Multimedia Tools and Applications (2019) 78:29357–29377
https://doi.org/10.1007/s11042-019-7433-7

Deep neural networks for human pose estimation from a very low resolution depth image

Piotr Szczuko[1]

Received: 1 February 2018 / Revised: 3 January 2019 / Accepted: 26 February 2019 /
Published online: 7 March 2019

Abstract
The work presented in the paper is dedicated to determining and evaluating the most efficient neural network architecture applied as a multiple regression network localizing human body joints in 3D space based on a single low resolution depth image. The main challenge was to deal with a noisy and coarse representation of the human body, as observed by a depth sensor from a large distance, and to achieve high localization precision. The regression network was expected to reason about relations of body parts based on depth image, and to extract locations of joints, and provide coordinates defining the body pose. The method involved creation of a dataset with 200,000 realistic depth images of a 3D body model, then training and testing numerous architectures including feedforward multilayer perceptron network and deep convolutional neural networks. The results of training and evaluation are included and discussed. The most accurate DNN network was further trained and evaluated on an augmented depth images dataset. The achieved accuracy was similar to a reference Kinect algorithm results, with a great benefit of fast processing speed and significantly lower requirements on sensor resolution, as it used 100 times less pixels than Kinect depth sensor. The method was robust against sensor noise, allowing imprecision of depth measurements. Finally, our results were compared with VGG, MobileNet, and ResNet architectures.

Keywords Depth image · Pose estimation · Artificial neural networks · Deep learning

1 Introduction

Many modern computer interfaces employ gesture tracking and user pose estimation. The foremost example is a Kinect sensor [1, 23] utilizing a depth camera and a dedicated algorithm based on random forests. Common approaches to joints tracking involve processing of 2D

✉ Piotr Szczuko
szczuko@multimed.org

[1] Faculty of Electronics, Telecommunications and Informatics, Multimedia Systems Department, Gdańsk University of Technology, Gdańsk, Poland

grayscale or color images [31, 32], 2D depth maps [7, 14, 23] or 3D body geometry scans obtained by a fusion of many depth images [20]. Depending on the used method the accuracy is influenced by numerous problems such as obscuration of body parts in case of crossed legs or arms, front-back confusion, pose ambiguity, and low accuracy of joints localizations and orientations. Usually a high processing power and high resolution sensors are required to achieve satisfactory precision. The approach presented here involves a very low resolution depth images and employs neural network-based regression run in real time on a medium performance CPU.

Kinect sensor was chosen as a reference for the results presented in this paper because there were no other as accurate and as well-established methods for extracting joints positions from depth image of human body. Kinect uses structured light to obtain depth map of size 640 × 480 pixels with 11-bit resolution expressing the distance from the sensor. The pose tracking algorithm is based on decision trees classifying each pixel as belonging to a particular body part, based on local features of the depth image. As a result the scanned geometry is first divided into body parts, and then their 3D space coordinates are estimated. The average precision of all joints localizations is 0.731, assuming that the distance less than 0 .1m from the ground truth is a true positive. For wrists and hands the precision reaches 0.67 (Section 5). The method requires relatively high resolution of the depth map and a high processing power (on a GPU 200 frames are processed per second, and on a 8-core CPU only 50 frames) [23].

The high requirements of Kinect were addressed in the presented research. The goal was to propose and evaluate a DNN (Deep Convolutional Neural Network) multiple regression method for estimating 3D coordinates of body joints, able to operate on a very low resolution depth image (100 less pixels than Kinect), and more efficient than the prior method.

Aims of the presented work were: to verify how selected neural network architectures perform with joint localization task, to speed up the localization process, and to reduce the required size of input depth map comparing to the reference Kinect algorithm [1, 23]. Low resolution depth images of 60 × 50 pixels were used. A very fast regression on body joints locations in 3D space was observed. Even in case of a sensor noise, large distance, occlusions and reaching off the screen, the achieved precision was comparable with Kinect (Section 6).

The reminder of this paper is organized as follows: related work is presented in Section 2. Section 3 documents the dataset, Section 4 describes MLP and DNN architectures, research methodology, and training procedure, Section 5 presents results of joints localization, Section 6 is dedicated to improvement upon those results by further training the best network, and Section 7 concludes the paper.

2 Related work

DNN networks are successfully used in many image processing applications, including pose estimation (Table 1). Such networks are usually composed of several layers, of three different types [11, 13]: 1) convolutional layers taking rectangular areas of a defined window size from a previous layer, then performing convolution with a given set of kernels, and returning filtered results, which are interpreted as feature maps; 2) pooling layers, reducing the size of feature maps, and introducing invariance to shifts or distortions, by e.g. choosing maximal or median value of a rectangular window slid with a defined step over the input; 3) fully connected layers decoding the internal state of the network and mapping features into a desired output format.

DNN architectures were already used for pose estimation by other researchers. Usually localization accuracy is expressed by mean average precision (mAP), by defining a threshold

Table 1 Pose estimation methods (by input data type, and size)

Authors	Image resolution	Dataset	Method	Inference time	Accuracy/Precision
2D color image, only bounding box location					
Mahendran [18]	n/a	12 classes	ResNet50, stages 1–4. Features extraction, rotation estimation	n/a	Given the bounding box, a class was determined, then rotation estimated with error 2.6–29 degrees
Howard [8]	224 × 224	Face attributes classification	MobileNet, Depthwise convolution	n/a	mAP 0.88
Jung [10]	224 × 224	Real traffic, Vehicles	ResNet 50 and 101 layers, classification and localization	n/a	Bounding box only, mAP 0.79
Simonyan [24]	224 × 224	1000 classes	VGG-D, classification and localization	n/a	Bounding box location 25% error
Mahendran [17]	224 × 224	12 classes	VGG-M, classification and localization	n/a	Median geodesic angle errors in range 8.99–35
2D color image, body parts pose estimation					
Li [15]	112 × 112	Defined actions, 15 subjects	Deep CNN pose regression	n/a	Average pose error 77–190 mm, mAP 0.2–0.95
Deeppose [32]	220 × 220	Movies, sports poses	DNN-based regressor, cascaded	0 .1s	69% of results are within 0.5·(body part's bounding box size) from GT
Tompson [31]	320 × 240	Movies, sports poses	Sliding window ConvNet for body parts heat maps	0. 05 s	90% of results are within 12 pixel from GT
Depth image, body parts pose estimation					
Our method	60 × 50	Synthetic poses	MLP and DNN	0.0 01 s – 0. 01 s	mAP 0.86
Ganapathi [5]	128 × 128	Real world actions	Model-based search	0 .1s	Average pose error 50–100 mm
Ye [34]	176 × 144	Real world actions [5]	Model based comparison, generative	60s – 150 s	mAP 0.95
Wang [33]	216 × 176	Daily poses, indoor	Deep CNN, body parts heat maps, joints regression	0. 04 s	mAP 0.91
Crabbe [4]	224 × 224	Stairs climb	Deep CNN	0. 01 s	Uses dedicated reduced "pose space", preventing comparison
Ly [16]	640 × 480	Synthetic and real poses	Evolutionary programming	70 min	84%
Kinect [23]	640 × 480	Synthetic poses	Random forests for parts classification	0. 02 s	mAP 0.731
Park [21]	640 × 480	Golf poses	Random regression forests, verification forests	0. 04 s	Average error 33 mm, mAP 0.98 (only for poses limited to golf swings)

for acceptable distance between estimated and actual location (absolute distance in millimeters, or relative to particular body part bounding box size, or as a fraction of torso size), and then treating each detection within this distance as a *true-positive*. The ratio of such *true-positives* to all estimated coordinates is mAP. Average absolute errors are also relevant for assessing the performance.

Significant efforts are made for pose estimation from 2D photos or video frames, important mainly for video indexation. In a Deeppose framework 2D color natural images were analysed to estimate upper body pose, achieving speed of 0.1s for 220 × 220 image on 12-core CPU [32]. Tompson et al. used deep convolutional network architectures to detect body joints inside two sliding windows of 128 × 128 and 64 × 64 pixels over 2D images of 320 × 240 pixels [31]. The processing speed was 0.05 s per image. A more complex approach with two strategies for DNN training was evaluated: training for pose regression and body part detectors, and a pre-training when the pose regressor is initialized using a weights of the network first trained for body part detection [15]. Input 2D color images with 112 × 112 pixels were processed by the architecture containing 3 conv. layers with pooling, effectively scaling down the input to 10 × 10 feature maps. Depending on the complexity of the analyzed pose the mean average errors were within a range 77 mm to 190 mm, and the mAP precision was within a range 0.2 to 0.95.

