3D Human Pose Estimation

Xinyuan Huang
huangxin@student.ethz.ch

Yufeng Zheng
zhengyuf@student.ethz.ch

ABSTRACT

Human 3D pose estimation is a crucial task in computer vision. It plays a key role in the field of human computer interaction, smart-city, robotics, etc. In this project, we estimated human 3D pose in 2 stages: from image to 2d joints, and from 2d joints to 3d joints. It successfully outperformed the baselines, and it only needs 4 to 5 hours of training on 1 single gpu.

1 INTRODUCTION

The accuracy of 3D human pose estimation has improved substantially in the recent years thanks to the development of deep convolutional neural networks. However, 3D pose estimation networks are still unable to yield as good results as 2D estimations. Despite the intrinsic ambiguity introduced by the extra dimension, is it possible that 3D pose estimating networks are failing to obtain 2D information due to some other reasons?
In the work of previous researchers, 2D pose estimation is mainly done by detection based methods instead of regression based ones. In detection based method, a Gaussian heat map is generated for each joint and the maximum is selected as the joint position. In comparison, regression based methods directly estimate the joint positions. Despite being computationally easier, regression based models might not exploit the structural connections between joints well enough. Therefore, Detection based methods sets most benchmarks for 2D pose estimation problem.
However useful they are in 2D problems, detection based methods are too computational expensive in 3D context. In the hope of developing a 3d pose network both accurate and computationally feasible, we decide to make full utilization of accurate 2D detection based models and regress our 3D joints based on 2D positions instead of directly on the image.

2 OUR APPROACH

We adopted a two staged approach for the 3d human pose estimation problem. In the first step, we detect 2d joint positions from images. In the second steps, we estimate joint positions in 3 dimensional space from 2d inputs. To detect 2d joint positions from images, we trained a two staged stacked hourglass model [4] from scratch. For lifting joints from 2d to 3d, we adopted the linear model from [3]. We will further elaborate our network in the following sections.

2.1 Data Augmentation

The training set contains about 310k image-pose pairs. From our experiments, we find that our stacked hourglass model overfits severely on the training set, which makes the training in the next stage very difficult. To increase the robustness of the stacked hourglass model, we use 5 classes of data augmentation.

2.1.1 flip. The images are randomly flipped. Since the training and testing dataset contains no upside down samples, we only flip horizontally.

2.1.2 shift. The images are shifted horizontally and vertically. Since the pictures in human36m dataset is a bounding box for subjects located in the middle of pictures, so we decide to shift horizontally or vertically according to the sizes of horizontal padding space and vertical padding space.

2.1.3 rotation. The images are rotated for a random degree within a certain range.

2.1.4 n. We randomly choose a patch of the image and set it to be black or white. We choose the patch width and height and patch location randomly.

2.1.5 scale. We randomly scale the image by a random factor.

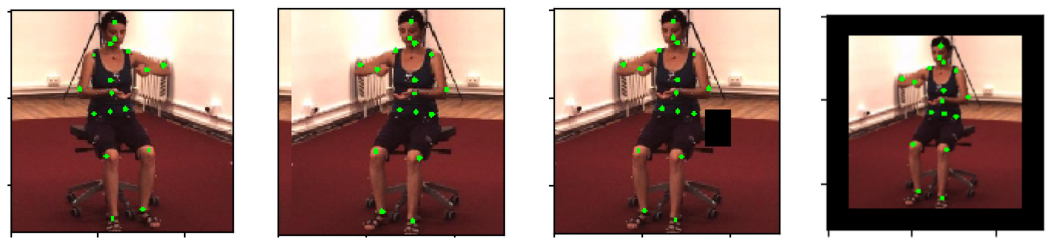
2.2 Stacked Hourglass model for 2d Detections

