WebGPU Image Super Resolution

•••

Fast model inference on the client

Overview

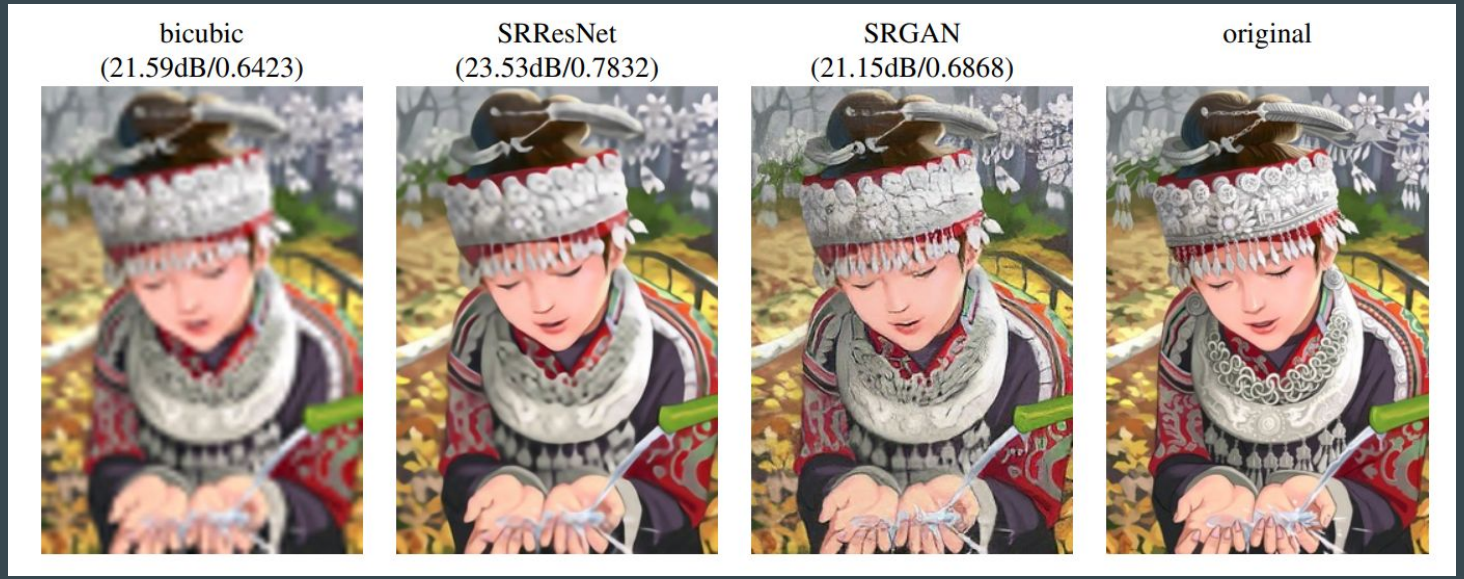
- Deep neural network models currently trained and deployed on company servers, not as scalable

- Client side inference greatly reduces the costs to companies and distributes load to clients. Clients can enhance their own experience by upgrading their hardware. Also more efficient to avoid transit / bandwidth.
    - A step further: presenting a huge number of models (the space is blowing up) and have the user select their favourite visually

- Example case study: Photopea photo editor. Author only hosts one static javascript application and pays for basic web hosting -- no server side work, and clients love this app!

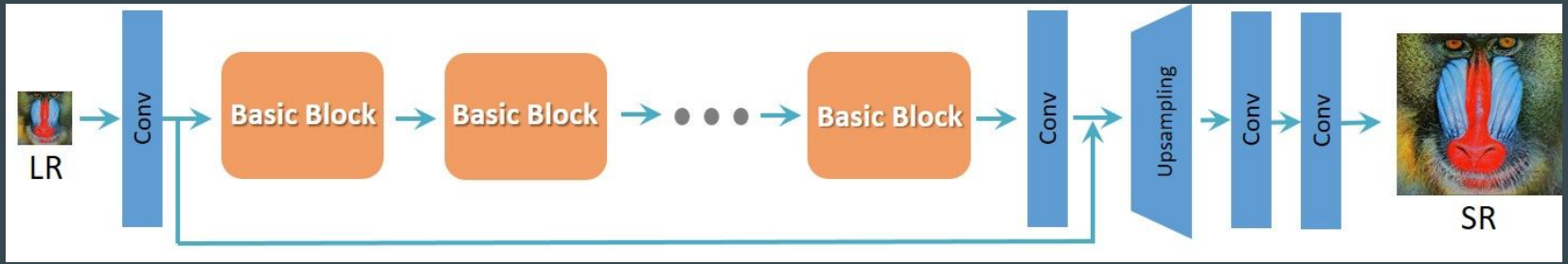
WebGPU Image Super Resolution

- A browser program that can enhance and enlarge images on the web.



WebGPU SuperSampler Architecture

- CPU: Javascript (WebAssembly as a secondary goal)
- GPU: WebGPU
- Libraries: WebGPU Blas
- Functionality: User upload low resolution images/select from preloaded images -> Supersampler generate a high resolution image in real time



Reference Implementation?

- There is a parallel library which seems to have a similar, general purpose goal: MIL Webdnn (https://github.com/mil-tokyo/webdnn) ; however, it is only an alpha version and doesn't seem to be completely open source (?)
- We may not be able to use it as a reference.
- Our implementation will aim to be specific to one chosen architecture and built for speed, rather than parse ONNX.

Schedule

- Milestone 1
  - Set up environment (Javascript)
  - Test webGPU framework (Hello Triangle)
  - Choose a specific super sampler architecture and digest it / verify that the pytorch or other framework model is working to the standards that we want.
- Milestone 2
  - Baseline implementation of super sampler inference on webGPU
  - Unit test each layer implementation
- Milestone 3
  - Optimize implementation for speed
  - Explore webassembly for faster CPU execution
- Final Presentation
  - Performance analysis

Potential Drawbacks

- Trained models can be relatively bandwidth-heavy on the first page load, but will be cached for all uses after that.
- User experience more heavily impacted by their hardware.
- Non-generalized model may not be worth the performance trade off, we will have to see. There is also not a great baseline for comparison (webDNN is alpha).

References

1. MIL WebDNN (mil-tokyo.github.io)
2. WebAssembly
3. milhidaka/webgpu-blas: Fast matrix-matrix multiplication on web browser using WebGPU (github.com)
4. xinntao/ESRGAN: ECCV18 Workshops - Enhanced SRGAN. Champion PIRM Challenge on Perceptual Super-Resolution. The training codes are in BasicSR. (github.com)
5. idealo/image-super-resolution: 🔎 Super-scale your images and run experiments with Residual Dense and Adversarial Networks. (github.com)
6. Bigjpg - AI Super-Resolution Image lossless enlarging / upscaling tool using Deep Convolutional Neural Networks
7. Super Resolution API | DeepAI