Assignment 1: Cat Pose Estimation with MMPose

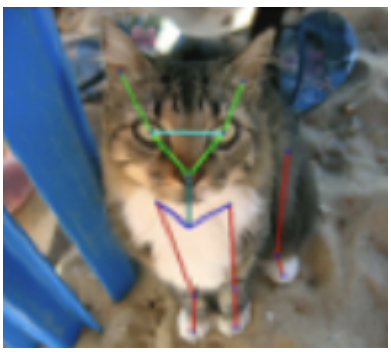
1 Introduction

In this assignment, you are going to develop a 2D cat pose estimation system based on MMPose. The system will detect all the body keypoints (or joints) of pet cats with state-of-the-art deep learning techniques.

1.1 What is 2D animal pose estimation?

2D animal keypoint detection (animal pose estimation) aims to detect the key-point of different species, including horses, dogs, macaques, and cheetah. It provides detailed behavioral analysis for neuroscience, medical and ecology applications.

In this assignment, we mainly focus on predicting the poses of pet cats. We predict 17 keypoints for each pet cat, e.g. eyes, ears, paws. The detailed description of these 17 keypoints can be found in Section 3.2.



1.2 What is MMPose?

MMPose is an open-source toolbox for pose estimation based on PyTorch. It is a part of the OpenMMLab project. MMPose is one of the most popular pose estimation frameworks. It supports a wide spectrum of mainstream pose analysis tasks, and various state-of-the-art pose models.

Learn more about MMPose at
https://github.com/open-mmlab/mmpose (https://github.com/open-mmlab/mmpose) and
https://mmpose.readthedocs.io/en/latest/ (https://mmpose.readthedocs.io/en/latest/)

1.3 What will you learn from this assignment?

The goals of this assignment are as follows:

1. Understand the basics of deep learning based pose estimation.
2. Gain experience with a major deep learning framework (PyTorch) and a powerful open-source library for pose-related tasks (MMPose).

3. Develop a deep learning system from scratch, including model design, model training and model testing.

2 Setup

You can work on the assignment in one of two ways: on a virtual machine on HKU GPU Farm, or locally on your own machine.

2.1 Working remotely on HKU GPU Farm (Recommended)

Note: after following these instructions, make sure you go to Working on the assignment below (i.e., you can skip the Working locally section).

As part of this course, you can use HKU GPU Farm for your assignments. We recomment you to follow the quickstart provided by the official website (https://www.cs.hku.hk/gpu-farm/quickstart) to get familiar with HKU GPU Farm.

After checking the quickstart document, make sure you have gained the following skills:

1. Knowing how to access the GPU Farm and use GPUs in interactive mode. We recommend using GPU support for this assignment, since your training will go much, much faster.
2. Geting familar with running Jupyter Lab without starting a web browser.
3. Knowing how to use tmux for unstable networks connections.

```
# Prepare basic environments

conda create -n open-mmlab python=3.7 -y
conda activate open-mmlab
conda install ipykernel
ipython kernel install --user --name=kernel_for_mmpose
pip install jupyterlab
pip install ipywidgets widgetsnbextension pandas-profiling
```

2.2 Working locally

If you have the GPU resources on your own PC/laptop. Here's how you install the necessary dependencies:

(Recommend) Installing GPU drivers: If you choose to work locally, you are at no disadvantage for the first parts of the assignment. Still, having a GPU will be a significant advantage. If you have your own NVIDIA GPU, however, and wish to use that, that's fine – you'll need to install the drivers for your GPU, install CUDA, install cuDNN, and then install PyTorch. You could theoretically do the entire assignment with no GPUs, though this will make training much slower.

Installing Python 3.6+: To use python3, make sure to install version 3.6+ on your local machine.

Virtual environment: If you decide to work locally, we recommend using virtual environment via anaconda for the project. If you choose not to use a virtual environment, it is up to you to make sure that all dependencies for the code are installed globally on your machine.

3. Working on the assignment

3.1 Learn about pose estimation

Before we start, let's learn some basics about 2d pose estimation. Here is a good blog (https://nanonets.com/blog/human-pose-estimation-2d-guide/) for you to have a general idea of 2d human pose estimation.

If you are interested, please read the review paper (https://arxiv.org/pdf/2012.13392.pdf) for more details.