We used the state-of-the-art stacked hourglass network of Newell et al. [4] to detect 2d joints' pixel positions from input image. As shown in Figure 2, multiple hourglass modules are stacked in a pipeline in the stacked hourglass network. Every single hourglass modules process information in a bottom-up, top-down manner. This repeated bottom-up, top-down processing enables the network processing features across all scales and capturing the various spatial relationships associated with the body.
Our implementation of Stacked Hourglass model is based on the tensorflow implementation from [2]. We trained a network consists of two stacked hourglass modules. For every joint, stacked hourglass model generates a heatmap for it which represents the position probability of the joint, and we used the position with the biggest probability value as our predicted joint position.
At first we used pictures of subjects S1,S5,S6,S7 as our training set, and used pictures of subject S8 as our validation set which works as an indicator for model selection. Since the settings in the dataset are not complex, we used five data augmentation strategies, flip, shift, rotation, occlusion and scaling, to avoid over fitting.
We found that data augmentation makes our dataset complex enough that the overfitting problem of the stacked hourglass model is not a big issue. Therefore we decided to include S8 into the training set in order to train our model with the most diversity of subjects.

2.3 Linear Model

The model we used is shown in Figure 3. Apart from the structure shown in the diagram, we also used a linear layer to map our input to 1024 dimension, and another linear layer to map the output to be of dimensionality 16*3. The input is a vector of 16 joint positions in 2d, which is predicted by our stacked hourglass network. We subtracted the root from the joints and applied data normalization. The outputs are 3 dimensional joint positions processed in the same way. We used L2 loss of joint position as our loss function. The optimal model is represented by Equation (1)

