Best Splitting of DNNs for distributed deployment in edge-cloud continuum with Quality Requirements

Simone Nicosanti
simone.nicosanti.01@students.uniroma2.eu, snicosanti@tudelft.nl

Tor Vergata University of Rome, Delft University of Technology

August 11th, 2025

Roadmap

Roadmap

- **Problem**: How can we optimize inference in edge devices?
- Why optimize such a thing?
  - Limited computational power
  - Limited energy (e.g. battery devices)
  - Limited memory
- **Thesis Target**: Optimize inference time and energy consumption with quality requirements

- The project originated from a public call for proposals in the context of occupational safety
  - Hence the importance of image AI processing
- **Target Model** : Yolo11
- Reasons:
  - State of the art model
  - High variety of tasks (detection, segmentation, classification, ecc.)
  - Easy to export in different formats

Model Format: ONNX

Easy to use:

- Libraries for import/export.
- Interface for submodel extraction.
- Interface for model analysis.

Execution Framework: OnnxRuntime

Reasons:

- Highly compatible with a lot of low level architectures (through its *ExecutionProvider*).
- Multi-Language and Multi-Platform.
- Easy support to per layer quantization.

Other instruments: NetworkX (for Graph Modeling); PuLP (for Problem Definition); CPLEX (for Problem Resolution).

Roadmap

System Architecture

High Level Interactions

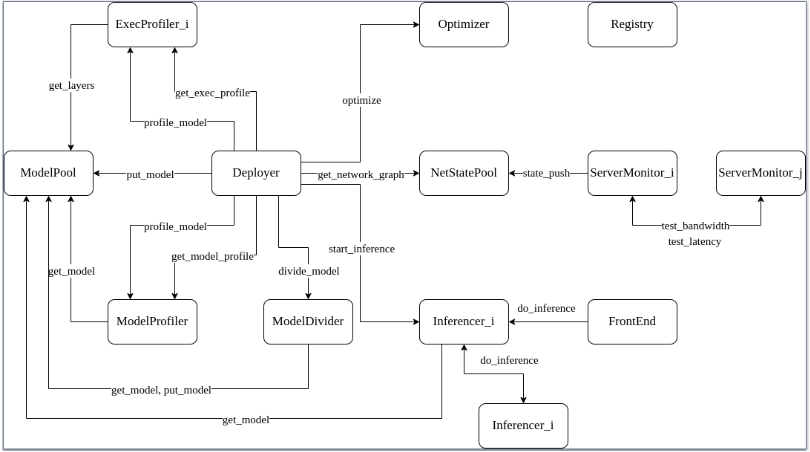

Figure 1: High Level Interactions

Roadmap

Base Definition

The main elements of our problem are:

- DNN. Modelled as a *logical graph* $G_D = (V_D, E_D)$. Where we call:
  - T_D the set of tensors moving between layers
- Server Network. Modelled as a *physical graph* $G_N = (V_N, E_N)$

Objective. Build a mapping $G_D \rightarrow G_N$ (graph partitioning problem).

Problem Variables:

- $x_{ik} \in \{0,1\}$ $\forall i \in V_D, \forall k \in V_N$: layer assignment.
- $y_{tn} \in \{0,1\}$ $\forall t \in T_D, \forall n \in E_N$: tensor assignment.
- $q_i \in \{0,1\}$ $\forall i \in V_D$: quantization activation.
- $x^q_{ik} \in \{0,1\}$ $\forall i \in V_D, \forall k \in V_N$: layer assignment and quantization.
- $y^q_{tn} \in \{0,1\}$ $\forall t \in T_D, \forall n \in E_N$: tensor assignment and quantization.

