Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing

E. Li, L. Zeng, Z. Zhou and X. Chen

IEEE Transactions on Wireless Communications, vol. 19, no. 1, pp. 447-457, Jan. 2020

Presenter: Yongwoo Kim

https://sites.google.com/view/yongwookim

June 23, 2020

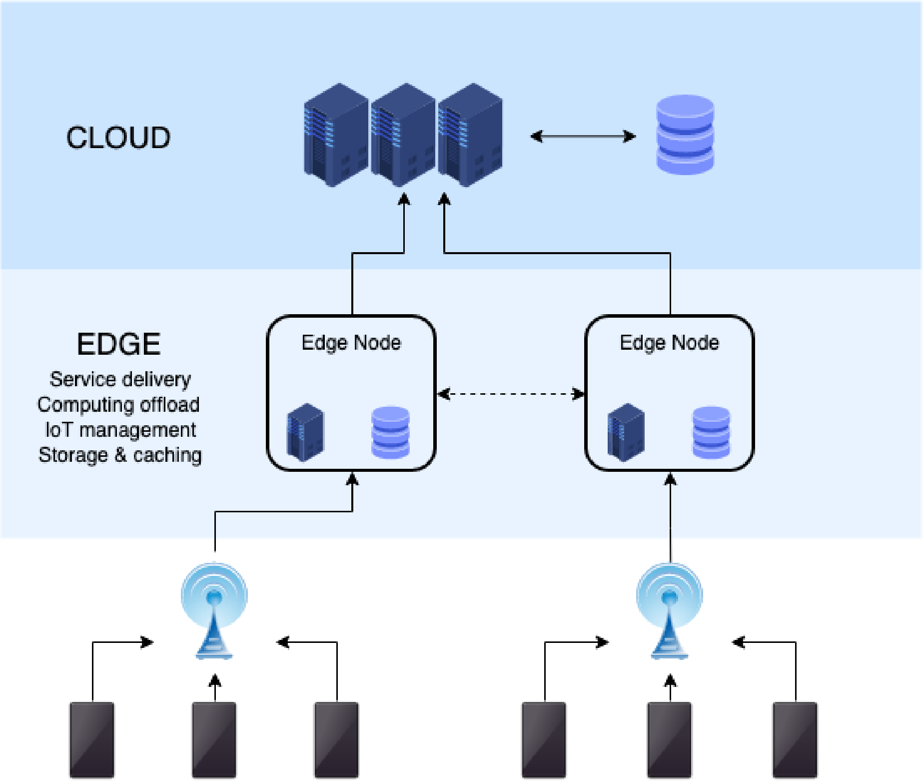
Neural Network Acceleration Study Season #2

Contents of presentation

- **Introduction and Background**
- **Key contribution: Edgent Framework**
  - DNN partitioning
  - DNN right-sizing
- **Performance Evaluation**
- **Conclusion and Discussion**

Introduction: Deep Neural Networks (1)

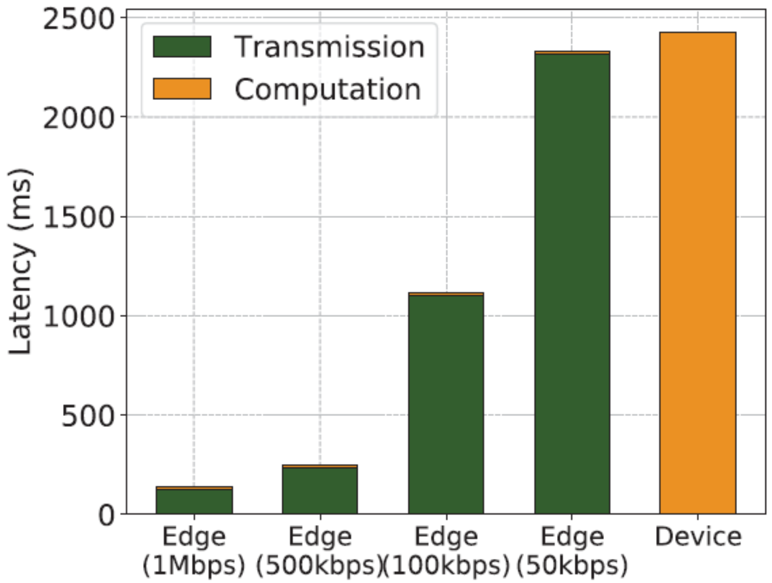
- **Mobile devices fail**
  - To the tremendous amount of computation required
- **Resort to cloud datacenter for intensive DNN computation**
  - Input data : generated mobile devices
  - Sent to remote cloud data center (computations)
  - Devices receive the execution results
- **Problems**
  - Intolerable latency
  - Extravagant energy



https://en.wikipedia.org/wiki/Edge_computing

Introduction: Deep Neural Networks (2)

- **Network bandwidth issue**
  - network bandwidth drop : from 1Mbps to 50kbps
  - Inference latency : from 0.123s to 2.317s
- **Mission-critical DNN-based applications**
  - Intelligent security
  - Industrial robotics
- **To solve this problems**
  - Propose *Edgent* framework
    1. DNN partitioning : some layers process edge, some layers process device
    2. DNN right-sizing : branchyNet framework (early-exit mechanism)



