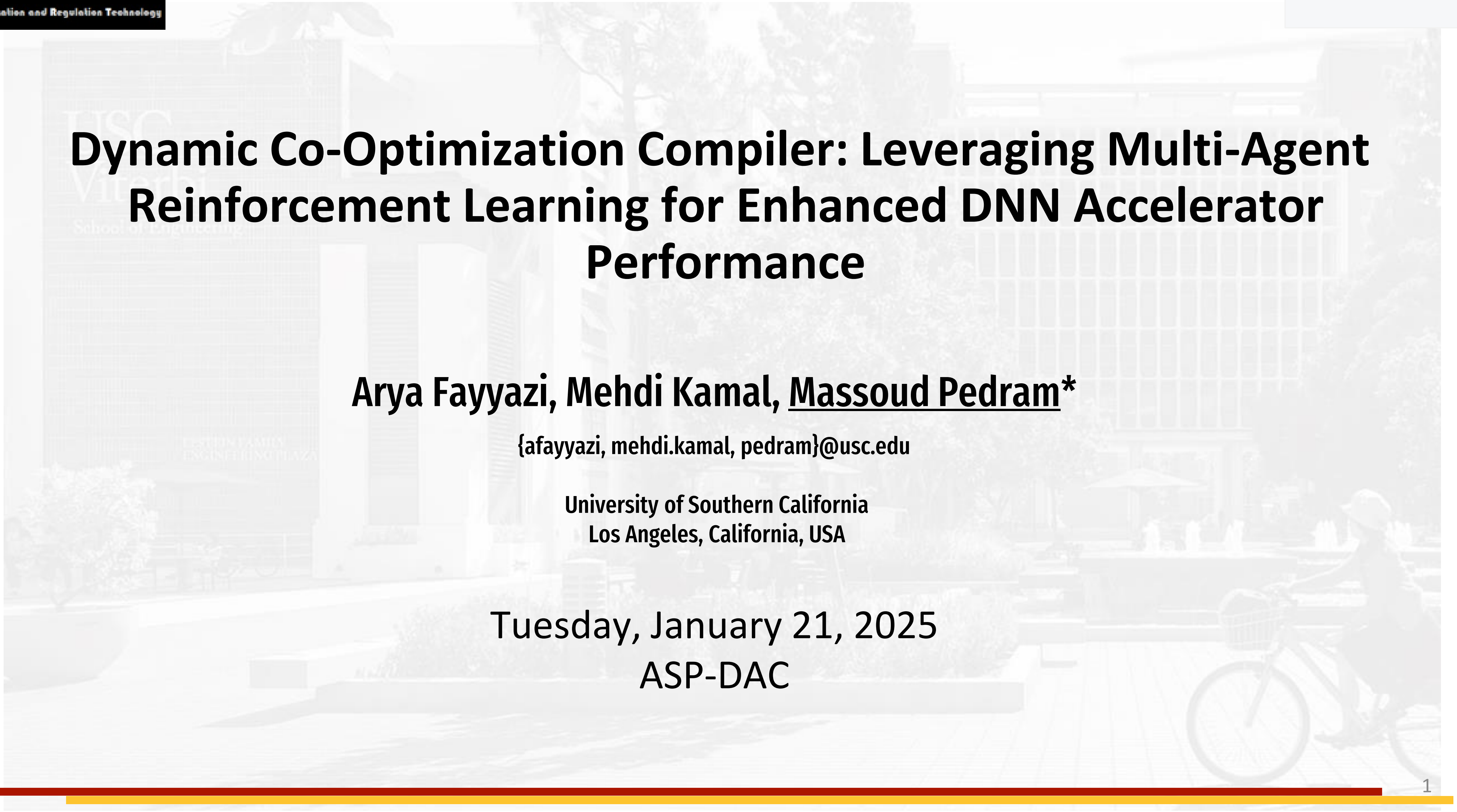


Dynamic Co-Optimization Compiler: Leveraging Multi-Agent Reinforcement Learning for Enhanced DNN Accelerator Performance

