WCET-Aware Partitioning and Allocation of Disaggregated Networks for Multicore Systems

Junjie Shi, Christian Hakert, *Member, IEEE*, Kay Heider, Mario Günzel, *Member, IEEE*, Nils Hölscher, Daniel Kuhse, Jian-Jia Chen, *Senior Member, IEEE*, Logan Kenwright, Sobhan Chatterjee, Nathan Allen, *Member, IEEE*, and Partha Roop

***Abstract*—The integration of machine learning into safety-critical cyber-physical systems has significantly increased computational demands, which are often met by modern multicore platforms. While complex memory subsystems, including local caches, make it challenging to maintain timing predictability, they also provide opportunities for worst-case execution time (WCET) optimization through improved data locality. To address this, we propose a multicore partitioning and allocation strategy that leverages sparse structures through neural network disaggregation to optimize the WCET. Our evaluation shows that disaggregated neural networks achieve a significantly reduced WCET, compared to fully connected monolithic neural networks of similar size.**

***Index Terms*—Cyber-physical systems, disaggregated neural network, worst-case execution time (WCET).**

I. Introduction

IN REAL-TIME systems, bounding the worst-case response time (WCRT) of each task is critical to ensure system correctness and safety. To meet increasing computational demands, especially in automotive and industrial domains, modern embedded systems are transitioning to multicore architectures that enable parallel task execution and improve efficiency. Accurate WCRT analysis requires precise estimation of worst-case execution times (WCETs), which are particularly sensitive to nonlocal memory accesses in complex memory hierarchies.

This challenge is exacerbated by the integration of machine learning models, such as neural networks, into real-time systems. Their computational complexity and dense data dependencies introduce significant variability, making WCET analysis more difficult. While neural networks offer ample parallelism for core allocation, nonlocal data transfers caused by tightly coupled layers can substantially increase WCET. Reducing this overhead demands a locality-aware execution strategy tailored to the neural network structure.

This letter addresses the problem of *WCET minimization for core allocation and scheduling of neural networks* by exploiting sparse dependency structures enabled through disaggregation [1], [2]. Disaggregation replaces a monolithic model with multiple independent subnetworks whose outputs are merged only after all subtasks complete. This structure promotes parallelism and reduces interdependencies.

Chatterjee et al. [2] illustrate the benefits of such disaggregation using a lane-changing module for autonomous vehicles based on the NGSIM dataset [3]. A three-class monolithic classifier is split into two binary classifiers for left and right lane changes, with the third class inferred. Despite using the same total number of neurons, the disaggregated model reduces per-layer size and connections, cutting resource usage by 54%, computation by 43%, and improving hardware timing by 85%, with only a 2% drop in accuracy. However, their work does not evaluate the WCET on software platforms.

To address this, we propose a WCET-aware scheduling approach using a mixed-integer quadratic constrained programming (MIQCP) formulation. Our method optimally allocates layers and neurons across CPU cores while penalizing nonlocal memory accesses, thereby enforcing locality and minimizing WCET. To the best of our knowledge, this is the first attempt at WCET-aware partitioning and allocation for neural-network-based workloads on multicore real-time systems. Our main contributions are as follows.

1) We propose a WCET-aware scheduler based on a MIQCP formulation that minimizes execution time through optimized layer partitioning and neuron allocation. Our approach applies to both monolithic and disaggregated neural networks, with notably better results for disaggregated models due to their sparse data dependencies.
2) We conduct a comprehensive numerical evaluation demonstrating substantial WCET reductions compared to a straightforward baseline with equivalent model sizes, achieving up to 95.3% reduction (21.5× faster) for monolithic networks and up to 98.4% reduction (63.7× faster) for disaggregated networks.

II. System Model and Specification

In this section, we present the system model, along with a detailed description of the disaggregated neural network

Received 11 August 2025; accepted 16 August 2025. Date of current version 17 October 2025. This work was supported in part by the German Academic Exchange Service (DAAD), Project under Grant 57706716; in part by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme under Grant 865170; in part by the Deutsche Forschungsgemeinshaft (DFG), as part of the project ARTS-NVM under Grant 502308721; and in part by the Memory Diplomat under Grant 502384507. This manuscript was recommended for publication by A. Shrivastava. *(Corresponding author: Christian Hakert.)*

Junjie Shi, Christian Hakert, Kay Heider, Mario Günzel, Nils Hölscher, Daniel Kuhse, and Jian-Jia Chen are with the Department of Computer Science under Chair XII, TU Dortmund University, 44227 Dortmund, Germany (e-mail: christian.hakert@tu-dortmund.de).

