Distributed Inference for Multiple DNN Models in IoT Environments

YoungHwan Jin HyungBin Park SuKyoung Lee
Yonsei University

Motivation

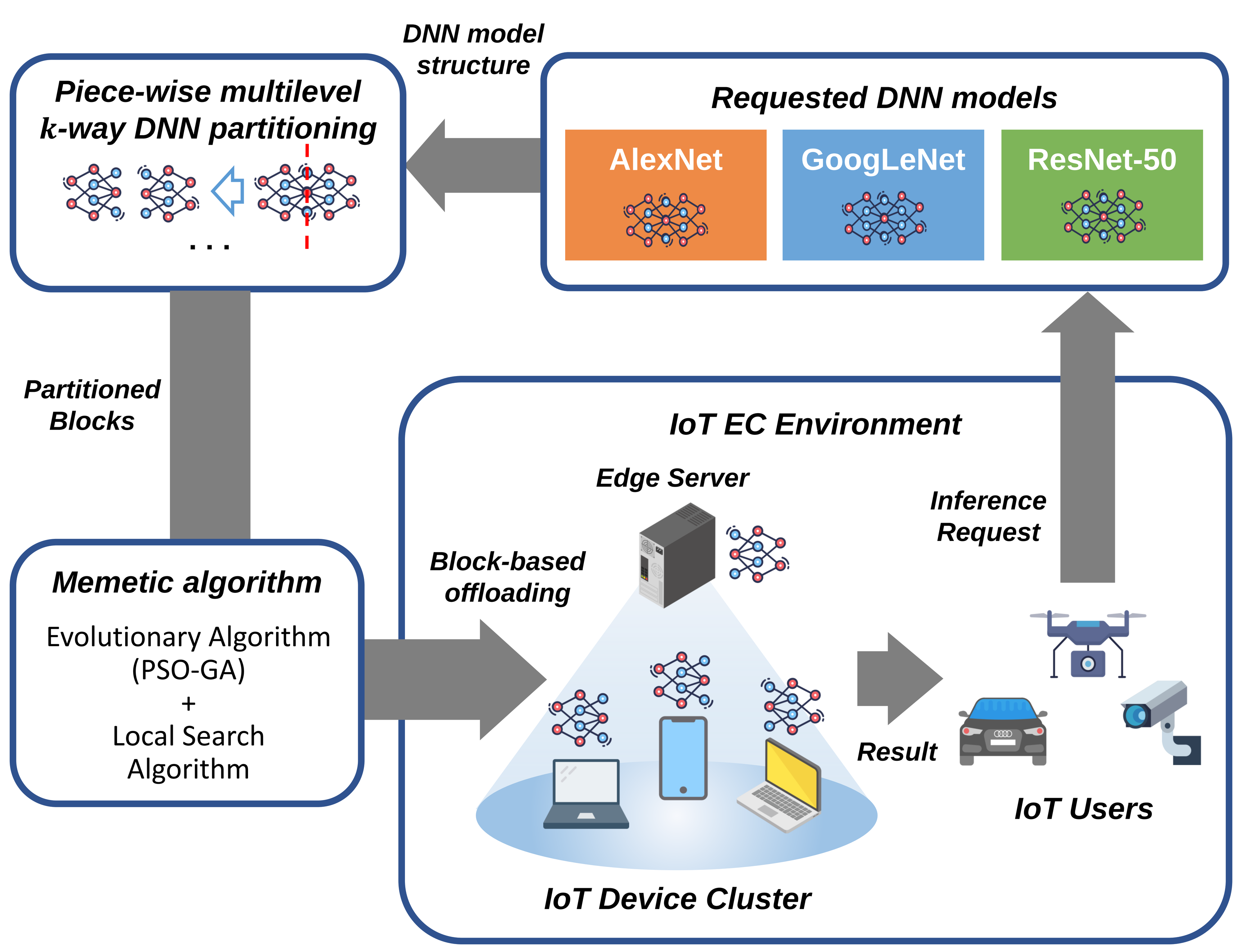
DNN inference offloading in IoT environment

Edge Computing (EC) - To support running DNN applications, EC has emerged as a solution, starting with support for a single DNN model.

Multiple DNN Request in IoT Environment - Applications such as augmented reality which require the use of multiple DNN models have become popular.

