DEZEMBER 2025 - JANUAR 2026 | ELEKTORMAGAZINE.DE

€ 9,95
CHF 18,00

GRATIS dank MOUSER ELECTRONICS

elektor MEDIA & LEARNING

DIGITALE BONUS AUSGABE

614B

SEIT 1961

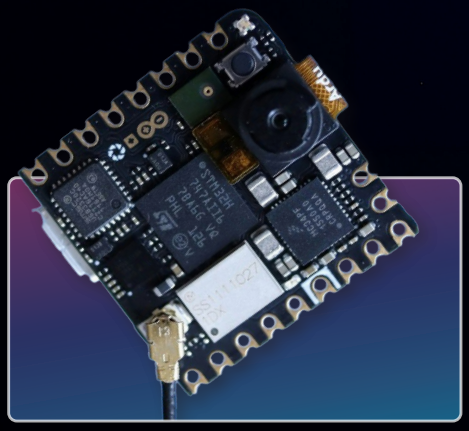
Artikel für Profis, Maker und Studierende

Gast-Ausgabe von

EDGE IMPULSE
a Qualcomm company

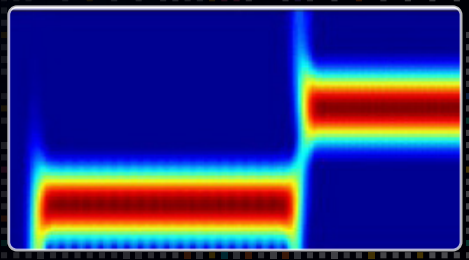
EDGE-KI FÜR ALLE

Smarter, schneller, sicherer.



Analoge Messgeräte mit KI ablesen Automatisierung klassischer Anzeigen mit Arduino

Flüssigkeitserkennung mit KI
Sensorfusion auf einem Seeed WIO Terminal



DSP und Edge-KI
Kleinere Modelle durch Signalverarbeitung

Treten Sie jetzt der Elektor-Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Zugang zum kompletten Online-Archiv (1970 - heute)!
- 8 x Elektor Magazin (gedruckt)
- 8 x Elektor Magazin (PDF)
- 10 % Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

  u. v. m. aus der Projektplattform Elektor Labs

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zum kompletten Online-Archiv
- 8 x Elektor Magazin (PDF)
- 10 % Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

  u. v. m. aus der Projektplattform Elektor Labs

www.elektormagazine.de/abo

elektor MEDIA & LEARNING

56. Jahrgang, Nr. 614B
Gast-Ausgabe 2025
ISSN 0932-5468

EDGE IMPULSE

Diese Ausgabe des Elektor-Magazins wurde in Zusammenarbeit mit Edge Impulse produziert und wird von Edge Impulse gesponsert.

Das Elektor Magazin wird 8 Mal im Jahr herausgegeben von
Elektor Verlag GmbH
Lukasstraße 1, 52070 Aachen (Deutschland)
Tel. +49 (0)241 95509190
www.elektor.de | www.elektormagazine.de

Content Director: C. J. Abate
Chefredakteur: Jens Nickel

Für alle Ihre Fragen
service@elektor.de

Mitglied werden
www.elektormagazine.de/abo

Abonnementspreise: ab dem 1. Dezember 2025 kostet die Print-Mitgliedschaft (Gold) 144,95 € pro Jahr für ein 1-Jahres-Abonnement.
Die digitale Mitgliedschaft (Green) beträgt 99,95 € für ein 1-Jahres-Abonnement und 169,95 € für ein 2-Jahres-Abonnement.

Anzeigen
Büsra Kas
Tel. +49 (0)241 95509178
busra.kas@elektor.com
www.elektormagazine.de/mediadaten

Druck
Senefelder Misset, Mercuriusstraat 35
7006 RK Doetinchem (Niederlande)

Distribution
IPS Pressevertrieb GmbH, Carl-Zeiss-Straße 5
53340 Meckenheim (Deutschland)
Tel. +49 (0)2225 88010

Bonus-Ausgabe: Mehr Intelligenz auf die Geräte!

C. J. Abate
(Content Director, Elektor)

Die kreative Zusammenarbeit zwischen Elektor und Edge Impulse geht weiter! Nach dem erfolgreichen Start der von Edge Impulse als Gastredaktion herausgegebenen Ausgabe 2025 von Elektor freuen wir uns, Ihnen noch mehr praxisnahe Projekte, praktische Einblicke und Know-how zu maschinellem Lernen zu bieten, damit Sie auch im neuen Jahr inspiriert bleiben.

Diese Bonus-Ausgabe von Elektor ist ein großartiger Start in ein weiteres Jahr der technischen Innovation! Egal, ob Sie ein professioneller Entwickler sind, der Embedded-ML für eine neue Industrieanwendung erforscht, oder ein neugieriger Maker, der mit Edge-KI-Technologie experimentieren möchte – diese Bonus-Ausgabe wird Ihnen viele Anregungen für Ihre Kreativität bieten. Im Magazin finden Sie Artikel zu einer breiten Auswahl an Edge-Impulse-Themen und Projekten: Echtzeit-Objekterkennung, Bildklassifizierung, Edge-KI im Operationssaal, Flüssigkeitsklassifizierung mit KI und vieles mehr!

Während Sie die Projekte und Beiträge in diesem Magazin entdecken, laden wir Sie ein, uns Ihre Gedanken unter *elektormagazine.com/labs* und *forum.edgeimpulse.com* mitzuteilen. Wir freuen uns auf Ihr Feedback und sind gespannt, was Sie entwickeln. Viel Spaß beim Lesen!

INHALT

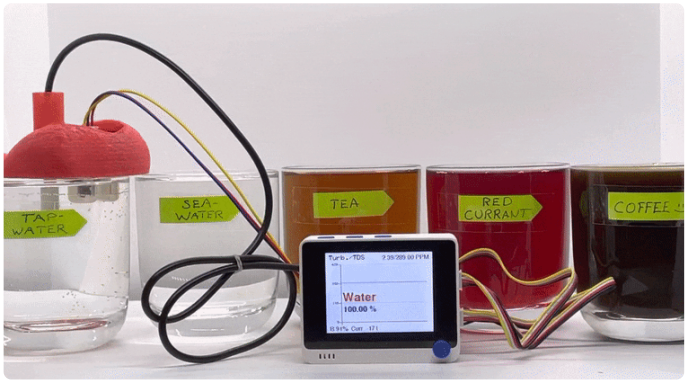
56. Jahrgang – Nr. 614B
Dez. 2025 / Jan. 2026

Gastredaktion: Edge Impulse

Die gedruckte Edge-Impulse-Gastausgabe von Elektor ist am Kiosk und im Elektor-Store erhältlich.

Flüssigkeits-erkennung mit KI

Sensorfusion auf einem Seeed WIO Terminal



Von Thomas Vikstrom (Finnland)

Dieses Projekt untersucht die Simulation menschlicher Geschmacksknospen mithilfe von TinyML und kostengünstigen Wasserqualitätssensoren auf dem Seeed Studio WIO Terminal. Es zeigt, wie Sensordaten zu gelösten Feststoffen und Trübung in Edge Impulse verarbeitet und trainiert werden können, um verschiedene Flüssigkeiten zu klassifizieren. Obwohl das Konzept vor allem experimentell ist, zeigt es Potenzial für Anwendungen in der Geschmacks- und Geruchstherapie, bei Weintests und in der Überwachung der Wasserqualität.

Wie viele andere wurde ich von Benjamin Cabés erstaunlicher künstlicher Nase [1] inspiriert, die verschiedene Gerüche erkennen kann. Ich dachte, es wäre interessant herauszufinden, ob es auch möglich ist, die Geschmacksknospen einer menschlichen Zunge zu simulieren, um verschiedene Arten von mehr oder weniger trinkbaren Flüssigkeiten zu erkennen. Dieses Tutorial zeigt, wie kostengünstige Wasserqualitätssensoren zusammen mit dem WIO Terminal von Seeed Studio erfolgreich fünf verschiedene Flüssigkeiten mithilfe von tinyML (tiny machine learning) erkennen konnten.

Use-Case-Erklärung

Dieses Projekt gehört hauptsächlich zur Grundlagenforschung, daher habe ich kein konkretes Problem zu lösen. Es sind jedoch viele Ideen entstanden, wie die Erkenntnisse aus diesem Projekt weiter genutzt werden könnten: olfaktorisches (Geruchs- und Geschmacks-)Training für Patienten mit Geruchs- und Geschmacksverlust; Weinverkostung (dafür wäre wahrscheinlich ein zusätzlicher pH-Sensor nötig) sowie Wasserqualitätsprüfung (z. B. zur Abwasserbehandlung).

Leider leiden viele Menschen unter teilweisem oder vollständigem Geruchsverlust (Anosmie) und Geschmacksverlust (Ageusie). Die Gründe für diesen Verlust sind unterschiedlich, aber insbesondere in den letzten Jahren wurde COVID-19 als eine wesentliche Ursache erkannt. Während die meisten Menschen nach COVID-19 oder anderen Atemwegserkrankungen ihre Geruchs- und Geschmackssinne vollständig wiedererlangen, berichtet eine Studie, dass etwa 5 % der Patienten nach einer COVID-19-Erkrankung noch lange Zeit an einem anhaltenden Geruchs- und/oder Geschmacksverlust leiden.

Benötigte Komponenten

Erforderlich

Wio Terminal [4]
Grove-TDS-Sensor/Messgerät für Wasserqualität [2]
Grove-Trübungssensor [5]

Optional, aber empfohlen

Wio Terminal Chassis – Akku (650 mAh) [6]
3D-Drucker zum Drucken von Zunge und Gehäuse

Was ist TDS und warum ist das wichtig?

Total Dissolved Solids (TDS) ist ein Maß für die gesamte Menge an anorganischen und organischen Feststoffen, die im Wasser gelöst sind. Typischerweise gilt: Je höher der TDS-Wert, desto mehr Substanzen sind im Wasser gelöst. Daher kann ein hoher TDS-Wert darauf hinweisen, dass das Wasser mehr Schadstoffe enthält, die ein Gesundheitsrisiko darstellen können, wie in **Bild 1** gezeigt. [2]

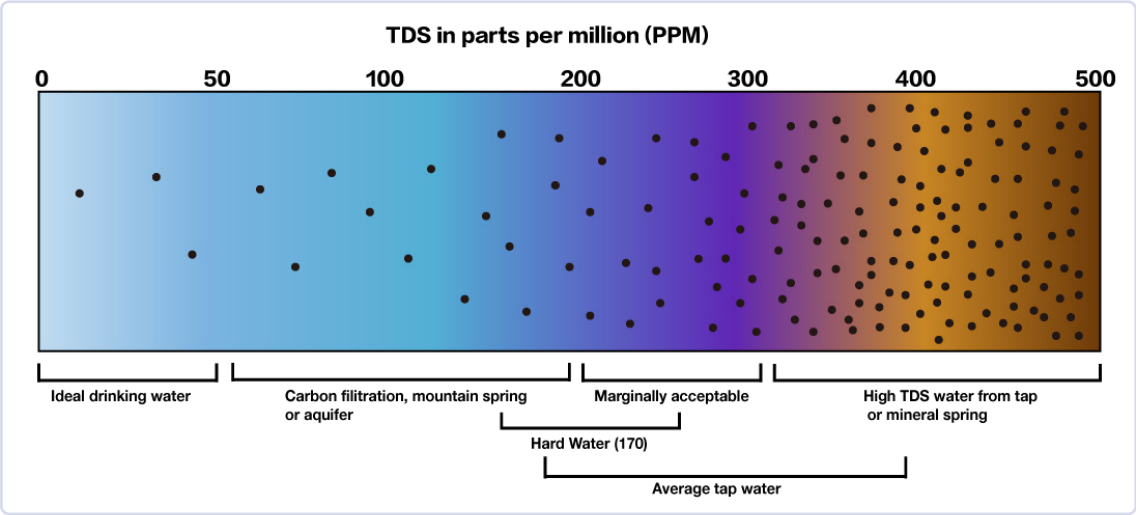


Bild 1. TDS-Konzept (Gesamtmenge gelöster Feststoffe) im Sensorabschnitt referenziert. (Quelle: Seeed Studio)

Was ist Trübung?

Der Grove-Trübungssensor kann die Trübung des Wassers (die Menge der enthaltenen Schwebstoffe) messen. Der optische Sensor dieses Moduls misst die Konzentration von Fremdstoffen im Wasser anhand der Streuung der Lichtwellen zwischen dem Phototransistor und der Diode. Mit einem optischen Transistor und optischen Dioden misst der Sensor die Menge des Lichts, das von der Lichtquelle zum Empfänger gelangt, um die Trübung des Wassers zu berechnen, wie in **Bild 2** gezeigt. [3]

3D-gedrucktes Gehäuse

Wenn Sie kein 3D-gedrucktes Gehäuse für das Board verwenden möchten, setzen Sie sie zumindest in separate Streichholzschachteln oder Ähnliches, um das Risiko zu minimieren, dass Flüssigkeiten darauf gelangen. Elektronik und Flüssigkeiten vertragen sich nicht!

Der Trübungssensor hat oben Löcher und ist daher nicht wasserdicht. Tauchen Sie ihn niemals vollständig in Flüssigkeiten ein. Durch eine 3D-gedruckte Zunge oder Ihr eigenes Sensorgehäuse können die Sensoren in der richtigen Tiefe gehalten werden, wodurch das Risiko verringert wird, dass der Trübungssensor durch eindringende Flüssigkeit beschädigt wird. Fast narrensicher wird das Ganze, wenn Sie zusätzlich Heißkleber oder Klebemasse (Blu Tack) auf die Löcher auftragen. Obwohl PETG- oder PLA-Filamente ebenfalls funktionieren, wird TPU-Filament wegen seiner Weichheit und Flexibilität empfohlen.

Zum Beispiel passen das Gehäuse und der Deckel für die Leiterplatten perfekt zusammen, da sie beim Zusammenfügen leicht zusammengedrückt werden können.

3D-Druck der Zunge und des Leiterplatten-Gehäuses

Ich habe zum ersten Mal TPU-Filament (von CCTree) verwendet und war überrascht, dass es mit meinem günstigen Bowden-Tube-3D-Drucker problemlos funktionierte. Ich habe mich nicht getraut, mit voller Geschwindigkeit (100 mm/s) zu drucken, und habe stattdessen eine moderate Geschwindigkeit von 50 mm/s bei 220 °C verwendet. Drucken Sie mit hoher Qualität – ich habe die maximale Qualität von 100 µm (0,1 mm) gewählt, wodurch der Drucker für jedes Teil stundenlang arbeitete. Alle STL-Dateien sind im GitHub-Repository [7] verfügbar.

Es ist keine Stützstruktur nötig; eine Brim- oder Raft-Schicht wird aber empfohlen. TPU-Filament haftet besser am Druckbett als PLA, daher ist eine beheizte Platte wahrscheinlich nicht notwendig; mein Drucker hat auch keine.

Die „Zunge“ ist in zwei Teile geteilt (**Bild 3**). Der untere Teil ist nicht unbedingt nötig, bietet aber etwas Schutz, wenn das Gerät nicht verwendet wird, und lässt es mehr wie eine echte Zunge aussehen! Wenn Sie mit sehr flexiblem TPU-Material drucken, möchten Sie den unteren Teil vielleicht etwas dicker machen als in den Designdateien vorgesehen.

Bild 2. 3D-gedruckte „Zungen“-Baugruppe zur Halterung der Sensoren in gleichmäßiger Tiefe.

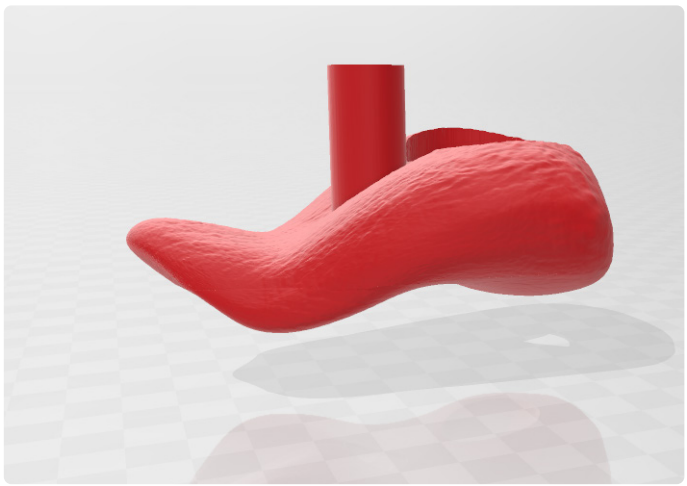


Bild 3. 3D-gedruckte Zunge und Gehäuseteile zur Positionierung und zum Schutz der Sensoren.

Bild 4. Leiterplatten in das 3D-gedruckte Gehäuse eingesetzt.

Hardware-Konfiguration

1. Setzen Sie die Leiterplatten zuerst in das 3D-gedruckte Gehäuse (**Bild 4**).
2. Schließen Sie das Kabel für den TDS-Sensor an Grove-Port A2 und das Kabel für den Trübungssensor an Grove-Port A4 am Batterie-Chassis an. Sie können natürlich auch jeden anderen analogen Port verwenden, solange Sie die entsprechenden Defines im Code ändern (`#define TDS_Pin A2` und `#define turbidity_Pin A4`). Wenn Sie keine Grove-Ports oder kein Batterie-Chassis verwenden, können Sie den 40-Pin-GPIO-Header auf der Rückseite verwenden.
3. Überprüfen Sie die Verbindungen (**Bild 5**).

Softwareinstallation

Um Daten sammeln und später das Setup praktisch testen zu können, müssen Sie das WIO Terminal vorbereiten.

1. Folgen Sie den Schritten in diesem Tutorial [8].
2. Fügen Sie die folgenden Bibliotheken über die Arduino-IDE hinzu mit *Sketch* -> *Include Library* -> *Add .ZIP-Library*:

- ei-tongue-arduino-1.0.2.zip [9]
- Wenn Sie das Batterie-Chassis nutzen und den Batteriestatus sehen möchten: *SparkFun BQ27441-G1A LiPo Fuel Gauge* Arduino Library [10]
- Besuchen Sie *Seeed_Arduino_Linechart* [11] und laden Sie das komplette Repo auf Ihr lokales Laufwerk herunter. Dann fügen Sie die .ZIP-Datei wie oben hinzu.

Bild 5. Grove-Ports mit dem WIO Terminal verbunden.

Daten-Erfassungsprozess

Um mit Edge Impulse Daten zu sammeln, sind nur wenige Schritte erforderlich.

Vorbereitungen

Besorgen Sie mindestens zwei Flüssigkeiten, z. B. Leitungswasser und Meerwasser. Verwenden Sie keine Flüssigkeiten, die die Sensoren beschädigen könnten, wie Öl, Säuren oder stark korrosive Flüssigkeiten. (Nachdem Sie die Sensoren in eine Flüssigkeit getaucht haben, spülen Sie sie gründlich mit Wasser und trocknen Sie sie ab.)
Kompilieren und laden Sie das Hauptprogramm *Tongue.ino* [12] auf Ihr WIO Terminal. Das Programm wird sowohl zum Datensammeln als auch zum späteren Inferencing verwendet. Es schreibt Sensordaten in das Terminal, die im nächsten Schritt vom Daten-Forwarder gesammelt werden.
Verwenden Sie den Edge-Impulse-Daten-Forwarder [13] und erzwingen Sie eine Frequenz von 5 Hz mit:

```
edge-impulse-data-forwarder --frequency 5
```

Wenn Sie folgende Meldung sehen

```
2 sensor axes detected (example values: [3,1.79]). What
do you want to call them? Separate the names with ',':,
```

geben Sie `TDS, Turbidity` ein, da dies die Sensornamen sind, die im Programm verwendet werden. Weitere Informationen zur Funktionsweise des Daten-Forwarders finden Sie unter [13].

Daten erfassen

1. Gehen Sie in Edge Impulse, erstellen Sie ein Projekt, falls noch nicht geschehen, und gehen Sie zu *Data acquisition*.
2. Ihr Gerät sollte nun auf der rechten Seite sichtbar sein (**Bild 6**).
3. Verwenden Sie eine Sample-Länge von 10 000 ms (= 10 s); die Frequenz ist automatisch 5 Hz, wie Sie es in den Vorbereitungsschritten festgelegt haben.

4. Das erste Label, für das Sie Daten aufzeichnen sollten, ist *Luft*. Tauchen Sie die Sensoren in keine Flüssigkeit, sondern halten Sie sie einfach in die Luft.
   Es ist oft nötig, ein Label zu haben, das als *Andere* oder *Hintergrund* dient. In diesem Fall ist das Label *air* sogar im Programm „hart-codiert", aber das Programm funktioniert technisch auch, wenn Sie dies nicht als Label verwenden.
5. Klicken Sie auf *Start sampling*, um Daten zu erfassen. Ich habe ungefähr 6 Minuten Daten pro Label gesammelt, aber ich empfehle, mit einer Minute pro Label zu beginnen. Machine Learning ist meist ein iterativer Prozess, starten Sie daher klein und passen Sie bei Bedarf an.
6. Wenn Sie Daten von Flüssigkeiten sammeln, stellen Sie sicher, dass die Sensoren tief genug eingetaucht sind. Beachten Sie die Animation am Anfang dieses Online-Tutorials [14]; die Flüssigkeiten sind sehr nahe an den Glasrändern. Benennen Sie die Flüssigkeiten am besten mit einem kurzen Label, da sie später auf dem WIO Terminal angezeigt werden. Versuchen Sie, für jede Flüssigkeit ungefähr die gleiche Datenmenge zu sammeln.

Wichtig

Sie sollten einen Teil der Datensätze als Testdaten zurückhalten, die später verwendet werden. Je nachdem, wie Sie Daten sammeln, geschieht das möglicherweise nicht automatisch. Wenn das Kreisdiagramm oben 100 %/0 % als Train/Test-Split anzeigt, müssen Sie manuell eingreifen. Eine Möglichkeit ist, im *Dataset*-Bereich auf *Test* zu klicken und dort Daten zu sammeln, aber insbesondere beim ersten Mal ist es einfacher, *Dashboard* zu wählen, nach unten zu scrollen und dann auf *Perform train/test split* zu klicken. So wird ein idealer Split von 80 %/20 % erstellt.

Training und Modell-Erstellung

Nachdem Sie einige Daten gesammelt haben, können Sie nun ein ML-Modell erstellen und trainieren.

Impuls erstellen und Features generieren: Wählen Sie zunächst *Create impulse* im Menü. Setzen Sie *Window size* auf *2000 ms*, *Window increase* auf *500 ms* und *Frequency* auf *5 Hz* (**Bild 7**). Wählen Sie *Raw data* als Processing-Block und *Classification* als Learning-Block. Klicken Sie anschließend auf *Save impulse*, um die Konfiguration zu übernehmen.
Gehen Sie dann im Menü auf *Raw data* und klicken Sie auf *Generate features*. Nach der Verarbeitung erscheint rechts im Feature Explorer eine grafische Darstellung (**Bild 8**). Der Grad der Trennung zwischen den Daten-Clustern zeigt, wie gut das Modell zwischen den Klassen unterscheiden kann – je klarer die Cluster getrennt sind, desto einfacher wird die Klassifizierung.

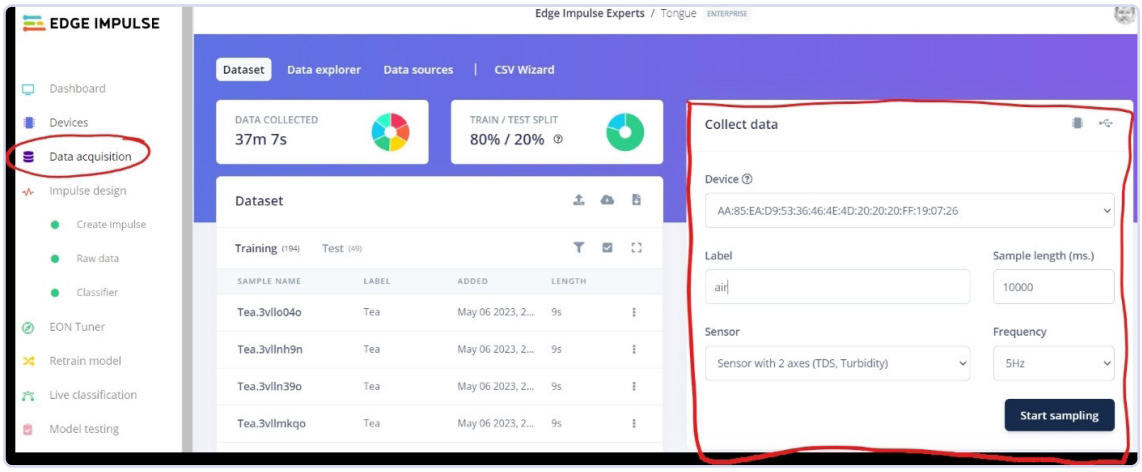


Bild 6. Datensammlungs-Ansicht beim Verbinden des Geräts und Aufzeichnen von Samples.

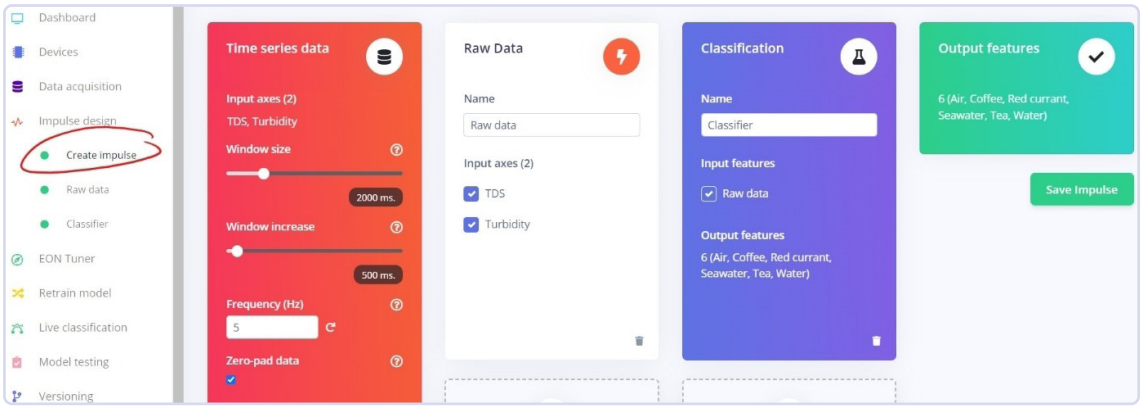


Bild 7. Schritte zum Erstellen eines Impulses und Generierung von Merkmalen.

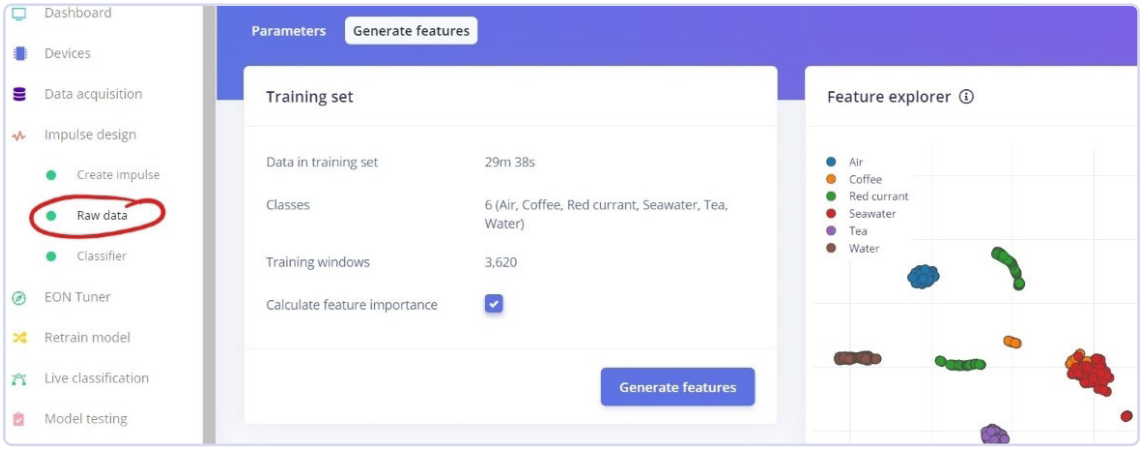


Bild 8. Merkmal-Generierungs-Schritt, der die Trennung der Klassen im Feature Explorer zeigt.

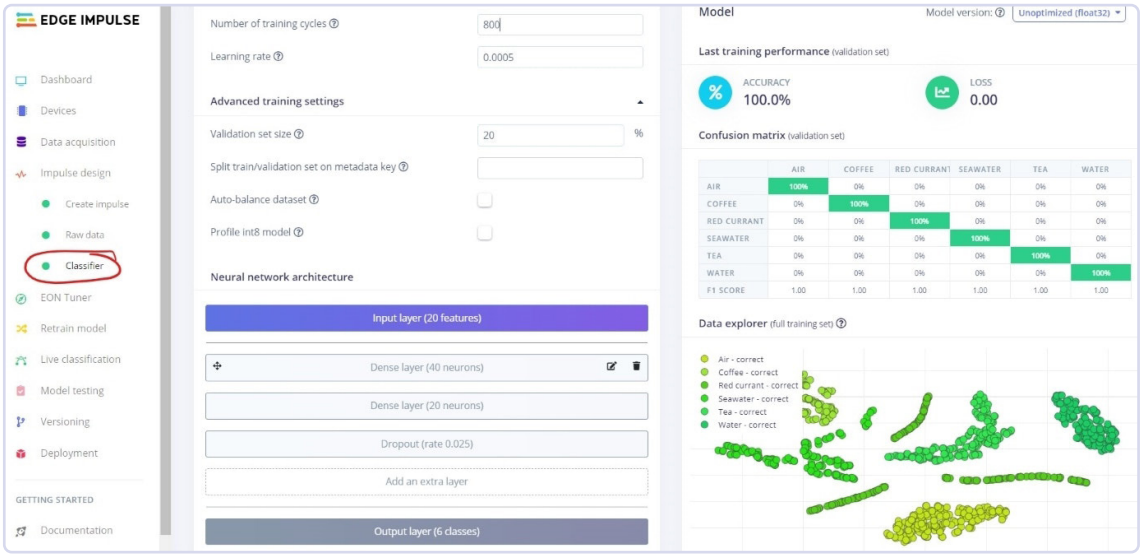


Bild 9. Classifier-Einstellungen (Dense Layers und Dropout).

In diesem Beispiel wurden für jede Probe identische Flüssigkeitskonzentrationen verwendet, was zu klar getrennten Clustern führt. In realen Szenarien sind jedoch Abweichungen unvermeidbar. Beispielsweise ist es beim Teekochen fast unmöglich, immer die exakt gleiche Stärke zu erreichen, was zu leichten Überschneidungen in der Datenverteilung führt.

Modell trainieren: Wählen Sie zum Trainieren des Modells *Classifier* aus dem Menü. Passen Sie die Einstellungen wie in **Bild 9** gezeigt an – die wichtigsten Parameter sind die *Dense layers* und die *Dropout rate*, Sie können aber auch andere Parameter ausprobieren, um die Leistung zu optimieren. Das unoptimierte float32-Modell, das mit diesen Parametern erzeugt wird, ist minimal, daher ist es meist nicht nötig, ein quantisiertes int8-Modell zu erstellen oder zu profilieren. In diesem Beispiel zeigte die int8-Version extrem schlechte Genauigkeit und war praktisch unbrauchbar; das float32-Modell funktionierte jedoch zuverlässig und effizient.
Nach der Konfiguration klicken Sie auf *Start training*. Sofern nicht Ihr Datensatz sehr groß oder das neuronale Netz besonders komplex ist, dauert das Training nur wenige Minuten. Die resultierende Genauigkeit hängt stark von der Qualität und Menge Ihrer Daten sowie vom Modellaufbau ab. Falls Sie mit dem float32-Modell eine unbefriedigende Genauigkeit erzielen, sammeln Sie weitere Daten. Überprüfen Sie auch, dass der Trübungssensor (der breitere Sensor) tief genug in die Flüssigkeit eingetaucht ist – anderenfalls könnte er Reflexionen aus der Luft statt aus der Probe erfassen.

Nachdem Sie neue Daten gesammelt haben, klicken Sie einfach auf *Retrain model*. Dadurch können Sie die Modellleistung verbessern, ohne die Features neu generieren zu müssen – das spart Zeit im Optimierungsprozess.

Modell testen: Für einen „Trockentest" – im wahrsten Sinne des Wortes – verwenden Sie Testdaten, die das Machine-Learning-Modell bisher nicht gesehen hat. Dieser Schritt simuliert reale Testbedingungen und zeigt, wie gut das Modell mit neuen Daten zurechtkommt.
Wählen Sie *Model testing* im Menü und klicken Sie auf *Classify all* (**Bild 10**). Nach kurzer Verarbeitung sehen Sie, wie das Modell bei unbekannten Daten abschneidet. Typischerweise ist die Genauigkeit etwas niedriger als beim Training, manchmal sogar höher. Die tatsächliche Modellleistung zeigt sich jedoch erst beim Testen mit echten Proben.
Sinkt die Genauigkeit deutlich gegenüber der Trainingsphase, deutet dies oft auf Overfitting hin – das Modell hat die Trainingsdaten zu genau gelernt und kann nicht mehr verallgemeinern. Reduzieren Sie dann die Modellkomplexität (z. B. weniger Neuronen), erhöhen Sie die Dropout-Rate oder sammeln Sie mehr Daten, um die Robustheit zu verbessern.

Modell-Einsatz (Nass-Test)

Wenn Sie das Modell im echten Leben testen möchten, in diesem Fall mit dem WIO Terminal, hat Edge Impulse dies erneut sehr einfach gemacht:

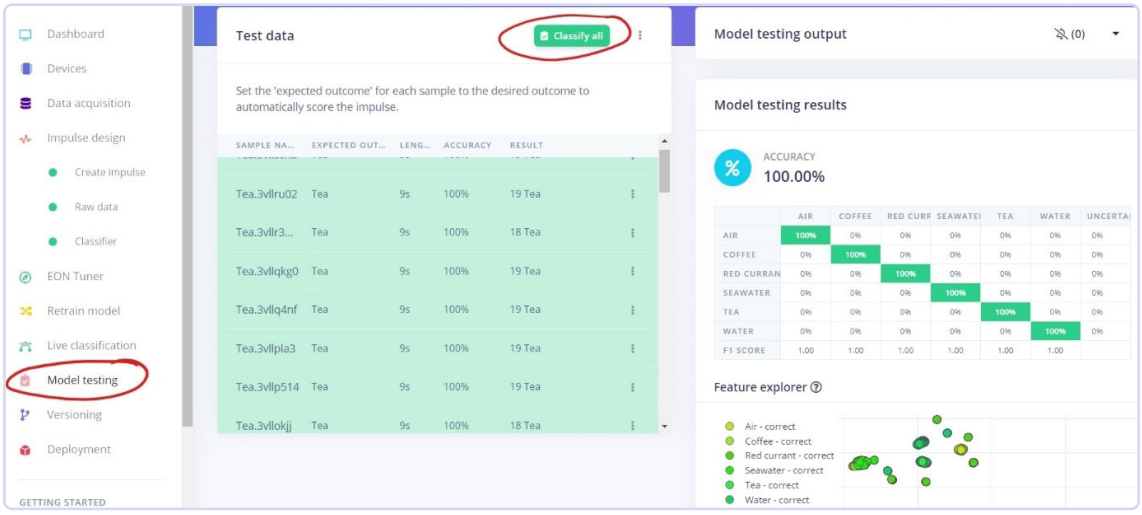


Bild 10. Modell-Test-Ansicht, die die Auswertung bei unbekannten (Test-) Daten zeigt.

- Wählen Sie *Deployment* im Menü (**Bild 11**).
- Wählen Sie *Arduino library*.
- Wählen Sie *Unoptimized (float32)* und klicken Sie auf *Build*.
- Nach wenigen Minuten erscheint ein Fenster, das zeigt, wie Sie die Arduino-Bibliothek in Ihren eigenen Code einbinden. Öffnen Sie in diesem Fall erneut die Datei *Tongue.ino* in der Arduino-IDE, binden Sie die neue Bibliothek wie angegeben ein und ersetzen Sie dann `<Tongue_inferencing.h>` durch die Header-Datei Ihrer eigenen Bibliothek. Wenn Sie Ihr Projekt in Edge Impulse z. B. *Liquids* genannt haben, würden Sie hier `<Liquids.h>` verwenden.

```
/* Includes ---- */
#include <Tongue_inferencing.h>
    // REPLACE this with your own library's header file

#include <SparkFunBQ27441.h>   // to read battery info
#include "seeed_line_chart.h"  //include the chart library
#include "TFT_eSPI.h"
```

Schließlich kompilieren und laden Sie das Programm auf das WIO Terminal. Wenn alles funktioniert, sehen Sie auf dem Gerät ein Liniendiagramm mit leicht variierenden Daten. Die Sensordaten werden auch im Arduino-IDE-Terminalfenster angezeigt.

Hinweise zum Programm *Tongue.ino*: Das Programm findet den höchsten Klassifikationswert und nimmt diesen als endgültige Vorhersage. Es gibt keinen Unsicherheits-Schwellenwert (d. h., alles wird in eine der Klassen eingeteilt). Sie können bei Bedarf eine Unsicherheitsklassifikation in das Arduino-Programm einbauen, indem Sie den Klassifikationswert mit einem Unsicherheits-Prozentschwellenwert vergleichen (z. B. 60 %).

Ergebnisse

Die ersten Ergebnisse beim Testen des Modells unter nassen Bedingungen waren sehr gemischt. Manchmal konnte das Modell eine Flüssigkeit immer korrekt zuordnen, manchmal scheiterte es völlig. Wie ich vermutet hatte, hatte ich anfangs nicht darauf geachtet, den Trübungssensor auf eine konstante Tiefe zu bringen. Daher habe ich komplett neu angefangen und alle Gläser bis fast zum Rand gefüllt, bevor ich die Daten erfasst habe. Das scheint die Lösung gewesen zu sein, denn das Modell funktioniert nun zu 100 %. Das zeigt erneut, dass Machine Learning oft ein iterativer Prozess ist und es besser ist, mit einer kleineren Datenmenge zu starten, ein ML-Modell zu bauen und zu testen und dann nach Bedarf anzupassen.
Wie bereits erwähnt, funktioniert das quantisierte int8-Modell extrem schlecht und ist unbrauchbar (**Bild 12**). Interessanterweise hat das unoptimierte float32-Modell sowohl in der Theorie als auch in der Praxis eine Genauigkeit von 100 %. Ich habe das nicht weiter untersucht, würde aber gerne wissen, wie dieser große Unterschied entsteht und wie man ihn beheben kann.
Außerdem habe ich das Modell nicht mit verschiedenen Konzentrationen getestet und vermute, dass es bei sehr stark verdünnten Flüssigkeiten Schwierigkeiten haben wird. Um dieses potenzielle Problem

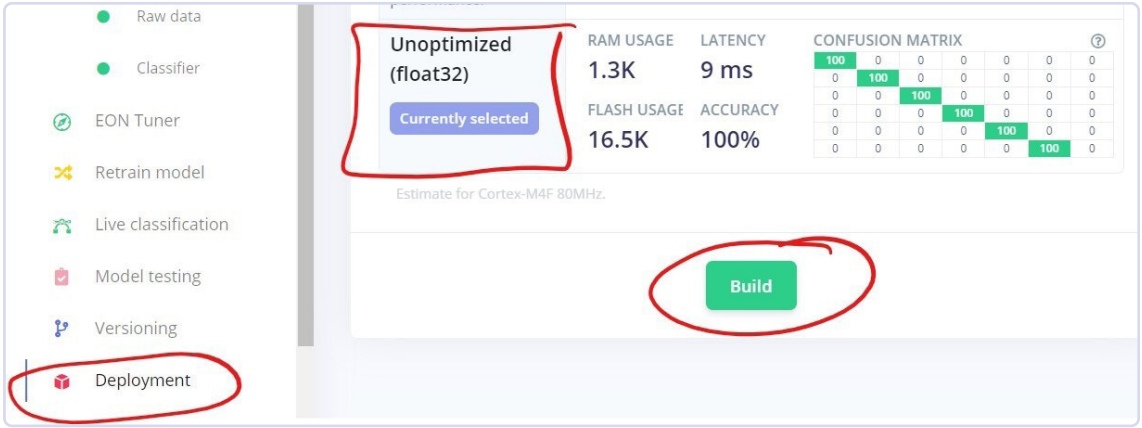


Bild 11. Bereitstellungs-Schritt zur Auswahl der Arduino-Bibliothek (unoptimiert, float32).

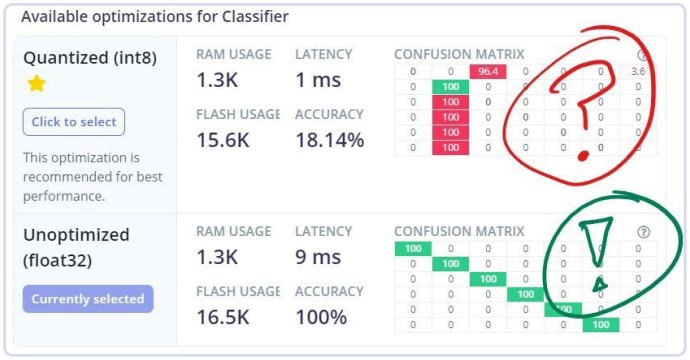


Bild 12. Das quantisierte int8-Modell funktioniert schlecht, während die float32-Version 100 % Genauigkeit erreicht.

zu lösen, müsste man Daten von Flüssigkeiten mit unterschiedlichen Verdünnungen sammeln und das Modell erneut trainieren. Beachten Sie außerdem, dass es wahrscheinlich völlig verschiedene Flüssigkeiten gibt, die mit den beiden verwendeten Sensoren gleiche Werte liefern. Wenn beide Sensoren ähnliche Werte messen, auch bei ganz unterschiedlichen Flüssigkeiten, betrachtet das Modell diese Flüssigkeiten als identisch. Das Problem kann durch „Sensorfusion" gelöst werden – also durch den Einsatz zusätzlicher oder anderer Sensortypen (z. B. Gassensoren für Geruch). Mit WIO Terminal und Grove-Komponenten sollte das recht einfach umzusetzen sein.
Trotz der oben genannten potenziellen Schwächen war ich positiv überrascht, dass das Modell an sich perfekt und konsistent funktioniert.

Quellen

Das Zungen-3D-Design ist eine Anpassung einer 3D-Datei auf Thingiverse [15]. Alle von mir erstellten 3D-Dateien sind unter der gleichen Lizenz wie die des ursprünglichen Urhebers (*Attribution-ShareAlike 3.0 Unported/CC BY-SA 3.0*).

250690-02

Über den Autor

Thomas Vikström ist Senior Lecturer für Informationstechnologie an einer finnischen Universität und beschäftigt sich seit vielen Jahren beruflich und privat mit KI, Robotik und Elektronik. In den letzten Jahren lag sein Fokus auf Edge-KI, insbesondere auf Mikroprozessor- und CPU-Ebene. Er hat zahlreiche projektbasierte Tutorials zu Machine Learning und verwandten Technologien veröffentlicht. Er freut sich über fachliche Kontakte und Diskussionen auf LinkedIn: www.linkedin.com/in/thomas-vikstrom.

Fragen oder Kommentare?

Haben Sie technische Fragen oder Feedback zu diesem Artikel? Bitte kontaktieren Sie die Elektor-Redaktion unter redaktion@elektor.de.

WEBLINKS

[1] „How I Built a Connected Artificial Nose (and How You Can Too!)", Blog von Benjamin Cabé, 2021: https://tinyurl.com/ye9d3tuc
[2] Grove–TDS-Sensor/Messgerät für Wasserqualität, Seeed Studio Store: https://www.seeedstudio.com/Grove-TDS-Sensor-p-4400.html
[3] Grove-Trübungssensor-Messgerät für Arduino V1.0, Seeed Studio Wiki: https://wiki.seeedstudio.com/Grove-Turbidity-Sensor-Meter-for-Arduino-V1.0/
[4] Wio Terminal, Seeed Studio Store: https://www.seeedstudio.com/Wio-Terminal-p-4509.html
[5] Grove-Trübungssensor, Seeed Studio Store: https://www.seeedstudio.com/Grove-Turbidity-Sensor-p-4399.html
[6] Wio Terminal Chassis - Akku (650 mAh), Seeed Studio: https://www.seeedstudio.com/Wio-Terminal-Chassis-Battery-650mAh-p-4756.html
[7] STL-Dateien auf GitHub: https://github.com/baljo/Tongue/tree/main/3D%20Model/STL-files
[8] „Wio Terminal Edge Impulse: Getting Started", Seee Seeed-Studio-Wiki: d Studio wiki: https://wiki.seeedstudio.com/Wio-Terminal-TinyML-EI-1/
[9] ei-tongue-arduino-1.0.2.zip, GitHub: https://github.com/baljo/Tongue/blob/main/ei-tongue-arduino-1.0.2.zip
[10] SparkFun BQ27441-G1A LiPo Fuel Gauge Arduino Library, GitHub: https://github.com/sparkfun/SparkFun_BQ27441_Arduino_Library/tree/master
[11] Seeed_Arduino_Linechart, GitHub: https://github.com/Seeed-Studio/Seeed_Arduino_Linechart
[12] Tongue.ino, GitHub: https://github.com/baljo/Tongue/blob/main/Tongue.ino
[13] Edge Impulse Data Forwarder: https://docs.edgeimpulse.com/tools/clis/edge-impulse-cli/data-forwarder
[14] Thomas Vikstrom, „Liquid Classification with TinyML – Seeed Wio Terminal + TDS Sensor", Edge Impulse Projects: https://tinyurl.com/yu74funx
[15] Original-3D-Datei der Zunge, Thingiverse: https://www.thingiverse.com/thing:644879

QOITECH

Do you care about battery life?

Trusted solutions for low-power profiling and debugging, battery validation, and battery life estimation of IoT and electronics.

www.qoitech.com

Bilderkennung mit Android-Smartphones und Raspberry Pi

Edge Impulse in Theorie und Praxis

Von Tam Hanna (Ungarn)

Die Entwicklung handelsüblicher ML-Aufgaben erfolgt meist durch Kombination vorhandener Algorithmen zu hauseigenen Pipelines. Das seit 2025 zu Qualcomm Technologies gehörende Unternehmen Edge Impulse hat diese „Theorie" in ein Produkt verwandelt und bietet eine Art Abstraktionsschicht für ML-Modellierer an. In diesem Artikel probieren wir die Plattform anhand einer Demo aus dem Bereich Bilderkennung aus. Der Schwerpunkt liegt hier beim Deployment auf einem Android-Smartphone und dem Raspberry Pi.

Im Prinzip arbeitet das – unter anderem unter [1] exzellent dokumentierte – System analog zu einem Cross-Plattform-Framework wie Qt. Die als „Impuls" bezeichnete ML-Pipeline wird im ersten Schritt in der Web-Anwendung *Edge Impulse Studio* generiert und danach von verschiedenen Compilern für alle möglichen Zielsysteme umgesetzt. Durch dieses Verfahren entstehen mehrere Vorteile. Erstens bietet Edge Impulse verschiedene Komfortfeatures an – einige davon werden wir uns in diesem Artikel näher ansehen. Zweitens lassen sich die generierten Modelle mit geringem Aufwand auf die verschiedensten Systeme portieren – der Kampf mit den nativen Entwicklungsumgebungen der jeweiligen Zielplattformen wird so auf ein Minimum reduziert. Insbesondere bei Aufgaben, bei denen es auf eine kurze Entwicklungszeit ankommt, ist die Nutzung von Edge Impulse sehr sinnvoll – die eingesparte Entwicklungszeit dürfte insbesondere bei kleinen Deployments die eventuell anfallenden Kosten amortisieren.

Edge Impulse steht aktuell in zwei Ausführungen zur Verfügung – eine freie (Developer) und eine Bezahlversion (Enterprise); wir wollen hier die kostenlose Basisvariante für den nichtkommerziellen Einsatz verwenden. Informationen zu den zusätzlichen Features der kommerziellen Variante finden sich unter [3] – zu beachten ist, dass es keine öffentlichen Preisinformationen seitens des Anbieters gibt.

Arbeit mit der Web-Oberfläche

Der erste Schritt ist das Besuchen der unter [2] bereitstehenden Webseite, wo wir uns für einen kostenlosen Account anmelden. Wer über ein Google-Konto verfügt, kann dieses für die Anmeldung heranziehen. Bei der Anmeldung wird außerdem noch ein Benutzername abgefragt. Danach steht die Benutzeroberfläche im Browser zur Verfügung.

Nach dem erfolgreichen Abnicken der diversen Einstellungsdialoge setzt uns Edge Impulse in dem in **Bild 1** gezeigten Projekt ab. Projekte dienen als „Container", die ML-Trainingsdaten, die Pipeline (den „Impuls") und die verschiedenen Parameter zur Verwaltung zusammenstellen. Zu beachten ist das links im Bild sichtbare Menü, das den Wechsel durch die verschiedenen Ansichten der Web-Applikation ermöglicht.

ML-Modelle leben und sterben mit der Qualität der gelieferten Trainingsdaten. Im Folgenden wollen wir ein ML-Modell zur Bildklassifizierung realisieren – das heißt, dass hierfür Foto-Trainingsdaten erforderlich sind. Im Fall einer normalen ML-Pipeline muss der Entwickler an dieser Stelle zur Digitalkamera greifen, fotografieren und die Bilder danach – idealerweise in Ordner unterteilt – an einer bestimmten Stelle in der IDE ablegen. Nutzer von Edge Impulse können ein bequemeres Verfahren wählen, falls sie über ein Smartphone mit ausreichend guter Kamera verfügen. Zur Verbindung des Smartphones mit dem Edge-Impulse-Projekt wechseln wir auf die Seite *Devices*, wo wir auf die Option *Connect a new device* klicken. Lohn der Mühen ist das Erscheinen eines Fensters, in dem sich unter anderem ein QR-Code findet. Wer ein aktuelles Android-Smartphone nutzt, kann diesen Code mit der Kamera einscannen und sieht auf seinem Telefon die in **Bild 2** gezeigte Erfolgsbestätigung.

Das Öffnen der im QR-Code gezeigten URL sollte übrigens nicht in einem anonymen Browsertab erfolgen, da Edge Impulse im Telefon ein Cookie platziert – spätere Aufrufe der URL sorgen dafür, dass sich die Web-Oberfläche automatisch mit dem Account bzw. Projekt verbindet.

Im nächsten Schritt folgt ein Klick auf die Option *Collect Images*, was die Plattform zum Einblenden einer Permissions-Anfrage veranlasst. Diese muss abgenickt werden, um den in **Bild 3** gezeigten *Capture*-Modus am Telefon zu aktivieren.

