Technologie internetu rzeczy
Analiza projektów dotyczących
TinyML

Marcin Mikuła
Tomasz Bochnak
Piotr Cygan
Jakub Łubkowski

Kryteria porównawcze projektów

1. Model uczenia
2. Umiejętność detekcji anomalii
3. Compatible hardware
4. Uczenie modelu

Porównanie projektów

Wyjaśnienia kryteriów porównawczych

1. **Model uczenia**
   Model uczenia maszynowego jest algorytmem, który po zapoznaniu się z danymi treningowymi jest w stanie przewidywać rozwiązania dla nowych danych. Modele uczenia maszynowego dzielą się na trzy główne kategorie: uczenie nadzorowane, uczenie nienadzorowane i uczenie wspomagane.
   - *Uczenie nadzorowane polega na nauczeniu modelu przewidywania etykiety dla danych wejściowych na podstawie danych treningowych z etykietami. Przykładami tego typu uczenia są klasyfikacja i regresja.*
   - *Uczenie nienadzorowane polega na nauczeniu modelu wykrywania struktur lub wzorców w danych bez użycia etykiet. Przykładami tego typu uczenia są klasteryzacja i redukcja wymiarów.*
   - *Uczenie wspomagane polega na wykorzystaniu człowieka w procesie uczenia, np. do oznaczania danych lub do interpretacji wyników.*

   Istnieje również wiele różnych metod i algorytmów, które mogą być używane do tworzenia modeli uczenia maszynowego, takich jak regresja liniowa, drzewa decyzyjne, sieci neuronowe, itp.

2. **Umiejętność detekcji anomalii**
   Wykrywanie anomalii rozumiane jest jako identyfikacja zdarzeń, które znacznie odbiegają od stanu normalnego. Zastosowanie wykrywania anomalii obejmuje kilka obszarów, w tym monitorowanie maszyn przemysłowych, transakcje finansowe, wykrycie oszustwa lub nieprawidłowe działanie urządzeń.

3. **Compatible hardware**
   Compatible hardware to sprzęt, który jest zgodny z innymi elementami systemu lub oprogramowaniem. Oznacza to, że może on być łatwo podłączony i współpracować z innymi urządzeniami lub oprogramowaniem, bez potrzeby dodatkowych modyfikacji lub specjalnego oprogramowania.
   Przykładami kompatybilnego sprzętu są dyski twarde USB, które mogą być podłączone do różnych komputerów, bez potrzeby instalowania specjalnego oprogramowania, lub karty graficzne, które są kompatybilne z płytami głównymi komputerów.
   Kompatybilność sprzętu jest ważna, ponieważ pozwala na łatwe i sprawne korzystanie z różnych urządzeń i oprogramowania, bez konieczności specjalistycznej wiedzy czy dodatkowych kosztów.

4. **Uczenie modelu**

Uczenie modelu to proces tworzenia i optymalizowania modelu uczenia maszynowego na podstawie danych treningowych. Celem tego procesu jest stworzenie modelu, który będzie mógł dokonywać skutecznych i trafnych przewidywań na podstawie nowych danych. Proces uczenia modelu jest iteracyjny, oznacza to, że może być on powtarzany wiele razy, aż do osiągnięcia zadowalających rezultatów.

