

# Access purpose-built ML hardware with Web Neural Network API

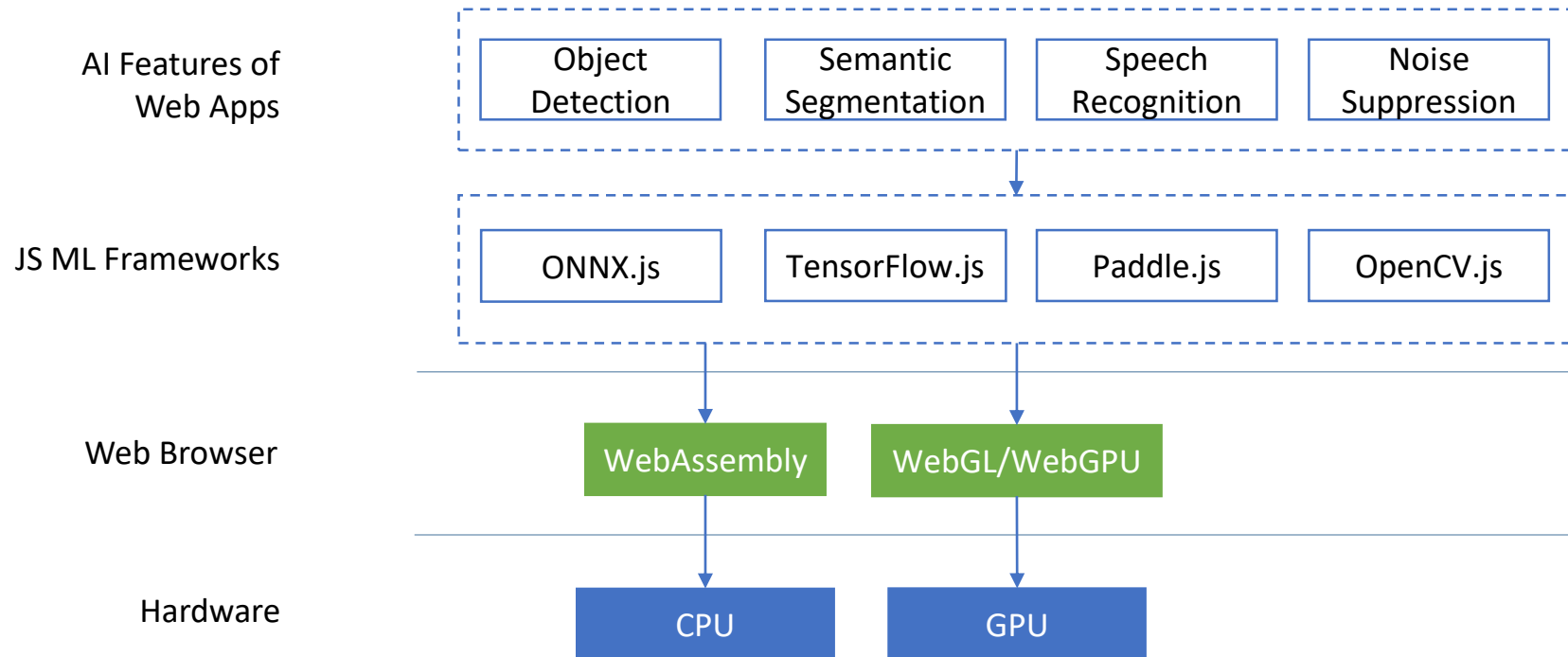
Ningxin Hu

Intel Corporation

