Cat Pose Estimation with MMPose Project Report

1. Introduction

Pose estimation is defined as a problem to determine the pose of an object. For pose estimation, the key is trying to detect the keypoints, which are joints for animals or people. It can be classified into two types based on dimensionality, 2D and 3D. On the other side, it can also be solved by the top-down approach and bottom-up approach. The first approach will be used in this project, which detects the objects first and then detects all the keypoints. While the latter one detects the keypoints first and combine to an object. The task is difficult to solve due to multiple reasons such that, the size of keypoints can be small and it can be difficult to detect under the influence of clothes, lights or etc.

Before the emergence of deep learning, there is no popular and common way to solve it due to the limitation of the existing graphical models. With the development of Deep Learning, especially Convolutional Neural Network (CNN), pose estimation has made tremendous progress. In 2014, Toshev and Szegedy introduced the Deep Learning method into pose estimation for the first time and CNN is used in the model. Lately, the combination of CNN and graphical method (Tompson, Goroshin, Jain, LeCun, & Bregler, 2015), and different network architecture (Newell, Yang, & Deng,2016; Carreira, Agrawal, Fragkiadaki, & Malik, 2016) have been proposed in this area. More recently, High-Resolution Network (HRNet) shows a high accuracy compared with existing methods (Sun, Xiao, Liu, & Wang, 2019). On the hand side, toolboxes for pose estimation have been established. The MMPose is a toolbox that provides a platform to train the model by different architectures and support various classical datasets for pose estimation (MMPose Contributors, 2020).

In this project, the goal is to predict the pose of cats in the 2D situation by top-down approach. To achieve this, the toolbox MMPose is applied. For the cat data, there are 17 keypoints to be detected with 1,000 training samples.

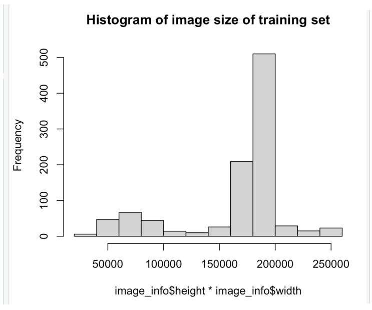
2. Method

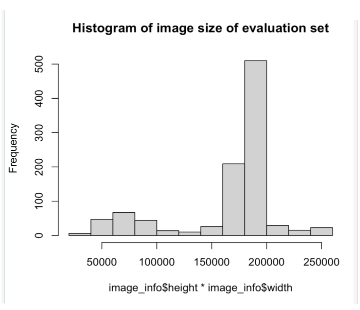
Before tuning any parameter, the mAP for the baseline configuration provided is 0.149. The adjustments are made to improve the accuracy. Firstly, the performance was improved by choosing the appropriate learning rate, 1e-4, and decreasing the batch size to 8. And the parameters in learning policy was changed to be larger values with warm iterations 500 and steps [20, 70], cooperating with increasing epoch to 70. The input image size changed to 256*256 and thus the heatmap size came to 64*64 correspondingly. Later, the neural network changed to be HRNet_w48. For data augmentation techniques, the flipping is kept being 0.5 while the rot_factor is set to be 40, and scale_factor is 0.5 in the random scale and rotation layer. And the sigma in TopDownGenerateTarget increased to 2 due to the growth of input size. With these adjustments, the mAP improved to 0.6481 in the best run. Then the pre-trained model has been adapted. With loading pre-trained hrnet_w48 in MMPose, changing the warm iteration to 50, steps to [10, 15] and using 20 epoches. the mAP increased to 0.846 dramaticly.

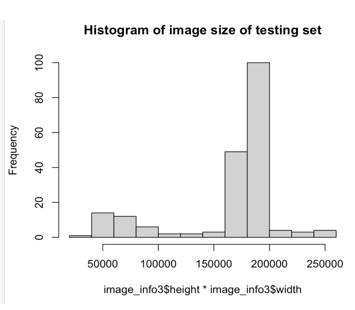
3. Experiments & Analysis

3.1 Dataset Analysis

In the training part, there are 1,000 images provided. And for the evaluation and testing set, it contains 124 and 200 corresponding. Firstly, the size of the image is explored. The sizes of images are different with various combinations of depth and width. The graphs below show the distribution of the total size, which is the multiplication of height and width. It is obvious that the size between 150,000 and 200,000 is most frequently and the distributions are similar for the three sets.

