MAESTRO: Supporting Real-Time Segmentation Networks on Surgical Systems via Edge Computing

Francesco Fiorella[1] FRANCESCO.FIORELLA@SLU.EDU
Andrea Pinto[1] ANDREA.PINTO.1@SLU.EDU
Lin Guo[1] LIN.GUO@SLU.EDU
[1] *Computer Science Department, Saint Louis University, USA*
Alessio Sacco[2] ALESSIO_SACCO@POLITO.IT
[2] *Department of Control and Computer Engineering, Politecnico di Torino, Italy*
Mohammad Mahmoud[3] MMAHMOUD96@SIUMED.EDU
[3] *School of Medicine, Southern Illinois University, USA*
Flavio Esposito[1] FLAVIO.ESPOSITO@SLU.EDU

Editors: Accepted for publication at MIDL 2025

Abstract

Deep neural networks (DNNs) enable accurate segmentation of surgical video streams, but their high computational and memory demands pose challenges for deployment on resource-constrained surgical systems. We present MAESTRO, an adaptive edge computing architecture that supports real-time execution of segmentation networks on surgical platforms. MAESTRO uses split learning to partition inference between the surgical device and an edge server, dynamically selecting the optimal cut layer to balance latency, energy consumption, and data privacy. We evaluate MAESTRO using a YOLOv11 model trained on the Dresden Surgical Anatomy Dataset (DSAD) and tested on Da Vinci robotic surgery videos. Experiments demonstrate up to 43% latency reduction and 56% energy savings compared to full offloading, while maintaining low data leakage risk. MAESTRO provides a flexible and efficient solution for deploying segmentation networks in real-time, privacy-sensitive surgical environments, and generalizes to other low-resource applications.

Keywords: Machine Learning, Split Learning, Segmentation, Edge Computing

1. Introduction

Real-time video segmentation with deep neural networks (DNNs) is critical in surgical applications, where timely and accurate visual feedback directly impacts clinical outcomes (Soper et al., 1994). However, the computational and energy demands of DNNs often exceed the capabilities of surgical devices, making deployment in constrained environments challenging (Shen et al., 2017). We introduce MAESTRO, an adaptive, serverless edge architecture that enables real-time execution of deep neural networks for multi-organ segmentation during laparoscopic surgery. MAESTRO leverages split learning (Vepakomma et al., 2018; Turina et al., 2021; Matsubara et al., 2022) and dynamic computation offloading to partition DNN inference between surgical systems and edge servers, optimizing latency, energy efficiency, and data privacy (Shi et al., 2016; Shiranthika et al., 2023).

Evaluated on laparoscopic video data from Da Vinci robotic systems, MAESTRO runs early DNN layers on a Coral Edge TPU and offloads the remainder to a serverless edge backend. Trained on the Dresden Anatomical Surgical Dataset (DSAD) (Carstens et al.,

© 2025 CC-BY 4.0, F. Fiorella, A. Pinto, L. Guo, A. Sacco, M. Mahmoud & F. Esposito.

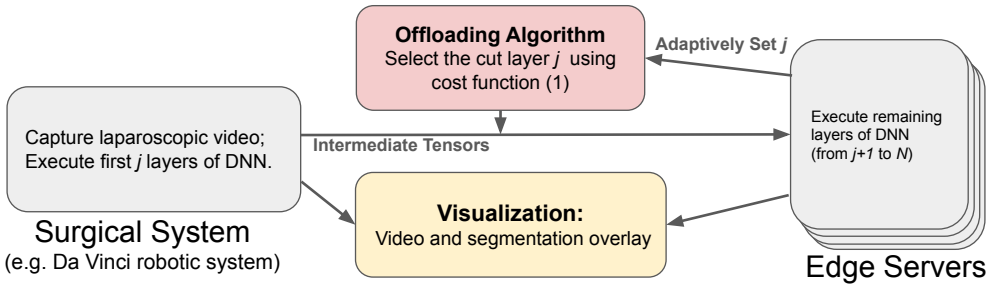

Figure 1: System architecture overview. The DNN model performs live multi-organ segmentation and is split between Surgical System and Edge. An Offloading algorithm decides which is the optimal number of layers to execute locally to protect the data.

