3D Human Pose Estimation - Project Report

Omar Elkhatib
eomar@student.ethz.ch

Tim Franzmeyer
frtim@student.ethz.ch

1 INTRODUCTION

3D human pose estimation lies at the heart of current deep learning research and has a wide field of application, such as human computer interfaces or pedestrian detection for autonomous cars. In this project we have employed a network that predicts the positions of the human's 17 joints, given an RGB image.

2 RELATED WORK

3D Human Pose Estimation has been particularly challenging due to the unavailability of large datasets with 3D labeling of joints. Only with the recent introduction of datasets such as MPII [1] and Human3.6M [3] has research accelerated significantly. Data acquisition in lab environments still represents a main drawback, as almost no labeled "in-the-wild" data is available.

So far, there have been two popular approaches for this task. The first attempts to build upon the well-studied work on 2D pose estimation, adding a subsequent 3D joint estimation. Another approach is to directly regress 3D poses from image data, without an intermediate step. Additionally, there are approaches which employ a hybrid solution as in [7].

In the work of Martinez et al. [6], an out-of-the-box 2D joint detector [8] is used to obtain 2D poses, which are then used to directly regress 3D joint coordinates. They show that with recent advancements in deep learning, it is possible to achieve state-of-the-art 3D pose estimation performance from 2D poses, using a simple fully-connected network architecture. Pavlakos et al. [9] directly estimate 3D poses, introducing a volumetric heatmap approach, where the individual joints are represented with volumetric heatmaps and optimized with a coarse-to-fine approach.

Other research explores different possibilities of pose representation, such as [12], which models the human skeleton with bone lengths instead of joint coordinates, explicitly enforcing constraints on human geometry.

More recent approaches attempt to predict high-fidelity models of 3D pose, which also capture the human 3D shape. Kanazawa et al. [4] estimate human shape and pose from in-the-wild images with only 2D ground-truth labels by adopting an adversarial network that discriminates realistic 3D human meshes. Alternatively, Pavlakos et al. [10] estimate human shape and pose in an end-to-end fashion using a parametric statistical shape model (SMPL).

3 APPROACH

In this section, the approaches investigated within this project will be presented and details about the individual subsystems are supplemented. Subsequently, the training procedure and the results are explained and discussed.

After briefly experimenting with different ResNet architectures to directly regress the 3D pose only showed marginal improvements, we decided to adopt similar approaches to the state-of-the-art models in literature. The work of [6] was especially promising due to the impressive performance, despite the simplicity of their architecture.

As such, we adopt the approach of first predicting 2D joint locations in an image and afterwards lifting the 2D joint predictions to 3D, resulting in two subsystems. In the first, 2D joint positions are predicted using a modified ResNet50. The output of the first network is then fed into a second network which regresses the 3D joint positions.

In this project, two general methods to combine the two subsystems were investigated. The first option is to first train both subsystems separately and subsequently predict on both models. Alternatively, the subsystems are trained jointly with a multi-objective loss function. The two subsystems will be presented in more detail in the next section, followed by an evaluation of the different training scenarios.

3.1 2D Pose Estimation with Heatmaps

3.1.1 Architecture. After noting that several publications use a modified version of ResNet50 [2] with great success, we decided to use a related approach for the first building block of our network. A similar network to the one proposed by [7] was used to estimate 2D joint positions from single input images. The original network consists of the following building blocks displayed in Figure 1, assuming an input size of 256, although this can be chosen arbitrarily: Block A simply is the ResNet50 model introduced by [2] until block 4f, using a stride of two in the first layers of blocks two, three and four. This results in an output size of [1024, 16, 16][1]. Block B represents a structure similar to a ResNet bottleneck block. Block C is another convolutional block which does not change the output size. In Block D, the tensor is scaled up to its final output width and height, further referred to as the heatmap size (HMSize). For the given input and architecture, this results in an HMSize of 64. This is done using a deconvolution with kernel size four and stride four. Block E produces the final output size, given by the number of joints and HMSize.

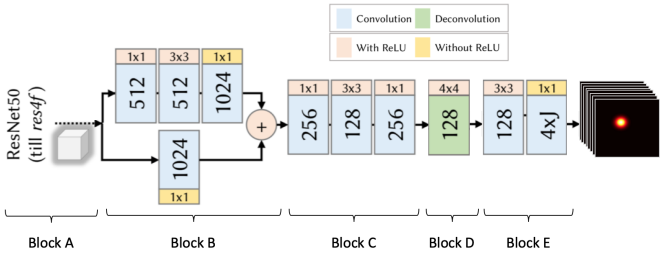

Figure 1: Network Architecture used for Heatmap Predictions of 2D Joint Positions

[1] [channels, height, width]

3.1.2 Groundtruth Representation. The groundtruth heatmaps are calculated given the 2D joint positions of the training data. For each joint, a heatmap of size HMSize by HMSize is created. A 2D, symmetric Gaussian is evaluated for each cell, with the maximum at the joint's pixel position. The heatmap is normalized to have a maximum value of 255, avoiding numerical instabilities.