Arya Fayyazi, Mehdi Kamal, Massoud Pedram*

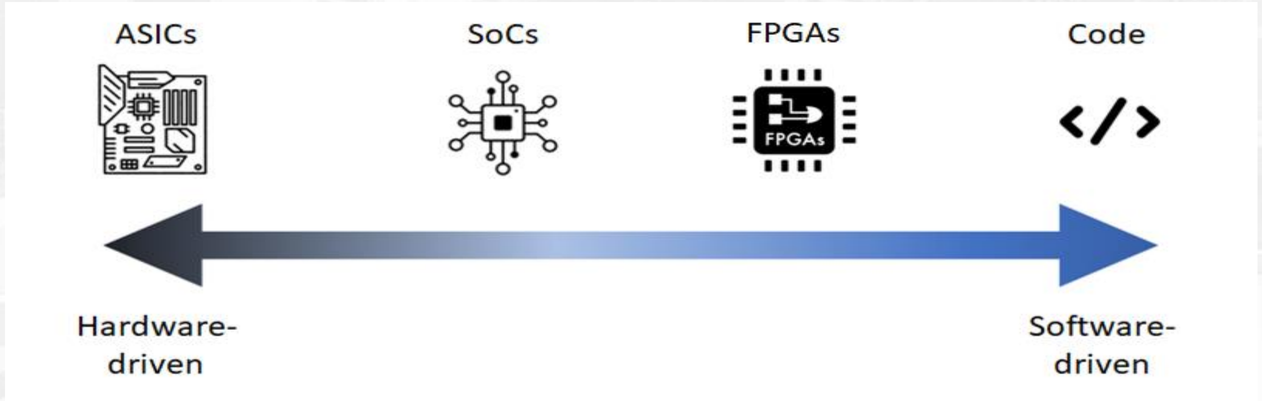
{afayyazi, mehdi.kamal, pedram}@usc.edu

University of Southern California
Los Angeles, California, USA

Tuesday, January 21, 2025
ASP-DAC

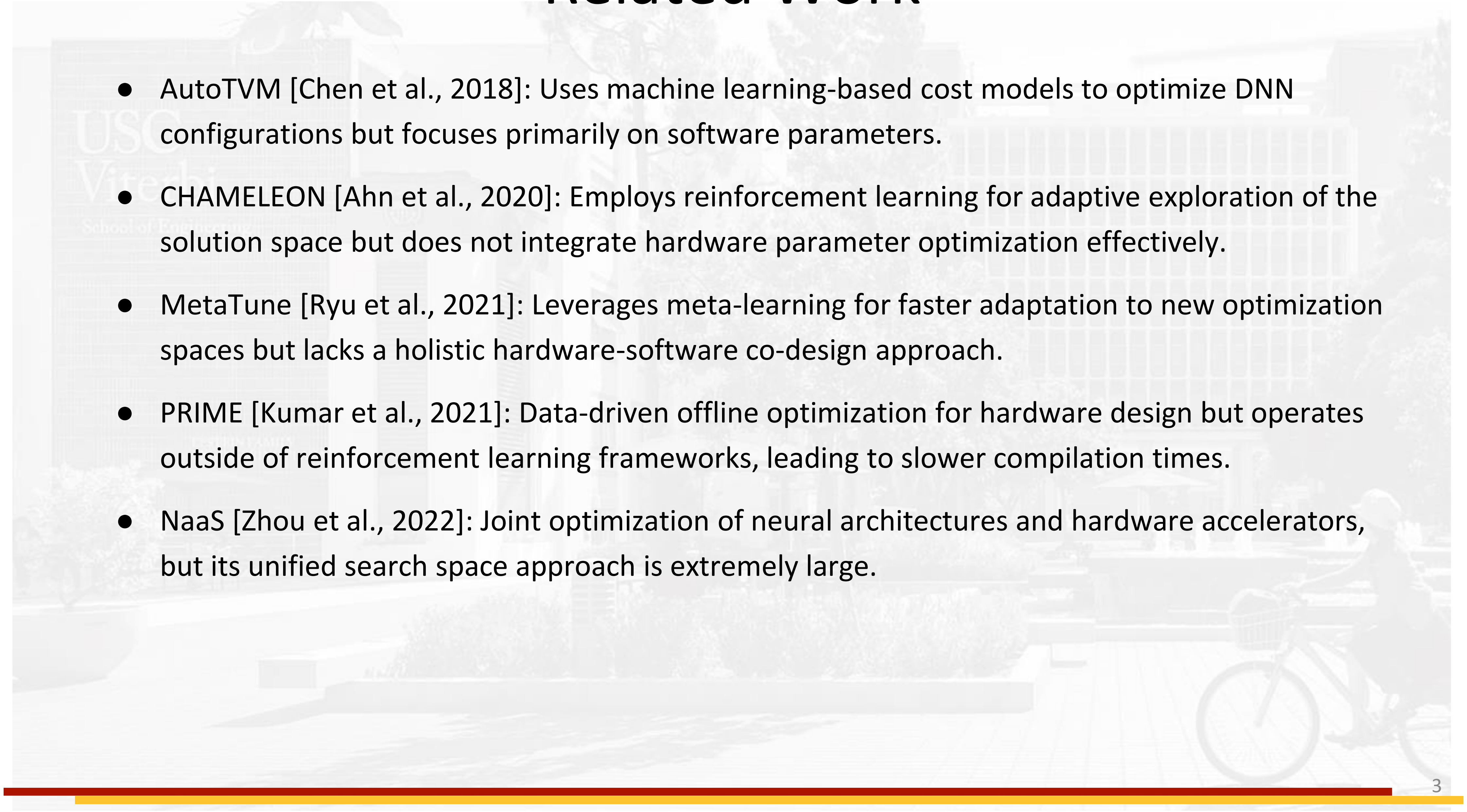
Motivation

- Increasing Complexity of neural network model
  - Advanced architectures and large-scale workloads demand more than mere software tweaks.
- Limitations of Existing Auto-Tuners
  - Traditional frameworks (e.g., TVM [Chen et al., 2018]) primarily focus on software optimizations, leaving hardware optimization potential largely untapped.
- Need for Hardware–Software Synergy
  - Jointly optimizing both layers is critical for peak performance but is vastly underexplored.