Logan Kenwright, Sobhan Chatterjee, and Partha Roop are with the Electrical, Computer, and Software Engineering, University of Auckland, Auckland 1010, New Zealand.

Nathan Allen is with the Faculty of Design & Creative Technologies, Auckland University of Technology, Auckland 1142, New Zealand.

Digital Object Identifier 10.1109/LES.2025.3600584

structure. We then formally define the problem, outlining the key challenges and objectives that this research addresses.

A. System Model

We consider a set $\mathbf{T}$ of n recurrent tasks to be scheduled on a set of identical (homogeneous) processors M. Each task $\tau_i \in \mathbf{T}$ is described by $\tau_i = (\boldsymbol{C_i}, T_i, D_i)$, where:

1) $C_i = \{C_{i,1}, C_{i,2}, \ldots, C_{i,|M|}\}$ represents the set of WCETs for task τ_i. That is, $C_{i,m}$ is the WCET of τ_i when it is executed using $m \in \{1, \ldots, |M|\}$ cores.
2) T_i is the period of τ_i.
3) D_i is the relative deadline of τ_i.

We consider tasks with constrained deadlines, i.e., $\forall \tau_i \in \mathbf{T}$, $D_i \leq T_i$. Additionally, we assume that all tasks release an infinite number of task instances, called jobs, sporadically, i.e., if a job of τ_i is released at time t the subsequent job is released no earlier than time $t + T_i$.

One of the primary examples considered in this work is disaggregated neural networks, which inherently exhibit high levels of parallelism. Each disaggregated neural network task can be partitioned across multiple processors, allowing concurrent execution of different segments or layers.

B. Neural Network Structure

In this work, we model a neural network task τ_i as a collection of subtasks $\tau_{i,j}$, where each neuron is treated as an individual subtask. The entire network is represented as a directed acyclic graph (DAG), with nodes corresponding to subtasks and edges capturing the data dependencies between them. Subtasks are organized into layers denoted by $\mathcal{L}^i = \ell^i_0, \ldots, \ell^i_{L-1}$. We assume that 1) subtasks within the same layer are independent and 2) there are no cyclic dependencies between layers. That is, it is not permissible for subtasks in layer ℓ^i_1 to depend on those in ℓ^i_2 if other subtasks in ℓ^i_2 simultaneously depend on ℓ^i_1.

In fully connected networks, each neuron in layer ℓ^i_j receives input from every neuron in the preceding layer ℓ^i_{i-1}, resulting in a dense dependency graph. In contrast, sparsely connected networks exhibit a significantly reduced number of interlayer dependencies, enabling more flexible computation.

C. Problem Definition

One of the main challenges in scheduling tasks on multicore systems is managing shared resources, particularly within the memory hierarchy, which includes caches, buses, and main memory. Upper bounds for the access latency of locally cached versus nonlocal memory contents can make a significant difference to a task's WCET. Consequently, the WCET of a task can be significantly tighter bounded when a majority of memory accesses can be guaranteed to be local within a CPUs cache. As a result, locality of memory accesses impacts the system WCRT, a critical factor in meeting timing guarantees in real-time systems.

In this work, we address the problem of WCET minimization for memory-intensive tasks. Specifically, given $m \in \{1, \ldots, |M|\}$ dedicated processors we aim to schedule the subtasks $\tau_{i,j}$ of a task τ_i to minimize its WCET $C_{i,m}$.

III. Worst-Case Execution Time Optimization

In this section, we study intratask scheduling to minimize the WCET of a task τ_i given a number of m dedicated processors. To that end, we first model the execution time of subtask in the presence of memory locality, in Section III-A. Afterwards, we formulate an MIQCP to minimize the execution time of τ_i, in Section III-B. This programming can be solved efficiently using solvers like Gurobi [4]. In Section III-C, we propose an initial condition for the solver.

