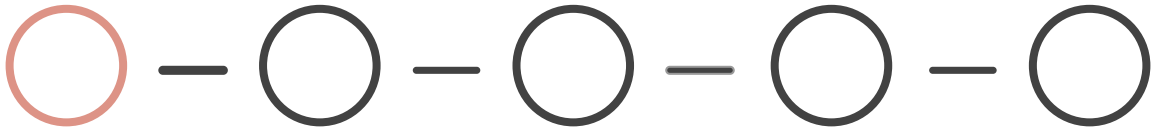
Co-Design of Convolutional Algorithms and Long Vector RISC-V Processors for Efficient CNN Model Serving

Sonia Rani Gupta[1], Nikela Papadopoulou[2], Jing Chen[1], Miquel Pericàs[1]
[1] *Chalmers University of Technology*
[2] *University of Glasgow*


Motivation
Background
Methodology
Evaluation
Conclusions

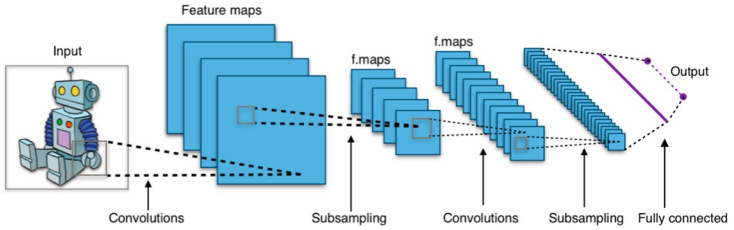
Outline

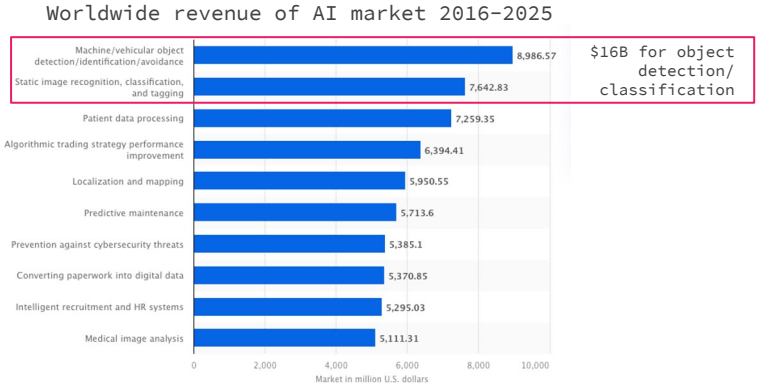
Motivation: Convolutional Neural Networks (CNNs)

CNNs remain highly relevant in computer vision

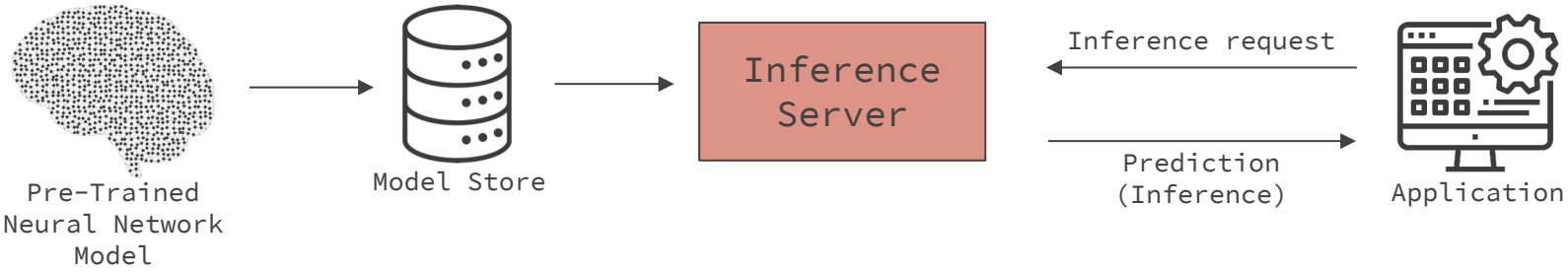
(despite the transformers craze!)

- Great at object detection and image classification
- Ideal for small tasks
- ResNet and other 2010s classics:
  high accuracy with low FLOPs
  - Perfect for edge computing
- ConvNext (the 2020s CNN):
  higher accuracy than ViT (transformers)
- Image recognition projected to have a 8.7% annual growth in the next years





