ELF: Accelerate High-resolution Mobile Deep Vision with Content-aware Parallel Offloading

Wuyang Zhang
wuyang@winlab.rutgers.edu
Rutgers University
Piscataway, NJ, USA

Zhezhi He
zhezhi.he@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Luyang Liu
luyangliu@google.com
Google Research
Mountain View, CA, USA

Zhenhua Jia
zjia@nvidia.com
NVIDIA Corporation
Holmdel, NJ, USA

Yunxin Liu
yunxin.liu@microsoft.com
Microsoft Research
Beijing, China

Marco Gruteser
gruteser@google.com
Google Research
Mountain View, CA, USA

Dipankar Raychaudhuri
ray@winlab.rutgers.edu
Rutgers University
Piscataway, NJ, USA

Yanyong Zhang
yanyongz@ustc.edu.cn
Corresponding author, University of Science and Technology of China
Hefei, China

Outline

- Motivations and Challenges
- ELF Overview
- Design Details
- Evaluation and Results
- Conclusions

Motivations

Existing offloading methods are insufficient:

- Use **low-resolution images** to make inference light-weight
- Only consider **single pair of server and client**, assuming no competing clients or extra edge resources available
  - Heterogeneous resource demand and highly dynamic workload
  - Resource fragmentation/waste
- To meet latency requirement and heterogeneous edge computing resources:
  - Offload smaller inference tasks **in parallel** to **multiple** edge servers

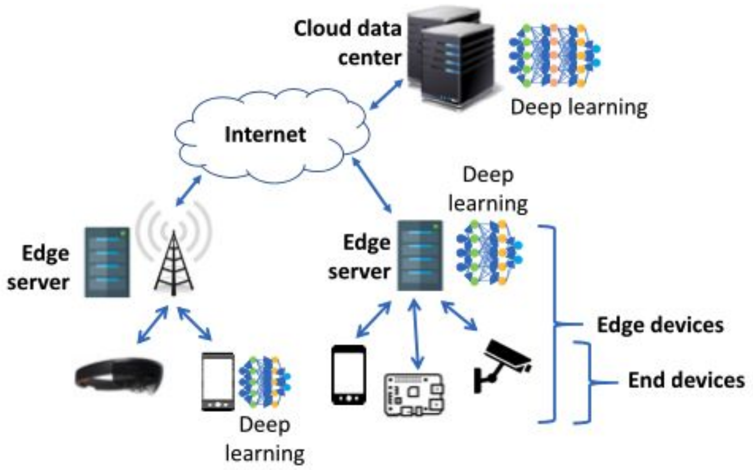

Fig. 1. ***Deep learning can execute on edge devices (i.e., end devices and edge servers) and on cloud data centers.***

Challenges

- How to partition the inference tasks while maintaining accuracy?
- How to distribute the tasks to multiple servers to minimize total latency?
- How to minimize the overhead of frame partitioning on mobile devices?

ELF: content-aware parallel offloading

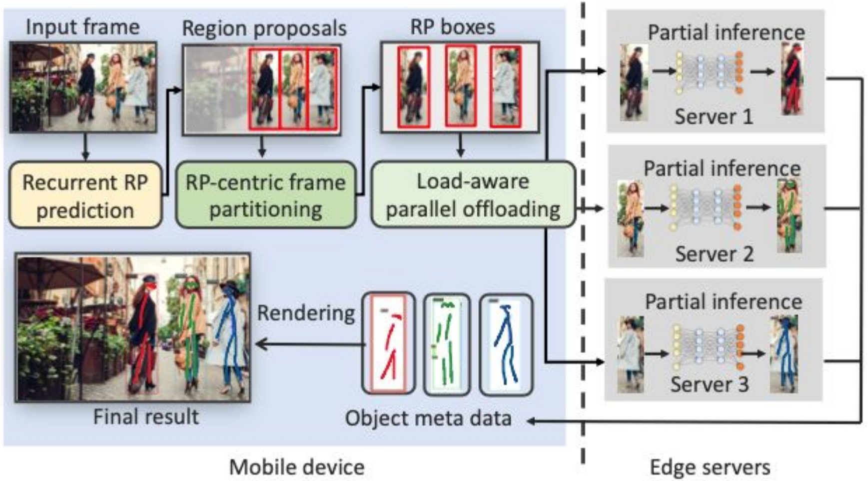

Figure 2: ELF system architecture. We explain the architecture using a multi-person pose estimation example with three edge servers

- Content-aware frame partitioning
- Load-aware parallel offloading

❑ First to propose a high-resolution mobile deep vision tasks acceleration system

❑ Prototype system with comprehensive experiments showing upto **4.85x speedup** and **52.6% less bandwidth** on 4 edge servers with **less than 1% accuracy loss**

How to partition the inference task?