Latency Modelling

Computation Latency

Let $f_k(i, q_i)$ = time for running layer i on server k with quantization q_i. Then we have:

- Layer i computation time on server k:
  $T^c_{ik} = [f_k(i,0) \cdot (1 - q_i) + f_k(i,1) \cdot q_i] \cdot x_{ik} = f_k(i,0) \cdot x_{ik} - (f_k(i,0) - f_k(i,1)) \cdot x^q_{ik}$
- Server k computation time:
  $T^c_k = \sum_{i \in V_D} T^c_{ik}$
- Total Computation Time:
  $T^c = \sum_{k \in V_N} T^c_k$

Transmission Latency

Let $g(t, n)$ = time for sending tensor t through network edge n. Then we have:

- Server k sending time for tensor t:
  $T^x_{tk} = \left[g(t,n) \cdot (1 - q_i) + g^{'}(t,n) \cdot q_i\right] \cdot y_{tn} = g(t,n) \cdot y_{tn} - \left(g(t,n) - g^{'}(t,n)\right) \cdot y^q_{tn}$
- Server k sending time:
  $T^x_k = \sum_{t \in T_D} \left(T^{x-self}_{tk} + T^{x-other}_{tk}\right) = T^{x-self}_k + T^{x-other}_k$
- Total sending time:
  $T^x = \sum_{k \in V_N} T^x_k$

Therefore we have: $T = T^c + T^x$

Energy Modelling

Computation Energy Let $h_k^c(t) = P_k^c \cdot t$ the function giving the computation energy consumption for server k. Then we have:

- Computation energy per server k: $E_k^c = h_k^c(T_k^c)$
- Total computation energy: $E^c = \sum_{k \in V_N} E_k^c$

Transmission Energy Let $h_k^{x-y}(t) = P_k^{x-y} \cdot t$ the giving the function giving the transmission energy consumption for server k in case y. Then we have:

- Transmission energy per server k: $E_k^x = h_k^{x-self}(T_k^{x-self}) + h_k^{x-other}(T_k^{x-other})$
- Total transmission energy: $E^x = \sum_{k \in V_N} E_k^x$

Therefore we have: $E = E^c + E^x$

Let:

- $V_Q \subset V_D$ a subset of layers for which quantization can be enabled.
- $\mathbf{q} = \{q_i\}_{i \in V_Q}$ the vector of q_i variables for quantizable layers.
- $\eta(\mathbf{q})$ a polynomial regression model of degree d giving as output the quantization noise obtained by the quantization scheme $\mathbf{q}$.

Such a regressor can be represented using linear constraints.

Important Constraints

- Unique assignment of layers to servers:

$$\forall i \in V_D \qquad \sum_{k \in V_N} x_{ik} = 1$$

- Coherence in tensor mapping. Said:
  - $i \in V_D$ source layer of tensor $t \in T_D$.
  - $V_D^t \subset V_D$ subset of layers receiving t as input.

  Then we have:

$$\forall t \in T_D, \forall n = (k,h) \in E_D \qquad \begin{cases} y_{tn} \leq x_{ik} \\ y_{tn} \leq \sum_{j \in V_D^t} x_{jh} \\ y_{tn} \geq x_{ik} + \frac{1}{|V_D^t|} \sum_{j \in V_D^t} x_{jh} - 1 \end{cases}$$

Final Definition

The **final problem definition** will be as follows:

$$\begin{aligned} min \quad & o(\omega_T \cdot T, \omega_E \cdot E) \\ subject\ \ to \quad & E_0 \leq J_0 \\ & \eta(\mathbf{q}) \leq \eta_{max} \\ & others... \end{aligned}$$

In order to tackle the scale problem of multi-objective optimization, both T and E are **normalized** in $[0, 1]$ interval using a min-max strategy, obtaining the following formulation:

$$\begin{aligned} min \quad & \omega_T \cdot T^{norm} + \omega_E \cdot E^{norm} \\ subject\ \ to \quad & E_0 \leq J_0 \\ & \eta(\mathbf{q}) \leq \eta_{max} \\ & others... \end{aligned}$$

Roadmap

Problem

- Number of quantized/not-quantized combinations can be very high ($2^{\#layers}$).
- Analysis of all possible combinations is infeasible.

Solution

- Consider only a subset of layer to be quantized.
- Those with higher number of FLOPS.

In this case, only 12 layers have been considered for quantization (most of which are *Conv* layers).