Der Autor möchte in diesem Projekt verschiedene Laborgeräte in Bildern unterscheiden, daher müssen wir für jeden der folgenden Schritte zunächst Bilder zusammenstellen. Im ersten Schritt folgt ein Klick auf das *Label*-Steuerelement, in dem wir als Label den String *drill* vergeben. Im nächsten Schritt greifen wir uns die Smartphone-Kamera und klicken den *Capture*-Button einige Male an, um ein Bildkollektion von (Tisch-)Bohrmaschinen zusammenzustellen – in der Dokumentation werden mindestens 30 bzw. 50 Bilder empfohlen, mehr sind logischer-

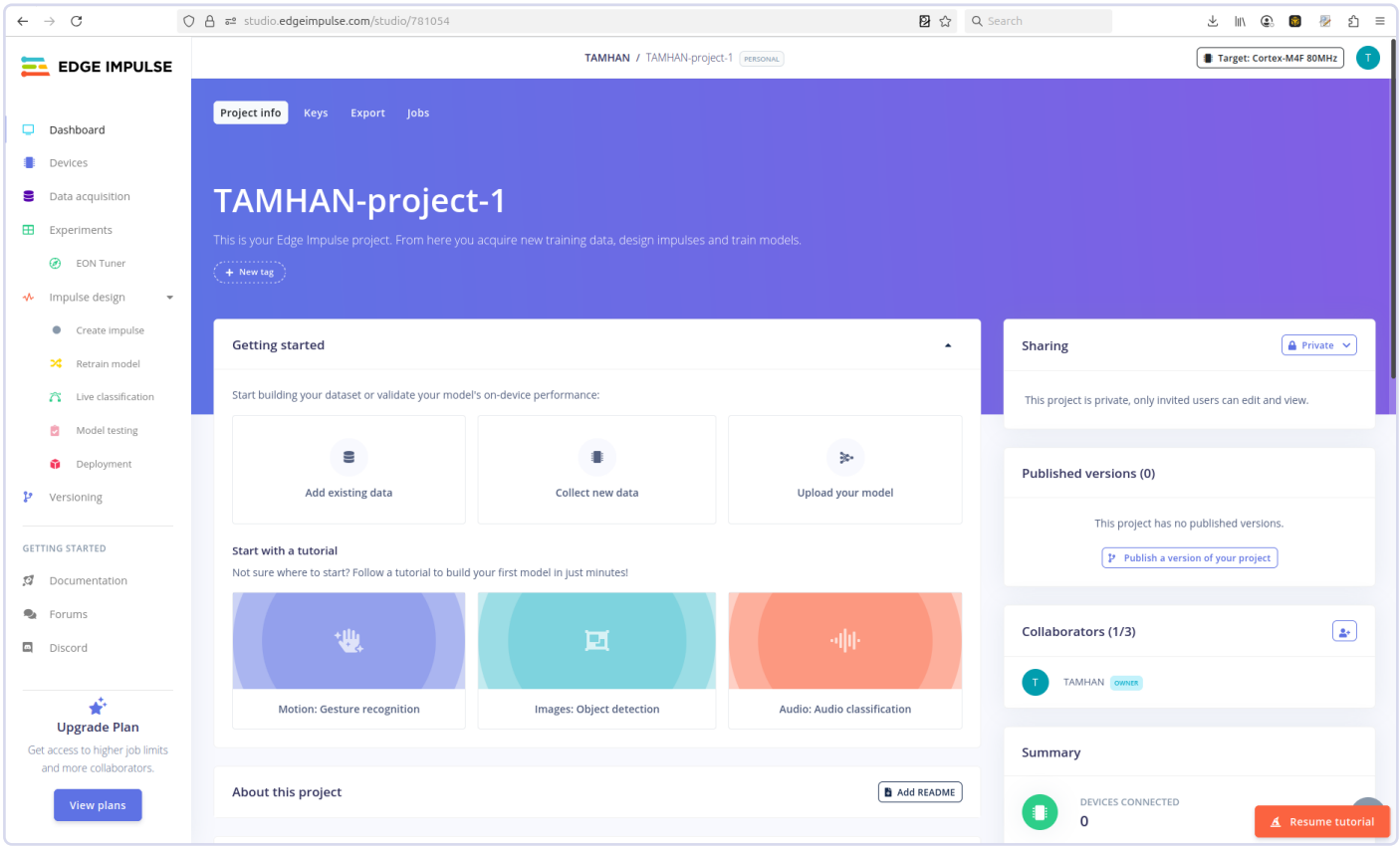


Bild 1. Noch ist das Projekt leer.

weise besser. Die Einteilung in Trainings- und Testdaten kann dabei eins zu eins übernommen werden.
Als Nächstes wird das Label *plant* vergeben. Auch hier sind rund 50 Fotos von Grünzeug aufzunehmen. Zu guter Letzt sind 50 zufällige Fotos des Raumes erforderlich – der Autor wird sie unter dem Label *random* ablegen. Nach getaner Arbeit erfolgt die Rückkehr in Edge Impulse Studio, wo wir uns links für die Option *Data Acquisitions* entscheiden.

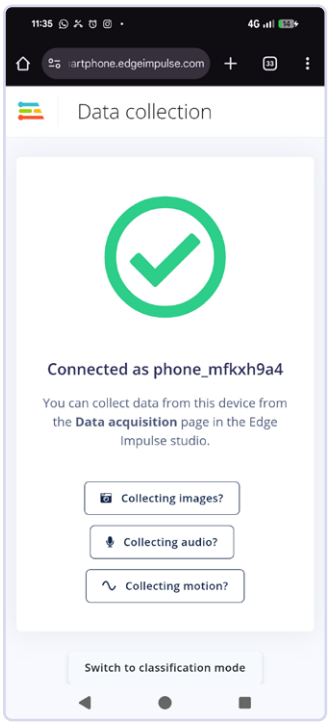


Bild 2. Das Telefon ist mit Edge Impulse verbunden.

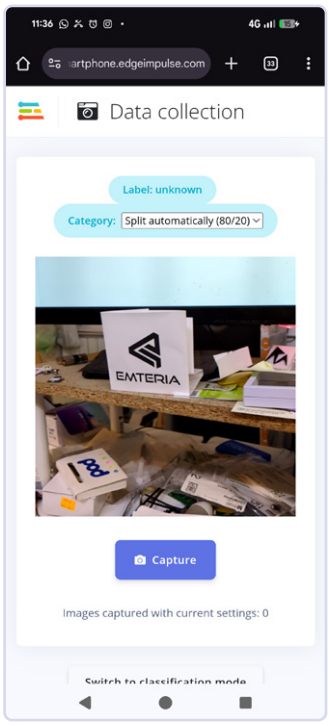


Bild 3. Die Kamera ist zur Datenakquise bereit.

Normalerweise sind die hochgeladenen Bilder sofort sichtbar, außerdem findet sich – wie in **Bild 4** gezeigt – in vielen Fällen ein Prompt, der auf die erwartete Art der zu realisierenden KI-Aufgabe hinweist, nämlich Objekterkennung (*Object Detection*), bei der auch mehrere Objekte in einem Bild erkannt werden. Da es bei uns eher um Bildklassifizierung (*Image Classification*) geht, müssen wir die Option *No* auswählen. In manchen Fällen vergeht etwas Zeit, bevor die Bilder mitsamt ihren Labels in der Arbeitsoberfläche auftauchen (**Bild 5**).

Die Signalverarbeitungskette

Im nächsten Schritt wählen wir die Option *Create Impulse*. Es erscheint eine an ein Flussdiagramm erinnernde Ansicht, die das Zusammenklicken der eigentlichen ML-Pipeline erlaubt. Im ersten Schritt folgt ein Klick auf *Add a processing block*, im daraufhin erscheinenden Dialog entscheiden wir uns für die Option *Image*. Im Feld *Add a Learning Block*

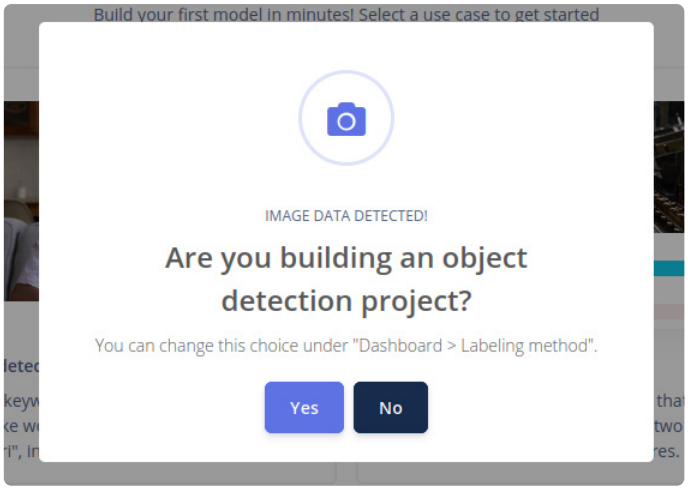


Bild 4. Bei uns geht es um Bildklassifizierung für ein einziges Objekt im Bild und nicht um Objekterkennung (bei der mehrere Objekte erkannt werden).

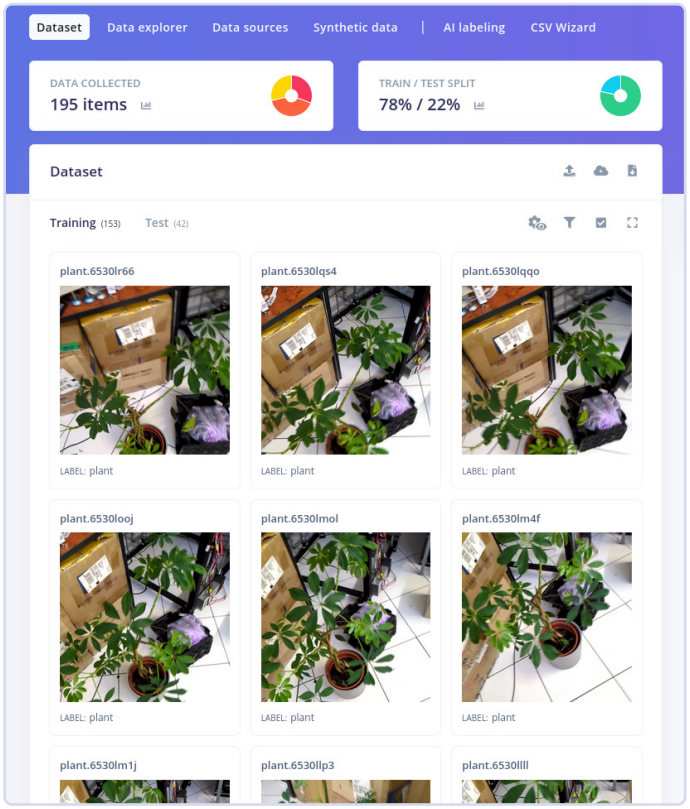


Bild 5. Bild und Label, in Edge Impulse vereint.

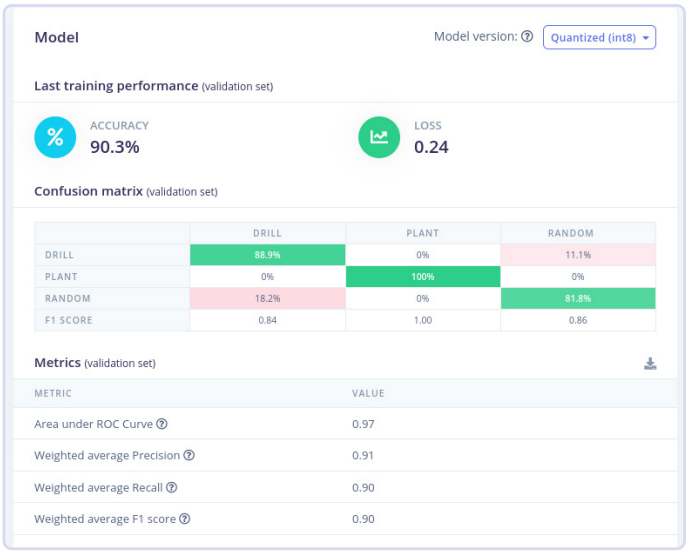


Bild 6. Der Trainingsprozess verlief mit brauchbarer Genauigkeit.

wählen wir die Option *Transfer Learning (Images)*. Zu guter Letzt ist ein Klick auf den grünen Button *Save Impulse* erforderlich, wodurch weitere Optionen in der ganz links eingeblendeten Werkzeugliste erscheinen. Im nächsten Schritt folgt ein Wechsel in die Rubrik *Image*, wo im Tab *Parameters* der Button *Save parameters* angeklickt wird. Das Benutzerinterface schlägt von Haus aus die Nutzung von RGB-Farben vor; dies ist ein Begehr, dem wir zustimmen können.

Nun finden wir uns in der Rubrik *Generate Features* wieder, wo wir auf den Button *Generate Features* klicken. Im Ausgabe-Panel *Feature generation output* wird man über den Fortschritt informiert. Nach der Ausgabe der Erfolgsmeldung *Job completed (success)* sollten Sie noch warten, bis im oben rechts eingeblendeten *Feature Explorer*-Fenster ein Diagramm erscheint. Danach ist die Parametrisierung von diesem Teil des ML-Systems abgeschlossen. Im nächsten Schritt folgt der Wechsel in die Rubrik *Transfer Learning*, wo wir die Option *Save & Train* wählen. Auch hier informiert das rechts eingeblendete Panel über die Fortschritte im Bereich des Trainings. Nach getaner Arbeit erscheint ein Ergebnisbericht, der – wie in **Bild 6** gezeigt – darüber informiert, wie „genau" das Modell mit den Trainingsdaten arbeitet.

Für einen ersten Praxistest bietet sich dann die Option *Model Testing* an. Hier wird deutlich, wie das Modell mit Testdaten umgeht, die es bisher noch nicht gesehen hat. Lohn der Mühen ist die Ausgabe eines Genauigkeitsberichts. Anzumerken ist, dass insbesondere in Laboren mit „viel Hintergrund" mit geringen Genauigkeiten zu rechnen ist. Außerdem ist zu berücksichtigen, dass unsere verwendeten Trainingsdatensätze vergleichsweise klein sind.

Einsatz des Modells auf dem Android-Smartphone

Nun möchte man natürlich wissen, wie gut das erstellte Modell – die Erkennung von Pflanzen und Bohrmaschinen in Kamerabildern – in der Realität funktioniert. Auf Wunsch kann Edge Impulse das Modell in eine kleine Anwendung packen, die im Smartphone-Browser läuft und schon ein paar Basisfunktionen mitbringt. Nach Anklicken des Links *Switch to classification mode* ist es mitunter erforderlich, abermals Zugriffsberechtigungen für die Kamera zu gewähren. Ist das Smartphone nicht mehr mit Edge Impulse verbunden, sollte man zur *Deployment*-Seite gehen und den QR-Code scannen, um das Modell auf dem Smartphone auszuführen. Danach erfolgt eine kleine Kompilation; **Bild 7** und **Bild 8** zeigen die Ergebnisse auf dem Smartphone des Autors.

Eine langlebigere Applikation, die außerhalb des Webbrowsers läuft, setzt etwas mehr Arbeit voraus. Dreh- und Angelpunkt ist hier abermals der „Impuls", der in der Edge-Impulse-Umgebung als Bibliothek, Package und Binary exportiert werden kann. Man erhält daraufhin einen Download-Ordner mit der Signalverarbeitung, dem ML-Modell und dem Code für die Inferenz. Erste Amtshandlung hierzu ist der Wechsel in die Rubrik *Deployment*, wo im Feld *Configure your deployment* eine Suchbox für verschiedene Vorlagen erscheint. Im ersten Schritt entscheiden wir uns hier für die Option *Android Library (C++)*, um ein auf die Bedürfnisse von Android-Smartphones optimiertes C++-Kompilat zu avisieren. Im danach eingeblendeten Optimierungs-

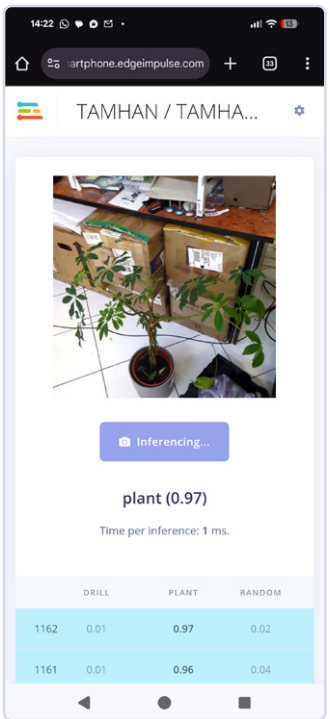


Bild 7. Die Erkennung der Pflanze …

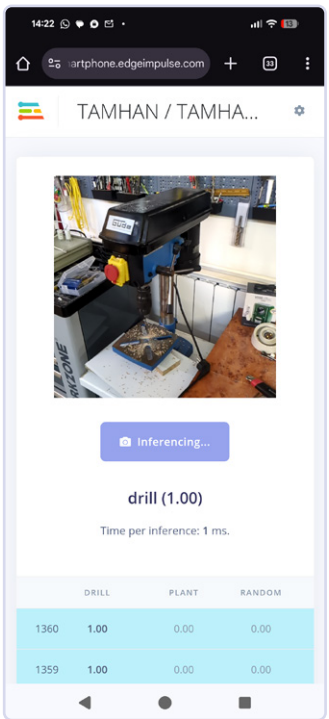


Bild 8. … und der Tischbohrmaschine funktionieren problemlos.

fenster bietet Edge Impulse verschiedene Optionen an, die ein Feintuning des Systemverhaltens ermöglichen. Für unsere Bedürfnisse ist dies nicht notwendig, weshalb wir an dieser Stelle auf den *Build*-Button klicken. Wie bei den bisherigen Experimenten wird auch diesmal auf der rechten Seite des Bildschirms ein Ausgabefenster eingeblendet, in dem die Entwicklungsumgebung über die Fortschritte berichtet. Hier gilt allerdings, dass die „Armierung" viel Zeit in Anspruch nimmt – in Tests des Autors steckte das System einige Minuten beim Schritt *Preparing the environment...* fest.
Nach dem erfolgreichen Durchlaufen des Prozesses wird eine.zip-Datei heruntergeladen – auf dem PC des Autors hieß sie *tamhan-project-1-cpp-android-v2.zip.* Dieses Archiv ist an einen bequem zugänglichen Ort im Dateisystem zu entpacken. Da die Verarbeitung von Kamera-Daten aufgrund der diversen Privatsphäre-Sicherungsmaßnahmen einen nicht unerheblichen Aufwand bei der Android-Programmierung darstellt, wollen wir in den folgenden Schritten auf ein schlüsselfertiges, von Edge Impulse zur Verfügung gestelltes Projektskelett zurückgreifen. Zu seiner Ausführung sind eine aktuelle Version von Android Studio (der Autor verwendet 2025.1.4 Canary 5) und eine Installation des Android-NDK unbedingt erforderlich. Die eigentliche Kamera-Android-Anwendung findet sich auf GitHub [4]. Zum Herunterladen ist es empfehlenswert, nach folgendem Schema das gesamte Repository auf den Rechner zu holen:

```
tamhan@TAMHAN18:~/Desktop/Stuff$ git clone https://github.
com/edgeimpulse/example-android-inferencing
```

Für unsere Experimente ist dann vor allem der Unterordner *example-android-inferencing/example_camera_inference* relevant, der im ersten Schritt in Android Studio über die *Open Project*-Funktion geladen wird. Das erstmalige Kompilieren eines von GitHub heruntergeladenen Projektskeletts benötigt etwas länger, weil das Build-System diverse betriebsnotwendige Komponenten aus den Google-Bibliotheksarchiven herunterladen muss.
Die erste Kompilation schlägt dann normalerweise mit einem Fehler der Bauart *:app debug:arm64-v8a failed to configure C/C++* fehl. An dieser Stelle ist es empfehlenswert, eine Aktualisierung des Gradle-Plugins durchzuführen. Android Studio bietet hierzu einen Assistenten an, im Fall des Autors erfolgte das automatisierte Update auf die Version 8.13.0. So die Rekompilation immer noch fehlschlägt, handelt es sich normalerweise um einen Verweis auf die Steuerungsdatei *CMakeLists.txt*. Spezifischerweise stirbt die Kompilation normalerweise in der Zeile 17, die sich folgendermaßen präsentiert:

```
include(edge-impulse-sdk/cmake/utils.cmake)
```

Dieser Fehler deutet auf das Fehlen von Bibliotheksdateien hin. Erste Amtshandlung ist das Herunterladen von TensorFlow – Edge Impulse kann dieses aus Gründen der Lizenzierung nicht als Teil des Projektskeletts zur Verfügung stellen. Zur Umgehung des Problems steht ein Shellskript zur Verfügung, das auf unixoiden Betriebssystemen nach der Zuweisung des Execute-Bits gemäß folgendem Schema zu aktivieren ist:

```
tamhan@TAMHAN18:~/Desktop/Stuff/example-android-inferen-
cing/example_camera_inference/app/src/main/cpp/tflite$
chmod +x download_tflite_libs.sh
tamhan@TAMHAN18:~/Desktop/Stuff/example-android-inferen-
cing/example_camera_inference/app/src/main/cpp/tflite$ ./
download_tflite_libs.sh
```

Wer mit Windows arbeitet, findet im genannten Verzeichnis auch eine klassische *.bat*-Datei. Die Ausführung erfolgt wie von der Windows-Administration gewohnt; Lohn der Mühen ist das Herunterladen der für die Ausführung notwendigen Code-Dateien.
Zur vollständigen Kompilierbarmachung müssen wir außerdem zum Archiv *tamhan-project-1-cpp-android-v2.zip* zurückkehren, das ja weiter oben aus Edge Impulse Studio heruntergeladen und an einem bequem zugänglichen Ort im Dateisystem entpackt wurde. Dort finden sich nun drei Ordner (*edge-impulse-sdk*, *model-parameters*, *tflite-model*) mit diversen Dateien, die für die Kompilation unbedingt notwendig sind – sie müssen in den Unterordner *example_camera_inference/app/src/main/cpp* des aus GitHub heruntergeladenen Projektskeletts wandern. An dieser Stelle bieten sich ein Neustart von Android Studio und eine Gradle-Resynchronisation an. Die Applikation lässt sich danach normalerweise kompilieren. Nach Quittierung der diversen von Android angezeigten Permission-Dialoge sieht man einen Live-Kameradatenstrom, in dem Pflanzen und Bohrmaschinen erkannt werden sollten.

Analyse des Datenverarbeitungsflows

Die vorliegende Applikation besteht – wie viele andere NDK-Programme – aus einem in C++ und einem in Kotlin gehaltenen Teil. Allgemeine Einstiegsinformationen in die Arbeit mit dem NDK findet man dabei unter anderem unter [5]. Die Arbeitsteilung zwischen Kotlin- und C++-Code sieht dann so aus, dass sich der Kotlin-Code um die Interaktion mit dem Betriebssystem und den zur Verfügung gestellten Services kümmert. Spezifischerweise errichtet der Code eine „Kamera-Pipeline", die – nach Abklärung der diversen Permissions – vom Betriebssystem regelmäßig mit Kameradatenframes befeuert wird. Die Versorgung der Edge-Impulse-Logik erfolgt dann durch den nach folgendem Schema aufgebauten Code:

```
private fun processImage(imageProxy: ImageProxy) {
    val bitmap = imageProxy.toBitmap()
    val byteArray = getByteArrayFromBitmap(bitmap)
    imageProxy.close()
    lifecycleScope.launch(Dispatchers.IO) {
        val result = passToCpp(byteArray)
        runOnUiThread {
            displayResults(result)
        }
    }
}
```

Erste Amtshandlung ist das Abernten der vom Betriebssystem zur Verfügung gestellten RGB-Informationen. Danach folgt die Aktivierung eines Dispatchers, der die Informationen aus Kotlin in den C++-Teil schickt.

C++-Interaktion über JNI

Auch wenn der Kotlin-Compiler es zu verbergen sucht – im Prinzip haben wir es hier nach wie vor mit Java zu tun. Eine Liste von brauchbarer Literatur zur Interaktions-Schnittstelle JNI findet sich unter [6].

`passToCpp` selbst ist dann schon eine in C++ gehaltene Methode, die über das JNI-Interface zur Ausführung gelangt. Ihre erste Amtshandlung ist das – hier aus Platzgründen nur teilweise abgedruckte – Konvertieren der Informationen, die danach an diverse zu Edge Impulse gehörende Funktionen weitergegeben werden:

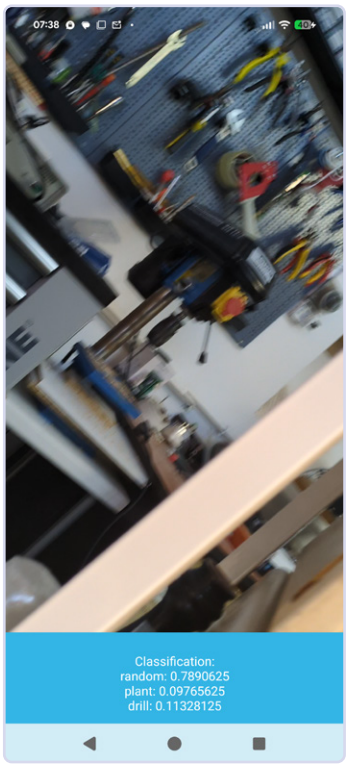


Bild 9. Die Inferenz-Pipeline funktioniert auch auf dem Android-Smartphone.

```
extern „C“ JNIEXPORT jobject JNICALL
Java_com_example_test_1camera_MainActivity_passToCpp(
        JNIEnv* env,
        jobject,
        jbyteArray image_data) {

    // Get byte array data from JNI
    byteData = env->
        GetByteArrayElements(image_data, nullptr);
    . . .
    ei_impulse_result_t result;

    signal_t signal;
    signal.total_length =
        EI_CLASSIFIER_INPUT_WIDTH *
        EI_CLASSIFIER_INPUT_HEIGHT;
    signal.get_data = &ei_camera_get_data;

    EI_IMPULSE_ERROR res =
        run_classifier(&signal, &result, false);
```

Die eigentliche Visualisierung der bereitgestellten Informationen erfolgt dann wieder im Kotlin-Code. Leider kommentiert das von Edge Impulse zur Verfügung gestellte Codebeispiel von Haus aus den für die Anzeige notwendigen Codeblock aus. Zur Lösung des Problems ist es im ersten Schritt erforderlich, das Widget `resultTextView` nach folgendem Schema als immer sichtbar zu deklarieren:

```
private fun displayResults(result: InferenceResult?) {
    resultTextView.visibility = View.VISIBLE
    boundingBoxOverlay.visibility = View.GONE
```

Weiter unten sind dann einige auskommentierte Zeilen zu „aktivieren“, um die vom C++-Teil angelieferten Informationen sichtbar zu machen:

```
if (result.objectDetections != null) {
    // Display object detection results
    val objectDetectionText =
       result.objectDetections.joinToString(„\n“) {
        „${it.label}: ${it.confidence},
         ${it.x}, ${it.y}, ${it.width}, ${it.height}“
    }
    . . .
```

Lohn der Mühen ist das in **Bild 9** gezeigte Verhalten.

Ausführung auf dem Raspberry Pi

Nachdem wir uns mit der lokalen Ausführung unseres Modells unter Android auseinandergesetzt haben, wollen wir uns im nächsten Schritt einer Plattform zuwenden, die in eigener Elektronik eingesetzt werden kann, nämlich einem Raspberry Pi 4. Dieser wird mit einem Kamera-Modul der zweiten Version verbunden. Prinzipiell kann auch ein Raspberry Pi 5 mit einer neueren Version der Kamera verwendet werden – zu beachten ist lediglich, dass das Kameradatenkabel zum vorliegenden Prozessrechner passen muss. Außerdem findet sich in der Dokumentation [7] die Passage *You must use 64-bit OS with _aarch64 and 32-bit OS with armv7l_**, die über an das Betriebssystem gestellte Ansprüche informiert.

Grundsätzlich gibt es zwei Vorgehensweisen: einerseits die Nutzung eines Docker-Containers, andererseits die Ausführung auf „Bare Metal“. Schon im Interesse der besseren Performance wollen wir hier auf die Bare-Metal-Ausführungsumgebung setzen – im ersten Schritt sind die folgenden drei Befehle auf dem Raspberry Pi erforderlich, um die Ausführungsumgebung als Ganzes vorzubereiten:

```
sudo apt update
sudo apt upgrade
curl -sL https://deb.
nodesource.com/setup_20.x
| sudo bash -
```

Spätestens nach der Aktualisierung der diversen Pakete ist ein Neustart erforderlich. Der Kamera-Support wird in manchen Versionen von Raspberry Pi OS nicht von Haus aus aktiviert, weshalb ein Umweg in das über Eingabe von `sudo raspi-config` aufrufbare Konfigurationsprogramm notwendig ist. Falls sich dort keine Option zur Aktivierung der Kamera vorfindet, ist sie normalerweise bereits aktiv.

Das eigentliche Deployment der benötigten Pakete erfolgt danach durch Eingabe der folgenden Kommandos:

```
sudo apt install -y gcc g++ make build-essential nodejs
sox gstreamer1.0-tools gstreamer1.0-plugins-good gstrea-
mer1.0-plugins-base gstreamer1.0-plugins-base-apps
sudo npm install edge-impulse-linux -g --unsafe-perm
sudo apt install -y gstreamer1.0-libcamera
```

Interessant ist, dass das Edge Impulse Linux-CLI in Node.js lebt. Während des Deployments der Pakete aus NPM heraus erscheinen einige Warnmeldungen in der Konsole, die auf veraltete Bibliotheksversionen hinweisen – im Rahmen unserer Experimente vor Ort sind diese ohne tiefergehende Relevanz. Die Verbindung mit Edge Impulse lässt sich dann auf der Kommandozeile durch Eingabe des folgenden Befehls auslösen:

```
pi@raspberrypi:~ $ edge-impulse-linux
```

Zu beachten ist, dass sich Personen, die sich mit einem Google-Konto bei Edge Impulse ausweisen, an dieser Stelle einen zusätzlichen Arbeitsschritt eingehandelt haben. Spezifischerweise muss ein kontenbezogenes Passwort festgelegt werden. Hierzu kehren wir in Edge Impulse Studio zurück, wo wir über das oben rechts eingeblendete User-Piktogramm in die Option *Account Settings* ▶ *Password* wechseln. Das Anklicken des Buttons *Set a password* ermöglicht dann die Eingabe eines Passworts – danach können wir auch schon zum Raspberry Pi zurückkehren, und das Einrichtungstool abermals zur Ausführung bringen. Nach dem erfolgreichen Durchlaufen des Assistenten – im Bereich der Geräte für Mikrofon und Kamera kann man die Standardeinstellungen belassen – erscheint der Raspberry Pi in der *Devices*-Ansicht von Impulse Studio wie in **Bild 10** gezeigt.

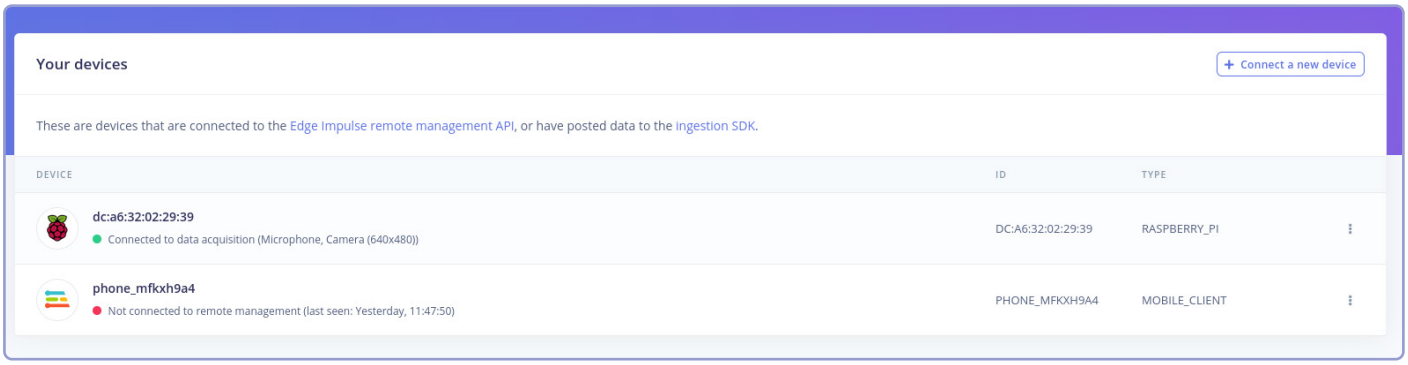


Bild 10. Der Raspberry Pi ist nun als Ziel ansprechbar.

Bild 11. Auch im Web-Interface steht eine Live-Vorschau zur Verfügung.

```
classifyRes 4ms. { drill: 0.0094, plant: 0.0961, random: 0.8945 }
classifyRes 4ms. { drill: 0.0102, plant: 0.1313, random: 0.8585 }
classifyRes 4ms. { drill: 0.0132, plant: 0.1411, random: 0.8457 }
classifyRes 4ms. { drill: 0.0089, plant: 0.1241, random: 0.867 }
classifyRes 4ms. { drill: 0.0082, plant: 0.1445, random: 0.8473 }
classifyRes 4ms. { drill: 0.0105, plant: 0.108, random: 0.8815 }
classifyRes 4ms. { drill: 0.0132, plant: 0.1269, random: 0.8599 }
classifyRes 4ms. { drill: 0.0107, plant: 0.1791, random: 0.8102 }
classifyRes 4ms. { drill: 0.012, plant: 0.1431, random: 0.8449 }
classifyRes 3ms. { drill: 0.0106, plant: 0.1659, random: 0.8235 }
classifyRes 4ms. { drill: 0.0121, plant: 0.1281, random: 0.8598 }
```

Bild 12. Kommandozeilen-Fans kommen ebenfalls auf ihre Kosten.

Falls sich das Einstellungsprogramm nach dem Festlegen eines Userpassworts nicht von selbst beendet, hilft das Senden von *Strg+C*. Im nächsten Schritt bietet sich der Start des Klassifizierers an, und zwar durch folgenden Befehl:

```
pi@raspberrypi:~ $ edge-impulse-linux-runner
```

Lohn der Mühen ist erstens die Ausgabe einer nach dem Schema `Want to see a feed of the camera and live classification in your browser? Go to http://192.168.1.103:4912` aufgebauten Meldung. Die Edge-Impulse-Runtime exponiert ein Webinterface, über das sich Informationen wie in **Bild 11** gezeigt abernten lassen. **Bild 12** zeigt, dass die Daten auch in der Kommandozeile zur Aberntung bereitstehen – hier ließe sich unkompliziert eine eigene Steuerungsanwendung anflanschen.

Zu beachten ist, dass die Kamera des Camera Module 2 ein komplett anderes Rauschverhalten aufweist. Da der Autor das Training hier mit der Kamera des Smartphones durchgeführt hat, sind die Ergebnisse mit dem Camera Module 2 nur wenig befriedigend. In jedem Fall sollte man darauf achten, dass die Bedingungen beim Training den tatsächlichen Bedingungen bei der Ausführung des Modells so gut wie möglich entsprechen.

Fazit

Wer seine ML-Aufgaben in Edge Impulse realisiert, kann sie mit minimalem Aufwand auf den verschiedensten Plattformen zur Ausführung bringen – die Bandbreite der unterstützten Systeme reicht vom Mikrocontroller über das Smartphone bis zu leistungsfähigen GPUs und NPUs. Insbesondere für Unternehmen, die keine KI-Spezialisten sind, bietet Edge Impulse einen attraktiven Weg zum schnellen Erreichen einer Lösung.

250741-02

Über den Autor

Ing. Tam Hanna befasst sich seit mehr als 20 Jahren mit Elektronik, Computern und Software; er ist freiberuflicher Entwickler, Buchautor und Journalist (www.instagram.com/tam.hanna). In seiner Freizeit beschäftigt sich Tam unter anderem mit 3D-Druck und dem Vertrieb von Zigarren.

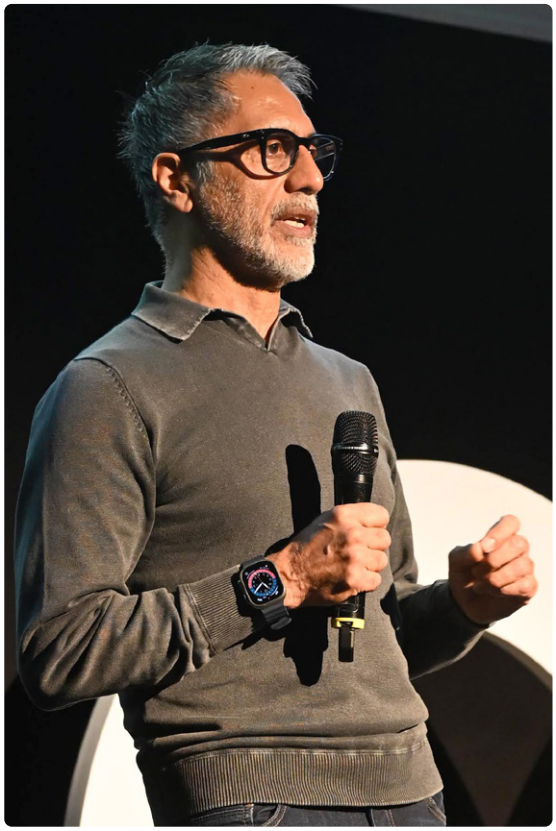
Fragen oder Kommentare?

Wenn Sie technische Fragen oder Anmerkungen zu diesem Artikel haben, nehmen Sie bitte Kontakt auf mit dem Autor (tamhan@tamoggemon.com) oder der Elektor-Redaktion unter redaktion@elektor.de.

WEBLINKS

[1] Shawn Hymel et al., „Edge Impulse: An MLOps Platform for Tiny Machine Learning“, Arxiv.org: https://arxiv.org/abs/2212.03332
[2] Edge Impulse Studio: https://studio.edgeimpulse.com/
[3] Edge-Impulse-Versionen: https://edgeimpulse.com/pricing
[4] Edge-Impulse-Kamera-Demo für Android, GitHub: https://github.com/edgeimpulse/example-android-inferencing
[5] „Get started with the NDK“, Android-Entwickler-Website: https://developer.android.com/ndk/guides
[6] Java and C/C++: JNI Guide, Reddit Forum: https://www.reddit.com/r/programming/comments/8cufpf/java_and_cc_jni_guide/
[7] Edge-Impulse-Dokumentation zum Raspberry Pi 4: https://docs.edgeimpulse.com/hardware/boards/raspberry-pi-4

Die nächste Generation intelligenter Systeme entwickeln



Manvinder „Manny" Singh, VP Produktmanagement

Von C. J. Abate (Elektor)

Für Entwickler, die an der nächsten Generation smarter Geräte arbeiten, eröffnet Edge-KI völlig neue Möglichkeiten. Manvinder „Manny" Singh, Vice President Product Management bei Qualcomm Technologies, erläutert, wie das Unternehmen die Entwicklung von Machine-Learning-Anwendungen vereinfacht, Embedded-Entwickler mit leistungsfähigeren Werkzeugen unterstützt und durch die Partnerschaft mit Edge Impulse eine neue Ära der Intelligenz direkt auf dem Gerät einleitet.

Abate: Sie arbeiten bereits seit 1996 bei Qualcomm. Wann wurde KI bei Qualcomm zu einem strategischen Thema?

Manvinder „Manny" Singh: KI gewann vor mehr als zehn Jahren strategisch an Bedeutung, insbesondere als wir das Potenzial von On-Device-Intelligenz erkannten, um das Nutzererlebnis von Mobilgeräten zu verändern. Qualcomm Technologies begann vor fast einem Jahrzehnt, KI-Fähigkeiten in seine Snapdragon®-Mobilplattformen zu integrieren. Dazu gehörte die Entwicklung des DSPs Qualcomm® Hexagon™, der GPU Qualcomm® Adreno™ und der CPU Qualcomm® Kryo™, die zusammen eine heterogene Computerarchitektur bildeten, die für KI-Workloads optimiert ist. Diese Komponenten ermöglichten Echtzeit-Inferenz für Aufgaben wie Computational Photography und Spracherkennung und legten das

Fundament für unsere Edge-KI-Strategie. Spätere Entwicklungen im Bereich neuronaler Netzbeschleuniger begannen sich mit unserer Führungsrolle im Bereich stromsparender Computer zu vereinen, sodass Edge-KI eine natürliche Weiterentwicklung darstellte.

Abate: Wann kam Edge Impulse bei Ihnen ins Blickfeld?

Singh: Das breite Portfolio von Qualcomm Technologies an SoCs, von Einsteiger- bis Hochleistungsmodellen, machte es zu einer Herausforderung, eine einheitliche KI-Entwicklungsumgebung zu schaffen. Entwickler hatten mit inkonsistenten Tools, fehlenden Abstraktionsschichten und eingeschränkter Unterstützung für hardwarebezogene Modelloptimierung zu kämpfen. Die interne Bewertung zeigte die Notwendigkeit einer End-to-End-MLOps-Plattform auf, die die KI-Entwicklung für alle Arten von Qualcomm-Technologies-Hardware vereinfacht. Edge Impulse geriet vor über einem Jahr in unser Blickfeld, da wir ein wachsendes Interesse der Entwickler an deren Plattform beobachteten. Die internen Plattformen von Qualcomm Technologies boten keine umfassenden Tools für Datenerfassung, -kennzeichnung und -transformation, was für die Entwicklung von Edge-ML-Modellen entscheidend ist. Edge Impulse hat dieses Problem gelöst. Ihr Fokus auf die Vereinfachung der ML-Entwicklung für Embedded-Geräte passt perfekt zu unserer Vision, Edge-KI zu demokratisieren.

Abate: Können Sie beschreiben, wie Qualcomms langfristige Vision für Edge-KI aussieht und wie Edge Impulse in diese Strategie passt?

Singh: Unsere langfristige Vision für Edge-KI besteht darin, intelligente, autonome Systeme in den Bereichen Industrie, Konsumgüter und Unternehmen zu ermöglichen. Edge-Impulse beschleunigt dies, indem es eine entwicklerfreundliche Plattform bietet, die Datenerfassung, Modelltraining und Einsatz auf Geräten, die von Qualcomm-Hardware angetrieben werden, miteinander verbindet.

Abate: Wie verbessert die Übernahme von Edge Impulse durch Qualcomm den End-to-End-ML-Workflow für Entwickler – von der Datenerfassung bis zur Bereitstellung und Modellüberwachung?

Singh: Edge Impulse bietet eine integrierte, intuitive Oberfläche, die die direkte Datenerfassung von Qualcomm-Technologies-Entwicklungsboards, assistiertes Labeling und synthetische Datenerzeugung, Signalverarbeitung, Modelltraining, automatische Modellanpassung bezüglich Leistung, Stromverbrauch und Speicherbedarf sowie nahtlose Bereitstellung über unsere SoCs hinweg unterstützt, alles ohne ML-Expertise. Entwickler können jetzt in wenigen Tagen – statt Monaten – vom Konzept zur Bereitstellung gelangen, dank vorintegrierter Entwicklungskits, integrierter Unterstützung verschiedener Modellarchitekturen und Zugang zu GPU-Training. Dies unterstützt auch die Umstellung von einer produktzentrierten auf eine plattformorientierte Strategie, wodurch Qualcomm Technologies für Long-Tail-Entwickler und Ökosystempartner relevanter wird.

Abate: Wie sehen Sie die Rolle von KI im industriellen IoT in den nächsten drei bis fünf Jahren?

Singh: KI wird zum Nervensystem des industriellen IoT und ermöglicht vorausschauende Wartung, adaptive Steuerung und Echtzeit-Entscheidungen. Wir werden einen Wandel von reaktiven zu proaktiven, sich selbst optimierenden Abläufen erleben, die Ausfallzeiten reduzieren und die Sicherheit verbessern.

Abate: Was sind angesichts der zunehmenden Verbreitung von Edge-KI die größten technischen Herausforderungen für Echtzeit-Inferenz auf ressourcenbeschränkten Geräten?

Singh: Die wichtigsten Herausforderungen sind das Gleichgewicht zwischen Rechenleistung und Energieeffizienz, die Robustheit der Modelle in realen Umgebungen und die sichere, latenzarme Inferenz. Der Betrieb von KI-Modellen auf ressourcenbeschränkten Edge-Geräten erfordert eine starke

Die Kombination aus multimodaler Sensorik und Echtzeit-Inferenz wird Anwendungen ermöglichen, die wir uns heute noch gar nicht vorstellen können.

Optimierung, Quantisierung, Pruning und Hardware-Beschleunigung, was ohne geeignete Tools technisch anspruchsvoll und fehleranfällig ist. Entwickler stehen vor einer fragmentierten Landschaft von SDKs, Compilern und Laufzeitumgebungen. Jede Hardwareplattform kann einzigartige Modellformate, spezielle Bereitstellungspipelines sowie manuelles Tuning für Speicher und Latenz erfordern, was die Entwicklung verlangsamt und die Lernkurve erhöht. Ohne strukturierte MLOps-Unterstützung fällt es Entwicklern schwer, die Modellleistung über die Zeit zu erhalten, besonders in industriellen Umgebungen, in denen Zuverlässigkeit entscheidend ist. Wir begegnen diesen Herausforderungen mit dedizierten KI-Beschleunigern und Software-Toolchains, die für den Edge-Einsatz optimiert sind.

Abate: Wie nah sind wir an vollständig autonomen IoT-Systemen, die vollständig durch Edge-KI betrieben werden? Und was bedeutet das für Industrie- und Konsumentenanwendungen?

Singh: Wir sind näher dran, als viele denken. Autonome Edge-KI verändert bereits das industrielle IoT: vorausschauende Wartung, Anomalieerkennung und Energieoptimierung in Fabriken und intelligenten Netzen, Einzelhandel: kassenlose Geschäfte, Regalintelligenz und Echtzeit-Bestandsverfolgung, Gesundheitswesen: Wearables, die Vitalwerte überwachen und Notfälle autonom erkennen, sowie Robotik und Drohnen: visuelle Inspektion, Navigation und Aufgabenerfüllung ohne menschliches Eingreifen usw. Volle Autonomie wird neue Effizienzniveaus, Sicherheit und Personalisierung ermöglichen, ohne auf Cloud-Konnektivität angewiesen zu sein.

Abate: Können Sie sich eine Zukunft vorstellen, in der Edge-KI Probleme oder Chancen erkennt, bevor Menschen sie wahrnehmen?

Singh: Absolut. Das ist das Versprechen der vorausschauenden KI. Mit genügend Kontextdaten und Echtzeitverarbeitung kann Edge-KI Anomalien erkennen, Ausfälle vorhersagen oder sogar Optimierungen vorschlagen – so können sich Menschen auf höhere Entscheidungsebenen konzentrieren.

Abate: Wenn Sie die „nächste Grenze“ für Edge-KI vorhersagen müssten, welche Anwendungen oder Szenarien werden die Branche und die Entwickler am meisten überraschen?

Singh: Ich glaube, wir werden Durchbrüche bei Edge-KI für Umweltsensorik, personalisierte Gesundheitsversorgung und autonome Robotik erleben. Die Kombination aus multimodaler Sensorik und Echtzeit-Inferenz wird Anwendungen ermöglichen, die wir uns heute noch gar nicht vorstellen können. Generative KI entwickelt sich rasant zu einer transformativen Kraft im Edge-Computing und eröffnet neue Möglichkeiten für autonome Systeme, industrielles IoT und personalisierte Nutzererfahrungen. Qualcomm Technologies steht an der Spitze dieses Wandels und entwickelt On-Device-GenKI-Funktionen, die die Möglichkeiten von Edge-Geräten neu definieren. Unsere hybride Edge-Cloud-Architektur bietet ein ausgewogenes Verhältnis von Leistung, Datenschutz und Skalierbarkeit, ideal für Unternehmensanwendungen, zum Beispiel Mitarbeiterassistenz: Chatbots und Q&A-Systeme, Echtzeit-Übersetzung, Zusammenfassung, Dokumentenerstellung usw.

Abate: Wenn Sie Ingenieuren und Entwicklern einen Rat für die Zukunft der Edge-KI geben könnten, wie würde dieser lauten?

Singh: Konzentrieren Sie sich darauf, den gesamten Stack zu verstehen – von den Sensordaten bis zur Modellausführung. Die wirkungsvollsten Lösungen kommen von denen, die Hardware-Beschränkungen mit Software-Innovation und benutzerzentriertem Design verbinden können.

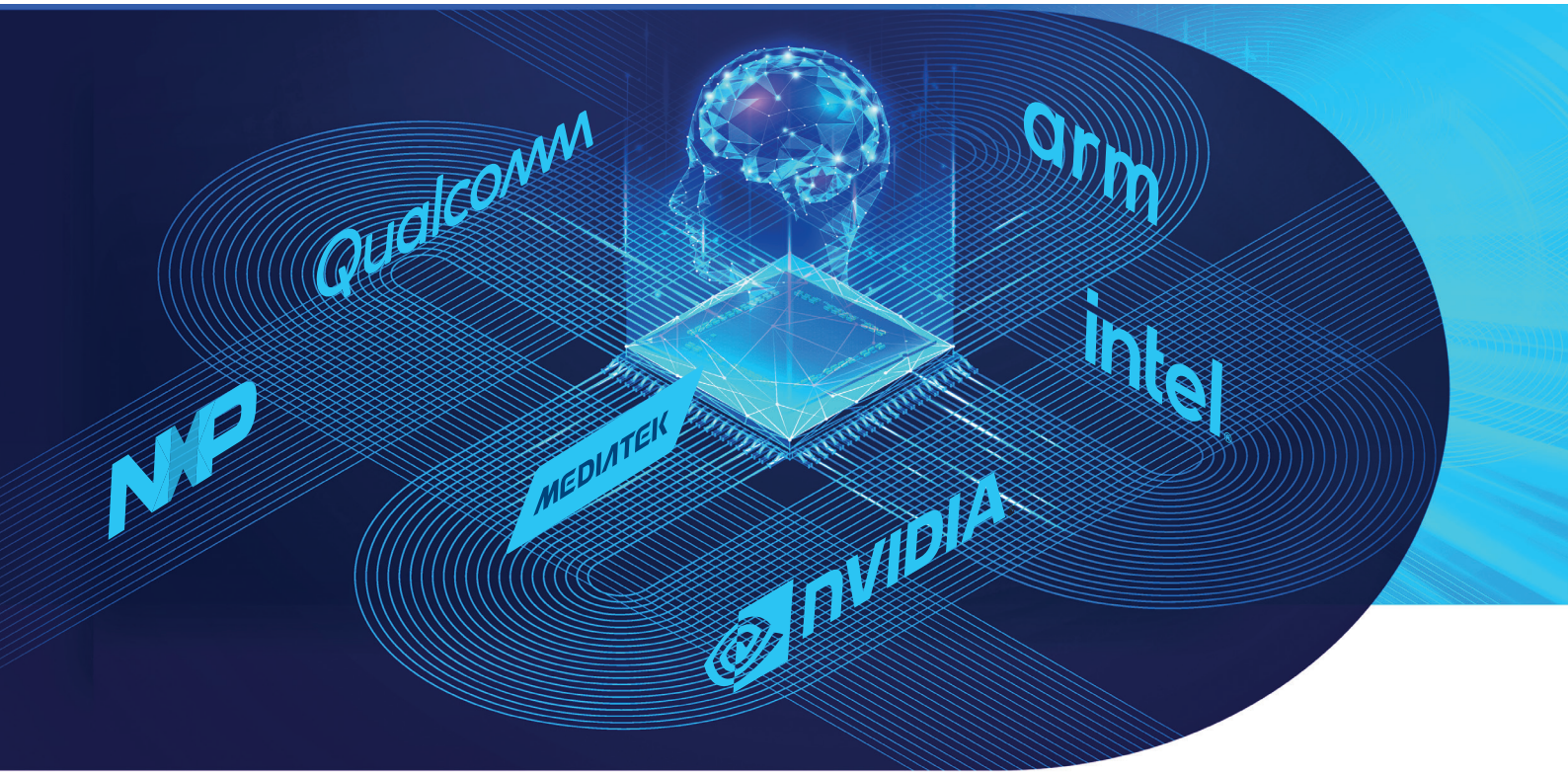
250785-02

KI wird zum Nervensystem des industriellen IoT und ermöglicht vorausschauende Wartung, adaptive Steuerung und Echtzeit-Entscheidungen.

EdgeAI

Computing Solutions

Accelerating AI Transformation in Embedded Markets



BREITES SPEKTRUM AN EMBEDDED SYSTEMEN

EdgeAI Servers

AI on Modules & Boards

AI-Ready Edge Systems

EdgeAI Systems

AI Acceleration Modules

Edge Ai Computers

ADVANTECH

Wussten Sie eigentlich, dass ADVANTECH ...

der Weltmarktführer für Embedded-Computing-Plattformen ist I lokalen Service und Entwicklungsdienstleistungen bietet I über die breiteste Produktpalette von Board-Level über Systeme bis hin zu vollständigen Cloud-Lösungen verfügt I unsere Kunden aus einer Vielzahl von kundenspezifischen Anpassungsoptionen wählen können **(DMS – Design & Manufacturing Services)** selbst fertigt und maßgeschneiderte lokale Logistikdienstleistungen bietet I Kontaktieren Sie uns unter: **00800-2426-8081** oder **embedded@advantech.eu**

Mehr entdecken

Edge-KI verbindet die Produktion mit dem Geschäftsergebnis

Source: Adobe stock / Justlight

Von Gledé Kabongo (Edge Impulse)

Die Industrie ist ein ausgesprochen spannendes Einsatzfeld für KI. In Lagerhallen und Logistikzentren weltweit erfassen Hersteller über Sensoren an unterschiedlichsten Maschinen enorme Datenmengen und erzeugen dabei täglich Terabytes an Informationen. Trotz dieses Reichtums an Daten bleibt jedoch die zentrale Herausforderung bestehen, aus Rohdaten verwertbare Erkenntnisse zu gewinnen – Erkenntnisse, die die Betriebseffizienz steigern, kostspielige Ausfallzeiten verhindern oder den Umsatz erhöhen könnten.

Ein Teil des Problems mit KI und Industrie besteht darin, dass Hardware-Intelligenz traditionell auf Cloud-Verarbeitung angewiesen ist, was Netzwerkverbindungen und ständige Verfügbarkeit erfordert. Industriemaschinen und die Sensoren, die sie überwachen, befinden sich jedoch oft an abgelegenen Standorten ohne zuverlässige und ständige Konnektivität und erzeugen manchmal sensible oder proprietäre Informationen, deren Übertragung Unternehmen nicht riskieren möchten. In solchen Situationen zeigt Edge-KI ihre Stärken, wenn die Sensoren selbst die Algorithmen ausführen, um beispielsweise Maschinenanomalien zu erkennen, Mitarbeiter zu identifizieren, die keine korrekte Schutzausrüstung tragen, oder eine Vielzahl weiterer wichtiger Erkenntnisse zu liefern – und dann darauf zu reagieren.
Wir haben Edge Impulse entwickelt, um ein Werkzeug für diesen bislang vernachlässigten Bereich bereitzustellen, und die Plattform definiert nun, wie Hersteller und Industrieunternehmen Embedded-Intelligenz nutzen, um Geschäftsziele zu erreichen.
Industrielle Anforderungen an Edge-KI sind vielfältig. In Bereichen wie Automatisierung, vorausschauender Wartung, Qualitätskontrolle und Asset-Tracking treiben wir das Feld mit marktführender Computer-Vision-Architektur voran, darunter das hochmoderne YOLO-Pro-Modell, das wir intern entwickelt haben, sowie den preisgekrönten Algorithmus FOMO-AD (Faster Objects, More Objects – Anomaly Detection).

