DEPLOYMENT AND CONFIGURATION OF YOLOV11 FOR OPERATIONAL USE

• Virtual Environment Setup and YOLOv11 Configuration

It is highly recommended to create a virtual environment for each project to manage dependencies and prevent conflicts between different library versions. While Python's default package manager, `pip`, can be used to create virtual environments, we will utilize `conda` to set up the virtual environment for our YOLOv11 project. Below are the steps for installing the YOLO11 environment:

1. **Anaconda/Miniconda Installation (Optional)**: If `conda` is not already installed, you need to install either the Anaconda or Miniconda software. In our case, the system we were using already had Anaconda installed, so there was no need for a separate installation.

2. **Create a Folder and Initialize the Environment**: First, create a folder to organize all the datasets, YOLO variants, Python files, and the virtual environment. Open this folder in PyCharm, VS Code, or any other coding platform, and then launch the bash terminal from within the platform. Alternatively, you can directly open a terminal from the folder itself.
   To create and initialize the environment, use the following command:

   ```
   conda create -p environment_name
   ```

   For instance, we created a folder named YOLO11, opened the terminal, and ran the command:

   ```
   conda create -p yolo11_env
   ```

   This created the yolo11_env environment folder inside the YOLO11 folder.

3. **Activate the Environment:** After creating the environment, activate it to ensure that all libraries and dependencies are installed exclusively within this environment. Use the following command to activate the environment:

   ```
   conda activate environment_name
   ```

   For example, we activated our environment with the following command:

   ```
   conda activate yolo11_env
   ```

4. **Install the Libraries:** While you can install libraries one by one, it is recommended to install them together in a single command. This allows the Conda package manager to resolve any potential dependency conflicts efficiently.
   - **Without GPU or if GPU is not required:** To install the ultralytics library without GPU support, use the following command:

     ```
     conda install -c conda-forge ultralytics
     ```

   - **With NVIDIA GPU (CUDA Environment):**
     If you plan to use an NVIDIA GPU, you need to install the environment in a CUDA-supported setup. Use the following command:

```
conda install -c pytorch -c nvidia -c conda-forge pytorch torchvision pytorch-cuda=CUDA_VERSION ultralytics
```

To determine the CUDA_VERSION for your system, run: `nvidia-smi`

Ensure that the CUDA version installed in the environment is equal to or lower than your device's CUDA version.

- Note: For installing CUDA versions equal to or lower than 11.3, replace `pytorch-cuda=CUDA_VERSION` with `cuda-toolkit=CUDA_VERSION`.

5. **Dataset:** The dataset folder should follow a specific structure. The data.yaml file within the folder contains the paths to all the images (train, validation, and test sets). Additionally, it defines the classes that the model will be trained on and the classes it will predict.

```
dataset
|
+---data.yaml
|
+---test
|   +---images
|   +---labels
+---train
|   +---images
|   +---labels
+---valid
    +---images
    +---labels
```

6. **Variants:** Different variants of the model (such as n, s, m, l, x) can be pre-downloaded and stored in the variants folder. These variants can then be used in the code as needed.

• EXPORT TO DIFFERENT SYSTEM

We can export our environment to a different system with the help of following steps:-

1. **Creation of .tar.gz archive:** Firstly we will pack our environment into .tar.gz file (Used for Linux and MacOs Systems) using `conda-pack`. Below are the steps for it.
    a. **Install conda-pack (If not already installed):** Activate the base environment and install conda-pack:-

```
conda activate base

conda install -c conda-forge conda-pack
```

    b. **Pack the Environment into a .tar.gz file:** Use the conda pack command and specify the output file with the .tar.gz extension:

```
conda pack -n your_env_name -o your_env_name.tar.gz
```

We have used the following command:-

```
conda pack -n yolo11_env -o yolo11_env.tar.gz
```

2. **Splitting (Optional):** Try transferring the environment file to your target machine, if there are any problems faced related to file size, you can try splitting the file into smaller parts with the help of following command:-

```
split -b 1G environment_name.tar.gz part_
```

We have used the following command: `split -b 2G yolo11_env.tar.gz part_`

3. **Transferring:** Above command will create files like part_aa, part_ab, etc. Transfer these parts to the target machine through pen drive or anything else, and then reassemble them on your target machine using:

```
cat part_* > environment_name.tar.gz
```

We have used the following command by after opening the terminal in which the parts were present:-

```
cat part_* > yolo11_env.tar.gz
```

4. **Extract the packed environment:** Use the following command to extract the environment:-

```
tar -xzf <packed_environment>.tar.gz -C <target_directory>
```

Replace <packed_environment>.tar.gz with your file name. Replace <target_directory> with the path where you want to extract the environment.

We used the following command:-

```
tar -xzf yolo11_env.tar.gz -C yolo11_env
```

5. **Navigate to the extracted directory:** Use the following command:-

```
cd <target_directory>/<environment_name>
```

We have used the following command:-

```
cd yolo11_env
```

6. **Fix paths using the conda-unpack script:** Fixing the paths are the main reason for using conda-pack and conda-unpack. It can be done using following command:-

```
./bin/conda-unpack
```

7. **Activate the environment:** To use the environment, activate the environment by the following command:-

```
source bin/activate
```