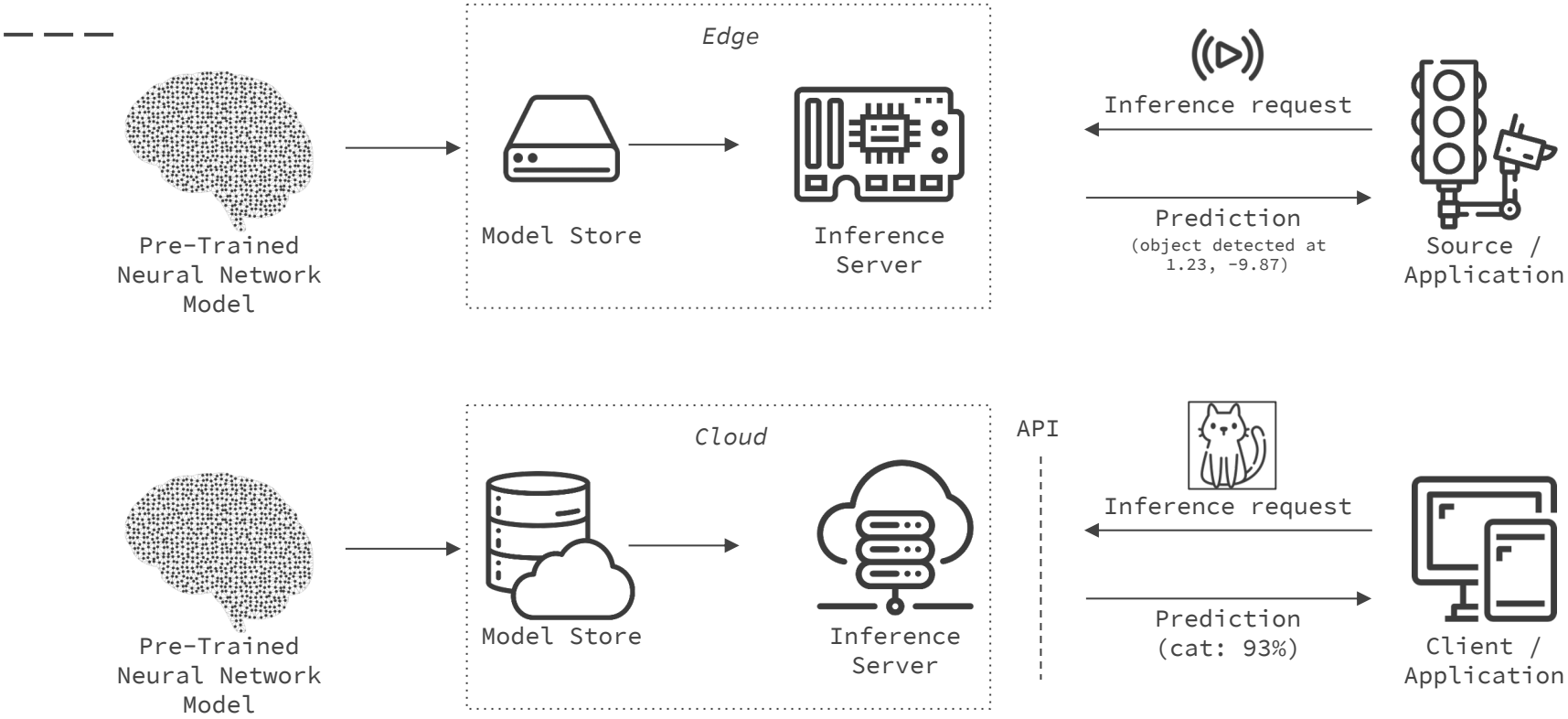
Motivation: Model Serving



Model serving is becoming the norm

- Inference is ~90% of a model's lifecycle
- Inference is ~90% of all ML infrastructure (and equally energy-consuming)

Motivation: Model Serving



Motivation: RISC-V as a model serving platform

Inference is traditionally dominated by GPUs

Vector-enabled processors provide an alternative path to training and inference

- POWER10 inline Matrix-Multiply Assist (MMA)
- A64FX Scalable Vector Extension
  - used to train LLM (Fugaku)
- RISC-V vector extension 1.0, RISC-V (integrated/attached) matrix extensions work in progress

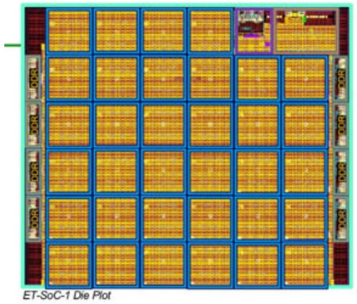
Why RISC-V?

- Open ISA enables customization and innovation
- No licensing fees + leverage open ecosystem
- Higher integration: host and accelerator tightly integrated in single core

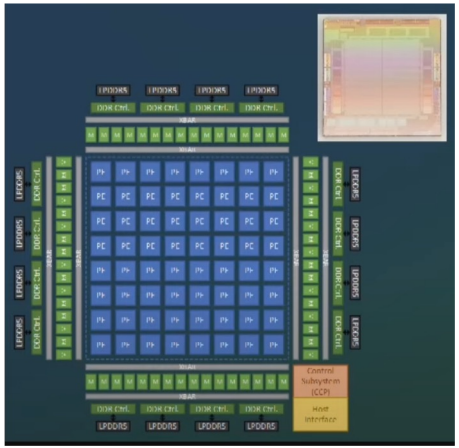
Many RISC-V startups focusing on AI:
Esperanto, Tenstorrent, Ventana Micro, Semidynamics

- Need for **codesign** to identify efficient design points enabling high performance, energy efficiency, high utilization

