ECE 449/590, Fall 2022
Project 3: Matrix and Tensor Operaions

Due: 10/23 (Sun.), by the end of the day (Chicago time)

1 Summary

In this project, we will implement a few matrix and tensor operations for EasyNN in C++. In particular, we will implement the following EasyNN operations.

- `Input(n)`: if the input `n` is a scalar, it will be passed via `add_kwargs_double()`. Otherwise `n` is a tensor and will be passed via `add_kwargs_ndarray()`.
- `Const(c)`: if constant `c` is a scalar, it will be passed via `add_op_param_double()`. Otherwise `c` is a tensor and will be passed via `add_op_param_ndarray()`.
- `a+b` and `a-b`: if `a` and `b` are both scalars, this is scalar addition or subtraction. Otherwise `a` and `b` are tensors with the same shape. Perform element-wise addition and subtraction to generate the result tensor of the same shape. Note that matrix is a tensor of 2 dimensions.
- `a*b`: if `a` and `b` are both scalars, this is scalar multiplication. If one of `a` and `b` is a scalar and the other is a tensor, multiply the scalar to each element of the tensor to generate the result tensor of the same shape. Otherwise `a` and `b` are both matrices with correct shapes so that `a*b` are well defined for matrix multiplication.

This project should be done individually. Discussions are encouraged. However, all the programs (except those from the lectures) and writings should be by yourself. COPY without proper CITATION will be treated as PLAGIARISM and called for DISCIPLINARY ACTION.

NEVER share your programs/writings with others.

2 Working with Your Projects

Please continue to work with your Git repository for Project 3. Here is a brief introduction of the files.

- `easynn.py`, `easynn_golden.py`, `easynn_cpp.py`, `Makefile`: same as those in Project 1 and 2. You should not modify them.
- `src`: this directory contains all your C++ implementations. Update them as needed.
- `easynn_test.cpp`: continue to use this file to test and to debug your C++ implementations. Start with a test that is as simple as possible and then move to more complicated cases.
- `grade_p3.py`: this is the grading script to verify whether your C++ implementations in `src` are correct or not. There are 10 questions. You should not modify this file.

After creating the shared library using "make", run the grading script to see if all questions pass.

```
make
python3 prj03_grade.py
```

3 Deliverables and Grading

We obtain a copy of all your source files in `src` as you push the changes to the central Git repository so there is no need for you to submit them to us using any other mechanisms. Moreover, please be advised that since to learn the use of Git and a CI system is among the objectives of this course, we will NOT accept project submissions outside the central Git repository, e.g. via emails. If you have difficulty accessing the central Git repository, it is your responsibility to act promptly to seek help from us well before the project deadline; otherwise, not able to access the central Git repository is NOT an excuse for late submissions.

Project 3 will have a full grade of 100 points. Each function, if passed, will give you 10 points. Since you are required to use the CI system to troubleshoot any issues with your code before the deadline, a failed function will earn 0 points.

The following submission checklist is provided for your convinience. Detailed instructions are available from Section IV of Guide to System Setup and Work Flow.

- ☐ Run `python3 grade_p3.py` in VM to make sure all 10 tests pass.
- ☐ Commit and push your changes to the central Git repository.
- ☐ Run `/home/ece449/show` on uranus.ece.iit.edu to access your grading report and correct any issues before the deadline.