CARBONDIS: Carbon-Aware DNN Inference Scheduling on Heterogeneous GPUs

Seyed Morteza Nabavinejad[1], Weiwei Jia[2], Shubbhi Taneja[1], Tian Guo[1]
[1]Worcester Polytechnic Institute [2]University of Rhode Island

***Abstract*—Deploying deep neural network (DNN) inference applications in data centers and clouds to empower various services is on the increase. The significant power consumption of these applications contributes to the carbon emissions of underlying infrastructures. Therefore, minimizing the carbon emissions of these applications leads to a reduced carbon footprint of data centers and clouds. This paper introduces CARBONDIS, a carbon-aware inference scheduler designed to maintain the latency of DNN inference applications while minimizing their carbon emissions by leveraging heterogeneous GPUs. CARBONDIS considers an inference architecture where high-end and low-end GPUs are used in tandem to serve inference jobs. By leveraging the varying computing capability and power consumption of heterogeneous GPUs, CARBONDIS can effectively balance the performance and carbon emissions. Evaluation using three types of GPUs, 10 DNN models, and real-world carbon intensity traces show that CARBONDIS can find a trade-off between carbon footprint and latency of the jobs and reduce the carbon emissions by 16% compared with a performance-centric approach.**

***Index Terms*—DNN inference, scheduling, GPU heterogeneity, carbon emissions**

I. INTRODUCTION

The proliferation of DNN inference applications has prompted extensive research in both academia and industry investigating various aspects of such applications. Traditionally, latency has been the focus of many studies, which emphasize maintaining the latency of applications below a specific latency constraint, e.g., determined in service level agreement (SLA). Within the category of latency-focused research, various objectives have been explored, such as throughout [1], [2], energy [3], [4], and accuracy [5], [6]. However, despite the growing prevalence of DNN inference workloads in data centers and clouds [7], much less research exists that delves into the sustainable impact of these jobs. Prior works based on sustainability tend to focus only on reducing carbon emissions for energy-hungry applications, such as deep learning training [8]–[10]. Further, a recent vision study argues that energy consumption and carbon emission are not always positively correlated when using green energy [11]. As such, there has been a noticeable interest from the community in understanding the role of carbon emissions on modern computing systems and infrastructures [12], [13].

To bridge the gap between research directions focusing on DNN inference systems and sustainability concerns of data centers and clouds, as well as contributing to the emerging interest in the impact of carbon emissions on the design of computing systems, in this work we study the DNN inference jobs deployed on heterogeneous GPUs concerning their latency requirements and carbon emissions. Two main challenges of designing such a system are as follows: 1) Heterogeneous GPUs have varying computing power, memory capacity, energy efficiency, and monetary cost. Therefore, leveraging a combination of them wisely can help to improve the latency and carbon emissions of DNN inference jobs. 2) The carbon intensity of the power grid, which is used to power such heterogeneous platforms, directly contributes to the carbon emissions of the jobs. The carbon intensity is an input to this problem and we cannot change it. We can only consider it in the decision-making process.

To address the challenges, we introduce CARBONDIS, a carbon-aware scheduling approach for DNN inference jobs in heterogeneous GPU platforms. A key insight of CARBONDIS design is leveraging the difference in various GPU types regarding idle and active energy consumption and its direct impact on overall carbon emissions. To handle the varying carbon intensity of the power grid, CARBONDIS leverages a carbon intensity ratio to determine the low and high carbon intensity periods and make scheduling decisions accordingly. Finally, to handle dynamic inference workloads that often have strict latency requirements, we design CARBONDIS to be as lightweight as possible.

Our main contributions are as follows:

- We study the impact of resource heterogeneity on the carbon emissions and latency of DNN inference jobs. Through experimentation, we demonstrate the promise of using heterogeneous GPUs to reduce the overall carbon emissions for these jobs while meeting latency constraints.
- We introduce CARBONDIS, a lightweight approach for scheduling DNN inference jobs on heterogeneous GPU platforms. By monitoring the carbon intensity of the power grid and estimating the latency of jobs's requests, CARBONDIS schedules the requests on low-end and high-end GPUs.
- We conduct trace-driven simulations to evaluate the performance of CARBONDIS and compare it against other performance-centric approaches. Our results show that CARBONDIS can reduce the carbon emissions by as much as 16% while meeting the latency constraint of the jobs.

The rest of the paper is organized as follows: we discuss the motivation behind our study and present background information in Section II. The problem formulation is presented

in Section III and our proposed approach is introduced in Section IV. We evaluate our approach and present results in Section V. Finally, we summarize the related works in Section VI and conclude the paper in Section VII.

II. Background and Motivation

In this section, we study the fluctuation in carbon intensity of the power grid to emphasize the importance of considering this fluctuation in efforts to reduce carbon emissions. Furthermore, through preliminary experiments, we show the benefit of heterogeneous GPUs in decreasing the carbon footprint of DNN inference jobs in the presence of a latency constraint.

A. Carbon Intensity Variation

The carbon intensity metric shows the environmental impact of electricity generation, measured as the amount of carbon dioxide (CO2) emitted per kilowatt-hour (kWh) of electricity produced. Fossil fuel-based electricity has a high carbon footprint, while renewable sources like wind, hydroelectric, or solar exhibit minimal to zero carbon emissions, resulting in negligible carbon intensity values. The power grid electricity is typically a mixture of renewable and non-renewable sources. Non-renewable sources are used to balance supply and demand, considering the intermittent nature of renewable sources like solar and wind [14]. This dynamic blend of power sources results in fluctuations in the carbon intensity of the power grid over time and across different regions.

To show the fluctuation in carbon intensity, we use three traces from different regions from ElectriciyMap [15] that belong to 2022: Australia-NSW, Canada-ON, and USA-CAL. In these traces, the average carbon intensity of electricity generated for that region is available for one year in half-hour time frames, i.e., 48 entries per day (some entries are missing, so the exact number is a bit lower). The average carbon intensity for one month (January) for these three traces is depicted in Figure 1. The significant carbon intensity variation, both within and across regions, emphasizes the importance of considering this variation in the design of any carbon-aware system.

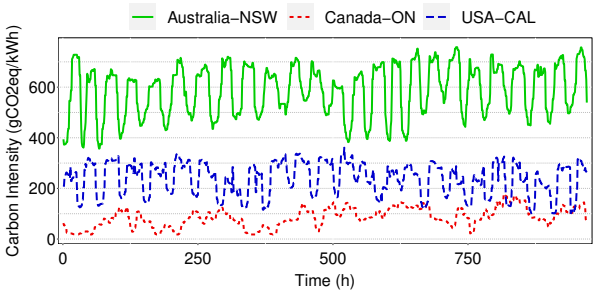


Fig. 1: Temporal carbon intensity variation of different regions. The carbon intensity varies both temporally and spatially and exhibits different magnitudes.