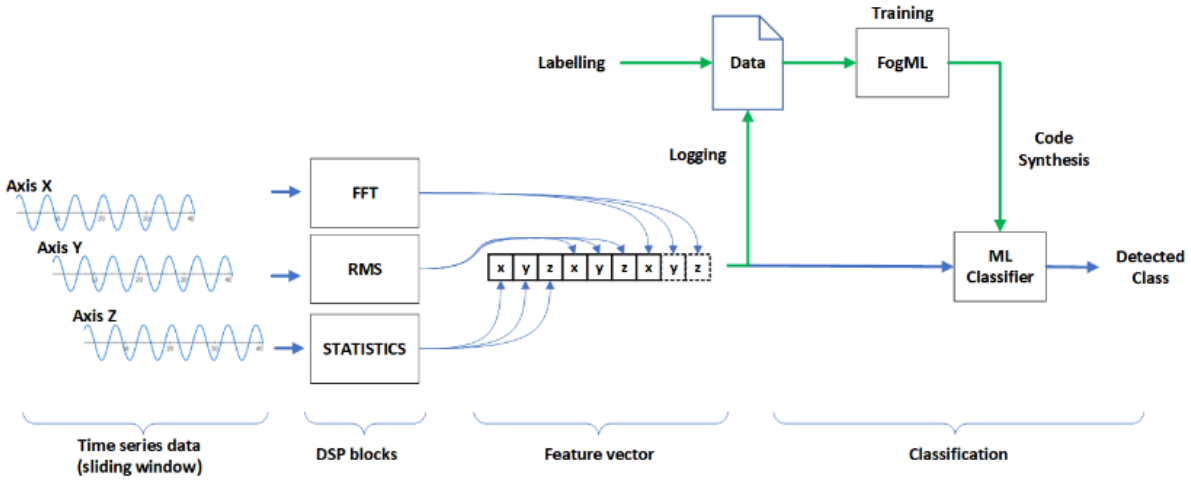
Nazwa projektu	FogML	DeepC	Emlearn	TinyML-CAM	sklearn-porter	Arm NN	CurrentSense TinyML	Edge Impulse
Model uczenia	sieci neuronowe	sieci neuronowe	drzewa decyzyjne, sieci neuronowe	sieci neuronowe	sieci neuronowe	sieci neuronowe	sieci neuronowe	sieci neuronowe
Umiejętność detekcji anomalii	tak	tak	tak	tak	tak	nie	nie	tak
Compatible hardware	tak	tak	tak	tak	tak	tak	tak	tak
Możliwe douczanie modelu	tak	tak	nie	tak	nie	nie	nie	tak

Analiza projektów

FogML

Model uczenia

Do rozwiązania problemu klasyfikacji FogML wykorzystuje algorytmy uczenia nadzorowanego takie jak: MLP, drzewa decyzyjne, sieci Bayesa i SVM, a do detekcji anomalii algorytm LOF.

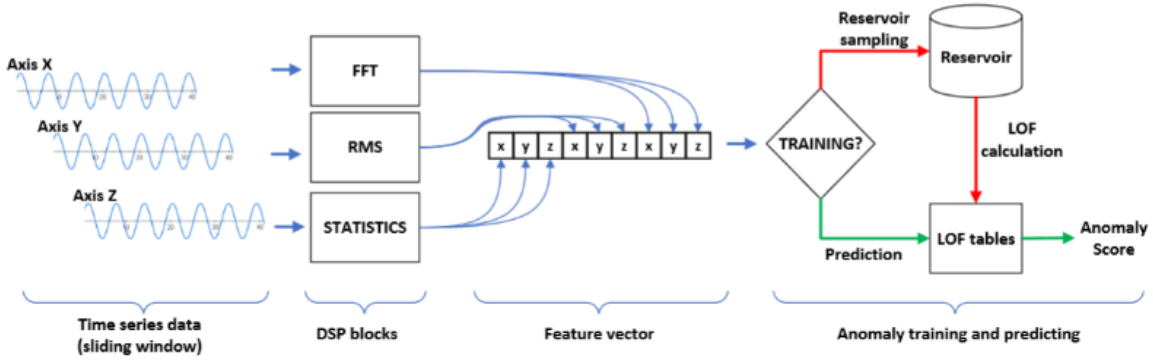


Rys 1. Schemat działania FogML dla klasyfikacji modeli

Detekcja anomalii

Urządzenia korzystające z FogML mogą wykrywać anomalie, ponieważ narzędzie wykorzystuje **algorytm LOF** w następujący sposób:
Faza wykrywania anomalii algorytmu LOF – nowe dane pomiarowe nie aktualizują zawartości zbiornika ani tablic gęstości, ale są analizowane jako tzw. detekcja nowości. Gęstość wokół nowego punktu jest obliczana przy założeniu, że zawartość zbiornika to jego sąsiedztwo. Na tej podstawie obliczany jest wynik anomalii.