Using depth information from time-of-flight sensors, instead of RGB image, simplifies segmentation, and allows for more accurate pose estimation not only in 2D image plane, but in 3D space. An approach similar to the Kinect method was implemented, dedicated strictly for processing sequences of poses in a golf swing [21]. It utilizes random regression forests cooperatively voting to localize joints in a 640 × 480 depth image, followed by a random verification forests that evaluate and optimize the votes to improve the accuracy of clustering that determine the final position of anatomic joints. For joints of interest mean average precision mAP of 0.98 and mean location error of 30 mm were achieved. Despite such high accuracy, the major drawback is limitation to defined action of playing golf. The method is able to process a single frame in 25 ms on a powerful i7-6850K CPU.

Another limited application was described in [4], with DNNs applied for regression of body joints in a depth images presenting action of climbing up the stairs. The accuracy is expressed in a dedicated "pose space" with reduced dimensionality, therefore a direct comparison with other methods is not possible. Input 224 × 224 depth images are processed by an deep convolutional architecture (5 conv. layers, and 3 max pooling in-between) with a speed of 100 frames per second on a GTX750 GPU.

Highly complex method of model-based search was implemented and joints locations were compared with motion capture ground truth data in a work by Ganapathi et al. [5]. Average pose error was 0.05–0.1m (depending on a test sequence), but the processing speed was low: 100 ms per 128 × 128 depth image.

A cascaded approach was also studied [33], where first confidence maps of body parts are generated by a fully convolutional network analyzing 216 × 176 pixels depth image (6 conv. layers with kernels from 7 × 7 to 3 × 3, with max pooling, and three conv. layers with kernels of 1 × 1), generating 23 images of 19 × 28 pixels. Then the image patches and body parts templates are processed by a second architecture to obtain optimal configuration of joints. For precision assessment the distance between the predicted and the true joint position was accepted when laying within a defined fraction of a torso diameter. For 0.2 of this diameter, the mAP is 0.91. This method is able to process 25 frames per second on a GTX-980Ti GPU.

A model based approach involving a comparison of a newly acquired depth map of 176 × 144 pixels with all poses available in the database (nearest neighbor search) was implemented

[34]. The initial search is refined with a generative approach, i.e. 3D pose is modified and rendered to achieve depth image closely aligned with the input. Achieved mean average precision is high 0.95, but the method is too slow for real-time (60–150 s per frame), due to its long generative phase.

Another generative approach with genetic optimization of pose of a 3D body model was successfully implemented [16]. Due to the nature of the applied evolutionary method, this implementation is extremely slow as for a single image it requires 10,000 iterations lasting ca. 70 min on a powerful CPU.

Often the problem of estimating object's pose in the image is reduced to the location and orientation of a rigid object within a frame. In numerous approaches the bounding box coordinates are inferred from the image of the object, serving as its 2D localization, supplemented with a rotation estimation. Such efforts are usually focused on vehicles, house appliances, etc., as well as silhouettes, but without detailed body pose information. Deep residual networks (ResNet [6]) were applied for that scenario to simultaneous localization and classification of vehicles in a traffic, obtaining the localization estimated as a bounding box coordinates with mAP of 79.24% [10].

Other authors proposed new architecture with a shared feature network, i.e. first a classifier network operated on extracted features, and finally a respective pose regression networks were engaged separately for each of resulting 12 classes [18]. The feature extractor was based on ResNet-50 stages 1 to 4, then 3 additional FC layers were added and trained for each class pose estimation. The pose angle is estimated with error within range 2.6 to 29 degrees, depending on the class.

Also the popular VGG architecture, with 13 convolutions and 3 fully connected layers (VGG-D variant) was used in a localization task, estimating bounding box size and location of rigid objects of 1000 classes. This particular VGG-D had 138 millions parameters, and processed 224×224 RGB images [24]. VGG and Alexnet were trained to estimate rigid object orientation in a 2D images, first applying classification network, and then dedicated pose network for each object class, inferring 3 orientation coordinates [17].

An approach similar to our work was done previously for face images [25]. A cascade of three ConvNets was proposed aimed at 5 facial keypoints positions estimation. The networks outputs are fused to increase accuracy. It was shown that ConvNets are able to extract face features, and networks perform multiple regression, and predict all coordinates simultaneously, thus the geometric constraints are implicitly encoded, and local minima due to possible occlusions, and large pose variations can be avoided. Next stages are cascaded to refine initial predictions, resulting in high accuracy and reliability.

A recent attempt at efficient, universal and easily scalable architecture was presented [8], resulting in a MobileNet architecture with a new type of depthwise convolutions, capable to both classify and to localize objects in images.

Architectures simpler than DNN, namely multilayered perceptron networks (MLP) were used specifically for 3D finger tracking in the depth image [30], and among other classifiers bag-of-features and SVM were employed for gesture recognition [29], and hand tracking [1, 14].

Summarizing, we have observed that convolutional networks are able to extract meaningful features from the image, and were also proven to perform accurate regression. It was stated that the same features are useful both for the classification and for localization [10, 18], and networks such as VGG performed exceptionally good in both classification and localization competitions [6]. Therefore we aimed at determining and evaluating a DNN architecture

performing both tasks at the same time. Our proposed output layer has separate neurons for triples of 3D coordinates representing joint localization in real space, and pairs of 2D coordinates in the image plain for classification of body part in the image (Section 4).

On the other hand the MLP network without convolutions was also evaluated in our work, as an alternative approach. In this architecture each input pixel obtains different weights (contrary to shared weights of DNN), and pooling is not performed, allowing for high spatial resolution. Pooling in DNN is mainly motivated by assuring invariance to shifts, but it must be stressed here, that it is expected that shifts must influence output values in localization task. See also Section 7. for additional discussion on other state-of-the-art architectures.

Other important aspect is that previous approaches dealt with relatively large input images, requiring the sensor to be high resolution and positioned close to the object. In our scope lies the challenge of processing low resolution depth images. The smallest input depth map was 128×128 pixels in the work of Ganapathi et al. [5], but they were not employing neural network, and achieved large errors up to 100 mm. On the other hand the most similar approach to ours was by Wang et al. [33], used deep convolutional neural networks for joints regression, but it required 12 times more pixels than our input format, and had a quite complex cascaded architecture: image patches localization, then joints positions optimization. Our effort is aimed at proposing and evaluating streamlined deep convolutional architecture, and showing a difference between wider and narrower configurations, and very simple multilayered perceptron network also trained as a joint localization regressor.

3 Depth images dataset

A dataset with ground truth positions of every relevant joint was required for the supervised learning. Such data are hard to obtain in real conditions, as the only way to register actual joints positions in 3D space is to use expensive and demanding motion capture systems. Moreover, for a real person it would be a tremendous effort to take thousands of different poses, required in this experiment. Therefore a 3D graphics software was used to model, pose and render human figure, creating depth image and appropriate ground truth for each pose. Similar artificial database was used for training of decision trees in Kinect algorithm, but this particular set and similar ones were not publicly available. Therefore a new dataset was created for this research [27] consisting of realistic very low resolution depth images of a simulated human figure [26]. It was assumed, that the sensor has a horizontal field of view 57° (same as Kinect), and the depth image was rendered to 60×50 pixels in 8-bit greyscale (3-bit less, i.e. 8 times lower depth resolution than Kinect).

The human figure was positioned in an empty space, without a background and other foreground objects. In real applications to obtain the same conditions a separation of foreground object based on depth information should be performed, e.g. by assuming the body to be the object closest to a camera, and discarding the ones positioned further by comparing their depth values.