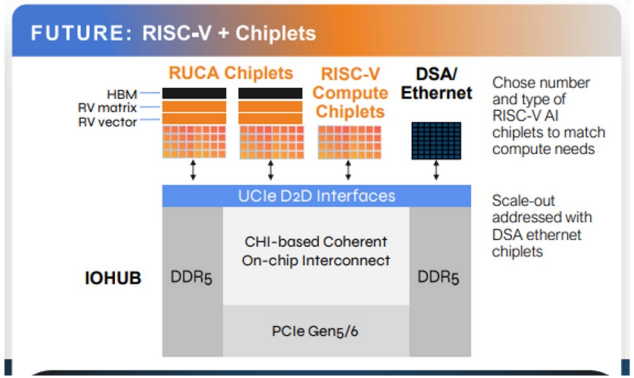
Motivation: Co-Design with RISC-V



ET-SoC-1 w/ 1088 ET-Minion Vec cores + 4 ET-Maxion OoO cores

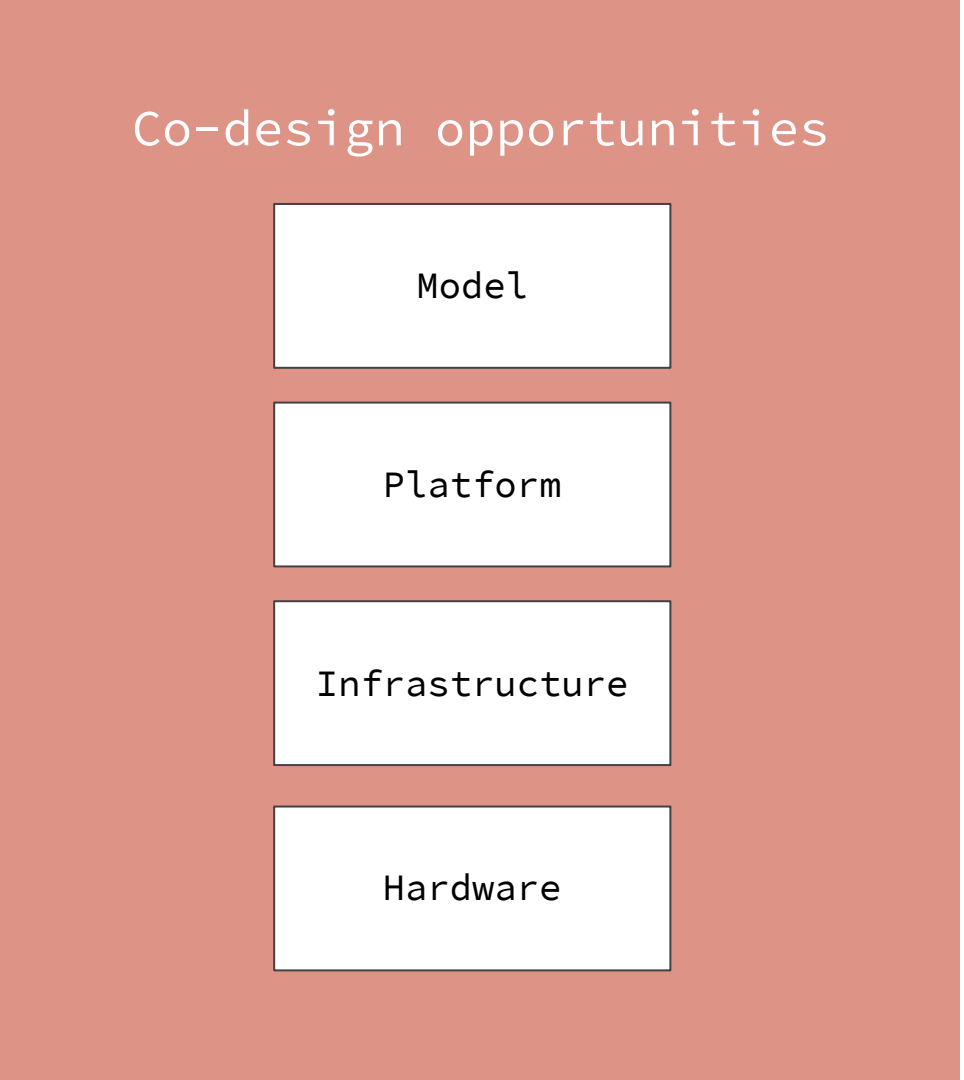


MTIA: 128 Vec core + 1 control core



Ventana's RUCA: RISC-V Unified Compute Architecture

Many AI RISC-V designs have been proposed, but which is the optimal design point?



Motivation: CNNs and co-design

Characteristics of CNNs:

- Multiple time-consuming convolutional layers
- No nice square matrices
- Widely varying dimensions between layers
- Many algorithmic implementations of convolutions

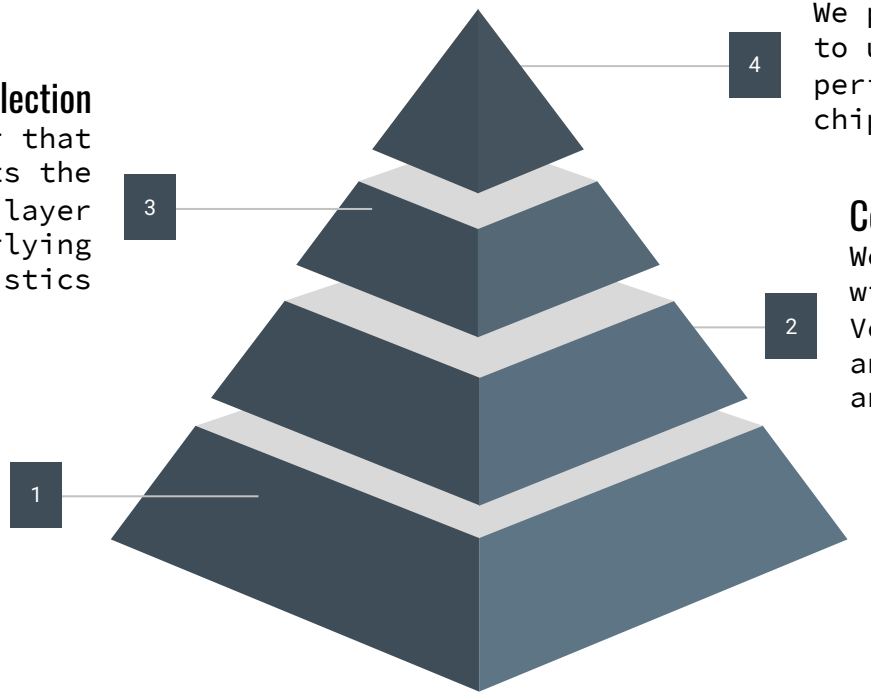
Optimizing the (FL)OPs is not enough!

