CRIME: A Collaborative Edge/Cloud Inference Framework for Recurrent Neural Networks

Roberta Chiaro, Chen Xie, **Daniele Jahier Pagliari**, Yukai Chen,
Enrico Macii and Massimo Poncino

Politecnico di Torino, Turin, Italy

Contact: daniele.jahier@polito.it

POLITECNICO
DI TORINO

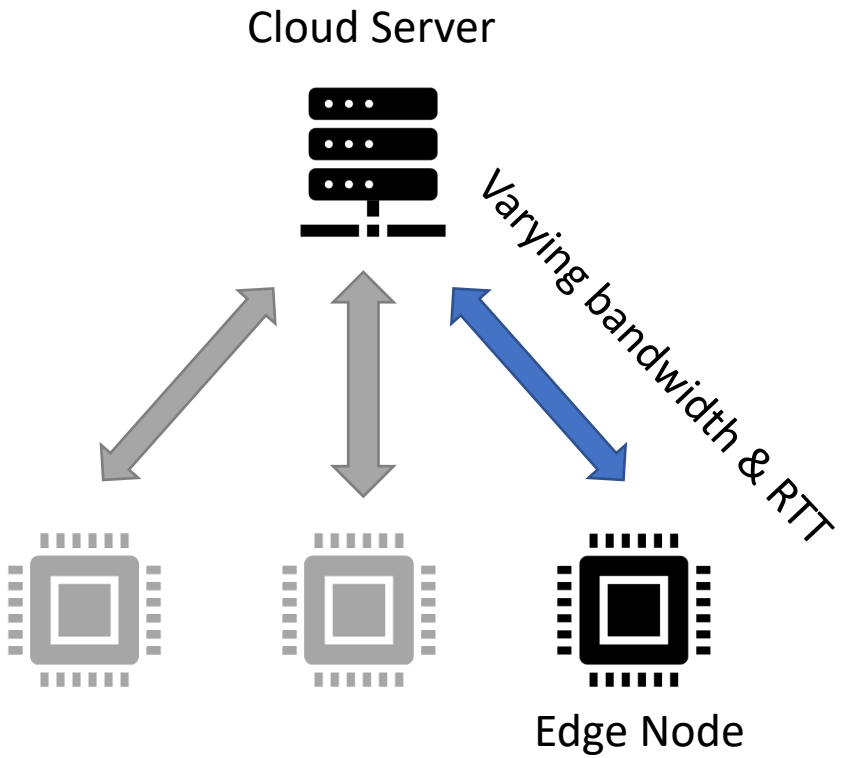
Motivation: ML/DL inference at the edge

- Potential benefits:

1. Lower and more predictable response latency
2. Avoid energy-hungry wireless transfers
3. Improve data privacy

Motivation: ML/DL inference at the edge

- For energy and latency, edge inference is not always optimal!
  - In particular on GP hardware (MPUs, MCUs)
- Edge vs cloud: time-varying trade-off
  - Speed/energy of edge and cloud compute
  - Speed/energy of tx/rx



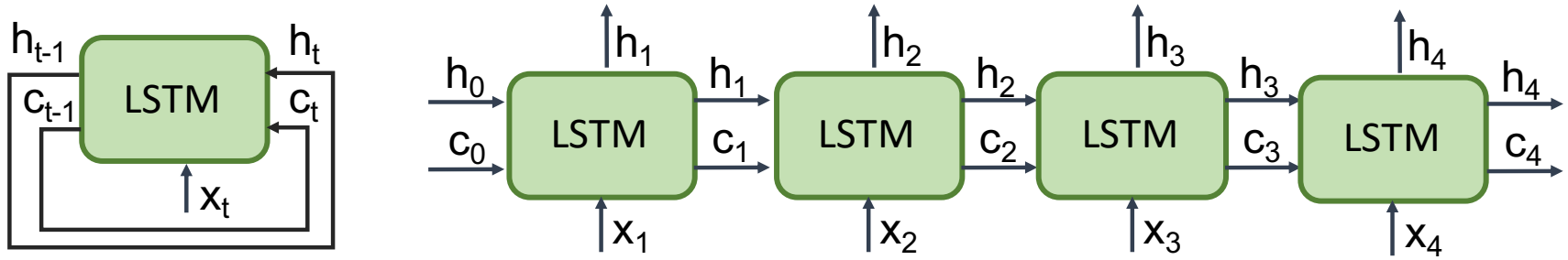
Collaborative Inference: dynamically map inference tasks on a network of collaborating devices