DNN Partitioning

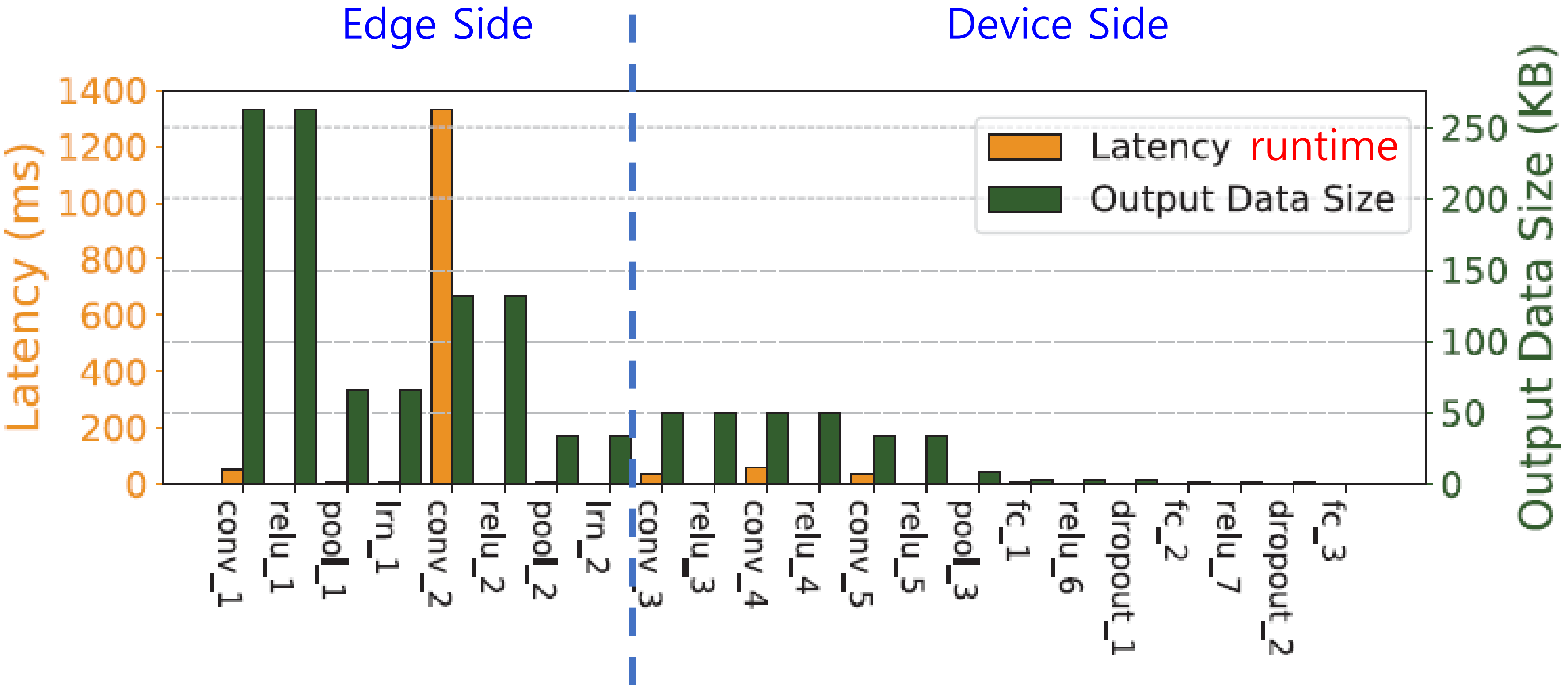


Fig. 3. AlexNet layer-wise runtime and output data size on Raspberry Pi.

