Thesis for The Degree of Doctrate of Philosophy

Adaptive Scheduling of Inference Pipelines on Multicore Architectures

Pirah Noor Soomro

Division of Computer Science Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2025

Adaptive Scheduling of Inference Pipelines on Multicore Architectures

Pirah Noor Soomro

Thesis supervisor:
Professor Miquel Pericàs, Chalmers University of Technology, Sweden

Thesis co-supervisors:
Dr. Nikela Papadopoulou, University of Glasgow, Scotland

Examiner & Chairman:
Professor Per Stenström, Chalmers University of Technology, Sweden

Opponent:
Research Scientist Lionel Eyraud-Dubois, Inria Centre at the University of Bordeaux, France

Grading Committee:
Established Researcher Vincenç Beltran, Barcelona Supercomputing Center, Spain
Associate Professor Jose Cano Reyes, University of Glasgow, Scotland
Associate Professor Salman Toor, Uppsala University, Sweden

Deputy Committee:
Professor Vincenzo Massimiliano Gulisano, Chalmers University of Technology, Sweden

ISBN 978-91-8103-261-1
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5719.
ISSN 0346-718X Department of Computer Science & Engineering

Division of Computer Science Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2025.

"Calmness is a sign of Intelligence"
- Imam Ali

Abstract

In today's data-driven world, machine learning (ML) algorithms, particularly Convolutional Neural Networks (CNNs), play a pivotal role in powering a myriad of applications across various domains. As the demand for realtime inference continues to escalate, optimizing CNN inference across diverse computational platforms becomes imperative. This thesis addresses this challenge by exploring the complexities posed by heterogeneous edge devices, chiplet-based architectures, and inference serving systems. Heterogeneous edge devices present unique challenges due to resource constraints and architectural diversity, while chiplet based architectures offer potential enhancements in inference performance. Leveraging innovative techniques such as online tuning algorithms, malleable and moldable inference pipelines, and adaptive scheduling strategies, our thesis proposes a comprehensive framework for optimizing DNN inference. This framework aims to advance system performance, reduce latency, and mitigate interference effects, thereby contributing to the development of more efficient and scalable AI systems capable of meeting the evolving demands of realtime inference across diverse computational platforms. The thesis addresses several key problem statements, including enabling runtime scheduling of inference pipelines on edge devices, fully online scheduling of inference pipelines on heterogeneous platforms, mitigating interference effects on inference pipelines in inference serving systems, and optimizing resource allocation in inference serving systems for adaptive SLO aware inference serving.

The contributions of this thesis are encapsulated in four papers, each focusing on distinct aspects of CNN inference optimization. These contributions include the development of comprehensive frameworks for online scheduling of CNN pipelines, leveraging platform knowledge for expedited seed generation, dynamic scheduling techniques to alleviate interference effects, and SLO aware scheduling techniques for optimizing resource allocation in inference serving systems. Through these contributions, this thesis seeks to advance the state of the art in CNN inference optimization and inference serving systems, paving the way for more efficient and scalable AI systems capable of meeting the demands of realtime inference across diverse computational platforms.

Keywords

CNN parallel pipelines, Online tuning, Design space exploration, Heterogeneous computing units, Processing on chiplets, Inference Serving Systems, Interference Mitigation

Acknowledgment

Truly, every milestone is achieved by the will of God. This journey has been lengthy and arduous, yet it has imparted invaluable lessons. I extend my heartfelt gratitude to my supervisor, Professor Miquel Pericàs, whose unwavering guidance has been instrumental. His profound expertise in research and astute insights made this journey significantly smoother. I am also indebted to my co-supervisors, Dr. Mustafa Abduljabbar and Nikela Papadopoulou, whose sharp observation and problem-solving acumen have been invaluable in advancing my research. I express sincere appreciation to Prof. Jeronimo Castrillon from Technische Universität Dresden for graciously hosting me during my research visit to his group and offering invaluable feedback on my work. My gratitude extends to my research group: Dr. Madhavan Manivannan, Bhavishiya Goel, Jing, Sonia, Hari, Minyu, Nufail, and Mohammad Eljamali, whose unwavering support, insightful discussions, and assistance have been invaluable throughout. Lastly, I am deeply grateful to my family for their constant support; to my Ami and Papa for their endless prayers and encouragement, even without fully understanding the technical details of my work, and to my siblings, Iqra, Kashi, and Hassan, for their continuous support and belief in me. I dedicate this thesis to my beloved daughter, Zimal Fatima Ali, whose presence brings joy and meaning to my life. A special acknowledgment goes to my husband, Ali Raza, for his steady support and encouragement throughout this journey.

This research is partly funded by the European Union Horizon 2020 research and innovation programme under LEGaTO [1] with grant agreement No.780681, PRIDE: Principles for Computing Memory Devices [2] with grant agreement No. CHI 19-0048 funded by Swedish Foundation for Strategic Research and Eurolab4HPC [3] with grant agreement No.800962. The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by Swedish Research Council [4] grant agreement No.2018-05973.

[1] https://legato-project.eu/
[2] https://www.pride-project.se/
[3] https://www.eurolab4hpc.eu/
[4] https://www.vr.se/

List of Publications

Appended publications

This thesis is based on the following publications:

I: Pirah Noor Soomro, Mustafa Abduljabbar, Jeronimo Castrillon, and Miquel Pericàs. "An online guided tuning approach to run cnn pipelines on edge devices"
Published in Proceedings of the 18th ACM International Conference on Computing Frontiers, pp. 45-53. 2021.

II: Pirah Noor Soomro, Mustafa Abduljabbar, Jeronimo Castrillon, and Miquel Pericàs. "Shisha: Online scheduling of CNN pipelines on heterogeneous architectures"
Published in Proceedings of International Conference on Parallel Processing and Applied Mathematics 2022.

III: Pirah Noor Soomro, Nikela Papadopoulou and Miquel Pericàs. "ODIN: Overcoming Dynamic Interference in iNference pipelines"
Published in Proceedings of the European Conference on Parallel Processing 2023.

IV: Pirah Noor Soomro, Nikela Papadopoulou and Miquel Pericàs. "Accordion: A malleable pipeline scheduling approach for adaptive SLO-aware inference serving"
Published in Proceedings of 22nd International Conference on Computing Frontiers 2025.

Other publications

I: Jing Chen, Pirah Noor Soomro, Mustafa Abduljabbar, Madhavan Manivannan, and Miquel Pericàs "Scheduling Task-parallel Applications in Dynamically Asymmetric Environments"
Published in 49th International Conference on Parallel Processing-ICPP: Workshops. 2020

Contents

Chapter 1

Introduction

In today's data driven world, the importance of machine learning (ML) cannot be overstated. ML algorithms power a plethora of applications, from image recognition and natural language processing to recommendation systems and predictive analytics. Central to many ML tasks is the concept of inference, where trained models make predictions or decisions based on input data. Among the various architectures used for inference, Convolutional Neural Networks (CNNs) have emerged as a cornerstone, demonstrating remarkable performance in tasks such as image classification, object detection, and semantic segmentation.