Collaborative Inference for RNNs

- Widely studied for on feed-forward NNs (MLP, CNNs, etc.)
- RNNs introduce new issues:
  - Temporal dimension to process sequences (text, speech, time-series, etc.)
- **Our work: first collaborative inference framework for RNNs!**
  - *D. Jahier Pagliari et al, Input-dependent edge-cloud mapping of recurrent neural networks inference, DAC 2020*
  - *D. Jahier Pagliari et al, CRIME: Input-Dependent Collaborative Inference for Recurrent Neural Networks, IEEE Transactions on Computers (in press)*

Background on RNNs

- RNNs = NNs with feedback
- Example: Long-Short Term Memory (LSTM) RNNs
  - Inputs at each step *t*:
    - New datum (x_t)
    - Prev. output (h_t = hidden state, c_t = cell state)
  - At inference time, the NN is actually unrolled n times (n = input length)

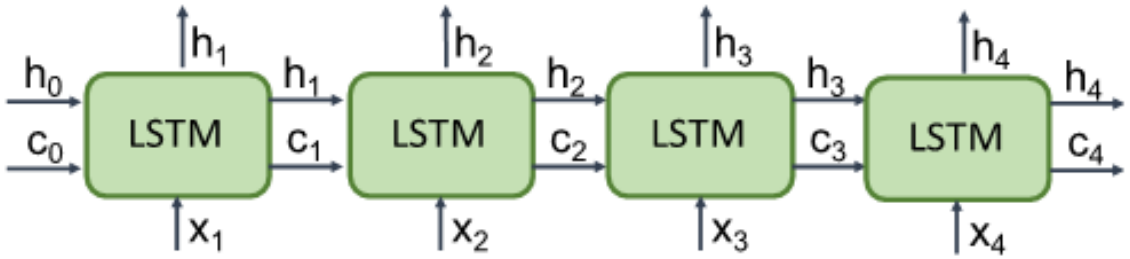


a) LSTM feedback loop

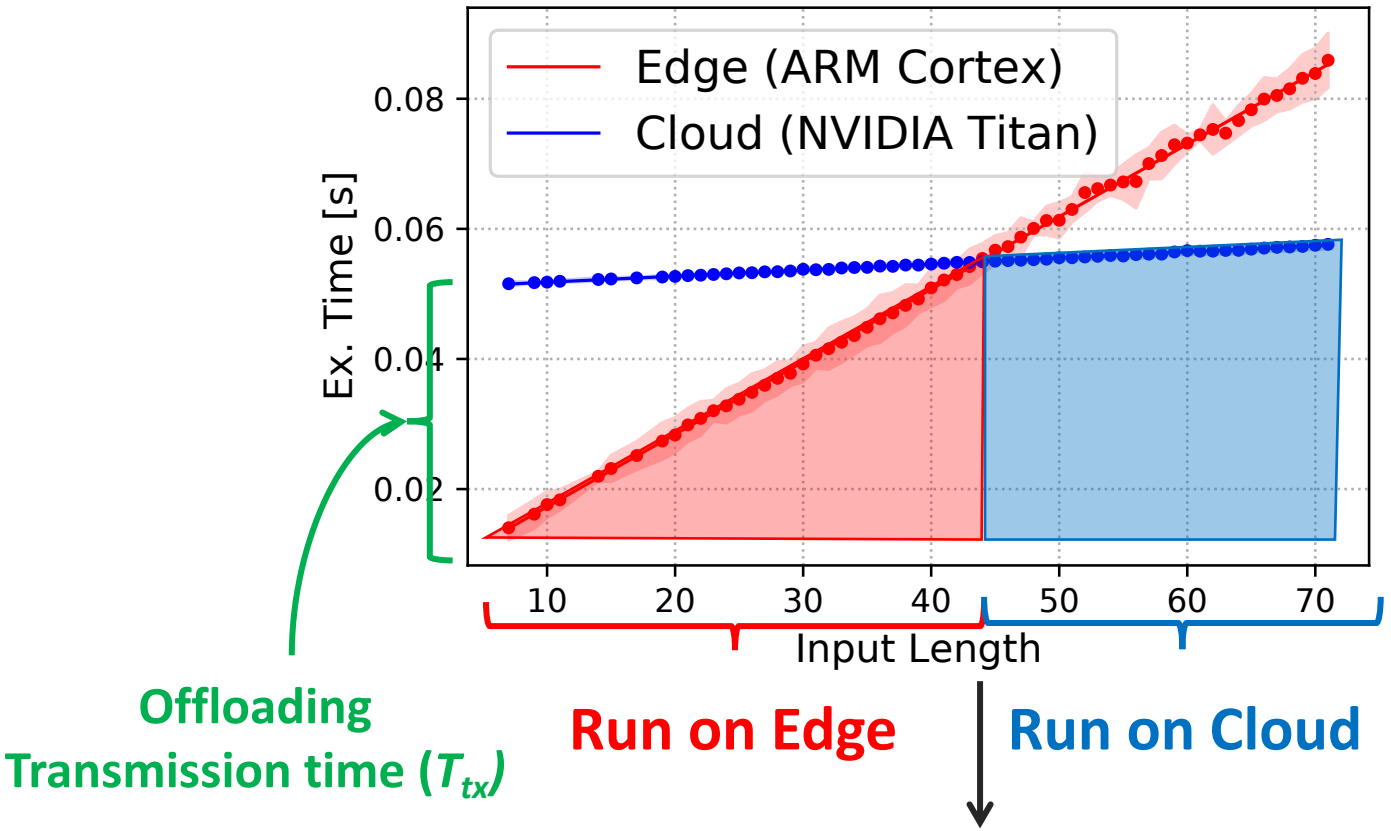
b) Unrolled LSTM for n=4

RNNs: Energy/latency characterization

- Each step involves the same operations
  - large MxVs + activation functions
  - Similar power and ex. time
- No inter-step parallelism
  - Each step requires the previous outputs
- Compute time and energy grow linearly with n!

