OGCNN: Oriented Gradient Convolutional Neural Networks For The OpenMonkeyChallenge

Avinash Akella akell011@umn.edu
Ryan Fredrick fredr321@umn.edu
Riley C. W. O'Neill oneil571@umn.edu

Abstract

Existing Pose Estimation methods tend to use complex preprocessing and transformations to make their models robust to rotation variations. While robustness to scale variations is an actively studied area, rotation invariance is not as thoroughly explored. Since monkeys take many positions, this is extremely desirable for the OpenMonkeyChallenge (OMC) [26]. We propose a novel method, which we call Oriented Gradient Convolutional Neural Network (OGCNN), that achieves promising keypoint detection on the OMC and is more robust to rotational variation.

1. Introduction

The OpenMonkeyChallenge [26] presents a unique and expansive machine learning dataset. It consists of 26 different species of Monkeys with 17 annotated pose key points (eyes, nose, hands, joints, etc) for each image. The goal of the challenge is to correctly identify these landmarks on new images of various species of monkey which differ from each other in varying degrees of size and body type.

Convolutional Neural Networks (CNNs) have been champions in Computer Vision, excelling on tasks of face recognition [20, 19], object recognition [23, 15, 18, 29], human pose estimation [31, 32], multi-person human pose estimation [30], medical image classification [11, 7], and traffic scenes related classification [1, 14]. In fact, their performance on the object recognition task has been described as "superhuman" [5]. Much of this success is attributed to large datasets, parameter sharing, and their immense capacity to learn [10]. CNNs have also been actively used for the pose estimation problem, whose objective is to locate or track the position of an object. Much of this research has been focused on human pose tracking, although some research exists for animal pose tracking as well.

The Simple Baselines' [24] approach was able to achieve pretty accurate results when the monkeys were in upright poses. However, when the images were rotated, the key point detection would often be incorrect [1]. The results from the OMC paper [26] align with our results. For Simple Baselines, the facial features were detected well, but the farther away from the face, the worse the accuracy was. For HigherHRNet, it performed similar to Simple Baselines, but was not quite as accurate for facial key points. Another method, Convolutional Pose Machines [22], detects the key points by producing belief maps for each key point and then uses those maps to make estimates for the next step in the model. This is able to avoid issues from previous models where abnormal poses, such as people diving or upside down, would have their arms incorrectly labeled as their legs, which would then completely throw off the labels for the torso and head.

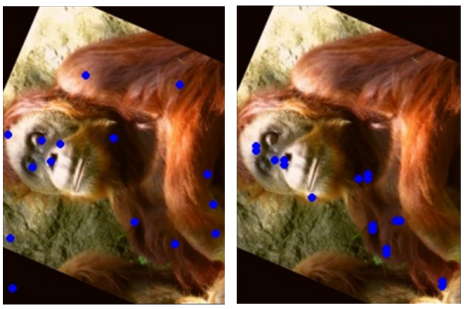

Figure 1. Ground Truth vs Prediction Labels for Simple Baselines

Three main qualities generally make CNNs very successful: 1. translational invariance, 2. multiple levels of automatic local feature tuning and extraction, and 3. relatively few parameters (which helps reduce overfitting) [9]. Further, the universal approximation theorem for CNNs allows them to approximate any function arbitrarily well given enough layers [28]. However, most CNNs are not organically rotation invariant: a small rotation to an image can throw the accuracy tremendously. For the OpenMonkey dataset in particular, monkeys can be in a variety of positions: upright, sideways, supine, upside down in a tree, etc, so a truly rotation invariant network is most desirable.

2. Related Work

Standard Pose Estimation methods, such as Simple Baseline[24], HigherHRNet [3], and OpenPose [2], employ data augmentation for ad-hoc "rotational invariance" - complex pre-processing and transformations to make their models more robust to rotation variations. Data augmentation in Simple Baseline includes scale(±30%), rotation($\pm 40°$) and flip. [3] similarly uses random rotation ($[30°, 30°]$), random scale ([0.75, 1.5]), random translation ([40, 40]), as well as random flip. While data augmentation improves generalizability, it is far from a guarantee of rotational invariance, and it also contributes to higher data complexity [4]. While robustness to scale variations is an actively studied area [3], rotation invariance is not as thoroughly explored. [2] mentions failure cases usually occur due to non-typical poses and upside-down examples. They find that increasing the rotation augmentation mitigates this issue up to some extent, but reduces the overall accuracy of their model on the COCO validation set by 5%.

