MAGMA: An Optimization Framework for Mapping Multiple DNNs on Multiple Accelerator Cores

Sheng-Chun Kao
Georgia Institute of Technology
felix@gatech.edu

Tushar Krishna
Georgia Institute of Technology
tushar@ece.gatech.edu

***Abstract*—As Deep Learning continues to drive a variety of applications in edge and cloud data centers, there is a growing trend towards building large accelerators with several sub-accelerator cores/chiplets. This work looks at the problem of supporting multi-tenancy on such accelerators. In particular, we focus on the problem of mapping jobs from several DNNs simultaneously on an accelerator. Given the extremely large search space, we formulate the search as an optimization problem and develop an optimization framework called M3E. In addition, we develop a specialized optimization algorithm called MAGMA with custom operators to enable structured sample-efficient exploration. We quantitatively compare MAGMA with several state-of-the-art methods, black-box optimization, and reinforcement learning methods across different accelerator settings (large/small accelerators) and different sub-accelerator configurations (homogeneous/heterogeneous), and observe MAGMA can consistently find better mappings.**

***Keywords*-Multi-tenancy; Multi-core accelerator; Genetic Algorithm; Scheduling**

I. Introduction

Accelerators for Deep Neural Network (DNN) models are commonplace today in both the cloud and edge. As AI workloads continue to drive up the demand for compute, there is a trend towards building large accelerators housing several sub-accelerator/arrays (summarized in Table I). Key examples include MCM-based SIMBA [82], wafer-scale Cerebras [8] or scaled-out platforms [3], [26], [40]. Some recent studies have also explored heterogeneous multi-accelerator designs enabled via reconfiguration [39] or separate *heterogeneous* sub-accelerators [49].

With the emergence of such platforms, enabling multi-tenancy, i.e., multi-DNN mappings on the an accelerator/platform, is a natural use-case. Data center workloads often run three categories of inference tasks: vision, language and recommendation, and in each task it involves variants of related DNN models [2], [69]. In this work, we target all three use cases and focus on batched-job tasks (jobs launched in bulk without latency constraint but with high-throughput need [2], [69], [77]), e.g., Google photo auto-editing, image tagging, video processing, and voice processing.

There have been a few recent works looking into the problem of mapping multi-DNN workloads on multiple accelerator cores. PREMA [15] develops a mapper for multi-tenant language tasks, however targeting single-core accelerator. AI-MT [3] successfully designs a mapper for homogeneous multi-core accelerators and shows performance improvement over vision and language tasks. Herald [49] targets heterogeneous multi-core accelerators and systematically analyzes the benefit of heterogeneity in dataflows across the accelerator cores for AR/VR workloads (vision tasks). These works demonstrate the impact of a mapping (of DNNs) for the new multi-tenant multi-core accelerators, which is of rising interest, as shown in Table I. However, all these approaches rely on manually-designed heuristics. This limits their scalability to diverse accelerator back-ends and emerging workloads. In this work, we develop an automatic mapping search process which includes two specific contributions (i) an optimization framework and (ii) a novel optimization algorithm.

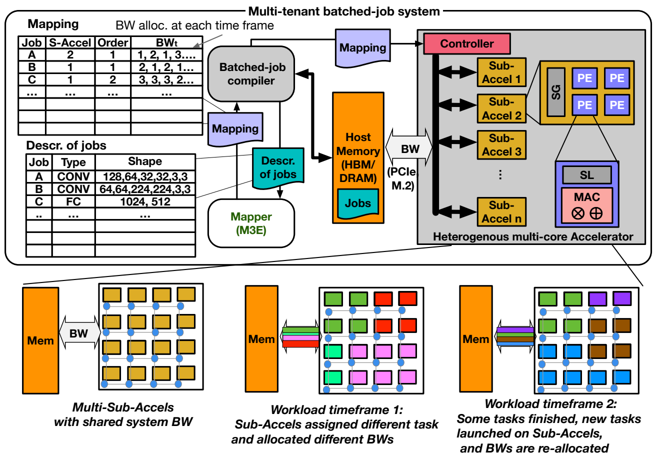

Multi-tenant batched-job system
Mapping
BW alloc. at each time frame
Job | S-Accel | Order | BWt
A | 2 | 1 | 1, 2, 1, 3....
B | 1 | 1 | 2, 1, 2, 1...
C | 1 | 2 | 3, 3, 3, 2...
Descr. of jobs
Job | Type | Shape
A | CONV | 128,64,32,32,3,3
B | CONV | 64,64,224,224,3,3
C | FC | 1024, 512
Batched-job compiler
Mapping
Descr. of jobs
Mapper (M3E)
Host Memory (HBM/ DRAM)
Jobs
BW
(PCIe, M.2)
Controller
Sub-Accel 1
Sub-Accel 2
Sub-Accel 3
Sub-Accel n
SG
PE
SL
MAC
Heterogenous multi-core Accelerator
Mem
BW
Multi-Sub-Accels with shared system BW
Workload timeframe 1: Sub-Accels assigned different task and allocated different BWs
Workload timeframe 2: Some tasks finished, new tasks launched on Sub-Accels, and BWs are re-allocated

Fig. 1: Multi-tenant multi-core accelerator.

We propose an optimization framework called Multi-workload Multi-accelerator Mapping Explorer (M3E). In the framework, (i) we develop an efficient encoding scheme to encode the search space of the mapping; (2) we develop several modules to enable the found mapping to effectively orchestrate the data movement across sub-accelerator cores; (3) we enable several commonly used black-box optimization algorithms and two reinforcement learning methods to be leveraged as the underlying optimization methods. In

Methods	Sample efficiency	Optimality
RLs (A2C, PPO2)	Good	Easier to fall into local optimum [9][19][98]
Population -based †	Poor	Has better global view [4][60]
MAGMA (this work)	Good (provided by designed operators*)	Has better global view

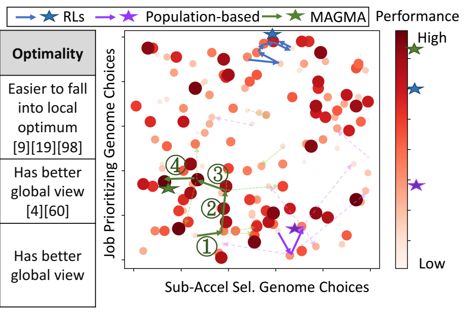


Fig. 2: The high-level characteristics of different optimization methods (see Table IV) and their effects on their exploration steps [4], [9], [19], [55], [90].
†:Population-based methods: stdGA, DE, CMA-ES, TBPSA, and PSO in Table IV.
*: *Crossover-gen* (MAGMA's operator) can perturb Job-Prior genome (X-axis) while respecting most of the Sub-Accel genome (Y-axis) (step-②). Likewise, *Crossover-gen* and *Crossover-accel* (MAGMA's operators) can perturb Sub-Accel genome while respecting most of the Job-Prior genome (step-③,④). *Crossover-rg* (MAGMA's operator), which respects segments of both genome simultaneously, becomes useful when a segment of Job-Prior genome is highly correlated with corresponding Sub-Accel genes. The designed operators will formulate the exploration in more structured manners, which follow the observed characteristics in this scheduling problem.

M3E, we break the multi-tenant mapping problem into two components: *sub-accelerator selection* and *job prioritization*. Sub-accelerator selection is where we assign each job a sub-accelerator to execute; job prioritization is where we order the jobs that are assigned to an sub-accelerator. Each component creates an immense design space by itself. The full design space is the combinatorial probability of both, which becomes as large as O(1e81) (Section IV-F). It also motivates us to design an sample-efficient optimization algorithm.

We propose a custom genetic algorithm-based optimization method for this mapping problem, termed Multi-Accelerator Genetic Mapping Algorithm (MAGMA)[1]. We design custom genetic operators in MAGMA, which allows it to structurally explore the design space and largely increase its sample efficiency.

Compared to prior work on multi-tenant DNN mapping [3], [15], [49], this work expands the scope of the problem-space in the following ways:

- We utilized optimization-based mapper to solve the mapping problem, while prior arts focus on manually designing a mapper.
- We target both homogeneous and heterogeneous DNN accelerator platforms.
- We target a diverse spectrum of models across vision, language and recommendation, which exhibit different bandwidth requirements.

Our solution, M3E includes the following novel features:

- ① We frame mapping into an optimization problem and develop key modules to present a complete optimization framework (M3E), which provides the infrastructure for the research community interested in new mapper design or the test-bed for the developed new optimization algorithms.
- ② A novel optimization algorithm called MAGMA, which includes several domain-aware operators to enable structured exploration of the large mapping space, as shown in Fig. 2. This makes MAGMA orders of magnitude faster and more sample-efficient than baseline optimization methods [28], [32], [33], [45], [73], [76], baseline genetic algorithm (GA) [33], and Reinforcement-learning (RLs) [63], [81], as our results demonstrate.
- ③ In our evaluation, we found MAGMA is (geomean) 1.4x and 1.41x better than Herald-like [49] and AI-MT-like [3] mappers, and 1.6x better than other comparing optimization methods in homogeneous multi-core accelerators. We found MAGMA is (geomean) 1.7-2.3x better than Herald-like, 39-52x better than AI-MT-like, 10-13x better than black-box optimizations, and 1.01-1.3x better than RLs in heterogeneous multi-core accelerators.

[1]This work is available at https://github.com/maestro-project/magma.

Table I: The comparisons of related works on multi-tenancy and multi-core accelerators.

Scale-out Accelerators	Multi-core (Multi-Sub-Accelerators)	Dataflow flexibility across cores	Multi-tenancy support	Multi-tenancy mapper
SIMBA[90]	Homogeneous (36 on-package chiplets)	No	No	NA
TPUv3 [42]	Homogeneous (4 [128x128] TPU cores)	No	No information in public	NA
Cerebras [8]	Homogeneous	Yes	No	NA
AI-MT [3]	Homogeneous (multiple systolic arrays)	No	Yes	Manual
PREMA [15]	Single-core (Single systolic array)	No	Yes	Manual
Planaria [41]	Heterogeneous (reconfigurable systolic array)	No	Yes	Manual
Herald [53]	Heterogeneous dataflow sub-accelerators	Yes	Yes	Manual
M3E (this work)	Homogeneous and Heterogeneous	Yes	Yes	MAGMA

II. Background

A. Characteristics of DNN Models

In this paper, we consider three types of tasks/ applications that are common in inference data centers [2], [69], [77] or edge multi-tenant systems [3], [49]: vision, deep recommendation system, and language model.