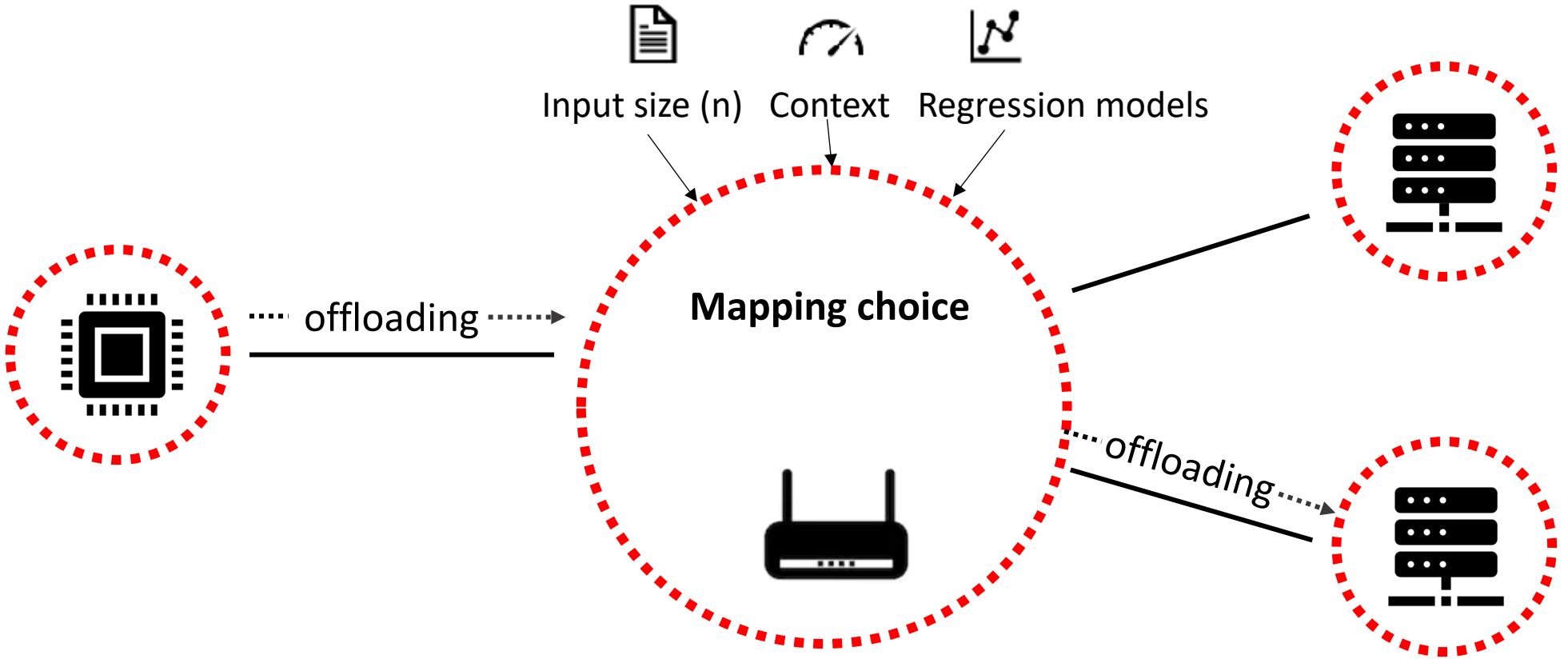


Collaborative RNN Inference: offloading choice



CRIME: fully distributed mapping engine

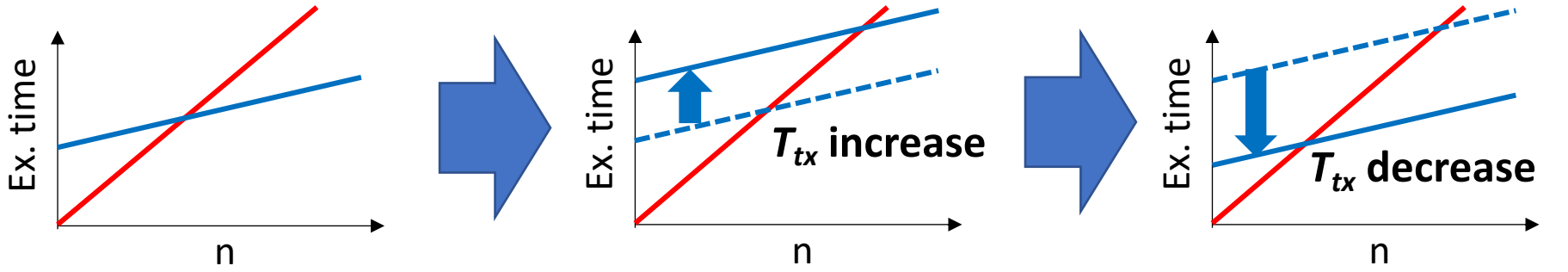
- Lightweight *mapping engine*: selects local processing or inference offloading for a given input



CRIME: Dynamic Adaptation

- Network conditions (*k*) and devices load change over time:
  - Context information and regression models need to be updated

- Two updating methods:
  1. Leverage offloading events: attach time-stamps to inputs/outputs
