ETH zürich

IIS – Integrated Systems Laboratory

A Microcontroller is All You Need:
Enabling Transformer Execution on Low-Power IoT Endnodes

Alessio Burello, **Moritz Scherer**, Marcello Zanghieri, Francesco Conti, Luca Benini
PhD Student, ETH Zürich
23.08.2021, IEEE COINS

TinyML and its importance in the IoT

- Artificial Intelligence on the edge is pervasive
  - Over 75 Billion end devices by 2025[1]
  - Extracting information on the edge improves privacy and security
  - Only sending relevant results lowers communication energy

- Most IoT data is sequential / time-series and time critical
  - Video
  - Accelerometer
  - Audio
  - RADAR / LIDAR

- **Most of this data lends itself very well to Deep Learning algorithms, CNNs especially**

[1]: https://www.businessinsider.com/75-billion-devices-will-be-connected-to-the-internet-by-2020-2013-10?r=US&IR=T

Neural Networks on Microcontrollers

- There is a significant body of literature on employing CNNs on the edge[2,3]
  - Need quantization to reduce compute and memory, typically 8 Bit
  - Most frameworks are still in their infancy
  - Data marshalling operations are costly

- Most frameworks ‘blindly’ implement a computational graph
  - Little checking for optimization potential
  - There is a large performance gap, even between identical networks
  - Very limited support for ‘exotic’ operators

- **CNNs are generally well supported, but there is lots of potential for optimization**

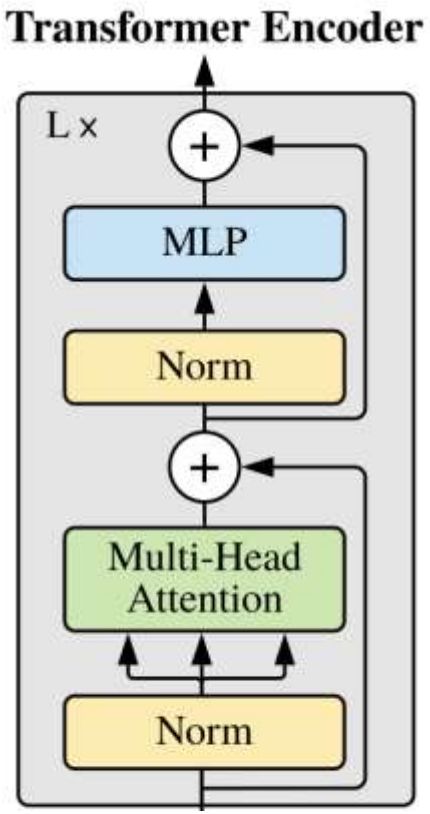
[2]: David et al.: “Tensorflow Lite Micro: Embedded Machine Learning on TinyML Systems”
[3]: Lai et al.: “CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs”

The Transformer Architecture

- Transformers are originally designed for language modelling
  - Inputs are 2D-Tensors of dimensions S(equence)xE(mbedding)
  - Multi-Head Attention relates all inputs globally

- Transformers perform well on many problems
  - Computer Vision[4]
    - Most accurate network on ImageNet is a Transformer[5]
  - Audio Processing[6]
  - And many more![7,8]

