Human Pose Estimation: Progress Report

Robert Lee, Julian Rocha, Wanze Zhang, Nicole Peverley, Rafay Chaudhry, Corey Koelewyn

March 14, 2021

1 The Problem

Human pose estimation (HPE) is the problem domain of identifying body keypoints to construct a body model. Many existing systems accept images as input, with some implementations accepting data such as point cloud or videos. The application of HPE is widespread and benefits many industries. In particular, HPE can revolutionize industries such as augmented reality, animation, gaming, and robotics [1]. There are examples where a person's gait can be used as an unique fingerprint for tracking or identifying individuals in videos [2]. Another subset of HPE is hand pose estimation which can be used to translate sign language. HPE is a difficult problem domain due to challenges such as variability in human appearance and physique, environment lighting and weather, occlusions from other objects, self-occlusions from overlapping joints, complexity of movements of the human skeleton, and the inherent loss of information with a 2D image input [3]. This largely unsolved problem enables us to explore many novel and creative approaches, enriching our learning experience. We are excited to explore these applications, but we decided to limit our scope to a general version of the problem so we could reference the abundance of research available.

There are many variations of HPE systems, which can be roughly categorized into 2D vs 3D, single-person vs multi-person, and different body models. Our group plans to focus on single-frame monocular RGB images containing a single individual rather than a photo with multiple individuals. We believe it will be more feasible to train a deep neural network (NN) with one individual based upon the research papers available and in the given timeframe. Given success with single individuals, we may explore multi-person HPE. Current state-of-the-art techniques for 2D single-person HPE can be categorized into two categories: regression on absolute joint position, or detection on joint locations with heat maps. Since a direct mapping from the input space to joint coordinates is a highly non-linear problem, heat-map-based approaches have proven to be more robust by including small-region information [4]. Thus, we have used the heat map approach.

There are three different types of models used with full body HPE: kinematic, contour, and volumetric, as shown in Fig. 1 [4]. A kinematic model consists of points on each human joint connected by straight lines, similar to a stick-figure skeleton. The contour model consists of 2D squares and rectangles that represent the body, and the volumetric model represents the body with 3D cylinders. Further high-fidelity models implement meshes that capture more details of the human pose. The kinematic model is the simplest model to perform loss metric computations, and thus is preferred by our group as a scope-limiting decision. Using a basic kinematic model will simplify the problem space and encourage us to focus on tangible results that can be used in real life applications. Our goal is to estimate a kinematic model for the individual in each picture.

While there were many existing HPE datasets, very few perfectly matched our chosen requirements for the project. Any available datasets would need to be cleaned and processed. Many datasets contained assorted images that used different types of models for the pose estimation or

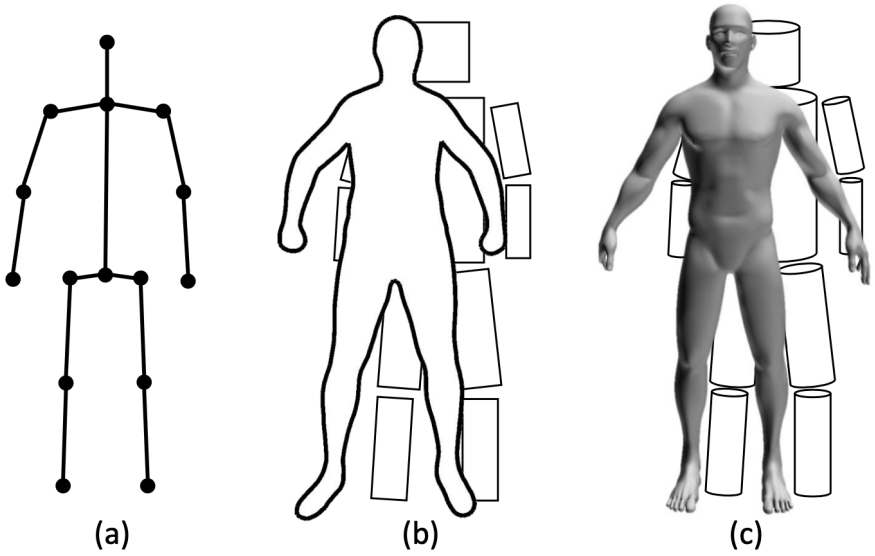


Figure 1: Common body models: (a) skeleton-based, (b) contour-based, (c) volume-based [4]

consisted of multiple people in each image. Our primary choice is the COCO dataset [5]. This dataset consists of 330 K images, of which 200 K are labelled. There are pre-sorted subsets of this dataset specific for HPE competitions: COCO16 and COCO17. These contain 147 K images labelled with bounding boxes, joint locations, and human body masks. We decided to use COCO17 for our generator because. We have decided against using DensePose [6], which is a highly detailed manually annotated subset of the COCO dataset, because it does not offer joint coordinate labels. We still hope to use the MPII dataset [7], which consists of 41 K labelled images split into 29 K train and 12 K test, for validating our model's performance if time permits.

