CV For Primates

Yichao Zhou, Hongjie Li
Department of Intelligence Science and Technology, School of EECS
Peking University
2100013086@stu.pku.edu.cn, 2100013080@stu.pku.edu.cn

Abstract

Primate behavior is an important variable of interest in primatology, psychology, and biology. The application of artificial intelligence will greatly facilitate researches in this field. However, there are currently few researches about applying modern computer vision techniques to primate behavior studies. Datasets and pre-trained models for human tasks cannot directly be used in corresponding primate tasks, while datasets specifically designed for primates can be hardly found. Here we implemented the detection, identification, as well as 2D pose estimation tasks for chimps with deep learning tools. We reproduced baselines for these tasks and tried to improve the performance of our models through data augmentation, contrastive learning as well as transfer learning. Although our results are still not satisfactory, they may be of use for further research in the future. Our work may help researchers who study primates to better detect and observe their behavior, even life-long tracking.

1 Introduction

Among all the species, primates are especially notable for its similarity to human beings. Biological studies about primates can help us better learn the structure and physiological properties of ourselves. Moreover, understanding the behavior of primates is crucial for the development of psychology and cognitive science. The past decade has witnessed the great development of AI, and computer vision is now ubiquitous in our daily life. Nevertheless, there are few studies about CV for primates, thus in many influential labs where primates are studied, those primate, manual methods to record the behavior of primates are still widely used, which is both time-consuming and error-prone. However, we humans have lots of intrinsic differences to the primates, which makes it impossible to simply apply CV models and datasets for human to primate studies. To sum up, CV for primates is a promising but challenging topic.

In this report, we take chimpanzees as an example. We studied and implemented the detection, identification, as well as 2D pose estimation tasks for chimps with deep learning tools. These tasks are all core topics of modern computer vision. We first summarize the related work, then introduce our attempts on the three tasks separately, and finally make conclusions and point out further works that can be done.

2 Related work

2.1 Detection

Object detection is a classic problem in computer vision, which receives a single RGB image as input, and produces a set of objects as output. For each object detected, the output contains its category label and bounding box information. The multiple types of output and larger images than classification adds complexity to the problem. The naive solution to the problem may be using a sliding window and apply a CNN on each crop to determine whether it's a object or background, which involves too many unnecessary trials and is impossible for implementation. R-CNN[1] is a milestone in object

detection, which used a Region Proposal Algorithm to produce around 2,000 proposals for each image and then run CNN and select several regions with highest scores, however, is still very slow. Fast R-CNN[2] made improvement by firstly running the whole image through a backbone network, and use the result to do proposals and find Regions of Interest. Faster-RCNN[3] made R-CNN even faster by inserting a Region Proposal Network to predict proposals from feature maps. YOLO[4] frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities to make further improvements.

2.2 Identification

Contrastive learning is a popular self-supervised learning that can be applied to classification tasks. Here is an overview of some of the important work in contrastive learning in recent years.

MoCo v1[5] summarizes the previous results of contrastive learning and builds a large and consistent dictionary through queues and momentum encoders, which can help improve the performance of contrastive learning.

SimCLR v1[6] proposes a relatively simple contrastive learning framework, and the accuracy of the model on classification tasks is greatly improved by introducing the MLP projection head. This paper also emphasizes the importance of data augmentation for contrastive learning, and gives the data augmentation method that is most helpful for contrastive learning through a large number of experiments.

MoCo v2[7] applies the MLP projection head and data augmentation mentioned in SimCLR v1[6] to the previous work (MoCo v1[5]), and achieves better results.

SimCLR v2[8] explores larger ResNet models and makes the projection head deeper to further improve performance. But the focus of this paper is that large models obtained through self-supervised contrastive learning can be used for semi-supervised learning.

BYOL (Bootstrap Your Own Latent)[9] proposes a new self-supervised learning method. The four papers mentioned above all rely on negative samples for contrastive learning, but BYOL[9] only requires positive samples, which is equivalent to turning the matching problem into a prediction problem.