- **Multi-Head (Self-)Attention is the key innovation in Transformers**

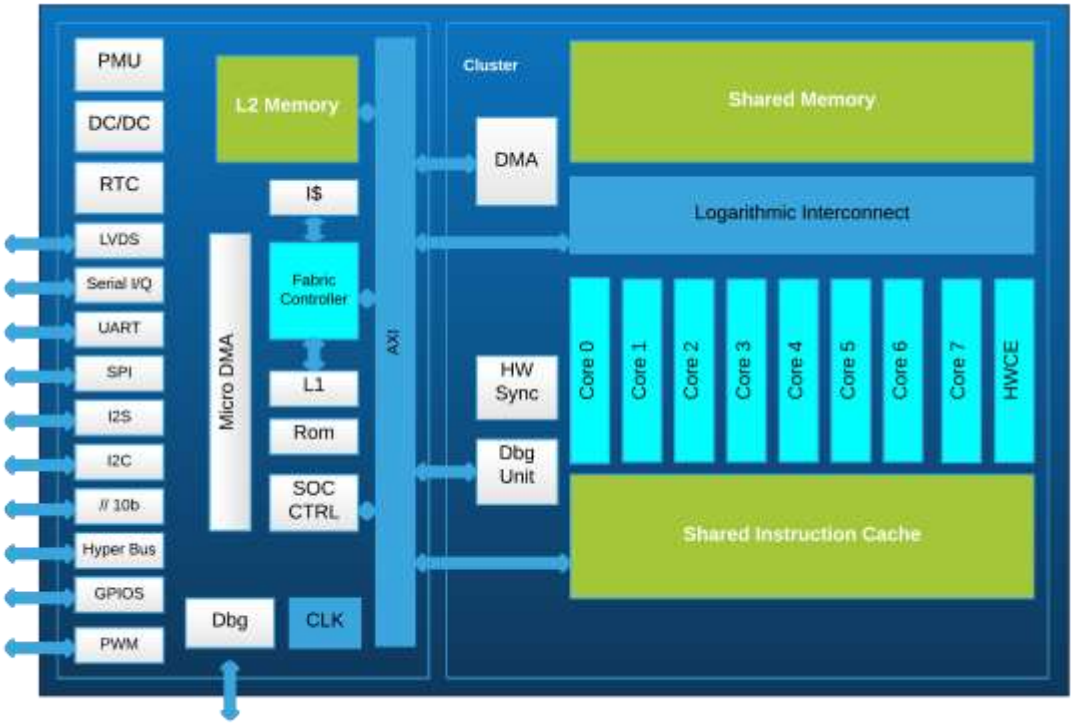


Dosovitsky et al.: “An Image Is Worth 16x16 Words: Transformers for Image Recognition At Scale”

[4]: Dosovitsky et al.: “An Image Is Worth 16x16 Words: Transformers for Image Recognition At Scale”
[5]: Zhai et al.: “Scaling Vision Transformers”
[6]: Gong et al.: “AST: Audio Spectrogram Transformer”
[7]: Chen et al.: ”TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation”
[8]: Strudel et al.: “Segmenter: Transformer for Semantic Segmentation”

PULP & GAP8

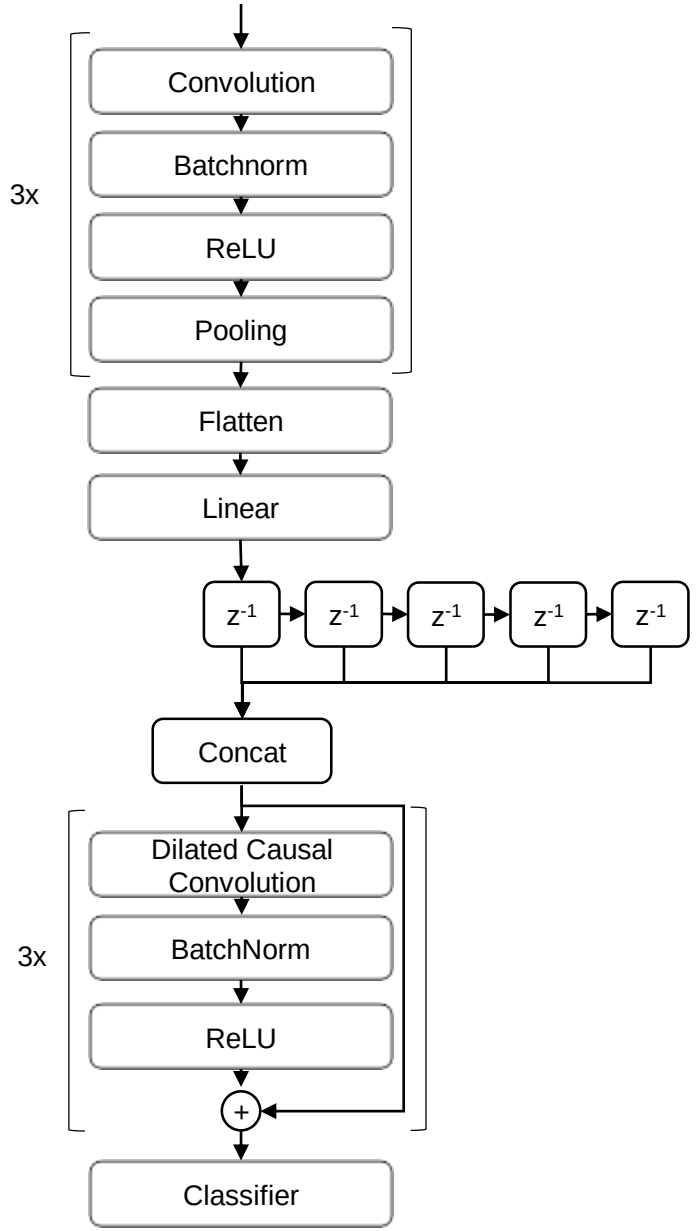
- GAP 8 is a microcontroller based on the open-source PULP-Open[9] system
  - PULP-Open is a project of ETH Zurich and University of Bologna
- GAP 8 features 8 32 Bit RISC-V cores (RI5CY) with specialized DSP-ISA extensions
  - 1 RI5CY Fabric Controller core
  - 512 kB L2 SRAM memory
  - 64 kB L1 low-latency cluster SRAM memory
  - Hardware Loops
- Support for 8 Bit SIMD
  - Sum-Dot-Product instructions
  - Vector Shuffle instructions
  - And many more



