WEBNN 最新技术进展

胡宁馨 ningxin.hu@intel.com
张敏 belem.zhang@intel.com

英特尔 SATG Web 平台工程
2023 年 6 月

WebML 客户端推理的优势

隐私

摄像头、麦克风等传感器数据保留在设备中

离线

初始资源缓存并离线后，不再依赖网络

延迟

无云端网络问题，浏览器实时推理

成本

无需云端算力支持

0 安装

浏览器中运行，无需额外安装，并易于共享

跨平台

在几乎所有平台上运行 AI 应用

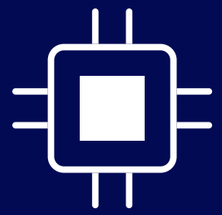
WebML 客户端推理

突发的
延迟敏感

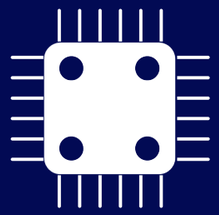
持续的
电量敏感

周期的
吞吐量敏感

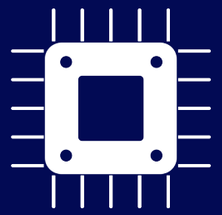
多样的客户端 AI 场景, 多种满足需求的计算单元



CPU
无处不在
低延迟, 单一推理任务



GPU
高并行性, 高 batch size
与 3D/渲染/媒体管道集成



NPU
专用低功耗AI加速器
高能耗比, 提升电源效率

Web 开发者的需求

> The web needs its own neural networks specification to leverage Apple Silicon, Tensor Cores, and others.

> Incredible new power unlocked for the free, open and competitive Web!

> Native Tensor support! It would be amazing to have Tensor objects and operations built into Chrome, and available as an“ML API” similar to the other Chrome APIs.

> Although some scientific computing libraries exist for JS/TS, having built-in support would be far more desirable!

> If go through the code of utils, maths, audio, tensor in JS, it is annoying that I had to implement these ops myself in JS.

> Delighted to find the working drafts of WebNN.

WebNN 简介

新兴的 W3C Web 标准 API

神经网络的统一抽象

通过原生 ML API 访问 AI 硬件加速器

接近原生的 AI 推理性能

目前在 Chrome 和 Edge Canary 中可用（runtime flag）

W3C Candidate Recommendation Draft

TABLE OF CONTENTS

1 **Introduction**

2 **Use cases**
2.1 Application Use Cases
2.1.1 Person Detection
2.1.2 Semantic Segmentation
2.1.3 Skeleton Detection
2.1.4 Face Recognition
2.1.5 Facial Landmark Detection
2.1.6 Style Transfer
2.1.7 Super Resolution
2.1.8 Image Captioning
2.1.9 Machine Translation
2.1.10 Emotion Analysis
2.1.11 Video Summarization
2.1.12 Noise Suppression
2.1.13 Detecting fake video
2.2 Framework Use Cases
2.2.1 Custom Layer
2.2.2 Network Concatenation
2.2.3 Performance Adaptation
2.2.4 Operation Level Execution
2.2.5 Integration with real-time video processing

3 **Security Considerations**
3.1 Guidelines for new operations

4 **Privacy Considerations**

5 **Ethical Considerations**

6 **Programming Model**
6.1 Overview
6.2 Device Selection

7 **API**
7.1 The navigator.ml interface

Web Neural Network API

W3C Candidate Recommendation Draft, 6 June 2023

▼ More details about this document

This version:
https://www.w3.org/TR/2023/CRD-webnn-20230606/

Latest published version:
https://www.w3.org/TR/webnn/

Editor's Draft:
https://webmachinelearning.github.io/webnn/

Previous Versions:
https://www.w3.org/TR/2023/CRD-webnn-20230519/

History:
https://www.w3.org/standards/history/webnn

Implementation Report:
https://wpt.fyi/results/webnn?label=master&label=experimental&aligned&q=webnn

Test Suite:
https://github.com/web-platform-tests/wpt/tree/master/webnn

Feedback:
GitHub
Inline In Spec

Editors:
Ningxin Hu (Intel Corporation)
Chai Chaoweeraprasit (Microsoft Corporation)