Einige Einsatzbereiche:

- Computer-Vision-Modelle laufen auf Edge-Geräten, um Produktfehler in Millisekunden zu erkennen und Korrekturmaßnahmen einzuleiten, bevor fehlerhafte Einheiten die Fertigungslinie verlassen.
- Vibrationsanalyse-Algorithmen, die auf Gerätesensoren implementiert sind, erkennen Lagerschäden schon Tage vor einem katastrophalen Ausfall und ermöglichen geplante Wartungen, die ungeplante Ausfallzeiten eliminieren.
- Anomalie-Erkennungssysteme überwachen Umweltbedingungen und können Abweichungen sofort melden, um den Warenwert zu schützen und gleichbleibende Qualität zu gewährleisten.

Betriebseffizienz

Laut einer Siemens-Umfrage von 2024 [1] kosten ungeplante Produktionsausfälle die 500 größten Unternehmen der Welt 11 % ihres Umsatzes, 1,4 Bio. US-Dollar, was dem jährlichen BIP einer großen Industrienation wie Spanien entspricht. Die Stillstandskosten können auch für kleine und mittelständische

Hersteller erheblich sein und bis zu 150.000 US-Dollar pro Stunde betragen.
Predictive Maintenance, also die Nutzung von Datenanalysen, Machine-Learning-Algorithmen und Sensoren, um Geräteausfälle vorherzusagen, bevor sie auftreten, ist weiterhin einer der wichtigsten Anwendungsfälle für Edge-KI. Durch kontinuierliche Überwachung des Gerätezustands und der Leistung können Unternehmen Unregelmäßigkeiten oder Verschleißerscheinungen frühzeitig erkennen, sodass sie Wartungsarbeiten planen können, bevor es zu spät ist – und gleichzeitig unnötige und kostspielige Wartungen an einwandfrei laufenden Maschinen vermeiden.
Die Edge-Impulse-Technologie beschleunigt die Entwicklung vorausschauender Wartungsalgorithmen, die Sensor-, Audio- und Bilddaten für industrielle Anwendungen und fast jegliche anderen Anwendungsfälle nutzen:

- Erkennung von Geräteausfällen mittels Vibrationsanomalie-Erkennung zusammen mit Feuchte-, Temperatur- und Drucksensoren
- Inspektion von Montagelinien mit Computer Vision, um die Produktivität der Mitarbeiter zu steigern und menschliche Fehler zu reduzieren
- Vorausschauende Wartung von Anlagen wie Motoren, Pumpen oder Maschinen durch Sensor- oder Audiodaten, um Wartungen proaktiv durchzuführen und Produktionsausfälle zu minimieren
- Schadensprävention bei Nutzpflanzen durch Sensorüberwachung der Umweltbedingungen und möglicher Naturkatastrophen, sodass frühzeitig Maßnahmen zur Schadensminderung ergriffen werden können

Wirkung und Leistung von Edge-KI in der Praxis

Die Vielseitigkeit von Edge-KI zeigt sich in zahlreichen industriellen Anwendungen und liefert messbare Ergebnisse etwa in den Bereichen Automobilindustrie, Fertigung, Energie, Qualitätssicherung, Logistik und mehr. Diese realen Anwendungen belegen, wie theoretische Fähigkeiten in konkreten geschäftlichen Mehrwert umgesetzt werden.

Fallstudie 1: Industrielles Asset-Tracking und Bestandsmanagement

Viele Industrieunternehmen sind auf eine genaue Nachverfolgung und Verwaltung ihres Lagerbestands angewiesen. Laufen solche Prozesse jedoch manuell, sind sie anfällig für menschliche Fehler und bieten keine Echtzeittransparenz. Palettenzählung, ein wichtiger Bestandteil der globalen Logistik, kann mit Edge-KI automatisiert werden, um Ineffizienzen und Wertverluste zu reduzieren. Edge-Impulse hat ein Vision-basiertes System entwickelt, das Paletten beim Durchlaufen eines Lagers präzise erkennen und verfolgen kann.
Geschäftlicher Nutzen: Deutliche Verkürzung der Markteinführungszeit – traditionell dauert das Sammeln von Daten, das Training von industrietauglichen Modellen und die Implementierung Monate bis Jahre. Mehr dazu unter [2].

Fallstudie 2: Predictive Maintenance für industrielle Services

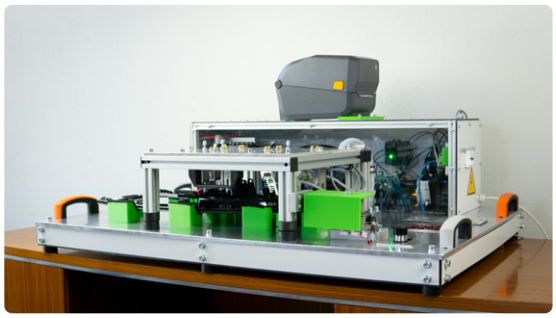
Atlas Machine & Supply, ein US-amerikanisches Industriedienstleistungsunternehmen mit mehr als 5.000 Kunden, benötigte eine einheitliche Predictive-Maintenance-Lösung für einen vielfältigen Kompressoren-Fuhrpark. Sie kombinierten die Arduino Opta micro PLC und die Arduino-Cloud, um Sensordaten zu zentralisieren und Echtzeit-Dashboards bereitzustellen. Das Unternehmen erweitert seine Lösung nun durch Edge-KI für vorausschauende Wartung. Die vollständige Beschreibung finden Sie unter [3].
Geschäftlicher Nutzen: Verbesserte Betriebszeiten, frühzeitige Fehlererkennung, stärkere Kundenbindung und ein flexibles, skalierbares System für gemischte Geräteflotten.

Fallstudie 3: Qualitätskontrolle – Automobilzulieferer
Mit Edge-KI für Echtzeit-Inspektion und Fehlererkennung können Hersteller den Inspektionsprozess automatisieren, indem sie Computer-Vision-Modelle auf Edge-Geräten einsetzen. Dadurch wird die Erkennung von Fehlern, Anomalien oder Abweichungen in Produkten in Echtzeit ermöglicht. Ein europäischer Hersteller von Komponenten für eine Luxusautomarke nutzt KI für Computer-Vision-End-of-Line-Qualitätskontrolle und setzt dabei die Anomalie-Erkennungsbibliothek FOMO-HD von Edge Impulse ein. Das Machine-Learning-Modell erkennt fehlerhafte Teile und falsche Teileplatzierung auf dem Montageband während der Fertigung.
Geschäftlicher Nutzen: Reduzierung von Maschinenstillständen, Produktionsausfällen und Maschinenschäden/Wartung. Dies führte zu höherer Produktivität und einer messbaren Kosteneinsparung von 1,6 Mio. US-Dollar.

Fallstudie 4: IDT-Industrieautomatisierung – italienischer Hersteller
IDT entwickelt intelligente Prüfvorrichtungen für End-of-Line-Qualitätskontrollen von Automobilkomponenten sowie Lösungen für Luxusautomobilhersteller mit kleinen/mittleren Serien.

Die Vielseitigkeit von Edge-KI zeigt sich in zahlreichen industriellen Anwendungen und liefert messbare Ergebnisse etwa in den Bereichen Automobilindustrie, Fertigung, Energie, Qualitätssicherung, Logistik und mehr.

Geschäftlicher Nutzen: Effizienzsteigerung durch offene und anpassbare Lösungen, wodurch Kunden eine Herstellerbindung vermeiden können. Schnellere Wertschöpfung bei neuen Produkteinführungen. Rückverfolgbarkeit durch bildbasierte Dokumentation.



Quelle: IDT

Wie die Industrie von Edge-KI profitieren kann

Edge-KI liefert messbare Ergebnisse, darunter Kosteneinsparungen durch reduzierte Ausfallzeiten, verbesserte Workflow-Zuverlässigkeit, höhere Produktqualität und Kundenzufriedenheit sowie schnellere Innovationszyklen. Doch wo sollte ein Unternehmen, das die Leistungsfähigkeit von Edge-KI nutzen möchte, beginnen? Die Integration von Edge-KI in die Kernprozesse und Produktangebote kann Wachstum fördern, das Kundenerlebnis verbessern und mit der richtigen Strategie sowie unterstützender Technologie einen Wettbewerbsvorteil bringen.

250814-02

WEBLINKS

[1] „The True Cost of Downtime 2024“, Siemens AG, 2024: https://tinyurl.com/423duc3f
[2] „Industrial Asset Tracking with NVIDIA Omniverse and TAO“, Edge-Impulse-Fallstudie: https://edgeimpulse.com/case-studies/industrial-asset-tracking-with-nvidia-omniverse-and-tao
[3] „Atlas Machine and Supply innovates predictive maintenance for industrial compressors with Arduino“, Arduino-Pro-Website: https://www.arduino.cc/pro/success-story-atlas-machine-and-supply/

Got Brilliant Electronics-Related Ideas?

Are you a maker, engineer, or educator with a brilliant idea?

Since 1961, Elektor has partnered with electronics experts to publish innovative educational content—from articles and books to kits and conferences and practical learning modules.

Now, we're expanding our expert network and inviting new contributors to join us in creating valuable products for a global audience.

Whether you've designed a clever circuit, developed a teaching tool, or have an idea for a hands-on project—pitch it to us. We'll help shape, publish, and market it—and you'll earn along the way.

- Share your expertise
- Generate income
- Join Elektor's expert community

Sign up and pitch your idea—together, we'll *launch* powerful new learning products.

Go to www.elektormagazine.com/submissions

elektor MEDIA & LEARNING

Künstlich oder künstlerisch?

Ein unterhaltsamer Rückblick auf die frühe KI-Berichterstattung in Elektor

Von Jan Buiting (Elektor)

Lange bevor der heutige KI-Boom einsetzte, dokumentierte Elektor die frühen Grundlagen der künstlichen Intelligenz mit praxisnahen Projekten und vorausschauenden Artikeln, die rückblickend erstaunlich weitsichtig wirken. Ob Experimente mit Kybernetik in den 1970er-Jahren oder das frühe Tüfteln an neuronalen Netzen – die Elektor-Autoren beschäftigten sich schon damals mit derselben Frage, die uns bis heute begleitet: Was bedeutet eigentlich maschinelle Intelligenz?

Man vergisst leicht, dass die Grundlagen des heutigen Hype-Themas künstliche Intelligenz bis zu den Anfängen der Embedded-Computertechnik, neuronale Netze und sogar Fuzzy-Logik zurückverfolgt werden können. Lange bevor ChatGPT und Deep Learning die Schlagzeilen beherrschten, erkundeten Elektronik-Enthusiasten bereits die Grenzen der Maschinenintelligenz durch praktische Projekte und theoretische Diskussionen. Hier finden Sie Beispiele aus dem Elektor-Archiv, die sich über fast zwei Jahrzehnte der KI-Forschung erstrecken – hauptsächlich „zum Schmunzeln“, aber auch, um zu würdigen, wie vorausschauend diese frühen Pioniere waren.

Die Grundlagenjahre (1975-1981)

Was ist Kybernetik? (1975)

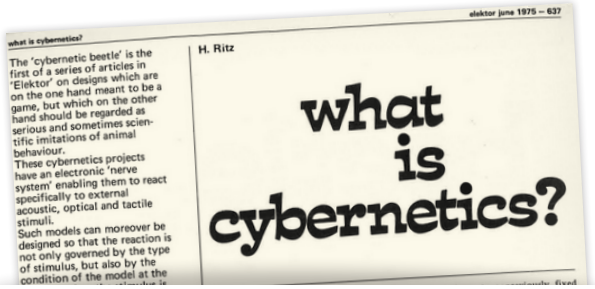
Die Reise beginnt mit einem der ersten Ausflüge von Elektor in intelligente Systeme. Dieser grundlegende Artikel führte die Leser in die Kybernetik ein – die Lehre von Kommunikation und Steuerung bei Maschinen und Lebewesen. Schon 1975 erforschte die Zeitschrift, wie Rückkopplungsschleifen und selbstregulierende Systeme ein intelligenteres Verhalten in elektronischen Geräten ermöglichen können. [1]

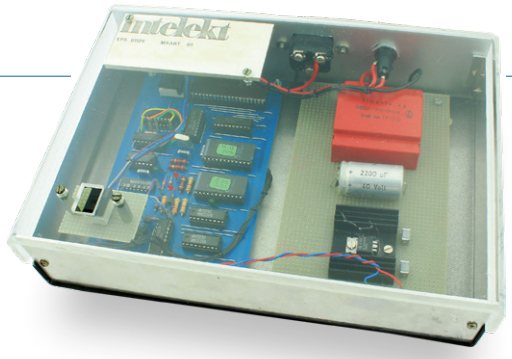
Der Intelektor-Schachcomputer von Elektor (April 1981)

Dieser Artikel stellt den Intelektor-Schachcomputer vor, der auf dem Intel-8088-Mikroprozessor basiert und eine portierte Version von Tiny Chess ausführt. Das Design legt Wert auf angemessene Geschwindigkeit und „Intelligenz“-Niveaus und macht ihn zu einem starken Gegner für viele Schachfreunde. [2]

Schon weil der Laborprototyp des Intelektor mehrere Umzüge und umfangreiche Aufräumaktionen im Elektor-Labor überlebte, wurde dieses berühmte Gerät als Teil der *Retronics*-Serie (2004–2020) erneut vorgestellt. [3]

Entscheiden Sie selbst, ob der Schachspieler mit dem Namen 8088 nun intelligent, künstlich intelligent oder quasi-intelligent ist –

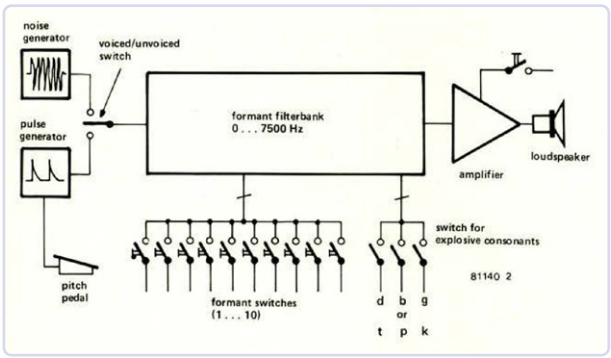




oder einfach nur den dummen Von-Neumann-Strategien folgt, die in ein EPROM programmiert wurden.

Mit Computern sprechen (1981)

Zur gleichen Zeit befasste sich Elektor bereits mit der Mensch-Maschine-Interaktion durch Spracherkennung und -synthese. Diese zweiteilige Serie zeigte, wie Computer menschliche Sprache verstehen und darauf reagieren können – eine Herausforderung, die noch Jahrzehnte im Mittelpunkt der KI-Forschung stehen sollte. [4][5]



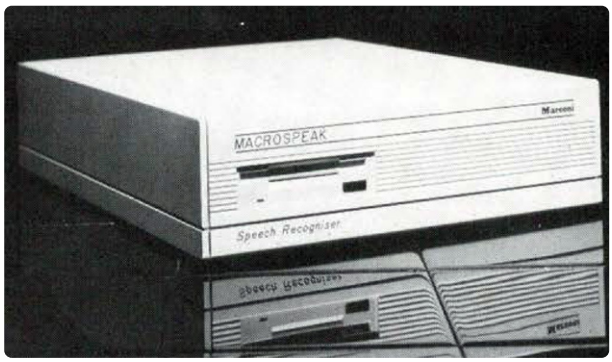
Die Expansionsphase (1986-1990)

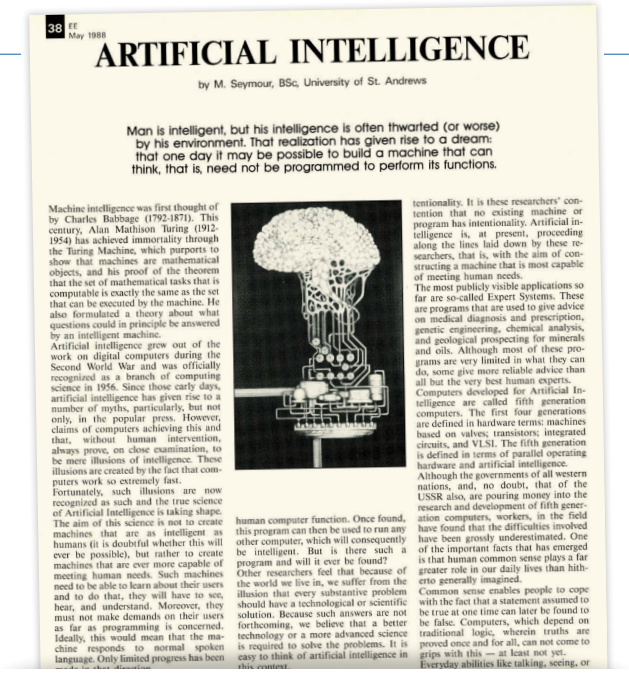
Die Zukunft der künstlichen Intelligenz (1986)

Mit dem Fortschreiten der 1980er Jahre nahm Elektor eine strategischere Sichtweise auf das Potenzial von KI an. Dieser Artikel untersuchte, wohin sich künstliche Intelligenz entwickeln könnte und welche Durchbrüche in den kommenden Jahren möglich sein könnten. [6]

Spracherkennungssystem (1987)

Aufbauend auf früheren Arbeiten zur Mensch-Maschine-Interaktion zeigte dieses Praxisprojekt den Lesern, wie sie ein eigenes Spracherkennungssystem aufbauen konnten – so wurde KI für Elektronikbegeisterte greifbar. [7]




38 EE May 1988

ARTIFICIAL INTELLIGENCE

by M. Seymour, BSc, University of St. Andrews

Man is intelligent, but his intelligence is often thwarted (or worse) by his environment. That realization has given rise to a dream: that one day it may be possible to build a machine that can think, that is, need not be programmed to perform its functions.

Machine intelligence was first thought of by Charles Babbage (1792-1871). This century, Alan Mathison Turing (1912-1954) has achieved immortality through the Turing Machine, which purports to show that machines are mathematical objects, and his proof of the theorem that the set of mathematical tasks that is computable is exactly the same as the set that can be executed by the machine. He also formulated a theory about what questions could in principle be answered by an intelligent machine.

Artificial intelligence grew out of the work on digital computers during the Second World War and was officially recognized as a branch of computing science in 1956. Since those early days, artificial intelligence has given rise to a number of myths, particularly, but not only, in the popular press. However, claims of computers achieving this and that, without human intervention, always prove, on close examination, to be mere illusions of intelligence. These illusions are created by the fact that computers work so extremely fast.

Fortunately, such illusions are now recognized as such and the true science of Artificial Intelligence is taking shape. The aim of this science is not to create machines that are as intelligent as humans (it is doubtful whether this will ever be possible), but rather to create machines that are ever more capable of meeting human needs. Such machines need to be able to learn about their users and to do that, they will have to see, hear, and understand. Moreover, they must not make demands on their users as far as programming is concerned. Ideally, this would mean that the machine responds to normal spoken language. Only limited progress has been

human computer function. Once found, this program can then be used to run any other computer, which will consequently be intelligent. But is there such a program and will it ever be found? Other researchers feel that because of the world we live in, we suffer from the illusion that every substantive problem should have a technological or scientific solution. Because such answers are not forthcoming, we believe that a better technology or a more advanced science is required to solve the problems. It is easy to think of artificial intelligence in this context.

tentionality. It is these researchers' contention that no existing machine or program has intentionality. Artificial intelligence is, at present, proceeding along the lines laid down by these researchers, that is, with the aim of constructing a machine that is most capable of meeting human needs.

The most publicly visible applications so far are so-called Expert Systems. These are programs that are used to give advice on medical diagnosis and prescription, genetic engineering, chemical analysis, and geological prospecting for minerals and oils. Although most of these programs are very limited in what they can do, some give more reliable advice than all but the very best human experts.

Computers developed for Artificial Intelligence are called fifth generation computers. The first four generations are defined in hardware terms: machines based on valves; transistors; integrated circuits, and VLSI. The fifth generation is defined in terms of parallel operating hardware and artificial intelligence.

Although the governments of all western nations, and, no doubt, that of the USSR also, are pouring money into the research and development of fifth generation computers, workers, in the field have found that the difficulties involved have been grossly underestimated. One of the important facts that has emerged is that human common sense plays a far greater role in our daily lives than hitherto generally imagined.

Common sense enables people to cope with the fact that a statement assumed to be true at one time can later be found to be false. Computers, which depend on traditional logic, wherein truths are proved once and for all, can not come to grips with this — at least not yet.

Everyday abilities like talking, seeing, or

Künstliche Intelligenz (Mai 1988)

Dieser umfassende Artikel von Mark Seymour behandelte das Potenzial, Maschinen zu erschaffen, die unabhängig denken können, ohne für jede Aufgabe einzeln programmiert zu werden. Er beschrieb den historischen Kontext von KI, einschließlich früher Entwicklungen und der Wiederbelebung des neuronalen Rechnens, das darauf abzielte, menschliche Gehirnfunktionen nachzubilden. Der Artikel vertrat die Ansicht, dass echte KI den Forschern zwar bislang entgangen sei, sich die Technologie jedoch dahin entwickle, dass Maschinen durch Beispiele lernen könnten – und nicht mehr nur durch explizite Programmierung. [8]

Der Artikel ging auf neuronale Computer ein, die neuronale Netzwerke des menschlichen Gehirns nachbilden, und hob Fortschritte in Verarbeitungsgeschwindigkeit und Problemlösungsfähigkeiten hervor. Außerdem wurden die Herausforderungen beim Bau großer neuronaler Netze und die Einsatzmöglichkeiten dieser Technologien bei komplexen Aufgaben wie Spracherkennung und Musterverarbeitung angesprochen.

Rückblickend war dieser Artikel bemerkenswert vorausschauend, da er nicht nur die praktischen Aspekte beim Aufbau eines neuronalen Netzwerks behandelte, sondern auch auf potenzielle Verbesserungen der KI durch neuronales Rechnen einging.

Simuliertes Sehen in Robotern (Mai 1988)

In derselben Mai-1988-Ausgabe veröffentlichte Arthur Fryatt „Simulating Sight in Robots“, was heute unter „Edge Computing“ fallen würde. Mehr eine Nachricht als ein Praxisartikel, beschreibt der Artikel ein Vision-Sensing-System für die Farbkontrolle bei der Sortierung von Obst und Gemüse im Autoselector, einer gemeinsamen Entwicklung des Essex Electronics Centre (einer Abteilung der Universität Essex) und Loctronic Graders.

Ihre Zusammenarbeit führte zunächst zur Einführung des Autoselector A, der eine monochrome Fernsehbildtechnik zur Erkennung von Graustufenunterschieden einsetzte. Mit der Einführung des Autoselector C wurde dann ein sehr bedeutender Fortschritt bei der Farbbildgebung erzielt, der es ermöglichte, bis zu 4096 Farben und Farbtöne in Bereichen mit nur 3 mm Durchmesser bei sehr hoher Geschwindigkeit zu unterscheiden.

Da das gesamte Produkt gescannt werden musste, entwickelte Loctronic Graders das Thrudeck, das ständig rotierende Produkte wie Tomaten, Zwiebeln, Kiwis oder Zitrusfrüchte mit Geschwindigkeiten von bis zu 2500 Stück pro Minute vor die Kamera brachte. Selbst wenn die Produkte unregelmäßig geformt waren, konnte das System jedes einzelne auf seinem ungleichmäßigen Weg über das Deck verfolgen, vermessen und zählen. [9]

Intelligenz, Intentionalität und Selbstbewusstsein (1989)

Diese philosophische Betrachtung ging tiefer darauf ein, was es wirklich bedeutet, wenn eine Maschine intelligent ist. Der Artikel beleuchtete Konzepte wie Intentionalität und Selbstbewusstsein – Fragen, die auch heute noch im Mittelpunkt der KI-Debatte stehen. [10]

Computer lernen aus menschlichen Fehlern (1990)

Zum Abschluss des Jahrzehnts untersuchte dieser Artikel, wie Computer ihre Leistung durch das Lernen aus Fehlern verbessern können – ein Konzept, das später grundlegend für maschinelle Lernalgorithmen werden sollte. [11]

49

SCIENCE & TECHNOLOGY

COMPUTERS LEARN FROM HUMAN MISTAKES

by R.A.J. Arthur

When a system fault develops on an earth satellite, a human controller may find it easy to correct. On an unmanned space vehicle, one fault exceeding the limited liability of an automatic control correction system can write off an asset valued in billions of pounds.

Now, however, the Turing Research Institute in Glasgow has produced a remarkable remedy, and it could also have applications in general industry. Professor Donald Michie, chief scientist at the institute, said: "At last we have opened the door to transferring cognitive skills from humans to machines."

He described the method as "like being able to take an X-ray photograph of a cognitive skill". A satellite equipped with this transplanted human expertise would be able to get itself out of trouble.

The institute is named after the late Alan M. Turing, the British mathematician and logician whose work had a lasting influence on the foundations of modern computing. It shares premises, and is closely linked, with the Scottish Human Computer Interaction (HCI) Centre, part of the University of Strathclyde. The executive director of both is Professor James Alty.

Human computer interface

The institute's unsurpassed on-line Artificial Intelligence (AI) library of 55,000 documents is the only one of its kind available electronically from remote sites.

The objective of the HCI Centre is to develop the greatly improved human computer interfaces that alone can make AI systems acceptable.

Professor Michie's approach aims to overcome a potentially dangerous impasse. Conventional automatic systems, as widely used in industry, may still encounter the unexpected but lack the adaptive ability to deal with it.

If a pedal comes off a bicycle, a human cyclist will reorganize a strategy for keeping going, but a robot cyclist will almost certainly fall off. Although human beings are not good at any one task, they are very good at re-learning on the job. The ability of machines to study human trial-and-error learning and extract a set of rules defining a successful strategy is the basis of the new technique. Machine learning is used as a tool for digging law-governed regularities out of data.

Learning process recorded

A machine's ability to learn from human examples is demonstrated in a computer-simulated exercise—balancing a pole on a cart free to move along a short piece of track without letting the pole fall over or the cart run off either end.

HOTOL (Horizontal Take-off and Landing Aircraft), the unmanned orbital craft for which the Turing Institute has researched a new method of control if a fault occurs.

Various human operators built up skill at this game while the machine recorded the entire learning process. From the record of these attempts it extracted a series of generalized 'rules' for success. This experiment was used simply to demonstrate the absolute reliability of the method.

The difficulty in designing an automatic attitude controller for systems on satellites is formidable. Machine simulation falls far short of handling so complex a task. Simulators always represent a large simplification. When pilots are trained on them for unstable craft such as helicopters, it is impossible to turn the trainee loose on the real thing without a period of conventional human instruction.

In the case of an unmanned orbital craft such as the HOTOL, there is no question of human control, nor can a perfect automatic controller be designed by mathematical means. However, the new method opens a possible route.

Behaviour examples

First, a human operator is trained as far as he can go on the simulator. Then, the machine's learning programs are allowed to ruminate over the record of his behaviour. From that perhaps messy and inconsistent record they distil the essence of what his strategy is likely to be.

The machine learns from examples of behaviour and this overcomes a big problem. A top practitioner carries so much of his skill in his subconscious memory that his verbal accounts of method are wholly inadequate. Extracting general rules from actual examples, the machine constructs its own standard strategy far better than the human one because human inconsistencies have been smoothed out.

The aim is to generate a streamlined strategy far superior to what a human being would develop on a satellite. Observation shows that the streamlined strategy of the machine is strikingly better than the original human example.

The close cooperation of the Turing Institute with the Scottish HCI Centre has already produced remarkable results. As the era of Artificial Intelligence develops, this kind of expertise will make its impact in most fields of human activity. ■

NEW PUBLIC CORDLESS TELEPHONE SYSTEMS

by John Williamson

In the 1980s, mobile communication has become one of the most dynamic sectors of the telecommunications industry and, as is widely accepted, the United Kingdom has established an international lead in the technological development and commercial exploitation of a number of mobile services.

Thanks to the introduction of private sector competition in the supply of service and equipment, Britain's two cellular telephone networks are individually among the world's largest.

The operators Cellnet and Racal Vodafone had a combined subscriber base of about 800,000 early this year, and conservative estimates put the total growth rate at 30,000 new customers a month.

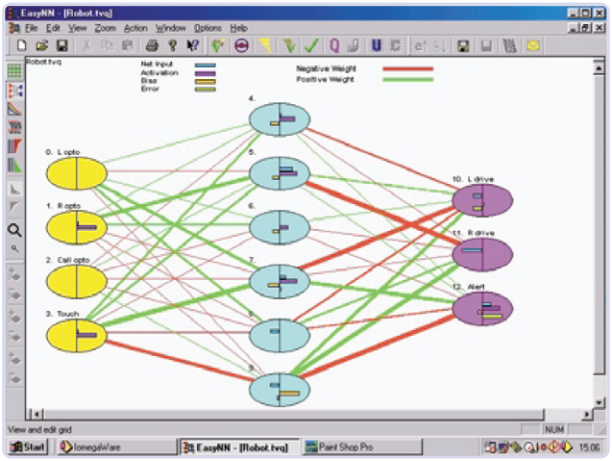
Local market volumes and competitive

ELEKTOR ELECTRONICS JULY/AUGUST 1990

Die Renaissance der neuronalen Netze (2001-2003)

Neuronale Netze in der Steuerung (2001)

Mit dem Beginn des neuen Jahrtausends rückten neuronale Netze erneut ins Interesse. Dieser Artikel untersuchte ihre Anwendung in Steuerungssystemen und zeigte, wie von der Biologie inspirierte Rechenmethoden komplexe ingenieurtechnische Probleme lösen können. [12]



Praxisreihe neuronale Netze (2003)

Diese umfassende Serie von Chris McLeod und Grant Maxwell, Absolventen der Robert Gordon University Aberdeen, brachte neuronale Netze in den Bereich von Amateur-Enthusiasten. Die Reihe erschien in vier Teilen:

- Eine Einführung in neuronale Netze [13]
- Backpropagation-Netze [14]
- Rückkopplungsnetze und kompetitive Netze [15]
- Anwendungen und große neuronale Netze [16]

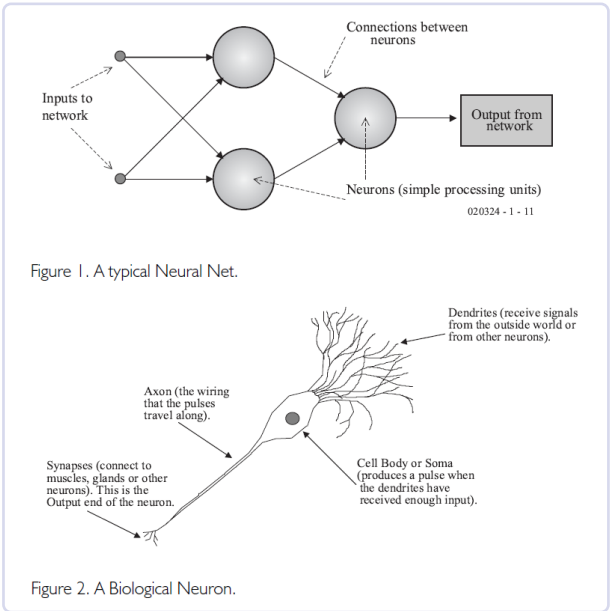


Figure 1. A typical Neural Net.

Figure 2. A Biological Neuron.

„Künstliche neuronale Netzwerke (kurz neuronale Netze oder ANN)", so beginnt die Serie, „sind eine populäre Form von künstlicher Intelligenz (KI). Sie basieren auf der Arbeitsweise von Gehirnzellen, und viele Forscher halten sie für unsere größte Hoffnung,

wahre Intelligenz in einer Maschine zu erreichen. Wer ein Fan der TV-Serie *Star Trek* ist, weiß, dass Data angeblich ein neuronales Netz als Gehirn hat, ebenso wie der Roboter in den *Terminator*-Filmen. Obwohl diese technologischen Wunder an der Spitze der Informatikforschung stehen, sind sie auch für engagierte Amateure gut zugänglich. Ziel dieser Artikel ist es, dieses faszinierende Thema auf praktische Weise vorzustellen, sodass Sie mit Ihren eigenen neuronalen Netzen experimentieren können."

Die Autoren schlussfolgerten in Teil 4, dass „das neuronale Netz als universelles logisches System betrachtet werden kann, das (vorausgesetzt, das Netz besteht aus drei Schichten) in der Lage ist, jede gewünschte Wahrheitstabelle zu erlernen. Wenn die Ausgabe der Neuronen eine Sigmoidfunktion ist, verhält es sich ähnlich wie *Fuzzy Logic* und kann analoge Ausgaben erzeugen – was nützlich sein kann, um Probleme aus der realen Welt zu behandeln, die nicht nur schwarz oder weiß sind."

Bemerkenswert ist, dass die Serie von einer Kursunterstützungs-Website der Robert Gordon University begleitet wurde, wobei diese Website inzwischen aus dem Netz verschwunden ist.

Rückblick und Ausblick

Besonders bemerkenswert an dieser Artikelsammlung aus fast drei Jahrzehnten ist, wie viele dieser zentralen Konzepte bis heute aktuell geblieben sind. Von Kybernetik und Rückkopplungssystemen über neuronale Netze bis hin zum maschinellen Lernen aus menschlichen Fehlern – die Elektor-Autoren erkundeten schon damals Ideen, die später die Grundlage der modernen KI bilden sollten.

Die im Zusammenhang mit dem Schachcomputer gestellte Frage, ob das Befolgen programmierter Strategien „Intelligenz" bedeutet, findet sich auch in heutigen Debatten um große Sprachmodelle wieder. Die Vision-Systeme zur Obstsortierung waren ein Vorläufer heutiger Anwendungen der Computer Vision. Die Serie zu neuronalen Netzen hat die Deep-Learning-Revolution vorweggenommen, die Jahrzehnte später die KI umkrempeln sollte.

Vielleicht am wichtigsten: Diese Artikel verfolgten stets einen praxisnahen Ansatz und ermutigten die Leser, eigene intelligente Systeme zu bauen und zu experimentieren. In einer Zeit, in der KI oft als geheimnisvoll und unzugänglich erscheint, erinnert das Elektor-Erbe daran, dass Verständnis durch Praxis entsteht – und dass der Weg von „künstlerischer" zu künstlicher Intelligenz von Neugier, Experimentierfreude und einem Augenzwinkern bei unseren ersten Versuchen, denkende Maschinen zu schaffen, geprägt ist.

Die Frage bleibt: Waren diese frühen Systeme wirklich intelligent oder nur sehr clever programmiert? Dass wir uns diese Frage auch heute noch stellen, zeigt vielleicht, dass der Weg zur künstlichen Intelligenz stets eher darin bestand, die richtigen Fragen zu stellen, statt endgültige Antworten zu finden.

250736-02

WEBLINKS

[1] H. Ritz, „Was ist Kybernetik? (Definition)", Elektor 1/1974: https://www.elektormagazine.de/magazine/elektor-197401/55408
[2] J. Kuipers, „Intelektor (Schachcomputer, 8088)", Elektor 4/1981: https://www.elektormagazine.de/magazine/elektor-198104/46241
[3] Jan Buiting, „Intelektor: Der Elektor-Schachcomputer (1981)", Elektor 3-4/2020: https://www.elektormagazine.de/magazine/elektor-140/57122
[4] Hans P. Baumann, „Computer und Sprache, Teil 1 (Spracherzeugung)", Elektor 4/1981: https://www.elektormagazine.de/magazine/elektor-198104/46239
[5] Hans P. Baumann, „Computer und Sprache, Teil 2", Elektor 6/1981: https://www.elektormagazine.de/magazine/elektor-198105/46253
[6] Prof. Margaret A. Boden, „The Future for Artificial Intelligence", Elektor 2/1986: https://elektormagazine.com/magazine/elektor-198602/46891
[7] „Speech Recognition System", Elektor 5/1987: https://elektormagazine.com/magazine/elektor-198705/47131
[8] M. Seymour, „Artificial Intelligence", Elektor 5/1988: https://elektormagazine.com/magazine/elektor-198805/47340
[9] Arthur Fryatt, „Simulating Sight in Robots", Elektor 5/1988: https://elektormagazine.com/magazine/elektor-198805/47341
[10] Dr. T. Farrimond, „Intelligence, Intentionality, and Self-Awareness", Elektor 11/1989: https://elektormagazine.com/magazine/elektor-198911/47684
[11] R. A. J. Arthur, „Computers Learn from Human Mistakes", Elektor 7-8/1990: https://elektormagazine.com/magazine/elektor-199007/32158
[12] Owen Bishop, „Computer mit Nervenzellen", Elektor 11/2001: https://www.elektormagazine.de/magazine/elektor-200111/1292
[13] Chris McLeod & Grant Maxwell, „Einführung in neuronale Netze", Elektor 3/2003: https://www.elektormagazine.de/magazine/elektor-200303/1577
[14] Chris McLeod & Grant Maxwell, „Back Propagation bei neuronalen Netzen", Elektor 4/2003: https://www.elektormagazine.de/magazine/elektor-200304/1598
[15] Chris McLeod & Grant Maxwell, „Rückkopplungsnetze und Wettbewerbs-Netze", Elektor 5/2003: https://www.elektormagazine.de/magazine/elektor-200305/1608
[16] Chris McLeod & Grant Maxwell, „Anwendungen und große neuronale Netzwerke", Elektor 6/2003: https://www.elektormagazine.de/magazine/elektor-200306/1620

Arduino Opta PLC: Dual-System-Industrietool

Arduinos Einstieg in Industrie-Controller bringt Intelligenz in Ladder Logic

Von Eoin Jordan (Edge Impulse)

Dual-System-Boards lösen Debatten und Kontroversen aus, eröffnen Entwicklern jedoch zugleich die Möglichkeit, Entwürfe neu zu bewerten, die zuvor nicht realisierbar waren. Ein Beispiel ist der Arduino Opta PLC, der Echtzeit-Steuerung unter rauen Einsatzbedingungen bereitstellt und zugleich Inferenz direkt auf dem Gerät sowie Anomalieerkennung mit Edge Impulse unterstützt.

Hybride Boards kombinieren zwei verschiedene Prozessorklassen – typischerweise einen Mikrocontroller zusammen mit einem leistungsfähigeren Applikationsprozessor – auf einer einzigen Leiterplatte. Sie sind meist spezialisiert und bekommen aktuell wenig Aufmerksamkeit. Ein Beispiel: In Jeff Geerlings umfassender Rezension des Arduino UNO Q [1] bezeichnet er das Board als „ein seltsames hybrides SBC … als wenn man eine Intel-CPU und einen Raspberry-Pi-RP2040-Mikrocontroller verheiraten würde". Geerlings Sorge? Solche hybriden Designs können mehr Strom verbrauchen, bringen Softwarekomplexität mit sich und verwischen die Grenze zwischen einfachen Mikrocontrollern und vollwertigen Applikationssystemen. Außerdem merkt er an: „Es fühlt sich immer noch so an, als verwaltet man zwei getrennte Dinge." Das sind berechtigte Punkte, und sein Beitrag bleibt eine der besten technischen Übersichten zum Gerät.

Doch es gibt noch eine andere wichtige Perspektive: Diese neuen hybriden Architekturen ermöglichen es Entwicklern, Entwürfe zu realisieren, die bisher nicht möglich waren. Der Arduino Opta PLC (**Bild 1**), ein weiteres hybrides Dualsystem, zeigt das sehr anschaulich. Als ich begann, den Opta gemeinsam mit meinem Kollegen Raul James aus unserem Embedded-Team zu erkunden, dessen tiefgehendes PLC-Know-how sich als äußerst wertvoll erwies, stellten wir fest, dass er Entwurfsmuster ermöglichte, die bisher nicht praktikabel waren. Beispielsweise lässt sich durch das Teilen von Inferenz-Ergebnissen über eine globale Variable und die Aufteilung des MCU-Speichers zwischen SPS-Logik und C++ mit dem Opta eine Echtzeitsteuerung realisieren, während gleichzeitig On-Device-Inferenz und Anomalieerkennung mit Edge Impulse möglich sind.

Genau diese Fähigkeit berücksichtigen die Kritiker nicht, und sie zeigt, worauf diese sogenannten „seltsamen" Architekturen hinauslaufen: Richtig eingesetzt, können sie Integrationsbarrieren überwinden. Schauen wir uns SPS (PLCs) an – ein Schlüsselfaktor für Hardware, die besonders von Funktionen profitiert, die über klassische Embedded-Entwicklung hinausgehen.

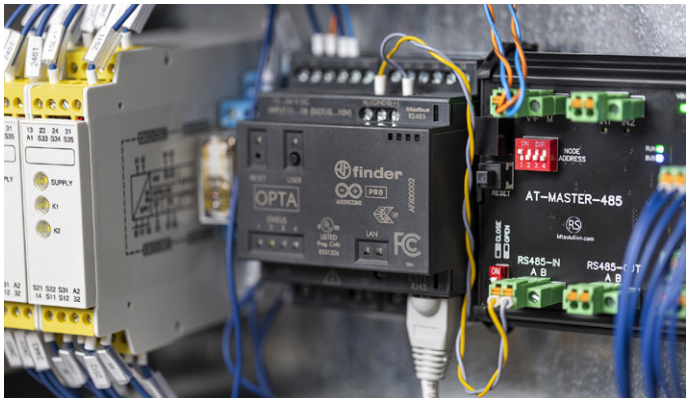


Bild 1. Arduino Opta PLC – ein duales Industrie-PLC- und MCU-Gerät in einem. (Quelle: Arduino)

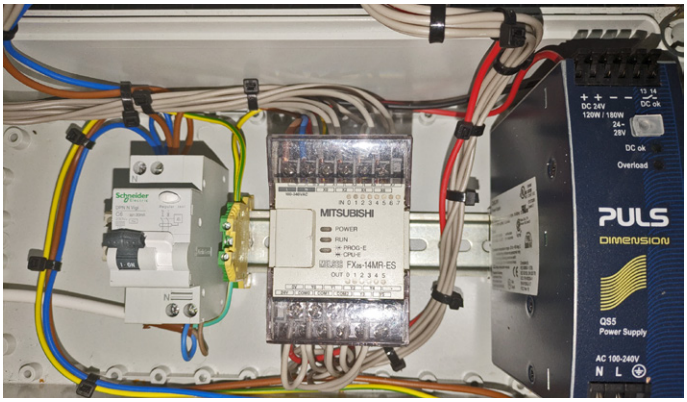


Bild 2. Eine moderne SPS: Mitsubishi FX-14MR-ES und ein PULS-Q55-Netzteil. (Quelle: Michael Madden, biomedizinischer Techniker)

Bild 3. Arduino Opta PLC (speicherprogrammierbare Steuerung). (Quelle: Arduino)

Eine kurze SPS-Einführung

Speicherprogrammierbare Steuerungen (SPS, engl. programmable logic controllers, PLCs) sind spezialisierte Industriecomputer, die häufig zur Automatisierung von Prozessen wie dem Steuern von Maschinen, Produktionslinien und anderen industriellen Systemen verwendet werden. Sie sind gebaut, um rauen Umgebungen standzuhalten – wie extremen Temperaturen, Feuchtigkeit und elektrischen Störungen – und eignen sich daher ideal für Echtzeitsteuerung unter anspruchsvollen Bedingungen. Ingenieure und Techniker aus Produktion, Energie und Logistik setzen SPS ein, weil sie einfach zu programmieren, langlebig, modular, äußerst zuverlässig (laufen jahrelang fehlerfrei) sind und sich direkt mit Industriesensoren und Aktoren verbinden lassen, um präzise Steuerungen zu ermöglichen (**Bild 2**).

Im Gegensatz zu klassischen Mikrocontrollern sind SPS für industrielle Echtzeitanwendungen konzipiert und unterstützen Programmiersprachen wie die sogenannte Kontaktplan-Programmierung (engl. „Ladder Logic" oder „Ladder Diagram", LD). Aufgrund der begrenzten Rechenleistung waren diese Controller traditionell nicht mit KI verbunden, aber durch Edge-KI eröffnen sich zahlreiche neue Anwendungsfälle, etwa Inferenz direkt auf dem Gerät, kombiniert mit fortschrittlicher digitaler Signalverarbeitung.

Vorstellung des Arduino Opta PLC

Der Arduino Opta PLC (**Bild 3**), entwickelt in Zusammenarbeit mit den italienischen Automatisierungsspezialisten Finder S.p.A., bringt das Arduino-Ökosystem in anspruchsvolle Industrieumgebungen. Entwickelt für SPS-Ingenieure, unterstützt er Kontaktplan-Programmierung und Funktionsblockdiagramme (FBD) [2] und kombiniert robuste Industrie-I/O, Langlebigkeit und Sicherheit mit der Flexibilität der Arduino IDE für Embedded-C++.

Schlüsselfunktionen des Arduino Opta

- Industrietaugliche Zuverlässigkeit: für anspruchsvolle Industrieumgebungen ausgelegt
- Optionale Konnektivität: integriertes Ethernet, WLAN und BLE
- Benutzerfreundlichkeit: Unterstützt Standard-PLC-Programmiersprachen und die Arduino-IDE.
- Edge-KI-fähig: Unterstützt Edge Impulse für Inferenz direkt auf dem Gerät.

Kurze Fakten zu Ladder Logic

Ladder Logic ist die bevorzugte Programmiersprache für SPS und ermöglicht es Ingenieuren, visuelle Darstellungen von Steuerungsschaltungen zu erstellen. Sie vereinfacht Automatisierungsprozesse, indem sie physikalische Relaislogik abbildet und damit für Fachleute zugänglich bleibt. Ladder Logic nutzt horizontale Linien, engl. *Rungs*, wie in **Bild 4** zu sehen, die logische Steuerungsanweisungen repräsentieren. Sie bleibt die De-facto-Programmiermethode für SPS, weil sie leicht zu visualisieren und zu debuggen ist und Technikern vertraut erscheint.

Allein mit Ladder Logic lassen sich jedoch Aufgaben wie Inferenz und Anomalieerkennung nicht umsetzen. Hier wird die Dual-Core-Architektur des Opta besonders interessant.

Dual-Core-Partitionierung

Die Dual-Core-Architektur des Opta kann so partitioniert werden, dass ein Kern eine Anwendung (wie z. B. einen C++-Edge-Impulse-Inferenz-Sketch) ausführt, während der andere über die Arduino PLC IDE deterministische Ladder-Logic-Programme abarbeitet. Diese Trennung bewahrt das SPS-Verhalten und ermöglicht gleichzeitig Anwendungen wie On-Device-Edge-KI (**Bild 5**).

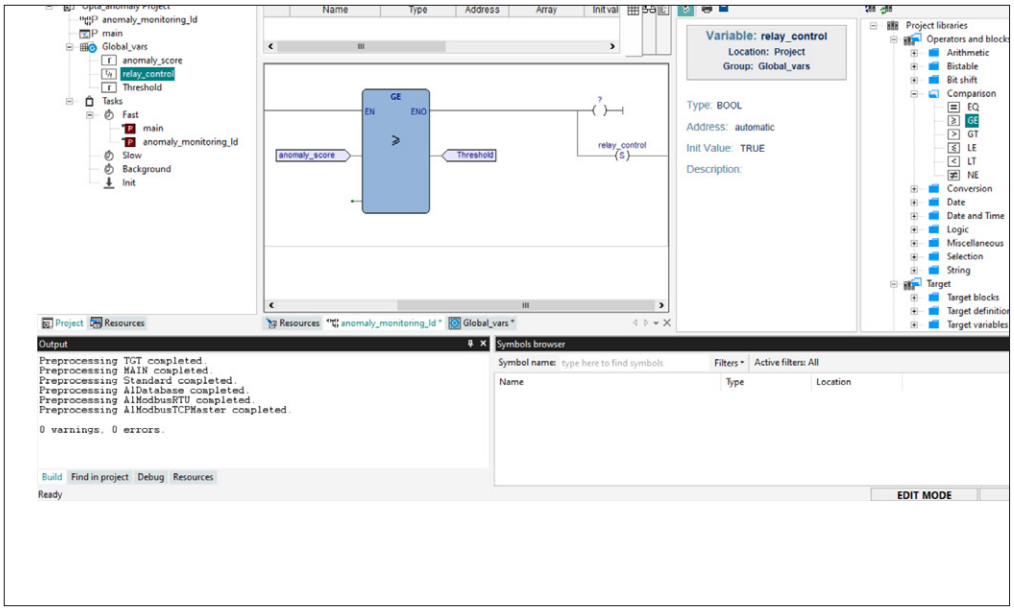


Bild 4. Ladder Logic mit einer Anomalieerkennungs-Rung (Arduino PLC IDE).

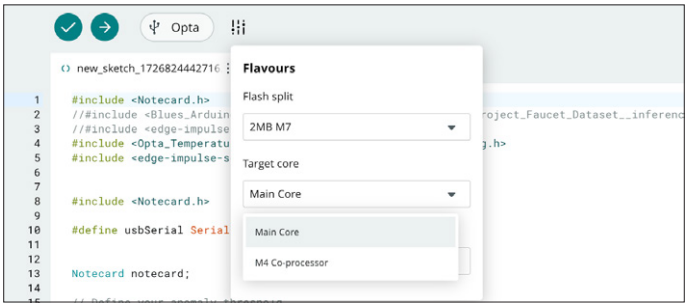


Bild 5. Flash-Partitionierung und Core-Zuweisung, bereitgestellt über die Arduino Cloud IDE.

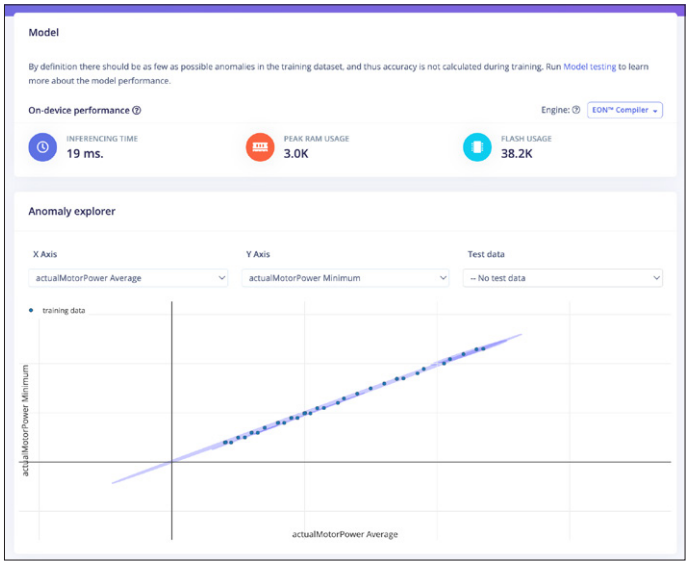
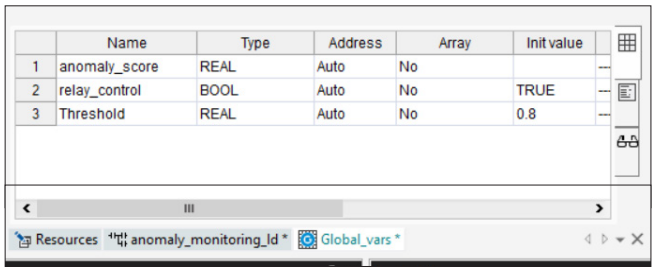


Bild 6. Gaußsches Mischmodell (GMM) – eines der Clustering-Verfahren bei Edge Impulse.



	Name	Type	Address	Array	Init value
1	anomaly_score	REAL	Auto	No	
2	relay_control	BOOL	Auto	No	TRUE
3	Threshold	REAL	Auto	No	0.8

Bild 7. Gemeinsame Variablen – globale Variablentabelle (Arduino PLC IDE).

SPS und Anomalieerkennung

Edge-KI kann SPS erweitern, indem maschinelles Lernen eingesetzt wird, um Zustände zu überwachen (Klassifikation), vorausschauende Wartung zu unterstützen (Regression) oder Fehlerbilder (Anomalieerkennung) in Industriesensoren wie Wägezellen oder in Motorfeedbacks zu erkennen. Anomalieerkennung benötigt besonders wenig Rechenleistung, läuft auf MCUs und ist für industrielle Anwendungen mit SPS von grundlegender Bedeutung.

Anomalieerkennung ist ein leistungsstarker, aber ressourcenschonender Algorithmus aus dem maschinellen Lernen, der Muster identifiziert, die vom normalen Systemverhalten abweichen. Statt auf gelabelte Fehlerdaten zu setzen, wird das Modell nur mit „normalen" Betriebszuständen trainiert. Während des Trainings clustert ein Gaußsches Mischmodell (GMM) diese typischen Muster (**Bild 6**). Werden später abweichende Daten erkannt, liefert das System einen höheren Anomalie-Score. Ist das Edge-Impulse-Modell einmal (als Arduino-Bibliothek) bereitgestellt, kann der Runtime-Sketch den `anomaly_score` in die globale Variablentabelle der SPS aus **Bild 7** schreiben.

