Whose Hands are These? Hand Detection and Hand-Body Association in the Wild Paper Implementation

Melike Çolak
Department of Computer Science
Hacettepe University
Ankara, Turkey
n22239753@cs.hacettepe.edu.tr

July 9, 2023

ABSTRACT

This study addresses the problem of detecting hands and associating them with their corresponding bodies under unconstrained conditions, as developed in the paper Whose Hands Are These? Hand Detection and Hand-Body Association in the Wild by Narasimhaswamy et al. Narasimhaswamy et al. (2022). As stated in the paper, the association of hands and bodies plays a significant role in many applications such as hand tracking and hand contact estimation. It is, for instance, helpful in identifying people when understanding hand gestures in human-human communication.

As implemented in the base paper, I used the Detectron2 library and the Mask RCNN model in my study. The advanced Mask R-CNN R 50 FPN 3x model in Detectron2 facilitated the simultaneous detection of hands and the body location of the associated person. The model first detects sets of hands and bodies and then employs a Hand-Body Association Network to predict association scores between them; I successfully implemented this process. These scores serve as the foundation for linking each detected hand to its respective body location, a task that becomes increasingly complex due to the potential presence of multiple individuals, varying overlaps, and occlusions.

Furthermore, I conducted an extensive examination of the new and challenging dataset, BodyHands, which consists of unconstrained images annotated with hand and corresponding body locations. In my reimplementation, I included a comprehensive discussion of the dataset, a detailed explanation of the model, and a thorough evaluation of the results. Toward the end of this study, I documented the challenges faced during the implementation and the solutions developed in response.

The experimental results aimed to be reproduced from the paper and the results I obtained are thoroughly examined in the study. Code and data can be found at `https://github.com/melikecolak/bodyhand`.

1 INTRODUCTION

Understanding the relationship between detected hands and their corresponding bodies can provide crucial information in many contexts, from gesture recognition in communication systems to identification methods in surveillance. Narasimhaswamy et al. (2022), investigate a new problem involving the detection of hands and the identification of the corresponding individual associated with each detected hand. They argue that this specific task plays a vital role in various downstream operations, including but not limited to hand tracking and hand contact estimation.

The task of associating hands with their corresponding individuals is a challenging one, particularly under unconstrained conditions where a scene may include multiple individuals with varying levels of overlap and occlusion. In response to this challenge, the researchers propose a novel end-to-end trainable convolutional network designed to detect hands concurrently and the body location of the respective individuals. Their method begins with the detection of a series of hands and bodies. Following this, they employ a unique Hand-Body Association Network to predict the association scores between the detected hands and bodies. These scores are then used to identify the body location associated with each detected hand.

In addition to this method, they introduce a new, highly challenging dataset named BodyHands. This dataset comprises unconstrained images complete with annotations for hands and their respective body locations. The authors conduct an extensive series of experiments on the BodyHands dataset as well as another public dataset. The results of these experiments effectively demonstrate the efficacy of their proposed method. The authors conclude their study by highlighting the significant benefits of hand-body association in two crucial applications: hand tracking and hand contact estimation. Their experimental results reveal that the methods used in hand tracking and hand contact estimation can be significantly improved through reasoning about the hand-body association.

Furthermore, the authors explain how detecting hands in an image and identifying the corresponding person for each detected hand can be beneficial for tasks like action recognition and scene understanding, particularly in multi-person images and videos. For instance, understanding hand gestures in human-human communication becomes easier. Also, assessing motor and social skills in children with mental disorders becomes possible by tracking their hands and how these hands interact with objects and people during tabletop games. Ultimately, the hand-body association is crucial in developing safety applications and assisting individuals working with hand-held tools in manufacturing settings.

2 DATA DESCRIPTION

In the section dedicated to data description, the authors detail the new BodyHands Dataset, an expansive collection of data assembled by Narasimhaswamy et al. (2022) to advance the development and assessment of hand-body association methodologies. The BodyHands dataset is characterized by its large-scale nature and includes unconstrained images, all annotated to indicate the location and correspondence of hands and bodies. Narasimhaswamy et al. (2022) also mentions that the foundation for the BodyHands dataset is the ContactHands dataset Narasimhaswamy et al. (2019), another comprehensive collection of unconstrained images annotated with hand polygon locations and their corresponding contact states. ContactHands, which contains images from well-known datasets such as MS COCO by Lin et al. (2014), PASCAL VOC by Everingham et al. (2010), Oxford-Hand by Mittal et al. (2011), TV-Hand, and COCO-Hand all compiled by Narasimhaswamy et al. (2019), was chosen as the source for various reasons.