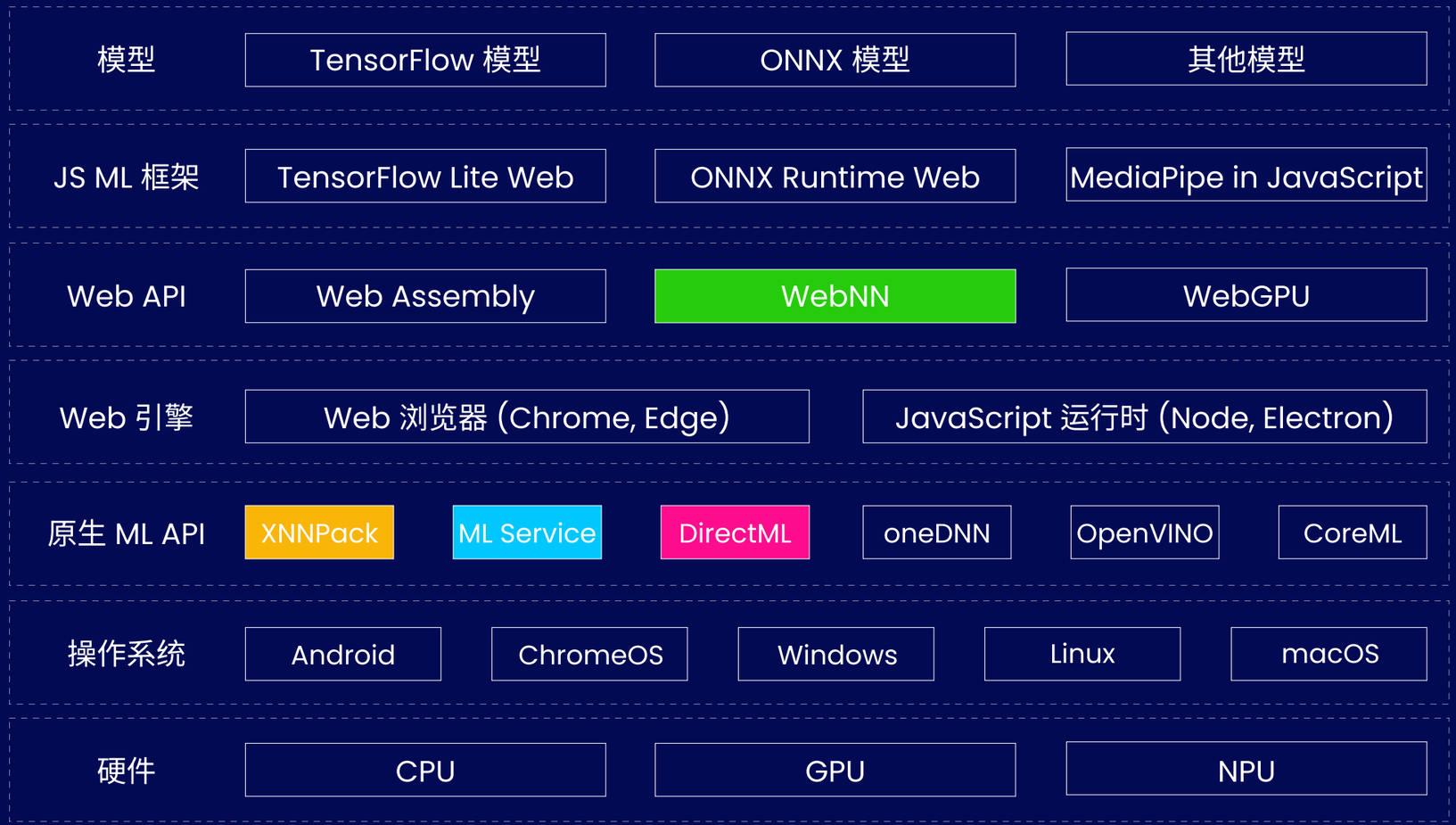
Explainer:
explainer.md

Polyfill:
webnn-polyfill / webnn-samples

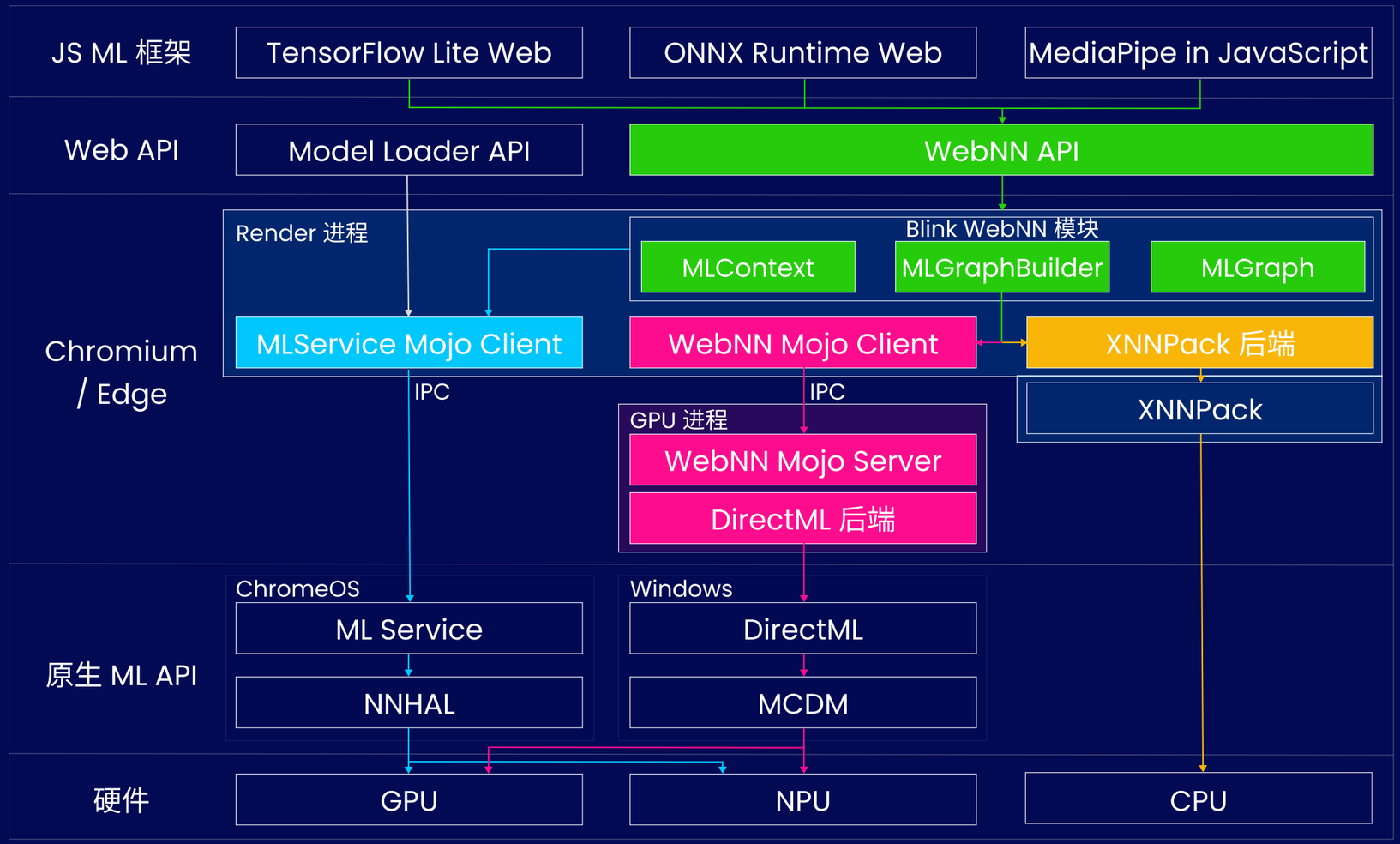
Abstract

This document describes a dedicated low-level API for neural network inference hardware acceleration.

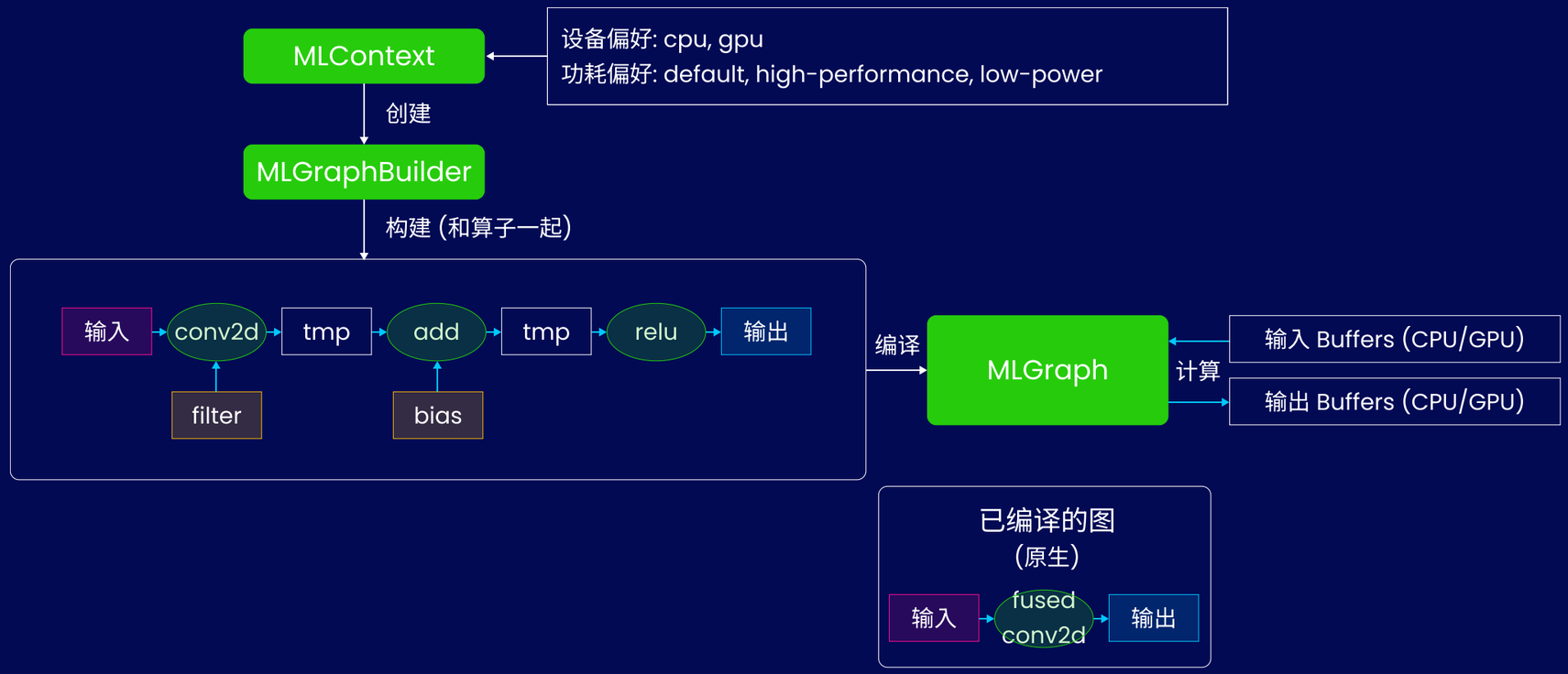
WebNN 架构


模型
TensorFlow 模型
ONNX 模型
其他模型
JS ML 框架
TensorFlow Lite Web
ONNX Runtime Web
MediaPipe in JavaScript
Web API
Web Assembly
WebNN
WebGPU
Web 引擎
Web 浏览器 (Chrome, Edge)
JavaScript 运行时 (Node, Electron)
原生 ML API
XNNPack
ML Service
DirectML
oneDNN
OpenVINO
CoreML
操作系统
Android
ChromeOS
Windows
Linux
macOS
硬件
CPU
GPU
NPU

WebNN 在 Chromium 中的实现



WebNN 编程模型


MLContext
设备偏好: cpu, gpu
功耗偏好: default, high-performance, low-power
创建
MLGraphBuilder
构建 (和算子一起)
输入
conv2d
tmp
add
tmp
relu
输出
filter
bias
编译
MLGraph
输入 Buffers (CPU/GPU)
计算
输出 Buffers (CPU/GPU)
已编译的图
(原生)
输入
fused
conv2d
输出
计算图图例
输入操作数
常量操作数
输出操作数
中间操作数
算子
WebNN API
其他 Web API
数据流
调用流