  2. If a device is not used for some time, use a special “ping” packet



Detected by time-stamps

Detected by “ping”

Results: Setup

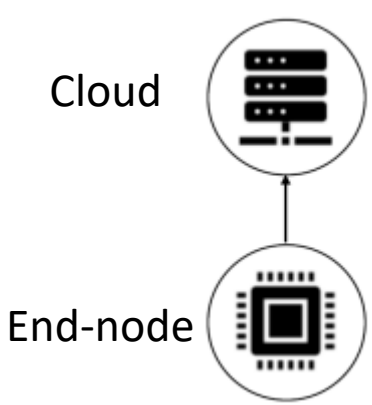
- **End node:** ARM Cortex A-53 + TensorFlow
- **Gateway**: NVIDIA Jetson TX2 + Tensorflow
- **Cloud Server:** NVIDIA Titan XP GPU + TensorFlow

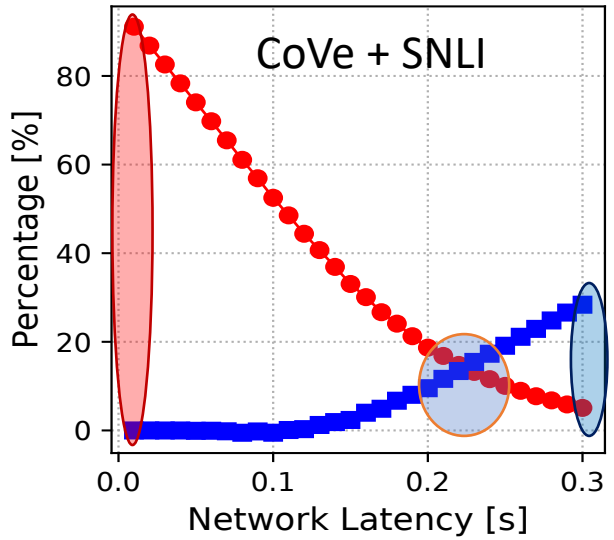
- CoVe (B. McCann et al, *"Learned in translation: Contextualized word vectors"*)
  - 2-layer LSTM
  - SNLI and SQuAD datasets

