Content-aware Input Scaling and Deep Learning Computation Offloading for Low-Latency Embedded Vision

Omkar Prabhune*
Purdue University
oprabhun@purdue.edu

Tianen Chen* †
University of Wisconsin–Madison
tianen.chen@wisc.edu

Younghyun Kim
Purdue University
younghyun@purdue.edu

Abstract

Deploying deep learning (DL) models for visual recognition on embedded systems is often constrained by their limited compute power and storage capacity, and has stringent latency and power requirements. As emerging DL applications continue to evolve, they place increasing demands on computational resources that embedded vision systems are unable to provision. One promising solution to overcome these limitations is computation offloading. However, for performance improvements to be realized, it is essential to carefully partition tasks, taking into account both the quality of the data and the communication overhead.

In this paper, we introduce a novel framework for content-aware offloading of DL computations, aimed at maximizing quality-of-service while adhering to latency constraints. Our proposed framework involves the embedded vision system/edge device intelligently compressing data in a content-aware manner using a lightweight model and transmitting it to a more powerful server. The framework consists of two key components: offline training for efficient content-aware data scaling and online control that adapts to the network variations in real-time. To illustrate the effectiveness of our approach, we apply it to multiple downstream tasks such as face identification, person keypoint detection, and instance segmentation, showcasing a significant enhancement in the overall quality of results for various applications.

1. Introduction

Embedded vision has been rapidly gaining traction across various domains, spanning from entertainment [2] to education [11] and healthcare [18]. Its utilization extends into increasingly critical applications, as evidenced by its incorporation into mission-critical contexts [1, 12]. Constraints on their storage, latency, weight, and power limit the computational capabilities of embedded vision systems. Low latency can be a critical requirement in applications such as extended reality, which worsens the challenge. Embedded vision devices are typically equipped with a low-power system-on-chip (SoC) which falls short of meeting the demands of compute-intensive DL tasks. The gap between the computing power available on edge devices and the compute requirement to run DL models is expected to widen as they continue to grow in complexity, facilitating more advanced applications.

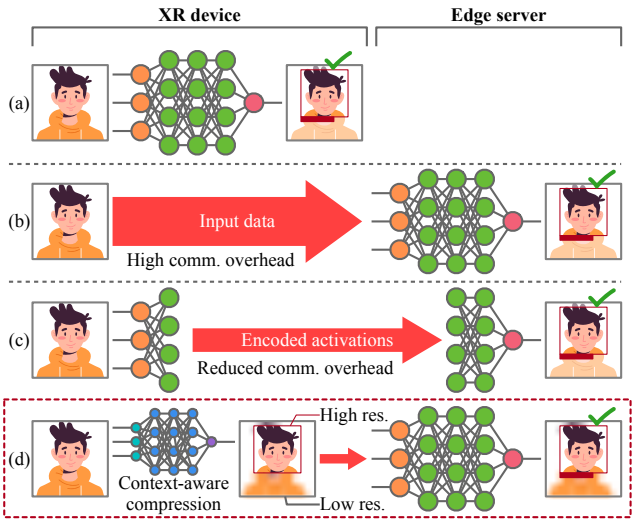


Figure 1. Computation offloading for face identification. (a) No offloading. (b) Full offloading of workload. (c) Partial offloading by partitioning workload. (d) Proposed content-aware offloading.

Computation offloading stands out as a promising solution for enabling compute-intensive DL applications on resource-constrained devices [13]. It involves splitting the workload and leveraging resource-rich edge servers to handle the computationally intensive segments. Consequently, the quality of service (QoS), such as latency and accuracy, can be significantly enhanced. However, it is essential to exercise caution during the workload partitioning process to ensure that it alleviates the burden on the embedded de-

*Equal contribution.
†Currently at Google.

vice while minimizing communication overhead. Inefficient computation offloading can lead to limited or even negative QoS improvements, owing to excessive communication overhead and potential data quality deterioration.

In Fig. 1, we illustrate the concept of computation offloading using DL-based face identification as an example. Fig. 1(a) is the baseline scenario where an embedded vision system such as an extended reality (XR) device handles the task entirely, which might not meet performance requirements. Fig. 1(b) shows complete offloading, where the entire task is executed on the edge server. Although the edge server offers more computational power, transmitting the raw input data results in substantial communication overhead. In partial offloading shown in Fig. 1(c), the DL model is partitioned into two segments so the transmitted intermediate data is minimized after the embedded device executes part of the original model. Here, the DL model is divided into two segments, minimizing the transmission of intermediate data after the embedded device completes the first segment. In this paper, we propose *content-aware computation offloading* as illustrated in Fig. 1(d), where the embedded device intelligently compresses input data using a lightweight model in a content-aware manner, thereby significantly reducing the communication overhead.