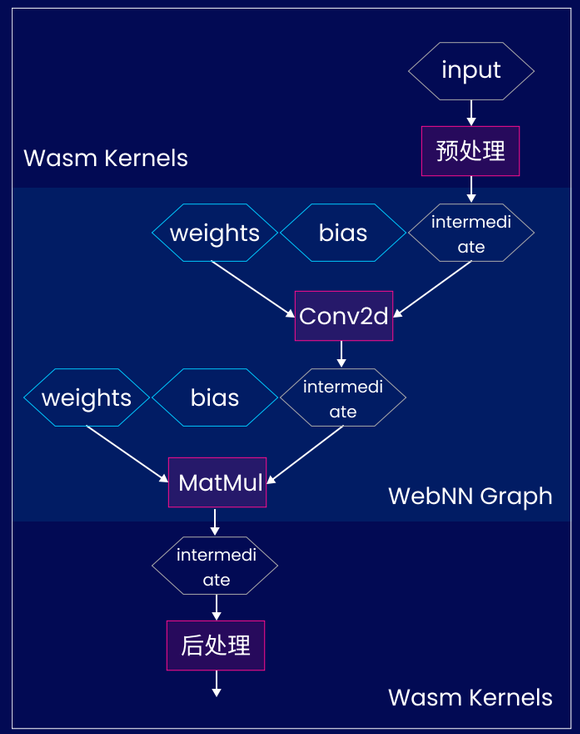
WebNN 操作符的实现状态

WebNN Spec	Web Platform Tests	XNNPack/CPU backend	External Delegate	Execution Provider
clamp	📈 🧪	✅ clamp ✅ Relu6	✅ ReluN1To1	✅ Clip
concat	📈 🧪	✅ concatenate2 ✅ concatenate3 ✅ concatenate4	✅ Concatenation	✅ Concat
conv2d	📈 🧪	✅ convolution_2d	✅ Conv2d ✅ DepthwiseConv2d	✅ Conv
convTranspose2d	📈 🧪	✅ deconvolution_2d	✅ TransposeConv ✅ Convolution2DTransposeBias	✅ ConvTranspose
add element-wise binary	📈 🧪	✅ add2	✅ Add	✅ Add
sub element-wise binary	📈 🧪	✅ subtract	✅ Sub	✅ Sub
mul element-wise binary	📈 🧪	✅ multiply2	✅ Mul	✅ Mul
div element-wise binary	📈 🧪	✅ divide	✅ Div	✅ Div
max element-wise binary	📈 🧪	✅ maximum2	✅ Maximum	🚀 Max
min element-wise binary	📈 🧪	✅ minimum2	✅ Minimum	🚀 Min
abs element-wise unary	📈 🧪	🔍 abs	✅ Abs	🚀 Abs
ceil element-wise unary	📈 🧪	🔍 ceiling	✅ Ceil	🚀 Ceil
floor element-wise unary	📈 🧪	🔍 floor	✅ Floor	🚀 Floor
neg element-wise unary	📈 🧪	🔍 negate	✅ Neg	🚀 Neg
elu	📈 🧪	✅ elu	✅ Elu	🚀 Elu

WebNN 操作符的实现状态

WebNN Spec	Web Platform Tests	XNNPack/CPU backend	External Delegate TensorFlow Lite for TensorFlow.js	Execution Provider ONNX RUNTIME
hardSwish	📈 🧪	✅ hardswish	✅ HardSwish	🚀 HardSwish
leakyRelu	📈 🧪	✅ leaky_relu	✅ LeakyRelu	✅ LeakyRelu
pad	📈 🧪	✅ static_constant_pad	✅ Pad	🚀 Pad
averagePool2d pooling	📈 🧪	✅ average_pooling_2d	✅ AveragePool2d ✅ Mean	✅ GlobalAveragePool ✅ AveragePool
maxPool2d pooling	📈 🧪	✅ max_pooling_2d	✅ MaxPool2d	✅ GlobalMaxPool ✅ MaxPool
prelu	📈 🧪	✅ prelu	✅ Prelu	🚀 Prelu
relu	📈 🧪	✅ clamp	✅ Relu	✅ Relu
resample2d	🚀 🚀	✅ static_resize_bilinear_2d	✅ ResizeBilinear	✅ Resize
reshape	📈 🧪	✅ static_reshape	✅ Reshape	✅ Reshape
sigmoid	📈 🧪	✅ sigmoid	✅ Logistic	✅ Sigmoid
split	📈 🧪	🔍 even_split2 🔍 even_split3 🔍 even_split4 🔍 static_slice (uneven split)	✅ Split	🚀 Split
slice	📈 🧪	🔍 static_slice	🚀 Slice	🚀 Slice
softmax	📈 🧪	✅ softmax	✅ Softmax	🚀 Softmax
transpose	📈 🧪	✅ static_transpose	✅ Transpose	✅ Transpose
30	29	35	34	32

WebNN 和主流 JavaScript ML 框架的集成



解释器核心 Interpreter Core

TensorFlow Lite Wasm 运行时

WebML 框架运行时 (由 Emscripten 构建)