2 Goals

The goal of this project is developing a HPE model that can perform with high accuracy and generalize well to unseen data. The success of the model will be measured using a subset of 2 quantifiable metrics common to the HPE literature [6].

The first metric that has been implemented for testing is Object Keypoint Similarity (OKS). OKS is commonly reported in the literature in terms of AR (average recall) and AP (average precision). It is implemented using the COCO keypoint evaluation server. COCO provides a Python API [9] to evaluate results and compute precision and recall of OKS across scales. It requires that our model outputs be formatted according to the COCO keypoint detection standard [10]. Beyond model evaluation, the API also provides methods for detailed analysis of errors with plots that can be further explored.

The second metric is Percentage of Correct Key Points (PCK) [7] which will need to be implemented ourselves, separately from the COCO evaluation API. A detected joint is considered correct if the distance between the predicted and the true joint is within a certain threshold (threshold varies). PCKh@0.5 is when the threshold = 50

After initial implementations of evaluation metrics and more research, we have decided to not pursue two metrics we mentioned in our formal proposal. The Percentage of Correct Parts (PCP) [7] metric, where a detected joint is considered correct if the distance between the predicted joints

and the true joint is at most half of the limb length, was determined to penalize shorter limbs and is not considered useful in practice. The other metric we chose not to include is Percentage of Detected Joints (PDJ) [7]. For PDJ a detected joint is correct if the distance between the predicted and the true joint is within a certain fraction of the torso diameter. This metric is not as sensitive to the shorter limb problem since smaller people have shorter limbs and smaller torsos. However, this metric can penalize images that have people in different folded positions or torsos that aren't visible.

The viability of this API has been explored and is allowing us to avoid implementing all of our own functions for computing evaluation metrics and align with existing industry-standard evaluation techniques. We will likely add to this evaluation by implementing our own metric for PCK.

For qualitative assessment of the model, a method to visualize the estimated keypoints and limbs overlayed on the input image has been implemented. This will be useful not only for visual demonstrations of the final product, but also to aid understanding the nature of any errors during the training/tuning phase. Should time permit, cross dataset evaluation will be used to demonstrate the model's ability to generalize on a completely different data. Since different datasets have different labelling standards, some work will likely be required to format and prepare new datasets for evaluation. The COCO evaluation API will unfortunately not work with other dataset's ground truth annotation data, however our implementation of PCK could still be used. The system should also perform with a reasonable latency.

3 Plans and Progress To Date

We completed consultations of research papers and online articles, and held weekly standup meetings. We completed the data generator and model training pipelines, enabling us to begin our first round of model training. The data generator has preliminary filtering and no augmentation, and the model training has variable architecture parameters and save/resume capability. We completed the majority of the Preparation milestone mentioned in our Proposal. The remaining task involves data augmentation, which was more complex than it appeared because the transformations need to be mirrored on the joint labels as well. A significant portion of the Architecture milestone is complete as well, with remaining work mostly in processing and evaluating the model output, and exploring techniques to improve training. A breakdown of our individual tasks, specific experiments and our initial results are outlined in Initial Results. The specific details of future experiments are outlined in Task Breakdown.

