Dissecting Efficient Architectures for Wake-Word detection

Cody Berger, Juncheng B. Li, Yiyuan Li, Aaron Berger, Dmitri Berger, Karthik Ganesan, Emma Strubell, Florian Metze

Contact: codyberger@cmu.edu, junchel@cs.cmu.edu

A **practical baseline** exploring real-world performance of **wake-word detection** architectures on **parallel vs. sequential devices**.

Efficiency and accuracy do not linearly translate from CPU to GPU between models. This is due to models' **structural difference**s and varying **abilities to exploit hardware optimization.**

Post-training quantization is a promising option for increasing model efficiency in real-world contexts, noticeably decreasing models' inference time while not harming accuracy.

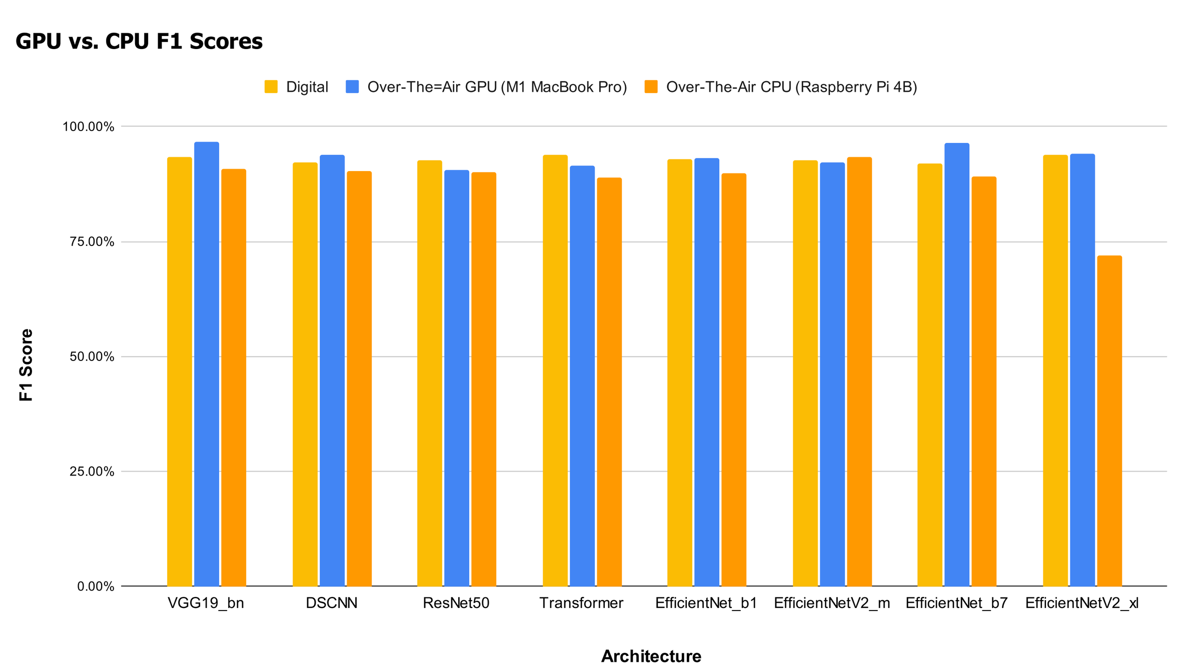
Methodology:

We trained eight models from six different architectures for **wake-word detection** using the Google **Speech Commands dataset.** Models were trained on GPU using PyTorch, and were not pretrained or fine-tuned.

Models were tested both **digitally and over-the-air** on both GPU and CPU.

In over-the-air trials, **real-time audio data** was sent as input to the models.

