COMPUTATIONAL IMAGING: COMPRESSIVE SENSING AND CONVEX OPTIMISATION PERSPECTIVES

PYTHON ENVIRONMENT INSTALLATION GUIDE

This course involves intensive programming tasks requiring specific Python environments for the effective execution of computational algorithms. This guide outlines the steps to install, configure, and utilize Python environments tailored for both CPU and GPU-based systems. By leveraging tools such as Miniconda and the provided configuration files, students can ensure consistency across platforms, enabling seamless execution of lab assignments and course projects.

1 Installing Miniconda

Miniconda is a lightweight package manager that simplifies the management of Python versions and dependencies across various platforms. Using Python installations provided by the system can sometimes interfere with the operating system or lead to version conflicts. Miniconda provides an isolated environment, ensuring stability and flexibility for managing different Python setups.

1.1 Quick Installation Commands

If you haven't installed a Conda-based environment manager on your system, Miniconda can be installed using the following commands. These commands will download the Miniconda installer and install it with the default settings in silent mode. For detailed instructions and troubleshooting, refer to the official Miniconda installation webpage.

1.1.1 If you are using Windows

Please open the Command Prompt as an administrator and run the following commands:

```
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe -o miniconda.exe
start /wait "" .\miniconda.exe /S
del miniconda.exe
```

After running these commands, you should be able to find "Anaconda Prompt (miniconda3)" and "Anaconda Powershell Prompt (miniconda3)" in the Start Menu. Open "Anaconda Prompt", you'll see "(base)" in the command prompt, which means the Conda environment is activated.

1.1.2 If you are using macOS

If your system uses Apple Silicon, please open the terminal and run the following commands:

```
mkdir -p ~/miniconda3
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh -o \
    ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh
```

If your system uses an Intel chip, replace "arm64" in the above command with "x86_64". After running these commands, you can reopen the terminal or directly activate Conda by running:

```
source ~/miniconda3/bin/activate
```

You can then see "(base)" in the command prompt. The Conda command can be initialised on all available shells by running:

```
conda init --all
```

It will modify shell configuration files (e.g., .bashrc, .zshrc) and let all new terminal windows have Conda activated automatically.

1.1.3 If you are using Linux

Please open the terminal and run the following commands:

```
mkdir -p ~/miniconda3
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -o \
    ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh
```

After running these commands, you can reopen the terminal or directly activate Conda by running:

```
source ~/miniconda3/bin/activate
```

You can then see "(base)" in the command prompt. The Conda command can be initialised on all available shells by running:

```
conda init --all
```

It will modify shell configuration files (e.g., .bashrc) and let all new terminal windows have Conda activated automatically.

1.2 Verifying Miniconda Installation

After installing Miniconda, verify the installation by running:

```
conda --version
```

If the command outputs the version number, the installation was successful.

2 Creating the Required Conda Environment

Once Miniconda is installed, you can create the required Python environment for the course. This section provides the full environment configuration files for both CPU and GPU setups and the command for creating the environment.

2.1 Environment configuration files

Conda environments can be quickly created using YAML files that specify the required dependencies and configurations. In this course, PyTorch and related libraries are used for computational imaging tasks.

Because PyTorch has different dependencies for CPU and CUDA systems, two separate YAML files are provided.

2.1.1 Configuration file for CPU systems

You can create a YAML file in your current directory named `b31xo_cpu.yml` with the following content:

```
name: b31xo_cpu
channels:
    - pytorch
    - conda-forge
    - defaults
dependencies:
    - python=3.11
    - numpy=2.0.1
    - scipy=1.14.1
    - matplotlib=3.9.2
    - matplotlib-inline=0.1.6
    - pytorch=2.5.1
    - torchvision=0.20.1=py311_cpu
    - torchaudio=2.5.1=py311_cpu
    - pywavelets=1.7.0
    - scikit-image
    - tqdm
    - notebook
    - ipykernel
    - ipywidgets
    - pip
    - pip:
        - ptwt==0.1.9
```

2.1.2 Environment for CUDA Systems

CUDA devices can greatly accelerate the training and inference of PyTorch. If your system has CUDA devices, you can create a YAML file `b31xo_cuda.yml` with the following content:

```
name: b31xo_cuda
channels:
    - pytorch
    - conda-forge
    - defaults
dependencies:
    - python=3.11
    - numpy=2.0.1
    - scipy=1.14.1
    - matplotlib=3.9.2
    - matplotlib-inline=0.1.6
    - pytorch=2.5.1
    - pytorch-cuda=12.4
```

```
  - torchvision=0.20.1=py311_cu124
  - torchaudio=2.5.1=py311_cu124
  - pywavelets=1.7.0
  - scikit-image
  - tqdm
  - notebook
  - ipykernel
  - ipywidgets
  - pip
  - pip:
      - ptwt==0.1.9
```

This YAML file assumes the CUDA driver on your system is compatible with CUDA 11.8. Please update the YAML file accordingly if a different CUDA version is desired.

2.2 Launching Conda

To use Conda, you must first activate it on your system:

- **Windows:** Open the Command Prompt (or Anaconda Prompt) and type:

```
conda activate
```

- **macOS/Linux:** Open a terminal and type:

```
source ~/miniconda3/bin/activate
conda activate
```

2.3 Creating the environment

Use the provided YAML files to create the Python environment tailored for your system.

2.3.1 For CPU systems

1. Go to the directory of file `b31xo_cpu.yml`.
2. Create the environment using:

```
conda env create -f b31xo_cpu.yml
```

3. Activate the environment:

```
conda activate b31xo_cpu
```

2.3.2 For CUDA systems

1. Go to the directory of file `b31xo_cuda.yml`.
2. Create the environment using:

```
conda env create -f b31xo_cuda.yml
```

3. Activate the environment:

```
conda activate b31xo_cuda
```

2.4 Verifying the Environment

Once the environment is activated, verify the installation with the following commands.

2.4.1 For all systems

```
conda list
python -c "import torch; print(torch.__version__)"
```

2.4.2 For CUDA systems

```
python -c "import torch; print(torch.cuda.is_available())"
python -c "import torch; print(torch.cuda.get_device_name(0))"
```

If CUDA is correctly configured, the output should display `True` and the name of your GPU.