Figure 2: Heatmap example will all joints overlayed - left: groundtruth, mid: predicted heatmap, right: groundtruth in green, predicted in red, yellow if equal

3.1.3 Loss Function. The predicted joint positions are computed with the argmax over the heatmap axes (height, width). As this operation is not differentiable, the L2 loss of the predicted and the groundtruth heatmaps is used as a surrogate loss for training purposes.

3.1.4 Model Variations - fixed HMSize.

- Ratio of HMSize and covariance: Determines the size of the gaussian bump. Tested the values: 3200, 64, 12.8 (best) and 6.4.
- Batchnorm implementation: Different implementations in Blocks B-E, while Block A (ResNet Block) stays unchanged. The three variations tested are batch normalization after each layer, a single batch normalization layer after each block, or no batch normalization at all (best).
- Model Simplification: Skip Block B to achieve a simpler architecture. Result: Performance better if Block B was included.
- Different kernel sizes of the deconvolutional kernel used in Block D: Tested 3, 4 (best, displayed in Figure 2) and 6.
- Subpixel accuracy refinement: Assuming a gaussian curve, a refinement of the heatmaps argmax was computed. Surprisingly, this did not improve model performance.

3.1.5 Model Variations - different HMSize.

- Multiple deconvolutional layers in Block D:
  a) 2 layers, stride (4, 2) - results in HMSize 128
  b) 2 layers, stride (2,2) - results in HMSize 64
  c) 3 layers, stride (2,2,2) - results in HMSize 128
- Stride of Resnet Layer 4 in Block A set to one - results in HMSize 128

A HMSize of 128 results in a slight improvement of model performance, but at the same time in a significant increase of memory consumption. As such, the batch size has to be reduced and computation time increases. In order to trade off performance and faster learning, an HMSize of 64 is used.

3.2 2D to 3D Regression

For the regression of 3D pose from 2D predictions, the simple network architecture of [6] illustrated in Figure 3 is employed. To integrate this model with the project skeleton code, we re-implemented the author's code [2] in high-level tensorflow. Furthermore, we investigated the effect of different components such as the loss function or max norm regularization.

The L1 and L2 loss functions were compared when training with 2D ground truth joint locations. As the L1-loss resulted in slower training and slightly worse performance, we opted for the L2 loss in all subsequent training.

In our experiments, the use of max norm regularization consistently introduced strong instabilities to the learning procedure, where the training loss would oscillate heavily even with small learning rates (0.0001). Increasing the max norm from the recommended value of one to a value of four improves the situation, but with significantly worse performance than the model without max norm. This result was unexpected because the combination of dropout and max-norm regularization was recommended by [6] as well as other literature [11].

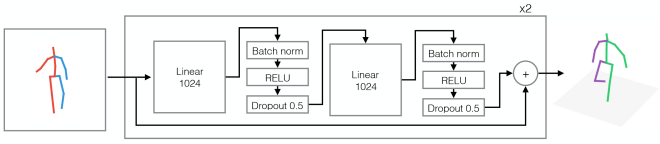

Figure 3: Simple fully-connected Network Architecture for 2D-to-3D Pose Regression

3.3 Training Scenarios

In the following part, the different approaches to combine and train the aforementioned subsystems will be introduced and elaborated.

3.3.1 Separate Learning. Due to the availability of 2D and 3D pose ground-truth, it is possible to train the two model subsystems separately with the different ground-truth labels. The two model subsystems are trained independently with their respective loss functions and are then combined during the prediction phase.

The first subsystem uses the above explained heatmap approach to predict joint locations. The 2D-to-3D regression module can either use the ground-truth 2D poses or the heatmap predictions as an input. Depending on the input used, the second subsystem can be trained in parallel with the first model.

3.3.2 Multi-task Learning. Another approach adopted within this project was combining both models into a single architecture to be trained end-to-end. In order to utilize the full extent of the training data, a multi-objective learning problem was set up that uses ground truth 2D pose for intermediate supervision of heatmaps and ground truth 3D pose for the final 3D pose loss.

As the output of the first subsystem is a set of heatmaps, these are converted to 2D joint positions in the 64x64 heatmaps. The result is then upscaled to the original image resolution of 256x256.

[2] *https://github.com/una-dinosauria/3d-pose-baseline*

Lastly, the 2D pose predictions enter the second subsystem that regresses 3D pose from 2D joint positions.

The two individual loss functions are added together to form the multi-objective loss (1), which weights the 2D and 3D losses relative to each other with the hyperparameter λ.

$$L_{tot} = L_{3d} + \lambda \cdot L_{2d} \tag{1}$$

3.4 Training Details

In order to quantify the performance of the model under different hyperparameters and architectural changes, the provided dataset was split into training and validation sets. To meaningfully split the reduced Human3.6M dataset, we adopted the same approach as related work and split the set across the subjects. For all experiments, all actions for subject S1 were used for the validation set, which represents approximately 15% of the dataset.