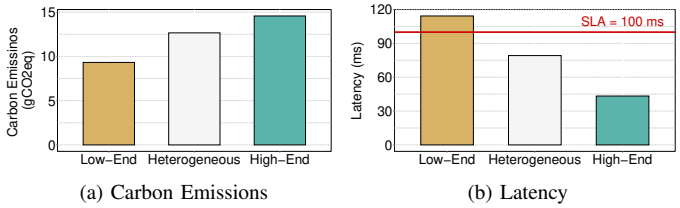
B. Impact of Heterogeneity on Carbon Emissions and Latency

To understand the benefits of using heterogeneous GPUs, we conduct preliminary experiments. We consider a low-end P4 GPU, and a high-end A100 GPU. The idle power of P4 is 25W, while for A100 it is 55W. It means that during idle times, the energy consumption, and hence, carbon emissions of A100 is more than twice that of P4. On the other hand, A100 is more energy-efficient than P4. It means that it can process the same request with lower energy consumption. In the experiments, we consider a DNN inference job where its DNN model is MobileNetV3-Large, and its input workload consists of 20000 requests with varying batch sizes and arrival times. The average inter-arrival time of requests is 200 ms. It means there would be times between requests when the GPUs would be idle. The latency requirement of this job is 100 ms, which means that the average latency of the requests should be less than that. We also consider the US-CAL-2022 trace as input for carbon intensity. More details on GPUs, input workload, and carbon intensity trace will be provided in the upcoming sections of the paper.

We conduct three experiments: 1) processing the entire input workload with a low-end GPU, 2) using a high-end GPU, and 3) dividing the workload evenly between low-end and high-end GPUs. The carbon emissions and latency results are shown in Figure 2. The low-end GPU achieves the lowest carbon emissions (9.32 gCO2eq) but fails to meet the latency requirement, indicated by the horizontal line in Figure 2b. In contrast, the high-end GPU meets the latency constraint but has the highest carbon emissions (14.56 gCO2eq) due to idle time and its high idle power, consistent with prior findings on low GPU utilization for SLA-compliant deep learning jobs [16], [17]. The heterogeneous platform balances these trade-offs. It satisfies the latency constraint while avoiding excessive reductions below it, resulting in lower emissions than the high-end GPU (12.65 gCO2eq). These findings suggest that properly leveraging a heterogeneous platform can reduce carbon emissions while meeting latency requirements.

III. Problem Formulation

Table I lists the notations used throughout the paper. Assume we have a set of latency-sensitive DNN inference jobs N, being deployed on a set of GPUs. Each job, consisting of a series of requests arriving at different times, is characterized



(a) Carbon Emissions (b) Latency

Fig. 2: Impact of heterogeneity on carbon emissions and latency of a DNN inference job. Using heterogenous GPUs can reduce carbon emission while meeting SLA.

by the DNN model, the average inter-arrival time of requests, and the average batch size of requests. The performance of a job is assessed by Service Level Agreement (SLA), which is defined as the high-percentile latency of the requests, similar to prior works on inference serving [1], [18], [19]. For inference servers, we consider two types of GPU: (i) a low-end GPU (e.g., Nvidia P4) that has low power consumption, but low computing power as well and processes the input batches slowly; (ii) a high-end GPU (e.g., NVIDIA A100) that consumes more power than the low-end one, but is faster and can reduce the latency compared with the low-end one.

The problem is *what type of GPU to use* and *how to schedule inference jobs' requests on GPUs* to minimize carbon emissions while meeting the SLA of the jobs.

For request r in job j, we represent its latency and power consumption on different GPU types, gt, as follows: $L^{gt}_{j,r}$, $P^{gt}_{j,r}$. We also have a binary decision variable $K_{j,r}$ that denotes if the request is scheduled on a low-end GPU ($K_{j,r} = 1$) or a high-end one ($K_{j,r} = 0$). We use CI_t to denote the carbon intensity of the power source, e.g., the power grid, at each time t.

If we denote the input workload of job j as W_j, then for each request in the workload, its energy consumption on each GPU type $E^{gt}_{j,r}$ can be calculated as follows:

$$E^{gt}_{j,r} = L^{gt}_{j,r} \times P^{gt}_{j,r} \quad , \quad \forall r \in W_j \tag{1}$$

Based on the energy consumption and carbon intensity, we can calculate the *active carbon emission* of each request, defined as carbon emissions from processing that request. Therefore, for each request, its carbon emissions, $CE_{j,r}$, and latency ,$L_{j,r}$, are as follows, based on which GPU is selected for the processing of the request (e.g., $gt1$ or $gt2$):

$$\begin{aligned} CE_{j,r} &= K_{j,r} \times E^{gt1}_{j,r} \times CI_t + (1 - K_{j,r}) \times E^{gt2}_{j,r} \times CI_t \\ L_{j,r} &= K_{j,r} \times L^{gt1}_{j,r} + (1 - K_{j,r}) \times L^{gt2}_{j,r} \end{aligned} \tag{2}$$

And for each job, the Tail Latency, TL, at m^{th} percentile is:

$$TL_j = TL(m^{th}, \{\forall r \in W_j : L_{j,r}\}) \quad , \quad \forall j \in N \tag{3}$$

TABLE I: List of Notations Used in this Paper

Notation	Definition
N	Set of Jobs
GS	Set of all the GPUs
SLA	Service Level Agreement
gt	GPU Type
g	GPU
IP_g	Idle Power of GPU *g*
ICE	Idle Carbon Emission of GPU
CE	Carbon Emission of Request
L	Latency of Request
AT	Arrival Time of Request
EL	Estimated Latency of Request
CT	Current Time
EXFT	Expected Finish Time of Request
ESFT	Estimated Finish Time of Request
CIT	Carbon Intensity Threshold
CIR	Carbon Intensity Ratio
ACI	Average Carbon Intensity
CCI	Current Carbon Intensity

In addition to active carbon emission, we also consider *idle carbon emission* which is defined as the carbon emissions of GPUs during their idle times, i.e., between the arrival of different requests. To this end, we denote the set of GPUs (and not GPU types), by GS. For each g in GS, we use $idle^g_t$ to denote whether g has been idle in time t ($idle^g_t = 1$) or not ($idle^g_t = 0$). Also, IP_g denotes idle power, i.e., the power consumed by GPU when it is idle and not processing any input. For a long time frame, T, the idle carbon emission, ICE_g of each GPU can be calculated as follows:

$$ICE_g = \sum_{t=1}^{T} (idle^g_t \times IP_g \times CI_t), \forall g \in GS. \tag{4}$$

The problem is formulated as following:

$$\begin{aligned} Minimize & \sum_{j \in N} \sum_{r \in W_j} CE_{j,r} + \sum_{g \in GS} ICE_g \quad , \\ S.T.: & \quad TL_j < SLA_j \quad , \quad \forall j \in N \end{aligned} \tag{5}$$

where the objective is to minimize the carbon emissions, considering both the active carbon emissions and the idle carbon emissions, and the constraint is to maintain the tail latency of the jobs according to the SLA.