Przełączanie między fazami uczenia i wykrywania anomalii zależy od sposobu użytkowania urządzenia i jego strategii działania



Rys 2. Schemat detekcji anomaliii

Compatible hardware

- Cortex M4 core
- Arm Cortex-M33

Uczenie modelu

W pierwszym etapie procesu model ML jest uczony przy użyciu pakietu scikit-learn wysokiego poziomu. Następnie na podstawie wyuczonego modelu dedykowany generator tworzy kod w C. Ostatecznie staje się integralną częścią oprogramowania sprzętowego urządzenia.

Faza uczenia algorytmu LOF – punkty danych w zbiorniku danych są przekazywane do algorytmu LOF, który oblicza tablice gęstości punktów.
Proces uczenia może być inicjowany przy starcie urządzenia lub cyklicznie poprzez aktualizację zasobnika zawartości.

Wnioski

Uczenie oraz douczanie modeli klasyfikacyjnych odbywa się poza urządzeniem, natomiast douczanie modeli detekcji anomalii może odbywać się na urządzeniu.

DeepC

Model uczenia

DeepC to nowa metoda nauki, która polega na uczeniu się za pomocą głębokiego uczenia. Jest to specjalizowana metoda, która jest często stosowana do rozwiązywania skomplikowanych problemów, takich jak rozpoznawanie obrazów, analiza danych i rozumienie języka naturalnego. DeepC to skuteczna metoda, która pozwala na automatyczne wykrywanie wzorców i trendów w danych, co jest niezwykle przydatne w wielu różnych dziedzinach.

Detekcja anomalii

Istnieje kilka różnych metod detekcji anomalii w DeepC:

a) **Autoenkodery**
Autoenkodery to modele uczenia głębokiego, które mogą być wykorzystane do kompresji i dekompresji danych. Można je wykorzystać do wykrywania anomalii poprzez porównywanie oryginalnych danych z ich dekompresją.

b) **Modele generatywne**
Modele generatywne, takie jak generatywny model sieciowy (GAN) lub model warunkowy Wasserstein GAN (CWGAN), mogą być wykorzystane do generowania danych, które są podobne do oryginalnych danych. Anomalie mogą zostać wykryte poprzez porównywanie oryginalnych danych z generowanymi danymi.

c) **Modele klasyfikacyjne**
Modele klasyfikacyjne mogą być wykorzystane do rozpoznawania anomalii poprzez przypisanie danych do różnych klas. Modele takie jak sieci neuronowe konwolucyjne (CNN) lub sieci recurrentne (RNN) mogą być specjalnie zaprojektowane do detekcji anomalii.

d) **Modele wykorzystujące metodę detekcji odstających (Outlier Detection)**
Modele tego typu są specjalnie zaprojektowane do wykrywania odstających punktów danych.

Compatible hardware

Arm
Armv7
Arm64
AMD64
ppc64le

Uczenie modelu

W przypadku nauki bezpośrednio na urządzeniu dane są zbierane z czujników urządzenia, są wstępnie przetwarzane i wykorzystywane do trenowania modelu uczenia maszynowego na samym urządzeniu. Takie podejście ma tę zaletę, że zmniejsza ilość danych, które należy przesłać do zdalnego serwera w celu szkolenia, zmniejszając ilość wymaganej komunikacji i zużycia energii. Model jest następnie optymalizowany pod kątem działania na ograniczonych zasobach urządzenia, takich jak pamięć i moc obliczeniowa.

Emlearn

Model uczenia

Emlearn to biblioteka Python do uczenia maszynowego inkrementalnego. Celem biblioteki jest zapewnienie prostej i elastycznej metody obsługi i trenowania modeli uczenia maszynowego inkrementalnie, umożliwiającej łatwe dostosowywanie się do zmieniających się strumieni danych i scenariuszy uczenia online.