F1 Scores and Accuracy

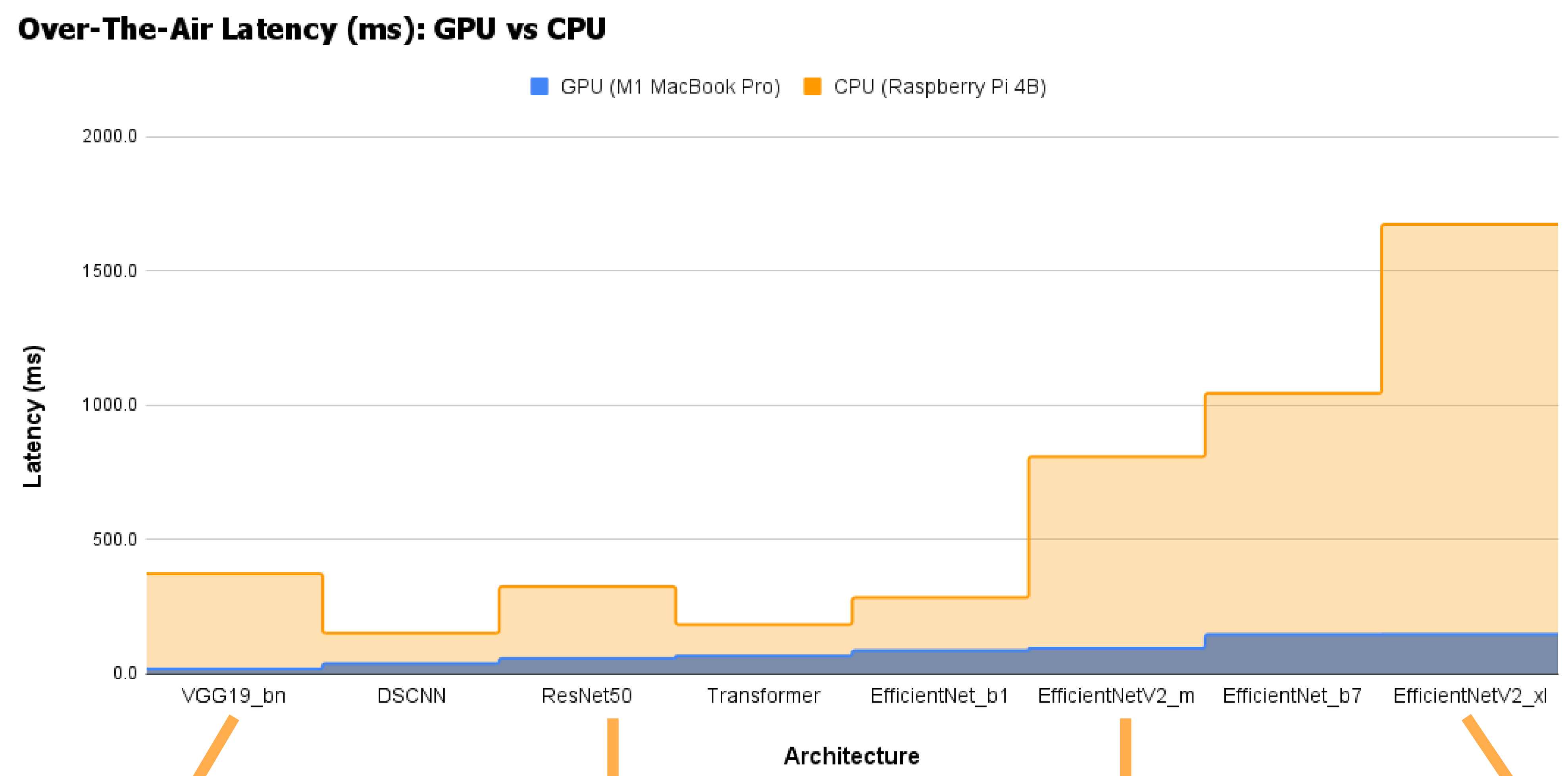


- Model performance often decreases moving from digital to over-the-air
- Almost all models performed better over-the-air on GPU than on CPU.

