AI & ML Workshop

Other ML Algorithms

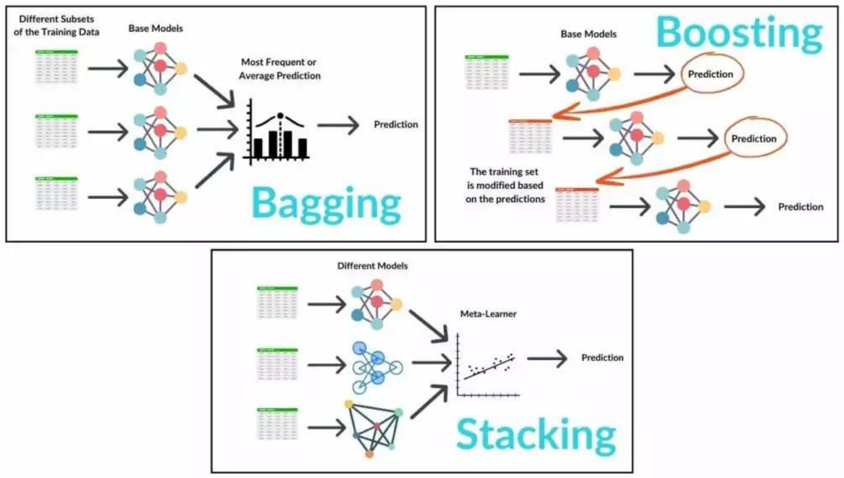
Muthukumar

Ensemble Learning

Ensemble Learning combines multiple models (called base learners) to improve prediction accuracy.

The idea is that a group of models performs better than any single model alone.

- Advantage:
  - Reduces variance, bias, and overfitting.
  - Improves generalization and robustness.
  - Especially useful when individual models are weak but diverse.
- Methods:
  - Bagging (Bootstrap Aggregating)
  - Boosting
  - Stacking


Different Subsets of the Training Data
Base Models
Most Frequent or Average Prediction
Prediction
Bagging
Base Models
Boosting
Prediction
Prediction
The training set is modified based on the predictions
Prediction
Different Models
Meta-Learner
Prediction
Stacking

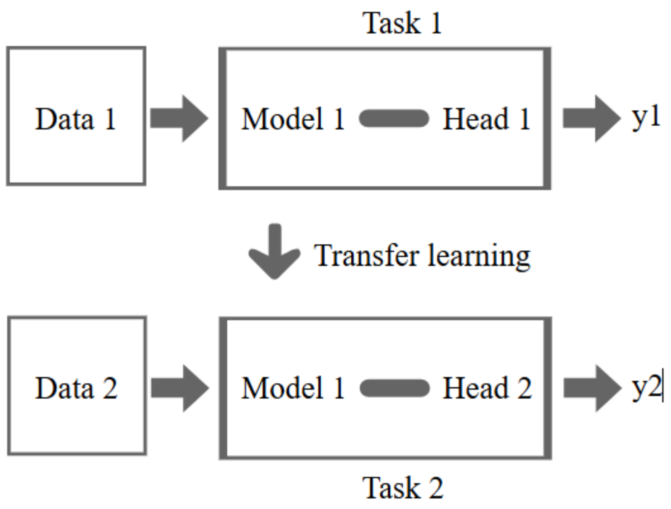
Transfer Learning

Transfer Learning is a technique where a model trained on one task or dataset is reused or adapted to a different but related task.

Often used to leverage knowledge from large datasets when you have limited labeled data in your target task.

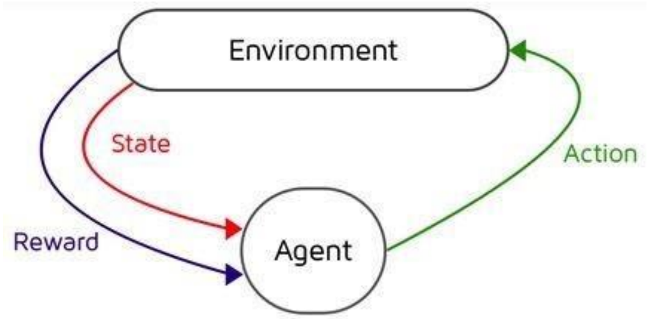
Advantages

- Reduces training time and computational cost.
- Improves accuracy on tasks with limited data.
- Enables faster model convergence and better generalization.



