Master Course Description for EE-400 A (ABET sheet)

Title: Tiny Machine Learning for Ultra Low-Power Edge Computing (TinyML)

Credits: 4

Coordinator: Radha Poovendran, Professor, Electrical and Computer Engineering

Goals:

This hands-on course studies the design and deployment of machine learning (ML) on ultra low-power edge devices (microcontrollers, embedded sensors). With billions of Internet of Things (IoT) devices worldwide, the ability to build compact yet accurate models under tight memory, compute, and energy budgets is essential for responsive, private, and reliable applications at the edge.

Students will build end-to-end TinyML pipelines in class – from sensing and data collection to model training, compression, and deployment. Applications span keyword spotting, visual wake words, anomaly detection, predictive maintenance, gesture recognition (magic wand), sign-language interpretation, and smart-lock audio recognition.

A recurring theme in the course is engineering under constraints: selecting algorithms, compression methods (quantization, pruning, knowledge distillation), and runtime designs that meet latency/footprint targets while preserving accuracy. We also highlight robustness and security of edge deployments ("Robust TinyML"), recognizing that adversaries can manipulate models and inputs in the wild.

Learning Objectives: At the end of this course, students will be able to:

1. Deploy TinyML models on power- and performance-constrained devices to solve real-world problems.
2. Implement ML algorithms such as clustering, regression, classification, and ensembles; compare baselines with compressed neural models.
3. Use Python libraries (NumPy, Pandas, Scikit-learn) for data preparation, training, and evaluation.
4. Use TensorFlow for deep learning and TensorFlow Lite / TFLite Micro for TinyML deployment.
5. 5Use C/C++ to integrate embedded inference (e.g., Arduino Nano 33 BLE Sense) and simple I/O (BLE/serial/LED).
6. Measure accuracy, latency, memory footprint, and (when feasible) energy; explain trade-offs due to compression.
7. Apply PTQ, QAT, pruning, and knowledge distillation appropriately and articulate the design trade-offs.

Textbook:

TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, Pete Warden & Daniel Situnayake; O'Reilly Media, 2020.

Reference Texts:

1. TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter, Gian Marco Iodice; Packt Publishing, 2022.
2. TensorFlow / TensorFlow Lite documentation.
3. TinyML Foundation resources (tinyml.org).
4. SensiML documentation (as applicable to Week 9 activity).

Prerequisites:

Either CSE 163, or EE 241; either AMATH 352, MATH 208, MATH 308, or MATH 136; and Either IND E 315, MATH/STAT 394, STAT 390, or EE 391; or instructor permission.

Familiarity with Python programming; basic C/C++ helpful for MCU deployment. No prior ML coursework required (basics covered early in the course).

Topics:

1. Introduction to TinyML; Python & toolchain overview; TinyML lifecycle and constraints [Week 1]
2. Compression foundations: pruning, PTQ vs QAT, and knowledge distillation; TFLite for TinyML [Week 2]
3. Keyword Spotting on streaming audio: data collection, training, metrics, on-device deployment [Week 3]
4. Visual Wake Words: datasets, MobileNets, transfer learning, deployment [Week 4]
5. Anomaly Detection: signal processing, unsupervised learning, thresholds, deployment [Week 5]
6. Wizard Magic Wand: BLE-based gesture tracking, CNN sketch, data collection/labeling, deployment [Week 6]
7. Predictive Maintenance: sensors (accel/gyro/baro/mag), data interfaces, TinyML framework, deployment [Week 7]
8. ASL Interpretation: motion features, neural modeling, TinyML pipeline, evaluation and deployment [Week 8]
9. Smart-Lock Audio Recognition: audio processing, TensorFlow/SensiML SDK, compile/flash, live inference UI [Week 9]
10. Lecture or additional time for the final project [Week 10]
11. Final Project Presentations (teams of 3–4; 9’ talk + 3’ Q&A). Final report due in Exam Week – Week 11.

Course Structure:

Integrated labs occur during the second part of class; short in-class quizzes reinforce weekly material.

A team-oriented project (3–4 students) runs through the quarter with proposal, presentation, and final report.

Computer Resources:

Python (NumPy, Pandas, TensorFlow/TFLite), TFLite Micro, Arduino IDE/CLI, BLE utilities, and (as applicable) SensiML SDK.

Students are expected to use personal laptops; ECE department computers may be used if available.

Hardware: Tiny Machine Learning Kit (Arduino Nano 33 BLE Sense).

Grading:

60% In-class lab completion/quizzes

40% Project (Proposal 5% + Final Presentation 15% + Final Report 20%)

Religious Accommodation Policy

Washington state law requires that UW develop a policy for accommodation of student absences or significant hardship due to reasons of faith or conscience, or for organized religious activities. The UW's policy, including more information about how to request an accommodation, is available at Religious Accommodations Policy – https://registrar.washington.edu/staffandfaculty/religious-accommodations-policy/. Accommodations must be requested within the first two weeks of this course using the Religious Accommodations Request form – https://registrar.washington.edu/students/religious-accommodations-request/.

ABET Student Outcome Coverage: This course addresses the following outcomes:

H = high relevance, M = medium relevance, L = low relevance to course.

(1) An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics. (H) Students must identify constraints (SRAM/Flash limits, latency, energy) and select/design algorithms and compression methods to meet them. We emphasize sound engineering principles over ad-hoc heuristics by quantitatively comparing baselines vs compressed models (PTQ/QAT/pruning/distillation) and analyzing failure modes for embedded inference.

(2) An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors. (M) The team project requires designing and implementing an embedded ML solution that meets functional/resource specs. Students consider privacy (on-device processing), safety (false positives/negatives), and deployment context.

(3) An ability to communicate effectively with a range of audiences. (M) Students submit a proposal, deliver a live demo/presentation, and write a final report in an engineering format with figures, metrics, and clear justifications.

(4) An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts. (H) The course discusses risks of ubiquitous sensing and adversarial manipulation in edge settings. Students document assumptions, limitations, and mitigations (e.g., robust preprocessing, anomaly detection, safe defaults).

(5) An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives. (H) Teams of 3–4 plan milestones, distribute tasks (data, modeling, firmware), integrate, and iterate based on profiling and user feedback.

(7) An ability to acquire and apply new knowledge as needed, using appropriate learning strategies. (H) Students learn evolving toolchains (TFLite Micro operator support, Arduino updates, SDK changes) via documentation and experimentation; they adapt workflows when encountering operator gaps or memory issues.

Prepared By: Radha Poovendran

Last revised: 10/19/2025