Numerous methods for true rotational invariance have been suggested. The simplest is to demand the filter matrix be centrosymmetric, which assures translational invariance as well. However, this severely handicaps the detectable features (one cannot get a basic first derivative filter), so this is seldom used. Other attempts for rotational invariance have been implemented, such as [13], which examines rotationally symmetric and Dihedral-4 symmetric networks, but such loses out on translational symmetry. [8] converts an input image to its polar representation and then applies a convolution with cylindrical sliding windows. This model had improved accuracy over conventional CNNs when data augmentation is not performed during training, but this improvement was usually quite small (eg. improving from 99.24% to 99.44%), and it is not translationally invariant.

2.1. Metrics

In accordance with the OMC, we implement the Mean per joint position error (MPJPE) [6]. It measures the normalized error between the prediction and ground truth for each landmark, normalizing w.r.t. the width of the bounding box. The smaller the MPJPE value, the better. The second metric we implement is the Probability of correct keypoints (PCK) [25]. PCK is the detection accuracy given error tolerance. The bigger the PCK value for a model, the better.

The final metric we use is Average Precision (AP). Calculation of Object Keypoint Similarity (OKS) is an intermediate step for this metric, which takes a landmark's relative tolerance (per landmark variance) into account. This is akin to the OKS calculation for the COCO challenge evaluation, but here we augment the tail landmark to match its tolerance with that of the wrist keypoint. The implementation specifics have been referred from the official OMC paper [26]. The higher the PA values, the better.

Model	Pre-trained	MPJPE	PCK@0.02	AP@0.02
HigherHRNet w32 512	Y	0.081	0.235	0.941
HigherHRNet w48 640	Y	0.474	0.0	0.058
SimpleBaseline ResNet 50	Y	0.122	0.117	0.823
SimpleBaseline ResNet 101	Y	0.254	0.0	0.411
SimpleBaseline ResNet 50	N	0.036	0.47	1.0

Figure 2. Baseline Results

2.2. Preliminary Implementations

We implemented a shallow U-Net model inspired from [17] to understand what works with the OpenMonkey dataset. Visually analyzing the results, we observe that the model detects facial keypoints very well, while its accuracy becomes progressively worse as we go further down the body, with the tail keypoint predictions being the worst.

To verify with models that have state of the art performance on human keypoint predictions, we chose to experiment with the Simple Baseline model [24] pre-trained on the Microsoft COCO dataset. This model is akin to our U-Net implementation, in that it uses a ResNet 50 [27] backbone, whose final layer outputs are sent through deconvolution layers to produce high resolution feature maps. This produces good results on the dataset, achieving an AP@0.02 of 0.823. Since the 26 monkey types are of different sizes and builds, we chose to use the HigherHRNet [3] model pre-trained on the Microsoft COCO dataset, as it claims to be scale invariant. The model uses the HRNet [21] model as the backbone, and upsamples its $\frac{1}{4}$ resolution output, unlike conventional architectures that upsample from $\frac{1}{32}$ resolution, thus producing much higher resolution feature maps for precise keypoint localization.

This model does improve our baseline metrics, with an AP@0.02 score of 0.941. This model does produce more precise predictions, but some keypoints, such as those of the tail, and forehead, are missed. This is primarily due to a difference in domain, as these models are pre-trained on the COCO dataset whose keypoints don't have any information about the tail or forehead.

To overcome this and come up with a more concrete baseline, we trained the Simple Baseline model on the

OpenMonkey dataset by cropping out the images at the given bounding box coordinates. This turned out to be the best model, with an AP@0.02 score of 1.0, and MPJPE as low as 0.03. So we chose to use this as our final baseline.

2.3. Limitations

Simple Baseline with a ResNet 101 back-end, and a HigherHRNet model with 48 channels instead of 32, seemed to perform worse. But all baselines used so far, perform badly when the input image is rotated, and the precision of predictions seems to worsen as the amount of rotation increases. The models also perform poorly when the images are blurry, or when a primate is not directly facing the camera.