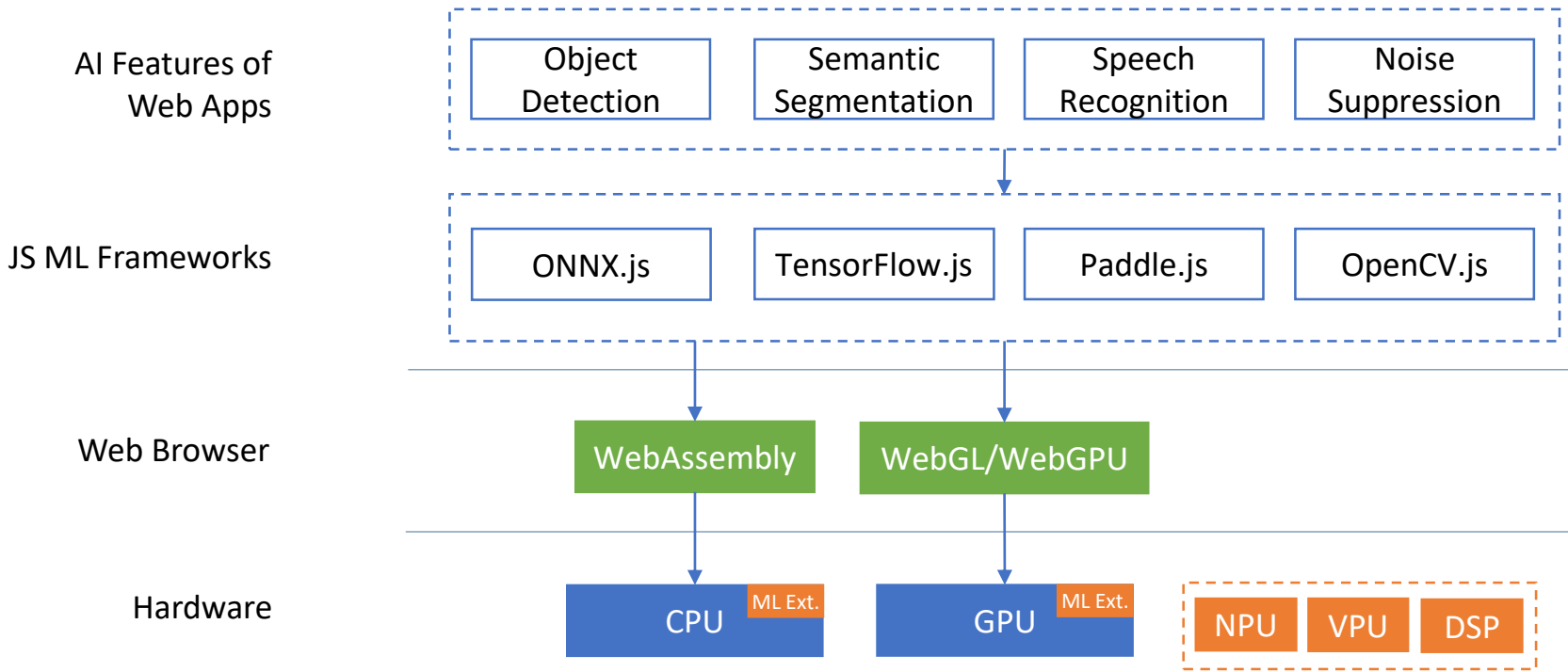
July 2020



# The JS ML frameworks and AI Web apps

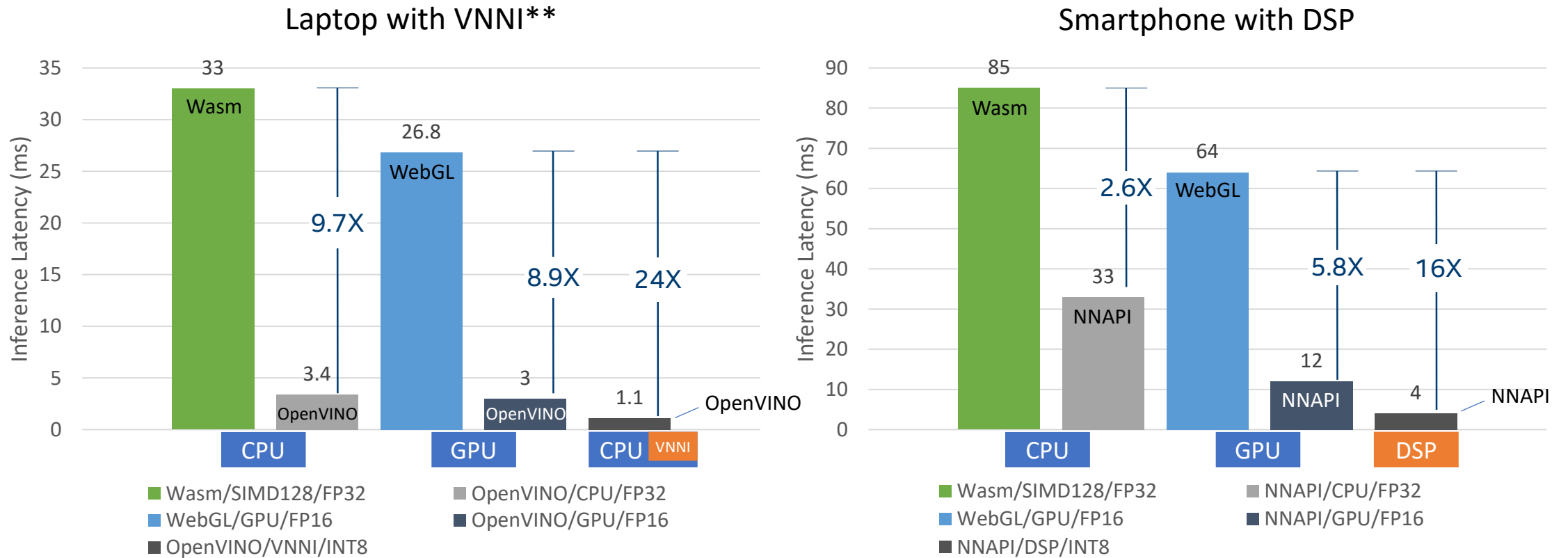


# The purpose-built ML