“Software and Hardware.” Altium Resources, Altium, files.resources.altium.com

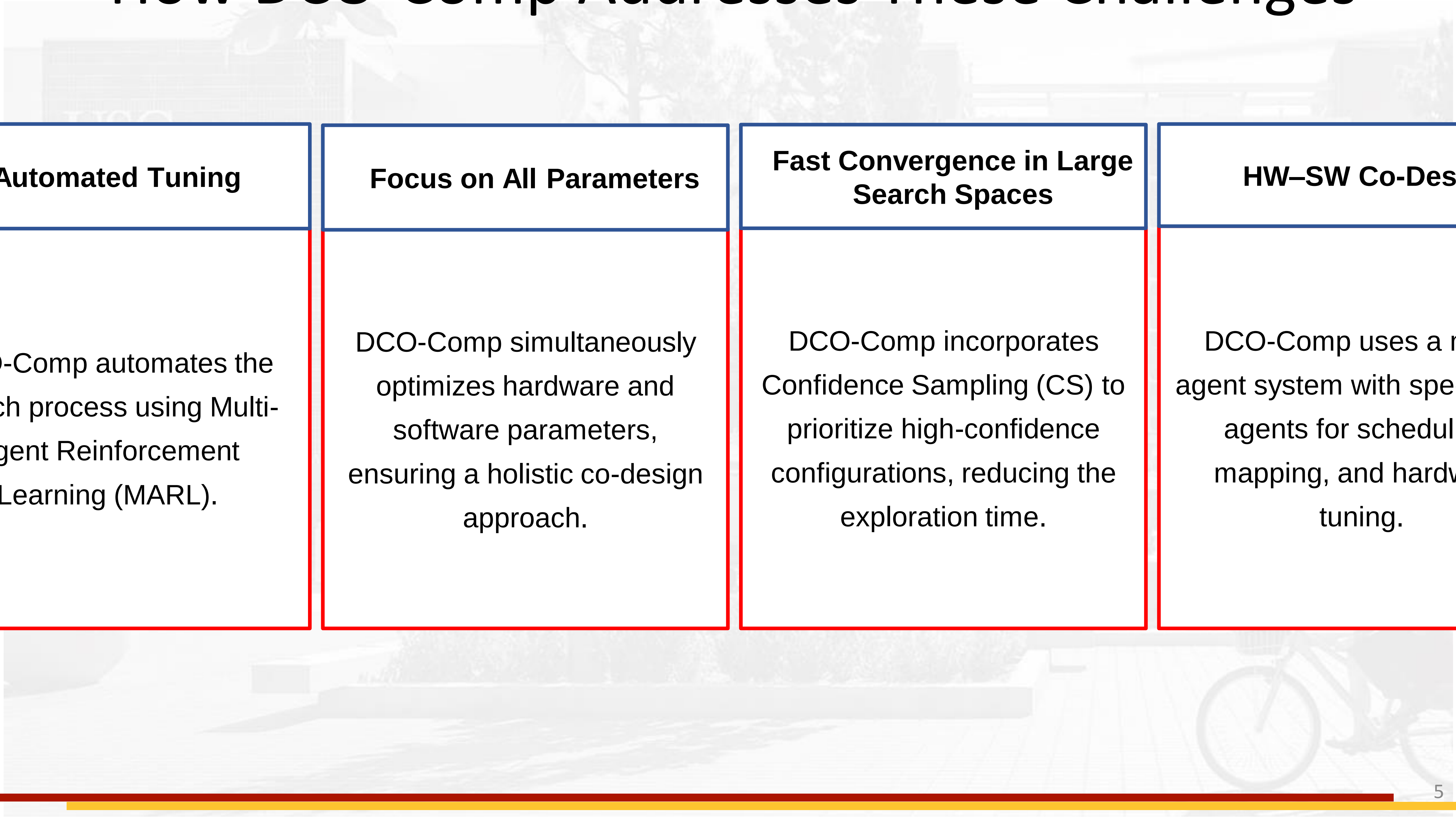
Related Work



- AutoTVM [Chen et al., 2018]: Uses machine learning-based cost models to optimize DNN configurations but focuses primarily on software parameters.
- CHAMELEON [Ahn et al., 2020]: Employs reinforcement learning for adaptive exploration of the solution space but does not integrate hardware parameter optimization effectively.
- MetaTune [Ryu et al., 2021]: Leverages meta-learning for faster adaptation to new optimization spaces but lacks a holistic hardware-software co-design approach.
- PRIME [Kumar et al., 2021]: Data-driven offline optimization for hardware design but operates outside of reinforcement learning frameworks, leading to slower compilation times.
- NaaS [Zhou et al., 2022]: Joint optimization of neural architectures and hardware accelerators, but its unified search space approach is extremely large.

Shortcomings of Existing Approaches

Manual Tuning Overhead	Partial Focus on Parameters	Slow Convergence in Large Search Spaces	Lack of HW–SW Co-Design
Hand-optimized kernels are difficult to design and generally non-scalable.	Automated frameworks (AutoTVM, CHAMELEON) concentrate on software parameters and often neglect hardware parameters.	Struggle to find near-optimal solutions quickly (MetaTune, PRIME).	Fail to do hardware and software co-optimizations (CHAMELEON, NaaS).



How DCO-Comp Addresses These Challenges

Automated Tuning

DCO-Comp automates the search process using Multi-Agent Reinforcement Learning (MARL).

Focus on All Parameters

DCO-Comp simultaneously optimizes hardware and software parameters, ensuring a holistic co-design approach.

Fast Convergence in Large Search Spaces

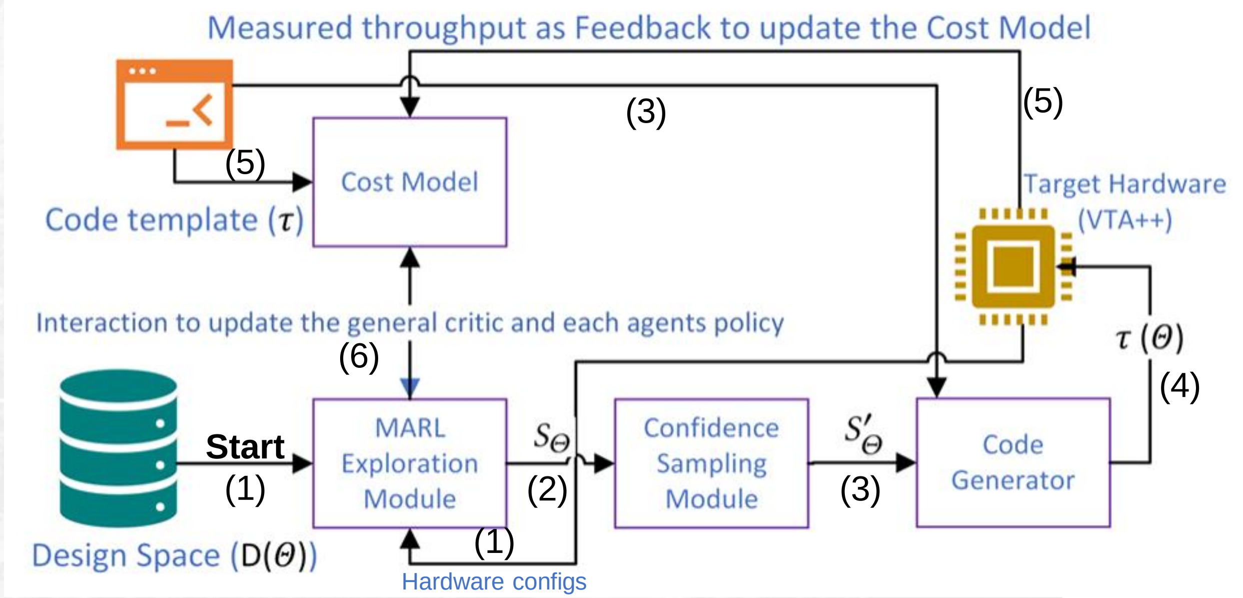
DCO-Comp incorporates Confidence Sampling (CS) to prioritize high-confidence configurations, reducing the exploration time.