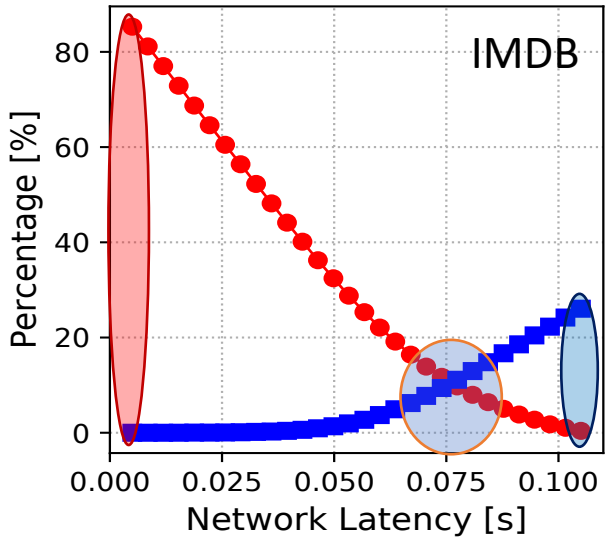
- IMDB (from Keras' official github repo)
  - 1-layer LSTM

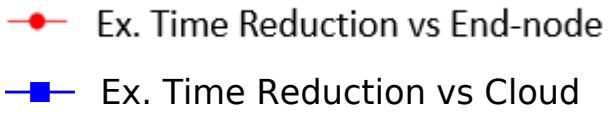
Results

- Ex. time reduction vs “edge only” and “cloud only” for a given T_{rtt}



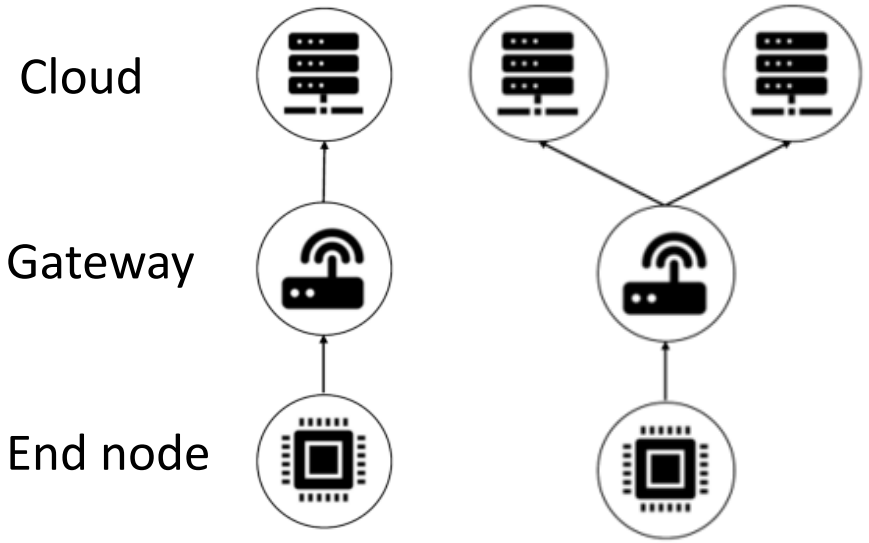






- Low-latency → Always offload → 80% saving vs “edge only”
- High-latency → Never offload → 30% saving vs “cloud only”
- Intermediate → 20% simultaneous saving vs both solutions