4 Task Breakdown

4.1 Julian

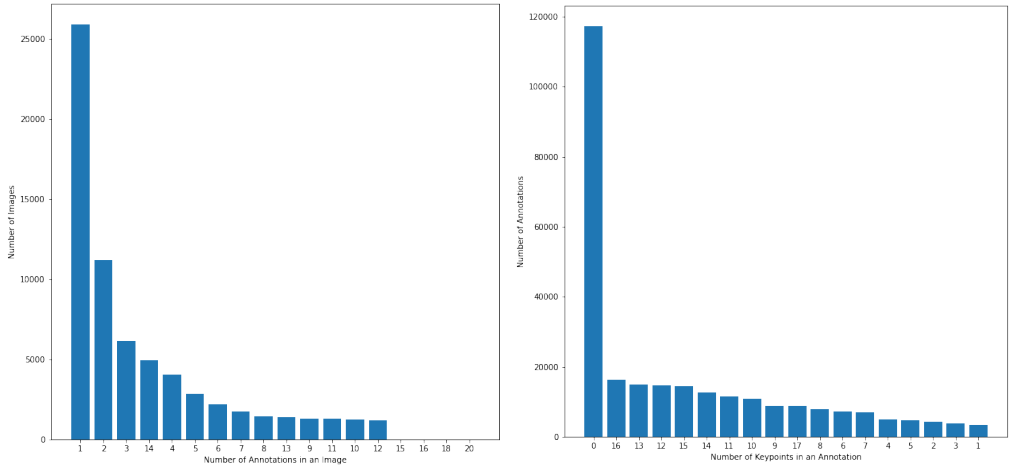
Julian is planning on performing experiments to see how adding data augmentation to the input pipeline will affect model performance. These experiments will include: varying bounding box size from 110% to 150%, flipping images horizontally, rotating images slightly, and various adjustments to brightness, contrast, and noise/grain. Augmentation will be done online between batch fetches and will replace examples. Due to the long training times of the current model, Julian may explore the option of converting a keras model to a TPU compatible model, which can be run on Google Colab TPU's to hopefully reduce training time. Finally, Julian would like to explore hyperparameter tuning and training of the model.

4.2 Robert

5 Initial Results

5.1 Julian - Data Generator Pipeline

Various metrics were gathered on the COCO dataset to help inform how the data should be processed as well as how the model should handle different scenarios. The two metrics most impactful to the project thus far are documented in Fig. 2. There are 66,808 images in the dataset containing a total of 273,469 annotations. Despite the fact that we have chosen to tackle single person and not multi person pose estimation, a large number of the COCO images contain more than one annotated person. It would be desirable if we did not have to discard these images, so cropping to a bounding box can allow us to convert a multi person image into a single person image. Most of the annotations in the dataset do not have all 17 key points of the body labelled. The model should not expect a perfect human image with all key points visible in frame. The model should instead output a dynamic number of keypoints based on what it can find. But what is the purpose of an annotation with 0 labelled keypoints? Fig. 3 shows two examples of these 0 keypoint annotations. Clearly the bounding boxes of these annotations denote people, but because there are no labeled keypoints, these examples may confuse our model. Therefore, despite the fact that 0 keypoint annotations make up 42.89% of the total dataset annotations, these annotations will be filtered out during training.



(a) Number of images with a given number of annotations

(b) Number of annotations with a given number of keypoints

Figure 2: COCO metrics

The COCO dataset is more than 20GB so keeping the entire dataset in memory during training is not an option. Image preprocessing needs to be done to get the images and annotations in a format that can be passed to the model. A data generator was developed to tackle these two tasks. To prevent the generator from being the bottleneck of the training process, it makes use of multiple cores. The generator fetches images from disk in batches and the next batch can be fetched and processed while the model is performing the front and back propagation on the current batch. The preprocessing responsibilities of the data generator include: cropping to the ground truth bounding box of a person, resizing to the models input resolution and dimensions, and converting ground truth annotations for each keypoint to a heatmap for the cropped images. Fig. 4 shows

Figure 3: Examples of annotations with zero labelled keypoints

an example of the transformations. Fig. 4 only shows the cropping for one person but since there are 4 annotated people, the image would get split into 4 images, each centered on the person of interest. Fig. 4 also only shows the heat map of the left hand, but since COCO annotations contain 17 keypoints, it produces 17 heatmaps per annotation.

5.2 Robert - Model Architecture and Training Pipeline

Various model architecture types were evaluated on the following criteria: complexity, feasibility for this project timeline, and research paper results. The main resource was a survey paper outlining top 2D single-person pose estimation methods shown in Table 1. The majority of the top performing systems were based on an Hourglass backbone. Thus, we selected the network shown in Fig. 5.

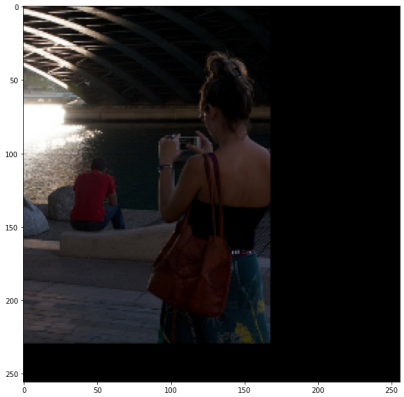
The network architecture was implemented in Keras with a TensorFlow backend. It includes capability to adjust the number of hourglass modules, feature channels, input and output resolution, and the type of 2D convolution block (separable vs normal). Since this is an extremely deep network, we elected to use intermediate training to help gradients propagate into the earlier stages of the network. To verify model training, the Tensorboard logs for training sessions were analyzed. The training and validation graphs are shown in Figs. 6 and 7, respectively, for a 4-stack network. The layer closest to the input side is numbered 0, and the output layer is numbered 3. We note that the trend of the loss, except for a blip in validation layer 3, has a downward trend, which is a good sign the model is learning something. We have not fully determined how the accuracy is assessed, so the absolute value of the loss is not analyzed. However, the accuracy seems to be improving with training. There is a small concern with the training loss flattening out and nearing 0, which may affect back propagation because of vanishing gradients.

