HW2: Model Inference of the Pretrained LeNet

We have trained a CNN model similar to LeNet with post-training quantization.

In this homework, we are going to implement the functional inference model in a high-level programming language, Python, and adjust the bit-width of the partial sums in each layer.

Action Items:

- [ ] Implement a high-level functional model for each layer of the CNN, including convolution, pooling, and fully-connected layers with 8-bit quantization of the input activations, output activations, and weights accordingly.
- [ ] Pass all unit tests in `nnunittest.py`.
- [ ] Fill in all TODOs in `homework2.ipynb`, `nnutils/LeNetModel.py`, and `nnutils/functional.py`.
- [ ] Answer all questions in `homework2.ipynb`.

How to launch Jupyter Notebook?

Choose either Option 1 or Option 2. If you are familiar with Jupyter Notebook, simply launch `homework2.ipynb` and start writing your homework. Make sure you have placed `homework2.ipynb`, `nnunittest.py`, and `nnutils` folder in the same directory.

```
\nnutils
    \LeNetModel.py
    \functional.py
    \...
\homework2.ipynb
\weights
\activations
\scale_hw.json
\...
```

Option 1: with Google Colaboratory on the Cloud

1. Open your Colab
2. Unzip hw2.zip and upload homework2.ipynb to Colab.
3. Upload external `nnutils` folder, `nnunittest.py`, and `parameters.zip`.

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

   - Or you can mount those files via Google drive. Check this link for detailed information.
   - Remember to unzip `parameters.zip` by `!unzip parameters.zip`.
4. Run `!pip install numba==0.55.1` in Colab before using it.

- We don't want to install `Numba` again or upload any files when checking your homework. Comment out all comments you use in Step 3 and Step 4 before submitting your homework.

Option 2: with Conda on your computer

1. Install miniconda
2. Create a Conda virtual environment

```
conda create --name vlsi
conda activate vlsi
```

3. Install the following packages for this homework

```
conda install -c conda-forge matplotlib
conda install -c anaconda jupyter
conda install -c numba numba
```

4. Type `jupyter notebook` and launch Jupyter Notebook!
   - Remember to unzip `parameters.zip` by `!unzip parameters.zip`.

What do I need to submit?

1. Make sure you have done everything in `LeNetModel.py` and `functional.py` under the `nnutils` folder and `homework2.ipynb`.
   - Remember to unzip `parameters.zip` by `!unzip parameters.zip`.

```
\nnutils
    \LeNetModel.py
    \functional.py
    \...
\homework2.ipynb
\weights
\activations
\scale_hw.json
\...
```

2. We don't want to install `Numba` again, upload/download any files to Colab, or retrain any models again when checking your homework. Comment out those lines of code for those processes!

```
# from google.colab import files
# uploaded = files.upload()

# for fn in uploaded.keys():
#     print('User uploaded file "{name}" with length {length} bytes'.format(
#         name=fn, length=len(uploaded[fn])))
...

# files.download(...)
...

# !pip install numba
```

3. Click `Kernel` and then click `Restart Kernel & Run All` on the Jupyter Notebook of `homework2.ipynb`.
    - Make sure everything goes smoothly without any warining or error messages while running your `homework2.ipynb`!
4. Upload `functional.py`, `LeNetModel.py`, `parameters.zip`, and `homework2.ipynb` to EECLASS. Do not zip these files or put them in a folder! Simply upload these four separate files.

Troubleshooting

Reloading modules

You might need to run and modify `functional.py`, `LeNetModel.py`, and `homework2.ipynb` back and forth. If you have edited the module source file using an external editor and want to try out the new version without leaving the Python interpreter, you should reload these modules.

There are two alternatives:

- Autoreload
    - IPython extension to reload modules before executing user code.
      `autoreload` reloads modules automatically before entering the execution of code typed at the IPython prompt.

```
%load_ext autoreload
%autoreload 2
```

    - Check this link.

- Reload
    - reload() reloads a previously imported module.

```
import importlib
importlib.reload(module)
```

    - Check this link.

Sometimes there may be the following error message:

```
super(type, obj): obj must be an instance or subtype of type
```

The straightforward solution is to restart the kernel and run it all.

- Check this link.