IV. CarbonDIS Design

Our approach, CarbonDIS, aims to minimize the carbon emissions of DNN inference applications deployed on GPU accelerators while maintaining their latency constraints. To this end, it considers the combination of low- and high-end GPUs and uses a carbon-aware scheduling algorithm to schedule the requests on GPUs. The challenge that CarbonDIS faces is which GPU to select for each request considering: 1) its arrival time and latency constraint, 2) the status of other jobs regarding SLA violation, 3) the latency of request on low-end and high-end GPUs, and 4) the carbon intensity of power grid and its impact on carbon emissions of the system.

To address the challenge, CarbonDIS uses a modular design with three main modules: 1) *Carbon Intensity Monitoring Module* that monitors the carbon intensity of the power grid, stores its history, and calculates the metrics needed for the scheduler unit. 2) *Latency Estimation Module* that estimates the latency of requests of jobs and calculates their estimated finish time that is used by the scheduler. 3) *Carbon-Aware Scheduler* that selects the low-end or high-end GPU for processing a request based on information from other modules.

Figure 3 shows a high-level overview of CarbonDIS and how its modules interact with inputs, heterogeneous GPU platform, and each other. Pseudo-code of CarbonDIS is presented in Algorithm 1. In the following, we discuss the CarbonDIS's modules in more detail.

A. Carbon Intensity Monitoring Module

As mentioned in Section II-A, carbon intensity varies over time. Therefore, it is important to determine how much the carbon intensity is low or high at each period, compared with the historical trend. This helps CarbonDIS to make decisions

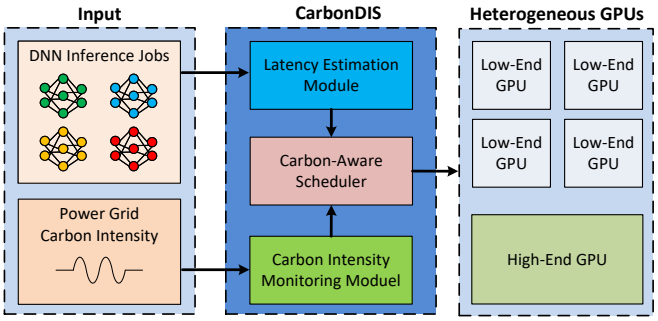


Fig. 3: CARBONDIS design. CARBONDIS takes the input of DNN inference jobs and carbon intensity and makes carbon-aware scheduling decisions to assign incoming requests to different GPUs.

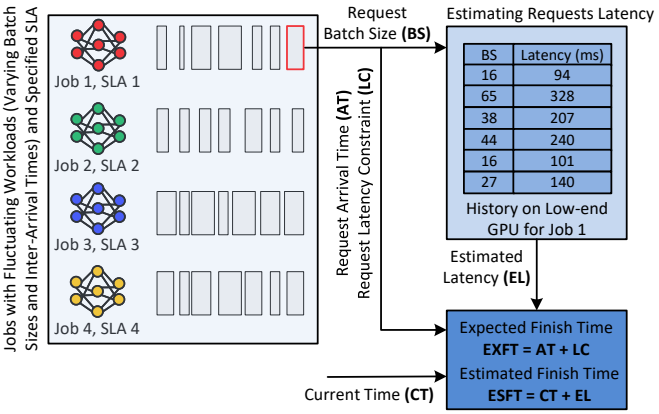
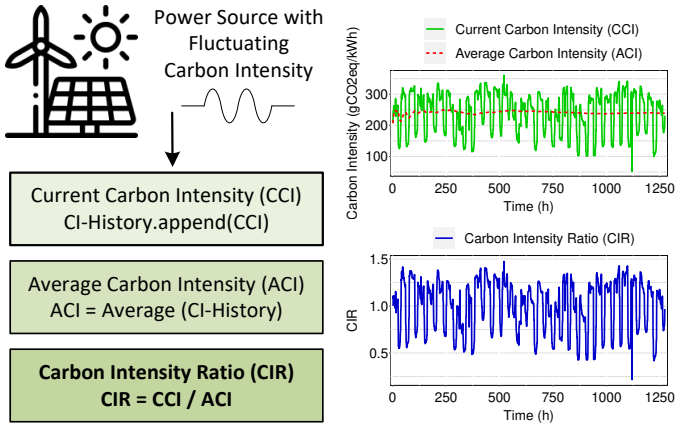


Fig. 5: Flow of latency estimation module



(a) Flow of carbon intensity monitoring module

(b) CCI, ACI, and CIR

Fig. 4: Flow of carbon intensity monitoring moduel, and CCI, ACI, and CIR for US-CAL-2022 trace during month of January.

on how to schedule the requests on GPUs to reduce carbon emissions. To this end, CARBONDIS monitors the variation and calculates the Average Carbon Intensity (ACI) over time. The carbon intensity can be obtained using different tools such as WattTime [20]. Having the carbon intensity at the time of making the decision Current Carbon Intensity (CCI), CARBONDIS divides the CCI by ACI to have a new metric called Carbon Intensity Ratio (CIR) where $CIR = \frac{CCI}{ACI}$. CIR helps CARBONDIS indicate how much the carbon intensity at the current time is lower or higher than the historical average. In Figure 4a, the flow of this module is depicted. Furthermore, in Figure 4b, the CCI, ACI, and CIR for January of trace US-CAL are shown. This figure highlights the CIR variation over time. When explaining the scheduling mechanism, we see how CARBONDIS uses this metric.

B. Latency Estimation Module

CARBONDIS uses a lightweight estimation mechanism to estimate the finish time of the requests and determine if they are going to meet their latency constraint or not. Due to the short nature of inference latency, the estimation mechanism should be simple and very fast, so it would not cause significant time overhead and excessively increase the latency of the requests. To this end, CARBONDIS uses a simple history-based estimation. For past processed requests, it stores their batch size and latency for low-end GPUs. Upon receiving a new request, CARBONDIS calculates the average latency of previous requests with the same batch size and considers that as an estimate for the processing latency of the current request. The overall flow of this module is shown in Figure 5.

C. Carbon-Aware Scheduler

Having the set of jobs, N, CARBONDIS sorts it concerning the number of requests violated their latency constraint in each job (line 7 of Algorithm 1). In this way, the job with the highest number of violations would be at the front of the set. This sorting aims to give higher priority to the job with the highest SLA violations. After sorting, CARBONDIS starts the scheduling from the front of the set. It checks the first job to see if it has a request waiting in its queue. If not, it goes to the next job. If there is a request, CARBONDIS estimates its latency on low-end GPU with the help of the latency estimation module (lines 8-10).

Having the estimated latency (EL), arrival time (AT), latency constraint (LC), and current time (CT), CARBONDIS calculates the expected finish time (EXFT) and estimated finish time (ESFT) of the request as follows:

$$EXFT = AT + LC \quad , \quad ESFT = CT + EL \tag{6}$$

Note that the CT is the time that the request will be executed. CT includes the time the request has been waiting in the queue, so we have $CT \geq AT$. It also obtains the *CIR* from the carbon intensity monitoring module whenever it wants to schedule a request. Having all the information needed, CARBONDIS proceeds to the next step, which is deciding which GPU to schedule the request on.