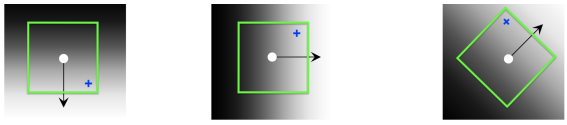

Figure 3. Reorienting the filter along the gradient direction (small plus sign used for reference). These points would all have the same first layer feature with OGCNN.

3. Methods

3.1. OGCNN

To install rotational invariance at the first layer, we propose the Oriented Gradient CNN (OGCNN), which reorients the convolutional filters along the image gradient direction (figure 3) and maintains translational invariance. This is achieved by using Sobel filters to obtain the gradient directions (note that we experiment with multiple filter sizes, as we describe next, for approximate scale invariance). We precompute the gradient directions at a fixed Sobel scale, bin them into b evenly spaced intervals in the unit circle ($b = 8$ seems quite intuitive, corresponding to the 8 pixel neighbors of a pixel), and store the bin labels.

Orienting the convolutional filter along the gradient allows direct estimation of a number of geometric quantities of interest, including Lie derivatives w.r.t. the image gradient. If instead using a 2-prong approach with a monkey mask proposer (setting the input background pixels to 0) and then an OGCNN segmenter, the OGCNN can readily approximate mean curvature of the boundary arc as well as the *circular area invariant* [16] - such is of great interest for segmentation related tasks. However, since image gradients on the monkey boundary generally point inward or outward (depending on intensity), intuition suggests that the OGCNN without a proposer network should be of similar capability when applied directly to the input image. We thus examine both the 2-prong approach (section 4.3) and the direct OGCNN (sections 4.1-4.2).

The model was implemented in PyTorch with Stochastic Gradient Descent with momentum and various losses (focal [12], Iou, L^2) on Gaussian blurred key points. Training was conducted on a Google Cloud Virtual Machine with a NVIDIA Tesla K80 GPU. The model is initialized with b filters $\{F_i\}_{i=1}^b$, wherein F_1 is randomly initialized and $F_i = R_i(F_1)$, where R_i rotates F_1 exactly $\frac{(i-1)2\pi}{b}$ radians about its center (corresponding to the center of the interval). The TorchVision Rotate function was used for this with bilinear interpolation. To ensure correct gradient updates, the $\{F_i\}_{i=1}^b$ are computed in each forward pass from F_1. For more computational efficiency in the future, each F_i could be stored as a model parameter and allowed to undergo the update, whereafter $F_1 \leftarrow \frac{1}{b}\sum_{i=1}^b R_i^{-1}(F_i)$ and the F_i are subsequently updated.

3.2. Network Architecture

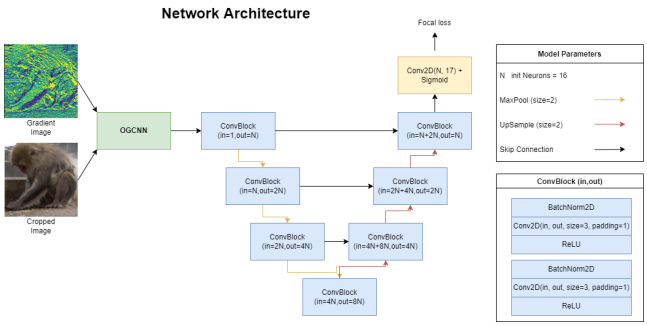

Figure 4. OGCNN Network Architecture

The OGCNN architecture consists of two main components: the OG layer, and the UNet. The UNet model consists of 3 blocks for downsampling, followed by 1 bottleneck block, and 3 upsampling blocks, where each block is a stack of two sandwich layers with BatchNorm, Convolution, and ReLU layers in each. There exist skip connections between corresponding upsampling and downsampling layers as shown in the architecture diagram. The output image has the same resolution as the input, and it is passed through a convolution and sigmoid layers to produce the final keypoint heat maps used for prediction. The OG block has gradient bins, along with the original image as inputs. We began with a single channel OGCNN and expanded it to use multiple channels to detect an array of local and global features. We also experimented with different loss functions and redcued UNet parameters to reduce train time.