- Equal partition does not work
  - Intersection reduces accuracy
  - Waste resource for redundant background pixels

(a) Equal partitioning (b) Ideal partitioning

Figure 1: Examples of video frame partitioning. The simple partitioning method in (a) split pixels of the same object into multiple parts and yield poor inference results. We can achieve much better partitioning using ELF, close to the ideal partitioning shown in (b)

Recurrent Region Proposal

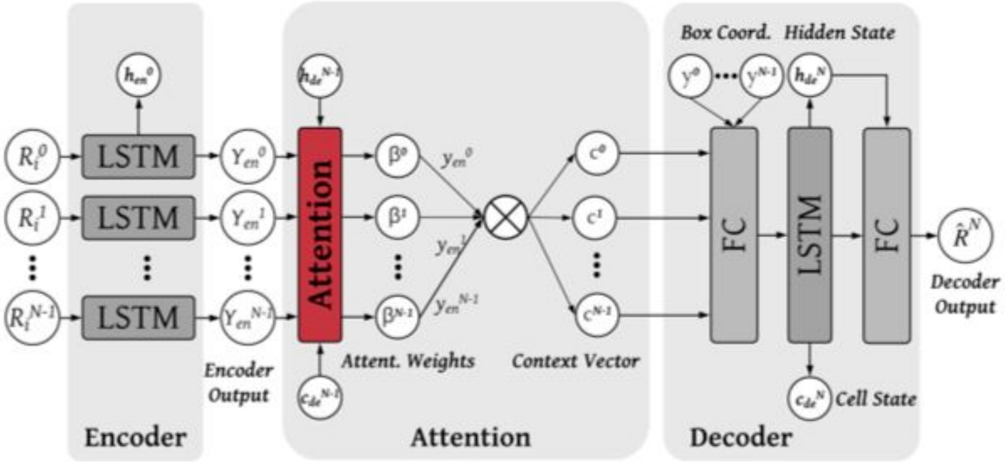

Figure 3: Our attention-based LSTM network

$$
\begin{aligned}
\min_{\theta} \mathcal{L}(\hat{\boldsymbol{R}}_i^t, \boldsymbol{R}_i^t) = &\min \textstyle\sum_i \big[(\hat{x}_{i,\mathrm{tl}}^t - x_{i,\mathrm{tl}}^t)^2 + (\hat{y}_{i,\mathrm{tl}}^t - y_{i,\mathrm{tl}}^t)^2 \\
&(\hat{x}_{i,\mathrm{br}}^t - x_{i,\mathrm{br}}^t)^2 + (\hat{y}_{i,\mathrm{br}}^t - y_{i,\mathrm{br}}^t)^2 + (\hat{a}_i^t - a_i^t)^2\big] \\
\text{s.t.} \quad &\hat{\boldsymbol{R}}_i^t = f(\{\boldsymbol{R}_i^{t-N}, \boldsymbol{R}_i^{t-N+1}, \ldots, \boldsymbol{R}_i^{t-1}\}, \boldsymbol{\theta})
\end{aligned} \tag{1}
$$

where the vector $\boldsymbol{R}_i^t$ denotes the i-th ground-truth RP at frame t, and $\hat{\boldsymbol{R}}_i^t$ is the predictive RP counterpart. Both $\boldsymbol{R}_i^t$ and $\hat{\boldsymbol{R}}_i^t$ consist of $[x_{\mathrm{tl}}, y_{\mathrm{tl}}, x_{\mathrm{br}}, y_{\mathrm{br}}, a_i]$ as the x, y coordinates of RP's top-left and bottom-right corners, and the area, respectively. $\boldsymbol{\theta}$ is the model parameters of LSTM. Also, a_i^t is the RP's area calculated based on x_{tl}, y_{tl}, x_{br}, and y_{br},. Further, N is the number of previous frames used in the prediction network $f(\cdot)$. Next, we explain our algorithmic effort in minimizing the prediction error as calculated in Eq. (1).