In the first step, CARBONDIS compares the *ESFT* and *EXFT*. Based on the comparison result, two paths are possible:

Algorithm 1 CARBONDIS

Input: CIT, CCI, CT, N

```
    #Carbon Intensity Monitoring Module
1:  ACI = average (CI-History)
2:  CIR = CCI/ACI
3:  CI-History.append(CCI)
4:  while N is not empty do
5:     sort(N, based on number of violations of jobs)
6:     for j ∈ N do
7:        r = j.getRequest() #getting the request at front of the queue
          #Latency Estimation Module
8:        EL = average (History(j, r.BS))
9:        EXFT = r.AT + j.LC
10:       ESFT = CT + EL
11:       if ESFT > EXFT then # might miss the constraint
12:          if High-end GPU is free then
13:             K = 0    # use high-end GPU
14:          else
15:             K = 1    # use low-end GPU
16:       else if ESFT ≤ EXFT & CIR > CIT then
          # can meet the constraint, but it is a high carbon intensity period
17:          if High-end GPU is free then
18:             K = 0    # use high-end GPU
19:          else
20:             K = 1    # use low-end GPU
21:       else if ESFT ≤ EXFT & CIR ≤ CIT then
          # can meet the constraint, and it is a low carbon intensity period
22:          K = 1    # use low-end GPU
```

1) If $ESFT > EXFT$, it means that the request might not be able to meet its latency constraint on the low-end GPU. Therefore, CARBONDIS checks the high-end one. If the high-end GPU is free, CARBONDIS schedules the request on it. But, if the high-end GPU is not free, CARBONDIS schedules the request on the low-end GPU (lines 11-15). 2) If $ESFT \leq EXFT$, it means that based on the estimation, the request can meet its latency constraint on the low-end GPU. In this case, before scheduling the request on low-end GPU, CARBONDIS checks the *CIR*.

If $CIR > CIT$, where *CIT* is a carbon intensity threshold that is an input for CARBONDIS, it means that the carbon intensity of the power source is relatively high. When the carbon intensity is high, it means that it is beneficial to reduce the energy consumption to reduce carbon emissions. Since the high-end GPU is more energy-efficient than the low-end one, scheduling the request on it would reduce the carbon emissions. So, CARBONDIS tests the high-end GPU to see if it is free. If so, then CARBONDIS would schedule the request on the high-end GPU (lines 16-20). On the other hand, if $CIR \leq CIT$, CARBONDIS directly schedules the request on a low-end GPU, without checking the high-end one (lines 21-22).

One point needs clarification in the design of CARBONDIS. We mentioned that the high-end GPU is more energy-efficient, and hence, it can lead to low carbon emissions So, the question is why not maximize its utilization by assigning requests to it whenever possible? The answer is that if we do so, we may end up scheduling the requests on high-end GPU that are not prone to SLA violations while taking the opportunity from others that need high-end GPU to avoid SLA violations. Hence, the current CARBONDIS scheduling mechanism tries to find a trade-off between carbon emissions and SLA violation. Later in Section V, we see the impact of excessive use of high-end GPU by another approach on SLA violations.

V. EVALUATION

In this section, we evaluate the effectiveness of CARBONDIS in minimizing carbon emissions while adhering to the SLA of jobs. The subsequent sections present results indicating that CARBONDIS successfully utilizes the diversity of GPUs, resulting in reduced carbon emissions when compared to a homogeneous approach. Moreover, its scheduling mechanism designed to be aware of carbon intensity, enables CARBONDIS to trade off carbon emissions with latency to meet the SLA of DNN inference jobs, a capability lacking in an alternative heterogeneous approach.

A. Experiment Setup

We evaluate CARBONDIS using trace-driven simulation with empirically profiled DNN inference data on different GPUs, carbon intensity traces, and workload traces, as described below.

DNN Models and Datasets. To show the applicability of our approach to different models, we use ten pre-trained image classification DNNs with ImageNet dataset [21]. We employ pre-trained models from TensorFlow Hub [22] and all these models can successfully deploy and run on the GPUs used in the experiments.

DNNs Profiling on GPUs. To conduct the experiments, it is essential to profile the latency and power consumption of models on GPUs across various batch sizes. We choose three different GPUs in the experiments: (1) Nvidia P4, (2) Nvidia T4, and (3) Nvidia A100 because these GPUs have varying computing power, memory capacity, and power consumption. The specifications of GPUs are listed in Table II. In this work, we consider P4 and T4 as low-end GPUs, and A100 as the high-end GPU in the experiments.

We perform a comprehensive profiling of each model on P4, T4, and A100 GPUs. The profiling involves processing input batches for each batch size ranging from 1 to 128 for a brief period (100 seconds). During the execution, the power consumption and latency are recorded for each input batch. After the completion of processing, the average latency and average power consumption of all the processed input batches are computed and stored as a profiling entry for that specific batch size. This profiling process is repeated for each combination of models and GPUs (10 models and 3 GPUs, 30 combinations in total). An example of the final profiling data is shown in Table III, illustrating part of the profiling results of the Inception-V3 model on all three GPUs. Similar tables are generated for each DNN model for all the GPUs.

TABLE II: Specifications of GPUs Used in the Experiments

GPU	Architecture	CUDA Core	Tensor Core	Memory	Idle Power
Nvidia P4	Pascal	2560	—	8GB GDDR5	25W
Nvidia T4	Turing	2560	320	16GB GDDR6	30W
Nvidia A100	Amper	6912	432	40GB HBM2e	55W

TABLE III: Example Profiling Data for Inception-V3 Model

Batch Size	P4		T4		A100	
	Latency (ms)	Power (W)	Latency (ms)	Power (W)	Latency (ms)	Power (W)
1	18	81.64	12	73.78	13.89	68.17
2	21	84.32	16	76.34	13. 67	73.63
3	26	85.09	19	80.88	13.70	83.03
4	29	86.70	22	82.59	13.81	90.11
5	32	88.06	26	83.64	14.22	88.68
6	37	88.66	30	84.66	14.35	111.14

Carbon Intensity Trace. We use two carbon intensity traces from the ElectricityMap [15]. The traces provide the half-hourly average carbon intensity of the power grid over time in different regions during the year 2022. Table IV shows more details on the traces. The high value of the coefficient of variation of traces indicates significant fluctuation in carbon intensity over time. Therefore, these traces can help evaluate the performance of CARBONDIS and other approaches regarding considering the carbon intensity fluctuation.

Jobs and Workload Traces. For the main experiments, there are two sets of jobs: Set 1, and Set 2, as depicted in Table V. Each set consists of five jobs and each job has its own DNN model. In both sets, the high-end GPU is the A100. For Set 1, the low-end GPU is P4 and for Set 2, the low-end one is T4. The carbon intensity trace used for Set 1 and Set 2 are US-CAL-2022 and AUS-NWS-2022, respectively. For each job, an input workload is generated consisting of 20000 requests with varying batch sizes. The workload generator introduced in [23] is used for generating the input workloads.