What is the problem?
- Response time increases with increasing workload
- Limited computing resource of single edge server

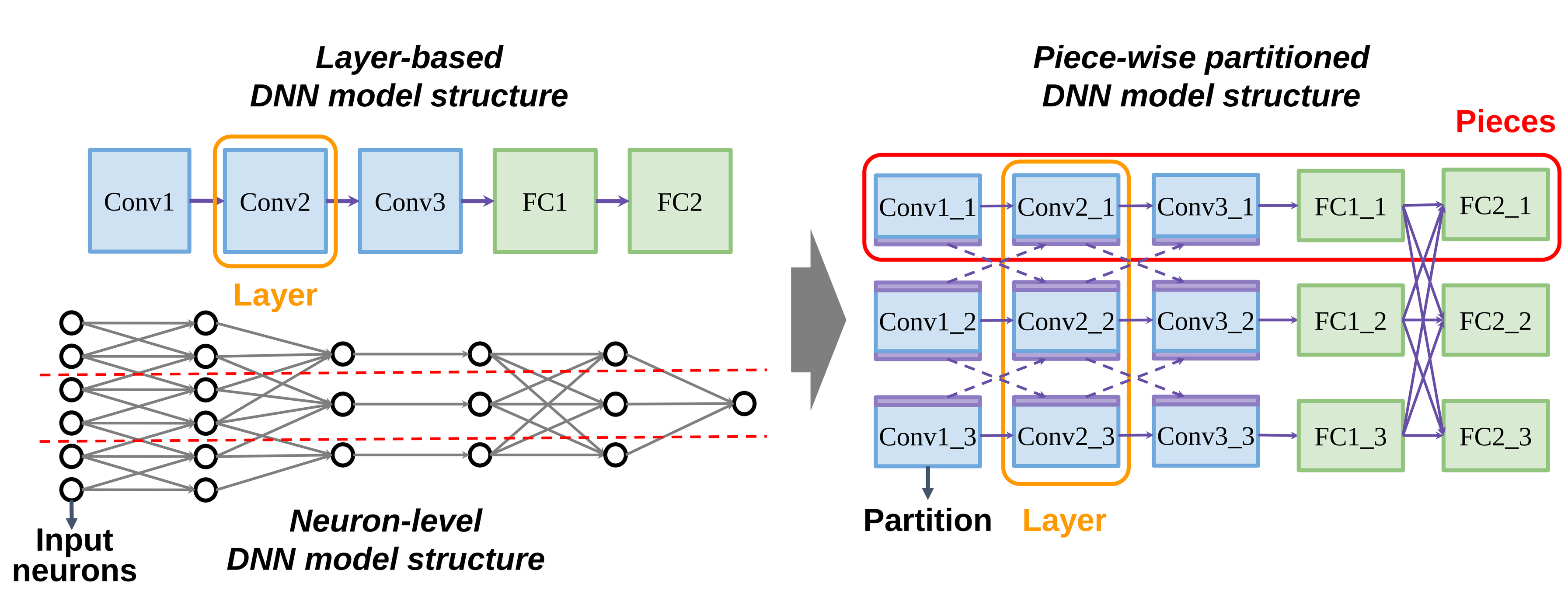
Solution
- Offload multiple DNN tasks to EC environment with additional computational support from nearby IoT device, aiming to minimize inference completion time.
- To find partitioning points to offload with faster completion time, we propose a piece-wise multilevel partitioning and scheduling algorithm.



Piece-wise Multilevel k-way DNN Partitioning

Why piece-wise partitioning

- Since each layer is composed of a set of neuron operations, piece-wise partitioning is applied to the input neuron such that **multiple devices can perform DNN inference operations in parallel**.
- This can **improve the degree of parallelism** compared to the layer-based DNN model structure. We construct a piece-wise partition by dividing the DNN layer in advance as much as the degree of parallelism is required.