- Cache and memory bandwidth are contenders in model sharing
- Performance - Energy - Cost tradeoffs

This work: Contributions

Convolutional Algorithms on RVV
We vectorize the Direct algorithm and use optimized versions of im2col+GEMM and Winograd for RISC-V Vector

Convolutional Algorithm Selection
We build a classifier that automatically selects the best algorithm per layer depending on the underlying architectural characteristics



Co-design Analysis on RVV
We study the 3 algorithms with 2 CNNs on a RISC-V Vector based architecture and vary the vector length and L2 cache size

Performance/Throughput/Area
We perform a Pareto analysis to understand tradeoffs in performance, throughput and chip area

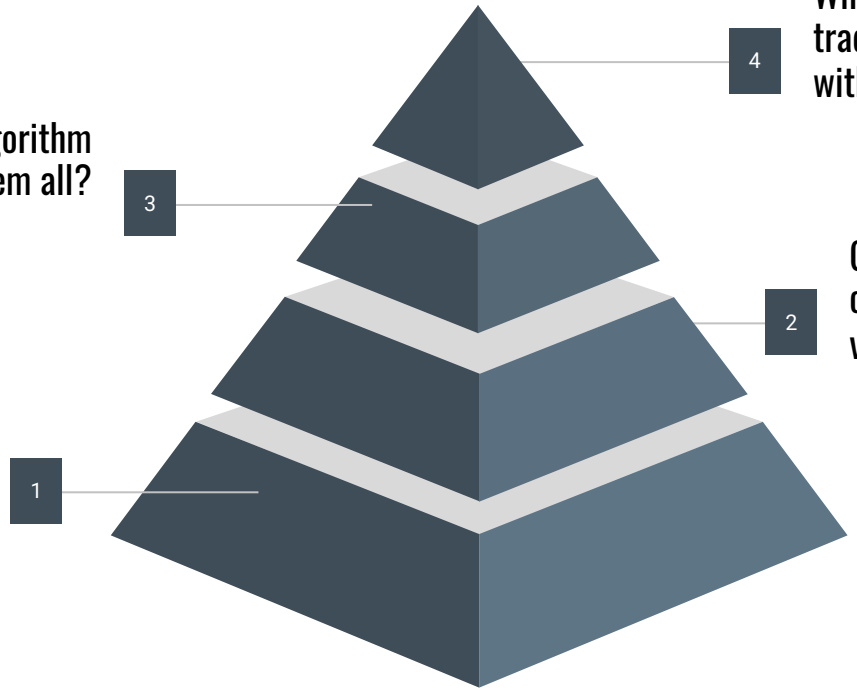
This work: Research Questions

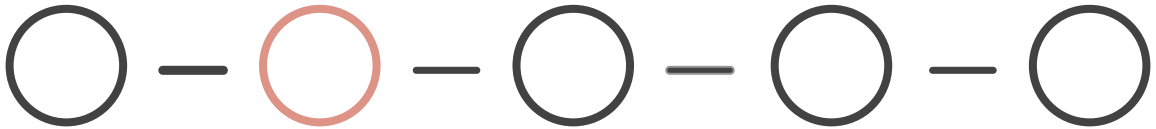
What are the throughput - area tradeoffs to efficiently serve CNNs with vector architectures?

Is there one convolutional algorithm to rule them all?

Can the performance of convolutions benefit from longer vectors and larger caches?

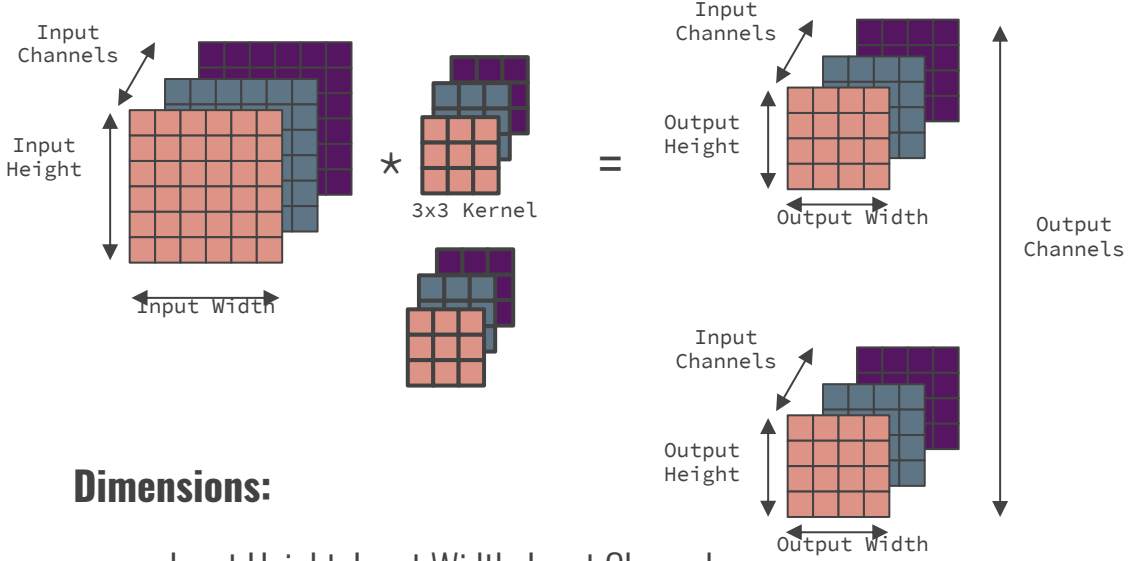
What is the performance of different convolutional algorithms for different layers on vector architectures?




Motivation
Background
Methodology
Evaluation
Conclusions

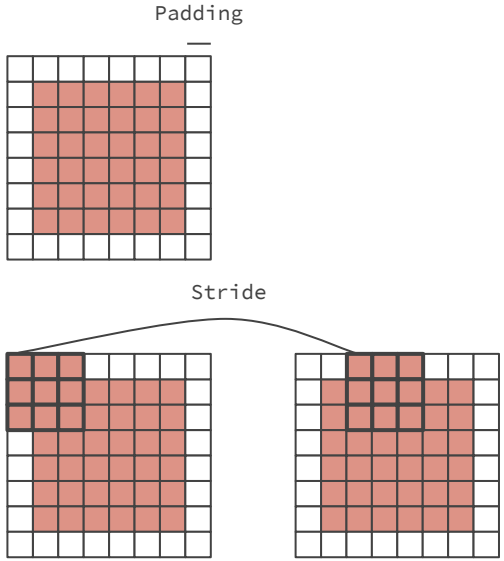
Outline

Background: Convolutional layers



Dimensions:

- Input Height, Input Width, Input Channels
- Kernel Width, Kernel Height
- Output Width, Output Height
- Output Channels



Additional Parameters:

- Padding
- Stride

Convolutional Algorithms

1. `Direct:` The kernel slides over the image, computing their dot product
    + No lowering required
    + No transformations required
1. `im2col + GEMM:` The input images are concatenated in a 2D-matrix (`im2col`), then multiplied with the kernel (`GEMM`)
    + Same complexity as the Direct
    + GEMM is optimized
    - The im2col transformation increases the memory footprint
3. `Winograd:` The input and weights are transformed into blocks, followed by block-to-block multiplications and output transformation
    + Lower complexity than GEMM
    - Suitable for 3x3 and 5x5 kernels due to numerical stability
3. `FFT:` The input and weights are transformed into with DFT, followed by multiplications and an inverse Fourier transformation of the output
    + Lower complexity than GEMM
    - Suitable for high-dimensional kernels
    - Requires a lot of zero-padding

Background: The RISC-V Vector extension (RVV)

- An instruction set extension for the RISC-V ISA
  - Spec v1.0 (draft v1.1)
- Vector-length agnostic extension
  - Operations are performed on vector registers
    - 32 vector registers of VLEN-bits
- The vector length `VLEN` is not fixed, but is implementation-specific
  - Hardware parameter
- For the programmer, the vector length `vl` used in an instruction is a software/runtime parameter
  - Depends on the `VLEN`, the datatype, and the number of grouped registers

Background: Convolutional Algorithms on Vector Processors

	Intel AVX	ARM NEON	ARM-SVE	RISC-V Vector	Other
Direct	Kelefouras et al, 2022 Santana et al, 2023	Wang et al, 2023	-	-	**Santana et al, 2023**
im2col + GEMM	Alaejos et al, 2023	Alaejos et al 2023	Dolz et al, 2022 Gupta et al, 2023a Martinez et al, 2024	**Gupta et al, 2023a** Martinez et al, 2024	-
Winograd	Dolz et al, 2023	Dolz et al, 2023	Dolz et al, 2023 Gupta et al, 2023a	**Gupta et al, 2023b**	Wang et al, 2021

Many recent works focusing on the emerging vector length agnostic ISAs (ARM SVE and RISC-V Vector)


Motivation
Background
Methodology
Evaluation
Conclusions

Outline

Methodology: Experimental Platform

Constants:

- Fork of `gem5` [1] supporting RVV v1.0
  - *Disclaimer #1: gem5 v23.1 supports RVV v1.0*
  - *Disclaimer #2: constant vector instruction latency*
- `RISCVMinorCPU` CPU model at 2GHz
- DDR3 1600 memory technology at 12.8GB/s per core
  - Close to per-core bandwidth of recent HBMs (~20GB/s)
- L1 cache fixed at 64KB

Variables:

- L2 cache: 1 up to 64MB
- Vector Length (HW parameter): 512 up to 4096 bits

[1] https://github.com/plctlab/plct-gem5

Methodology: Algorithms for Convolutions

1. `Direct`: New in this work, starting from existing work for Intel AVX and NEC SX-Aurora [1]
2. `im2col + GEMM`: Use two vectorized/optimized variants from our previous work on RVV [2]
   - *`im2col + GEMM – 3 loops`*: an implementation of the "naive" GEMM
   - *`im2col + GEMM – 6 loops`*: an implementation of the blocked GEMM
3. `Winograd`: Use a vectorized and optimized version from our previous work on RVV [3]

- All implemented in the Darknet [4] framework.
- We use EPI-builtins (intrinsics) to vectorize for RVV v1.0

Implementing the Direct algorithm for RVV

Naive version → Optimized version

```
for n = 0, N do
  for oc = 0,OC do
    for ic = 0, IC do                                   Vectorize
      for oh = 0,OH do
        for ow = 0,OW do                                Unroll
          for kh = 0, KH do
            ih = oh * Cstr + kh – Cpad
            for kw = 0, KW do
              iw = ow * Cstr + kw – Cpad
```

Reorder

```
O[n,oc,oh,ow] += I[n,ic,ih,iw] * K [oc,ic,kh,kw]
O[n,oc,oh,ow] += I[n,ih,iw,ic] * K [oc,ih,kw,kc]
```

Change format from NCHW to NHWC

3x speedup

[1] Alexandre de Limas Santana, Adrià Armejach, and Marc Casas. 2023. Efficient Direct Convolution Using Long SIMD Instructions. In Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '23). Association for Computing Machinery, New York, NY, USA, 342–353.

[2] Sonia Rani Gupta, Nikela Papadopoulou, and Miquel Pericàs. 2023. Accelerating CNN inference on long vector architectures via co-design. In 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 145–155

[3] Sonia Rani Gupta, Nikela Papadopoulou, and Miquel Pericàs. 2023. Challenges and Opportunities in the Co-Design of Convolutions and RISC-V Vector Processors. In Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23). Association for Computing Machinery, New York, NY, USA, 1550–1556

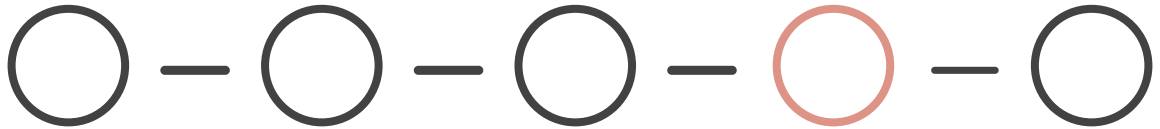
[4] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/