The 2D pose is normalized as it enters the regression model and the predicted 3D pose is later denormalized. The Adam optimizer [5] has been used for all training runs, as it showed best performance in initial trials. Weights of the 2D-to-3D regression model are initialized with Kaiming initialization [2], whereas the ResNet heatmap architecture uses a variance scaling initializer.

When performing separate learning, the L2 distance of predicted and ground truth 2D points was used as the performance metric for the heatmap network and the mean per joint position error (MPJPE) score for the regression model. During multi-task learning, solely the MPJPE value was of importance.

3.4.1 Hyperparameters. Table 1 shows the hyperparameters considered for model selection and their respective final values. To obtain these values, we used grid search for the learning rate and the loss weighting parameter λ. All other parameters were validated with separate learning and carried over to the multi-task learning.

Table 1: Final Hyperparameter Values used for Submission

Hyperparameter	Final Value
Learning Rate	0.001 / 0.0001 (Scheduling)
LR Decay Rate	Disabled
Loss weighting λ	0.01
Dropout	0.5
Max-Norm	Disabled
Heatmap Parameters	as in 3.1.4/3.1.5

3.4.2 Learning Rate Scheduling. For the training of the networks, two approaches with regards to learning rate scheduling have been experimented with. The first reduces the learning rate step-wise once the validation loss flattens out. The second approach uses exponential learning rate decay to continuously decrease the learning rate throughout training.

For the step-wise learning rate decrease, we chose a starting learning rate of 0.001. The network takes around 15 hours for the validation loss to flatten out, after which the network is fine-tuned with a learning rate of 0.0001 for six hours.

Experimenting with exponential learning rate decay, values were chosen such that the learning rate would decay from the original 0.001 value to 0.0001 in around the same time the validation loss previously flattened out[3].

4 RESULTS

To compare the different models and training combinations, we focused on the MPJPE score as the primary metric. For the separate learning case, the MPJPE could only be computed for the second subsystem. Therefore, to gauge the performance of the heatmap subsystem, we first qualitatively evaluate the heatmaps then feed the predictions into the trained regression model to calculate an indirect MPJPE score. For the best combination of separately trained subsystems, we achieved a score of 141.97 mm on the public test set. For this result, the regression model was trained using ground-truth 2D pose with a MPJPE of 56.23 mm. All attempts to train this model with heatmap predicted 2D poses led to strong instability and unsatisfactory performance.

Therefore, the next focus was on training both systems together in the multi-task learning setting. Using the hyperparameters listed in Table 1 and the step-wise learning rate scheduling detailed in 3.4.2, our best model achieved a MPJPE of 76.89 mm on the entire validation set as well as a submission score of 100.52 mm. These results show that the multi-objective formulation improves the performance of the overall pipeline, when compared to training both models separately and predicting final 3D poses sequentially.

5 CONCLUSION

This work shows that multi-task learning is a viable method for predicting 3D human pose from single RGB images. After trying several training scenarios with the same architecture we noticed minimal performance gains, leading us to believe that architectural changes are required in order to improve overall performance.

The heatmap subsystem, in particular, seems to be the bottleneck of the performance, since the 2D-to-3D regression model trained with ground truth 2D poses can achieve substantially lower MPJPE scores of $\sim$ 56 mm. Furthermore, training the regression subsystem with 2D predictions has been unviable due to training instability. This leads us to think that a better heatmap prediction subsystem is necessary to boost performance further. One possibility for this would be the stacked hourglass model, which was employed by [6] as a 2D joint predictor and has shown significantly better performance.

Moreover, we would have liked to implement data augmentation to reduce overfitting, but it became evident how untrivial this is for the combined model. Without sufficient knowledge about the coordinate frame in which the 3D poses are represented, it was not possible to preserve equivariance once the input images are transformed. Hence we were able to implement data augmentation, e.g. flipping the image, for the heatmap network when using separate learning, though we were unable to implement it for multi-task learning. Since multi-task learning gave significantly better performance than separate learning, we decided to focus on other possibilities for improvement instead of data augmentation.

[3] With a batch size of 32, the 10000 steps represent a single epoch of training time. The learning rate decays to approx. 1/10th the original value after 4 epochs with decay rate 0.6 or 10 epochs with decay rate 0.8

REFERENCES

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2014. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. 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 (jul 2014), 1325–1339.

[4] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. 2018. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7122–7131.

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

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

[7] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 44.

[8] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*. Springer, 483–499.

[9] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. 2017. Coarse-to-fine volumetric prediction for single-image 3D human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7025–7034.

[10] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. 2018. Learning to estimate 3D human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 459–468.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[12] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. 2017. Compositional human pose regression. In *Proceedings of the IEEE International Conference on Computer Vision*. 2602–2611.