Generally speaking, 2d pose estimation can be categorized into top-down and bottom-up approaches. Top-down approaches first use an object detection model to detect the bounding box of each instance, and then apply pose estimation model to detect keypoints. Bottom-up approaches first detect all the keypoints in the image, and then group the keypoints into instances.

3.2 Task description

In this assignment, you are going to train a top-down 2d cat pose estimation model with MMPose. We have provided the ground-truth bounding boxes of the train/val/test dataset. So you do not need to care about the object detection part. Our goal is to train a pose estimation model.

You will be provided with a cat dataset. The dataset can be found here (https://github.com/jin-s13/COMP7606). The dataset file is named as "Cats.tar". Please download and extract it using the command:

```
tar -xvf Cats.tar
```

You will get two directories:, i.e. "images" and "annotations".

"images" contain 1,328 images in total. 1,000 images for training, 128 images for validation and 200 for testing.

"annotations" contain ground-truth annotations. The annotation file follows the COCO-type. Please read COCO (https://cocodataset.org/#format-data) to learn more about the COCO data format.

We have provided full annotations for train/validation set, including the ground-truth bounding boxes and the ground-truth keypoint annotations. But for test set, "test_info.json", we only provide the ground-truth bounding boxes, and the keypoints are all ones (placeholders). In your experiments, you may use the train/val split to tune your model. And finally, you need to submit the prediction results of the test set. (More information about the submission can be found in Section 3.5.)

Each cat is annotated with 17 keypoints. The detailed description of these 17 keypoints are listed as follows:

Keypoint ID	Keypoint Name
0	L_Eye
1	R_Eye
2	Nose
3	Neck
4	Root of tail
5	L_Shoulder
6	L_Elbow
7	L_F_Paw
8	R_Shoulder
9	R_Elbow
10	R_F_Paw
11	L_Hip
12	L_Knee
13	L_B_Paw
14	R_Hip
15	R_Knee
16	R_B_Paw

'L' means left; 'R' means right; 'F' means front; 'B' means back.

3.3 Get Started

We have provided the colab notebook `Tutorial.ipy` for you to begin with.

https://github.com/jin-s13/COMP7606/blob/main/Tutorial.ipynb (https://github.com/jin-s13/COMP7606/blob/main/Tutorial.ipynb)

It contains a baseline model training script. Please follow the detailed instruction to (1) install MMPose (2) prepare data (3) run the model training (4) visualize (5) run the model testing (6) prepare for submission.

The training data and the validation data (images + full annotations) are provided for you to test your model's performance locally. You may use the performance on the validation set to tune your model training pipeline.

The test data (images only & without annotations) are also released. You need to submit the result json of the test data.

3.4 Improve the baseline model

IMPORTANT: Please do NOT modify the core inference & evaluation codes of MMPose. e.g. `$MMPOSE/mmpose/core/` . It may lead to scoring errors.

There are lots of ways to improve the baseline model provided in the `Tutorial.ipy` . Here are some suggestions for you.

3.4.1 Hyper-parameter tuning

There are lots of important hyper-parameters.
(1) About optimizer: learning rate, batch size, warm-up and training epochs, etc.
(2) About model: Input size, gaussian heatmap size, etc.

3.4.2 Different neural network architectures

You may choose a better neural network architecture (ResNest, HRNet) from MMPose, or you may also design your own customized neural networks.

3.4.3 Try different data augmentation techniques

Common augmentation techniques include random scaling/rotation/flipping. There are also many hyper-parameters (the rotation factor, the scaling factor) to tune.

You may also try more advanced data augmentation techniques.

3.4.4 Weight initialization

The weight initialization of a neural network can affect how fast the model converges, how good the converged point is, and if the model converges at all.

Transfer learning is a deep learning technique whereby a model is developed for one task but then re-used as a starting point for a separate task. This technique helps leverage the pre-trained weight of that model in our new model. Besides, transfer learning saves a lot of time as training a deep learning model from scratch requires a lot of training data and computing power.

It is allowed to use any pre-trained weights, but please describe it in the report.

3.4.5 Try different optimization methods

Popular optimizers are SGD, RMSProp, and ADAM. MMPose has supported multiple types of optimizers.

3.4.6 Others

There are also many other possible techniques to explore. But please make sure do not change the evaluation API. And make sure TAs can run your codes successfully.