The researchers' primary aim was to create a dataset that would allow them to develop and train methods capable of robustly detecting and associating hands and bodies, irrespective of factors like shape, size, skin tone, and motion blur. Additionally, they intended to develop a mechanism to detect and associate hands and bodies in challenging scenarios, especially in situations with mutual occlusions between people. Finally, they wanted to employ hand-body association in established applications, such as hand contact estimation, which necessitated the inclusion of contact state annotations. All these requirements were suitably met by the ContactHands dataset.

The BodyHands dataset is comprised of 20,490 images, featuring 57,898 annotated polygons for hands and 63,095 axis-parallel rectangular bounding boxes representing people. The dataset includes 19,810 people with one annotated hand, 19,044 people with two annotated hands, and 24,241 annotated people who have no annotated hands—likely due to the hands being either occluded or too small to annotate. Figure 1 presents an annotated image taken from the BodyHands dataset. The image distinctly illustrates that only one human body has been labeled. The reason behind this is that, within the parameters of this dataset, a body is only labeled if it includes an associated hand class. Thus, even if multiple bodies are present in the image, those without associated hand classes remain unlabeled.

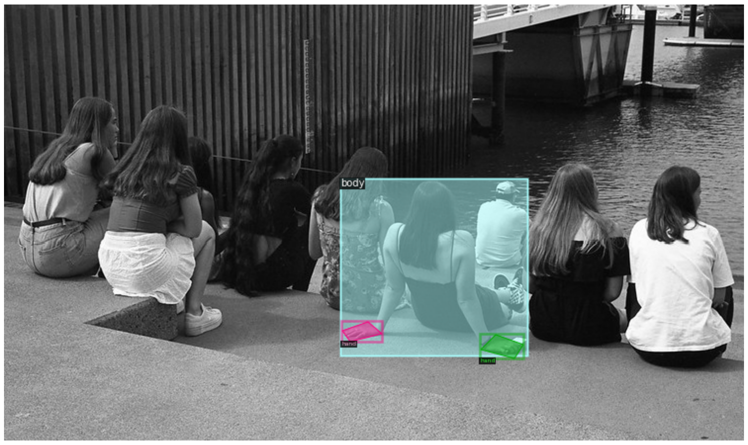


Figure 1: An example of an image with body and hand classes in the HandBody dataset.

3 MODEL DESCRIPTION

In this section, I introduce the model I used for the hand-body association problem, which is based on Detectron2. This framework, developed by FAIR (Facebook AI Research), represents the next generation of their platform for object detection and segmentation. Detectron2 provides CUDA and PyTorch implementations of cutting-edge neural network architectures and includes pre-trained models for object detection, instance segmentation, person keypoint detection, and more. It effectively serves as a computer vision model zoo with notable models such as Faster R-CNN, Mask R-CNN, RetinaNet, and DensePose, along with recent additions like Cascade R-CNN, Panoptic FPN, and TensorMask.

Detectron2 supports tasks like keypoint detection, object detection, and semantic segmentation, utilizing datasets registered in COCO JSON format. The COCO models in Detectron2 were trained on train2017 and evaluated on val2017, with default settings that differ from the original Detectron's standard settings due to the inclusion of scale jittering in data augmentation. I also considered baselines for Faster/Mask R-CNN based on three different backbone combinations: ResNet+FPN, ResNet conv4, and ResNet conv5 with dilations (known as Dilated-C5).

In my study, I used the Mask R-CNN R 50 FPN 3x model, which is celebrated for its robust object detection and pixel-level segmentation capabilities. The various aspects of this model are signified by its nomenclature: Mask R-CNN represents the model type, R50 refers to the ResNet50 backbone used for feature extraction, FPN denotes the Feature Pyramid Network employed for processing these extracted features and 3x stands for the model's training schedule. A detailed explanation of the Mask R-CNN R 50 FPN 3x model is a specialized type of neural network designed for highly effective object detection, classification, and pixel-level segmentation. This model has been instrumental in my work, where it has showcased its proficiency in tackling complex image-understanding tasks.