[9]:https://github.com/pulp-platform

TinyRadarNN Network

- Network predicts gesture from short range radar data
  - Input data consists of sequential samples from radar
- Network consists of CNN frontend + TCN backend[11]
  - Each 2D sample is fed to a CNN for feature extraction
  - 5 feature vectors at a time are fed into TCN for final prediction
- 8 Bit weights & feature maps
- 73.66% LOOCV Accuracy
- 91 kB Model size



[10]: Available at: https://tinyradar.ethz.ch/
[11]: Scherer et al.: “TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars”

Motivation & Contribution

How can we enable efficient execution of transformers on microcontrollers?

- Novel transformer-based network for TinyRadarNN dataset
- Full 8 Bit integerization pipeline for transformer networks
- Optimize kernels to avoid data marshalling operations in self-attention
- Benchmark kernels and network on GAP8 and ARM Cortex-M devices

Network Design

- Replaced dense convolutions by depthwise-separable convolutions
  - Reduce number of parameters in frontend
- Subsampled input vectors by 2x to reduce computations
- Replaced TCN by Transformer Encoder
  - 6 layers
  - Same embedding dimension

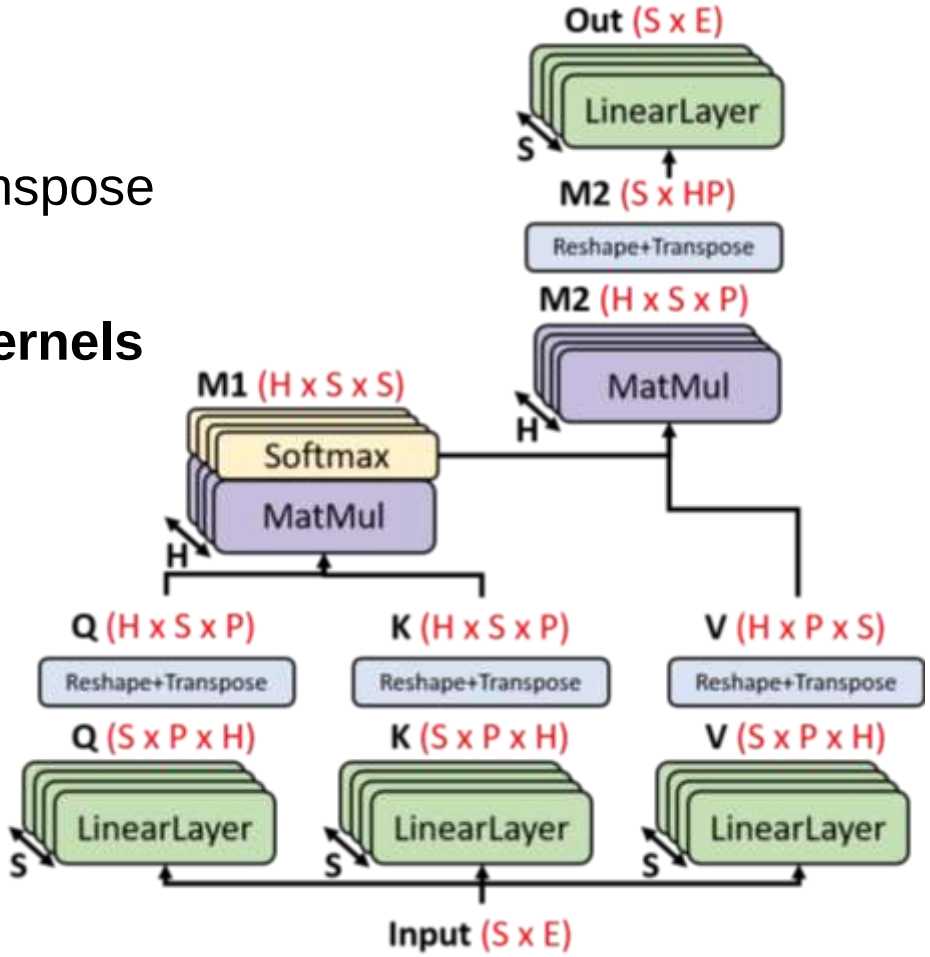
- 8 Bit weights & feature maps
- 77.15% LOOCV Accuracy
  - **3.5% increase in accuracy**
- 263 kB model size