Vision. Most of vision models [29], [48], [79], [87], [92] are dominated by convolution layers (2D/depth-wise/point-wise) (CONV) and many of them have a MultiLayer Perceptron (MLP) or fully connected layer (FC) at the end [48], [87].

Recommendation. Recommendation models are either dominated by MLP, attention, or embedding lookup layers [13], [27], [30], [65]. With respect to the compute and HW cost, the MLPs and the attention layers are modeled as several FCs. We assume the embedding lookups are kept in the CPU host.

Language. Language models are often dominated by embedding lookup, MLPs, RNNs, and attention layers [21], [22], [72], [74]. At long sequence range, attention layers becomes a heavy job owing to its quadratic complexity for both compute and memory [95], [96].

B. Multi-core Accelerator

*1) **Accelerator Architecture**:* An accelerator houses multiple cores (termed *sub-accelerators* in this paper). As shown in Fig. 1, all sub-accelerators share the "system BW" via an interconnection network. We define system BW as the minimum of main memory (e.g., HBM/DRAM) BW and host-to-accelerator (e.g., PCIe) BW. The specific interconnection network architecture can vary depending on the target technology (on-chip [3] versus on-package [82] versus wafer-scale [8]) and the scheduler is agnostic to this. In this work, we target accelerators with both homogeneous and heterogeneous sub-accelerators. The motivation for heterogeneity among sub-accelerators comes from diverse dataflow preferences for different kinds of layers. For instance, certain sub-accelerators could be optimized for convolutions [11], [66] to service vision models, some for GEMM [41], [67] to service NLP models, and some for and embeddings to service recommendation models [36]. Recent work Herald [49] also shows that heterogeneous accelerators are beneficial for multi-DNN workloads.

*2) **Sub-Accelerator Architecture**:* Each sub-accelerator in our system is a conventional DNN accelerator that is comprised of an array of Processing Elements (PE). Each PE has a MAC to compute partial sums, and local scratchpad (called *SL* in this paper) to store weights, activations, and partial sums. Each sub-accelerator also houses a shared global scratchpad (*SG*) to prefetch activations and weights from HBM/DRAM for the next tile of computation that will be mapped over the PEs and SLs. Networks-on-Chip (NoCs) are used to distribute operands from the SG to the SLs and write the outputs back to the SG.

Dataflow. Given a DNN layer, each sub-accelerator employs a dataflow (*aka local-mapping*). The dataflow determines the loop order, parallelism dimensions, and tile sizes for running the layer on the sub-accelerator. From the data movement perspective, a tile is a basic data movement unit from DRAM/HBM to the SG. The tile sizes are bound by the size of the SG buffer. The SG is double-buffered [12], [50] to try and hide the data-fetching latency of the current tile from DRAM/HBM behind the compute latency. However, *if the bandwidth to main memory is insufficient to hide the fetch latency (based on the bandwidth allocation determined by the scheduler), the accelerator will stall.* For instance, the NVDLA [66] dataflow keeps weights in the outermost loop (i.e., weight-stationary) and schedules the weight tiles spatially over the array along the input and output channels for parallelism.

III. Problem Formulation

The focus of this paper is *designing an automated mapper for multi-tenant DNN accelerators housing several homogeneous or heterogeneous cores*. The output of our mapper is a *global mapping*, i.e., mapping of independent jobs across the accelerator cores over space and time. We describe the problem formulation and assumptions in detail here. In Section IV, we formulate this mapping problem as an optimization problem and propose an optimization framework. In Section V, we propose an algorithm targeting this mapping problem.

Jobs. We focus on systems targeting multi-tenancy and batched-job tasks [2], [69], [77]. Batched-job tasks are usually not latency sensitive and targeting high throughput; in contrast, single-job tasks are often latency-sensitive and targeting minimum response time. Moreover, multi-tenant batched-job tasks also has much less layer dependency issue than single-tenant single-job tasks for following reasons. i) As shown by the observation in AI-MT [3], multi-tenancy, running multiple neural networks together, can largely alleviate the layer dependency problem as layers from different neural networks can be freely scheduled without any dependency issue. ii) In batched-job tasks, where often hundreds to thousands of activations are running by the same model (e.g., video processing), in practice, the activations will be broken down to mini-batches because of memory concern. These mini-batches are independent with each others, which further alleviate the dependency issue when scheduling these mini-batches. In this paper, we refer to a "job" as a mini-batch of a layer — a job is a mini-batch of activations and a set of weight parameters of a layer, where the layer belongs to one of the independent models in the multi-tenant system.

Group. At the host, we run a light-weighted control program to divide a pool of queuing jobs into multiple "dependency-free [3]" groups (Jobs inside each group have no dependency). The group size is a hyper-parameter. Larger group size could increase the search space of the mapper and increase the expected performance; however, larger group size also increase the difficulty for the mapper which could lead to sub-optimal performance. Also, group size should be larger or equal to number of sub-accelerators to avoid some sub-accelerators being idle completely.

Mapping. Mapping is the strategy to orchestrate the jobs assignment and job prioritizing across sub-accelerators over space and time. It could also be called *global mapping* (a global control of the data movement across host and sub-accelerators), while dataflow (Section II-B2) is called *local mapping* (a local control of the data movement inside one sub-accelerator).

Mapper. The mapper for multi-tenant multi-core accelerators for DNN workload is a comparatively new and rarely

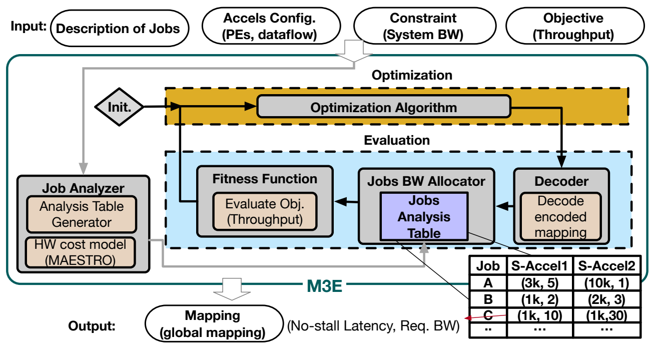


Fig. 3: The structure and flow† of M3E.

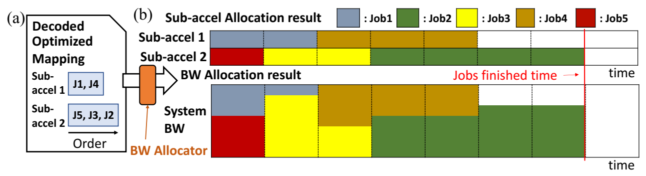


Fig. 4: (a) Mapping description from the decoder. (b) The example BW and sub-accelerators allocation results.

explored topic. Recent works such as PREMA [15], AI-MT [3], and Herald [49] have motivated the need of an effective mapper for new DNN accelerators targeting multi-tenant DNNs and multiple sub-accelerator cores.

A mapping for such accelerators could be a manually-designed heuristic or found by a mapping optimizer. The manual-designed mapping heuristic [3], [49] is often highly tuned for specific accelerators/ systems or tasks/ workloads. The challenge with this approach is that whenever the underlying accelerators or the targeting tasks change, the mapper need to be re-designed or at least undergo another lengthy cycle of engineer-intensive tuning process.

In this work, unlike the previously mentioned manual-designed mappers [3], [49], we explore the potential of the mapping optimization method. It has the benefit of ”automatics”, i.e., we simply need to re-run the search of the optimizer whenever the underlying accelerator evolves or target task changes, which happens more and more frequently at the era of the fast-paced hardware evolution. We discuss the proposed mapping optimization framework next.

IV. Optimization Framework (M3E)

We propose a mapping optimization framework for multi-tenant heterogeneous accelerator. The structure of our proposed framework called M3E is shown in Fig. 3.

At a high-level, the M3E consists of an optimization phase and an evaluation phase. At evaluation phase, the candidate mapping is evaluated by the cost model. At optimization phase, the optimization algorithm tweak the mapping based on the feedback from the evaluation phase. The optimization-evaluation loop happen iteratively until the targeting objective converges or after a fixed set of time epochs.

The mapping consists of two key components:

- **Sub-accelerator selection:** the assignment of each job to a specific sub-accelerator.
- **Job prioritization:** execution order of jobs on a given sub-accelerator.

To successfully frame a problem into an optimization process, there are two critical pillars: encoding (how the search space is described) and optimization algorithm (how the search space is explored). We describe them as follows.

A. *Encoding*

An encoded mapping should encode the joint strategy of job prioritization and sub-accelerator selection. We encode them into a series of values with two separate sections, as shown in Fig. 5(a). The sub-accelerator selection section decides which job goes to which sub-accelerator. The job-prioritizing section decides the order of the jobs in each sub-accelerator. The length of the section is equal to group size. A full mapping consists of two sections with total length 2x group size. We describe the encoding using the example in Fig. 5(a) assuming two sub-accelerators and a group size of 5.

Sub-accelerator Selection Section. Each value describes the sub-accel ID for the corresponding job. For example, jobs J1 and J4, are assigned to sub-accel 1, and J2, J3, and J5 are assigned to sub-accel 2 as shown in the sub-accel selection part of the decoded assignment in Fig. 5(a).

Job Prioritizing Section. Each value describes the priority of the corresponding job. The priority value ranges from 0 to 1, where 0 is the highest priority. We order the job assigned to a certain sub-accelerator by the order of priority value. For example, J1 runs before J4 in sub-accel 1 as shown in the job prioritizing part of the decoded assignment in Fig. 5(a).

B. *Optimization Algorithms Supported*

With an encoded search space and the support of building blocks of M3E (described later in Section IV-D), we support several popular optimization algorithms, as shown in Table IV. We include multiple black-box optimization methods such as Differential Evolution [73], Covariance Matrix Adaptation Evolution Strategy [28], and Particle Swarm Optimization [45]. In addition, we also include two widely-used reinforcement learning methods — Advantage Actor-Critic (A2C) [63] and Proximal Policy Optimization (PPO2) [81]. Finally, we also support our novel optimization algorithm called MAGMA (Section V). With the established framework, M3E can also easily be extended to support other algorithm proposals.