hardware



# The performance gap: Web and native

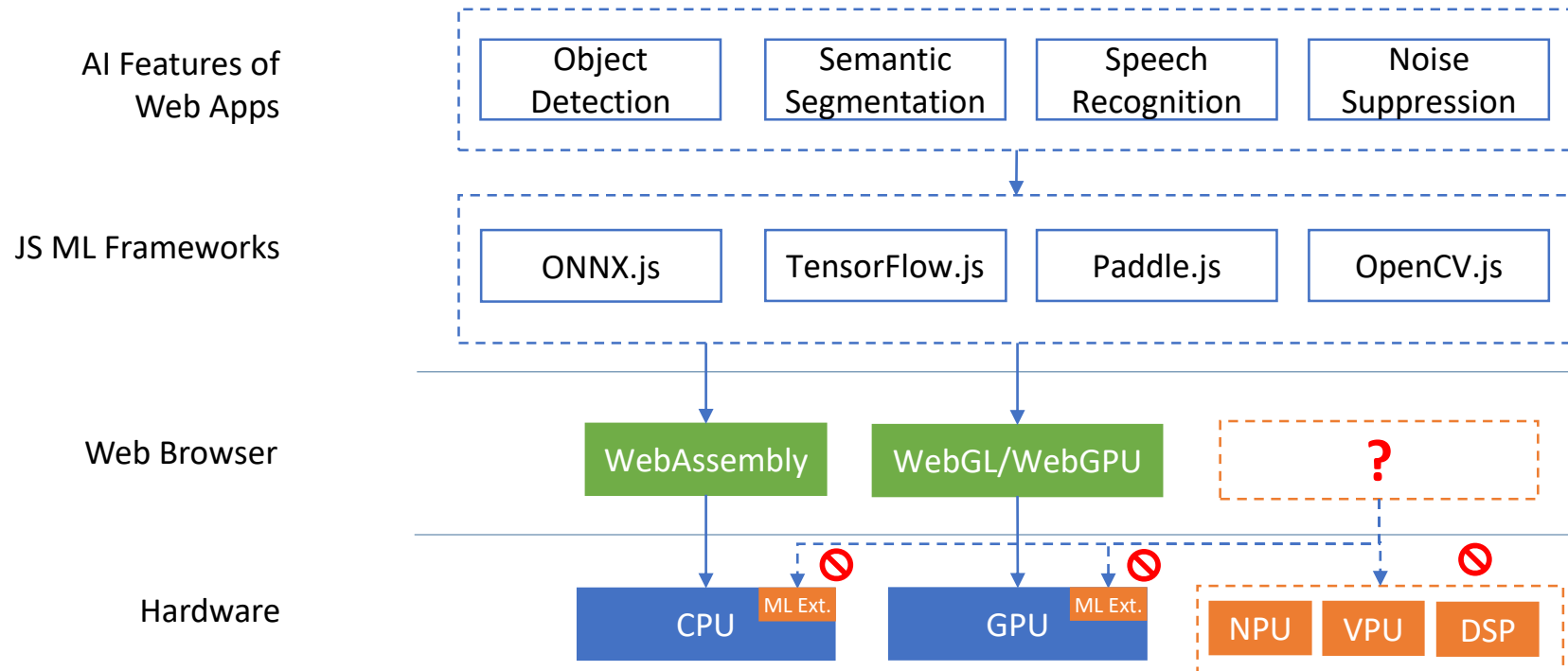
## MobileNet\* Inference Latency (smaller is better)



