MLIoT: An End-to-End Machine Learning System for the Internet-of-Things

Sudershan Boovaraghavan, Anurag Maravi, Prahaladha Mallela, Yuvraj Agarwal

University of Minnesota
Driven to Discover℠

Authors

Sudershan Boovaraghavan
PhD Student

Anurag Maravi
Research Intern

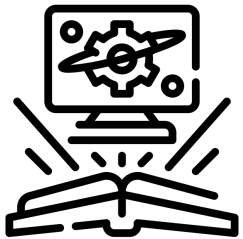
Prahaladha Mallela
Master's Student

Yuvraj Agarwal
Associate Professor

Carnegie Mellon University

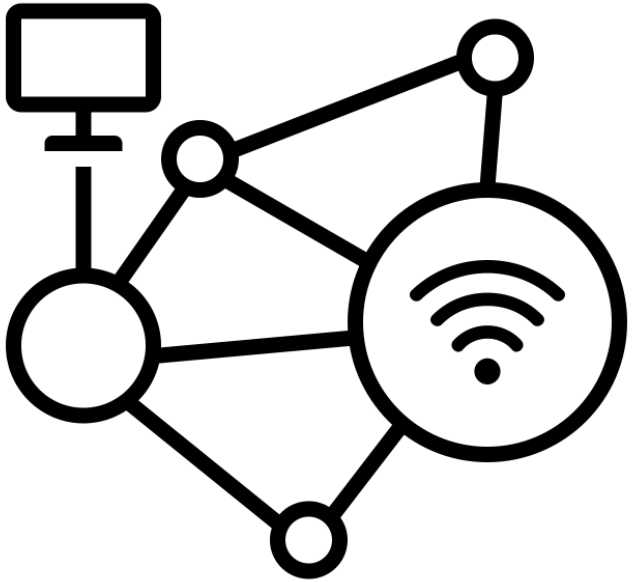
Institute for Software Research

IoTDI 2021 - ACM/IEEE Conference on Internet of Things Design and Implementation, 2021



What is IoT?

- Internet of Things
- System of internet-connected objects that are able to collect and transfer data over a wireless network
- Make predictions based on ML training



Limitations/Issues with IoT Applications

- **Issues**:
  - Current systems rely on general-purpose pre-trained ML models
  - Retraining based on the sensors and events in a dynamic setting isn't possible due to the architecture of the systems
  - Accurate pre-trained models need a significant amount of labeled training data, and computation resources, which is uncommon in IoT scenarios
  - These systems are optimized for specific, and often dedicated hardware resources, and do not adapt to changes in resource availability
- **Goal**:
  - Train a generalized ML model once and then deploy it to make predictions and have it adapt to environmental changes over time

Proposal: MLIoT

- End-to-end ML System tailored towards supporting the entire lifecycle of IoT applications including initial training, serving, and retraining processes
- Adapts to changes in IoT environments or compute resources by enabling retraining, and updating models on the fly
- Maintains accuracy and performance

Key Contributions

1. Design and implementation of MLIoT
2. Propose optimizations for training and serving, and report on their efficacy
3. Evaluate MLIoT on several hardware devices and across a set of IoT benchmark datasets (image, audio, multi-modal sensor data)

IoT Applications

- Activity Recognition using Multi-Modal sensors
  - Use multiple datasets from the Mites to evaluate MLIoT
- Audio Based Activity Recognition
  - Use audio benchmarks and compare MLIoT with Ubicoustics
- Object Recognition using Image Data
  - Use the MNIST dataset which has labelled images for handwritten digits to evaluate MLIoT

Challenges for ML systems in IoT Settings

- Adapting to Different IoT Application Requirements
- Adapting to Device Capabilities and Resource Availability
- Adapting to Changes in Environmental Context
- Customization to IoT Data Types
- Need for an End-to-End ML Service for IoT

MLIoT Goals

- Balance user requirements with the available resources
- Use various devices and models to see how results compare
- Enable both the initial training, and re-training and re-optimizations of the models based on user driven corrective feedback