HW–SW Co-Design

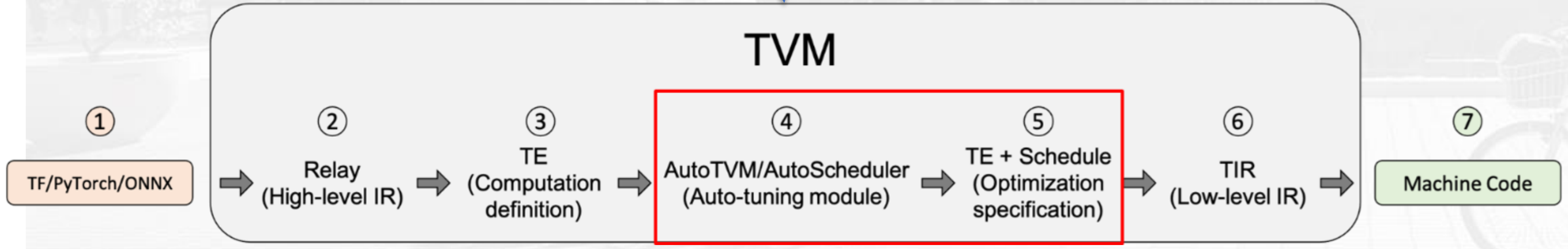
DCO-Comp uses a multi-agent system with specialized agents for scheduling, mapping, and hardware tuning.

DCO-Comp : Dynamic Co-Optimization Compiler

- **Multi-agent RL for setting hardware and software knobs**
- **Confidence sampling for improved search efficiency**
- **Seamless integration with the TVM pipeline**



This diagram illustrates the overall search flow of DCOC. The numbers indicate the sequence of actions performed at each step. The process iterates in a loop, continuously refining configurations through feedback until the optimal configuration is found.



TVM
1 TF/PyTorch/ONNX
2 Relay (High-level IR)
3 TE (Computation definition)
4 AutoTVM/AutoScheduler (Auto-tuning module)
5 TE + Schedule (Optimization specification)
6 TIR (Low-level IR)
7 Machine Code

MARL Exploration Module

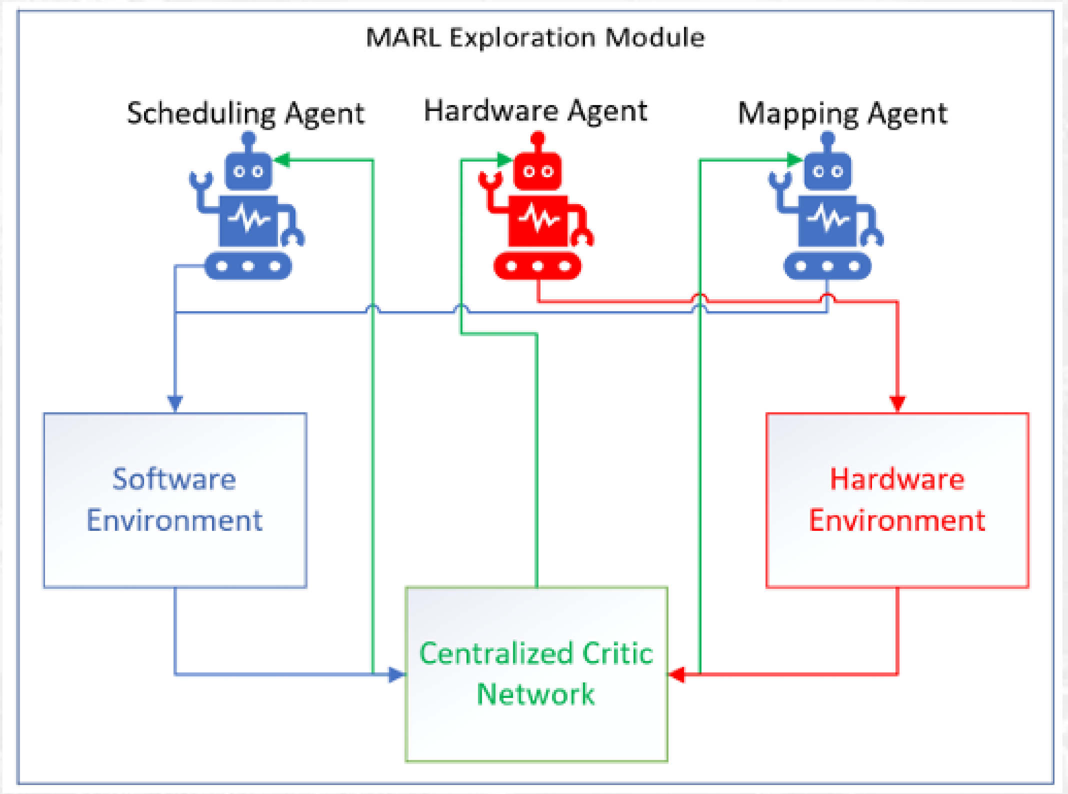
Three specialized agents:

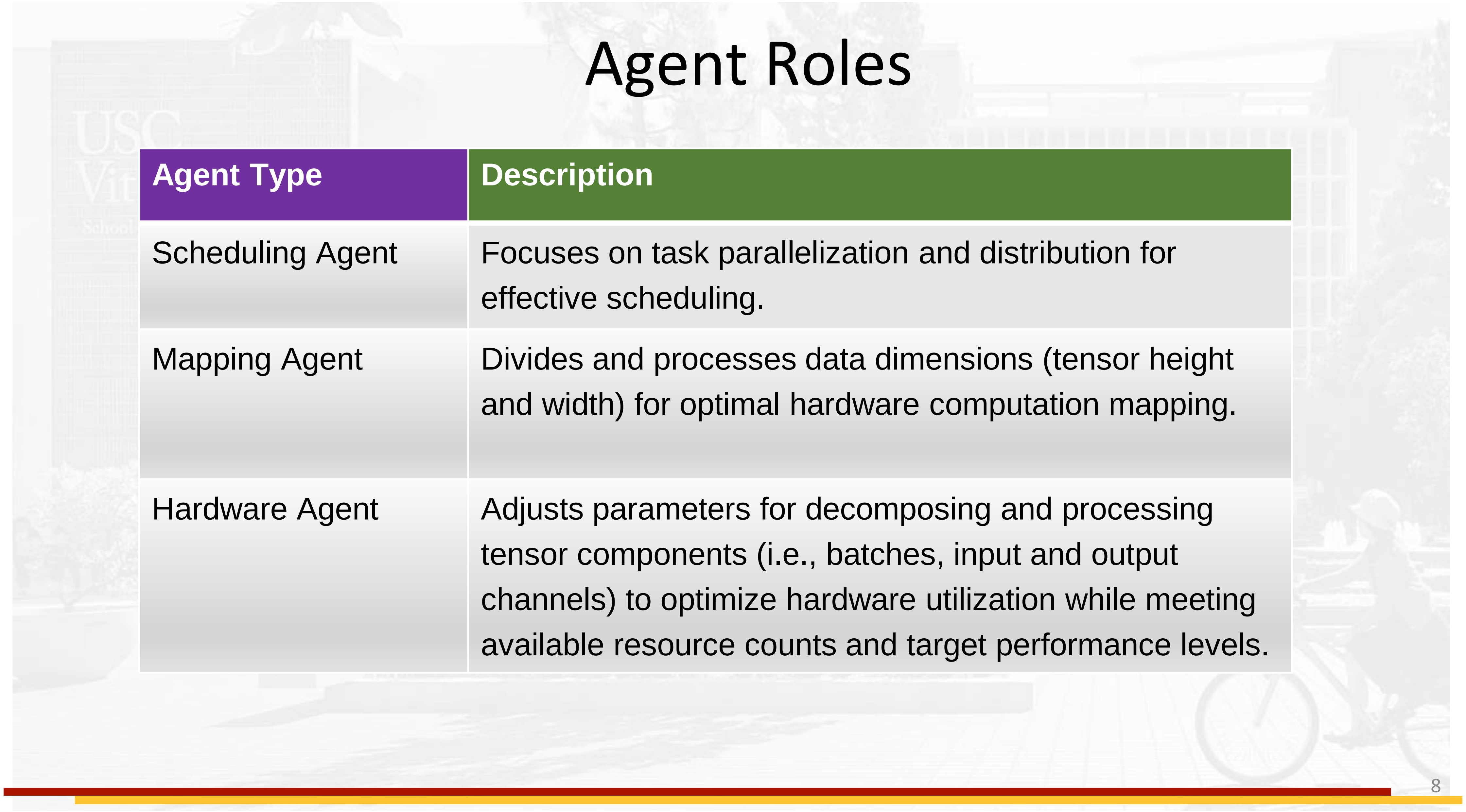
- Scheduling Agent
- Mapping Agent
- Hardware Agent

Shared Critic (value network)

Policy networks for each agent

Centralized Training, Decentralized Execution (CTDE)





Agent Roles

Agent Type	Description
Scheduling Agent	Focuses on task parallelization and distribution for effective scheduling.
Mapping Agent	Divides and processes data dimensions (tensor height and width) for optimal hardware computation mapping.
Hardware Agent	Adjusts parameters for decomposing and processing tensor components (i.e., batches, input and output channels) to optimize hardware utilization while meeting available resource counts and target performance levels.

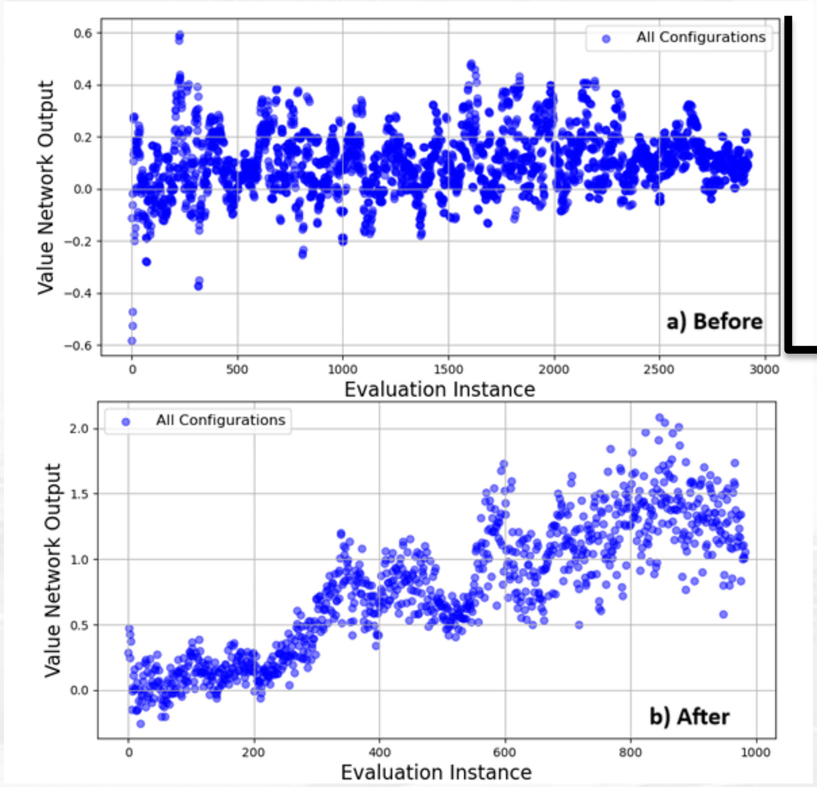
Knobs Controlled by Each Agent

Knobs in the design space to optimize convolution layers targeting VTA++ hardware [Banerjee et al., 2021]