C. *Objective and Constraints*

We target throughput as our main objective. However, other objective can also be set (e.g., latency, energy) or formulated (e.g., energy-delay-product, performance-per-watt). The objective can simply be specified as an input to the M3E, as shown in Fig. 3. We consider BW constraint (however

Algorithm 1 BW Allocator

```
Input: Mapping description
Output: $BW_t^{alloc}$, t=1,2...T
Get $Lat_t$, an array of no-stall latency for the parallel jobs at time t, t=0
Get $BW_t^{req}$, an array of required BW for the parallel jobs at time t, t=0
$CurJobs_t = Lat_t \times BW_t^{req}$
while $CurJobs_t$ is not empty do
    if sum($BW_t^{req}$) $< BW_{sys}$ then
        $BW_t^{alloc} = BW_t^{req}$
    else
        $BW_t^{alloc} = \frac{BW_t^{req} \times BW_{sys}}{sum(BW_t^{req})}$
    end if
    $runtimes = \frac{CurJobs_t}{BW_t^{alloc}}$
    $runtime$ = min($runtimes$)
    $CurJobs_t$ -= $runtime \times BW_t^{alloc}$
    $accel_{next}$ = argmin($CurJobs_t$)
    $t$ += $runtime$
    Fetch the next $Lat$ and $BW^{req}$ of $sub-accel_{next}$, compute $CurJob_t$
    and insert into $BW_t^{req}$, $Lat_t$ and $CurJobs_t$.
end while
```

similarly, other constraints can also be specified). The BW constraints include accelerator-to-host BWs (e.g.,PCIe, M.2) and host memory BW (eg., DRAM/HBM BW), which are common constraints in the practical deployment scenario [49], [69]. For simplicity of the optimization framework, we take the minimum of the two (the more stringent BW constraint), as the BW constraint known by the optimization framework, and we name this constraint — system BW.

D. Building Blocks of M3E.

*1) **BW Allocator**:* However, in a multi-core accelerator, system BW to the accelerator becomes a global shared resources between cores (sub-accelerators). To evenly allocate the same amount of BW to all the sub-accelerators is an often applied heuristics. However, it will increase the possibility of compute resource under-utilization. E.g., in a normal single-accelerator case, depth-wise CONV jobs are often more memory-intensive than regular 2D CONV jobs, which can make the accelerator under-utilized when running depth-wise CONV while it is fully-utilized when it runs 2D CONV. In the multi-core accelerator, where the system BW is a global shared resources [69], it gives us a chance to reallocate the BW to alleviate the under-utilization problem by proving more BW to core running memory-intensive jobs and proving only adequate BW to cores running compute-intensive jobs, which motivates the BW allocator (Algorithm 1). The BW allocator is reallocating the BW based on the memory BW intensity of different jobs running on different sub-accelerators.

In detail, receiving the mapping, the BW allocator lookup those jobs' no-stall latency (Section IV-D2) and required BW from the job analysis table (Section IV-D2), and allocates the system BW to each sub-accelerator at each time frame with the ratio of their required BWs. It outputs the detailed BW allocation results, as shown in Fig. 4(b).

Fig. 4(b), a BW allocation results, as an example, we can tell, jobs J1 and J5 will be launched in Sub-accel-1 and Sub-accel-2, concurrently. Sub-accel-2 will be allocated more BW because it is running a more BW-intensive job. When Sub-accel-2 finishes J5 and launches J3, the BW will be re-allocated to reflect the change of live running jobs in the accelerators, where Sub-accel-1's BW is reduced and reallocated to Sub-accel-2 in Fig. 4.

*2) **Job Analyzer**:* The job analyzer takes the jobs description as input and estimates the no-stall latency and its required BW for each sub-accelerator using a cost model (described below) to generate a job analysis table as Fig. 3 shows. This table serves as a performance lookup table by the BW allocator (Section IV-D1) within the optimization loop.

*3) **HW cost model for Sub-Accelerators**:* In M3E, we leverage MAESTRO [1] as our underlying cost model for each sub-accelerator because of its ability to support diverse accelerator dataflows and configurations[2]. It supports most of the common DNN layers such as CONV, depth-wise CONV, and fully connected. Given a DNN layer, a HW resource configuration (PE, SL size, SG size, NoC latency, and BW), and a mapping/dataflow strategy, MAESTRO estimates the statistics such as latency, energy, runtime, power, and area.

*4) **Job Analysis Table**:* Job Analyzer profiles each job in the task by the cost model [1] and stores the profiling results in the Job Analysis Table. In the optimization process, Job Analysis Table serves as a quick look-up table for fitness evaluation to avoid frequently querying the cost model. The profiling has two main information: no-stall latency and no-stall bandwidth, described next.

No-stall Latency. We define no-stall latency as the latency for running each job on each sub-accelerator, assuming it has sufficient memory bandwidth (i.e., not memory-bound).

No-stall Bandwidth. We define no-stall bandwidth as the minimum bandwidth requirement from each sub-accelerator to make it stay compute-bound, not memory-bound.

E. M3E Workflow

Set-up: At the start, the user/host sets up the optimizer by feeding in the jobs descriptions, configurations (number of PEs, dataflow) of each sub-accelerators, the system constraint (system BW), and objective (e.g., throughput).

Pre-process: **Job analyzer** Job Analyzer prepares the Job Analysis Table, as shown in Fig. 3.

Optimization Loop: *Optimization phase:* optimization algorithm updates the encoding mapping based on the feedback from the evaluation block. *Evaluation phase:* **Decoder** decodes encoded mapping into a mapping description, as shown in Fig. 4(a). **BW Allocator** takes in the mapping description and allocates the BW for each sub-accelerator.

[2]In this paper, we explore heterogeneity with the aspect of different specialized DNN accelerators configurations (PEs, buffer size, dataflows). However, M3E is general enough, so that it could also consider generic architectures such as CPUs/GPUs/TPUs by plugging in their cost models.

Table II: Terminology used in MAGMA Algorithm.

Term	Description
Gene	An encoded value that represents accel. sel. or job prio. of a job.
Genome	A series of genes that represent the entire schedule about accel. sel. or job prio. of a batch of jobs.
Individual	A series of genomes that fully represent the schedule of a batch of jobs.
Generation	An entire set of individuals forms a generation. The generation evolves with time by mutation/crossover and selection of the well-performing individuals to the next generation.
Crossover	Blend two parents' genes to reproduce children's genes.
Mutation	Randomly perturb a parent's genes to reproduce children's genes.

Fitness function extracts the objective and sets it as fitness value.

This finishes one loop/ epoch of optimization. The optimization loop stops when M3E reaches the set sampling budget (the number of allowed sampling data points in a search process).

F. Search Space

The full search space of the proposed framework is the combinatorial combination of the choices for sub-accelerator selection and job prioritizing. Assuming the accelerator has 4 sub-accelerator and we use the group size of 60. The size of the design space is $(60!)/(15!)^4 \times (15!)^4 = 60! = O(1e81)$ which is extremely massive. Therefore the *sample efficiency*[3] of the optimization methods, which decides the convergent rate, becomes a key factor. We describe our proposed sample-efficient optimization method, next.

V. Optimization Algorithm (MAGMA)

MAGMA is a GA-based search technique. Its key difference from standard GA is that it customizes the optimization algorithm's exploration momentum and mechanism (i.e., genetic operators in GA context) for the target search space.

A. Why GA?

Research shows GA reaches competitive performance with deep reinforcement learning [78], [90], and hyper-parameter optimization problem. STOKE [80] and Tensor Comprehensions [99] use GA to search the space of DNN code optimization. From a search time perspective, GA is light and fast [78], [90] comparing to many optimizations methods since the optimization mechanism in GA uses simple operations (e.g., crossover and mutations). A key challenge with standard GA however is that it is not sample-efficient. We address this issue using our customized operators (Section V-B2).

B. MAGMA Algorithm Details

*1) **Terminology and Basics of GA**:* We list the common terminology of GA we use throughout the paper in Table II, namely *gene, genome, individual, generation*. The basic mechanism in GAs is to create a population of individuals in each generation. All individuals are evaluated and sorted based on their fitness. The best performing individuals are used as parents to create a new population of individuals using genetic operators (Section V-B2). The goal of GA is to perturb genes (i.e., components of the schedule) and retain well-performing ones across generations.

[3]Performance improvement over the number of sampling budget.

*2) **Genetic operators**:* **Standard GA Operators.** The standard genetic operators in GA consist of mutation and crossover. The standard mutation operator randomly mutates some genes. The standard crossover operator samples a pivot point and exchanges the genes of parents according to that pivot point. The sampling efficiency of the GA relies on the efficiency of the genetic operators to sample high-quality next generation.

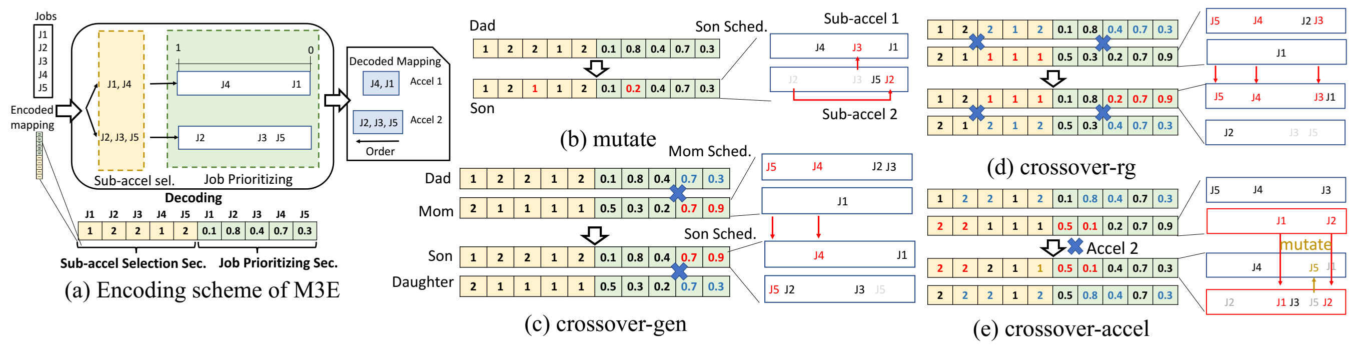
MAGMA Operators. In MAGMA, we inherit the standard mutation mechanism and design *three specialized crossover genetic operators*. Different crossover operators are designed to preserve different dependency of genes while exploration. They allow us to explore the scheduling problem in a more strategical manner. We describe the genetic operators next.