The batch size of requests follows a normal distribution. The mean and standard deviation of batch sizes vary for different jobs. For the inter-arrival time of requests, poison distribution is used, with each job having its specific inter-arrival time distribution. The average inter-arrival time of jobs and their SLA is selected such that there would be enough time between requests (concerning their average latency on low-end GPU and SLA of the job) to avoid excessive violations that make it impossible to meet the SLA of the jobs. On the other hand, they are tight enough to make it challenging to schedule the requests on GPUs. In other words, they are not that relaxed that any scheduling algorithm, regardless of its efficiency, can meet the SLA of all the jobs.

Systems Comparison. We compare the performance of CARBONDIS to the following approaches:

- **HIGHPERF**: This approach is a performance-centric one, that only focuses on improving the tail latency of the jobs and avoiding SLA violation as much as possible. But, it does not consider energy consumption and carbon emissions. It only uses high-end GPUs for each job. For example, if there are 5 jobs, it uses 5 high-end GPUs.

TABLE IV: Carbon Intensity Traces Used in Experiments

Trace	Average Carbon Intensity (ACI)	Standard Deviation	Coefficient of Variation
US-CAL-2022	207.13	69.23	33.42%
AUS-NWS-2022	605.59	83.67	13.82%

- **LOWENERGY**: This method ignores latency constraints and focuses on minimizing energy consumption, resulting in reduced carbon emissions. By ignoring latency, it often fails to meet SLA. We use this approach to provide an upper bound for latency violations and a lower bound for carbon emissions. Unlike HIGHPERF, this method only employs low-end GPUs per job, e.g., 5 low-end GPUs for 5 jobs.
- **DYBATCH**: We implement a variant of the approach described in [31][1] where a fairness-driven scheduler monitors the resource utilization of all the applications and then allocates the time-sharing resource (e.g., high-end GPU in this work) to them so that every application would have a fair share of that resource. Similar to CARBONDIS, the implementation of DYBATCH in our work considers one low-end GPU for each job, and then a high-end one shared between all the jobs. The high-end GPU has the role of a shared resource in the original implementation of the DYBATCH.
- **RANDOM**: Similar to CARBONDIS and DYBATCH, this approach also uses a heterogeneous setup. However, it does not have a specific scheduling algorithm for assigning the requests to high-end GPU. It randomly selects a job and assigns its request to high-end GPU.

B. Experimental Results

1) Tail Latency of Jobs: First, we present the results regarding the latency performance of each approach to see how effective they are in maintaining the SLA of the jobs in each set. The tail latency of each job in each set is presented in Figure 6. As can be seen, both CARBONDIS and HIGHPERF can meet the SLA of all the jobs in each set. However, other approaches fail to meet the SLA for one or more jobs.

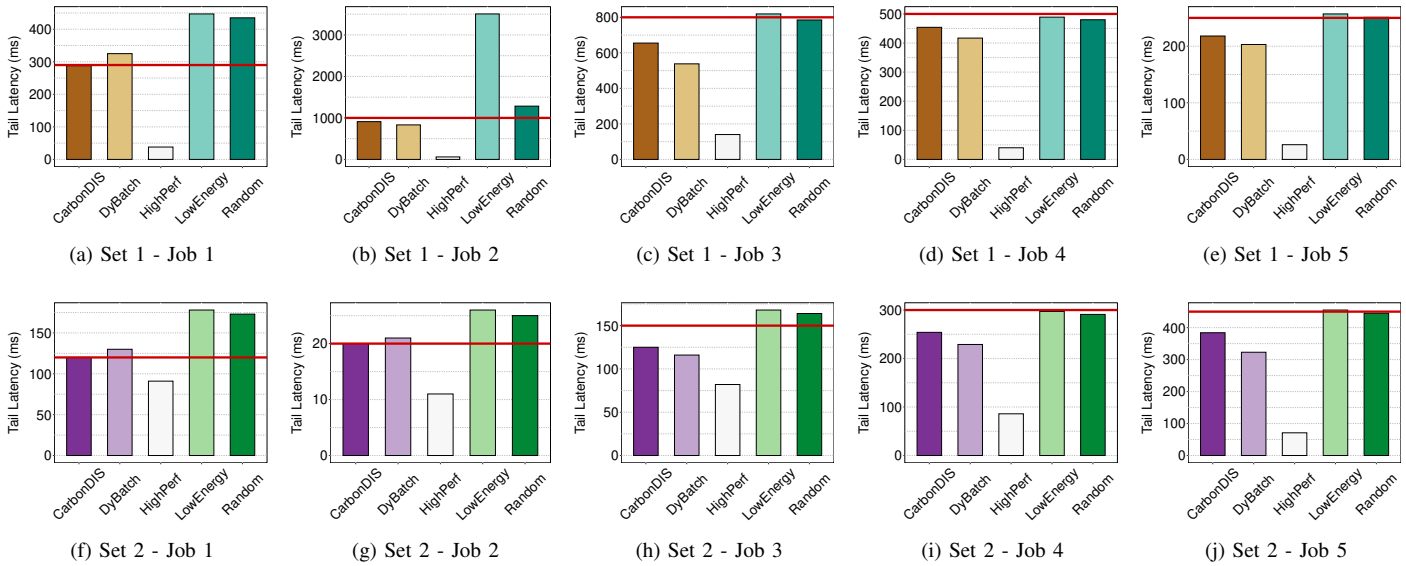
Among the approaches, HIGHPERF has the best performance regarding latency as it only uses high-end GPUs. The 95^{th} tail latency of all the jobs under this approach is significantly lower than the SLA. LOWENERGY, on the other hand, has the worst performance. As it only focuses on energy and carbon emissions by using low-end GPUs, it yields very high tail latency. DYBATCH and RANDOM also fail to meet the SLA for some jobs. For example, DYBATCH fails to meet the SLA for Job 1 in Set 1 and Set 2. These results show that while DYBATCH and RANDOM use heterogenous GPUs similar to CARBONDIS, they fail to meet the SLA due to their scheduling mechanism. To better understand the performance of each approach, we show the cumulative distribution function (CDF) of requests's latency for Job 1 from Set 1 under each approach in Figure 7. We see that for a significant share of requests, the latency resulting by DYBATCH is significantly less than CARBONDIS. We can explain it as follows: when considering the SLA, what is important is the tail latency and not the latency of individual requests. There is no benefit in processing some requests very fast, while the tail latency is

[1]We did not use the batching mechanism in DYBATCH because in this work we assume that the requests are arriving in the system in the form of pre-defined batches.

TABLE V: Specifications of Sets of Jobs Used in the Experiments. For both sets, we use A100 as the high-end GPU.

