



“Porting SYCL accelerated neural network frameworks to edge devices”

Hector Arroyo

Dylan Angus, Svetlozar Georgiev, Mehdi Goli

Tue 5 Dec 2023



Established 2002 in
Edinburgh, Scotland.

Grown successfully to around
100 employees.

In 2022, we became a **wholly
owned subsidiary** of Intel.



Committed to expanding the
open ecosystem for
heterogeneous computing.

Through our involvement in
oneAPI and SYCL
governance, we help to
maintain and develop open
standards.



Developing at the forefront
of **cutting-edge research.**

Currently involved in two
research projects - **SYCLOPS**
and **AERO**, both funded by
the Horizon Europe Project.

In today's presentation:

Introduce our work in porting PointNet neural network for edge devices through SYCL.

1. Edge Computing -> Examples, devices and relevancy
2. Use case -> Aerial point cloud segmentation with PointNet, hardware setup, backend framework
3. Neural network development -> PointNet background, data collection, training and results
4. Enabling with DCP++
 - 4.1 Benchmark configuration and initial performance
 - 4.1 Performance comparison after graph improvements
 - 4.3 Joint matrix support, Kernel tuning and energy analysis
5. Video Demo

What is Edge Computing?

- Distributed computing paradigm
- Encourages data to be processed and stored close to the source of origination.
- A direct result of compute resources being moved to a cloud-based framework
- Relevant in areas where bandwidth and latency are restricted and network stability, privacy, or security are un-reliable or insecure.

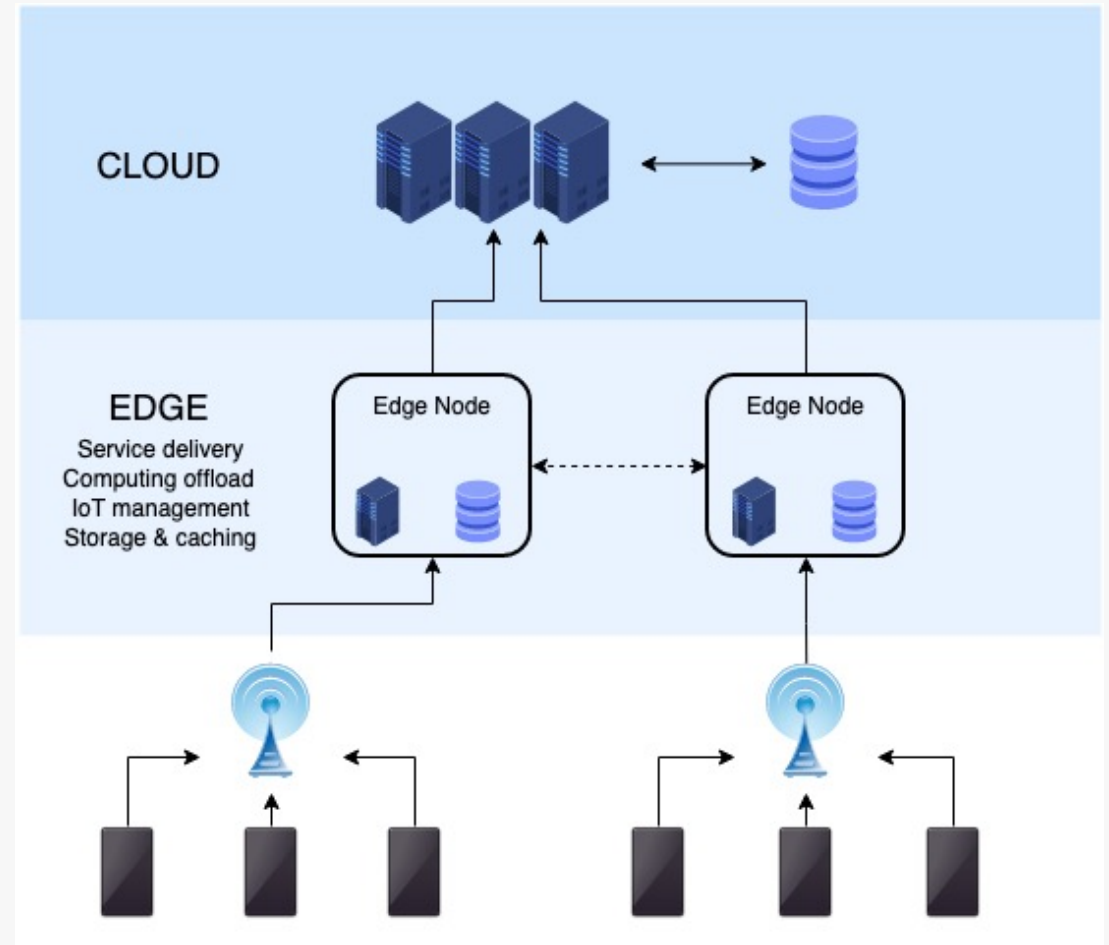


Figure 1. The edge computing infrastructure [14]

Edge Computing Examples

- Autonomous agriculture robots [3]
 - Often have numerous cameras connected to the host
 - Processing needed in areas where there can be no reliable connection to a cloud-based platform
- Bridge surveying drones [4]
 - Mapping and path-planning are needed with low latency
 - Benefit from a lightweight, compact, low-powered device
 - Especially when there are size and energy consumption requirements.



Figure 2. Example of agriculture robot. [3]



Figure 3. Example of drone surveying robot. [4]

What are Edge Devices?

Any piece of hardware that initially processes data as close to the origin as possible

- They can be the first entry point node into a wider (typically) cloud-based network of processing nodes filtering out unnecessary data and/or doing initial processing before passing on to the rest of the network.
- Work as small but compact computers, leverage onboard accelerators to tackle various Robotics, Computer Vision and AI tasks directly on the device without needing an external connection

Why are they important?

- Reduces latency and memory usage in network by filtering data
- In areas where bandwidth and latency are restricted and network stability, privacy, or security are un-reliable they act as localised processing nodes.

Use Case: Aerial Radar Point Cloud Segmentation

- SOTA to perform aerial segmentation of point clouds (radar data) to classify 5 different object classes typically presented in the aerial domain.
- Radar gathers points from around the scene and generates a point cloud of 2048 points at a frequency of 10Hz-20Hz. However this point count is highly variable in the aerial domain
- Using the PointNet Neural Network model we digest the point cloud and perform a classification task.
- A Jetson Xavier NX edge device is used to accelerate the model through the Onnxruntime framework.
- The Jetson benefits from being lightweight and portable, while pro due to latency and energy constraints.
- Full publication pre-print can be accessed in arXiv:
<https://arxiv.org/abs/2311.03221>



Figure 4. Radar and drone used in use-case.

Jetson Xavier NX

OS: Ubuntu 18.04

GPU:

- 384 NVIDIA CUDA cores
- 48 Tensor cores
- 2x NVDLA Engines
- GPU Max Frequency
 - 1100 MHz

CPU:

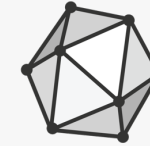
- 6-core NVIDIA Carmel ARM v8.2 64-bit
- 6 MB L2 + 4 MB L3 Cache
- CPU Max Frequency
 - 2-core @ 1900MHz
 - 4/6-core @ 1400MHz



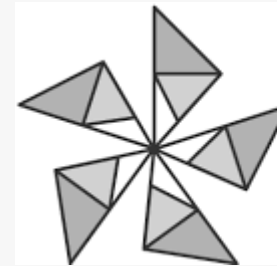
Figure 5. Jetson Xavier NX Device. [12]

Framework

- Open Neural Network Exchange (ONNX) [5] is an open-source artificial intelligence ecosystem of technology companies and research organizations that establish open standards for representing machine learning algorithms and software tools.
- ONNXRuntime [6] is the model loading library that implements the ONNX standard for different backends (E.g., CPU, CUDA, SYCL)
- Using the implemented SYCL backend of ONNXRuntime we build on work introduced in [2]
- Namely we enabled ONNXRuntime on an edge device and then tune the kernels [8] of our SYCL backend through portDNN[9] and portBLAS [7]