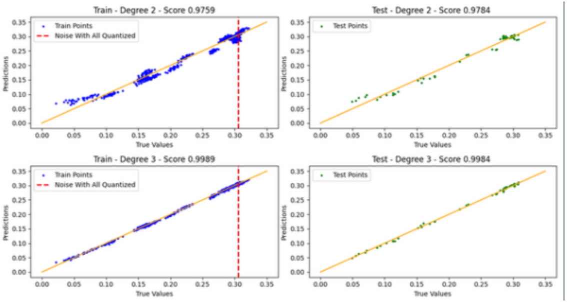

Figure 2: Noise Regressor Fit - Degree 2 and 3

Quantization noise defined as:
$\rho = \frac{1}{n}\sum_{i=1}^{n} mean(|o_i - o_{q,i}|)$

Machines GCP:

- Device
  - Machine Type: c3-standard-4
  - Docker --cpus: 0.5
- Edge
  - Machine Type: c3-standard-4
  - Docker --cpus: 1.0
- Cloud
  - Machine Type: n1-standard-4
  - GPU: nvidia-tesla-t4

Network Config:

- Device $\rightarrow$ Edge
  - Max Bandwidth: 5 MB/s
  - Latency: 5 ms
- Edge $\rightarrow$ Device
  - Max Bandwidth: 20 MB/s
  - Latency: 5 ms

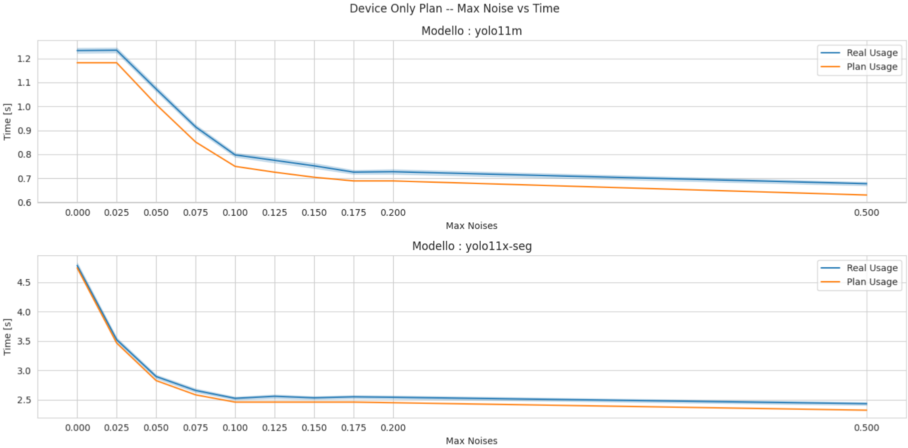

Figure 3: Device Only: Latency vs Quantization Noise

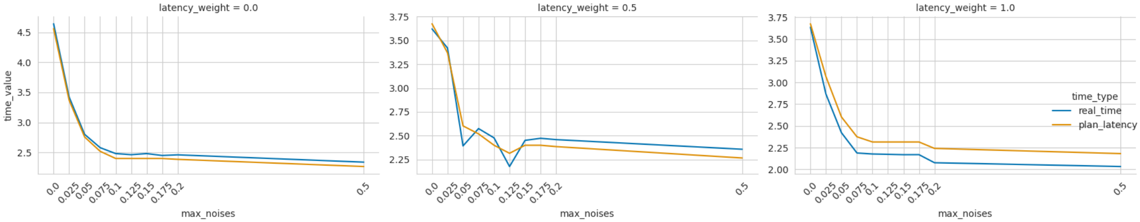

Figure 4: Device & Edge: Latency vs Quantization Noise

Latency Test

Device & Edge & Cloud

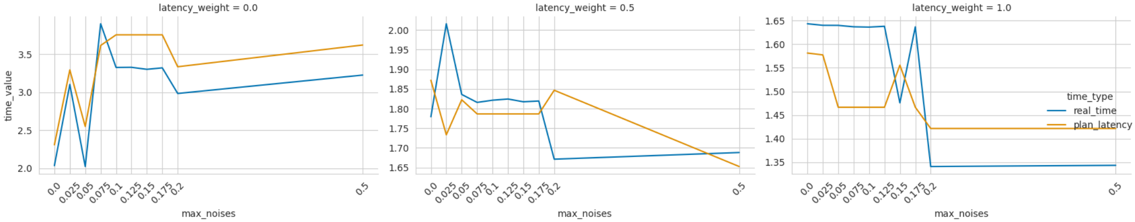

Figure 5: Device & Edge & Cloud: Latency vs Quantization Noise