Developing AI applications for the edge: **A new approach**

Exciting advances are happening on the HW, SW, and model fronts

by **Bill Eichen,** VP of Business Development at **DeGirum**

The past few years have seen a dramatic surge in the number of hardware accelerators for running AI inference at the edge. These offerings, instead of making life easier for developers, have turned the process of developing edge AI applications into a complex, expensive, time-consuming, and frustrating experience. Complex because it requires bringing together hardware, software, business logic, and machine learning (ML) models together. Expensive because developers need to invest time and money to bring up different hardware options. And time-consuming because each hardware option is accompanied by its own software stack and toolchain for porting models.

Choosing the right AI hardware

Let's delve deeper into the world of hardware (HW) selection to understand what makes it such an interesting and complex problem. You will have no doubt come across the various terms used by HW vendors to publicize the strength of their products: Trillion Operations Per Second (TOPS), TOPS/Watt, Frames Per Second (FPS), FPS/Watt, FPS/$, FPS/TOPS (believe me, I am not making up this metric), and so on. We will see why these metrics alone cannot guide HW selection.

A box of chocolates

The problem is that TOPS is like a box of chocolates (you never know what you're going to get), as Forest Gump might have said. Metrics like TOPS do not translate to real application performance in a standardized way.

Consider two hardware options: A and B. If A has 2x TOPS compared to B, this does not necessarily mean that—for a given ML model—A will have 2x FPS compared to B. This is true even if A and B are different products from the same company.

Or consider two ML models—M1 and M2—running on the same hardware. If M1 has 2x more compute than M2, this does not necessarily mean that M2 will have 2x FPS compared to M1. This is true even if M1 and M2 are the same model architecture running at different input resolutions.

One might argue that a normalized metric like TOPS/Watt addresses the above issue. Unfortunately, this metric is not measured in a standardized way. Some vendors measure only the power consumption of the matrix multiplication unit, which is responsible for most of the computation, while not accounting for power drawn by other components. Comparing SoCs with pure accelerators is another challenge, as end users are typically interested in overall power budget rather than the narrow scope for which numbers are published.

Everything and nothing

"What about absolute performance metrics like FPS," you might ask. "They measure the real performance of the hardware and should allow us to compare apples-to-apples, right?"

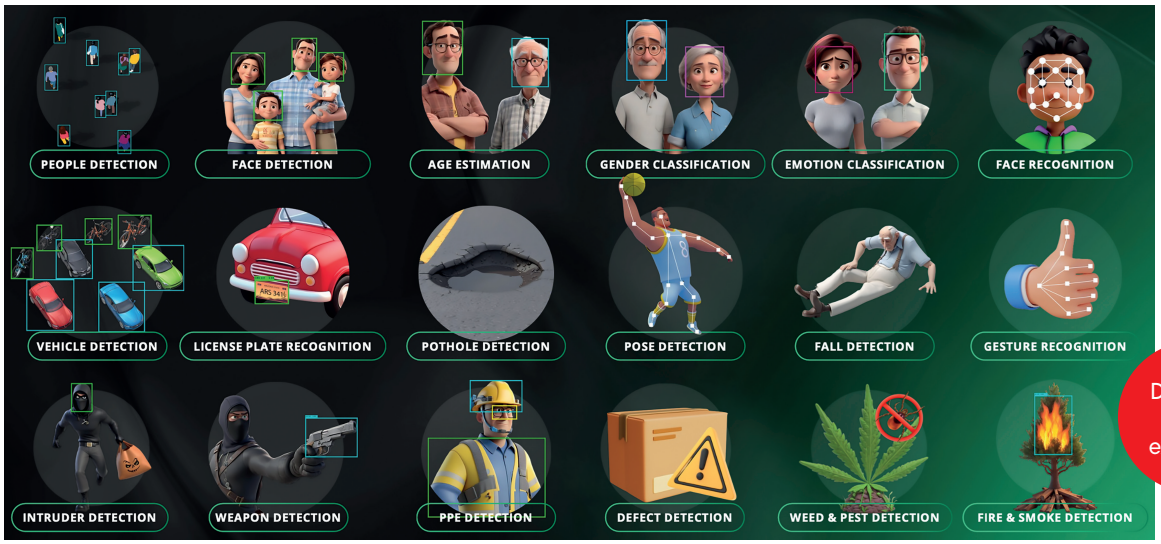
The reality is that FPS is a metric that tells you everything and nothing at the same time. While FPS can give us an idea of the power of the hardware, it is far from being perfect. For a given hardware, if you know FPS for one ML model, you cannot reliably estimate the FPS for other models. Even if FPS numbers for all the models you are interested in are published, there is no guarantee that you will achieve those numbers in real usage. This is because the conditions under which benchmarks are generated might not match the real use case. Application software load on the system can degrade the ML performance, or the real use case may not be able to supply inputs in batch sizes for which benchmarks are reported.

Compatibility conundrum

The varying capabilities of different hardware options also bring another important point into focus: the ML model compatibility conundrum. Anyone who has spent time porting an ML model to new hardware knows how frustrating this process can be. The trained model (typically a PyTorch or TensorFlow checkpoint) needs to be exported to a format (like ONNX or TFLite) that can be compiled for a hardware target.

For hardware that can run only quantized models, there needs to be an intermediate quantization step that may require providing some calibration data. These intermediate steps use tools that are not very reliable: we are in luck when they work, but in deep trouble when they don't. This is not to mention the hassle of getting all pre-processing and post-processing methods to match the original training settings to ensure that the ported models perform as well as the trained model.

CONTINUES ON PAGE 12



DeGirum model examples

> The problem is that TOPS is **like a box of chocolates** (you never know what you're going to get), as Forest Gump **might have said.**

Even without developing a solution, one can see that the right approach to solving these issues should have the following features:
1. Easy-to-use software that allows development of complex real-world AI applications.
2. Software that works with multiple hardware options.
3. A mechanism that allows hardware evaluation in a short amount of time with little to no investment in buying the hardware.
4. Reliable toolchains to optimize and port ML models to different hardware options.

DeGirum PySDK

Users can install DeGirum PySDK and get started with AI application development in under five minutes. You can even run the examples in Google Colab, which means you do not have to install any software (SW) locally to experiment with our SW.

Our PySDK supports three types of inference: Hosted, AI Server, and Local. It supports multiple HW options: DeGirum Orca, Intel CPUs/GPUs, Nvidia GPUs and SoCs, Google Edge TPUs, Arm SoCs (Raspberry pi etc), and AMD CPUs. And it supports advanced functionality such as tracking, slicing, and zone counting with simple to use APIs.

DeGirum's AI Hub

Users can now sign up for our AI Hub, which offers the following features:

Remote access: Access to multiple types of AI HW accelerators.

Hosted compiler: Go from PyTorch checkpoint to compiled model in a single click (including quantization). SOTA model architectures such as YOLOv5 and YOLOv8 are supported.

Inference in browser: Get performance estimates for different models on different hardware right in the browser without the need to install any SW.

Rich model zoos: Access to model zoos with the latest models such as YOLOv5 and YOLOv8 trained on different datasets and compiled for different hardware options.

Closing thoughts

This is an exciting time. We are in the middle of enabling AI on a vast number of use cases at the edge. Exciting advances are happening on the HW, SW, and model fronts at breakneck speed. For application developers to keep up with the pace of emerging technologies, there is a need for tools that can shorten the time between model development to model deployment.

www.degirum.ai

DeGirum application area examples