After summarizing the model in general, we can now move on to the model details. The Mask R-CNN component of the name denotes the model type. This model is an extension of the popular Faster R-CNN object detection model and adds a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. This added functionality empowers the model to perform instance segmentation, which goes a step beyond object detection by precisely delineating the boundaries of each identified object. ResNet50 backbone, which forms the basis for feature extraction in this model. ResNet50, a 50-layer deep version of the ResNet architecture, utilizes residual learning and skip connections to learn complex features without succumbing to the problem of vanishing gradients. ResNet50 is known for its ability to capture an extensive range of features from input images, from low-level details to high-level semantic features. The other part of this model is Feature Pyramid Network, a structure integrated into the model for handling multi-scale objects. By leveraging the inherent multi-scale, pyramidal hierarchy of deep convolutional networks, the FPN enhances the model's capability to detect objects of different sizes in an image. The FPN constructs a multi-level feature pyramid from a single-scale input and is particularly effective in improving the detection performance for small objects. Finally, the 3x in the model's name signifies the training schedule, indicating that the model's training extends over three times the standard period. This longer training duration allows the model to better generalize and learn more complex patterns from the training data.

In summary, the Mask R-CNN R 50 FPN 3x model employs a multi-step, multi-layered process to perform object detection, classification, and segmentation. Each step, from feature extraction with ResNet-50, processing through the FPN, and object location identification with a Region Proposal Network, to class predictions and bounding box regressions in the final Head stage, is meticulously designed to ensure optimal performance. This makes the model adept at analyzing and understanding complex images, detecting and segmenting objects of varying scales and shapes, and delivering accurate, high-quality results.

The functioning of Mask R-CNN R 50 FPN 3x involves a sequence of steps. It commences with feature extraction via a ResNet-50 backbone, progresses to a Feature Pyramid Network, identifies object locations with a Region Proposal Network, and goes through the Region of Interest (ROI) Alignment stage before finally arriving at the Head stage. Here, class predictions, bounding box regressions, and mask outputs are produced.

This intricate series of operations empowers Mask R-CNN R 50 FPN 3x to deliver object detection, classification, and segmentation capabilities for each detected object. This makes it especially effective for complex image analysis and understanding tasks, and for detecting and segmenting objects of diverse scales and shapes. I successfully incorporated this model into my study, which effectively showcased the power and versatility of Detectron2's architecture.

4 Experimental results and comparisons

In this study, I first utilized the Pascal VOC format to register a dataset for the hand-body association problem. Each image in the dataset comes with annotations indicating which hand belongs to which body. I created a function named new-load-voc-instances to load the dataset. This function retrieves the filenames, heights, and widths of the images in the dataset. Moreover, it constructs a dictionary for each hand and body in the image, encapsulating an array of information. Coordinates of hands and bodies, classes of hands and bodies, as well as a body-id that signifies which body is associated with each hand, are included in this dictionary.

Subsequently, I introduced another function called new-register-pascal-voc. This function introduces the dataset I have loaded to Detectron2 and indicates what it is. This process allows the Detectron2 library to appropriately use the corresponding dataset for subsequent analyses. Through this code snippet, I successfully loaded and introduced a hand and body annotated dataset to Detectron2. Following these steps, I started training the model that would work on the hand-body association problem. The code segment details the setup and configuration for training an instance segmentation model with Detectron2, utilizing a model architecture from Detectron2's model zoo, specifically the COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml. This configuration employs the Mask R-CNN framework with a ResNet-50 backbone and a Feature Pyramid Network (FPN) to facilitate multi-scale prediction. Notably, this model has been pre-trained on the COCO dataset and undergone an extended training schedule, three times the standard length. The training is explicitly designated to operate on a CPU.

The designated datasets for training and testing are the BodyHands train and BodyHands test, respectively. In terms of data loading, the configuration calls for a single process. The model's weights at initialization are derived from the pre-trained weights on the COCO dataset.

Training is set to process ten images at each step, with an initial base learning rate of 0.00025. This learning rate is dynamically updated during the training process according to a pre-defined schedule. The training is set for a maximum of 1000 iterations. Each image in the training will be supplied with 128 region proposals, while the model is set to predict two classes. In the evaluation, the minimum score threshold for a region proposal to be counted as a positive instance is 0.5, meaning only predictions with a confidence score above this threshold are considered. The model is instantiated with the DefaultPredictor, a simple high-level API for inference in Detectron2. Though this setup is apt for this specific task, parameters may require adjustments when applied to different tasks, contingent on the respective size, complexity, and nature of the training data.

One significant aspect of this configuration is the dynamic learning rate. Although it starts at 0.00025, it is not a fixed value. During training, the learning rate will adapt according to a schedule based on training progress, allowing it to decrease as the model begins to converge, improving the fine-tuning of the model parameters. This feature, often known as learning rate scheduling or learning rate decay, is a standard practice in training deep learning models to prevent overshooting the optimal solution in the high-dimensional parameter space. In the given parameter configuration, the training process was performed over a total of 1000 iterations, which equates to roughly 60 epochs, spanning an approximate duration of 12 hours. On average, each epoch took about 12 minutes to complete. The evolution of the training and validation accuracy values during the training process is depicted in Figure 2.