Dieser Score kann als Brücke zwischen maschinellem Lernen und der SPS-Steuerlogik dienen; er kann in eine geteilte globale Variable geschrieben und direkt in Ladder Logic ausgelesen werden, um bei anomalen Werten Alarme, Maschinenstopp oder Schutzroutinen auszulösen. Nach dem Training und Deployment des Modells lässt es sich als Arduino-Bibliothek für den Sketch der Arduino-Partition nutzen.

Wenn der Score einen Schwellenwert überschreitet, kann die Logik die Variable `relay_control` umschalten, um die Produktion zu stoppen oder einen Alarm auszulösen (z. B. LED einschalten oder Relais aktivieren). Wie das in einem Sketch aussieht, zeigt **Bild 8**.

Mit integriertem Modell kann man Inferenz oder Anomalieerkennung auf den von Motor oder Sensor erfassten Daten ausführen, und der Opta PLC führt ein Ladder-Logic-Programm aus, das die geteilte Variable `anomaly_score` liest. Die Logik kann dann auf Grundlage dieses Wertes auf erkannte Anomalien reagieren.

Vorteile der Kombination aus Ladder Logic und Anomalieerkennung

- Echtzeit-Steuerung: Ermöglicht Echtzeit- und intelligente Steuerungssysteme, die auf datenbasierte Entscheidungen des Edge-Impulse-Modells zugreifen.
- Einfaches Troubleshooting: Bietet eine klare, visuelle Darstellung der Steuerlogik, sodass sich das Systemverhalten und seine Reaktionen auf Anomalien einfacher überwachen und debuggen lassen.
- Nahtlose PLC-Integration: Die Arduino-PLC-IDE bietet eine durchgängige Umgebung für die Kombination klassischer Steuerlogik mit moderner Machine-Learning-Funktionalität, sodass Kompatibilität und Bedienkomfort gewährleistet sind.

Neue Optionen für die industrielle Automatisierung

Industriemaschinen kosten oft Hunderttausende Euro in der Entwicklung, und Ausfallzeiten sind teuer für Produktionslinien. Selbst wenn man Datenschutz und IP-Bedenken außer Acht lässt, ist Konnektivität schlicht nicht immer gegeben. Mit On-Device-Machine-Learning können Ingenieure jedoch Algorithmen integrieren, um Bediener auf zuvor beobachtete Zustände oder Anomalien hinzuweisen. Mit Edge Impulse lässt sich KI direkt auf bestehenden, von SPS gesteuerten Maschinen ausführen und als Expertenwissen nutzen – ganz ohne komplexe Netzwerke.

Industriemaschinen operieren im Bereich Edge-Computing; in puncto Konnektivität sind diese Systeme meist nur teilweise, wenn überhaupt, vernetzt.

Erwähnenswert ist auch, dass Sie mit Edge Impulse bestehende Inhouse-Systeme weiterentwickeln oder maßgeschneiderte Lösungen für jede industrielle Anforderung – für Einzelmaschinen und unter fertigungsspezifischen Bedingungen – konzipieren können.

Weiterführende Möglichkeiten

Unser begleitendes Opta-PLC-Tutorial [3] zeigt Schritt für Schritt, wie Sie Anomalieerkennung auf einem Arduino Opta PLC mit Edge Impulse implementieren und integrieren können.

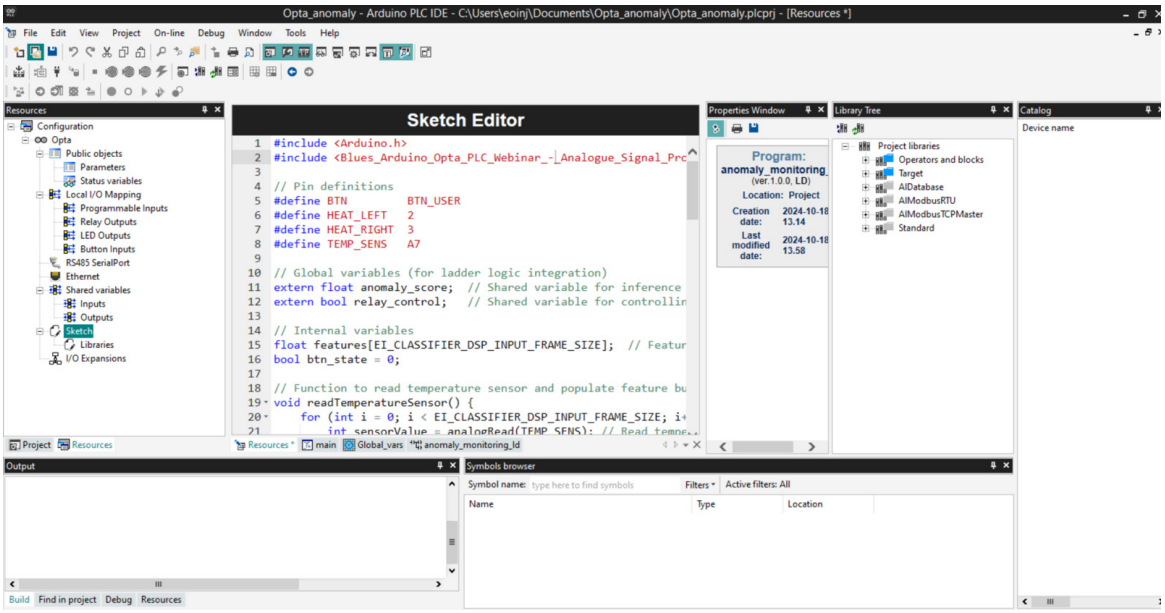


Bild 8. Ladder Logic – Sketch-Integration (Arduino PLC IDE).

Für Praxisbeispiele schauen Sie sich unser Webinar [4] an, in dem Edge Impulse, der Arduino Opta PLC und IDT kombiniert werden, um eine industrielle Architektur zu demonstrieren – mit Beispielen aus der Automobilteilefertigung von IDT.
Das Webinar bietet auch eine praxisnahe Einführung in eine weitere Methode, um Systeme wie den Opta mit anderen verteilten Geräten zu vernetzen, die Edge Impulse-Modelle nutzen, etwa den Nicla Sense ME. Die iterative Natur des Entwicklungsprozesses ist Teil des Modell-lebenszyklus-Managements. Diese Workflows zur Modellintegration, Überwachung und Bereitstellung finden Sie in unseren speziellen Lifecycle-Management-Dokumenten 5].

Deterministische SPS-Steuerung plus moderne KI

Der Opta PLC ist eines dieser seltsamen und zugleich faszinierenden Geräte, die zeigen, dass „anders" kein Nachteil ist. Es ist inspirierend zu sehen, wie Hardware wie diese bisher undokumentierte Möglich-keiten eröffnet, deterministische SPS-Steuerung und moderne KI in einem einzigen, nahtlos integrierten Industriecontroller zu vereinen.
Durch die Kombination klassischer Ladder Logic mit einer C++-Laufzeit, die Edge-Impulse-Modelle ausführt, bringt der Opta deterministische Steuerung und adaptive Intelligenz in ein einziges Gerät – ohne externe Gateways, Industrie-PCs oder Cloud-Anbindung.
Die dazugehörige Arduino PLC IDE erleichtert die Integration von maschinellem Lernen in Ladder Logic über geteilte Variablen und ermöglicht On-Device-KI in bestehenden Industrieprozessen mit minimalem Aufwand.

250882-02

Über den Autor

Eoin Jordan arbeitet im Developer-Relations-Team von Edge Impulse und verfügt über mehr als 13 Jahre Erfahrung in den Bereichen Networking, Cloud, Edge und IoT. Er engagiert sich leidenschaftlich für Edge-Intelligenz und promoviert derzeit in diesem Fachgebiet, während er die Community zu diesem wachsenden Themenfeld unterstützt. Er wirkte an Entwicklung und Implementierung der Traffic Analysis Hub Ontology (TAHO) [6] mit, einem semantischen Modell zur Darstellung und Analyse komplexer Finanz- und Anti-Trafficking-Datensätze innerhalb lokaler Bankinfrastrukturen.

WEBLINKS

[1] "The Arduino Uno Q is a weird hybrid SBC," Jeff Geerling blog, 2025: https://www.jeffgeerling.com/blog/2025/arduino-uno-q-weird-hybrid-sbc
[2] "Introduction to Programming in PLC IDE with Finder OPTA," Finder S.p.A.: https://tinyurl.com/plc-programming-introduction
[3] "Temperature anomaly detection on Opta PLC," Edge Impulse tutorial: https://docs.edgeimpulse.com/tutorials/end-to-end/temp-anomaly-detection-opta-plc
[4] "Accelerating Industrial Automation with Arduino and Edge Impulse," Edge Impulse webinar, 2025: https://edgeimpulse.com/all-events/accelerating-industrial-automation-with-arduino-and-edge-impulse
[5] "What is edge MLOps?," Edge Impulse course: https://docs.edgeimpulse.com/knowledge/courses/edge-ai-fundamentals/what-is-edge-mlops
[6] Eoin Jordan on ResearchGate: https://www.researchgate.net/profile/Eoin-Jordan-4

Von den Daten bis zur Implementierung

Der Edge-Impulse-Workflow

Erstellen

Entwerfen Sie Signalverarbeitungs- und ML-Pipelines, die auf Ihre Daten und Ihr Gerät zugeschnitten sind.

> Impuls-Design

Sammeln

Sammeln und kennzeichnen Sie Bild-, Audio- oder Sensordaten, um einen repräsentativen, hochwertigen Datensatz zu erstellen.

> Daten-Explorer
> KI-Kennzeichnung
> Synthetische Daten

Überwachen

Erfassen Sie Live-Gerätedaten, bewerten Sie die Drift und verbessern Sie Ihre Modelle kontinuierlich im Einsatz.

> Modellüberwachung

EDGE IMPULSE

Edge Impulse vereint alle Schritte zur Erstellung von KI-Modellen für Edge-Geräte in einer benutzerfreundlichen End-to-End-Plattform. Ob Sie für Sensoren, Kameras oder Audiogeräte entwickeln – Edge Impulse ermöglicht es Ihnen, Daten zu erfassen, effiziente Modelle zu trainieren und diese nahezu überall bereitzustellen – von MCUs über MPUs bis hin zu GPUs und darüber hinaus. So läuft der Prozess innerhalb der Plattform ab.

Trainieren

Trainieren und bewerten Sie Modelle mit der integrierten Cloud-Rechenleistung, indem Sie Genauigkeits- und Leistungsmetriken verfolgen.

Optimieren

Finden Sie automatisch das ideale Gleich-gewicht zwischen Genauigkeit, Latenz und Speicherbedarf.

> EON Tuner
> EON Compiler

Bereitstellen

Erzeugen Sie einsatz-bereite Binärdateien oder SDKs für Ihre Edge-Geräte und andere Hardware-Ziele mit nur wenigen Klicks.

> Bereitstellungs-konfiguration

Legen Sie los!

Erstellen, trainieren und implementieren Sie Ihre eigene Edge-KI-Lösung mit Edge Impulse: schnell, leistungsstark und einfach.

GlobalSense setzt neue Maßstäbe in der Fahrzeugdiagnose mit Edge-KI

Quelle: Adobe Stock / Oleksandr

Von Mike Senese (Edge Impulse)

GlobalSense revolutioniert die Fahrzeugdiagnose durch KI-gestützte Geräuschanalyse zur Fehler- und Kollisionserkennung. Alltägliche Geräusche werden in aussagekräftige Daten verwandelt, um Lücken herkömmlicher Diagnosesysteme zu schließen.

GlobalSense ist ein in San Diego, Kalifornien, ansässiges Technologie-Startup, das einen neuen Ansatz für die Automobil-Diagnose und Ereigniserkennung entwickelt. Mitbegründer und CEO Dr. Saeid Safavi gründete GlobalSense im August 2021. Das Unternehmen verändert die Art und Weise, wie Geräusche als leistungsfähiges Werkzeug genutzt werden können, um tiefere Bedeutungen aus gewöhnlichen Ereignissen zu extrahieren. Das erste Ziel ist es, die erheblichen Lücken herkömmlicher Diagnosewerkzeuge für Kraftfahrzeuge zu schließen, indem eine innovative Lösung zur Fehlerdiagnose und Kollisionserkennung mit einem neuen Onboard-Gerät entwickelt wird, das durch Audioerkennungs-Algorithmen unterstützt wird.

„Unser Fokus liegt auf akustischer Erkennung, die unserer Meinung nach einen neuen Sensormechanismus darstellt, der in vielen Branchen eingeführt werden kann", sagt Safavi.

Die Herausforderung: Geräuschklassifikation

GlobalSense konzentrierte sich zunächst auf die integrierte Erkennung von Ereignissen im Fahrzeug (Unfälle, Einbrüche usw.), die stark auf beschleunigungssensorgestützte Systeme angewiesen sind. Diese Systeme weisen jedoch häufig eine hohe Fehlerquote bei bestimmten Arten von Vorfällen auf. Safavi erklärt: „Diese Geräte basieren ausschließlich auf Beschleunigung und Verzögerung als einzigem Sensormechanismus und verpassen etwa 30 % der Kollisionen, einfach weil einige Kollisionen nicht in den Schwellenwert-Algorithmus passen, auf den sie angewiesen sind – wie Überschläge oder seitliche Kollisionen."

Techniker verlassen sich hingegen oft auf eine Mischung aus OBD2-Fehlercodes und persönlicher Erfahrung, um Probleme zu erkennen. Ein erfahrener Mechaniker kann ein Pleuellagerschlagen, Riemenschlupf oder Zündaussetzer einfach am Geräusch erkennen – etwas, das aktuelle digitale Werkzeuge nicht zuverlässig reproduzieren können. Diese Abhängigkeit von menschlicher Wahrnehmung hinterlässt eine große Diagnoselücke, insbesondere in Umgebungen, in denen Geschwindigkeit, Genauigkeit und Skalierbarkeit entscheidend sind – etwa bei Auktionen, Händlern und Fuhrparkbetreibern. Gleichzeitig kommen Bewertungs- und Schlichtungsprozesse im Automobilbereich häufig ins Stocken, weil Zustandsberichte unvollständig oder umstritten sind. Aktuelle Inspektionssysteme übersehen subtile, aber kostspielige Probleme, was zu unnötigen Rückgaben, Schlichtungsstreitigkeiten und Vertrauensverlust führt.

GlobalSense hat sich seit der Gründung auf die Fehlererkennung im Automobilbereich konzentriert – eine anspruchsvolle Aufgabe, die durch fehlende Daten, die Komplexität des Problems und die große Variabilität der akustischen Signaturen von Fahrzeugkomponenten wie

Quelle: Positioning Universal

Motoren, Getrieben und Zubehörteilen erschwert wird. Der gleiche Fehler kann bei unterschiedlichen Marken und Modellen völlig unterschiedlich klingen.

Lösung: Edge-AI + automatisierte Datenpipeline

Nach einigen anfänglichen Versuchen, eigene KI-Sequenzen zu entwickeln, entschied sich GlobalSense dafür, den umfassenden Workflow von Edge Impulse zur Erstellung von Edge-AI-Algorithmen zu nutzen, um die Entwicklung der erforderlichen geräuschbasierten Diagnosemodelle zu beschleunigen. Die Integration ermöglichte es GlobalSense, seine Modelle schneller und präziser als bei den anfänglichen Versuchen zu optimieren.

„Bevor Edge Impulse ins Spiel kam, musste man alles intern machen", sagt Safavi. „Der Zyklus vom Training der Daten bis zur Implementierung und zum Testen in der Hardware war deutlich länger."

Die Zusammenarbeit mit Edge Impulse ermöglichte es GlobalSense, seine geräuschbasierten Sicherheits- und Schutzmodelle für die Automobilindustrie produktreif zu machen. Infolgedessen hat einer der Kunden von GlobalSense, Positioning Universal, diese Technologie in seine Aftermarket-Ortungsgeräte integriert, die sich nun in der Serienproduktion für den Weltmarkt befinden.

Autosonix: Plattform zur Fehlerdiagnose

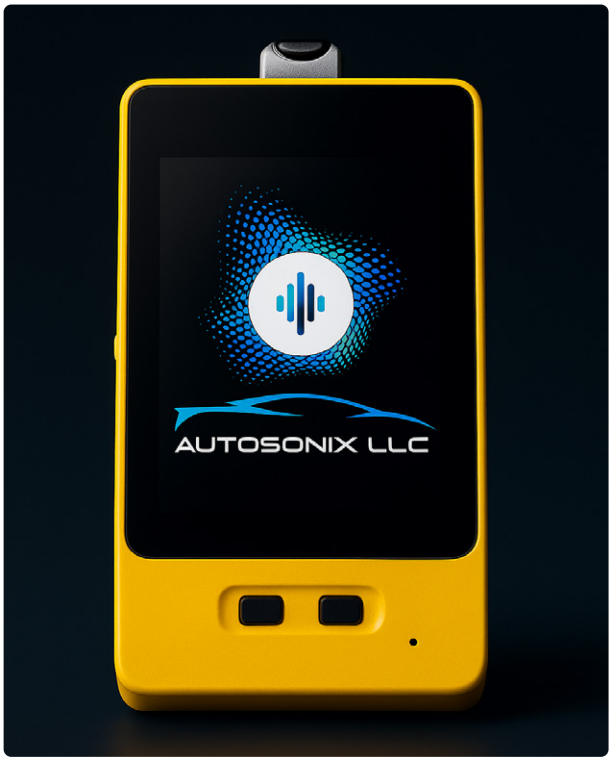
AutoSonix ist ein neues Produkt, das entwickelt wurde, um die langjährigen Herausforderungen der Fehlerdiagnose und Wartung im Automobilbereich zu lösen. „Um diese Herausforderungen zu bewältigen, brauchten wir mehr Daten, viel mehr davon", erklärt Safavi. Diese Hürde wurde durch eine Partnerschaft mit Condition Now, einem Dienstleister für Fahrzeugdaten, überwunden. Die Zusammenarbeit ermöglichte es GlobalSense, eine Infrastruktur für die großflächige Erfassung von Geräusch- und OBD-Daten in Kooperation mit mehreren Autoauktionen aufzubauen.

Diese Auktionen, die bereits die Dienste von Condition Now nutzen, stellten GlobalSense tägliche Aufnahmen aus Flotten von Tausenden Fahrzeugen zur Verfügung. Die Partnerschaft führte auch zur Optimierung der Datenerfassungsinfrastruktur, zur Entwicklung von Kundenportalen und vor allem zur Schaffung von Prozessen zur Erfassung und Aufbereitung von Daten in Formaten, die für Edge- und Cloud-KI-Engines am besten geeignet sind. Mobile Apps und Spezialwerkzeuge wurden ebenfalls entwickelt, um konsistente und hochwertige Dateneingaben zu gewährleisten.

„Mit diesem noch nie dagewesenen Volumen hochwertiger Daten von Condition Now konnten wir eine Plattform entwickeln, die in der Lage ist, spezialisierte Erkennungsmodelle zu trainieren und zu aktualisieren sowie Echtzeitverarbeitung zu unterstützen. Dadurch konnten Genauigkeit und Zuverlässigkeit erheblich gesteigert werden", sagt Safavi.

Aus dieser Zusammenarbeit entstand die Gründung von AutoSonix LLC, einer gemeinsamen Gesellschaft von GlobalSense und Condition Now. AutoSonix revolutioniert die Nutzung von Geräusch-, Bewegungs- und Onboard-Fahrzeugdaten, um verwertbare Erkenntnisse im großen Maßstab zu liefern. Das Ziel ist es, die Lücken herkömmlicher Diagnose- und Inspektionswerkzeuge zu schließen, indem akustische KI,

> *Die Edge-Impulse-Toolchain hat dazu beigetragen, unsere Markteinführungszeit für einige Produkte um eine Größenordnung zu verkürzen.*



Quelle: AutoSonix

OBD2-Daten und Edge Intelligence in einer schlanken Lösung vereint werden.

„Automobildiagnosen haben sich jahrzehntelang auf subjektive Intuition von Mechanikern oder unvollständige Scan-Tool-Daten gestützt. AutoSonix macht daraus einen objektiven, wiederholbaren und skalierbaren Prozess", ergänzt Safavi.

AutoSonix hat seine Plattform um Edge-KI herum aufgebaut, die direkt auf dem Gerät oder Mobiltelefon verarbeitet, um Echtzeit-Feedback, Datenschutz und Skalierbarkeit zu gewährleisten. Durch die Erfassung der akustischen Signatur eines Fahrzeugs, die Verknüpfung mit Bewegungsdaten von Smartphones und die Integration von OBD2-Telemetrie liefert AutoSonix objektive Diagnoseergebnisse in Sekunden, ohne dass ein Cloud-Rechenzentrum für die Erstbewertung erforderlich ist.

Dieser Edge-First-Ansatz reduziert nicht nur die Latenz, sondern gewährleistet auch den Datenschutz, da nur relevante Ereignisse und Signaturen übertragen werden – und keine Roh-Audioaufnahmen. Das System integriert sich nahtlos in die Arbeitsabläufe von Händlern und Auktionen über eine mobile App, sodass selbst ungeschulte Benutzer hochwertige Diagnosen mit minimalem Aufwand erstellen können.

Edge-Impulse-Integration

GlobalSense hat die Plattform von Edge Impulse eingesetzt, um einen schlanken, iterativen Ansatz für die Modellentwicklung und -bereitstellung zu schaffen. Als täglich Daten von versteigerten Fahrzeugen eintrafen, lud GlobalSense diese auf die Plattform hoch, um Modelle zu testen, zu verfeinern und basierend auf der tatsächlichen Leistung anzupassen. Die intuitive Oberfläche und die umfangreichen Funktionen von Edge Impulse ermöglichten es dem Team, Probleme schnell zu erkennen und zu beheben und so die Effizienz des Modelltrainingsprozesses erheblich zu steigern. Dazu gehörte auch die nahtlose Integration der eigenen Datenaufbereitungs- und Nachbearbeitungsalgorithmen sowie der eigenen KI-Modelle direkt in den Edge-Impulse-Workflow. Diese Integrationen werden nun zu einem wichtigen Bestandteil der AutoSonix-KI-Plattform und ihres nahtlosen Workflows.

Lokale Cloud-Unterstützung

Um die Edge-Fähigkeiten der AutoSonix-Plattform zu verbessern, arbeitet GlobalSense eng mit mimik Technology zusammen, dem Entwickler der Plattform Hybrid Edge Cloud (HEC), die es intelligenten Geräten ermöglicht, als lokale Cloud-Äquivalente für dezentralisierte, agentische, KI-gesteuerte Anwendungen zu fungieren. Diese Zusammenarbeit verspricht eine energiesparende, nahtlose Bereitstellung der AutoSonix-Edge-KI-Modelle auf Geräten wie Mobiltelefonen, OBD-Systemen und mehr, während der Datenschutz optimiert und der Bedarf an ständiger Cloud-Konnektivität eliminiert wird.

GlobalSense/AutoSonix nutzt eine Reihe von AWS-Diensten und -Ressourcen, darunter EC2, RDS, S3, Containerdienste, Kubernetes und Elastic Load Balancing. Diese Dienste unterstützen verschiedene Funktionen – von der Datenerfassung und -speicherung bis hin zur Bereitstellung von Diensten für aktuelle Kunden wie Autoauktionen.

Chip-Partnerschaft

Neben Edge Impulse arbeitet GlobalSense mit wichtigen Partnern aus der Technologie- und Automobilbranche zusammen, um die Datenerfassung und -kennzeichnung zu verbessern. Eine Partnerschaft mit Syntiant, einem KI-fokussierten Chiphersteller, stellte GlobalSense spezialisierte Hardware für die Verarbeitung von Audiodaten zur Verfügung und half, den Energieverbrauch der Modelle zu senken.

„Wir arbeiten seit über dreieinhalb Jahren mit Syntiant zusammen und haben deren Chip in unser Kundenprodukt integriert – mit dem Ziel, im Laufe der Zeit 2,5 Millionen Chips zu nutzen", sagt Safavi. „Es ist eine für beide Seiten vorteilhafte Partnerschaft – Syntiant profitiert von unserer Kundenbasis, und wir nutzen deren Technologie und Ressourcen, um unser Angebot zu verbessern."

Ergebnisse

Durch die Partnerschaft konnte GlobalSense damit beginnen, sein Kundenportfolio aufzubauen. Dazu gehören Hersteller von Aftermarket-Geräten für Autos, Autoauktionen, Händler und Versicherungen. GlobalSense ist in der Lage, über die Automobilbranche hinaus in andere Bereiche zu expandieren, in denen Audiodaten für vorausschauende Wartung und Anomalieerkennung eingesetzt werden können.

„Edge Impulse hat es uns ermöglicht, für einen Kunden potenziell mehrere Millionen Dollar pro Jahr einzusparen, indem wir die kleinstmöglichen Modelle bereitstellen und so unsere gesamten BOM-Kosten reduzieren", sagt Safavi. „Darüber hinaus hat die Edge-Impulse-Toolchain dazu beigetragen, unsere Markteinführungszeit für einige Produkte um eine Größenordnung zu verkürzen."

Mit der leistungsstarken Plattform von Edge Impulse hat GlobalSense eine skalierbare, leistungsfähige Lösung geschaffen, die eine entscheidende Lücke in der Automobilindustrie und darüber hinaus schließt.

Die nächsten Schritte

Während GlobalSense sich zunächst auf den Automobilsektor konzentriert – wo die Lösungen bereits Inspektionen, Schlichtung und Bewertung optimieren – reicht das Anwendungspotenzial der zugrunde liegenden Technologie weit über den Automobilbereich hinaus. Fuhrparkbetreiber, Speditionen und Anbieter von Industriemaschinen können von derselben geräusch- und bewegungsbasierten Diagnosetechnik profitieren.

Über den Automobilbereich hinaus richtet GlobalSense den Blick – oder das Ohr – auf verschiedene Branchen, darunter industrielles IoT, HLK, Gas- und Ölindustrie und mehr. Die bahnbrechende Arbeit mit geräuschbasierten Sensoren bringt eine völlig neue Perspektive in viele Bereiche und nutzt Edge-KI, um Diagnoseprozesse genauer, kostengünstiger und skalierbarer zu machen.

„Der Automobilbereich ist für uns der Einstiegspunkt, weil wir die Branche kennen", sagt Safavi. „Aber während wir Perfektion und Erfahrung in dieser speziellen Sensormechanik aufbauen, planen wir, sie auf industrielles IoT, HLK, Fehlererkennung oder andere Anwendungen mit beweglichen Teilen und Geräuschen auszuweiten."

Die Partnerschaft von GlobalSense und AutoSonix mit Edge Impulse ermöglichte es ihnen, eine bahnbrechende geräuschbasierte Diagnoselösung zu entwickeln, die der Automobilbranche echten Mehrwert bietet. Diese Zusammenarbeit zeigt das Potenzial von Edge-KI zur Transformation komplexer, datenintensiver Bereiche und bietet einen Einblick in die Zukunft der geräuschbasierten vorausschauenden Wartung in vielen Branchen.

250684-02

Analoge Messgeräte mit KI ablesen

Automatisierung klassischer Anzeigen mit Arduino Nicla Vision

Von Constantin Craciun (Rumänien)

Analoge Messgeräte sind in industriellen Umgebungen weit verbreitet, müssen jedoch manuell abgelesen werden und lassen sich nur schwer in digitale Systeme integrieren. Dieses Projekt zeigt, wie sich herkömmliche Anzeigen mit Computer Vision und Machine Learning modernisieren lassen. Es beschreibt die Erzeugung eines synthetischen Datensatzes, das Training eines kompakten ML-Modells und dessen Einsatz auf dem Arduino Nicla Vision zur automatischen Echtzeit-Überwachung. So entsteht eine kostengünstige Lösung, um bestehende Geräte zu digitalisieren, ohne die Originalhardware ersetzen zu müssen.

Analoge Messgeräte werden häufig in industriellen Umgebungen eingesetzt, um verschiedene Prozessvariablen wie Druck, Temperatur oder Durchfluss zu messen. In vielen Fällen werden analoge Anzeigen digitalen vorgezogen, da die meisten analogen Messgeräte, die an alten Maschinen angebracht sind, nicht einfach ersetzt werden können oder ein Austausch zu kostspielig wäre. Allerdings haben sie mehrere Nachteile, wie etwa die Notwendigkeit der visuellen Inspektion durch einen menschlichen Bediener und die Schwierigkeit, sie in digitale Systeme zur Automatisierung von Aufgaben zu integrieren.
Durch den Einsatz von Computer Vision und maschinellem Lernen können diese Nachteile überwunden werden, indem analoge Messgeräte mit digitaler Erfassung nachgerüstet werden. Computer-Vision-Systeme können automatisch Messwerte von analogen Anzeigen und Displays ablesen, wodurch das manuelle Ablesen und Aufzeichnen entfällt. Darüber hinaus ermöglicht diese Methode eine kontinuierliche Überwachung der analogen Werte in Echtzeit, was eine genauere Erkennung von Trends und Analysen ermöglicht, die Wartungszeit reduziert und definierte Warnmeldungen zur Vermeidung von Ausfällen erlaubt.

Unsere Lösung

In diesem Projekt zeigen wir Ihnen, wie Sie Computer Vision und maschinelles Lernen einsetzen können, um das Kesseldruck-Messgerät einer Heizungsanlage abzulesen. Wir verwenden das Arduino-Kameramodul Nicla Vision (**Bild 1**), um die Trainingsdaten zu erfassen und das

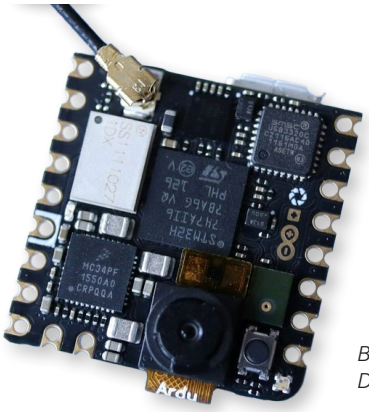


Bild 1.
Die Arduino Nicla Vision.

Projektanforderungen

Hardware-Anforderungen
Arduino Nicla Vision [1]
Micro-USB-Kabel

Software-Anforderungen
Edge-Impulse-Konto
OpenMV-IDE [2]

ML-Modell auszuführen, sowie die Edge-Impulse-Plattform, um ein Bildklassifizierungsmodell zu erstellen, zu trainieren und bereitzustellen. Die Nicla Vision, ein Gemeinschaftsprodukt von Arduino und OpenMV, ist die perfekte Wahl für diesen Anwendungsfall, da sie über einen leistungsstarken Prozessor mit einer Zwei-Megapixel-Farbkamera verfügt, TinyML unterstützt und sich problemlos in Edge Impulse integrieren lässt. Außerdem bietet sie Wi-Fi- und Bluetooth-Low-Energy-Konnektivität, sodass Sie Ihre Daten in die Cloud senden können, ohne ein weiteres Entwicklungsboard verwenden zu müssen. Und all diese Funktionen sind auf einer wirklich kleinen Platine mit weniger als 23 × 23 mm untergebracht!

Bild 2. 3D-gedrucktes Gehäuse zum Schutz der Nicla-Vision-Platine.

Hardware-Aufbau

Passend zur Nicla-Vision-Entwicklungsplatine haben wir ein schickes Gehäuse im 3D-Druckverfahren hergestellt (**Bild 2**), das mit einer Aluminiumstange am Kessel angebracht werden kann und auf das analoge Messgerät ausgerichtet ist. Obwohl sich der Kessel im Gebäudeinneren befindet, wo keine Gefahr von Wasserschäden besteht und die Staubbelastung gering ist, bietet das Gehäuse einen zusätzlichen Schutz gegen Umwelteinflüsse.
Gewindeeinsätze wurden mit einem Heißluftföhn in das Basisteil eingebracht (**Bild 3**), sodass der Deckel mit M3-Schrauben befestigt werden kann. Dies gewährleistet einen guten Sitz des Deckels auf der Basis und ermöglicht das wiederholte Öffnen und Schließen des Gehäuses (**Bild 4**), ohne dass ein 3D-gedrucktes Gewinde beschädigt wird. Außerdem haben wir uns für eine mit GoPro kompatible Halterung für den Deckel entschieden – eine übliche Wahl in der Open-Source-Hardware-Community, die dieses Design mit zahlreichen anderen online verfügbaren Halterungen kompatibel macht.
Je nach Aufbau kann die Beleuchtung im Tagesverlauf variieren, daher können Sie auch eine Lichtquelle hinzufügen, um eine konstante Ausleuchtung zu gewährleisten. Dies ist ein entscheidender Aspekt, um die Leistung des ML-Modells sicherzustellen, da die Lichtverhältnisse die erkannten Merkmale stark beeinflussen können.

Software-Aufbau

Auf der Softwareseite beginnen wir mit der Installation der OpenMV-IDE und der Erstellung eines Edge-Impulse-Kontos [3].

Datenerfassung

Um das Modell darauf zu trainieren, zu erkennen, wann der Druck außerhalb des Normbereichs liegt, definieren wir drei mögliche Kategorien: niedrig, normal und hoch, wie in **Bild 5** dargestellt. Da wir nicht in der Lage sind, Bilder des Messgeräts mit dem Zeiger in allen möglichen Positionen aufzunehmen, generieren wir den Bilddatensatz mithilfe von Python. OCI Labs stellt dafür einen hervorragenden Code [4] zur

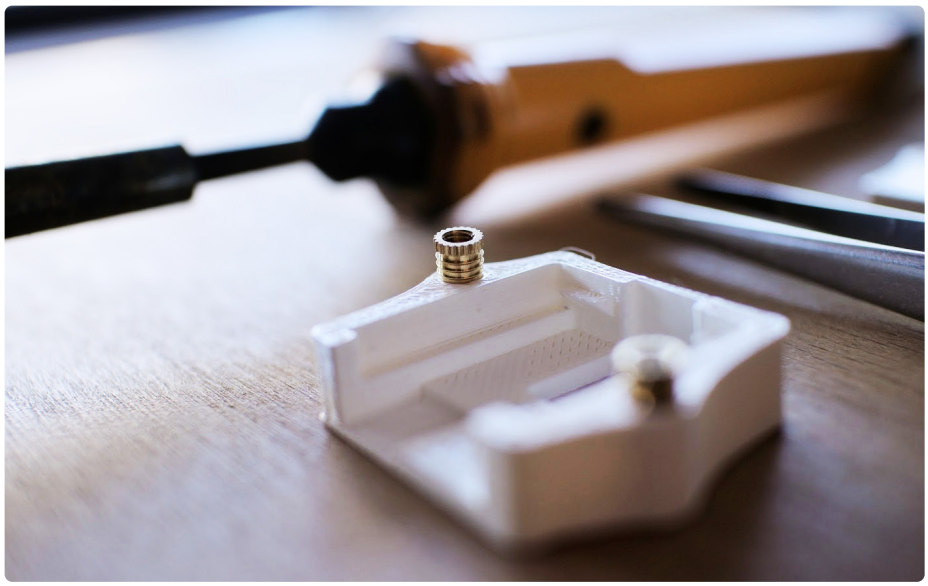


Bild 3. Der Deckel wird mit M3-Schrauben und Gewindeeinsätzen befestigt.

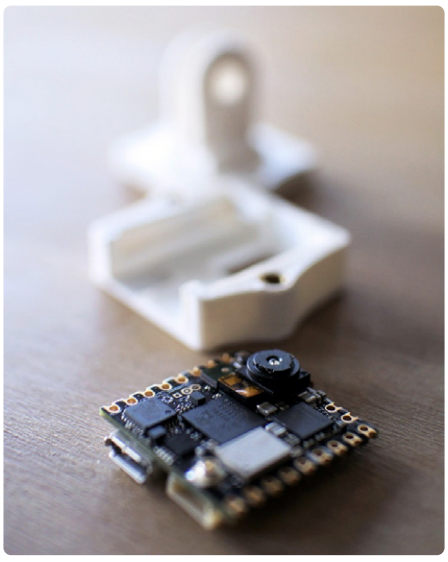


Bild 4. Die Nicla-Vision-Platine außerhalb des Gehäuses.

Bild 5. Druckbereiche des Messgeräts: niedrig, normal und hoch.

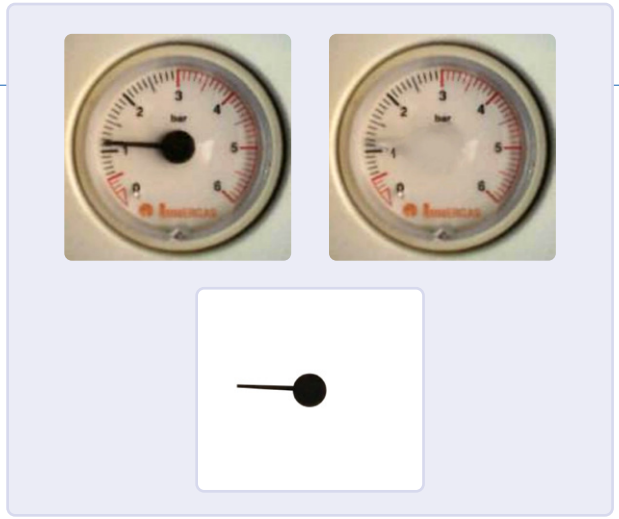


Bild 7. Originalbild des Messgeräts neben dem Hintergrund ohne Zeiger und dem herausgetrennten Zeiger.

Verfügung, den wir für unseren Anwendungsfall angepasst haben. Zuerst schließen Sie die Nicla-Vision-Platine an Ihren Laptop an, öffnen OpenMV und klicken unten links auf die *Connect*-Schaltfläche. Wenn Sie die Platine beim ersten Mal nicht verbinden können, doppelklicken Sie auf die Taste an der Nicla Vision, um die Platine in den Bootloader-Modus zu versetzen (die grüne LED sollte blinken). Gehen Sie dann zu *Tools* → *Dataset Editor* → *New Dataset* und wählen Sie einen Ordner aus, in dem Sie die Bilder speichern möchten (**Bild 6**). Klicken Sie auf *New Class Folder* und geben Sie dem Ordner einen Namen. Machen Sie einige Fotos und wählen Sie das beste aus. Wir verwenden es als Ausgangspunkt, um den Rest des Datensatzes zu generieren. Für den nächsten Schritt benötigen Sie Bildverarbeitungskenntnisse. Trennen Sie den Zeiger aus diesem Bild mit einem beliebigen Bildbearbeitungswerkzeug ab und erstellen Sie auch ein Bild, auf dem der Zeiger vom Messgerät entfernt wurde, wie in **Bild 7** gezeigt. Um Bilder für jeden Wert zu generieren, der einer Messgeräteanzeige entspricht, wird das Zeigerbild auf die jeweiligen Winkel gedreht und auf den Hintergrund überlagert. Sie finden den Code dafür im Jupyter-Notebook zur synthetischen Messgeräte-Bilderzeugung [5]. Nach dem Ausführen des Codes erhalten Sie Ergebnisse ähnlich wie in **Bild 8** und **Bild 9**.

Jetzt, wo wir einen vollständigen Datensatz haben, können wir ein neues Edge-Impulse-Projekt erstellen. Nach dem Login in Ihr Edge-Impulse-Konto erscheint der Bildschirm *Project Creation*. Klicken Sie auf *Create new project*, geben Sie dem Projekt einen aussagekräftigen Namen und wählen Sie *Developer* als gewünschten Projekttyp.

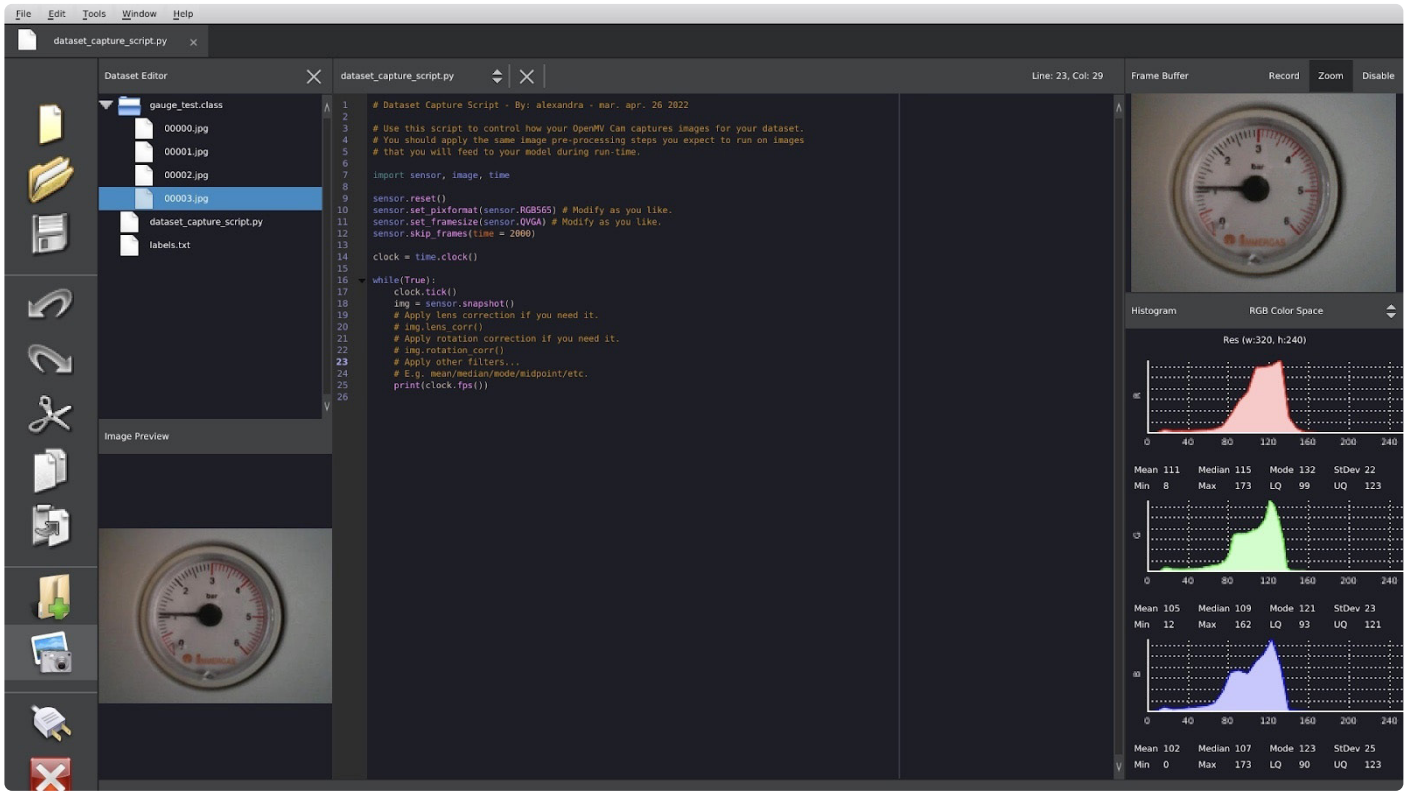


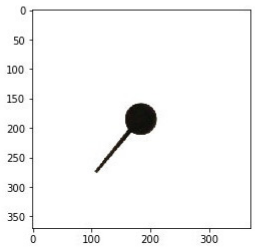
Bild 6. Aufnahme von Referenzbildern des analogen Messgeräts mit der OpenMV-IDE.

Synthetic Images - Correct Needle placement

- Determine what angle the needle needs to be set to in order to display at the 0 gauge number.

```
In [4]:
image = Image.open(NEEDLE).convert('RGBA')
image_rot_0 = image.rotate(53, expand=False, resample=Image.BICUBIC) # adjust this value to rotate the needle to
image_rot_0.save('./images/needle_rot_0.png')
plt.imshow(image_rot_0)
```

```
Out[4]: <matplotlib.image.AxesImage at 0x7f043371e5e0>
```




```
In [8]:
# Gauge background
image = Image.open(GAUGE).convert('RGBA')
img_copy = image.copy()
img_copy.paste(image_rot_0.convert('L'), (0, 0), image_rot_0.convert('RGBA'))

plt.imshow(img_copy)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x7f0431dbe670>
```

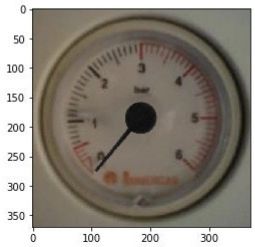


Bild 8. Drehen und Überlagern des Zeigers auf das Ziffernblatt zur Erstellung beschrifteter Trainingsbilder.

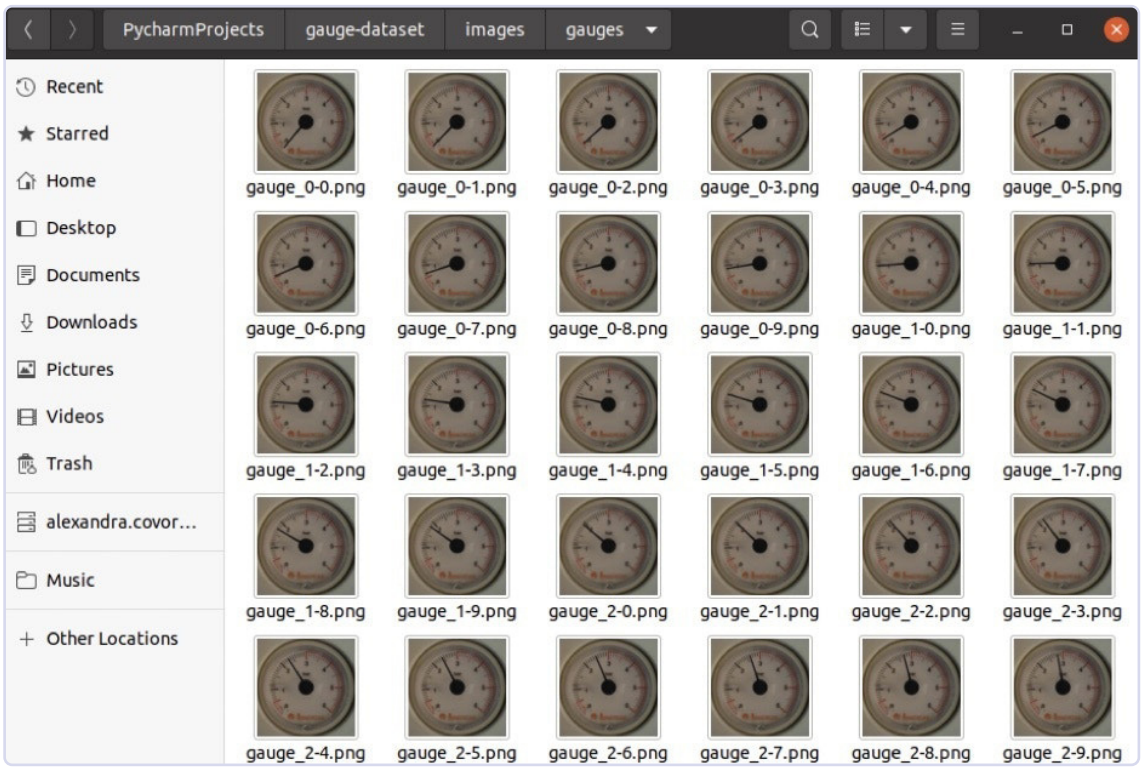


Bild 9. Der fertige Datensatz aus synthetisch generierten Bildern verschiedener Zeigerstellungen.

Anschließend wählen Sie *Images* als Datentyp. Danach wählen Sie *Image Classification* und öffnen das Menü *Data acquisition*. Klicken Sie auf *Upload existing data* und laden Sie die soeben generierten Bilder hoch, auf denen der Zeiger in der Normalposition steht. Achten Sie darauf, sie entsprechend zu kennzeichnen (**Bild 10**). Wiederholen Sie diesen Vorgang für die anderen beiden Kategorien.

Impuls erstellen

Um einen „Impuls" (ein Lernmodell) zu erstellen, gehen Sie zu *Impulse Design* und stellen Sie die Bildgröße auf 96 × 96 px ein, fügen Sie einen *Image*-Verarbeitungsblock und einen *Transfer-Learning*-Block hinzu. Wir werden kein Modell von Grund auf neu trainieren, sondern die Möglichkeiten eines vortrainierten Modells nutzen und seine letzten Schichten mit unserem Datensatz erneut trainieren, was viel wertvolle Zeit und Ressourcen spart. Dieses Verfahren wird als Transferlernen bezeichnet. Die einzige Einschränkung bei dieser Methode besteht darin, dass wir die Bilder unseres Datensatzes auf die Größe der Bilder umskalieren müssen, mit denen das Modell ursprünglich trainiert wurde, also entweder 96 × 96 px oder 160 × 160 px. Wir haben uns für 96 × 96 px entschieden, da die Nicla-Vision-Platine nur 1 MB RAM und 2 MB Flash-Speicher zur Verfügung hat.
Die Ausgabefunktionen werden unsere Kategorien sein (**Bild 11**), d. h. die zuvor definierten Labels (hoch, niedrig und normal).

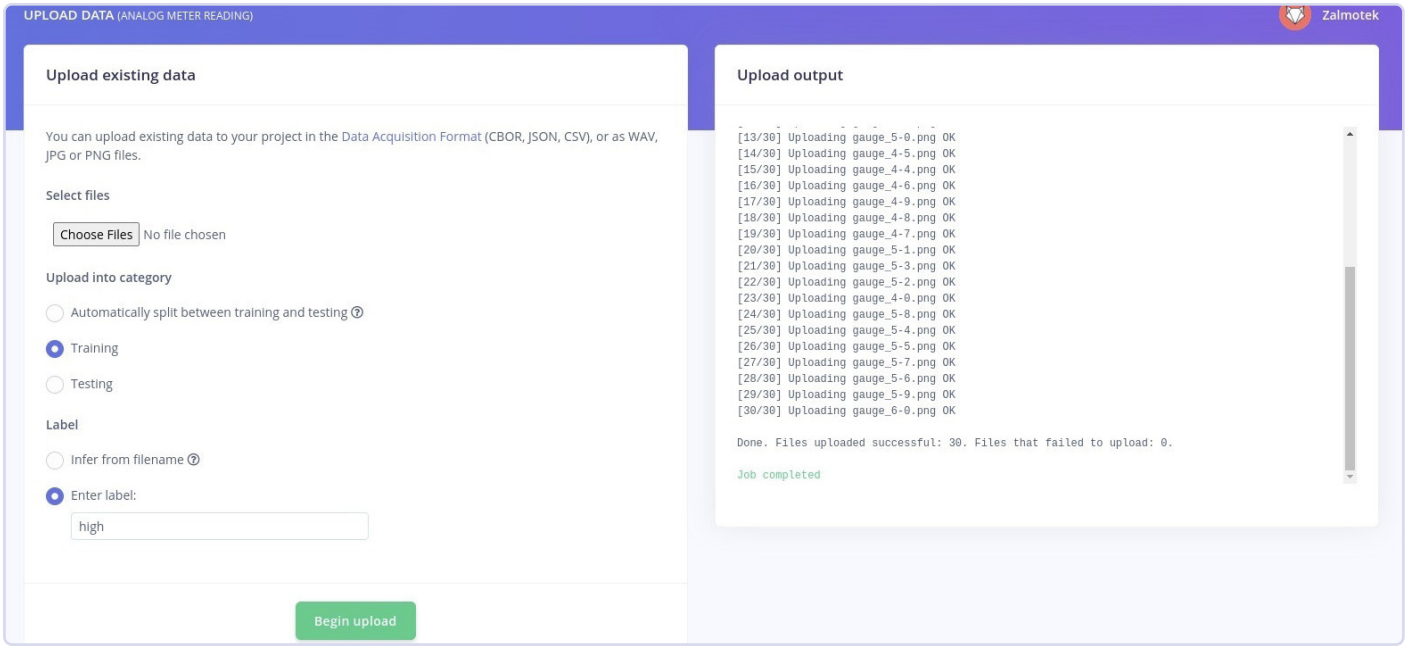


Bild 10. Hochladen des erzeugten Datensatzes zu Edge Impulse und Kennzeichnung der Trainingsdaten.

