Evaluating Approaches to Human Body Orientation Estimation

Caden Kroonenberg, Ahmni Pang-Johnson, Nick Velicer
Joe Nordling, Aydan Smith, Derek Zhang

Introduction

Human body orientation estimation (HBOE) is the task of approximating the angle a person is facing relative to the camera. It is used for applications such as autonomous driving in order to provide rich information about a vehicle's environment for the purpose of predicting where people may move in the near future. Because a person is most likely to move in the direction they are facing, orientation may be used as an indicator when attempting to predict a person's general future trajectory. Current approaches to HBOE are dominated by machine learning methods of two general varieties: predicting a keypoint skeleton which yields 3D orientation (from which 2D orientation may be inferred) and predicting 2D orientation using classification models. In this report, we follow the latter method and analyze the effectiveness of two different machine learning architectures for HBOE and investigate the effect of three different image preprocessing methods on this process.

Dataset

In order to use a machine learning model to estimate human orientation from an image, it is necessary to train and test the model on a robust dataset with labels corresponding to meaningful descriptions of orientation. The Common Objects in Context - Monocular Estimation of Body Orientation in the Wild (COCO-MEBOW) dataset provides labels for approximately 130,000 individuals using approximately 55,000 images from the COCO dataset [1][2]. Labels for the images are an integer value from 0-360, divided into 72 bins (see Figure 1). While the COCO dataset was not exclusively created for human orientation estimation, the data points provided, such as the 17 unique key points allow for a large subset of the images to be utilized for estimating human body estimation. As such, the COCO-MEBOW dataset is a subset of the COCO dataset, customized for human body orientation estimation.

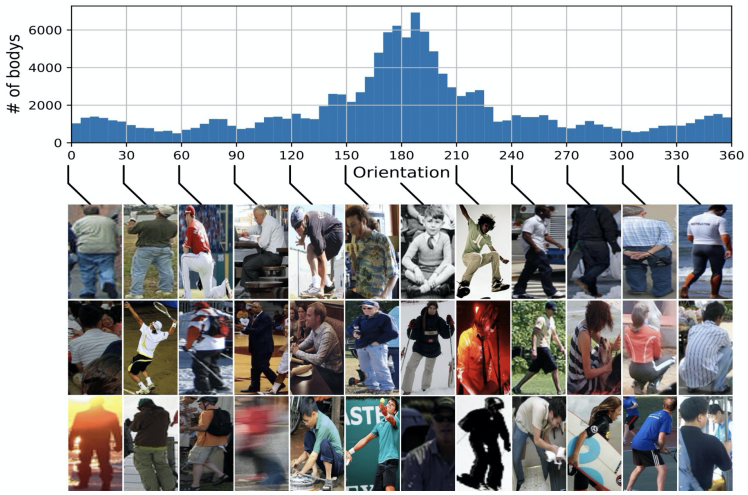

Figure 1: Distribution and examples of body orientation in the COCO-MEBOW dataset [1]

The COCO-MEBOW dataset provides images of subjects in a variety of backgrounds while other datasets primarily feature subjects indoors. Furthermore, subjects in the COCO-MEBOW dataset may be occluded which mimics real-world applications in which the subject may not be in clear view of the camera. This diversity of data yields improved robustness, which in turn allows models to be trained with higher generality. For the purpose of using an HBOE model for autonomous navigation applications, this is ideal as subjects may appear in a variety of environments, perhaps while partially occluded.

Methods

To validate the effectiveness of the COCO-MEBOW dataset, Wu et al. designed a machine learning model architecture for HBOE. This model used an HRNet (High-Resolution Net) as a backbone and ResNet as a head unit, as such the authors of the paper dubbed this model the HRNet+Head model. HRNet is a general purpose convolutional neural network. It is differentiated from other models due to its ability to use high resolution images throughout the whole process. Due to this, HRNet is commonly used for detecting objects and image classification. This model is composed of 11 layers and assumes the human instances have already been detected, with inputs being cropped-out 256x192 images of a single human [1]. The model output is a vector of size 72, with each value corresponding to the probability of each orientation bin being the best one to represent the body orientation of the input image. We will utilize this model in our investigation of the dataset and the effectiveness of image preprocessing in HBOE.

