Von rohen Daten zum geschliffenen KI-Modell

von Oliver Roth

Wofür man Embedded KI auch einsetzen mag, letztlich ist immer das KI-Modell Kernstück der Lösung. Es ist die Instanz, die Muster erkennt, Vorhersagen macht und Entscheidungen trifft, und zwar mithilfe der Daten, die ihr vorab beim Training und später im laufenden Prozess zur Verfügung stehen. Doch welche Daten eignen sich im industriellen Umfeld, und was ist bei Modellentwicklung, Training und modellbasierter Validierung im Feld zu beachten?

Bild: Pattarin/stock.adobe.com

Eines gilt unabhängig davon, ob man KI für vorausschauende Wartung, Prozessoptimierung und Qualitätskontrolle oder andere Zwecke nutzen will: Ohne geeignete Daten geht nichts. Und die sind im industriellen Umfeld oft Mangelware. Denn auch oder gerade wenn Maschinen samt Steuerung seit Jahren zuverlässig arbeiten, werden meist nur wenige Parameter erfasst. Die meisten Signale, die auf Fehler, Verschleiß oder Anomalien hindeuten und für die Entwicklung von KI-Modellen nützlich wären, fehlen dagegen.

Daten und Signale: Woher nehmen?

Hinzu kommt, dass viele Embedded-Systeme nur über ein begrenztes Speicher- und Übertragungsvolumen verfügen. Datenlogging ist selten vorgesehen, und wenn doch, dann nicht in einer für das Training geeigneten Form. Labeling, also die Klassifizierung von Datenpunkten, ist ein weiteres Problem: Oft entscheiden erfahrene Bediener, ob ein bestimmter Betriebszustand als ‚normal' oder ‚kritisch' zu werten ist. Diese Einschätzungen sind jedoch selten schriftlich dokumentiert oder reproduzierbar.

Was aber tun, wenn die Signale fehlen, die das KI-Modell eigentlich bräuchte, also jene zeitabhängigen physikalischen Größen oder digitalen Werte, die von Sensoren, Aktoren oder Systemkomponenten stammen können? In der Praxis hat sich ein schrittweises Vorgehen bewährt: Zuerst wird geprüft, ob vorhandene Daten zumindest als Näherung dienen können. Ist das nicht ausreichend, helfen zusätzliche temporär angebrachte Sensoren, die mit mobilen

Web-Tipp

Dieser Beitrag ist der 2. Teil der Artikelserie „Der Weg zur Embedded KI". Lesen Sie Teil 1 online:
https://bit.ly/3Zq0mEG