ONNX



ONNX
RUNTIME

Neural Network baseline model: PointNet

- PointNet [10] is a Neural Network Model Capable of digesting a point cloud and performing Classification, Part Segmentation and Semantic Segmentation.
- Leverages heavy matrix multiplication-based operators to perform tasks
- Well suited to handle the point cloud generated by the radar used on top of the drone
- Relies heavily on matrix-multiplication based operators which we will show benefit greatly from our kernel tuning

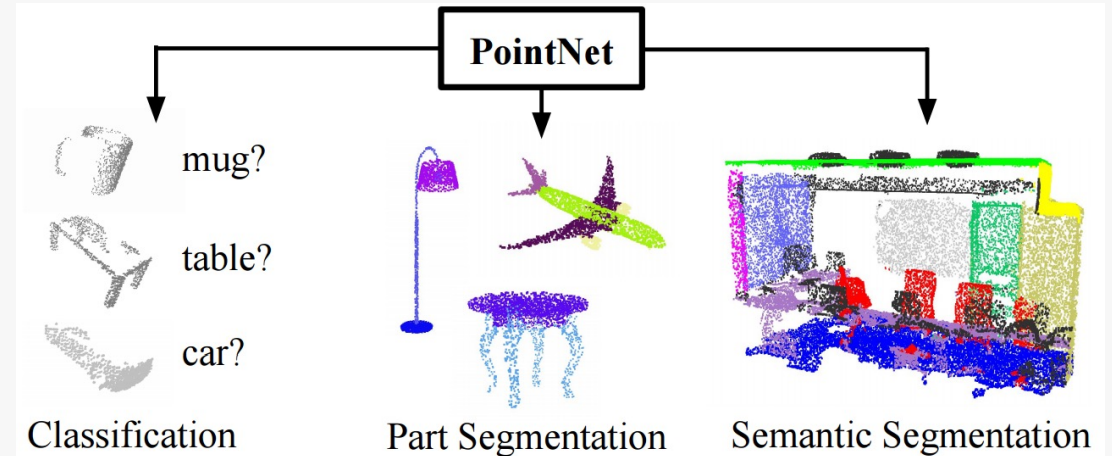


Figure 6 showing different scene understanding use cases of Point Net. [10]

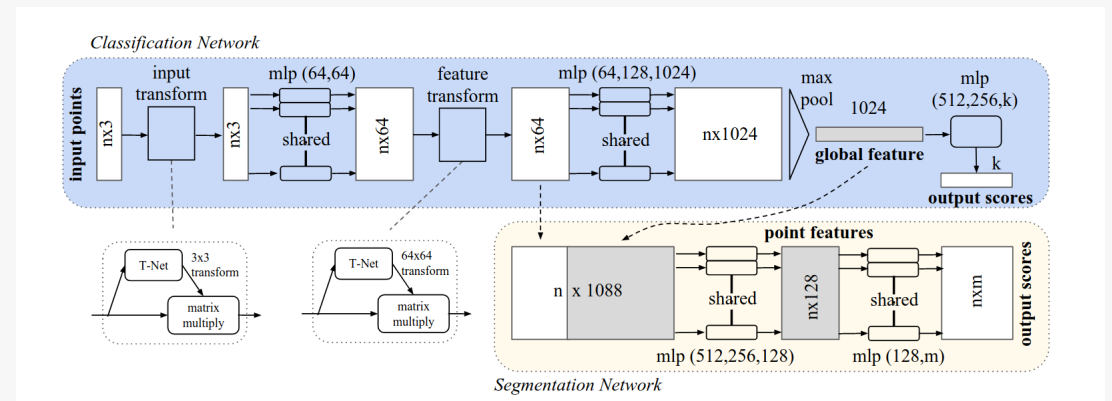


Figure 7 showing Point Net model architecture. [10]

Data Acquisition and Labelling

- Dataset acquisition and labelling: carried out in Strathaven Airfield (Glasgow)



Figure 8 showing different scene understanding use cases of Point Net. [10]
(Image source: Google Maps)

- The model is also able to discretize between ground and infrastructure returns.

	Class	Number of returns	%
Ground	1	779760	70.16
DJI M300 RTK	2	152065	13.68
Airplane	3	1487	0.14
DJI Mini	4	17539	1.58
Infrastructure	5	160589	14.45
Total		1111440	100

Figure 9: Overall dataset return point count.

- 5 object classes are captured and labelled in the radar point clouds, including three flying vehicles shown below




	<p>DJI M300 RTK drone</p> <ul style="list-style-type: none"> • Dimensions: 810 x 670 x 430 mm • Weight: 6.3 kg (with two batteries) • Materials: PVC, Carbon fiber • Max speed: 23 m/s (83 km/h)
	<p>DJI Mini drone</p> <ul style="list-style-type: none"> • Dimensions: 245 x 289 x 56 mm • Weight: 246 g • Materials: PVC, Carbon fiber • Max speed: 16 m/s (58 km/h)
	<p>Ikarus C42 aircraft</p> <ul style="list-style-type: none"> • Dimensions: 6.2 x 9.45 x 2.34 m • Empty weight: 265 kg • Materials: Aluminum, composites • Max speed: 175 km/h

Figure 10. Specifications of mobile targets included in the study

Neural Network Training

1) Prove the superiority of a DL model by comparing it with a simpler ML model

2) Find an optimized PointNet architecture for aerial radar point clouds (sparse and highly variable point count throughput)

3) Train the final model with Tensorflow and get performance results on the held-out data

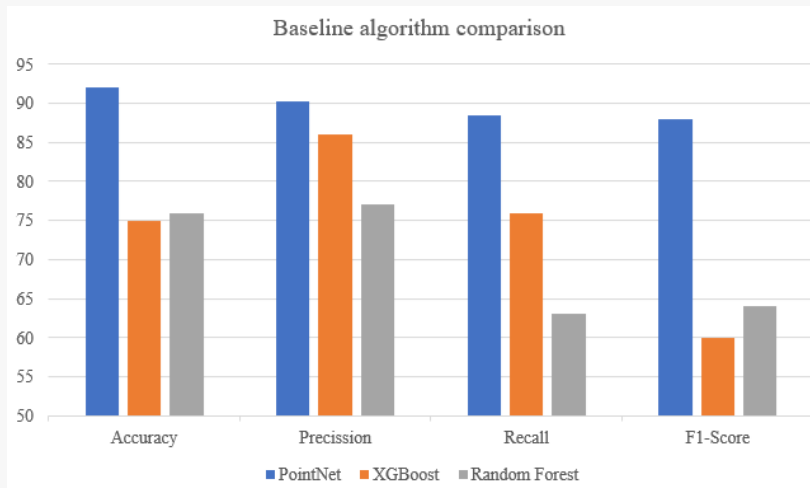


Figure 11. ML vs DL approaches for aerial point cloud segmentation

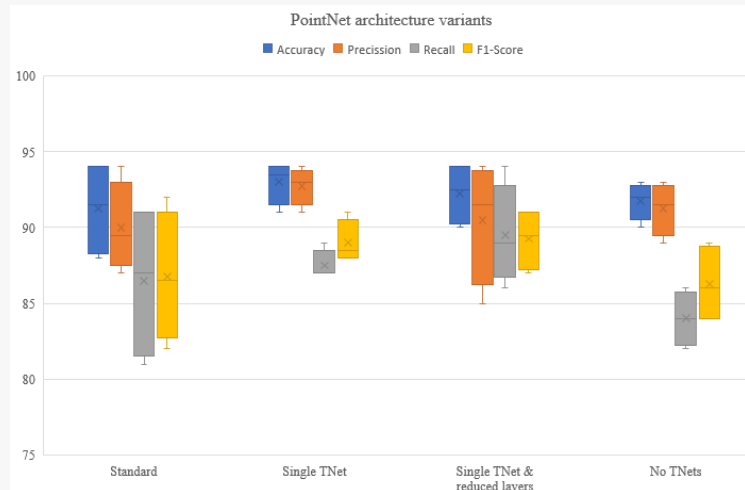


Figure 12. PointNet optimization search

	Precision	Recall	F1-Score	Accuracy
Ground	0.92	0.98	0.95	0.98
DJI M300 RTK	0.92	0.88	0.9	0.89
Airplane	0.98	0.99	0.98	0.99
DJI mini	0.86	0.67	0.75	0.67
Airfield Infr	0.99	0.92	0.96	0.92
Combined	0.93	0.89	0.91	0.94