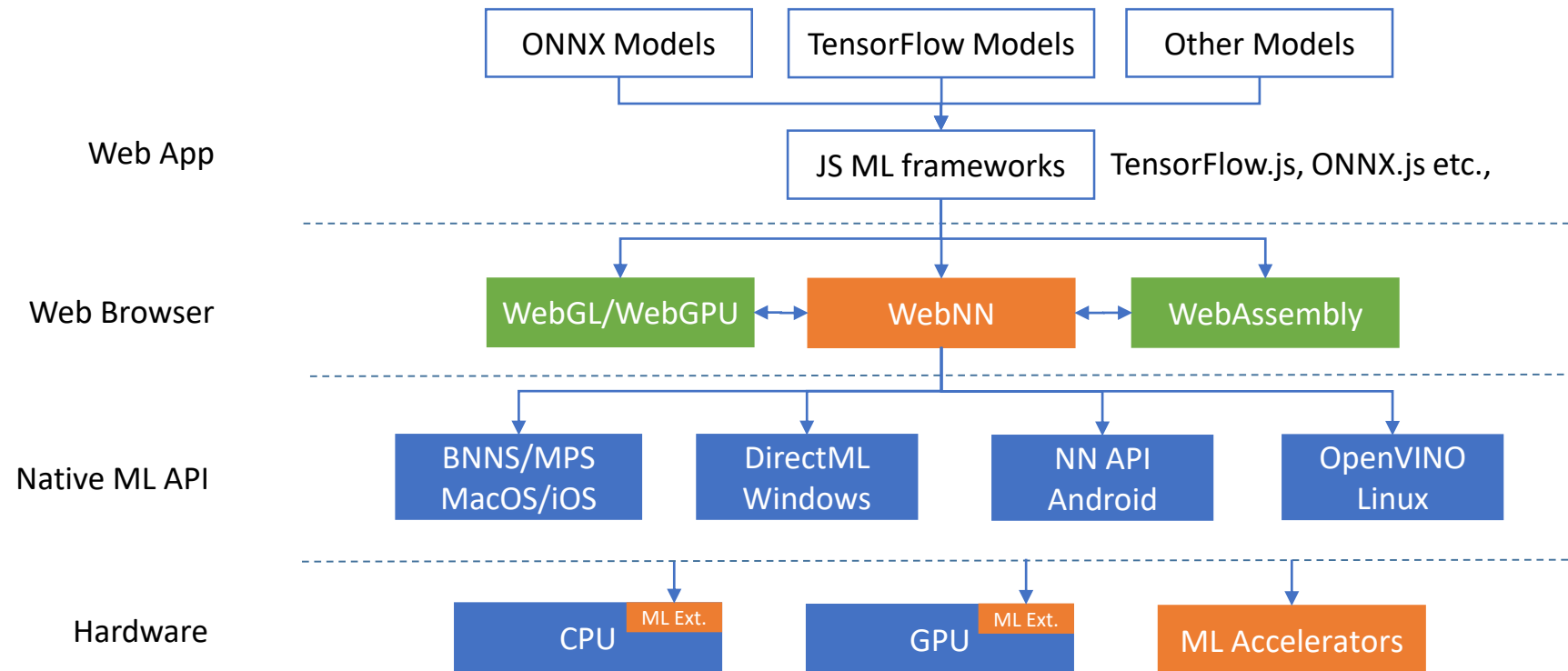
\* Batch size: 1, input size: 224x224, width multiplier: 1.0

\*\* VNNI: Vector Neural Network Instruction

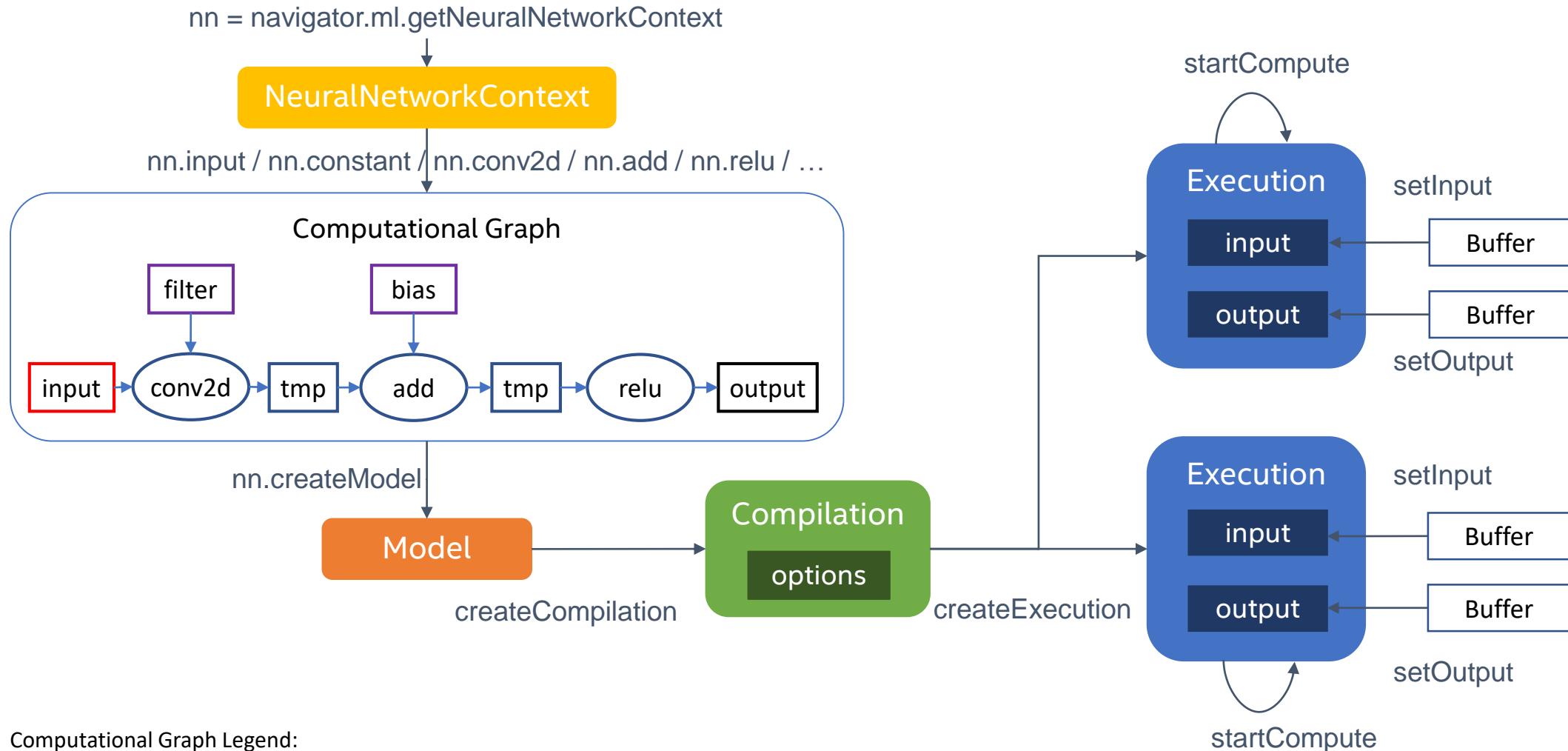
# The Web is disconnected from ML hardware