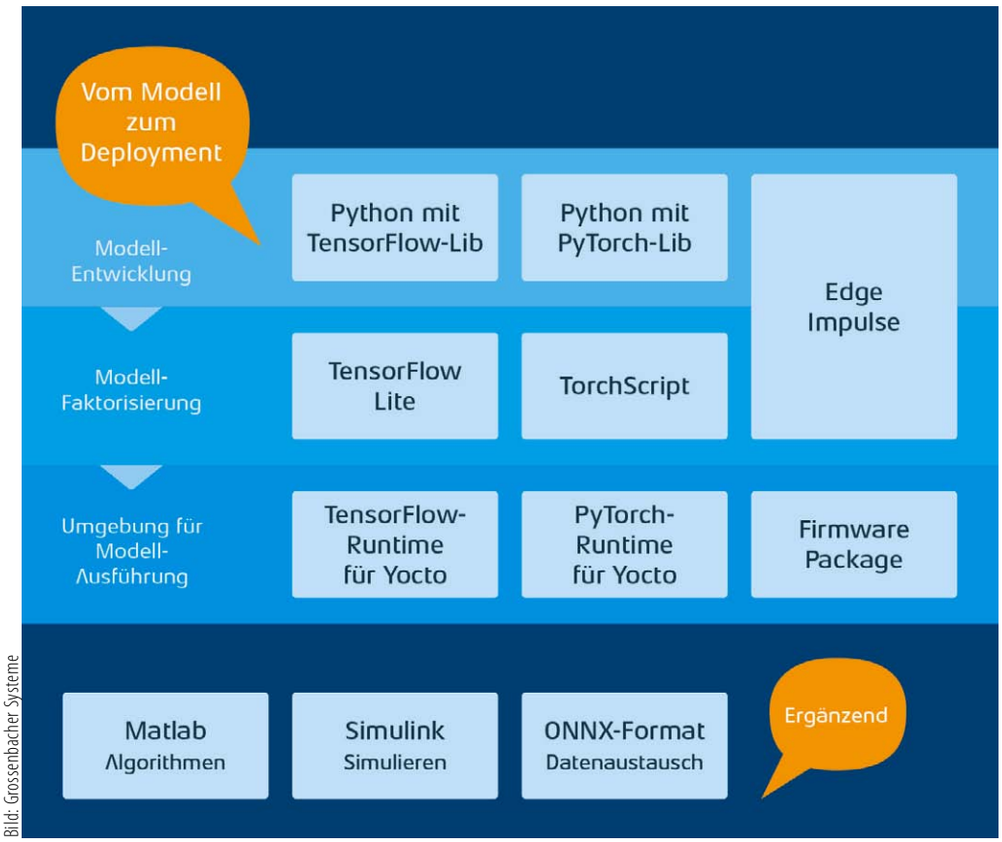

Bild: Grossenbacher Systeme

Eine Auswahl an KI-Software und ihre Anwendungsfelder.

Loggern verbunden werden. Diese Maßnahmen können innerhalb weniger Tage einen Datensatz liefern, der für ein erstes Modelltraining ausreicht.

In manchen Fällen lässt sich auch mit synthetischen Daten arbeiten: Eine gezielte Variation bekannter Muster, ergänzt durch Simulation oder einfache Replikation existierender Daten, kann helfen, die Varianz im Trainingsset zu erhöhen. Wichtig ist dabei Transparenz: Der Ursprung der Daten, real oder synthetisch, sollte dokumentiert werden.

Training und Validierung: Rein ins echte Leben!

Sobald erste Daten vorliegen, stellt sich die nächste Frage: Wo soll trainiert werden? Welche Tools sollen die Daten übernehmen? Viele setzen zunächst auf bewährte Tools wie ‚Python' mit ‚TensorFlow' oder ‚PyTorch'. Das ist naheliegend, denn leistungsfähige Frameworks, eine große Community und viele Tutorials erleichtern den Einstieg. Doch der Weg vom trainierten Modell bis zur tatsächlichen Inferenz, also der Anwendung eines ‚fertigen' Modells auf einem Gerät, ist lang und manchmal voller Hindernisse. Ein häufiges Hindernis ist, dass sich das Modell als zu groß für die Zielhardware erweist, ein anderes, dass Modelle, die im Labor mit hoher Genauigkeit arbeiten, im realen Einsatz deutlich abweichende Ergebnisse liefern. Die Ursachen für die letztgenannte Problematik sind vielfältig: Zu nennen sind beispielsweise ein Rauschen in den Sensorsignalen, nicht vorhergesehene Zustände oder schlicht eine andere Verteilung der Eingabedaten im Feld.

Deshalb empfiehlt es sich, frühzeitig mit realen Systemen zu testen. Der Begriff ‚Nullserie' beschreibt diesen Ansatz sehr gut: Geräte werden mit Logging-Funktion ausgestattet, sammeln aber noch keine produktiven Entscheidungen. In dieser Phase kann das Modell ‚mitlaufen', seine Ausgaben werden protokolliert und später mit den tatsächlichen Ereignissen verglichen.

Auch die Zusammenarbeit mit ausgewählten Versuchskunden hat sich

computer & automation
NEWSLETTER

Link up!

Tool	Sprache	Besonderheit	Code-Footprint
TensorFlow	Python	Große Community, viele Features	hoch
Edge Impulse	C/C++	Webinterface, Zielsystem-fokussiert	niedrig
Matlab mit codegen	Matlab	Automatischer C-Code Export möglich	optimierbar, niedrig

bewährt. Diese Nutzer liefern nicht nur wertvolle Daten, sondern auch praktisches Feedback zur Integration des KI-Modells in die Benutzeroberfläche und in bestehende Arbeitsabläufe.

Vortrainierte Modelle: Es geht auch zügig

Oft ist jedoch der Aufwand für das Training eines komplett neuen Modells ohnehin nicht zu rechtfertigen – etwa wenn es sich um klassische Aufgaben wie Klassifikation oder Anomalie-Erkennung handelt. Hier kommt Transfer Learning ins Spiel: Man nutzt ein vortrainiertes Modell, etwa ein MobileNet oder ein anderes kleineres Convolutional Neural Network (CNN), und trainiert es weiter oder passt es an den eigenen Anwendungsfall an.

Das spart Ressourcen, reduziert den Bedarf an Trainingsdaten und hilft dabei, das Modell gleich in der passenden Architektur aufzusetzen. Wichtig ist dabei, das Zielsystem im Blick zu haben: Ein Modell, das auf dem Laptop gut läuft, kann auf dem Embedded-Gerät zu groß oder zu langsam sein. Um vortrainierte Modelle anzupassen, stehen diverse Tools zur Verfügung (**siehe Tabelle oben**).