Results: multiple offloading levels

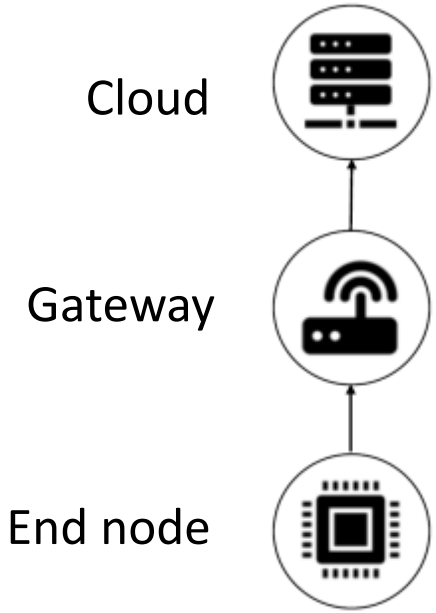


T_{tx} profile from RIPE Atlas database

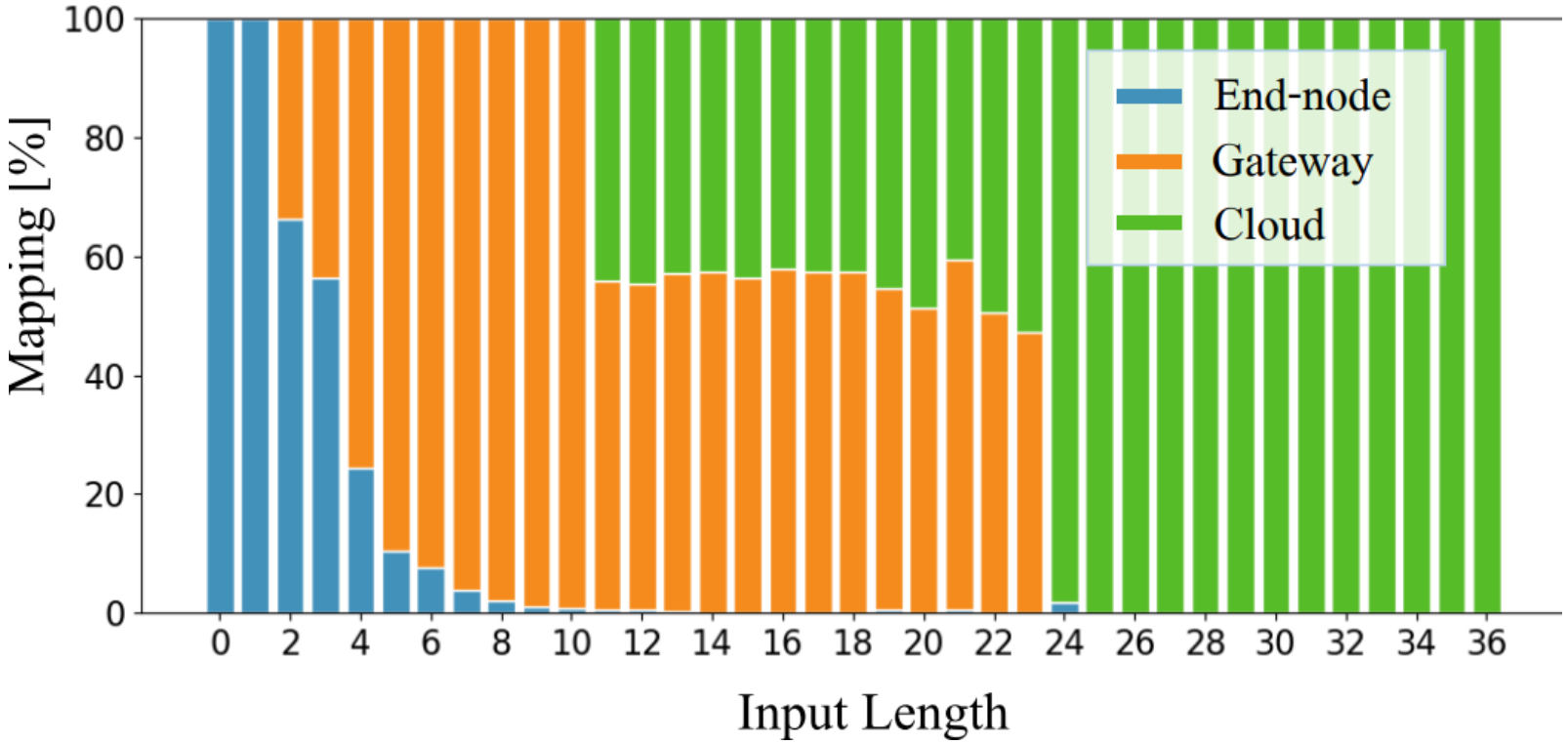
T_{tx} profile: real Bluetooth Low Energy (BLE) connection