As the demand for real time inference continues to rise, there is an urgent need to optimize CNN inference across diverse computational platforms. This thesis explores this scenario comprehensively, delving into the challenges and opportunities presented by heterogeneous edge devices, chiplet based architectures, and inference serving systems. Heterogeneous edge devices pose unique challenges due to resource constraints and architectural diversity. Conversely, chiplet based architectures offer potential enhancements in inference performance by integrating diverse computational elements, albeit with their own set of challenges.

Furthermore, inference serving systems play a critical role in providing predictions as a service, yet face hurdles such as interference from co-located workloads, which can impact performance and violate service level objectives (SLOs). Dynamic adaptation strategies are essential to optimize resource allocation and ensure consistent performance in such systems. Additionally, scheduling multiple CNN inference pipelines on inference serving systems introduces complexities in managing throughput and latency.

By synthesizing insights from these diverse domains, our thesis aims to propose a comprehensive framework for optimizing CNN inference. Leveraging innovative techniques such as online tuning algorithms, malleable and moldable inference pipelines, and adaptive scheduling strategies, our approach seeks to advance system performance, reduce latency, and mitigate interference effects. Ultimately, our research endeavors to contribute to the development of more efficient and scalable AI systems, capable of meeting the evolving demands of real time inference across diverse computational platforms.

1.1 Background

This section sets the stage for discussing CNNs, outlining their structure and operations, including parallel implementations and inference pipelines. It also highlights the challenges of deploying CNNs on diverse computing platforms and introduces inference serving systems. Subsequent sections will delve deeper into these topics, exploring optimization strategies and addressing quality-of-service considerations in inference delivery.

1.1.1 Convolutional Neural Networks

The forward pass of CNNs primarily involves convolutional and fully-connected layers. Convolutional layers constitute the most computationally intensive aspect of CNNs. These layers consist of a set of learned filters (weights) convolved across the height, width, and depth of the input tensor. The core operation within convolutional layers is the dot product between the weight tensor and local input regions, akin to matrix multiplication. The computational complexity of convolutional layers is represented by Equation 2.1, where $[H, W, C]$ denotes the dimensions of the input tensor, and $[R, S, K]$ represents the dimensions of the convolutional kernel.

$$W_C = H \times W \times C \times R \times S \times K \tag{1.1}$$

Fully connected layers, occurring towards the end of CNN architectures, feature dense connectivity, with each neuron connected to all activations of the preceding layer. This dense connectivity results in a substantial number of parameters and intensive computations, as illustrated by Equation 1.2, where F denotes the number of output categories.

$$W_{fc} = H \times W \times C \times F \tag{1.2}$$

Pooling layers, positioned between convolutional layers, serve to downsample the spatial dimensions of the input tensor. With no learned parameters, these layers involve simple computations, utilizing the input tensor dimensions as computational weights.

1.1.2 Parallel Implementations of CNNs

Various parallelization schemes are employed to accelerate CNN computations. Data parallelism involves partitioning the work of a minibatch among multiple computational resources, while model parallelism divides the work based on neurons in each layer. Layer pipelining, as described by Ben-Nun et al. [1], partitions work by distributing network layers among computational resources, combining model parallelism within the layer with overall layer pipelining. This hybrid parallelism offers benefits such as reduced communication volume and cached weights, minimizing memory round trips.

1.1.3 Inference Pipelines

The computations in CNNs are structured as layers, where the output of one layer feeds into the next. This arrangement forms a linear task chain

or dependency, defining the task graph of a CNN. CNN inference processes streaming input data on this persistent task graph, which can be divided into sub DAGs representing pipeline stages, where a single stage can contain multiple CNN layers and a single stage (sub DAG) is assigned to a separate execution resource. Optimal pipeline efficiency is achieved when the execution time of all stages is balanced, minimizing end-to-end latency and reducing the size of the latency gap, commonly referred to as the "bubble" [2]. The performance of the pipeline is primarily determined by the slowest stage, or bottleneck, making it essential to distribute layers in a pipeline to minimize bottleneck latency.

1.1.4 Diversity in computing platforms

Modern edge devices feature powerful and energy efficient compute resources, enabling on-device execution of CNNs. This enhances realtime performance and mitigates communication delays stemming from network issues [3]. While prevalent DNN frameworks like Tensorflow [4], Caffe [5], Torch [6], and Theano [7] excel on computing platforms with discrete GPUs and highperformance CPU clusters, they do not directly address the resource constraints inherent to edge devices, such as power, memory, and compute capability [8]. As a result, inference performance on CPUs is often comparable to GPUs in edge devices (embedded devices), leading many vendors to favor CPUs for inference tasks [9]. However, this transition to edge devices introduces a fresh set of challenges, chiefly stemming from the diverse architectures of System on Chips (SoCs) [9, 10]. Modern edge devices frequently integrate heterogeneous execution units on the same chip, comprising cores with varying power-performance-area characteristics but sharing a common Instruction Set Architecture (ISA) [11]. For instance, an edge device, the NVIDIA Jetson TX2 [12] houses a dualcore NVIDIA Denver 2 64-bit CPU alongside a quadcore ARM A57 cluster, illustrating this heterogeneity. Similarly, advancements like Intel's Meteor Lake [13] and Apple's A14 Bionic [14] feature asymmetric multicore designs, combining highperformance and power saving cores. This trend towards heterogeneity is further augmented by Multi-Chip-Module (MCM) integration, employing technologies like interposer based packaging to enhance latency and bandwidth [15, 16]. As chip manufacturers blend these technologies to design highperformance processors, heterogeneity emerges not only at the core level but also within the memory subsystem and Network on Chip (NoC). To effectively harness such architectures, applications must be optimized considering varying levels of heterogeneity. Additionally, to accommodate the diversity of hardware platforms, the optimization process should ideally be rapid and preferably conducted online.

1.1.5 Inference Serving Systems

Machine learning's widespread adoption necessitates efficient prediction delivery, giving rise to inference serving systems [17–21]. These systems host pre-trained model pipelines, or inference pipelines, on cloud infrastructure, catering to inference queries from users and applications. Such systems often operate under stringent quality-of-service (QoS) requirements, expressed as service level

objectives (SLOs), for query response times and throughput [22]. However, due to resource constraints and high demand, inference pipelines frequently share resources with other workloads. This co-location, whether within the inference serving system itself or as part of broader multi-tenancy practices in cloud environments [23, 24], can introduce interference that jeopardizes inference performance, potentially leading to SLO violations.