Mutation. During mutation, we randomly select multiple genes (according to the mutation rate) and mutate them to random values. Fig. 5(b) shows an example when mutating at the third and second genes of two genomes respectively. On the right side of the figure, it shows how the son's genes/schedule are generated by the dad's mutation. J3 is moved to sub-accel 1 because of the first mutation. J2 is moved to a higher priority in sub-accel 2 because of the second mutation. In our experiments, we use a mutation rate of 0.05.

Crossover-gen. This is a genome-wise crossover. First, we randomly sample a type of genome to crossover. Next, we randomly sample a pivot point and exchange the genes of the genomes. There are two benefits of genome-wise crossover. First, we keep the perturbation to the level of the genome, which potentially keeps the good characteristics of the other un-touched genomes, and therefore is more stable throughout the evolution. Second, we eliminate the order dependency of the genomes. The genomes are independently representing their features, where the order of them provides no information (, i.e., representing Sub-accel Sel. genome first and Job Prio. Genome later does not make the J5 of Sub-accel Sel. and J1 of Job Prio. strongly correlated despite their being next to each other.). Therefore, a genome-wise crossover, which operates genomes independently, enables us to perturb the gene without unnecessary assumptions of the genome order. Crossover-gen becomes the major crossover function, which we set the crossover rate as 0.9.

Fig. 5(c) shows an example that we pick the second genome (Job Prio.) as the crossover region and the third location of the region as the pivot point. With the respect of schedule change after crossovering, in the example, the orders of J4 and J5 in mom's schedule are passed to son's schedule.

Crossover-rg. This is a range crossover mechanism



(a) Encoding scheme of M3E (b) mutate (c) crossover-gen (d) crossover-rg (e) crossover-accel

Fig. 5: (a) Then encoding scheme of M3E. The genetic operators of MAGMA and their implying update on the mapping: (a) mutation, (b) crossover-gen, (c) crossover-rg, and (d) crossover-accel.

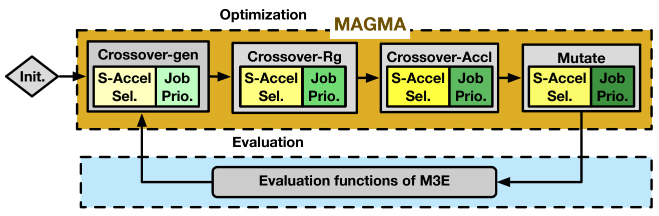


Fig. 6: The algorithm flow of MAGMA.

structured to preserve the the dependency of genes across genomes. For example, in Fig. 5(a), the first and the sixth genes are dependent, since they are both representing some features for J1. We randomly pick a range of genome (e.g., the 3rd to the 5th locations of each genome) and simultaneously crossover all the genes falling into the picked region from both genomes, and thus the cross-genome dependency is preserved. With the respect of scheduling change after crossovering, the order and accel selection of J3, J4, and J5 are exchanged between two individuals, as shown in Fig. 5(d). Crossover-rg has crossover rate of 0.05.

Crossover-accel. This is a crossover method to preserve the dependency of job ordering within an sub-accelerator. We randomly select a sub-accelerator and pass the job ordering information of this sub-accelerator to the children. For example, in Fig. 5(e), we select sub-accel 2. Next, we check the Sub-accel Sel. genome of Mom, copy the genes related to sub-accel 2 (the first and second genes of both genomes in (e)), and paste them to son's genomes.

To increase load balancing, the original jobs assigned to sub-accel 2 in Son will be randomly mutated. Crossover-accel has crossover rate of 0.05.

*3) **Hyper-parameter Tuning**:* The above mentioned mutation, crossover rates, populations sizes, and elite ratios are hyper-parameters in MAGMA. We applied a hyper-parameter search via a Bayesian optimization framework [7] to select a set of hyper-parameters that makes MAGMA achieve the highest performance across multiple workloads.

Table III: Accelerators configurations/ settings of the experiments.

Setting	Description	# of sub-accels	(height of PE array, dataflow style, buffer)
S1	Small Homog	4	4x(32, HB, 146KB)
S2	Small Hetero	4	3x(32, HB, 146KB), 1x(32, LB, 110KB)
S3	Large Homog	8	8x(128, HB, 580KB)
S4	Large Hetero	8	7x(128, HB, 580KB), 1x(128, LB, 434KB)
S5	Large Hetero BigLittle	8	3x(128, HB, 580KB), 1x(128, LB, 434KB) 3x(64, HB, 291KB), 1x(64, LB, 218KB)
S6	Large Scale-up	16	7x(128, HB, 580KB), 1x(128, LB, 434KB) 7x(64, HB, 291KB), 1x(64, LB, 218KB)
2D PE array: h (height) × w (width) (PEs), w=64, in the experiments			

C. *Warm-Start of MAGMA*

In this section, we present the techniques we implement to enable the warm-start of the algorithm. Warm-start is a well-known technique in black-box optimizations to enable faster convergence or reaching better objective value. Warm-start works as follows. There are series of tasks to be solved by the optimization algorithms. If the current task is the same or similar to the previous solved tasks, we can take the previous solution to initialize the algorithms. We also implement an warm-start engine, which recognize if the current task fall within the same types of tasks (Vision, Recommendation, or Language), i.e., whether it is similar to the previous solved task. If tasks are within the same type of task, the warm-start engine will take over the initialization job from Init engine (initializing algorithm randomly). In our experiment (Section VI-G), we found warm-start is a useful add-on technique in MAGMA.

VI. Evaluations

A. *Methodology*

1) Target DNN Models: We consider three different types of tasks/ applications with their corresponding models collected from PyTorch: Vision ([29], [37], [79], [87], [93], [94], [107]), Language ([17], [18], [21], [22], [46], [47], [52], [53], [54], [57], [61], [74], [75], [103]), and recommendations ([13], [27], [65], [109], [110]). The models are the ones used for the batched high-throughput applications (e.g., photo auto-editing, image tagging, and video / voice processing) that we are targeting.

Table IV: Supported optimization algorithms in M3E.

Alg.	Description
AI-MT-like	A manual-tuned mapper for multi-core accelerator targeting vision and language workloads.
Herald-like	A manual-tuned mapper for multi-core heterogenous accelerator targeting vision workloads.
stdGA	Genetic Algorithm. *We use mutation rate: 0.1, crossover rate: 0.1.*
DE	Differential Evolution. *We use weighting for local DV: 0.8, weighting for global DV: 0.8, in the experiment.*
CMA-ES	Covariance Matrix Adaptation-ES. *We use 1/2 of the best performing individuals as an elite group in the experiment.*
TBPSA	Test-based Population-Size Adaptation. *We set the initial population size as 50 and let it evolve in the experiment.*
PSO	Particle Swarm Optimization. *We use weighting for global best: 0.8, weighting for parent best: 0.8, with momentum ω: 1.6.*
RL A2C	Advantage Actor-Critic. *We use policy and critic networks composed by 3 MLP layers with 128 nodes, discount factor: 0.99, learning rate: 0.0007, RMSProp optimizer.*
RL PPO2	Proximal Policy Optimization. *We use policy and critic networks composed by 3 MLP layers with 128 nodes, discount factor: 0.99, clipping range: 0.2, learning rate: 0.00025, Adam optimizer.*
MAGMA	A GA-based optimization algorithm that houses domain-specific genetic operators for multi-core heterogenous accelerator mapping problem.

2) Task and Benchmark: We categorize the jobs into Vision, Language (Lang), Recommendation (Recom), and Mix (a complex tasks with vision, language, and recommendation model involved simultaneously), four different types of tasks. We build a benchmark including different tasks motivated by the Facebook's inference accelerator jobs [2], [69], edge data centers for AI applications [77], Herald [49]. AI-MT [3], and others [83]. In the benchmark, we collect models from the four different types of tasks and create several workloads. Each workload contains hundreds to thousands of jobs (one job include a batch of activations and weight parameters of a layer). We chopped them into several "dependency-free" group similar to prior works [3]. The objective of the optimization algorithm is to execute these group with the highest possible throughput. We set the default group size to be 100 but also study the effect of group size in Section VI-G.

3) Accelerators: We consider two classes of accelerator: Small and Large. For each class, we consider multi-core homogeneous and heterogeneous accelerator settings with different PEs, dataflow, and on-chip buffer. We construct six different multi-core accelerators, motivated by [3], [41], [49], [82], as our test-bed in Table III. S1 and S3 represent homogeneous accelerators. S2, S4-6 represent heterogeneous accelerators. The accelerators are modeled with MAESTRO [1]. We uniformly set one dimension of the 2D PEs array to 64[4] and scale the PEs array size by increasing the other dimension. We consider three kinds of PEs configuration: 32×64 for Small accelerator [23], [25], [56], [64], [111], 64×64 and 128×64 for Large accelerator. The dataflow style (discussed next) and target tile sizes determine the buffer sizes for both SL and SG [50].

[4]Based on our observation, most of the popular models that we collected, especially language and recommendation ones, are manually designed to have the tensor shape formed by the multiples of 64. Setting one dimension to 64, which aligns with the tensor shape, ensures higher utilization rate.

Sub-Accelerator Dataflow Styles. For our evaluations, we pick two distinct dataflow styles for the heterogeneous sub-accelerators: High Bandwidth usage dataflow style (HB) (inspired by NVDLA) [66]) and relatively Low Bandwidth usage dataflow style (LB) (inspired by Eyeriss [11]). The HB-style parallelizes across channel dimensions, and shows high-efficiency on late layers for CNN-based (vision) models, while the LB-style parallelize across activations dimensions and excels on the early layers of CNN-based models [50]. For Language and Recommendation, we found the HW-style is more compute efficient but BW intensive, while LB-style is less compute efficient but also less BW demanding (Fig. 7). Therefore we house both these sub-accelerators in a BW constrained accelerator platform to act as a good test for our optimizer to learn and exploit their difference. M3E is general enough to run with any heterogeneous combination of two or more accelerator styles.

System BW. The accelerators are executing under frequency 200MHz and bit-width of 1 Byte. For the system BW, at the Small accelerator, we consider the BW to be range from 1GB/s to 16GB/s, which is the range of DDR1-DDR4 BW [98] and PCIe1.0 - PCIe3.0 [70] BW; at the Large accelerator, we consider the BW to be range from 1GB/s to 256GB/s, which is the range of DDR4-DDR5 [62] and HBM BW [42] and PCIe3.0 - PCIe5.0 and upcoming PCIe6.0 BW [70].

