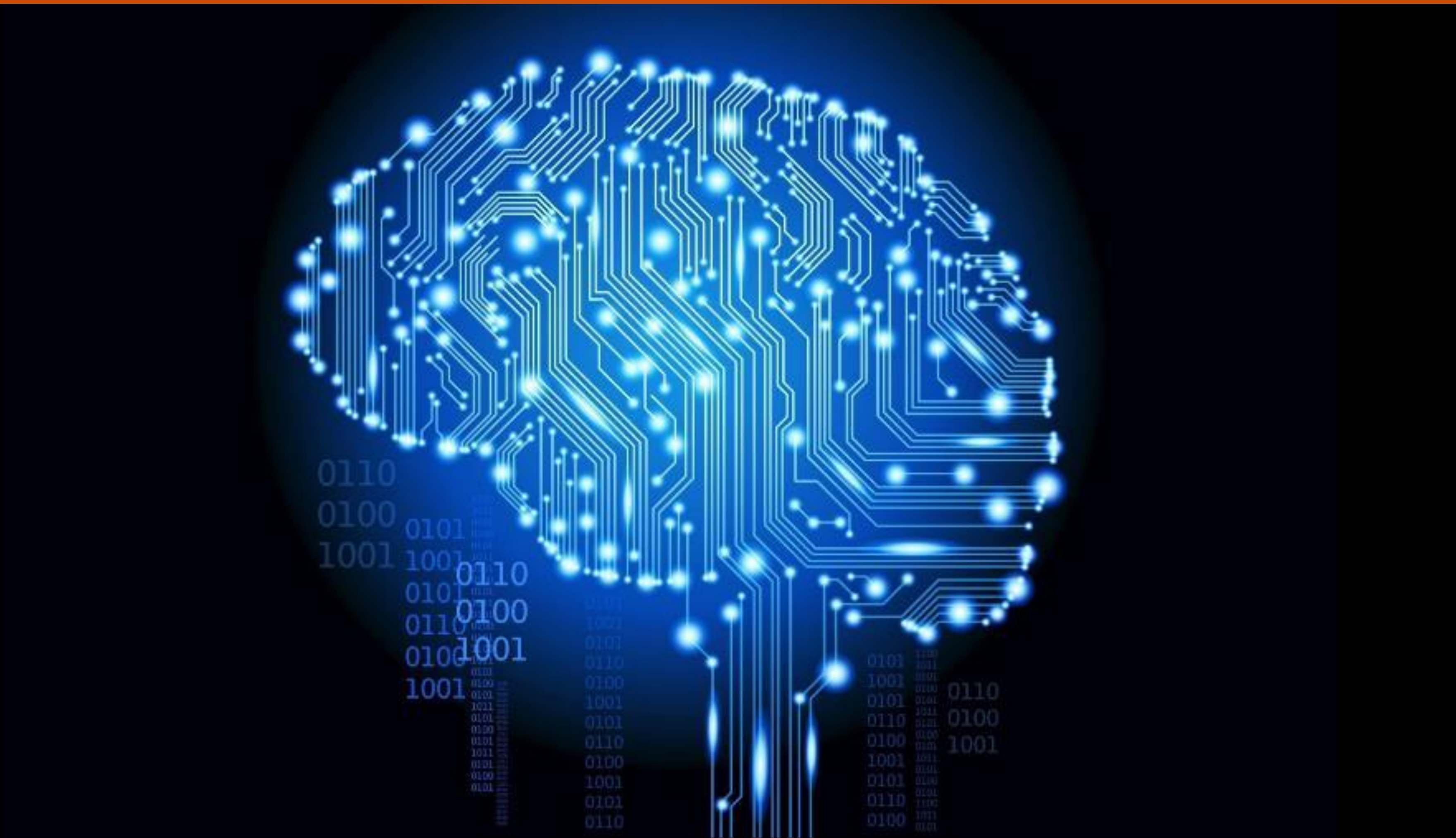
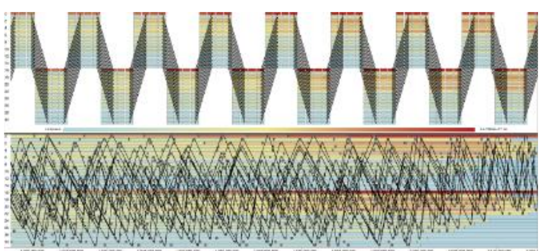
AI, Performance and Modeling

Alexandru Calotoiu



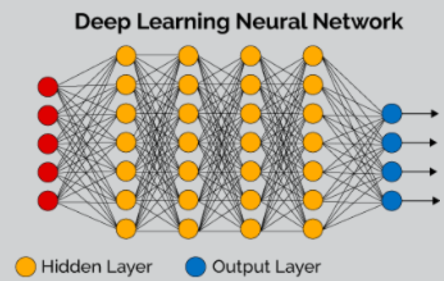
Performance

ORNL - Summit



LLNL - Ravel

Artificial Intelligence

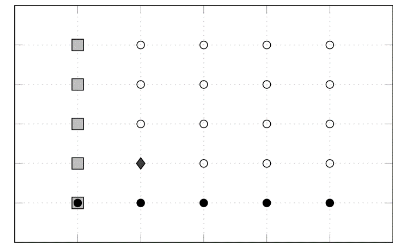


Outline

Improving the performance of AI

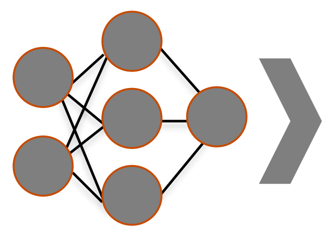
A. Mazaheri, J. Schulte, M. Moskewitz, F. Wolf, A. Jannesari