A notable point in this evolution is the abrupt dip in performance observed in the 5th epoch, which is speculated to be an artifact of an imbalanced distribution present in the validation dataset. However, as the training progressed, the optimization of the learning rate throughout the process fostered a beneficial alignment of the model with the data. This was manifested as an encouraging good fit situation. The parallel trend observed in training and validation accuracy values suggested a harmonious convergence, indicating an effective learning process free of overfitting or underfitting complications. This parallel trend is a positive indication that the model generalized well from the training data to unseen data in the validation set.

While the current accuracy values are satisfactory, there is always room for improvement in machine learning models. To enhance the training process, several strategies could be implemented. To enhance the training process, several strategies could be implemented. Increasing the number of iterations might lead to improved performance as the model would have more exposure to the data. However, the risk of overfitting - where the model performs exceptionally well on training data but poorly on unseen data - should be considered.

Adjusting the batch size also offers potential for improvement. Larger batch sizes allow the model to process more data points simultaneously, potentially enabling more accurate gradient updates and faster learning. The limitation here is computational resources, as larger batches require more memory. Optimizing the learning rate or employing learning rate decay strategies could provide significant benefits. Adjusting the learning rate helps strike a balance between speedy learning and ensuring no crucial features in the dataset are overlooked, which is critical for effective model optimization. Lastly, advanced techniques like sophisticated optimization algorithms could further boost the model's performance. The choice of which strategies to employ depends heavily on the dataset's specific characteristics and the computational resources at hand.

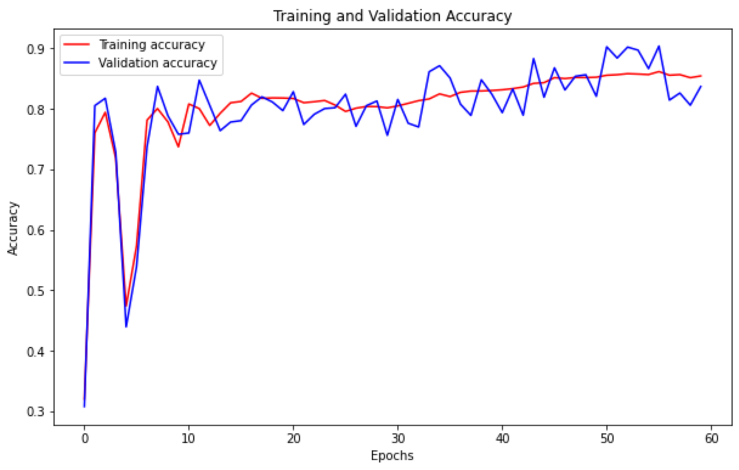
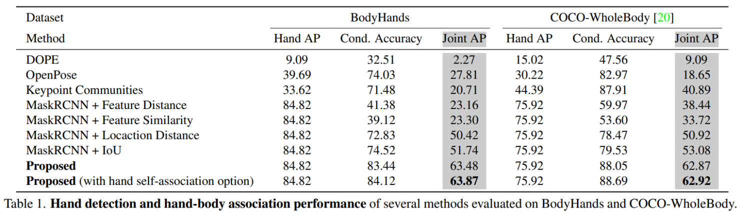


Figure 2: Train and validation accuracy values of the presented model.

While conducting the evaluation, I utilized Detectron2's COCOEvaluator function to assess the model against a dataset that comprised 5983 instances of the hand class and 4341 instances of the body class. This function allows for straightforward computation of evaluation metrics in the COCO format. After the evaluation, the model achieved an Average Precision (AP) score of 81.22 for the hand class. The principal objective during the training phase was to exceed the hand's AP score of 84.82, a benchmark set by the model trained on the BodyHands dataset, as referenced in Figure 3 of the original article.

However, despite substantial efforts, surpassing this score proved challenging due to certain issues encountered during the process. These challenges, which will be discussed in depth in the Challenges and Discussions section, played a significant role in the training process, providing valuable learning experiences and highlighting areas for potential improvement. Although the benchmark AP score wasn't surpassed on this occasion, these insights gained will undoubtedly contribute to future improvement efforts.