More specifically, our framework introduces an intelligent input scaling mechanism on the embedded device using a lightweight model. The compressed data is then transmitted to and processed by the full DL model hosted on the edge server. We refer to this approach as *coarse segmentation*, which efficiently identifies regions-of-interest (ROIs) within the input while reducing the fidelity of non-ROIs to minimize communication overhead, all while preserving the integrity of ROIs. The lightweight model executes a simplified version of the original model's task but remains distinct from it. For instance, in a face identification scenario, the lightweight model focuses on "face detection," a less complex task than "face identification." This process involves preserving the resolution of detected faces while down-scaling the background, generating compressed input data. Subsequently, the edge server performs the more intricate identification task using this compressed, but ROI-preserved data.

The paper's contributions can be summarized as follows:

- We introduce an efficient DL workload offloading approach centered on content-aware coarse segmentation, enabling ROI-preserving data scaling for embedded devices. We introduce a novel training objective, coarse segmentation, to train a lightweight model for the efficient identification of ROIs.
- We propose an optimization framework capable of dynamically managing system operations to adhere to latency constraints in varying wireless conditions. This framework comprises offline characterization of DL workloads and online adaptation of data scaling.
- We implement a comprehensive end-to-end pipeline of our proposed framework in the context of face identification, person keypoint detection, and instance segmentation, showcasing improved accuracy when compared to baseline computation offloading.

2. Related Work

Computation offloading is a promising strategy for facilitating resource-intensive tasks on devices with limited resources. Achieving effective computation offloading hinges on the efficient reduction of communication overhead between the client and the server. In the context of video analysis, leveraging inter-frame similarities can substantially decrease this communication overhead. Since a video frame closely resembles a previously processed or transmitted frame, there is considerable potential to diminish both computation and communication requirements, as noted in [8, 10]. A recent work presents a technique to encode intermediate features to compress the data [17].

Accurate ROI detection plays a pivotal role in optimizing data scaling and encoding. It is essential that this detection process remains computationally efficient to ensure that the advantages of computation offloading are not undermined. One of the fundamental techniques involves straightforward frame-to-frame subtraction to identify significant changes between frames. Advanced ROI detection methods leverage various strategies, including exploiting historical frame data [10], lightweight local object detection technique [6, 16], and the integration of multi-camera networks to reduce overlap [4, 15].

3. Content-aware Input Scaling

Our aim is to design a computation offloading method for embedded systems for efficient DL workloads under latency constraints. We propose a new technique called *content-aware input scaling* to determine key ROI and scale the fidelity of the input data in order to achieve lower communication costs. For the rest of this paper, we focus on a face identification task as the application of the system, but the proposed method can be generally applicable to any compute-heavy DL workload. Further, we show the generalizability of our approach to other typical downstream tasks such as person keypoint detection and instance segmentation.

3.1. Computation offloading under latency constraints

Let us consider a scenario where an embedded device (the client) performs a face identification task that compares a face in a captured image to a database of faces to find a matching identity. First, embedded devices generally do not have the computing power to efficiently perform such a

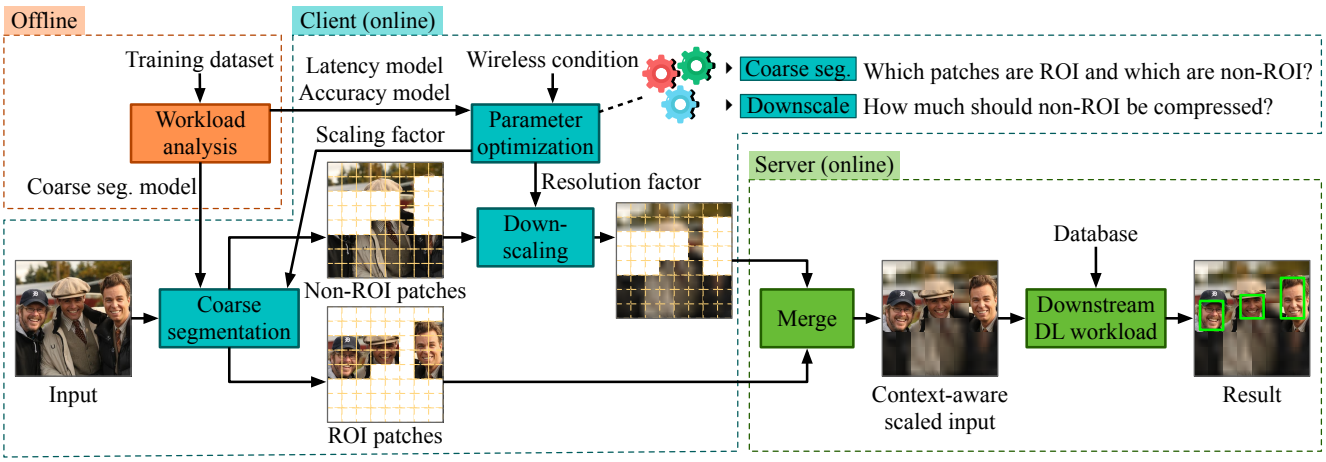


Figure 2. Computation offloading of face identification from an embedded vision system such as extended reality (XR) device (client) to a server. Image from IMDb-Face dataset [14].

complex task. In addition, due to the memory and storage constraints, storing the entire face database in the client will be inefficient or infeasible due to the large storage requirement. Moreover, a face database contains sensitive personal data, which cannot be in distributed devices due to privacy concerns. For these reasons, the captured image cannot be processed on the client but must be sent to an edge server (the server) for processing.

