Time-efficient Offloading for Machine Learning Tasks between Embedded Systems and Fog Nodes

Darren Saguil darren.saguil@uoit.net
Akramul Azim akramul.azim@uoit.ca

OntarioTech UNIVERSITY

Introduction

Machine Learning allows you to **gain insights and analysis** from seemingly unrelated features

ML Model

Embedded Systems can leverage this to provide novel functions, such as automation, navigation, and classification

Motivation

- Machine learning is **computationally expensive** and embedded systems are **resource-constrained**, so it is the status quo to **perform all Machine Learning applications on external devices**. This may lead to some problems, such as:
  - The runtime of the ML models are bottlenecked by the transmission time
  - Losing connection can impact the device's functionality[1]

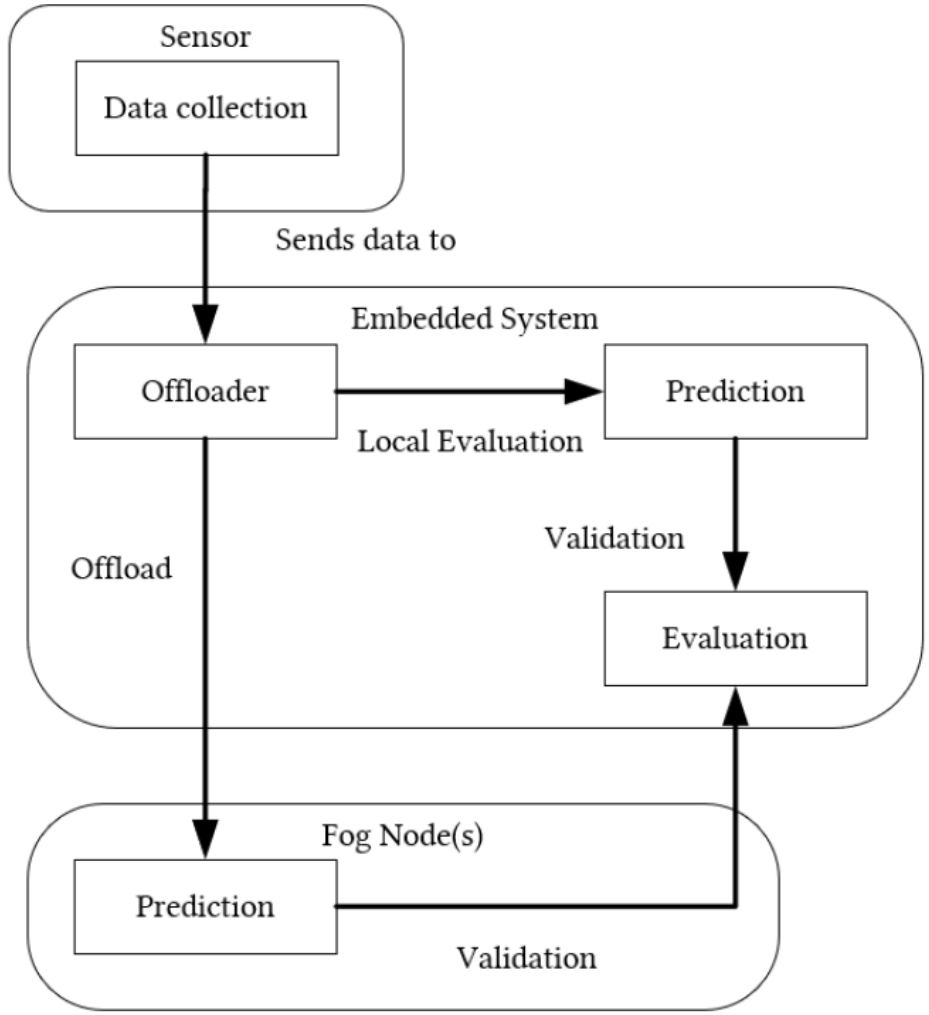
- ML algorithms come in many different complexities, and embedded systems may be able to run some, but not all of them

- Our contribution was to find a **time-efficient distribution threshold** to lessen the reliance of embedded systems on fog servers by running some inputs locally, and offloading when needed