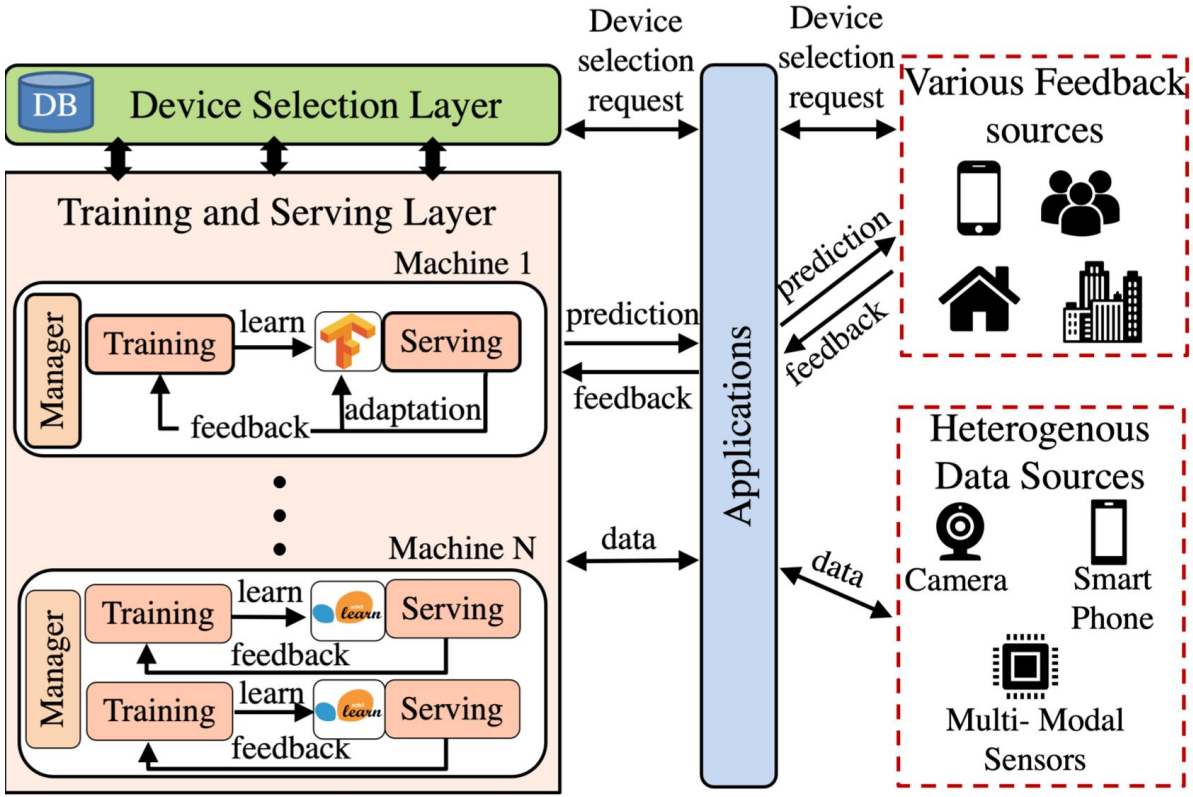
Related Work

- Serving-Only
  - These systems place the trained models in containers and optimize model inference requests
  - fail to address one or more key requirements for IoT scenarios: focus on serving static pre-trained models, no adaptation to environmental changes, do not support heterogeneous devices, or have limited, if any, support for policies
- Training-Only
  - Most of the training focused systems optimize for deep neural network models with many hyperparameters, which is very resource and time intensive
  - the available data is sparse to train complex models, the trained models need to be specific to the sensor sources, environmental context, application requirements and thus need more closely coupled (re-)training and serving systems
- Hybrid Training-Serving
  - simplify ML development through a general-purpose machine learning system with both training and serving of models
  - generally run on the cloud incurring a higher cost for better workload environments and restrict users to a specific set of algorithms or libraries, so users are on their own when they step outside these boundaries
  - not designed to adapt to unpredictable operational environments