SimSiam[10] summarizes the above work and proposes a very simple Siamese network. It works well without negative samples, large batch size and momentum encoder.

As for the identification task, we take inspiration from SimCLR v2[8], and choose SimSiam[10] as the model for training.

2.3 2D pose estimation

As is known, 2D pose estimation problem is a keypoint detection problem. The keypoints for a chimp are the major body parts. If we have detected all the keypoints correctly, we can easily get the 2D pose estimation given links among those body parts. And this chimpanzee pose estimation problem is similar to the Multi Person pose estimation problem, due to the similarity between chimps and humans, and the fact that there may be several chimps in an image.

In general, previous attempts to multi person 2D pose estimation problem can be divided into two classes. One is top-down framework, which first detects bounding boxes for objects in the image, and then do pose estimation in each box independently. Such framework is used in AlphaPose [11]. The other is bottom-up methods, which first detects all the keypoints in the image, and then links the keypoints of each object together. Such framework is used in OpenPose [12].

Both the frameworks have their advantages and drawbacks. We have discovered that, the top-down framework highly depends on the quality of the bounding box detection. If some keypoints are not in the bounding box, then it is impossible for the pose estimator to find out a satisfying estimation. And if there are two bounding boxes for one single object, two groups of pose estimation will be done, so NMS is required. Meanwhile, traditional top-down methods spent plenty of time generating bounding boxes to ensure precision. AlphaPose[11] came up with several novel methods to deal with these problems, which allowed better performance and efficency.

Although the bottom-up framework doesn't rely on the bounding boxes. It only cares about detecting small local regions. One problem is that, if two target objects are very close to each other, take a mother chimpanzee holding its baby as an example, the model would probably confuse the two chimp's body parts, as they are small and hard to discriminate. OpenPose[12] introduces Part Affinity Fields to solve this problem. Nevertheless, small body parts detectors are not robust enough with frequent scale variation.

There are also some approaches which locate the target object and detect its body parts simultaneously, called one-stage approaches, hoping to achieve higher efficiency than the two-stage approaches introduced above. One example is DirectPose[13]. However, these approaches seem to perform worse than top-down framework.

As previous work mostly focused on human pose estimation, there are only a few available animal pose estimation datasets, such as Animal-Pose dataset[14] and AP-10K dataset[15]. Although the AP-10K dataset includes chimpanzee and gorilla, which may be useful to the task, these datasets consists of various animal species and are too general. MacaquePose[16] is a dataset more available for this chimpanzee task, including more than 13,000 images of macaque monkeys both from the Internet and zoos.

3 Detection

In this part, we trained a detection model using YOLOv5 v7.0 to detect all the chimpanzees in each frame. We choose YOLOv5 for its outstanding performance in most cases of object detection. Our implementation process is as follows:

3.1 Clean the data

The initial dataset we got had 1251 images for training, and 100 images for validation. After cleaning null data, we had 1188 images for training and 94 images for validation. As far as we are concerned, a few null data may be of use, as it 'tells' the model that sometimes there may be no objects, and gets the model to learn about the whole environment. However, there were too many null data in the initial dataset, accounting for 5% to 6%, which we thought is not that meaningful. Moreover, the model can still learn the environment in images with chimps. So we cleaned the dataset and removed the null data. In detail, some label files under `label_detect` are null, and we used a python script to delete these label files and the corresponding images. Since the size of null label file is 0kb, the script can easily finish the task without opening label files.

3.2 Reproduce existing baseline

YOLOv5 provides several pre-trained models, and we choose yolov5s to reproduce existing baseline. The results are as follows in Tab. 1:

Table 1: Baseline for detection

Model	Precision	Recall	AP_0.5	AP_0.5:0.95
yolov5s	0.8815	0.79549	0.87423	0.6922

3.3 Improve the performance

We first tried other pre-trained models provided by yolov5 and changed the epochs of training. The results are as follows in Tab. 2.