Figure 13. Result metrics on the held-out data

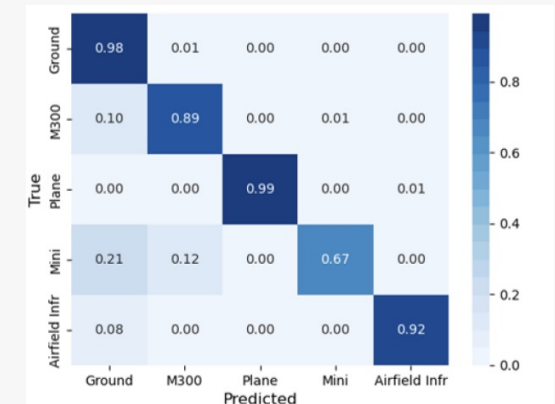


Figure 14. Confusion matrix on the held-out data

Results Visualization

Bird's eye view projection

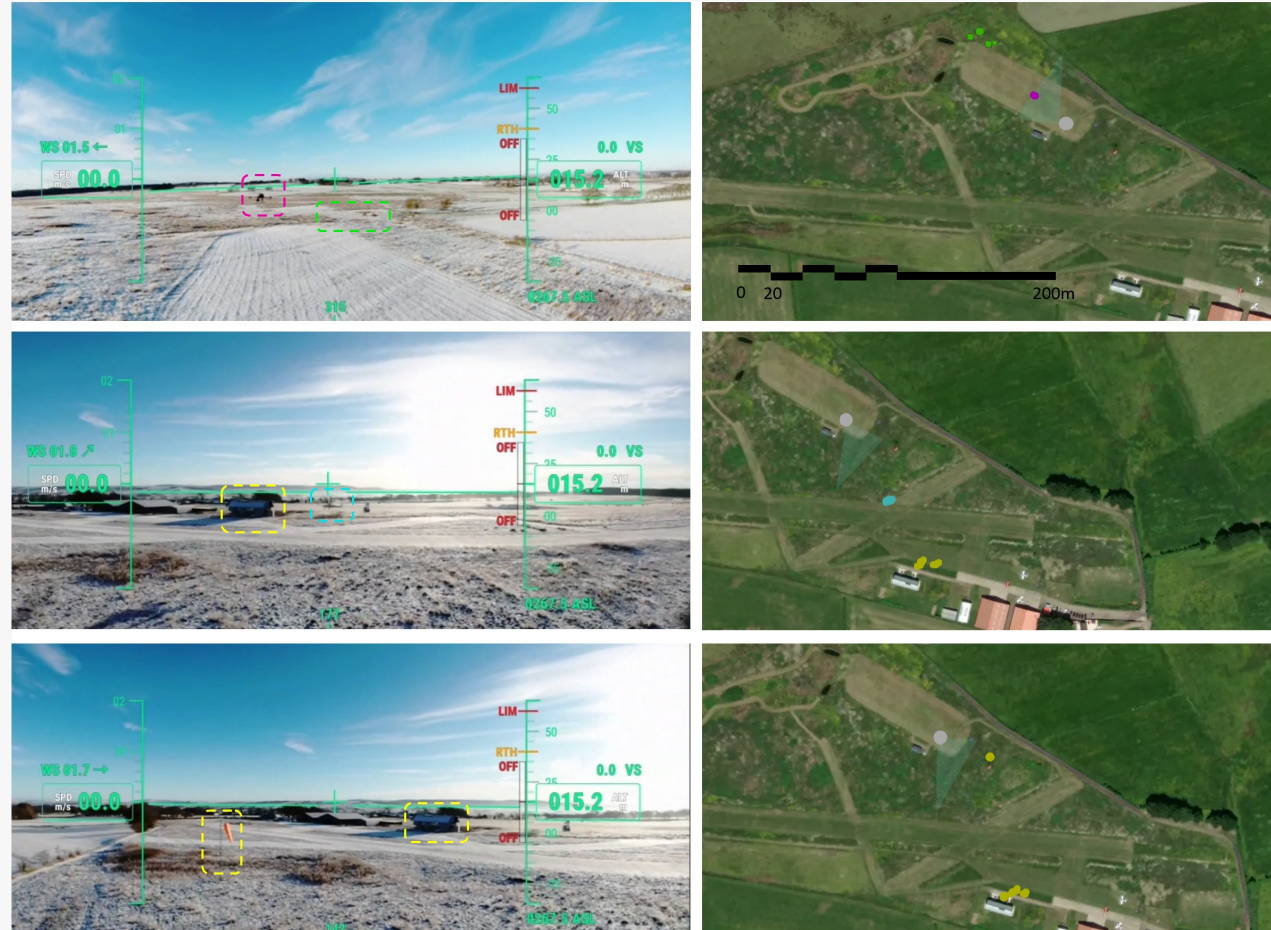


Figure 15. PointNet inference results projected into bird's eye view (BeV)

First person view projection

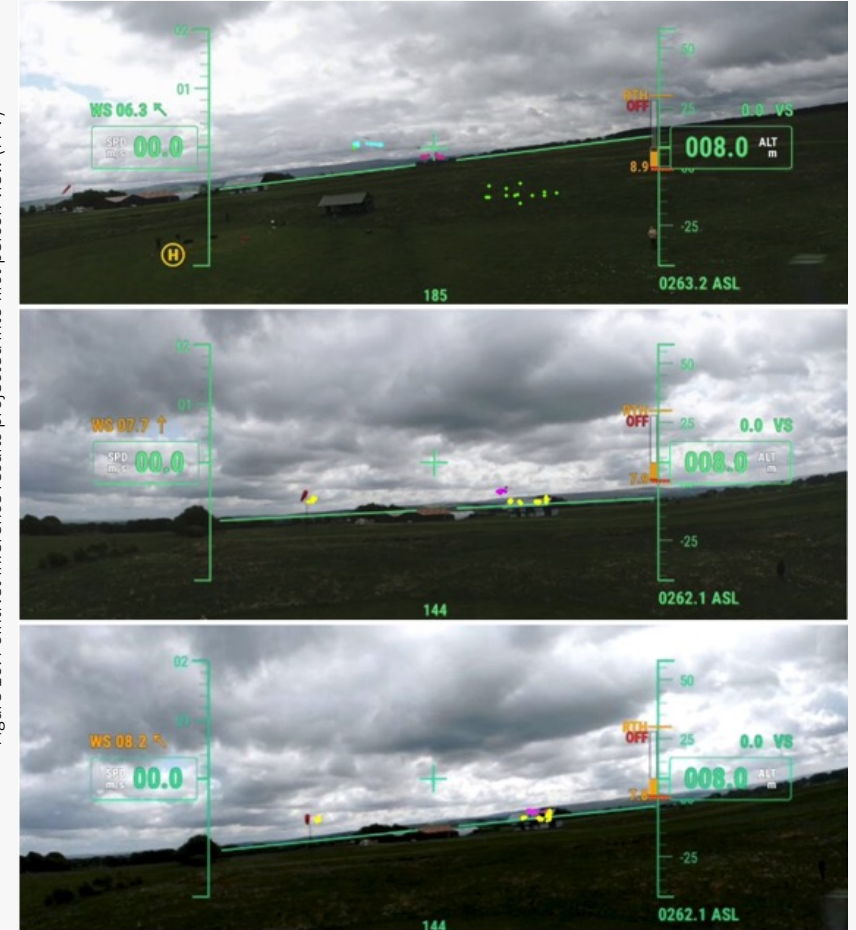


Figure 16. PointNet inference results projected into first person view (FPV)

- Detected target (illustration only)
- Location and orientation of radar drone
- Ground
- Infrastructure
- DJI RTK M300
- Airplane