Test	Three-Levels				Three-Levels and Two-Servers				
	Ex. time reduction [%]			Ex. time incr. [%]	Ex. time reduction [%]				Ex. time incr. [%]
	vs end-node	vs gateway	vs cloud	vs oracle	vs end-node	vs gateway	vs cloud1	vs cloud2	vs oracle
SNLI	35.57	5.93	25.33	0.32	35.56	45.38	26.11	25.13	0.72
SQuAD	26.40	1.49	31.92	0.99	23.22	35.12	32.98	29.51	1.18
$SNLI_{200}$	4.52	20.22	37.48	0.85	4.75	50.87	44.73	37.69	1.15
IMDB	-0.46	68.49	59.22	0.46	-0.70	80.45	64.35	59.12	0.71

Results:

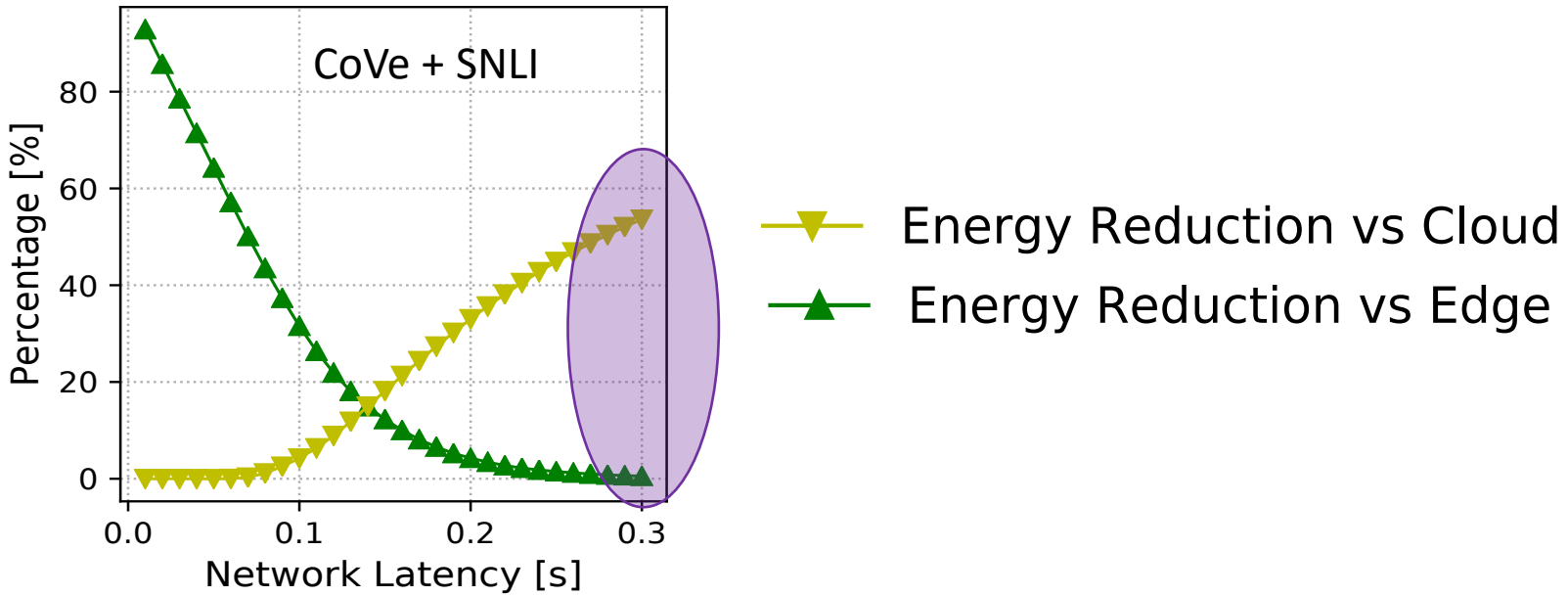


Mapping visualization for SQUAD dataset



Results

- CRIME targeting energy minimization
  - an adaptation of the cost function evaluated for mapping choices



- Local processing is more convenient for energy

Conclusions

- Collaborative inference can improve execution time and energy of RNNs
- CRIME can determine the optimal inference device dynamically, adapting to variations in network status or in the devices loads
- CRIME adapts to every network topology
- CRIME can be extended to other NN architectures

Thank You

Questions?