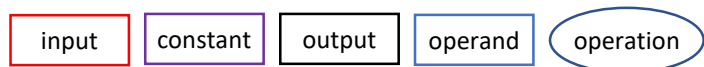
# WebNN: the architecture view



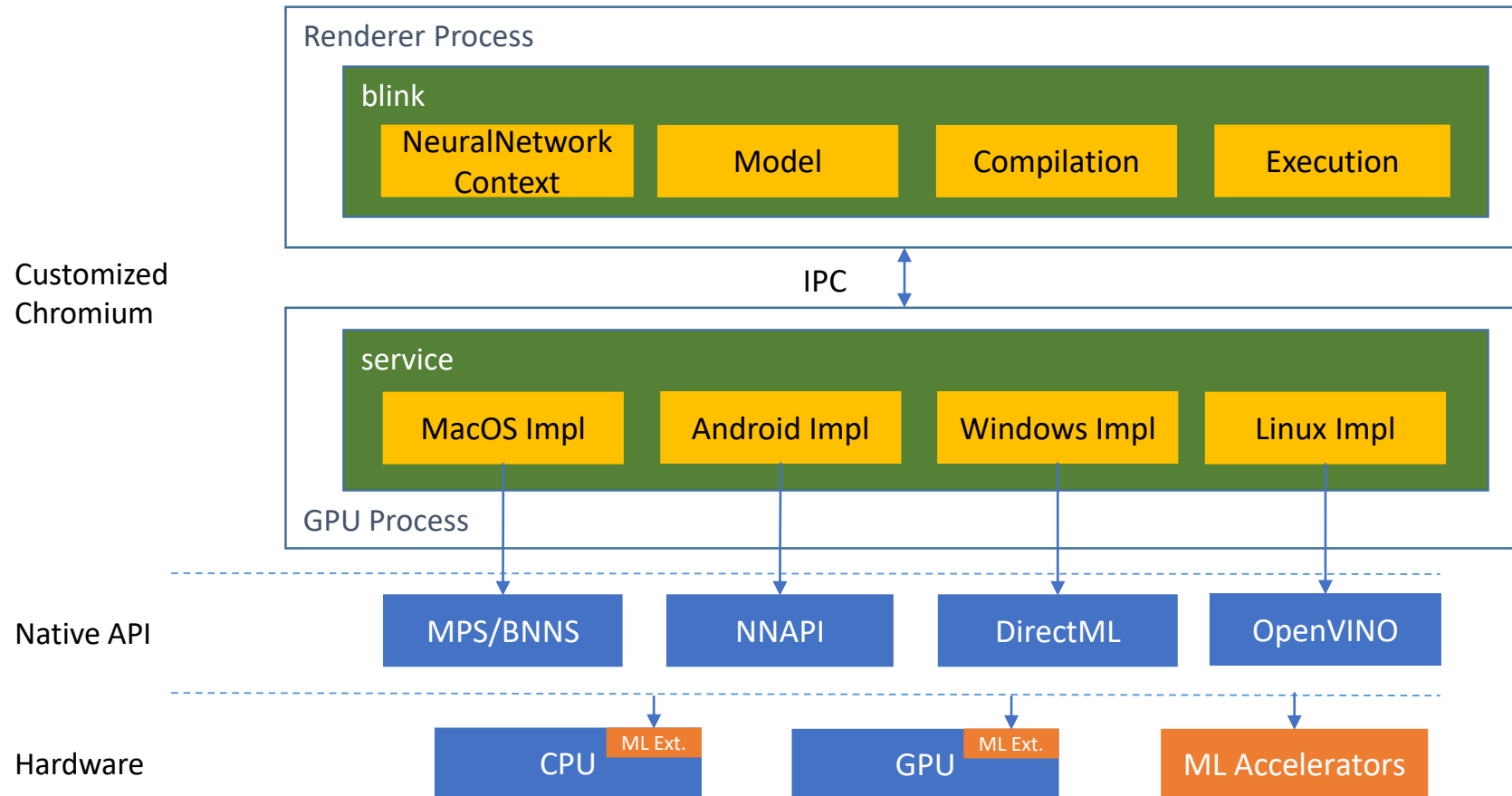
# WebNN: the programming model



Computational Graph Legend:



# WebNN: the proof-of-concept implementation





# WebNN: the demos

Image Classification / WebNN (FAST) / MobileNet V1 Quant (Caffe2)

103 FPS

Inference Time: 1.74 ms

