To be, or not to be

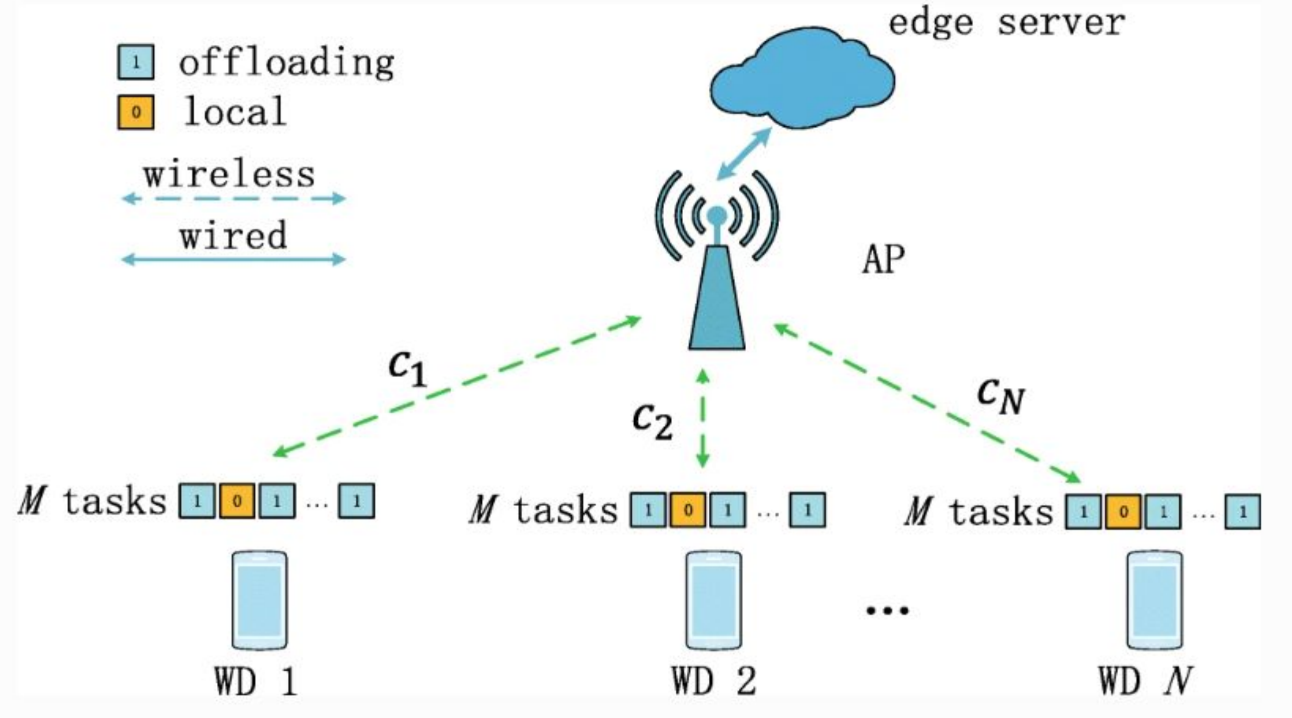
Edge Computing Version

Abhishek Verma (av2783)
Ishita Jaisia (ij2056)
Utkarsh Kumar (uk2012)

Introduction

- In edge computing, computing is done at the location or close to the data source, minimizing the need for data to be processed in the remote data center.
- Wireless communication technology has enabled transmission of computation tasks from wireless devices to nearby access points.
- Deploying computation servers at these access points (rather than remote data centers) bridges the gap between edge devices and edge server.
- Applications include online gaming, AR/VR, smart utility grid analysis, safety monitoring of oil rigs, streaming video optimization, drone-enabled crop management, etc.

Problem Statement



Previous Work

- This offloading problem is NP-hard because of exponential time complexity.
- Different low-complexity algorithms solve the binary computation offloading problem.
- One such algorithm based on game theory requires multiple communications per task between the edge server and the device.
- However, almost no algorithm overcomes the trade-off between time delay and optimal solution.