The dataset represents a wide variation of the upper body poses with biologically correct random joints positions, with wrists positions uniformly covering the available space in front of the figure. The body absolute position was changed randomly to introduce shifts in x, y and z directions relative to the fixed sensor. The movement ranges were limited to allow only far reaching hands to leave the frame, whilst the rest of the figure remained visible. 200,000 depth images were created, randomly split in 60/40 proportion into training and testing sets. In

testing set 56,000 images had hands inside the frame, 24,000 had one or both hands reaching out of the frame.

Each depth image was described by the *root* joint location in meters in 3D space (reflecting the figure absolute location in front of the camera), 3D coordinates in meters of both *hands, elbows, arms*, relative to the *root*, and 2D coordinates in pixels of both *hands, elbows,* and *arms* on the image plane. The pose state was defined as a set **P** of 3D coordinates of 7 joints:

$$\mathbf{P} = \left\{ \left(j_{\mathrm{x}}, j_{\mathrm{y}}, j_{\mathrm{z}} \right) \right\} \tag{1}$$

where a joint $j \in \{root, right\ arm, right\ elbow, right\ hand, left\ arm, left\ elbow, left\ hand\}$.

Pose estimation errors were defined twofold, as an error along particular axis $\varepsilon^{j}_{\mathrm{loc,dim}}$ (3) and as an Euclidean distance in space between true and estimated values $\varepsilon^{j}_{\mathrm{dist}}$ for a particular joint j (4):

$$\varepsilon^{\mathrm{j}}{}_{\mathrm{loc,dim}} = |j_{\mathrm{dim}} - j^{*}{}_{\mathrm{dim}}| \tag{3}$$

$$\varepsilon^{\mathrm{j}}{}_{\mathrm{dist}} = \| \left(j_{\mathrm{x}}, j_{\mathrm{y}}, j_{\mathrm{z}} \right), \left(j^{*}{}_{\mathrm{x}}, j^{*}{}_{\mathrm{y}}, j^{*}{}_{\mathrm{z}} \right) \| \tag{4}$$

where $\mathrm{dim} \in \{x, y, z\}$, j^{*} is an estimated joint coordinate, and $\| \cdot , \cdot \|$ is a distance metric between two points in the 3D space.

In the 2D image plane the localization of joints is a set **L** of pixel coordinates:

$$\mathbf{L} = \left\{ \left(i_{\mathrm{x}}, i_{\mathrm{y}} \right) \right\} \tag{5}$$

and errors are expressed as:

$$\varepsilon^{\mathrm{i}}{}_{\mathrm{loc,dim}} = |i_{\mathrm{dim}} - i^{*}{}_{\mathrm{dim}}| \tag{6}$$

$$\varepsilon^{\mathrm{i}}{}_{\mathrm{dist}} = \| \left(i_{\mathrm{x}}, i_{\mathrm{y}} \right), \left(i^{*}{}_{\mathrm{x}}, i^{*}{}_{\mathrm{y}} \right) \| \tag{7}$$

where $\mathrm{dim} \in \{x, y\}$ are image row and column coordinates, and Euclidean distance was used as a metric $\| \|$.

4 Training of neural network architectures

The applied methodology consisted of qualitative measurement of joint localization regression accuracy and comparison with the baseline Kinect algorithm. First the dataset was preprocessed to fulfill good practices of machine learning, then four neural networks were defined, with justification of their architectures, then the supervised training with error backpropagation was performed, and the results were presented and discussed. The most accurate network was further trained and tested on augmented dataset. Accuracy was expressed as distances between the estimate and actual joint position, and precision is calculated facilitating comparison with the Kinect.

The dataset consists of accurate 8-bit depth renderings of the human body model. Such an idealized representation is not suitable for machine learning, potentially leading to fast overfitting. Therefore the depth images were preprocessed by adding noise with a mean $\mu = 0$, and a standard deviation $\sigma = 1.25$ cm for a random half of training images, and $\sigma = 2.5$cm for the other half, thus in the first case 98% of pixels were within range $\pm 2 \cdot \sigma = \pm 2.5$cm from the actual depth, and within $\pm 2 \cdot \sigma = \pm 5$ cm in the latter case. All testing images were also noised ($\mu = 0$, $\sigma = 1.875$). The noise simulates rough surface of the body and cloths, and assure robustness to presence of wearable trinkets and accessories. Next, an average image over training subset was calculated and subtracted from every training and testing sample (Fig. 1), eliminating influence of absolute distance.

The output values of **P** and **L** (root location, joints relative locations, and image plane coordinates) were normalized to a range $< 0, 1 >$ using *min-max* rule, where each new normalized value x^*_i of x_i was calculated based on the minimal and maximal value of x (8):

$$x^*_i = (x_i - \min(x))/(\max(x) - \min(x)) \tag{8}$$

Normalization ranges $\{min(x), max(x)\}$ were obtained from the training dataset individually per each output and were used again to scale the network output back to original ranges for expressing absolute errors ε^i_{loc}, ε^i_{dist}, ε^j_{loc}, ε^j_{dist}. The normalization helps each output value and its gradient to influence the network weights in the same amount, increasing training speed [12].

Four architectures were considered, a Multi-Layered Perceptron (MLP), and three Deep Convolutional Neural Networks (DNN) described below.

Networks were trained to act as a multiple regression networks, and to return continuous values of predicted coordinates **P** and **L**, minimizing $\varepsilon^j_{loc,\,dim}$ and ε^i_{loc}.

MLP was composed of an input layer fed with 3000 depth image pixels, one hidden layer with 500 neurons, and an output layer with 33 neurons (Fig. 2). In each neuron a rectified

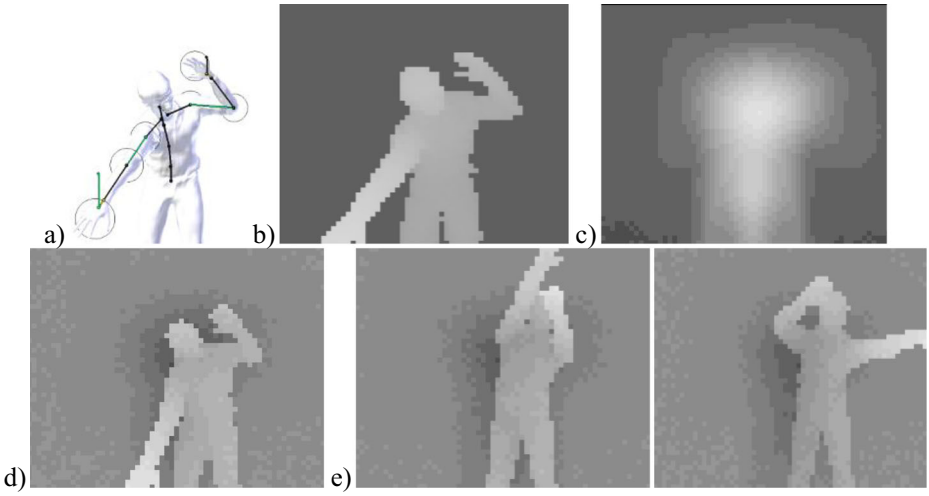

Fig. 1 Depth image preprocessing: **a** human model and upper body skeleton (circles mark joints of interest), **b** rendered depth image, **c** mean image of the training set, **d** noisy input with the mean subtracted, **e** samples of two other training images

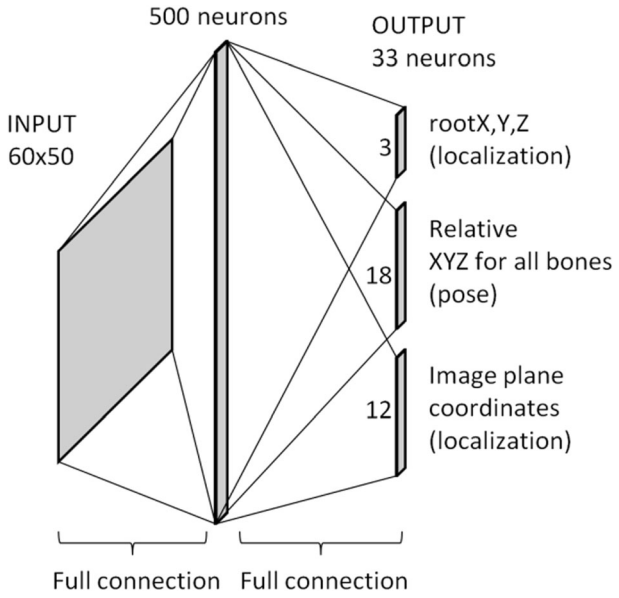

Fig. 2 Architecture of the MLP network

linear activation (ReLU) was used, following the suggestion that ReLUs are able to learn faster than *tanh* and *sigmoid* due to a lower computational complexity and the lack of saturation [19].