Detekcja anomalii

Emlearn posiada kilka algorytmów detekcji anomalii, które mogą być wykorzystane do detekcji anomalii w danych ciągłych, takich jak szeregi czasowe lub dane transmisji. Przykładami takich algorytmów są:

a) **One-Class SVM**

jest to metoda polegająca na trenowaniu modelu SVM na danych normalnych i zastosowaniu go do wykrywania danych nienormalnych. Model SVM jest trenowany tylko na przykładach z normalnych danych i zakłada, że wszystkie przykłady poza zbiorem treningowym są nienormalne.

b) **Mahalanobis Distance**

ta metoda polega na określeniu odległości Mahalanobis między każdym przykładem a średnią i odchyleniem standardowym znormalizowanych danych treningowych. Przykłady, które są dalekie od średniej, są oznaczane jako nienormalne.

c) **Isolation Forest**

Jest to metoda polegająca na losowym wybieraniu cech i podziałach danych tak, aby izolować przykłady nienormalne.

Compatible hardware

- ESP8266
- AVR Atmega
- ARM Cortex M
- ESP32

Uczenie modelu

Uczenie oraz douczanie modeli klasyfikacyjnych, oraz detekcji anomalii odbywa się poza urządzeniem. Proces uczenia się modelu klasyfikacyjnego polega na szkoleniu modelu w celu poznania cech różnych klas, tak aby mógł dokładnie przewidywać klasę nowych danych. Zwykle odbywa się to za pomocą technik, takich jak nadzorowane uczenie się. W

uczeniu nadzorowanym model jest szkolony przy użyciu oznaczonych danych, w których klasy są jasno zdefiniowane. Model uczy się identyfikować funkcje, które są powiązane z każdą klasą, i używa tych funkcji do przewidywania.

Model wykrywania anomalii jest zwykle szkolony poza urządzeniem, przy użyciu zestawu danych zawierającego zarówno normalne, jak i nieprawidłowe przykłady. Model jest następnie wdrażany na urządzeniu osadzonym, gdzie można go wykorzystać do identyfikacji anomalii w nowych danych.

TinyML-CAM

Model uczenia

TinyML-CAM to biblioteka python, która została zaprojektowana, aby zapewnić prosty i elastyczny sposób na uruchomienie modeli uczenia głębokiego na urządzeniach typu edge, takich jak mikrokontrolery, mikroprocesory i urządzenia IoT. Używa ona techniki nazywanej Warunkową Mapą Uwagi (CAM), aby optymalizować modele uczenia głębokiego dla urządzeń typu edge. Pozwala na kompresję i optymalizację modeli uczenia głębokiego dla urządzeń typu edge za pomocą CAM. posiada gotowe modele do klasyfikacji obrazów i wykrywania obiektów. zapewnia prosty i elastyczny API do uruchamiania modeli na urządzeniach typu edge.

Detekcja anomalii

TinyML-CAM nie posiada wbudowanej obsługi detekcji anomalii

Compatible hardware

Biblioteka zapewnia wsparcie dla popularnych platform mikrokontrolerów, takich jak Arduino, ESP32 i STM32

Uczenie modelu

Proces uczenia modelu TinyML-cam zazwyczaj polega na trenowaniu modelu na mocnym komputerze za pomocą dużego zbioru etykietowanych obrazów. Następnie wytrenowany model jest wdrażany na urządzeniu kamerowym, gdzie może być używany do prognozowania na nowych obrazach.

uTensor

Model uczenia

uTensor to biblioteka uczenia maszynowego dla mikrokontrolerów i innych urządzeń o ograniczonych zasobach. Umożliwia ona na uruchomienie modeli uczenia głębokiego na urządzeniach takich jak mikrokontrolery, mikroprocesory czy urządzenia IoT. Biblioteka ta jest oparta na popularnym frameworku TensorFlow i umożliwia proste i elastyczne API do uruchamiania modeli na urządzeniach o ograniczonych zasobach.