1.2 Problem Statements

1.2.1 Enabling runtime scheduling of inference pipelines on edge devices

Frameworks leveraging pipeline parallelism, such as PipeIt [25] for heterogeneous core clusters and graphi [26] for many-core platforms, employ offline analytical performance models to construct efficient pipeline stages. However, these models face limitations as platforms become more complex and heterogeneous. One limitation stems from the reliance on prediction models that utilize workload characteristics and profiled execution times of representative kernels, as seen in PipeIt and AUGUR [27]. These models often overlook realtime performance degrading factors like resource contention, potentially leading to sub-optimal configurations and performance loss. Furthermore, as platforms evolve with increased heterogeneity and hierarchy, the shortcomings of analytical models are expected to exacerbate. Another challenge arises from the need to repeat performance sampling and throughput maximization whenever platform configurations change. This necessitates additional efforts and resources. Addressing these challenges requires an online approach that relies on realtime performance measurements. However, the complex design space poses a significant obstacle. To date, there is no online solution capable of effectively navigating this design space, quickly converging to near-optimal solutions, and adapting to the performance asymmetry present at runtime.

Question: *How to develop an online solution that effectively prunes the design space, rapidly converges to near-optimal solutions, and adapts to performance asymmetry in heterogeneous platforms?*

1.2.2 Fully online scheduling of inference pipelines on heterogeneous platforms

Current approaches for partitioning and scheduling CNN pipelines rely on sophisticated cost models and exploration algorithms to navigate the design space effectively [28, 29]. However, these approaches face several challenges, including scalability, sensitivity to environmental changes, and a lack of consideration for heterogeneous architectures. While sophisticated cost models have been proposed, they often require extensive training and are sensitive to changes in the execution environment and architectural parameters. Moreover, they do not adequately account for the increasing heterogeneity in future computing platforms, including diverse core performance and memory bandwidth characteristics. As heterogeneity continues to rise in future platforms, static pipeline partitioning and scheduling become increasingly inflexible. To address this, a fully online and scalable approach to CNN pipeline scheduling is needed.

Such an approach should not only consider core heterogeneity but also address heterogeneity in memory bandwidth. Additionally, it should minimize overhead to ensure practicality in real world applications.

Question: *How to design a fully online and scalable CNN pipeline scheduling approach that targets both core heterogeneity and heterogeneity in memory bandwidth, while minimizing overhead for practical deployment?*
Compared to Question 1, here we focused more on designing a solution that doesn't look into performance database to find a solution, instead it should rely on runtime performance data. We further explored possibilities to develop an algorithm which should consider memory bandwidth heterogeneity while finding a balanced pipeline configuration.

1.2.3 Mitigating Interference effects on inference pipelines in inference serving systems

In inference serving systems, inference pipelines often face performance degradation due to interference from co-located workloads, leading to violations of SLOs [23,24,30,31]. Mitigating this interference is crucial for ensuring the reliability and efficiency of critical applications. Various scheduling techniques have been proposed to address interference effects on critical workloads, including inference pipelines [21,31–34]. These techniques often rely on offline profiling and machine learning models to partition resources effectively. However, they may lack adaptability to dynamic changes in workload characteristics and interference patterns. Pipeline parallelism, particularly in the form of layer pipelining, offers a promising solution to improve throughput and reduce latency for inference pipelines [35–38]. Online scheduling techniques have emerged to find near-optimal pipeline schedules, leveraging heuristics to navigate the vast search space efficiently [39–42]. By rebalancing pipeline stages dynamically, these techniques can adapt to changes in workload interference, optimizing resource utilization in realtime.

Question: *How to develop dynamic scheduling techniques for inference pipelines that effectively mitigate interference effects from co-located workloads in inference serving systems, ensuring high throughput and low latency while dynamically adapting to changing workload conditions?*

1.2.4 Inter-Pipeline scheduling for adaptive SLO-aware inference serving

Inference serving systems play a crucial role in providing machine learning services to businesses, ensuring quick and accurate inference responses while minimizing costs. These systems operate under hard cost constraints, often utilizing specialized ML hardware to achieve interactive latencies [43]. However, under bursty workloads, static resource allocation solutions can lead to increased costs due to the need for additional resources. To address these challenges, modern inference serving systems implement various techniques, such as multi-tenancy, adaptive batching, model selection, and accuracy scaling, to proactively assign resources and maintain high resource utilization while meeting Service Level Objectives (SLOs) [17,44–47]. Despite these efforts, factors like variable query patterns and changes in the execution environment can lead to SLO

violations, resulting in resource overprovisioning or underutilization. Existing approaches may revise scheduling decisions or drop/defer queries to mitigate violations, adding operational costs and complexity.

Question: *How can resource allocation in inference serving systems be further optimized by dynamically readjusting resources per inference query, while ensuring SLO requirements are met and costs are minimized?*
In this work we looked into not just one but scheduling and balancing multiple pipelines to accommodate SLO requirements.

1.3 Contributions

This thesis builds upon four papers, aiming to automate the scheduling of CNN pipelines on heterogeneous platforms and inference serving systems. The first paper represents the inaugural attempt at developing an online search scheme for CNNs. The main contributions of Paper I are:

- Development of a comprehensive framework for generating and online scheduling of CNN pipelines. We devised a tuning algorithm that utilizes task moldability and online performance measurements to discover a near optimal schedule for maximizing pipeline throughput. This scheme adapts to performance variations between core clusters.

- We introduced a tensor template language interface to describe CNN descriptors, facilitating the formulation of initial schedules (referred to as seeds) for online design space exploration.

In Paper II, our focus shifts to chiplet based heterogeneous architectures. We present an approach that leverages readily available information about the computing platform and CNN structure without requiring human intervention. The main contributions of Paper II are:

- Introduction of an expedited method for seed generation by leveraging platform knowledge. This approach is compared to a range of representative exploration algorithms, including the scheme proposed in Paper I.

- Elimination of the requirement for generation and preprocessing of the design space before the online phase, addressing a limitation observed in Paper I, particularly when larger platforms and deeper CNNs are used as use cases.

- Demonstration of the scalability of the scheme proposed in Paper II with CNNs containing over 50 compute intensive layers.

In Paper III, we investigated the ramifications of co-locating inference pipelines within inference serving systems. Our primary emphasis was on addressing interference, a significant challenge, by devising dynamic scheduling techniques to alleviate its effects. The principal contributions of Paper III are as follows:

- An online solution - ODIN, that dynamically detects interference and adjusts the execution of inference pipelines on a given set of processing elements.
- Unlike other approaches, ODIN does not rely on offline resource utilization profiles for inference. Instead, it utilizes only the runtime observed execution times of pipeline stages, making it easily applicable to any system.
- We propose a heuristic based pipeline scheduling algorithm that optimizes the pipeline's overall throughput while minimizing the impact of interference on the execution unit. This algorithm considers both the execution times of pipeline stages and the extent of performance degradation due to interference.

In paper IV we looked at scheduling techniques for multiple inference pipelines co-located on inference serving system with a goal to combat response delays and reduce cost due o additional resources during peak time with bursty workload. The principal contributions of Paper IV are as follows:

- The proposed solution optimizes resource allocation within an inference serving system by leveraging SLO-aware scheduling techniques. This minimizes the need for additional resources per inference query, effectively mitigating costs.
- We leverage malleable resource allocation to models through malleable inference pipelines, allowing for flexible assignment of resources during peak loads. Our adaptive scheduling algorithm ensures that, in the event of resource scarcity, existing pipelines can be downscaled without violating SLO requirements, thereby maintaining service quality.