Using AI to model performance



M. Ritter, A. Calotoiu, T. Hoefler, T. Reimann, S. Rinke, F. Wolf

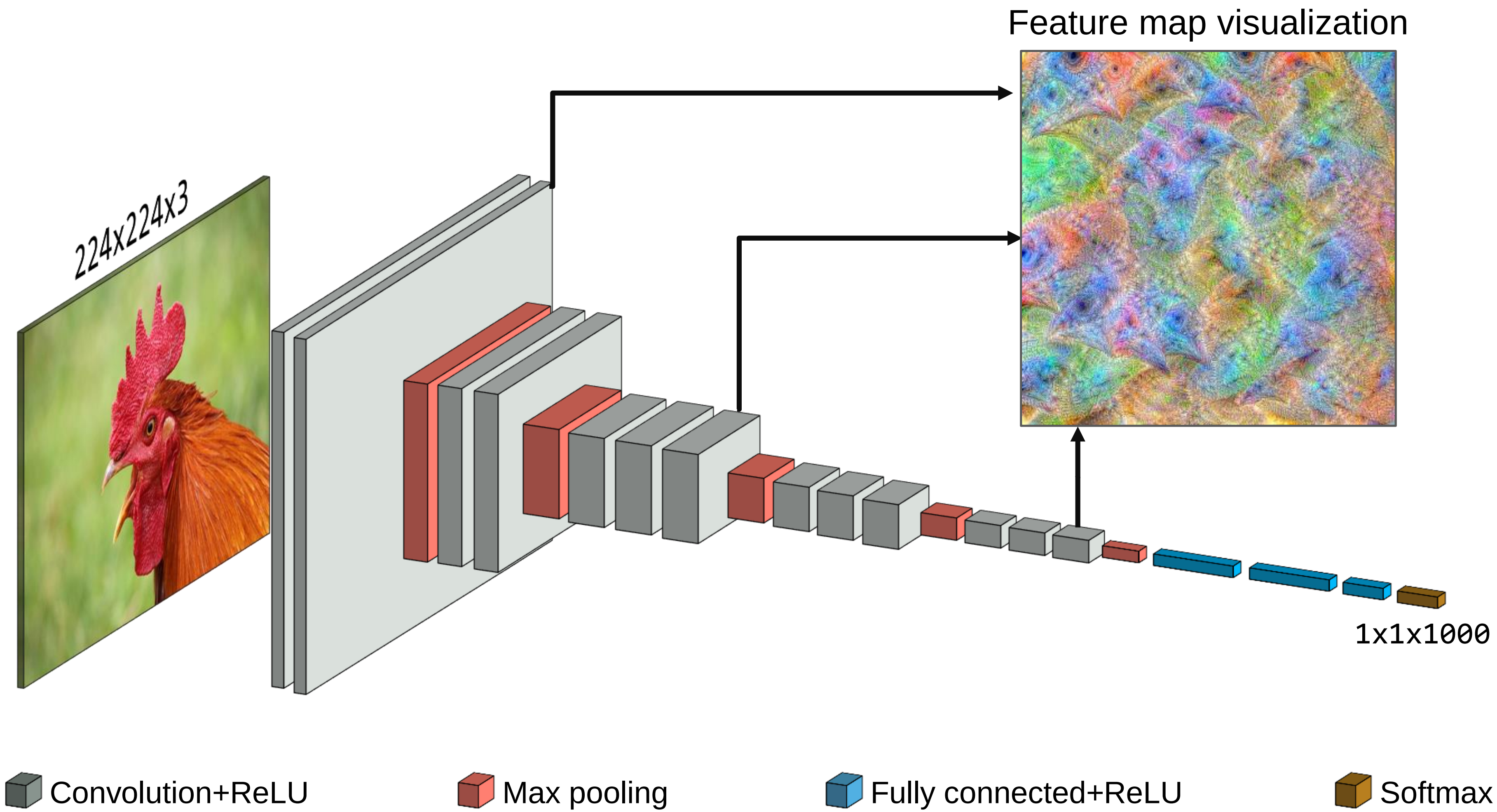
Using AI to improve performance



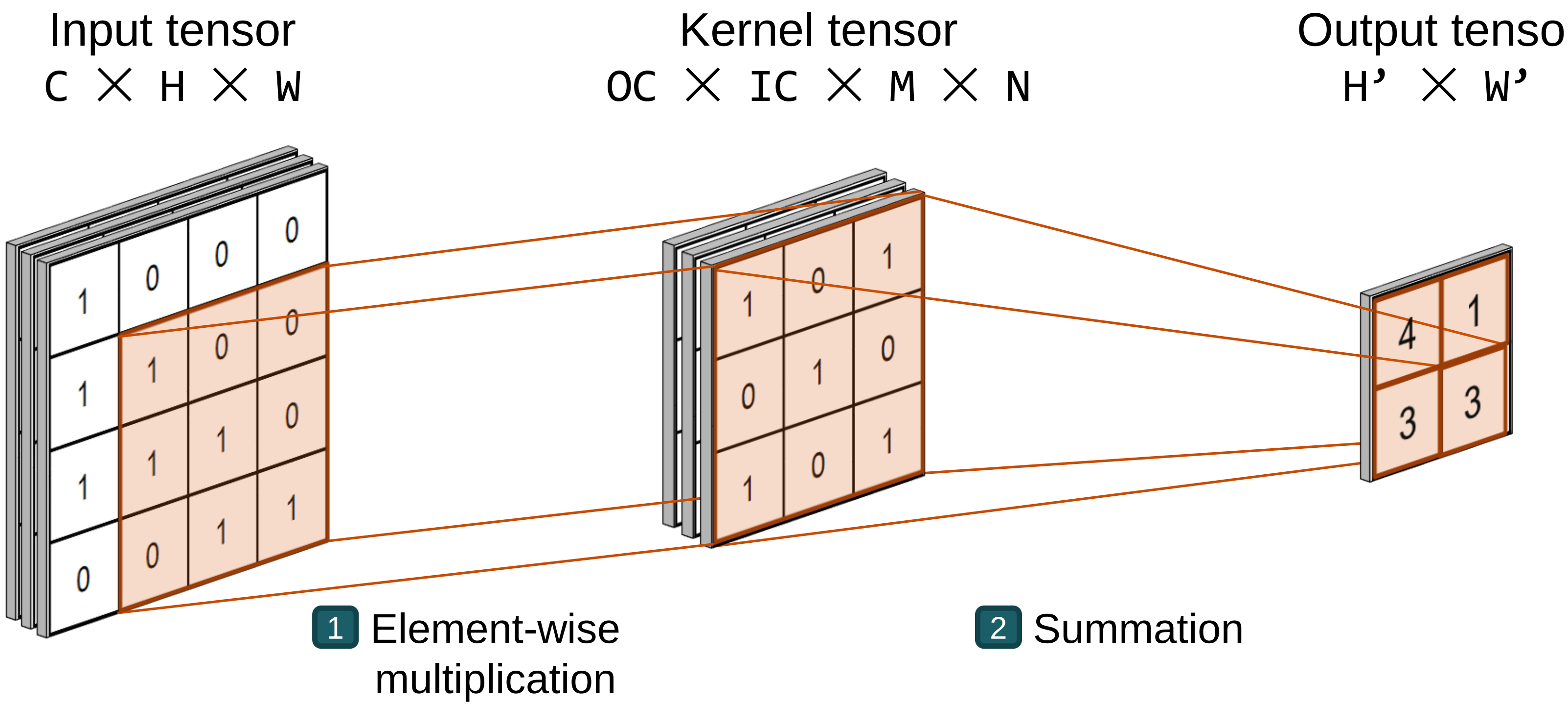
```
for(i = 0; i <= 255; i += 2)
{
    do_something(i);
    do_something(i + 1);
}
```

R. Mammadli, M. Pradel, M Selakovic, F. Wolf

Enhancing the Programmability and Performance Portability of GPU Tensor Operations



Convolution & tensors



- Dominate computation (>90% of runtime)
- Similar to generalized matrix-matrix multiply → Massive GPU parallelism
- Difficult to implement efficiently on GPUs

Challenges of implementing convolutions on GPUs

- Fundamental issues (when targeting a particular GPU):

Data movement

Scheduling, resource-management, parallelism

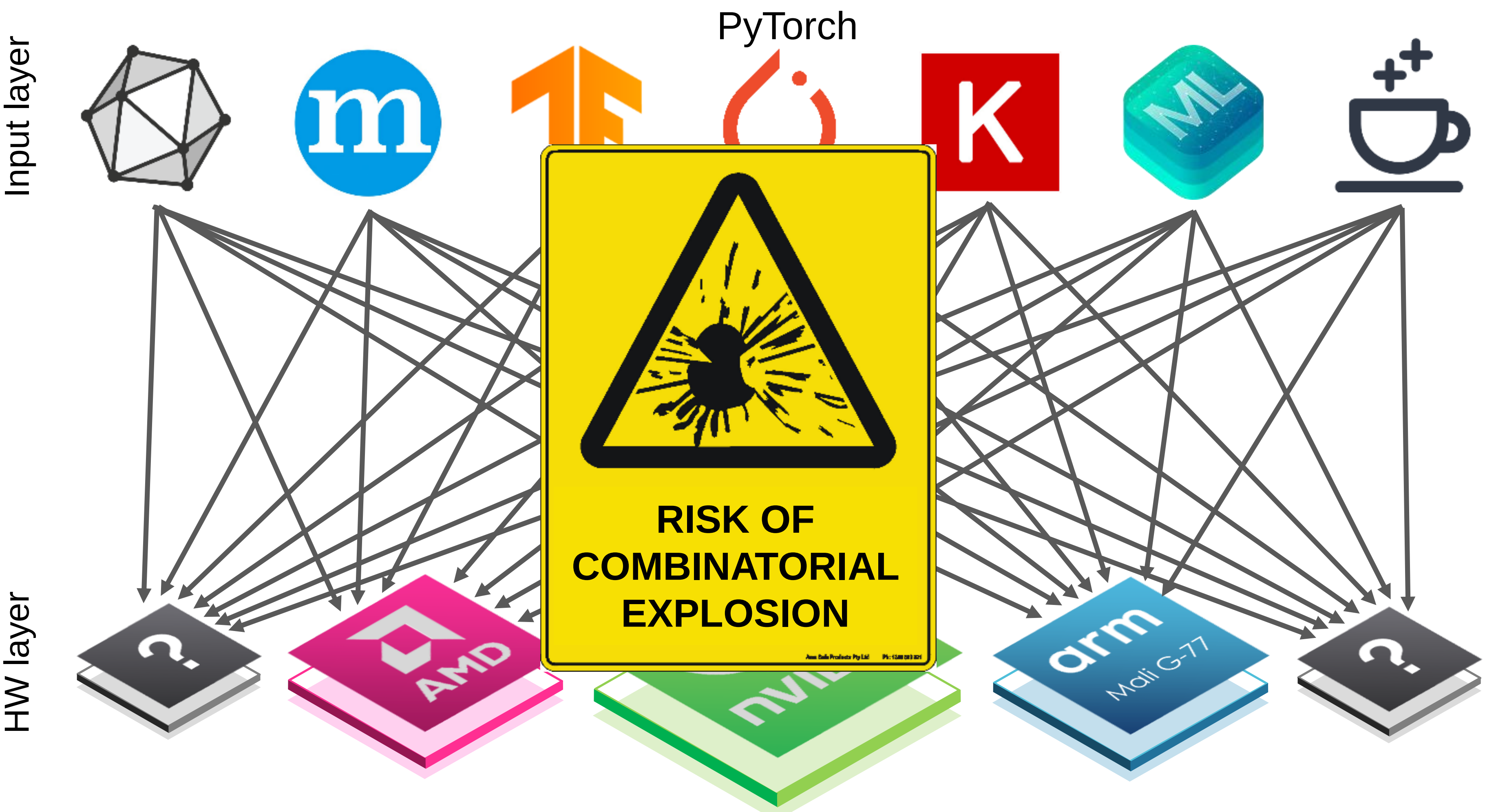
Managing overheads

- Portability issues (when targeting multiple types of GPUs):