Detekcja anomalii

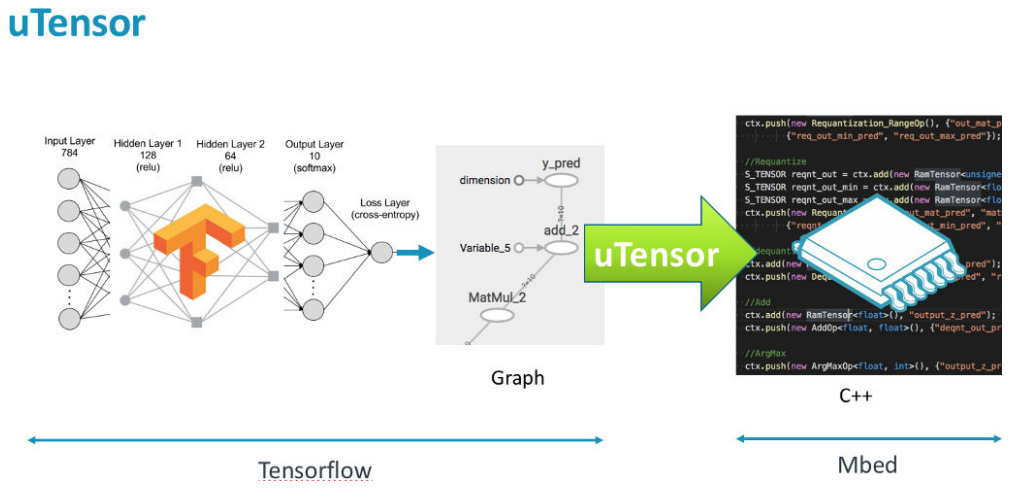
Nie posiada ona wbudowanej funkcjonalności do detekcji anomalii, jednak można użyć uTensor do uruchomienia modelu detekcji anomalii na urządzeniu z ograniczonymi zasobami. Można wytrenować model detekcji anomalii na komputerze z użyciem np. TensorFlow

Compatible hardware

- Arduino (AVR, SAM, SAMD)
- ESP32
- STM32
- Raspberry Pi (ARMv7, ARMv8)
- Nordic nRF5x (ARM Cortex M4)
- TI CC3220 Linux (x86, x64, ARM)
- Windows (x86, x64)
- MacOS (x64)

Uczenie modelu

uTensor jest projektowany do uruchamiania już wytrenowanych modeli uczenia maszynowego na urządzeniach o ograniczonych zasobach. Nie posiada on funkcjonalności do treningu lub dostosowywania modeli bezpośrednio na urządzeniu. Jego celem jest umożliwienie uruchamiania już wytrenowanych modeli na urządzeniach o ograniczonych zasobach, takich jak mikrokontrolery, mikroprocesory czy urządzenia IoT.



sklearn-porter

Model uczenia

sklearn-porter to biblioteka python, która pozwala na konwersję modeli uczenia maszynowego z pakietu scikit-learn do różnych języków programowania, takich jak C, Java, lub JavaScript. Dzięki temu możliwe jest wykorzystanie tych modeli na różnych platformach, takich jak urządzenia IoT, aplikacje mobilne czy serwery. Biblioteka sklearn-porter działa poprzez wykorzystanie specjalnie przygotowanego kodu, który składa się z klas i funkcji, które odpowiadają za działanie modelu, który chcemy przenieść na inny język programowania. Dzięki temu, że biblioteka sklearn-porter jest oparta na popularnym pakiecie scikit-learn, jest łatwa w użyciu i pozwala na szybką konwersję modeli.

Detekcja anomalii

Sklearn-porter nie posiada wbudowanej funkcjonalności do wykrywania anomalii

Compatible hardware

Sklearn-porter nie jest narzędziem przeznaczonym do działania na sprzęcie, jest to biblioteka językowa, która pozwala na konwersję modeli uczenia maszynowego z pakietu scikit-learn do innych języków programowania. Dzięki temu modele te mogą być uruchamiane na różnych platformach, takich jak urządzenia IoT, aplikacje mobilne lub serwery, jednak sklearn-porter nie jest przeznaczony do działania na sprzęcie, jego celem jest konwersja modeli.