A. Locality-Aware Execution Model

The execution time of each subtask $\tau_{i,j}$ comprises a base WCET $WCET_{\text{base}}$ (achieved if all processed data is locally available) and penalties for moving data. We assume that there is a constant WCET penalty $WCET_{\text{penalty}}$ that is added to $WCET_{\text{base}}$ for every subtask that $\tau_{i,j}$ requires data from that has to moved between the cores. Specifically, if there are ξ subtasks on different cores than $\tau_{i,j}$, whose data $\tau_{i,j}$ depends on, then the execution time for $\tau_{i,j}$ is

$$\text{WCET}_{\text{full}}(\xi) :- \text{WCET}_{\text{base}} + \xi \cdot \text{WCET}_{\text{penalty}} \tag{1}$$

Note that to obtain a safe upper bound on the WCET without modelling the cache behavior, we assume that the full penalty is considered even if data is potentially cached on the core.

B. Formulation of the MIQCP

To model the intratask schedule of τ_i, we abstract the schedule of subtasks $\tau_{i,j}$ on the level of layers. That is, we specify the start time and finishing time of each layer on each processor. Given the set of layers $\mathcal{L}^i = \{\ell^i_0, \ldots, \ell^i_{L-1}\}$ and the set of m processors $\mathcal{P} = \{p_0, \ldots, p_{m-1}\}$, we define the start times SPC and finishing times FPC by

$$\text{SPC}\colon \mathcal{L}^i \times \mathcal{P} \to \mathbb{N} \qquad \text{and} \qquad \text{FPC}\colon \mathcal{L}^i \times \mathcal{P} \to \mathbb{N} \tag{2}$$

Those start and finishing times deliver a static execution plan of the neural network. The time that is required by a layer on a processor is then defined by $TR = \text{FPC} - \text{SPC}$.

The global start and finish time of each layer is denoted by $S : \mathcal{L}^i \to \mathbb{N}$, with $S(\ell^i_j) = \max_{p_k \in \mathcal{P}} SPC(\ell^i_j, p_k)$, and $F\colon \mathcal{L}^i \to \mathbb{N}$, with $F(\ell^i_j) = \max_{p_k \in \mathcal{P}} \text{FPC}(\ell^i_j, p_k)$, respectively. The whole network finishes at

$$\max_{\ell^i_j \in \mathcal{L}^i} F\left(\ell^i_j\right), \tag{3}$$

i.e., this is the cost function of our WCET analysis.

Our formulation of the optimization problem assumes that layers are executed nonpreemptively, i.e., if a layer ℓ^i_j is started on a processor p_k, then another layer $\ell^i_{j'}$ cannot start on p_k before ℓ^i_j is finished. That is, for all $\ell^i_j \neq \ell^i_{j'} \in \mathcal{L}^i$ and $p_k \in \mathcal{P}$, either $\text{SPC}(\ell^i_j, p_k) \geq \text{FPC}(\ell^i_{j'}, p_k)$ or $\text{FPC}(\ell^i_j, p_k) \leq \text{SPC}(\ell^i_{j'}, p_k)$ must hold. Although this assumption might introduce pessimism, it enables good performance and scalability of our solution. Furthermore, if there is only one layer ℓ^i_{L-1} without any outgoing data dependencies, i.e., the *last layer*, then with this assumption the cost function simplifies to $F(\ell^i_{L-1}) = \max_{p_k \in \mathcal{P}} \text{FPC}(\ell^i_{L-1}, p_k)$.

To find a suitable schedule of layers, in the optimization problem, we partition the full number of neurons of each layer

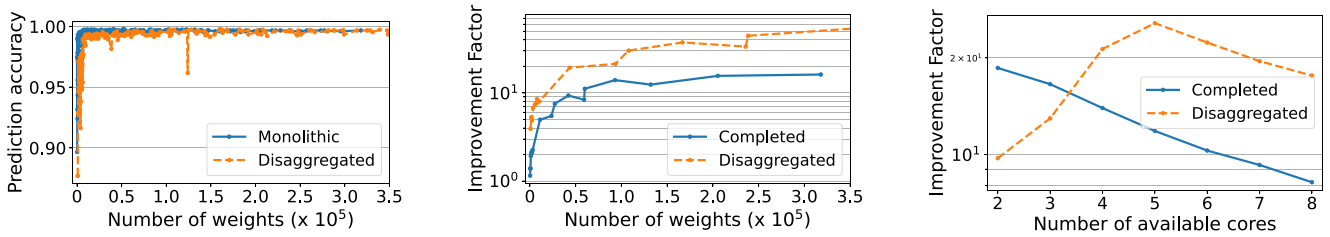

Fig. 1. Performance comparison across structural complexity, output class count, and core availability. (a) Prediction accuracy for tasks with varying structures. (b) WCET minimization results using four cores. (c) WCET minimization results across varying core counts.

