May 2023

CONSTRAINED MACHINE LEARNING MODEL DEPLOYMENTS FOR OPERATIONAL TECHNOLOGY NETWORKS

Robert Barton

Indermeet Gandhi

Jerome Henry

CONSTRAINED MACHINE LEARNING MODEL DEPLOYMENTS FOR OPERATIONAL TECHNOLOGY NETWORKS

AUTHORS:
Robert Barton
Indermeet Gandhi
Jerome Henry

ABSTRACT

Presented herein are techniques for formulating level and device specific machine learning (ML) models for operational technology (OT) networks that can be deployed closer to an end device (in a respective level) for constrained devices.

DETAILED DESCRIPTION

Operational technology (OT) industrial networks, such as Industrial Internet of Things (IIoT) networks are organized into different levels. These levels range from Level 0 (Process Control) to Level 4 (Enterprise) and each level is responsible for a specific function in an IIoT environment. The industrial Ethernet (IE) switches deployed at each level facilitate communication between different OT devices within the same level.

In Level 0, the IE switches connect to various OT devices such as sensors, actuators, and programmable logic controllers (PLCs). These devices are responsible for controlling the physical processes in the IIoT environment. The IE switches at Level 0 typically use proprietary protocols to communicate with the OT devices.

At Level 1, the IE switches connect to the controllers that manage the physical processes in the IIoT environment. These controllers include Distributed Control Systems (DCS), Supervisory Control and Data Acquisition (SCADA) systems, Programmable Automation Controllers (PAC), and the like. The IE switches at Level 1 typically use standard Ethernet protocols such as Modbus, OPC, and Ethernet/IP to communicate with the controllers.

In Level 2, the IE switches connect to devices responsible for managing production processes, such as Manufacturing Execution Systems (MES) and Human-Machine Interfaces (HMI). The IE switches at Level 2 use standard Ethernet protocols such as TCP/IP and HTTP to communicate with these devices.

In Level 3, the IE switches connect to devices responsible for managing the data collected from the OT devices in the lower levels. These devices can include Historians, Data Warehouses, and Analytics engines. The IE switches at Level 3 use standard Ethernet protocols such as Message Queues Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and Representational State Transfer (RESTful) Application Programming Interfaces (APIs) to communicate with these devices.

Finally, in Level 4, the IE switches connect to the devices responsible for managing the business processes in the IIoT environment. These devices include Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) systems, and Supply Chain Management (SCM) systems. The IE switches at Level 4 use standard Ethernet protocols such as Simple Object Access Protocol (SOAP) and Extensible Markup Language-Remote Procedure Call (XML-RPC) to communicate with these devices.

The constrained IE switches deployed at each Purdue model can be used to run different processes. In a world where machine learning (ML) is distributed to the edge, they can also run different ML models (for training, inference, or reinforcement). For example, the IE switches at Level 0 may use a proprietary protocol to communicate with the OT devices, while the IE switches at Level 1 may use a standard Ethernet protocol such as Modbus. These are different protocols and hence, the data generated, and the ML models used for inferencing that should be executed at each level are different.

Examining this further, edge computing needs to be viewed as a constrained resource that has contextual awareness of which level of the OT network at which it is deployed. For example, an ML implementation may deploy training at the edge with the goal of maximizing resources and computing as close as the source as possible. However, due to the lack of compute resources at the edge (e.g., Graphics Processing Units (GPUs)), the learning or inference task may leverage nodes that reside at different levels of the Purdue model, thereby, causing a violation of the security model.

Thus, there is a need for a method that distributes ML computing at the edge without violating the Purdue structure.

Presented herein are techniques for formulating level and device specific ML models for OT networks that can be deployed closer to an end device (in their respective levels) for constrained devices.

Consider various steps for achieving the techniques of this proposal. For example, IE switches can be deployed at each International Electrotechnical Commission (IEC) 63443 / Purdue level of an OT network.

Increasingly, artificial intelligence (AI) and/or ML tools are being developed for OT applications, such as predictive maintenance, sensor anomaly detection, and much more. Due to the processing requirements of AI/ML, this function has generally been relegated to the cloud only - edge processing has not had enough compute power (GPUs) to perform AI/ML tasks.