Google Colab training pipeline was finalized. We discovered that Colab had an issue accessing a mounted Google Drive folder containing more than 5,000 items. We first explored using symlinks that link on the Colab virtual machine (VM) disk to the mounted Google Drive shared folder. The files still needed to be visited once to add them to the cache. While this improved performance, we determined that it would be faster to download the dataset on the VM disk every time. Thus, we wrote a script that consistently sets up the Colab environment.

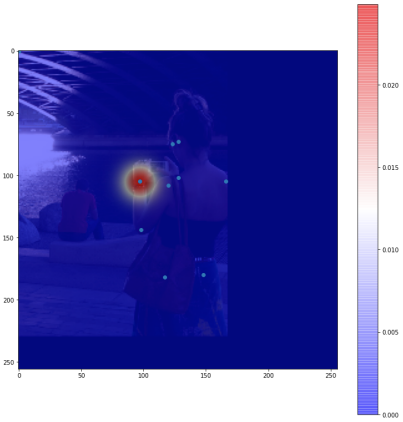
Finally, since Colab instances are time-limited, model saving and resuming was essential. This was implemented by command line arguments when the training script was run.

(a) Original image

(b) Original image with ground truth annotations overlaid



(c) Processed image



(d) Processed image with translated keypoints and left hand heat map overlaid

Figure 4: Example of transformations applied by the data generator

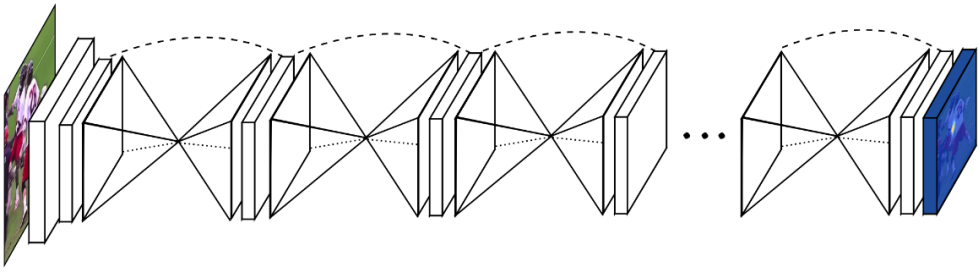


Figure 5: A stacked hourglass network for HPE [8]

References

[1] Pose Estimation Guide — Fritz AI. [Online]. Available: https://www.fritz.ai/pose-estimation/

[2] W. Zeng and C. Wang, "Human gait recognition via deterministic learning," *Neural Networks*, vol. 35, pp. 92–102, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360801200192X

[3] L. Sigal, *Human Pose Estimation*. Boston, MA: Springer US, 2014, pp. 362–370. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_584

[4] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, p. 102897, Mar 2020. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2019.102897

[5] Common Objects in Context. [Online]. Available: https://cocodataset.org/#home

[6] DensePose. [Online]. Available: http://densepose.org/

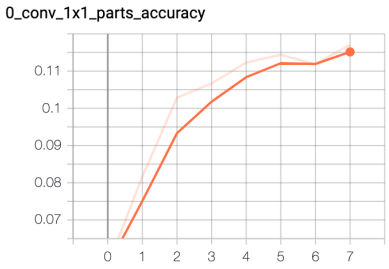
[7] MPII Human Pose Database. [Online]. Available: http://human-pose.mpi-inf.mpg.de/#dataset

[8] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," 2016.