Enabling with DPC++

- From Source

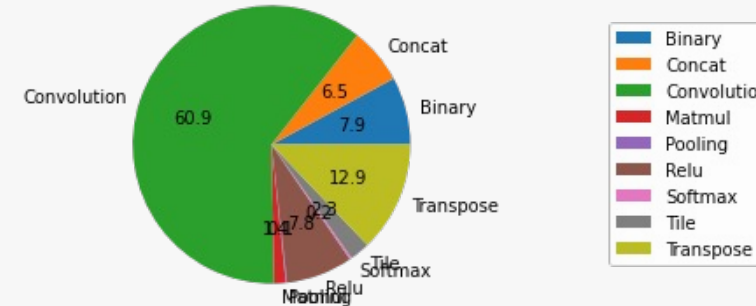
- Cloned source from <https://github.com/intel/llvm>
- Commit Hash 4a4702e2e992
- Compiled with AArch64 as the host target and CUDA as the device target
- CUDA version 10.2

Benchmark Configurations

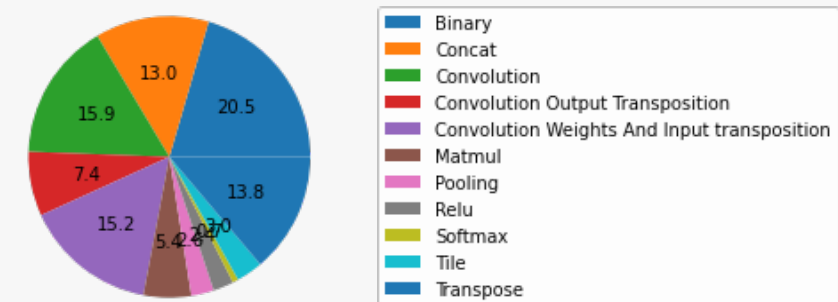
- Configurations Details and Workload Setup:
 - Hardware:
 - 384-core NVIDIA Volta™ GPU with 48 Tensor Cores
 - 6-core NVIDIA Carmel ARM®v8.2 64-bit CPU
 - 8 GB 128-bit LPDDR4x
 - Software:
 - Ubuntu 18.04.5 LTS
 - Kernel Version: 4.9.201-tegra
 - L4T 32.5.1 [JetPack 4.5.1]
 - CUDA 10.2.89
 - CUDNN: 8.0.0.180
 - SYCL Open Source Clang 17.0.0
 - Compiler switches: `-fsycl-targets=nxptx64-nvidia-cuda`
 - NVIDIA-NVCC V10.2.89
 - Compiler switches: `-O3 -gencode -arch=compute_72, code=sm_72`
- All Benchmarks were run for 1000 iterations on the 15W 2CPU configuration mode unless otherwise specified.
- We compared results were compared to the CUDA backend of Onnxruntime to the SYCL backend targeting the GPU through DPC++.
- Testing Date: Performance results are based on testing by Codeplay as of April 6th, 2023, and may not reflect all publicly available updates
- Performance varies by use, configuration and other factors.

Initial Operator Performance

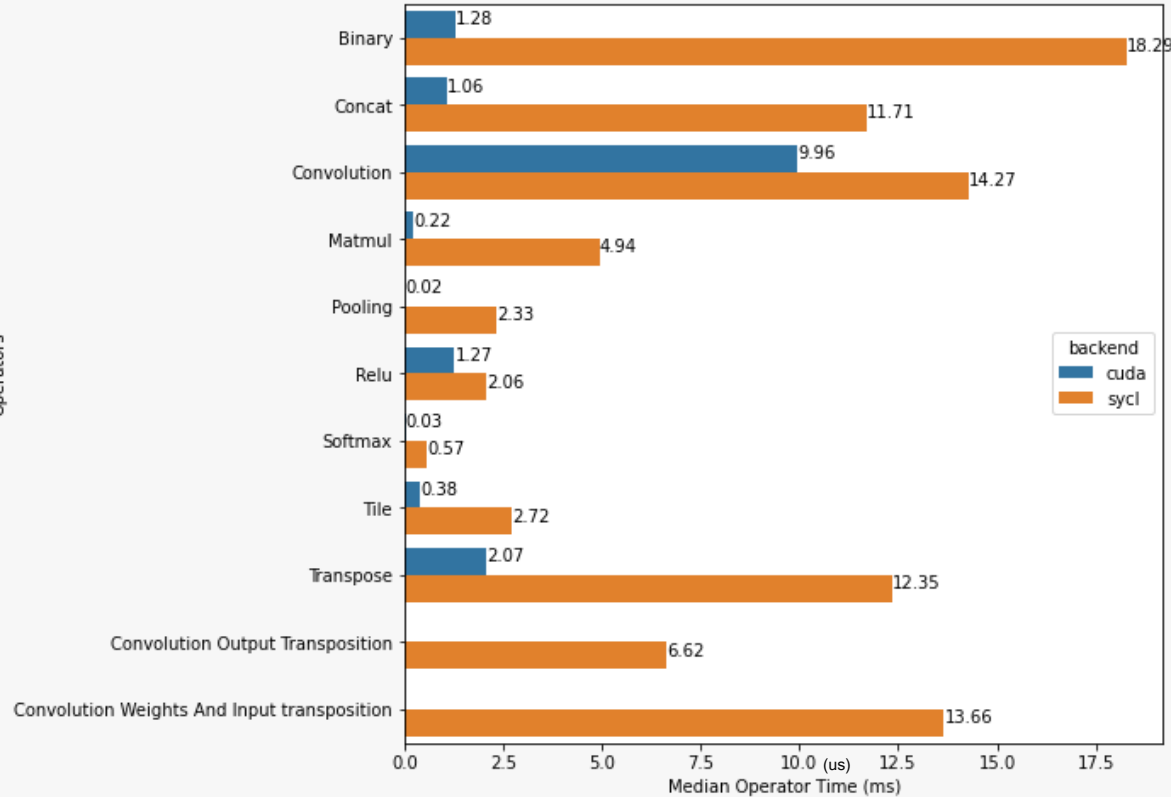
CUDA Operator Time Distribution



SYCL Operator Time Distribution



SYCL (89.5 ms) vs CUDA (16.3 ms) Operator Time



See backup for workloads and configurations. Results may vary.