Multi-Head Self-Attention

- Self-Attention uses Matrix Multiplications + Transposition
  - Many different data formates required -> Reshape + Transpose
  - Data marshalling is costly
  - **Merge multiplication + transposition in specialized kernels**
- Self-Attention uses Softmax activation
  - Calculcating exponentials in full precision is expensive
  - **Use polynomial approximation** as in I-BERT[12]

$$exp(x_i) \approx a \times (x_i + b)^2 + c$$

$$\text{Softmax}(x)_i = \frac{exp(x_i)}{\sum_{i=1}^{N} exp(x_i)}$$

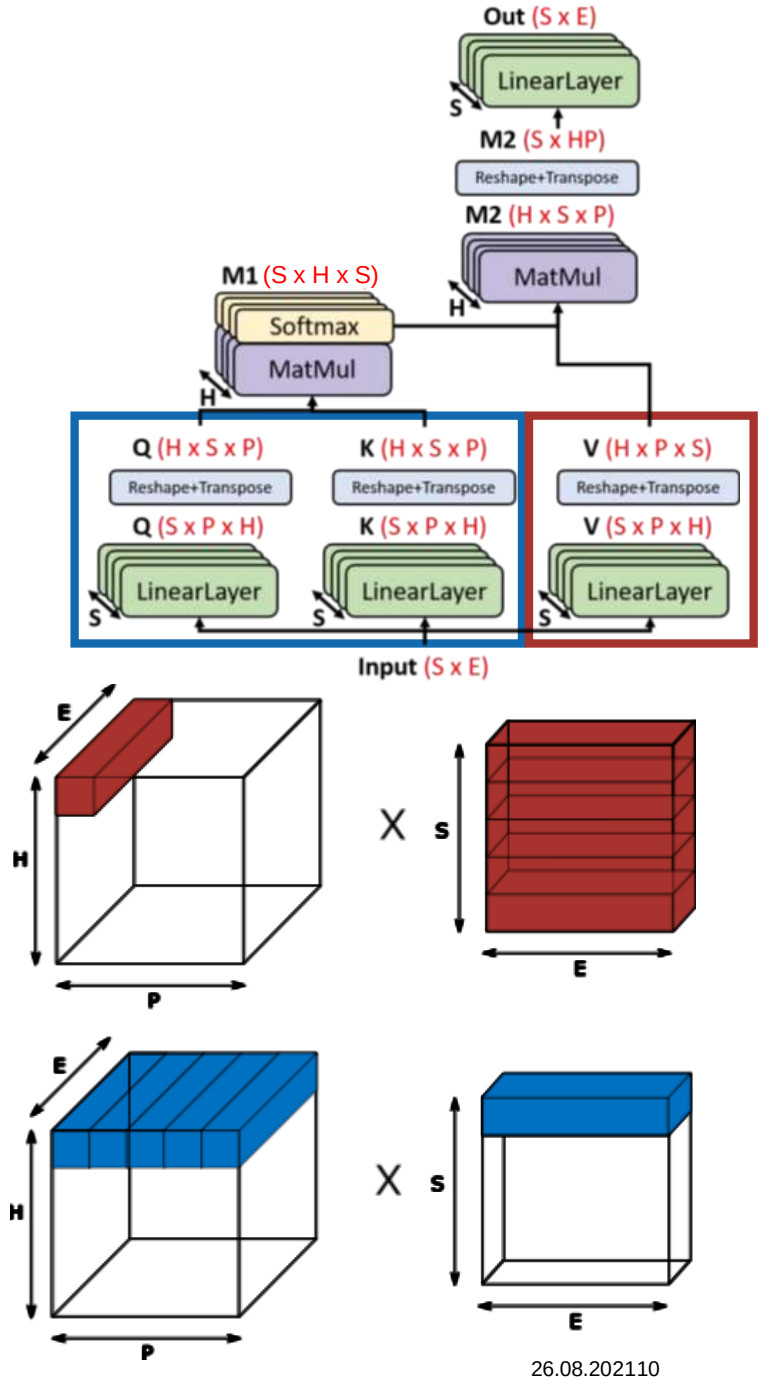


[12]: Kim et al.: "I-BERT: Integer-only BERT Quantization"

