TensorFlow.js

Opportunities & Challenges for TensorFlow.js and beyond

Jason Mayes

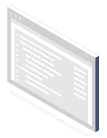
Senior DA - **TensorFlow.js**, Google

@jason_mayes

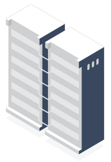
W3C Workshop 2020

TensorFlow.js is an open source library for **machine learning in JavaScript**.

ML in the browser / client side means **lower latency, high privacy, and lower serving cost**. We also support **Node.js** server side for larger more complex models.

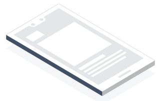


Browser



Server

node

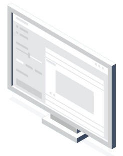


Mobile

React Native

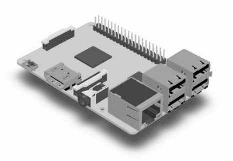
WeChat

PWA



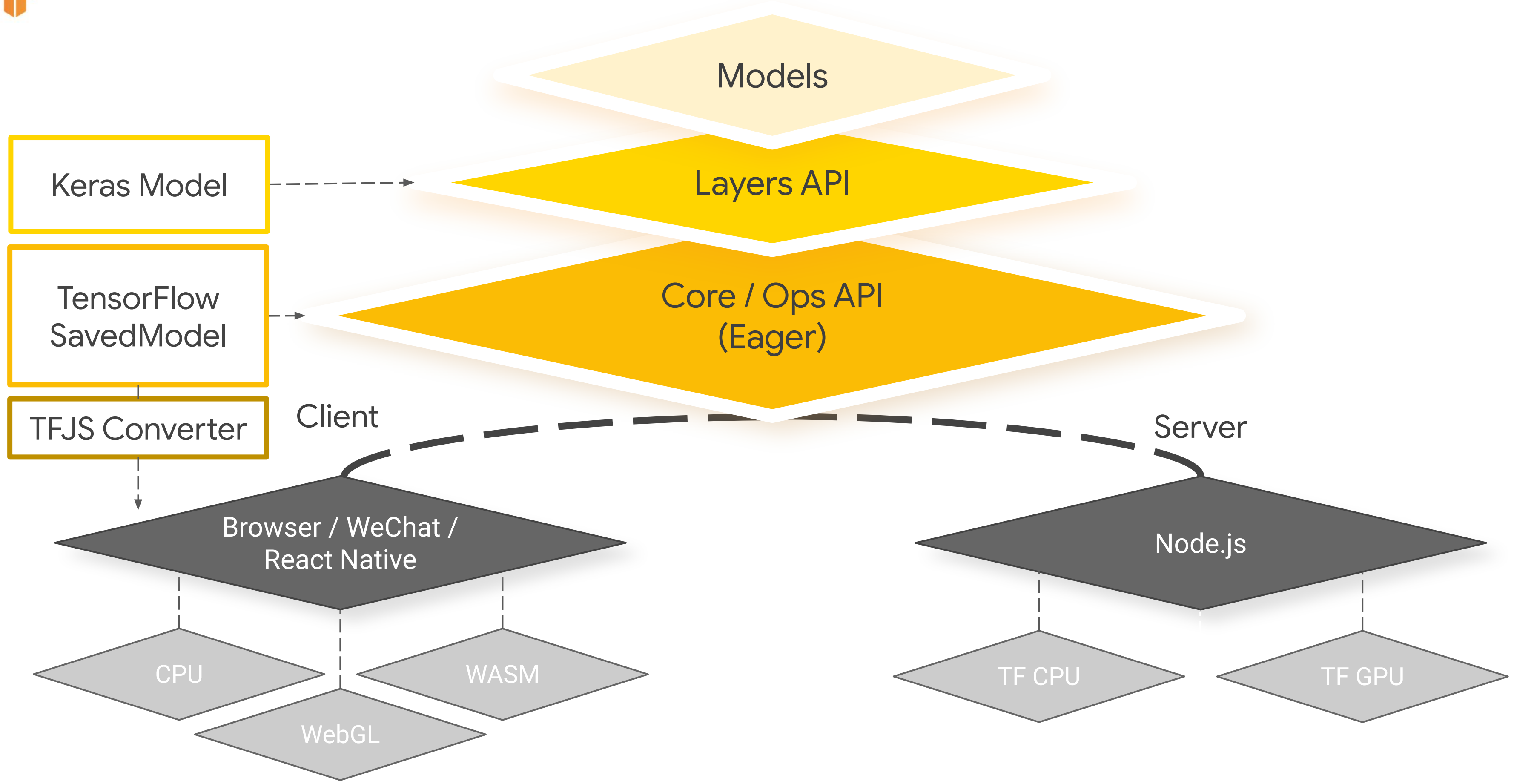
Desktop

Electron