to the different cores by a function NLC: $\mathcal{L}^i \times \mathcal{P} \to \mathbb{N}$. An additional constraint ensures that all neurons are computed on a core. All executed networks in the system are merged to a single set of layers, that can be scheduled in a constraint order, where layers from one network need to execute in order. We assume that this data dependencies between layers are checked in the form of a function ID: $\mathcal{L}^i \times \mathcal{L}^i \to \{0, 1\}$, with

$$ID\left(\ell_j^i, \ell_{j'}^i\right) = \begin{cases} 1 \text{ if } \ell_j^i \text{ is data dependent of } \ell_{j'}^i \\ 0 \text{ else} \end{cases} \tag{4}$$

The time required to run a layer ℓ_j^i on a core p_k is then

$$TR\left(\ell_j^i, p_k\right) = \text{NLC}\left(\ell_j^i, p_k\right) \cdot \text{WCET}_{\text{full}}\left(\xi(\ell_j^i, p_k)\right) \tag{5}$$

where an upper bound $\xi(\ell_j^i, p_k)$ on the number of data dependencies is computed as

$$\xi\left(\ell_j^i, p_k\right) = \sum_{\ell_\xi^i \in \mathcal{L}^i \setminus \{\ell_j^i\},\, p_\psi \in \mathcal{P} \setminus \{p_k\}} ID\left(\ell_j^i, \ell_\xi^i\right) \cdot \text{NLC}\left(\ell_\xi^i, p_\psi\right) \tag{6}$$

We ensure that the required time is respected by the condition

$$TR\left(\ell_j^i, p_k\right) = \text{FPC}\left(\ell_j^i, p_k\right) - \text{SPC}\left(\ell_j^i, p_k\right) \tag{7}$$

for all $\ell_j^i \in \mathcal{L}^i$ and $p_k \in \mathcal{P}$. Please note that this also ensures that $\text{SPC}(\ell_j^i, p_k) \leq \text{FPC}(\ell_j^i, p_k)$ and $S(\ell_j^i) \leq F(\ell_j^i)$, because $TR(\ell_j^i, p_k) \geq 0$ holds by definition.

C. Initial Conditions

In order to accelerate the convergence process of the programming, we derive a heuristic solution, which is used as an initial condition for the solution of the MIQCP. Iterating through each layer of the neural network, we assign each layer's neurons to a core using a round-robin approach. The intuition for this policy is that neurons within a layer cannot be dependent on each other, and thus can be allocated to different cores without suffering penalties. We note that processors do not differ, as we assume a homogeneous system. This initial candidate solution is encoded within the P matrix as follows:

$$P_{j,k} = \begin{cases} 1 \text{ if } j \bmod m = k \\ 0 \text{ otherwise} \end{cases} \quad \forall \ell_j^i \in \mathcal{L}^i \, \forall p_k \in \mathcal{P} \tag{8}$$

If the problem has the same number of layers and processors, i.e., $|\mathcal{L}^i| = |\mathcal{P}|$, the result of our heuristic would be the identity matrix $P = I_{|\mathcal{L}^i|}$. It is entirely possible that the initial candidate misleads the solver, resulting in a longer solve time.

IV. Evaluation

To evaluate the effectiveness of the proposed WCET minimization approach, we conduct a numerical analysis across various configurations. We consider monolithic (nondisaggregated) and disaggregated neural networks with equivalent computational loads, measured by the total number of weights in the model. As a dataset for evaluation, we choose the Room Occupancy Estimation [5], which has 16 features and 4 classes.