As is seen in the table, the model yolov5m always performs better than yolov5s in our chimp detection task. It's not unusual for the higher model complexity. And it is predictable that yolov5l and yolov5x can perform even better. However, bigger models require more training time that is impractical on our devices, and real-time detection is also crucial in applications. So we didn't try bigger models and fixed the model yolov5m for further improvements. In the table above, we also changed maximum epoch numbers and only find slight differences. It doesn't really matter whether we train 150 or 200 epoches, or even more because we observed that the model had already converged before 200 epoches.

Table 2: The performance of using different modeles and epochs. Changing epochs gives little improvement, and yolov5m performs better than yolov5s.

Model	Epochs	Precision	Recall	AP_0.5	AP_0.5:0.95
yolov5s	150	0.8815	0.79549	0.87423	0.6922
yolov5s	200	0.8872	0.79191	0.87494	0.69246
yolov5m	150	0.86052	0.84104	0.8772	0.71597
yolov5m	200	0.86603	0.80336	0.87319	0.71624

We believed that further improvement of performance needs to start with data, so we went through the dataset manually and made the following two findings. Firstly, in very few images, the chimpanzee only shows a very small part of the body. If there is no corresponding label file, we can't even find them from the image. Secondly, in a few images, the chimpanzees are extremely blurred. Some degree of ambiguity is beneficial for the learning process, however, these excessive ambiguous images probably do not improve the performance. Moreover, as for the following tasks of identification and pose estimation, these images are extremely difficult and completely of no use. After carefully going through all the images, we thought it necessary to delete some of the previously mentioned bad data. Also, a few images are incompletely labelled or incorrectly labelled. In practice, we use Roboflow(`https://app.roboflow.com/`) to process and relabel the dataset.

We have also considered improving performance through data augmentation, and our attempts are as follows in Tab. 3.

Table 3: The performance of data augmentation for detection. No significant improvement.

Model	Data augmentation	Precision	Recall	AP_0.5	AP_0.5:0.95
yolov5m	Hue+Saturation	0.89692	0.80656	0.880523	0.70952
yolov5m	Contrast+Brightness	0.91568	0.80337	0.89674	0.70761
yolov5m	Hue+Brightness	0.89023	0.80337	0.87785	0.70209

The first set of data augmentation was inspired by SimCLR v1[6]. The paper shows that at least in contrastive learning, crop+color is a better data augmentation method. Considering that YOLOv5 uses the Mosaic data augmentation method by default in training, we only changed the color of the images.

We did the latter two sets of data augmentation because we took the fact that chimpanzees are black into account. Changing the brightness of the image might not have a significant effect on the chimpanzees in the image, but it may change the background more significantly. Also, we found that if we change the contrast, the chimpanzees would be easier to be found in the image. We thought these changes can help the model better recognize the outline of chimps, which might be useful in the detection task.

We conducted all the data augumentation on Roboflow as well, where hue and saturation are randomly changed between -25% and +25%, brightness is between -10% and +10% as we found -25% and +25% performs much worse than the baseline. The training set becomes 3 times larger due to the random changes.

An image demo is showed below in Fig. 1.

4 Identification

In this part, we trained a classification model to identify chimpanzees in each frame based on the detection results. Our implementation process is as follows:

4.1 Clean the data

The format of the dataset for identification and detection is the same, and since we need to solve a classification problem, we first processed the data. For each image, we cropped the chimpanzee in the image according to the label file corresponding to the image, and resized all the extracted images to 640×640. Images were stored in folders corresponding to the name of the chimpanzee in them.

Figure 1: An image demo for detection part.

4.2 Reproduce existing baseline

We wanted the baseline to be simple and reliable, so instead of using contrastive learning, we used pre-trained models provided by YOLOv5 to reproduce existing baseline, since after the latest update, YOLOv5 can not only train the detection model, but also the classification model. The results are as follows in Tab. 4.