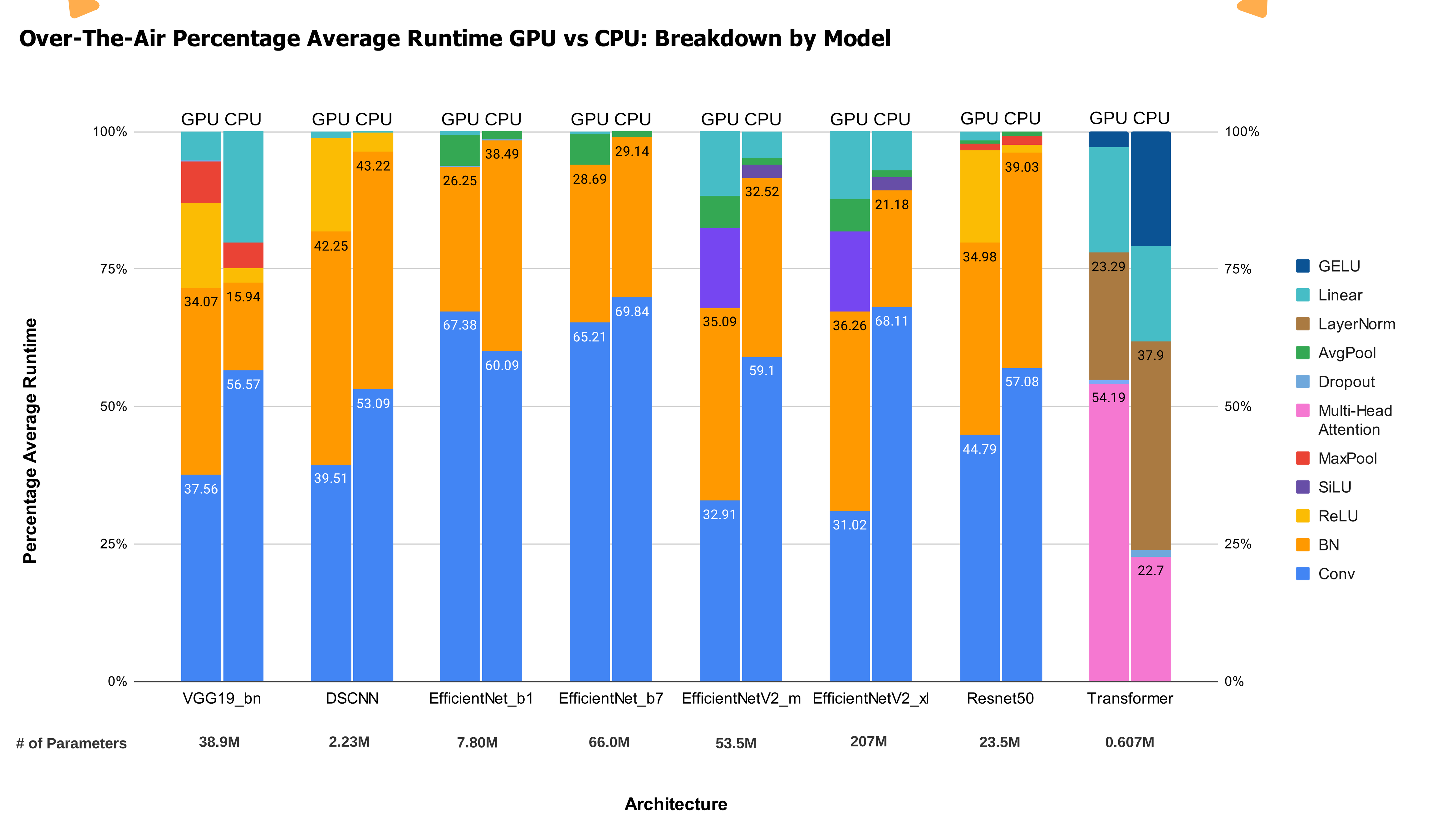
Is NAS Efficiency Transferable?

Models designed by Neural Architecture Search on GPU demonstrate poor efficiency on CPU, suggesting **NAS GPU optimization isn't necessarily applicable to CPU.**