We use a multilayer perceptron (MLP) [6], well-suited for embedded systems due to its performance and efficiency. We employ model-based optimization (MBO), also known as *Bayesian Optimization* [7], for hyperparameter tuning, using BoTorch [8], a PyTorch-based library. For WCET optimization, a nonpreemptive optimal schedule is derived using the MIQCP model in Gurobi, with a 48-h time limit for the monolithic and disaggregated neural networks. If the optimizer fails to converge, its heuristic solution is used. Scheduling is performed for systems with 1 to 8 cores, and each schedule's feasibility is verified through simulation. We set the cache miss penalty to $\frac{3}{5}$ of the base WCET for each neuron, which is derived by microbenchmarks on several systems. We evaluate our approach against a baseline heuristic scheduler implemented as a greedy, locality-aware round-robin algorithm. In this baseline, each core selects the next available layer based on recently computed values. If no dependent layer is ready, either due to unmet dependencies or task completion, the core proceeds to the next available layer. To quantify the performance difference, we compute the ratio of the baseline WCET to that of our MIQCP-based scheduler, referred to as the Improvement Factor. A value of 1 indicates identical WCETs, while a value of 2 means the MIQCP-based schedule achieves half the WCET of the baseline.

Fig. 1(a) shows the prediction accuracy results for both the original monolithic neural network and the disaggregated neural network structure. For each neural network the total number of weights is calculated by multiplying the weights of each classifier by the number of classifiers (classes). For clarity, we present a subset of results, focusing on configurations with up to 3.5×10^5 weights. The disaggregated neural network achieves prediction accuracy comparable to that of the original neural network, with differences within 2 %.

For evaluation of WCET minimization, the number of cores is fixed to four CPU cores in Fig. 1(b). The MIQCP scheduler consistently achieves greater WCET reductions for disaggregated neural networks compared to monolithic neural

networks of equivalent size. In absolute terms, our MIQCP scheduler reduces WCET by approximately 2–3 times compared to the baseline for monolithic neural networks, and by approximately 3–4 times for disaggregated neural networks. These results demonstrate the substantial benefits of our approach in optimizing WCET, particularly for disaggregated structures.

Furthermore, we evaluate the performance of our MIQCP scheduler with a varying number of CPU cores in Fig. 1(c). For this analysis, we fix the size of the MLPs to 10^5 weights and vary the number of CPU cores between 2 and 8. For disaggregated neural networks, a sweet spot for WCET minimization emerges between 3 and 5 cores. For monolithic neural networks, WCET minimization decreases steadily with more CPU cores but lacks a clear sweet spot, reflecting their inability to efficiently leverage parallelism due to the absence of independent partitions.

V. Related Work

This section reviews relevant research in two key areas: 1) memory-aware WCET optimization and analysis, and 2) time-predictable neural networks.

a) Memory-aware WCET optimization and analysis: To achieve less pessimistic WCET estimation, numerous approaches have been proposed over the past decades. In order to provide tighter WCET bounds, Zhang and Yan [9] proposed a method to analyze worst-case cache interferences and bound the WCET for threads running on multicore processors with shared direct-mapped L2 instruction caches. Herrmann et al. [10] addressed heterogeneous resources, such as dual-memory clusters with multicores and dedicated accelerators (FPGA or GPU), by proposing a memory-aware list scheduling approach using an intricate ILP formulation.

b) Time-predictable neural networks: Several studies have investigated the WCET estimation of neural networks directly by either measuring or estimating the execution time of monolithic networks, which are either single core only or lack sound formal guarantees. Tizpaz-Niari and Sankaranarayanan [11] apply statistical methods to estimate the extreme-case WCET of a neural network. Silva et al. [12] introduced ACETONE, a framework that compiles neural networks into C code and provides sound timing bounds using the Otawa tool.

In terms of scheduling multiple networks, Ling et al. [13] decompose neural networks into duo-blocks, dynamically allocating them to a CPU or GPU based on current load to avoid deadline misses in real-time systems. While their partitioning approach is relevant, BlastNet's scheduling lacks formal guarantees. Other studies [14], [15] also use network partitioning to improve performance on edge devices but do not specifically address timing analysis.

None of these works investigates the WCET-aware partitioning and allocation for neural-network-based workloads on multicore real-time systems.

VI. Conclusion

This work addresses the challenge of optimizing WCET for neural network-based workloads in safety-critical systems on multicore platforms. We proposed a WCET-aware scheduler based on a MIQCP formulation that optimizes layer partitioning and neuron allocation to minimize execution times. Our approach is applicable to both monolithic and disaggregated neural networks, achieving significantly better performance in disaggregated networks by leveraging their sparse data dependency structures. Comprehensive evaluations demonstrated substantial WCET reductions, while maintaining comparable prediction accuracy across diverse datasets and configurations.