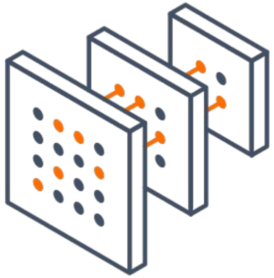
IoT

RaspberryPi
(via Node)


Models
Layers API
Core / Ops API (Eager)
Keras Model
TensorFlow SavedModel
TFJS Converter
Client
Server
Browser / WeChat / React Native
Node.js
CPU
WASM
WebGL
TF CPU
TF GPU

3 key user journeys

Run, Retrain, Write

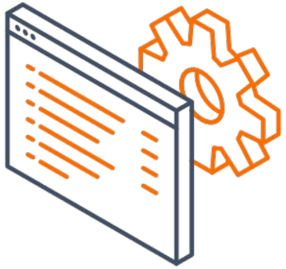


Run existing models

Pre-packaged JavaScript **or** Converted from Python

Retrain existing models

With transfer learning



Write models in JS

Train from scratch

For anything you may dream up

Augmented Reality

Gesture-based interaction

Sound recognition

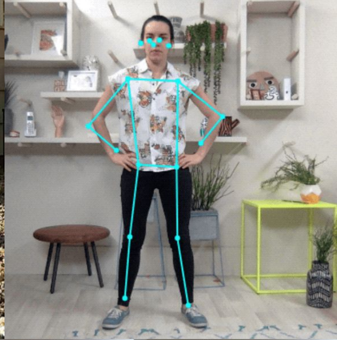
Accessible web apps

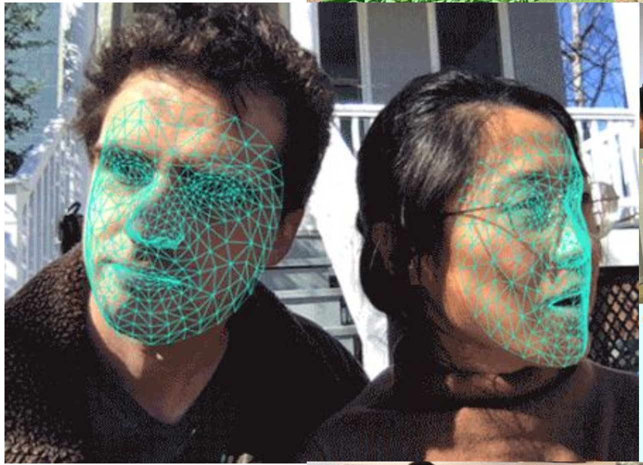
Sentiment analysis, abuse detection, NLP