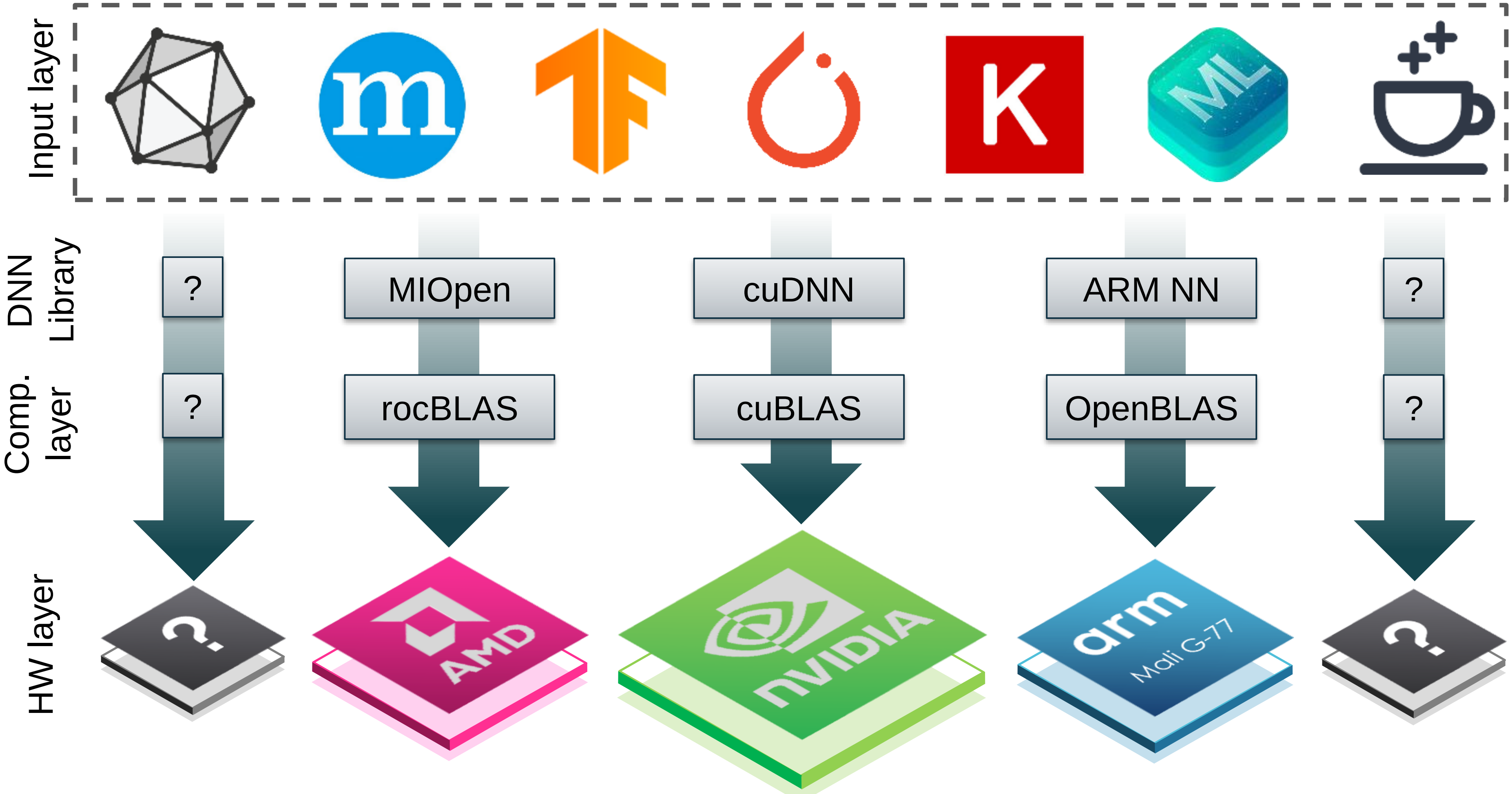
Incompatible GPU programming models

GPU-hardware-specific constraints

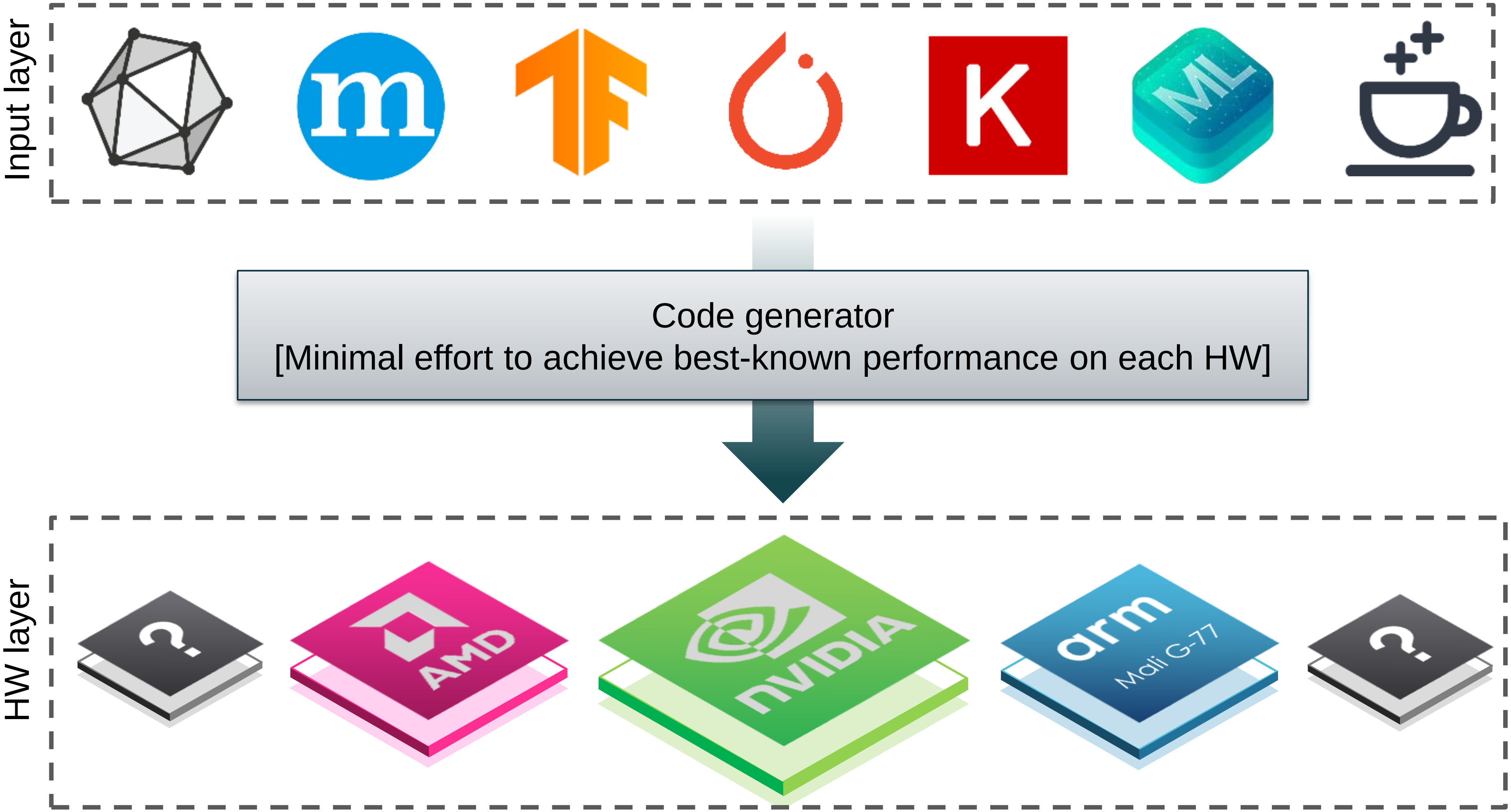
From design to production



From design to production with pain



From design to production without pain
[Performance portable approach]