$$f^* = \min_f \sum_{i=0}^{N} (f(x) - y)^2 \tag{1}$$



(a) Flip **(b) Shift** **(c) Occlusion** **(d) Scale**

Figure 1: Caption place holder

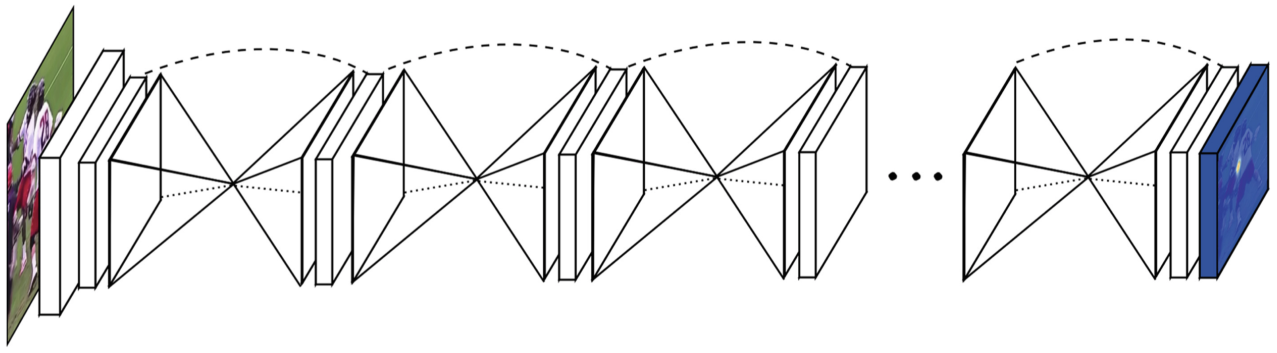


Figure 2: Architecture of stacked hourglass model

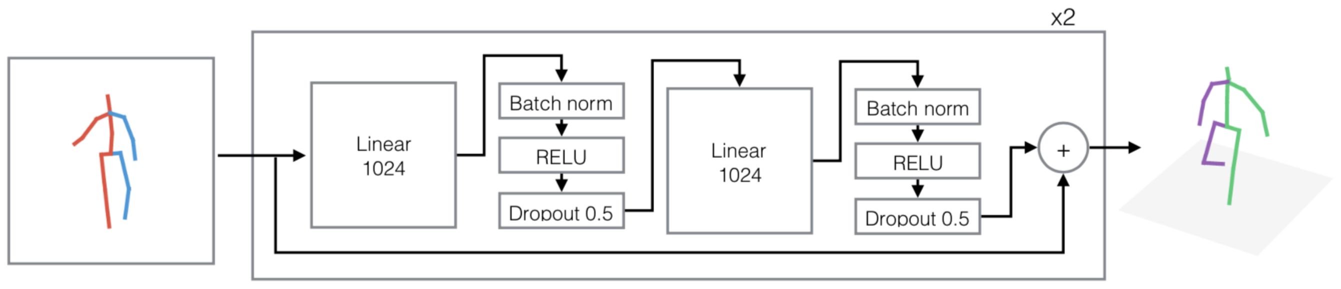


Figure 3: Diagram for our 2d to 3d Linear Model

Since we used noisy input from stacked hourglass detection, the linear model severely overfits on the training set. To make the model more robust, we added batch normalization, dropout and we also utilized norm clipping which force the back propagated gradients to be less or equal than 1. Our implementation of linear model is based on pytorch implementation from [1].

2.4 Loss Function

For training linear model, apart from the normal L2 loss of joint positions, we also experiment with the compositional loss function proposed in [5] and weighted L2 loss of joint positions.
For a variable *var*, it is normalized as:

$$\hat{var} = \frac{var - mean(var)}{std(var)} \tag{2}$$

We use $\mathbf{J} = [J_1, J_2, ..., J_{17}]$ to denote the predicted 3d joint positions, and use $\mathbf{J}^{gt} = [J_1^{gt}, J_2^{gt}, ..., J_{17}^{gt}]$ to denote the ground truth 3d joint positions.
The normal L2 loss of joint positions is

$$L_{2,normal} = ||\hat{\mathbf{J}} - \hat{\mathbf{J}}^{gt}||_2^2 \tag{3}$$

2.4.1 Compositional loss function. The compositional loss function utilizes the relative positions between different joints.
The relative positions between different joints are

$$\Delta\hat{J}_{u,v} = \hat{J}_u - \hat{J}_v, \forall 1 \le u < v \le 17 \tag{4}$$

$$\Delta\hat{J}_{u,v}{}^{gt} = \hat{J}_u{}^{gt} - \hat{J}_v{}^{gt}, \forall 1 \le u < v \le 17 \tag{5}$$

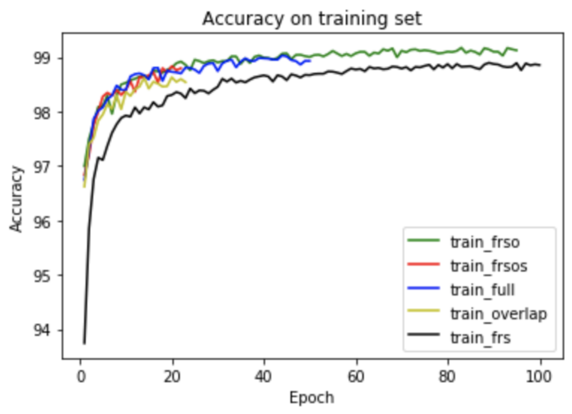


Figure 4: Accuracy on training set for stacked hourglass model. Frs: data is augmented by one random strategy in flip, rotation and shift. Frso: data is augmented by one random strategy in flip, rotation, shift and occlusion. Frsos: data is augmented by one random strategy in all five strategies. Full: data is augmented by one random strategy selected from all five strategies, and all available subjects are used as training set. Overlap: data is augmented randomly by all five strategies. Unlike other experiments, each sample can be processed by multiple strategies.

The compositional loss function is defined as

$$L(P) = \sum_{(u,v)\in P} ||\Delta\hat{J}_{u,v} - \Delta\hat{J}_{u,v}{}^{gt}||_1,\ P = \{(u,v)|1 \le u < v \le 17\} \tag{6}$$

In our experiment, the best accuracy of linear model doesn't have noticeable improvement when using compositional loss function, but given the same accuracy obtained, the usage of compositional loss function shortened the training time.