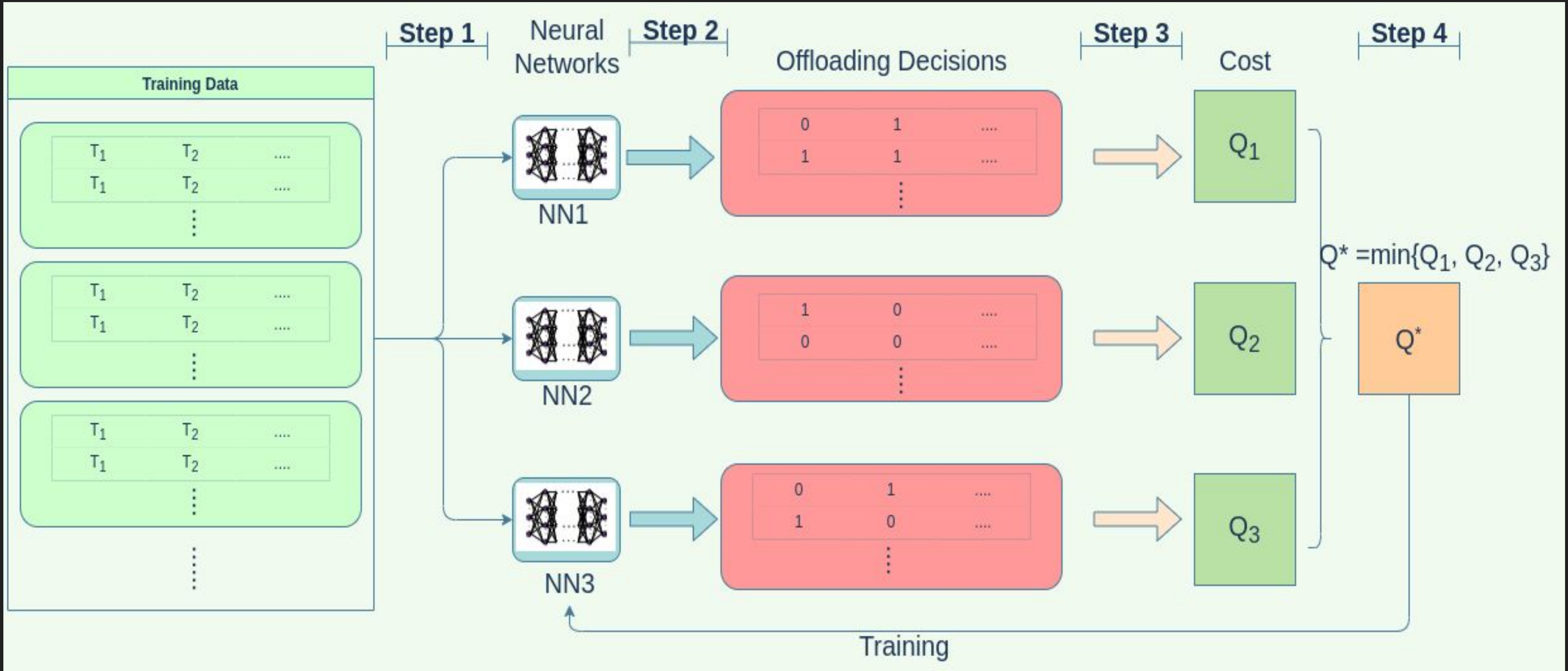
Our Contribution

- Design (and implementation) of a deep learning algorithm: given an input task and its characteristics, outputs a binary offloading decision.
- Compare the performance of this algorithm with baselines including the exponential algorithm.
- Profile the training time and inference time on bare-metal machines, VMs and Kubernetes cluster.