2023), the system is assessed across latency, energy usage, and resistance to data reconstruction attacks. These metrics inform a dynamic partitioning algorithm that adapts to varying runtime conditions. While developed for surgical support, MAESTRO's design generalizes to other real-time, resource-constrained DNN applications.

2. Method

MAESTRO is a split learning-based architecture designed to enable real-time execution of deep neural networks (DNNs) for multi-organ segmentation during laparoscopic surgery. The system partitions the DNN between a resource-constrained surgical device and an edge server. In the MAESTRO workflow, video frames are captured and preprocessed locally. A subset of the DNN, up to a selected cut layer j, is executed on the surgical device, producing an intermediate feature tensor. This tensor is transmitted to the edge server, which processes the remaining layers and returns the segmentation output for real-time visualization.

To determine the optimal partition point j^*, MAESTRO employs a cost-aware offloading algorithm. The system collects runtime metrics, including local and remote inference times, transmission delay, energy consumption, and data leakage risk. Regression models are trained to estimate performance under different partitioning configurations. A cost function is then used to evaluate each candidate cut layer:

$$C(j) = w_1 \cdot (T_{M,j} + T_{U,j} + T_{C,j}) + w_2 \cdot E_{M,j} + w_3 \cdot R_{M,j} \tag{1}$$

where $T_{M,j}$, $T_{U,j}$, and $T_{C,j}$ represent the local execution time, transmission time, and edge execution time, respectively. $E_{M,j}$ is the energy consumed locally, and $R_{M,j}$ is the risk of data leakage based on the similarity between the input and intermediate tensor. The weights w_1, w_2, w_3 allow tuning the relative importance of each factor. The optimal cut layer j is selected with the minimal cost. MAESTRO continuously monitors runtime conditions and adapts the partitioning as needed. This dynamic strategy ensures efficient and privacy-preserving DNN inference in real-time surgical environments. The full DNN partitioning algorithm is detailed in Appendix A.

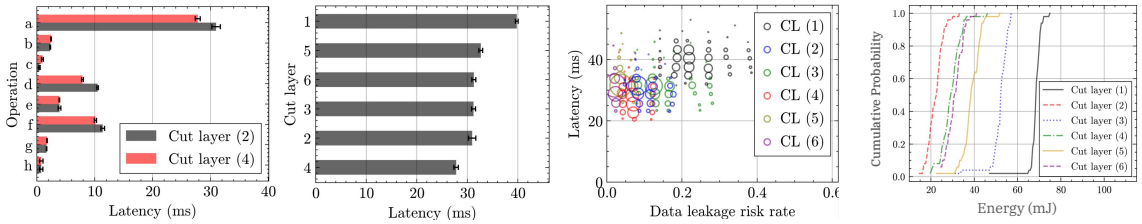

Figure 2: MAESTRO performance across different cut layers. (Left) Latency breakdown by operation shows how a split to layers 2 and 4 reduces both server load and transmission time. (Center-left) Total latency increases with deeper cuts. (Center-right) Trade-off between latency and data leakage risk. (Right) Energy consumption is lowest for cut layers 2 and 4.

3. Experimental Results

We trained a YOLOv11 model (Khanam and Hussain, 2024) on the DSAD dataset (Carstens et al., 2023) and evaluated it using private laparoscopic videos recorded with Da Vinci robotic systems. In our setup, early layers of the network were executed on a Coral Edge TPU, while remaining layers were offloaded to a GPU-equipped edge server.

As shown in Fig. 2, configurations with 2 and 4 local layers delivered the best trade-offs. The latency breakdown (leftmost plot) highlights that executing fewer layers on the device (e.g., cut layer 2) increases server-side latency and transmission cost, while deeper cuts (e.g., layer 4) reduce those components but shift more computation to the device. Overall, both configurations achieve significant reductions in total latency—up to 43%—compared to full offloading (center-left). Similarly, energy consumption (rightmost plot) is minimized for these configurations, yielding savings of up to 56%.