1.4 Thesis Outline

The remainder of this thesis follows a structured layout. Chapter 2 offers summaries of each paper. Chapter 3 contains concluding remarks and suggestions for future research directions. In addition, the four papers are appended at the conclusion of this thesis.

Chapter 2

Summary of the papers

2.1 Summary of Paper I

In recent years, convolutional neural networks (CNNs) have emerged as powerful tools in various fields such as image classification [48, 49] and natural language processing [50]. While training CNNs typically occurs in the cloud, inference, which involves a single forward pass of a neural network, is increasingly being performed on edge and mobile devices due to latency concerns [51]. Edge devices are becoming more powerful, allowing for realtime execution of CNN applications and reducing reliance on cloud based inference [3]. However, optimizing CNN inference for edge devices presents challenges due to resource constraints and the heterogeneous nature of these platforms [8]. Existing deep neural network (DNN) frameworks are primarily optimized for server-side computing platforms with discrete GPUs and high performance CPU clusters [4–7]. However, the parallelization strategy commonly employed in these frameworks, layer wise data parallelism, may not be optimal for heterogeneous compute devices [9, 10]. An alternative approach, model parallelism, which groups consecutive CNN layers into pipeline stages, has shown promise in addressing these challenges [52]. However, efficiently exploring the complex design space of model parallelism remains a significant challenge [25, 26]. This paper proposes an online tuning algorithm for optimizing pipeline parallelism in CNNs to maximize throughput on heterogeneous platforms. The algorithm leverages evolutionary search techniques and computational hints derived from network layer descriptors to efficiently navigate the design space and find near-optimal configurations. The contributions of this work include the development of a multi-layer solution integrating a tensor template language for CNN descriptions and the XiTAO runtime [53] for moldable task execution.

2.1.1 Proposed Approach

The proposed approach consists of two primary components: pre-processing to generate seed heuristics and online tuning followed by pipelining. Figure 2.1 provides an overview of both modules. The CNN network is described using a tensor template language, and computational hints are derived from network layer descriptors. The online tuning phase employs the Pipe-search algorithm,

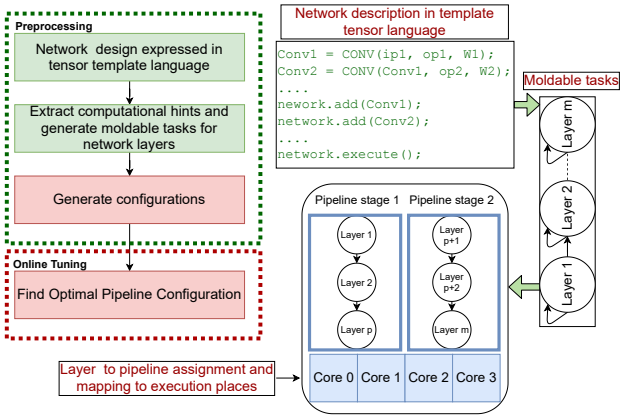

Figure 2.1: An overview of framework for generation and schedule exploration for moldable CNN pipeline

which utilizes evolutionary search techniques guided by computational hints and realtime performance profiling. Moldable tasks are used to represent CNN layers, enabling dynamic mapping to resources during execution [53]. The pre-processing step involves generating seed heuristics based on network layer descriptors and computational hints. These seeds serve as initial configurations for the online tuning phase. During online tuning, the Pipe-search algorithm explores the design space to identify near-optimal configurations for maximizing throughput. realtime performance profiling enables the algorithm to adapt to runtime conditions and platform heterogeneity [54].

2.1.2 Evaluation

The evaluation of the proposed approach involves comparing the Pipe-search algorithm with exhaustive search and random walk. Experimental results demonstrate that Pipe-search significantly reduces convergence time while achieving near-optimal throughput maximizing pipeline configurations. Specifically, Pipe-search converges 70X faster than exhaustive search and 11X faster than random walk. Moreover, the algorithm finds configurations that yield balanced pipelines for state-of-the-art CNNs, with 79% of configurations tested considered good candidates for near-optimal cases.

To assess the effectiveness of the algorithm, experiments are conducted using various CNN architectures on heterogeneous platforms. The results show that Pipe-search outperforms exhaustive search and random walk in terms of convergence time and solution quality. Moreover, the algorithm is capable of adapting to different platform configurations and optimizing pipeline parallelism for maximum throughput.

2.1.3 Conclusion

In conclusion, this paper presents an online tuning algorithm for optimizing pipeline parallelism in CNNs on heterogeneous platforms. By leveraging evolutionary search techniques and computational hints, the proposed approach

efficiently explores the design space to find near-optimal configurations for maximizing throughput. Experimental results demonstrate the effectiveness and adaptability of the algorithm across various platform configurations. Overall, this work contributes to the development of efficient CNN inference techniques for edge and mobile devices.

2.2 Summary of Paper II

Multicore processors are becoming increasingly heterogeneous, with designs featuring a mix of high performance and power saving cores, as seen in Intel's Meteor Lake [13] and Apple's A14 Bionic [14]. This trend extends to Multi-Chip-Module (MCM) integration, enabling lower design costs and improved yield by reducing chip area, particularly when combined with interposer based packaging for lower latency and high bandwidth transmission to memory devices such as High Bandwidth Memory (HBM) [16]. The resulting heterogeneity, spanning cores, memory subsystems, and Network on Chip (NoC), presents challenges and opportunities for application optimization, particularly for CNNs, which have high computational, bandwidth, and memory capacity requirements. Parallel pipelining, partitioning networks across devices and requiring only input exchange among stages, offers a promising solution. In chiplet architectures, CNNs can be efficiently pipelined by distributing layers across chiplets, reducing weight transfer, and enabling manageable load balancing among heterogeneous computing units. However, effective partitioning and scheduling of pipelines necessitate sophisticated cost models, often requiring extensive training and lacking adaptability to heterogeneous architectures. As future HPC platforms are expected to exhibit increased heterogeneity, static partitioning and scheduling become less flexible. Online autotuning of pipeline schedules becomes essential for performance portability, provided it can achieve acceptable configurations with low overhead. Existing approaches like PipeSearch [40] generate databases of pipeline configurations, which are space intensive and slow for larger systems and deeper CNNs. This paper proposes Shisha, leveraging static information from CNNs and target platforms to reduce exploration points and find near optimal solutions efficiently.