Table 4: Baseline for identification

Model	Image size	Batch size	Epochs	Accuracy_top1	Accuracy_top5
yolov5m-cls	640×640	16	100	0.71636	0.91636

4.3 Improve the performance

4.3.1 Attempts on contrastive learning

Just like the previous part, we first went through the dataset for identification and found two difficulties.

One of them is that the labels for identification are very limited. We only had 662 images for training and 275 images for validation, but need to classify chimpanzees into 17 categories. We believed that only supervised learning may not be able to handle this problem, and considered using contrastive learning to improve performance.

Another difficulty is that the number of different chimpanzees in the dataset is very uneven and distributed in a long tail. There are 312 images of a chimpanzee named Azibo in the dataset, accounting for a third of the total number of images. But some chimpanzees appear less than five times in the dataset. We tried to find more images of chimpanzees on the official website of the zoo, but we found that the images of chimpanzees in the current dataset were taken earlier. Compared with the images on the website, some chimpanzees have even grown up a lot, so we thought it might not be a viable idea. We hoped to use semi-supervised learning to solve this problem, because in semi-supervised

learning, unlabeled images are labeled as training data, which is equivalent to increasing sampling. Unlabeled images can be directly cropped from the video, which is not a difficult task.

In practice, we choose SimSiam[10] as the model for training, because it is new enough and the network structure is relatively simple. Our specific plan is divided into the following two parts. First, we use Simsiam and the existing dataset for training, and we can get two models, one of which is a feature extraction model and the other is a classification model. Due to the long-tail distribution of the data, the classification model may not be generalized enough. Therefore, we only keep the feature extraction model at this step. Then, we retrain a classification model through semi-supervised learning on the basis of the feature extraction model obtained in the first step. In this step, in order to prevent the initial random classification model from affecting the existing feature extraction model, a threshold can be set for the accuracy of the model. When the accuracy rate is lower than the threshold, only the classification model is trained, otherwise the whole model is trained.

Unfortunately, the first step of the plan did not go well. In the beginning, We used images of size 224×224 for training, because we believed larger images can contain more information. However, due to limited device performance, we had to set the batch size very small. This in turn made the model difficult to converge, and the accuracy rate was only 0.32. The loss curve is as follows in Fig. 2.

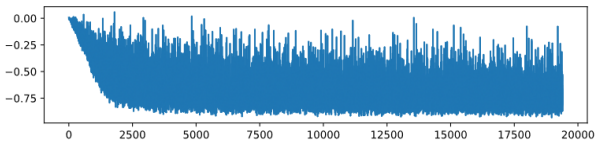

Figure 2: Loss curve of images size = 224×224 and batch size = 8.

After we resized all the images to 32×32 and set batch size to 128, the model can converge, but the accuracy rate fluctuated significantly. The loss curve and accuracy curve are as follows in Fig. 3 and Fig. 4 (the abscissa of the former is iterations, while the abscissa of the latter is epochs).

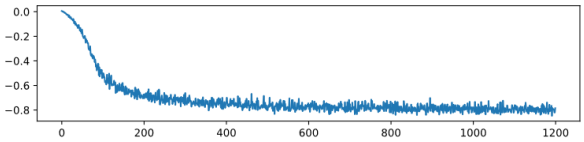

Figure 3: Loss curve of images size = 32×32 and batch size = 128.

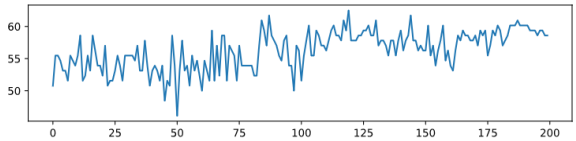

Figure 4: Accuracy curve of images size = 32×32 and batch size = 128.