DNN Right-Sizing

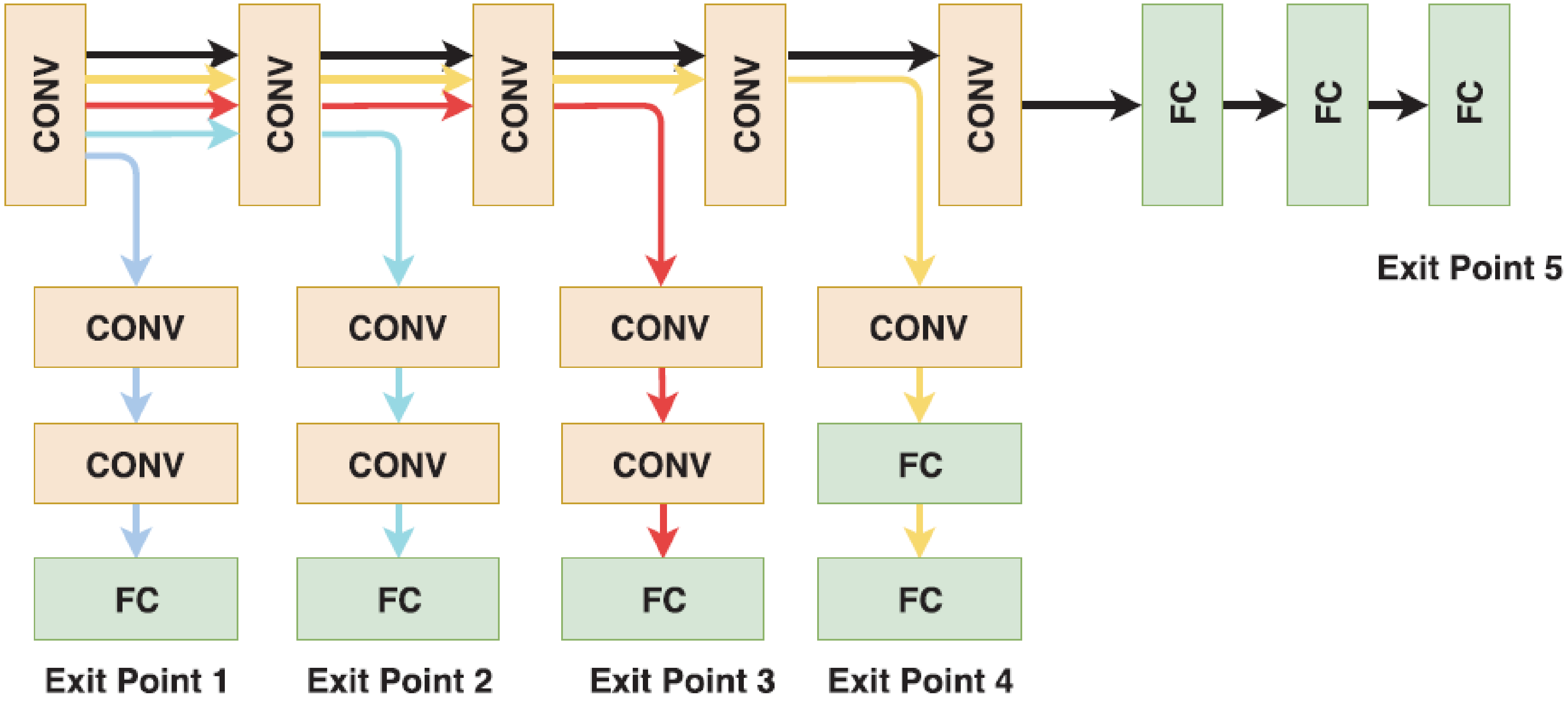
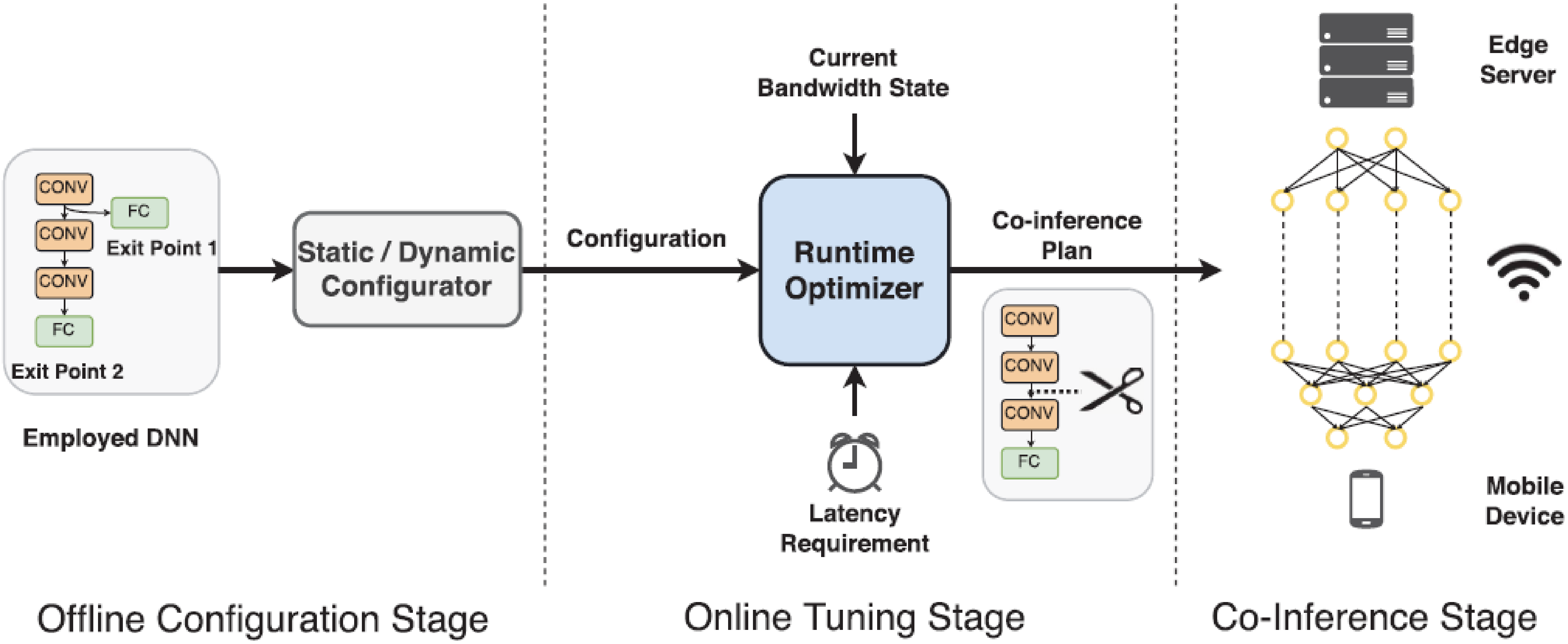


Fig. 4. A branchy AlexNet model with five exit points.