Region Proposal Indexing

- Many vision applications output labels in random orders, making it hard to track and match RP across frames
- ELF:
    - RP position shift < 0.02
    - RP area shift < 0.2

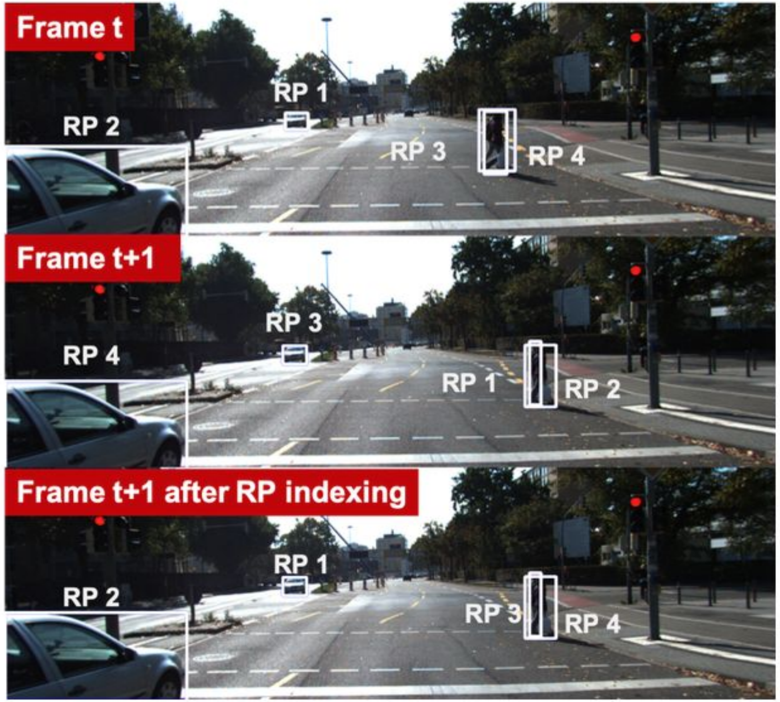

Figure 4: An example result for RP indexing

RP Expansion

- Expand bounding box by p% to cover all related pixels

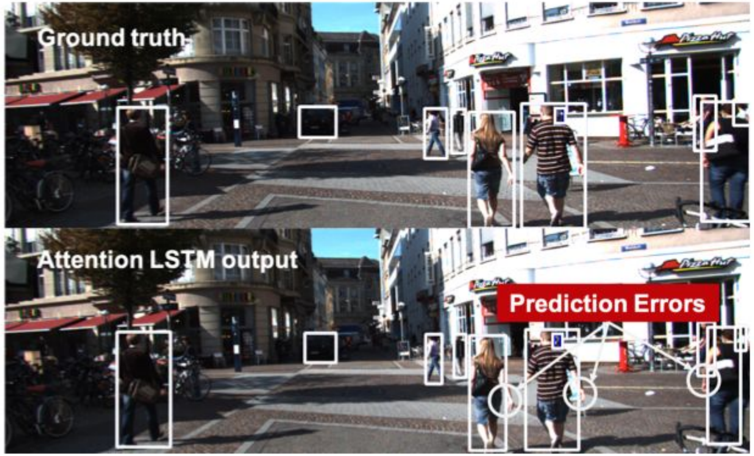

Figure 5: An example prediction error. Part of the objects are outside of the predicted RP bounding box

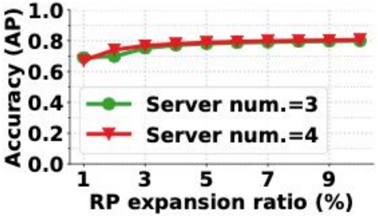

Figure 14: Inference accuracy vs RP expansion ratio

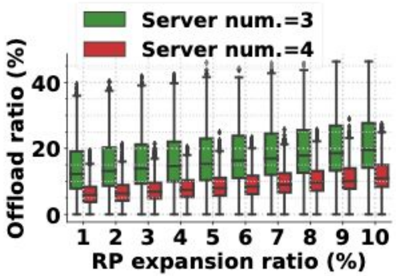

Figure 15: Offload ratio vs RP expansion ratio

Objects When First Appear

- Low Resolution Compensation (LRC)
    - Down-sample high-resolution frames for new object detection
    - Run LRC once per n frames