2.2.1 Proposed Approach

The system comprises nodes with high performance cores (Fast Execution Places - FEP) connected to high bandwidth memory and nodes of slower cores (Slow Execution Places - SEP) connected to low bandwidth memory (Figure 2.2). Our goal is to run throughput maximizing CNN inference pipelines on such architectures. Pipeline configurations involve assigning CNN layers to stages and mapping stages to Execution Places (EPs). Shisha operates in two phases: seed generation and online tuning. In the seed generation phase, a meaningful starting configuration is determined using static information about the CNN and platform heterogeneity. This configuration balances workload among stages while considering EP assignments. The seed generation phase calculates layer weights using Equation 2.1 and ranks EPs based on performance. Layers are grouped into pipeline stages, aiming for balanced weights across stages.

$$W = H \times W \times C \times R \times S \times K \tag{2.1}$$

Online tuning refines the seed configuration for faster convergence. It starts from the seed and gradually adjusts load distribution by moving layers between adjacent stages. This guided exploration aims to avoid slow configurations and improve overall throughput.

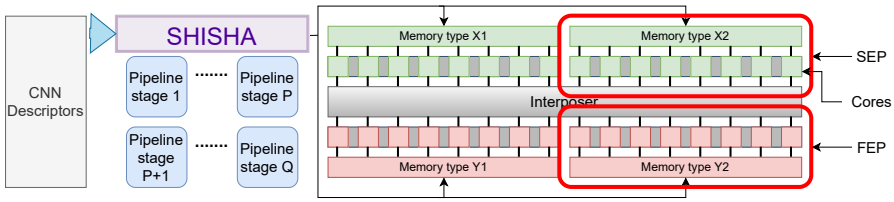

Figure 2.2: System overview in Paper II

2.2.2 Evaluation

In our evaluation, Shisha emerges as a potent solution, offering superior performance compared to existing methods. The seed generation phase is pivotal, as it sets the initial configuration for subsequent tuning. We find that seeds generated by Shisha significantly impact both convergence time and solution quality. Notably, solutions initiated with Shisha seeds consistently achieve better throughput and faster convergence compared to those with randomly generated seeds. The online tuning phase of Shisha further enhances its efficacy. Through incremental adjustments to the pipeline configuration, Shisha effectively balances workload distribution among pipeline stages, leading to improved overall throughput. The selection of assignment and balancing schemes plays a crucial role in this phase. Heuristics such as workload balancing to the lightest Fast Execution Place (FEP) contribute significantly to faster convergence and the attainment of near-optimal solutions. In comparing Shisha with other exploration algorithms, including Pipe-Search, Hill Climbing, Simulated Annealing, and Random Walk, Shisha demonstrates superior exploration efficiency and convergence speed. Remarkably, Shisha explores only a fraction of the design space (˜0.12%) yet achieves convergence approximately 35 times faster than alternative methods. Even when compared to Exhaustive Search, Shisha matches the best solutions found for complex CNN architectures like ResNet50 and YOLOv3. Moreover, we conduct a sensitivity analysis of the parameter α, which governs the extent of exploration in Shisha. Our findings reveal that higher values of α lead to better solutions, particularly in scenarios with greater performance heterogeneity between EPs. This underscores the adaptability and effectiveness of Shisha in optimizing CNN inference pipelines across diverse computing platforms.

2.2.3 Conclusion

In this paper, we introduce a rapid method for scheduling CNN pipelines on heterogeneous computing platforms comprising both fast and slow cores. Our approach is versatile, applicable to various hardware architectures including GPUs, FPGAs, asymmetric multicores, and chiplets. By leveraging compile time information alongside a succinct and directed online search, we effectively autotune CNN layers into parallel pipelines. The experimental evaluation validates the efficacy of our method. We demonstrate that Sisha's solutions are comparable to those obtained through exhaustive search of the design space. Furthermore, our results highlight Shisha's scalability with larger networks and

computing platforms, showcasing its potential for realworld applications. In conclusion, Shisha offers a promising solution for optimizing CNN inference pipelines across heterogeneous hardware architectures. Its ability to swiftly converge to high-quality solutions, coupled with its scalability, underscores its value in accelerating neural network inference tasks.

2.3 Summary of Paper III

In today's digital landscape, the demand for machine learning based predictions has skyrocketed, leading to the proliferation of inference serving systems. These systems, such as Clipper [17], Pretzel [18], and TensorFlow Serving [19], deploy pre-trained model pipelines on cloud infrastructure to serve inference queries to users and applications. However, as these systems often operate in resource-constrained environments with high demand, they frequently co-locate inference pipelines with other workloads, either within the same inference serving system or as part of multi-tenant cloud environments. This co-location can lead to interference, which adversely affects the performance of inference pipelines and results in violations of service level objectives (SLOs) [22].

Various scheduling techniques have been proposed to mitigate the impact of interference on critical applications, including inference pipelines. Traditional approaches focus on offline profiling and predictive modeling to preemptively partition resources and mitigate interference [23, 24, 32]. However, these techniques often require extensive offline profiling and are less adaptable to dynamic workload changes. Recent advancements have led to the development of online interference mitigation techniques tailored specifically for inference pipelines [21, 31, 34]. These techniques dynamically adapt the execution of inference pipelines in response to realtime interference, thereby improving their resilience to co-located workloads.

In this paper, we propose ODIN, a dynamic solution for online interference mitigation in inference pipelines. Unlike traditional techniques that rely on offline profiling, ODIN operates entirely online, leveraging runtime observations to adapt pipeline execution dynamically. We demonstrate the effectiveness of ODIN in maintaining quality-of-service (QoS) for inference pipelines under varying interference conditions, outperforming traditional scheduling techniques such as the least loaded scheduler (LLS) [23]. Our evaluation includes extensive experiments across different interference scenarios, showcasing ODIN's superior performance in terms of latency, throughput, and SLO conformance.

2.3.1 Proposed Approach

ODIN's approach to online interference mitigation revolves around dynamically detecting interference and adaptively rebalancing pipeline stages to optimize throughput and latency. Unlike traditional techniques that rely on offline profiling and predictive models, ODIN operates entirely online, making it more adaptable to dynamic workload changes and varying interference conditions. ODIN operates within a system consisting of multiple execution places, each comprising multiple cores 2.3. It continuously monitors the execution times of pipeline stages and triggers rebalancing when interference is detected. The core of ODIN's methodology lies in its heuristic algorithm, which intelligently moves layers between pipeline stages to reduce the workload on affected stages while maximizing pipeline throughput.

At the heart of ODIN is its heuristic based approach to pipeline stage rebalancing under interference. This approach employs two key heuristics: determining the direction of moving layers and avoiding local optima. By intelligently applying these heuristics, ODIN is able to efficiently explore the

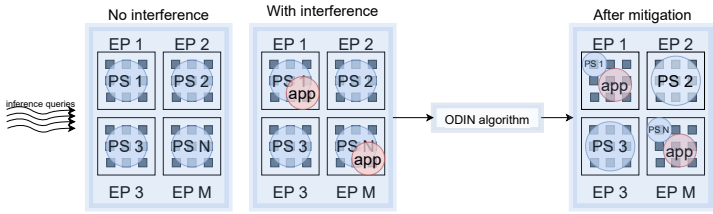

Figure 2.3: System overview

solution space and adapt pipeline execution to varying interference conditions in realtime.