Solution: Template metaprogramming

Metaprogramming is writing programs that output programs. It addresses:

- **OpenCL/CUDA/Vulkan incompatibilities**
  - Use templates and keywords to bridge syntax differences
- **Load/Store/Multiply sequences**
  - Use code generation to emit long sequences
- **Control Overhead**
  - Specialize code for particular inputs to remove conditionals
- **Loop Overhead**
  - Specialize code to make fixed-length and/or pre-unrolled loops
- **Wide range of input sizes**
  - Select different algorithms (*variants*)

MetaGPU abstraction layer

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Single source (portability)
- Compatibility layer over our target APIs
- Abstracts away the syntactic differences for the basic GPU programming concepts shared by our target APIs
- Simple to use, very similar to OpenMP

1 Tuning parameters

```
#pragma metagpu tuning_knobs
{
  int wg_size_x;
  int unroll_lvl;
}
```

2 Data layout

```
#pragma metagpu data_layout \
in(a,b) out(c) shared(in_smem)
{
  float const * const a;
  float const * const b;
  float * c;
  float in_smem[%(dim)*%(dim)];
}
```

3 Kernel body

```
#pragma metagpu kernel_body {
 for(k=0;k<%(dim);k+=unroll_lvl){
   %(sm_loads);
   BARRIER_SYNC;
   %(inner_loop_body);
 }
}
```

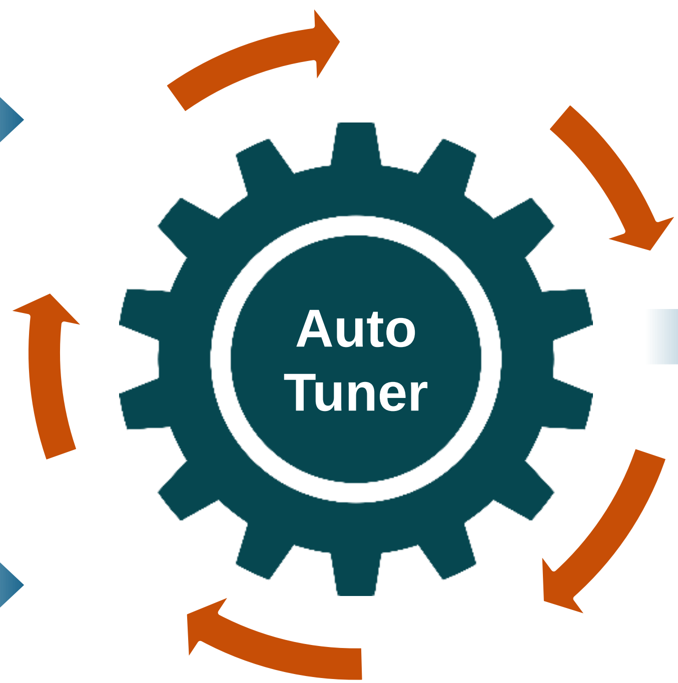
Performance auto-tuning

Tuning knobs

- Thread blocking
- Register blocking
- Loop unrolling factor
- Vector width
- Convolution variants

Tensor operation kernel

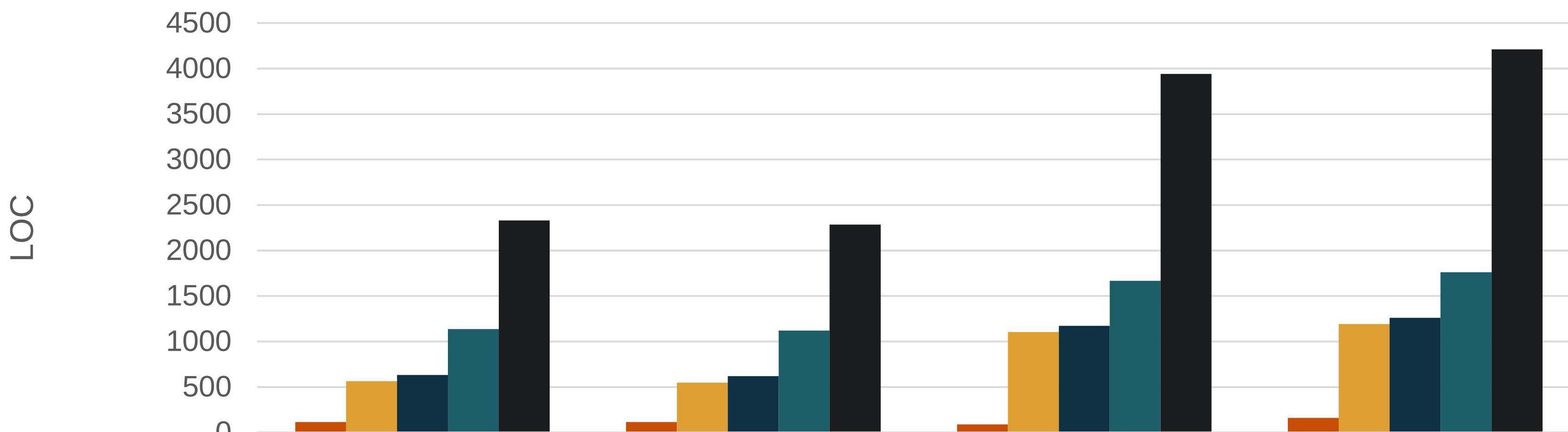
```
float g[7][7];
float Gg[8][7];
float tmp[8][8];

const GASQ float *B = filts_ref + (k * 3 +
c) * 7 * 7;
for (int i = 0; i < 7; ++i) {
  for (int j = 0; j < 7; ++j) {
    g[i][j] = B[7*j + i];
  }
}
```



Lowest runtime kernel

Quantitative programmability analysis

- Programming effort metric: $$Effort[\%] = \left(\frac{LOC_{MetaGPU}}{LOC_{TotalUniqueLines}}\right) * 100$$