Methodology

- A simulated sensor sent machine learning inputs to a component on the embedded system called the “Offloader”. This component determined:
  - which model the input is being sent to
  - whether to send the input to local or remote model
- Pre-Runtime, the WCET of each model is measured using a validation set. The offloader compared this WCET to a threshold which was measured during runtime. It consisted of:
  - T_L, the time wirelessly transfer the data
  - T_F, the time execute the model on the external device

$$WCET_T = 2 * T_L + T_F$$

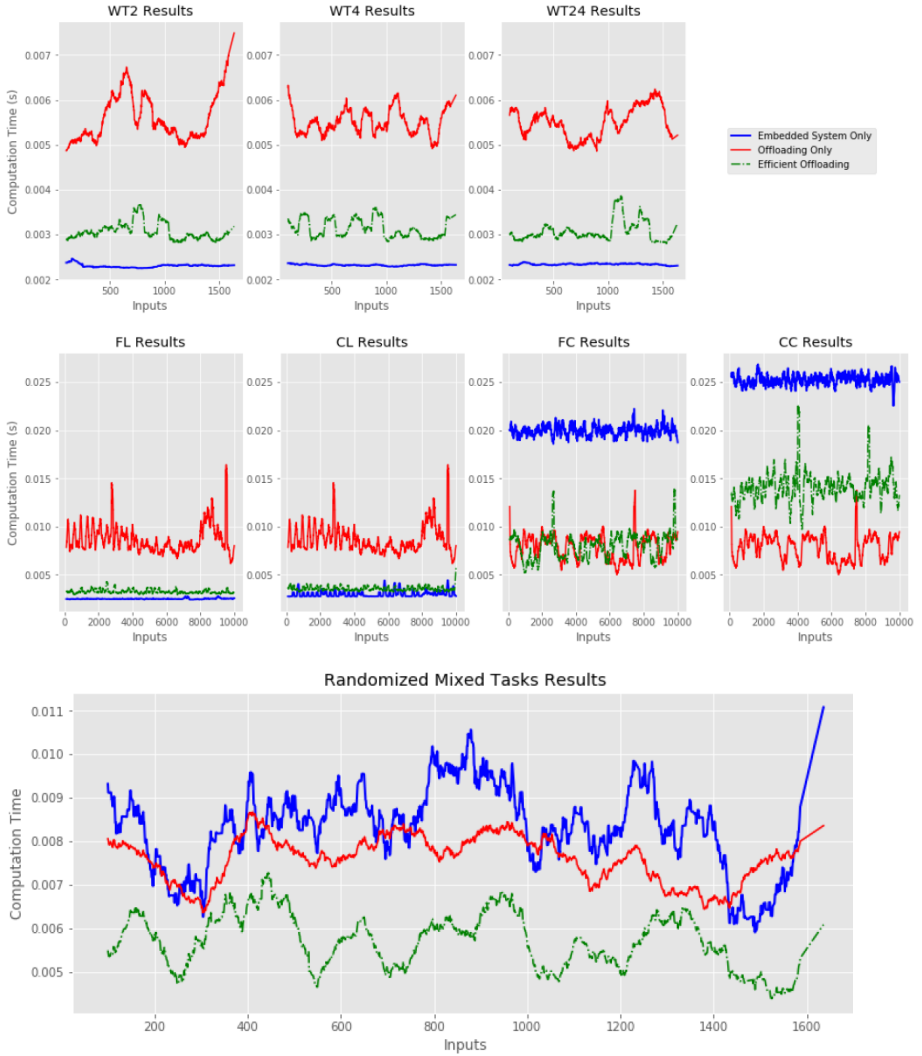


Results and Analysis

- The system model was ran using both linear (1D) and image datasets on a **Multilayer Perceptron (MLP)** and a **Convolutional Neural Network (CNN)**

- The top graphs show the time taken to run every model locally and the time taken to run the model externally. It shows that the models using the MLP can be run locally

- The bottom graph shows the results of a theoretical device running multiple models of varying complexities. When using the proposed offloader, it shows that only offloading inputs that bypass the threshold can reduce the total runtime



Conclusion

- The status quo of only performing every embedded system's Machine Learning application on external devices can be improved. The **transmission time is a severe bottleneck**, and simple Machine Learning applications can bypass it by running locally
- One of the main factors which determines if a Model's input should be offloaded is the model's complexity. For example, the CNN used in this experiment had a large Dense Layer with 128 output nodes. This made the runtime much longer, as opposed to the MLP's dense layer of only 32 output nodes.
- Runtime is not the only aspect to observe when running Machine Learning applications. Energy consumption and temperature should also be looked at.
- The Machine Learning models themselves could also be partitioned, instead of offloading the functionality of the entire models[2]

References

1. Sam Leroux, Steven Bohez, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. The cascading neural network: building the internet of smart things. *Knowledge and Information Systems*, 52:791–814, 2017.
2. Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS ’17, pages 615–629, New York, NY, USA, 2017. ACM.