Uczenie modelu

sklearn-porter nie jest narzędziem do treningu lub dostosowywania modeli uczenia maszynowego. Jest to biblioteka, która pozwala na konwersję modeli już wytrenowanych z pakietu scikit-learn do innych języków programowania, takich jak C, Java czy JavaScript. Nie jest to narzędzie przeznaczone do treningu modeli bezpośrednio na urządzeniu, jego celem jest konwersja modeli już istniejących.

Arm NN

Model uczenia

ARM NN (Neural Network) to lekki silnik inferencji sieci neuronowej dla procesorów Arm Cortex-A. Jest to biblioteka przeznaczona do uruchamiania modeli sieci neuronowych na procesorach Arm Cortex-A, zapewniając wysoką wydajność i niskie zużycie energii. ARM NN zawiera interfejs programowania aplikacji (API) umożliwiający tworzenie i uruchamianie modeli sieci neuronowych na tych procesorach. Obsługuje on różne frameworki takie jak TensorFlow, Caffe, ONNX itp.

Detekcja anomalii

ARM NN jest silnikiem inferencji sieci neuronowej dla procesorów Arm Cortex-A, nie posiada on wbudowanej funkcjonalności do detekcji anomalii, jednak można użyć ARM NN do uruchomienia wcześniej wytrenowanego modelu detekcji anomalii na tych procesorach.

Compatible hardware

ARM NN wspiera szeroką gamę platform, w tym, ale nie tylko: procesory ARM Cortex-A (takie jak Qualcomm Snapdragon, Samsung Exynos i Apple A-series) procesory ARM Cortex-M (takie jak te stosowane w mikrokontrolerach) procesory ARM Cortex-R (takie jak te stosowane w systemach czasu rzeczywistego) GPU i inne specjalistyczne akceleratory sprzętowe (takie jak Intel Movidius, Google Edge TPU) Linux, Windows, QNX, VxWorks, Android, iOS.

Uczenie modelu

ARM NN jest przeznaczony do uruchamiania już wytrenowanych modeli sieci neuronowych na procesorach Arm Cortex-A. Nie posiada on funkcjonalności do treningu lub dostosowywania modeli bezpośrednio na urządzeniu.

CurrentSense-TinyM

Model uczenia

CurrentSense-TinyML to biblioteka przeznaczona do uruchamiania modeli sieci neuronowych na urządzeniach z ograniczonymi zasobami, takich jak mikrokontrolery. Biblioteka ta zawiera algorytmy do analizy prądu i detekcji anomalii w prądzie, co pozwala na zwiększenie bezpieczeństwa urządzeń IoT poprzez wykrywanie nieautoryzowanego dostępu lub innych nieprawidłowych działań. CurrentSense-TinyML jest oparty na TensorFlow Lite Micro

Detekcja anomalii

Biblioteka CurrentSense-TinyML stosuje algorytm uczenia maszynowego do wykrywania anomalii. Algorytm ten analizuje pobór prądu i wykrywa nieprawidłowości w jego zużyciu, które mogą wskazywać na nieautoryzowany dostęp lub inne nieprawidłowe działanie. Algorytm uczenia maszynowego jest wytrenowany na podstawie danych historycznych poboru prądu, dzięki czemu jest w stanie rozpoznawać anomalie. Konkretnie, CurrentSense-TinyML używa algorytmów detekcji outlierów, takich jak algorytm LOF (Local Outlier Factor), który pozwala na wykrywanie punktów, które są odstające od pozostałych.