Set 1 — Low-End GPU: P4 — Carbon Intensity Trace US-CAL-2022					
Job #	DNN Model	Average Inter-Arrival Time (ms)	Average Batch Size	Standard Deviation of Batch Size	95^{th} Tail Latency (SLA) (ms)
1	Efficient-V2-b3 [24]	480	30	20	290
2	Efficient-V2-s [24]	750	50	10	1000
3	Inception-ResNet-V2 [26]	650	35	15	800
4	Inception-V3 [28]	590	65	10	500
5	MobNet-V2 [30]	270	70	25	250

Set 2 — Low-End GPU: T4 — Carbon Intensity Trace AUS-NSW-2022					
Job #	DNN Model	Average Inter-Arrival Time (ms)	Average Batch Size	Standard Deviation of Batch Size	95^{th} Tail Latency (SLA) (ms)
1	MobNet-V3-Large [25]	360	55	30	120
2	MobNet-V3-Small [25]	210	25	10	20
3	NasNet-Mob [27]	480	45	15	150
4	ResNet-V2-50 [29]	540	60	20	300
5	ResNet-V2-152 [29]	720	40	15	450



(a) Set 1 - Job 1 (b) Set 1 - Job 2 (c) Set 1 - Job 3 (d) Set 1 - Job 4 (e) Set 1 - Job 5

(f) Set 2 - Job 1 (g) Set 2 - Job 2 (h) Set 2 - Job 3 (i) Set 2 - Job 4 (j) Set 2 - Job 5

Fig. 6: 95^{th} tail latency of jobs for Set 1 and Set 2. The horizontal line in the plots shows the SLA of each job. A bar passing that line means an SLA violation.

longer than expected. That is exactly the difference between DyBatch and CarbonDIS. DyBatch can significantly reduce the latency for some requests across jobs, way lower than their latency constraint. However, as it cannot meet the SLA for some jobs, that significant reduction is not useful. CarbonDIS, unlike HighPerf and DyBatch that strive to reduce the latency of individual requests as much as possible, spreads out the latency distribution of requests for better SLA guarantees, compared with DyBatch. We also see in the next section how this feature of CarbonDIS helps it reduce the carbon emissions compared with HighPerf. Spreading out the latency distribution for SLA guarantee has been used in previous works as well [32].

2) Carbon Emissions Breakdown: In this section, we delve into the carbon emissions results to understand the environmental impact of each approach. The total carbon emissions, as per Equation (5), comprise active emissions during request processing and idle emissions when GPUs await requests. Figure 8 shows the carbon emissions breakdown of different approaches for each set. As the high-end GPUs are more energy-efficient and HighPerf only uses high-end ones, its active carbon emissions are less than the other approaches in both sets. However, the higher idle power of high-end GPUs leads to higher idle carbon emissions, surpassing the other approaches. Since the high idle carbon emissions of

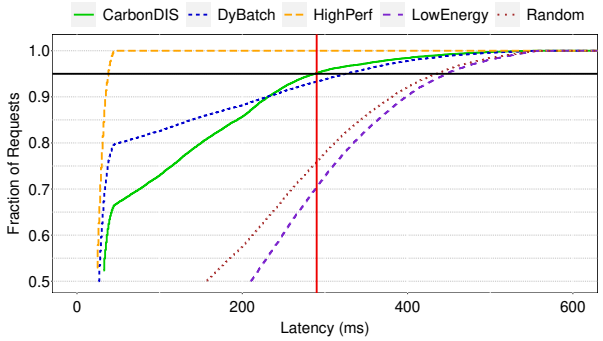


Fig. 7: Request latency CDF for Job 1 in Set 1. We show this CDF as an example to understand the distribution of requests' latency under each approach. The horizontal line shows the 95% of the requests and the vertical line shows the latency constraint. If a curve cuts the vertical line before the horizontal line, it means SLA violation. Among all the approaches, only CarbonDIS and HighPerf can meet the SLA.

HighPerf outweigh its lower active carbon emissions, its total carbon emissions exceed other approaches.

LowEnergy has the lowest carbon emissions, as its idle carbon emissions are very low due to the low idle power of low-end GPUs. Random, while using the same het-

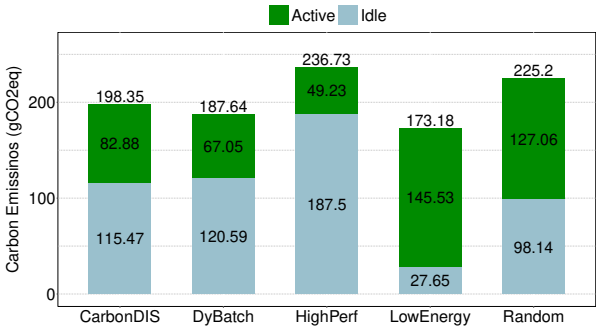
erogeneous GPUs as CARBONDIS and DYBATCH, shows higher carbon emissions compared with them because of its primitive scheduling mechanism. CARBONDIS' carbon emissions stands between HIGHPERF and DYBATCH. It can reduce carbon emissions by 16.21% and 11.22% compared with HIGHPERF in Set 1 and Set 2, respectively. However, it experiences a 5.39% and 4.72% increase in carbon emissions compared to DYBATCH in Sets 1 and 2, respectively. Using the combination of low-end and high-end GPUs allows CARBONDIS to reduce its idle carbon emissions compared with HIGHPERF, resulting in lower total carbon emissions despite its higher active carbon emissions originating from less energy-efficient low-end GPUs. Conversely, compared to DYBATCH, lower usage of high-end GPU leads to higher carbon emissions in CARBONDIS. Note that while CARBONDIS slightly increases carbon emissions compared to DYBATCH, its scheduling mechanism successfully meets the SLA of all jobs in both sets, unlike DYBATCH.

Comparing results between Set 1 and Set 2, CARBONDIS's carbon emissions reduction compared to HIGHPERF is more significant in Set 1. This is primarily due to the higher average inter-arrival time of job requests in Set 1, leading to increased idle carbon emissions for HIGHPERF. Additionally, the higher idle power of T4 compared to P4 contributes to the differences, with the gap between the idle power of T4 and A100 being less than that between P4 and A100. This bridges the gap between idle carbon emissions of CARBONDIS and HIGHPERF.

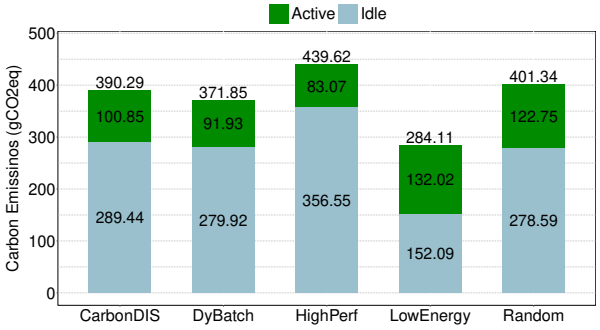
3) High-End GPU Usage Pattern: In this section, we study the usage pattern of high-end GPU by CARBONDIS and DYBATCH, aiming to understand the difference between these approaches that lead to their different latency and carbon emissions performance. Figure 9 indicates the frequency of high-end GPU usage for individual jobs in Sets 1 and 2 by each approach. Notably, DYBATCH consistently exhibits a higher rate of high-end GPU usage across all jobs in both sets compared to CARBONDIS. This behavior is expected, given DYBATCH's carbon-oblivious nature that emphasizes maximizing the time-sharing resource utilization, the high-end GPU in this work, to improve latency. In contrast, CARBONDIS adopts a more balanced approach, considering both carbon intensity and the probability of latency constraint violations. Therefore, its high-end GPU usage is more restrained than DYBATCH. CARBONDIS delays high-end GPU allocation to a request, factoring in the ESFT and EXFT of that request. This occasionally results in the high-end GPU being available for subsequent requests. Moreover, CARBONDIS adjusts usage frequency based on carbon intensity, employing the high-end GPU more during high carbon intensity periods. This stands in contrast to DYBATCH, prioritizing high-end GPU use without considering carbon intensity.