2.3.2 Evaluation

Our evaluation demonstrates the significant performance improvements ODIN offers over traditional scheduling techniques in mitigating interference and enhancing the quality-of-service (QoS) of inference pipelines. We compare ODIN's latency and throughput with those of the least loaded scheduler (LLS) across various interference scenarios. In a system with 4 execution places of 8 cores each, serving inference queries with VGG16 and ResNet50 models, ODIN consistently outperforms LLS. On average, ODIN achieves 15.8% better latency with $\alpha = 10$ and 14.1% with $\alpha = 2$. Similarly, ODIN achieves 19% higher throughput than LLS across all scenarios. Notably, ODIN's adaptive nature allows it to sustain high throughput and low latency even under challenging interference conditions.

We evaluate ODIN's ability to maintain quality-of-service (QoS) by profiling the number of queries violating service level objectives (SLOs) for different throughput targets. ODIN demonstrates superior performance, with less than 20% of SLO violations for SLO levels lower than 85%. In contrast, LLS exhibits higher SLO violations, especially for stringent SLO targets. Moreover, ODIN sustains 70% of the original throughput under interference, outperforming LLS, which achieves only 50% SLO conformance. These results highlight ODIN's effectiveness in ensuring QoS even in the presence of interference.

We analyze ODIN's scalability on high numbers of execution places using ResNet152 models. Scaling from 4 to 52 execution places, ODIN maintains consistent latency and increases throughput, indicating high parallelism and efficient utilization of resources. Even with deep neural network models, ODIN adapts pipeline configurations effectively, achieving throughput comparable to peak performance under no interference.

2.3.3 Conclusion

In conclusion, we propose ODIN, a dynamic solution for online interference mitigation in inference pipelines. By adaptively rebalancing pipeline execution in response to realtime interference, ODIN enhances the performance and quality-of-service (QoS) of inference serving systems. Our evaluation demonstrates the effectiveness of ODIN in mitigating interference, outperforming

traditional scheduling techniques, and maintaining high throughput and low latency under varying interference conditions.

2.4 Summary of Paper IV

Machine learning (ML) inference applications are increasingly pervasive in various domains, including social media, e-commerce, and healthcare [55–59]. Meta (formerly Facebook) has reported serving trillions of inference requests daily, with over 90% of production AI resources dedicated to inference tasks [60, 61]. As businesses increasingly rely on ML services for their products, the demand for quick and cost effective inference responses is escalating. Inference serving systems play a critical role in meeting these demands, providing the infrastructure to execute inference queries based on predefined Service Level Objectives (SLOs) that encompass model accuracy, response time, and cost constraints [61]. However, traditional inference serving systems face challenges in dynamically allocating resources to adapt to fluctuating workloads while maintaining performance and cost efficiency. Techniques such as multi-tenancy, adaptive batching, model selection, and accuracy scaling have been proposed to optimize resource utilization and meet SLO requirements [17, 44–47]. Nevertheless, factors like variable query patterns and changes in the execution environment can lead to SLO violations or resource overprovisioning [62–64]. To address these challenges, we propose a novel approach, termed Accordion, which leverages malleable inference pipelines and adaptive scheduling to optimize resource allocation dynamically.

2.4.1 Proposed Approach

The Accordion framework comprises a collection of computing units known as execution places (EPs) (Figure 2.4), organized in a multi chip architecture [34]. Each EP consists of multiple cores and is capable of executing inference tasks in a data parallel manner. The system handles incoming queries categorized based on priority, such as high and low priority, to prioritize resource allocation [65]. Additionally, Accordion incorporates a predictor to estimate query processing times and dynamically adjusts resource allocations to minimize waiting times [66]. Accordion's resource allocation strategy involves categorizing incoming queries into high and low priority queues, with a focus on minimizing waiting times for high priority queries [66]. The scheduler periodically checks for new queries and allocates resources accordingly, utilizing a predictor to estimate waiting and processing times based on preprofiled performance data [65]. Queries are dynamically allocated resources based on their priority and SLO requirements, ensuring efficient resource utilization and timely query processing. A key feature of Accordion is its malleable inference pipelines, which allow resources to be dynamically adjusted for each query to meet its SLO requirements [47]. The system utilizes an adaptive scheduling strategy to fetch resources from existing pipelines without violating their SLOs, ensuring efficient resource utilization [34]. By dynamically readjusting pipeline resources per query, Accordion optimizes system throughput and minimizes SLO violations, even under fluctuating workloads.

2.4.2 Evaluation

We conducted extensive experiments to evaluate the performance of Accordion against several baseline approaches in a simulated inference serving environment.

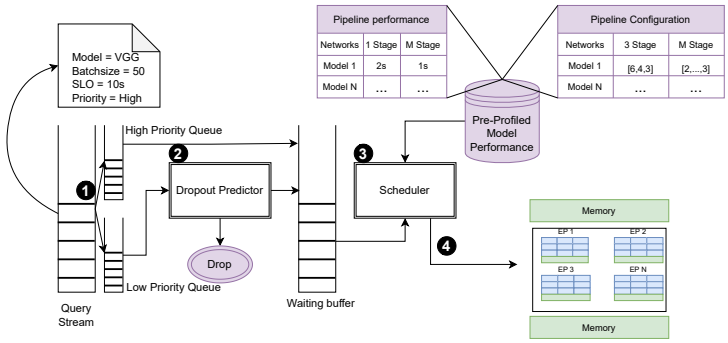

Figure 2.4: Scheduling architecture

The experiments were performed on a cluster comprising execution places (EPs), organized in a multichip architecture [34]. Each EP consists of multiple cores capable of executing inference tasks in a data parallel manner. We utilized a diverse set of query streams with varying workload patterns, including batch sizes and arrival rates, to mimic realworld scenarios [41, 62]. Additionally, we defined strict, moderate, and relaxed Service Level Objectives (SLOs) to assess system performance under different latency targets [46]. We compared Accordion against several baseline approaches, including SLO-Conforming, Fastest, Buffer Based Solution (BBS), and 1-Stage pipeline. SLO-Conforming allocates resources to queries based on their SLO requirements, while Fastest assigns maximum resources for the fastest execution [34, 64]. BBS buffers incoming queries to match them with ongoing ones, while 1-Stage assigns one EP per query [66]. Accordion demonstrated superior performance in minimizing end-to-end latency compared to the baselines. On average, Accordion achieved a 1.6x reduction in latency compared to SLO-Conforming and Fastest, and a 2.2x reduction compared to the 1-Stage pipeline. The adaptive resource allocation strategy of Accordion effectively optimized query processing times under varying workload conditions.