Reinforcement Machine Learning

Reinforcement learning is a type of machine learning where an agent learns to take actions in an environment in order to maximize a reward.



Highlights:

- Machine trained to make specific decisions
- Machine interacting with its environment
- Trial and error
- Reward system: providing feedback when an artificial intelligence agent performs the best action in a particular situation
- Sequence of successful outcomes is reinforced to develop the best solution for a given problem.

Tools:

- OpenAI Gym - Toolkit for developing and comparing reinforcement learning algorithms. Provides a standardized set of environments.
- TensorFlow Agents - RL library for TensorFlow with implementations of many algorithms. Integrates well with other TF tools.
- PyTorch RL - Reinforcement learning framework built on PyTorch.

Federated learning

- Allows a model to be trained across decentralized devices holding local data samples, without exchanging them.
  - Vertical: models to be trained across complementary features without sharing the raw data.
  - Horizontal: enables time or geographic learning collaboration without sharing.
  - Transfer: knowledge gained in one location or/and domain to improve learning in another.
  - Cross-Silo: collaboration while keeping data within boundaries.
  - Hybrid: combine federated learning with other techniques.
- Tools:
  - TensorFlow Federated (TFF): open-source framework specifically designed for federated learning
  - PySyft: library for encrypted, privacy-preserving machine learning.

Applications of different MLs

ML Paradigm	Example	Domain	Sensor Input	Embedded/TinyML Use Case
Ensemble Learning	Adaptive Random Forest (ARF) for vibration anomaly detection	Advanced Manufacturing / IIoT	IMU + vibration sensors on CNC machines	Multiple decision trees run in parallel on edge device; ensemble reduces false positives and improves robustness of predictive maintenance in noisy industrial environments
Transfer Learning	Pre-trained HAR CNN fine-tuned on-device for personalized activity recognition	HAR (Wearables)	IMU (accel + gyro)	Start with a CNN trained on WISDM/UCI HAR dataset; fine-tune last layers on user's local IMU data → deployed to STM32/Edge MCU for personalized recognition
Reinforcement Learning	Q-learning for adaptive UAV flight stabilization	UAV Flight Control	IMU (roll, pitch, yaw rates)	UAV onboard MCU uses IMU feedback as state; Q-learning agent adjusts control gains in real time for turbulence rejection, reducing reliance on fixed PID parameters
Federated Learning	Collaborative ESC fault detection across factory devices	ESC in IIoT	ESC current + IMU vibration from motors	Multiple ESC controllers at different sites locally train small neural nets on fault/no-fault data, then share only model updates (not raw data); federated aggregation improves global ESC diagnostic accuracy while preserving data privacy

Streaming data analytics

process of analyzing and extracting insights from continuously flowing data in real-time or near-real-time.

Address:

- **General:** collected and processed in fixed intervals, efficient storage, and provide timely insights.
- **Latency:** Balancing the need for low latency with the complexity of analytics.
- **Scalability:** Handling large volumes of streaming data while maintaining performance.
- **Data Quality:** Ensuring the accuracy and reliability of data in real-time.
- **Integration with Batch Processing:** Coordinating seamlessly with batch processing systems for comprehensive data analysis.
- **Complex Event Processing:** Handling and processing complex events and patterns in streaming data.

Tools:

- **Apache Flink:** A distributed stream processing framework for big data processing and analytics.
- **Apache Kafka:** A distributed streaming platform that can be used for building real-time data pipelines and streaming applications.
- **Apache Storm:** An open-source stream processing system for processing data in real-time.
- **Spark Streaming:** A micro-batch processing library integrated with Apache Spark for real-time analytics.