The DNN architectures used by us resemble the AlexNet [11]. The input for DNNs was an image of 60 × 50 pixels, first convoluted with 20 kernels with size 5 × 5, resulting in feature maps of size 60 × 50. These 20 maps were expected to grasp key local features of the depth image such as particular gradient orientations, edges or points, as well as reduce the impact of noise. Next, for each map the max pooling was performed within a sliding window of 2 × 2 with a stride of 2 × 2, resulting in 20 feature maps of size 30 × 25. After the next convolution layer and pooling more general feature maps of size 15 × 13 × 50, 15 × 13 × 33, or 15 × 13 × 20 were obtained, depending on the chosen number of feature filters in the second convolutional layer. In consequence three networks were created DNN50, DNN33, DNN20 varying in the size of second convolutional layer (Fig. 3).

The justification for chosen sizes is that DNN33 was expected to model 33 feature maps closely correlated with 33 output values, in one-to-one relation. Next, the number of feature maps in DNN20 was reduced to 60% (20 feature maps) as an attempt to train a more generalized network, and purposefully taking into account that some of output values were correlated, e.g. location in pixels in the image (i_x, i_y) with location along X- and Z-axis in space (j_x, j_z),. Finally the last architecture DNN50 involved 50% more features (50 feature maps in total) as an attempt to verify if such an increased capabilities should improve accuracy, possibly by modeling of subtle nonlinear relations between the camera distortions, and changes in size and distances due to an optical perspective, and extrapolation for hands locations outside of camera field of view.

Results of convolutions and pooling were passed onto a fully connected layer of 500 neurons dedicated to decoding internal state of the network, and finally to a fully connected layer of 33 neurons forming desired outputs. ReLU activation function was used in all neurons.

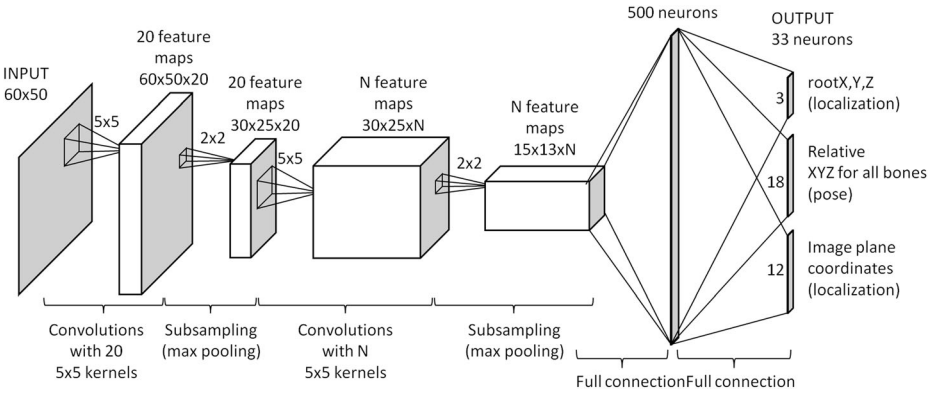

Fig. 3 Architecture of the DNN networks used in the experiment. N in the second convolution stage was 20, 33 or 50 for DNN20, DNN33 and DNN50 respectively

Architectures deeper than the presented one were not verified, as it was assumed that further convolution and pooling performed on 15 × 13 large feature maps of the stage 3 would lead to significant degradation of internal representation of joints coordinates. Pooling usually is expected to introduce shift invariance in classifier networks, but in our case the exact pixel location should be transformed into respective 3D coordinates. Filtration and pooling would obscure that dependency.

On the other hand, wider and narrower architectures were evaluated in our previous work [28], revealing that at least 20 kernels are required for satisfactory accuracy, and using over 50 kernels does not increase the accuracy any further.

The training was conducted in R environment [22] with MXNet package for neural networks [2, 3] on a 4 core i5–6500@3.2GHz CPU, with 16GB RAM, and on DGX Station.

For all networks the learning rate was 0.05, and momentum 0.3. The validation error was observed, and the training was stopped once the mean absolute validation error slowed its decrease and became lower than 0.0001 per epoch, or after a given number of epochs.

One epoch of MLP network training took approx. 5.5 s, and one DNN epoch took 75 to 87 s for DNN20 and DNN50 respectively (Table 2). MLP was trained for 6500 epochs, and DNNs for 2260 epochs.

The mean absolute error (MAE) for training data and evaluation data is presented on Fig. 4, visualized for a particular epoch number, and for the elapsed time. It can be noticed that:

- all DNNs generalize better than MLP, as the gap between the training and evaluation error was ca. 0.01 for DNNs and ca. 0.018 for MLP,
- DNNs achieve higher accuracy than MLP, as DNNs training errors were ca. 0.01 lower than MLP, and DNNs evaluation errors were ca. 0.018 lower than the MLP evaluation error,
- DNNs after 24 h of training surpassed MLP in the training error, and achieved lower evaluation error 12 h,
- the random large changes in the evaluation error for DNN33 before epoch 2260 were a result of too long learning, as the model started to overfit to training data, and lost its generalization capabilities.

Table 2 Training and inference speed for a given network architecture and CPU type, compared to baseline Kinect

	MLP	DNN20	DNN33	DNN50	Kinect
Average length of 1 training epoch [seconds, the lower the better]:					
Desktop i5@3.2GHz, 4 cores	5.5	75	78	87	–
Regression speed [frames per second, the higher the better]:					
Desktop i5@3.2GHz, 4 cores	32,520	1121	1119	1032	–
Mobile i5@2.5GHz, 4 cores	14,134	241	234	210	–
Mobile i5@1.6GHz, 4 cores	8000	150	134	130	–
Mobile i5@0.8GHz, 4 cores	5200	73	72	62	–
Unspecified 8 cores CPU	–	–	–	–	50
XBOX360 GPU	–	–	–	–	200

5 Neural networks performance and accuracy evaluation

For MLP network the pose estimation and joints location speed reached over 35,000 images per second, and for DNN over 550 images per second (averaged over all 60 k testing samples, measured on a desktop workstation with 4 cores, i5–6500@3.2GHz CPU) (Table 2). Thus the achieved speed exceeded the reference value of 200 images reported for Kinect algorithm run on the XBOX360 GPU [23].

Networks inference was also performed on a mobile CPU i5-5200U with adjusted clock speed. The achieved framerate depended on the architecture and clock speed, but even for 800 MHz the inference was faster than Kinect algorithm tested on a modern 8-core CPU [23].

The measured regression speed leaves large overhead for real-time applications, as the joint localization ought to be executed at reception of each new depth frame from a sensor. Assuming the modern Time-of-Flight camera is used with up to 90 frames per second, the neural network regressor can be run on any modern CPU. Considering the trade-off between the localization error (Section 6) and the speed, for applications where average precision of 0.7 is acceptable (only slightly below baseline Kinect mAP), one can apply MLP architecture achieving over 5000 fps on the slowest of tested configurations. For example, compact

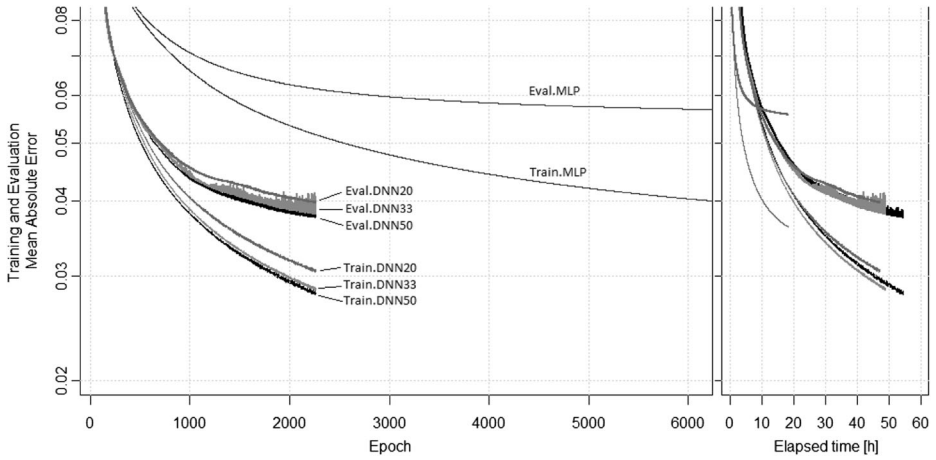

Fig. 4 Mean absolute errors during the training and evaluation for a given number of epochs and time

Raspberry Pi 3 computer, equipped with 1.4GHz quad-core CPU would handle the DNN33 network for joint localization in real-time.

