3D Hand Pose Estimation - Report Project 1

Lea Reichardt
relea@student.ethz.ch

Amro Abdrabo
aabdrabo@student.ethz.ch

Dominic Weibel
doweibel@student.ethz.ch

ABSTRACT

In this work, we present an approach to automatically detect hand pose and shape from RGB images with the help of a neural network. We enrich the provided FreiHand dataset [14] with online data augmentation and regress our model directly to the 21 3D joint positions. Using ResNet101, we achieve a score of 9.91 calculated with the procrustes-aligned mean per joint.

1 INTRODUCTION

Up to 70% of our communication is nonverbal with the hands playing a major role. Understanding hand gestures is not only important to understand sign language, but also in understanding the context, mood, or intentions of a person. With this work, we present an approach to automatically detect hand-joint positions from RGB images with the help of a neural network.

Challenges. Two major challenges are inherent to this task: Firstly, even in a lab setting, the picture of a hand is partially occluded. One never sees the complete hand - interacting with the world by grabbing or holding objects further makes the observable portion of the hand smaller. Secondly, hands are small and the provided FreiHand dataset [14] images are of low resolution as well. This leads to the effective information in a picture about the hand itself being relatively small.
To address these challenges, we try out different networks and combine them with data augmentation techniques.

Related work. A widely used network architectures for pose estimation is the encoder and decoder network [13], [8]. The main idea is for an encoder to produce a robust, disentangled, and informative latent vector representation of the hand. Romero et al. [11] has demonstrated that such a latent vector can be represented using two vectorial parameters, the shape, denoted by $\beta \in \mathbb{R}^{10}$, and pose, denoted as $\theta \in \mathbb{R}^{21\times3}$. They also created the MANO hand model [11] that provides us a parameterization of 3D hands using only few parameters, and functionality to visualize the 3D model.

Iqbal et al. [7] propose to first use an encoder-decoder network to regress to 2D heatmaps for the keypoints and then lift the keypoints to the third dimension via a CNN. Kulon et al. [8] train an encoder ResNet50 to represent the hand in a latent vector, which is then fed into a spatial mesh convolutional decoder to reconstruct the 3D hand model. Boukhayma et al. [4] use a modified ResNet34 with optional heatmaps as inputs to train an encoder for predicting the MANO hand pose and shape parameters and additionally the camera intrinsics. This information is used to generate the 3D hand model via the aforementioned MANO model [3, 4, 6].

More recently, extensive research has increased the complexity of the encoder where now multiple encoders are pipelined, or chained, into each other such as Lin et al. [9], which relies on the recent advent of transformers for use as the encoders.

Method. We enrich the provided FreiHand dataset [14] with online data augmentation such as background change, rotation and synthetic clutter and train a ResNet101 to directly predict the 3D joint locations. We directly regress on these 3D coordinates. By training this network for 200 epochs, we achieve a public score of 9.91.

2 METHOD

In this paper, we tried two main approaches, one with multiple chained transformers as encoders, and the other simpler variant with a ResNet as an encoder. A fundamental limitation of state-of-the-art approaches is the requirement of extra representations such as heat maps, location maps, or delta maps which encode the pixel's proximity to the joint's center of mass. Problematic is the increased use of memory for training as well as the sparse availability of these.

Even more recently, the limitation is both time and memory, as the encoders now possess more parameters to be trained, hence more memory and a longer expected duration to convergence.

2.1 Data Augmentation

In order to enlarge our dataset, reduce overfitting and tackle the aforementioned challenges, we provide several online data augmentations:
Background. We compute a mask that separates the background from the hand and replaces the background with a randomly selected image from datasets containing various images [1, 2, 5].
Rotation. We rotate the image and the keypoints by a randomly chosen angle in the range $[-180, 180]$.
Flipping. With a $\frac{1}{2}$ chance, we flip the image and the keypoints around the y-axis to ensure the training data contains a similar amount of left- and right-hand images.
Occlusion. Since occlusion is one of the biggest challenges of hand pose estimation, we add randomly generated shapes to the image to simulate occlusion.
Figure 1 shows the intermediate results of our online data augmentation pipeline on an example training image.

2.2 Model and Training

Since both our approaches resulted in similar scores, we explain both architectures.

Pipelining ResNets. Pipelining multiple encoders promised good results. The basic architecture can be seen in figure 2. After augmenting the input, we use a dimensionality reduction architecture of three ResNets to reduce after each layer the dimensions of the embedding. As already mentioned, our limitations are both time and memory, thats why we choose as the first two layers ResNet50 to catch as much image features as possible, followed by a shallower ResNet34 that just outputs the xyz coordinates for the 21 joints. The layers are trained on one loss, which is backpropagated through all

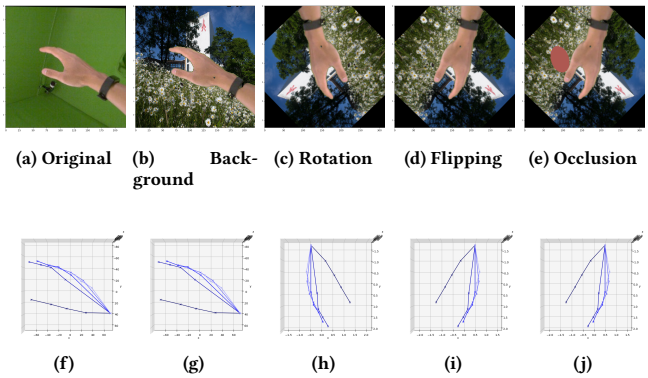


Figure 1: (a)-(e) shows the augmented images in the order they are applied in our pipeline. (f)-(g) shows the corresponding keypoint positions.