Under a latency constraint, computation offloading must be performed with additional overheads taken into account, including data compression and transmission. In order to minimize the overhead the compression should have three crucial properties:

- Compressive: It should reduce the amount of data transmission substantially to reduce communication overhead.
- Efficient: The compression process itself should be lightweight to minimize compute overhead.
- Adaptive: The compression ratio should be able to adapt to varying wireless conditions to meet the latency constraint.

In the rest of this section, we describe how we design content-aware input scaling to meet these requirements.

3.2. Content-aware input scaling pipeline

The proposed content-aware input scaling pipeline is composed of four main components as illustrated in Fig. 2.

- *Workload analysis* is an offline step that analyzes the latency and accuracy of the target DL workload. The analysis is discussed in Sec. 3.4. We also generate a coarse segmentation model in this step, which is used in the next component.
- *Coarse segmentation* is an online step performed by the client. The input is partitioned into ROI patches and non-ROI patches using the coarse segmentation model. This step is further discussed in Sec. 3.3.

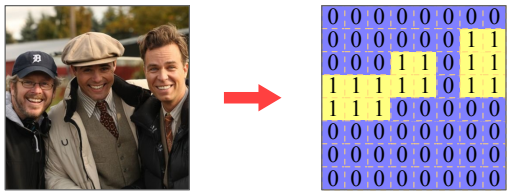


Figure 3. Training coarse segmentation model.

- *Parameter optimization* derives two control parameters, scaling factor and resolution factor, used in the coarse segmentation and downscaling steps, respectively, as discussed in Sec. 3.5.
- *Downscaling* is where the non-ROI patches are downscaled based on the resolution factor determined in the parameter optimization step.

The ROI patches and the downscaled non-ROI patches are then transmitted to the server, where they are merged and processed by the downstream DL workload (face identification in this example). The following subsections describe each component of the pipeline in detail.

3.3. Coarse segmentation

The coarse segmentation step is to partition an input image into ROI patches that contain ROIs and non-ROI patches that do not. In order to meet the goals discussed in Sec. 3.1, we introduce a new task called *coarse segmentation* and propose a lightweight convolutional neural network (CNN)-based coarse segmentation method.

In coarse segmentation, the objective is to predict if a patch contains the pixels of ROI or not, where a patch is one piece of an image equally divided into $N \times N$. For the downstream task of face identification, we consider face pixels as our ROI. Each patch is labeled either '1' if the patch

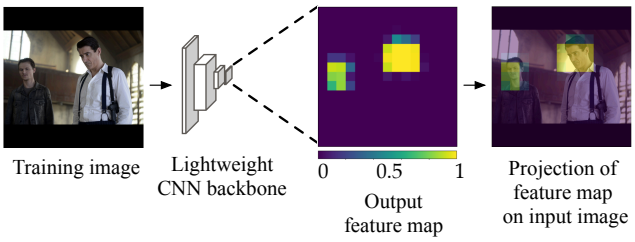


Figure 4. Design flow of client-side coarse segmentation model.

contains any face pixels or '0' otherwise. Thus, the ground truth label is a 2-D binary mask of dimensions $N \times N$ as shown in Fig. 3.

In order to perform coarse segmentation, we redesign and adapt a lightweight CNN backbone of the image classifier to predict the ROI via its feature maps. Different layers in a CNN classifier learn a hierarchy of features, from simple and local features such as edges to more complex and global features such as object parts and, eventually, entire objects or patterns. The deeper layers capture increasingly abstract representations, making the network capable of recognizing and classifying complex patterns in the input data. Analysis of intermediate feature maps also shows that the regions that are activated exhibit a strong correlation with the regions containing semantic objects [14]. This indicates the possibility of approximating the positions of important regions within the image by analyzing the feature maps. Therefore, we redesign the model and training objective for a lightweight CNN backbone of an image classifier such that it predicts ROI via its feature maps.

Fig. 4 shows our approach to performing coarse segmentation. We train a lightweight CNN backbone such that the output feature map shows high activations at the spatial locations of ROI and low activations otherwise. After applying the sigmoid activation function, the output is interpreted as the probability of the block containing ROI pixels. When this feature map is projected on the original image dimensions, we obtain the ROI and non-ROI patches as shown in Fig. 4.

For training the model, the original RGB image serves as the input training image and the coarse segmentation mask serves as the ground truth. Loss is calculated by comparing the output feature map and the ground truth mask and the model weights are updated using backpropagation.