C. Sensitivity Analysis

In this section, we study the impact of CIT on the performance of CARBONDIS. We conduct a set of experiments for Set 1, and in these new experiments, we change the value of CIT from 0.8 to 1.2, with steps of 0.1 ($CIT = \{0.8, 0.9, 1, 1.1, 1.2\}$). The number of high-end GPU usage, the carbon emissions, and the number of violations are presented in Figure 10.

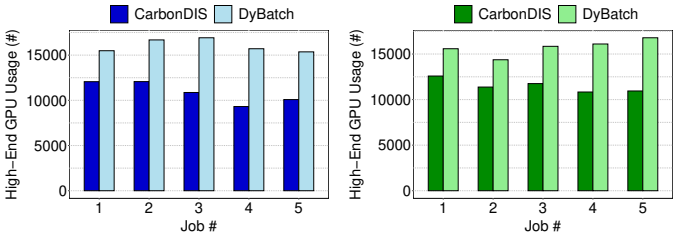


(a) Set 1



(b) Set 2

Fig. 8: Carbon emissions results for different approaches in Sets 1 and 2. The results are broken down to show the share of active and idle states in total carbon emissions.



(a) Set 1 (b) Set 2

Fig. 9: Number of usage of high-end GPU by each job

Increasing the *CIT* results in less usage of high-end GPU per job, as CARBONDIS would less frequently use the high-end GPU when it is estimated that the request can meet its latency constraint on the low-end GPU ($ESFT \leq EXFT$). In other words, higher *CIT* causes the condition at line 16 of Algorithm 1 to be true in less cases. Less usage of high-end GPU affects both the carbon emissions and the number of latency constraint violations, as depicted in Figure 10b and Figure 10c, respectively.

Less usage of high-end GPU, due to higher *CIT*, means that more requests are processed by low-end GPUs. As the energy

efficiency of these GPUs is lower than the high-end ones, they consume more energy for processing the same request. This increase in energy consumption leads to higher carbon emissions. Therefore, we see that the active carbon emissions is on the increase. On the other hand, since the low-end GPUs are now less idle, their idle carbon emissions, and consequently, the total idle carbon emissions are decreasing. Since the increase in active carbon emissions is more significant than the decrease in idle one, the total carbon emissions are on the increase.

Finally, we discuss the impact of *CIT* on the number of latency constraint violations. Increasing the *CIT* results in fewer violations in Job 1, but leads to a slight increase for the rest of the jobs. Because of its tight latency constraint, Job 1 can benefit more from the high-end GPU than other jobs. A low *CIT* leads to more frequent usage of the high-end GPU for requests that do not need it to meet their latency constraint, and hence, less opportunity for requests that need it. However, with the increase in *CIT*, Job 1's requests that are more prone to violation have a higher chance of using the high-end GPU. Other jobs see a slight increase in violations as their high-end GPU use diminishes, impacting their ability to meet latency constraints for a few more requests. Overall, increasing *CIT* correlates with a decrease in constraint violations across all jobs (see last bar group of Figure 10c).

From the experiments, we can conclude that by fine-tuning the *CIT*, we can find a trade-off between carbon emissions and latency constraint violation. *CIT* has a direct relation with carbon emissions and an inverse relation with the number of violations. Larger *CIT* leads to higher carbon emissions, but a lower number of violations.

VI. Related Work

With the increasing adoption of DNN inference in data centers, cloud, and edge environments, research efforts have focused on improving performance through various strategies [23], [33], [34]. These include optimizing DNN architectures via techniques like pruning [35] and quantization [36], adapting architectures for diverse hardware platforms such as FPGA [37] and GPU [38], and developing advanced resource allocation and scheduling mechanisms [18], [34], [39]. Within this landscape, our work aligns with the focus on scheduling and resource management. Previous research has addressed various objectives, such as improving throughput, accuracy, cost efficiency, and power consumption. For example, methods such as P-CNN [40] and Clockwork [18] enhance throughput using batching and latency profiling, while techniques such as Clipper [1] and Cocktail [34] balance accuracy and throughput through dynamic model selection and ensembling. Other works such as Scrooge [41] optimize cost by profiling resource usage, and energy-focused approaches such as BatchDVFS [42] and EAIS [3] combine batching with DVFS to reduce energy consumption while meeting latency constraints.

Despite addressing performance, accuracy, cost, and energy efficiency, these approaches often overlook carbon emissions. Clover [7] is a notable carbon-aware system that reduces emissions through model switching and partitioning; however, it does not account for the heterogeneity of GPUs or input scheduling. To fill this gap, our work introduces the first carbon-aware DNN inference scheduler explicitly designed for heterogeneous GPUs. Using GPU heterogeneity, our approach optimizes scheduling to minimize carbon emissions while maintaining performance, thus addressing an important yet underexplored aspect of sustainable AI.

VII. Conclusion

Sustainability has become an important consideration when designing and operating computing systems. In this work, we studied the sustainable impact of a popular workload, DNN inference based on the key insight of leveraging heterogeneous GPU resources to better match the dynamic workload and temporal-variant carbon intensity. We introduced CarbonDIS, a carbon-aware DNN inference scheduling approach for heterogeneous GPUs. Our experiments indicate that compared to a performance-centric approach, CarbonDIS effectively reduces carbon emissions while meeting SLA requirements for all submitted jobs. Our finding suggests that by considering the workload fluctuation and carbon intensity variation, a scheduling algorithm can optimize the use of heterogeneous GPUs to meet the latency requirement of DNN inference jobs while minimizing carbon emissions. As a future direction, we plan to study the impact of a carbon intensity prediction module on the performance of the scheduling algorithm.

Acknowledgment

This work was supported in part by NSF Grants #2105564, #2236987, #2402383, #2348066, a VMWare grant, and Google Cloud credits. We also thank Electricity Maps for its carbon intensity dataset.

References

[1] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *USENIX NSDI*, 2017, pp. 613–627.

