DeepEdge: A Network Edge for Deep Learning Workloads

Rahul Anand Sharma
Carnegie Mellon University

Ranveer Chandra
Microsoft Research

Abstract

Several applications require extremely low latency. This has led to a design in which computation is being performed closer to the user at the network Edge. A high compute workload that has recently gained significant interest is running Deep Neural Networks (DNNs) on streaming data. However, running all the DNNs at the Edge might get too computationally intensive for a server with limited GPU and CPU resources. In this paper, we propose a new Edge architecture, called DeepEdge, for handling Deep Learning workloads on a network edge. DeepEdge enables multiple cameras to stream data and their images to be analyzed in real-time on the network Edge even though the Edge machine has limited GPU and CPU resources. It leverages insights about the structure of DNNs to schedule tasks in a way that is able to achieve good performance for small workloads, and gracefully degrades when the workload increases.

1 Introduction and Motivation

Several applications of the Internet of Things (IoT) need to operate in scenarios where connectivity to the cloud is weak or non-existent. These cloud connectivity issues commonly arise for verticals, such as agriculture, oil and gas, energy, and mining. A commmon technique to overcome weak connectivity is to use a powerful Gateway device, also sometimes referred to as the Edge. The Edge performs computation, such as drone video processing [4] to avoid shipping gigabytes of data to the cloud, or asset monitoring to prevent machine downtime [1]. The Edge is also being considered as an important component of the 5G standardization process [2]. Within the 5G standard, Edge compute is expected to enable next generation applications with unique requirements [3], such as low latency, lesser cloud bandwidth consumption, and improved battery life of mobile devices.

Existing Edge frameworks typically run standard IoT workloads, such as, streaming analytics, device management, etc. However, there has been recent interest in running another compute-heavy workload on the Edge, that of DNNs. DNNs are multi-layered machine learning frameworks that build models from large amounts of data. They have been shown to be extremely useful for various types of data, especially for images, audio, and video streams.

For example, in an IoT application, several cameras could be mounted on the ground, or on the mining truck, etc., and the DNNs can be trained to flag security alerts, or detect anomalies in the system. Instead of running DNNs in the cloud, there are several benefits of running the DNN at the Edge. It helps the system run offline, saves bandwidth, and also reduce latency to the application.

Despite the benefits of running DNNs on the Edge, it is a non-trivial to support a typical DNN on the Edge scenario for the IoT or 5G networks. These scenarios require the Edge to support multiple cameras, and multiple DNN models, e.g. for body detection, cow detection, etc. Ideally, each DNN on each camera stream is expected to run on a dedicated GPU to help it execute at frame rate, e.g. 30 fps. However, a typical Edge server has limited GPU and CPU resources. Hence it is challenging to support multiple cameras, each with multiple DNN models, to achieve good performance on a typical Edge device.

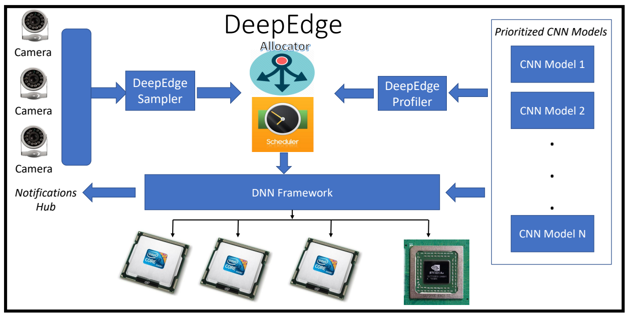


Figure 1: The DeepEdge system components. On the left each camera is streaming video data which is handled by DataEdge Sampler. On the right we define the DNN Models that the user wants to run on incoming video streams. DeepEdge Scheduler and Allocator takes both of these as inputs along with Offline trained model profiler for optimal resource allocation and scheduling.

2 DeepEdge Overview

The goal of the DeepEdge system is to support Deep Learning workloads on Edge devices with a limited number of GPU and CPU cores. It strives to achieve these goals, while ensuring three desired properties. **Performance:** to minimize the time taken to process a given workload. **Graceful Degradation:** With an increase in the number of camera streams, and DNN models, DeepEdge throttles workloads without starving any stream or model. **Support for available DNN models:** there are several 1000s of DNN models developed using many frameworks such as TensorFlow, Caffe, CNTK etc. that have been built by engineers and data scientists worldwide. Instead of shipping Edge devices with pre-trained models, our goal is to support a network operator to plug-and-play any of the existing models that are available on the Internet. This design also allow us to model complex interactions across DNN models. DeepEdge can trigger a face recognition model in CNTK whenever a person detection model fires a positive result in TensorFlow.

We have built DeepEdge, and compared its performance against traditional schedulers and resource allocators used in existing operating systems. Our results show that DeepEdge gives 10X speedup over alternative approaches.