System Model

Cost	Computing on edge server	Computing on local device
Energy (electricity) cost	energy to upload task + energy to perform computation on edge server	energy to perform computation on local device
Time delay	time to upload task + edge server computation time	local device computation time

Algorithm


Step 1
Neural Networks
Step 2
Offloading Decisions
Step 3
Cost
Step 4
Training Data
T1 T2 ....
T1 T2 ....
NN1
NN2
NN3
0 1 ....
1 1 ....
1 0 ....
0 0 ....
0 1 ....
1 0 ....
Q1
Q2
Q3
Q* =min{Q1, Q2, Q3}
Q*
Training

Experimental Assumptions

Number of Wireless device	3
Number of tasks per device	3
Local computation time of mobile devices	4.75×10^{-7}s/bit
Processing energy consumption	3.25×10^{-7}J/bit
Uplink bandwidth limit	150 Mbps
Weight of the energy consumption at edge server (α)	1.5×10^{-7} J/bit
Weight of time delay (β)	1 J/bit

Experimental Setup

- We compare the performance of the algorithm with baseline models on the NYU HPC cluster.
- Profile the training time and inference time on bare-metal machines, VMs and Kubernetes cluster.

NYU HPC
Model Name: Intel(R) Xeon(R) Platinum 8268 CPU
CPU Frequency: 2.90 GHz
Number of CPUs: 4
Threads Per Core: 1

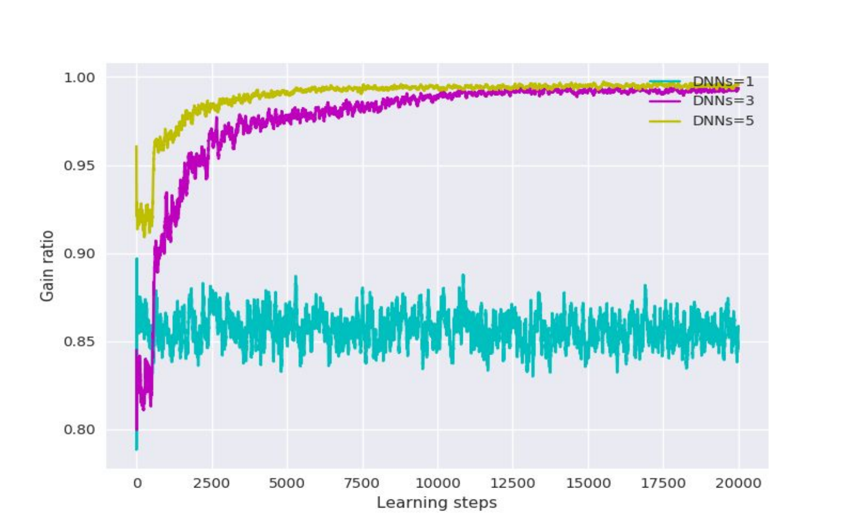
IBM VM
Model Name: Intel Xeon Processor (Cascadelake)
CPU Frequency: 2.40 GHz
Number of CPUs: 4
Threads Per Core: 2

Local
Model Name: 11th Gen Intel(R) Core(TM) i5-1135G7
CPU Frequency: 2.40 GHz
Number of CPUs: 8
Threads Per Core: 2

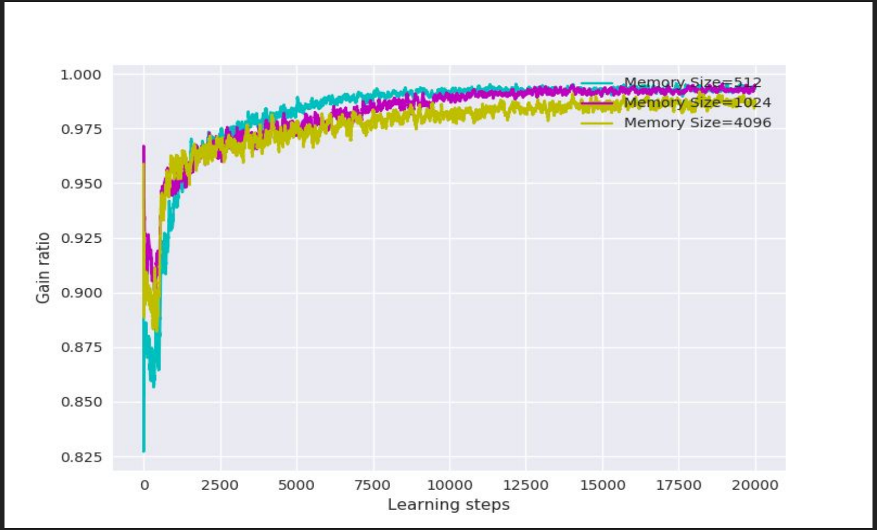
Kubernetes
Model Name: Intel Xeon Processor (Cascadelake)
CPU Frequency: 2.40 GHz
Number of CPUs: 4
Threads Per Core: 2