Evaluation Metric In all experiments, we use throughput as the optimization objective.

B. Mapper Settings.

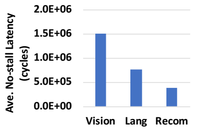
Baseline Manual-tuned Mapper. We use the mapper from Herald [49] and AI-MT [3] as the baseline methods (Herald-like and AI-MT-like). Note that the mapper in Herald [49] is manual-designed targeting multi-core heterogeneous system with Vision tasks, and the mapper in AI-MT is manual-designed targeting multi-core homogeneous system with Vision and Language tasks. In our evaluation, we also tested their performance in Recommendation and Mix tasks and on both homogeneous and heterogeneous accelerators.

Optimization methods. We enable many commonly-used optimization methods in M3E. The specific hyper-parameters settings are listed in Table IV. For fair comparisons, all optimization methods are given the same sampling budget, 10K data points. Note that batched-job tasks are not latency sensitive, where the jobs and optimization process are executed off-line. Therefore the search time of different methods is not our main concern; instead, the objective of these tasks are to utilize the underlying hardware as efficient as possible, i.e, maximizing the throughput of the underlying hardware.

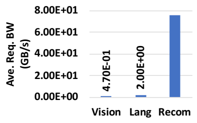
MAGMA. MAGMA is also one of the optimization methods. We set the number of individuals in a generation, to be as large as group size. We also constraint MAGMA to have the same 10K sampling budget, and use population

		Ave. no-stall latency		Ave. Req. BW (GB/s)	
		(HB,64)	(LB,64)	(HB,64)	(LB,64)
Vision	MobileNetv2	3.1E+05	1.1E+06	6.8E-01	8.8E-02
	Resnet50	8.5E+04	7.0E+06	5.2E-01	3.6E-04
	Shufflenet	9.6E+04	4.9E+05	1.5E+00	2.3E-02
	Ave.	1.7E+05	2.8E+06	9.0E-01	3.7E-02
Lang	GPT2	1.7E+04	3.5E+06	3.4E-01	1.6E-05
	MobileBert	5.5E+02	1.1E+05	1.1E+01	5.0E-04
	TransformerXL	4.5E+03	9.2E+05	1.3E+00	6.4E-05
	Ave.	7.4E+03	1.5E+06	4.1E+00	1.9E-04
Recom	DLRM	2.4E+02	9.7E+05	5.6E+01	2.5E-04
	WideDeep	2.8E+02	1.1E+06	3.0E+01	5.2E-06
	NCF	4.6E+01	1.8E+05	3.7E+02	6.4E-05
	Ave.	1.9E+02	7.6E+05	1.5E+02	1.1E-04

(a) Jobs analysis



(b) Ave. no-stall latency



(c) Ave. BW requirement

Fig. 7: (a) The average per-job no-stall latency and required BW for no-stall across different models on high (HW) and low (LB) bandwidth mapping style. (b) Average no-stall latency and (c) average BW required for no-stalls across all involved jobs.

size of 100, and thus have 100 epochs for optimizing. As for search time, we run the experiments on a desktop with Intel i9-9820 CPU. MAGMA takes about 0.25 seconds per epoch, and 25 seconds for a full optimization process.

C. *Latency-BW Characteristics of DNNs*

We start by showing the latency characteristics and bandwidth requirements of the DNN models from the three types of tasks when running by itself on two separate dataflow styles (HB and LB). We show three of the models from each type of tasks and the average across all the models in that type of tasks in Fig. 7(a). The average values across all model across both accelerators are plotted in Fig. 7(b-c). From Fig. 7, in general, we can see that the per-job latency of the Vision models is higher because more compute is needed in the CONV dominant models. However, CONV is generally less memory-bound than FC. The data also shows that usually Vision has the lowest BW requirement, and Recommendation has the largest.

D. *Homogeneous Accelerators*

We examined the Small homogeneous accelerator (S1) with system BW=16 GB/s across different tasks. As shown in Fig. 8, Herald-like and AI-MT-like mapper works rather well across four different tasks. The result form AT-MT-like shows that even though it is designed considering vision and language tasks, it also work adequately well in recommendation and even mix task. Likewise, Herald-like is designed for vision task and work well when applying to others. For optimization methods, they can reach similar performance as Herald-like and AI-MT-like. Note that, these optimization methods are not originally designed for this specific mapping task. However, they work adequately well working in the M3E framework. Overall, MAGMA outperforms others. MAGMA reach performance (geomean) 1.4x and 1.41x better than Herald and AI-MT, and (geomean) 1.6x better than other optimization methods.

E. *Heterogeneous Accelerators*

We examined the Small (S2) and Large (S4) heterogeneous accelerators across different tasks in Fig. 9. In the following results, we will focus on presenting the result of Mix task, since it is a more complex task and is a fair realistic use-case in nowadays' inference data centers [2], [69]. In Fig. 9, we also present Vision task result as baselines, since both Herald-like and AI-MT-like has Vision task as target.

Small Heterogeneous Accelerators (S2). As shown in Fig. 9(a), Herald-like performs well, while AI-MT-like has comparatively lower performance. It shows the different characteristic of two algorithms. Herald-like is designed for heterogeneous accelerator while AI-MT-like is targeting homogeneous accelerator, which explains the performance difference. For optimization methods, many of them reach comparable performance to Herald-like, while PSO and CMA have lower performance (comparable to AI-MT-like). For a more complex Mix task (Fig. 9(b)), AI-MT-like has comparatively worse performance. Many optimization methods undergo lower performance, too. However, the two RLs methods stands out. Overall, MAGMA outperforms others in both tasks. By geomean, MAGMA is 2.3x better than Herald, 39.5x better than AI-MT, 13.4X better than optimization methods excluding RLs. RLs and MAGMA have compatible performance and MAGMA achieve slightly better result 1.01x better.

Large Heterogeneous Accelerators (S4). In large accelerator, the mapping task becomes more complex since the design space of the mapping grows. As shown in Fig. 9(c)(d), Herald-like perform rather well in Vision task. However, at a more complex Mix task and a more complex large accelerator case, Herald-like starts to undergo lower performance. Many more basic optimization process cannot tackle the large and complex design space as well (Fig. 9(d)). However, RLs starts to shine and reach good performance. Overall, MAGMA outperforms others in both tasks. By geomean, MAGMA is 1.7x better than Herald, 52x better than AI-MT, 10x better than optimization methods excluding RLs, and 1.3x better than RLs. Note that the contribution of this work is both the framework M3E (which enables the other optimization methods) and algorithm MAGMA. Before this work, the best performing mapper in Large heterogeneous accelerator setting is Herald-like, which harvests only 20% of maximum throughput enabled by M3E in Mix task, as shown in Fig. 9(d).

Fig. 10 sketches how different methods explore and exploit, leading to their performance difference. For ablation study of sampling budget in different methods, we add a set of experiments, where we let all methods run until they converge, as shown in Fig. 11.

BW-limited Environment. We examine the performance of Small accelerator at BW=16GB/s and Large accelerator at BW=256GB/s. However, in a heavy-loaded inference data

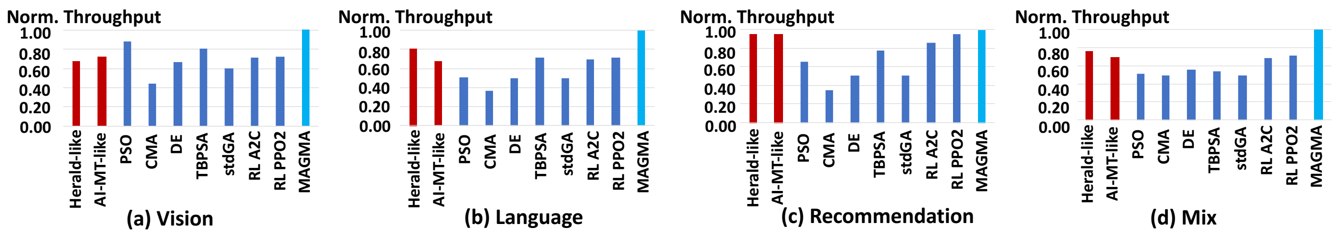


Fig. 8: The experiment results on multi-core homogeneous small accelerator (S1) with BW=16 across four tasks. Throughput values are normalized by the value of MAGMA. The absolute throughput values of MAGMA in (a-d) are: 249, 397, 194, and 329 GFLOPs.

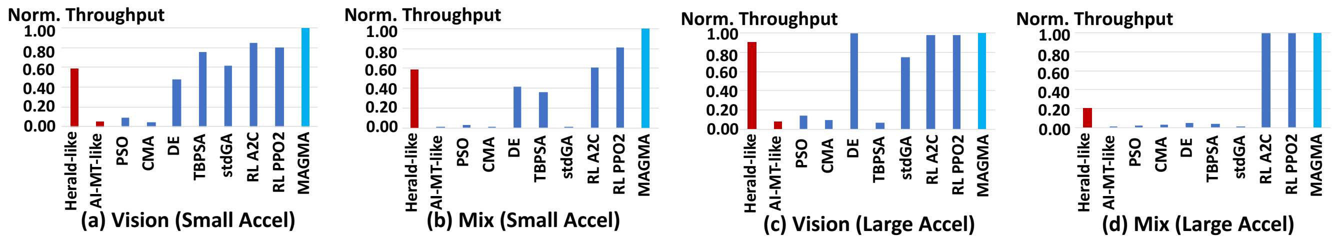
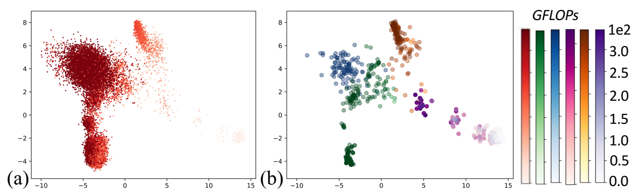


Fig. 9: The experiment results on multi-core heterogeneous (a)(b) small (S2, BW=16) and (c)(d) large (S4, BW=256) accelerator on Vision and Mix tasks. Throughput values are normalized by the value of MAGMA. The absolute throughput values of MAGMA in (a-d) are: 254, 271, 254, and 383 GFLOPs.