Accordion significantly reduced SLO violations across all query streams and SLO targets. Under strict SLO targets, Accordion exhibited a 1.13x reduction in violations compared to the baselines. Moreover, Accordion achieved a 4.2x reduction in violations under moderate SLO targets and maintained only 2% violations under relaxed SLO targets. These results highlight the effectiveness of Accordion in meeting stringent latency requirements. Accordion maintained high system occupancy levels, utilizing resources efficiently throughout the execution of query streams. For query streams with high workload intensity, Accordion sustained system occupancy at 100% for approximately 70% of the time, outperforming other baseline approaches. This efficient resource utilization contributed to the overall performance improvement of the system. Accordion demonstrated superior throughput compared to the baselines, processing a higher number of queries per unit time. On average, Accordion achieved a 1.6x increase in system throughput compared to SLO-Conforming and Fastest, and a 2x increase compared to the 1-Stage pipeline. This enhanced throughput is

attributed to Accordion's dynamic resource allocation and malleable pipeline approach, which optimized query processing under varying workload conditions.

2.4.3 Conclusion

Paper IV presents Accordion, a novel approach for optimizing resource allocation and query processing in inference serving systems. By leveraging malleable pipelines and adaptive scheduling strategies, Accordion effectively addresses the challenges of fluctuating workloads and stringent latency requirements. Through extensive experimentation and comparison against traditional baseline approaches, Accordion demonstrates superior performance in terms of minimizing end-to-end latency, reducing SLO violations, maximizing system occupancy, and improving overall throughput. The results underscore the potential of Accordion to enhance the efficiency and scalability of inference serving systems in realworld deployments.

Chapter 3

Conclusion and Future Directions

The culmination of this thesis represents a significant stride forward in the optimization of Convolutional Neural Network (CNN) inference across heterogeneous platforms and inference serving systems. Through the exploration and refinement of various techniques across four papers, this research contributes to the advancement of resource-efficient and responsive machine learning inference systems. The overarching scope of this thesis encompasses the development and evaluation of innovative solutions to address the challenges posed by diverse computational platforms and inference serving systems. Each paper makes distinct contributions to this endeavor:

Paper I: Introduces a comprehensive framework for online scheduling of CNN pipelines. By leveraging task moldability and online performance measurements, this paper pioneers an approach to dynamically discover near-optimal schedules, thus maximizing pipeline throughput. Additionally, the introduction of a tensor template language interface facilitates the formulation of initial schedules, laying the foundation for efficient design space exploration.

Paper II: Shifts the focus to chiplet based heterogeneous architectures and presents an expedited method for seed generation based on readily available platform knowledge. By eliminating the requirement for preprocessing of the design space, this paper demonstrates scalability with deep CNNs, thus addressing a critical limitation observed in previous approaches.

Paper III: Investigates interference effects within inference serving systems and proposes ODIN, an online solution for dynamic interference detection and adjustment of inference pipeline execution. By leveraging runtime observed execution times and a heuristic based scheduling algorithm, this paper effectively mitigates interference effects, optimizing overall pipeline throughput while minimizing performance degradation due to interference.

Paper IV: Explores scheduling techniques for multiple co-located inference pipelines within inference serving systems, focusing on SLO-aware resource allocation and adaptive scheduling. By leveraging malleable inference pipelines and adaptive scheduling algorithms, this paper optimizes resource allocation to mitigate costs during peak loads while ensuring SLO requirements are met.

The importance of Convolutional Neural Networks (CNNs) is ubiquitous,

serving as the backbone for numerous automated applications. However, an intriguing shift in processor design, exemplified by Processing In Memory (PIM) technology [67, 68], presents a new set of challenges and opportunities for optimizing existing applications. Adapting CNN pipelines to PIM architectures necessitates a thorough analysis of the performance gains achieved by executing CNN operators within the memory. Understanding the impact of this architectural shift on CNN performance is essential for maximizing efficiency. Investigating the optimal placement of large CNN parameters on PIM based chiplets is crucial for leveraging the benefits of PIM technology effectively. This research direction aims to optimize memory utilization and access patterns to enhance overall system performance. Given the critical importance of energy and power efficiency, future research should focus on developing pipeline scheduling techniques that prioritize reducing energy consumption while maximizing pipeline throughput. This entails exploring novel scheduling algorithms and optimization strategies tailored to PIM architectures.

Moreover, for more managed inference serving systems, several promising directions for future work include: Concealing model variant selection behind a high level API enables users to specify performance and cost objectives effortlessly. This research direction aims to simplify the model selection process for users while ensuring optimal utilization of available resources. Abstracting the choice of hardware behind the same high level API allows the system to dynamically select the appropriate hardware type(s) based on performance and cost Service Level Objectives (SLOs). This approach ensures efficient resource allocation and cost effective inference serving. Abstracting resource management from users involves automatically optimizing resource allocation to meet query cost and performance goals for different models under varying loads. This includes transparently sharing resources across model instances, users, and managing memory based on observed load and popularity to enhance provider resource utilization.

In summary, future research in CNN inference optimization and inference serving systems should focus on addressing the challenges posed by emerging processor architectures, as well as advancing techniques for efficient resource management and model selection. By exploring these research directions, we can further advance the development of efficient and scalable AI systems capable of meeting the demands of realtime inference across diverse computational platforms.

Bibliography

[1] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–43, 2019.

[2] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *arXiv preprint arXiv:1811.06965*, 2018.

[3] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems," 2015.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[6] "Torch," http://torch.ch, accessed: 2021-01-20.

[7] "Theano," http://deeplearning.net/software/theano/, accessed: 2021-01-20.

[8] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur *et al.*, "Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications," *arXiv preprint arXiv:1811.09886*, 2018.

[9] C.-J. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia *et al.*, "Machine learning at facebook: Understanding inference at the edge," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 331–344.

[10] C. Gao, A. Gutierrez, M. Rajan, R. G. Dreslinski, T. Mudge, and C.-J. Wu, "A study of mobile device utilization," in *2015 ieee international symposium*

on performance analysis of systems and software (ispass). IEEE, 2015, pp. 225–234.

[11] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-isa heterogeneous multi-core architectures: The potential for processor power reduction," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.* IEEE, 2003, pp. 81–92.

[12] "Nvidia jetson tx2," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, accessed: 2021-01-20.

[13] "Intel technology roadmaps and milestones," Feb 2022. [Online]. Available: https://www.intel.com/content/www/us/en/newsroom/news/intel-technology-roadmaps-milestones.html#gs.z47liy

[14] "Apple a14 bionic: Specs and benchmarks." [Online]. Available: https://nanoreview.net/en/soc/apple-a14-bionic

[15] Kannan *et al.*, "Enabling interposer-based disintegration of multi-core processors," in *2015 48th Annual IEEE/ACM MICRO*. IEEE, 2015, pp. 546–558.

[16] Cho *et al.*, "Design optimization of high bandwidth memory (hbm) interposer considering signal integrity," in *2015 IEEE EDAPS*, 2015, pp. 15–18.

[17] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system." in *NSDI*, 2017.

[18] Y. Lee, A. Scolari, B.-G. Chun, M. D. Santambrogio, M. Weimer, and M. Interlandi, "Pretzel: Opening the black box of machine learning prediction serving systems." in *OSDI*, vol. 18, 2018, pp. 611–626.