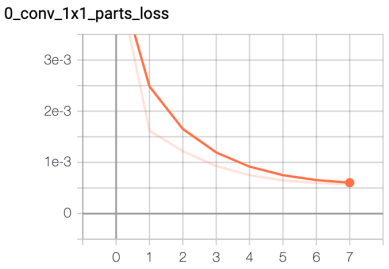
6 Appendix

Table 1: A summary of 2D single-person human pose estimation methods [4]

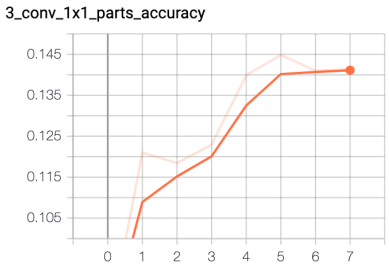
Methods	Backbone	Input size	Highlights	PCKh (%)
Regression-based				
(Toshev and Szegedy, 2014)	AlexNet	220×220	Direct regression, multi-stage refinement	-
(Carreira et al., 2016)	GoogleNet	224×224	Iterative error feedback refinement from initial pose.	81.3
(Sun et al., 2017)	ResNet-50	224×224	Bone based representation as additional constraint, general for both 2D/3D HPE	86.4
(Luvizon et al., 2017)	Inception-v4+ Hourglass	256×256	Multi-stage architecture, proposed soft-argmax function to convert heatmaps into joint locations	91.2
Detection-based				
(Tompson et al., 2014)	AlexNet	320×240	Heatmap representation, multi-scale input, MRF-like Spatial-Model	79.6
(Yang et al., 2016)	VGG	112×112	Jointly learning DCNNs with deformable mixture of parts models	-
(Newell et al., 2016)	Hourglass	256×256	Proposed stacked Hourglass architecture with intermediate supervision.	90.9
(Wei et al., 2016)	CPM	368×368	Proposed Convolutional Pose Machines (CPM) with intermediate input and supervision, learn spatial correlations among body parts	88.5
(Chu et al., 2017)	Hourglass	256×256	Multi-resolution attention maps from multi-scale features, proposed micro hourglass residual units to increase the receptive field	91.5
(Yang et al., 2017)	Hourglass	256×256	Proposed Pyramid Residual Module (PRM) learns filters for input features with different resolutions	92.0
(Chen et al., 2017)	conv-deconv	256×256	GAN, stacked conv-deconv architecture, multi-task for pose and occlusion, two discriminators for distinguishing whether the pose is 'real' and the confidence is strong	91.9
(Peng et al., 2018)	Hourglass	256×256	GAN, proposed augmentation network to generate data augmentations without looking for more data	91.5
(Ke et al., 2018)	Hourglass	256×256	Improved Hourglass network with multi-scale intermediate supervision, multi-scale feature combination, structure-aware loss and data augmentation of joints masking	92.1
(Tang et al., 2018a)	Hourglass	256×256	Compositional model, hierarchical representation of body parts for intermediate supervision	92.3
(Sun et al., 2019)	HRNet	256×256	high-resolution representations of features across the whole network, multi-scale fusion.	92.3
(Tang and Wu, 2019)	Hourglass	256×256	data-driven joint grouping, proposed part-based branching network (PBN) to learn representations specific to each part group.	92.7



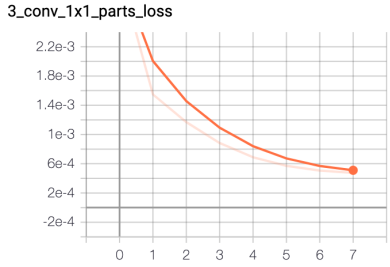
(a) Training Layer 0 Accuracy



(b) Training Layer 0 Loss

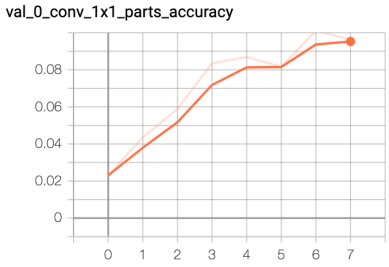


(c) Training Layer 3 Accuracy



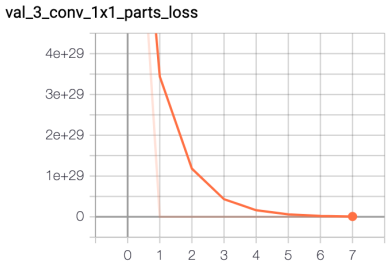
(d) Training Layer 3 Loss

Figure 6: Training accuracy and loss vs. epochs

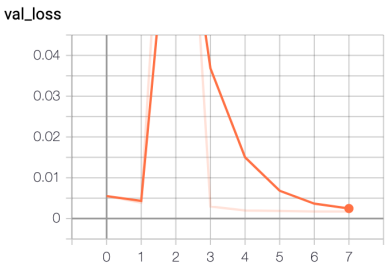


(a) Validation Layer 0 Accuracy

(b) Validation Layer 0 Loss



(c) Validation Layer 3 Accuracy



(d) Validation Layer 3 Loss

Figure 7: Validation accuracy and loss vs. epochs