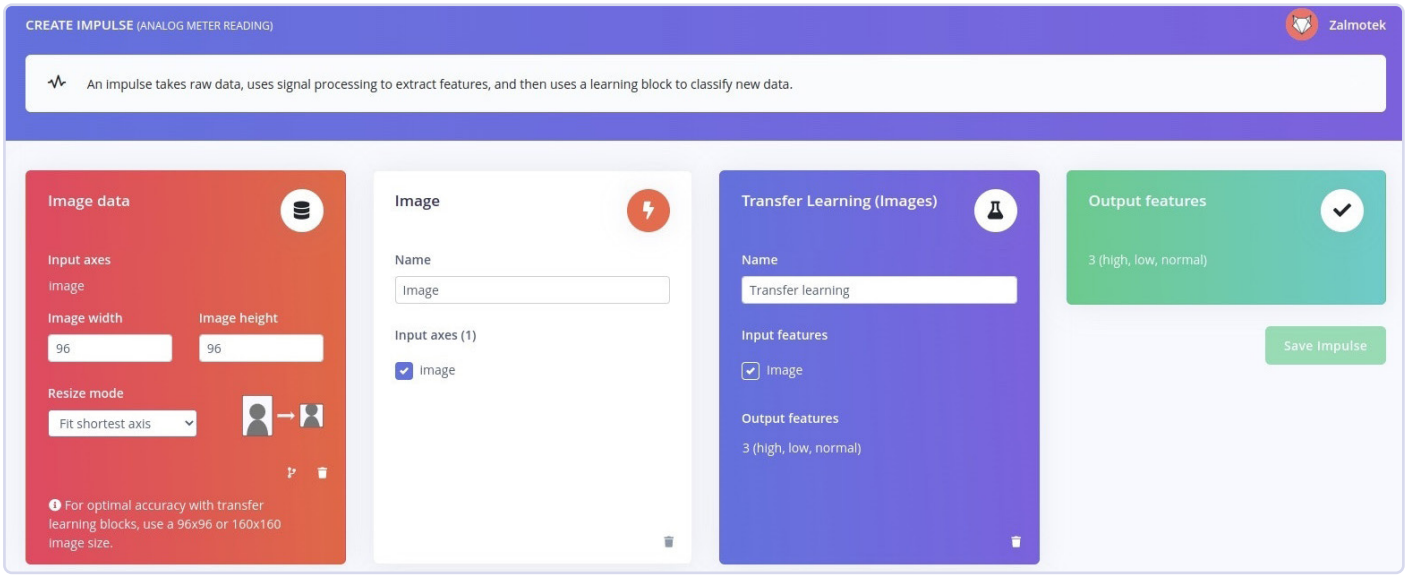


Bild 11. Erstellung des Impulses in Edge Impulse.

Merkmale generieren

Nun wechseln wir zum *Image*-Menü im Menü *Impulse Design* und klicken auf *Save Parameters* und *Generate Features* (**Bild 12**). Dadurch werden alle Bilder auf 96 × 96 px skaliert und optional die Farbtiefe auf *RGB* oder *Grayscale* geändert. Wir haben den Standardmodus *RGB* gewählt. Sie können die erzeugten Merkmale auch im *Feature explorer* visualisieren, gruppiert nach Ähnlichkeit (**Bild 13**). Eine gute Faustregel ist, dass gut getrennte Cluster im Feature Explorer für das Machine-Learning-Modell leichter zu erlernen sind.

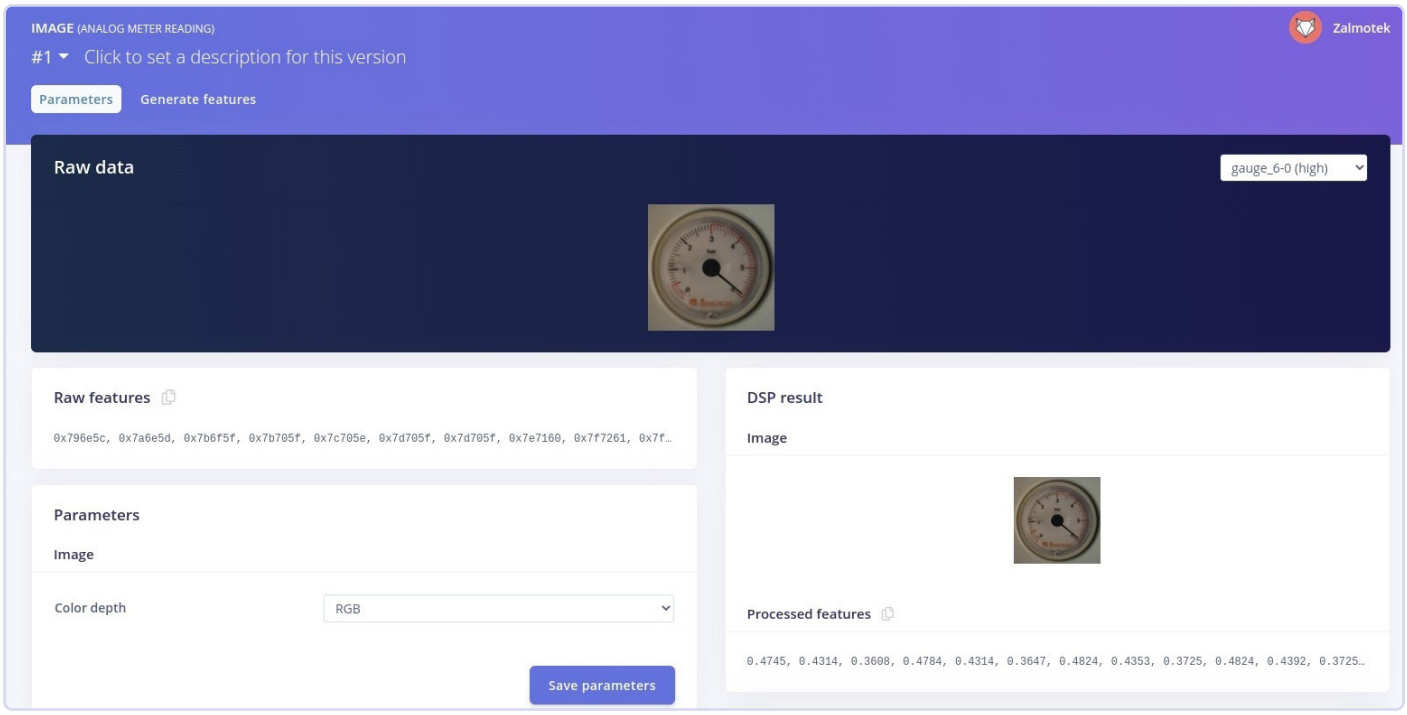


Bild 12. Rohmerkmale und digitale Signalverarbeitungsergebnisse (DSP) für ein Eingabebild des Messgeräts.

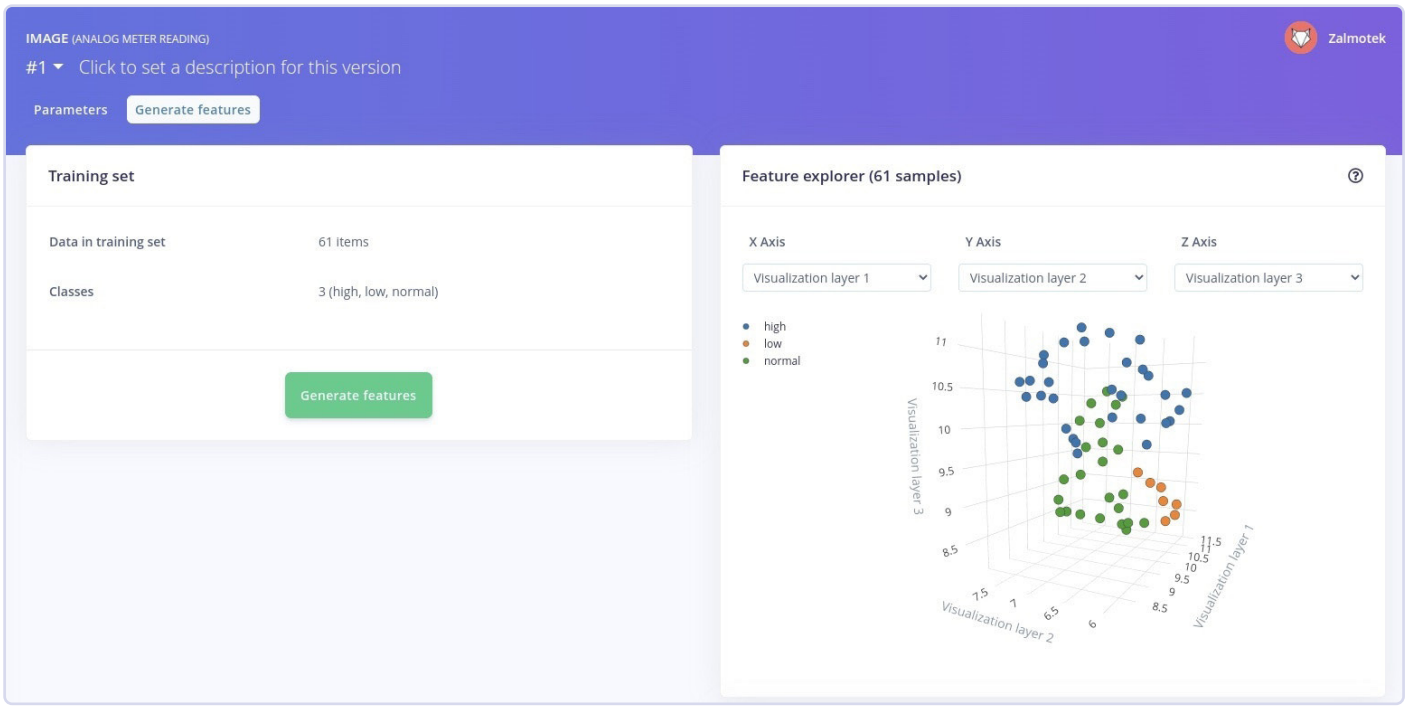


Bild 13. Visualisierung der Feature-Explorer-Verteilung der Klassen (niedrig, normal, hoch) nach Merkmalserstellung.

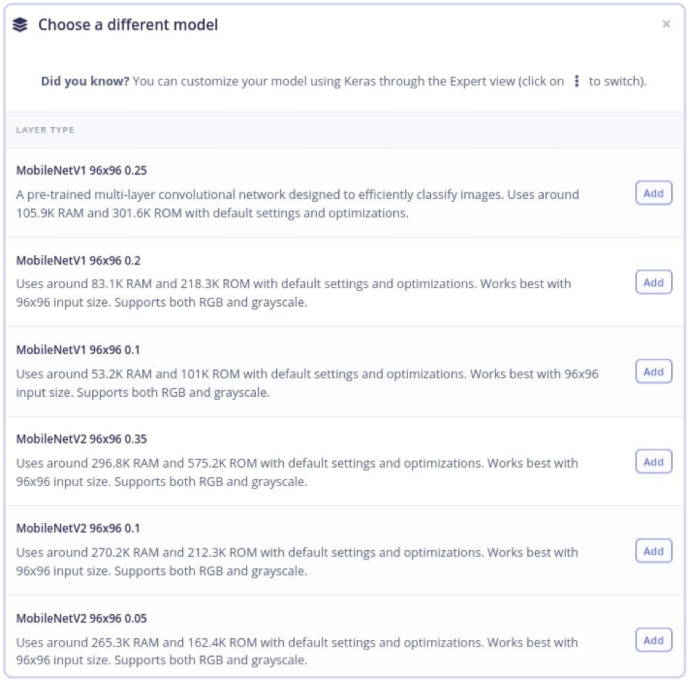


Bild 14. Seite zur Modellauswahl.

Das Modell trainieren

Da wir nun die Merkmale haben, um das Training zu starten, können wir das neuronale Netz im Menü *Transfer Learning* trainieren. Bei der Modellauswahl müssen wir die Speicherbeschränkungen der Nicla-Vision-Platine (1 MB RAM und 2 MB Flash-Speicher) berücksichtigen, deshalb haben wir das Modell *MobileNetV2 96x96 0.05* gewählt, das recht leichtgewichtig ist. Sie können das Modell auswählen und die Speicheranforderungen überprüfen, indem Sie auf *Choose a different model* klicken (**Bild 14**).

Sie können vorerst die Standardeinstellungen des *Neural Network* verwenden und nach dem Training des Modells zurückkehren, um diese für eine bessere Genauigkeit zu optimieren. Es gibt keine eindeutige Antwort darauf, welche Einstellungen für das Modell am besten sind, da dies von Fall zu Fall unterschiedlich ist. Sie können mit verschiedenen Werten experimentieren, wobei Sie Unter- und Überanpassung vermeiden sollten. Ein Beispiel für unsere Einstellungen sehen Sie in **Bild 15**.

Das Modell bereitstellen

Nachdem wir unser Modell nun erstellt, trainiert und validiert haben, ist es an der Zeit, es auf der Nicla-Vision-Platine bereitzustellen. Gehen Sie im Edge-Impulse-Menü zu *Deployment*, wählen Sie *OpenMV Firmware* (**Bild 16**) und klicken Sie unten auf der Seite auf die *Build*-Schaltfläche. Dadurch wird eine OpenMV-Firmware generiert und als ZIP-Datei heruntergeladen. Entpacken Sie diese, und Sie finden darin mehrere Dateien, darunter *edge_impulse_firmware_arduino_nicla_vision.bin* und *ei_image_classification.py*, die für uns relevant sind.

Der nächste Schritt besteht darin, die heruntergeladene Firmware mit dem ML-Modell auf die Nicla-Vision-Platine zu laden. Gehen Sie zurück zu OpenMV und dann zu *Tools → Run Bootloader (Load Firmware)*, wählen Sie die *.bin*-Datei aus und klicken Sie auf *Run*. Danach wählen

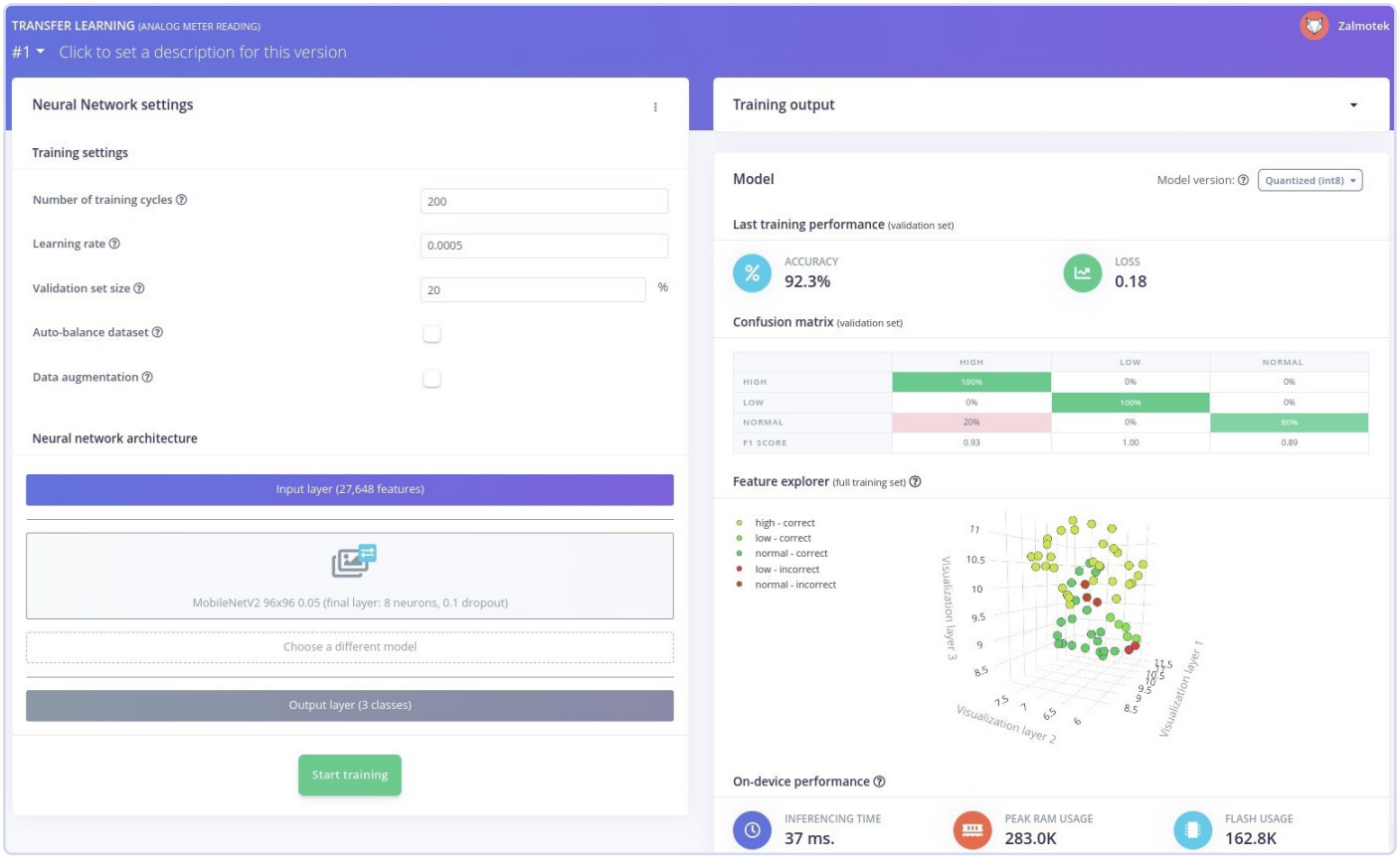


Bild 15. Konfiguration des neuronalen Netzes und Leistungsmetriken. Erreichte Genauigkeit: 92,3 %.

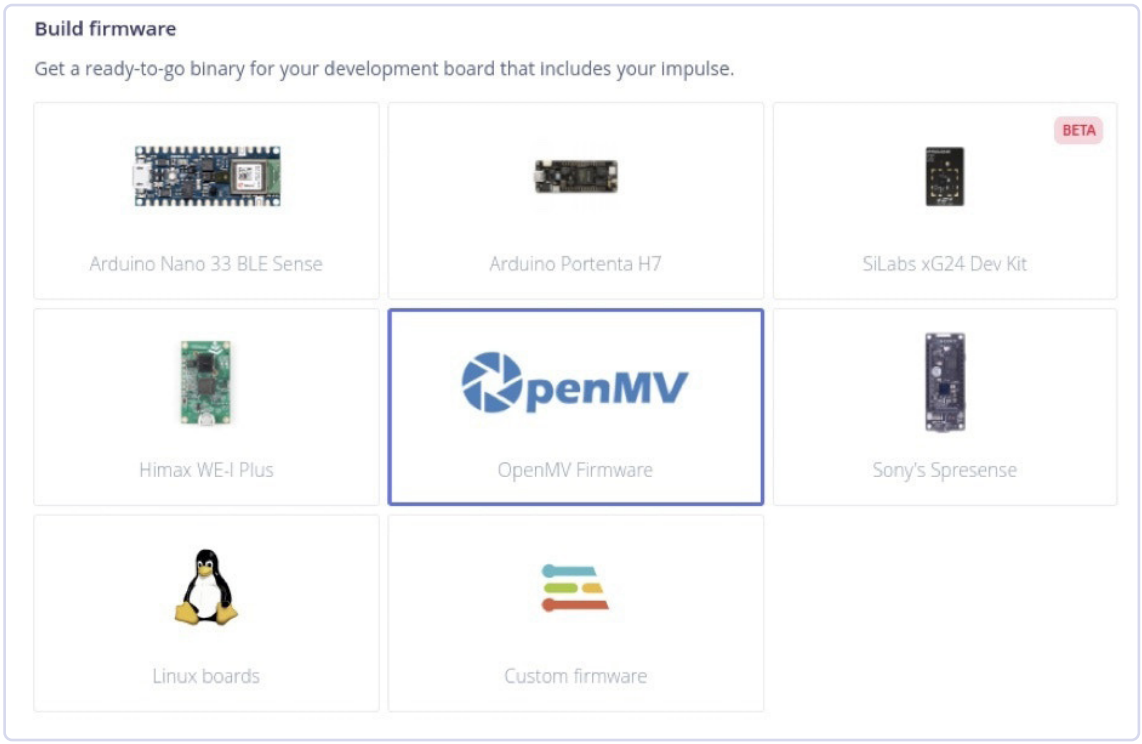


Bild 16. Bereitstellungsoptionen zur Generierung von Firmware-Binärdateien.

Sie *File* → *Open File* und wählen die Python-Datei aus dem Archiv aus. Um Speicherplatz zu sparen, passen Sie die folgenden Codezeilen in Ihrer Datei an:

```
sensor.set_framesize(sensor.QQVGA)     # Set frame size
to QQVGA (160x120)
sensor.set_windowing((96, 96))         # Set 96x96 window
```

Da das Modell mit 96 × 96 px großen Bildern trainiert wurde, hätte es keinen Sinn, größere Bilder zu verwenden, da diese lediglich mehr Speicherplatz benötigen (**Bild 17**). Stellen Sie außerdem die Bildgröße der Kamera entsprechend ein. Wenn Sie mit der Maus über *set_framesize* fahren, finden Sie alle verfügbaren Optionen. In unserem Fall funktioniert QQVGA (160 × 120 px) gut. Klicken Sie abschließend auf *Connect* und *Run* – jetzt können Sie das Modell testen!

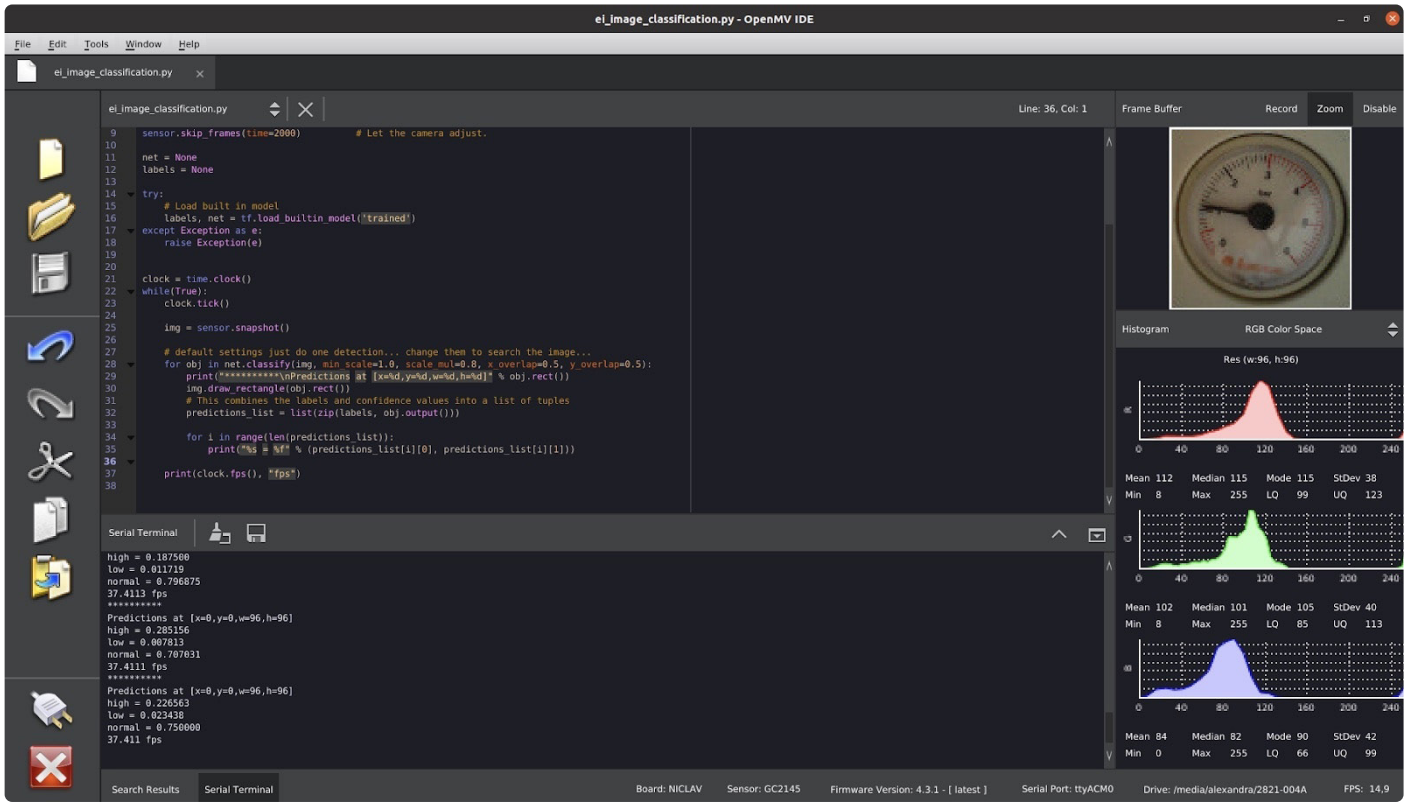


Bild 17. Die OpenMV-IDE mit angezeigten Klassifikationsergebnissen.

Fazit und Ausblick

Das Ablesen analoger Messgeräte ist für die Überwachung des Energieverbrauchs in verschiedenen Bereichen von entscheidender Bedeutung. Ein Hauptgrund dafür ist, dass Computer-Vision-Techniken genutzt werden können, um schnell und einfach Hunderte oder sogar Tausende von Messwerten zu analysieren, um Muster und Trends für verschiedene Datentypen wie Tageszeit, Verbrauchsniveau und zeitliche Veränderungen zu erkennen. Darüber hinaus ermöglicht das Ablesen analoger Messgeräte eine höhere Genauigkeit bei der Überwachung des Energie-, Gas- oder Wasserverbrauchs, da es Echtzeitdaten liefert, die nicht den gleichen Ungenauigkeiten wie Smart Meter unterliegen. Dadurch wird das analoge Ablesen zu einem wichtigen Instrument für Unternehmen und Versorgungsunternehmen bei ihren Bemühungen, die Effizienz zu steigern und Kosten zu senken. Letztlich können Organisationen durch Investitionen in analoge Ablesetechnologie dazu beitragen, eine nachhaltige Zukunft zu sichern, indem sie die ihnen zur Verfügung stehenden Ressourcen besser verwalten.

Arduino Nicla ist ein nützliches Computer-Vision-System, das sich für verschiedene Anwendungen eignet und Objekte in Echtzeit sogar unter schwierigen Bedingungen erkennen und verfolgen kann. Es lässt sich sehr gut an spezielle Anforderungen anpassen – von der Überwachung von Verkehrsströmen bis hin zum Ablesen analoger Messgeräte, wie in unserem Beispiel. In Kombination mit Edge Impulse lassen sich damit schlanke, aber leistungsstarke Computer-Vision-Lösungen mit Edge-KI realisieren.

250688-02

Fragen oder Kommentare?

Haben Sie technische Fragen oder Feedback zu diesem Artikel? Kontaktieren Sie bitte das Elektor-Redaktionsteam unter redaktion@elektor.de.

Über den Autor

Constantin Craciun ist Hardware-Ingenieur und CEO von Zalmotek, einem in Bukarest ansässigen Unternehmen, das auf Hardware-Prototyping spezialisiert ist. Er engagiert sich leidenschaftlich für Fortschritte an der Schnittstelle von KI und Robotik und treibt Innovationen in den Bereichen maßgeschneiderte Elektronik, Wearables und vernetzte Geräte voran.

WEBLINKS

[1] Arduino Nicla Vision, Arduino Store: https://store.arduino.cc/products/nicla-vision
[2] OpenMV-IDE, Download: https://openmv.io/pages/download
[3] Edge-Impulse-Login: https://studio.edgeimpulse.com/login
[4] Erzeugung synthetischer Messgerätebilder von OCI Labs, GitHub: https://tinyurl.com/deepguagegithub
[5] Jupyter-Notebook-Code, GitHub: https://github.com/Zalmotek/edge-impulse-meter-reading-nicla-vision-arduino

KI-Kamera OpenMV AE3

Embedded Vision neu definiert auf Alif Ensemble E3

Von Brian Tristam Williams (Elektor)

Die AE3-Kamera von OpenMV nutzt das Dual-M55 Ensemble E3 SoC von Alif mit zwei Ethos-U55-NPUs, 13 MB On-Chip-RAM und einem 1-MP-Global-Shutter-Sensor – und ermöglicht so KI der Desktop-Klasse mit der Leistung eines Mikrocontrollers.

Die AE3 von OpenMV (**Bild 1**) ist das neueste Modell einer Reihe von Python-programmierbaren KI-Kameras, die darauf abzielen, Computer Vision auf kleinste, batteriebetriebene Embedded-Systeme zu bringen. Das Unternehmen, gegründet von Ingenieur Kwabena Agyeman und Entwickler Ibrahim Abdelkader, begann vor über einem Jahrzehnt als Open-Source-Projekt, das maschinelles Sehen ebenso zugänglich machen wollte, wie Arduino es für Mikrocontroller erreicht hat. Seitdem hat OpenMV eine starke Anhängerschaft unter Hobbyisten, Forschern und Produktentwicklern aufgebaut, die maschinelles Sehene ohne den Overhead von Linux oder Cloud-Infrastruktur benötigen. Die Kameras kombinieren stromsparende Hardware, MicroPython-Skripting und eine intuitive IDE, um die Entwicklung von KI-Anwendungen direkt auf dem Gerät für jeden praktikabel zu machen.

Bild 1. Die winzige KI-Kamera OpenMV AE3.

Die AE3 setzt diese Tradition fort – bewegt sich jedoch in eine völlig neue Leistungsklasse. Basierend auf dem Ensemble-E3-SoC von Alif Semiconductor bietet sie Beschleunigung für neuronale Netze und einen riesigen On-Chip-Speicher auf einem Board, das kaum größer als eine Münze ist.

Das Board wurde Anfang 2025 zusammen mit dem höherwertigen N6-Modell über eine Kickstarter-Kampagne [1] vorab vorgestellt, die mehr als 150.000 US-Dollar einbrachte – das Dreifache des Ziels. Während die Hardware in ihrer finalen Iteration ist, erwartet OpenMV die vollständige Software-Reife bis Ende 2026.

Dennoch ermöglicht die AE3, wie wir später sehen werden, bereits jetzt Echtzeit-Objekterkennung und Segmentierung mit erstaunlichen Bildraten.

Die Ursprünge von OpenMV

OpenMV begann mit der Idee, eine kleine Kamera und einen Mikrocontroller zusammenzubringen, um lokale Computer-Vision-Aufgaben zu erledigen – einfaches Farb-Tracking, Linienverfolgung und Marker-Erkennung – alles in Python programmierbar. Frühere Boards wie die H7 haben bewiesen, dass MicroPython auch für ernsthafte Embedded-Anwendungen taugt. Die AE3 ist die fünfte Generation und steht für OpenMVs Wechsel von MCU-basiertem Rechnen zu echter KI-Inferenz. Sie vereint alle Fähigkeiten früherer Kameras – Gesichtserkennung, AprilTags (ähnlich wie QR-Codes, aber zur Positionsbestimmung von Robotern im realen Raum [2]), Barcodes, optisches Fluss-Tracking – und ergänzt sie durch Hardware-Beschleunigung für neuronale Netze – leistungsstark genug, um kompakte YOLO-Modelle in Echtzeit auszuführen.

▲ *Bild 2a. Vorderseite der OpenMV AE3 mit USB-C-Stecker als Größenvergleich.*

Aktueller Entwicklungsstand

Die AE3 (zu finden auf der OpenMV-Shopseite [3]) ist das erste OpenMV-Board, das Alif Semiconductors Ensemble-Familie von Multicore-Mikrocontrollern verwendet. Die AE3 zielt auf das stromsparende Ende dieser Reihe ab. Sowohl die AE3 als auch die N6 [4] laufen mit MicroPython und LiteRT (ehemals TensorFlow Lite für Mikrocontroller), sodass trainierte Modelle direkt auf dem Gerät ausgeführt werden können, ohne einen Host-PC zu benötigen.

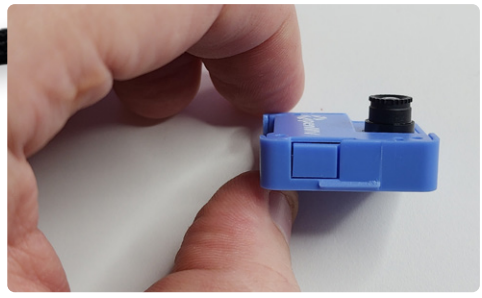
Was die AE3 auszeichnet, ist ihre Effizienz: Sie liefert etwa das 50-Fache der Leistung älterer H7-Modelle bei weniger als halbem Stromverbrauch. Das macht sie geeignet für batteriebetriebene Sensoren, autonome Drohnen, Smart-Agrar-Kameras und jedes System, das im Einsatz vor Ort „sehen" können muss, ohne die Cloud zu nutzen.

Spezifikationen der OpenMV AE3

Wie in **Bild 2a** bis **2d** zu sehen, ist das Modul in einem spritzgegossenen Gehäuse untergebracht. Es ist nur etwa einen Zoll (25 × 25 mm) groß und beherbergt dennoch eine vollständige KI-Computing-Plattform mit Funkmodulen, Sensoren und einer Global-Shutter-Kamera.

Der Name AE3 deutet darauf hin, was sich darunter befindet (**Bild 3a** und **3b**): Unter anderem Alif Semiconductors Ensemble E3, ein System-on-Chip (SoC), das erstaunliche Fähigkeiten in ein Chipgehäuse von nur 6 × 7 mm packt:

- **Zwei CPUs (ARM Cortex-M55)** – jede unterstützt Helium (M-Profile Vector Extension) für DSP- und Matrixoperationen.
- **Zwei NPUs (ARM Ethos-U55)** liefern zusammen 250 GOPS Durchsatz für neuronale Netze.
- **13 MB On-Chip-SRAM** – enorm für einen Mikrocontroller – plus 5 MB MRAM für Code und Modelle
- **32 MB externer Octal-SPI-Flash**, direkt in den Adressraum des Prozessors gemappt mit etwa 200 MB/s Lesegeschwindigkeit über das neue ROMFS-Dateisystem von OpenMV
- **Integrierte GPU** für schnelles Bild-Skalieren und Komposition
- **Stromverbrauch**: Typischer aktiver Strombedarf etwa 60 mA während der Inferenz; Tiefschlaf im Bereich einiger zehn Mikroampere.

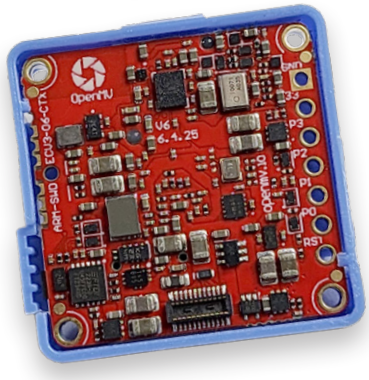


◀ *Bild 2b. Kamera-Modul von rechts, mit der konfigurierbaren Universaltaste.*

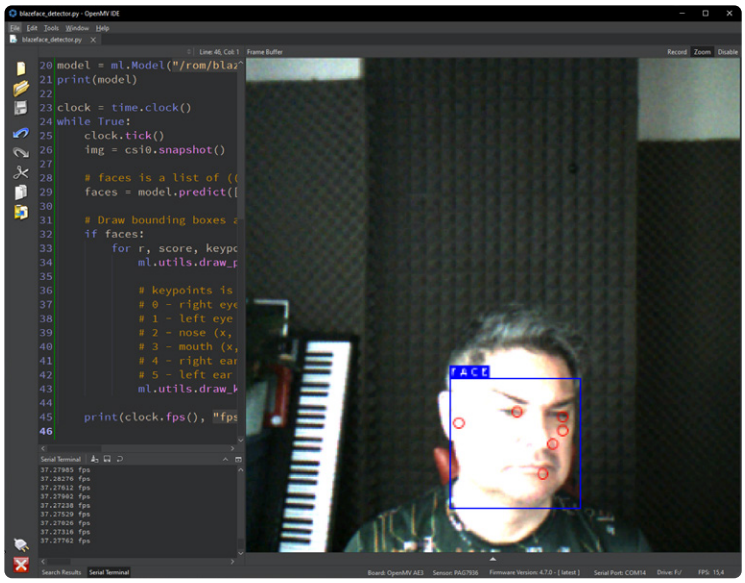
◀ *Bild 2c. Unterseite, zeigt die QWIIC- und USB-C-Anschlüsse.*

◀ *Bild 2d. Rückseite des Gehäuses mit den externen Anschlüssen.*

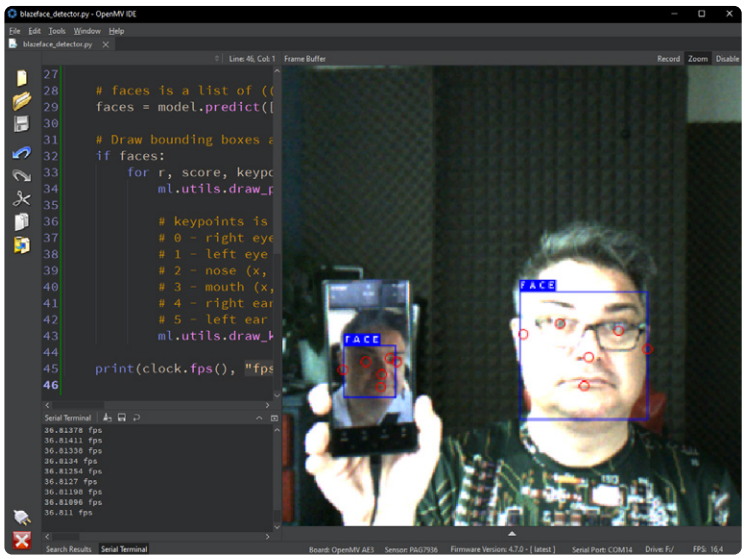
◀ *Bild 3a. Ohne Gehäuse ist die Technik im Inneren sichtbar, einschließlich des E3-Dual-Core-Mikrocontrollers von Alif Ensemble in der Mitte. Der USB-C-Anschluss (unter dem QR-Code) verdeutlicht nochmals, wie winzig der Chip ist (6 × 7 mm).*



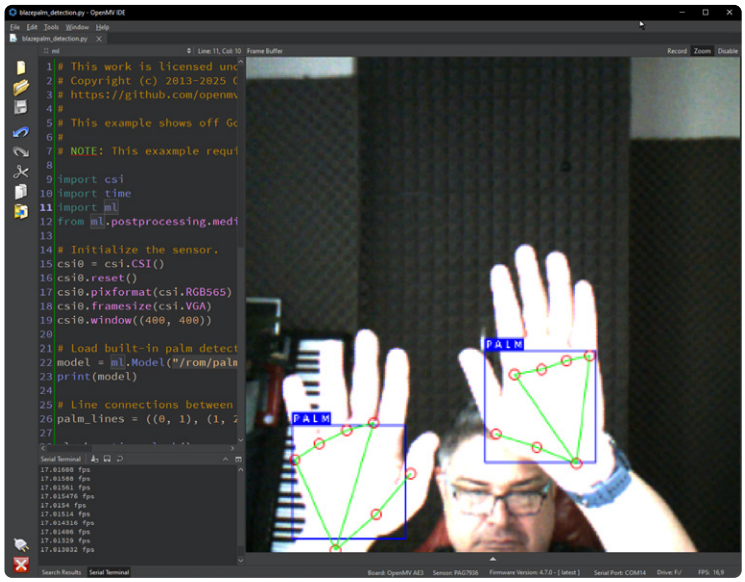
◀ *Bild 3b. Rückseite des Boards. Eine Stiftleiste könnte leicht an die GPIO-Reihe auf der rechten Seite gelötet werden.*



▲ *Bild 4a. blazeface_detector.py in Aktion – ein Gesicht.*



▲ *Bild 4b. blazeface_detector.py in Aktion – mehrere Gesichter.*



▲ *Bild 5. blazepalm_detection.py läuft.*

Peripherie und Sensoren

Für ein so kleines Board kommt die OpenMV AE3 mit einer vollen Palette an Onboard-Peripherie, darunter:

- **Hauptbildsensor:** 1 MP Global-Shutter-Farbsensor (wie bei der N6), bis zu 400 fps (Frames/s) bei QVGA
- **Wi-Fi + Bluetooth LE 5** mit optionaler U.FL-Antenne
- **Time-of-Flight-Entfernungssensor**, neben der Kamera, aber außerhalb des Sichtfelds
- **3-Achsen-Beschleunigungssensor + Gyroskop**, mittig auf der Rückseite der Leiterplatte
- **Mikrofonanschluss**, **RGB-Status-LED**, **Benutzertaste** und ein **Recovery-Schalter** zum Wiederherstellen der Firmware
- **Qwiic/Stemma-QT-Anschluss** für externe Sensoren
- **Board-to-Board-Erweiterungsstecker** mit JTAG und etwa 10 GPIOs (SPI, I²C, UART, PWM usw.)
- **USB-C-Highspeed-Schnittstelle** für Programmierung und Streaming

Die OpenMV IDE

Wie bei Arduino-Boards oder Espressifs ESP32-Modulen hätte ich vermutet, dass ich die AE3 einfach an meinen PC anschließen und Code aus der Arduino IDE, PlatformIO oder sogar Thonny hochladen könnte. Aber OpenMV ist einen anderen Weg gegangen und hat seine eigene Entwicklungsumgebung, die OpenMV IDE [5], gebaut, weil keines der bestehenden Tools den kompletten Workflow – Live-Bildströme, Peripherie-Steuerung und MicroPython-Skripting auf einem Kameramodul – handhaben konnte.

Die IDE, erstmals veröffentlicht mit der ursprünglichen OpenMV Cam im Jahr 2015, wurde entwickelt, um Embedded-Vision-Entwicklung allen zugänglich zu machen. Sie kombiniert eine Live-Framebuffer-Ansicht mit einem Python-Editor und einer seriellen Konsole, sodass Sie sehen können, was die Kamera sieht, während Ihr Code läuft. Sie können Skripte anpassen, neuronale Modelle ausführen oder Fokus und Belichtung on-the-fly einstellen – alles ohne Firmware-Neukompilierung.

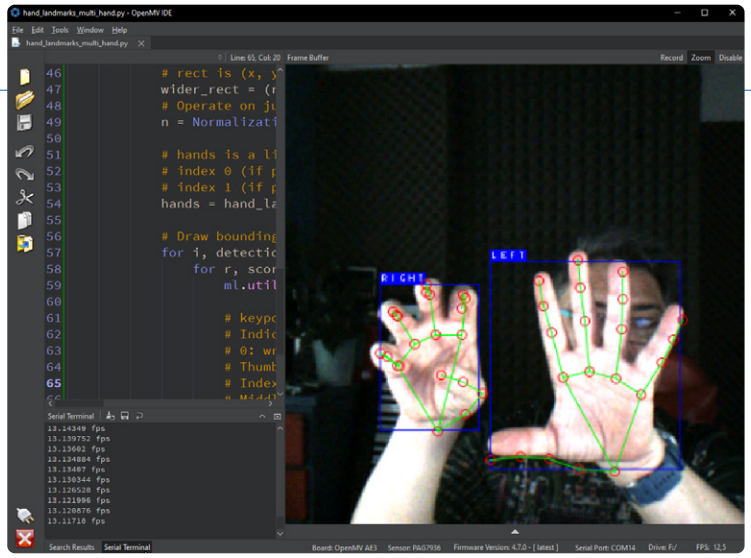
Die OpenMV IDE bietet Entwicklern eine maßgeschneiderte Umgebung für die direkte Arbeit mit der Kamera. Bilddaten sind ein zentraler Bestandteil des Workflows, mit Live-Video, sofortigem Feedback und einfacher MicroPython-Programmierung – Dinge, für die allgemeine IDEs (noch) nicht konzipiert sind. Ein Beispiel: Die aktualisierte OpenMV IDE, basierend auf Qt Creator, verbindet sich über USB-C für Live-Video-Streaming, Skript-Upload und REPL-Zugriff (Read-Eval-Print-Loop). Sie unterstützt mehrere Tabs zum Editieren, Dateiverwaltung auf dem Dateisystem der Kamera und Ein-Klick-Deployment.

Einige praktische Beispiele

Da sich die Software noch in aktiver Entwicklung befindet, kam ich in den Genuss einer persönlichen Einführung durch den OpenMV-Gründer Kwabena Agyeman. Wir haben mehrere Demo-Skripte ausprobiert, die der IDE beiliegen und im GitHub-Repository des Projekts [6] zu finden sind. Diese Beispiele sind eine hervorragende Möglichkeit, um zu sehen, was die AE3 bereits kann – noch bevor die Firmware ihre vollständige Reife erreicht.
Ich habe mehrere LiteRT-Demos ausgeführt, darunter

- *blazeface_detector.py* mit einem einzelnen Gesicht (**Bild 4a**) und mehreren (**4b**), bei 37 bzw. 36 Bildern pro Sekunde
- *blazepalm_detection.py* (**Bild 5**) mit 17 fps, nützlich für einfache Handgesten wie „STOP!“.
- *hand_landmarks_multi_hand.py* (**Bild 6**) – Erkennung einzelner oder mehrerer Hand-Landmarken, bis hin zu den Fingerknochen. Für 21 Punkte pro Hand sind 13 fps ziemlich flott.
- *yolo_lc_person_detector.py* (**Bild 7**) – der bekannte Personendetektor, etwa 17 fps, unabhängig von der Anzahl der Personen im Bild

Jedes Beispiel wurde direkt aus der Vielzahl an Beispielen aus dem OpenMV-GitHub-Repo kopiert und direkt auf dem Gerät ausgeführt, mit Live-Ergebnissen im IDE-Fenster. Der Dual-Core-Ensemble-Prozessor der AE3 hat alle Aufgaben mit beeindruckender Reaktionsfähigkeit gemeistert und problemlos Bewegungen in Echtzeit erkannt, während Gesichter, Hände und Posen identifiziert wurden.
Bemerkenswert war der enorme Sprung in der Geschwindigkeit. Frühere OpenMV-Boards wie die H7 schafften 0,6 Bilder pro Sekunde bei vergleichbaren neuronalen Netzmodellen. Die AE3 hingegen bewältigt YOLO-Modelle mit etwa 20 bis 30 Bildern pro Sekunde und erreicht über 120 fps mit kleineren FOMO-Netzen. Im Labor brachte Kwabena sie sogar auf 175 fps – etwa das Hundertfache der vorherigen Generation bei einem Bruchteil des Stromverbrauchs.
Die Beispiele demonstrieren auch das innovative Memory-Mapped-Dateisystem (ROMFS) von OpenMV. Als Computerfan der 80er-Jahre habe ich früher mit Memory-Mapped-Speicher gearbeitet (bei dem externe Speicherchips einem Segment des Adressraums der CPU zugeordnet sind), ebenso mit Memory-Mapped-Peripheriegeräten (bei denen man durch Lesen und Schreiben bestimmter Speicheradressen mit der Hardware kommunizierte) und sogar mit gemeinsam genutztem Speicher (wo CPU und z. B. Videoprozessor abwechselnd auf denselben Speicher zugreifen). Zugegeben, die Idee eines Memory-Mapped-Dateisystems war mir jedoch noch nicht gekommen.
Mit dem OpenMV-ROMFS können Modelle direkt aus dem Octal-SPI-Flash ausgeführt werden, ohne ins RAM geladen zu werden, was nahezu sofortigen Start und eine saubere, ordnerartige Struktur für mehrere Projekte ermöglicht. Es ist eine elegante Lösung, die Komfort mit hoher Durchsatzleistung verbindet – etwa 200 MB/s Lesegeschwindigkeit.
Entwickler sind nicht auf OpenMVs eigene Datensätze angewiesen. Der Workflow unterstützt Modelle, die mit Edge Impulse oder Roboflow trainiert und als LiteRT-Dateien exportiert werden und einfach auf die Kamera übertragen werden. Wichtig: Dieselben Python-Skripte funktionieren sowohl auf der AE3 als auch auf der höherwertigen N6 – dank eines gemeinsamen MicroPython-Frameworks, das Hardware-Details abstrahiert.
Ebenfalls praktisch: Die IDE liefert Live-Feedback. Während jedes Modell läuft, werden Begrenzungsrahmen und Schlüsselpunkt-Overlays im Videostream angezeigt, was es einfach macht, Ergebnisse zu überprüfen, Parameter zu justieren und sofort zu



▲ *Bild 6. hand_landmarks_multi_hand.py in Aktion.*

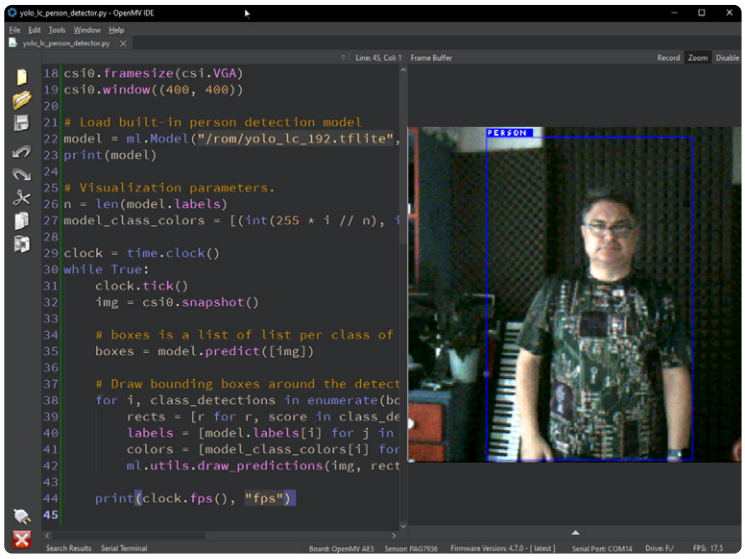


Bild 7. yolo_lc_person_detector.py läuft.

sehen, wie verschiedene Netze dieselbe Szene interpretieren. Es ist eine ungewöhnlich reibungslose Entwicklungserfahrung für ein Mikrocontroller-basiertes Vision-System – eine Brücke zwischen der Zugänglichkeit für Hobbyisten und professioneller Leistung.

Was kann man damit machen?

Mit der Kombination aus hoher Leistung, niedrigem Stromverbrauch und kompakter Bauform eröffnet die AE3 zahlreiche potenzielle Anwendungen, wie:

Tragbare Analytik: Überwachung von Industrieanlagen oder Produktionslinien vor Ort, ohne Cloud-Verbindung
Batteriebetriebene Wildkameras: Erkennung von Arten oder Verhaltensweisen und selektive Übertragung relevanter Clips statt kontinuierlicher Aufnahmen
Gesten- und Bewegungserkennung: Natürliche Interaktion mit Robotern, Displays oder Consumer-Geräten
Wearable-Vision-Systeme: Leichtgewichtige, am Kopf oder Körper getragene Anwendungen, die von Onboard-Inferenz und geringer Wärmeentwicklung profitieren
Automatisierte Inspektion: Echtzeit-Fehlererkennung in der Fertigung oder Qualitätskontrolle
Umweltsensorik: Nutzung des eingebauten Beschleunigungssensors, Time-of-Flight-Sensors und Mikrofons für multimodale KI-Eingaben

Die AE3 ist nicht nur auf Computer Vision beschränkt; die integrierten Sensoren ermöglichen *Fusion*-basierte Anwendungen, die Ton, Bewegung und Tiefenwahrnehmung kombinieren. Entwickler könnten ebenso gut akustische Ereigniserkennung, Umweltüberwachung oder Mensch-Maschine-Schnittstellen erforschen – alles mit einem handflächengroßen, programmierbaren Modul.
Wie immer sind die interessantesten Anwendungen aber meist nicht die vorhersehbaren. Die eigentliche Überraschung wird sein, was die Menschen tatsächlich damit umsetzen. Haben Sie Ideen, die ich noch nicht erwähnt habe? Lassen Sie es mich wissen! Meine Kontaktdaten finden Sie im Kasten **Fragen oder Kommentare?**.

250836-01

Fragen oder Kommentare?

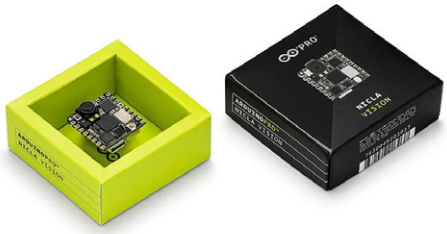
Haben Sie Fragen an den Autor oder eigene Ideen? Sie erreichen mich unter brian.williams@elektor.com oder das Elektor-Redaktionsteam unter redaktion@elektor.de.

Über den Autor

Brian Tristam Williams interessiert sich seit seinem ersten „Mikrocomputer" im Alter von zehn Jahren für Computer und Elektronik. Sein Weg zu Elektor begann mit dem Kauf seiner ersten Ausgabe im Alter von 16 Jahren. Seitdem verfolgt er die Entwicklungen in Elektronik und Computertechnik mit großer Neugier. 2010 stieß er zum Elektor-Team. Heute konzentriert er sich besonders auf aktuelle Technologietrends, vor allem KI und Einplatinencomputer wie den Raspberry Pi.

Passendes Produkt

› **Arduino Pro Nicla Vision**
www.elektor.de/20152



WEBLINKS

[1] OpenMV N6 und AE3, Kickstarter: https://tinyurl.com/openmvkickstarters
[2] AprilRobotics/apriltag auf GitHub: https://github.com/AprilRobotics/apriltag
[3] AE3-Kameramodul, OpenMV-Store: https://openmv.io/products/openmv-ae3
[4] N6-Kameramodul, OpenMV-Store: https://openmv.io/products/openmv-n6
[5] OpenMV IDE: https://openmv.io/pages/download
[6] Tensorflow-Machine-Learning-Beispiele, OpenMV-GitHub-Repository: https://tinyurl.com/openmvgithubexamples

Vom Cloud- zum Edge-Computing

Erkenntnisse aus der Praxis

Von Michelle Hoogenhout, PhD (Hydrostasis)

Von der Datenerfassung über das Feature-Design bis hin zur Beobachtbarkeit: Erfolgreiche Edge-KI-Strategien betrachten stets das große Ganze. Das in San Diego ansässige Start-up Hydrostasis, das ein tragbares System zur Überwachung des persönlichen Flüssigkeitshaushalts zur Verbesserung der Gesundheit entwickelt, hat seine KI-Verarbeitung aus Rechenzentren direkt auf das tragbare Gerät verlagert. Die dabei gewonnenen Erfahrungen sind äußerst wertvoll – sowohl für Unternehmen, die einen ähnlichen Schritt planen, als auch für alle, die gerade erst mit eigenen Edge-Projekten beginnen.