	Related Work	Requirements in IoT						
		Approach	Sources	Adaptive Model Selection	Expressive Polices	Adaptation to Heterogeneous Hardware	Adaptation to Environment	End-to-End
Academic Systems	Clipper [14]	Serving	Any Src	✓	◆	✗	✗	✗
	InferLine [15]	Hybrid	Any Src	✓	◆	✓	✗	✗
	Ubicoustics [38]	Serving	Audio	✗	✗	◆	✗	✗
	Helix [66]	Training	Text	✗	✗	✓	✗	✗
	Project Adam [11]	Training	Image	✗	✗	◆	✗	✗
	KeystoneML [63]	Training	Text	✗	✗	✗	◆	✗
	Mites [49]	Hybrid	Multimodal	✗	✗	✗	✓	✓
	Velox [13]	Hybrid	Any Src	✗	✗	✗	✓	✓
	Laser [67]	Hybrid	Text(Ads)	✗	✗	✗	◆	✓
Commercial	TensorFlow Serving [51]	Serving	Any src	✗	◆	✗	✗	✗
	AWS IoT Greengrass [5]	Serving	Any Src	✗	✗	✓	✗	◆
	Microsoft's ML.Net [2]	Hybrid	Any Src	✗	✗	✗	◆	✓
	Google's Cloud ML [26]	Hybrid	Any Src	✗	✗	✗	✓	✓
	Google's TFX [6]	Hybrid	Any Src	✓	✗	✗	✓	✓
	MLIoT	Hybrid	Any Src	✓	✓	✓	✓	✓

✓: Available ✗: Not Available ◆: Partial features available

MLIoT Design



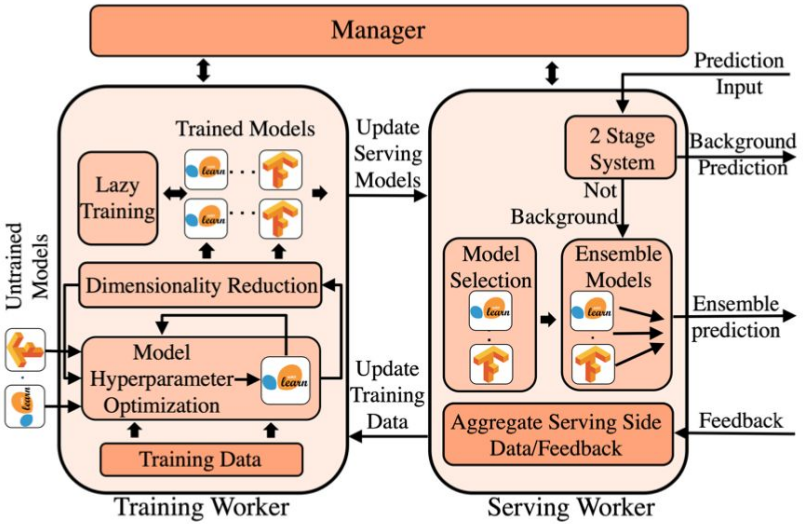
DSL Benchmarks

- CPU and Memory Utilization
- Prediction Latency
- Training Time and Accuracy
- Runtime Metrics

TSL

- Training Worker
  - creates initial model training using labeled data
  - retrains models when corrective feedback is provided by users to improve accuracy
  - streamline model training to provide high-quality ML models that are tailored to the specific IoT task
  - goal with TW model selection is to create an ensemble of models to send to the Serving Worker

- Serving Worker
  - obtains the trained models from the TW
  - uses model ensemble to make predictions
  - collects user feedback on the ensemble-based predictions and forwards them to the TW



Optimizations

- Hyper-parameter optimization and dimensionality reduction of models
- Lazy training to increase accuracy over time, while reducing latency to start serving
- Model adaptation to account for drift using corrective feedback from users
- Two-stage serving that increases accuracy while improving serving latency
  - First Stage is a binary classifier with two classes - “Background” vs. other classes
  - Second Stage uses the model ensemble obtained to serve predictions

- w is the weight
- $f(x)m_i$ is the prediction made by model m
- c is the class

$$Prediction = argmax_c(\sum_{i=1}^{n} w_i(if\ f(x)_{m_i} = c))$$

Experiment

- Five traditional models (KNN, Ridge Regression, RandomForest, SVM-Linear, SVM-RBF)
- Two Neural Networks (MLP & XGboost)

Table 2: Summary of IoT Benchmark Data Characteristics

Dataset	Data Type	Features	Training Samples	Testing Samples	Classes
MNIST [40]	Image	28*28	42000	18000	10
Mites [49]	Multimodal	1172	25787	16735	16
Microphone	Audio	96 * 64	2633	1734	14