4. Results

4.1. 1-Channel OGCNN with UNet

The preliminary results of the 1-channel OGCNN with a shallow UNet backbone and IoU loss were unsatisfactory. For efficiency, it was trained and validated on 40% of the

original training and validation sets. After just over 12 hours of training (and needing more), it achieved an L^2 loss of .225 (averaged over all the pixels in an image and all images) on the pixel difference between the key point heat map and the model output with IoU score of .1.

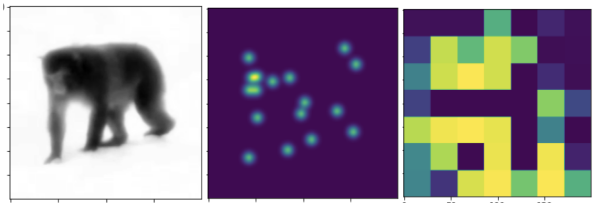

Figure 5. Preliminary OGCNN results. Left: Gray image. Middle: heat maps for key points. Right: Model output.

4.2. Multi-Channel OGCNN With UNet

The use of the focal loss ($L = (1-p)^{\gamma}\log(p)$) [12], which gives more weight to under-represented classes (used for all subsequent training), paired with several OG filters at different scales, provided improved accuracy, better scale invariance, and more informative features as it was able to model more complex relationships. However, it does not handle occlusions very well, i.e. if a monkey is not fully visible. The performance results are listed as Unet OGCNN in Figure 7.

4.3. 2-Pronged Approach

Another experiment was to test the 2-pronged approach of segmentation and subsequent keypoint detection. For this we used an FCN-ResNet model to generate monkey masks, apply it to the original image, and pass that as the input to the OGCNN network. This gave better keypoint accuracy as indicated by a lower MPJPE score, and fewer disruptions caused by occlusions as the network is able to better focus on the monkey body. With this, the generated masks are imperfect, and the network struggles to converge smoothly; a better proposer network could be trained if ground truth monkey masks were had. Interestingly, this performs very similar to the OGCNN network, as hypothesized by us earlier. The performance metrics are under SimpleBasline Resnet 50 OGCNN Masked in Figure 7.

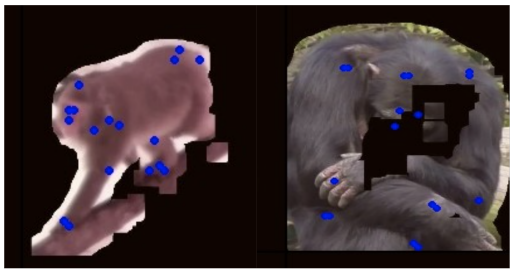

Figure 6. Some identified monkey masks and the model predicted key points. Left: quite good. Right: poor.

4.4. Mix with regular CNNs

The last approach was to integrate the OGCNN network with the Simple Baseline, by training it with OGCNN layer outputs from the UNet architecture. The ResNet backbone of Simple Baseline produced rich features, and performed better on monkeys in varying positions and orientations. But it doesn't quite beat the Simple Baseline; occlusions are still an issue, and it takes longer to train than other models.The performance metrics are listed in Figure 7 under SimpleBaseline ResNet50 OGCNN.

Model	MPJPE	PCK@0.02	AP@0.02
SimpleBaseline ResNet 50	0.036	0.47	1.0
UNet OGCNN	0.228	0.115	0.411
SimpleBaseline ResNet 50 OGCNN	**0.057**	**0.294**	**0.9411**
SimpleBaseline ResNet 50 OGCNN Plus	0.055	0.353	0.9411
SimpleBaseline ResNet 5 OGCNN Trained	**0.115**	**0.235**	**0.7647**
SimpleBaseline ResNet 50 OGCNN Masked	0.106	0.235	0.7647

Figure 7. Comparison of results on unrotated images. The OGCNN with ResNet 50 comes closest to beating the ResNet 50.