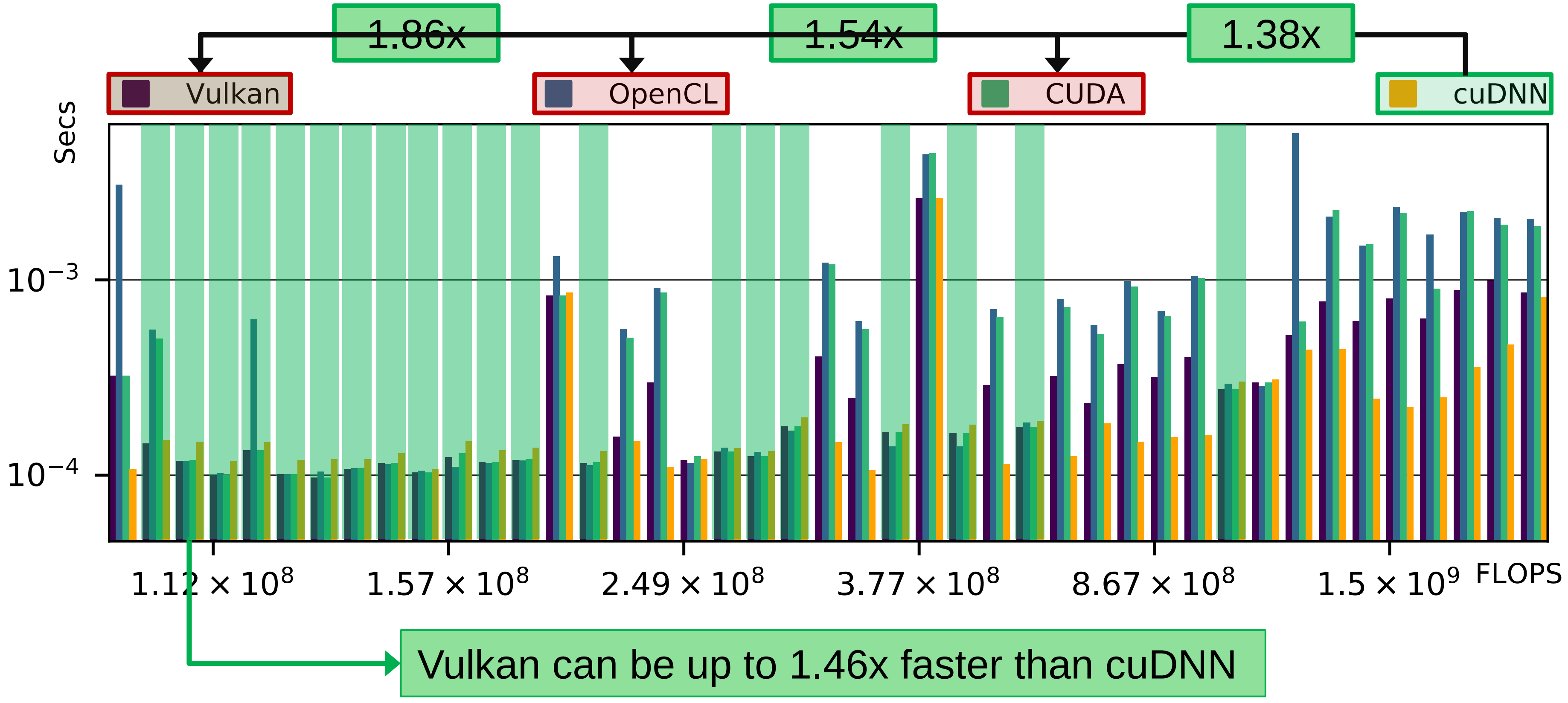


	Direct	Tiled	GEMM	1x1
LOC_MetaGPU	113	115	89	160
LOC_CUDA	562	[illegible]	[illegible]	1190
LOC_OpenCL	631	[illegible]	[illegible]	1259
LOC_Vulkan	1137	[illegible]	[illegible]	1761
LOC_Total	2330	[illegible]	[illegible]	4210
Effort	4,85%	5,03%	2,26%	3,80%

On average: ~4 % of the total effort

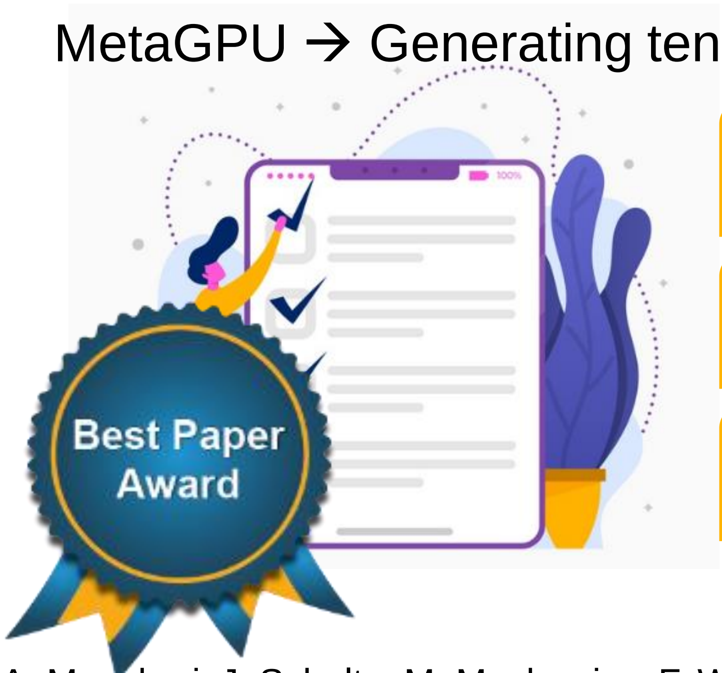
Performance portability [Nvidia GPU]

- Runtime comparison of kernels generated by our method with cuDNN vendor library on Nvidia GTX 1080 Ti.



Conclusion

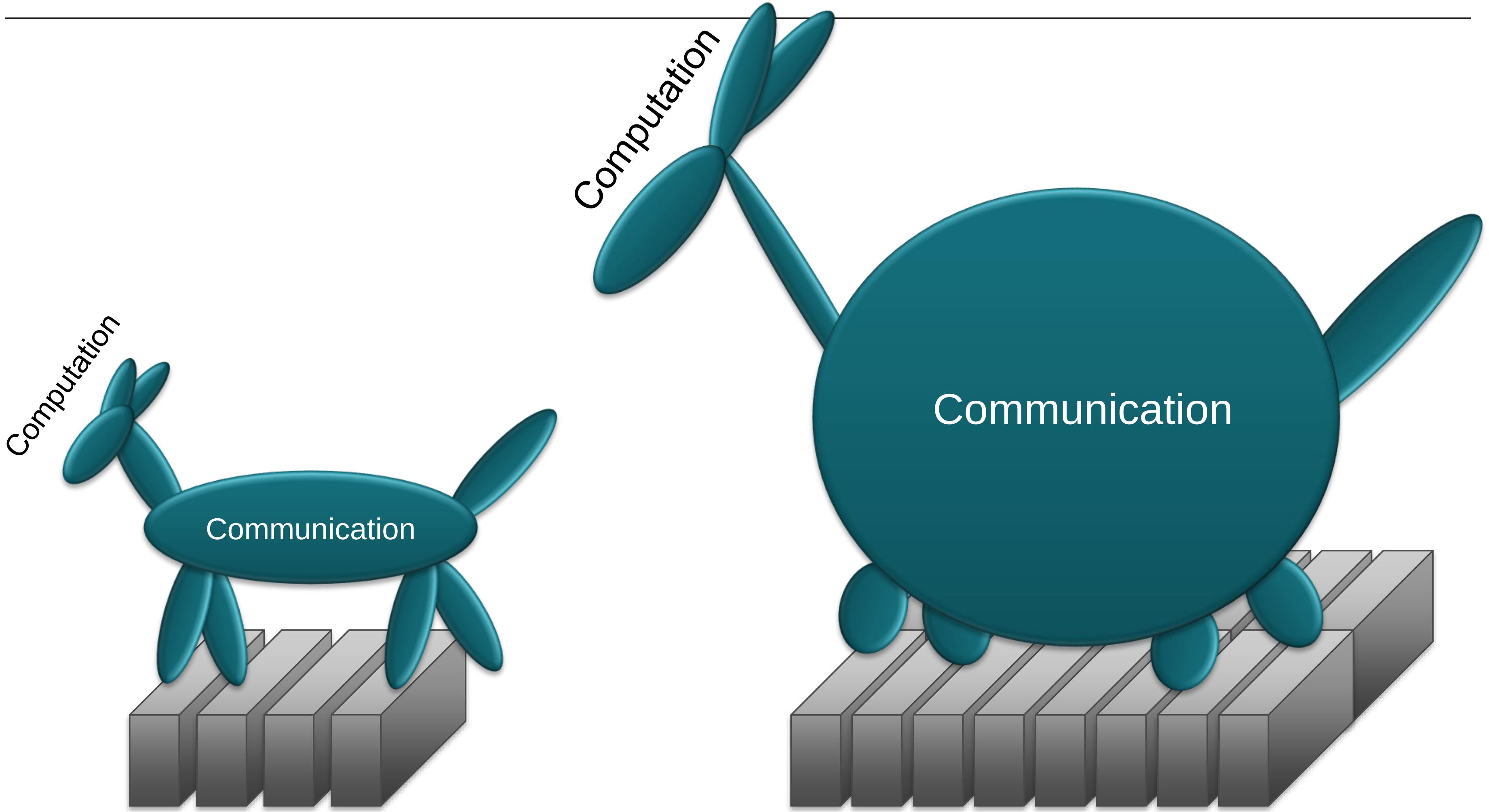
- Comparative analysis of the GPU programming interfaces CUDA, OpenCL, and Vulkan
- MetaGPU → Generating tensor ops in CUDA/OpenCL/Vulkan



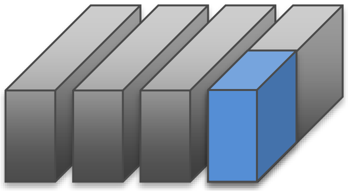
1. ~96% programming effort reduction
2. Performance portability on three different architectures
3. Vulkan can yield faster runtime than OpenCL

A. Mazaheri, J. Schulte, M. Moskewicz, F. Wolf, A. Jannesari: Enhancing the Programmability and Performance Portability of GPU Tensor Operations. In *Proc. of the 25th Euro-Par Conference, Göttingen, Germany*, volume 11725 of *Lecture Notes in Computer Science*, pages 213–226, Springer, August 2019