MediaPipe JS | TensorFlow Lite Web | ONNX Runtime Web

TensorFlow Lite Wasm 运行时 | ONNX Runtime 运行时

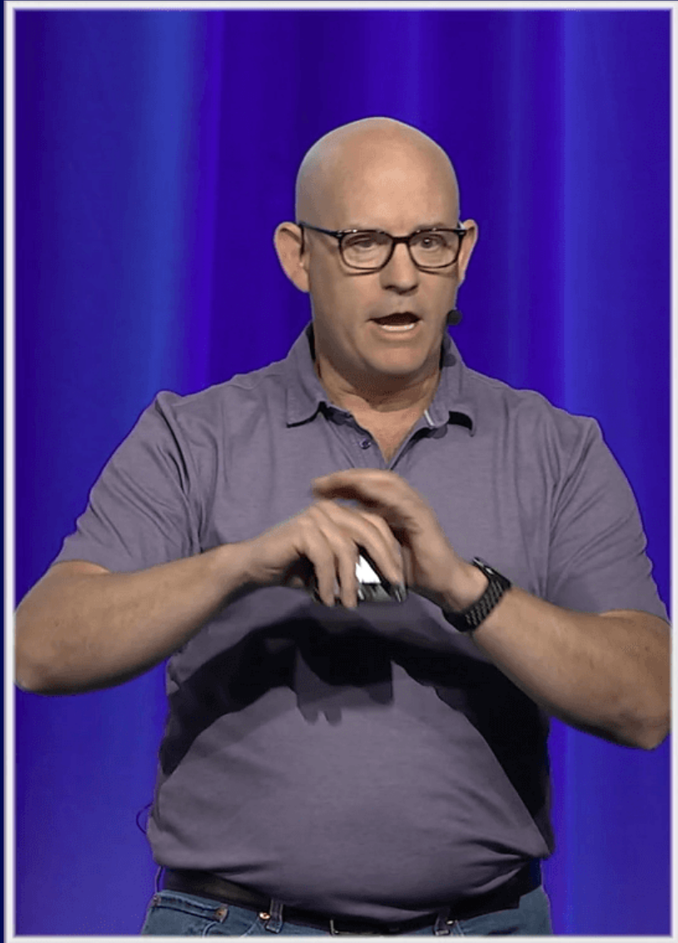
Wasm Kernels | WebNN Graph | WebGPU Kernels | WebGL Kernels

浏览器 / Node.js (WebNN)

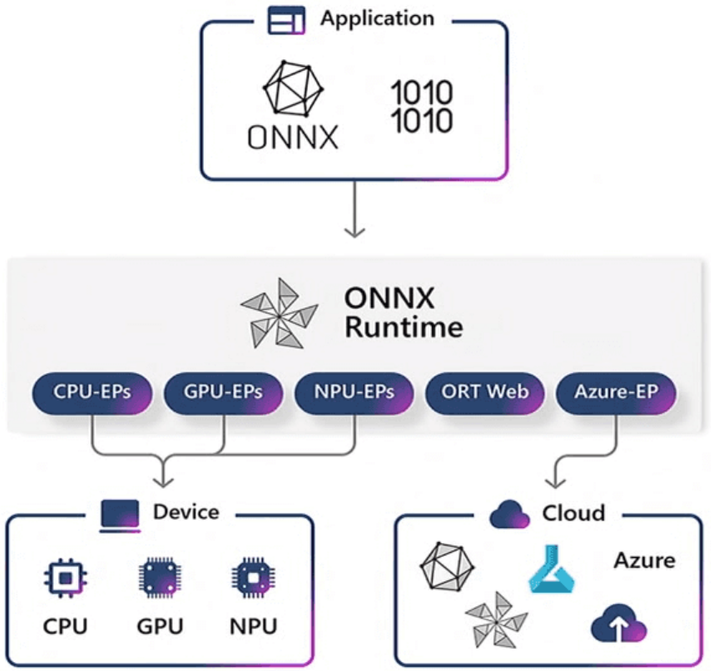
原生 CPU Kernels | 原生 GPU Kernels | 原生 NPU Kernels

CPU | GPU | NPU

WebNN 和 ONNX Runtime Web 的集成



ONNX Runtime



Execution Provider	Status	Company
Azure	Public Preview	Microsoft
QNN	Public Preview	Qualcomm
OpenVino	Public preview	Intel
Vitis AI	Public preview	AMD
DirectML	GPU support across IHVs	Microsoft
WebNN	Working with partners to deliver	Microsoft, Intel

Microsoft Build 2023 Deliver AI-powered experiences across cloud and edge, with Windows

WebNN 代码示例

```
const context = await navigator.ml.createContext({powerPreference: 'low-power'})

// The following code builds a graph as:
// constant1 ---+
//              +--- Add ---> intermediateOutput1 ---+
// input1    ---+                                    |
//                                                   +--- Mul---> output
// constant2 ---+                                    |
//              +--- Add ---> intermediateOutput2 ---+
// input2    ---+

// Use tensors in 4 dimensions.
const TENSOR_DIMS = [1, 2, 2, 2];
const TENSOR_SIZE = 8;

const builder = new MLGraphBuilder(context);

// Create MLOperandDescriptor object.
const desc = {type: 'float32', dimensions: TENSOR_DIMS};

// constant1 is a constant MLOperand with the value 0.5.
const constantBuffer1 = new Float32Array(TENSOR_SIZE).fill(0.5);
const constant1 = builder.constant(desc, constantBuffer1);

// input1 is one of the input MLOperands. Its value will be set before execution
const input1 = builder.input('input1', desc);

// constant2 is another constant MLOperand with the value 0.5.
const constantBuffer2 = new Float32Array(TENSOR_SIZE).fill(0.5);
const constant2 = builder.constant(desc, constantBuffer2);
```

```
// input2 is another input MLOperand. Its value will be set before execution.
const input2 = builder.input('input2', desc);

// intermediateOutput1 is the output of the first Add operation.
const intermediateOutput1 = builder.add(constant1, input1);

// intermediateOutput2 is the output of the second Add operation.
const intermediateOutput2 = builder.add(constant2, input2);

// output is the output MLOperand of the Mul operation.
const output = builder.mul(intermediateOutput1, intermediateOutput2);

// Compile the constructed graph.
const graph = await builder.build({'output': output});

// Setup the input buffers with value 1.
const inputBuffer1 = new Float32Array(TENSOR_SIZE).fill(1);
const inputBuffer2 = new Float32Array(TENSOR_SIZE).fill(1);
const outputBuffer = new Float32Array(TENSOR_SIZE);

// Execute the compiled graph with the specified inputs.
const inputs = {
  'input1': inputBuffer1,
  'input2': inputBuffer2,
};
const outputs = {'output': outputBuffer};
const results = await context.compute(graph, inputs, outputs);

console.log('Output value: ' + results.outputs.output);
// Output value: 2.25,2.25,2.25,2.25,2.25,2.25,2.25,2.25
```