Dataset	BodyHands			COCO-WholeBody [20]		
Method	Hand AP	Cond. Accuracy	Joint AP	Hand AP	Cond. Accuracy	Joint AP
DOPE	9.09	32.51	2.27	15.02	47.56	9.09
OpenPose	39.69	74.03	27.81	30.22	82.97	18.65
Keypoint Communities	33.62	71.48	20.71	44.39	87.91	40.89
MaskRCNN + Feature Distance	84.82	41.38	23.16	75.92	59.97	38.44
MaskRCNN + Feature Similarity	84.82	39.12	23.30	75.92	53.60	33.72
MaskRCNN + Locaction Distance	84.82	72.83	50.42	75.92	78.47	50.92
MaskRCNN + IoU	84.82	74.52	51.74	75.92	79.53	53.08
Proposed	84.82	83.44	63.48	75.92	88.05	62.87
Proposed (with hand self-association option)	84.82	84.12	**63.87**	75.92	88.69	**62.92**

Table 1. **Hand detection and hand-body association performance** of several methods evaluated on BodyHands and COCO-WholeBody.

Figure 3: Scores according to different performance metrics of the methods implemented in the base paper according to the dataset and method.

5 Challenges and discussions

When starting the project, it was difficult for me to understand and apply a complex research paper for the first time. The first difficulty I had was that I realized quite late that the results I wanted to implement in milestone 2 were in Table 1, not Table 2. The metrics I specified in Milestone were valid for the video dataset. But I had planned to experiment using a dataset of images called HandBody, which my hardware barely sufficed. I hope you don't mind this confusion in target experimentation. I am very sorry that I could not inform you about this situation earlier.

After extensive research, I followed a specific approach, starting with analyzing the dataset. Then I decided to use the Detectron2 model in my implementation. In Detectron2, it is necessary to format the dataset with appropriate labels before training a model. I downloaded the BodyHands dataset and attempted to register it using a code I wrote for the Pascal VOC label format (XML). However, some images in the dataset were missing hand and body labels, resulting in errors during registration. I identified and removed 16 images with missing labels using a Python code. Then, I read the hand and body annotations along with the coordinates for polygon generation from the XML files and registered them as the train-test split.

The training phase was the most challenging part for me. Because Detectron2 is known for its efficient performance on GPUs. Unfortunately, as my computer did not have a GPU, I had to configure everything to run on the CPU. Without CUDA and cuDNN support, the processing tasks took longer on the CPU. For example, I had to wait a considerable amount of time for the dataset to load before each training attempt. This situation affected the training parameters. Although the default batch size in Detectron2 is 512, my hardware only allowed a maximum batch size of 128. Another important parameter was the number of workers. Setting the number of workers to 0 meant that data loading would occur on the main process only. When the number of workers was set higher than 0, data loading would happen in parallel across multiple worker processes. Since I didn't have a GPU, I set the number of workers to 0, disabling parallel processing. As a result, data loading was slower when working with a large dataset like BodyHands.

Furthermore, each training session consisting of 60 epochs, lasting approximately 10 hours, didn't provide enough time for extensive hyperparameter tuning. My aim was to experiment with different parameter settings and observe the model's behavior. Despite getting close to the target AP score and requesting additional time, I couldn't achieve the desired outcome.

To improve this project, there are several potential solutions. First, acquiring a GPU or utilizing cloud computing resources with GPU capabilities would significantly speed up the training process and allow for larger batch sizes. Additionally, increasing the dataset size and diversity through data augmentation techniques would enhance the model's ability to generalize to different scenarios. Furthermore, fine-tuning hyperparameters, such as learning rate, weight decay, and optimization algorithms, can potentially improve the model's performance. Finally, leveraging transfer learning by starting with a pre-trained model and customizing it on the BodyHands dataset could lead to better results. This approach takes advantage of the pre-learned features and can reduce training time and data requirements.

By considering these solutions and implementing them in the project, there is a higher chance of achieving improved results and surpassing the limitations encountered during the implementation phase.

REFERENCES

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338, 2010.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Arpit Mittal, Andrew Zisserman, and Philip HS Torr. Hand detection using multiple proposals. In *Bmvc*, volume 2, pp. 5. Citeseer, 2011.

Supreeth Narasimhaswamy, Zhengwei Wei, Yang Wang, Justin Zhang, and Minh Hoai. Contextual attention for hand detection in the wild. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9567–9576, 2019.

Supreeth Narasimhaswamy, Thanh Nguyen, Mingzhen Huang, and Minh Hoai. Whose hands are these? hand detection and hand-body association in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4889–4899, 2022.