Methodology: Experimental Setup

Network Models

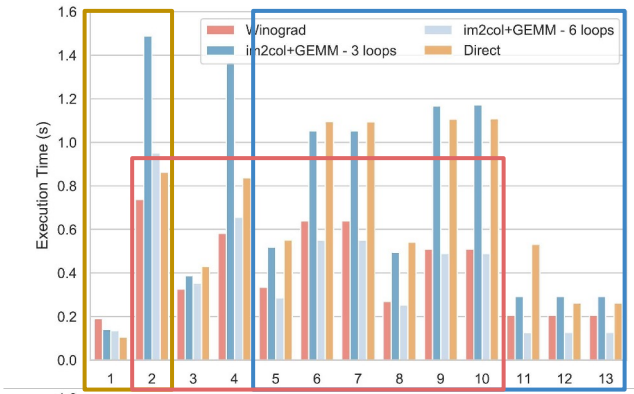
- YOLOv3 for object detection
  - 107 layers in total
  - 75 convolutional layers
    - ~96% of the execution time
  - *We use the first 20 layers, out of which 15 are convolutional*
- VGG16 for image classification
  - 25 layers in total
  - 13 convolutional layers
    - ~64% of the execution time
  - 3 fully connected layers (use our optimized GEMM)

Please refer to the paper for layer dimensions

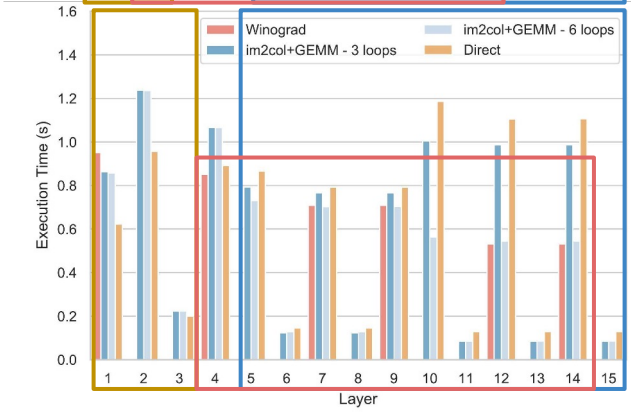

Motivation
Background
Methodology
Evaluation
Conclusions

Outline

Evaluation: Performance comparison of convolution implementations



VGG-16



YOLOv3

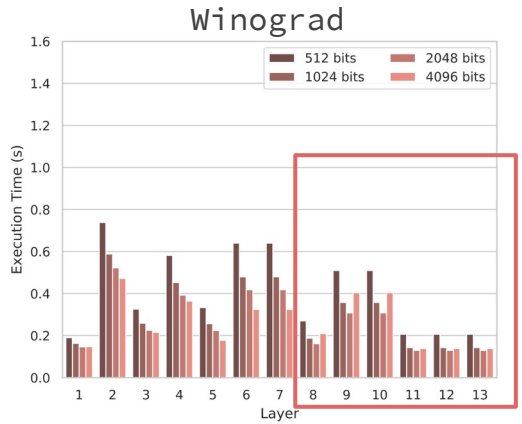
Baseline architecture: 1MB L2 cache, 512 bits vector length

Lower is better!

1. `Direct:` Better when input/output dimensions are high but channels are low
2. `im2col+GEMM:` Better with skinny matrices and high input/output channels
   - the 6-loop implementation prevails
3. `Winograd:` Better or comparable performance for many layers, worse when the input channels are high
   - only applicable with 3x3 kernels
   - high transformation overheads

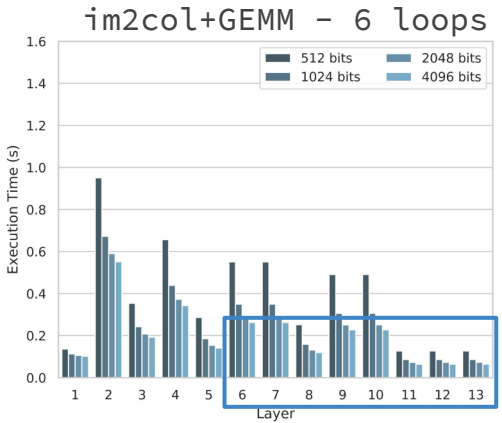
Evaluation: Co-design - scaling the vector length

Fixed L2 Cache size: 1MB | **Varied vector length:** 512-4096 bits | **VGG-16**



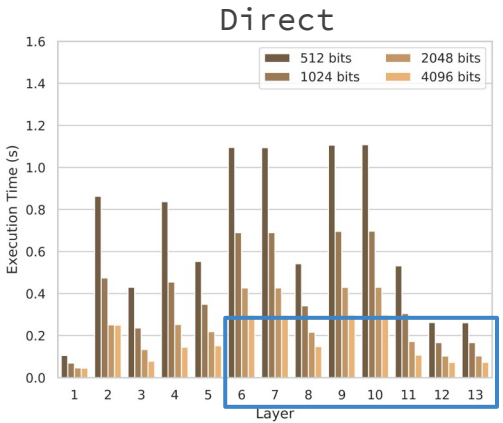
1.3-1.7x speedup for 2048 bits (baseline 512 bits)

Limited scalability from 2048 to 4096 bits for skinny matrices
- Increased loop iterations



1.4-2.1x speedup for 4096 bits (baseline 512 bits)

The 6-loop variant decreases the pressure to the L2 cache compared to the 3-loop variant

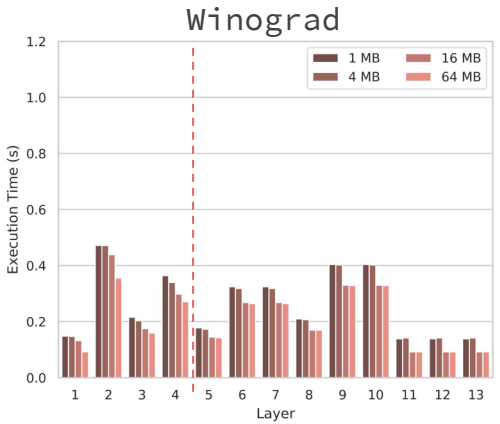


2.4-5.8x speedup for 4096 bits (baseline 512 bits)

im2col+GEMM can still offer better performance for specific layers with 2048 bits