Algorithm Performance Evaluation

Algorithm performance for different number of DNNs

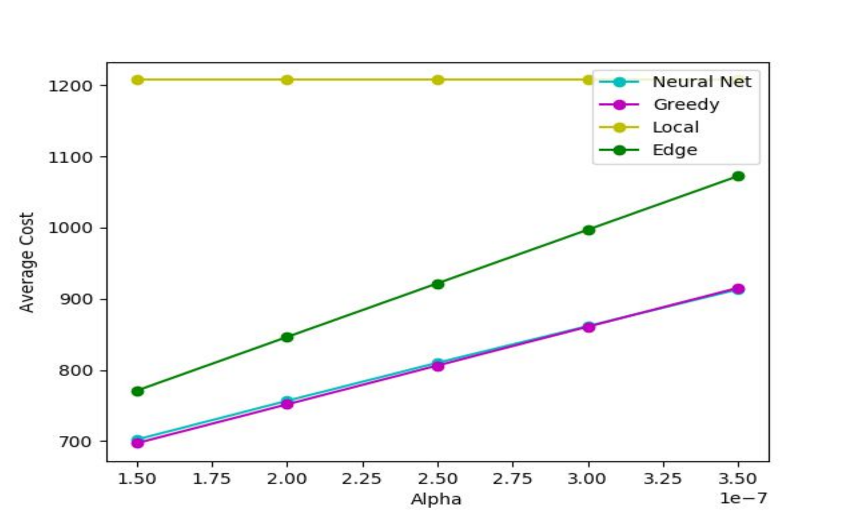


Algorithm performance with different memory size