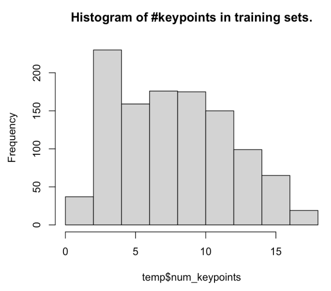




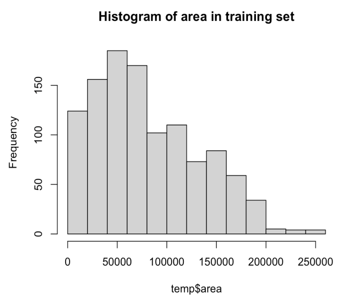


Graph 1 *Graph 2* *Graph 3*

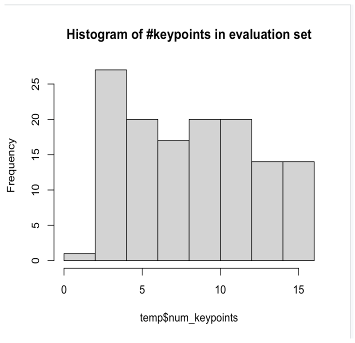
Then the JSON files are explored. In the training set, there are 1100 rows in the annotations part, as there may be more than one cat in an image. From Graph 4 which shows the histogram for the number of keypoints, we can find that there exist samples with no keypoints. And most of the samples do not contain all 17 keypoints. For the attribute, area, the distribution is right-skewed. And for the evaluation set, the situations are similar. There exist images that have more than one cat and images with zero keypoints. These special cases will be major examples in the next section of qualitative evaluation.



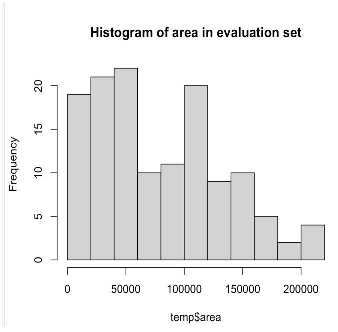
Graph 4



Graph 5



Graph 6

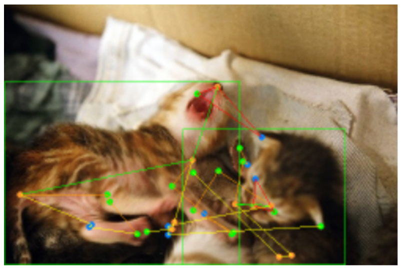


Graph 7

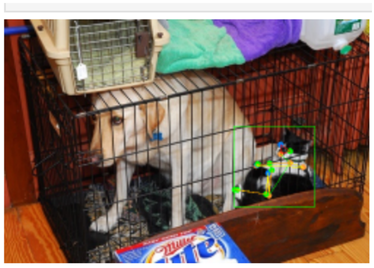
3.2 Qualitative Evaluations

As the top-down method used, the evaluation is also be discussed in two parts. For object detection, in the case with another person in the same picture (Graph 8), it successfully detects the cat and then detects key points. And for the case with two cats in one image (Graph 9), it detects both cats. For the next case with another dog in the image (Graph 10), it can detect the cat and does not misclassify the dog. Graph 11 is a special case. By checking with the annotations, the cat with a head looking from the back is not regarded as a valid cat in the data provided and the model works well.

Graph 8

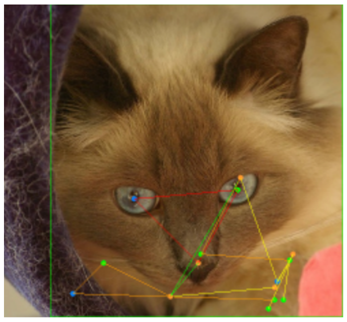


Graph 9

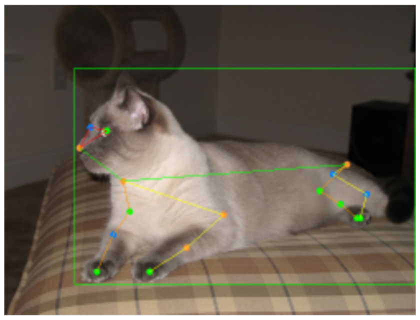


Graph 10

Then for the keypoints detection, some mistakes are shown here. For Graph 11, there are four keypoints in the annotation, however, the model can detect more than four points. And similarly, in Graph 12, there are 12 keypoints, but the model detects more than 12 keypoints.



Graph 11



Graph 12

<u>3.3 Ablation studies</u>.

Firstly, for the learning rate of the optimizer, it should be adjusted to an optimal level. The value that is too large or too small will decrease the performance as it cannot reach the optimal point efficiently. From the experiment results (Table 1), the one with 4 decimal places is the most suitable one.

learning rate	Default Setting(1e-5)	1e-3	1e-4	1e-7
mAP	0.149	0.082	0.275	0

Table 1

learning policy	Linear; 50	Linear; 100	Linear; 500	Exp; 50
mAP	0.296	0.278	0.275	0.286

Table 2

And then for the learning rate policy, we found the most efficient way is to choose linear warmup and the number of iterations can choose 50 in the current stage.