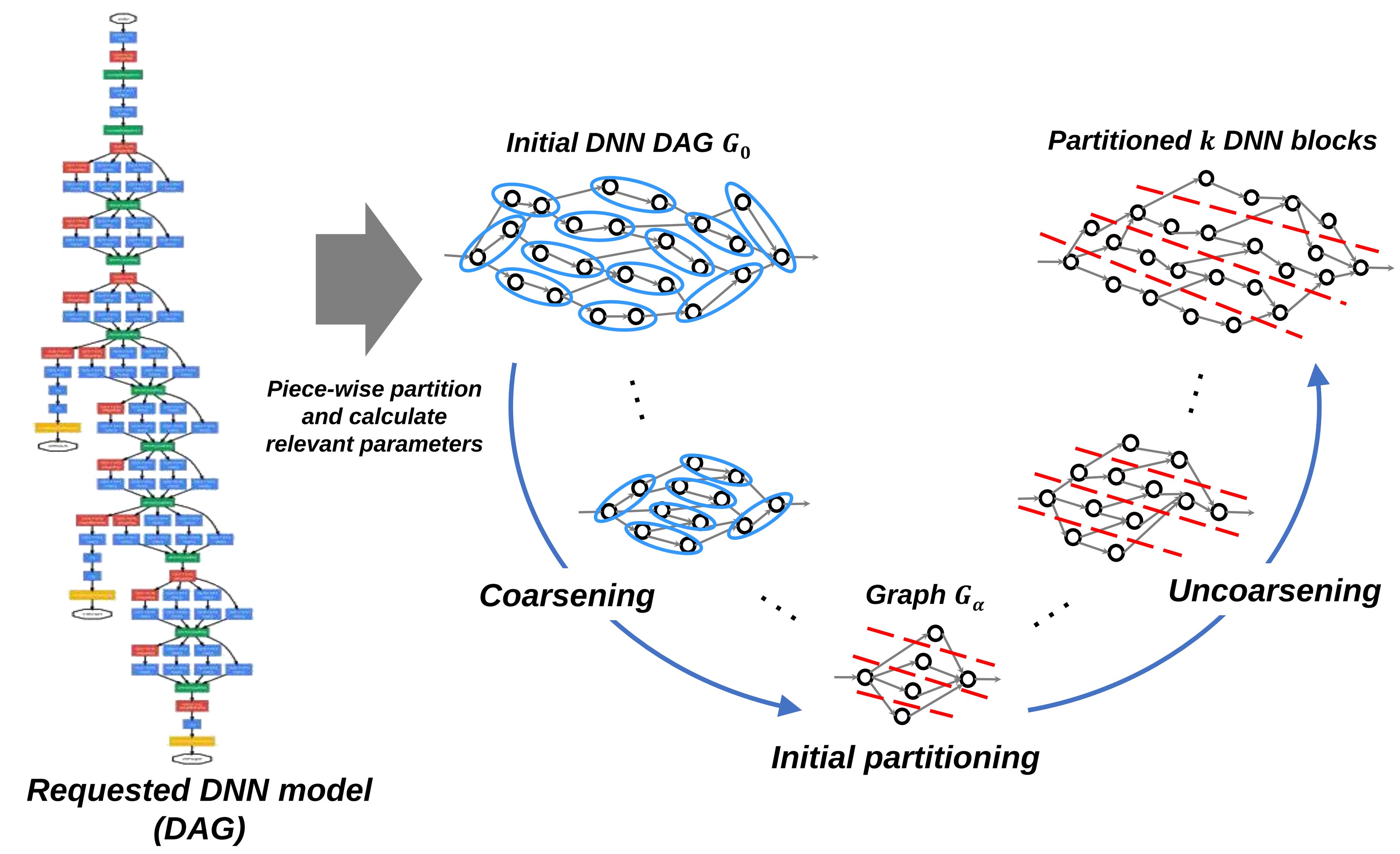


We consider the DNN model as a directed acyclic graph (DAG) and introduce a multilevel partitioning scheme to quickly find the optimal partitioning point from the large-scale DAG structure. The proposed piece-wise multilevel k-*way* partitioning algorithm aims to **maximize the degree of parallelism of DNN models by grouping partitions with the same piece order into blocks**. The algorithm has three main phases.

(I) Coarsening Phase - The initial DNN graph G_0 is transformed into a sequence of smaller graphs $G_1, \ldots, G_\alpha$ through a coarsening phase.

(II) Initial Partitioning Phase - The graph G_α is divided into k blocks $S_\alpha = \{B_1, \ldots, Bk\}$.

(III) Uncoarsening Phase - The blocks S_α of G_α is projected back to G_0 via the intermediate blocks $S_{\alpha-1}, \ldots, S_0$. In each level $\in [0, \ldots, \alpha - 1]$, boundary FM algorithm is applied. All boundary partitions are moved to adjacent blocks in order of gain to minimize transmission delay.



Block-based Multiple DNN Model Offloading

Memetic algorithm
Particle Swarm Optimization with Genetic Algorithm (PSO-GA) is adapted to the block-based offloading. Its local search algorithm guarantees near-optimal solutions in early generations.