2.4.2 Weighted L2 loss of joint positions. During the training, we found that stacked hourglass models achieve lower accuracy for wrist, elbow, and foot compare to other joints. In order to compensate for such difference, we introduced weighted L2 loss of joint positions in the following 2d to 3d model. Our weighted L2 loss of joint positions is

$$L_{2,weighted} = ||(\hat{\mathbf{J}} - \hat{\mathbf{J}}^{gt}) \cdot \mathbf{w}||_2^2,\ \mathbf{w} \in R^{17} \tag{7}$$

We set $\mathbf{w}$ to make linear model put more emphasis on wrist, elbow and foot. The introduction of weighted loss improved our mpjpe score by 3.

3 RESULTS

Our best model achieves an mpjpe score of 101.9 on submission system.

3.1 Stacked hourglass models

To train stacked hourglass models, we tried multiple augmentation settings. The training accuracy and validation accuracy are shown in Figure 4 and Figure 5. Due to the limit of computational resources and time, we couldn't train all the models for enough epochs. But one can see from Figure 5 that the addition of 'occlusion' improved the accuracy by about 0.8 percent.

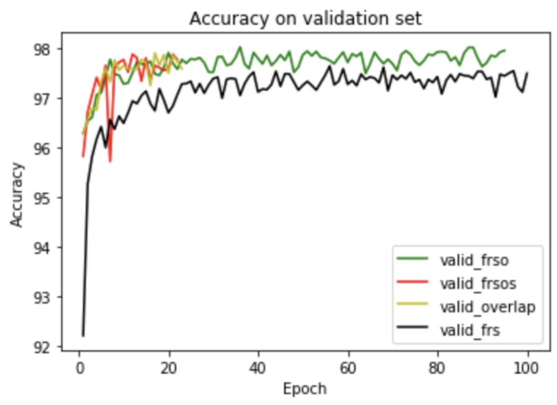


Figure 5: Accuracy on validation set(S8) for stacked hourglass model.

Our final stacked hourglass model with four stacked hourglass modules is trained on S1,S5,S6,S7 for 15 epochs, with S8 as validation set. The data augmentation methods we used include. Every epoch consists of 32000 pictures. For each image, we randomly decide whether to augment it or not. If a picture is to be augmented, we randomly choose a strategy.

3.2 Linear model

We experiment with our linear model with S8 as validation. But the final linear model is trained on all samples of S1,S5,S6,S7,S8. The input is obtained from our trained stacked hourglass model. The labels are the ground truth 3d joint positions. We trained linear model for 80 epochs. Each epoch is the whole dataset.

3.3 Experiment details

We first tried to train stacked hourglass model with 2 stack modules, and all the data we used for the images in this report is based on 2 stacked model. However, 3 hours before the submission deadline, we decided to submit the result from a model with 4 stack modules, with no scaling and no shifting. It also has lower learning rate and faster learning rate decay for the 2d to 3d model. It turned out that it achieved much better result than others. Please excuse us that we didn't experiment with parameters further based on our best model. Also regarding reproducing the result, please consult README file in GitLab! The code for reproducing is on the branch 'final'.

REFERENCES

[1] [n. d.]. A simple baseline for 3d human pose estimation in PyTorch. https://github.com/weigq/3d_pose_baseline_pytorch
[2] Walid Benbihi. [n. d.]. Tensorflow implementation of Stacked Hourglass Networks for Human Pose Estimation. https://github.com/wbenbihi/hourglasstensorlfow
[3] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. 2017. A Simple yet Effective Baseline for 3D Human Pose Estimation. In *The IEEE International Conference on Computer Vision (ICCV)*.
[4] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision – ECCV 2016*.

[5] 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.