Compatible hardware

- STM32F407
- STM32L476
- nRF52840

Uczenie modelu

Biblioteka CurrentSense-TinyML jest przeznaczona do uruchamiania już wytrenowanych modeli na mikrokontrolerach z ograniczonymi zasobami. Nie posiada ona funkcjonalności do treningu lub dostosowywania modeli bezpośrednio na urządzeniu. Jego celem jest umożliwienie uruchamiania już wytrenowanych modeli na mikrokontrolerach, zapewniając wysoką wydajność i niskie zużycie energii

Edge Impulse

Model uczenia

Edge Impulse to platforma do tworzenia modeli uczenia maszynowego dla urządzeń edge (krawędziowych). Platforma ta pozwala na łatwe połączenie danych z sensora, przetwarzanie ich i szybkie tworzenie i dostosowywanie modeli uczenia maszynowego

Detekcja anomalii

W Edge Impulse istnieją dwa główne rodzaje detekcji anomalii:

- Detekcja punktu anomalii: polega na wykrywaniu pojedynczych, nieoczekiwanych wartości w danych. Jest to przydatne, gdy chcesz wykryć pojedyncze incydenty, takie jak awaria maszyny lub nietypowy ruch.

- Detekcja zmiany trendu: polega na wykrywaniu zmian w trendzie danych. Jest to przydatne, gdy chcemy wykryć długotrwałe zmiany w danych, takie jak zmiana wydajności maszyny lub zmiana warunków środowiskowych.

Edge Impulse umożliwia również ich kombinację oraz konfigurację parametrów takich jak progi detekcji czy czas potrzebny na reakcję.

Compatible hardware

- Arduino,
- ESP32,
- STM32, Raspberry Pi,
- ATmega
- ATtiny,
- MSP430,
- MSP432,
- SAM3X,
- SAM3N

Uczenie modelu

W Edge Impulse, model wykrywania anomalii jest zwykle trenowany poza urządzeniem za pomocą podejścia uczenia nadzorowanego, gdzie model jest trenowany na zbiorze danych z przykładami zarówno normalnego, jak i błędnego zachowania. Podobnie, klasyfikacja również jest trenowana za pomocą metody uczenia nadzorowanego poza urządzeniem, gdzie model jest uczony na zbiorze danych z przykładami różnych klas. Platforma zapewnia narzędzia do zbierania danych, przetwarzania i trenowania modelu, które mogą być używane do trenowania modelu klasyfikacji. Edge Impulse umożliwia również uczenie na urządzeniu poprzez udostępnienie biblioteki o nazwie Edge Impulse Inferencing Library (EIIL), która może być zintegrowana z firmware urządzenia krawędziowego. Biblioteka zapewnia niezbędną funkcjonalność do zbierania danych, przetwarzania i trenowania modelu, a także do wdrażania wytrenowanego modelu na urządzeniu krawędziowym.

Podsumowanie

TinyML to dziedzina, która zajmuje się zastosowaniem uczenia maszynowego w urządzeniach o niskim zużyciu energii, takich jak urządzenia IoT, sensory, układy zintegrowane itp. Celem TinyML jest umożliwienie tym urządzeniom wykonywanie zaawansowanych obliczeń i przewidywań na miejscu, bez konieczności przesyłania danych do zewnętrznego centrum obliczeniowego.

Plusy TinyML w porównaniu do ML (Machine Learning):

- Wymagają mniej zasobów sprzętowych i energii, co oznacza, że mogą działać na małych, niedrogich urządzeniach, takich jak sensory czy układy zintegrowane.
- Szybkie reakcje dzięki lokalnym obliczeniom na urządzeniu.
- Możliwość pracy w trudnych warunkach, takich jak brak dostępu do sieci czy niskie możliwości obliczeniowe.
- Możliwość zbierania i przetwarzania danych bezpośrednio na miejscu, co zwiększa prywatność i bezpieczeństwo danych.

Minusy TinyML w porównaniu do ML:

- Ograniczona moc obliczeniowa w porównaniu do tradycyjnych systemów ML.
- Trudniejsze tworzenie i utrzymanie modeli, ponieważ muszą one być bardziej zoptymalizowane pod kątem rozmiarów i zasobów.
- Mniejsza dokładność w porównaniu do modeli uczenia maszynowego trenowanych na dużych zbiorach danych.
- Ograniczona skalowalność w przypadku rosnącego zapotrzebowania na moc obliczeniową.