Edgent Framework



Edgent Framework

- **Offline configuration stage**
  - Pre-defined Static/Dynamic configurator
  - Composed of the trained branchy DNN and optimal selection for different bandwidth states
- **Online tuning stage**
  - Current bandwidth state, latency requirement
  - Runtime optimizer → Co-inference Plan
- **Co-inference stage**
  - Selected exit point and partition point
  - Execute server side and device side

Edgent for Static Environment

- **Offline configuration stage**
  - Profile layer-wise inference latency on the mobile device and edge server
  - Train the DNN model with multiple exit points via BranchyNet framework

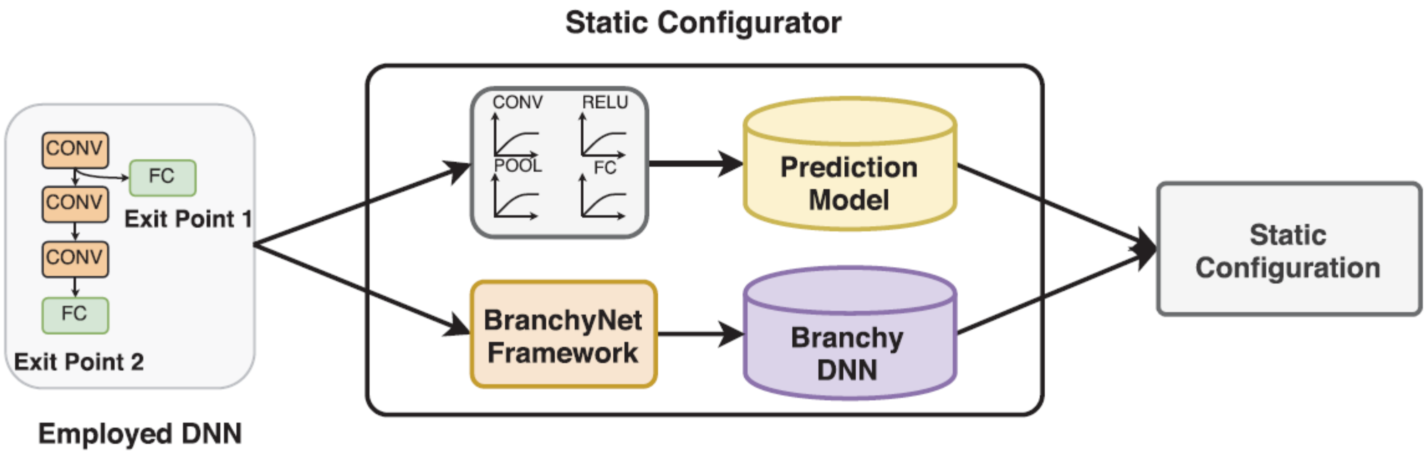
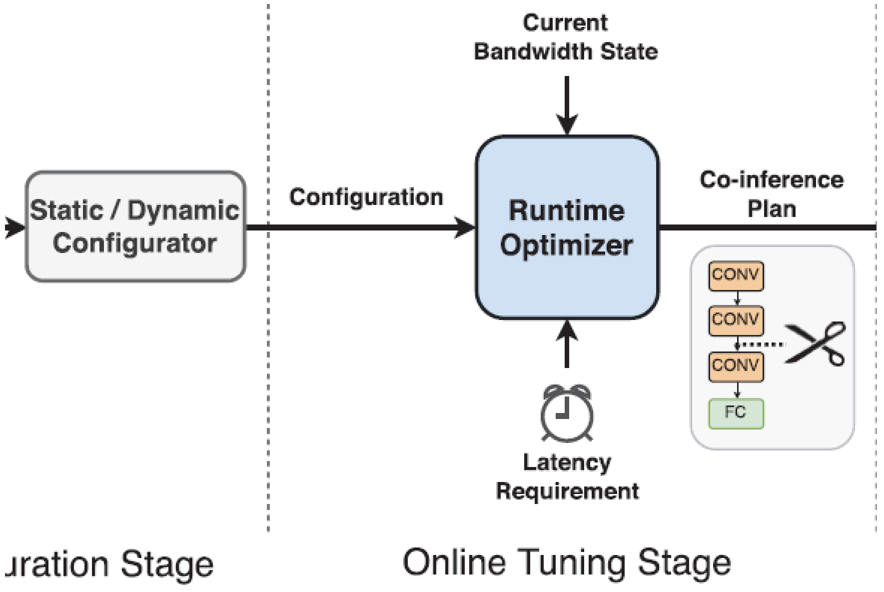


Fig. 6. The static configurator of Edgent.