Online Machine Learning

training and updating of machine learning models continuously for real-time data.

Addresses:

- **General:** Dynamic Adaptation, Efficient Resource Utilization, Scalability, Timely model updates, accuracy, and decision.
- **Concept Drift:** Adapting to changes in data distribution over time, known as concept drift, poses a challenge for maintaining model accuracy.
- **Data Quality and Noise:** Handling noisy and unreliable data in real-time environments.
- **Model Complexity:** Balancing the complexity of models with the need for real-time updates.
- **Resource Management:** Efficiently managing computational resources for continuous learning.
- **Security Concerns:** Ensuring the security of online ML models and preventing vulnerabilities.

Tools:

- **TensorFlow Extended (TFX):** Google's end-to-end platform for deploying production-ready machine learning models.
- **Apache Flink ML:** A stream processing framework that supports real-time machine learning.
- **Scikit-Multiflow:** An extension of scikit-learn for online machine learning on streaming data.
- **MLlib Streaming:** Part of Apache Spark, it supports online machine learning on large-scale distributed data.

Streaming & Online ML

- Streaming data analytics:
  - analyze and extract insights from data streams in real-time.
  - analyzing streaming data by detecting patterns, simple processing (aggregations, filtering), and complex event processing.
- Online ML:
  - continuously update and improve machine learning models as new data becomes available.
  - continuous model training and updating, adapting to changes in the underlying data distribution, feature updates, and hyperparameter tuning.
- integration of online ML with streaming analytics allows for dynamic model updates and predictions in real-time.
  - **Health Monitoring in IoT Devices:**
    - **Use Case:** Real-time health monitoring of patients through wearable devices.
    - **Integration:** Streaming analytics processes continuous health data streams, while online ML models adapt to changes in individual health conditions over time.
    - **Frameworks:** Apache Kafka, Apache Flink or Apache Storm for streaming analytics, Edge Impulse for online ML on edge devices.

Streaming & Online ML

Aspect	**Streaming Data Analytics**	**Online ML**
Input	Raw IMU stream (50 Hz)	Raw IMU stream (50 Hz)
Processing	Fixed thresholds, RMS, FFT, orientation checks	Incremental model update (logistic regression / adaptive forest)
Adaptation	None — one-size-fits-all	Personalized per user
Deployment	Low compute, very efficient	Higher compute, memory needed
Outcome	Alerts when thresholds are exceeded	Alerts improve over time (fewer false positives, better personalization)

AutoML

automated processes and tools to develop and accelerate the machine learning model development.

Addresses:

- **Data Preprocessing:** Handling missing data, encoding categorical variables, and scaling features.
- **Feature Engineering:** Automatically generating relevant features from raw data.
- **Model Selection:** Identifying the most suitable algorithm for a given dataset.
- **Hyperparameter Tuning:** Optimizing the hyperparameters of machine learning models.
- **Ensemble Methods:** Combining multiple models to improve performance.