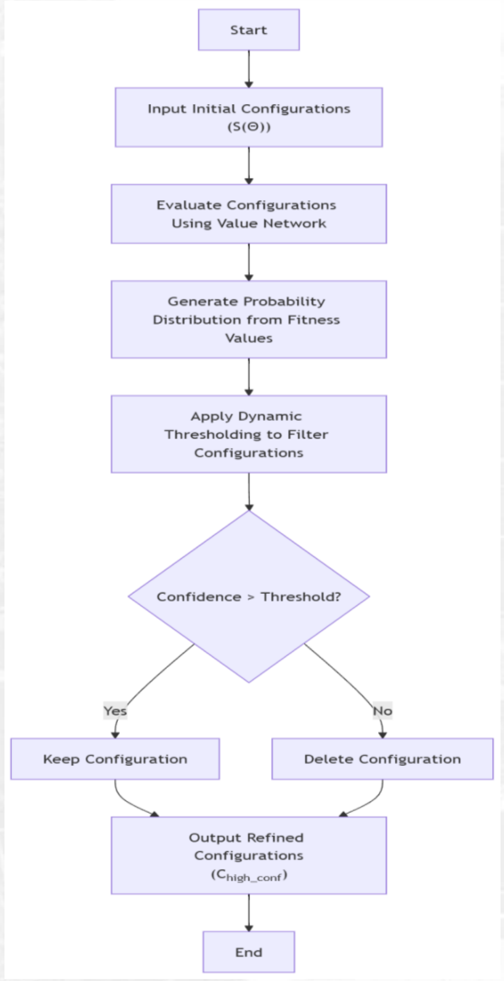
Agent Type	Knobs
Scheduling Agent	• Horizontal threading (h_threading) • Output channel threading (oc_threading)
Mapping Agent	• Tile across height (tile_h) • Tile across width (tile_w)
Hardware Agent	• Tile across batch size (tile_b) • Tile across input channels (tile_ci) • Tile across output channels (tile_co)

Confidence Sampling (CS)

- Probability-guided selection of configurations
- Filters low-confidence solutions
- Reduces measurement overhead



Configurations over time for ResNet-18 model a) before and b) after applying the CS method.



Start
Input Initial Configurations (S(Θ))
Evaluate Configurations Using Value Network
Generate Probability Distribution from Fitness Values
Apply Dynamic Thresholding to Filter Configurations
Confidence > Threshold?
Yes
No
Keep Configuration
Delete Configuration
Output Refined Configurations (C_high_conf)
End

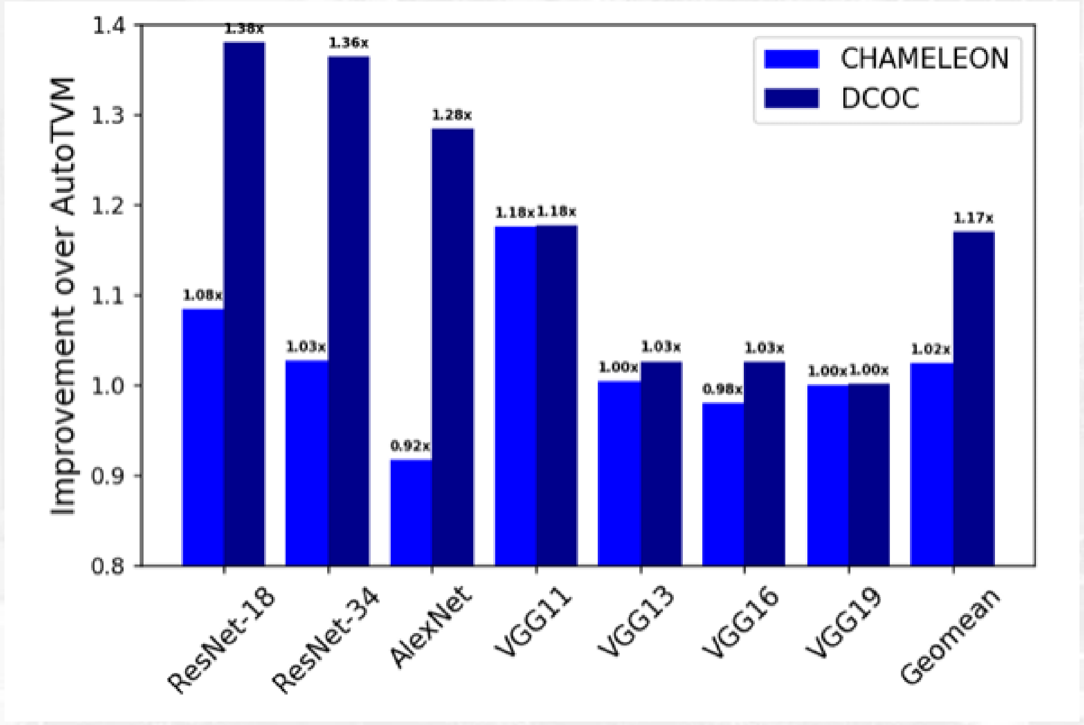
Fewer required configurations, also the sampling process gravitates towards configurations that exhibit superior performance over time, highlighting the efficacy of this method.