Als Machine-Learning-Entwickler fühlen wir uns in der Cloud wohl. Wir haben leistungsstarke GPUs, praktisch unbegrenzten Speicher und den Luxus, komplexe Modelle schnell zu iterieren. Aber was passiert, wenn Ihr Unternehmen diese Modelle auf einem winzigen Gerät mit begrenzter Leistung, Speicher und Rechenleistung ausführen muss? Genau diese Reise haben wir 2017 als Wearables-Startup angetreten, das sich auf die Überwachung der Flüssigkeitszufuhr für Gesundheit und Langlebigkeit konzentriert.

Wir hatten bedeutende Meilensteine erreicht: ein marktreifes Produkt und zufriedene Kunden. Unser nächstes ambitioniertes Ziel erforderte jedoch eine grundlegende Veränderung. Wir mussten unsere Algorithmen zur Überwachung der Flüssigkeitszufuhr überall und jederzeit funktionsfähig machen – ohne auf App-Konnektivität oder Mobilfunknetze angewiesen zu sein. Das bedeutete, dass unsere Intelligenz von der Cloud direkt auf das Gerät selbst übertragen werden musste.

Wenn Sie ein ML-Ingenieur sind und eine Edge-Bereitstellung in Betracht ziehen, finden Sie hier, was wir während unseres Übergangs vom Cloud-first- zum Edge-native-Denken herausgefunden haben.

Warum wir Edge Impulse gewählt haben

Seien wir ehrlich: Viele junge Startups stehen vor derselben Realität. Wir sind klein, anpassungsfähig und können es uns oft nicht leisten, zusätzliche Spezialisten wie eigene Firmware-Entwickler in unser Team aufzunehmen. Wir mussten die Lücke zwischen Cloud-Bereitstellung und einem funktionierenden Edge-Algorithmus auf schnelle, flexible und kosteneffiziente Weise überbrücken.

Edge Impulse wurde zu unserer Brücke über diesen Abgrund. Die Plattform ermöglichte es uns, weiterhin Modelle mit den uns bekannten und beliebten Python-Toolkits zu entwickeln, während automatisch optimierte C++-Pakete für das Edge-Deployment erstellt wurden. Für die meisten unserer Modelle entfiel so die Notwendigkeit, Algorithmen manuell zu portieren oder sich mit Embeded-Optimierungstechniken auseinanderzusetzen, die wir noch nicht beherrschten.

Die entscheidende Erkenntnis hier betrifft nicht eine bestimmte Plattform – es geht darum, zu erkennen, wann man Werkzeuge benötigt, die den Prozess angesichts der speziellen Fähigkeiten und Einschränkungen des Teams beschleunigen können.

Vier wichtige Lektionen für Ihr nächstes Edge-Deployment

1. Dateninfrastruktur: Ihr zukünftiges Ich wird es Ihnen danken

Was wir gelernt haben: Die Datenpraktiken, die während der anfänglichen Entwicklung als „gut genug" erscheinen, werden zu schmerzhaften Engpässen, wenn man auf Edge-Plattformen migrieren oder Modelle unter anderen Bedingungen neu trainieren muss.

Das Problem: Unsere frühe Datenspeicherung war inkonsistent. Einige Datensätze existierten in verschiedenen Formaten, die Kennzeichnungskonventionen unterschieden sich je nach Experiment, und die Dokumentation der Vorverarbeitung befand sich an verschiedenen Orten. Dies führte zu Reibungen bei jedem Versuch, Daten auf externe Plattformen zu importieren oder unsere Modelle an Edge-spezifische Anforderungen anzupassen.

Die Lösung: Etablieren Sie von Anfang an rigorose, standardisierte Prozesse für das Speichern sowohl von Rohdaten als auch von verarbeiteten Daten. Erstellen Sie klare Benennungsregeln, halten Sie konsistente Datenstrukturen ein und dokumentieren Sie jeden Vorverarbeitungsschritt. Wenn Sie mit Sensordaten arbeiten, die für Edge-Geräte bestimmt sind, wird dies umso wichtiger, da Sie möglicherweise Ihre gesamte Pipeline mit anderen Speicher- und Rechenleistungsanforderungen rekonstruieren müssen.

Diese Investition in die Dateninfrastruktur zahlt sich später enorm aus. Glauben Sie uns – Ihr zukünftiges Ich wird Ihnen danken, wenn Sie Ihre Datensätze schnell für neue Plattformen anpassen oder Modelle mit unterschiedlichen Optimierungszielen neu trainieren können.

2. Produktfunktionen vs. Modellleistung: Der versteckte Kompromiss

Was wir gelernt haben: Optimierungen der Benutzererfahrung können Ihre Trainingsdatenqualität auf eine Weise verschlechtern, die erst viel später offensichtlich wird.

Unser Beispiel: Wir haben intermittierende Messungen eingeführt, um die Batterielaufzeit zu verlängern – ein klarer Gewinn für die Kundenerfahrung. Dies führte jedoch zu inkonsistenten und reduzierten Abtastraten, die unsere Trainingsdaten weniger zuverlässig machten. Wir mussten Daten über längere Zeitfenster aggregieren, um die Konsistenz zu wahren, was sich auf die Granularität und Reaktionsfähigkeit des Modells auswirkte.

Die übergeordnete Erkenntnis: Produktentscheidungen und ML-Pipeline-Entscheidungen sind stärker miteinander verknüpft, als es zunächst erscheint. Eine Funktion, die das Nutzerengagement verbessert, könnte Datenverschiebungen verursachen. Eine Energieoptimierung, die die Batterielaufzeit verlängert, könnte Lücken in Ihren Trainingsdaten erzeugen, die die Modellleistung beeinträchtigen.

Empfehlung: Etablieren Sie eine Rückkopplungsschleife zwischen Ihren Produkt- und ML-Teams. Dokumentieren Sie, wie jede Produktentscheidung Ihre Datenpipeline beeinflusst, und behalten Sie diese Kompromisse während der Modellentwicklung im Blick. Berücksichtigen Sie die ML-Auswirkungen frühzeitig bei der Produktplanung und nicht erst im Nachhinein.

3. Gehen Sie über Metriken hinaus: Bauen Sie Modell-Observability auf

Was wir gelernt haben: Genauigkeitswerte, Präzision und Recall sind wichtig, aber sie verraten Ihnen nicht, warum Ihre Modelle in der realen Welt scheitern – insbesondere auf ressourcenbeschränkten Geräten, bei denen die Fehlerursachen andere sein können als in Cloud-Umgebungen.

Die Herausforderung: Edge-Geräte bringen neue Variablen mit sich, die die Modellleistung auf Weisen beeinflussen können, die ML-Ingenieuren aus der Cloud-Entwicklung nicht offensichtlich sind. Temperaturschwankungen, Stromschwankungen und Hardware-spezifische Quantisierung (Modellkompression) können zu unerwarteten Fehlern und Verzerrungen führen.

Die Lösung: Schaffen Sie Systeme, mit denen Sie tatsächliche Vorhersagen und Fehlerbilder inspizieren können. Egal, ob Sie untersuchen, warum ein Regressionsmodell unter bestimmten Bedingungen systematisch zu hohe Werte liefert, oder verstehen möchten, welche Klassen Ihr neuronales Netz am häufigsten verwechselt – Sie benötigen Einblick in den Entscheidungsprozess des Modells.

Dies ist für das Edge-Deployment entscheidend, da das Debugging auf dem Gerät deutlich schwieriger sein kann als in der Cloud. Entwickeln Sie Observability-Tools, die Ihnen helfen, das Modellverhalten vor dem Deployment zu verstehen, und schaffen Sie Logging-Mechanismen, die Edge-spezifische Probleme erfassen können, ohne die Ressourcen Ihres Geräts zu überfordern.

4. Denken Sie Edge-first bei der Datenverarbeitung

Was wir gelernt haben: Verarbeitungs-Pipelines, die in der Cloud problemlos funktionieren, können auf Edge-Geräten zu echten Engpässen werden – und manchmal entscheiden sie über ein tragfähiges oder nicht tragfähiges Produkt.

Realitätscheck: Ihre Python-Vorverarbeitungs-Pipeline, die ganz entspannt komplette Datensätze in den Speicher lädt und komplexe Transformationen ausführt, wird auf einem Gerät mit 256 KB RAM nicht funktionieren. Die Reihenfolge der Verarbeitungsschritte, die in der Cloud beliebig schien, ist plötzlich entscheidend, wenn jeder Rechenschritt den Akku beeinflusst.

Geca, ein tragbarer Flüssigkeitsmonitor von Hydrostasis.

Strategie: Wenn ein Edge-Deployment auf Ihrer Roadmap steht, entwerfen Sie Ihre Datenvorverarbeitung von Anfang an unter Berücksichtigung der Gerätebeschränkungen. Das bedeutet:

- Speichernutzung bei jedem Verarbeitungsschritt berücksichtigen
- Rechenaufwand mit Blick auf die Batterielaufzeit bewerten
- Prüfen, ob Ihre Feature-Engineering-Ansätze auf Embedded-Prozessoren umsetzbar sind
- Vorverarbeitungs-Pipelines so gestalten, dass sie schrittweise auf Streaming-Daten und nicht nur als Batch-Verarbeitung laufen können

Das bedeutet nicht, dass Sie zu früh optimieren müssen – aber Sie sollten die Edge-Einschränkungen bereits in der Cloud-Entwicklungsphase im Hinterkopf behalten.

Das große Ganze: Ihre ML-Pipeline neu denken

Edge-Deployment verändert grundsätzlich, wie Sie über Machine-Learning-Systeme nachdenken. Es geht nicht nur um Modelloptimierung oder Quantisierung – es geht darum, Ihre gesamte Pipeline unter Berücksichtigung von Ressourceneinschränkungen neu zu gestalten.

In der Cloud optimieren wir für Modellleistung und Entwicklungsgeschwindigkeit. Bei Edge-Modellen optimieren wir für die Schnittmenge aus Leistung, Energieeffizienz, Speichernutzung und Rechenressourcen. Dieser Perspektivenwechsel beeinflusst alles – von der Datenerhebung über die Modellarchitektur bis hin zu den Bereitstellungspipelines.

Ausblick: Fangen Sie frühzeitig an zu diskutieren

Egal, ob Sie noch ganz am Anfang stehen oder bereits Modelle auf Geräte bringen müssen: Entscheidend ist, dass Sie frühzeitig über Einschränkungen und Anforderungen sprechen.

Beginnen Sie damit, Ihre Must-haves von Nice-to-haves zu unterscheiden. Verstehen Sie die spezifischen Einschränkungen, innerhalb derer Sie arbeiten werden – Energiebedarf, Speicherbeschränkungen, Rechenleistung und Latenzanforderungen. Diese Einschränkungen sollten Ihre Entwicklungsentscheidungen schon lange vor dem eigentlichen Deployment beeinflussen.

Am wichtigsten ist: Verstehen Sie, dass Edge-Deployment nicht nur eine technische Herausforderung ist – es ist ein strategischer Wandel, der Ihre Herangehensweise an ML-Entwicklung von der Datenerhebung über das Modelldesign bis zur Bereitstellungsinfrastruktur prägt.

Die Reise vom Cloud- zum Edge-Computing hat uns gelehrt, dass erfolgreiches Edge-ML nicht bedeutet, Cloud-Modelle auf kleine Geräte zu zwängen. Es geht darum, ein anderes Paradigma zu akzeptieren, in dem Einschränkungen Innovation antreiben und Ressourcenbewusstsein jede Entscheidung prägt. Das Ergebnis, wenn es gut gemacht ist, ist ML, das nicht nur technisch beeindruckend, sondern wirklich transformativ für Nutzer ist, die Intelligenz überall benötigen.

250831-02

Über die Autorin

Michelle Hoogenhout verfügt über mehr als 15 Jahre Erfahrung darin, biologische Signale und Verhaltensdaten in verwertbare Informationen zur Förderung der menschlichen Gesundheit zu verwandeln. Sie ist spezialisiert auf den strategischen Aufbau von Data-Science-Abteilungen und die Entwicklung tragbarer Technologien – von der Firmware über die App bis hin zur ML-Implementierung – um leistungsfähige Produkte zu schaffen. Michelle promovierte in Psychologie (Neuropsychologie) an der University of Cape Town und absolvierte ein Forschungsstipendium in neuropsychiatrischer Genetik an der Harvard T.H. Chan School of Public Health. Sie ist leitende Data Scientist bei Hydrostasis (www.hydrostasis.com) und erreichbar unter mhoogenhout@hydrostasis.com.

Q&A mit Pete Bernard

(Executive Director, EDGE AI FOUNDATION)

Die Zukunft der Edge-KI

Intelligenz skalieren – von der Hardware bis zur Cloud

Fragen vom Elektor-Team

Edge-KI verändert grundlegend, wie Intelligenz in die reale Welt gelangt. Pete Bernard, Executive Director der EDGE AI FOUNDATION, erläutert, wie Edge-KI globale Herausforderungen auf zugängliche Weise lösen kann – und was erforderlich ist, um die nächste Million Innovatoren im Edge-Bereich zu befähigen.

Pete Bernard, Executive Director (Quelle: EDGE AI FOUNDATION)

Elektor: Was war der Auslöser für das Rebranding von tinyML zur EDGE AI FOUNDATION (edgeaifoundation.org), und was sagt das über die Richtung aus, in die sich das Ökosystem entwickelt?

Pete Bernard: Wir sind den Impulsen der Community gefolgt – die tinyML Foundation wurde bereits 2018 gegründet und war in den Anfangsjahren maßgeblich daran beteiligt, herauszufinden, *wie* man KI-Modelle in sehr ressourcenbeschränkten Umgebungen betreiben kann. Aber unsere Community-Gespräche entwickelten sich über die ursprünglichen tinyML-Grenzen hinaus, hin zu breiteren Diskussionen und Forschung rund um maschinelles Sehen, generative KI, physische KI usw. sowie die Verbindung von Cloud und Edge. Daher waren ein Rebranding und ein Umdenken bezüglich des Umfangs der Organisation notwendig. Das ist sehr gut angekommen, und wir haben nun eine deutlich vielfältigere und wachsende Edge-KI-Community, die vom einzelnen Chip bis zur Cloud und vom Pre-Seed bis zur Big Tech reicht.

Elektor: Sie sehen Anwendungen in Bereichen wie Gesundheitswesen, Fertigung, Energie und darüber hinaus. Welche Branchen finden Sie derzeit am spannendsten und warum?

Bernard: Es gibt enormes Potenzial, wenn Sie Endpunkte mit Intelligenz ausstatten. Manche Optimierungen in Prozessen, Erträgen und Sicherheit sind offensichtlich, aber wir sehen auch Kombinationen von KI-Modellen, die zusammenarbeiten, etwa VLMs (Visual Language Models), die mit Sensornetzwerken kombiniert werden, um Sprachintelligenz und agentische KI in die reale Welt zu bringen. Stellen Sie sich vor, Sie betreten ein Krankenzimmer und fragen, wie es dem Patienten geht, und erhalten die Antwort in Ihrer eigenen Sprache. Oder Sie fragen ein Gerät, wie es ihm geht, und es sagt Ihnen, welche Wartung erforderlich ist. Oder ein Hirnimplantat passt sich an, um einem Parkinson-Patienten zu helfen. Gerade diese neuartigen Kombinationen sind wirklich faszinierend, und ich denke, das ist erst der Anfang.

Elektor: Die Foundation hat Tausende von Studierenden weltweit erreicht. Wie wird Bildung Ihrer Ansicht nach die nächste Generation von Edge-KI-Innovatoren beeinflussen, insbesondere in aufstrebenden Märkten?

Bernard: Wir haben weltweit über 100.000 Studierende, die Edge-KI durch unseren Stipendienfonds lernen, der Bildungsprogramme im globalen Süden fördert, durch unsere Unterstützung eines Open-Source-Lehrplans für Universitäten und Schulen sowie durch unsere *EDGE AI Challenges* und *Grant*-Programme. Als Beispiel haben wir im März ein mehrwöchiges Programm in Malawi finanziert, um Edge-KI Hunderten von Lehrkräften und Studierenden nahezubringen, mit praktischen Experimenten und Anleitungen zum von uns unterstützten Open-Source-Lehrplan. Dasselbe werden wir Ende September in Kolumbien tun. Unser großer Multiplikator besteht darin, Lehrkräfte auszubilden und zu befähigen, dies in Ingenieurstudiengängen und Schulen selbst zu unterrichten. Im Herbst starten wir außerdem das Programm *EDGE AI Certification*, um das Wissensniveau zu Edge-KI auf zugängliche und produktneutrale Weise zu zertifizieren.

Das Gute ist, dass Edge-KI schon immer auf Effizienz – bei Kosten, Stromverbrauch und Größe – ausgerichtet war; deshalb ist es tatsächlich die einfachste und günstigste Art, KI in der realen Welt umzusetzen.

Elektor: Sie sind an KI-Governance-Diskussionen mit Organisationen wie dem Weltwirtschaftsforum beteiligt. Was ist derzeit am dringendsten in Bezug auf Richtlinien rund um KI im Edge-Bereich?

Bernard: Derzeit rennen wir alle, so schnell wir können, und ehrlich gesagt fehlt es uns an Transparenz (woher stammen diese Daten?) und es fehlen Bereiche wie die Sicherheitszertifizierung. Wenn Sie agentische KI in die Welt der physischen KI bringen, entstehen großartige, aber potenziell auch extrem riskante Szenarien. Zwei meiner aktuellen Schwerpunkte sind Sicherheitskonzepte und Best Practices für reale agentische KI. Zum Beispiel: Was sind Best Practices, um einen KI-Agenten „anzuheuern"? Welche Protokolle gibt es für den Datentransfer zwischen Edge und Cloud und anderen Systemen? Und wo sind die Grenzen für Agenten im Edge-Bereich, sprich: Wo braucht man einen Menschen in der Schleife?

Elektor: Wie kann Edge-KI vermeiden, eine „Luxustechnologie" zu werden, und stattdessen zu einem Werkzeug zur Lösung kritischer Herausforderungen in unterversorgten Regionen werden?

Bernard: Das Gute ist, dass Edge-KI schon immer auf Effizienz – bei Kosten, Stromverbrauch und Größe – ausgerichtet war; deshalb ist es tatsächlich die einfachste und günstigste Art, KI in der realen Welt umzusetzen. Wir sehen lokale Anwendungen in der Landwirtschaft, in Wassersystemen sowie bei Sicherheits- und Schutzszenarien, wo kein nuklear betriebenes Rechenzentrum und keine Netzanbindung nötig sind. Und der geringe Stromverbrauch ermöglicht es vielen Systemen, sich selbst zu versorgen und sehr nachhaltig zu sein.

Elektor: Was treibt die nächste Welle von Innovation und Akzeptanz bei Edge-KI an – Entwicklungen bei der Hardware oder Software? Oder ist es eher ein wachsendes Bewusstsein, das hinzukommen muss?

Bernard: Wir sind als Branche im Halbleiterbereich extrem erfolgreich – Beschleunigungsarchitekturen, Hochleistungs-Speicher, neue Verfahren für niedrigen Stromverbrauch – aber ich denke, die nächste Welle wird sehr softwaregetrieben sein – mit besseren Tools, besserer Portabilität sowie einfacherer und sichererer MLOps und Workload-Orchestrierung. Wir müssen es einfach machen, Cloud-Training und Edge-Inferenz zu nutzen, Edge-Lernen zu stärken, Modelle zwischen Plattformen zu verschieben, KI-Modelle vor Ort zu verwalten, zu aktualisieren und auszutauschen, die eventuell abdriften, und Edge-Ressourcen wie jede andere Cloud-Ressource zu betrachten – mit den gleichen Tools und Ansätzen wie bei virtualisierten Cloud-Workloads. Am Ende führt das alles zu wiederholbaren und kommerzialisierten Lösungen, die einfacher zu implementieren und zu skalieren sind und die gewünschten Geschäftsergebnisse liefern.

Elektor: Gibt es Anwendungsfälle, die die unerwartete, menschliche Seite von Edge-KI zeigen – etwas, das über Effizienzgewinne hinausgeht?

Bernard: Manchmal halten wir Dinge wie sauberes Wasser und reichlich Nahrung für selbstverständlich, aber in vielen Teilen der Welt ist das ein alltäglicher Kampf, deshalb hat Edge-KI in diesen Szenarien einen echten, unmittelbaren Einfluss. Ich denke auch, Edge-KI, die dem Umweltschutz dient (z. B. Überwachung, wer im Wald etwas Illegales tut) und Lösungen für ein würdevolles Altern zu Hause sind weitere Beispiele, die jeder als real und notwendig nachvollziehen kann.

Quelle: EDGE AI FOUNDATION

Elektor: Wenn Sie fünf Jahre in die Zukunft schauen, wie sieht Erfolg für die EDGE AI FOUNDATION in Bezug auf Technologie, Menschen und den Planeten aus?

Bernard: Aus der Bildungsperspektive wollen wir eine Million Menschen mit Edge-KI befähigen. Das ist gut machbar, wenn wir unser Stipendienprogramm und unsere Bildungsangebote weiter ausbauen. Um Edge-KI zu etablieren, wollen wir sie zur „Standarddesign-Option" für die meisten KI-Szenarien in Fertigung, Industrie, Automobil, Einzelhandel, Gastgewerbe, Landwirtschaft, Energie, Schifffahrt und mehr machen. Dafür ist echte Zusammenarbeit in unserer Community nötig, aber wir müssen die KI zu den Daten bringen, nicht die Daten zur KI (in die Cloud). Das ergibt einfach Sinn.

250722-02

Über Pete Bernard

Pete Bernard ist Technologie-Manager mit langjähriger Erfahrung mit Edge-Computing, Halbleitern und vernetzten Geräten. Nach 14 Jahren im Silicon Valley wechselte er zu Microsoft, wo er die Entwicklung von Produkten wie Windows IoT, Azure Percept, Windows on Snapdragon und anderen leitete. Er ist Gründer von EDGECELSIOR, einer Initiative, die sich auf die Konvergenz von Halbleitern, 5G und KI konzentriert. Seit April 2024 ist er Executive Director der EDGE AI FOUNDATION. Bernard hält sieben Patente in den Bereichen Sicherheit, Bildverarbeitung, Wearables und Streaming.

Echtzeit-Objektzählung mit Edge-KI

Hohe Auflösung und Geschwindigkeit mit dem Jetson Nano & TensorRT

Von Jallson Suryo (Indonesien)

Dieser Artikel beschreibt ein leistungsfähiges Echtzeit-Zählsystem auf Basis eines NVIDIA Jetson Nano und der FOMO-Modelle von Edge Impulse. Statt herkömmlicher Sensorik ermöglicht die Lösung eine präzise Erfassung kleiner, schnell bewegter und zufällig verteilter Objekte. Die Ergebnisse werden in Echtzeit auf einem LCD-Display ausgegeben.

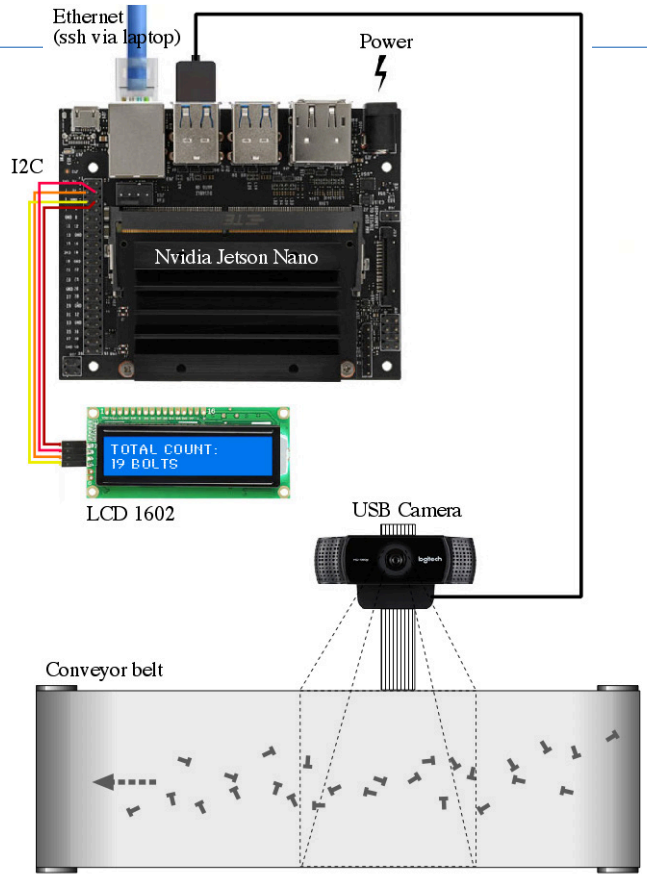


Bild 1. Projektübersicht.

Das Problem

Objektzählsysteme sind in der Fertigungsindustrie essenziell für das Bestandsmanagement und die Lieferkette. Meist werden Näherungssensoren oder Farbsensoren eingesetzt, um Objekte zum Zählen zu erkennen. Näherungssensoren detektieren das Vorhandensein oder Fehlen von Objekten anhand ihres Abstands zum Sensor, während Farbsensoren Objekte nach Farbe oder anderen visuellen Merkmalen unterscheiden können. Es gibt jedoch Einschränkungen dieser Systeme; sie haben typischerweise Schwierigkeiten, viele kleine Objekte zu erkennen, insbesondere wenn sie nicht in einer Reihe oder geordnet sind. Dies kann durch ein relativ schnelles Förderband noch verstärkt werden. Diese Bedingungen führen zu ungenauen Zählergebnissen.

Unsere Lösung

Nach einigen Experimenten mit Computer Vision auf dem Jetson Nano in einem früheren Projekt [1] bin ich überzeugt, dass ein Computer-Vision-System mit Objekterkennung in der Lage ist, viele kleine Objekte selbst auf schnellen Förderbändern präzise zu zählen. Um das zu zeigen, untersuchen wir die Möglichkeiten der FOMO-Modelle von Edge Impulse [2], die für die GPU im Jetson Nano optimiert wurden. In diesem Projekt läuft das Förderband relativ schnell, mit vielen kleinen Objekten in zufälligen Positionen, und die Anzahl der Objekte wird live auf einem 16 × 2-LCD angezeigt (siehe **Bild 1**). Geschwindigkeit und Genauigkeit sind die Ziele dieses Projekts.

Wie funktioniert es?

Das Projekt nutzt den FOMO-Algorithmus von Edge Impulse, der Objekte in jedem Kamerabild auf dem laufenden Förderband schnell erkennen kann. FOMO kann Anzahl und Positionskoordinaten dieser Objekte bestimmen. Ziel ist es, die GPU-Leistung des NVIDIA Jetson Nano bei der Verarbeitung hochauflösender Bilder (720 × 720 Pixel im Vergleich zu typischen FOMO-Projekten, die oft mit 96 × 96 arbeiten) zu bewerten – und das bei optimaler Inferenzgeschwindigkeit.
Das Machine-Learning-Modell (*model.eim*) wird mit der *TensorRT*-Bibliothek, optimiert für die GPU, über das Linux-C++-SDK bereitgestellt. Das Edge-Impulse-Modell wird nahtlos in den Python-Code integriert, um das Zählen der Objekte zu ermöglichen. Unser Algorithmus vergleicht aktuelle Koordinaten mit denen vorheriger Frames, um neue Objekte zu erkennen und Doppelzählungen zu vermeiden.

Hardware-Anforderungen

- NVIDIA Jetson Nano Developer Kit
- USB-Kamera (z. B. Logitech C922)
- Mini-Förderbandsystem mit Kamerahalterung
- Objekte, z. B. Schrauben
- Ethernet-Kabel
- PC zum Zugriff auf Jetson Nano per SSH

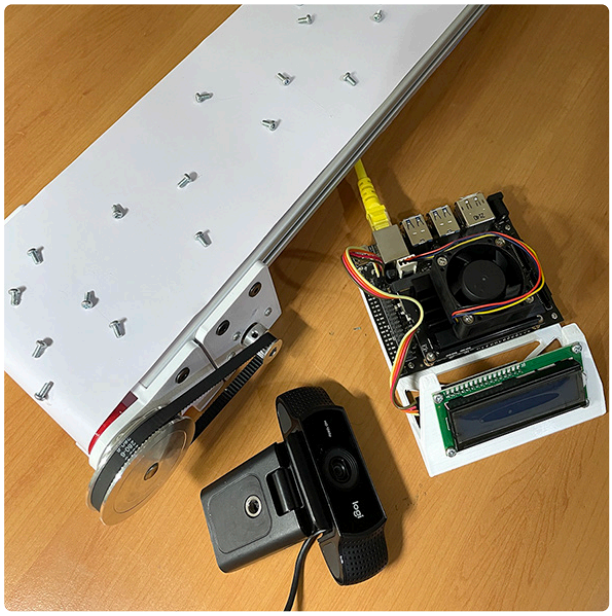


Bild 2. Jetson Nano, Kamera und Förderband.

Bild 3. Datenvariation.

Software und Online-Dienste

- Edge Impulse Studio
- Edge Impulse CLI
- Edge Impulse Linux Python SDK
- NVIDIA Jetpack SDK
- Terminal

Schritte

1. Daten/Bilder vorbereiten

Zur Vereinfachung nutzen wir in diesem Projekt einen PC, der mit einer USB-Kamera vom Typ Logitech C922 verbunden ist, die 720p bei 60 fps aufnehmen kann, um Bilder für die Datensammlung zu erfassen (siehe **Bild 2**). Machen Sie Fotos von oben aus leicht unterschiedlichen Winkeln und bei verschiedenen Lichtbedingungen, damit das Modell unter verschiedenen Bedingungen funktioniert und Überanpassung (Overfitting) vermieden wird. Die Objektgröße ist ein entscheidender Faktor für die FOMO-Leistung. Sie sollten den Abstand der Kamera zu den Objekten konstant halten (siehe **Bild 3**), da deutliche Größenunterschiede das Modell verwirren und die automatische Beschriftung erschweren.

2. Datenerfassung und Beschriftung

Besuchen Sie *studio.edgeimpulse.com*, loggen Sie sich ein oder erstellen Sie ein Konto, und eröffnen Sie ein neues Projekt. Wählen Sie den NVIDIA Jetson Nano als Zielgerät und wählen Sie im Bereich *Projekt*-Info auf dem *Dashboard Bounding boxes (object detection)* als Methode zum Beschriften. Gehen Sie dann zu *Data acquisition*, klicken Sie auf *Add data* und anschließend auf *Upload data*. Wählen Sie Ihre Fotos aus, teilen Sie diese automatisch auf *Training* und *Testing* auf, und klicken Sie auf *Upload data*, wie in **Bild 4** gezeigt.

Bild 4. Das Modul „Upload data".

Bild 5. Manuelles Labeling.

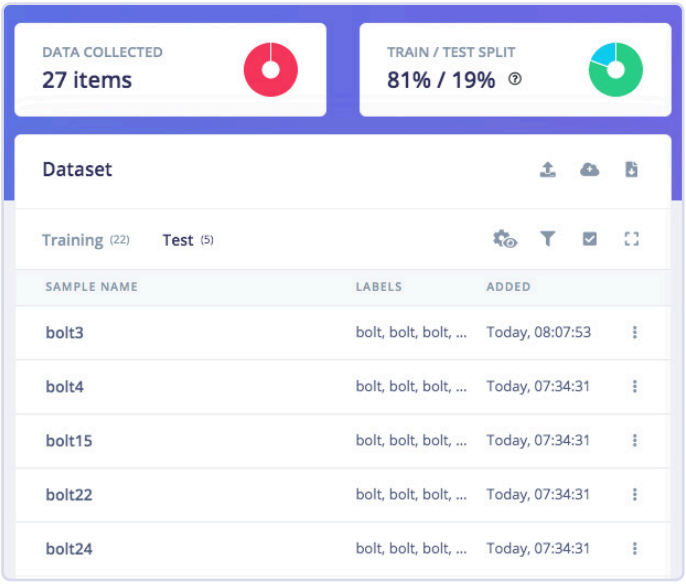


Bild 6. Verhältnis von Trainings- zu Testdaten: 81 % / 19 %.

Anschließend klicken Sie auf den Tab *Labeling queue*, ziehen einen Rahmen um ein Objekt und beschriften es, dann klicken Sie auf *Save labels*, siehe **Bild 5**. Wiederholen Sie das für alle Bilder. Das geht schnell, da Edge Impulse Vorschläge für Rahmen und Beschriftungen machen kann, indem Objekte zwischen Frames verfolgt werden.
Das Verhältnis zwischen Trainings- und Test-Samples wurde auf 80/20 festgelegt (**Bild 6**). Damit steht dem Modell genug Datenmaterial für das Lernen zur Verfügung, während ein Teil für die Testphase reserviert bleibt, um die Leistung vor dem Einsatz zu überprüfen.

3. Modell mit FOMO trainieren und bauen

Ist der Datensatz fertig, gehen Sie zu *Create impulse* und setzen Sie Breite und Höhe des Bildes auf 720 × 720 (**Bild 7**). Wählen Sie dann *Fit shortest axis*, als Verarbeitungsblock *Image* und als Lernblock *Object Detection*. Im *Image*-Block wählen Sie *Grayscale* als Farbtiefe und klicken auf *Save parameters* (**Bild 8**). Sie werden automatisch zur Merkmalsgenerierung weitergeleitet. Klicken Sie auf *Generate features* (**Bild 9**), um eine grafische Darstellung der extrahierten Merkmale zu erhalten. Im Block *Object detection* lassen Sie die Standardeinstellungen unverändert, außer der Anzahl der Trainingszyklen (auf 120 erhöhen). Wählen Sie *FOMO (MobileNet V2 0.35)* als Modell und starten Sie das Training mit *Save & train*. Den Fortschritt sehen Sie rechts. Ist das Training erfolgreich, können Sie mit dem Testen weitermachen. Gehen Sie zu *Model testing* in der Navigation links und klicken Sie auf *Classify all*. Unser Ergebnis liegt über 90 % (**Bild 10** und **Bild 11**), also geht es weiter mit dem nächsten Schritt – der Bereitstellung (*Deployment*).

4. Modell-Deployment für Jetson Nano GPU

Klicken Sie auf *Deployment*, suchen Sie nach *TensorRT*, wählen Sie *Float32* und klicken Sie auf *Build* (**Bild 12**). Dadurch wird eine NVIDIA-TensorRT-Bibliothek erstellt, die das Modell auf der Jetson-Nano-GPU ausführt. Nach dem Download öffnen Sie die ZIP-Datei. Jetzt ist das Modell bereit für das Deployment mit dem Edge Impulse C++ SDK direkt auf dem NVIDIA Jetson Nano.

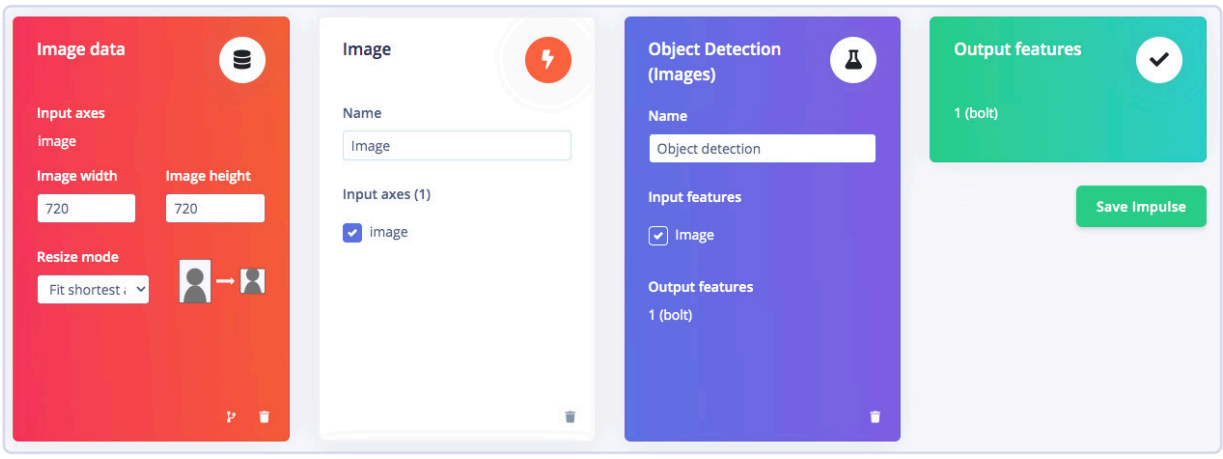


Bild 7. Die verschiedenen Blöcke.

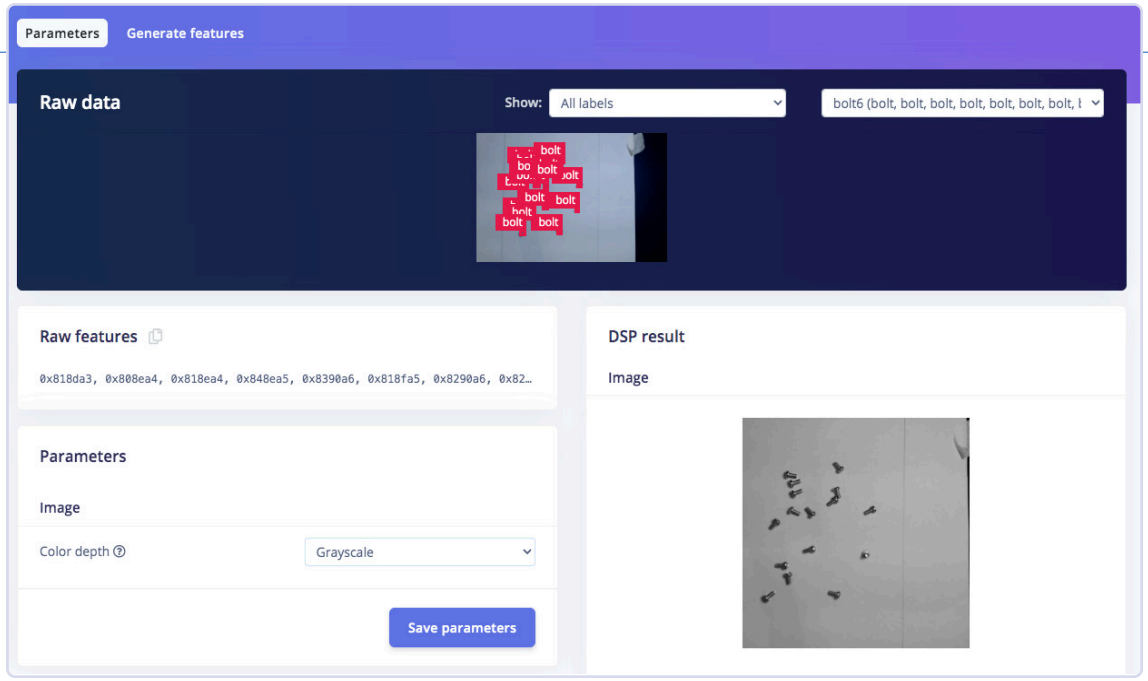


Bild 8. Speichern des Verarbeitungsblocks.

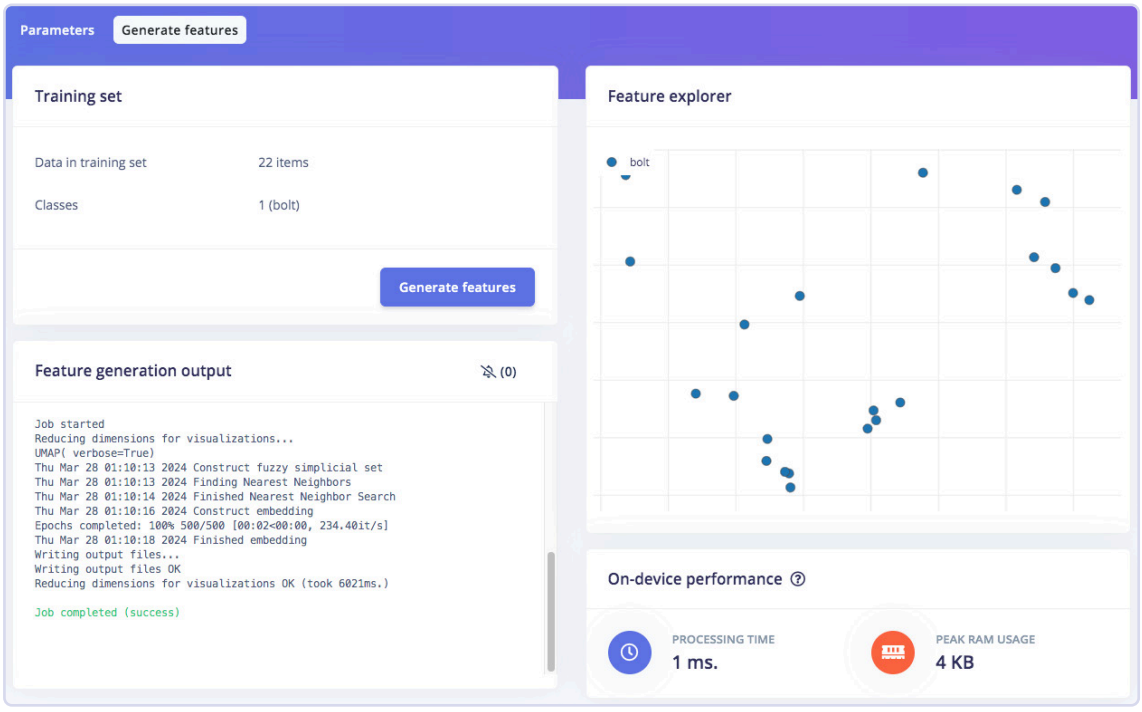


Bild 9. Die Seite „Generate features“.

Auf dem Jetson Nano sind einige Vorbereitungen nötig. Auf dem Gerät sollten das native Ubuntu OS und JetPack laufen, die normalerweise schon auf der SD-Karte installiert sind. Weitere Infos zum Download und Flashen der SD-Karte finden Sie hier [3]. Dann per SSH vom PC oder Laptop mit Ethernet verbinden und die Edge-Impulse-Firmware im Terminal einrichten:

```
wget -q -O - https://cdn.edgeimpulse.com/firmware/
linux/jetson.sh | bash
```

Für den C++-Compiler installieren Sie Clang:

```
sudo apt install -y clang
```

Dann installieren Sie die Edge-Impulse-Inferenzanwendung und weitere Hilfsprogramme mit diesen Befehlen:

```
git clone https://github.com/edgeimpulse/
example-standalone-inferencing-linux
```

```
cd example-standalone-inferencing-linux && git
submodule update --init --recursive
```

Installieren Sie dann OpenCV und die Abhängigkeiten:

```
sh build-opencv-linux.sh
```

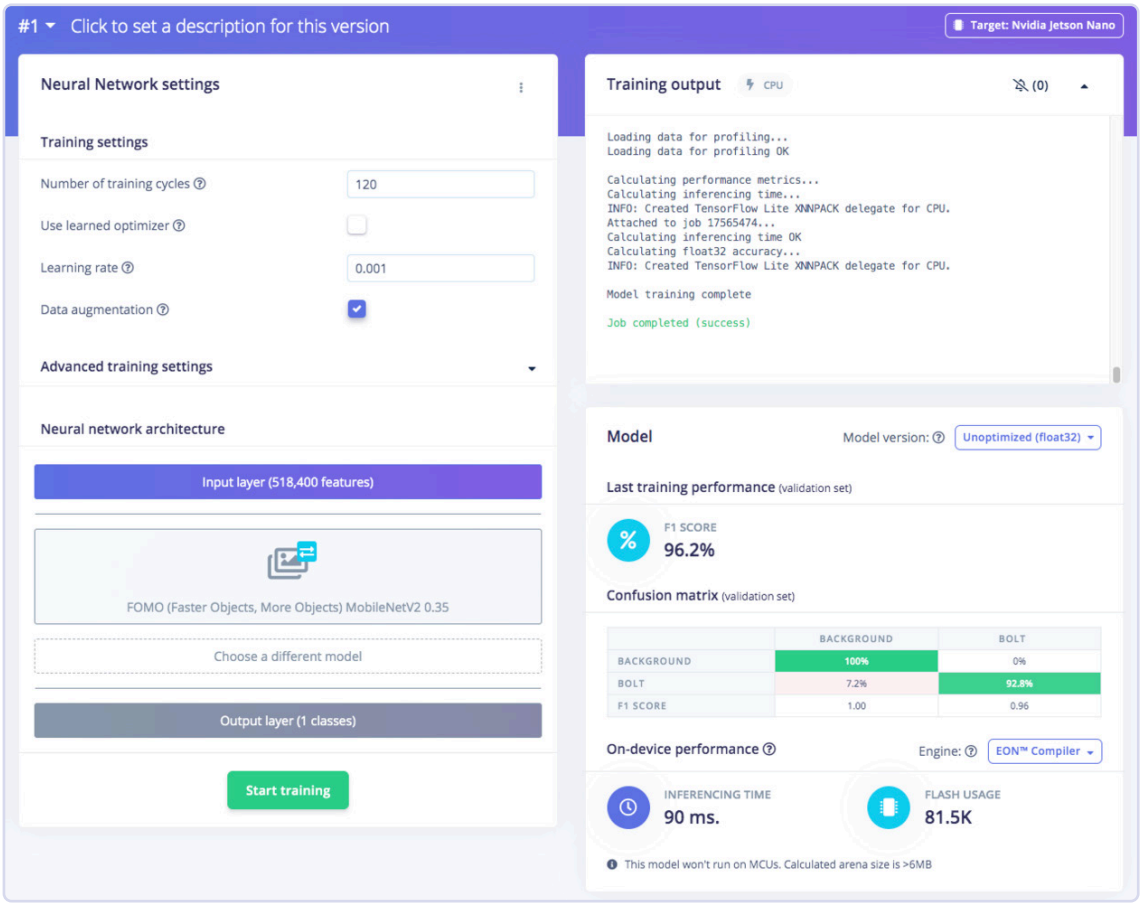


Bild 10. Die Trainingsergebnisse.

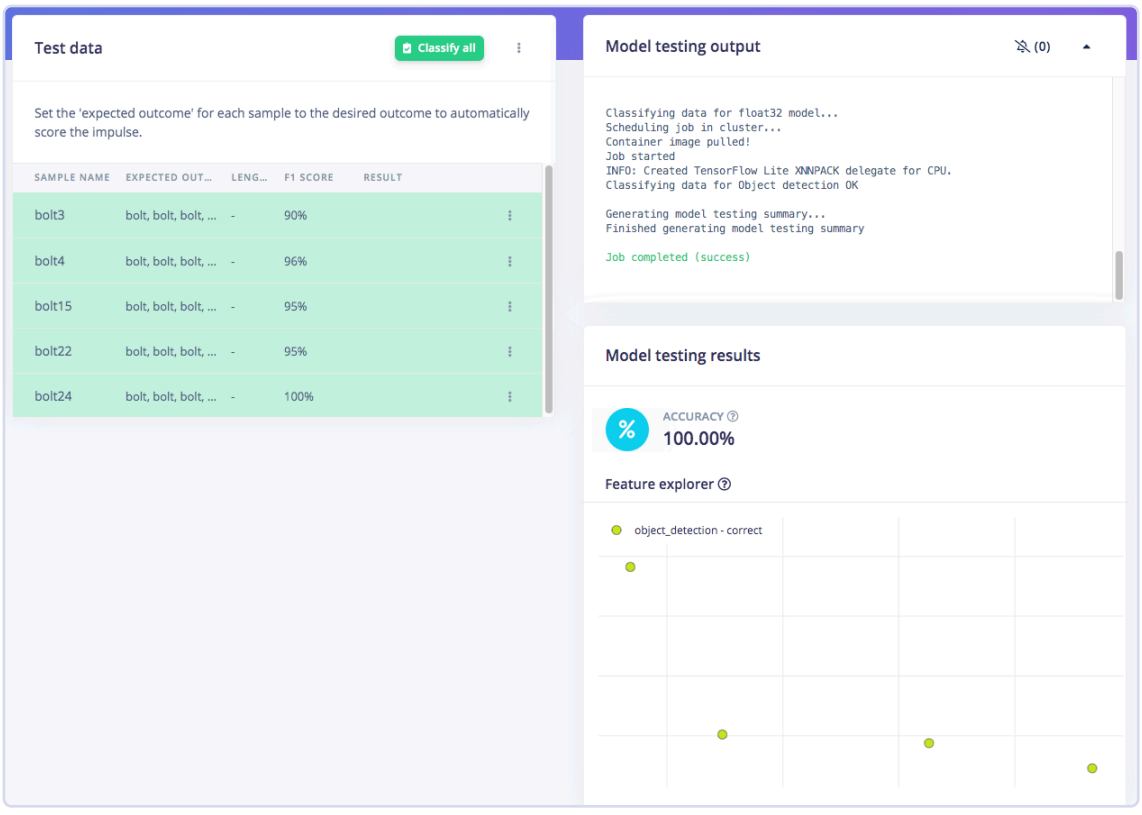


Bild 11. Testdaten und Ergebnisse.

Bauen Sie ein spezifisches Modell für die NVIDIA-Jetson-Nano-GPU mit TensorRT und clang:

```
APP_EIM=1 TARGET_JETSON_NANO=1 make -j
```

Das Ergebnis ist eine lauffähige Datei: */build/model.eim*
Wenn Ihr Jetson Nano mit einer dedizierten Stromversorgung läuft (nicht mit Akku), können Sie die Leistung mit diesem Befehl maximieren:

```
sudo /usr/bin/jetson_clocks
```

Nun kann das Modell in einer Hochsprache laufen, wie im Python-Programm im nächsten Abschnitt. Um sicherzustellen, dass das Modell funktioniert, kann das Tool Edge Impulse Linux CLI Runner mit Kamera auf dem Jetson Nano verwendet werden, während das Förderband läuft. Der Kamerastream ist im Browser sichtbar (**Bild 13**). Die IP-Adresse erscheint beim Start von Edge Impulse Runner. Führen Sie folgenden Befehl aus und folgen Sie den Anweisungen:

```
edge-impulse-linux-runner --model-file /model.eim
```

Die Inferenzzeit liegt bei etwa 15 ms – eine sehr schnelle Erkennung.

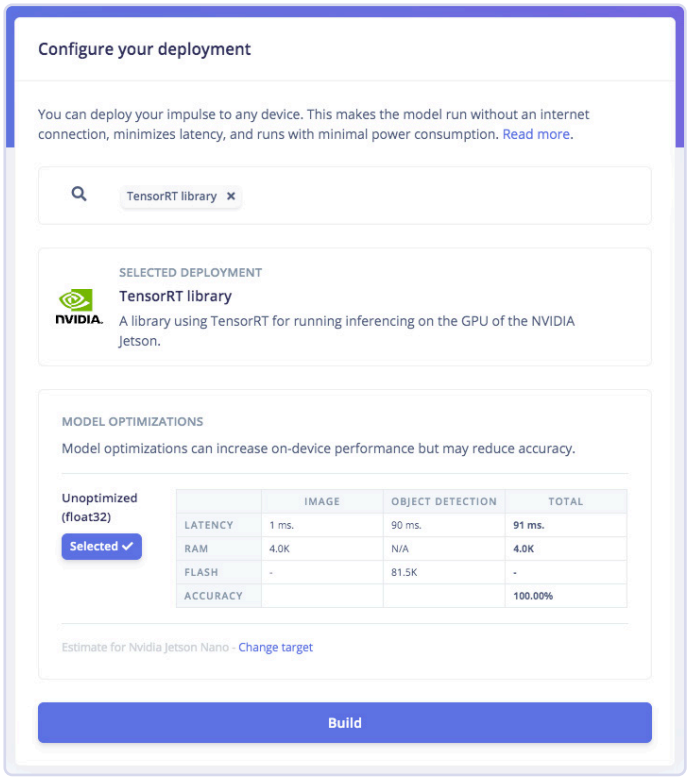


Bild 12. TensorRT-Bibliothek für das Deployment ausgewählt.