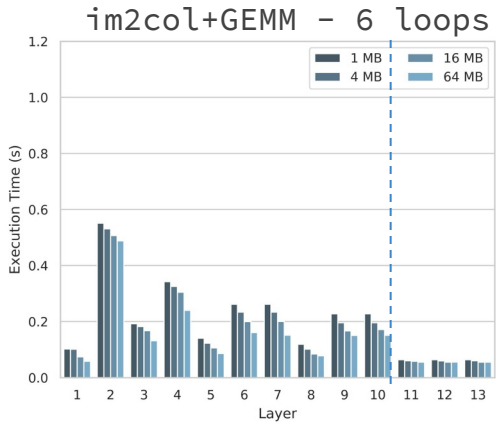
Evaluation: Co-design - scaling the L2 cache (I)

Fixed vector length: 512 bits | **Varied L2 cache:** 1-64MB | **VGG-16**



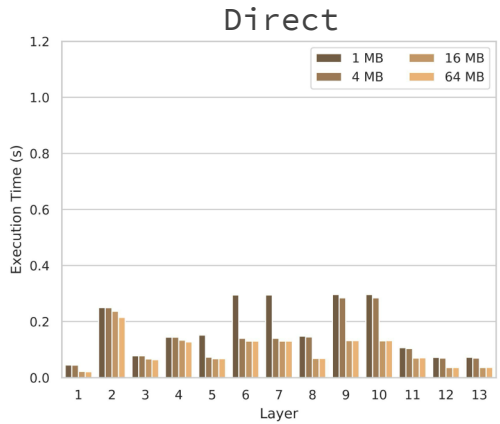
1.3-1.5x speedup for layers #1-#4 (baseline 1MB)

Limited scalability from increasing the L2 cache for layers with high numbers of input/output channels



1.1-1.76x speedup for layers #1-#10 (baseline 1MB)

No further benefit from increasing the cache above 16MB for extremely skinny matrices



1.1-1.4x speedup for up to 16MB (baseline 1MB)

Better scalability as the L2 cache increases, resulting in better execution times

Evaluation: Co-design - scaling the L2 cache (II)

Some additional takeaways:

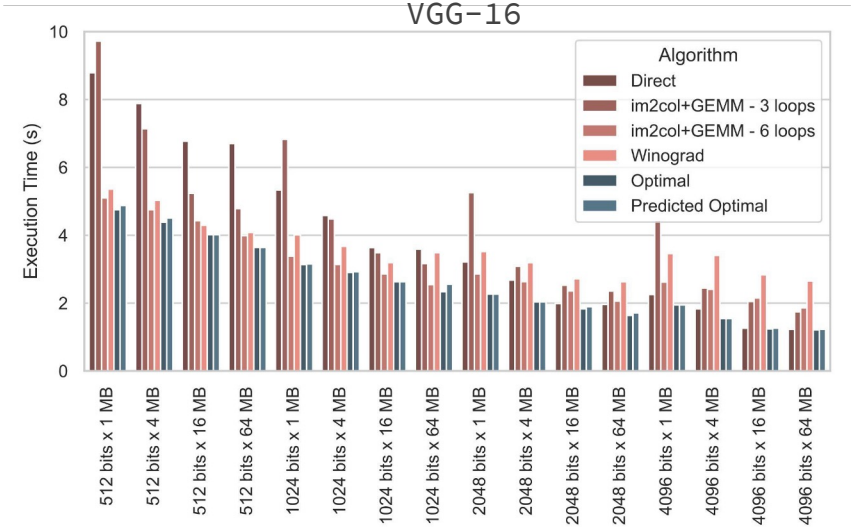
- `Winograd` uses fixed tile sizes and is oblivious to larger caches
- The 3-loop variant of `im2col+GEMM` is sensitive to the cache and shows high scalability
- `im2col+GEMM` (both variants) exhibit on average **2x** better performance with a larger cache
- The `Direct` can experience a large performance boost (up to **2.8x**) for high input/output dimensions, in configurations with long vectors (2048 bits or 4096 bits)

So which one is the right algorithm to use?

Evaluation: Algorithm selection

Random Forest Classifier for **Algorithm Selection per Layer**

- Non-linear but small classifier - can be used at runtime
- `Input:` Convolution dimensions/parameters, Hardware configuration
- `Output:` Algorithm with lowest execution time for the specific layer
- 5-fold cross-validation (80% training, 20% testing)
- 92.8% accuracy



Selecting the optimal algorithm per layer improves the performance compared to using a single algorithm:

- `YOLOv3:` **1.33x** compared to Direct only, **2.11x** compared to im2col+GEMM only
- `VGG-16:` **1.85x** compared to Direct only, **1.73x** compared to im2col+GEMM only

Evaluation: Performance-Area Tradeoffs for Single Model

We approximate the area of a RVV chip at 7nm (single core)