Self-Attention – Initial Projection

- *Input-Reuse Linear* Kernel for H x S x P output layout
  - Use ExP weights first, then traverse sequence
  - Outputs in H -> P -> S order
  - Parallelize over Head dimension
- *Weight-Reuse Linear* Kernel for H x P x S output layout
  - Use all inputs first, then traverse weights
  - Outputs in H -> S -> P order
  - Parallelize over Head dimension

- **All accesses are contiguous**

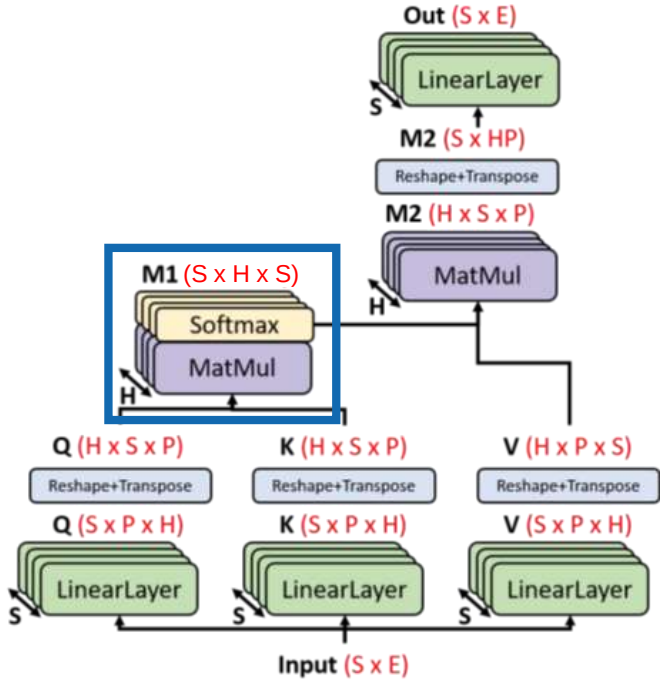


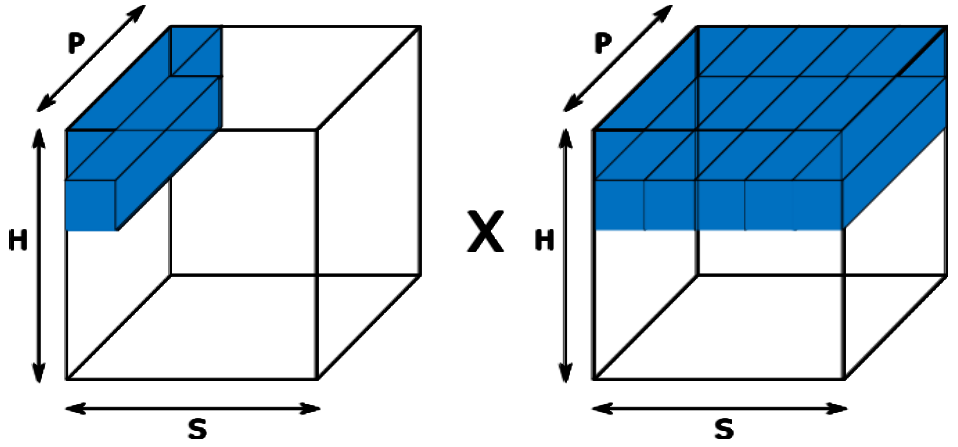
Self-Attention – MatMul + Softmax

- The softmax matmul Q x K^T produces data in S x H x S format
  - Access Q in S -> H -> P order
    - Non-sequential accesses but regular stride
  - Access K in H - > S -> P order
  - Parallelize over Head dimension

- Softmax is calculated over the last dimension
  - Exponential function is approximated with polynomial
  - Exponentiation can be merged into matmul kernel