4.5. Performance on Rotated images

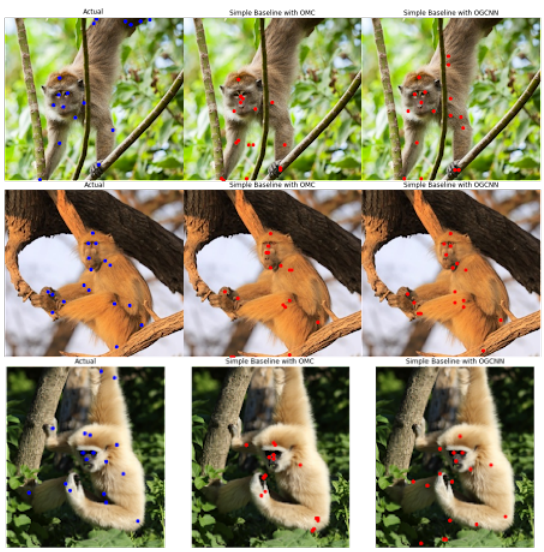

Figure 8. Comparison of Simple Baseline's performance on OpenMonkey image (middle) and OGCNN output image (right) as inputs. Notably the performance of the OGCNN beats the baseline on the upside-down monkey (top).

The performance of all the models, both ours and state-of-the-art, is significantly lower for rotated images. But we observe that the Simple Baseline with OGCNN input layer performs comparatively well compared to the Simple Baseline alone, thus beating the baseline, in a way. In Figure 8, the keypoint predictions for lower half of the top monkey are equally bad in both the cases. But the OGCNN output images help the network better identify facial features. To test this empirically, we run the model with rotated OpenMonkey challenge images, and rotated OGCNN output images as inputs separately and compare performance. The latter has lower MPJPE scores (0.238 compared to 0.266), and much better AP@0.02 (0.411 compared to 0.352). So we believe that the rotational invariance property of OGCNN does help the Simple Baseline model make better predictions, as we think it helps the network generalize better to monkeys in different orientations.

Model	Pre-trained	MPJPE	PCK@ 0.02	AP@0.02
OMC rotated images	Y	0.266	0.0	0.352
OGCNN rotated images	**Y**	**0.238**	**0.0**	**0.411**

Figure 9. Comparison of results on rotated images. The OGCNN with ResNet 50 comes closest to beating the ResNet 50.

5. Conclusion

It remains relatively surprising that the OGCNN underperforms the unaltered baseline methods given that it can approximate key mathematical features of interest in just the first layer. We attribute this to a few possibilities:
1. The "zero image-gradient problem" - if a pixel neighborhood bears zero Sobel gradient magnitude, the pixel is binned into the first bin by default (arctan(0,0) = 0), whereby this orientation is arbitrarily overrepresented with "noise." To relegate this in future implementations, such pixels should be assigned to a random bin or binned by neighboring consensus.
2. Likewise, some extremely small gradients may be attributable to noise in pixel values - a region that ought to be regarded as flat assumes an arbitrary bin. Adding a small, nonlinear gradient magnitude dependency (e.g. thresholding) to the outputs of the OG layers could resolve both of these, but such may get too far into feature engineering.
3. Vanishing gradient problem: In training it seems like the loss converges very slowly and flattens after a few epochs. Changing the optimizer doesn't help very much, and although adding momentum or changing the learning rates helps achieve a dip in the loss initially, after a few epochs it struggles to converge too. The OGCNN layer weights cease changing and something akin to vanishing gradients could be an issue, despite having skip connections in our UNet architecture. While the OG outputs look to features of interest, the OGCNN may be more limited in its capabilities.
4. Network architecture: 3. raises the question of if conventional CNN architectures truly suit the OGCNN layer, i.e. if the use of OG layers removes some form of homogeny in parameter weight sharing. Future experiments could look at orienting the *entire network* along the image gradient direction, albeit the question of how to do so after maxpool operations is unclear.

In total, the OGCNN method is a mathematically innovative and sound approach, but not without flaws. Through experimentation we were able to empirically show that this method does bring us closer to rotational invariance, and helps existing state of the art models like Simple Baseline generalize better. You can find our Codalab submission under the name: *Avinash_Akella*. And the associated code at our Github repository.