The DeepEdge system achieves the above goals through the components described in Figure 1. It allows a network operator, who is the primary user of the Edge device, to download the DNN models. Right now this is through Github, although in the future, this could be from something like a DNN App store. Various data streams, e.g. cameras, are registered with the DeepEdge service. And for each stream, the network operator specifies the corresponding DNN models to be run on them. The key components of DeepEdge are described below

DeepEdge Model Profiler: After a new model, with a given architecture and framework, has been downloaded to the Edge, the system need to know its resource requirements, such as how much time would it take to run on some number of CPU cores, or on the GPU, or when sharing the CPU cores with other workloads. Since running all possible scenarios is extremely time consuming, and difficult in an operational system, the Model Profiler introduces a new ML-based technique for estimating the resource requirements of each DNN. DeepEdge Model Profiler learns the dependency of the tunable DNN Parameters such as Sampling rate, Batch Size, Precision and system resources such as CPU, GPU, Memory utilizations on the performance throughput of the DNN models.

DeepEdge Sampler: Cameras usually stream at 30 fps with more advanced cameras streaming at as high as 120 fps. This results in lot of redundant information to be processed for each stream, traditional systems handle this by doing key frame extraction to identify unique frames by using metrics such as Structural Similarity(SSIM) or PSNR. DeepEdge Sampler does this in a much better way where it selects an optimal sampling rate assignment for each stream which depends on the relative importance of the DNN and the incoming stream. This is one of the key components which has a direct correlation with the time and resource requirement of a DNN.

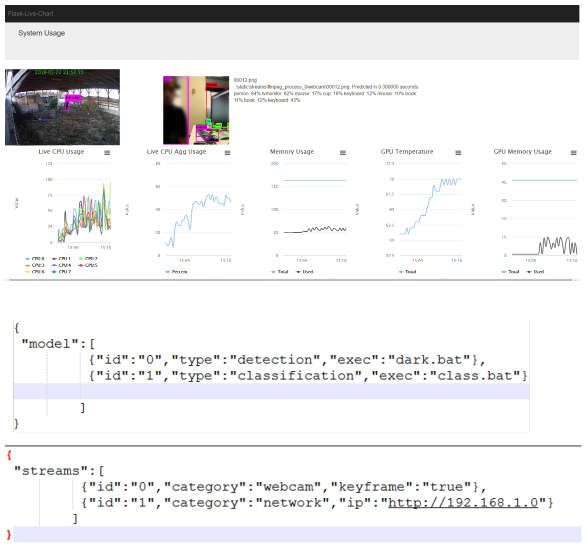
DeepEdge Allocator: The Allocator uses the Model Profiler to handle system resource allocation for the DNN workloads. The Allocator consists of two subcomponents

System Allocator: This handles the allocation of CPU/GPU Core to each DNN based on resource requirement of a DNN and current system utilization.

DNN Parameter Allocator: takes the input from DeepEdge CPU Allocator and Model Profiler to assign various DNN parameters to each of the DNN workloads so as to maximize the user specified optimization criteria.

DeepEdge Allocator takes the input as learned model profiler for each DNNs and current system utilization and formulate it as an optimization problem for assigning system resource and DNN parameters to each DNN while maximizing the defined objective function and following the constraints that arise from hardware limitations. The output of DeepEdge Allocator is then fed into DeepEdge Scheduler that decides the execution scheme for each of the DNNs.

DeepEdge Scheduler: The Scheduler leverages the insights about DNN structures to handle when and how to run each DNN model. Compared to traditional user workloads, DNN workloads are predictable. Convolutional layers require less memory but more compute time whereas fully connected layers require more memory but less compute time. The Scheduler exploits the inherent properties of these layers to handle execution of the DNNs in an efficient way while maintaining the resource allocation suggested by DeepEdge Allocator.



```
00012.png
..\static\streams\ffmpeg_process_0webcam\00012.png: Predicted in 0.300000 seconds.
person: 84% tvmonitor: 82% mouse: 17% cup: 18% keyboard: 12% mouse: 10% book:
11% book: 12% keyboard: 43%
```

```
{
 "model":[
          {"id":"0","type":"detection","exec":"dark.bat"},
          {"id":"1","type":"classification","exec":"class.bat"}

         ]
}
```

```
{
 "streams":[
          {"id":"0","category":"webcam","keyframe":"true"},
          {"id":"1","category":"network","ip":"http://192.168.1.0"}
         ]
}
```

Figure 2: Snapshot of DeepEdge running multiple incoming Data Streams on a Laptop with live resource utilization viewer

3 Conclusion and Ongoing Work

In this paper we have proposed a new architecture for the IoT Edge, called DeepEdge, which was necessitated by our work on FarmBeats – an IoT system for Agriculture. DeepEdge runs on a PC form factor device, and is able to support multiple camera streams and DNN models. DeepEdge leverages new techniques for profiling the DNN models, which are then used to schedule the workloads based on the structure of the DNN, and allocate the resources for each model and camera stream based on their importance. Our initial results are very promising. We are actively exploring the more recent advances in resource allocation using reinforcement learning, and also extending the framework to work with new DNN ASICs.

References

[1] Ams 6500 machinery health monitor. http://www.emerson.com/en-us/automation/asset-management/asset-monitoring/condition-monitoring/ams-6500-machinery-health-monitor.

[2] Mobile broadband standard. http://www.3gpp.org/release-15.

[3] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI White Paper*, 2015.

[4] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, and et al. Farmbeats: An iot platform for data-driven agriculture. In *NSDI*, 2017.