Performance modeling at a discount: Predicting performance at scale is hard



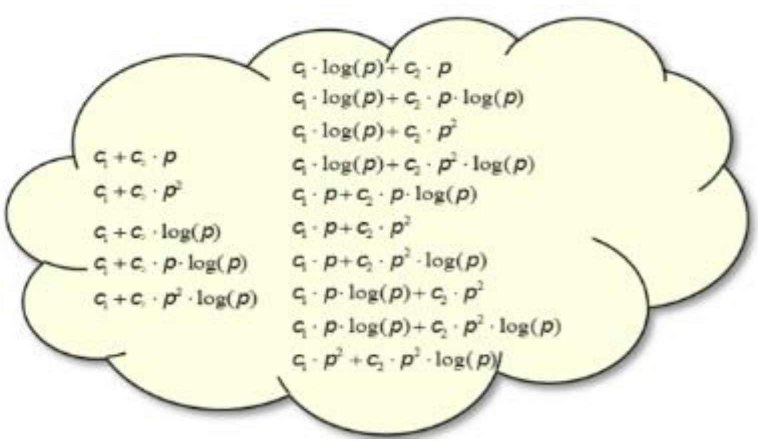
Extra-P



Small-scale measurements

$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(p)$$

Performance model normal form (PMNF)



Generation of candidate models and selection of best fit

Kernel [2 of 40]	Model [s] t = f(p)
sweep → MPI_Recv	$4.03\sqrt{p}$
sweep	582.19

http://www.scalasca.org/software/extra-p/download.html

Automatic empirical performance modeling with multiple parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