Edgent for Static Environment

- **Online tuning stage**
  - Runtime optimizer component : search optimal exit point and partition point



Edgent for Static Environment

- **Online tuning stage**

Algorithm 1 Runtime Optimizer for Static Environment

Input:

M: the number of exit points in the DNN model

$\{N_i | i = 1, \cdots, M\}$: the number of layers in the branch of exit pint i

$\{L_j | j = 1, \cdots, N_i\}$: the layers in the branch of exit point i

$\{D_j | j = 1, \cdots, N_i\}$: layer-wise output data size in the branch of exit point i

$f(L_j)$: the prediction model that returns the j-th layer's latency

B: current available bandwidth

$Input$: input data size

$Latency$: latency requirement

Output:

Selection of exit point and partition point

Procedure

for $i = M, \cdots, 1$ **do**

  Select the branch of i-th exit point

  for $j = 1, \cdots, N_i$ **do**

    $ES_j \leftarrow f_{edge}(L_j)$

    $ED_j \leftarrow f_{device}(L_j)$

  end for

  $A_{i,p} = \underset{p=1,\cdots,N_i}{\operatorname{argmin}} \left(\sum_{j=1}^{p-1} ES_j + \sum_{k=p}^{N_i} ED_j + Input/B + D_{p-1}/B\right)$

  if $A_{i,p} \leq Latency$ **then**

    return Exit point i and partition point p

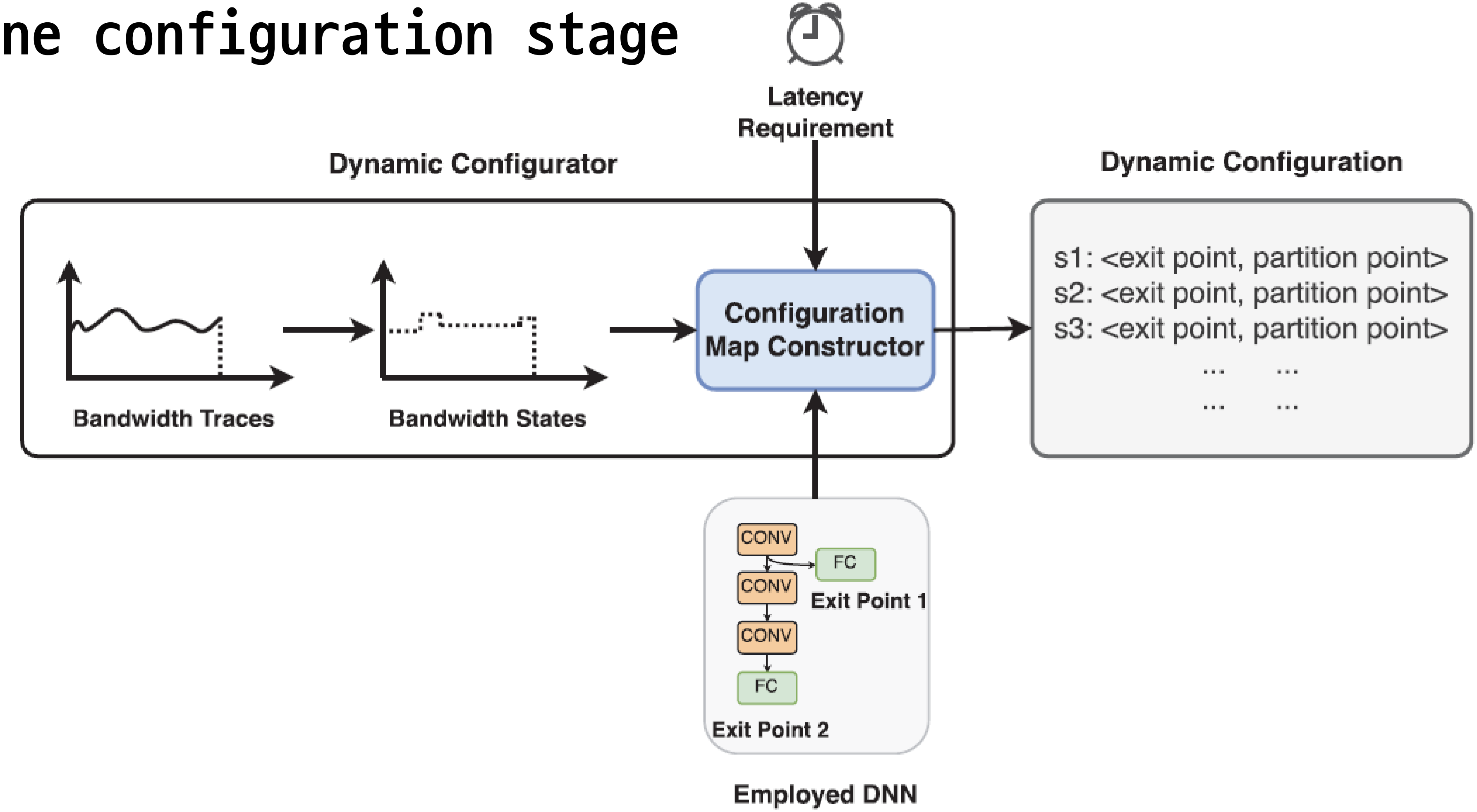
  end if

end for

return NULL ▷ can not meet the latency requirement

Edgent for Dynamic Environment

- Offline configuration stage



Edgent for Dynamic Environment

- **Offline configuration stage**

$$reward_{step} = \begin{cases} \exp(acc) + throughput, & t_{step} \le t_{req}, \\ 0, & else, \end{cases}$$