The joints localization errors were expressed as distances along particular dimensions in 3D space $\varepsilon^{j}_{\text{loc, dim}}$ (Fig. 5) and as distances in x and y image plane coordinates $\varepsilon^{i}_{\text{loc, dim}}$ (Fig. 6). Errors were the largest for hands localization along the depth Y-axis in space, and along the vertical axis on the image. Hands locations were expressed by wrist position, what caused mistakes because the fingertips were the most extreme parts of the whole arm, probably disturbing the classification and localization (Fig. 8). Moreover the Y-direction is perpendicular to the image plane (depth) and its value in the depth image is highly influenced by the added noise. MLP network generally achieved the lowest accuracy, due to the simplicity of this architecture, as expected. DNN33 was the most accurate (see medians, 3rd quartiles, and upper whiskers), and DNN50 and DNN20 were slightly worse. This confirmed initial assumptions on DNN architecture – it was expected that matching the number of feature maps in the last convolutional layer of DNN33 with the number of 33 outputs would allow the most accurate modeling (Section 4). It can be observed that generally the errors are the largest for hands, their 4th quartile range exceeds 0.1m. Arms and elbows are located with high accuracy (less than 0.05 m), being a satisfactory result, taking into account the low resolution of input image. In the pixel space of 2D image plane (Fig. 6), the localization of arms and elbows is, in half of the test images, done with subpixel accuracy, as the median lies below 1pix. The largest errors are for hands, on extreme reaching 8 pixels, but 3rd quartile range is below 3pix.

The absolute distances $\varepsilon^{j}_{\text{dist}}$ were analyzed as well (Fig. 7). For DNNs errors have medians of 2, 4 and 6 cm for root, arms and elbows respectively, 10 cm for hands, and reaching 28 cm for hands. Generally MLP architecture accuracy was the worst. The main contribution to the large distance error was the low accuracy of localization of joints along Y-axis (depth), partially caused by the artificially introduced sensor noise. Generally, the noise increased distances between the ground truth coordinates and the estimated result. In case of hands the Y-axis error median reached 5 cm, up to 10 cm in the 3rd quartile, what relates to the absolute distance error up to 15 cm in the 3rd quartile. In our previous work [26], it was verified that increasing the noise variance in range 0.005–0. 01 m does not influence this accuracy, so it must be concluded that the proposed networks, trained with the presented dataset does not allow for more accurate localization of the hands, and the conditions for precision increase will

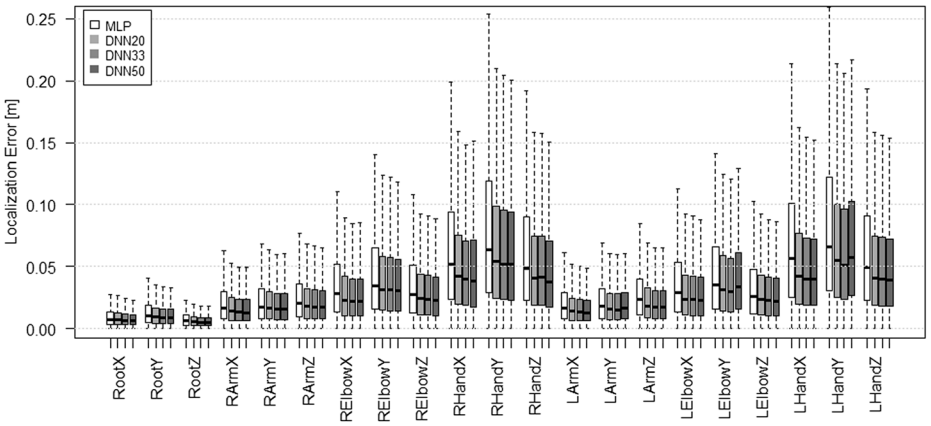

Fig. 5 Absolute errors in localization of joints (in m) along X,Y,Z-axes for MLP and DNNs architectures

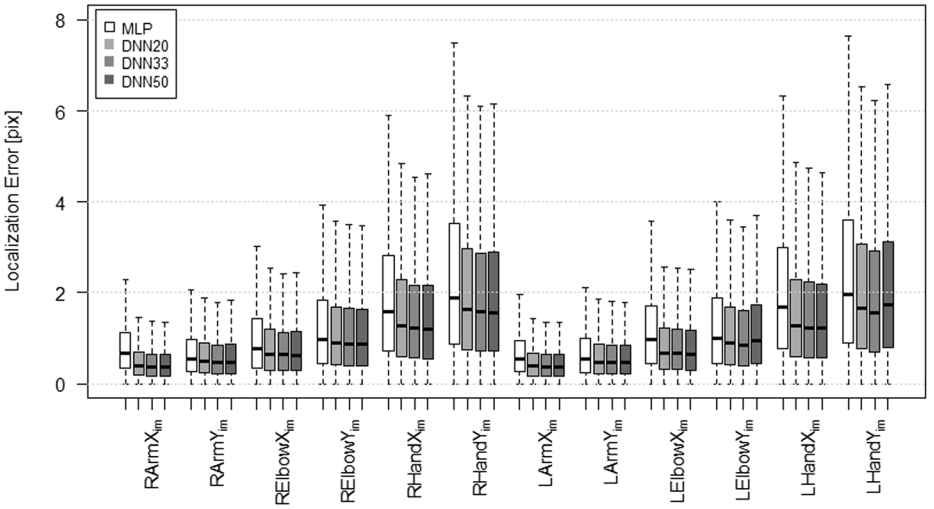

Fig. 6 Absolute errors in image plane localization of joints (in pixels) for MLP and DNNs architectures

be studied in the future. One of the scenarios for improvement will involve tracking the hand not only by the wrist location, but also by its fingertips, which, when straightened, can be even 15 cm apart from the wrist, what can be observed on fifth and sixth sample frame in Fig. 8.

Obtained results were compared with the original Kinect results by Shotton, et al. [23]. For Kinect it was assumed that a joint located within a distance 0 .1m from the ground truth location was counted as a true positive, and the reported average precision over all joints was 0.731, for hands reaching 0.67. The average precisions for the presented neural networks approaches were calculated following the same convention. The precision surpassed the reference results for arms and elbows locations but was lower for both hands (Fig. 9), probably

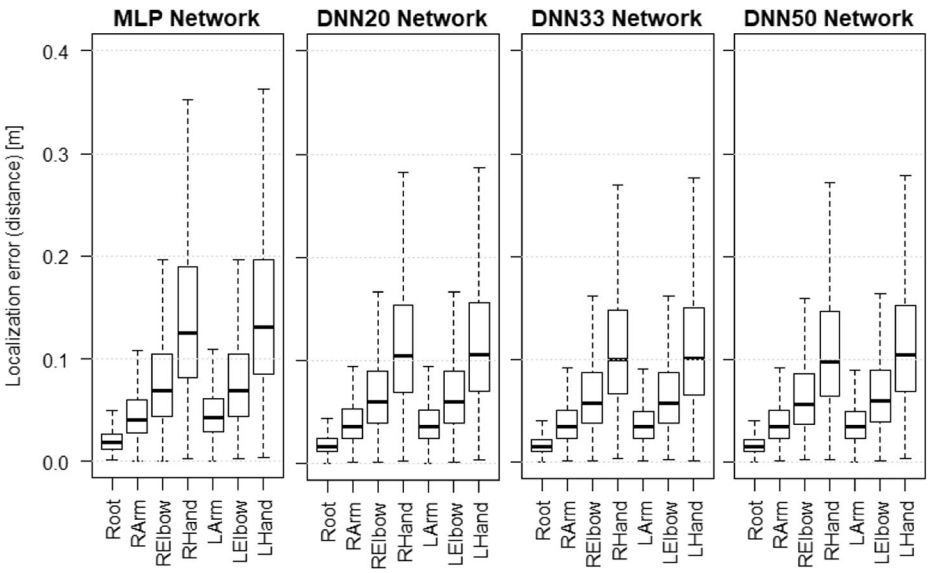

Fig. 7 Localization errors (distances in 3D space) for joints for MLP and DNNs architectures