References

[1] X. Yang, P. S. Roop, H. A. Pearce, and J. W. Ro, "A compositional approach using Keras for neural networks in real-time systems," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2020, pp. 1109–1114. [Online]. Available: https://doi.org/10.23919/DATE48585.2020.9116371

[2] S. Chatterjee, N. Allen, N. D. Patel, and P. S. Roop, "Exploring compositional neural networks for real-time systems," in *Proc. 22nd ACM-IEEE Int. Symp. Formal Methods Models Sys. Design (MEMOCODE*, 2024, pp. 46–57. [Online]. Available: https://doi.org/10.1109/MEMOCODE63347.2024.00010

[3] U.S. Department of Transportation Federal Highway Administration, "Next generation simulation (NGSIM) vehicle trajectories and supporting data," 2016. Accessed: May 26, 2025. [Online]. Available: https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajector/8ect-6jqj

[4] (Gurobi Optim., LLC, Beaverton, OR, USA). *Gurobi Optimizer Reference Manual*. 2024. [Online]. Available: https://www.gurobi.com

[5] A. P. Singh, V. Jain, S. Chaudhari, F. A. Kraemer, S. Werner, and V. Garg, "Machine learning-based occupancy estimation using multivariate sensor nodes," in *Proc. GLOBECOM Workshops*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOCOMW.2018.8644432

[6] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences," *Atmosph. Environ.*, vol. 32, no. 14, pp. 2627–2636, Jan. 1998.

[7] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Global Optim.*, vol. 13, pp. 455-492, Dec. 1998. [Online]. Available: http://link.springer.com/article/10.1023/A:1008306431147

[8] M. Balandat et al., "BoTorch: A framework for efficient monte-carlo Bayesian optimization," in *Proc. 35th Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–15. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html

[9] W. Zhang and J. Yan, "Accurately estimating worst-case execution time for multi-core processors with shared direct-mapped instruction caches," in *Proc. 15th IEEE Int. Conf. Embed. Real-Time Comput. Syst. Appl. (RTCSA)*, 2009, pp. 455–463. [Online]. Available: https://doi.org/10.1109/RTCSA.2009.55

[10] J. Herrmann, L. Marchal, and Y. Robert, "Memory-aware list scheduling for hybrid platforms," in *Proc. Int. Parallel Distrib. Process. Symp. Workshops*, 2014, pp. 689–698. [Online]. Available: https://doi.org/10.1109/IPDPSW.2014.80

[11] S. Tizpaz-Niari and S. Sankaranarayanan, "Worst-case convergence time of ML algorithms via extreme value theory," in *Proc. IEEE/ACM 3rd Int. Conf. AI Eng.-Softw. Eng. AI (CAIN)*, 2024, pp. 211–221. [Online]. Available: https://doi.org/10.1145/3644815.3644989

[12] I. D. A. Silva, T. Carle, A. Gauffriau, and C. Pagetti, "ACETONE: Predictable programming framework for ML applications in safety-critical systems (artifact)," *Dagstuhl Artifacts Ser.*, vol. 8, no. 1, pp. 1–2, 2022. [Online]. Available: https://doi.org/10.4230/DARTS.8.1.6

[13] N. Ling, X. Huang, Z. Zhao, N. Guan, Z. Yan, and G. Xing, "BlastNet: Exploiting duo-blocks for cross-processor real-time DNN inference," in *Proc. 20th ACM Conf. Embed. Netw. Sensor Syst., SenSys*, 2022, pp. 91–105. [Online]. Available: https://doi.org/10.1145/3560905.3568520

[14] J. Jeong and H. Yang, "Optimal partitioning of distributed neural networks for various communication environments," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, 2021, pp. 269–272. [Online]. Available: https://doi.org/10.1109/ICAIIC51459.2021.9415248

[15] Q. Li, L. Huang, Z. Tong, T. Du, J. Zhang, and S. Wang, "DISSEC: A distributed deep neural network inference scheduling strategy for edge clusters," *Neurocomputing*, vol. 500, pp. 449–460, Aug. 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.05.084