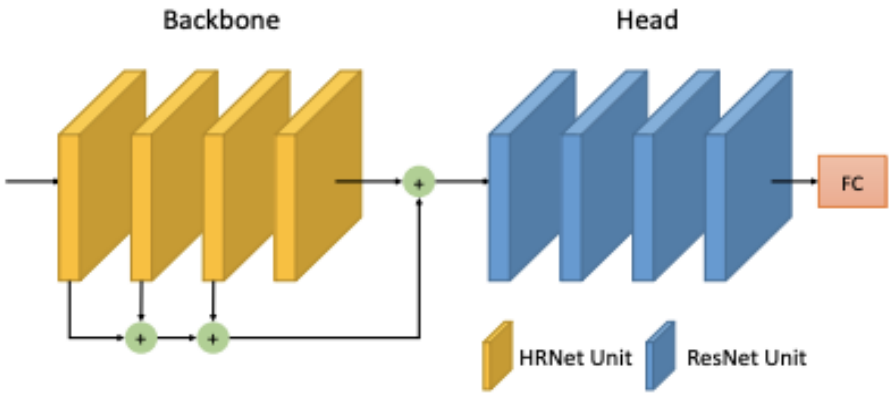

Figure 2: Network architecture for HRNet+Head model [1]

In addition to the HRNet+Head model, a ResNet model was tested as an alternate method for HBOE. This architecture was implemented using the PyTorch Python library with rationale for this being that the HRNet+Head model was also developed using the same library. The ResNet variation used in our experiments was ResNet50, a commonly used architecture for image classification tasks. ResNet50 is composed of 50 residual blocks, with each block being two layers of weights separated by a ReLU activation function and ending in a ReLU activation with the new values being added to the identity of the original values, provided by a skip connection from the beginning of the block [3]. This architecture is promising for HBOE due to its proven ability to perform well in image classification tasks [3].

The HRNet+Head model implementation from Wu et al. included some preprocessing methods including: normalizing all images to 256x192 and employing flipping and scaling augmentation [1]. In addition to the preprocessing models used in the HRNet+Head model, three preprocessing methods will be employed to analyze their effect on HBOE accuracy. These methods (shown in Figure 3) are: histogram equalization, Gaussian blur, and unsharp masking.

Figure 3: Preprocessing methods visualized: no preprocessing (top left), Gaussian blur (top right), histogram equalization (bottom left), unsharp masking (bottom right).

Histogram equalization is a preprocessing method used to optimize contrast and training efficiency. The process transforms the input image to have a uniform distribution of grayscale values, which increases image contrast. This should make it easier for the model to discern a person from the background, thus making it easier to tell which direction they are facing. The method of implementation for the histogram equalization technique is the PyTorch equalize() function which applies a non-linear mapping to the provided dataset [4].

Gaussian blurring applies a kernel with a Gaussian distribution to the image. This replaces pixel values with an average of its surrounding pixel values, giving pixels further away from the center less weight. This can reduce the noise in an image and potentially increase the accuracy of the model. Our implementation utilizes the PyTorch GaussianBlur() method with a 9x9 kernel and a standard deviation value between 0.1 and 2.0 (chosen uniformly at random) [5].

Unsharp masking, also known as inverse Gaussian blur, applies Gaussian blur to a copy of an image, and then subtracts the transform from the original, effectively creating a negative mask of the original with more highlighted edges and better isolation of high-contrast features. This method loses a large amount of the original color and shadow data of the image, and is primarily intended to isolate the strongest components of the image. Our implementation uses scipy's gaussian_filter() with a standard deviation value of 5.0 [6].

Experiments

Training procedures for the ResNet model closely mimicked the HRNet+Head model training procedure in order to ensure model comparison could be primarily based on the difference in architectures. Each model was trained using the designated training and testing MEBOW datasets for 80 epochs with the training set containing ~128K samples and the testing set containing ~5K samples. In each epoch, the full training set is passed through the model, updating model weights based on loss after processing each image using the standard Adam optimizer. Note: the loss used in training the ResNet50 model was mean squared error while the HRNet+Head model used cross entropy loss[1].

To test preprocessing methods, three extra variations of each model architecture were trained; one variation for each preprocessing method. Preprocessing was applied to the dataset as it was loaded into the training procedure. This yielded eight models in total (six models trained with preprocessing and two trained without).

Each of the ResNet models were trained on four NVIDIA GeForce GTX 1080 GPUs working in parallel with a batch size of 64. Training took from 9 to 24+ hours to train depending on the preprocessing method. Each of the HRNet+Head models were trained on an M1 Mac chip and took ~16 hours to train using a batch size of 64.