WebNN 与 ONNXRuntime Web 集成的代码示例

```js
import { InferenceSession } from "onnxruntime-web";

// ...

// Initialize the ONNX model
const initModel = async () => {
  ort.env.wasm.numThreads = 1; // 4
  ort.env.wasm.simd = true;
  ort.env.wasm.proxy = true;

  const options: InferenceSession.SessionOptions = {
    // provider name: wasm, webnn
    // deviceType: cpu, gpu
    // powerPreference: default, high-performance
    executionProviders:
      [{ name: "wasm"}], // WebAssembly CPU
  };

  // ...
};

const results = await model.run(feeds);
const output = results[model.outputNames[0]];
```

WebAssembly Backend

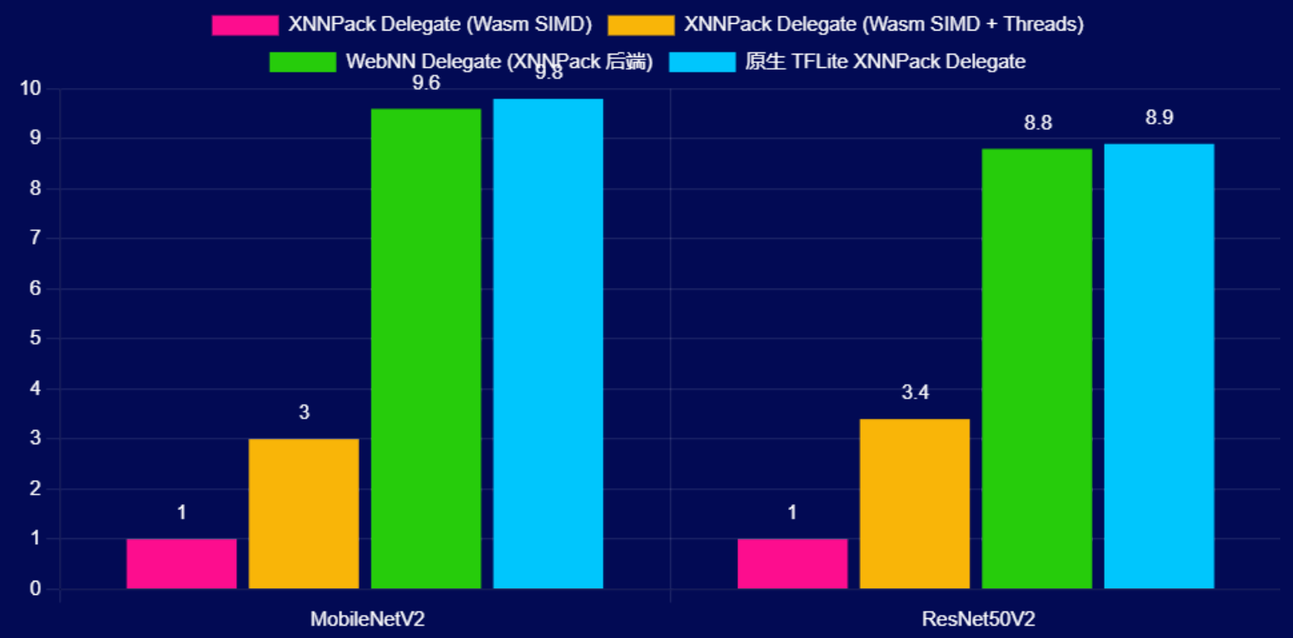
```js
import { InferenceSession } from "onnxruntime-web";

// ...

// Initialize the ONNX model
const initModel = async () => {
  env.wasm.numThreads = 1; // 4
  env.wasm.simd = true;
  env.wasm.proxy = true;

  const options: InferenceSession.SessionOptions = {
    // provider name: wasm, webnn
    // deviceType: cpu, gpu
    // powerPreference: default, high-performance
    executionProviders:
      [{ name: "webnn", deviceType: "gpu", powerPreference: 'default' }],
  };

  // ...
};

const results = await model.run(feeds);
const output = results[model.outputNames[0]];
```