Experimental Settings & Platforms

Item	Description
Compiler Integration	• DCO-Comp is integrated into the TVM framework to enable end-to-end evaluation.
Comparison Baselines	• *AutoTVM*: Baseline automated tuner in TVM. • *CHAMELEON*: Existing RL-based tuner, with code from its original repository. *(Both baselines were extended to have hardware co-optimization for a fair comparison.)*
Hardware/Simulator	• Evaluations performed on an AMD EPYC 7763 64-Core CPU. • DCO-Comp targets the VTA++ simulator, which is configurable and representative of FPGA/GPU-like architectures.
Benchmarked Models	• *AlexNet*, *VGG* series (VGG-11, VGG-13, VGG-16, VGG-19) • *ResNet* series (ResNet-18, ResNet-34). • All models are compiled with the same total number of hardware measurements (e.g., 1,000) to ensure a fair comparison.
Hyperparameter Setup	• Key hyperparameters (e.g., for GBT in TVM, MAPPO for multi-agent RL) are consistent with standard practices from AutoTVM and CHAMELEON. • DCO-Comp employs offline-tuned parameters to optimize multi-agent RL exploration and Confidence Sampling.
Performance Metrics	• *Throughput* (in GFLOPS or equivalently, 1 / inference time) as runtime performance metrics. • *Compilation Time* (total optimization overhead) for each framework.

Improvement in Throughput

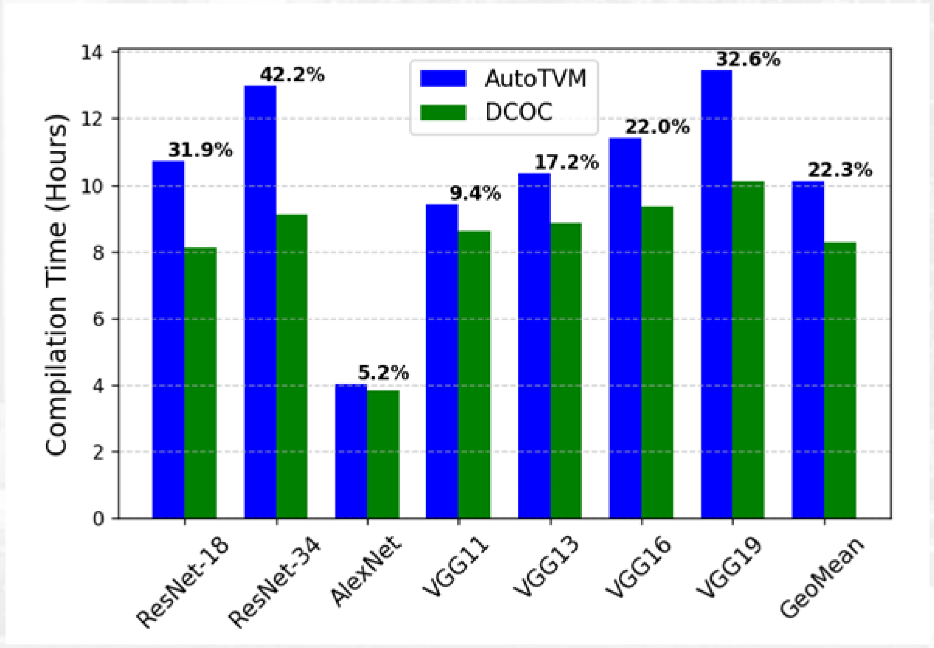
- DCO-Comp yields up to 37.95% improvement (17% on average)
- Gains across ResNet, AlexNet, VGG series



Comparing the achieved throughput of different frameworks over AutoTvm on VTA++

Reduction in Compilation Time

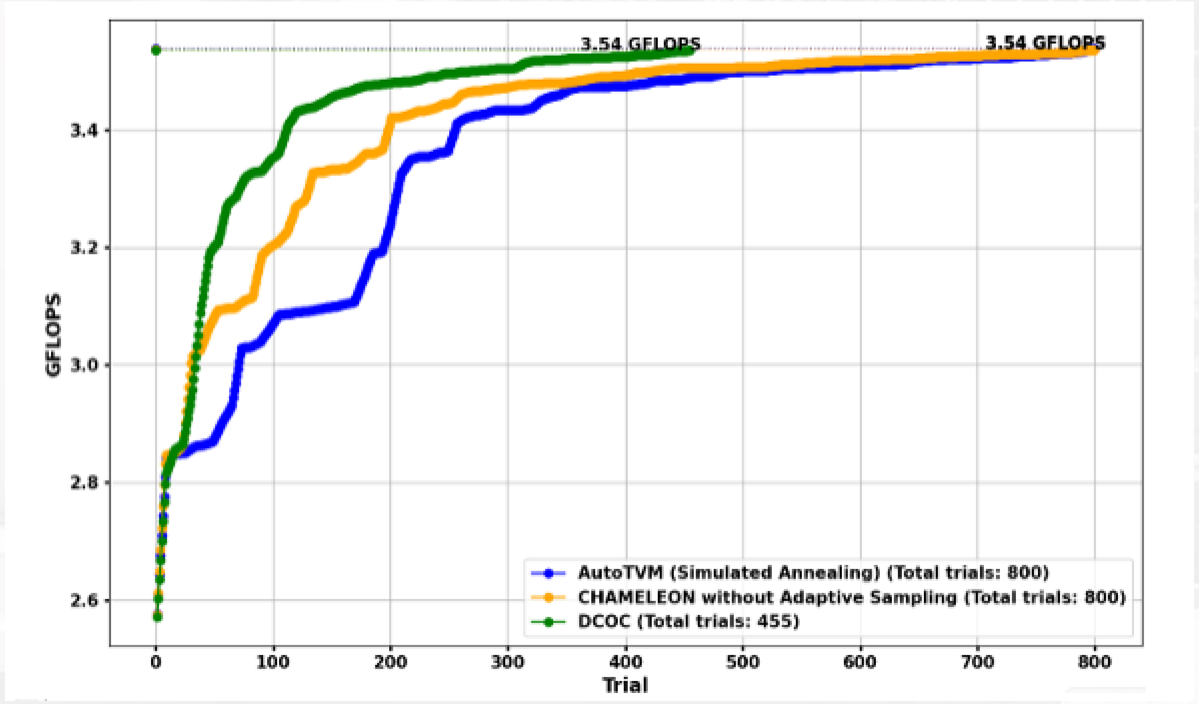
- DCO-Comp's improved search approach → up to 42.2% speedup in compile time over AutoTVM (22.3% on average)
- Confidence Sampling + MARL drastically cut down exploration overhead



