A Simple Workflow to Deploy and Update Neural Networks on Microcontrollers for Inference with the Pelion IoT Platform

Yue Zhao*[1], Austin Blackstone*[2], Neil Tan[3], Damon Civin[1]
Arm, San Jose, CA[1]; Austin, TX[2]; Taipei, Taiwan[3]

Background & Motivation

Neural networks have demonstrated state of the art performance in a variety of complex tasks, such as computer vision and speech recognition. In many applications, low hardware cost, low communication cost, energy-efficiency, and/or privacy concerns are desired. Microcontrollers provide these properties and are widely used in industrial settings.

In spite of their limited memory and compute resources, many studies have shown the possibility of deploying complex neural networks on microcontrollers. This is achieved through model compression methods like pruning and quantization. For example, popular architectures such as CNN and LSTM have deployed on microcontrollers for keyword spotting and image recognition tasks. CMSIS-NN, a library of optimized kernels for neural networks, has been developed to maximize performance and minimize memory footprint on Arm Cortex-M processors targeted for IoT devices.

However, in practice, **one needs to update the neural networks or other software, often remotely.** In this work, **we demonstrate a complete workflow to enable scalable remote updates from the cloud to the neural networks deployed on microcontrollers**. This system is based on **TensorFlow, Mbed OS enabled microcontrollers and the Pelion Device Management platform.**

The demo will run on a DISCO-F413ZH (Cortex M4 based running at 100MHz with 320 kB SRAM and 1.5 MB Flask) board connecting to Pelion through WIFI.

Mbed OS & Pelion Device Management & uTensor

- **Mbed OS** is an open-source real-time operating system (RTOS). It is designed specifically for IoT devices. The associated toolchain allows for straightforward compilation of C/C++ applications.
- **Pelion Device Management** securely connects and recognizes devices with standards-based communication using Open Mobile Alliance's LWM2M model and Constrained Application Protocol (CoAP). It also offers an end-to-end management and monitoring of remote updates. The Device Management Portal provides visualization of raw data and results of inference.
- **uTensor** is an extremely light-weight framework that converts a TensorFlow model saved in protobuf file to embedded C++ code. This program is ready to be compiled with Mbed OS and flashed to the devices.
- Disco-F413ZH dev board:
  ARM® 32-bit Cortex® -M4 CPU
  100 MHz max CPU frequency
  1.5 MB Flash, 128-Mbit Quad-SPI
  320 KB SRAM, 8-Mbit PSRAM; 512K word × 16bits
  Integrated Wi-Fi ® module 802.11 b/g/n
  Connector for microSD™ card
  240 × 240-pixel LCD
  I2S audio codec

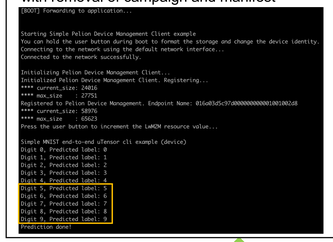
A Simple Workflow to Deploy and Update Neural Networks

Pelion Update

Remotely updating firmware

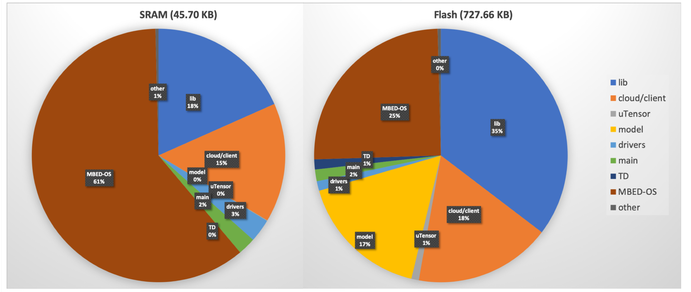
Remotely onto devices via device ID and update certificates

States: initializing, checking manifest, publishing and finishing with removal of campaign and manifest