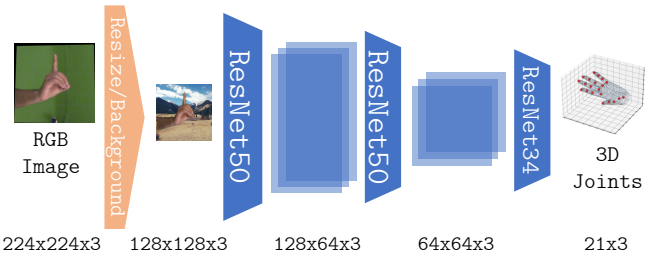


Figure 2: Pipelined ResNet architecture

three layers and minimises the distance between ground truth and predicted joint coordinates:

$$L = ||\hat{x}_{3D} - x_{3D}||_1 \quad (1)$$

Single ResNet. Our second approach was to use a single ResNet to generate either hand parameters (β, θ) or directly the 3D coordinates of the 21 joints locations. In order to reduce memory consumption and speed up training, we downsize our input to images of 128x128 pixels. To make the network more robust to overfitting, we used all data augmentations. We try out different ResNet architectures with the final, fully connected layer adjusted to predict directly the 3D coordinates of the 21 joints - the output of our network is therefore $21 \times 3 = 63$. To minimise the distance between ground truth and predicted 3D keypoints, we again use a l1 loss (see equation 1).
Starting with ResNet34, we already achieve moderate predictions. Increasing the depth using ResNet101 lead to a even better score. Table 1 shows a comparison of the two architectures, using data augmentation. ResNet34's loss reached a plateau after 150 epochs, in contrary to ResNet101.

Optimizer. We used the adam optimizer since its less sensitive to wrong order of learning rate.

Architecture	ResNet34	ResNet101	
Epochs	150	150	200
Perf metric	13.83	11.42	9.91

Table 1: Performance of different ResNets

Method Name	Loss	Perf Metric (val)	Perf Metric (test)
1: Res34 cos. anneal. lr (150 ep.)	0.446	14.32	n/s
2: Res34 const lr (175 ep.)	0.462	14.09	n/s
3: Res34 late cos. anneal. lr (200 ep.)	0.198	12.43	13.83
4: 2xRes50 + Res34 (240 ep.)	0.180	n/s	13.14
4: Res101 (200 ep.)	0.098	n/s	9.91

Table 2: Performance of our most promising experiments

2.3 Postprocessing

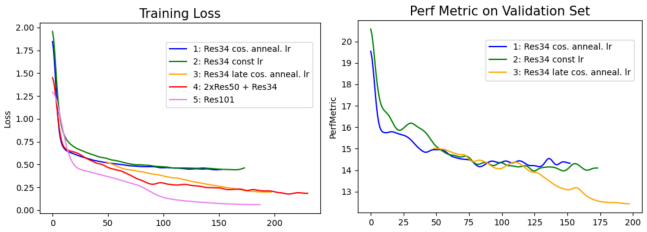
Since there are different orderings of joints for the FreiHand dataset, one challenge was to make sure the ordering is correct for each stage. The convention used in this paper is the AIT ordering, which is different from the ordering of MANO and FreiHand. Although order preprocessing does not present a limitation to static image mesh reconstruction, it can lead, for high frame rates, to the appearance of notable lags for real time hand shape and motion capture algorithms.

3 EVALUATION

In this section we show the results of our final model, as well as an overview of the performance of our most promising runs. The submission score metric, which we refer to as Perf Metric, is the procrustes-aligned mean per joint euclidean distance (PA-MSE). Furthermore, we define the loss used in these experiments in equation 1. Figure 3 shows the loss and Perf metric of our experiments during the training phase. Run (1) through (3) are all performed with the ResNet34 model, but with different approaches for the learning rate scheduling: (1) uses the cosine annealing lr scheduler provided by pytorch, whereas (2) uses the constant $lr = 1e-04$. (3) is pretrained for 50 epochs using $lr = 1e-04$, and trained another 150 epochs using the cosine annealing lr scheduler. Run (4) shows the training loss of the pipelined ResNet model, and (5) shows the loss of the ResNet101 model that produced our final score. In table 2 we present the final loss and Perf Metric of the above mentioned experiments.