- Combination of back-of-the-envelope calculations and PCACTI

Examine Performance / Area tradeoffs

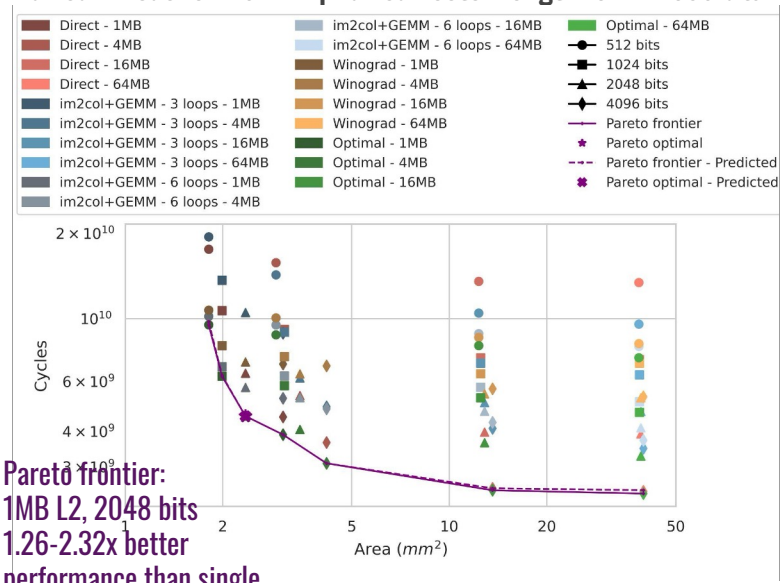
- Single core: Performance = Cycles

Optimal algorithm always better than single algorithm

All Pareto points with 1MB cache

VGG-16 | Single core/model instance

Varied L2 cache: 1-64MB **| Varied vector length:** 512 - 4096 bits



Pareto frontier:
1MB L2, 2048 bits
1.26-2.32x better performance than single algorithm in same area

Evaluation: Throughput-Area Tradeoffs for Model Serving

We approximate the area of a multicore RVV chip at 7nm

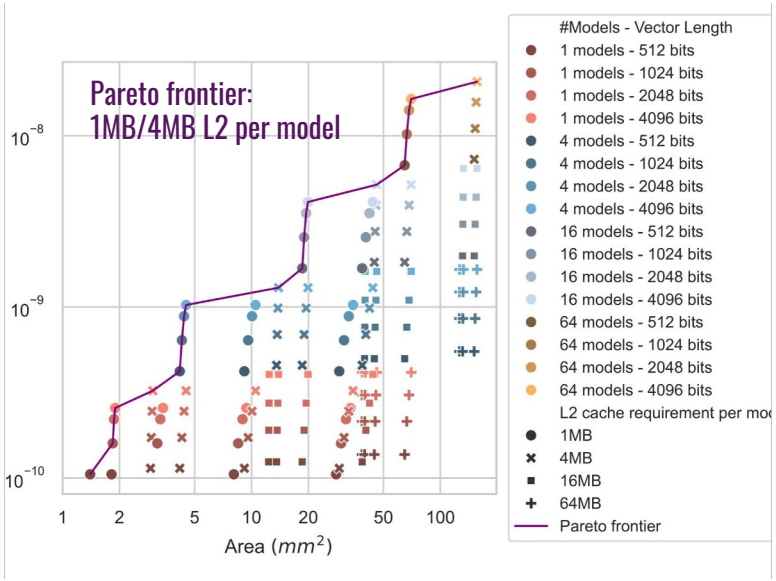
- 1 up to 64 cores
- 1 - 256MB of L2 to cache
- 512 - 4096 bits of vector length

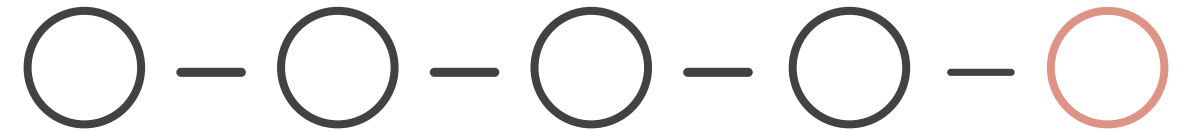
We co-locate 1 up to 64 model instances

- No core/cache oversubscription
- Assumption #1: cache perfectly partitioned
- Assumption #2: sufficient memory bandwidth
- Assumption #3: optimal algorithm per layer

Throughput increases superlinearly with area!

VGG-16 | Multiple Model Instances (1 - 64)




Motivation
Background
Methodology
Evaluation
Conclusions

Outline

Conclusions

- **RQ1: What is the performance of different convolutional algorithms for different layers on vector architectures?**
  - Winograd is a good choice for layers with 3×3 kernel size
  - im2col+GEMM - 6 loop is the best choice for skinny matrices with large numbers of input/output channels
  - Direct performs best when the input output dimensions are high, but the number of channels is relatively small
- **RQ2: Can the performance of convolutions benefit from longer vectors and larger caches?**
  - All algorithms benefit from longer vector lengths (2048 bits) about 2x compared to 512 bits
  - im2col+GEMM and Direct benefit about 1.5x from a larger L2 cache (64MB compared to 1MB), but the Direct can have higher speedups if the vector length is large (4096 bits)
- **RQ3: Is there one convolutional algorithm to rule them all?**
  - We should select the optimal algorithm per layer, it depends on the layer dimensions and the hardware
  - A simple non-linear classifier can help with the selection
  - Algorithm selection can boost performance more than 2x compared to using a single algorithm
- **RQ4: What are the throughput - area tradeoffs to efficiently serve CNNs with vector architectures?**
  - Multiple cores with smaller L2 cache per model and long vector lengths can offer superlinear throughput / area increase

Thank you

soniar@chalmers.se
nikela.papadopoulou@glasgow.ac.uk
chjing@chalmers.se
miquelp@chalmers.se

Acknowledgements

- Funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No.956702 (**eProcessor**), under Framework Partnership Agreement No.800928 and Specific Grant Agreement No. 101036168 (E**PI SGA2**),and under grant agreement No. 101034126 (**The European PILOT**)
- Funding from the project **PRIDE** from the Swedish Foundation for Strategic Research with reference number CHI19-0048, and the project **P4PIM** from the Swedish Research Council (VR) with project ID 2020-04892
- Resources provided by the National Academic Infrastructure for Supercomputing in Sweden (**NAISS**), partially funded by the Swedish Research Council through grant agreement no. 2022-06725