Table 3: Hardware platforms used in our experiments

Device	Type	Processors	Memory	Average RTT	Local /Cloud
M1	Raspberry-Pi 4	4 x ARM A72	4GB	~3ms	Local
M2	Intel NUC	4 x i5-5250U	8GB	~14ms	Local
M3	Virtual Machine	2 core	4GB	~128ms	Cloud
M4	Virtual Machine	8 core	16GB	~64ms	Cloud
M5	Virtual Machine	16 core	32GB	~84ms	Cloud

Results

- Training time, CPU and memory usage vary widely based on the models
- Performance is machine dependent and data type dependent
- The average latency with a two-stage system is 65ms, compared to the average latency of the baseline system of 105ms
- MLIoT improves accuracy 50% - 75%
- MLIoT (DSL) does not schedule any more additional tasks on a machine since servicing them would lead to a higher serving latency than the threshold

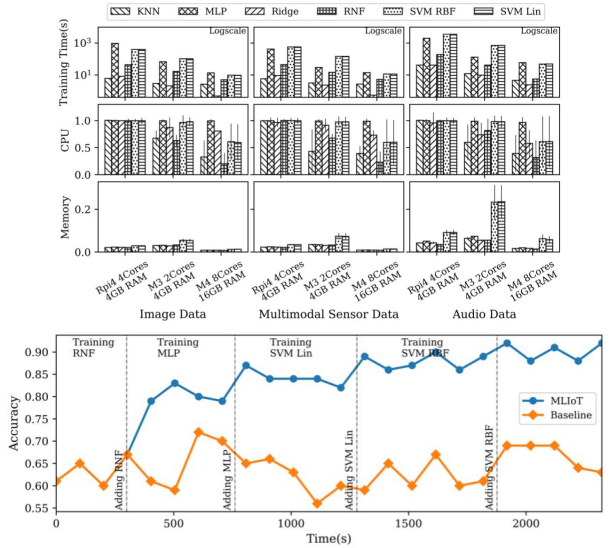


Figure 6: Change in accuracy with Lazy training. The latency to start serving is reduced while accuracy improves as the ensemble is updated with more models trained in the background.

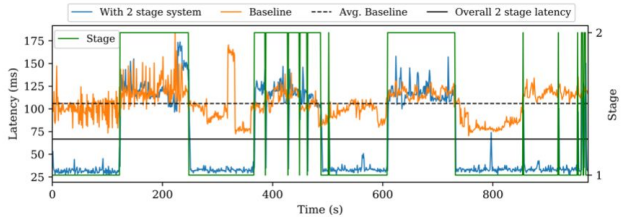


Figure 8: Effect of Two-stage serving on the End-to-End Latency of MLIoT and the corresponding stage where the prediction is made (green line). The Two-stage system's binary model classifies background with lower latency when compared to baseline.

*Note: baseline model is without DR and HPO

Results (Part 2)

Table 4: Comparing MLIoT with other ML systems: Ubicoustics [38] Mites.io [49] and a general-purpose system: TensorFlow Extended [6]. Numbers in parenthesis are percentage increase/decrease in accuracy (higher is better) or latency (lower is better).

System	Top 1 Accuracy	Latency (s)
Audio Data		
Ubicoustics [38] - Pretrained Model	0.52	0.08
MLIoT- Best Effort (7 ensemble)	**0.89 (+71%)**	0.1 (+25%)
MLIoT- Low Latency (3 ensemble)	0.86 (+65%)	**0.06 (-25%)**
TFX [6] - General Purpose	0.67 (+29%)	0.35 (+337%)
Multi-modal Sensor Data		
Mites [49] - Supervised Model	0.48	0.05
MLIoT- Best Effort (7 ensemble)	**0.84 (+75%)**	0.09 (+80%)
MLIoT- Low Latency (3 ensemble)	0.72 (+50%)	**0.04 (-20%)**

Discussion

- What improvements are needed to make it ~100% effective?
- Are there any alternate architectures that can improve the efficacy?
- Are the experimental results representative and achievable in real-world implementation at scale?
- What is the cost of implementing this?