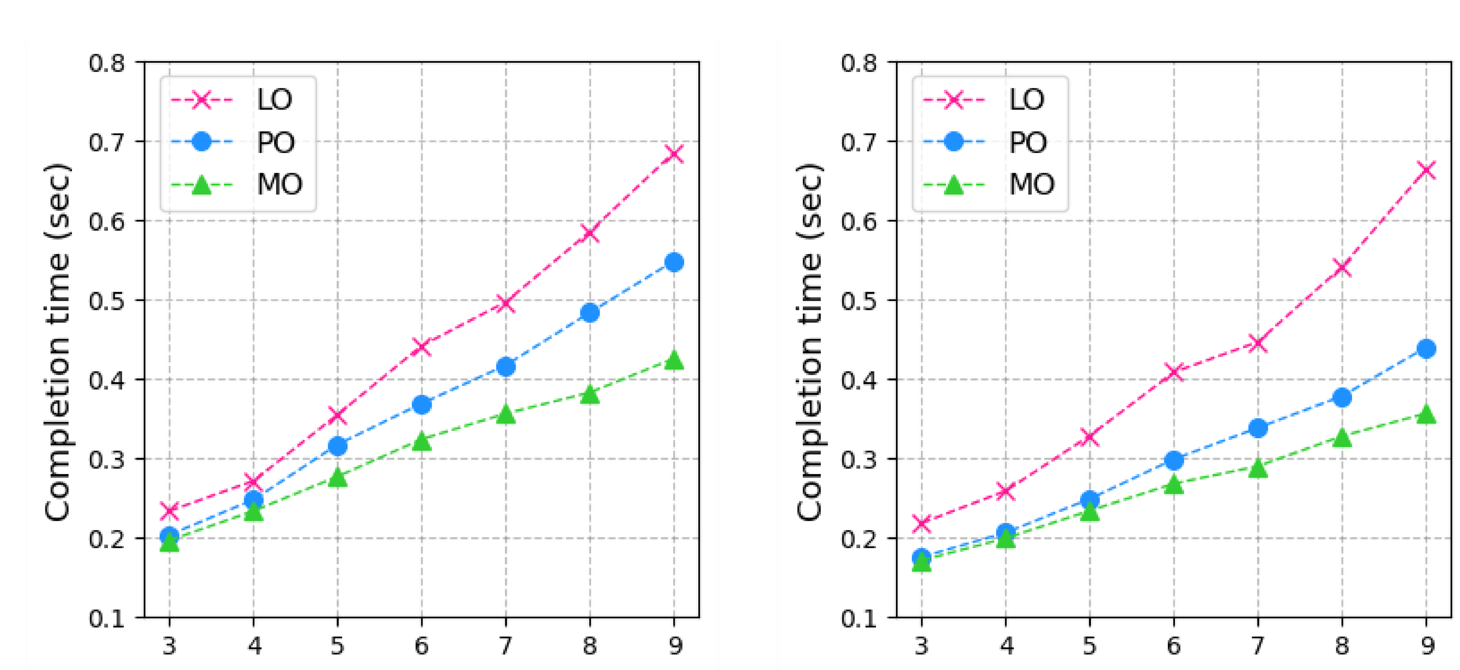
Memetic Algorithm for DNN block offloading

Input: Set of all DNN blocks S from the partitioning algorithm
Output: X and Y with the minimum completion time
$Y \leftarrow$ set execution orders in descending order of rank
while convergence criteria are not satisfied do
Find candidate solutions of X with PSO-GA
for each particle of PSO-GA do // Local optimization
  Randomly select $B \in S$ and offload the B to the device d which minimizes the latency
end for
end while

Experiment setting
- Devices: An edge server, Jetson TX2, Raspberry Pi 4B
- DNN models: AlexNet, GoogleNet, and ResNet50

Layer-based offloading (LO) - Layer-based partitioning & offloading with PSO-GA.
PSO-GA offloading (PO) - Piece-based partitioning & offloading with PSO-GA.
MA offloading (MO) - Our piece-based partitioning & offloading with MA.

The proposed MO showed performance improvements of $\mathbf{16.2-33.1\%}$ $\mathbf{(26.2-40.6\%)}$ over LO, when $D = 3$ ($D = 6$). There is greater performance improvement going from $D = 3$ to $D = 6$, and the performance increase is due to the piece-wise partitioning which allows all resources to be utilized by decoupling the dependencies within a layer. MO showed performance improvements of $\mathbf{9.4-20.6\%}$ $\mathbf{(8.4-18.2\%)}$ over PO when $D = 3$ ($D = 6$), demonstrating the effectiveness of the local search of MA over PSO-GA.



Comparison of completion time performance based on the number of DNN models for $D = 3$ (left) and $D = 6$ (right)