For the given epoch 20, it is found that batch size 8 would be more efficient compared to 16 and 32 as the result in Table 3 as smaller size brings more iterations. The optimal point can be found in less time. However, further reducing the batch size will increase the number of iterations and thus influence the training time. As a result, smaller numbers have not been tried. It should be noted that the batch size has a relationship with the number of epochs for the training time and convergence rate. As a smaller size will increase the training time, so currently 8 is kept and possible to be adjusted later with increasing the number of epochs to save time in experiments.

Batch size	default(16)	8	32
mAP	0.296	0.318	0.26

Table 3

Epoch	20	40	70
mAP	0.539	0.5393	0.5725

Table 4

And for the step number of learning rate changing, [20, 30] is found to be efficient and the mAP improves to 0. 342.Then for the input size, with increasing it to 256*256, change the heatmap size corresponding and increase the sigma of TopDownGenerateTarget in the data augmentation following the hint in the script, the performance improves a lot to be 0.539. With further trial to be a larger number 512*512, the performance decreases. And it increases the training time heavily. So, the size is selected to be 256*256.

Then for the training epochs, with increasing the number of epochs, the performance will increase. However, if it is set to be too large, it requires more training time and more memory space for the training log. Also, it comes with the problem of overfitting. From experiment results (Table 4), we can find that the performance improves slightly from 20 to 40 and improves a lot from 40 to 70. With further increasing the training, the performance will increase a little but the training time increases heavily. And it may come with the memory exceed problem because of the log files in the working directory. In some experiments, it's obvious to observe overfitting problems as the loss for the training dataset becomes smaller while the performance on the evaluation becomes worse. As a trade-off, the 70 is used eventually.

Further, the different architecture of the neural network can be tried. With adding the depth of ResNet to 101, and changing the pre-train to Resnet101 also, the performance improves a little. And then hrnet_w48 also been tried. It turns out that the hrnet_w48 has the best performance with 0.629. It follows the idea mentioned in part (a). The HRNet has been proved to have better performance on the pose estimation.

network	ResNet101	HRNet_w48
mAP	0.613	0.629

Table 5

flip_prob	0.5	0.7	0.3
mAP	0.633	0.592	0.626

Table 6

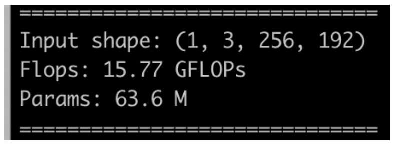
After changing the architecture of the model and increasing the total number of iterations, the learning rate policy is adjusted for the second time. From the experiments, we can find that decreasing the learning rate too early has no benefits to the model performance. After trials, the epochs to decrease the learning rate is selected to be 20 and 50 and the number of warmup iterations is changed to be 500.

For the data augmentation, the current level for the probability of implementing flip, 0.5, which is a random case, is found to be most efficient. With experiments of 0.3 and 0.7, the performance decreases. Table 6 shows the comparison of different flip_prob given other parameters are the same in one experiment. Later, the TopDownGetRandomScaleRotation layer is adjusted to be the level with rot_factor 40 and scale_factor 0.2. With this adjustment, the performance improved to be 0.646. And we also tried to add one more TopDownHalfBodyTransform layer, with parameters num_joints_half_body equals 8 and prob_half_body equals 0.3. The experiment shows that it decreases the performance by about 0.03 in one experiment.

Finally, we tried to adopt pre-trained models. With loading the pre-trained HRNet_w48 model in MMPose by load_from in Line 3, the performance improved a lot. At the same time, it is found that with using the pre-trained model, the total epochs, the learning policy can be changed also. Large values are not required to achieve high performance. Thus, we use 20 epochs and the warmup iteration moved back to 50. The epochs to change the learning rate changed to 10 and 15. The performance improved to 0.8462. The reason why we can get such a high performance as we use adequate pre-trained model, so the initial weights are more efficient and eliminates the training and fine-tuning.

3.4 Model Complexity & Runtime Analysis

Graph 13 shows the model complexity with flops and parameters by the tool provided by MMPose.



```
==============================
Input shape: (1, 3, 256, 192)
Flops: 15.77 GFLOPs
Params: 63.6 M
==============================
```

Graph 13

Reference

Carreira, J., Agrawal, P., Fragkiadaki, K., & Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4733-4742).

MMPose Contributors(2020). OpenMMLab Pose Estimation Toolbox and Benchmark. https://github.com/open-mmlab/mmpose.

Newell, A., Yang, K., & Deng, J. (2016, October). Stacked hourglass networks for human pose estimation. In *European conference on computer vision* (pp. 483-499). Springer, Cham.

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5693-5703).

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 648-656).

Toshev, A., & Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1653-1660).

Yang, Y., & Ramanan, D. (2012). Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence, 35(12), 2878-2890.*