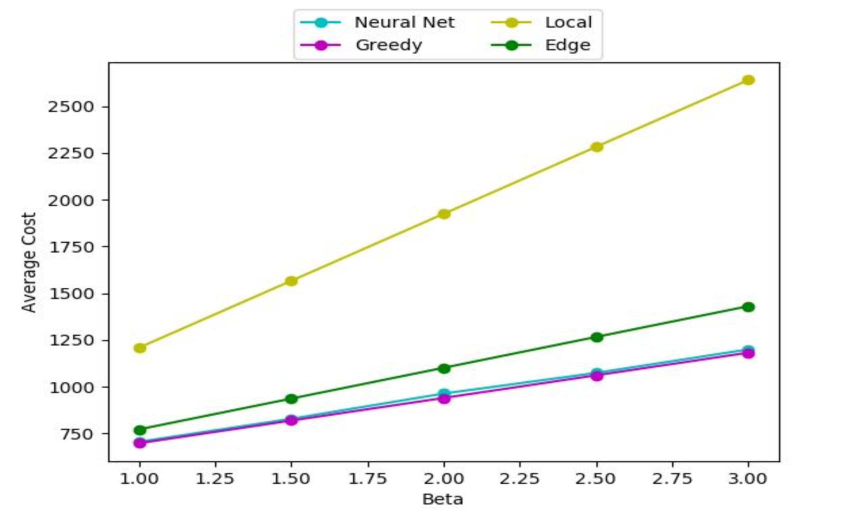


Algorithm Performance Evaluation

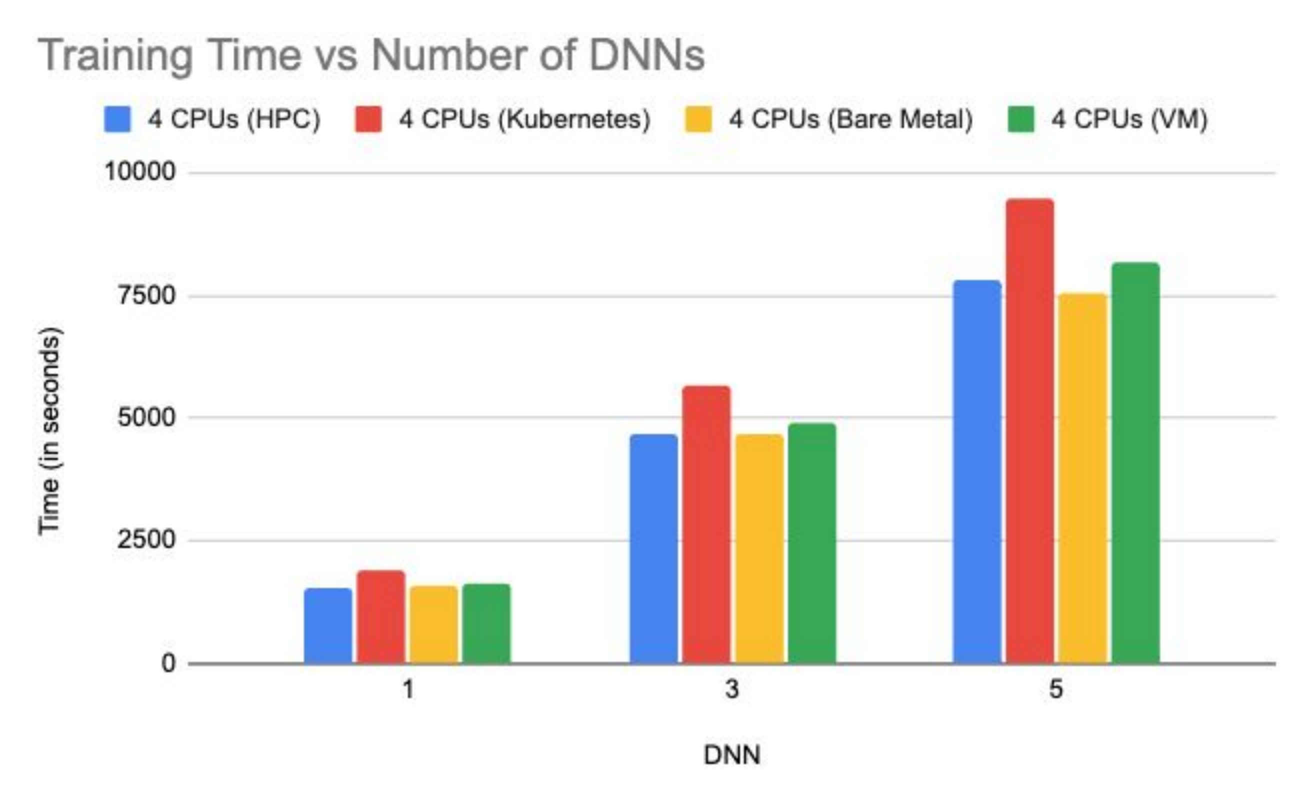
Average Energy Cost vs Joules/bit required to transfer data from edge device to server