[19] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke, "Tensorflow-serving: Flexible, high-performance ml serving," *arXiv preprint arXiv:1712.06139*, 2017.

[20] E. Liberty, Z. Karnin, B. Xiang, L. Rouesnel, B. Coskun, R. Nallapati, J. Delgado, A. Sadoughi, Y. Astashonok, P. Das *et al.*, "Elastic machine learning algorithms in amazon sagemaker," in *SIGMOD*, 2020, pp. 731–737.

[21] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "Infaas: Automated model-less inference serving." in *USENIX Annual Technical Conference*, 2021, pp. 397–411.

[22] C. Zhang, M. Yu, W. Wang, and F. Yan, "Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving." in *USENIX Annual Technical Conference*, 2019, pp. 1049–1062.

[23] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 77–88, 2013.

[24] ——, "Quasar: Resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[25] S. Wang, G. Ananthanarayanan, Y. Zeng, N. Goel, A. Pathania, and T. Mitra, "High-throughput cnn inference on embedded arm big. little multi-core processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[26] L. Tang, Y. Wang, T. L. Willke, and K. Li, "Scheduling computation graphs of deep learning models on manycore cpus," *arXiv preprint arXiv:1807.09667*, 2018.

[27] Z. Lu, S. Rallapalli, K. Chan, and T. La Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1663–1671.

[28] Adams *et al.*, "Learning to optimize halide with tree search and random programs," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[29] Anderson *et al.*, "Efficient automatic scheduling of imaging and vision pipelines for the gpu," *Proceedings of the ACM on Programming Languages*, vol. 5, no. OOPSLA, 2021.

[30] G. Yeung, D. Borowiec, R. Yang, A. Friday, R. Harper, and P. Garraghan, "Horus: An interference-aware resource manager for deep learning systems," in *ICA3PP*. Springer, 2020, pp. 492–508.

[31] D. Mendoza, F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "Interference-aware scheduling for inference serving," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, 2021, pp. 80–88.

[32] Q. Chen, H. Yang, J. Mars, and L. Tang, "Baymax: Qos awareness and increased utilization for non-preemptive accelerators in warehouse scale computers," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 681–696, 2016.

[33] Q. Chen, H. Yang, M. Guo, R. S. Kannan, J. Mars, and L. Tang, "Prophet: Precise qos prediction on non-preemptive accelerators to improve utilization in warehouse-scale computers," in *Proceedings of the 22nd ASPLOS'17*, pp. 17–32.

[34] L. Ke, U. Gupta, M. Hempsteadis, C.-J. Wu, H.-H. S. Lee, and X. Zhang, "Hercules: Heterogeneity-aware inference serving for at-scale personalized recommendation," in *HPCA*. IEEE, 2022, pp. 141–144.

[35] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: generalized pipeline parallelism for dnn training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 1–15.

[36] Huang *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, vol. 32, 2019.

[37] Wang *et al.*, "High-throughput cnn inference on embedded arm big. little multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2254–2267, 2019.

[38] D. Kang, J. Oh, J. Choi, Y. Yi, and S. Ha, "Scheduling of deep learning applications onto heterogeneous processors in an embedded device," *IEEE Access*, vol. 8, 2020.

[39] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, "Deep learning inference parallelization on heterogeneous processors with tensorrt," *IEEE Embedded Systems Letters*, vol. 14, no. 1, pp. 15–18, 2021.

[40] P. N. Soomro, M. Abduljabbar, J. Castrillon, and M. Pericàs, "An online guided tuning approach to run cnn pipelines on edge devices," in *CF'21*.

[41] ——, "Shisha: Online scheduling of cnn pipelines on heterogeneous architectures," in *PPAM 2022*.

[42] H.-Y. Chang, S. H. Mozafari, C. Chen, J. J. Clark, B. H. Meyer, and W. J. Gross, "Pipebert: High-throughput bert inference for arm big. little multi-core processors," *Journal of Signal Processing Systems*, pp. 1–18, 2022.

[43] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[44] M. LeMay, S. Li, and T. Guo, "Perseus: Characterizing performance and cost of multi-tenant serving for cnn models," in *IC2E*. IEEE, 2020, pp. 66–72.

[45] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "Infaas: A model-less and managed inference serving system," *arXiv preprint arXiv:1905.13348*, 2019.

[46] Gupta. *et al.*, "Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 982–995.

[47] S. Ahmad, H. Guan, B. D. Friedman, T. Williams, R. K. Sitaraman, and T. Woo, "Proteus: A high-throughput inference-serving system with accuracy scaling," 2024.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[50] M. M. Lopez and J. Kalita, "Deep learning applied to nlp," *arXiv preprint arXiv:1703.03091*, 2017.

[51] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 117–122.

[52] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," *arXiv preprint arXiv:1807.05358*, 2018.

[53] M. Pericàs, "Elastic places: An adaptive resource manager for scalable and portable performance," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 2, pp. 1–26, 2018.

[54] D. Skinner and W. Kramer, "Understanding the causes of performance variability in hpc workloads," in *IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium, 2005.* IEEE, 2005, pp. 137–149.

[55] Jiang. *et al.*, "Artificial intelligence in healthcare: past, present and future," *Stroke and vascular neurology*, vol. 2, no. 4, 2017.

[56] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 3–18.

[57] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 conference of the ACM special interest group on data communication*, 2018, pp. 253–266.

[58] F. Romero, M. Zhao, N. J. Yadwadkar, and C. Kozyrakis, "Llama: A heterogeneous & serverless framework for auto-tuning video analytics pipelines," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 1–17.

[59] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and {Delay-Tolerance}," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 377–392.

[60] Facebook Engineering, "Building meta's genai infrastructure," https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/, accessed on April 22, 2024.

[61] Gupta *et al.*, "The architectural implications of facebook's dnn-based personalized recommendation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 488–501.

[62] (2018) Twitter Streaming Traces. [Online]. Available: https://archive.org/details/archiveteam-twitter-stream-2018-04

[63] N. J. Yadwadkar, F. Romero, Q. Li, and C. Kozyrakis, "A case for managed and model-less inference serving," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2019, pp. 184–191.

[64] W. Seo, S. Cha, Y. Kim, J. Huh, and J. Park, "Slo-aware inference scheduler for heterogeneous processors in edge platforms," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 4, pp. 1–26, 2021.

[65] "NVIDIA Triton Inference Server Documentation," https://docs.nvidia.com/deeplearning/triton-inference-server/, accessed: 2024.

[66] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, "Optimizing inference serving on serverless platforms," *Proceedings of the VLDB Endowment*, vol. 15, no. 10, 2022.

[67] Zhang *et al.*, "Top-pim: Throughput-oriented programmable processing in memory," in *Proceedings of the 23rd International Symposium on HPDC*. New York, NY, USA: Association for Computing Machinery, 2014, p. 85–98.

[68] J. Torrellas, "Flexram: Toward an advanced intelligent memory system: A retrospective paper," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 3–4.