(c)

Method	Exhaustively Sampled*	MAGMA	PPO2	stdGA	PSO	CMA
GFLOPs	254	254	101	16	68	19

Fig. 10: (a) The full map space, and (b) the explored map space and (c) the reached performance of different methods in problem (Mix, S2, BW=16). The axes of (a-b) are 2-dimensional projection by PCA [71]. From (b)(c), we can observe CMA, PSO, stdGA, and PPO2 converge to different local optima. MAGMA globally samples a wide region at the start (characteristics as a GA-based algorithm [4]) and quickly converges to an optimum with better performance than other methods (sample efficiency provided by the the designed operators).

Exhaustively sampled*: Running random sampling for around 2 days and more than 1 million samples collected. It represents the best-effort optimum point. (Note that all other methods are run with the set 10K sampling budget.)

center, the BW is a precious resource, where a big portion of it could also be occupied by other applications and leads to a more BW-limited environment. At a more BW-limited environment, mappers become crucial for smartly ordering the jobs to exploit the limited BW. We examine the effect of BW by a BW sweep in Fig. 12. For both Small and Large accelerators, with the decrease of BWs, MAGMA stands out

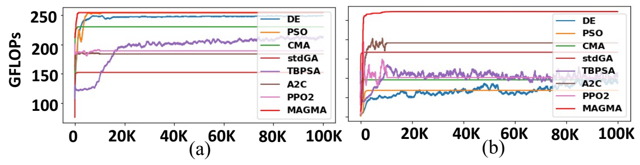


Fig. 11: The convergence curve of different methods across 100K samples on (a) (Vision, S2, BW=16) and (b) (Mix, S3, BW=16). Most of the methods converge before the set 10K sampling budget in (a)(b) (and all experiments in this paper). For few cases, some algorithms require more sampling budget to converge, and we show one of them in (a): TBPSA requires around 20K samples. However, in the end, they all converge to some lower performance points than the points found by MAGMA.

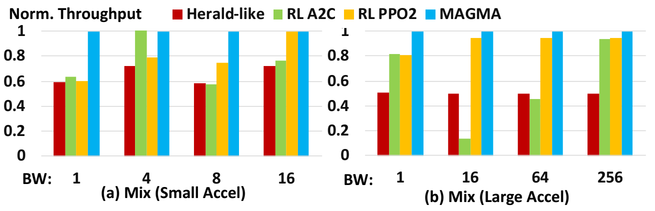
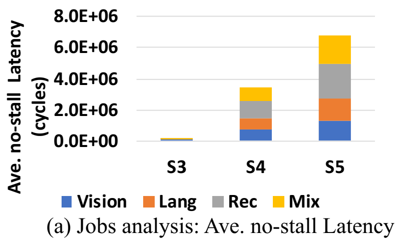
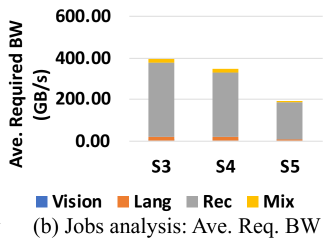


Fig. 12: Performance comparisons on multi-core heterogeneous (a) small (S2) and (b) large (S4) accelerator on Mix tasks, given different BWs. Throughput values are normalized by the value of MAGMA.

more obviously by reaching better relative performance. For example, in Fig. 12(a), MAGMA is (geomean) 1.2x better than others when BW=16GB/s, but MAGMA is 1.6x better


(a) Jobs analysis: Ave. no-stall Latency


(b) Jobs analysis: Ave. Req. BW

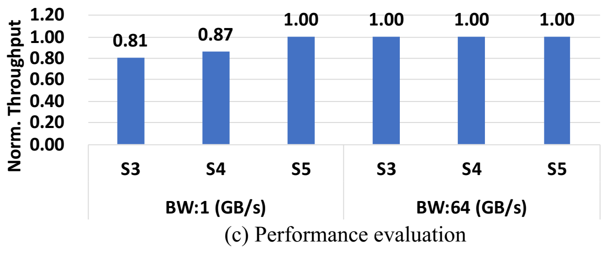

(c) Performance evaluation

Fig. 13: Jobs analysis of the averaged per-job (a) no-stall latency and (b) required BW. (c) Performance evaluation of MAGMA on S3, S4, and S5 with different BW. In (a-b), we concatenate the four independent no-stall latency (averaged required BWs) into a stacked bar and show the total values. In (c), throughput values are normalized by the value of S5.

than other when BW=1GB/s. Similar observation can be made in Fig. 12(b).

1) Sub-accelerator Combinations: In this experiment, we examine the performance change in different settings, S3 (Large Homogeneous), S4 (Large Heterogeneous), S5 (Large Heterogeneous, BigLittle) of the Large accelerator.

Homogeneous versus Heterogeneous. In the following experiment, we discuss the performance implication of a homogeneous versus heterogeneous accelerator using MAGMA algorithm. The LB-style sub-accelerators usually take larger runtime but lower BW requirements than HB-style in language and recommendation tasks, as shown in Fig. 7(a). The jobs analysis in Fig. 13(a-b) reflect the fact that S4, in general, induces more no-stall latency but requires less BW than S3. Therefore, when BW is limited (BW=1), the heterogeneous setting enables accelerator to leverage the difference of BW requirement among sub-accelerators to relax the BW contention. Thus S4 reaches better performance than S3 at BW=1 in Fig. 13(c). However, when the BW is mostly sufficient (BW=256GB/S), the performance will reflect more of the behavior of the no-stall latency. Thus S3 reaches better performance.

Bigs versus BigLittle. We consider an accelerator with a smaller setting, BigLittle (S5), comparing to Bigs (S3, S4). It is obvious when the BW budget is sufficient (BW=256GB/S), BigLittle will perform worse than both of the Bigs (S3, S4) as shown in Fig. 13(c), and can be verified by the jobs analysis in Fig. 13(a). However, BigLittle has smaller BW requirement because of its smaller sub-accelerator size, as shown in Fig. 13(b). Therefore, as shown in Fig. 13(c), when the BW is limited (BW=1), BigLittle (S5) with the least amount of resources reaches the best performance. This observation shows that in a multi-core heterogeneous system, in addition to making the compute cores more powerful (adding more compute resources to them), striking the balance between each cores is another key consideration for boosting the performance.

F. Flexible Accelerator

In this experiment, we consider accelerators where the PE array dimensions are configurable, such as FPGAs [16],

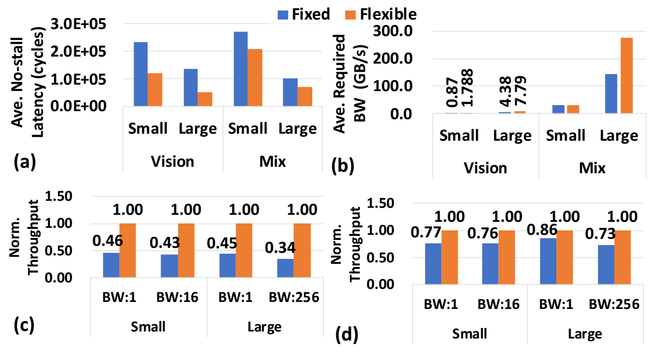


Fig. 14: Jobs analysis of the averaged (a) per-job no-stall latency and (b) required BW of fixed and flexible PEs arrays. Performance evaluation of MAGMA with fixed or flexible PEs array on (c) Vision and (d) Mix. Throughput values are normalized by the value of flexible accelerator.

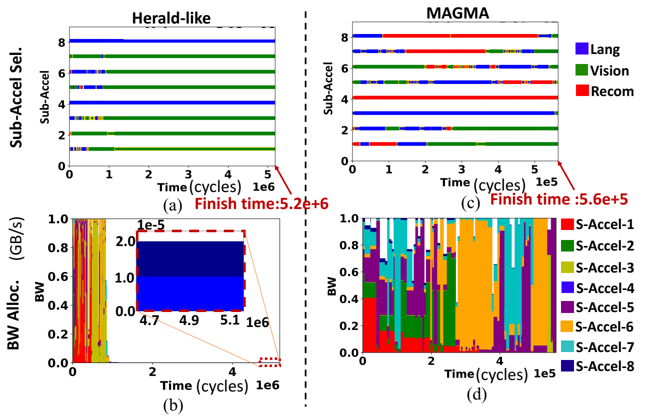


Fig. 15: The visualization of found solution by Herald-like and MAGMA. (a)(c) shows the respective sub-accelerator allocations, and (b)(d) shows the respective BW allocations. (Mix task, S5, BW=1).

CGRA [51], or programmable accelerators [5], [104], [108], and demonstrate their performance by applying mapping found by MAGMA.

Accelerator Configuration. We extend the setting of S1 (Small, fixed) and S3 (Large, fixed) to have flexible accelerators. The number of PEs in the sub-accelerator are

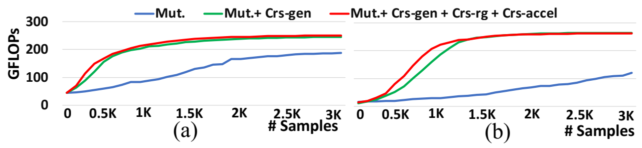


Fig. 16: The convergence curve of MAGMA with three level of genetic operations: Mutation only, Mutation and Crossover-gen, and MAGMA with all four operators on (a) (Vision, S2, BW=16) and (b) (Mix, S3, BW=16). Mutation is the base operator in MAGMA. However, when MAGMA is restricted to only the mutation operator, the sample efficiency decreases seriously. We can see that adding Crossover-gen is essential for algorithm to find good solution within limited samples, and finally, Crossover-rg and Crossover-accel further help approach an optimum faster.

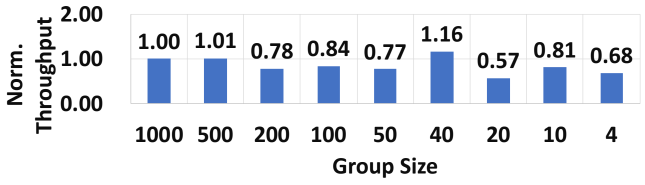


Fig. 17: The reached performance of MAGMA given the same task and setting (Mix, S2, BW=16) with different group sizes. Throughput values are normalized by the value of group size=1000,

fixed (the same as in Table III). However, the *shape* of 2D PE arrays is flexible, that is we can configure the routing among the PEs. This enables the sub-accelerator to run various dataflows or mappings [51]. The maximum size of SLs are fixed as 1KB in each PE, and SGs are fixed as 2MB in each sub-accelerator.