Expanded PMNF

$$f(x_1,\dots,x_m)=\sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

$n=3$

$m=3$

$I=\left\{\frac{0}{4},\frac{1}{4},\dots,\frac{12}{4}\right\}$

$J=\{0,1,2\}$

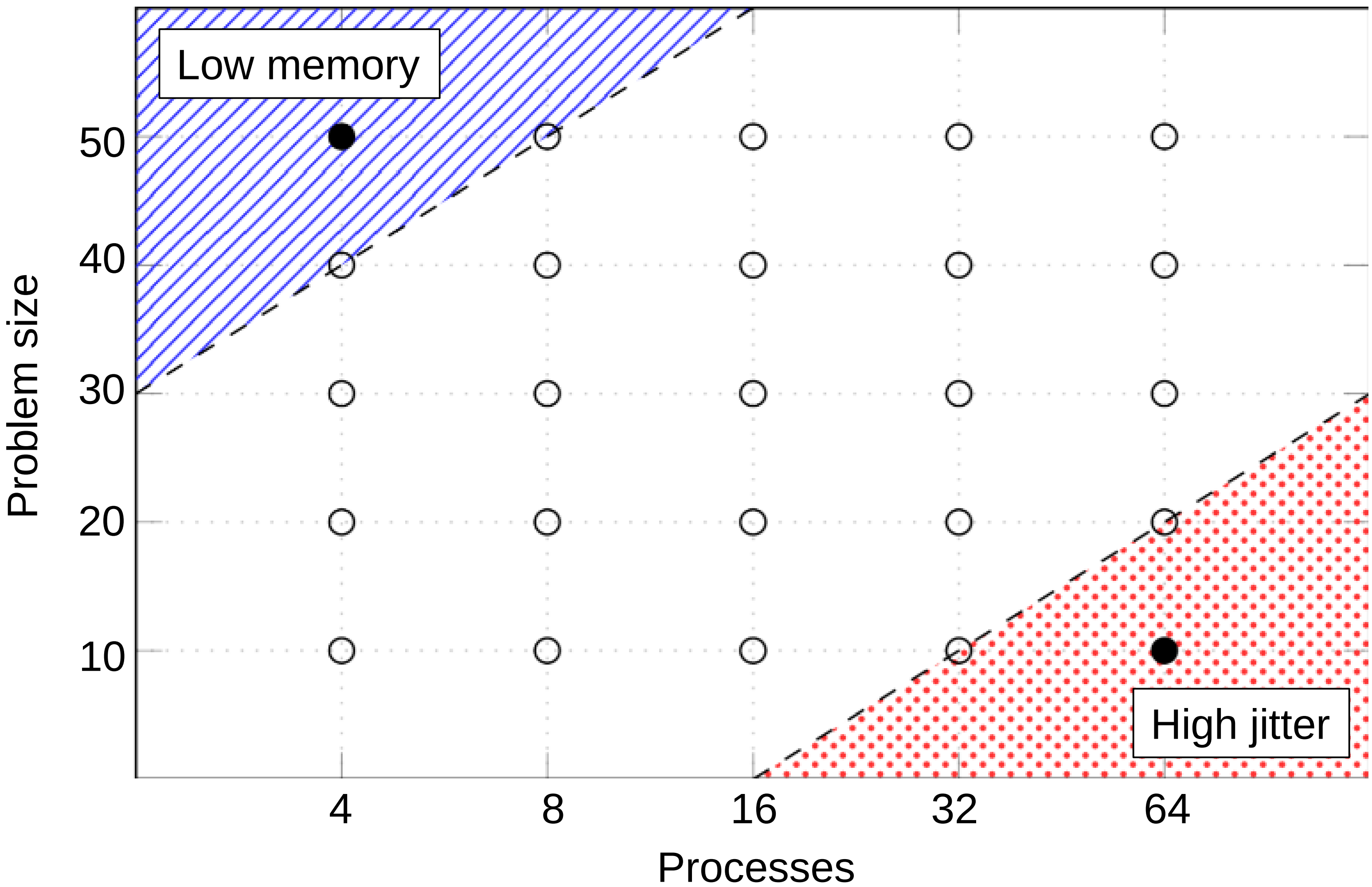
Search space explosion

- Total number of hypotheses to search: 34.786,300,841,019

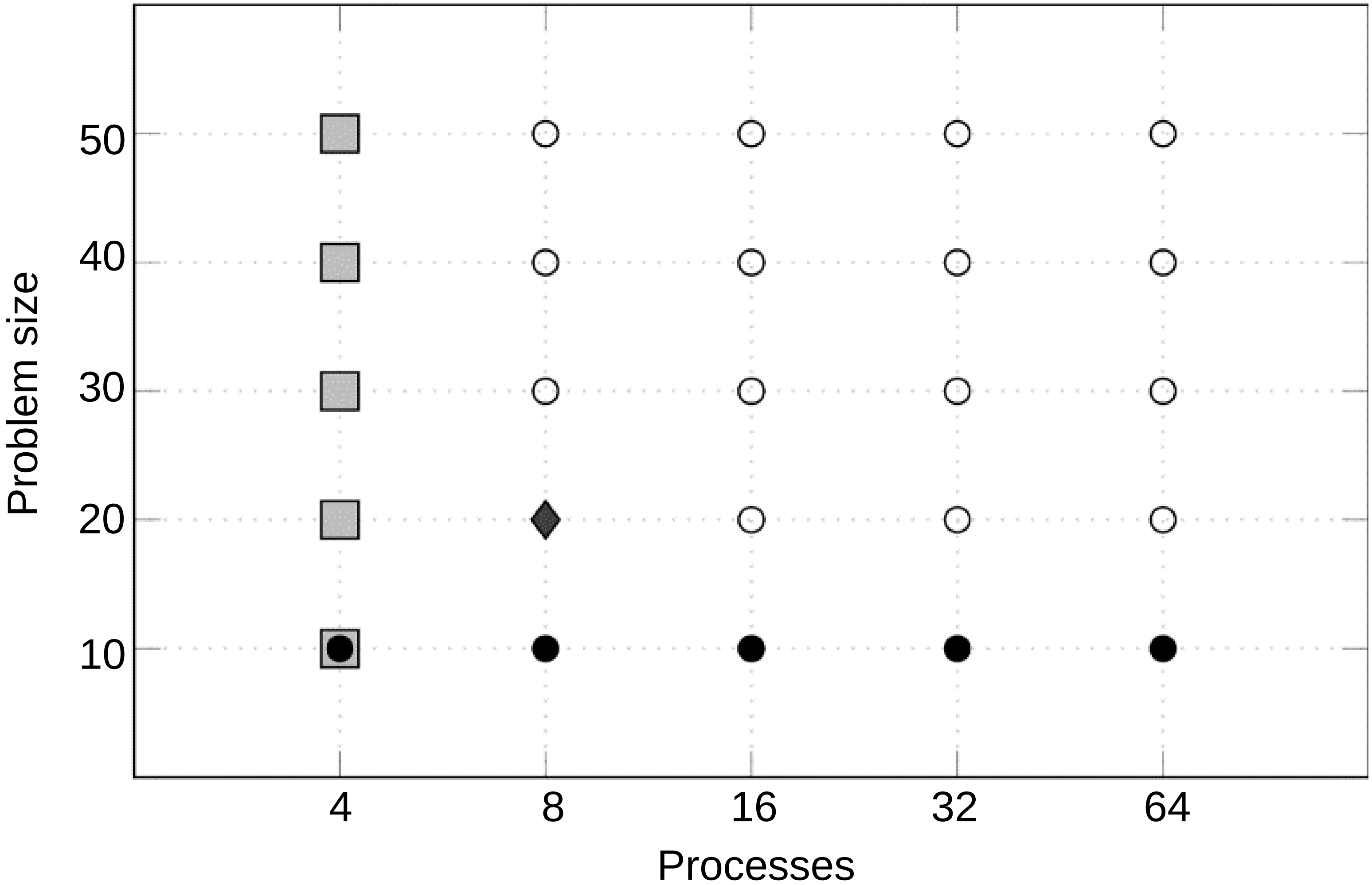
Heuristics make the search faster

- **Hierarchical search**
- Modified golden section search

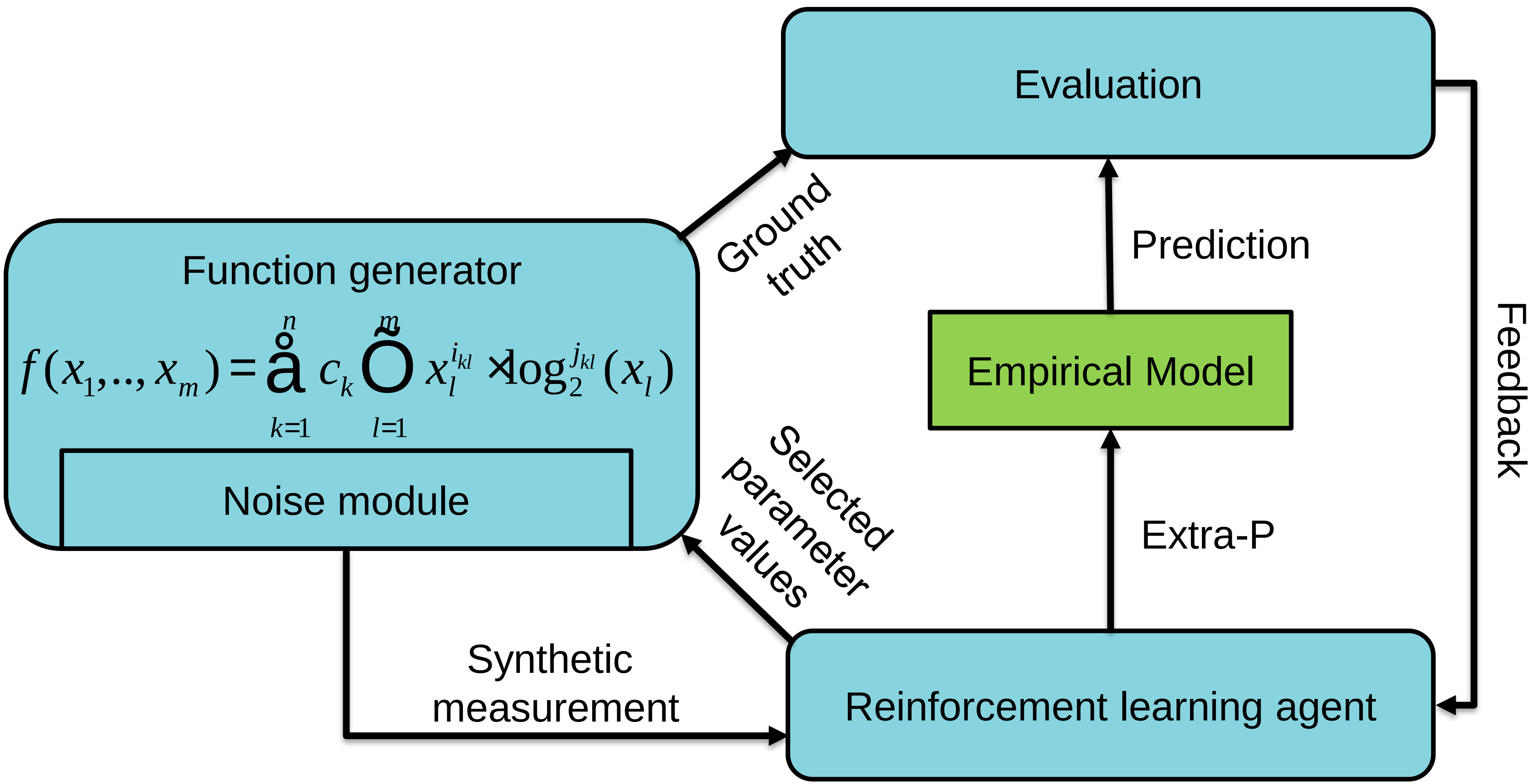
Gathering measurements: A difficult and expensive task



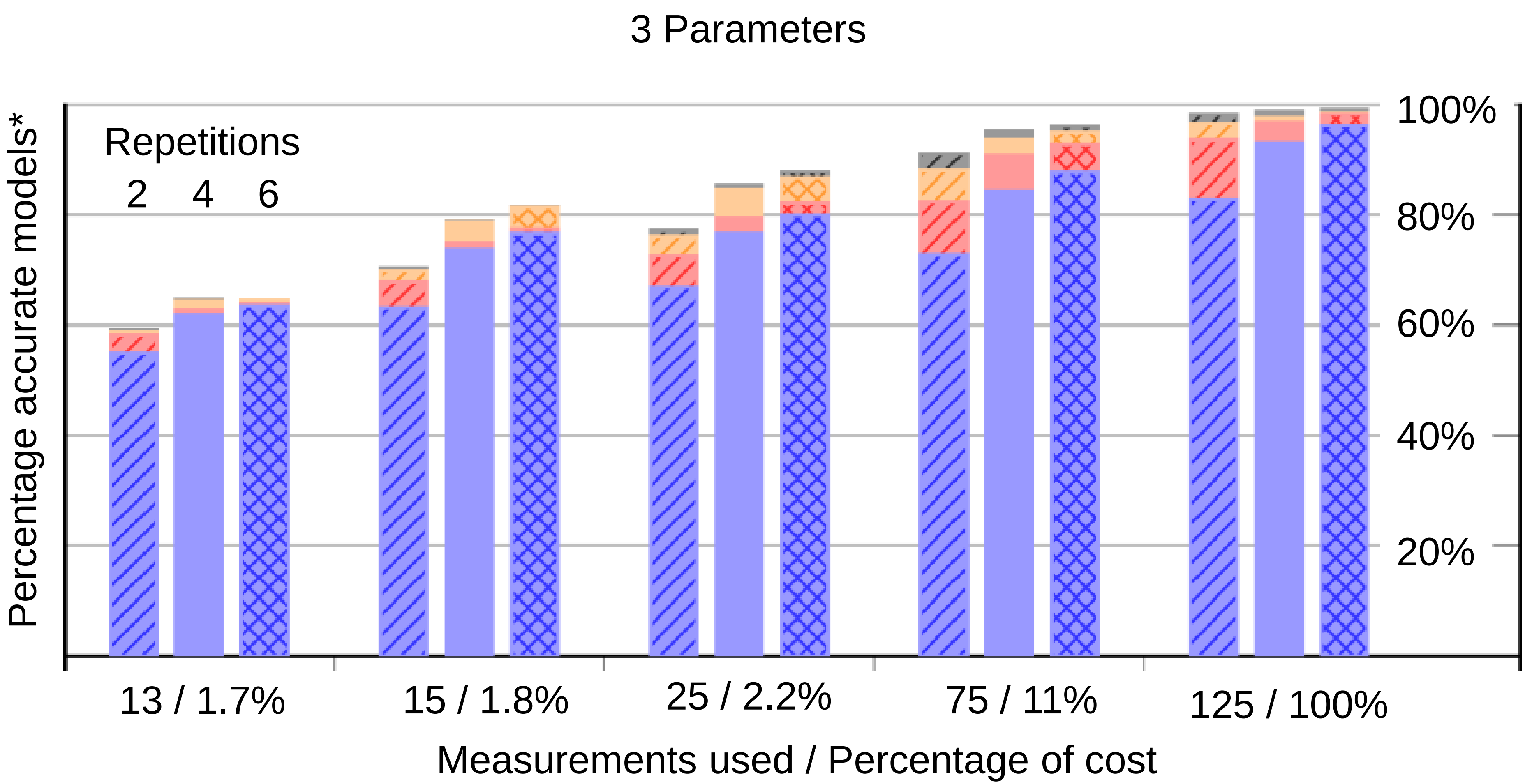
Sparse modeling – Let‘s model with less



Parameter value selection strategy



Sparse modeling accuracy



Cheap is good!

