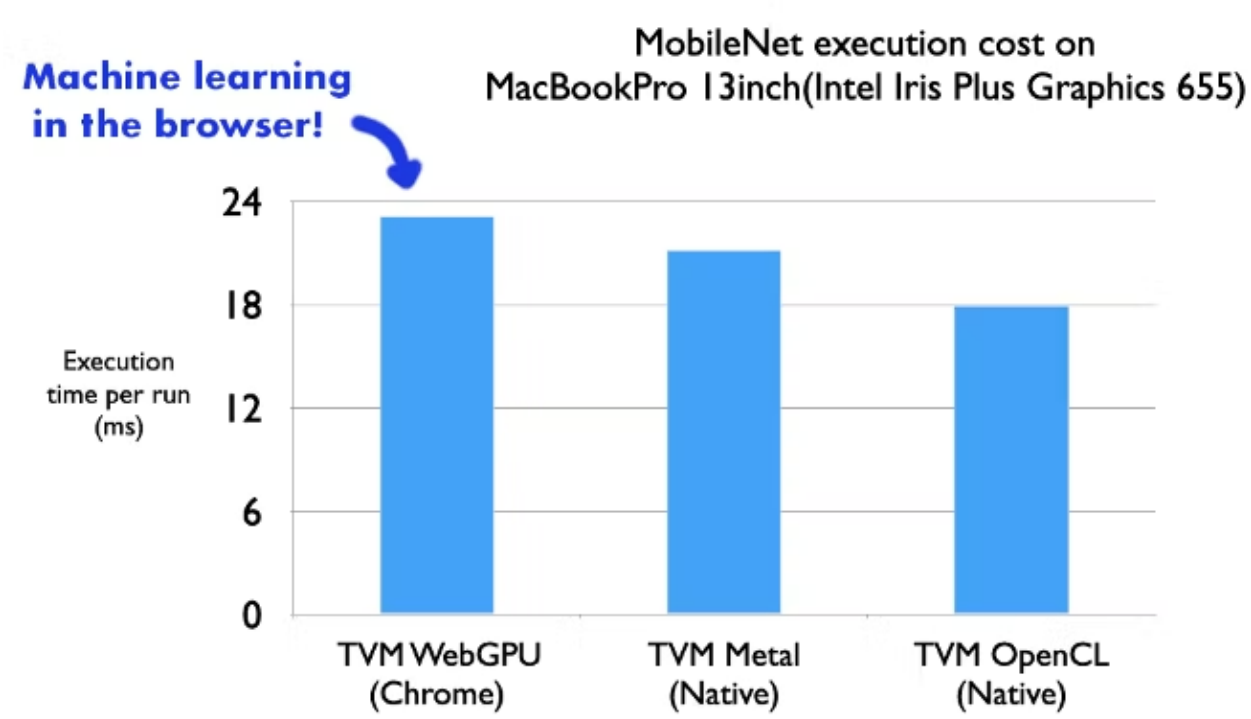
OctoML

Jason Knight May 14, 2020 4 minutes



In this article

By Tianqi Chen, Jared Roesch, and Jason Knight

TL;DR

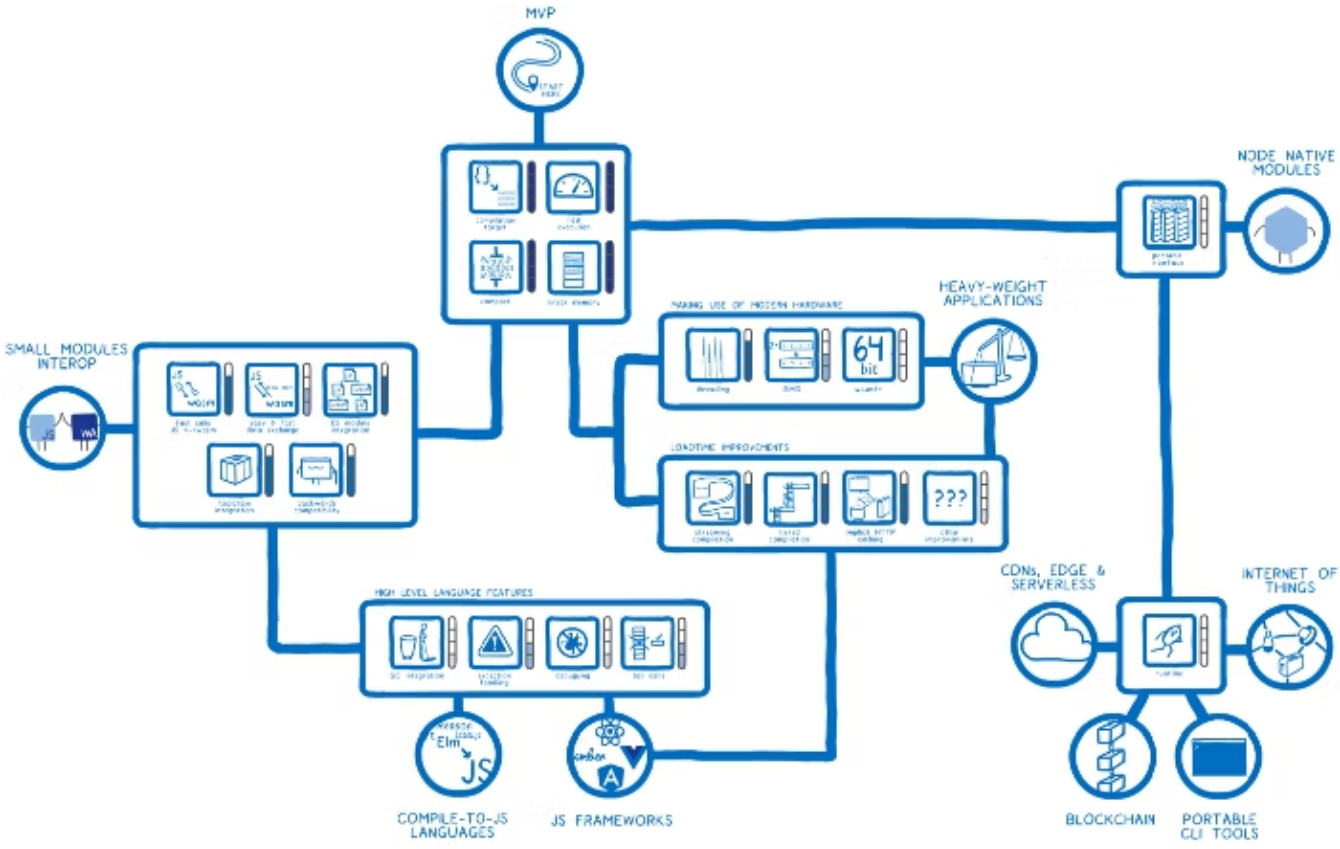
Easy ML deployment in the browser with near-native GPU performance and WebAssembly

We introduced support for WASM and WebGPU backends to the Apache TVM deep learning compiler Our initial experiments shows that TVM's WebGPU backend can get **close to native GPU performance** when deploying models in the browser.

Introduction — faster browser applications

WebGPU and WebAssembly are two up-and-coming browser standards aimed at making browser based applications as efficient as native applications while retaining hardware portability. WebGPU first builds on top of the lower level graphics API movements to enable more full featured GPU accelerated compute capabilities as compared to its predecessor WebGL, and WebAssembly (WASM) is a similar effort for unlocking CPU based compute capabilities. And best of all, WASM is most likely already enabled in the browser you're reading this in, with WebGPU coming soon.

For some of the exciting capabilities and use cases behind WASM particularly, please check out Lin Clark's excellent WASM "skill tree"



WASM's post MVP skill tree

Leveraging more of these WASM capabilities such as SIMD support and multi-threading as they mature is an exciting area of future work, but let's see how far we can get with WASM and WASI (standard interfaces) to feed WebGPU compute shaders.

Machine learning in the browser

While efforts like TensorFlow.js and ONNX.js attempt to bring machine learning to the browser, there still exist non-trivial gaps in performance between the web versions and native ones. One of the many reasons behind these gaps are the lack of standard and performant access to the GPU on the web. WebGL lacks