Next, we determine the number of patches that will be downscaled in resolution using a parameter called *scaling factor* $0 \leq S_f \leq 1$. It is defined as

$$S_f = 1 - \frac{N_{ds}}{N^2}, \tag{1}$$

where N_{ds} is number of downscaled patches. These patches are selected based on the output feature map of the coarse segmentation model. The patches are sorted based on their

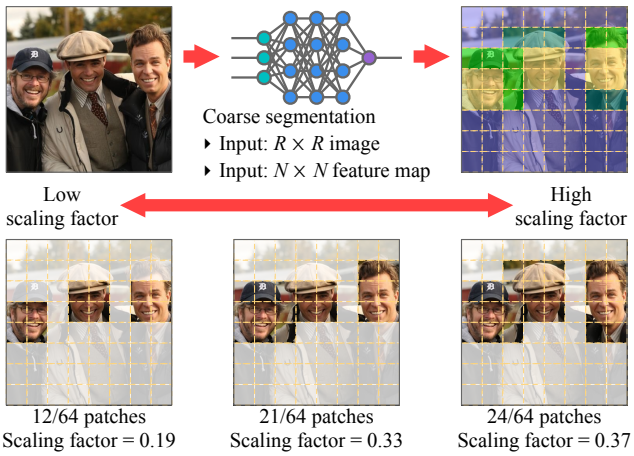


Figure 5. Example of coarse segmentation with three different scaling factors. $N = 8$.

probability scores from the feature map. A low probability score indicates that the patch is not important for the downstream task and hence can be downscaled in resolution. Therefore, we downscale N_{ds} patches with the lowest probability scores. Fig. 5 shows the coarse segmentation of an image with three different scaling factors. Within Fig. 5, we see that as the scaling factor increases, the amount of high-resolution original input image increases. Our system tunes a balance between high accuracy, high scaling factor, and low latency, low scaling factor inputs. This parameter will be explicitly defined within our system design and dynamically adjusted depending on application constraints.

3.4. Workload analysis

The workload analysis is an offline step to understand the computation and communication requirements of the workload in the context of the DL task. This step builds a latency profile and an accuracy profile for a given training dataset. It also produces a coarse segmentation model, which has been described in Sec. 3.3.

First, a latency model is built based on empirical measurement and analytical estimation. The total latency on the client side to process an image is the sum of the acquisition latency L_{acq} to obtain the source image, the processing latency L_{proc} to split the image into ROI and non-ROI patches, and the transmission latency L_{tx}. We do not consider the processing latency on the server side, since it is generally negligible due to its high performance. The total latency should be less than or equal to the latency constraint L_c.

$$L_{acq} + L_{proc} + L_{tx} \leq L_c \tag{2}$$

Since the image resolution R is fixed, L_{acq} is constant. The majority of L_{proc} is for coarse segmentation with some for image partitioning, and this is also constant for a fixed R on

a given client device. Therefore, these two latency factors can be empirically measured on the target client platform.

On the other hand, L_{tx} is the most dominant and variable latency factor. First, we downscale non-ROI patches by a resolution factor R_f, which is the percentage of how much we downscale the individual patch resolution height and width. Without downscaling, the baseline transmission latency L_{tx}^{base} would be directly proportional to the size of the image and inversely proportional to the data rate.

$$L_{tx}^{base} = c\frac{R^2}{\gamma}, \tag{3}$$

where c is a constant that relates the image size to the data size, and γ is the data rate. Then, with the proposed scaling, for a given S_f and R_f, L_{tx} is reduced to

$$L_{tx} = c\frac{R^2}{\gamma}\left(S_f + {R_f}^2(1 - S_f)\right). \tag{4}$$

Since $0 \leq R_f \leq 1$, L_{tx} increases as either S_f or R_f increases.

This step also generates an accuracy profile. It characterizes the accuracy of the downstream task by running it with different S_f or R_f settings. This step is performed offline on a validation dataset. As the quality of the image also increases as S_f or R_f increases, the accuracy also increases. Considering these latency and accuracy characteristics, the next step is to find the optimal values of S_f or R_f that maximize the accuracy while meeting the constraint (Eq. (2)) as discussed in the next subsection.

3.5. Parameter optimization

In order to determine optimum S_f and R_f, we use a combination of latency analysis combined with downstream task accuracy $A_s(S_f, R_f)$ calculated offline on the validation set. While the parameters are selected online, the latency and accuracies are already calculated, ready to conform to constraints. We dynamically choose parameters (S_f, R_f) such that we scale the input image to match application-defined latency constraint L_c while maintaining the highest level of accuracy based on validation data accuracy for the downstream task.

Let C_a represent all possible (S_f, R_f) configurations corresponding to unique latencies, L. We find a subset C_s as follows:

$$C_s = \{A_i : L_{tx}(S_f, R_f) < L_c\}, \tag{5}$$

where C_s is the subset of configurations satisfying the latency constraint within C_a, and A_i is the accuracy of the particular configuration. Thus, to obtain the maximum accuracy for a given latency constraint, we select the configuration that maximizes accuracy within C_s:

$$(S_{fo}, R_{fo}) = \underset{A_i \in C_s}{\arg\max}(A_o) \tag{6}$$

where (S_{fo}, R_{fo}) is the optimum scaling and resolution factors that satisfy constraints, and A_o is the highest accuracy within the subset of configurations C_s. With this, we can say that (S_{fo}, R_{fo}) is the most latency-constrained, highest-accuracy configuration for content-aware resolution scaling for a given data rate.