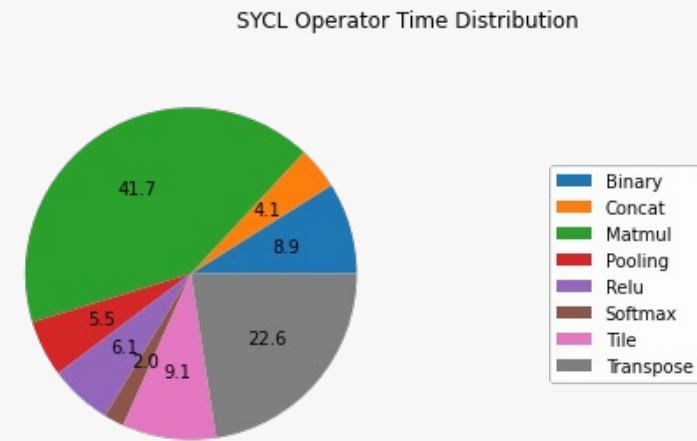
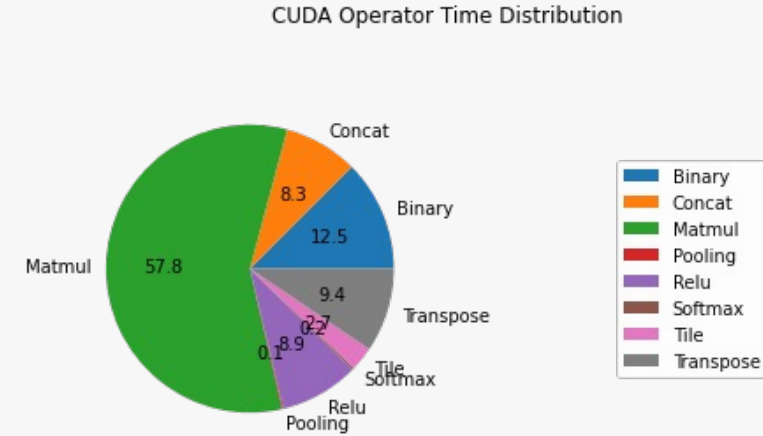
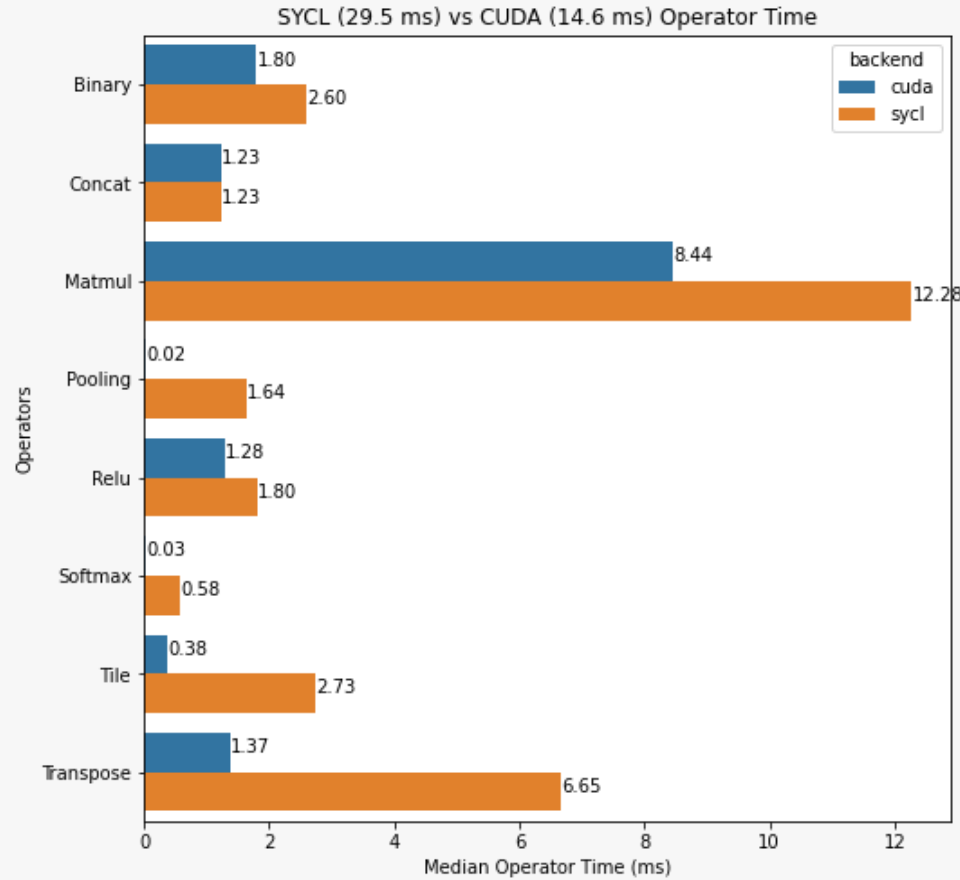
Fallbacks

- ONNX only supports NCHW format while portDNN primarily supports and is optimised for NHWC
- Unnecessary transposes were added into the model during conversion to ONNX format.
- Concat, Broadcasted BinaryOp, Softmax and Pooling implementations need to be optimised
- CUDA backend leveraged Jetson's TensorCores while the SYCL backend didn't

Graph Improvements

- Removed Unnecessary transposes and other redundant operators
- Converted 1D Convolutions to MatMul
- Tiled the static binary operator input offline to avoid runtime broadcast

Operator Performance After Graph Improvements



See backup for workloads and configurations. Results may vary.

Nvidia Tensor Cores

- Programmable dedicated matrix-multiply-and-accumulate units
- Each Tensor Core provides a 4x4x4 matrix processing array which performs the operation $D = A * B + C$, where **A**, **B**, **C** and **D** are 4x4 matrices
- Each Tensor Core performs 64 floating point FMA mixed-precision operations per clock (FP16 input multiply with full-precision product and FP32 accumulate)
- Interface provide specialized matrix load, matrix multiply and accumulate, and matrix store operations to enable cores.

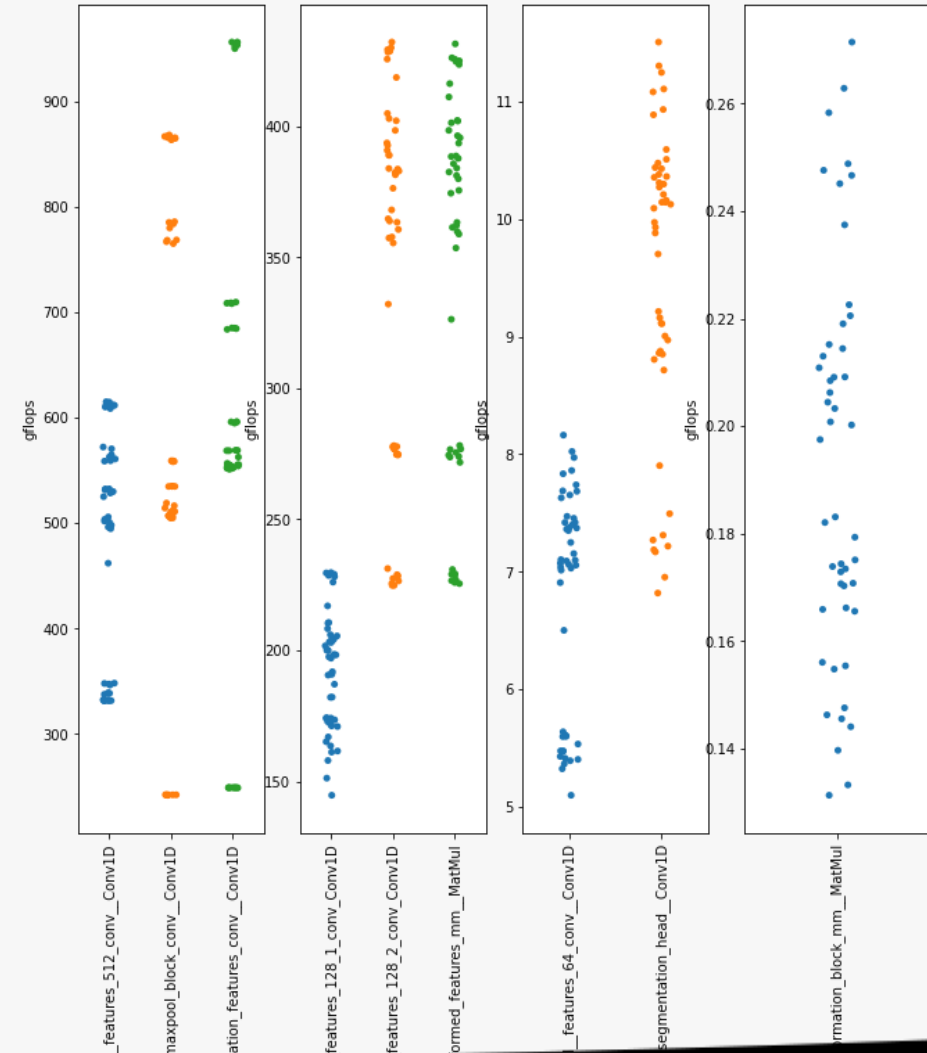
Joint Matrix Support

- Experimental matrix extension to DPC++
- Generalised interface intended to unify different tensor hardware: *Intel® Advanced Matrix Extensions (Intel® AMX)*, *Intel® Xe Matrix Extensions (Intel® XMX)* and *Nvidia® Tensor Cores*
- portBLAS now supports sub-group based collective GEMM operation which is used by the MatMul operation
- Support can be enabled or disabled at runtime through the environmental variable `SB_ENABLE_JOINT_MATRIX`

Kernel Tuning

- portBLAS and portDNN exposes underlying kernel configurations through templates.
- Allowing kernels to be tuned to specific devices as seen and shown in [7]
- We were able to extend the configuration space to include the joint matrix parameters as well.

