WebGPU Image Super Resolution Milestone 2

Paul (San) Jewell
Andrew Zhu

Project Overview

- A WebGPU based image super resolution program
- Input is fed to a neural network to generate the output
- Essentially a neural network inference engine

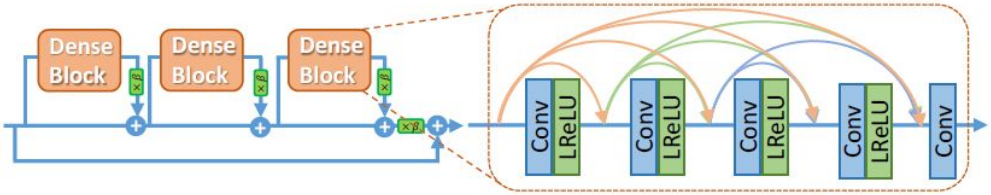
General Progress

- Reimplemented prototype with test cases in python ; verified match to pytorch with <1E-6 error
- Naive implementation in webGPU completed and verified except the last three layers

Super Resolution Neural Network Architecture

1. Input: RGB image (3 * height * width)
2. First Convolution Layer (Channel 3 -> 64)
3. 23 x Residual in Residual Dense Block

Residual in Residual Dense Block (RRDB)



Where we are

4. Second Convolution + Residual Layer
5. 2 x Super Resolution Layer (width x 2, height x 2)
6. 2 x Super Resolution Layer (width x 2, height x 2)
7. Last Convolution Layers (Channel 64 -> 3)
8. Output: RGB image (3 * (4 * height) * (4 * width))

Progress - Python

Why?

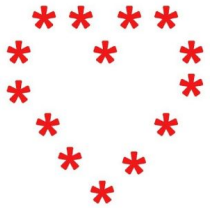
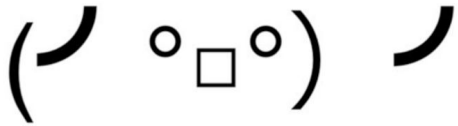
- ~400 layers to check, many chances for implementation errors, easier debugging.
- Easy to try a performance improvement idea without writing GPU code

Completed

- Completely unrolled original pytorch version to individual layers
- Made a simulated ‘cuda-like’ version with pure python (numpy arrays without slicing tricks), with a test case to compare to the Pytorch version

Challenges

- Browser localstorage insufficient (5MB?!) Needed to make this a bit more robust ; used IndexedDB instead. (New API I didn't know of!)
- Just… LOTS of small issues trying to reproduce what pytorch does exactly.
- WebGPU is very tedious compared to even CUDA, with lots and lots of boilerplate similar to vulkan. Some errors (which show as warnings) do not appear on google _at all_ and I must step through each source line to see which causes the problem. (looking at you "binding sizes too small for bind group" >.>) ; why do uniforms need to be strided?


Goals for Next Milestone

- Second, faster implementation
  - Layer fusion
  - Reduce number of CPU-GPU memory swaps
  - Saturate GPU computation
  - Naive 2D convolution vs matrix multiply 2D convolution
  - Float 32 to Float 16
- User interface prettiness with modern progress indicator
- Retrain on a few other data sets and compare

References

- https://web.dev/gpu-compute/
- https://github.com/austinEng/webgpu-samples
- https://github.com/xinntao/ESRGAN