WebNN Backend

WebNN 性能数据（标准化）

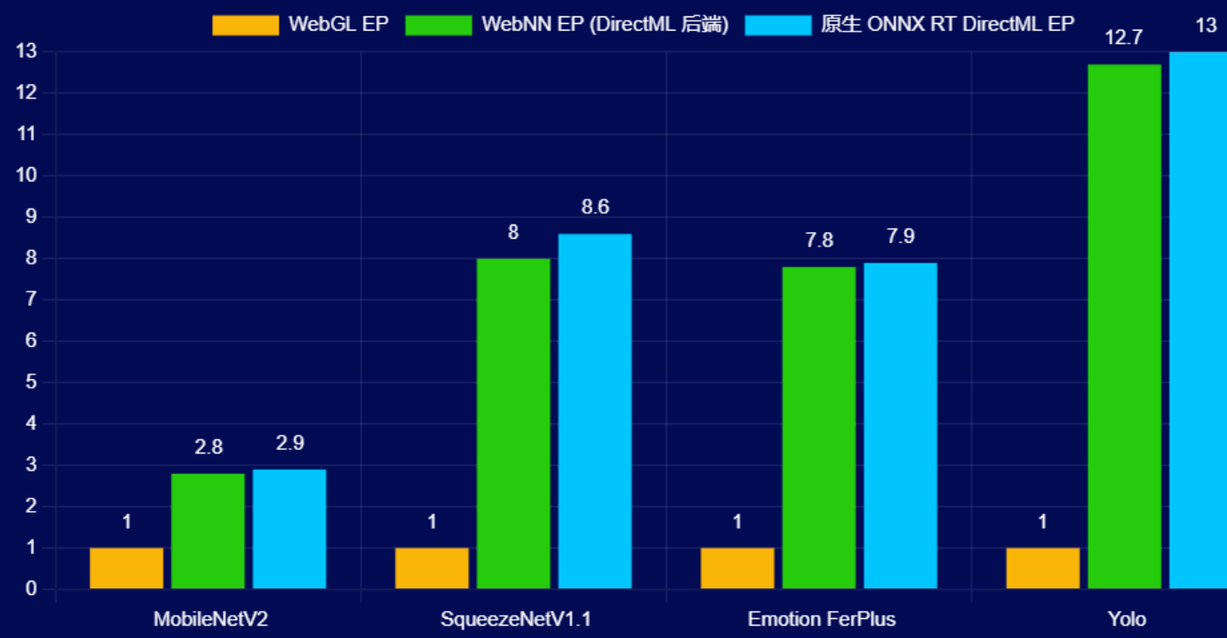
TensorFlow-Lite Web / CPU

越高越好



ONNXRuntime Web / GPU

越高越好

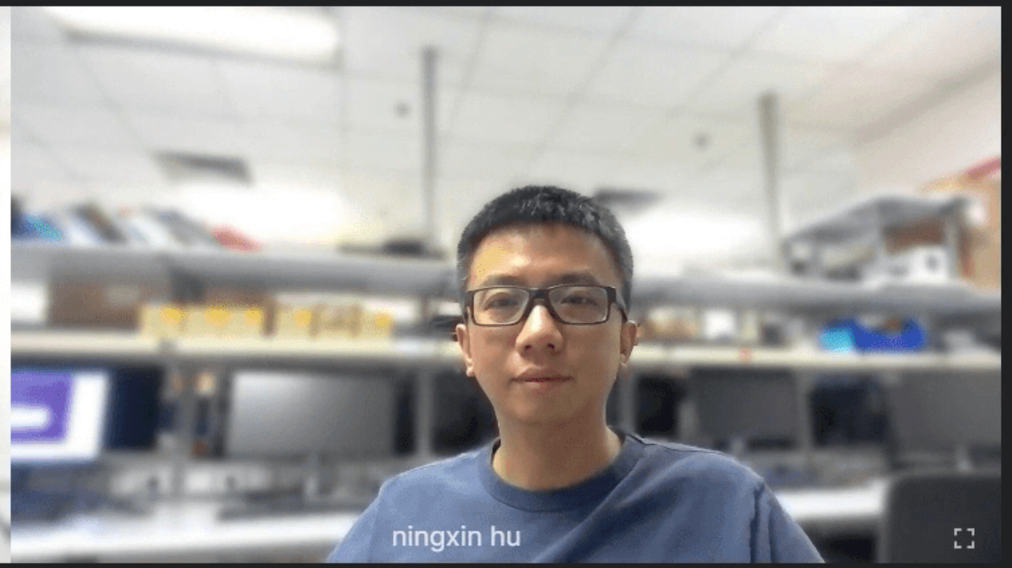


10.239.115.77:4173/_gather

Not secure https://10.239.115.77:4173/_gathering/Belem%20Zhang

GatherEase
Gathering with Others on the Web

12:26:15 · Tue, May 09 Belem Zhang



Participants (2)