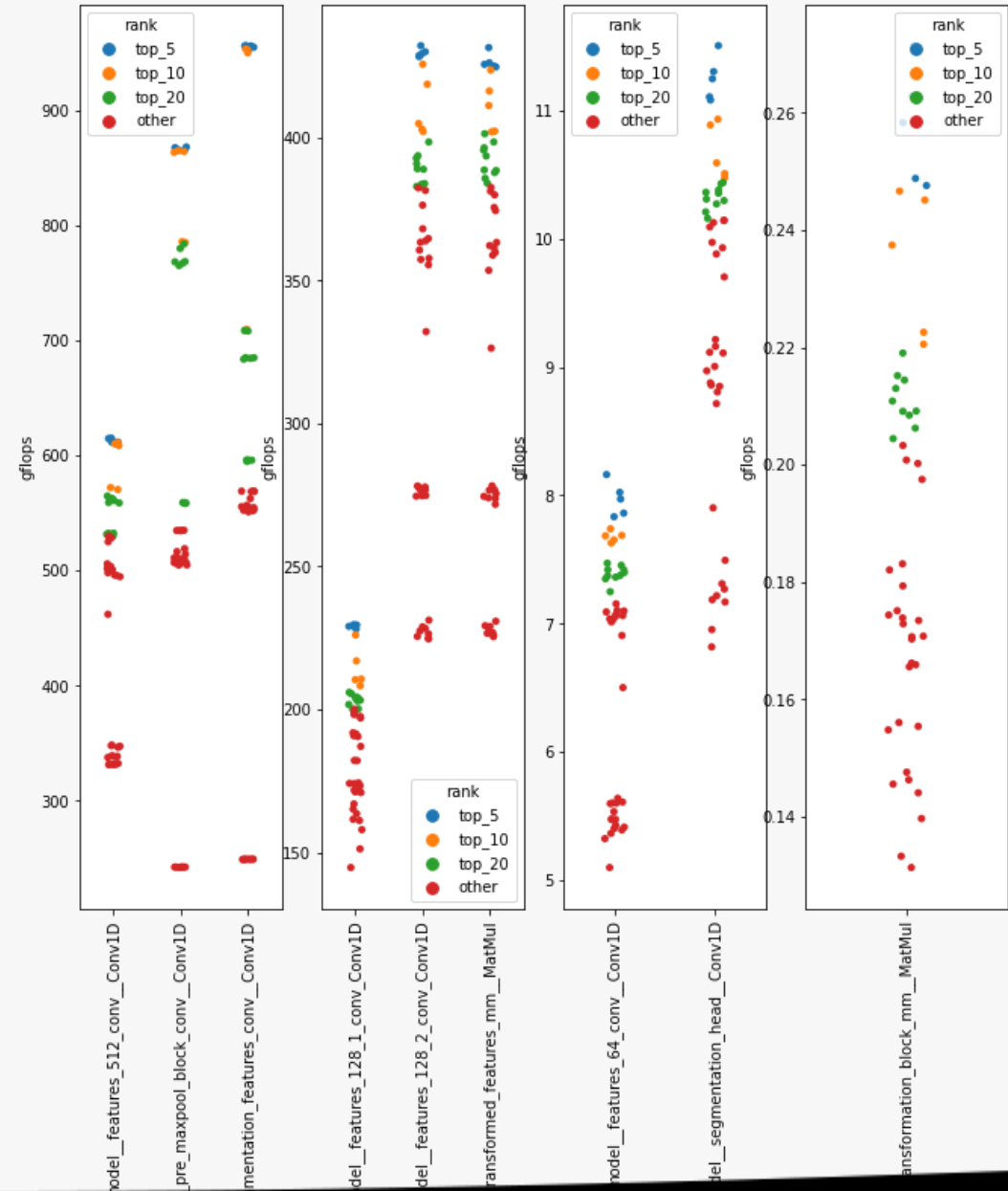
MatMul performance for each kernel configuration



See backup for workloads and configurations. Results may vary.

Configuration Selection

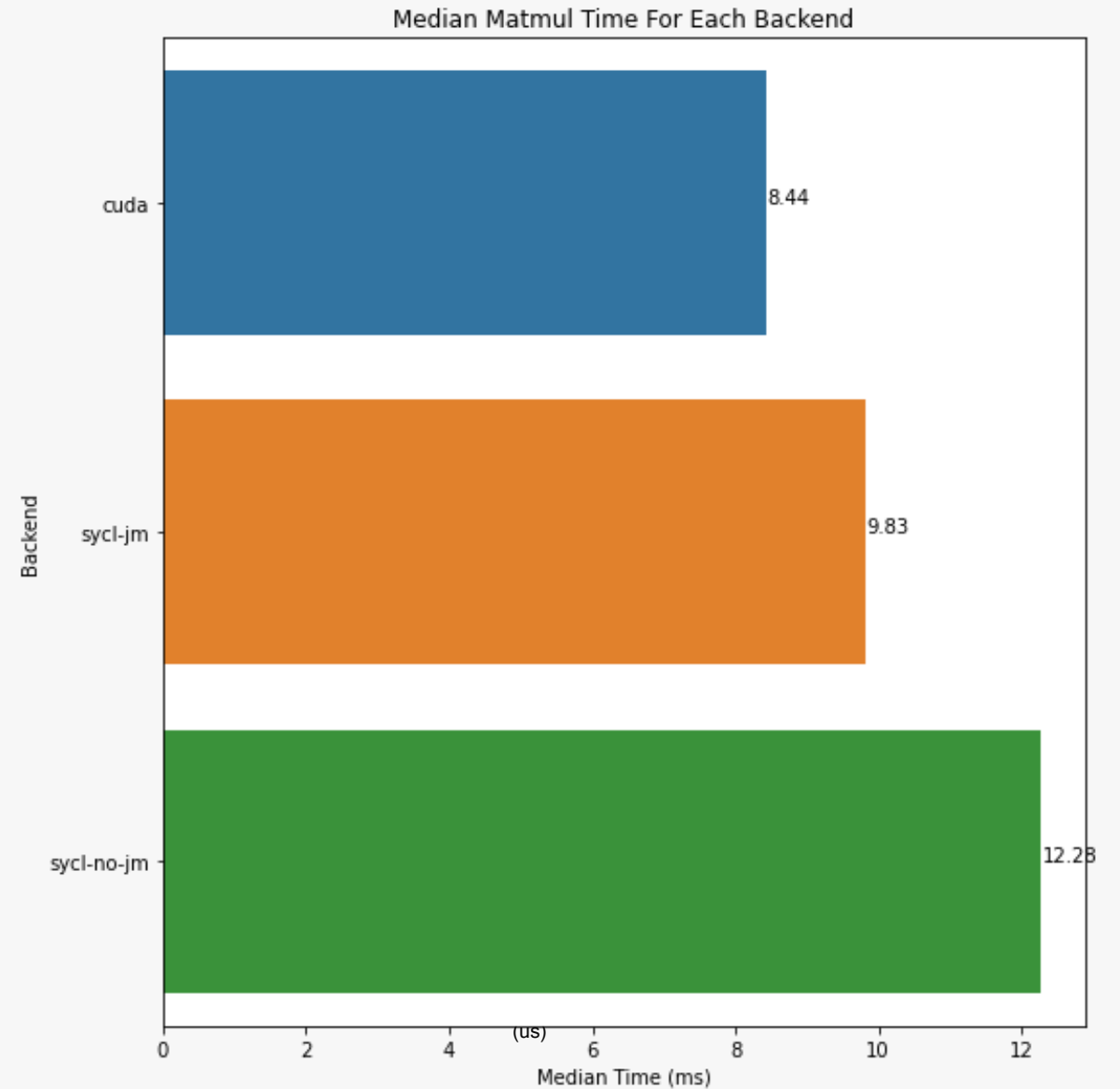
- The same top configurations do well when grouped with similar sized operations
- This allows us to save on library size as we can reuse the same kernel specification for multiple nodes



See backup for workloads and configurations. Results may vary.