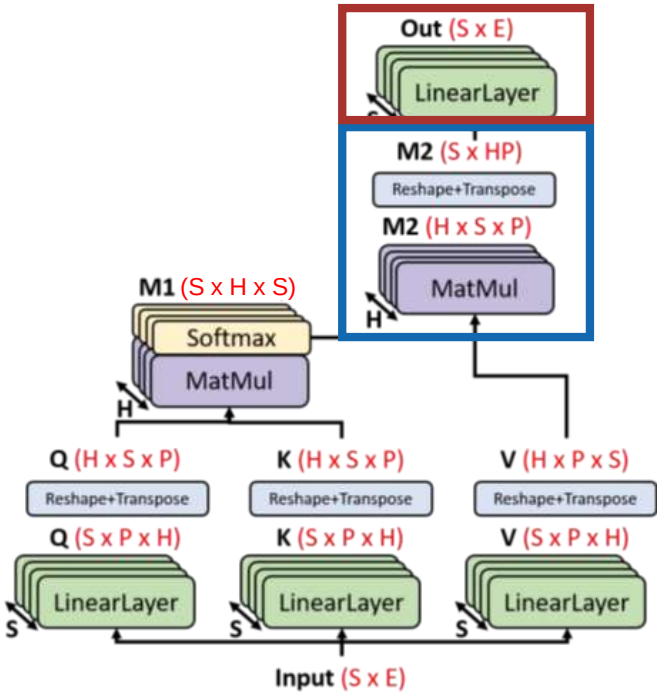
- **Kernel fully avoids explicit transpose / reshape operations!**

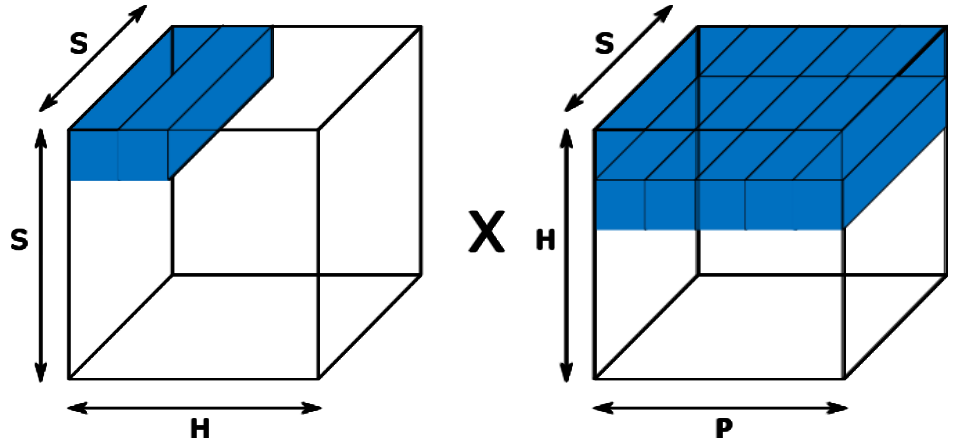




Self-Attention – Attention Matmul & Final Projection

- The Attention Matmul is standard
  - M1 is accessed in sequential S -> H -> S order
  - V is accessed in sequential H -> P -> S order
  - H and P dimensions are collated implicitly
  - Parallelize over Head dimension
- The Final Projection layer is a standard Linear layer
  - Use any well-optimized implementation like CMSIS-NN / PULP-NN
  - Parallelize over output Embedding dimension