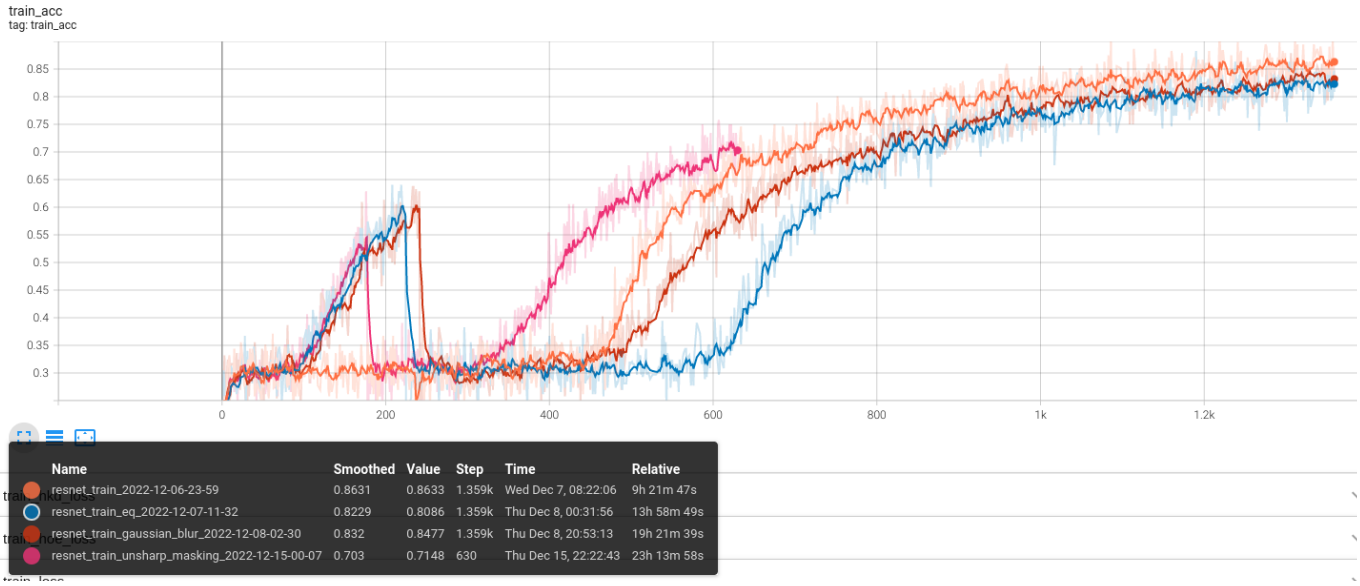

Figure 4: ResNet training accuracy over time

Figure 4 shows training accuracy over time for the ResNet models. One time step denotes 30 batches of 64 having been processed on each of the four GPUs the process was parallelized on. For the training set being of size ~128K, a time step would occur 17 times per epoch, yielding 1360 total logged time steps. Notably, the accuracy of the models trained with histogram equalization and Gaussian blur preprocessing dropped drastically around time step 250, which occurred in the 12th or 13th epoch. This same anomaly occurred in the model trained with unsharp masking around time step 180. Due to time constraints, the reason for these anomalies was not explored.

Results

After training the models as described above, the models were evaluated on the test set of the MEBOW dataset. Metrics for evaluation are standard metrics for HBOE tasks: Accuracy-45°, Accuracy-30°, Accuracy-15° and Accuracy-5°, where Accuracy-X° is the percentage of samples predicted within X° from the ground-truth orientation [1].

Pre-Processing	Model	Acc.-45°	Acc.-30°	Acc.-15°	Acc.-5°
None	ResNet	0.951	0.930	0.856	0.622
	HRNet+Head	0.928	0.884	0.680	0.447
Histogram Equalization	ResNet	0.931	0.906	0.822	0.576
	HRNet+Head	0.908	0.835	0.620	0.394
Gaussian Blur	ResNet	0.940	0.916	0.839	0.598
	HRNet+Head	0.909	0.879	0.672	0.425
Unsharp Masking	ResNet				
	HRNet+Head	0.863	0.803	0.572	0.373

Figure 5: 2D human pose estimation accuracy evaluation on MEBOW test set

Our results, shown in Figure 5, suggest that the ResNet model trained with no preprocessing performs best for HBOE. In fact, our results further suggest that the ResNet50 architecture outperforms the HRNet+Head architecture in general. This especially holds true regarding metrics of higher precision; ResNet and HRNet+Head perform somewhat similarly for Accuracy-45°, but ResNet clearly outperforms the HRNet+Head architecture for more precise metrics such as Accuracy-15° and Accuracy-5°.