Good Luck!

3.5 What to submit? (4 items in total)

1. Prepare a final report in PDF format (no more than 4 pages)

a) Introduction

```
Briefly introduce the task & background & related works.
```

b) Methods

```
Describe what you did to improve the baseline model performance.
For example, this may include but not limited to:
(i) Hyper-parameter tuning, e.g. learning rate, batch size, and training epochs.
(ii) Different neural network architectures. You may choose a better architecture
(ResNest, HRNet) from MMPose, or you may also design your own customized neural
networks.
(iii) Try different data augmentation techniques, random scaling/rotation/flipping.
There are also many hyper-parameters (the rotation factor) to tune.
(iv) Model pre-training.
```

---

c) Experiments & Analysis (*IMPORTANT!*)

```
Analysis is the most important part of the report. Possible analysis may include but
(i) Dataset analysis (dataset visualizations & statistics)
(ii) Qualitative evaluations of your model.
Visualize the results of some challenging images and see if your model accurately
predicts the poses. Failure case analysis is also suggested.
(iii) Ablation studies. Analyze why better performance can be achieved when you
made some modifications, e.g. hyper-parameters, model architectures,
data augmentation. The performance (AP) on the validation set should be given to
validate your claim.
(iv) Model complexity & runtime analysis. MMPose has provided a useful tool for
this. You may choose to analyse it.
```

---

2. Codes

a) All the codes.
b) **README.txt** that describes which python file has been added/modified.

3. Models

a) Model checkpoint (.pth)
b) Model config file (.py)
c) Model training log (XXX.log)

4. Result json on the test set

a) The result json obtained by model inference on the test set.

```
# Please organize all these items in a zipped directory. Please rename the zipped f
```

```
123456.zip
├── report.pdf
├── codes
|    ├── README.txt
|    ├── mmpose
|        ├── mmpose
|            ├── tools
|            ├── configs
|            ...
|── models
|    |-- model.pth
|    |-- config.py
|    |-- training.log
|
├── result_keypoints.json
```

3.6 When to submit?

The deadline is March 6th (Sunday).

Late submission policy:

10% for late assignments submitted within 1 day late.
20% for late assignments submitted within 2 days late.
50% for late assignments submitted within 7 days late.
100% for late assignments submitted after 7 days late.

3.7 Need More Support?

If you encounter any questions about the MMPose, please feel free to raise an issue at https://github.com/open-mmlab/mmpose/issues (https://github.com/open-mmlab/mmpose/issues). Don't worry, the MMPose team will help to solve your problems.

For any questions about the assignment, please contact the TA (Sheng Jin) via email (js20@connect.hku.hk (mailto:js20@connect.hku.hk)).

Marking Scheme:

The marking scheme has two parts, (1) the performance ranking (70% marks) and (2) the 4-page final report (30% marks):

1. For the performance ranking part (70%): TA will rank the mAP. The students who obtain higher mAP will get higher marks.
   (1) Rank top-10% will get the full mark of this part.
   (2) Rank 10%-20% will get 90% mark of this part.
   (3) Rank 20%-30% will get 80% mark of this part.
   (4) Rank 30%-40% will get 70% mark of this part.
   (5) Rank 40%-50% will get 60% mark of this part.
   (6) Others will get 50% mark of this part.

2. For the 4-page final report part (30%):
   The marks will be given mainly based on the richness of the experiments & analysis.
   (1) Rich experiments + detailed analysis: 90%-100% mark of this part.
   (2) Reasonable number of experiments + analysis: 70%-80% mark of this part.
   (3) Basic analysis: 50%-60% mark of this part.
   (4) Not sufficient analysis: lower than 50%.

Reference

1. MMPose Contributors. Openmmlab pose estimation tool-box and benchmark.https://github.com/open-mmlab/mmpose (https://github.com/open-mmlab/mmpose), 2020
2. Zheng, Ce, et al. “Deep learning-based human pose estimation: A survey.” arXiv preprint arXiv:2012.13392 (2020).
3. Sun, Ke, et al. “Deep high-resolution representation learning for human pose estimation.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
4. Xiao, Bin, Haiping Wu, and Yichen Wei. “Simple baselines for human pose estimation and tracking.” Proceedings of the European conference on computer vision (ECCV). 2018.