#	Label	Probability
1	notebook	63.78%
2	laptop	20.50%
3	mouse	3.83%

WebNN image classification on a laptop with VNNI

Image Classification / WebNN (LOW) / MobileNet V2 Quant (TFLite)

58 FPS

Inference Time: 7.60 ms

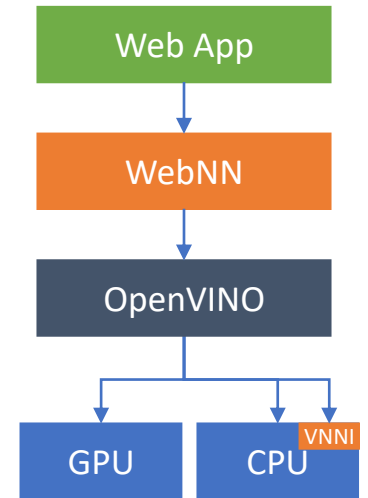
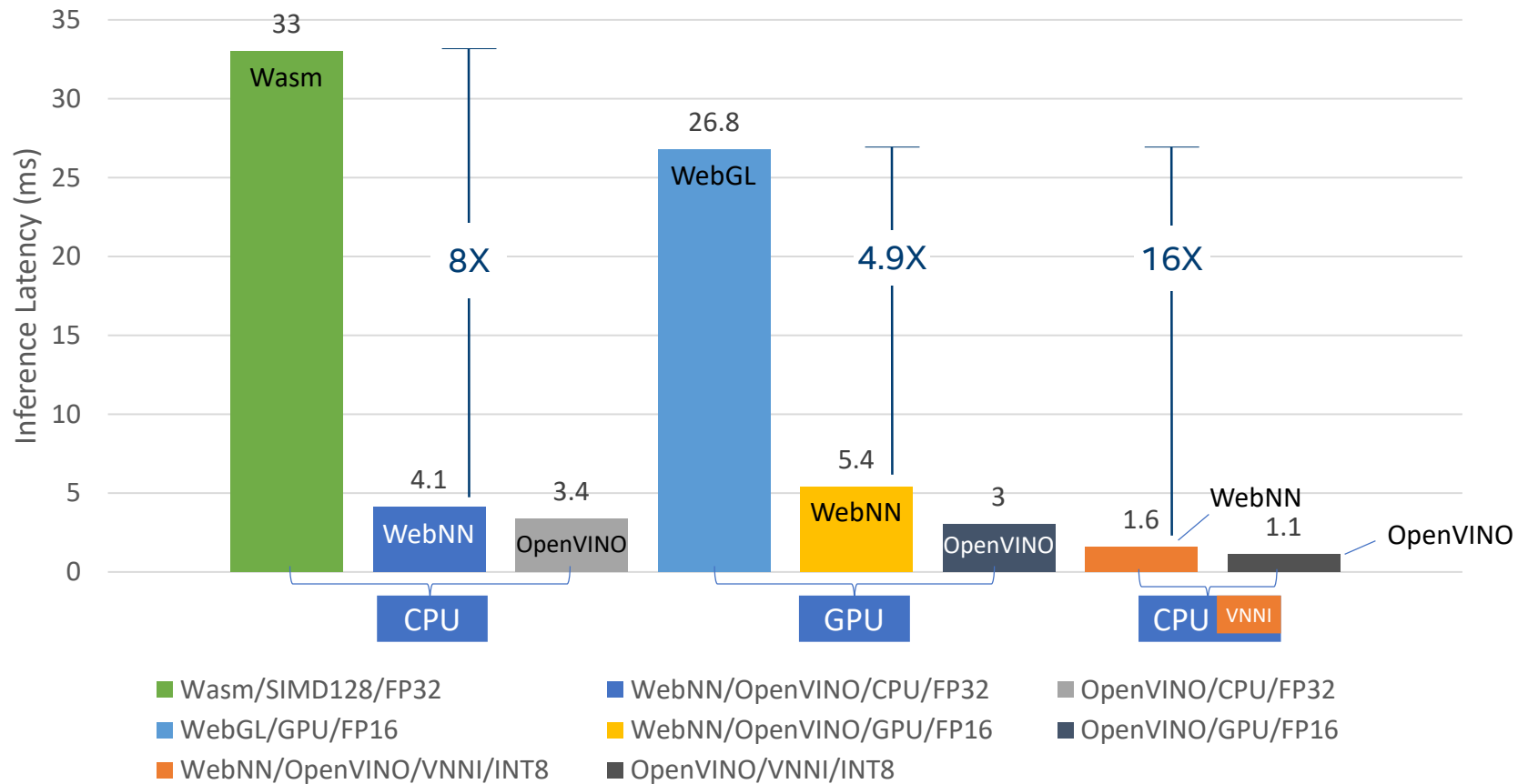
#	Label	Probability
1	notebook	79.69%
2	laptop	14.84%
3	mouse	0.78%