MatMul Performance With Tuned Joint Matrix

- Tuned Joint Matrix implementation yielded a ~20% improvement from baseline
- Tuned Joint Matrix implementation has ~86% performance of the CUDA implementation



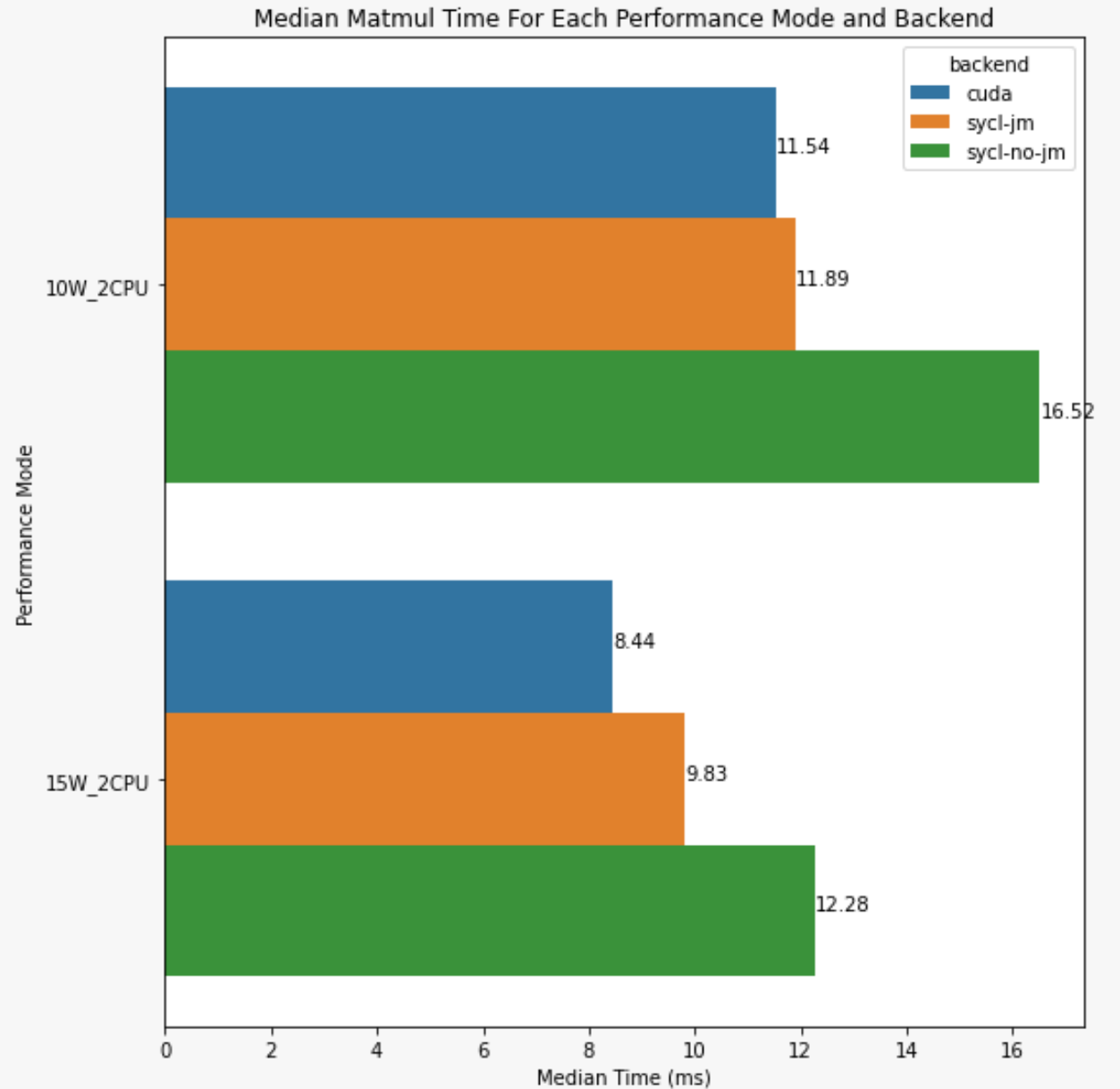
See backup for workloads and configurations. Results may vary.

Energy Analysis

- The Jetson is highly configurable
- GPU, CPU and EMC clock frequencies are adjustable
- Control over the number of CPU cores being used
- Made use of the 15W mode with 2 CPU Cores and 10W mode with 2 CPU Cores configurations

Energy Analysis

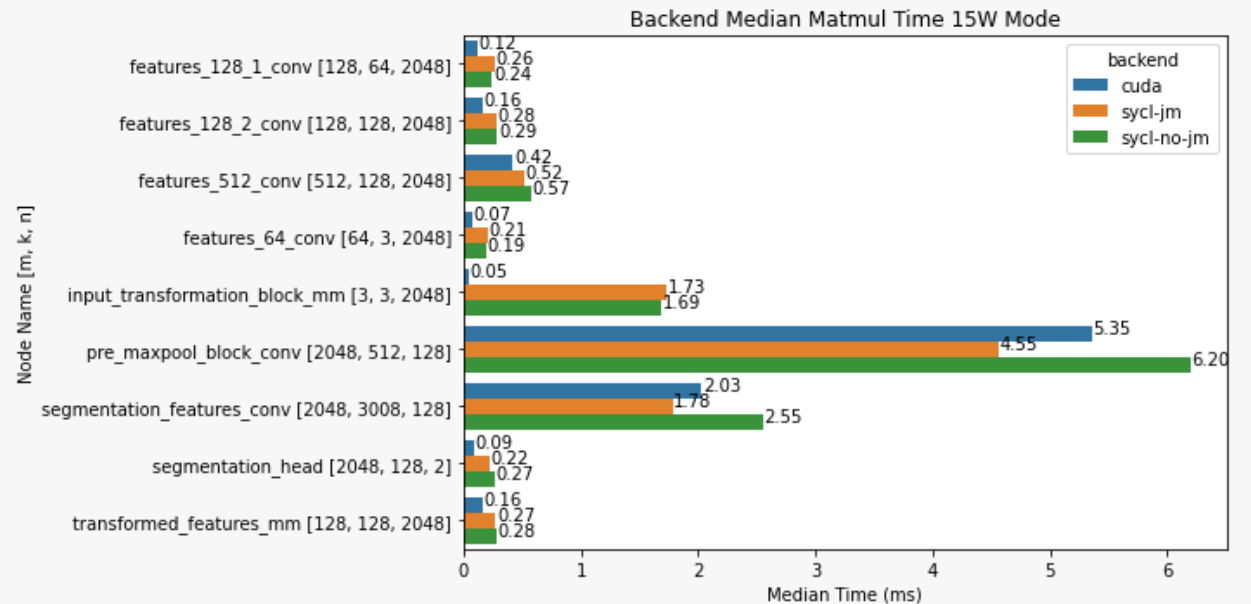
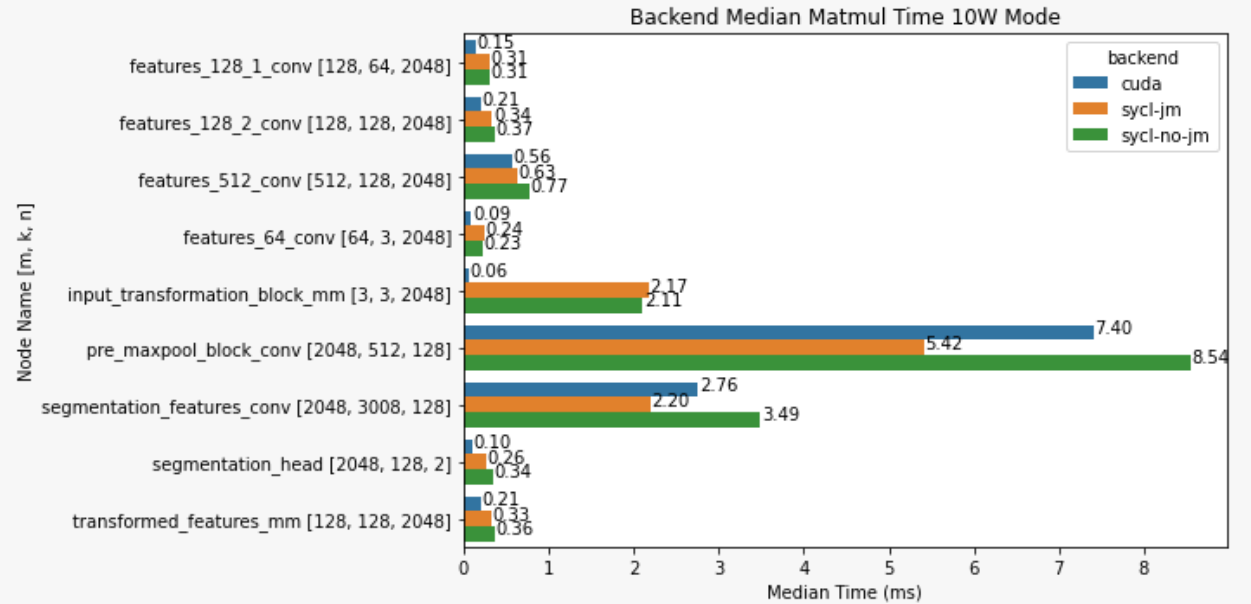
- JM SYCL backend achieved 97% of the performance of the CUDA backend in 10W mode vs the 86% performance in the 15W mode
- JM SYCL backend in 10W had a 28% improvement from its baseline
- JM SYCL backend in 10W outperformed the 15 mode SYCL backend without JM



See backup for workloads and configurations. Results may vary.