Algorithm 2 Configuration Map Construction

Input:
$\{s_i | i = 1, \cdots, N\}$: the bandwidth states
$\{C_j | j = 1, \cdots, M\}$: the co-inference strategy
$R(C_j)$: the reward of co-inference strategy C_j
Output:
Configuration Map

```
Procedure
for i = 1, ···, N do
    Select the bandwidth state s_i
    reward_max = 0, C_optimal = 0
    for j = 1, ···, M do
        reward_{c_j} ← R(C_j)
        if reward_max ≤ reward_{c_j} then
            reward_max = reward_{c_j}, C_optimal = C_j
        end if
    end for
    Get the corresponding exit point and partition point
        of C_optimal
    Add S_i :< exit point, partition point > to the Con-
        figuration Map
end for
return Configuration Map
```

Edgent for Dynamic Environment

- **Online tuning stage**

Algorithm 3 Runtime Optimizer for Dynamic Environment

Input:
$\{B_{1,\cdots,t}\}$: the accumulated bandwidth measurements until the current moment t
$\{C_j|j = 1, \cdots, t\}$: the co-inference strategy
$\{s_i|i = 1, \cdots, t\}$: the bandwidth states
$D(B_{1,\cdots,t})$: the bandwidth state detection function that returns the current bandwidth state
$find(s)$: find the co-inference strategy corresponds to the given state s

Output:
Co-inference strategy

Procedure
$C_t = C_{t-1}$
$s_t = D(B_{i,\cdots,t})$
if $s_t \neq s_{t-1}$ **then**
    $C_t \leftarrow find(s_t)$
end if
$s_{t-1} = s_t$
$C_{t-1} = C_t$
return C_t

Experimental Setup

- **Raspberry Pi and desktop PC**
  - Static bandwidth env. : WonderShaper tool [45]
  - Dynamic bandwidth env. : Belgium 4G/LTE bandwidth logs [46] emulation
  - Use BraychNet framework : AlexNet as the toy model
  - AlexNet model : 5 exit point, cifar-10 dataset

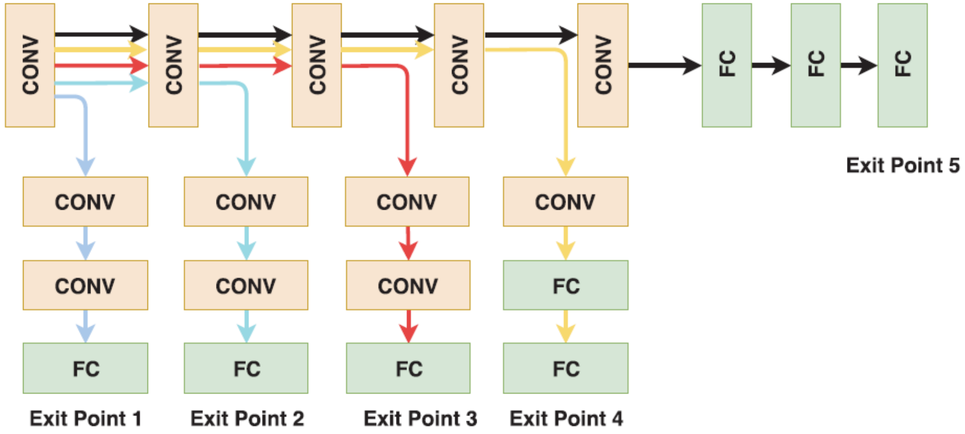
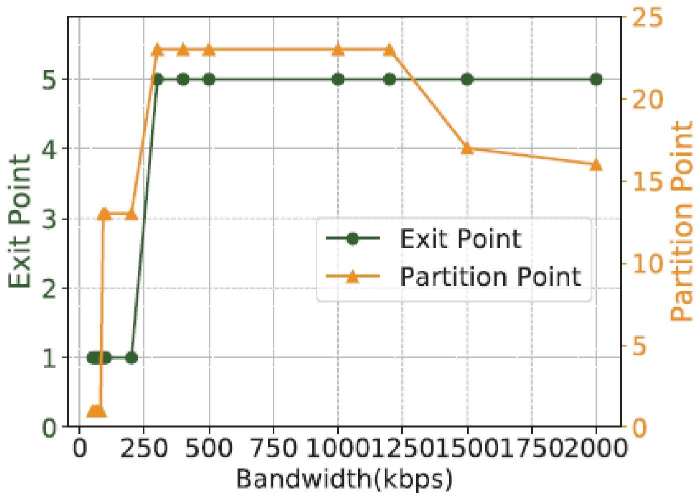


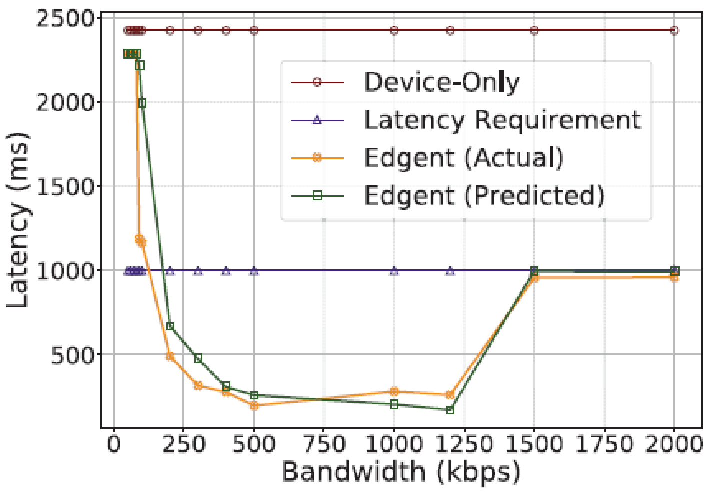
Fig. 4. A branchy AlexNet model with five exit points.