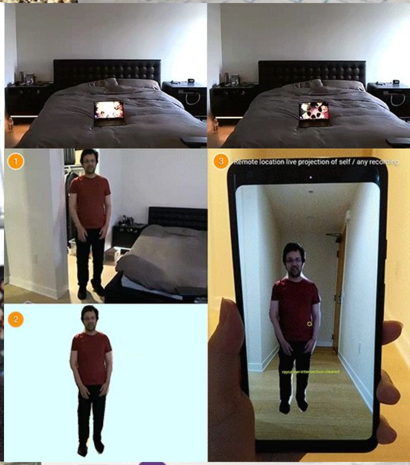
Conversational AI

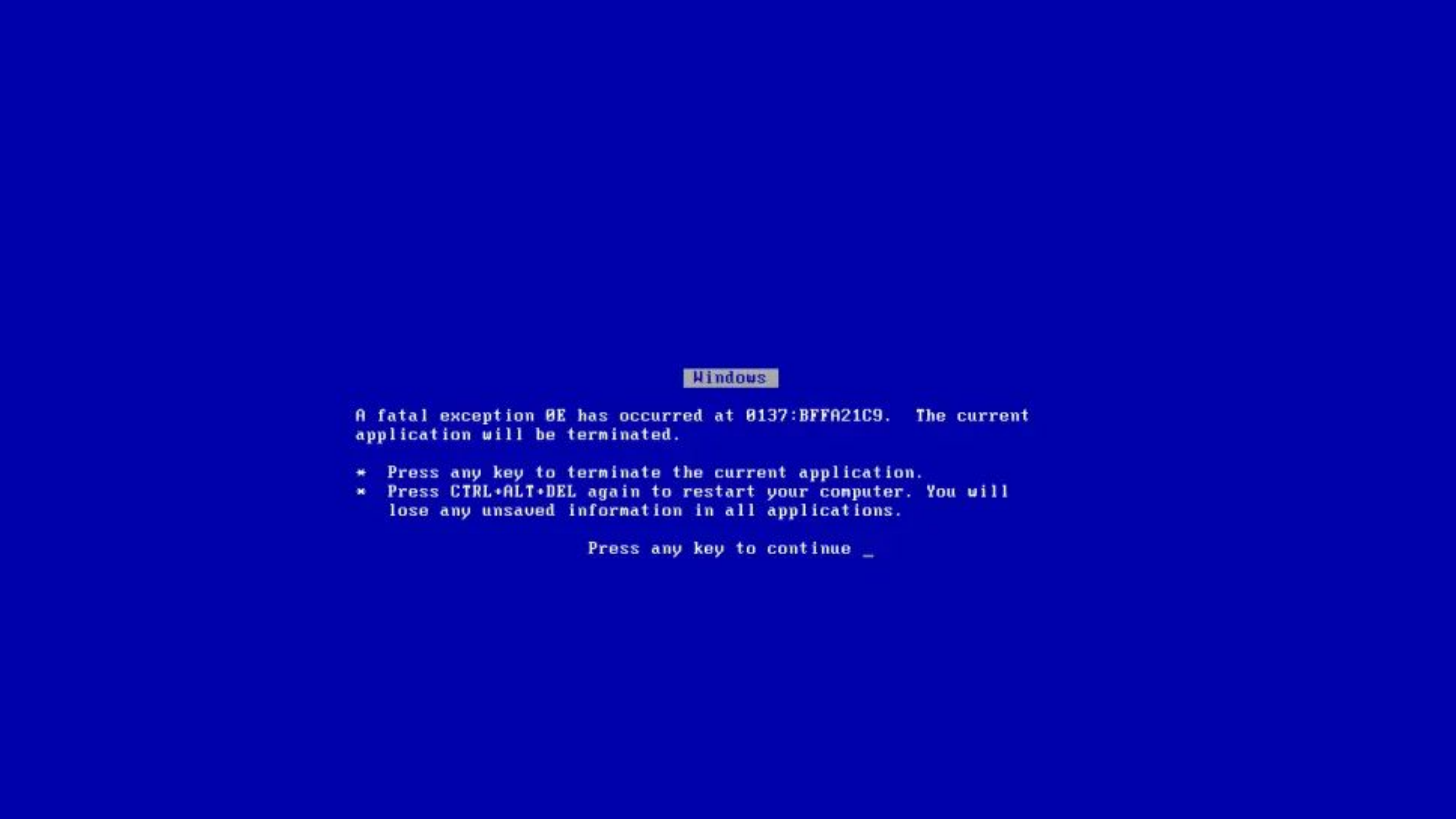
Web-page optimization

And much more when combined with other web technologies...









```
                              Windows

A fatal exception 0E has occurred at 0137:BFFA21C9.  The current
application will be terminated.

*  Press any key to terminate the current application.
*  Press CTRL+ALT+DEL again to restart your computer. You will
   lose any unsaved information in all applications.

                      Press any key to continue _
```

TensorFlow.js

Limitations / Roadblocks

What have we learnt from creating TensorFlow.js?