BZ Belem Zhang

NH Ningxin Hu

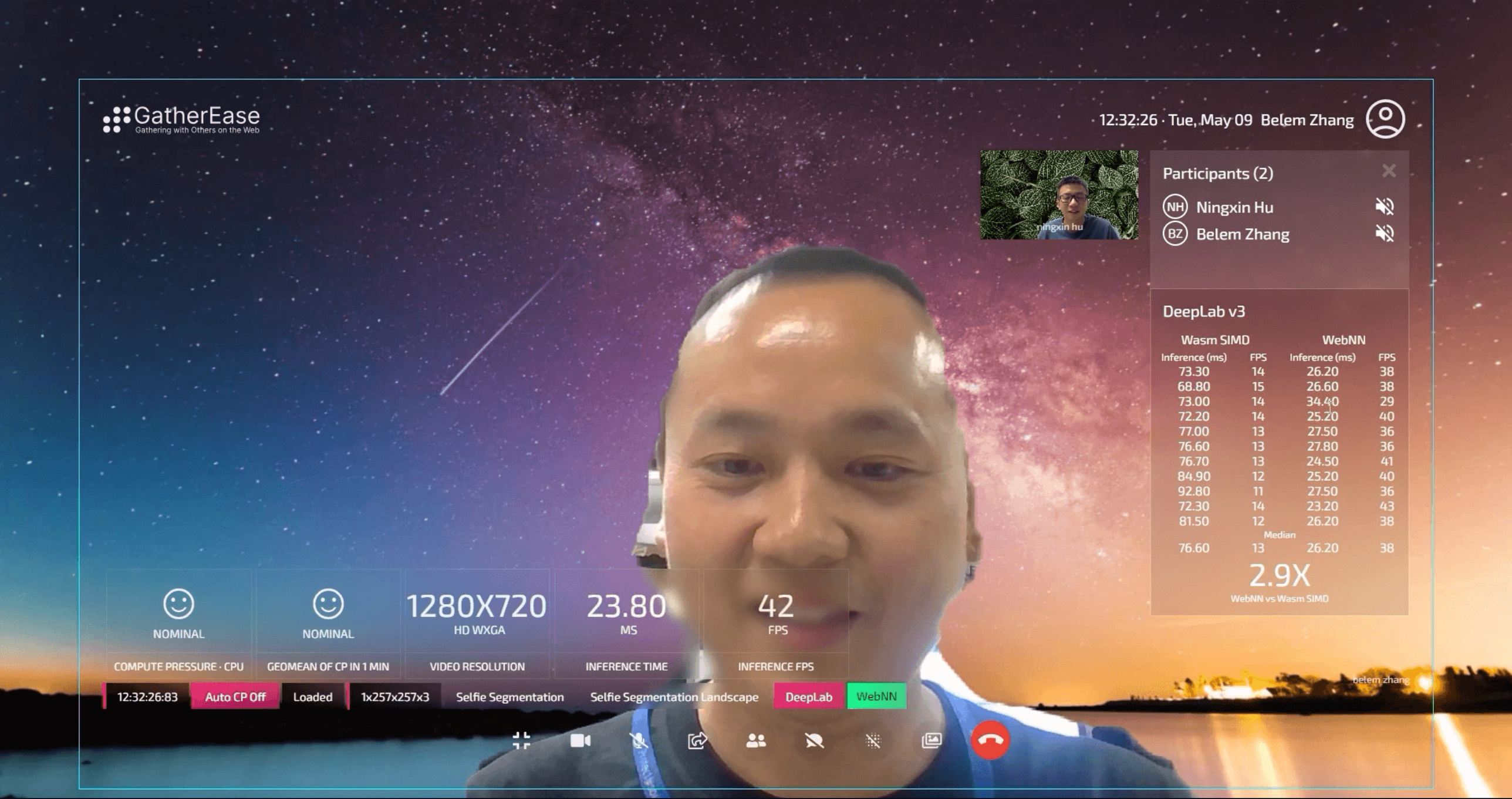
Selfie Segmentation

Wasm SIMD		WebNN	
Inference (ms)	FPS	Inference (ms)	FPS
9.70	103	4.70	213
10.10	99	4.90	204
9.20	109	5.10	196
9.80	102	5.10	196
10.90	92	5.60	179
9.80	102	8.50	118
10.20	98	4.70	213
8.70	115	5.00	200
10.20	98	5.30	189
10.50	95	5.80	172
13.20	76	6.00	167
Median			
10.10	99	5.10	196

2.0X
WebNN vs Wasm SIMD

☺NOMINAL	☺NOMINAL	1280X720 HD WXGA	4.80 MS	208 FPS
COMPUTE PRESSURE · CPU	GEOMEAN OF CP IN 1 MIN	VIDEO RESOLUTION	INFERENCE TIME	INFERENCE FPS

Auto CP Off | Loaded 1×256×256×3 | Selfie Segmentation | Selfie Segmentation Landscape | DeepLab | WebNN


GatherEase
Gathering with Others on the Web
12:32:26 · Tue, May 09 Belem Zhang
ningxin hu
Participants (2)
NH Ningxin Hu
BZ Belem Zhang
DeepLab v3
Wasm SIMD
WebNN
Inference (ms) FPS Inference (ms) FPS
73.30 14 26.20 38
68.80 15 26.60 38
73.00 14 34.40 29
72.20 14 25.20 40
77.00 13 27.50 36
76.60 13 27.80 36
76.70 13 24.50 41
84.90 12 25.20 40
92.80 11 27.50 36
72.30 14 23.20 43
81.50 12 26.20 38
Median
76.60 13 26.20 38
2.9X
WebNN vs Wasm SIMD
NOMINAL
NOMINAL
1280X720
HD WXGA
23.80
MS
42
FPS
COMPUTE PRESSURE · CPU
GEOMEAN OF CP IN 1 MIN
VIDEO RESOLUTION
INFERENCE TIME
INFERENCE FPS
12:32:26:83
Auto CP Off
Loaded
1x257x257x3
Selfie Segmentation
Selfie Segmentation Landscape
DeepLab
WebNN
belem zhang

W3C Machine Learning for the Web

社区组

讨论和探索新想法，孵化机器学习推理的新提案

39 个组织代表, 126 名参与者

Adobe agora HUAWEI moz://a SAMSUNG SAP

工作组

基于社区组孵化的提案，标准化机器学习推理的 Web API

17 个组织代表, 43 名参与者 (3 名特邀专家)

BBC ByteDance BERKELEY LAB NVIDIA

Alibaba Group 阿里巴巴集团 Baidu 百度 Google intel Microsoft

谢谢！

web
neural network

https://webnn.dev

WebNN 交流群

联系张敏