Privacy considerations are reflected in the latency-risk trade-off (center-right), where deeper cut layers correlate with lower data leakage risk. However, this comes at the cost of increased latency. Our results show that cut layers 2 to 4 provide the best balance between real-time performance, energy efficiency, and privacy.

These findings confirm MAESTRO's ability to adaptively select cut layers that respond to system constraints and application priorities, demonstrating its practicality for real-time deployment in surgical video analysis.

4. Discussion and conclusion

We presented MAESTRO, an edge computing architecture for real-time multi-organ segmentation in laparoscopic surgery using split learning. Designed for integration with the Da Vinci Xi platform, MAESTRO improves computational efficiency and responsiveness, enabling intelligent, privacy-aware surgical assistance. Experimental results confirm the feasibility of deploying split learning on constrained edge devices, highlighting MAESTRO's potential for broader real-time medical imaging applications. Beyond organ segmentation, the method is extensible to other deep learning-based surgical video understanding tasks, making it a flexible foundation for future smart operating room systems.

Acknowledgments

This work has been supported by SIU School of Medicine and by NSF award OAC 2430236.

References

M. Carstens, F.M. Rinner, S. Bodenstedt, et al. The dresden surgical anatomy dataset for abdominal organ segmentation in surgical data science. *Scientific Data*, 10:3, 2023. doi: 10.1038/s41597-022-01719-2.

Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024.

Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Comput. Surv.*, 55(5), December 2022. ISSN 0360-0300. doi: 10.1145/3527155. URL https://doi.org/10.1145/3527155.

Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19(1):221–248, 2017. doi: 10.1146/annurev-bioeng-071516-044442. PMID: 28301734.

Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

Chamani Shiranthika, Parvaneh Saeedi, and Ivan V. Bajić. Decentralized learning in healthcare: A review of emerging techniques. *IEEE Access*, 11:54188–54209, 2023. URL https://api.semanticscholar.org/CorpusID:259035436.

N. Soper, L. Brunt, and K. Kerbl. Laparoscopic general surgery. *The New England journal of medicine*, 330 6:409–19, 1994. doi: 10.1056/NEJM199402103300608.

Valeria Turina, Zongshun Zhang, Flavio Esposito, and Ibrahim Matta. Federated or split? a performance and privacy analysis of hybrid split and federated learning architectures. In *IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 250–260. IEEE, 2021.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Appendix A. DNN Partitioning Algorithm

The MAESTRO system uses an adaptive partitioning algorithm to determine the optimal cut layer in the deep neural network. This algorithm evaluates performance trade-offs across latency, energy consumption, and data privacy, and selects the partition point that minimizes a weighted cost function. Runtime conditions are continuously monitored to adjust the cut point dynamically as needed.

Algorithm 1 Adaptive DNN Partitioning

Require: Number of layers $\overline{N}$ to test, Layers $\{L_i\}_{i=1}^{N}$, Number of Exploratory Rounds ER

▷ Phase 1: Data Collection

for $j = 1$ to N **do**
  for $r = 1$ to ER **do**
    Execute layers L_1 to L_j on IoT device
    Measure $T_{M,j}^{(r)}$, $E_{M,j}^{(r)}$
    Transfer output of L_j to edge server
    Measure $T_{U,j}^{(r)}$, $R_{U,j}^{(r)}$
    Execute layers L_{j+1} to L_N on edge server
    Measure $T_{C,j}^{(r)}$
  end for
end for
Send collected data to edge server

▷ Phase 2: Model Training

Train regression models to predict $T_{M,j}$, $T_{U,j}$, $T_{C,j}$, $E_{M,j}$

▷ Phase 3: Optimal Partition Selection

for $j = 1$ to N **do**
  Predict $T_{M,j}$, $T_{U,j}$, $T_{C,j}$, $E_{M,j}$ (regression models)
  Compute cost $C(j)$ using Equation (1)
end for
$j^* \leftarrow \arg\min_j C(j)$

▷ Phase 4: Continuous Adaptation

while system is operational **do**
  Monitor B, K, and resource usage
  Update regression models with new data periodically
  if significant changes detected **then**
    Re-evaluate j^*
  end if
end while