Float 32

JS / WASM support Float 32, but not 16

In machine learning there are times you want to change the accuracy of the weights of the model to be say for example Float 16 instead of 32. This allows you to:

1. Use less memory to store the model at runtime
2. Increase execution speed

Currently JS / WASM only has support for Float 32 on the lower end of the scale.

Right now TensorFlow.js quantization only has the effect of reducing the file size sent from server to client, but as soon as we load into memory we then need to use Float 32 again meaning we miss out on the runtime benefits of this quantization.

Float 16

What if we could support model quantization to use less memory and gain faster inference in JS at run time?

Whilst the model accuracy may decrease, the 10% drop or so may be acceptable if it means it can run on devices that have less memory available and to run faster on lower end devices.

This would need to be available in JS and WASM.

Garbage Collection - WebGL

The JS garbage collector is great, but it does not deal with WebGL memory

In TensorFlow.js we have a tidy() function that understands when to clean up tensors that will no longer be used.

People new to machine learning however may not be familiar with this concept, especially if are used to JS cleaning up automatically which can lead to memory leaks.

Garbage Collection - WebGL

How could we clean up WebGL memory too?

Currently WebGL is the primary way we get graphics card acceleration for Machine Learning in JS until WebGPU is more widely available.

Is this a situation that can be solved with future version of WebGPU spec or is this something we can address for WebGL too which may benefit other use cases too?

Graphics card acceleration

Currently we use WebGL to execute ops in the machine learning model

It would be more efficient if the browser exposed lower level APIs to the graphics card for more efficient utilisation of the hardware.

Graphics card acceleration

What lower level support do we need for efficient ML when using the graphics card?

Clearly WebGPU is on the way, but with an ML specific lens, are there any other specific needs that may need to be part of such an API in the future?

Model Security

Many production use cases require that their model can not be stolen and used elsewhere.

We have seen a number of use cases where deploying to the front end is preferable for privacy / latency / offline usage but the bottleneck right now is the concern their hard work will be copied and used elsewhere.

Some models can be a significant part of a companies' IP and take a lot of money to develop. Whilst the resulting service may be free to use, companies are reluctant to give away the model itself.

Model Security

How can we prevent an end user from copying a front end deployed ML model in browser?

Could we have a secure way to download a set of files / JS code such that it is not exposed for inspection / saving locally and instead provide a way to communicate with that code to get pass data to it and get results without being able to intercept the downloaded model / pre / post processing code in any way?

Model Warmup

Often it can take a couple of uses of the model before it runs at optimal inference speed

Currently developers have tried approaches to warm up the model on page load by sending zeros to the model as input so the next time it is called by the user for a real task it is ready to do so efficiently.

We have also seen developers experiment with having more efficient models running initially in WASM, whilst a more complex model loads in WebGL which it then switches to when ready.

Model Warmup

What if there was a standard way to specify a better model is available and should be prepared and swapped to when ready?

Taking a hypothetical example of detecting an object in an image maybe the use case would be as follows:

1) Initially download a lightweight model like COCO-SSD that loads fast but only gives us bounding box data
2) In the background download a more advanced image segmentation model (but maybe takes several seconds to load up) to the upgrade the resolution of what is detected in the image and swap to that model when it is warmed up and ready to use.
3) How could this be defined in a library agnostic way? What other considerations should there be here?

See what the community has made

#MadeWithTFJS

TensorFlow