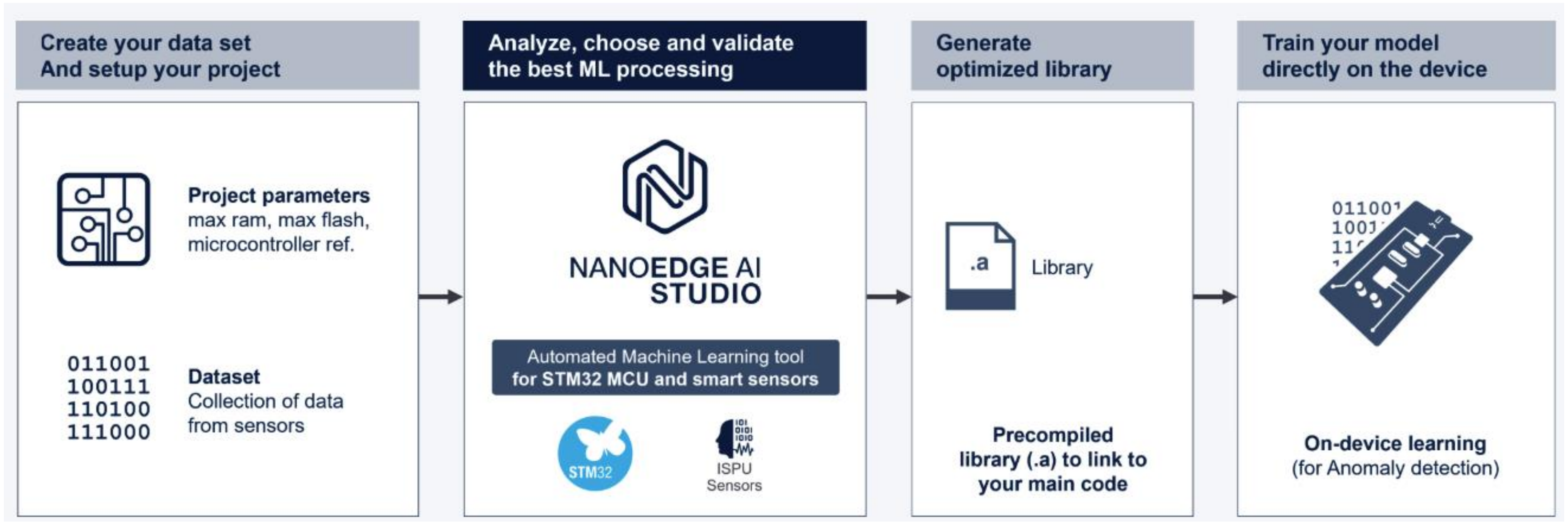
Tools

- **Auto-Sklearn:** An automated machine learning toolkit based on scikit-learn.
- **Google AutoML:** Provides various AutoML services for different tasks like image classification, natural language processing, and tabular data.

AutoML in ST

STM NanoEdge AI Studio is a tool that simplifies the creation of optimized machine learning (ML) models for STM32 microcontrollers.

It automates the process of building ML libraries by handling data logging, model training, and code generation with minimal user input.


Create your data set
And setup your project
Project parameters
max ram, max flash, microcontroller ref.
011001
100111
110100
111000
Dataset
Collection of data from sensors
Analyze, choose and validate the best ML processing
NANOEDGE AI STUDIO
Automated Machine Learning tool for STM32 MCU and smart sensors
STM32
ISPU Sensors
Generate optimized library
.a
Library
Precompiled library (.a) to link to your main code
Train your model directly on the device
On-device learning (for Anomaly detection)

TinyML

Tiny Machine Learning is the deployment of machine learning models on resource-constrained edge devices.

specialized lightweight models are used in the TinyML.

- Linear Models
- Decision Trees
- Random Forests
- k-Nearest Neighbors (k-NN)
- Support Vector Machines (SVM)
- Neural Networks (Tiny Neural Networks)
- Spiking Neural Networks (SNN)
- Gaussian Mixture Models (GMM)

Tools/Framework:

- TensorFlow Lite for Microcontrollers, Edge Impulse, Arm CMSIS-NN, and uTensor.

ML Algorithms for TinyML / Embedded ML

- Linear Models - Linear Regression, Logistic Regression, Linear SVM
- Decision Trees and Random Forests (Shallow)
- k-Nearest Neighbors (with small k and dataset)
- Support Vector Machines (with Linear or Polynomial Kernel)
- Small Neural Networks (Tiny DNNs)
  - 1D/2D CNNs
  - MLPs (Dense networks): For classification/regression.
  - RNNs/GRUs: For time series, sensor data

Frameworks & Applications

- Frameworks Supporting TinyML
  - TensorFlow Lite for Microcontrollers
  - CMSIS-NN (for ARM Cortex-M cores)
  - Edge Impulse Studio
  - uTensor, TVM, microTVM
  - Arduino ML

- Sample Applications
  - Wake-word detection
  - Gesture recognition (e.g., IMU-based)
  - Vibration monitoring
  - Air quality estimation
  - Predictive maintenance on industrial sensors