4 DISCUSSION

Altough pipelining three ResNets seemed promising, it didn't give the expected results. After training for 240 epochs, the perf metric was still around 13. Looking for possible explanations leads into several directions. Training a network of this combined size comes with the caveat of a huge number of parameters. Therefore, we either should train even more or supervise the training by introducing a regularization loss. One could also train the individual networks first or use a pretrained version, which we didn't.



(a) Loss **(b) Perf Metric**

Figure 3: Training loss and perf metric plots of our experiments.

Single ResNet101 on the other hand outperformed each other approach. We attribute this to its size and architecture which is perfectly made for these kind of tasks. What surprised us was, that training it without the cluttering data augmentation lead to a better loss, but evaluating with the perf metric resulted in almost the same scores. This can stem from the test images containing more occlusion than the test set. We proceeded similarly with the flip-augmentation.

4.1 Circumstances

Since we all were forced to work from home, one challenge we faced was the communication. Although we met via zoom regularly, it still occurred that we had misunderstandings and implemented the same things or solved errors twice.

4.2 Limitations

Although the provided dataset with 20'000 different hand images is very large, the variety of the hands might be a limitation: As far as we can tell, all hands belong to white male humans. Therefore, our model would be less accurate predicting female hands or hands from humans with a different skin color. Also, the dataset only features right hands. However, this can be resolved by flipping the image and the ground truth target keypoints.

A physical limitation is the hardware. With limited availability of the Tesla GPU and the limited memory of the GTX 1080 Ti, on the cluster, and RTX 3070, on the author's computer, it was difficult running memory intensive models such as [9] and [10].

5 CONCLUSION

As we have shown, a simple common network such as ResNet101 is able to predict quite accurately the 3D Joint locations, given enough training epochs and enough diverse data. We made sure about the latter by augmenting the data in a way that our network never really trained on the same image twice. Such a model works well, is easily adapted to different output dimensions and can be trained fast.

A UNSUCCESSFUL IDEAS

A.1 Metro

According to [9], this method outperformed all precedent methods on the FreiHand scoreboard. However, the major problem with the approach developed in the paper, namely the use of transformers pipelined, is that it is very memory expensive. We tried several GPUs, none of which could execute due to time or memory limitations. The problem is compounded by the fact that the errors which could occur after the forward method of the transformers, or possibly during the forward method (where the memory crash happens), remain unnoticed.

A.2 Boukhayma

One of the main problems encountered with [4] is that the paper mentions that the PCA-projected pose space is 10 while in the GitHub implementation by the author, the projection is done on a space of dimensionality 6. This confusion resulted in the naive usage of 45 as the projection dimensionality, thereby undoing the usefulness of PCA in distilling the essential information. Further, the PCA is done on the output of a dense layer with output dimension 22, which means that the size of the bottleneck is 22 [3], which from an information-theoretic perspective is unlikely to capture the variability of the output size of 21×3. Thus, when training his model with its original settings on FreiHand, the model became saturated at a loss close to 16. Upon changing the ResNet34 backend to a more complex model such as ResNet50, the validation loss did not decrease, suggesting a problem with the MANO decoder. Another problem is the lack of training code or complete model in the code of the GitHub repository [3]. It was also not mentioned how many epochs were required to achieve the desired results. The author had defined several classes such as Bottleneck and DeconvBottleneck which were remnants of a an earlier write-up of the code. It was only upon finding [12] that it became clear that these classes were not used.

REFERENCES

[1] [n. d.]. Flowers Recognition. https://kaggle.com/alxmamaev/flowers-recognition
[2] [n. d.]. Pothole and Plain Road Images. https://kaggle.com/virenbr11/pothole-and-plain-rode-images
[3] boukhayma. 2020. 3dhand. https://github.com/boukhayma/3dhand.
[4] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 2019. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10843–10852.
[5] G Chaladze and L Kalatozishvili. 2017. Linnaeus 5 dataset for machine learning.
[6] hassony2. 2019. Manopth. https://github.com/hassony2/manopth.
[7] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. 2018. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 118–134.
[8] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. 2020. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4990–5000.
[9] Kevin Lin, Lijuan Wang, and Zicheng Liu. 2021. End-to-End Human Pose and Mesh Reconstruction with Transformers. In *CVPR*.
[10] Gyeongsik Moon and Kyoung Mu Lee. 2020. I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation. In *European Conference on Computer Vision (ECCV)*.
[11] Javier Romero, Dimitrios Tzionas, and Michael J Black. 2017. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)* 36, 6 (2017), 1–17.
[12] yueyang. 2021. 3D-Hand-Shape-and-Pose-from-Images-in-the-Wild. https://github.com/yueyang130/3D-Hand-Shape-and-Pose-from-Images-in-the-Wild.
[13] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. 2020. Monocular Real-time Hand Shape and Motion Capture using Multi-modal Data. In *Proceedings of the IEEE International Conference on Computer Vision*. 0–0.
[14] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. 2019. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of IEEE/CVF*. 813–822.