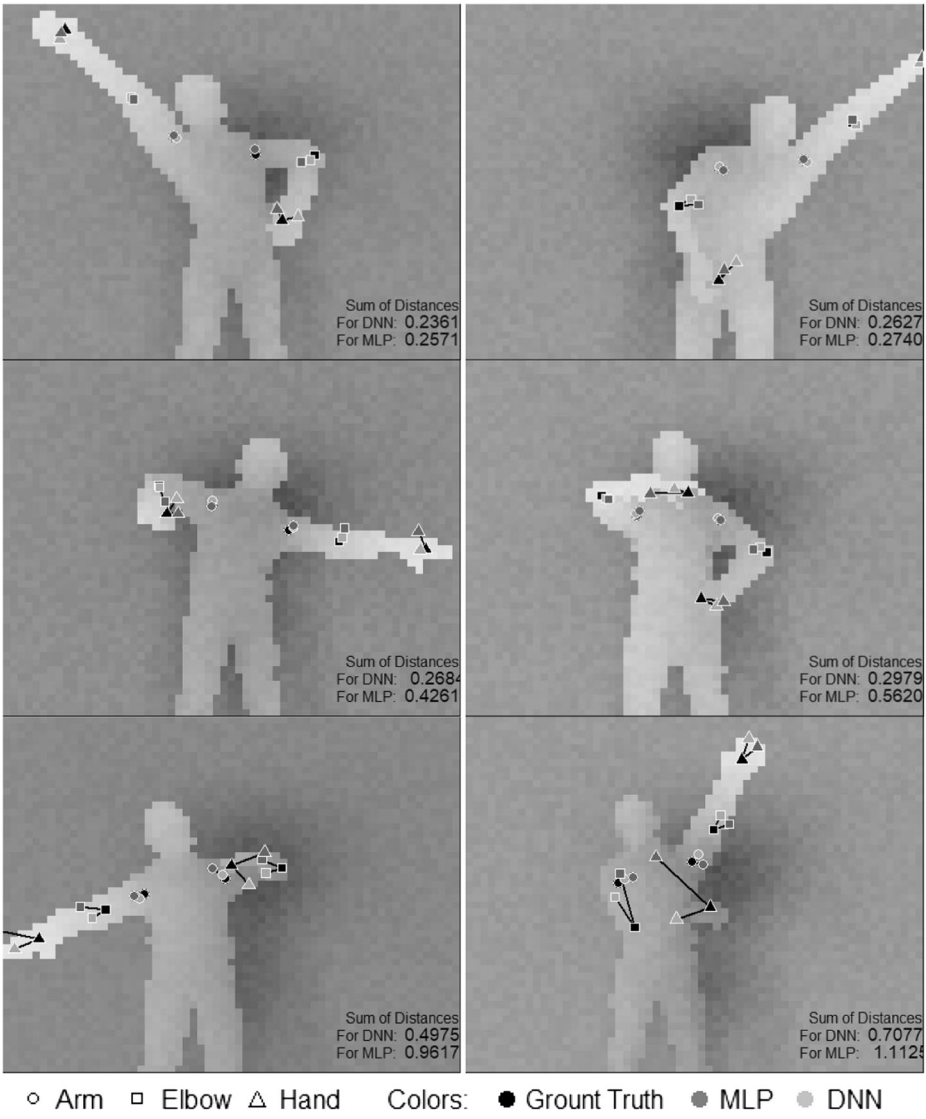

Fig. 8 Poses with low and high localization errors (sums of distances between Ground Truth and estimated locations). For simplification only locations of hands, elbows and arms are shown, GT and respective estimations are connected with lines

due to a high depth noise added to all images or too short training time, what was verified in the next step (Section 6).

It must be stressed here that the testing set was significantly more demanding comparing to the Kinect input images – it had a lower resolution, and lower bit depth and significant amount of additive depth noise (Section 3). In half of the cases (results above median) the hands errors exceeded 10 cm, but in 75% of cases (3rd quartile) errors were less than 15 cm, what is considered acceptable in case of that low sensor resolution.

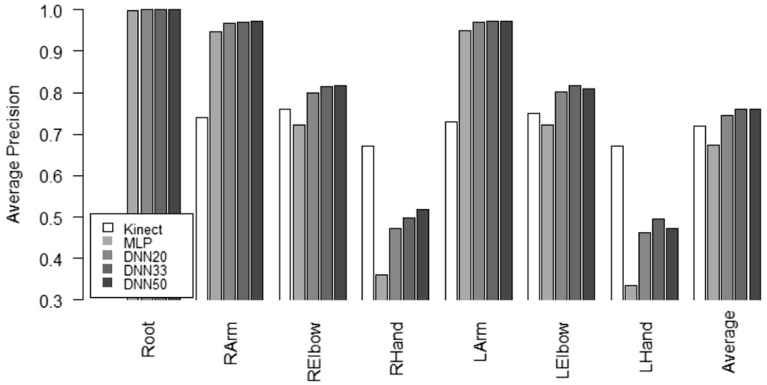

Fig. 9 Comparison of Kinect, MLP, and DNNs average precision for initial training set. Rightmost "Average" was calculated based on hands, elbows, and arms only, as precision for the root was not available for Kinect. Kinect results recreated based on [23]

6 Localization accuracy improvements

The best performing DNN33 network was further trained on the initial training set (Fig. 10a). Once the process was restarted at epoch 2260 the previously small learning rate was reset to 0.05 (training error spike), but the model recovered generalization capability, and the error decreased in more stable manner in consecutive epochs. It can be observed that evaluation error tended to saturate on 0.037 but training error still had decreased. To prevent overfitting the process was stopped at epoch 4000.

Finally, the DNN33 was evaluated on an augmented dataset. The depth images database contained 200,000 poses. In the initial experiment (Section 5) only 120,000 of them were used for training, thus excluding significant number of possible test poses. Therefore, a database augmentation was performed by shifting all 200,000 images one column and one row in random direction, and filling the missing pixels with repetition of their neighbor, and adding Gaussian noise with a mean $\mu = 0$ cm and $\sigma = 0.625$ cm. Augmented images differ enough from the training set, therefore the achieved results were reliable without the risk of exact match with training samples. Comparing results obtained on the augmented set to the initial testing set, the hands localization median error decreased by 1 cm, maximal error decreased by 5 cm (Fig. 10c), and maximal pixel error decreased by almost 2 pixels (Fig. 10d), maximal 3D distance error decreased by 8 cm (Fig. 10b). Average precision for hands was increased by 0.2, and was equal to the one achieved by Kinect (Fig. 11).

7 Comparison with state-of-the-art architectures

It was already stated in Section 2, that numerous architectures were used for localization and detection of human figure in color and depth images. Although our approach differs significantly (in network size and in input resolution), an attempt was made to use those other architectures for the same task of localization and regression and test with the same input data. Following pre-trained architectures were chosen: MobileNet-512 SSD [8], Resnet-18 and Resnet-34 [6], Squeezenet [9], VGG-16 and VGG-16 with reduced number of layers [24].