6. Contributions

All authors contributed equally to the reports & presentations. AA: extensive literature review, baseline and OGCNN implementations, training, metrics, testing, result generation. RF: extensive literature review, training, result verification. RO: OGCNN conceptualization & implementations, training, testing, data preparation, theory lit. review.

References

[1] D. R. Bruno, and F. S. Osório, "Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes." *IEEE Xplore*, 2017. 1

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. "Realtime multi-person 2d pose estimation using part affinity fields." *CVPR*, 2017. 2

[3] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang. "HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation." *arXiv preprint*, 2019. 2

[4] R. Gens, P. Domingos. "Deep symmetry networks." *NIPS*, 2014. 2

[5] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[6] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov. "Learnable Triangulation of Human Pose." *International Conference on Computer Vision (ICCV)*, 2019. 2

[7] Y. Jiang, L. Chen, H. Zhang, and X. Xiao. "Breast cancer histopathological image classification using convolutional neural networks with small SE-ResNet module." *PloS one*, vol. 14, no. 3, 2019. 1

[8] Jinpyo Kim, Wookeun Jung, Hyungmo Kim, and Jaejin Lee, "CyCNN: A Rotation Invariant CNN using Polar Mapping and Cylindrical Convolution Layers." *Arxiv*, 2020. 2

[9] A. Khan, Anabia Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artif Intell Rev*, April 2020. 1

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in neural information processing systems*, vol. 25, no. 2, 2012. 1

[11] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen. "Medical image classification with convolutional neural network." *Springer Journal of Big Data*, 2019. 1

[12] S.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar. Focal Loss for Dense Object Detection. *IEEE ICCV*, 2017. 3, 4

[13] S.C. B. Lo, M. T. Freedman, S. K. Mun, H.-P.Chan. "Geared rotationally identical and invariant convolutional neural network systems." *ArXiv*, 2018. 2

[14] R. Madan, D. Agrawal, S. Kowshik, H. Maheshwari, S. Agarwal, and D. Chakravarty, "Traffic Sign Classification using Hybrid HOGSURF Features and Convolutional Neural Networks," 2019. 1

[15] D. Maturana, and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition." *IEEE Xplore*, 2015. 1

[16] R. C. W. O'Neill, et al. Computation of circular area and spherical volume invariants via boundary integrals. *SIIMS*, 2020. 3

[17] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." *LNCS,* 2015. 2

[18] S. Song, and J. Xiao, "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images." *ArXiv,* 2015. 1

[19] Y. Sun, X. Wang, and X. Tang. "Deep learning face representation from predicting 10,000 classes." *IEEE Xplore*, 2014. 1

[20] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. "Deepface: Closing the gap to human-level performance in face verification." *IEEE Xplore,* 2014. 1

[21] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. "Deep high-resolution representation learning for visual recognition." *CoRR*, abs/1908.07919, 2019 2

[22] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional Pose Machines. *CVPR*, 2016. 1

[23] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes." *CVPR,* 2015. 1

[24] Bin Xiao, Haiping Wu, and Yichen Wei, "Simple Baselines for Human Pose Estimation and Tracking." In ECCV. 2018 1, 2

[25] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013. 2

[26] Y. Yao, A. Mohan, E. Bliss-Moreau, K. Coleman, S. M. Freeman, C. J. Machado, J. Raper, J. Zimmermann, B. Y. Hayden, and H. S. Park. "OpenMonkeyChallenge: Dataset and Benchmark Challenges for Pose Tracking of Non-human Primates." *BioArXiv*, 2021. 1, 2

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." *ArXiv*, 2015. 2

[28] D. X. Zhou. Universality of Deep Convolutional Neural Networks. *arXiv preprint*, 2018. 1

[29] Y. Zhou, and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection." *ArXiv,* 2017. 1

[30] Y. Raaj, H. IDrees, G. Hidalgo, and Y. Sheikh "Efficient Online Multi-Person 2D Pose Tracking with Recurrent Spatio-Temporal Affinity Fields" *CVPR*, 2019. 1

[31] A. Toshev and C. Szegedy "DeepPose: Human Pose Estimation via Deep Neural Networks" *CVPR*, 2014. 1

[32] F. Zhang, X. Zhu, M. Ye "Fast Human Pose Estimation" *CVPR*, 2019. 1