Figure 32, is a comparison of models AP on different OKS levels. In the Figure 33, are examples of predictions.

In this case, results showed that the heatmap-based approach is superior to the regression-based. The reason for the lower accuracy of both DePOTR and DePOTR-HM may be that the models do not generalize enough. In the DETR model, individual query vectors specialized to detect objects with a certain size and position in the image. However, this is not possible with the DePOTR. Query vectors in DePOTR are interpreted differently. To each keypoint class is assigned one query vector. This might cause query vectors to better learn to predict keypoints in positions and poses that appear in the dataset more often. Resulting in poorer accuracy of less frequent poses.

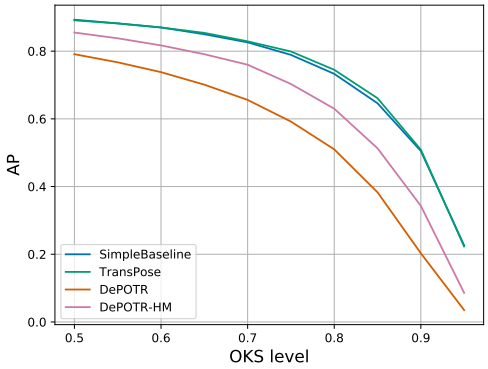


Figure 32: Comparison of AP on different OKS levels.

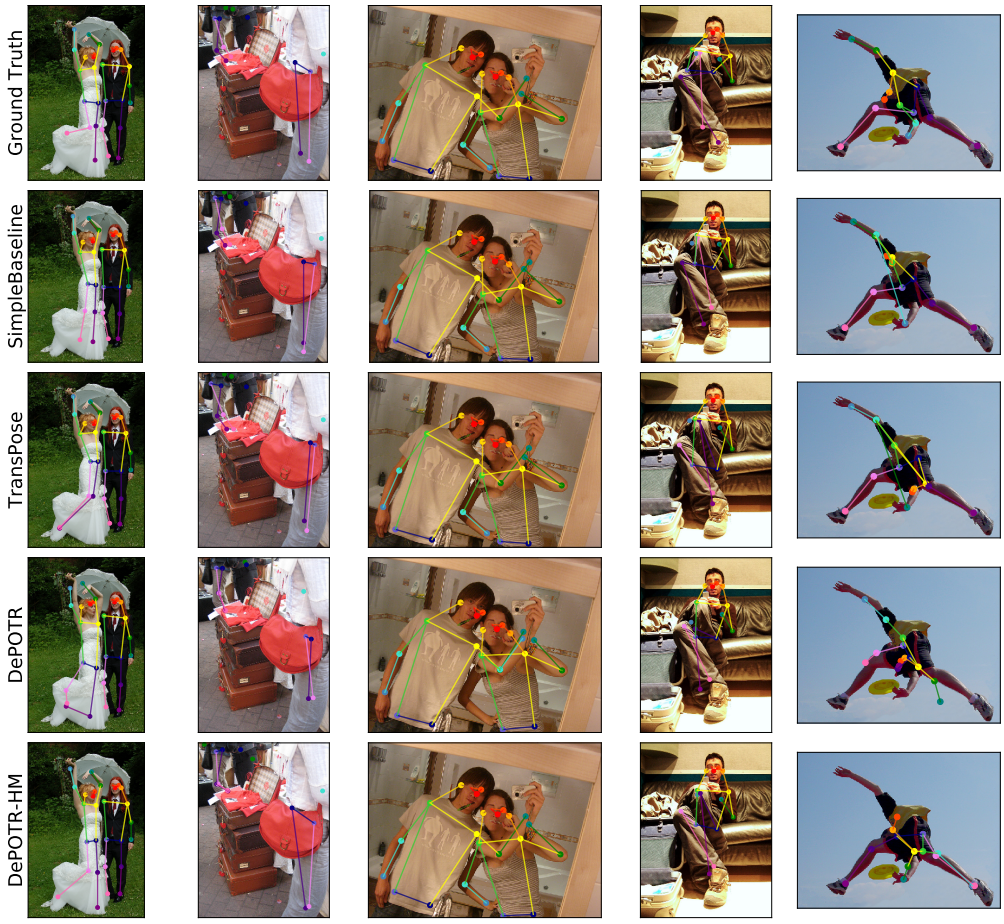


Figure 33: Examples of predictions of all models.

7 Conclusion

The first part of this work has described the task of human pose estimation and some challenges of this task. Also, various approaches to solving this task and methods that successfully solved this task were described. The two basic types of models are regression-based models and heatmap-based models. Heatmap-based models generally perform better.

The second part described modifications of the DePOTR model, which was originally designed for 3D hand pose estimation. Two variants were derived from the model. The first maintained a regression approach for estimation. Only changes in the output of the model and minor changes in loss function were necessary. The second variant of the model was modified so that the output of the model was heatmaps from which the position of keypoints was subsequently determined. This was achieved by adding a small structure consisting of transposed convolution layers and convolution layers. Besides modifications to the model, it was necessary to adjust the input and output of the model so that it could be trained on the COCO dataset.

Then followed the training of the models. First, were performed experiments with a subset that accounted for approximately 10% of the entire dataset. These experiments helped determine which settings of the model to use when training with the entire dataset. Both variations of the model were then trained with different backbones and input image sizes. The regression-based model did not achieve better results with a larger input image and a deeper backbone. The best achieved results on the test set were 55.7% AP. Model based on heatmaps achieved significantly better results. Also, an increase in image size and deeper backbone was beneficial. The best result on the test set was 65.0%. It should be noted that this improvement was achieved only by additional processing of the transformer output. This improvement was still not enough to outperform the results of the TransPose and SimpleBaseline models.

Further changes to the model architecture would probably be needed to further significantly improve the results.

All source files are available at: https://github.com/strakaj/Automatic-Human-Pose-Estimation-using-Neural-Network

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] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[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.

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[5] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018.

[6] Myung Jin Choi, Vincent YF Tan, Animashree Anandkumar, and Alan S Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 12:1771–1812, 2011.

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[9] 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.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] 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, jul 2014.

[12] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010. doi:10.5244/C.24.12.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[15] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[16] 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.

[17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. MSCOCO keypoint detection evaluation metric. https://cocodataset.org/#keypoints-eval.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[19] Hrúz Marek, Gruber Ivan, Boháček Matyáš, Kanis Jakub, and Zdeněk Krňoul. DePOTR: Deformable Transformer for Hand Pose Estimation. https://github.com/mhruz/POTR, 2022.

[20] 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.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] Matteo Ruggero Ronchi and Pietro Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[24] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[25] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[26] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[27] 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.

[28] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*, sep 2018.

[29] 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*, 43(10):3349–3364, 2020.

[30] 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.

[31] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Towards explainable human pose estimation by transformer. *arXiv e-prints*, pages arXiv–2012, 2020.

[32] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.