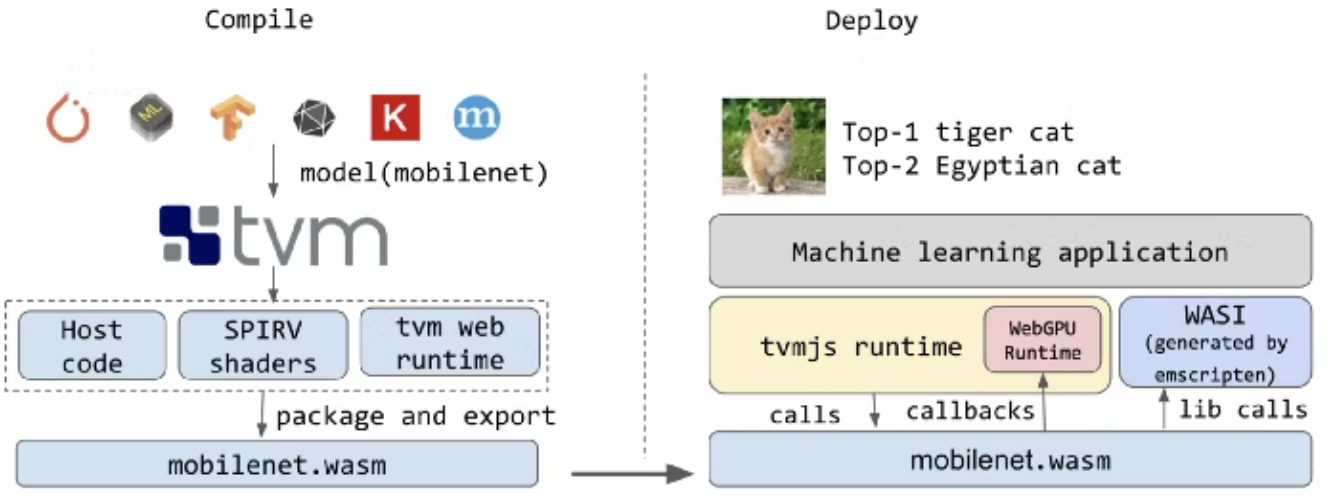
important features such as compute shaders and generic storage buffers that are necessary for high performance deep learning.

WebGPU offers the potential to bridge this gap with its first-class compute shader support.

To explore the potential of using WebGPU for deploying machine learning in the browser, we enhanced the deep learning compiler Apache (incubating) TVM to target WASM (for host code that computes the launching parameters and calls into the device launch) and WebGPU (for device execution). Our preliminary results are quite positive — for the first time, we can deploy machine learning applications on the web while still getting near native performance on the GPU.

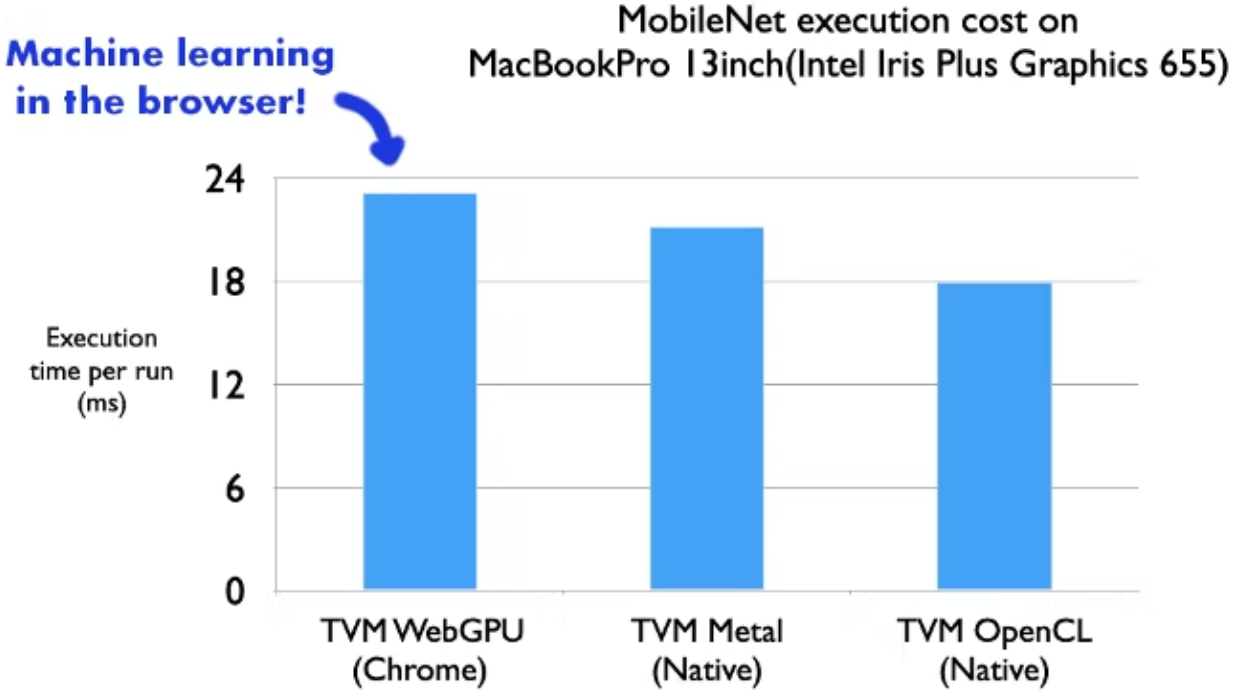
Please check out our companion Apache TVM technical blog post for the technical details, but in short, we were able to achieve near native GPU performance in the browser through WASM and WebGPU by combining:

- TVM's existing SPIR-V code generation capabilities to target WebGPU compute APIs
- TVM's existing LLVM backend for it's WASM generation support
- A new tvmjs TypeScript runtime
- WASM's standard interfaces (WASI) for system library calls (eg malloc, stderr)
- Chrome's nightly WebGPU support



One important advantage of the compilation based approach is the reuse of infrastructure. We are able to effortlessly relative to other approaches target the web by reusing the infrastructure for optimizing GPU kernels for native platforms such as CUDA, Metal and OpenCL. If the mapping of the WebGPU API to native APIs is efficient we can expect similar performance with very little work. More importantly, the AutoTVM infrastructure allows us to specialize the compute shaders for specific models, enabling the generation of the best compute shaders for our specific model of interest.

Performance



We ran a quick experiment comparing the execution of a full Mobilenet execution via TVM's WebGPU backend and native targets that use native GPU runtimes (Metal and OpenCL). We find that WebGPU get's quite close to matching the performance of Metal. Assuming Chrome WebGPU's runtime targets Metal instead of OpenCL on the MacOS, we can safely assume there is little to no performance loss when targeting the GPU through the browser here.

We also expect further performance boosts by applying AutoTVM tuning on the GPU being used by the browser since these results are based off previously auto-tuned schedules from a GTX 1080 Ti, which is quite different from the Intel graphics GPU. To make this easier, we expect our upcoming improvements to the uTVM runtime (stay tuned to this blog for an update in the next 1–2 weeks) for embedded microcontrollers will help make autotuning easier in the browser since WASM targets share many similarities to embedded devices.

Looking to the Future

Our results suggest many interesting opportunities for machine learning on the web. Notably, WebGPU is an API that is still evolving and its implications could

go beyond web applications. For example one could target native APIs of WebGPU as it matures and becomes standardized through WASI, enabling standalone WASM applications that make use of WebGPU.

The TVM community is also actively working on a Rust based runtime that would enable much more robust WASM support and enable easier interaction with projects like wgpu, and the Rust WASM ecosystem. As an open source project, we are looking for contributors who can bring in new ideas and help push the project in these exciting directions.

The proposed approach provides effective machine learning support for most WASM's application scenarios. The close to native performance could also unlock better federated learning capabilities on the browser. The same compiled package should also be able to run on native WASM executors to provide sandbox for the applications.

Show me the Details/Code

- Detailed technical version of this blog post
- Example project for image classification
- Apache TVM on github

Acknowledgement

We would like to thank the emscripten project for providing the WASM compilation infrastructures as well as the JS library support on the web. We would also like to thank the WebGPU community for various helpful discussions. Thanks to Fletcher Haynes for valuable feedback on the post.

Share

OUR BLOG

Related Posts

ALL POSTS

2020-12-16

On the Apple M1, Beating Apple's Core ML 4 With 50% Model Performance Improvements

Apple's release of an Arm-based chip, called the M1, was a seismic shift in the personal computing landscape.

Sayce Falk

2021-03-04

Up to 9x performance improvements with TVM's new auto-scheduler

Autoscheduling enables higher performance end to end model optimization from TVM, while also enabling users to write custom operators even easier than before.

Jason Knight

OctoML

Models

Stable Diffusion

Llama 2

Whisper

Resources

Docs

Blog

Apache TVM

Company