Front-facing

WebNN image classification on a smartphone with DSP

# WebNN: the PoC performance

MobileNet\* Inference Latency on Laptop with VNNI\*\*  
(smaller is better)

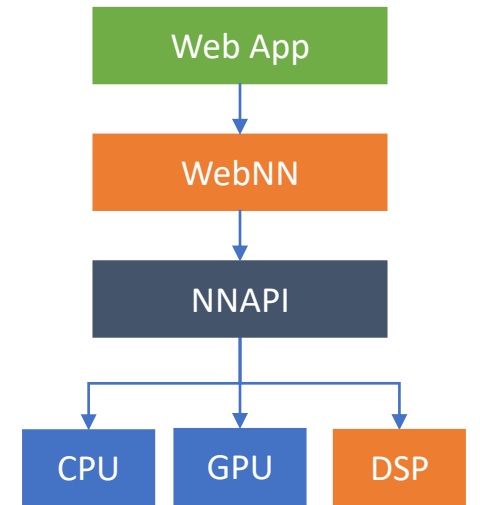
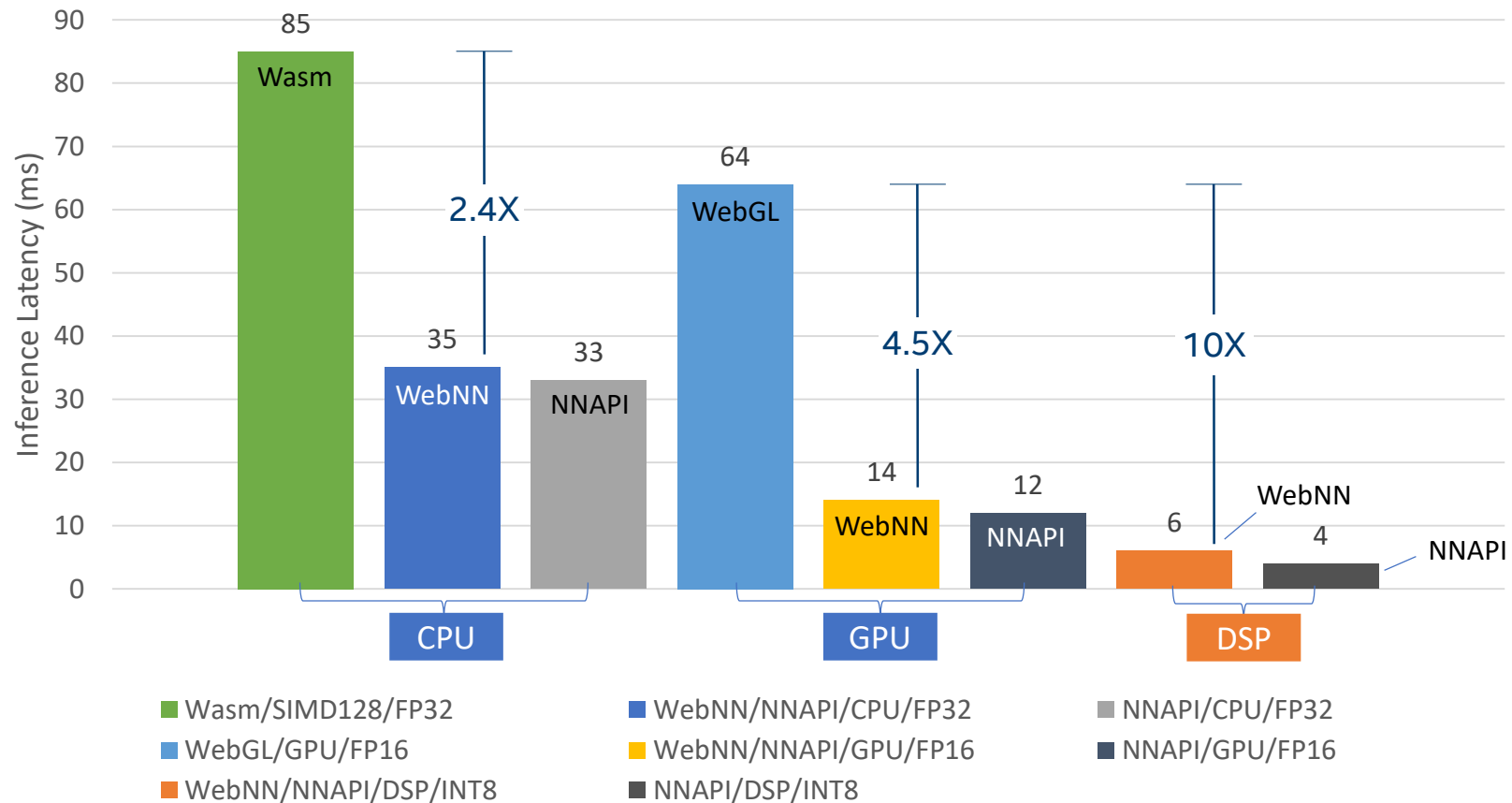