4. Experiments

In this section, we demonstrate the efficacy of our content-aware efficient edge offloading scheme for object detection.

4.1. Experimental setup

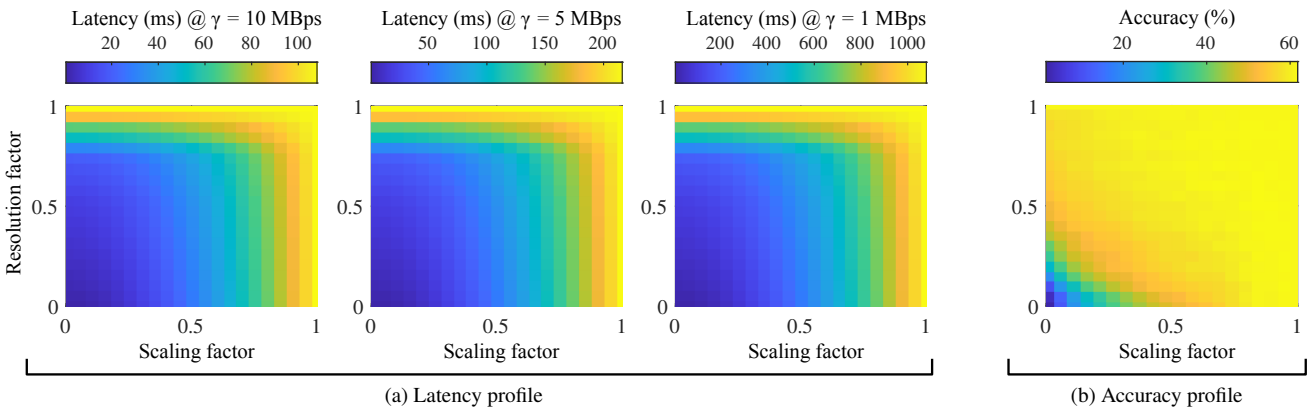
Within our system model, we use the Raspberry Pi 4 as our embedded device (client) running a redesigned MobileNetV3-Small backbone as our coarse segmentation model. We use the Raspberry Pi for flexibility in software modification, but the proposed methodology is largely hardware-agnostic.

Face identification involves detecting faces in an input image and predicting the individuals by matching them against a gallery of labeled images of individuals. While we use face identification for this experiment, coupled similar lightweight models and downstream tasks can also be used in our agnostic design as discussed in Sec. 5.

4.2. Dataset

For training the coarse segmentation model, we use 5000 training images and 2000 test images from the IMDb-Face dataset [14]. The images in this dataset are official photos, lifestyle photos, and movie snapshots of celebrities sourced from the IMDb website. The images display large variations in terms of scale, pose, lighting, and occlusion of faces, as well as number of faces per image. Specifically, movie snapshots provide a diverse dataset for testing the robustness of our methodology. The ground truth coarse segmentation mask is created using the bounding box annotations provided in the dataset. If a bounding box is present in an image patch (partially or completely), that patch is marked as a region of interest. Here, we treat the region inside the face bounding box as the region of interest. Thus, the regions within a bounding box for a face were labeled as '1' and '0' otherwise.

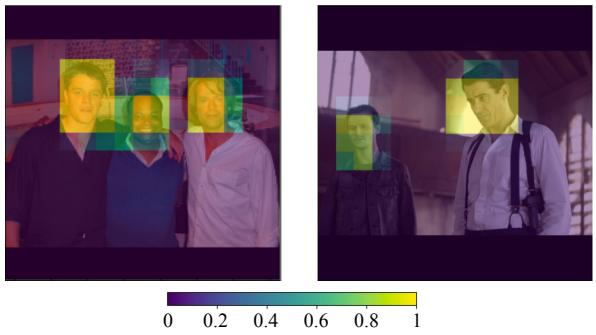
For the downstream task of face identification, we use a subset of 100 identities from the IMDb-Face dataset which comprises annotated faces of celebrities as our test dataset. We construct an annotated gallery of images of at most ten images per individual against which the test images are matched for identification. The test set comprises 1190 images that do not overlap with the gallery. We ensure that the training data for coarse segmentation does not overlap with the test data for the face identification downstream task.



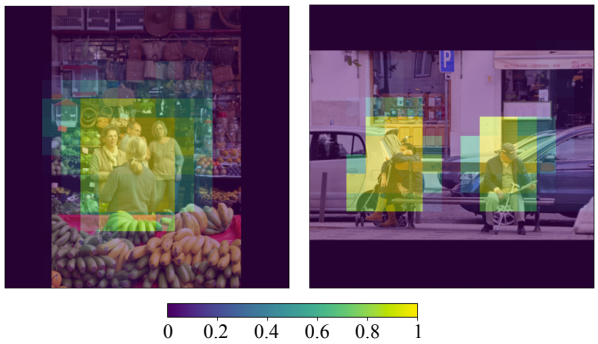
(a) Latency profile

(b) Accuracy profile

Figure 6. (a) Latency model $L_{tx}(S_f, R_f)$ for three data rates, 10, 5, and 1 MBps. (b) Accuracy profile $A_{tx}(S_f, R_f)$.