Average Energy Cost vs Joules/bit required for computation

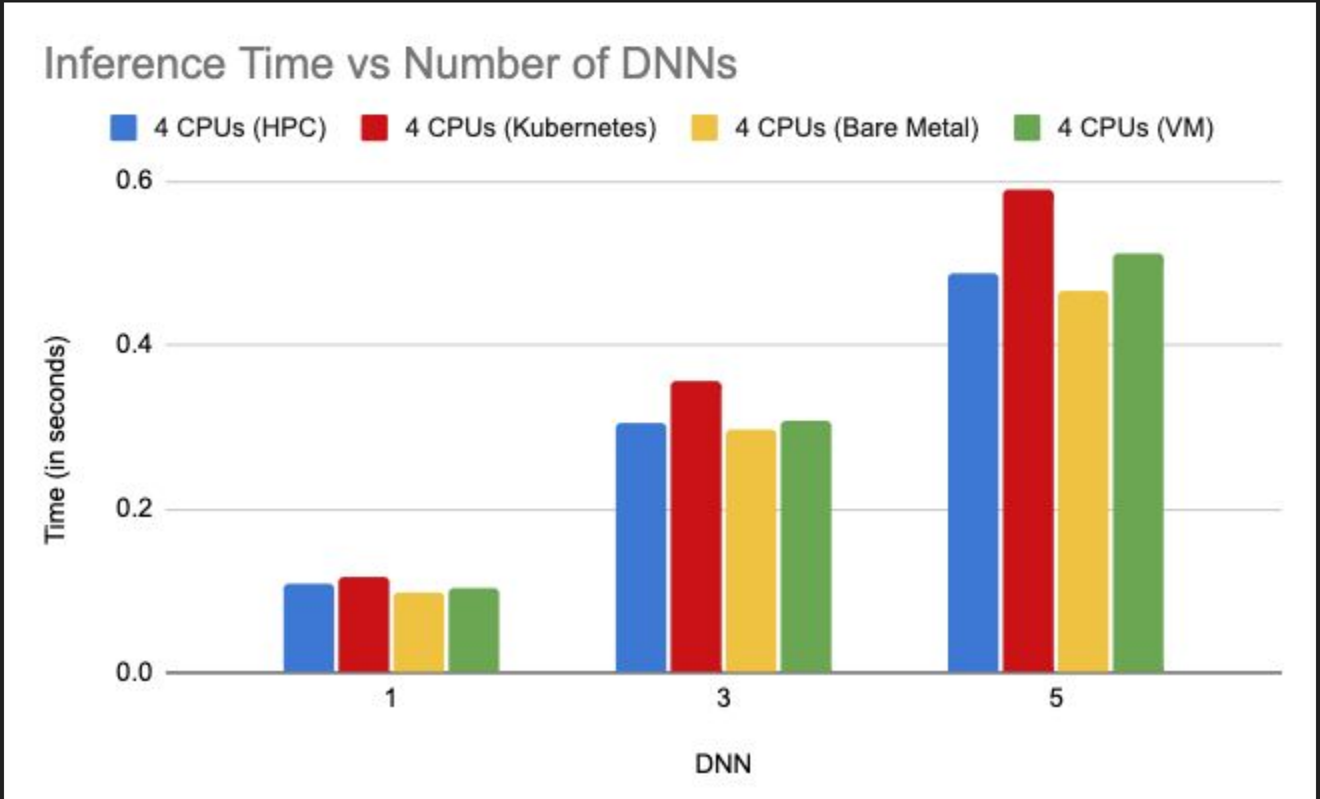


Cross-Platform Training Time



Cross-Platform Inference Time

[15 edge devices, 3 tasks per device]



Conclusion and Future Work

1. The proposed algorithm can generate near-optimal offloading decisions in less than one second.
2. Scalability with respect to number of DNNs can be increased by training and inference in parallel.
3. Profiling the inference algorithm on edge devices with minimal computation power will lead to more insightful results.
4. Running the algorithm on edge simulation frameworks will validate the real-life usefulness of the algorithm.

Thoughts

1. Apart from the resource usage, profiling an algorithm on its assumptions is always useful and can lead to interesting insights.
2. Kubeflow is cool! You don’t need to write yaml(s) for every single job. You can just run jupyter notebooks on the Kubeflow K8 cluster.
3. Installing utilities and libraries on vanilla VM is an arduous task.
4. It was a challenge to come up with a cost function for the DNNs as it involved researching about various costs associated with edge computing.