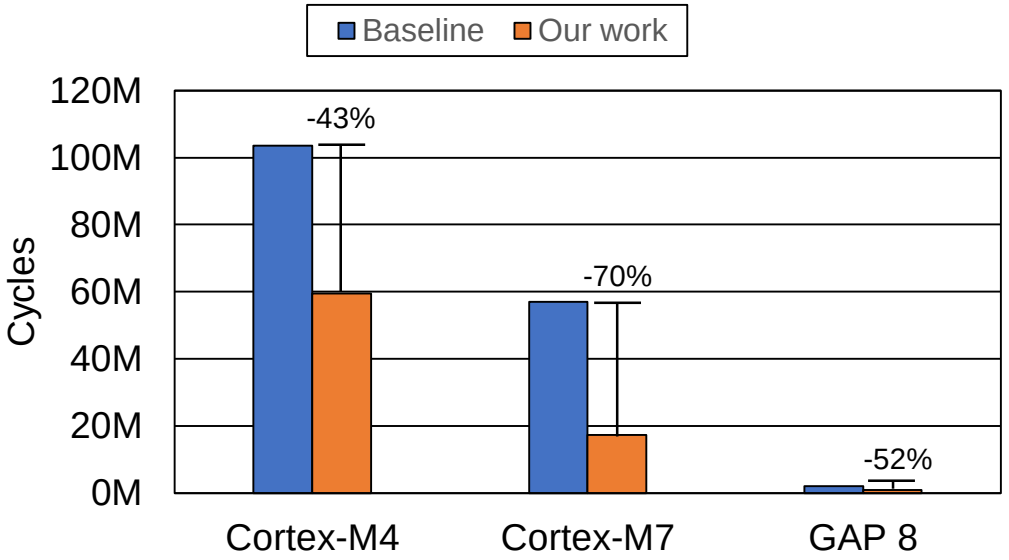
Experimental setup

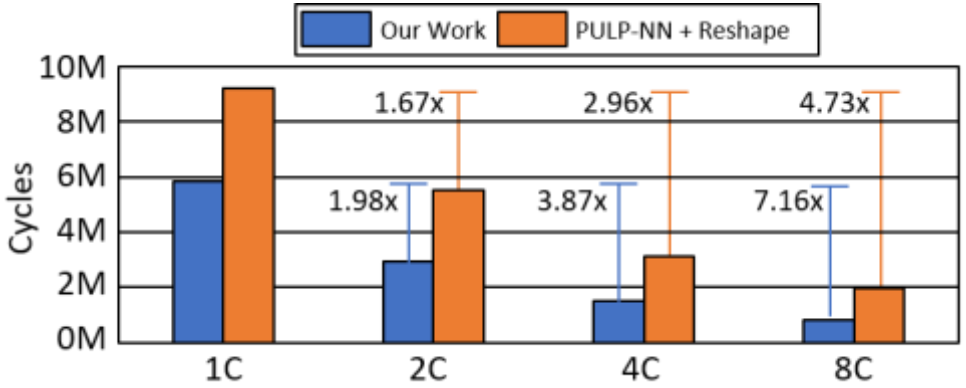
- Implemented 8 Bit attention kernels on three platforms
  - GAP 8 (RV32IMC + XPulpv2)
  - STM32H7 (Cortex-M7)
  - STM32L4 (Cortex-M4)
- Self-attention benchmarks for the following parameters:
  - H = 16
  - P = 64
  - S = 32
  - E = 64
- Benchmarked the full network on all three devices
  - Our solution, PULP-NN[13] and CMSIS-NN[14]

[13]: Garofalo et al.: “PULP-NN: Accelerating Quantized Neural Networks on Parallel Ultra-Low-Power RISC-V Processors”
[14]: Lai et al.: “CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs”

Attention Results

- Large single-core speedup across all platforms
  - Cortex-M4: 1.75x
  - Cortex-M7: 3.35x
  - GAP 8: 2.1x
- Close-to-optimal speedup on multiple cores
  - 7.16x for 8 cores
  - Parallelization over head dimension
- **Comparable efficiency to convolutions**
  - Our attention layer: 11.29 MAC/cycle
  - PULP-NN Convolutions: 12.86 MAC/cycle
- **No data marshalling in Attention layers!**





Network Results

- All network performance numbers are improved by large margins
  - Accuracy, Operations, Latency & Energy
- Only increase in model size
  - Still fits in 512 kB L2 memory

	Original		Our Network
Accuracy	73.66%	↑ 3.5%	77.15%
Model size	93 kB	↑ 3x	263 kB
Operations	42 MMACs	↓ 14x	3 MMACs
Latency	58 ms	↓ 6.3x	9.24 ms
Energy	4.52 mJ	↓ 9.6x	0.47 mJ

Conclusion

- Our kernel library allows to extract performance similar to convolutional layers from Attention
  - **No explicit reshaping or transposition operations**
- Parallelizing over the head dimension allows for optimal scaling
  - 1.5x better than conventionally optimized library

- Network backend size is increased, frontend size is decreased
  - Transformer significantly increases accuracy
- While transformer layers pack a lot of parameters, computations are reduced
  - 9.6x improvement in inference energy over original implementation, **one order of magnitude**
  - **All performance metrics are fully dominated**

Moritz Scherer
PhD Student, ETH Zürich
scheremo@iis.ee.ethz.ch

ETH Zürich
Integrated Systems Lab
ETZ J 69.2
Gloriastrasse 35
8092 Zürich