(a) Face ROI predictions



(b) Person ROI predictions

Figure 7. Sample outputs from coarse segmentation model for (a) face ROI detection and (b) person ROI detection.

4.3. Client-side coarse segmentation model

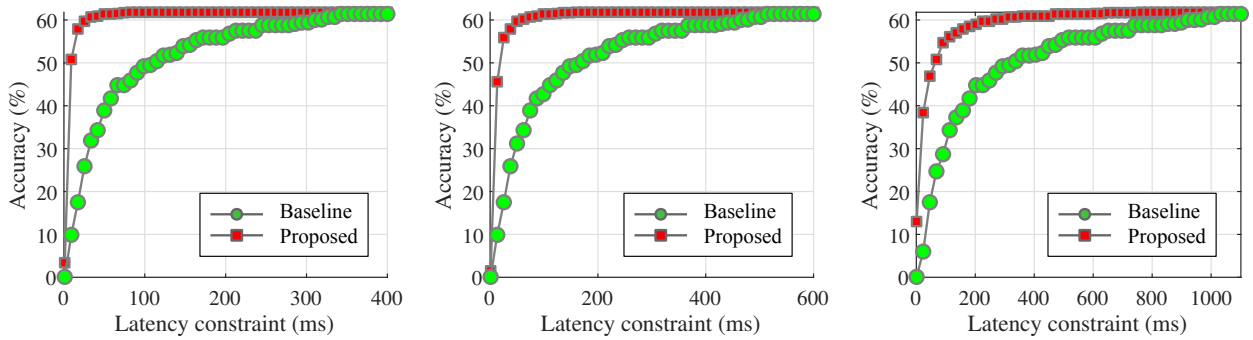
We redesign MobileNetV3-Small [5] classification model to adapt to the task of coarse segmentation. MobileNetV3 architecture design is well-suited for deployment on resource-constrained environments due to its efficiency and competitive performance as compared to other CNN backbones [5].

In the MobileNetV3-Small model, we remove the last 2 building blocks as well as the convolution and pooling layers that follow these blocks. We add a point-wise convolution layer to this trimmed model to produce a feature map with a channel dimension of 1. The input resolution is 600×600 pixels ($R = 600$), and the output feature map resolution is 15×15 ($R = 15$).

We use the binary cross-entropy loss to train the model for 500 epochs with Adam optimizer and a learning rate of 0.001. We use cross-validation to select the best weights with minimum validation loss. On evaluating the model on test data, we obtain a PR-AUC (area under the precision-recall curve) score of 0.82. The model is trained in the TensorFlow framework and converted to TFLite for portability on Raspberry Pi. Post-training dynamic range quantization is used to convert the model from TensorFlow to TFLite. The execution time of the coarse segmentation model and image partitioning (L_{proc}) is 46 ms per image. This short execution time is compared to full face identification on Raspberry Pi, taking more than 2.8 seconds for a small subset of 3 identities. Thus, coarse segmentation is much faster than full face identification on the Raspberry Pi without any of the privacy concerns that accompany a face database on the client. Fig. 7a shows some sample outputs of the trained model for face ROI detection.

4.4. Latency and accuracy profiling

In our design, since typical Wi-Fi connections greatly depend on varying transmission upload rates, we analytically find latency according to the methods described in Sec. 3.4. We can visualize the latency surface when plotting the surface of Eq. (4) in Fig. 6a with an image resolution of $R = 600$, and an estimated client upload data rate of 10 MBps, 5 MBps, and 1 MBps. We choose 20 different S_f values sampled from 0 to 1. We use 21 different R_f uniformly spaced from 0 to 1. This forms 420 different configurations of (R_f, S_f). Fig. 6a provides analytical latency calculations given different data rates, such that we observe latency of transmission increases



(a) γ = 10 MBps (b) γ = 5 MBps (c) γ = 1 MBps

Figure 8. Accuracy of face identification after computation offloading for different latency constraints and data rates.

proportionally to S_f and R_f.

Fig. 6b displays the accuracy profile of our face identification model given different S_f and R_f configurations, which are calculated offline. This accuracy profile is generated by evaluating the accuracy on a validation dataset that is scaled using the proposed content-aware scaling for all 420 (R_f, S_f) configurations.