We tried some adjustments to the hyperparameters of the model, but it didn't help to solve the problem of fluctuating accuracy curve. Later, we found that even with the same data and hyperparameters for training, the final accuracy on the test set fluctuated significantly. In several identical training runs, the difference between the lowest accuracy rate and the highest accuracy rate is nearly 0.1. Therefore, we suspected that the main reason for the fluctuation of the accuracy rate may be the long-tail distribution of the data.

In fact, we have always ignored the impact of the long-tail distribution of data on the first part of our plan. This is mainly because directly using the pre-trained model provided by YOLOv5 achieved a higher baseline than we expected. The highest accuracy obtained by contrastive learning is 0.53. Although the accuracy rate has improved a lot from the initial 0.32 to 0.53, it is still much lower than the baseline. We did not think such results are sufficient to support the second step of our plan, and we believed that some optimization of the results of the baseline based on the yolov5m-cls model may be an easier idea to implement, and may have better results.

4.3.2 Work based on yolov5m-cls

It should be noted that we actually tried to use yolov5s-cls as a pre-trained model, and got a higher accuracy rate than yolov5m-cls. But by comparing the training logs, we found that the test_loss of the former is larger than the latter and not as stable as the latter. We therefore speculated that using yolov5s-cls as a pre-trained model may lead to overfitting, and used yolov5m-cls instead.

Just like the previous part, we also changed maximum epoch numbers and improved the performance. The results are as follows in Tab. 5. Due to limited device performance, we could not continue to increase maximum epoch numbers.

Table 5: The performance of using different maximum epoch numbers. We achieved better accuracy_1 with epochs of 150.

Model	Image size	Batch size	Epochs	Accuracy_top1	Accuracy_top5
yolov5m-cls	640×640	16	100	0.71636	0.91636
yolov5m-cls	640×640	16	150	0.73091	0.88364

YOLOv5 provides a default data augmentation method using the `albumentations` library during training, without changing the original picture, and we did not consider other data augmentations. Specifically, images are randomly cropped and flipped horizontally with a 50 percent probability. In addition, brightness, contrast and saturation are randomly changed between -40% and +40%.

Furthermore, we had previously found that image size and batch size have a strong impact on the convergence of contrastive learning. Inspired by this, we also tried changing the image size and batch size, and further improved the performance. The results are as follows in Tab. 6.

Table 6: The performance of using different image size and batch size. Setting the image size to 224×224 and batch size to 64 can obtain better results.

Model	Image size	Batch size	Epochs	Accuracy_top1	Accuracy_top5
yolov5m-cls	640×640	16	150	0.73091	0.88364
yolov5m-cls	224×224	64	150	0.76727	0.93091
yolov5m-cls	128×128	128	150	0.76	0.90545
yolov5m-cls	32×32	256	150	0.72727	0.89818

An image demo is showed below in Fig. 5. If the two numbers are different, it means that the prediction is wrong, otherwise the prediction is correct.

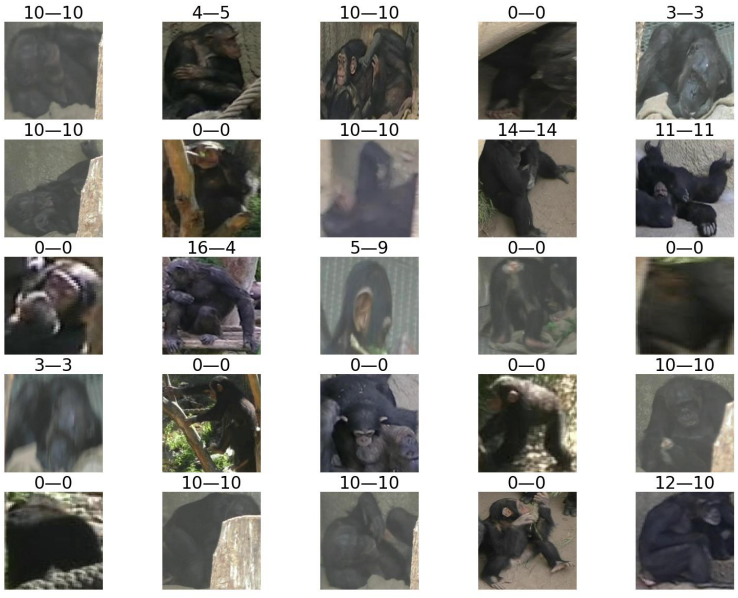

Figure 5: An image demo for identification part.