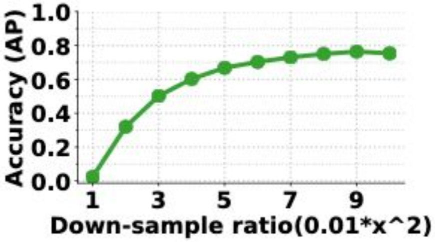

Figure 16: Inference accuracy vs downsample ratio

Content and Resource-aware Partitioning and Offloading

Accordingly, the optimization objective can be written as:

$$
\begin{aligned}
&\min\ \max(\{T_k^t\}) \quad k \in [1, \ldots, N'], \\
&s.t.\ T_k^t = T_{\text{rps},k}^t + T_{\text{lrc},k}^t \cdot \mathbf{1}_{(t \bmod n=0)} \cdot \mathbf{1}_{(\arg\max\{p^t\}=k)}, \\
&T_{\text{rps},k}^t \approx \frac{C_{\text{rps},k}^t}{p_k^t},\ T_{\text{lrc},k}^t \approx \frac{C_{\text{lrc},k}^t}{p_k^t}
\end{aligned}
\tag{6}
$$

- Estimate resources through passive profiling
- Estimate RP computation cost based on RP's area

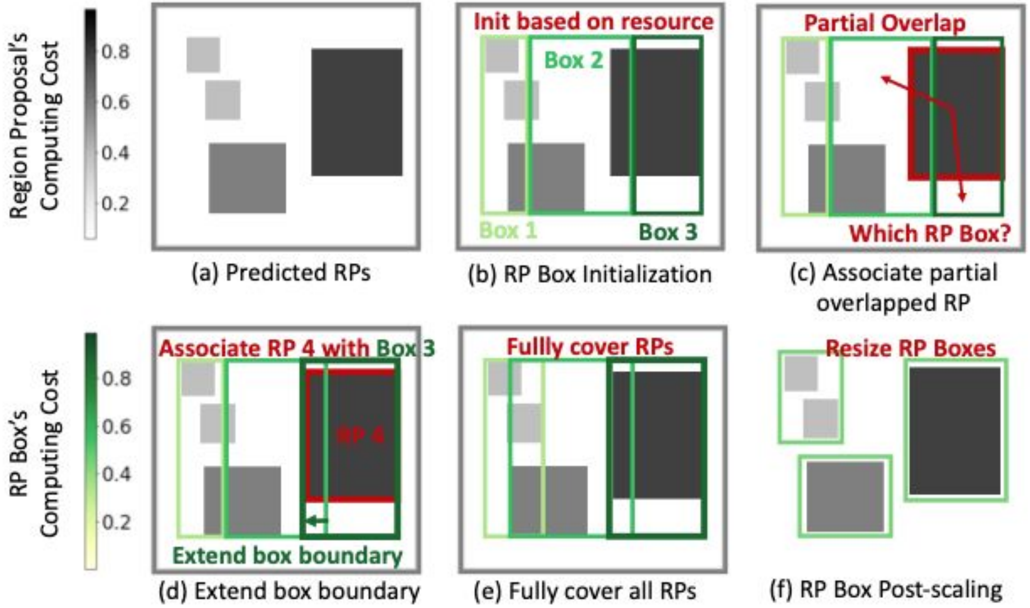

Figure 6: RP-centric frame partitioning pipeline

Experiment Setup

- Integrated ELF with 10 state-of-the-art DL networks with 3 types of applications
  - Instance segmentation, multi-object classification, multi-person pose estimation
- 4 mobile platforms
  - Pixel4, Nexus 6P, Jetson Nano, Jetson TX2
- Upto 5 edge servers: each with Tesla P100 GPU
- Networks: WiFi6 and LTE
- Baseline
  - Single Offloading (SO)
  - existing offloading work: Filter-Forward

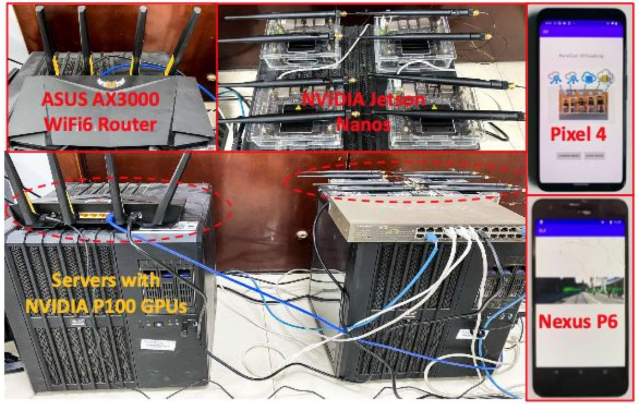

Figure 7: Our experimental evaluation hardware platform

Latency and Accuracy

- ELF-1: average 1.39x, upto 5.43x with ELF-5
- FF: 1.56x, not increasing with # servers