Comparing the compilation time of different frameworks (The percentage show the speedup of DCOC compared to AutoTVM).

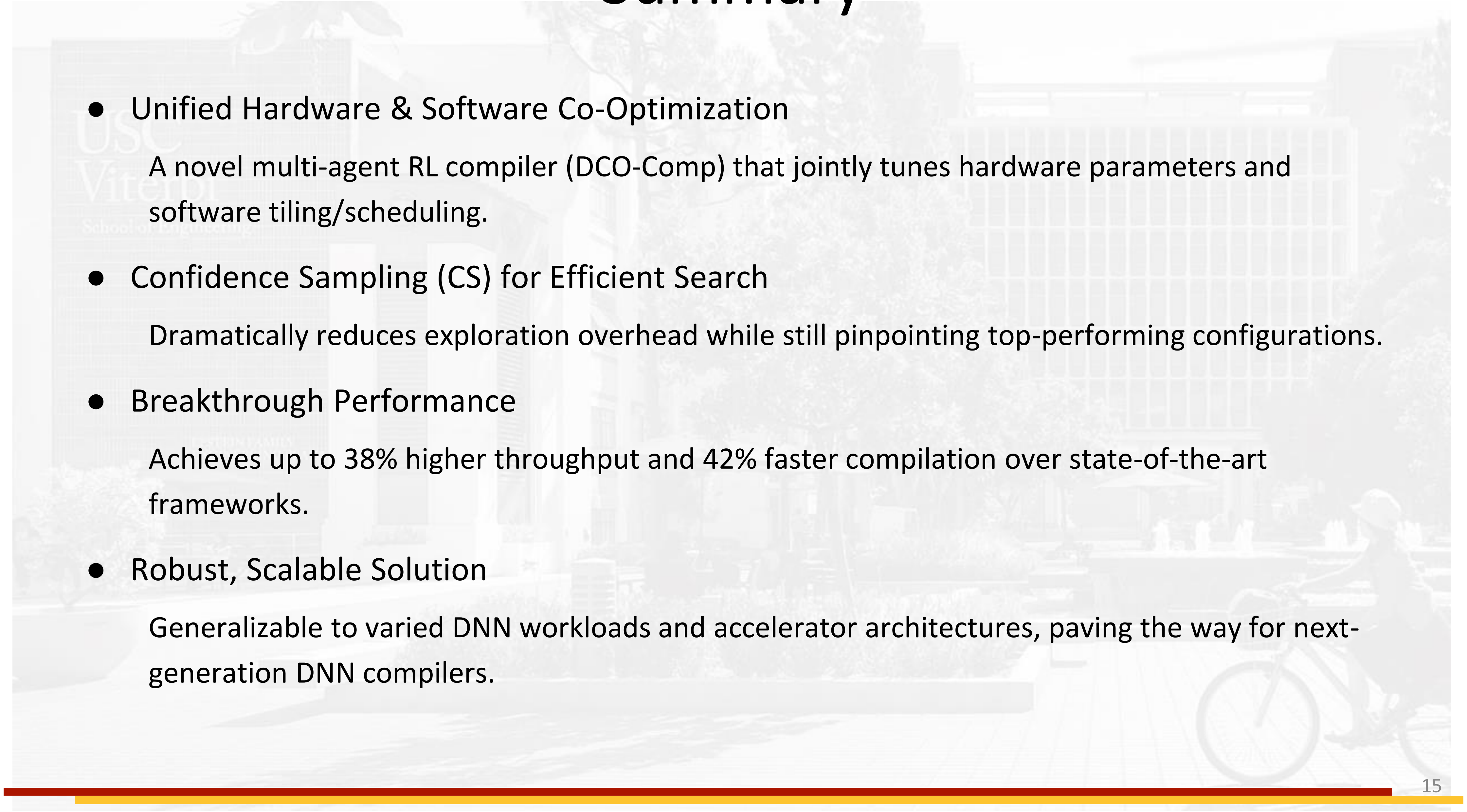
Performance vs. #Trials (GFLOPS)

- DCO-Comp converges to high GFLOPS faster (455 vs 800 Trials for AutoTVM and CHAMELEON)
- Fewer hardware measurements needed to reach near-peak performance



Comparing the compiled code performance (GFLOPS per Trial) of different frameworks for ResNet-18 model.

Summary



- Unified Hardware & Software Co-Optimization

  A novel multi-agent RL compiler (DCO-Comp) that jointly tunes hardware parameters and software tiling/scheduling.

- Confidence Sampling (CS) for Efficient Search

  Dramatically reduces exploration overhead while still pinpointing top-performing configurations.

- Breakthrough Performance

  Achieves up to 38% higher throughput and 42% faster compilation over state-of-the-art frameworks.

- Robust, Scalable Solution

  Generalizable to varied DNN workloads and accelerator architectures, paving the way for next-generation DNN compilers.

THANK YOU

Questions

For codes and additional questions, please reach out to

afayyazi@usc.edu