1. Gather the minimum set of measurements

2. Create a model

1. Gather one additional measurement

2. Evaluate previous model

3. Create new model

How Deep Learning Makes Compiler Optimization More Effective

TECHNISCHE UNIVERSITÄT DARMSTADT

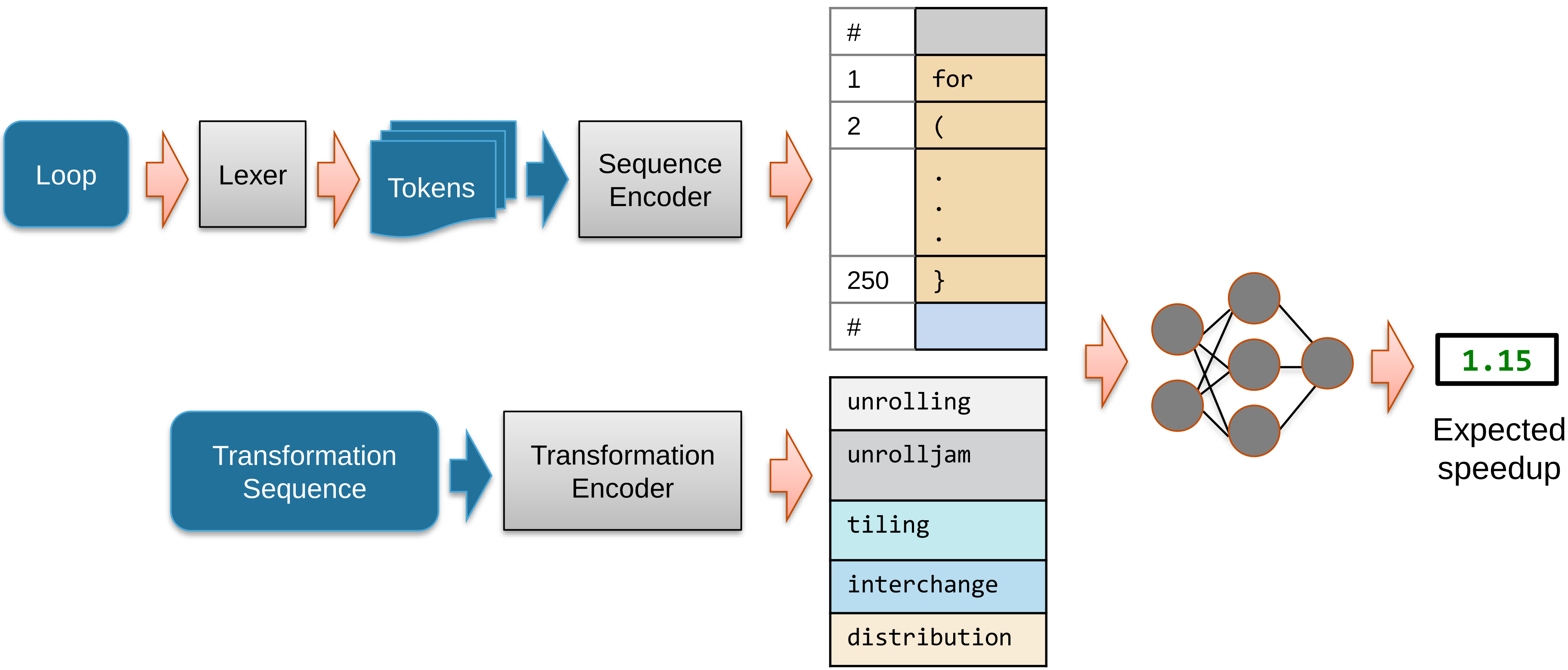
Problem

- Compiler optimization results inconsistent when processing semantically-equivalent code [Gong et al., 2018]
- Programs usually spend most of the time in loops
- Which loop representation will yield the best performance?

```
/* original loop */
for(i = 0; i < 256; ++i) {
    do_something(i);
}

/* unrolled, factor = 2 */
for(i = 0; i <= 255; i += 2)
{
    do_something(i);
    do_something(i + 1);
}
```

Example transformation

Approach



Loop
Lexer
Tokens
Sequence Encoder
#
1
for
2
(
.
.
.
250
}
#
Transformation Sequence
Transformation Encoder
unrolling
unrolljam
tiling
interchange
distribution
1.15
Expected speedup

Results

Static mode (w/o validation)

1.14x speedup

Dynamic mode (w/ validation)

Top-1 1.23x speedup

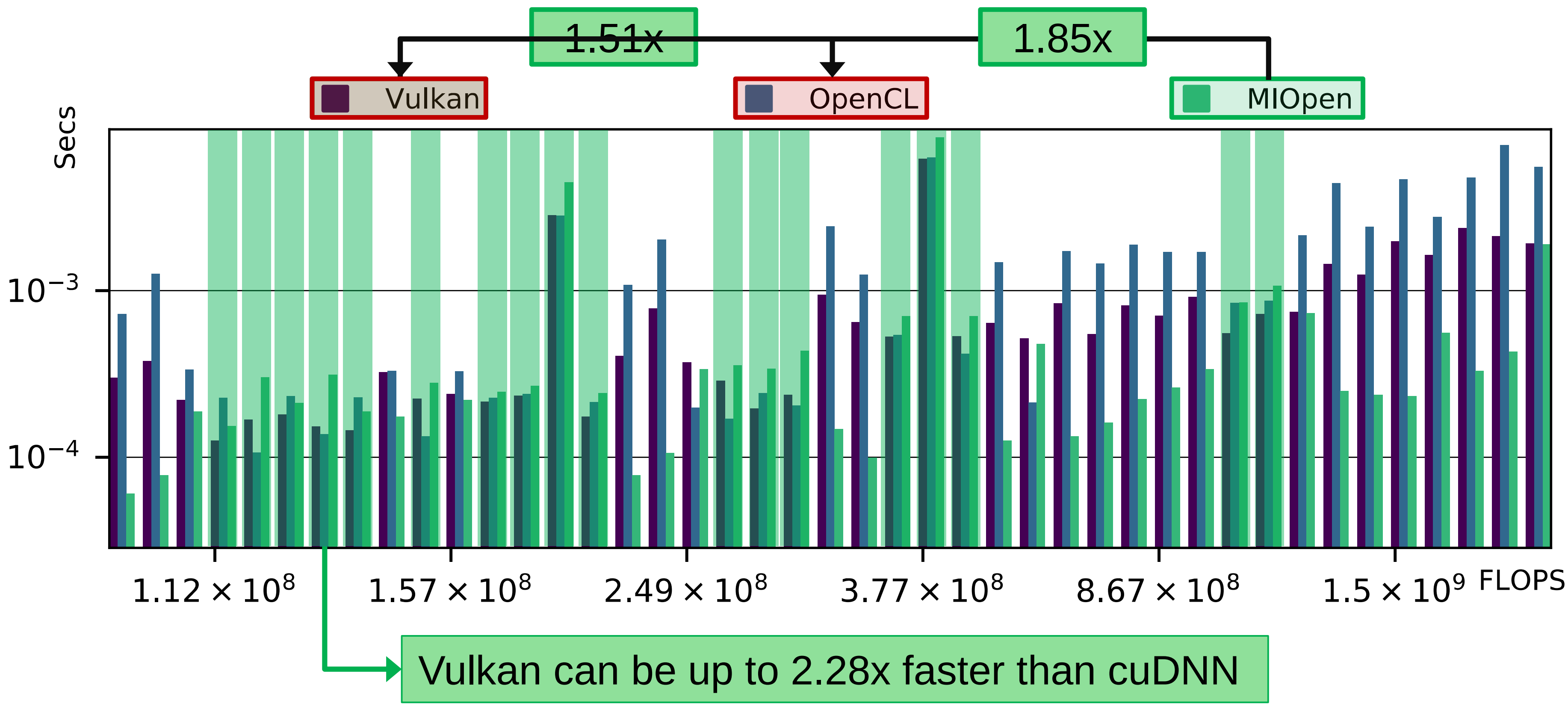
Top-3 1.28x speedup

Top-5 1.29x speedup

Thank you!

Performance portability [AMD GPU]

- Runtime comparison of kernels generated by our method with the MIOpen vendor library on AMD Radeon RX 580.



Performance portability [Mobile GPU]

- Vulkan performance with and without auto-tuning on Mali G71.