MatMul Shape Analysis

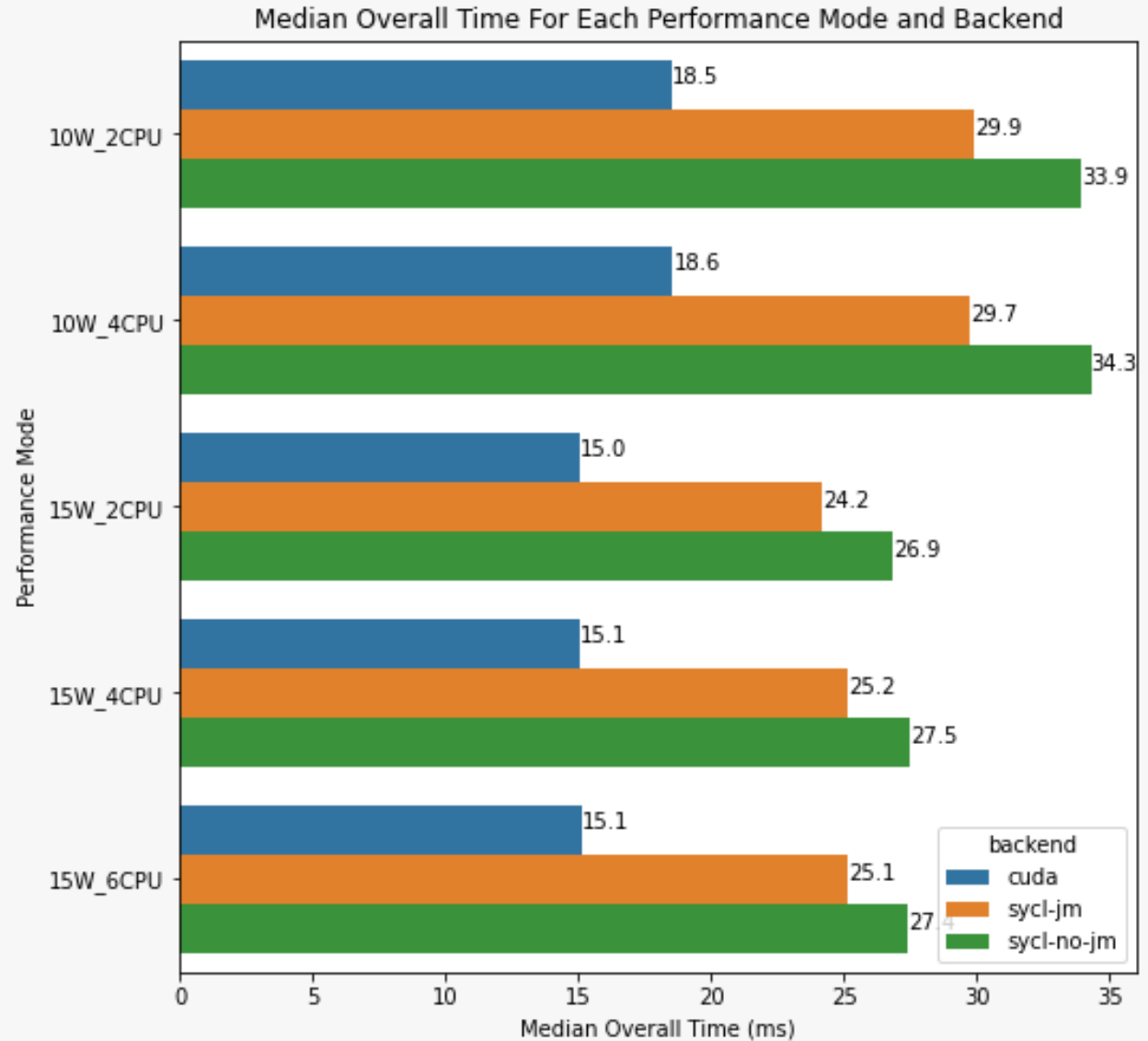
- SYCL backend performs worse for smaller sizes. This may be fixed by on-going work in SYCL Graphs
- Found a performance bug for the shape [3, 3, 2048]
- JM SYCL outperforms CUDA for the two largest matrix shapes



See backup for workloads and configurations. Results may vary.

Energy Analysis of Overall Execution

- Overall comparable performance between the SYCL backend and the CUDA backend.
- For both the 10W and 15W mode the SYCL backend is operating at about 62% the performance of CUDA with major room for improvement.
- More than capable of operating at the 10Hz-20Hz execution frequency needed for use case.



See backup for workloads and configurations. Results may vary.

Video Demo



References

- [1] Yeung, Tiffany. “What Is Edge Computing?” *NVIDIA Blog*, 3 Jan. 2023, <https://blogs.nvidia.com/blog/2019/10/22/what-is-edge-computing/>.
- [2] Tanvir, M., Narasimhan, K., Goli, M., El Farouki, O., Georgiev, S. and Ault, I. (2022). Towards performance portability of AI models using SYCL-DNN. *International Workshop on OpenCL*, <https://doi.org/10.1145/3529538.3529999>.
- [3] Bayer. “Agricultural Robotics.” *Agricultural Robotics | Bayer Global*, 25 Jan. 2023, <https://www.bayer.com/en/agriculture/article/ripe-robots>.
- [4] Victor. “UAV Bridge Inspections: Award Winning Drone Inspection Services.” *Baltimore Inspection Services*, Baltimore Inspection Services, 10 Aug. 2021, <https://baltimoreuav.co.uk/bridge-inspection-services/>.
- [5] Bai, Junjie, et al. ‘ONNX: Open Neural Network Exchange’. GitHub Repository, GitHub, 2019, <https://github.com/onnx/onnx>.
- [6] Developers, Onnx Runtime. ONNX Runtime. 2021, <https://onnxruntime.ai/>.
- [7] Aliaga, José & Reyes, Ruyman & Goli, Mehdi. (2017). SYCL-BLAS: Leveraging Expression Trees for Linear Algebra. 1-5. 10.1145/3078155.3078189.
- [8] Lawson, John, and Mehdi Goli. ‘Performance Portability through Machine Learning Guided Kernel Selection in SYCL Libraries’. *Parallel Computing*, vol. 107, 2021, p. 102813, <https://doi.org/10.1016/j.parco.2021.102813>.
- [9] Rod Burns, John Lawson, Duncan McBain, and Daniel Soutar. 2019. Accelerated Neural Networks on OpenCL Devices Using SYCL-DNN. In *Proceedings of the International Workshop on OpenCL (IWOCCL'19)*. ACM, New York, NY, USA, Article 10, 4 pages. DOI: <https://doi.org/10.1145/3318170.3318183>
- [10] Qi, Charles R., et al. ‘PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation’. *ArXiv Preprint ArXiv:1612.00593*, 2016.
- [11] Arroyo, Hector, et al. ‘Segmentation of Drone Collision Hazards in Airborne RADAR Point Clouds Using PointNet’ *ArXiv Preprint | arXiv.2311.03221*
- [12] Yeung, Tiffany. “What Is Edge Computing?” *NVIDIA Blog*, 3 Jan. 2023, <https://blogs.nvidia.com/blog/2019/10/22/what-is-edge-computing/>.
- [13] “Nvidia Jetson Xavier NX for Embedded & Edge Systems.” *NVIDIA*, <https://www.nvidia.com/en-au/autonomous-machines/embedded-systems/jetson-xavier-nx/>.
- [14] NoMore201 (2019). Available at: https://commons.wikimedia.org/wiki/File:Edge_computing_infrastructure.png [Accessed 1 Dec. 2023].

Notices & Disclaimers

Performance varies by use, configuration and other factors.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

© Codeplay Software Ltd.. Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Questions?