Experimental Results

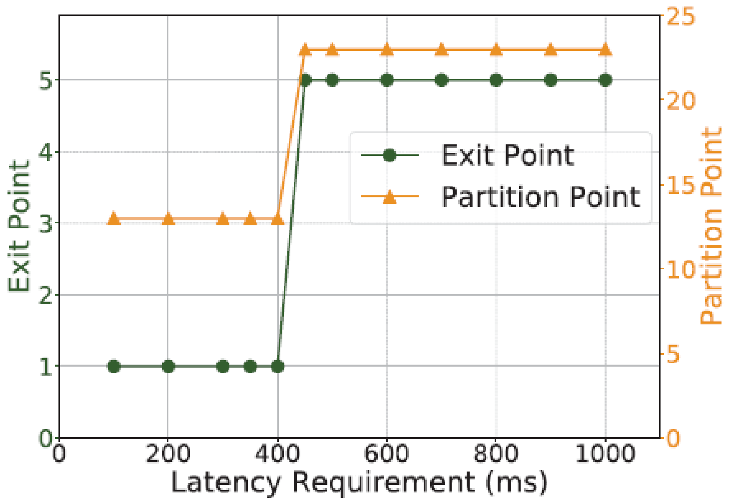
- **Static Bandwidth Environment**



(a) Selection under different bandwidths



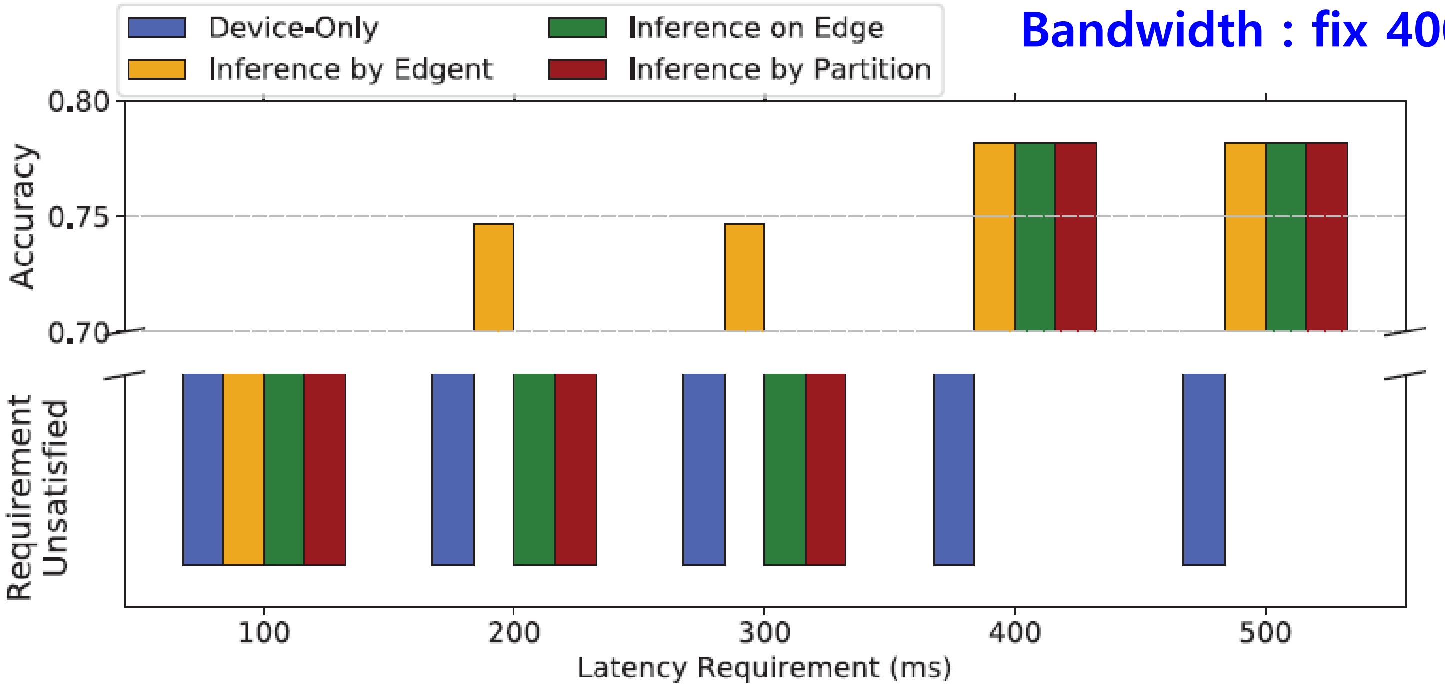
(b) Model runtime under different bandwidths