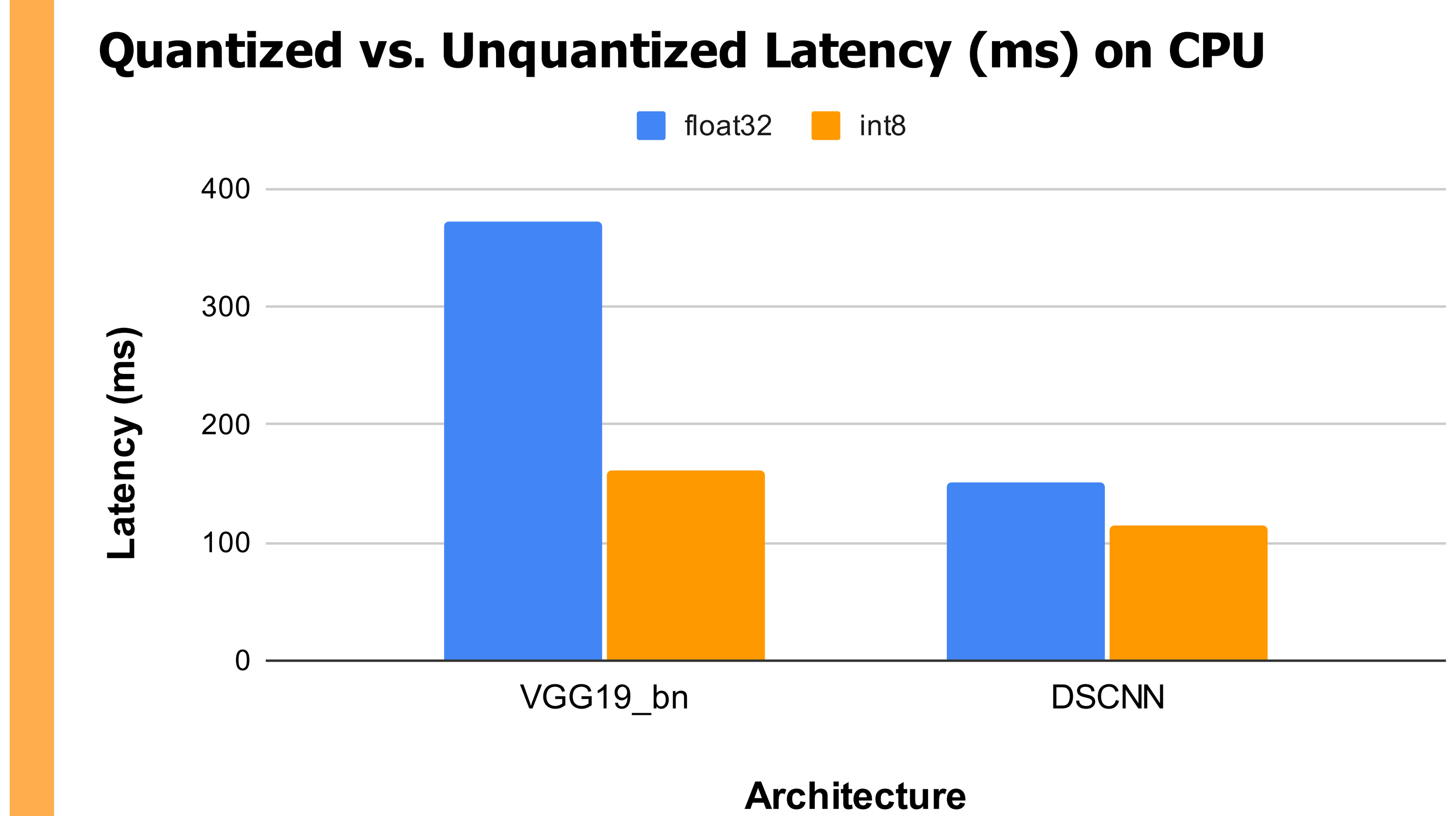
Layer Efficiency in the Spotlight



- **The fastest model on GPU is not always the fastest model on CPU.** Model structure has a strong impact on latency changes between GPU and CPU.
- For convolution-heavy models, this results from **hardware optimization for matrix multiplication.** Regular convolution is more parallelizable than MBConv blocks, but also has higher work, making it perform better on GPU than CPU, and vice versa.



Quantization



- Post-training quantization yields a noticeable **decrease in latency**; however, the decrease is **below the theoretical threshold** and is not consistent between models.
- This may result from **caching**, as VGG19_bn uses computation-heavy regular convolution, which is more likely to have poor locality than DSCNN's MBConv convolution.