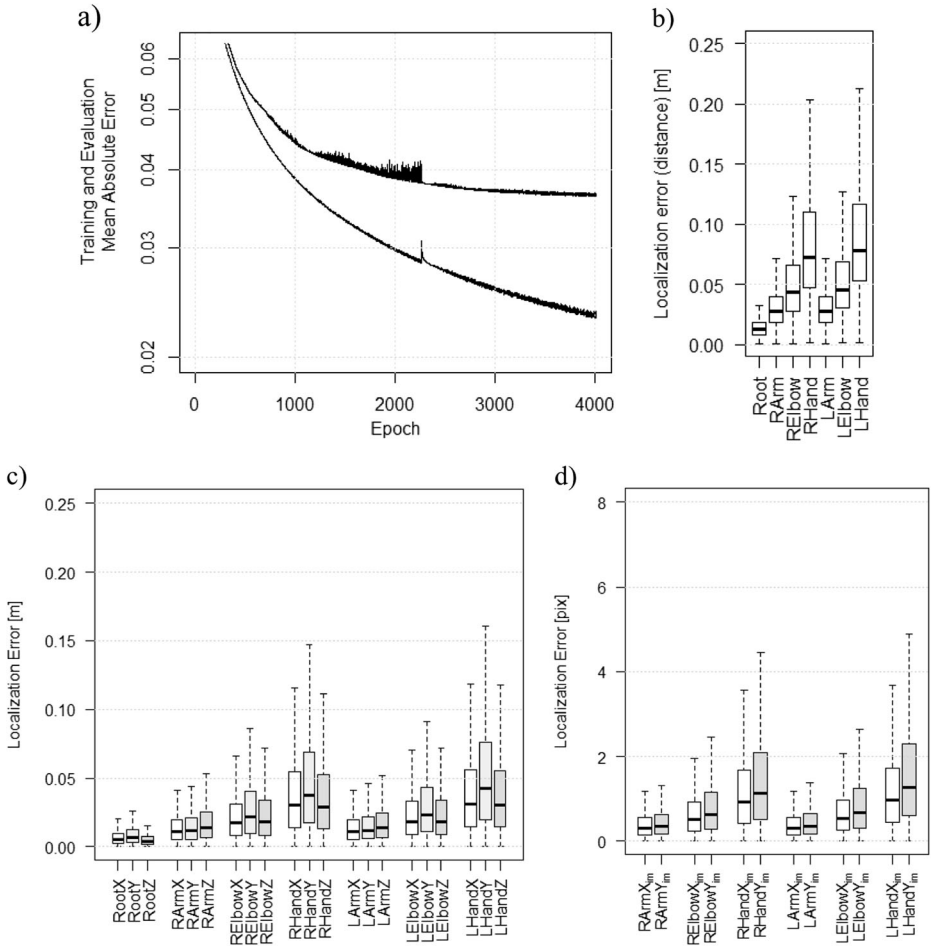

Fig. 10 Results for DNN33 network trained for 4000 epochs and tested on augmented set: **a** training and evaluation errors, **b** 3D space localization error, **c** localization errors along each axis, **d** localization errors in pixels

Other networks were more than 100 times larger than our DNN33, and were omitted as significantly different than the considered convolutional networks.

Two transfer learning approaches were explored:

- **F**: full pre-trained network, i.e. all layers up to the last flattening operation are kept, and one fully connected layer is added at the end, with 33 output regression neurons,
- **S**: pre-trained network is shortened to only first few layers: no more than 2 convolutions and no more than 2 pooling stages are used, and two fully connected layers are added at the end, first with 500 and second one with 33 neurons (following our DNN design).

As a result we can use either all features of the pre-trained network or only the most basic low level features from first layers, and then train last FC layers to exploit the features for localization of body joints.

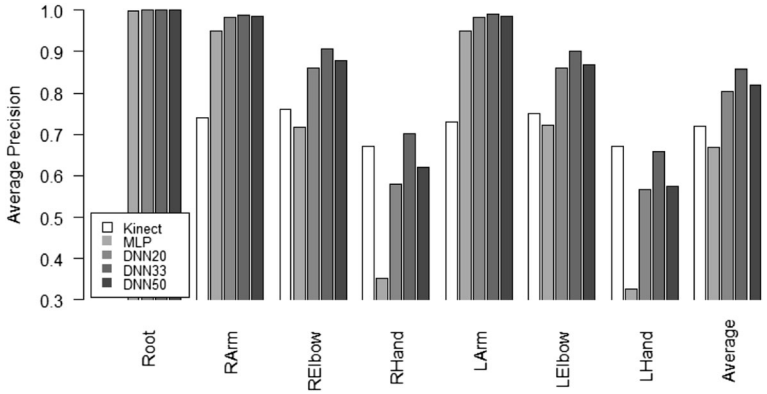

Fig. 11 Comparison of Kinect and DNN average precision for augmented set, with DNN33 trained up to 4000 epochs

Following configurations were used in this transfer learning scenarios:

MobileNet-512 SSD F: flatten4_output + FC33
MobileNet-512 SSD S: conv_2_conv2d_output + FC500 + FC33
Resnet-18 F: flatten0_output + FC33
Resnet-18 S: plus0_output + FC500 + FC33
Resnet-34 F: flatten0_output + FC33
Resnet-34 S: stage1_unit1_relu2_output + FC500 + FC33
Squeezenet v1.1 F: flatten_output + FC33
Squeezenet v1.1 S: fire2_concat_output + FC500 + FC33
VGG-16 F: flatten0_output + FC33
VGG-16 S: pool1_output + FC500 + FC33
VGG-16 reduced F: pool5_output + FC33
VGG-16 reduced S: pool1_output + FC500 + FC33

For each network the same training and evaluation dataset was provided, the training was performed for 2000 epochs, and errors were observed (Table 3).

Squeezenet, VGG-16 reduced, and VGG-16 employ many convolutions and pooling stages, in consequence introducing a significant decrease of the analysis resolution. They are dedicated mainly to image classification, where the resolution loss is not harmful, but rather intended to achieve shifts invariance. Here for the localization task it can be observed that they are not capable to accurately regress on joints coordinates. The low accuracy of full network is improved for VGGs, in scenarios S when only few first layers are used, therefore benefiting from highly detailed representation of the input after only two convolutions.

Residual networks (Resnet-18 and Resnet-34) are designed to preserve information from previous, more detailed layers, therefore allow to take into account low level, local, and high resolution features, resulting in a very high accuracy. Moreover, the generalization gap (difference between training and evaluation results) is the smallest here, indicating high suitability of this architectures for the joint coordinates regression task.

MobileNet was designed as an example of new paradigm of depthwise convolution and trained for a single shot detection. This network is able to extract meaningful and detailed features from the input image, and the achieved accuracy is the highest among tested networks.

Table 3 Comparison of training and evaluation errors after 2000 epochs of transfer learning of state-of-the-art architectures

Networks	Our DNN	Our MLP	Mobile Net SSD		Resnet-18		Resnet-34		Squeeze net		VGG reduced		VGG-16	
			F	S	F	S	F	S	F	S	F	S	F	S
Training error	.025	.040	.017	.019	.015	.060	.013	.048	.146	.149	.224	.037	.185	.039
Evaluation error	**.037**	.058	**.019**	**.028**	**.020**	.060	**.020**	.049	.146	.149	.224	**.038**	.185	**.040**
Relative net size	×1	×1.4	×4.0	×23	×10	×6.1	×19	×6.1	×1.1	×10	×13	×13	×13	×13

Evaluation errors below 0.04 are in bold. Network sizes are expressed as a relative to our DNN33 with 751k weights

The best results were achieved for MobileNet, Resnet-18, Resnet-34, surpassing our proposed architecture, although these networks are significantly larger (from 4 up to 23 times larger than our DNN33), and thus significantly slower. Depending on the implementation, these solutions could be not suitable for real-time operation.

Finally, a comparison of network sizes reveals that the increase of architecture complexity does not guarantee increase of accuracy. Carefully constructed networks achieve high accuracy, when their architectures are aimed at extraction and preservation of local low level features, either by mechanisms of residual blocks, or depthwise convolution, or simply a low number of convolutions as in our case.

8 Conclusions

Proposed neural network architectures proven to be a good tool for multiple regression on body joints, and were capable to model relations between body parts appearance in depth image, to detect and localize joints, and to transform their depth and position in the image into a 3D space coordinates. This satisfactory precision and relatively small errors were achieved despite the low resolution input and resolution loss introduced by the pooling operations. MLP without the convolution and pooling was not able to this precisely grasp non-linear relations between joint locations and depth pixels, achieving lower accuracy. MLP would be still usable in specific scenarios demanding fast processing speed, high accuracy for arms and elbows, and when accuracy for hands slightly lower than Kinect is accepted.