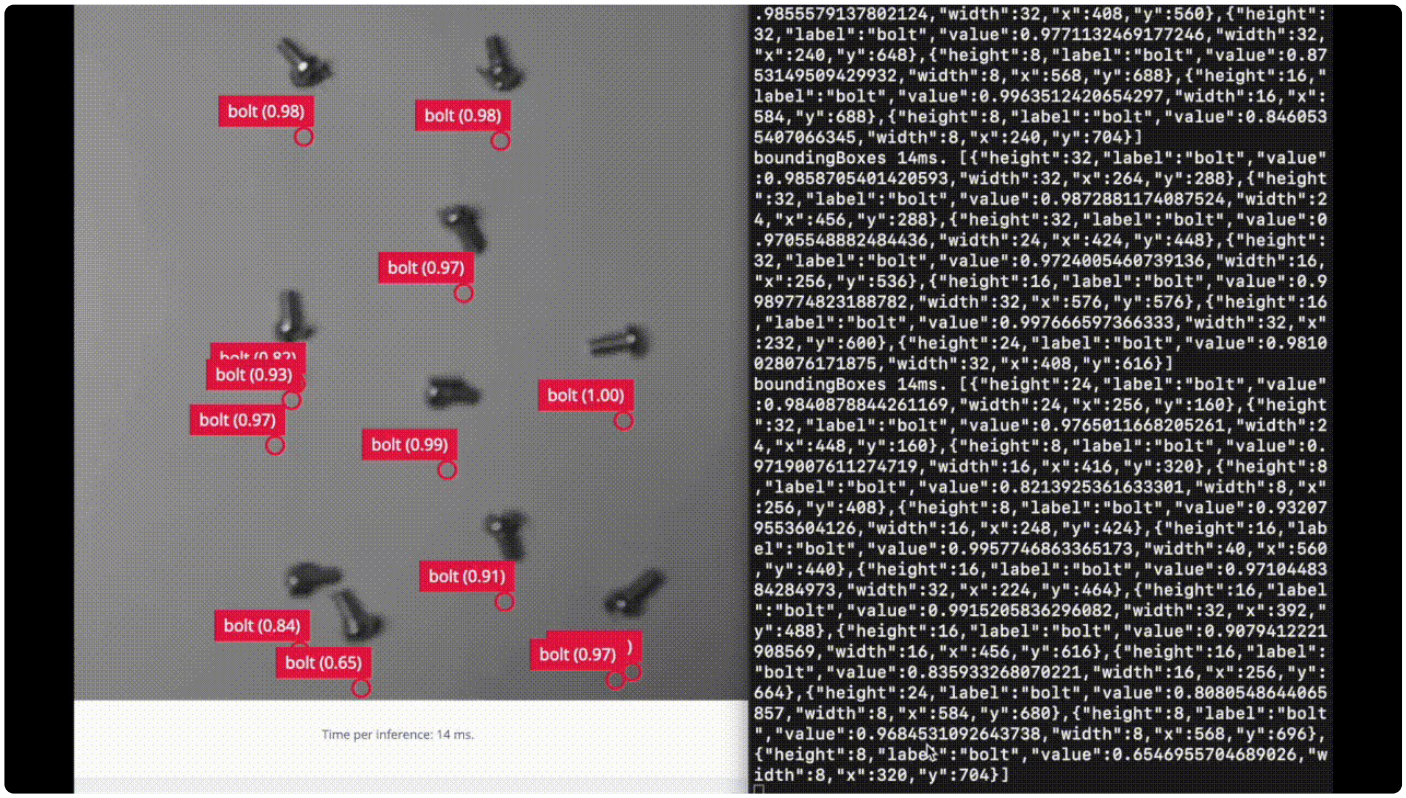


Bild 13. Der Videostream im Browser.

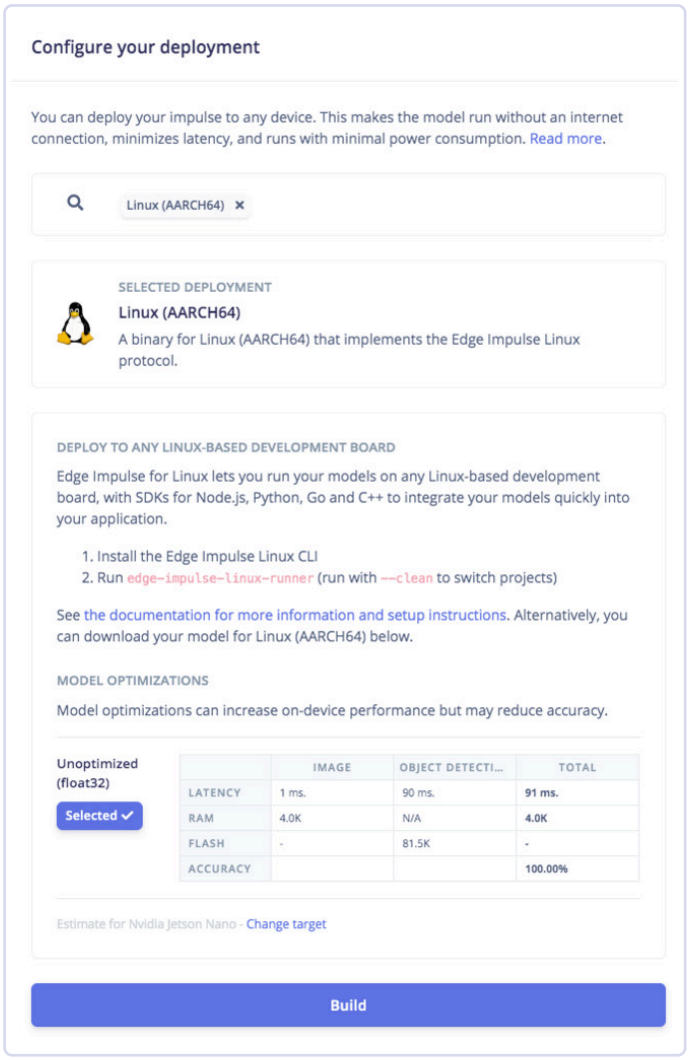


Bild 14. Deployment auf die CPU.

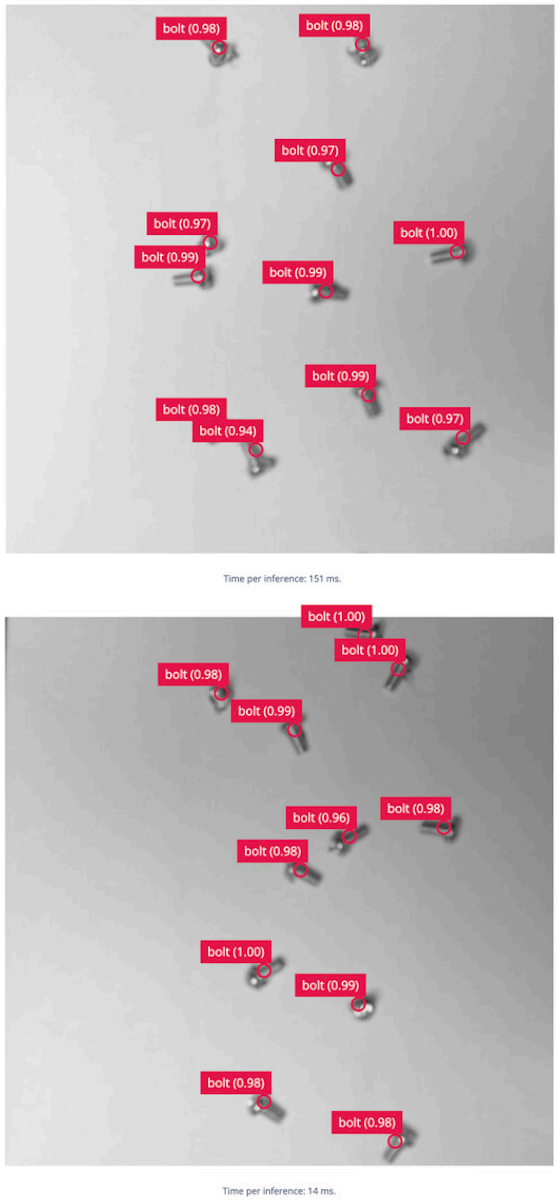


Bild 15. CPU-Performance vs. GPU-Performance.

Zum Vergleich wurde auch ein CPU-basiertes Deployment (Linux-AARCH64-Modell) mit demselben Befehl ausgeführt. Die Inferenzzeit beträgt dort etwa 151 ms (**Bild 14**).
In **Bild 15** sieht man, dass die GPU etwa zehnmal schneller arbeitet – beeindruckend!

5. Programm zur kumulierten Zählung (Python)

Vor dem Start mit Python muss das Edge Impulse Linux Python SDK installiert werden, und das Repository aus den Edge-Impulse-Beispielen wird geklont. Folgen Sie dazu den Schritten in [4].
Mit der beeindruckenden Echtzeit-Inferenzleistung im Runner schreiben wir jetzt ein Python-Programm, das die kumulierte Anzahl der Objekte anhand der Kameraaufnahme berechnet. Das Programm ist eine Modifikation von Edge Impulses *classify.py* im Ordner *examples/image* aus dem *linux-python-sdk*-Repository. Es wurde in ein Objektverfolgungsprogramm umgewandelt, welches das bipartite Zuordnungsproblem löst, sodass dasselbe Objekt in verschiedenen Frames verfolgt werden kann, um Doppelzählungen zu vermeiden. Details und den Python-Code finden Sie zum Download unter [5].
Das Repository kann geklont werden, danach starten Sie das Programm mit dem Pfad zu *model.eim*:

```
python3 count_moving_bolt.py path/to/directory/model.eim
```

Ein Demo-Video zu den Ergebnissen ist unter [5] verfügbar. Die Verzögerung im Videostream und der zugehörigen Ausgabeberechnung entsteht durch das Rendern des 720×720-Fensters durch OpenCV, nicht durch die Inferenzzeit des Objekterkennungsmodells. In diesem Demo-Test werden 30 Objekte pro Zyklus (unsere Schrauben) auf dem Förderband verwendet, um einen Vergleich mit der Anzeige auf dem Zähler zu zeigen (**Bild 16**).

250686-02

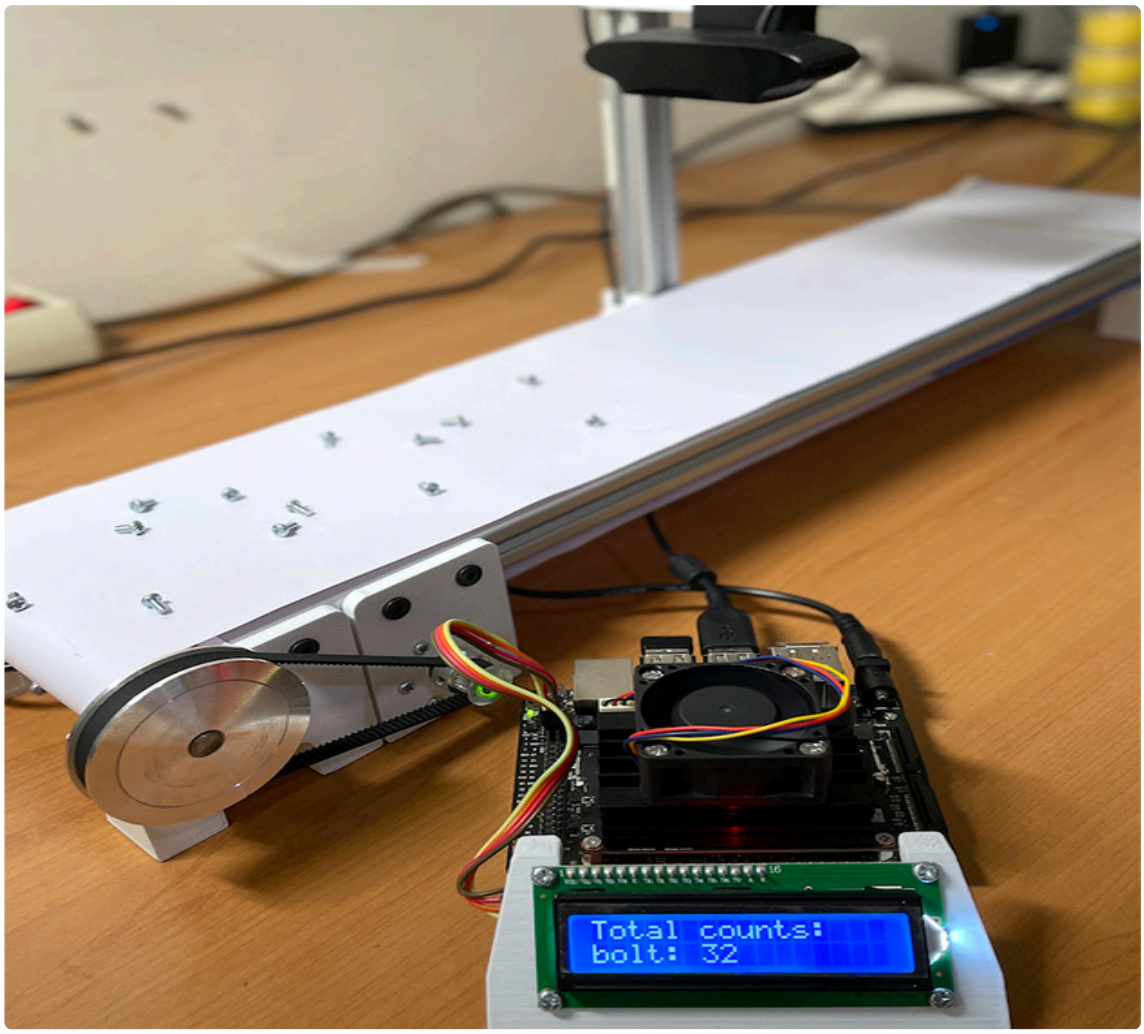


Bild 16. Hochauflösendes, schnelles Objektzählen mit NVIDIA Jetson Nano (TensorRT).

WEBLINKS

[1] „Counting for Inspection and Quality Control – Nvidia Jetson Nano (TensorRT)", Edge Impulse: https://docs.edgeimpulse.com/projects/expert-network/quality-control-jetson-nano
[2] Edge Impulse FOMO: https://docs.edgeimpulse.com/studio/projects/learning-blocks/blocks/object-detection/fomo
[3] Herunterladen und Flashen der SD-Karte : https://docs.edgeimpulse.com/projects/expert-network/high-speed-counting-jetson-nano
[4] Linux-basiertes Python-SDK : https://docs.edgeimpulse.com/tools/libraries/sdks/inference/linux/python
[5] Python-Programm, GitHub: https://github.com/Jallson/High_res_hi_speed_object_counting_FOMO_720x720
[6] „Hi-resolution, hi-speed Object Counting; DEMO TEST with 30 bolts per cycle", YouTube: https://youtube.com/watch?v=ouqvACe48ts
[7] Öffentlich zugänglicher Projektlink: https://studio.edgeimpulse.com/public/207728/live

DSP entmystifiziert

Von Alex Elium (Edge Impulse)

Digitale Signalverarbeitung (DSP) ist ein zentraler Bestandteil der Edge-KI. Erfahren Sie, wie Verfahren wie schnelle Fourier-Transformation, Spektrogramme und Wavelets Ihren Machine-Learning-Modellen helfen, ähnlich wie Menschen zu „hören".

Wenn Sie das nächste Mal einen Zeitreihendatensatz vorliegen haben – sei es von einem gewöhnlichen Mikrofon oder einem exotischen Wandler –, nutzen Sie DSP zur Merkmalsextraktion, um Ihr Modell deutlich zu verkleinern.

DSP, kurz für digitale Signalverarbeitung (digital signal processing), verwandelt Zeitreihendaten in eine kompaktere und nützlichere Form, während die wichtigsten Merkmale erhalten bleiben. Wenn Sie noch die Anfänge von Napster miterlebt haben, haben Sie dies zum ersten Mal erlebt, als Sie eine kleine MP3-Datei Ihres Lieblingssongs heruntergeladen haben, statt zu versuchen, eine riesige, unkomprimierte WAV-Datei über eine DFÜ-Verbindung zu übertragen. Das damals populär gewordene MP3-Format war nur einen Bruchteil so groß und klang für die Zuhörer dennoch fast identisch. Der Trick war DSP. Durch das Codieren der Audiodaten abhängig von der Energieverteilung über die Frequenzbänder statt des Übertragens der Rohwellenform reduzierte sich die Dateigröße drastisch, ohne dass das Wesentliche der Musik verloren ging. Für Sie rockten die White Stripes immer noch, und es klang mehr oder weniger wie das Original.

Warum DSP mit ML verwenden?

Was hat das nun mit Edge-KI zu tun? Auch wenn das viel zitierte Versprechen der KI so klingt, als müsse man „dem Modell einfach nur genügend Daten geben, und dann findet es schon das Richtige raus", ist dies für die meisten Entwickler nicht praktikabel. In der realen Welt des Edge Computing sind Daten in der Regel teuer, zeitaufwendig zu erfassen oder schwer zu bekommen – meist alles zusammen. Die Reduzierung der Eingabegröße für Ihr Modell, auch bekannt als Reduzierung der Dimensionalität Ihrer Daten, ermöglicht es Ihnen, ein kleineres Modell zu trainieren. Kleinere Modelle benötigen weniger Daten, um einen brauchbaren Zustand zu erreichen. Wenn Sie es quantifizieren möchten, lautet eine grobe Faustregel: Sie brauchen etwa ein Daten-Sample für jedes zu trainierende Gewicht. Weniger Knoten – weniger Datenbedarf. Und wenn das Modell auf einem Mikrocontroller laufen soll, profitieren Sie auch davon, dass weniger Gewichte weniger Flash-Speicher und weniger RAM für weniger Multiplikationen benötigen.

Einstimmen auf die richtige Frequenz

Um die DSP-Vorverarbeitung für Machine-Learning-Modelle – oder genauer gesagt, die Merkmalsextraktion – zu verstehen, sollten wir uns den Frequenzbereich ansehen. Ein Signal kann von einem Mikrofon oder einem anderen Wandler stammen. Zur Veranschaulichung konzentrieren wir uns auf Audio, da es intuitiv ist, obwohl alles, was hier besprochen wird, gleichermaßen

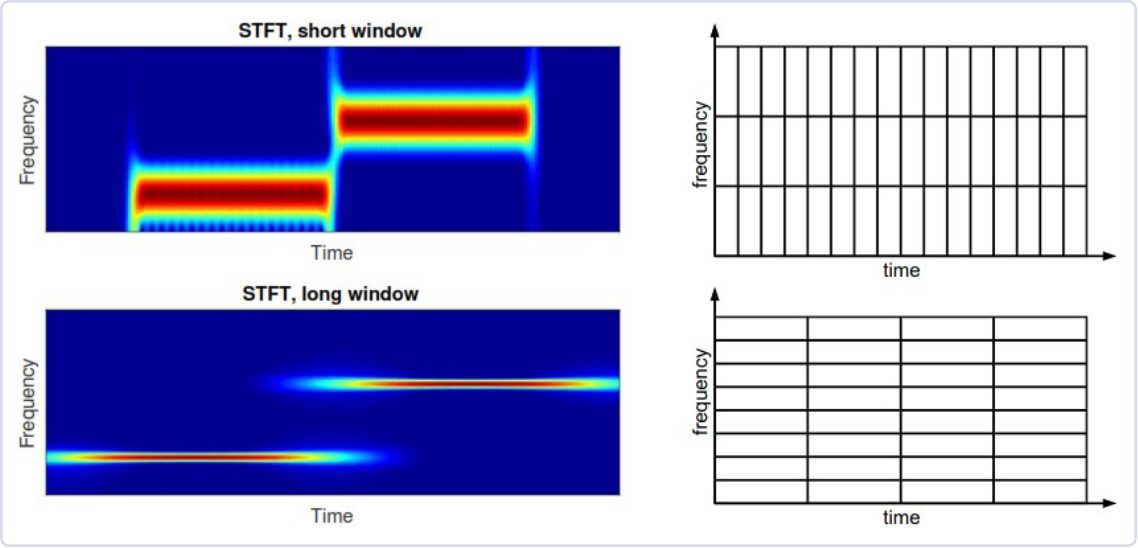


Bild 1. STFT (Spektrogramm) mit kurzen und langen Fenstern, zeigt den Unterschied in Frequenz- und Zeitauflösung. (Quelle: Panoradio SDR [1])

für andere Zeitreihensignale wie z. B. Beschleunigungsdaten gilt. Wandler erzeugen in der Regel ein Zeitreihensignal, aber das ist nicht immer die effizienteste Art, das Signal auszudrücken, etwa beim Streaming über das Internet oder beim Training eines ML-Modells. Oft ist es am besten, die zugrundeliegenden Frequenzkomponenten des Signals darzustellen. Das klingt vielleicht zunächst ungewohnt, ist aber tatsächlich die Art, wie wir Klang wahrnehmen.

Stellen Sie sich vor, jemand spielt einen einzelnen Ton auf einer Gitarre und nimmt diesen mit einem Mikrofon auf. Die schwingende Saite erzeugt eine Grundfrequenz (zusätzlich zu weiteren Obertönen, die wir zur Vereinfachung ignorieren). Auf einem Oszilloskop sehen Sie diesen Ton im sogenannten Zeitbereich, der in diesem Fall ein wenig so aussehen würde wie die schwingende Saite selbst. Über einen Lautsprecher abgespielt, ist das Signal als Gitarrenton erkennbar. Ein ausgebildeter Musiker könnte sogar die genaue Tonhöhe erkennen – also die Frequenz –, was eine wesentlich kompaktere Beschreibung des Klangs liefert als die Oszilloskopanzeige.

Hören wie eine Maschine

Genauso wie Menschen Frequenzen wahrnehmen, möchten wir, dass unsere Machine-Learning-Modelle das auch tun. Nun können wir nicht einfach Ohren abschneiden und an Leiterplatten befestigen, aber zum Glück hat ein Mann namens Joseph Fourier ein mathematisches Äquivalent erfunden. Die Fourier-Transformation wandelt Zeitreihensignale in Frequenzspektren um. Häufiger begegnet uns die FFT (Fast Fourier Transform), die schlicht ein Algorithmus ist, um diese Transformation – Sie ahnen es – schnell durchzuführen.

Die FFT bietet Entwicklern viele Parameter zur Einstellung. Die wichtigste Frage ist, ob man ein Modell nur mit einer Liste von Frequenzamplituden trainieren sollte oder besser eine Darstellung verwendet, die einem Bild ähnelt. Edge Impulse bietet diese Auswahl in Form zweier verschiedener DSP-Verarbeitungsblöcke an: *Spectral Analysis* und *Spectrogram* (und einige weitere Varianten).

Es ist wichtig, einige Grundregeln und Begriffe zu klären. Obwohl einige Modelle das fortlaufende Einlesen von Daten unterstützen, kann Edge Impulse sowohl das Training als auch die Inferenz auf einem gesamten Datenblock durchführen, den wir „Fenster" nennen. (Wir können zur Inferenzzeit den Klassifikator auch mit kleineren Datenmengen aufrufen und alte Berechnungen wiederverwenden, wie in einem Rollpuffer, aber das ist hier nicht wichtig.) Die *Fenstergröße* ist ein weiterer wichtiger Parameter beim Experimentieren; längere Fenster erfordern mehr Ressourcen und Sie brauchen in der Regel ein Fenster, das lang genug ist, um Ihr längstes Ereignis zu erfassen. Wenn Sie zum Beispiel erkennen wollen, ob jemand „Hallo Edge Impulse" sagt und das maximal eine Sekunde dauert, sollten Sie ein einsekündiges Fenster verwenden.

Für jede Inferenz wird das gesamte Fenster mittels DSP-basierter Merkmalsextraktion transformiert. Diese Merkmale dienen dann als Eingabe für das neuronale Netz (oder ein anderes Modell), und das Ergebnis ist entweder eine Klassifikation (z. B. „Katze" oder „Hund") oder ein Regressionswert (z. B. eine numerische Bewertung von 1 bis 10).

Spektrogramme: Die Grundlage der Merkmalsextraktion

Lassen Sie uns einige nützliche Algorithmen zur Merkmalsextraktion durchgehen, die in Edge Impulse „Blöcke" genannt werden. Wir beginnen mit dem Spektrogramm, weil die anderen Blöcke darauf aufbauen. Wenn wir zum Gitarrenton von vorhin zurückkehren und diesen in ein Spektrogramm umwandeln, würden Sie ein paar Linien sehen – die Y-Koordinaten der Linien zeigen die Frequenzen, mit denen die Saite schwingt, und die X-Achse zeigt Startzeit und Dauer der Schwingung.

Wegen der Einschränkungen des FFT-Algorithmus, der dieses Spektrogramm erzeugt, müssen Sie das Signal in einem bestimmten Intervall abtasten, was bedeutet, dass Ihre Start- und Stoppzeiten nicht exakt sind. Dieses Intervall ist einer der oben erwähnten Parameter, und die Anpassung kann einen enormen Einfluss auf die endgültige Modellleistung haben. Leider gibt es keine beste Wahl für alle Fälle. Tatsächlich gibt es einen grundsätzlichen Zielkonflikt auf physikalischer Ebene (für Sie als Super-Nerds: es ist ein bisschen wie das Heisenberg'sche Unschärfeprinzip: Längere Intervalle liefern genauere, höher aufgelöste Frequenzinformationen, verlieren aber an Auflösung in Bezug auf das „Wann" der Signale). Ein kurzes, impulsartiges Signal, das viel kürzer als das Intervall ist, kann völlig übersehen werden.

Weil das so wichtig ist, können Sie auf Plattformen wie Edge Impulse mit verschiede-

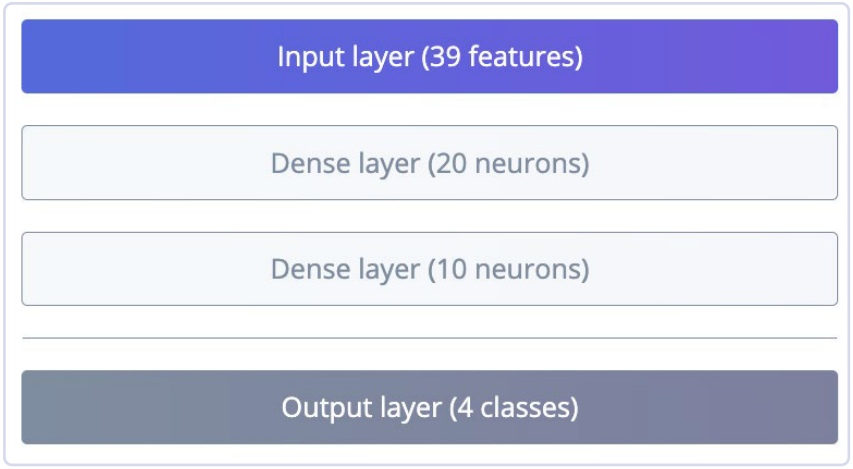


Bild 2. Zweischichtiges Netzwerk mit 39 Merkmalen pro 1.000-Punkte-Input-Fenster.

nen Einstellungen experimentieren und die Auswirkungen auf Zwischenoutputs und auf die endgültige Modellgenauigkeit prüfen.

Da Spektrogramme wie Bilder aussehen (**Bild 1**), werden sie mit Modellarchitekturen trainiert, die ursprünglich für Fotografien entwickelt wurden – mit Faltungsschichten (convolutional layers). Das Modell kann Merkmale wie schnell ansteigende Frequenzen oder Lücken in verschiedenen Tönen lernen, was zur Trennung der Klassen beiträgt.

Zweidimensionale Faltungsschichten schaffen räumliche Invarianz, das heißt, das Modell lernt, wichtige Merkmale unabhängig vom Ort im Spektrogramm zu erkennen. Zum Beispiel möchten Sie nicht, dass Ihr Modell Hunde nur dann erkennt, wenn sie in der oberen linken Ecke eines Bildes erscheinen, nur weil Ihre Trainingsdaten dort viele Hunde zeigten. Ein Hund ist überall ein Hund! Dasselbe gilt für Merkmale im Spektrogramm.

Spektralanalyse: kompakte Merkmalsextraktion

Der Spektralanalyse-Block arbeitet ähnlich, indem er eine FFT in einem definierten Intervall durchführt. Allerdings erzeugt er statt eines zweidimensionalen Outputs ein einfacheres, eindimensionales Ergebnis, indem er einfach die maximale Leistung in jedem Frequenzband über das gesamte Eingabefenster hinweg ermittelt. Auch hier werden FFTs in festgelegten Abständen über das Eingabefenster hinweg berechnet, statt nur eine große FFT durchzuführen – das spart viele Ressourcen und kann auch kurzlebige Signale erfassen. Auch Edge Impulse Studio bietet Einstellungen für FFT-Größe und Intervall.

Das Reduzieren der Merkmale auf ein einfaches Array wirkt sich auch auf die Modellarchitektur aus. (Keine Sorge, Edge Impulse erledigt das alles automatisch.) Räumliche Invarianzen, wie oben erwähnt, sind kein Thema mehr, da das gesamte Eingabefenster in eine einzige Array-Darstellung transformiert wurde. Die Zeitachse entfällt also. Ist das wichtig? Es kommt darauf an! Genau deshalb ist schnelles Experimentieren in ML-Plattformen wie Edge Impulse so wichtig.

Angenommen, Spektrogramm- und Spektralanalyseblock liefern ähnliche Genauigkeiten, dann benötigt letzterer wegen der kleineren Eingabegröße weniger Ressourcen (zur Erinnerung: Er betrachtet nur das Maximum über die Zeit und entfernt die Zeitachse). Zusätzlich zu den Eingabegrößenunterschieden kann eine flache Array-Eingabe mit einfachen, voll verbundenen Schichten verarbeitet werden statt mit größeren Faltungsschichten.

Sehen wir uns ein Beispiel an. In **Bild 2** und **Bild 3** gibt es zwei Netzwerke für zwei verschiedene Projekte (wir nennen sie *Impulse*). Bild 2 zeigt ein einfaches neuronales Netz mit zwei versteckten Schichten. Dieses Projekt verwendet einen Merkmalsextraktionsblock vom Typ *Spectral Analysis DSP*, um 39 Merkmale für jedes Eingabefenster mit 1.000 Punkten zu extrahieren. Der Prozess besteht darin, wiederholt eine 64-Punkte- FFT in festgelegten Abständen durchzuführen und dann die maximale Leistung in jedem Frequenzbereich der generierten FFTs zu berechnen. Wir berücksichtigen nur 33 Frequenzmerkmale, da die zweite Hälfte einer FFT doppelte Informationen enthält. Wir fügen auch einige weitere statistische Merkmale hinzu, wie RMS und Kurtosis, was die Gesamtanzahl leicht über 33 steigen lässt.

Bild 3 zeigt das Netzwerk für die Verarbeitung eines Spektrogrammeingangs. Der Input wird aus internen Gründen als flaches Array mit 3.960 Merkmalen angezeigt, aber die erste Schicht, eine Reshape-Schicht, weist das Netz an, das „Bild“ als 40 × 99 Inputs zu verarbeiten. Zwei Faltungsschichten

Bild 3. Netzwerk mit zwei Faltungsschichten – die Eingabeschicht ist ein flaches Array von 3.960 Merkmalen.

füttern eine abschließende dichte Schicht. Der Unterschied beim Ressourcenbedarf zeigt sich am besten in der Zahl der Gewichte – oder noch besser in deren Flash-Speicherbedarf: Das voll verbundene Modell benötigt 15,5 kB Flash, das Faltungsmodell aber 4,4 MB!

Warum sollte man sich angesichts der Modellgrößen also überhaupt mit einem Spektrogramm abmühen? Manche DSP-Algorithmen entfernen zu viele Informationen, sodass das Modell die Klassen nicht mehr effektiv trennen kann – also z. B. nicht mehr zwischen „an" und „aus" oder etwas ganz anderem unterscheiden kann. In solchen Fällen brauchen Sie ein vollständiges Spektrogramm. Diese Art Modell nennt man Keyword-Spotting (KWS), und der größere 2D-Modell-Input ist nötig, um Wörter zu erkennen.

Noch mehr Möglichkeiten: Wavelet-Transformation und Flatten

Wir haben jetzt die wichtigsten Werkzeuge der DSP vorgestellt, aber wie in jedem guten Werkzeugkasten gibt es noch mehr. Werfen wir einen kurzen Blick auf weitere Möglichkeiten in Edge Impulse.

Für den Spektralanalyse-Block gibt es in Edge Impulse auch einen anderen Algorithmus, genannt Wavelet-Transformation, der manchmal nützlichere Informationen liefern kann als eine FFT; Edge Impulse Studio kann Merkmale mit über 40 verschiedenen Wavelet-Kernen erzeugen. Wenn Ihnen jetzt schon der Mausfinger weh tut bei dem Gedanken an so viele Experimente, keine Sorge: Edge Impulse bietet einen Autotuner, der vorschlägt, ob Sie FFT oder Wavelet verwenden sollten und welche Einstellungen optimal sind. Dieser Autotuner funktioniert mit allen unseren Verarbeitungsblöcken.

Wir haben auch einen sehr einfachen Block namens *Flatten* , mit dem Sie einige einfache Statistiken auswählen können. Manchmal reicht das schon aus, aber oft ist es hilfreich, diesen Block zusammen mit einem anderen, etwa der Spektralanalyse, zu nutzen und beide Merkmalsmengen für das Modelltraining zu kombinieren.

Weitere Blöcke

Edge Impulse bietet zwei Varianten von Spektrogrammen, die oft in der Sprachverarbeitung genutzt werden. Der Block *Mel-Frequency Energy (MFE)* startet mit einem normalen Spektrogramm, mittelt dann aber die höheren Frequenzbereiche zu einer kompakteren Darstellung (kleinerer Input – kleineres Modell.) Die Mittelungsformel wurde von Paul Mermelstein 1976 veröffentlicht, als Computer noch ganze Wände füllten! Die Grundlage der Formel ist, dass das menschliche Ohr Unterschiede in tiefen Frequenzen besser wahrnimmt als in hohen. Ein neuronales Netz zur Worterkennung kann das ausnutzen, denn auch unsere Sprache hat sich an unser Hörvermögen angepasst.

Edge Impulse bietet noch einen ähnlichen Block namens *Mel-Frequency Cepstral Coefficients (MFCC)*, der einen zusätzlichen Schritt zur oben beschriebenen MFE durchführt: Es wird eine diskrete Cosinus-Transformation (DCT) angewendet und einige Werte werden verworfen. Dieser Schritt wurde bereits im Originalartikel von 1976 empfohlen, aber Experten streiten sich, ob das mit modernen Trainingsmethoden (z. B. neuronalen Netzen) noch nötig ist. Der beste Weg: beide ausprobieren!

Brauchen Sie immer Merkmalsextraktion?

All das heißt nicht, dass DSP für alle Zeitreihendatensätze verwendet wird. Wenn die Samples hauptsächlich aus kurzlebigen, ungewöhnlich geformten Signalen – sogenannten *Transienten* – bestehen, kann eine Transformation dieser Signale alle nützlichen Informationen zerstören. In solchen Fällen bringt das Training des Modells direkt mit den Rohdaten oft die besten Ergebnisse. Erinnern Sie sich: 2D-Faltungsnetze erlauben es, dass Objekte in Bildern lageunabhängig erkannt werden („Der Hund kann überall im Foto sein"). 1D-Faltungsschichten und -netze tun das Gleiche für Zeitreihen und erlauben es Modellen, Muster überall im Eingabefenster zu erkennen.

Und wie ist es bei Bildern? Auch die werden manchmal vorverarbeitet. In Edge Impulse geht es meist um die Verkleinerung der Eingabebilder. Das Herunterrechnen auf 200 × 200 oder 96 × 96 Pixel reduziert die Modellgröße und erhält oft dennoch eine akzeptable Genauigkeit. Manche einfachen Datensätze funktionieren sogar schon mit 32 × 32! Diese kleinen Größen sind besonders geeignet für eingeschränkte Systeme wie Mikrocontroller.

Die Macht der DSP

Wenn Sie das nächste Mal einen Zeitreihendatensatz vorliegen haben, sei es von einem gewöhnlichen Mikrofon oder einem exotischen Wandler, nutzen Sie DSP zur Merkmalsextraktion, um Ihr Modell deutlich zu verkleinern. Und wenn Sie ohnehin wenig Trainingsdaten haben, können Sie mit DSP noch mehr Genauigkeit aus Ihren wenigen Samples herausholen.

250786-02

Über den Autor

Alex Elium ist Forschungsingenieur bei Edge Impulse. Er verfügt über mehr als 20 Jahre Erfahrung in Embedded-Software und Signalverarbeitung.

Fragen oder Kommentare?

Haben Sie technische Fragen oder Rückmeldungen zu diesem Artikel? Bitte wenden Sie sich an den Autor im Edge-Impulse-Forum (forum.edgeimpulse.com, @AlexE) oder an das Elektor-Redaktionsteam unter redaktion@elektor.de.

WEB LINK

[1] Panoradio SDR: https://panoradio-sdr.de/

Edge-KI-Arbeitsplätze

Gute Organisation sorgt für gute Daten und mehr

Source: Adobe Stock

Haben Sie sich schon einmal gefragt, wo die Innovatoren von Edge Impulse ihre Machine-Learning-Ideen zum Leben erwecken? Interessieren Sie sich für die Werkzeuge, die Hardware und die Aufbauten hinter den Modellen, die auf Edge-Geräten laufen? Einige Mitarbeitende von Edge Impulse öffnen die Türen zu ihren Arbeitsplätzen und zeigen, wie sie an der Schnittstelle von Hardware und KI entwickeln, testen und implementieren.

Das Developer-Relations-Team spielt bei einem Softwareunternehmen wie Edge Impulse eine wichtige Rolle. Es hilft externen Nutzern, neue Funktionen zu entdecken und zu nutzen, sammelt dabei Informationen darüber, wie diese genutzt werden und welche Ergänzungen eventuell erforderlich sind. Die DevRel-Ingenieure von Edge Impulse erstellen einen Großteil der öffentlich zugänglichen Inhalte des Unternehmens, darunter technische Dokumentationen, Webinar-Anleitungen und Erklärvideos, führen Workshops für Community-Mitglieder durch und nehmen an Veranstaltungen teil. Daher gehören der praktische Umgang mit derselben Hardware und Software wie bei den Nutzern sowie die Vorbereitung auf die nächste Veröffentlichungsrunde zum Arbeitsalltag.

Werfen wir einen Blick auf drei Arbeitsplätze von Mitgliedern des Edge-Impulse-Teams. Lesen Sie weiter, um zu erfahren, was für ihre Arbeit wichtig ist.

250825-02

Louis Moreau
(Ort: Lille, Hauts-de-France, Frankreich)

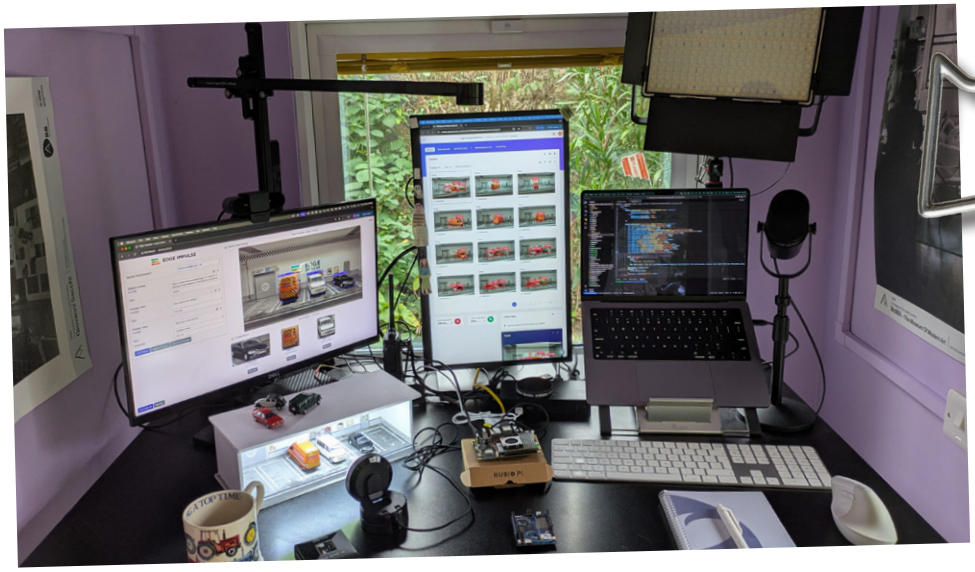
Mein Schreibtisch sieht vielleicht chaotisch aus, aber er ist auf seine eigene Weise organisiert. Ich halte häufig verwendete Hardware – wie Arduino-, ESP32- und Raspberry-Pi-Boards – in Reichweite für schnelle Tests. Andere Hardware, die ich seltener benutze, wie meine Lötstation, das Oszilloskop oder das Multimeter, bewahre ich in den Aufbewahrungsboxen auf, sodass sie trotzdem leicht zugänglich sind.

Ich nutze diesen Bereich auch als Aufnahmestudio. Ich habe mehrere Kameras und Mikrofone bereit, und ich schätze das natürliche Tageslicht dabei sehr. Es verbessert die Videoqualität erheblich.

Jim Bruges

(Ort: York, England)

Ich nutze mein Büro für viele verschiedene Zwecke, deshalb sieht man oft eine Mischung aus unfertigen Demos, Edge-Hardware und Aufnahme-Equipment auf meinem Schreibtisch. Im Moment erstelle ich einen Datensatz für eine Parkplatzüberwachungs-Demo mit dem Diorama unter dem linken Computermonitor und einem Rubik-Pi-3-Entwicklerboard. Ohne meinen vertikalen Monitor könnte ich nicht arbeiten; das ist zwar nicht jedermanns Sache, aber nur so kann ich Hunderten Slack-Kanälen folgen!



Marc Pous

(Ort: Barcelona, Spanien)

Das ist mein Schreibtisch mit einem Multi-Monitor-Setup, das für meinen Arbeitsablauf entscheidend ist.

Ich versuche immer, die Hardware meines aktuellen Projekts (meist mit Arduinos und Raspberry Pis) direkt vor mir zu haben, mit den benötigten Kabeln für schnelle Iterationen. Alles andere hat seinen Platz im hölzernen Organizer, sodass Werkzeuge und Kabel, die ich brauche, immer griffbereit sind. Außerdem habe ich mir ein System gebaut, um die Lade- und Netzwerkkabel meines Setups zu verstecken.

Und natürlich brauche ich immer ein Getränk während meiner Arbeitssessions. Denken Sie daran: Immer genug trinken.

Zeigen Sie Ihren Elektronik-Arbeitsplatz!

Möchten Sie, dass die Redaktion von Elektor Ihren Elektronik-Arbeitsplatz auf elektormagazine.com oder auf den Seiten des *Elektor-Magazins* vorstellt? Nutzen Sie unser Online-Formular zur Einsendung von Arbeitsplätzen, um Details über den Ort zu teilen, an dem Sie entwickeln, konstruieren, testen und programmieren! **www.elektormagazine.com/workspaces**

Edge-KI im Operationssaal

Ein tragbares KI-System zur Echtzeiterkennung chirurgischer Instrumente

Von Eivind Holt (Norwegen)

Dieses Projekt untersucht das Konzept eines tragbaren Geräts, das chirurgische Instrumente und Materialien automatisch verfolgt, um das Zurücklassen von Operationsgegenständen im Patienten zu verhindern. Durch den Einsatz von Machine Learning und synthetischer Datengenerierung mit NVIDIA Omniverse und Edge Impulse wird gezeigt, wie Objekterkennungsmodelle auf kompakter, ressourcenbeschränkter Hardware wie dem Arduino Nicla Vision ausgeführt werden können.

Jedes Jahr bleibt bei einer kleinen, aber signifikanten Anzahl chirurgischer Eingriffe ein Gegenstand unbeabsichtigt im Patienten zurück – ein medizinischer Fehler, der als Retained Surgical Body (RSB) bekannt ist. Trotz strenger Zählroutinen vor, während und nach der Operation passieren immer noch Fehler. Schon ein einziger derartiger Vorfall kann zu schweren Komplikationen oder zum Tod führen.

In der aktuellen chirurgischen Praxis werden Geräte und Materialien sorgfältig organisiert und nachverfolgt. Instrumente sind in Sets für bestimmte Eingriffe verpackt, während Tupfer gezählt, markiert und für eine einfache visuelle Überprüfung ausgelegt werden. Trotz dieser etablierten Routinen treten Fehler mit einer geschätzten Häufigkeit von 0,3 bis 1 pro 1000 abdominalen Eingriffen [1] auf. Bestehende technologische Hilfsmittel zur Minimierung von RSB beruhen typischerweise auf Röntgen- oder RFID-Erkennung. Röntgenscans erfordern mobile Geräte und Schutzkleidung, und während Metallwerkzeuge leicht zu erkennen sind, müssen Textilmaterialien wie Tupfer zur Detektion mit Metallstreifen versehen werden. RFID-basierte Systeme nutzen eingebettete passive Schaltungen, die mit Handscannern erkannt werden können, aber diese Lösungen erhöhen die Komplexität und die Kosten im chirurgischen Ablauf.

Um das Risiko von RSB weiter zu reduzieren, werden neue Technologien erforscht, die chirurgische Teams bei der automatischen Nachverfolgung

Projektanforderungen

Hardware
- Arduino Nicla Vision [2]
- NVIDIA GeForce RTX 3090 (jede RTX ist geeignet)
- 3D-Drucker Formlabs Form 2
- Chirurgisches Instrumentarium

Software
- Edge Impulse Studio [3]
- NVIDIA Omniverse [4] mit Replicator [5]
- Visual Studio Code [6]
- Blender [7]
- Autodesk Fusion 360 [8]

Bild 1. Matte und reflektierende Oberflächen.

von Instrumenten und Materialien unterstützen. Dieses Projekt ist ein Proof-of-Concept für ein tragbares Gerät, das helfen soll, die während der Operation verwendeten Instrumente und Einwegmaterialien zu überwachen. Durch die automatische Zählung der erfassten Gegenstände dient es als zusätzliche Absicherung gegen RSB.

Objekterkennung mit neuronalen Netzen

Für diese Lösung kann die webbasierte Entwicklungsplattform von Edge Impulse für maschinelles Lernen genutzt werden. Das neuartige FOMO-Verfahren (Faster Objects, More Objects) [9] ist ein Algorithmus für maschinelles Lernen, der visuelle Objekterkennung auf stark ressourcenbeschränkten Geräten ermöglicht, indem ein neuronales Netz mit mehreren Faltungsschichten trainiert wird. Dies funktioniert hervorragend mit dem Kamera-Board Arduino Nicla Vision, das hier verwendet wird.

Herausforderungen

Reflektierende Oberflächen

Als ob die Objekterkennung auf stark ressourcenbeschränkten Geräten nicht schon herausfordernd genug wäre, stellt dieser Anwendungsfall eine besondere Herausforderung dar, denn die meisten in der Chirurgie verwendeten Werkzeuge haben eine verchromte Oberfläche. Aufgrund der reflektierenden Eigenschaften von Chrom – insbesondere starke Spiegelung und Glanzlichter – variieren die Merkmale eines Gegenstands, beurteilt nach seinem Pixel-Abbild, stark; in diesem Zusammenhang werden sie auch als *Features* bezeichnet. Menschen sind ziemlich gut darin, stark reflektierende Objekte zu interpretieren, aber es gibt viele Beispiele, bei denen selbst wir uns täuschen lassen.

Anzahl der Objekte und Klassen

Unser neuronales Netz wird in Code übersetzt, der auf einem stark ressourcenbeschränkten Gerät kompiliert und ausgeführt wird. Einer der limitierenden Faktoren ist das verfügbare RAM, das die Anzahl der Parameter direkt einschränkt. Zusätzlich muss die Bildgröße vom Kamerasensor auf nur 96 × 96 Pixel beschränkt werden, und es gibt eine Begrenzung für die Anzahl der erkennbaren Klassen. Außerdem ist die Anzahl der erkennbaren Objekte in einem Bildrahmen auf 10 festgelegt.

Es gibt Spielraum, mit erweiterten Parametern zu experimentieren, aber es ist besser, diese Begrenzungen zu akzeptieren und kreativ zu denken. Das Ziel des Geräts ist nämlich nicht, bestimmte Gegenstände oder Typen zu identifizieren, sondern das OP-Team zu alarmieren, wenn die Anzahl nicht stimmt. Mit diesem Ansatz können Gegenstände mit ähnlichen Formen und Oberflächen gruppiert werden. Abgesehen davon wird die RAM-Größe selbst bei kleinsten Geräten in naher Zukunft sicherlich steigen. Die Anzahl der für das Training verwendeten Bilder beeinflusst den Speicherbedarf nicht.

Manuelle Datenerfassung und -kennzeichnung

Mit der Kamera des Arduino Nicla Vision wurden zunächst etwa 600 Bilder aufgenommen und in Edge Impulse gekennzeichnet (beschriftet, d. h. mit *Labels* versehen). Die meisten Bilder enthielten mehrere Gegenstände, und ein Teil waren nicht gekennzeichnete Bilder von anderen Hintergrundobjekten. Das mit diesen Daten trainierte Modell erwies sich schnell als unbrauchbar für die Erkennung reflektierender Gegenstände, diente aber als Ausgangsbasis.

Um die Chromoberflächen als Problem zu isolieren, wurden mehrere Chrominstrumente matt lackiert (**Bild 1**) und einige Plastik- sowie Stoff-

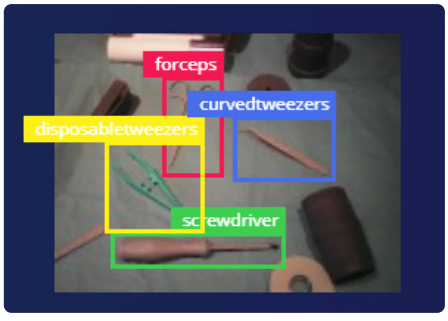


Bild 2. Aufnahme matter Objekte.

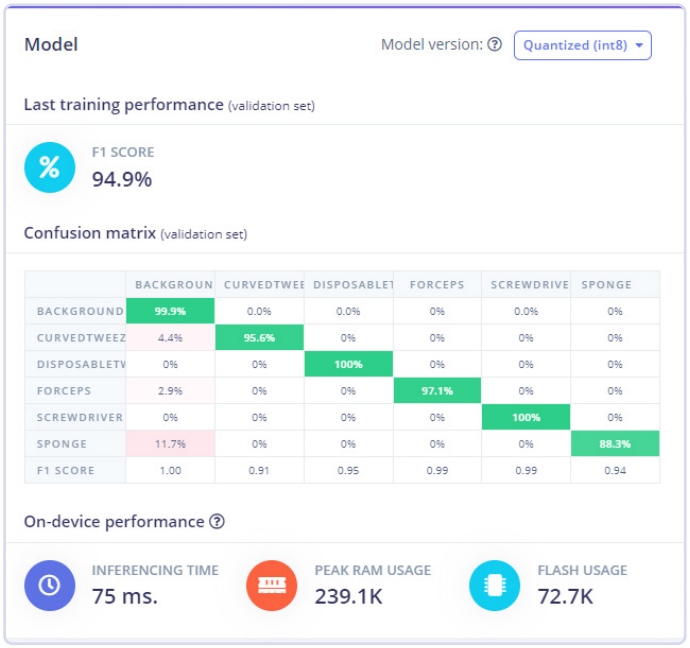


Bild 3. Leistung bei matten Objekten.

gegenstände genutzt, um einen neuen, manuell aufgenommenen und gekennzeichneten Datensatz gleicher Größe zu erstellen. Bei jedem Bild wurden die Gegenstände verstreut und die Kameraposition verändert (**Bild 2**). Dieses Modell funktionierte hervorragend (**Bild 3**) und kann unter [10] eingesehen werden. Ein Video mit Live-Inferenz von der Gerätekamera ist unter [11] zu sehen. Nur trainierte Objekte werden gekennzeichnet. Flackern kann durch Mittelwertbildung reduziert werden.

Kompensation von Reflexionen

Obwohl die oben genannten Lösungen bei den ersten Tests erfolgreich waren, stellten die Reflexionen der Chrominstrumente weiterhin eine Hürde dar, ebenso wie der Umfang der benötigten Trainingsdaten. Schauen wir uns an, wie das Problem angegangen wurde und wie eine Lösung aussieht.

Synthetische Trainingsdaten

Ein zentraler Bestandteil jeder ML-Lösung sind die Daten, mit denen das Modell trainiert, getestet und validiert wird. Bei einem Modell zur visuellen Objekterkennung sind dies viele Bilder der zu erkennenden Objekte. Außerdem muss jedes Objekt auf jedem Bild gekennzeichnet werden. Edge Impulse bietet ein intuitives Tool, um Boxen um die betreffenden Objekte zu zeichnen und Labels zu vergeben. Bei großen Datensätzen kann die manuelle Kennzeichnung sehr aufwendig werden; zum Glück gibt es in Edge Impulse ein automatisches Labeling-Tool. Andere Tools zur Datenverwaltung verfolgen unterschiedliche Ansätze, etwa die Nutzung großer, fertiger Bilddatensätze. Oft sind diese Datensätze aber zu allgemein und ungeeignet für spezifische Anwendungsfälle.

NVIDIA Omniverse Replicator

Ein zentrales Ziel dieses Projekts ist die Erforschung der Erstellung synthetischer Objektbilder, die direkt mit Labels versehen sind. Dies geschieht durch die Erstellung einer 3D-Szene in NVIDIA Omniverse und die Nutzung des Replicator Synthetic Data Generation Toolkits, um Tausende von leicht unterschiedlichen Bildern zu erzeugen – ein Konzept, das als Domain Randomization bekannt ist. Mit einer Grafikkarte vom Typ NVIDIA RTX 3090 lassen sich etwa zwei Raytracing-Bilder pro Sekunde erzeugen. Die Erstellung von 10.000 Bildern dauert also rund fünf Stunden.