Industrial networking components increasingly support edge compute resource capabilities; however, these are often constrained and often do not support GPUs. Still, using very small models (e.g., TinyML), industrial network components do have the ability to support minimal ML jobs.

Next, an asset inventory system can build the topology of the OT network and creates a contextual inventory of all networking nodes in the network. Included in this topology are nodes that support edge computing and an estimate of their compute capability. Furthermore, the industrial assets are inventoried (e.g., using a network visibility tool, etc.).

A catalogue of OT-specific ML models that are available for use can then be provided. These may include vendor-specific models for anomaly detection, preventive maintenance, etc. (e.g., a model published by a device vendor). Some of these models may be very specific to a type of IoT device (e.g., a PLC, a drive motor, a vehicle logic unit, etc.).

The models can also be associated with performance parameters. For example, in one instance, the parameter can be indicated as an 'O' value representing the computation cost (for training, for inference). In another instance, the parameters can represent the computation time on key reference platforms (for training, in epoch or grade steps units, for inference, in samples examined per unit of time).

Thereafter, each asset, at each level of the Purdue model, can be associated with its memory and compute availability. It is well-known that the central processing unit (CPU) of network devices tend to run hot (e.g., the CPU consumption is high even when the device is idle). Thus, the memory/compute availability can represent the difference between 100% utilization and the idle level. Similarly, each asset can be associated with a volume of

produced data per unit of time (that depends on the device role details and location in a chain).

By examining the asset inventory, an administrator can monitor each asset memory/compute utilization (e.g., current, and/or min/mean/max over an interval), the associated volume of data generated, and can select which type of ML application can be used on each platform. The system next examines all nearby edge compute devices as candidates to publish a desired inference model. The objective is to push the model to a nearby edge compute node that have capacity to support its training, and/or the inference task (if the model is already trained).

For the next step, for each Purdue level, various data can be observed, such as the memory/compute availability on each platform, the platform count and the type of tasks needed (training, inference), each associated with a priority level, and the system starts by distributing the high priority tasks. It should be clear that the output of some tasks at lower levels may be needed for upper levels tasks to be performed, and the deployment takes this constraint into account (e.g., by assigning a priority premium for tasks displaying such dependency).

In many cases, training a model on a single source of data causes high bias (and is therefore undesirable). Thus, the system can evaluate–for each training case–the optimal training structure (i.e., one that completes on lowest amount of time for data from all relevant sources at a given Purdue level, while minimizing inter-device data transfer).

In some cases, a single unit may be able to receive data from all others and perform the training compute task. In other cases, training may need to occur on more than one device. Distributions techniques are inserted into the target assets (e.g., mini-batching on local data, before the resultant weights are shared with the neighboring platform and the next mini-batch is performed; a task supervisor or the like distributing the compute task (e.g., TinyML or equivalent) between platforms with a 'dining philosopher' approach to distributing data between 'n' nearest neighbors).

In some cases, the model may be trained such that inference is needed. In such cases, the memory/compute allocation can be used to find the platform nearest to each production source that can complete the inference task faster than the data sampling rate.

Once the deployment and device types have been identified, the system breaks the aggregated ML model into smaller modular ML models, each model with its own weights and feature variables to consider for each level and use-cases.

For example, consider a deployment involving electrical power distribution in which Level 0 may include process sensors and actuators for which ML models are responsible for predictive maintenance of transformers in order to detect anomalies and predict failure(s). While in Level 1, the use-cases (ML models) may focus on basic control processes, such as the adaptive control of voltage regulation systems to optimize control parameters based on real-time sensor data. Next, in Level 2, the ML models may involve forecasting power based on real-time demand, and so on.

Thus, specific models utilized at different levels may be capable of solving use-case(s) suited for a particular level and for a particular device type. Stated differently, one of the ML models can only infer anomaly detection for a microswitch to determine if the electrical power consumption is exceeding the baseline threshold in order to take real-time corrective actions at Level 0. In other scenarios, the device specific ML model can make real-time predictions at Level 2.

Thus, techniques herein may provide for the ability to formulate and deploy level and device specific ML models for OT networks closer to the end devices at each of multiple levels for different constrained devices that may be utilized at each level.