Dataflow Strategy. We pick the dataflow strategy of the sub-accelerator to maximize the utilization of the PEs array. In order to maximize the utilization, we will align the PEs array dimension to be the factor of the the parallelizing dimension of the tile as much as possible. For example if the parallelizing dimension of the tile is (2, 15), which is going to map over the y and x dimension of the PEs array with 16 PEs. The potential PE array shape could be 2×8 while aligning to the factor of y dimension, or 3×5, 5×3, and 1×15 while aligning to the factor of x dimension. We examine these combinations, evaluate their expected latency by the HW cost model, and pick the lowest latency one as our PE array configurations.

Evaluations. From the performance analysis in Fig. 14(a-b), we can observe that for both Vision and Mix tasks, *flexible* outperforms *fixed* in ave. per-job no-stall latency, owing to its ability to maximizing the utilization rate of the PEs array. However, it would also incur higher BW requirement. It is because the flexible mapping we found is to maximize the PE utilization rate, which also increases the number of data to fetch per tile to keep PEs busy.

From all scenario in Fig. 14(c-d), *flexible* outperforms *fixed*.

Table V: The performance of warm-start on (a) Mix, S4, BW=1. (b) The averaged performance across different tasks and different accelerator (S1-S6) under BW=1. All the values are normalized by the values of Trf-100-ep of each columns. Raw (highlighted in orange) is the throughput without warm-start. Trf-0-ep (highlighted in green) is warm-start and before further optimization. Trf-1-ep is warm-start with one epoch of optimization, and likewise for Trf-30-ep. Trf-100-ep (highlighted in blue) represents a full optimization process.

Mix-S4 BW=1	Insts0 (Optim -ized)	Insts1 (Warm -start)	Insts2 (Warm -start)	Insts3 (Warm -start)	Insts4 (Warm -start)	Ave. (Warm -start)
Raw	0.02	0.04	0.02	0.09	0.05	0.03
Trf-0-ep	1.00	0.32	0.60	0.78	0.58	0.51
Trf-1-ep	1.00	0.43	0.73	0.96	0.88	0.68
Trf-30-ep	1.00	0.99	0.98	0.99	0.99	0.99
Trf-100-ep	1.00	1.00	1.00	1.00	1.00	1.00

(a) Perf. on Mix, S4, BW=1

Averaged across S1-S6				
BW=1	Mix	Vision	Lang	Rec
Raw	0.02	0.04	0.014	0.004
Trf-0-ep	0.48	0.28	0.52	0.88
Trf-1-ep	0.67	0.40	0.79	0.95
Trf-30-ep	0.97	0.93	0.97	0.99
Trf-100-ep	1.00	1.00	1.00	1.00

(b) Ave. perf, across S1-S6

The results conclude that with flexible accelerators (ASIC of FPGA), we could further increase the accelerator performance without providing additional compute HW resources (PEs) if the accelerators (or sub-accelerators) have configurable PEs array shape.

G. More about MAGMA Algorithm

Analysis of found solutions. To understand the effect of different mapping, we show the detailed sub-accelerator selection and the corresponding BW allocation results of mapping found by two of the mappers, Herald-like and MAGMA. We showcase what is actually happening on the accelerator in an execution duration of one group of jobs in Fig. 15. We found that MAGMA can distribute the BW-intensive jobs (Recommendation, Language) across the runtime to balance the BW requirement (Fig. 15(c-d)). In contrast, Herald-like (Fig. 15(a-b)) tries to use BW intensively at the beginning, which causes BW competition. Finally, it causes longer finish time of a group of jobs comparing to MAGMA.

Warm-start of MAGMA. In the following, we show the usefulness of warm-start technique in MAGMA. In the experiments, we use MAGMA to optimize on a group of jobs, Insts0. Then, we test, and optimize on the other four different group of jobs. Table V(a) shows that by directly applying previous knowledge (Trf-0-ep), we could achieve **16x better performance** than the usual starting points, randomly initialization (Raw). By warm-start followed by one epoch/ step of optimization (Trf-1-ep), we could already receive 93% of the expected performance gain of a full optimization (Trf-100-ep). We execute the same experiment for different types of tasks and for different setting (S1-S6) (Table V(b)). We can observe for BW-intensive tasks, Language and Recommendation, the previous knowledge become more important, and therefore the performance gain from the warm-start become significant. Overall, by warm-start and before further optimization is run (Trf-0-ep), MAGMA can achieve **7.4x** to **152x** better performance than the the usual starting points (Raws).

Ablation Study of Operators. Fig. 16 shows the importance of the four designed operators. We can see that we could achieve the best sampling efficiency when all four operators are included.

Ablation Study of Group Size. Throughout the evaluation, we use the benchmark with a set group size of 100. It establishes a fair comparisons of the performance of different mappers. Also, in practice, group size is often a pre-defined system parameter as the formulated benchmark. However, a larger (or smaller) group size is also valid. We execute a group size sweep in Fig. 17 using MAGMA algorithm. It tells that increasing or decreasing the group size does not affect the overall performance drastically. However, a too small group size (e.g., 4) will lead to lower performance.

VII. Related Works

Mapping DNN Jobs on Single Accelerator. Several mappers have been proposed for the problem of mapping a single DNN layer efficiently on an accelerator. These include manual-designed mapping search [58], [84], heuristic-based mapping search [68], [89], [101], [106] and optimization/ML methods [31], [35], [43], [44], [91]. These works fall within the local mapping phase within individual accelerator cores.

Multi-tenant Mapping for DNN Accelerators. Prophet [10] builds a runtime prediction model for multi-tenancy on GPU. AI-MT [3] develops a heuristic for DNN job mapping for multi-PE arrays. Prema [15] explores preemptive multi-tenancy on a NPU. Herald [49] and Planaria [39] use manual-designed mapping for assigning jobs to sub-accelerators or reconfigurable PEs array. SCARL [14] utilizes RL for the mapping problem. In this work, we target multi-DNNs mapping and compared MAGMA against prior arts [3], [49], black-box optimizations [28], [32], [33], [45], [73], and RLs [63], [81].

Multi-tenant Scheduling for CPUs and GPUs. Multi-tenancy has been investigated for decades for multi-tasking on a single CPU and job ordering in CPU clusters [85], [105] or in GPUs [6], [38]. GAs [20], [34], [86], [88], [100], PSO [102], CMA-ES [24], and other optimizations have also been used. Some works leverage RL for jobs ordering over clusters such as DeepRM [59], Decima [60] and Thamsen *et al.* [97]. However, they presume a unified abstraction of the underlying cluster, where heterogeneity of the system is not considered.

VIII. Conclusions

This work presents a mapping optimizer for multi-tenant DNN accelerators. The key takeaways are as follows. (i) Heuristic and optimization methods have been used successfully for the design space of local-mapping (i.e., dataflow design). However, global-mapping forms a new drastically different search space. A new mapper for global mapping is needed for upcoming platforms (Table I). (ii) The search space for this (global-) mapping is extremely enormous. The search sample-efficiency of baseline optimization methods is not sufficient to find optimized solutions. (iii) We develop an optimization algorithm called MAGMA that customizes its exploration momentum and mechanism (genetic operators in this work) for the target search space and outperform the existing related works and other well-known optimization methods.

Acknowledgment

This work was supported by NSF Award 1909900. We thank Suvinay Subramanian for valuable technical discussions and helping us with positioning this work.

References

[1] "Maestro tool," http://maestro.ece.gatech.edu/, 2020.

[2] M. Anderson, B. Chen, S. Chen, S. Deng, J. Fix, M. Gschwind, A. Kalaiah, C. Kim, J. Lee, J. Liang *et al.*, "First-generation inference accelerator deployment at facebook," *arXiv preprint arXiv:2107.04140*, 2021.

[3] E. Baek, D. Kwon, and J. Kim, "A multi-neural network acceleration architecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 940–953.

[4] P. Bajpai and M. Kumar, "Genetic algorithm–an approach to solve global optimization problems," *Indian Journal of computer science and engineering*, vol. 1, no. 3, pp. 199–206, 2010.

[5] S. Bang, J. Wang, Z. Li, C. Gao, Y. Kim, Q. Dong, Y.-P. Chen, L. Fick, X. Sun, R. Dreslinski *et al.*, "14.7 a 288μw programmable deep-learning processor with 270kb on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 250–251.

[6] T. Beisel, T. Wiersema, C. Plessl, and A. Brinkmann, "Cooperative multitasking for heterogeneous accelerators in the linux completely fair scheduler," in *ASAP 2011-22nd IEEE International Conference on Application-specific Systems, Architectures and Processors*. IEEE, 2011, pp. 223–226.

[7] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[8] Cerebras, "Cerebras cs-1," 2020. [Online]. Available: https://www.cerebras.net/

[9] S. Chang, J. Yang, J. Choi, and N. Kwak, "Genetic-gated networks for deep reinforcement learning." in *NeurIPS*, 2018, pp. 1754–1763.

[10] 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 Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 17–32.

[11] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *International Symposium on Computer Architecture (ISCA)*, 2016.

[12] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[13] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[14] M. Cheong, H. Lee, I. Yeom, and H. Woo, "Scarl: attentive reinforcement learning-based scheduling in a multi-resource heterogeneous cluster," *IEEE Access*, vol. 7, pp. 153 432–153 444, 2019.

[15] Y. Choi and M. Rhu, "Prema: A predictive multi-task scheduling algorithm for preemptible neural processing units," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 220–233.

[16] E. S. Chung, J. Fowers, K. Ovtcharov, M. Papamichael, A. M. Caulfield, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, M. Abeydeera, L. Adams, H. Angepat, C. Boehn, D. Chiou, O. Firestein, A. Forin, K. S. Gatlin, M. Ghandi, S. Heil, K. Holohan, A. E. Husseini, T. Juhász, K. Kagi, R. Kovvuri, S. Lanka, F. van Megen, D. Mukhortov, P. Patel, B. Perez, A. Rapsang, S. K. Reinhardt, B. Rouhani, A. Sapek, R. Seera, S. Shekar, B. Sridharan, G. Weisz, L. Woods, P. Y. Xiao, D. Zhang, R. Zhao, and D. Burger, "Serving DNNs in Real Time at Datacenter Scale with Project Brainwave," *IEEE Micro*, vol. 38, no. 2, pp. 8–20, 2018. [Online]. Available: https://doi.org/10.1109/MM.2018.022071131

[17] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[18] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[19] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," *arXiv preprint arXiv:1712.06560*, 2017.