Lösungsüberblick

Wir gehen die folgenden Schritte durch, um ein Objekterkennungsmodell auf einem Mikrocontroller-Entwicklungsboard zu erstellen und auszuführen. Eine aktuelle Python-Umgebung mit Visual Studio Code wird empfohlen. Ein 3D-Geometrie-Editor wie Blender ist erforderlich, falls keine 3D-Modelle im USD-Format (Universal Scene Description) vorliegen.

- Installation von Omniverse und Replicator und Einrichten des Debuggings mit Visual Studio Code
- Erstellen einer 3D-Szene in Omniverse
- Arbeiten mit 3D-Modellen in Blender
- Import von 3D-Modellen in Omniverse, Beibehalten von Transformationen, Zuweisung von Materialien
- Festlegen von Metadaten für Objekte
- Erstellung eines Skripts für Domain Randomization
- Erstellung einer Label-Datei für Edge Impulse Studio
- Erstellung eines Objekterkennungsprojekts in Edge Impulse Studio und Hochladen des Datensatzes
- Training und Deployment des Modells auf das Gerät
- 3D-Druck eines Schutzgehäuses für das Gerät
- Verwendung des Objekterkennungsmodells in einer Anwendung

Installation von Omniverse und Replicator und Debugging mit Visual Studio Code

Für NVIDIA Omniverse:

- Installieren Sie Omniverse Isaac Sim von NVIDIA [12].
- Starten Sie Omniverse Isaac Sim.
- Gehen Sie zu *Window Extensions* und installieren Sie *Replicator*.
- Installieren Sie *Embedded VS Code for NVIDIA Omniverse* [13].

Erstellen einer 3D-Szene in Omniverse

- Erstellen Sie eine neue Bühne/Szene (USD-Datei), wie in **Bild 4** gezeigt.
- Erstellen Sie eine texturierte Ebene als Bereich zum Verteilen der Objekte.
- Erstellen Sie eine größere texturierte Ebene als Hintergrund.
- Fügen Sie einige Lichter hinzu.

Wenn Sie eine leistungsstarke, wärmeerzeugende Grafikkarte verwenden, sollten Sie eventuell in den Viewports von Code die Bildwiederholrate stärker begrenzen. Standardmäßig liegt die Grenze bei 120 FPS, was bei höchster Renderqualität viel Wärme erzeugt. Setzen Sie *UI FPS Limit* und *Present thread FPS Limit* auf 60. Leider wird diese Einstellung nicht Session-übergreifend gespeichert und muss jedes Mal wiederholt werden, wenn Projekte geöffnet werden.

Arbeiten mit 3D-Modellen in Blender

Die zu erkennenden Objekte müssen als 3D-Modell und mit einer Oberfläche (Material) dargestellt werden. Omniverse bietet eine Bibliothek

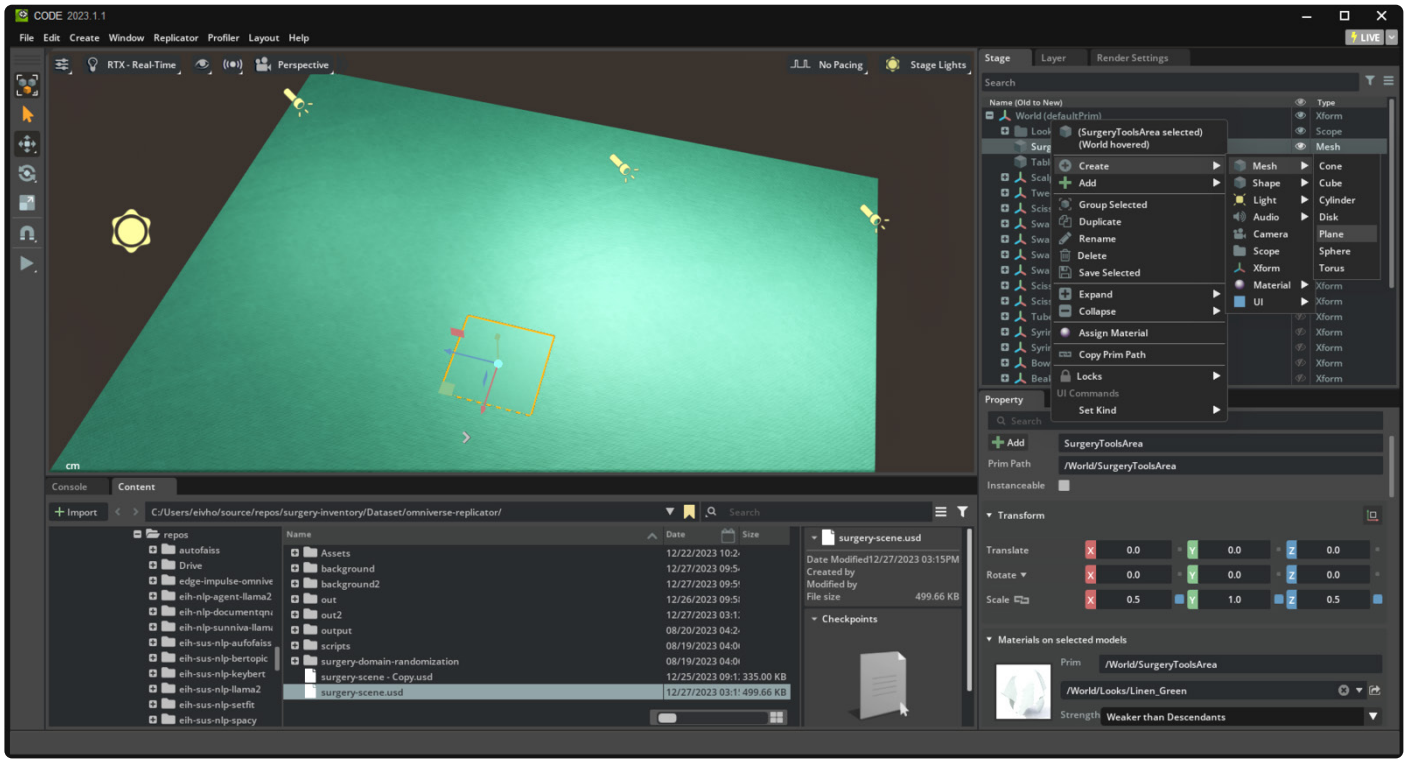


Bild 4. Erstellen der Bühne.

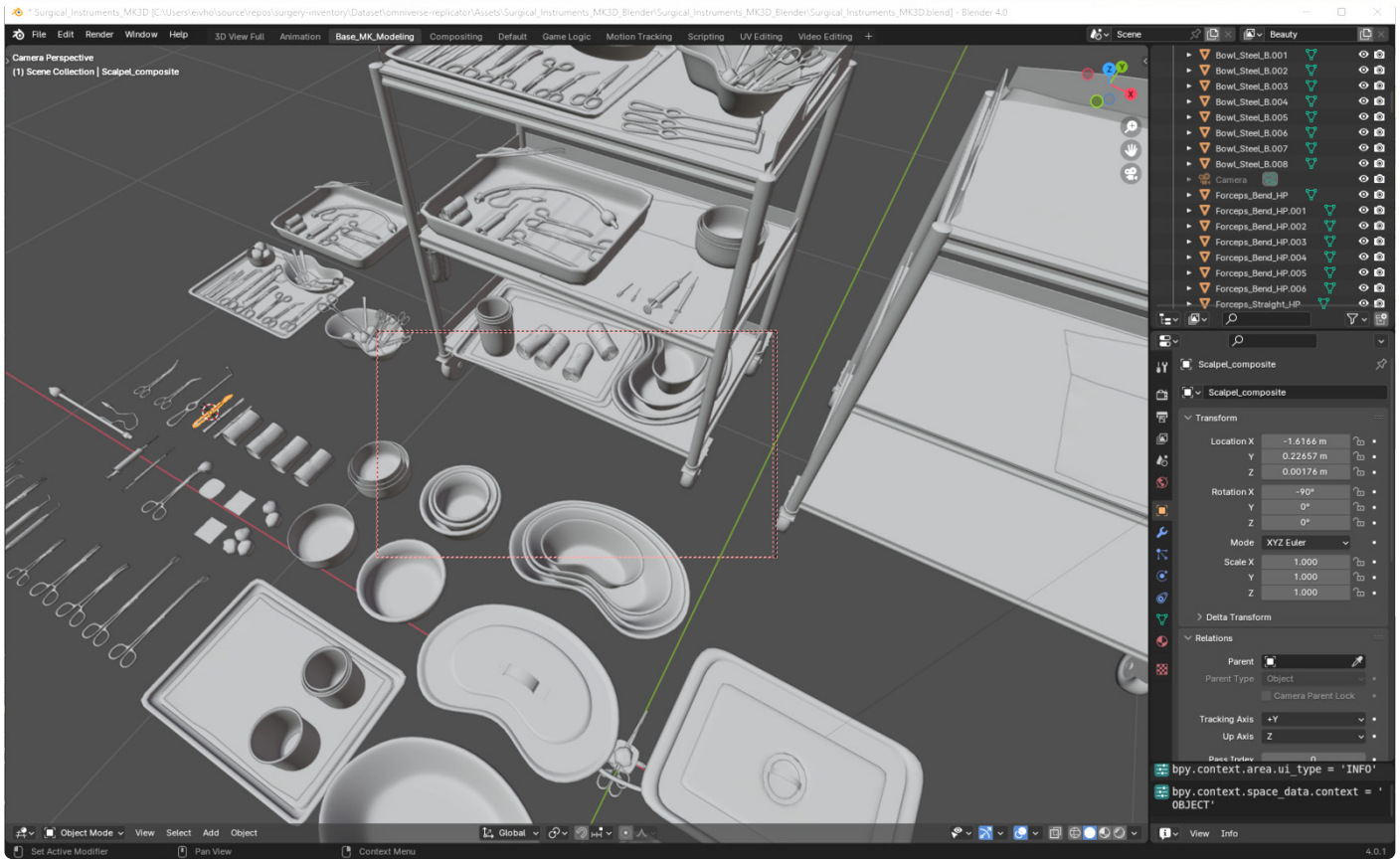


Bild 5. Export eines Modells aus Blender.

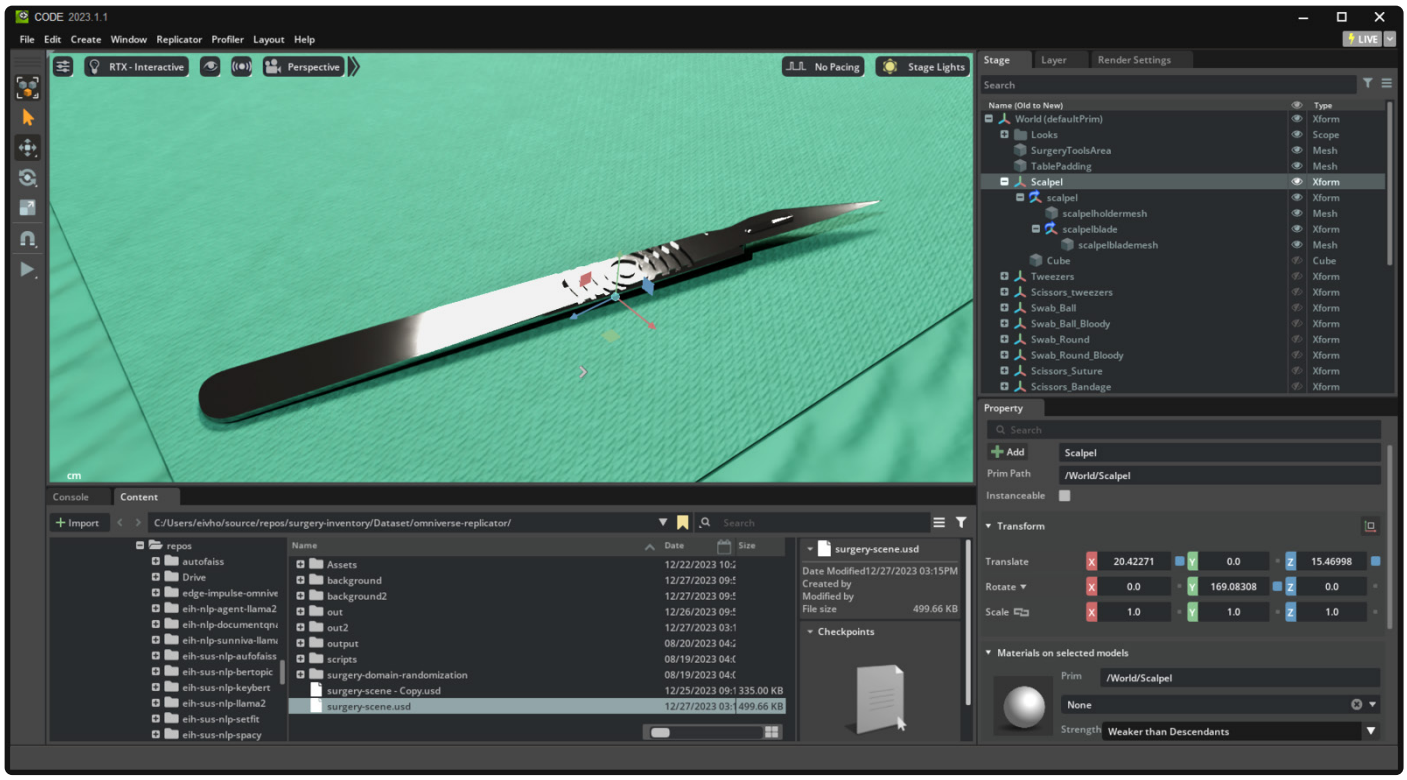


Bild 6. Import eines 3D-Modells in Omniverse.

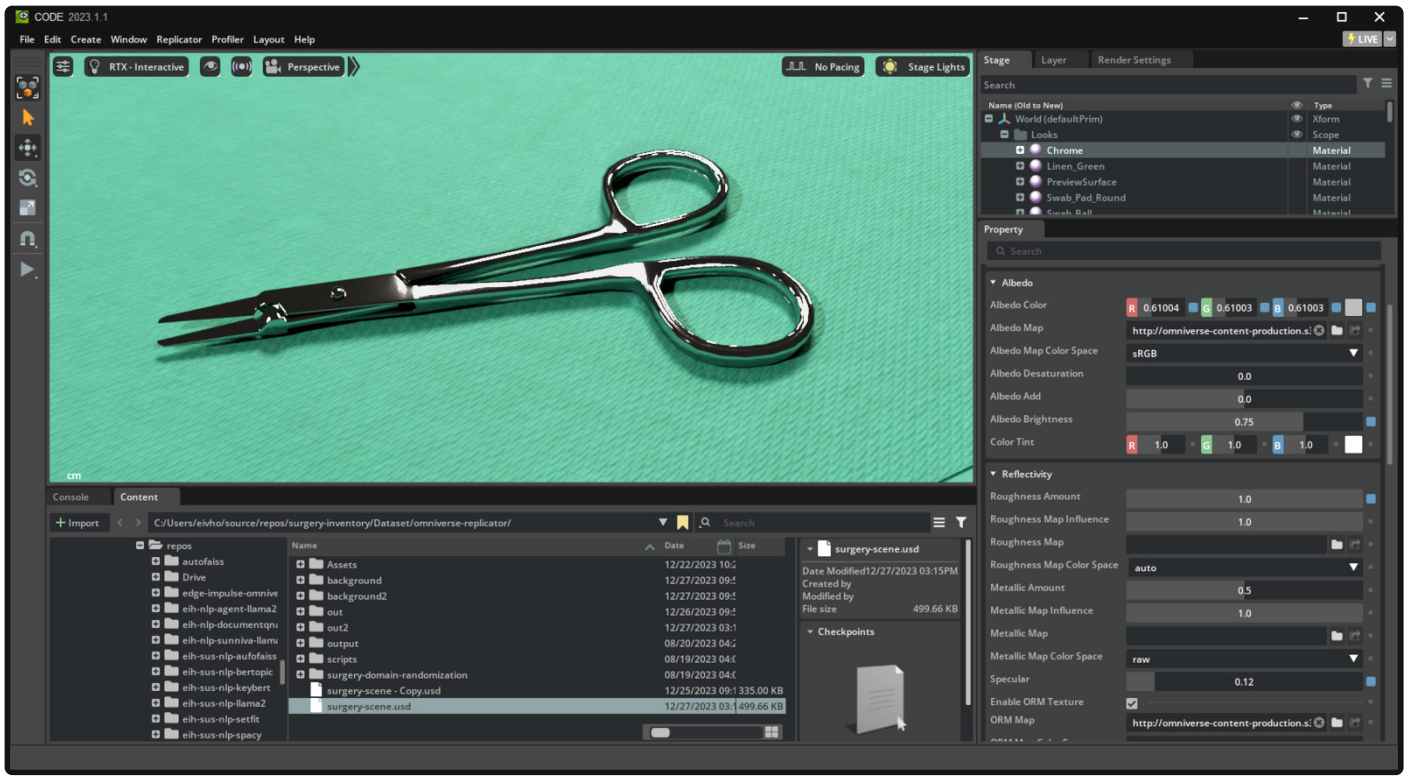


Bild 7. Chrom-Material.

mit fertigen Assets; weitere Modelle können mit Editoren wie Blender (**Bild 5**) erstellt oder auf Seiten wie Turbo Squid [14] gekauft werden.

Eine Szene mit mehreren geometrischen Modellen sollte jeweils als Einzelmodell im USD-Format exportiert werden. Omniverse hat mittlerweile experimentelle Unterstützung für den Import von BSDF-Parametern für Materialien, aber dies ist noch nicht ausgereift. In diesem Projekt wurden Materialien oder Texturen nicht direkt importiert.

Import von 3D-Modellen in Omniverse

Um zu vermeiden, dass benutzerdefinierte Skalierungen oder andere Transformationen beim Export überschrieben werden, empfiehlt es sich, jedem Modellbaum einen Top-Knoten vom Typ *Xform* hinzuzufügen. So kann das Objekt später verschoben werden, ohne Anpassungen zu verlieren (**Bild 6**).

Das Replicator-Toolkit hat eine Funktion zum Verteilen von Objekten auf einer Fläche in seiner API. Um Objektüberschneidungen (weitgehend) zu vermeiden, können einige Verbesserungen vorgenommen werden. Im Screenshot wurde eine Grundform als Begrenzungsbox hinzugefügt, um Abstand zwischen den Objekten zu gewährleisten und dünne Objekte beim Verteilen besser zu berücksichtigen. Die Begrenzungsbox kann unsichtbar gemacht werden. In Replicator 1.9.8 scheint etwas Überlappung unvermeidlich zu sein.

Für die Chromoberflächen wurde ein Material aus einem der in Omniverse bereitgestellten Modelle wiederverwendet (**Bild 7**). Für repräsentative Raytracing-Ergebnisse wechseln Sie in den Modus *RTX - Interactive*; *RTX - Real-Time* ist eine vereinfachte Renderpipeline.

Bild 9. Generierung synthetischer Bilder.

Für stoffbasierte Materialien (**Bild 8**) wurden einige Texturen aus den Ursprungsmodellen übernommen. Eine bessere Einrichtung der Shader mit passenden Texturkarten könnte die Ergebnisse weiter verbessern (**Bild 9**).

Setzen von Metadaten für Objekte

Um Bilder für das Training mit Labels zu erzeugen, kann eine Funktion des Replicator-Toolkits im Menü *Replicator Semantics Schema Editor* genutzt werden.

Hier kann für jeden Top-Knoten, der ein Erkennungsobjekt darstellt, ein Schlüssel-Wert-Paar gesetzt werden. Wählt man *class* als *Semantic Type*

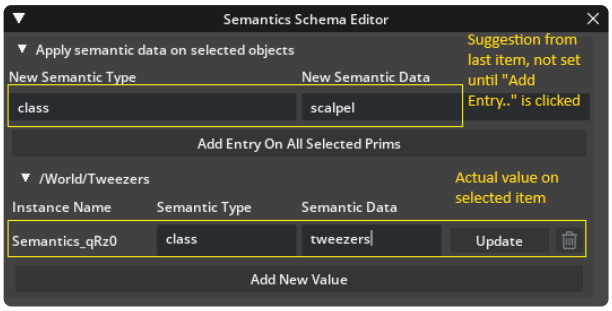


Bild 10. Vorschlag für den Semantics Schema Editor.

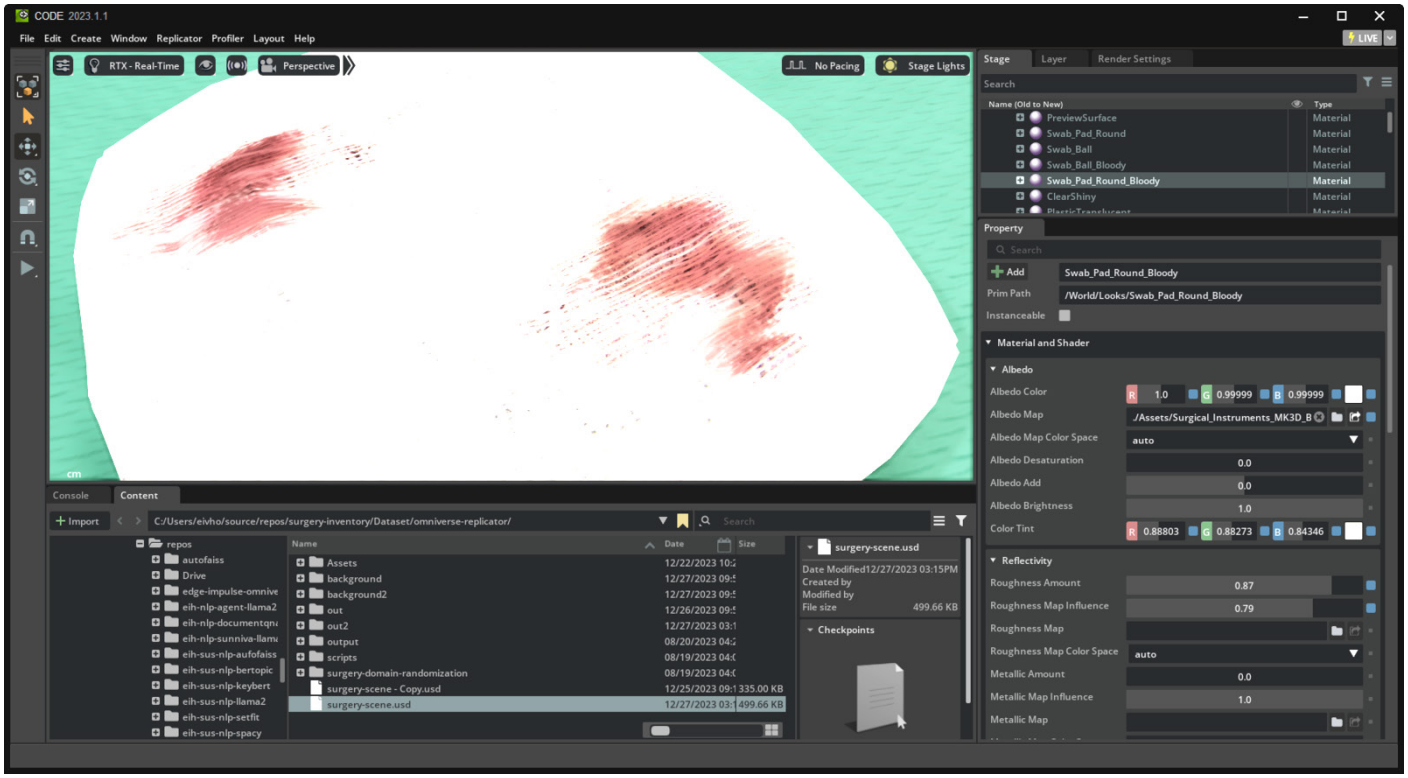


Bild 8. Stoff-Material.

und z. B. *tweezers* (Pinzette) als *Semantic Data*, können diese Strings später als Labels exportiert werden. Die Oberfläche könnte noch intuitiver gestaltet sein, da leicht verwechselt werden kann, welches Feld die tatsächlich gesetzten semantischen Daten anzeigt und welche Felder das Labeling vieler Objekte erleichtern.

Der Semantics Schema Editor kann auch für mehrere ausgewählte Objekte genutzt werden. Er bietet praktische Funktionen, um die Knotennamen für die automatische Kennzeichnung zu verwenden (**Bild 10**).

Erstellen eines Skripts für Domain Randomization

Ein Python-Skript kann geschrieben werden, um die Randomisierung (zufällige Variation) der für das Training generierten Bilder zu automatisieren. Statt mit einer leeren Bühne zu starten und jedes Modell, jedes Licht und jede Kamera programmatisch zu laden, werden die meisten Elemente wie oben beschrieben manuell hinzugefügt. Das Skript – meist mit der Endung .py und in der Nähe der Bühnen-USD-Datei abgelegt – übernimmt die Automatisierung. Das folgende Beispiel *replicator_init.py* [15] demonstriert diesen Ansatz.

Um die im Skript generierten Elemente vom manuell erstellten Inhalt zu trennen, wird zunächst eine neue Ebene in der 3D-Bühne erzeugt:

```
python
with rep.new_layer():
```

Als Nächstes wird festgelegt, dass Raytracing als Bildausgabe genutzt wird. Eine Kamera wird erstellt und die Position fest codiert. Für jede Aufnahme wird sie auf die Objekte ausgerichtet. Dann werden mit den zuvor gesetzten semantischen Daten Referenzen auf Objekte, Hintergrundobjekte und Lichter erzeugt. Schließlich wird das Renderelement festgelegt, indem die Kamera und die gewünschte Auflösung gewählt werden. Die Zielauflösung von 96 × 96 Pixel verursacht offenbar Artefakte, daher wird eine etwas höhere Auflösung von 128 × 128 Pixeln gewählt. Edge Impulse Studio übernimmt später die Skalierung auf die gewünschte Größe.

```
python
rep.settings.
set_render_pathtraced(samples_per_pixel=64)
camera = rep.create.camera(position=(0, 24, 0))
tools = rep.get.prims(semantics=[
    ("class", "tweezers"), ("class", "scissors"),
    ("class", "scalpel"), ("class", "sponge")])
backgrounditems = rep.get.prims(semantics=
   ("class", "background")])
lights = rep.get.light(semantics=[
    ("class", "spotlight")])
render_product = rep.create.render_product(camera,
(128, 128))
```

Aufgrund der asynchronen Natur von Replicator muss die Randomisierungslogik als Callback-Methoden definiert und wie folgt registriert werden:

```
python
    rep.randomizer.register(scatter_items)
    rep.randomizer.register(randomize_camera)
    rep.randomizer.register(alternate_lights)
```

Bevor die eigentliche Randomisierung beginnt, wird definiert, was während jedes Renderings geschieht:

```
python
with rep.trigger.on_frame(num_frames=10000,
rt_subframes=20):
rep.randomizer.scatter_items(tools)
rep.randomizer.randomize_camera()
rep.randomizer.alternate_lights()
```

Mit `num_frames` wird festgelegt, wie viele Renderings erstellt werden sollen. `rt_subframes` lässt die Renderpipeline mehrere Frames durchlaufen, bevor das Ergebnis gespeichert und weitergegeben wird. Ein hoher Wert ermöglicht fortgeschrittene Raytracing-Effekte wie Reflexionen, erhöht aber die Renderzeit. Jede Randomisierungsmethode wird mit optionalen Parametern aufgerufen.

Zum Schreiben jedes Bilds und der semantischen Informationen auf die Festplatte wird eine bereitgestellte API genutzt. Anpassungen am Writer sind mit Replicator 1.9.8 unter Windows noch fehleranfällig. *BasicWriter* wird verwendet, ein separates Skript erstellt später ein Label-Format, das mit Edge Impulse kompatibel ist.

```
python
writer = rep.WriterRegistry.get("BasicWriter")
writer.initialize( output_dir="out",
    rgb=True,
    bounding_box_2d_tight=True)
writer.attach([render_product])
asyncio.ensure_future(rep.orchestrator.step_async())
```

Mit `rgb` wird festgelegt, dass die Bilder als PNG-Dateien geschrieben werden. `bounding_box_2d_tight` sorgt dafür, dass Dateien mit Labels (aus den zuvor definierten Semantiken) und Begrenzungsrahmen (**Bild 11**) als Rechtecke geschrieben werden. Das Skript endet mit einer Einzeliteration in Omniverse, um die Ergebnisse zu visualisieren.

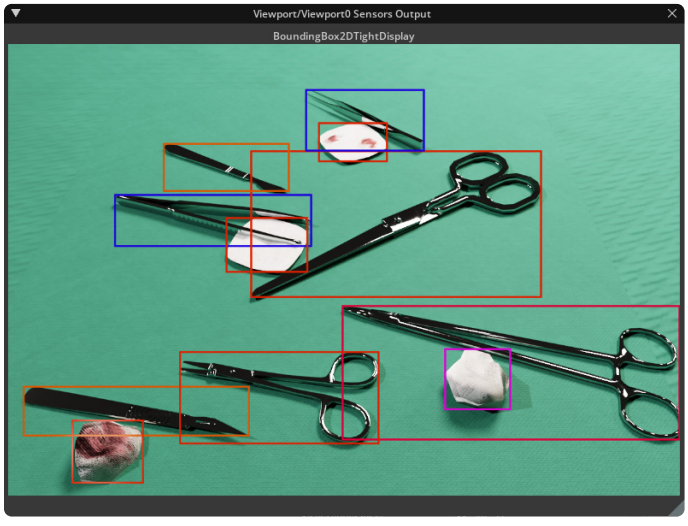


Bild 11. Begrenzungsrahmen.

Listing 1. Python-Skript.

```
python
def scatter_items(items):
      table = rep.get.prims(path_pattern='/World/SurgeryToolsArea')
      with items as item:
            carb.log_info("Tool: " + tool)
            logger.info("Tool: " + tool)
            rep.modify.pose(rotation=rep.distribution.uniform((0, 0, 0), (0, 360, 0)))
            rep.randomizer.scatter_2d(surface_prims=table, check_for_collisions=True)
      return items.node
def randomize_camera():
      with camera:
            rep.modify.pose(
            position=rep.distribution.uniform((-10, 50, 50), (10, 120, 90)),
            look_at=(0, 0, 0))
      return camera
def alternate_lights():
      with lights:
           rep.modify.attribute("intensity", rep.distribution.uniform(10000, 90000))
      return lights.node
```

Die Bounding Boxes können angezeigt werden, indem das Sensor-Widget aktiviert, *BoundingBox2DTight* markiert und *Show Window* ausgewählt wird.

Das Einzige, was noch fehlt, ist die Definition der Randomisierungslogik (siehe **Listing 1**).

Für `scatter_items` wird auf den Bereich zugegriffen, der die Gegenstände enthält. Für jedes Objekt wird eine Zufallsrotation (0 bis 360 Grad auf der Ebene) erzeugt, und mit `scatter_2d` die Platzierung zufällig verteilt. `surface_prims` nimmt ein Array möglicher Flächen, `check_for_collisions` versucht Überlappungen zu vermeiden. Die Reihenfolge der Schritte ist wichtig, um Überschneidungen zu verhindern.

Für die Kamera wird einfach die Position auf allen drei Achsen zufällig gewählt und sichergestellt, dass sie auf die Bühnenmitte zeigt.

Die Lichter werden in ihrer Helligkeit zufällig zwischen bestimmten Werten variiert. Im bereitgestellten Beispiel werden für die Objekterkennung und für Hintergrundobjekte separate Bildsätze erstellt. Der Prozess läuft einmal für die OP-Gegenstände, dann wird die folgende Zeile geändert von

```
python
rep.randomizer.scatter_items(tools)
```

zu

```
python
rep.randomizer.scatter_items(backgrounditems)
```

Beim Rendern der relevanten Gegenstände müssen die Hintergrundobjekte manuell oder programmatisch ausgeblendet werden und umgekehrt. Auch der Ausgabeordner sollte geändert werden, um Überschreiben zu vermeiden.

Ob es zum Erzeugen guter Trainingsdaten besser ist, relevante Objekte und Hintergrundobjekte auf getrennten Bildern oder zusammen erscheinen zu lassen, ist umstritten. In diesem Projekt wurden mit beiden Ansätzen gute Ergebnisse erzielt.

Erstellung der Label-Datei

Edge Impulse Studio unterstützt eine Vielzahl von Bildlabel-Formaten für die Objekterkennung. Leider muss das Output-Format des BasicWriter von Replicator konvertiert werden, damit es über die Weboberfläche oder per Ingestion-API [16] hochgeladen werden kann.

Dafür gibt es ein einfaches Python-Programm, *basic_writer_to_pascal_voc.py* [17]. Mit einer kurzen Beschreibung für ChatGPT wurde die Konvertierung von Replicator-Ausgabe in das von Edge Impulse [18] benötigte Format beschrieben. In der Shell rufen Sie das Programm wie folgt auf:

```
python basic_writer_to_pascal_voc.py
```

Oder nutzen Sie den Debugging-Modus von Visual Studio Code, indem Sie in *launch.json* das Eingabeverzeichnis wie folgt setzen:

```
"args": ["../out"]
```

Dadurch wird eine Datei *bounding_boxes.labels* erzeugt, die alle Labels und Bounding Boxes je Bild enthält.

Erstellung eines Objekterkennungsprojekts in Edge Impulse Studio und Hochladen des Datensatzes

Sehen Sie sich das bereitgestellte Objekterkennungsprojekt von Edge Impulse [19] an oder folgen Sie einem Leitfaden zur Erstellung eines neuen Projekts [9]. Um Objekte mit reflektierenden Oberflächen zu erkennen, sind viele Trainingsbilder nötig – die exakte Anzahl hängt von vielen Faktoren ab, Experimentieren ist ratsam. Es empfiehlt sich, zunächst relativ klein zu starten, z. B. mit 1000 Bildern der zu erkennenden

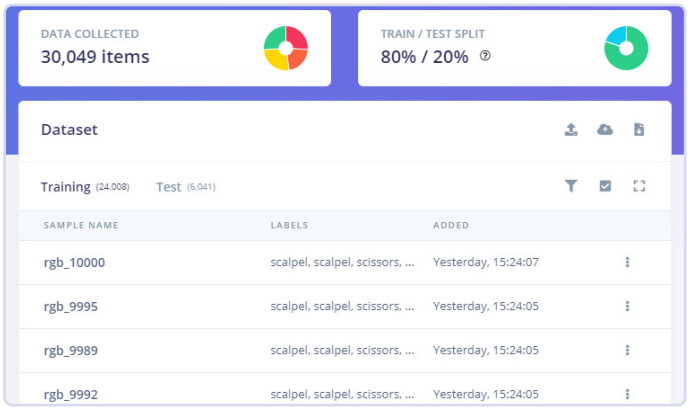


Bild 12. Datenerfassung in Edge Impulse Studio.

Objekte. Für dieses Projekt wurden mehr als 30.000 Bilder generiert (**Bild 12**), viel mehr als nötig. Um praxistaugliche Ergebnisse zu erzielen, sind auch viele Bilder von zufälligen Hintergrundobjekten nötig. Für dieses Projekt wurden dafür andere OP-Geräte verwendet, diese müssen nicht einzeln gelabelt werden. Dennoch erzeugt Edge Impulse Studio eine Labeling-Queue für jedes Bild, das keine Labels enthält. Damit nicht jedes Bild durchgeklickt werden muss, um zu bestätigen, dass es keine Labels enthält, erstellt das beschriebene Programm eine Datei namens *bounding_boxes.labels* mit leeren Labels für Objekte mit der semantischen Klasse *background*. Das beste Verhältnis zwischen zu erkennenden Objekten und Hintergrundbildern muss experimentell ermittelt werden, aber ein Hintergrundanteil von 1 % bis 2 % ist ein guter Ausgangswert.

Edge Impulse erstellt pro Bild eine eindeutige Kennung, sodass mehrere Iterationen möglich sind, auch mit denselben Dateinamen. Laden Sie einfach alle Bilder eines Batches mit der Datei *bounding_boxes.labels* hoch. So lassen sich Tausende gelabelte Bilder einfach erzeugen, und man kann beobachten, wie die Performance bei der Erkennung reflektierender Objekte steigt. Achten Sie darauf, die Anzahl der Labels für jede Klasse möglichst anzugleichen.

Edge Impulse hat eine praktische GUI-basierte Erweiterung [20] entwickelt, um Bilder direkt von Replicator ins Projekt hochzuladen. Derzeit werden dabei nur Bilder hochgeladen, in Zukunft könnten auch Labels unterstützt werden.

Training und Deployment des Modells auf das Gerät

Abschließend kann das Objekterkennungsmodell [21] erstellt und trainiert werden. Das Zielgerät muss gesetzt werden, und beim Arduino Nicla Vision ist zu beachten, dass das RAM nur für 96 × 96 Pixel ausreicht. Ein „Early Stop"-Feature wäre wünschenswert, aktuell bleibt nur Experimentieren mit der Anzahl der Trainingszyklen. Data Augmentation sollte bei Tausenden Bildern vermieden werden – es verbessert die Ergebnisse nicht (**Bild 13**).

3D-Druck eines Schutzgehäuses für das Gerät

Um das Gerät zu schützen und das Tragen zu erleichtern, wurde ein Gehäuse in CAD entworfen und 3D-gedruckt. Es empfiehlt sich, zu-

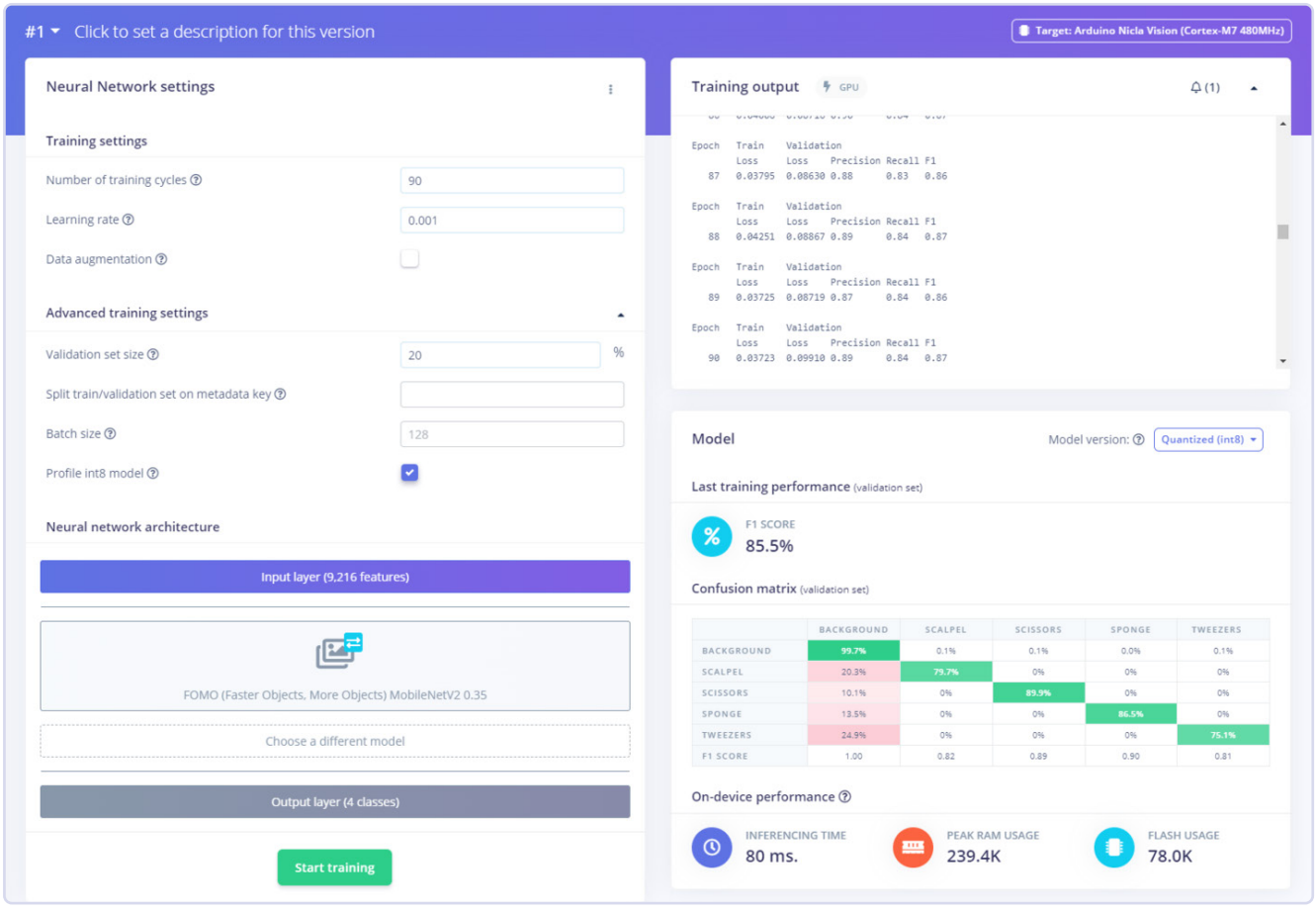


Bild 13. Die Leistung des Modells.

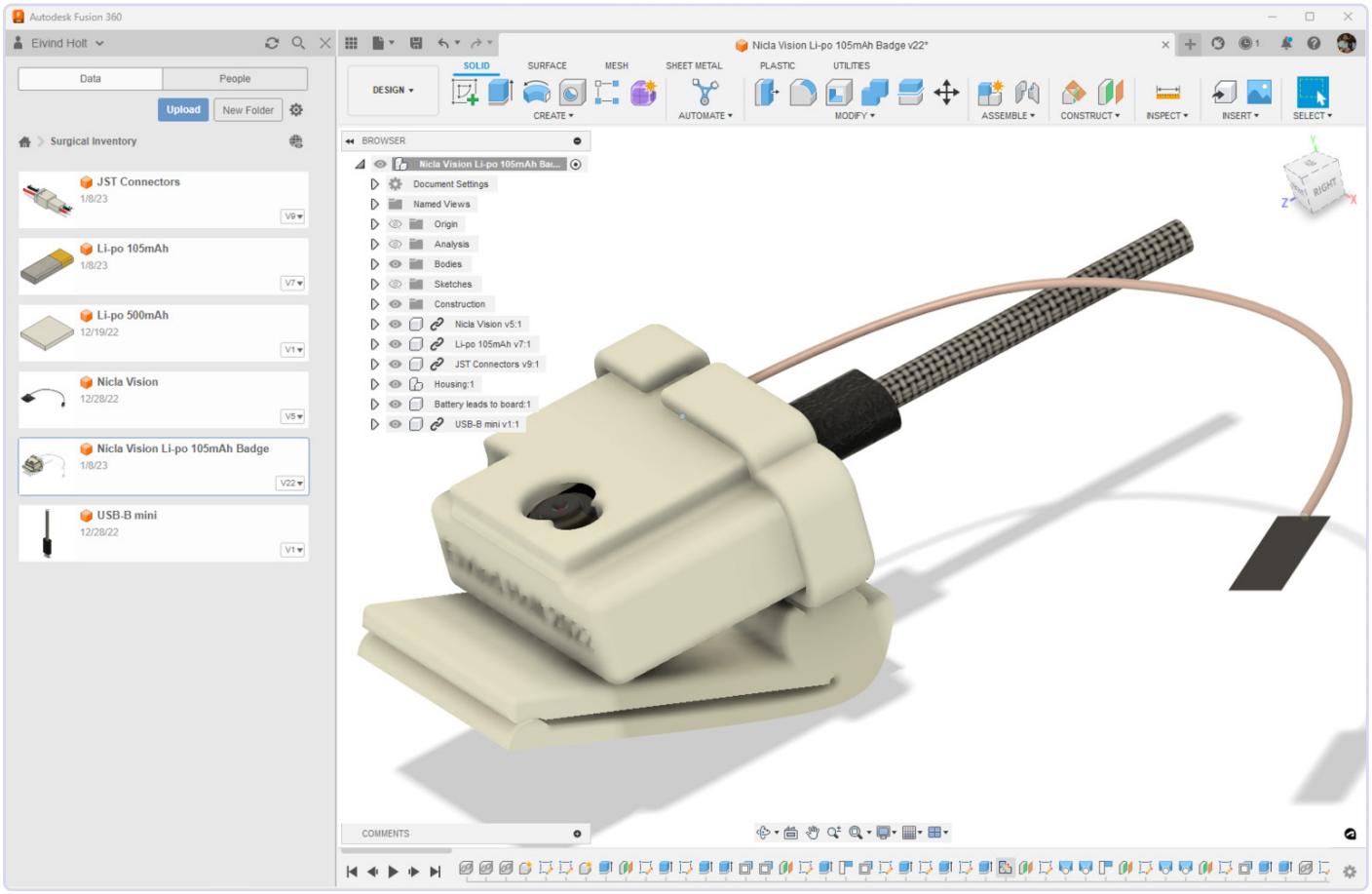


Bild 14. Das Gehäuse im CAD-Programm.

nächst einfache 3D-Modelle aller Komponenten zu erstellen (**Bild 14, Bild 15** und **Bild 16**). Dies reduziert Iterationen und Überraschungen bei der Montage deutlich.

Verwendung eines Objekterkennungsmodells in einer Anwendung

Ein trainiertes Modell kann als Arduino-kompatible Bibliothek kompiliert werden. Ereignisse können die Inferenz auslösen, und der Arduino Nicla Vision kann die Anzahl erkannter Objekte per Bluetooth LE übertragen. Ein BLE-Dongle oder Smartphone kann diese Ereignisse empfangen und an eine Web-API zur Integration mit anderen Systemen weiterleiten. So kann z. B. das medizinische Informationssystem die erkannten Gegenstände protokollieren. Der E-Health-Standard HL7 FHIR [22] ermöglicht die Definition von Geräten und Materialien für Behandlungen. Testumgebungen wie Open DIPS [23] eignen sich gut, um Integrationen mit Krankenhaussystemen auszuprobieren.

Ergebnisse

Die Ergebnisse dieses Projekts zeigen, dass Trainings- und Testdaten für Objekterkennungsmodelle mit 3D-Modellen synthetisiert werden können, wodurch der manuelle Aufwand für die Bilderstellung und Kennzeichnung verringert wird (**Bild 17**). Besonders beeindruckend ist die Detektion unvorhersehbar reflektierender Oberflächen auf stark ressourcenbeschränkter Hardware durch eine große Anzahl von Bildern. Eine Demo zur Detektion von Chrominstrumenten auf einem Arduino Nicla Vision ist auf YouTube [24] verfügbar.

Das GitHub-Repository dieses Projekts finden Sie unter [25]. Weitere Lektüre: *How to Train an Object Detection Model for Visual Inspection with Synthetic Data* [26].

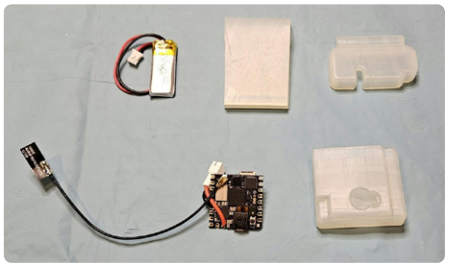


Bild 15. Das Geräteschutzgehäuse.

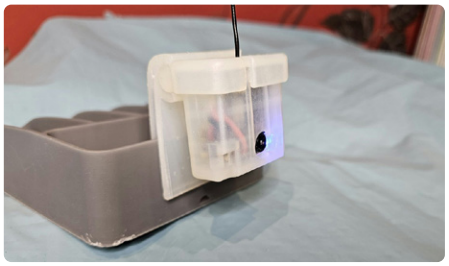


Bild 16. Das montierte Geräteschutzgehäuse.

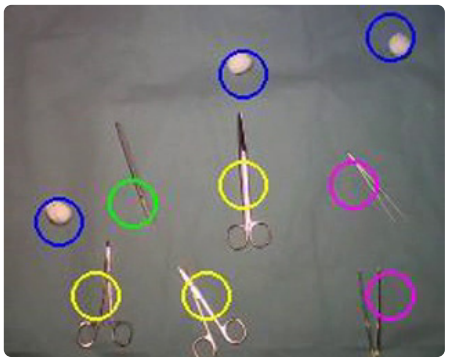


Bild 17. Erkennung von Chrom-Objekten.

Die nächste Generation der Objekterkennung

Das Gebiet der visuellen Objekterkennung erlebt aktuell eine spannende Entwicklungsphase, ermöglicht durch zahlreiche wichtige Fortschritte. Man stelle sich einen Dienst vor, der 3D-Modelle als Eingabe akzeptiert und daraus eine vielfältige Bildersammlung für das Training generiert. Durch stetige Verbesserungen bei generativen Diffusionsmodellen – insbesondere bei der Text-zu-3D-Konvertierung – stehen wir kurz davor, noch leistungsfähigere Möglichkeiten zur Erstellung synthetischer Trainingsdaten zu erschließen. Dieser Fortschritt ist nicht nur ein technologischer Sprung, sondern wird die Herangehensweise an die Objekterkennung revolutionieren und den Weg für eine neue Generation hochinnovativer und effektiver Objekterkennungslösungen ebnen. Die Auswirkungen sind groß: Sie ermöglichen neue Genauigkeit und Effizienz in verschiedensten Anwendungen.

250685-02

Über den Autor

Eivind Holt arbeitet seit 22 Jahren an der Entwicklung des elektronischen Krankenakten-Systems DIPS [27] in Norwegen. Als Tech Lead für Forschung und Innovation treibt er durch den Einsatz von KI, Sensortechnik und anderen Zukunftstechnologien eine effizientere Gesundheitsversorgung voran. In seiner Freizeit entwickelt und teilt er neuartige, sensorbasierte Projekte, die Machine Learning und energieeffiziente Kommunikation nutzen.

Fragen oder Kommentare?

Haben Sie technische Fragen oder Rückmeldungen zu diesem Artikel? Kontaktieren Sie den Autor unter https://linkedin.com/in/eivholt oder die Elektor-Redaktion unter redaktion@elektor.de.

WEBLINKS

[1] „Retained Surgical Foreign Bodies after Surgery“, PubMed Central: https://pmc.ncbi.nlm.nih.gov/articles/PMC5320916/
[2] Arduino Nicla Vision, Edge-Impulse-hardware-Seite: https://docs.edgeimpulse.com/hardware/boards/arduino-nicla-vision
[3] Edge Impulse Studio: https://studio.edgeimpulse.com/login?next=%2Fstudio
[4] NVIDIA Omniverse: https://www.nvidia.com/en-us/omniverse/
[5] Replicator, NVIDIA: https://docs.omniverse.nvidia.com/extensions/latest/ext_replicator.html
[6] Visual Studio Code: https://code.visualstudio.com
[7] Blender: https://www.blender.org
[8] Autodesk Fusion 360: https://www.autodesk.com/no/products/fusion-360/overview
[9] Edge Impulse FOMO (Faster Objects, More Objects): https://docs.edgeimpulse.com/studio/projects/learning-blocks/blocks/object-detection/fomo
[10] Surgery Inventory, Edge Impulse project: https://studio.edgeimpulse.com/public/94601/latest
[11] „Surgery Inventory TinyML Edge Impulse matte objects demo“, YouTube: https://www.youtube.com/watch?v=8B8JAnl4Aq8
[12] „Getting started with Omniverse Replicator“, NVIDIA: https://docs.omniverse.nvidia.com/extensions/latest/ext_replicator/getting_started.html#getting-started-with-omniverse-replicator
[13] Embedded VS Code für NVIDIA Omniverse, GitHub: https://github.com/Toni-SM/semu.misc.vscode
[14] „3D Models for Every Budget“, Turbosquid: https://www.turbosquid.com
[15] replicator_init.py, GitHub: https://github.com/eivholt/surgery-inventory-synthetic-data/blob/main/omniverse-replicator/replicator_init.py
[16] Ingestion API: https://docs.edgeimpulse.com/apis/ingestion
[17] basic_writer_to_pascal_voc.py, GitHub: https://github.com/eivholt/surgery-inventory-synthetic-data/blob/main/omniverse-replicator/basic_writer_to_pascal_voc.py
[18] Object detection: https://docs.edgeimpulse.com/tools/specifications/data-annotation/object-detection
[19] Surgery Inventory Synthetic, Edge-Impulse-Projekt: https://studio.edgeimpulse.com/public/322153/latest
[20] Edge Impulse Data Ingestion Omniverse Extension, GitHub: https://github.com/edgeimpulse/edge-impulse-omniverse-ext/tree/main
[21] „Design and train our Object Detection Model“, Edge Impulse tutorial: https://docs.edgeimpulse.com/tutorials/end-to-end/object-detection-centroids#3-designing-an-impulse
[22] HL7 FHIR, CI-Build, fhir.org: https://build.fhir.org/device.html
[23] Open DIPS: https://open.dips.no
[24] „Surgery Inventory TinyML Edge Impulse chrome objects demo“, YouTube: https://www.youtube.com/watch?v=1k0pfPwzTw4
[25] GitHub Repo: https://github.com/eivholt/surgery-inventory-synthetic-data
[26] „How to Train an Object Detection Model for Visual Inspection with Synthetic Data“, NVIDIA developer blog: https://developer.nvidia.com/blog/how-to-train-an-object-detection-model-for-visual-inspection-with-synthetic-data/