Je schwächer die Hardware in Sachen RAM, CPU-Power und Speicherplatz ist, desto wichtiger wird der Code-Footprint und damit der Leistungshunger des KI-Modells. Wer das von Anfang an berücksichtigt, spart sich spätere Enttäuschungen.

Deshalb bietet Grossenbacher Systeme skalierbare Ansätze für die Hardware und verteilt die Modellentwicklung bei Bedarf auf mehrere Submodelle, die containerisiert von verschiedenen Processing Units bearbeitet werden.

Modelltest auf Embedded-Hardware

Der Sprung in die reale Umgebung ist ein kritischer Moment und sollte möglichst frühzeitig erfolgen. Denn erst dann zeigt sich, ob das Modell auch unter Produktionsbedingungen funktioniert. Dabei geht es nicht nur um Genauigkeit; Latenz, Ressourcenverbrauch und Stabilität spielen ebenso eine Rolle. Wichtig ist, dass das Modell auf der Zielhardware läuft und seine Ausgaben protokolliert und mit den erwarteten Werten verglichen werden. Wo weichen die Ergebnisse ab? Welche Sensorwerte führen zu Unsicherheiten? Warum reagiert das Modell auf ein seltenes Ereignis falsch?

Viele Fehlerquellen werden erst dann sichtbar: Das Modell hat versehentlich ein irrelevantes Muster gelernt, etwa, dass eine bestimmte Schwingung immer mit einer Störung einhergeht, obwohl es sich nur um ein temporäres Geräusch handelt. Oder die Quantisierung hat die Gewichte so verändert, dass Entscheidungsgrenzen verrutschen. Solche Effekte lassen sich nur in der realen Anwendung entdecken. Abhilfe schafft man nicht durch neue Trainingsrunden, sondern durch gezieltes Debugging. Nur so entsteht Vertrauen in das Modell und in die Systemarchitektur.

Roll-Out und Updates

Ein funktionierendes Modell ist dennoch kein Endpunkt. Denn jetzt beginnt die Skalierung: Wie bringe ich das Modell auf viele Geräte? Wie sorge ich dafür, dass alle Systeme im Feld die richtige Version verwenden, und wie erkenne ich, wenn sich die Modellgüte verändert?

Als Antwort können nur leistungsfähige Over-the-Air-Lösungen in Betracht kommen, die versionierte, sichere und dokumentierte Updates erlauben. Deshalb setzt Grossenbacher auf ein eigenes Tool für Field Device Management, das Remote-Updates des KI-Modells, des Betriebssystems oder gar einzelner Konfigurationsdateien ermöglicht. Zusätzlich hilft in diesem Fall auch das Abholen von KI-Modell-Diagnose-Daten über das Field Device Management Tool, um Modellüberwachung zu ermöglichen. So kann man jederzeit erkennen, wenn ein Modell zunehmend falsche Entscheidungen trifft – ein Indikator für Modelldrift. Er entsteht, wenn sich die Umgebung des KI-Modells mit der Zeit verändert: Es wird ungenauer oder weniger zuverlässig, weil sich die zugrunde liegenden Datenmuster ändern.

Deshalb kann es sinnvoll sein, Mechanismen zu etablieren, mit denen das KI-Modell während seiner gesamten Einsatzdauer beim Kunden (re-)trainiert und korrigiert werden kann, quasi ein Rezept für ‚lebenslanges Lernen'. Hierzu bieten sich ebenso verschiedene Mechanismen an, zum Beispiel Federated Learning oder Reinforcement Learning.

Fazit: Wer die Wahl hat, braucht Kompetenz

Grundsätzlich ist die Entwicklung von KI-Modellen im industriellen Umfeld keine Geheimwissenschaft (mehr). Zahlreiche vortrainierte Modelle sind unter Open-Source-Lizenzen verfügbar und mithilfe diverser Werkzeuge auf Embedded-Hardware nutzbar. Doch der Teufel steckt im Detail, also der situations- und bedarfsgerechten Auswahl der geeigneten Modelle, Strategien, Tools und Prozessoren, die sich teilweise gegenseitig beeinflussen und bedingen. Deshalb ist eines sicher: Die Bedeutung kompetenter Entwicklungspartner für Embedded-KI-Projekte nimmt nicht ab. ag

Oliver Roth
ist CEO von Grossenbacher Systeme und der Amalthea Gruppe, zu der unter anderem auch Sabo Mobile IT gehört.