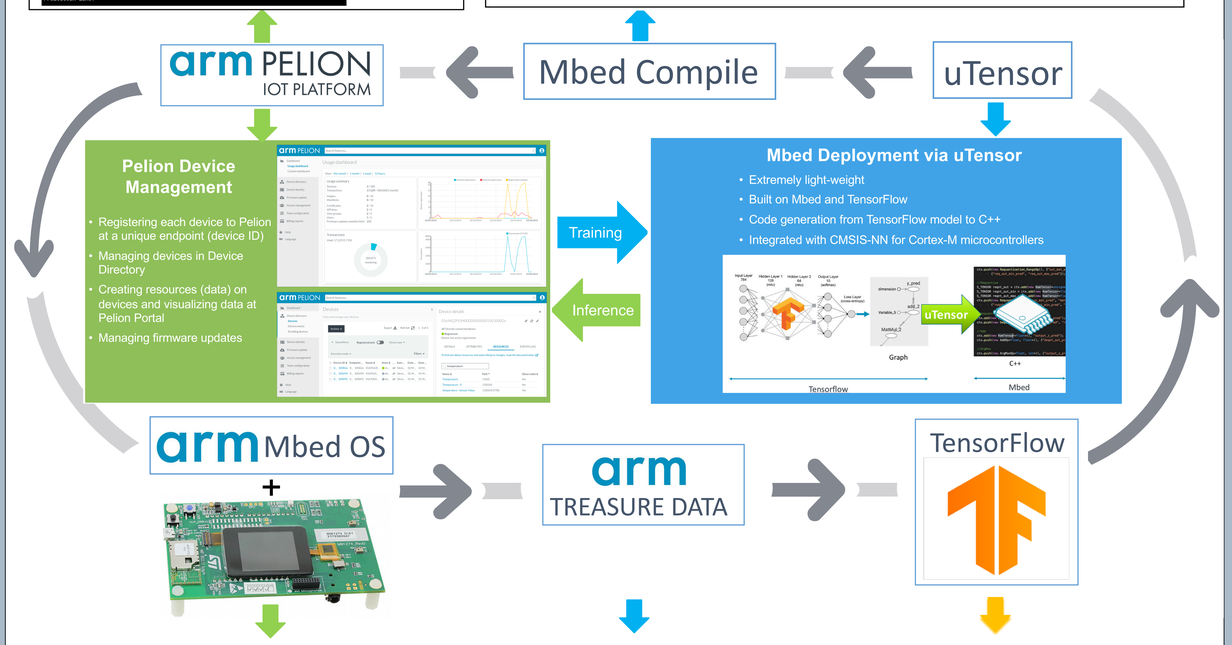


The output from the serial terminal shows that the updated model predicts all the digits correctly,in contrast to the old model, indicating the model update is successful.

Mbed Compiler (Mbed CLI)

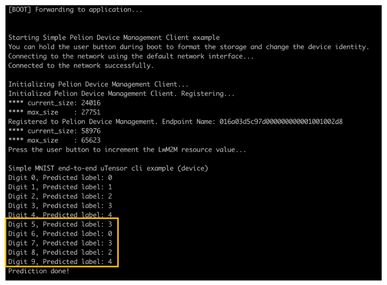


Summary of Static RAM and Flash distribution in library, Mbed-OS, Cloud/Client, uTensor, etc.



Inference on Mbed Enabled MCU

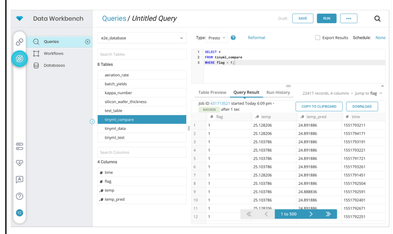
- 320 KB in SRAM, 1.5 MB in FLASH, M4 100 MHz
- Connecting to WIFI and registering with Pelion
- Loading the digit images that are saved in FLASH
- Returning inferences from a neural network
- Also possible to post data to cloud databases



The output from the serial terminal shows that the flashed model only predicts the digits from 0 to 4 correctly.

Treasure Data Database

- Creating a database API key
- Creating a schema for a table
- Supporting machine learning in Hivemall SQL
- Supporting SQL queries with Presto for other environment (e.g. Jupyter Notebook)



A snapshot of Treasure Data database interface.

Training with TensorFlow

- 55,000 images in training set, 5000 images in validation set, and 10,000 images in test set
- Dense neural networks (DNNs) and convolutional neural networks (CNN) are trained in 100 epochs with a batch size of 50.
- For each architecture, two versions are trained. For example, DNN-1 and CNN-1 are only trained with the digits from 0 to 4, while DNN-2 and CNN-2 are trained with all the digits from 0 to 9.
- Saving models to checkpoints
- Writing a graph into a protobuf (pb) file

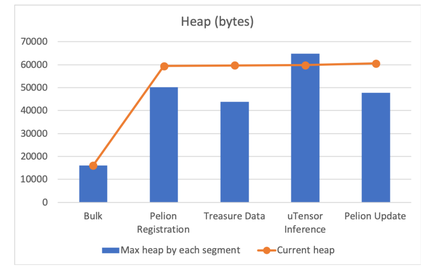
Model	Train	Val.	Test	FLASH	SRAM
DNN-2	99.9%	97.7%	97.7 %	136 KB	6 bytes
CNN-2	99.6%	99.4%	99.2%	122 KB	6 bytes

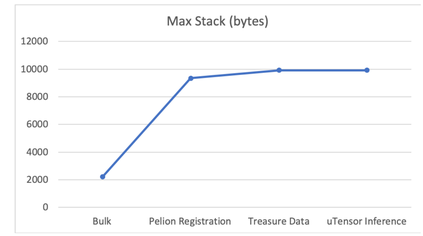
Table for the training, validation and test accuracy together with SRAM and FLASH memory for DNN and CNN.

Model Training

The DNN and CNN are trained with the MNIST dataset of handwritten digits in TensorFlow. The DNN has 3 fully-connected layers with 128, 64 and 10 neurons. The CNN has 3 convolutional layers with 5 by 5 filters (16, 32, 64 features), 3 max-pooling layers with the strides of (1, 2, 2, 1), 4 dropout layers (0.9, 0.75, 0.75, 0.5), a fully-connected layer and a softmax layer. Two models are trained for each architecture to test if the update is successful. DNN-1 and CNN-1 are only trained with the digits from 0 to 4, while DNN-2 and CNN-2 are trained with all the digits from 0 to 9. The test accuracy is 97.8% for DNN-2 and 99.2% for CNN-2. Although it is not state-of-the-art, we focus on testing the remote update.

Dynamic Heap and Stack Usage Monitoring





In addition to SRAM and Flash at compile time, the above figures show the heap (top) and the stack (bottom) usage for each segment at runtime, which is extracted from Mbed Stats APIs. As shown, the heap usage spikes temporarily with the Pelion registration/update, TD API and the CNN inference. But extra heap is released except for the Pelion registration. The CNN inference also consumes 1.1 KB in stack.

Summary

In this work, a complete PoC workflow is shown to enable scalable remote deployment and updates for neural networks on microcontrollers, based on TensorFlow, Mbed OS enabled microcontrollers and the Pelion IoT platform. In the future, we will look into more compelling use cases, optimize data transfer with Fluentd, find out the correlation between model complexity and heap/power usage, and explore Pelion delta update.