[20] R. C. Corrêa, A. Ferreira, and P. Rebreyend, "Scheduling multiprocessor tasks with genetic algorithms," *IEEE Transactions on Parallel and Distributed systems*, vol. 10, no. 8, pp. 825–837, 1999.

[21] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[23] H. Du, S. Leng, F. Wu, X. Chen, and S. Mao, "A new vehicular fog computing architecture for cooperative sensing of autonomous driving," *IEEE Access*, vol. 8, pp. 10 997–11 006, 2020.

[24] G. Emadi, A. M. Rahmani, and H. Shahhoseini, "Task scheduling algorithm using covariance matrix adaptation evolution strategy (cma-es) in cloud computing," *Journal of Advances in Computer Engineering and Technology*, vol. 3, no. 3, pp. 135–144, 2017.

[25] F. Fu, Y. Kang, Z. Zhang, F. R. Yu, and T. Wu, "Soft actor-critic drl for live transcoding and streaming in vehicular fog computing-enabled iov," *IEEE Internet of Things Journal*, 2020.

[26] Google, "Cloud tpu," 2020. [Online]. Available: https://cloud.google.com/tpu/

[27] U. Gupta, S. Hsia, V. Saraph, X. Wang, B. Reagen, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference," *arXiv preprint arXiv:2001.02772*, 2020.

[28] N. Hansen, "The cma evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[31] K. Hegde, P.-A. Tsai, S. Huang, V. Chandra, A. Parashar, and C. W. Fletcher, "Mind mappings: Enabling efficient algorithm-accelerator mapping space search extended abstract."

[32] M. Hellwig and H.-G. Beyer, "Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound-the pccmsa-es," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 26–36.

[33] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[34] E. S. Hou, N. Ansari, and H. Ren, "A genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed systems*, vol. 5, no. 2, pp. 113–120, 1994.

[35] Q. Huang, M. Kang, G. Dinh, T. Norell, A. Kalaiah, J. Demmel, J. Wawrzynek, and Y. S. Shao, "Cosa: Scheduling by constrained optimization for spatial accelerators," *arXiv preprint arXiv:2105.01898*, 2021.

[36] R. Hwang, T. Kim, Y. Kwon, and M. Rhu, “Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations,” *arXiv preprint arXiv:2005.05968*, 2020.

[37] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[38] W. Joo and D. Shin, “Resource-constrained spatial multi-tasking for embedded gpu,” in *2014 IEEE International Conference on Consumer Electronics (ICCE).* IEEE, 2014, pp. 339–340.

[39] S. G. B. H. A. Joon, K. Kim, S. K. B. R. Yatham, N. Alla, H. Sharma, M. Alian, E. Ebrahimi, N. S. Kim, C. Young, and H. Esmaeilzadeh, “Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks.”

[40] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, “A domain-specific supercomputer for training deep neural networks,” *Commun. ACM*, vol. 63, no. 7, p. 67–78, Jun. 2020. [Online]. Available: https://doi.org/10.1145/3360307

[41] 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.

[42] H. Jun, J. Cho, K. Lee, H.-Y. Son, K. Kim, H. Jin, and K. Kim, “Hbm (high bandwidth memory) dram technology and architecture,” in *2017 IEEE International Memory Workshop (IMW).* IEEE, 2017, pp. 1–4.

[43] S.-C. Kao and T. Krishna, “Confuciux: Autonomous hardware resource assignment for dnn accelerators using reinforcement learning,” in *2020 53th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE, 2020, pp. 1–14.

[44] S.-C. Kao and T. Krishna, “Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD).* IEEE, 2020, pp. 1–9.

[45] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[46] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, “Ctrl: A conditional transformer language model for controllable generation,” *arXiv preprint arXiv:1909.05858*, 2019.

[47] N. Kitaev, Ł. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” *arXiv preprint arXiv:2001.04451*, 2020.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[49] H. Kwon, L. Lai, T. Krishna, and V. Chandra, “Herald: Optimizing heterogeneous dnn accelerators for edge devices,” *arXiv preprint arXiv:1909.07437*, 2019.

[50] H. Kwon, M. Pellauer, and T. Krishna, “An analytic model for cost-benefit analysis of dataflows in dnn accelerators,” *arXiv preprint arXiv:1805.02566*, 2018.

[51] H. Kwon, A. Samajdar, and T. Krishna, “Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects,” *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[52] G. Lample and A. Conneau, “Cross-lingual language model pretraining,” *arXiv preprint arXiv:1901.07291*, 2019.

[53] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab, “Flaubert: Unsupervised language model pre-training for french,” *arXiv preprint arXiv:1912.05372*, 2019.

[54] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

[55] C. Li, S. Yang, and T. T. Nguyen, “A self-learning particle swarm optimizer for global optimization problems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 627–646, 2011.

[56] X. Li, Y. Qin, H. Zhou, D. Chen, S. Yang, and Z. Zhang, “An intelligent adaptive algorithm for servers balancing and tasks scheduling over mobile fog computing networks,” *Wireless Communications and Mobile Computing*, vol. 2020, 2020.

[57] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[58] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, “Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks,” in *International Symposium on High Performance Computer Architecture (HPCA)*, 2017.

[59] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource management with deep reinforcement learning,” in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 50–56.

[60] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, “Learning scheduling algorithms for data processing clusters,” in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 270–288.

[61] L. Martin, B. Muller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. de la Clergerie, D. Seddah, and B. Sagot, “Camembert: a tasty french language model,” *arXiv preprint arXiv:1911.03894*, 2019.

[62] Micron, “Ddr5 sdram,” 2020. [Online]. Available: https://www.micron.com/products/dram/ddr5-sdram

[63] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[64] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Intelligent resource allocation in dynamic fog computing environments," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*. IEEE, 2019, pp. 1–7.

[65] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini *et al.*, "Deep learning recommendation model for personalization and recommendation systems," *arXiv preprint arXiv:1906.00091*, 2019.

[66] NVIDIA, "NVIDIA Deep Learning Accelerator (NVDLA)," https://nvldla.org, 2018.

[67] NVIDIA, "Nvidia volta, tensor core gpu architecture," 2020. [Online]. Available: https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/

[68] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2019, pp. 304–315.

[69] 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.

[70] PCI-SIG, "Pce spec," 2020. [Online]. Available: https://pcisig.com/newsroom

[71] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[72] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[73] K. V. Price, "Differential evolution," in *Handbook of Optimization*. Springer, 2013, pp. 187–214.

[74] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[75] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[76] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. frommann-holzbog, stuttgart, 1973," *Step-Size Adaptation Based on Non-Local Use of Selection Information. In PPSN3*, 1994.

[77] D. Richins, D. Doshi, M. Blackmore, A. T. Nair, N. Pathapati, A. Patel, B. Daguman, D. Dobrijalowski, R. Illikkal, K. Long *et al.*, "Missing the forest for the trees: End-to-end ai application performance in edge data centers," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 515–528.

[78] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[79] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[80] E. Schkufza, R. Sharma, and A. Aiken, "Stochastic superoptimization," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 305–316, 2013.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[82] Y. S. Shao *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *MICRO*, 2019, pp. 14–27.

[83] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: a gpu cluster engine for accelerating dnn-based video analysis," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 322–337.

[84] Y. Shen, M. Ferdman, and P. Milder, "Maximizing cnn accelerator efficiency through resource partitioning," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 535–547.

[85] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya, "Libra: a computational economy-based job scheduling system for clusters," *Software: Practice and Experience*, vol. 34, no. 6, pp. 573–590, 2004.

[86] P. Shroff, D. W. Watson, N. S. Flann, and R. F. Freund, "Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments," in *5th Heterogeneous Computing Workshop (HCW'96)*, 1996, pp. 98–117.

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

[88] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," in *5th IEEE heterogeneous computing workshop (HCW'96)*, 1996, pp. 86–97.

[89] A. Stoutchinin, F. Conti, and L. Benini, "Optimally scheduling cnn convolutions for efficient memory access," *arXiv preprint arXiv:1902.01492*, 2019.

[90] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[91] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, and Y. Cao, "Throughput-optimized opencl-based fpga accelerator for large-scale convolutional neural networks," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 16–25.

[92] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[93] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[94] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[95] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," *arXiv preprint arXiv:2011.04006*, 2020.

[96] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *arXiv preprint arXiv:2009.06732*, 2020.

[97] L. Thamsen, B. Rabier, F. Schmidt, T. Renner, and O. Kao, "Scheduling recurring distributed dataflow jobs based on resource utilization and interference," in *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2017, pp. 145–152.

[98] Transcend, "The tranfer rate of ddr1, ddr2, ddr3, and ddr4," 2020. [Online]. Available: https://www.transcend-info.com/Support/FAQ-292

[99] N. Vasilache, O. Zinenko, T. Theodoridis, P. Goyal, Z. DeVito, W. S. Moses, S. Verdoolaege, A. Adams, and A. Cohen, "Tensor Comprehensions: Framework-Agnostic High-Performance Machine Learning Abstractions," *arXiv preprint arXiv:1802.04730*, 2018.

[100] L. Wang, H. J. Siegel, and V. P. Roychowdhury, "A genetic-algorithm-based approach for task matching and scheduling in heterogeneous computing environments," in *Proc. Heterogeneous Computing Workshop*, 1996, pp. 72–85.

[102] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *2010 International Conference on Computational Intelligence and Security*. IEEE, 2010, pp. 184–188.

[101] X. Wei *et al.*, "Automated systolic array architecture synthesis for high throughput cnn inference on fpgas," in *DAC*, 2017, pp. 1–6.

[103] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[104] S. Yin, P. Ouyang, S. Zheng, D. Song, X. Li, L. Liu, and S. Wei, "A 141 uw, 2.46 pj/neuron binarized convolutional neural network based self-learning speech recognition processor in 28nm cmos," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 139–140.

[105] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Job scheduling for multi-user mapreduce clusters," Technical Report UCB/EECS-2009-55, EECS Department, University of California . . . , Tech. Rep., 2009.

[106] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[107] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[108] S. Zheng, P. Ouyang, D. Song, X. Li, L. Liu, S. Wei, and S. Yin, "An ultra-low power binarized convolutional neural network-based speech recognition processor with on-chip self-learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 12, pp. 4648–4661, 2019.

[109] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 5941–5948.

[110] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[111] X. Zhu, S. Chen, S. Chen, and G. Yang, "Energy and delay co-aware computation offloading with deep learning in fog computing networks," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp. 1–6.