5 2D Pose Estimation

5.1 Practice with AlphaPose

Since AlphaPose performs better in human pose estimation, we firstly learned about AlphaPose through the paper [11] and online information. In practice, we first wanted to train a primate 2D pose estimation baseline using AlphaPose and the dataset MacaquePose[16]. It took us a lot of time to download the dataset because it's too big, including more than 13,000 images. Then we discovered that this dataset annotates 17 keypoints for each macaque in the images and their segmentation information, rather than bounding boxes. However, we had difficulty in converting the annotations into COCO format, which is needed in AlphaPose training, lacking useful dataset annotation tools. In addition, it's predictable that enormous time is needed to train on such a big dataset, considering our limited devices.

At last we tried to use the pre-trained models on human pose estimation for an around 30s-long video of chimpanzees cut from given data. However, the result is poor, showed in Fig. 6. Most scenes are blank, and only a few marked a hand correctly. Although chimpanzee and human are both primates, they differ a lot in color and face appearance, which makes it hard to detect a chimp using a human model.

Figure 6: A best result achieved by AlphaPose using pre-trained human pose estimation models.

5.2 Practice with MMPose

Then, one day, we happened to find that open-mmlab has a pose estimation toolbox, MMPose[17], which allows both bottom-up and top-down frameworks, and we still followed the top-down framework. To our surprise, MMPose has pre-trained models on MacaquePose. We created a video demo using the model, which can be a baseline. The video demo is better than the previous one, however, it always makes wrong prediction and is still not satisfying. See an image in the result video in Fig. 7. Since macaques and chimps bear more similarities between each other, the model can detect chimps and their keypoints, however, the precision is extremely low.

Then we realized that the image Fig. 7 above is a mistake. We just use the program `demo/top_down_video_demo_full_frame_without_det.py` in mmpose, and as its name shows, the program didn't do detection. Instead, it treated the whole image as a bounding box and directly do pose estimation, like there are only one chimpanzee. This would surely lead to a terrible result.

For further improvement, we have to make a chimpanzee 2D pose estimation dataset to conduct transfer learning. We noticed that coco-annotator is a tool for image annotation and can produce annotation files in COCO format for keypoint detection. We looked through the data provided and found 120 images which showed clear pose, which is suitable for the dataset. Then we use coco-annotator to annotate them by marking bounding boxes and 17 keypoints for crucial body parts for all

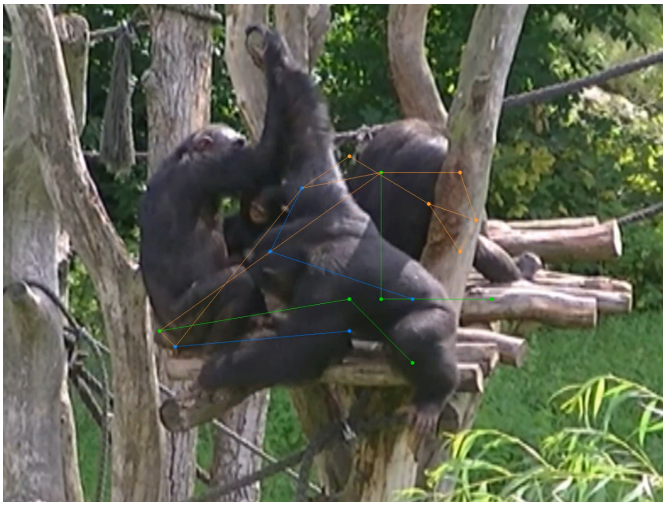

Figure 7: A image result achieved by MMPose using pre-trained models of MacaquePose, which proves to be a mistake for it treated the whole image as a chimp.