The results also show a decrease in system performance when utilizing data preprocessing, which can be explained by an overall net negative of compelling image information after trading detail loss for contrast or smoothness. While not fatal to the system accuracy, the preprocessing provided more of a detriment to the final results than was hoped for. However, this does speak well to the robustness of the models overall, as they performed relatively close to the system accuracy with no pre-processing given varying ranges of alteration to the data set's clarity. It should be noted that due to time constraints, the ResNet model with unsharp masking preprocessing did not finish training due to time constraints. However, from its training accuracy curve shown in Figure 4, it is likely that it performed comparably to the other ResNet models. One may speculate that it will perform the best, given that its training accuracy shows a big upward trend around time step 400, while other models took until around time step 500-600 to experience the same large trend. Interestingly, the unsharp masking model took far longer to train than the other models, taking more than twice as long to train compared to the vanilla ResNet model.

Conclusion

Our experiments successfully tested two different model architectures on the COCO-MEBOW dataset for the purpose of HBOE. Furthermore, the efficacy of several preprocessing methods was tested, although the results did not align with our hypotheses that their employment would yield improved accuracy. In general, it was found that none of the preprocessing methods tested yielded increased performance over the standard dataset. Furthermore, it was found that the ResNet50 architecture consistently outperformed the HRNet+Head architecture proposed in the COCO-MEBOW paper, especially when using metrics of high precision.

Due to time constraints, only the ResNet50 architecture was tested. In future work, alternate variations of ResNet could be tested; deeper ResNet variations such as ResNet101, ResNet152 might yield improved performance and slimmer variations such as ResNet34 might yield faster training time. Furthermore, alterations to training hyperparameters like number of training epochs or the learning rate could be tested for potential increases in performance.

The data preprocessing, while originally intended to increase prediction accuracy, also has compelling promise for the time component of model training. Given the length of time required to train just one of the models, using these processing techniques to preserve important image information more accurately after compression or resizing may yield faster convergence while training. Future optimizations such as hyperparameter tuning and more aggressive preprocessing transforms can also be balanced for information preservation to complement this, as opposed to accuracy on the original data set.

References

[1] Wu, Chenyan, et al. "MEBOW: Monocular Estimation of Body Orientation in the Wild." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[2] 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 *Proceedings of the European Conference on Computer Vi- sion*, pages 740–755. Springer, 2014.

[3] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] "Equalize" Equalize - Torchvision Main Documentation, https://pytorch.org/vision/stable/generated/torchvision.transforms.functional.equalize.html.

[5] "GaussianBlur" GaussianBlur - Torchvision Main Documentation, https://pytorch.org/vision/stable/generated/torchvision.transforms.GaussianBlur.html.

[6] "Scipy.ndimage.gaussian_filter" Scipy.ndimage.gaussian_filter - SciPy v1.9.3 Manual, https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html.

Contributions

Aydan Smith - Resolved cuda dependencies in the pretrained model so that it could run on our machines. Worked on testing the pretrained model with no pre processing to determine the baseline accuracy. Worked on both presentations and the paper with an emphasis on the areas that I worked on and described above.

Joseph Nordling - Configured pretrained models to train on M1 chip to accelerate training process. Implemented and retrained the pretrained models with each of the preprocessing methods detailed in the paper. Worked on the presentation and final report.

Caden Kroonenberg - Implemented ResNet architecture in PyTorch for evaluation. Trained and evaluated ResNet models on MEBOW dataset, which entailed parallelizing the procedures with Cuda for execution on multiple NVIDIA 1080 GPUs. Reached out to one of the COCO-MEBOW authors, Chenyan Wu, in order to gain access to MEBOW annotations. Wrote the project proposal. Detailed the aforementioned work in both presentations and the final report.

Derek Zhang - Researched various preprocessing methods for training optimization. Developed code to apply a Gaussian Blurring transform to the training images.

Nick Velicer - Researched preprocessing methods and finalized implementation for unsharp masking using inverse Gaussian blur, as well as presenting materials for current and future work with image preprocessing. In addition, detailing unsharp masking, preprocessing impacts, and future implementation in the presentation and final paper.

Ahmni Pang-Johnson - Researched preprocessing methods for training optimization, developed code to apply histogram equalization. Worked on the slides and report related to these topics.