Differences between three DNN architectures and feedforward simple MLP were assessed. The highest accuracy was achieved for root, arms, and elbows, surpassing previous results, and for hands reaching similar precision as Kinect. The method achieved high accuracy, very fast processing speed, and has low requirements on the sensor resolution, as it uses 100 times less pixels than Kinect depth sensor. The method was robust against the sensor noise.

Our method was compared with state-of-the-art architectures, and it was shown that similar accuracy is obtained by transfer learning of pre-trained networks, usually requiring more than 4 times larger and slower architectures.

The method is suitable for estimating body poses for gesture-controlled computer interfaces, e.g. during actions of selecting and manipulating objects in space and onscreen menus by hands movements, similar to many Kinect applications.

The ongoing work focuses on introducing larger variation of body poses, including front and side orientations and legs movement, localization of knees and feet, and on increasing training speed by applying GPGPU platforms.

Acknowledgements This work has been partially supported by Statutory Funds of Electronics, Telecommunications and Informatics Faculty, Gdansk University of Technology.

References

1. Cai Z, Han J, Liu L et al (2017) RGB-D datasets using Microsoft Kinect or similar sensors: a survey. Multimed Tools Appl 76:4313. https://doi.org/10.1007/s11042-016-3374-6
2. Chen T, Li M, Li Y, Lin M, Wang N, Wang M, Xiao T, Xu B, Zhang C, Zhang Z (2015) MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems. arxiv: 1512.01274. Accessed 17 Sept 2018
3. Chen T, Kou Q, He T (2018) MXNet, R package version 1.0. https://mxnet.io, https://github.com/apache/incubator-mxnet/tree/master/R-package. Accessed 17 Sept 2018
4. Crabbe B, Paiement A, Hannuna S, Mirmehdi M (2015) Skeleton-free body pose estimation from depth images for movement analysis. In: IEEE Intl Conf computer vision workshops, pp 312–320. https://doi.org/10.1109/ICCVW.2015.49
5. Ganapathi V, Plagemann C, Koller D, Thrun S (2010) Real time motion capture using a single time-of-flight camera. In: IEEE Conf computer vision and pattern recognition (CVPR), pp 755–762
6. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385
7. Hesse N, Stachowiak G, Breuer T, Arens M (2015) Estimating body pose of infants in depth images using random ferns. In: Proc. IEEE international conference on computer vision workshops, pp 35–43
8. Howard AG, Zhu M, Chen B, Kalenichenko D et al (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, pp 1–9
9. Iandola FN, Moskewicz MW, Ashraf K, Han S, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. arXiv:1602.07360
10. Jung H, Choi M, Jung J, Lee J, Kwon S, Jung WY (2017) ResNet-based vehicle classification and localization in traffic surveillance systems. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 934–940. https://doi.org/10.1109/CVPRW.2017.129
11. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inf Proces Syst 1:1097–1105
12. LeCun Y, Bottou L, Orr GB, Müller KR (2002) Efficient BackProp, neural networks: tricks of the trade. Lect Notes Comput Sci 1524:9–50
13. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444. https://doi.org/10.1038/nature14539
14. Leite DQ, Duarte JC, Neves LP et al (2017) Hand gesture recognition from depth and infrared Kinect data for CAVE applications interaction. Multimed Tools Appl 76:20423. https://doi.org/10.1007/s11042-016-3959-0
15. Li S, Chan AB (2015) 3D human pose estimation from monocular images with deep convolutional neural network. In: Cremers D, Reid I, Saito H, Yang MH (eds) Computer vision – ACCV 2014. ACCV 2014. Lecture notes in computer science, vol 9004. Springer. https://doi.org/10.1007/978-3-319-16808-1_23
16. Ly DL, Saxena A, Lipson H (2012) Co-evolutionary predictors for kinematic pose inference from RGBD images. In: Proc. 14th annual conference on genetic and evolutionary computation (GECCO '12), pp 967–974. https://doi.org/10.1145/2330163.2330297

17. Mahendran S, Ali H, Vidal R (2017) 3D pose regression using convolutional neural networks. In: IEEE conference on computer vision and pattern recognition workshops. https://doi.org/10.1109/CVPRW.2017.73
18. Mahendran S, Ali H, Vidal R (2018) Convolutional networks for object category and 3D pose estimation from 2D images. arXiv:1711.07426
19. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
20. Núñez JC, Cabido R, Montemayor AS et al (2017) Real-time human body tracking based on data fusion from multiple RGB-D sensors. Multimed Tools Appl 76:4249. https://doi.org/10.1007/s11042-016-3759-6
21. Park S, Chang YJ, Jeong H, Lee J-H, Park J-Y (2017) Accurate and efficient 3D human pose estimation algorithm using single depth images for pose analysis in golf. CVPR, IEEE. https://doi.org/10.1109/CVPRW.2017.19
22. R Core Team (2018) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna. https://www.r-project.org. Accessed 17 Sept 2018
23. Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M, Moore R (2013) Real-time human pose recognition in parts from single depth images. Commun ACM 56(1):116–124. https://doi.org/10.1145/2398356.2398381
24. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. ICLR, arXiv:1409.1556v6
25. Sun Y, Wang X, Tang X (2013) Deep convolutional network cascade for facial point detection. In: IEEE conference on computer vision and pattern recognition. https://doi.org/10.1109/CVPR.2013.446
26. Szczuko P (2017) ANN for human pose estimation in low resolution depth images. In: IEEE Conf signal processing: algorithms, architectures, arrangements, and applications (SPA), pp 354–359. https://doi.org/10.23919/SPA.2017.8166892
27. Szczuko P (2018) Very low resolution depth images of 200,000 poses – open repository. https://github.com/szczuko/poses. Accessed 17 Sept 2018
28. Szczuko P (2018) CNN architectures for human pose estimation from a very low resolution depth image. In: 11th international conference on human system interaction (HSI). https://doi.org/10.1109/HSI.2018.8431338
29. Takahashi M, Fujii M, Naemura M et al (2013) Human gesture recognition system for TV viewing using time-of-flight camera. Multimed Tools Appl 62:761. https://doi.org/10.1007/s11042-011-0870-6
30. Togootogtokh E, Shih TK, Kumara W et al (2017) 3D finger tracking and recognition image processing for real-time music playing with depth sensors. Multimed Tools Appl. https://doi.org/10.1007/s11042-017-4784-9
31. Tompson JJ, Jain A, LeCun Y, Bregler C (2014) Joint training of a convolutional network and a graphical model for human pose estimation. In: Advances in neural information processing systems, pp 1799–1807 arXiv:1406.2984
32. Toshev A, Szegedy C (2014) Deeppose: human pose estimation via deep neural networks. In: Proc. IEEE conference on computer vision and pattern recognition, pp 1653–1660, arXiv:1312.4659. https://doi.org/10.1109/CVPR.2014.214
33. Wang K, Zhai S, Cheng H, Liang X, Lin L (2016) Human pose estimation from depth images via inference embedded multi-task learning. In: ACM on multimedia conference (MM '16), pp 1227–1236. https://doi.org/10.1145/2964284.2964322
34. Ye M, Wang X, Yang R, Ren L, Pollefeys M (2011) Accurate 3D pose estimation from a single depth image. In: Computer vision (ICCV), IEEE international conference, pp 731–738

Piotr Szczuko received the M.S. degree in telecommunication in 2000, and Ph.D degree in 2008 from the Gdansk University of Technology. Since 2008 he has been an Assistant Professor in the Multimedia Systems Department of Gdansk University of Technology. His research focuses on applications of classification methods in multimedia and human-computer interaction applications, especially in depth images and video. He is also interested in: machine vision, artificial intelligence and automatic inference methods, classification and perception of acoustic and visual signals. He is an author of over 80 conference papers and journal articles. Dr. Szczuko was a recipient of Best Paper Awards of IEEE SPA 2015, MCSS 2014, and MCSS 2011 and Prize of Prime Minister of Poland for his PhD dissertation in 2009.