(c) Selection under different latency requirements

Experimental Results

- **Static Bandwidth Environment**

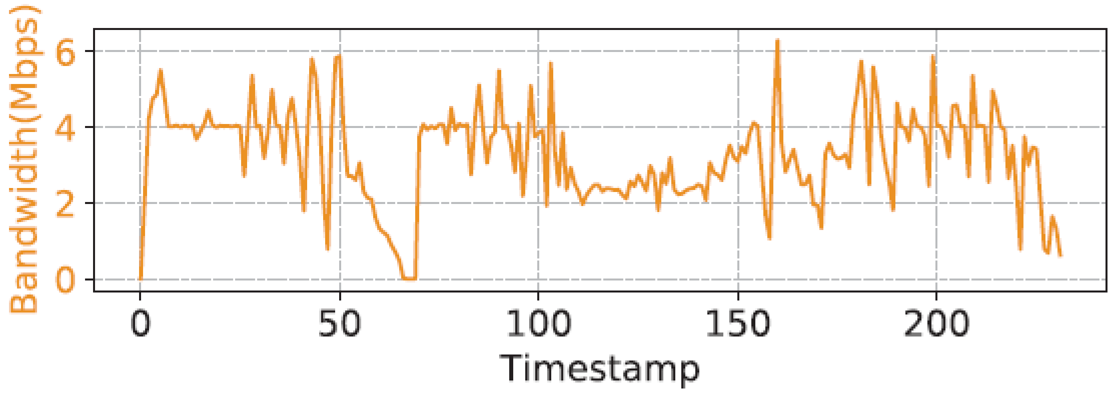


Bandwidth : fix 400kbps

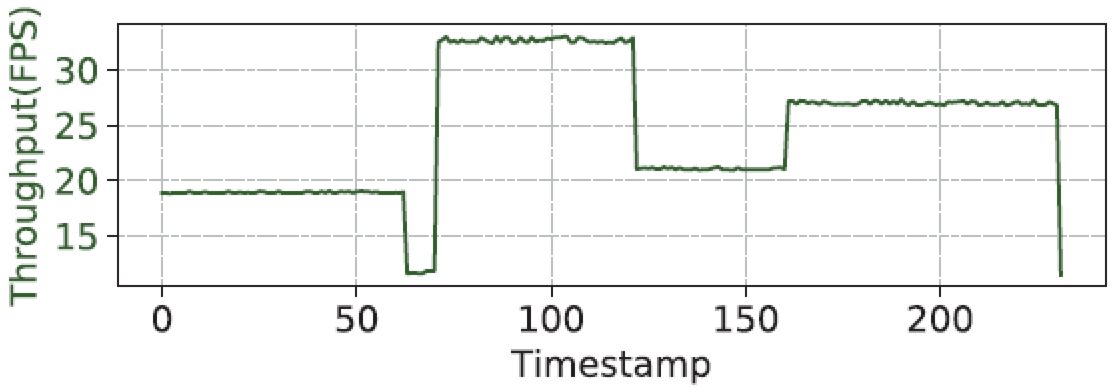
Fig. 9. The accuracy comparison under various latency requirement.

Experimental Results

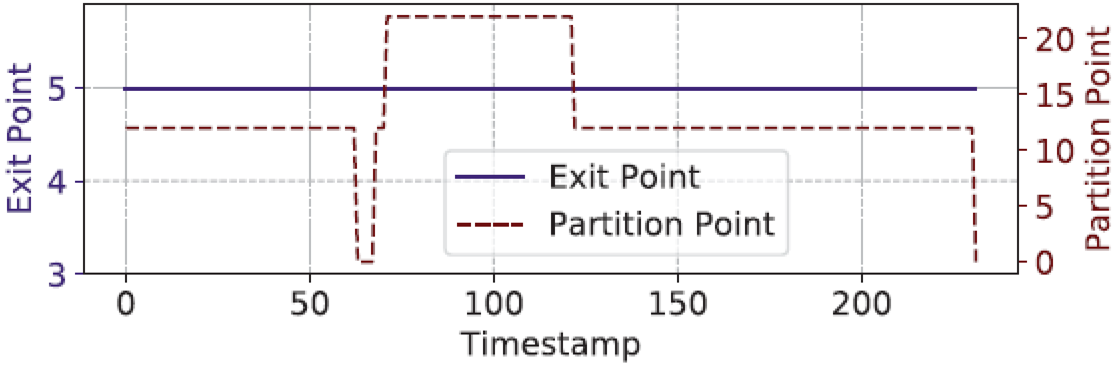
- **Dynamic Bandwidth Environment**



(a) A example bandwidth trace on Belgium 4G/LTE dataset [46]



(b) The throughput of DNN model inference



Conclusion

- **Propose Edgent Framework**

  - Two design knobs to optimize DNN inference latency

  1. DNN partitioning : profiling layer-wise runtime
  2. DNN right-sizing : branchy network → early-exit mechanism
  3. Optimizing → low-layency edge intelligence

- **Proposed prototype implementation : Raspberry Pi & PC**

  ➢ **show feasibility and effectiveness**

Thank you