the chimps whose pose is identifiable in the images. Annotating images from scratch is a boring and time-consuming work. As our time is limited, we only annotated 60 images at last, 50 for training, 10 for validation. Because we only want to do fine-tuning on the baseline produced by MacaquePose, smaller sample size may be acceptable, and many of the images have more than one chimp, which is also beneficial for training multi-object pose estimation models rather than single-object models.

Using the dataset we made and ResNet50 as network architecture, we tested the 2D primate pose estimation baseline pre-trained by MacaquePose, with the results in Tab. 7. We also tried to train a pose estimation network from scratch. The default epoch number is 210. However, after 210 epochs of training, the model still seemed not to converge, so we added the epoch number to 300. After that, the highest AP50-95 is achieved at 280 epochs(validation are done after every 10 epochs of training, step policy for learning rate configuration, learning rate drops at 220 and 260 epoch). As is seen in Tab. 7, the results are better than the baseline. However, as the dataset is rather small, it is easy to result in over-fitting. After that, we tried to conduct fine-tuning on the baseline. Fine-tuning requires smaller learning rate and fewer training epochs, so we decrease the default learning rate from 5e-4 to 1e-4, as well as epochs from 210 to 160(step policy for learning rate configuration, learning rate drops at 120 and 150 epoch). Then the highest AP50-95 is achieved at 120 epochs, listed in Tab. 7. The results in Tab. 7 clearly show the improvement made by transfer learning. After fine-tuning, the network do perform better in pose estimation.

Table 7: The performance of MacaquePose Baseline, Training from scratch and fine-tuning on the baseline. After fine-tuning, the network perform much better, with all the metrics in the table improved. Network Architecture = ResNet50, samples-per-gpu = 16, workers-per-gpu = 2.

Model	AP_50-95	AP_50	AP_75	AR_50-95	AR_50	AR_75
Baseline	0.063	0.152	0.014	0.107	0.267	0.067
Scratch	0.339	0.705	0.220	0.367	0.733	0.267
Fine-tuning	0.511	0.857	0.465	0.540	0.867	0.533

Then we tried different network architectures. MMPose support ResNet50, ResNet 101, ResNet152, introduced by [18], as well as HrNet-w32 and HrNet-w48, introduced by [19]. Certainly, increasing the depth or width of these networks would produce better results. Using our limited devices, we only tested ResNet50, ResNet101, and HrNet-w32, listed in Tab. 8. Also, the results of macaque pose estimation using the MacaquePose baseline can be seen in the MMPose[17] documents(`https://mmpose.readthedocs.io/en/latest/topics/animal.html`), which can be a reference. All these data prove the superiority of HrNet, as HrNet carefully organizes high-to-low resolution subnetworks to maintain high-resolution representations through the whole process.

Using the model HrNet-w32, we tried to change the learning rate strategy to achieve better performance. We increased the learning rate from 1e-4 to 1.5e-4, decreased training epoch number from

Table 8: The performance of ResNet50, ResNet101, and HrNet-w32 we fine-tuned. ResNet101 is deeper and needs more training epochs. HrNet-w32 performs best among the three network architectures. Samples-per-gpu = 16, workers-per-gpu = 2.

Model	Epoch	Best AP epoch	AP_50-95	AP_50	AP_75	AR_50-95
ResNet50	160	120	0.511	0.857	0.465	0.540
ResNet101	190	160	0.524	0.733	0.733	0.547
HrNet-w32	160	110	0.553	0.861	0.658	0.567

160 to 110(using step policy for learning rate configuration, learning rate drops at 80 and 100 epoch.) The best AP_50-95 increased a bit, see in Tab. 9.