[2] W. Liu, J. Geng, Z. Zhu, Y. Zhao, C. Ji, C. Li, Z. Lian, and X. Zhou, "Ace-sniper: Cloud-edge collaborative scheduling framework with dnn inference latency modeling on heterogeneous devices," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

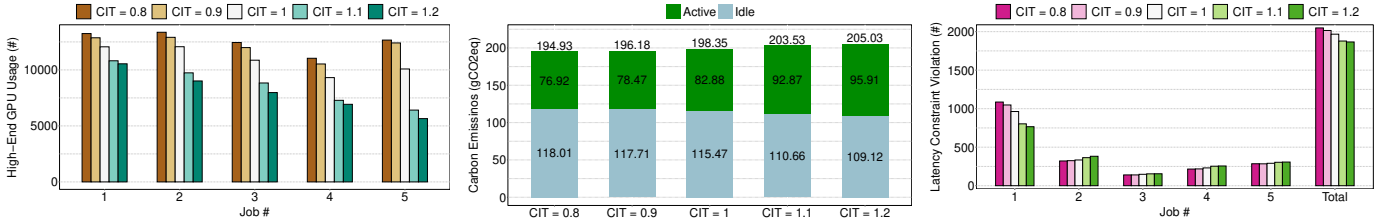
[3] C. Yao, W. Liu, W. Tang, and S. Hu, "Eais: Energy-aware adaptive scheduling for cnn inference on high-performance gpus," *Future Generation Computer Systems*, vol. 130, pp. 253–268, 2022.

[4] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2020.

[5] S. S. Ogden and T. Guo, "Layercake: Efficient inference serving with cloud and mobile resources," in *IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing*, 2023, pp. 191–202.

[6] S. Zhang, W. Cui, Q. Chen, Z. Zhang, Y. Guan, J. Leng, C. Li, and M. Guo, "Pame: Precision-aware multi-exit dnn serving for reducing latencies of batched inferences," in *Proceedings of the 36th ACM International Conference on Supercomputing*, 2022, pp. 1–12.

[7] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Clover: Toward sustainable ai with carbon-aware machine learning inference service," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–15.



(a) Number of High-End GPU Usage Per Job (b) Carbon Emissions (c) Latency Constraint Violations Per Job

Fig. 10: Impact of CIT on the performance of CARBONDIS. We show that CIT can effectively trade-off between carbon emissions and latency constraint violations, providing the flexibility for carbon-aware inference systems like CARBONDIS.

[8] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., 2019, pp. 3645–3650.

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://arxiv.org/pdf/1908.09791.pdf

[10] Y. Zhao and T. Guo, "Carbon-efficient neural architecture search," in *Workshop on Sustainable Computer Systems*, ser. HotCarbon '23, 2023. [Online]. Available: https://doi.org/10.1145/3604930.3605708

[11] N. Bashir, T. Guo, M. Hajiesmaili, D. Irwin, P. Shenoy, R. Sitaraman, A. Souza, and A. Wierman, "Enabling sustainable clouds: The case for virtualizing the energy system," in *ACM Symposium on Cloud Computing (SoCC)*, 2021, p. 350–358.

[12] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, "Let's wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud," in *Middleware'21*, 2021, pp. 260–272.

[13] A. Souza, N. Bashir, J. Murillo, W. Hanafy, Q. Liang, D. Irwin, and P. Shenoy, "Ecovisor: A virtual energy system for carbon-efficient applications," in *ASPLOS'2023*. New York, NY, USA: Association for Computing Machinery, 2023, p. 252–265. [Online]. Available: https://doi.org/10.1145/3575693.3575709

[14] D. Maji, P. Shenoy, and R. K. Sitaraman, "Carboncast: multi-day forecasting of grid carbon intensity," in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2022, pp. 198–207.

[15] E. Map, "Electricity map," Accessed: January 27, 2024. [Online]. Available: https://www.electricitymaps.com

[16] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," in *OSDI'18*, 2018, pp. 595–610.

[17] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of large-scale multi-tenant {GPU} clusters for {DNN} training workloads," in *ATC'19*, 2019, pp. 947–960.

[18] A. Gujarati, R. Karimi, S. Alzayat, W. Hao, A. Kaufmann, Y. Vigfusson, and J. Mace, "Serving dnns like clockwork: Performance predictability from the bottom up," in *OSDI'20*, 2020, pp. 443–462.

[19] J. Wu, L. Wang, Q. Jin, and F. Liu, "Graft: Efficient inference serving for hybrid deep learning with slo guarantees via dnn re-alignment," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[20] WattTime, "WattTime," accessed: 2024-2-3. [Online]. Available: https://watttime.org/

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] TensorFlow, "Tensorflow hub," Accessed: January 27, 2024. [Online]. Available: https://www.tensorflow.org/hub

[23] 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," in *ISCA'20*. IEEE, 2020, pp. 982–995.

[24] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[25] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *IEEE/CVF ICCV*, 2019, pp. 1314–1324.

[26] 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.

[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR'16*, 2016, pp. 2818–2826.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV'16*. Springer, 2016, pp. 630–645.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR'18*, 2018, pp. 4510–4520.

[31] S. Zhang, W. Li, C. Wang, Z. Tari, and A. Y. Zomaya, "Dybatch: Efficient batching and fair scheduling for deep learning inference on time-sharing devices," in *CCGRID'20*. IEEE, 2020, pp. 609–618.

[32] R. S. Kannan, L. Subramanian, A. Raju, J. Ahn, J. Mars, and L. Tang, "Grandslam: Guaranteeing slas for jobs in microservices execution frameworks," in *EuroSys'19*, 2019, pp. 1–16.

[33] Y. Yang, L. Zhao, Y. Li, H. Zhang, J. Li, M. Zhao, X. Chen, and K. Li, "Infless: a native serverless system for low-latency, high-throughput inference," in *ASPLOS'22*, 2022, pp. 768–781.

[34] J. R. Gunasekaran, C. S. Mishra, P. Thinakaran, B. Sharma, M. T. Kandemir, and C. R. Das, "Cocktail: A multidimensional optimization for model serving in cloud," in *USENIX NSDI*, 2022, pp. 1041–1057.

[35] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4876–4883.

[36] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated dvfs and precision control for deep neural networks," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 136–140, 2019.

[37] P. D'Alberto, V. Wu, A. Ng, R. Nimaiyar, E. Delaye, and A. Sirasao, "xdnn: Inference for deep convolutional neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 2, pp. 1–29, 2022.

[38] J. Jeong, S. Baek, and J. Ahn, "Fast and efficient model serving using multi-gpus with direct-host-access," in *Proceedings of the Eighteenth European Conference on Computer Systems*, 2023, pp. 249–265.

[39] Y. Wu, H. Wu, D. Luo, Y. Xu, Y. Hu, W. Zhang, and H. Zhong, "Serving unseen deep learning models with near-optimal configurations: a fast adaptive search approach," in *SoCC'2022*, 2022, pp. 461–476.

[40] M. Song, Y. Hu, H. Chen, and T. Li, "Towards pervasive and user satisfactory cnn across gpu microarchitectures," in *HPCA'17*. IEEE, 2017, pp. 1–12.

[41] Y. Hu, R. Ghosh, and R. Govindan, "Scrooge: A cost-effective deep learning inference system," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 624–638.

[42] S. M. Nabavinejad, S. Reda, and M. Ebrahimi, "Coordinated batching and dvfs for dnn inference on gpu accelerators," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2496–2508, 2022.