MMPose Installation Guide for Windows (11/16/2023)

1. Gather GPU information

Find your graphics card type by going to Task Manager -> Performance -> GPU. If it is not an NVIDIA card, you cannot use it. (You probably can’t use any \*GPUs that are a decade old either.)

Example: NVIDIA GeForce GTX 1070

Go to the following website and find your CUDA capability based on your card:

<https://developer.nvidia.com/cuda-gpus>

If your capability is less than 3.5, you will not be able to use your GPU.

Example: 6.1

Find your driver version by opening NVIDIA control panel -> System Information

Example: 536.40

Swap to the components tab and find NVCUDA64.DLL. This is the CUDA version.

Example: NVIDIA CUDA 12.2.79 driver

(Updating your driver may allow you to use a newer version of CUDA. For a list of all CUDA versions you can use, go to <https://docs.nvidia.com/deploy/cuda-compatibility/>. If your CUDA version is below 10.2, you will not be able to use the \*GPU)

1. Set up environment

Download and install Miniconda from the official website:

<https://docs.conda.io/projects/miniconda/en/latest/>

Open Anaconda Powershell Prompt (miniconda3)

Create a conda environment:

conda create –name openmmlab python=3.8 -y

conda activate openmmlab

(This will create an environment that will hold all of the tools needed to run MMPose. Activating it lets you access the tools from any directory. If you ever close and reopen Anaconda Powershell Prompt, you MUST conda activate openmmlab again. The following command only work when it is active.)

1. Install pytorch and torchvision (and torchaudio)

While the environment is activated, install the correct version of pytorch and torchvision with the correct version of cudatoolkit. Go to <https://pytorch.org/get-started/locally/> or <https://pytorch.org/get-started/previous-versions/> and use pip install. Select CPU if your CUDA capability or version is too low.

Example: pip3 install torch torchvision torchaudio –index-url https://download.pytorch.org/whl/cu121

1. Verify cuda compatibility

Enter the following

python

import torch

torch.version.cuda

torch.cuda.is\_available()

torch.backends.cudnn.enabled

If the version is correct, and the functions returns True, next we verify that CUDA sees our GPU:

device = torch.device(‘cuda’)

torch.cuda.get\_device\_properties(device)

And finally, we check if torch can use CUDA:

torch.tensor([1.0, 2.0]).cuda()

Example: tensor([1., 2.], device=’cuda:0’)

(if cuda.is\_available() returns False, you have installed the wrong version of CUDA or torch. If torch.tensor().cuda() gives an error, it means torch cannot access your GPU.)

1. Install MMEngine and MMCV

exit() the python script, returning to your active conda environment.

Install mmengine:

pip install -U openmim

mim install mmengine

Next, go to <https://mmcv.readthedocs.io/en/latest/get_started/installation.html>

Under the section ‘Install with pip’, put in your version of OS, CUDA, and torch to generate the correct version of MMCV

Example: Windows, cuda 12.1, torch 2.1.x, mmcv2.1.0

Then we can install that version of mmcv and mmpose using mim:

mim install “mmcv>=2.1.0”

mim install “mmpose>=1.1.0”

(if you could not find any versions of mmcv in the drop down, try mim install “mmcv>=2.0.1”)

MMPose requires Python 3.7+, CUDA 9.2+, and PyTorch 1.8+

For use with GPU, CUDA 10.2+ is required.

MMCV, a necessary component of MMPose has various builds which contain cuda and torch. See the following link under ‘Install with pip’ for pre-built packages:

<https://mmcv.readthedocs.io/en/latest/get_started/installation.html>

To make sure your version of cuda and torch work together, search for it on pyTorch’s page:

<https://pytorch.org/get-started/locally/> (or <https://pytorch.org/get-started/previous-versions/> for older builds)

1. Run mmpose

Edit mmposetest.py and uncomment if name == main, testmmpose()

With the environment activated, navigate to PoseMatcherBackend and run the command:

python mmposetest.py

On the first run, 2 or 3 files will be download. These will be used as a checkpoint to run the inference faster. You can ignore the warnings, as they are not harmful.

If the image pops up with keypoints overlaid, everything is working normally. Re-comment the if name == main, testmmpose() section.

1. Install flask

With the environment activated, just run the command:

pip install Flask

pip install -U flask-cors

1. Run backend locally.

python PoseMatcherBackendDemo.py

This will run the local flask backend on the address produced.

Example: Running on <http://127.0.0.1:5000>

Go to the website to confirm that it is live.

1. Simulate a frontend request

Open Command Prompt

navigate to PoseMatcherBackend's file

in Command Prompt, enter the command:

python frontsimulation.py testimages/salute.jpg testimages/selfie.jpg

(if command prompt can’t find python, remember that it was installed with miniconda. Go to ‘edit the system environment variables’ -> Environment Variables and under the System variables section, edit PATH. Add the new path C:\Users\ericr\miniconda3)

A new file should pop up in the folder called encoded\_images.txt. In Command Prompt, run the following command:

curl -X POST -H "Content-Type: application/json" -d @encoded\_images.txt http://127.0.0.1:5000/upload

The Anaconda Powershell Prompt should run MMPose, and then Command Prompt should dump a bunch of random characters.

In the templates folder, edit image\_verification and paste the extremely long string of data where it says <COPY RAW DATA HERE>. (from /9j/4AAQ to KKAP//Z without the quotes)

Then double click image\_verification to see the image.

\*if running without a GPU, select a version of pytorch that doesn’t require GPU. Also edit mmposetest.py in the analyze() and testmmpose() functions, remove device=’cuda’.