Table 9: The performance of HrNet-w32 using different learning rates and learning strategies. We achieved better AP_50-95 by changing learning rates .Samples-per-gpu = 16, workers-per-gpu = 2.

Model	Epoch	learning rate	AP_50-95	AP_50	AP_75	AR_50-95
HrNet-w32	160	110	0.553	0.861	0.658	0.567
HrNet-w32	110	110	0.576	0.857	0.640	0.593

6 Conclusion

In this paper, we reproduced baselines for detection, identification and 2D pose estimation tasks for chimpanzees and tried to improve the performance of our models through data augmentation, contrastive learning as well as transfer learning. As sophomores, we felt greatly honored to do such an interesting and challenging task. We tried our best, however, we did face lots of problems in the process, and we have to acknowledge that our research has many shortcomings. This is our first semester with professional courses, so we didn't have enough knowledge for machine learning. Most of our work is based on the existing models, such as YOLOv5 and MMPose. We modified the models and datasets to improve performance to a certain extent compared with the baselines, however, is far from satisfactory.

Further works can be done in following aspects:

- First, we can try to use more tricks of transfer learning in training in the detection part. In this part, we noticed that after data augmentation, the model learns faster at the beginning, but the final performance does not improve. We originally planned to spend some time researching the cause of this phenomenon, but eventually abandoned this plan due to more time spent on the other two parts of the task.
- Second, we are very dissatisfied with not being able to achieve a satisfactory result through contrastive learning. We think that machine learning methods that specialize in dealing with long-tailed data can be considered to improve performance.
- As for the pose estimation part, better results might be achieved by fine-tuning on a larger chimpanzee dataset. DeepLabCut[20] is another tool that worth trying.
- Finally, as mentioned above, due to the lack of machine learning knowledge, our research is based on existing models, and we have not made major optimizations and improvements to them. We need to enhance the ability to modify existing models to better fit our tasks, not just improve the performance of existing models on our tasks.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015. 1

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 2

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2, 4

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2

[8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020. 2

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 2

[10] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2, 6

[11] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 8

[12] Z Cao, G Hidalgo Martinez, T Simon, S Wei, and YA Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. in ieee transactions on pattern analysis and machine intelligence. 2019. 2, 3

[13] Zhi Tian, Hao Chen, and Chunhua Shen. Directpose: Direct end-to-end multi-person pose estimation. *arXiv preprint arXiv:1911.07451*, 2019. 3

[14] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9498–9507, 2019. 3

[15] Hang Yu, Yufei Xu, Jing Zhang, Wei Zhao, Ziyu Guan, and Dacheng Tao. Ap-10k: A benchmark for animal pose estimation in the wild. *arXiv preprint arXiv:2108.12617*, 2021. 3

[16] Rollyn Labuguen, Jumpei Matsumoto, Salvador Blanco Negrete, Hiroshi Nishimaru, Hisao Nishijo, Masahiko Takada, Yasuhiro Go, Ken-ichi Inoue, and Tomohiro Shibata. Macaquepose: A novel "in the wild" macaque monkey pose dataset for markerless motion capture. *Frontiers in behavioral neuroscience*, 14:581154, 2021. 3, 8

[17] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. `https://github.com/open-mmlab/mmpose`, 2020. 8, 9

[18] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 9

[19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 9

[20] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 2018. 10

A Appendix

A.1 Author contributions

Yichao Zhou reproduced the baseline for the identification part, and did the 2d pose estimation part.

Hongjie Li did the rest major part of identification part.

The two authors collaborated in finishing the detection part, where Yichao Zhou produced the baseline and made the dataset on Roboflow, and Hongjie Li carried out most experiments.

For the report, the two authors wrote their own parts. For other parts, Yichao Zhou wrote the abstraction and introduction part, and Hongjie Li wrote the conclusion part.