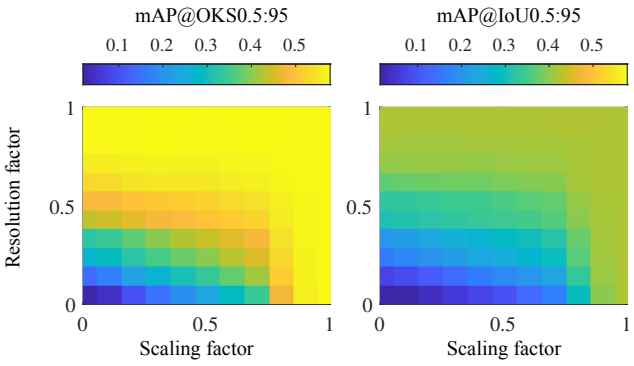
As discussed in Sec. 3.5, given the latency constraints, we find C_s, i.e. a subset of (R_f, S_f) configurations using the latency profile. The accuracy profile is then looked up to choose the (R_f, S_f) configurations with the highest accuracy, which is the optimum configuration for the given latency constraint. This configuration is used to downscale the image in the proposed content-aware manner and is transmitted to the edge server which performs the downstream tasks.

4.5. Server-side downstream task

As a baseline design, we consider an offloading system that transmits an input image by downscaling it without content awareness. The entire image is downscaled to meet the latency constraint without partitioning into ROI and non-ROI.

For the downstream task, we use off-the-shelf (pre-trained) ArcFace [3] model for face identification. We evaluate our approach against the baseline using the dataset described in Sec. 4.2. In Fig. 8, accuracy scores for face identification are plotted as a function of latency constraint for various data rate settings. Our method outperforms the baseline resolution scaling across all the data rates with 16.5%–17.3% higher accuracy on average.

Given the latency, our method achieves better performance by dynamically selecting (R_f, S_f) configuration based on the latency and accuracy profiling done in the offline stage. In the baseline approach where the entire image is downscaled to meet the latency requirement, important details in the image (e.g. facial features in this case) are lost due to downscaling, which hampers the accuracy of the face identification task. On the other hand, our method better preserves the important details while meeting the latency requirement by intelligently preserving the resolution of important regions in the image leading to better performance. Across different data rates, the improvement in accuracy is more pronounced when the latency requirement is stringent as shown in Fig. 8.



(a) Keypoint detection (b) Instance segmentation

Figure 9. Accuracy profiling for (a) Person keypoint detection (b) Person instance segmentation.

5. Generalizability to Other Downstream Tasks

In this section, we show the generalizability of our approach by applying it to other typical downstream tasks such as person keypoint detection and instance segmentation.

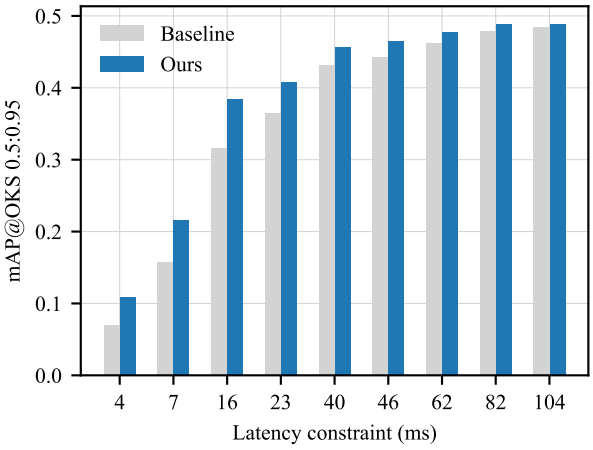
The person keypoint detection task consists of simultaneous person detection and their keypoint localization. For this task, we regard the pixels corresponding to persons as our regions of interest. We use the COCO dataset [9] for training and evaluation of the approach. This dataset consists of annotations of 17 keypoints for the instances of persons in an image. The coarse segmentation model is trained to predict the regions in the image that contain persons. To generate

ground truth for the coarse segmentation task, we use the instance segmentation masks in COCO annotations. If an instance segmentation mask is present in a patch (partially or fully), the patch is marked as '1' and '0' otherwise. From the training dataset, we keep out 400 images to generate our accuracy profile for various configurations of S_f and R_f as shown in Fig. 9.

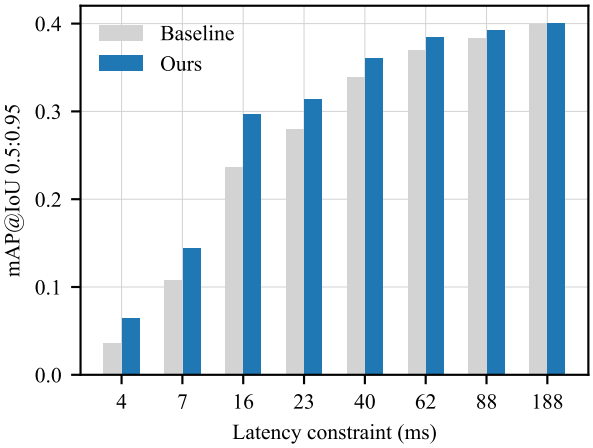
We use the same lightweight model as face ROI detection described in Sec. 4.3. Binary cross-entropy loss is used to train the model for 200 epochs with Adam optimizer and a learning rate of 0.001. We use the value of cross-validation (20% of data split from the training set) to select the best weights with minimum validation loss. On evaluating the trained coarse segmentation model on test data (referred to as val2017 in the COCO dataset), we obtain an accuracy of 91.5%. Fig. 7b shows some sample outputs of the trained model.