\* Batch size: 1, input size: 224x224, width multiplier: 1.0

\*\* VNNI: Vector Neural Network Instruction

# WebNN: the PoC performance – cont'd

MobileNet\* Inference Latency on Smartphone with DSP  
(smaller is better)



\* Batch size: 1, input size: 224x224, width multiplier: 1.0

# Call for Participation

<https://www.w3.org/community/webmachinelearning/>

<https://webmachinelearning.github.io/webnn/>

## TABLE OF CONTENTS

1	<b>Introduction</b>
2	<b>Use cases</b>
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
2.2	Framework Use Cases
2.2.1	Custom Layer
2.2.2	Network Concatenation
2.2.3	Performance Adaptation
3	<b>API</b>
3.1	Navigator
3.2	ML
3.3	OperandDescriptor
3.4	Operand
3.5	NeuralNetworkContext
3.5.1	batchNormalization
3.5.2	element-wise binary operations
3.5.3	element-wise unary operations
3.5.4	concat
3.5.5	conv2d
3.5.6	gemm
3.5.7	leakyRelu
3.5.8	matmul
3.5.9	pooling operations

## Web Neural Network API

Draft Community Group Report, 1 July 2020



### This version:

<https://webmachinelearning.github.io/webnn/>

### Issue Tracking:

[GitHub](#)

### Editors:

Ningxin Hu ([Intel Corporation](#))

Chai Chaoweerasit ([Microsoft Corporation](#))

### Explainer:

[explainer.md](#)

Copyright © 2020 the Contributors to the Web Neural Network API Specification, published by the [Machine Learning for the Web Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable [summary](#) is available.

## Abstract

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

## Status of this document

This specification was published by the [Machine Learning for the Web Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn more about [W3C Community and Business Groups](#).

## § 1. Introduction

We're working on this section. Meanwhile, please take a look at the [explainer](#).

## § 2. Use cases

### § 2.1. Application Use Cases

This section illustrates application-level use cases for neural network inference hardware acceleration. All

Thanks

# Appendix

- WebNN spec: <https://webmachinelearning.github.io/webnn/>
- WebML CG: <https://www.w3.org/community/webmachinelearning/>
- NNAPI: <https://developer.android.com/ndk/guides/neuralnetworks>
- DirectML: <https://docs.microsoft.com/en-us/windows/win32/direct3d12/dml-intro>
- MPS: <https://developer.apple.com/documentation/metalperformanceshaders>
- OpenVINO: <https://docs.openvino toolkit.org/>
- TensorFlow.js: <https://js.tensorflow.org/>
- ONNX.js: <https://github.com/microsoft/onnxjs>
- Paddle.js: <https://github.com/PaddlePaddle/Paddle-Lite/tree/develop/web>
- OpenCV.js: [https://docs.opencv.org/3.4.10/d5/d10/tutorial\\_js\\_root.html](https://docs.opencv.org/3.4.10/d5/d10/tutorial_js_root.html)
- AI-benchmark: <http://ai-benchmark.com/>
- TensorFlow.js benchmark: <https://tensorflow.github.io/tfjs/e2e/benchmarks/>
- Wasm SIMD128: <https://github.com/WebAssembly/simd>
- AVX512-VNNI: [https://en.wikichip.org/wiki/x86/avx512\\_vnni](https://en.wikichip.org/wiki/x86/avx512_vnni)