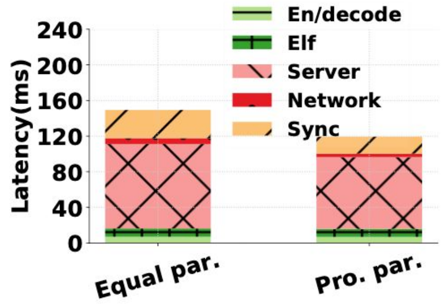

Figure 12: latency vs RP box partitioning schemes

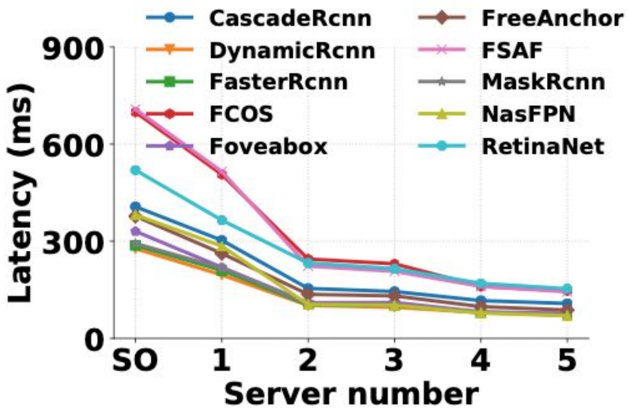

Figure 8: End-to-end latency vs server numbers

Table 1: Comparisons of inference accuracy (AP) in three deep vision applications: instance segmentation [33], object classification [63], and pose estimation [67]

Deep Vision Applications	Accuracy (AP)				
	TX2	Nano	SO	FF [19]	Elf
Instance Segmentation	0.803	0.803	0.803	/	0.799
Object Classification	0.672	0.672	0.672	0.605	0.671
Pose Estimation	0.661	0.661	0.661	/	0.654

Overhead on Mobile Device

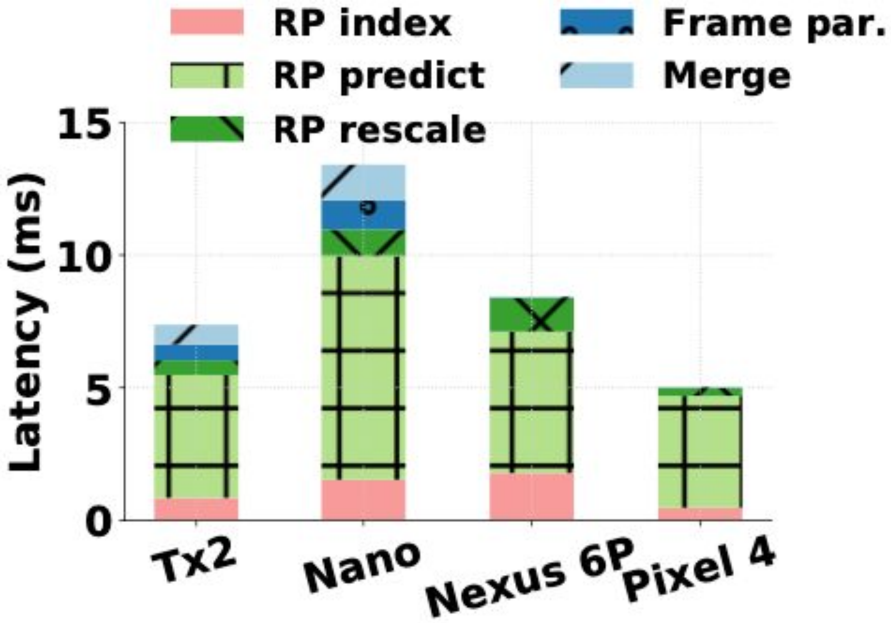

Figure 13: System overheads of ELF functions

Conclusions

- Recurrent region proposal prediction
- Content-aware video frame partitioning algorithms
- Upto 4.85x speedup and 52.6% less bandwidth on 4 edge servers with less than 1% accuracy loss
- Future work
    - Add model parallelism
    - Consider network conditions on task assignments
    - Efficient model design to better benefit from parallel offloading