For the downstream task of person keypoint detection, we use pre-trained YOLOv8n-pose model [7]. Evaluation of our approach is done using COCO validation set. For a given data rate and different latency constraints, images are scaled using our proposed content-aware manner by selecting the optimum (R_f, S_f) configuration. These intelligently downscaled images are input to the keypoint detector model and evaluated against the ground truth. This performance is compared against the evaluation metrics obtained for images that have been downscaled using the baseline approach. The scores mAP@OKS0.5:0.95 metric are shown in Fig. 10a. Specifically in the low latency constraint region (16 ms), our approach outperforms the baseline by 6.8% mAP score.

Similar experiments are performed on the person instance segmentation task. Evaluation on the COCO dataset and comparison with the baseline is shown in Fig. 10b. For low latency constraint (16 ms), our approach outperforms the baseline by 5.9% mAP score.



(a) Person keypoint detection



(b) Person segmentation

Figure 10. Comparison of the proposed approach with the baseline for the downstream tasks of (a) person keypoint detection, and (b) person segmentation for γ = 5 MBps.

6. Conclusions

Computation offloading, if properly designed, can significantly improve the performance and latency of DL applications on embedded vision systems. We introduced a method to improve computation offloading from an embedded device to an edge server. Our method optimizes the scaling of wirelessly transmitted images by identifying ROI to preserve its fidelity while reducing the fidelity of non-ROI background. We used face identification as the downstream DL task as an example and demonstrated a significant improvement in accuracy under the same latency constraint. We achieved 16.5%–17.3% higher downstream task accuracy for three different data rates under the same latency constraints as the baseline naive resolution scaling. Similarly, performance gains are obtained for other downstream tasks of keypoint detection and instance segmentation.

For future work, our method can be extrapolated to a variety of different downstream tasks, such as pose segmentation. Additionally, multiple other control knobs, such as variable resolution adjustment for higher R_f around ROI or variable data rate fluctuation, can be examined for accuracy preservation.

7. Acknowledgements

This work was supported by the National Science Foundation under awards CNS-1845469 and CNS-2112562, and by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00261534).

References

[1] Manish Bhattarai, Aura Rose Jensen-Curtis, and Manel Martínez-Ramón. An embedded deep learning system for augmented reality in firefighting applications. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1224–1230, 2020. 1

[2] Shaveta Dargan, Shally Bansal, Munish Kumar, Ajay Mittal, and Krishan Kumar. Augmented reality: A comprehensive review. *Archives of Computational Methods in Engineering*, 30(2):1057–1080, 2023. 1

[3] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2019. 7

[4] Hongpeng Guo, Shuochao Yao, Zhe Yang, Qian Zhou, and Klara Nahrstedt. CrossRoI: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, pages 186–199, 2021. 2

[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 6

[6] Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu, and Yunxin Liu. Flexible high-resolution object detection on edge devices with tunable latency. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 559–572, 2021. 2

[7] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLOv8, 2023. 8

[8] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 359–376, 2020. 2

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. 7

[10] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pages 1–16, 2019. 2

[11] Jesús López-Belmonte, Antonio-José Moreno-Guerrero, Juan-Antonio López-Núñez, and Francisco-Javier Hinojo-Lucena. Augmented reality in education. A scientific mapping in Web of Science. *Interactive Learning Environments*, 31(4):1860–1874, 2023. 1

[12] Oceane Peretti, Yannis Spyridis, Achilleas Sesis, Georgios Efstathopoulos, Anastasios Lytos, Thomas Lagkas, and Panagiotis Sarigiannidis. Augmented reality training, command and control framework for first responders. In *South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–5, 2022. 1

[13] Xukan Ran, Haoliang Chen, Zhenming Liu, and Jiasi Chen. Delivering deep learning to mobile devices via offloading. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network)*, pages 42–47, 2017. 1

[14] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3119–3127, 2015. 3, 4, 5

[15] Ziyi Wang, Xiaoyu He, Zhizhen Zhang, Yishuo Zhang, Zhen Cao, Wei Cheng, Wendong Wang, and Yong Cui. Edge-assisted real-time video analytics with spatial–temporal redundancy suppression. *IEEE Internet of Things Journal*, 10 (7):6324–6335, 2022. 2

[16] Zheng Yang, Xu Wang, Jiahang Wu, Yi Zhao, Qiang Ma, Xin Miao, Li Zhang, and Zimu Zhou. EdgeDuet: Tiling small object detection for edge assisted autonomous mobile vision. *IEEE/ACM Transactions on Networking*, 2022. 2

[17] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*, pages 476–488, 2020. 2

[18] James Zhang, Victor Lu, and Vikas Khanduja. The impact of extended reality on surgery: a scoping review. *International Orthopaedics*, 47(3):611–621, 2023. 1