KI Vision für jedermann

Edge-Entwicklung in der Cloud

Komplexer werdende Anforderungen an die industrielle Bildverarbeitung verändern den Machine-Vision-Bereich – von Standard-Vision-Industriekameras hin zu intelligent mit ihrer (Geräte-)Umwelt interagierenden Lösungen. Anwendern unterschiedlicher Branchen eröffnen solche auf künstlicher Intelligenz (KI) basierenden Systeme neue Anwendungsmöglichkeiten. Doch zugleich stellt es Hersteller vor neue Herausforderungen, solche Technologien effizient und nachhaltig in kleine Edge-Geräte zu implementieren. Zudem soll es Anwendern möglich sein, selbstständig und flexibel Systemanpassungen vorzunehmen, ohne dass hierfür grundlegende, plattformspezifische Programmierarbeiten anfallen. Abhilfe schafft hier eine cloudbasierte Entwicklungsplattform, die KI-Vision-Systeme einer breiten Anwendergruppe mit unterschiedlichen Vorkenntnissen zugänglich macht.

Edge-Geräte

Bildverarbeitungsgeräte sind heute allgegenwärtig, und sie verwenden längst nicht mehr nur regelbasierte Bildverarbeitung, sondern arbeiten mit maschinellen Lernalgorithmen. Durch die Verlagerung der Bildverarbeitung und der Erzeugung von Ergebnissen in das Edge-Gerät selbst ändert sich der Ansatz und die Handhabung von Bildverarbeitungsaufgaben erheblich. Vor diesem Hintergrund ist es notwendig, Antworten auf wesentliche Fragen zu finden, wie z. B.:

- Welche Schnittstellen braucht das Gerät, hardware- und softwareseitig?
- Welche Werkzeuge können für die Steuerung, Programmierung oder Konfiguration des Geräts bereitgestellt werden?
- Wer sind die Zielgruppen, die diese Geräte implementieren, und welche Fähigkeiten benötigen sie, um bei diesem Vorhaben erfolgreich zu sein?

Herausforderung „Embedded“

Eingebettete Bildverarbeitungssysteme sind hochoptimierte Geräte, die in der Regel neben einer CPU auch Komponenten wie FPGAs (Field-Programmable Gate Array), GPUs (Graphical Processing Unit) oder ASICs (Application-Specific Integrated Circuit) verwenden, um ihre speziellen Aufgaben äußerst effizient zu erfüllen. Als offen programmierbare Plattform ist ein solches Edge-Device eine Spielwiese für Experten, die wissen, wie man die verschiedenen Komponenten anspricht und welche Aufgaben sie erfüllen können. Doch wer schon einmal Anwendungen für ein Embedded Device programmiert hat und die dafür notwendige Entwicklungsumgebung einrichten musste, weiß, dass dies erhebliche Vorkenntnisse erfordert. Nicht nur Wissen über die Einrichtung der Entwicklungsumgebung und deren Zusammenspiel mit dem Edge-Device, sondern auch über den Umgang mit Schnittstellen, Kommunikationsprotokollen, Debuggern, Toolchains und einer IDE (Integrated Development Environment).

Neue Herangehensweise

Die potenziellen Fähigkeiten einer neuen Generation von KI-basierten Geräten werden nicht nur die Herangehensweise an Bildverarbeitungsaufgaben verändern, sondern auch zu ganz neuen Anwendungsfällen führen. Das wiederum wird auch ganz neue Nutzergruppen für die Bildverarbeitung und KI begeistern. Für diese neuen Zielgruppen ist die Einstiegshürde in die Welt der Embedded Vision allerdings noch viel zu hoch. Gerätehersteller müssen dieses Ziel im Auge behalten: Jeder sollte in der Lage sein, seine eigene KI-basierte Bildverarbeitung zu entwerfen und auf einem Edge-Device auszuführen, auch ohne das oben erwähnte Spezialwissen. Und damit ergeben sich völlig neue Anforderungen an die Embedded-Vision-Entwicklung. Das klassische codebasierte Programmier-SDK für Software-Ingenieure ist nicht mehr länger das optimale Arbeitsmittel für alle und reicht auch nicht aus, um den gesamten Embedded Vision Workflow abzudecken.

Entwickeln in der Cloud

Die Entwicklung eines KI-Vision-Workflows für ein Edge-Device muss heute nicht mehr plattformspezifisch mit einer direkten Geräteanbindung erfolgen. Wenn die grundlegenden Gerätefunktionen in einem universell konfigurierbaren, modularen Funktionsbaukasten zur Verfügung ste-

Autor:
Dipl.- Ing. Heiko Seitz
Technischer Autor
IDS Imaging Development Systems GmbH
www.ids-imaging.de

hen, muss das Edge-Device nicht jedes Mal neu programmiert oder Sourcecode für eine andere Plattformarchitektur cross-compiliert werden. Das Training der ML-Algorithmen kann ohnehin schon völlig unabhängig vom Gerät erfolgen. Was dann noch fehlt, ist lediglich eine „Beschreibung" des Anwendungsworkflows. Das heißt, wie die Funktionen miteinander kombiniert werden müssen. Auch für dieses Zusammenspiel muss man heute keine spezielle textbasierte Programmiersprache mehr beherrschen.

„High-Level-Entwicklungsumgebung"

Mit dieser „High-Level-Entwicklungsumgebung" in der Cloud müssen sich Anwender weder mit plattformspezifischer Programmierung noch mit der speziellen Syntax einer Programmiersprache auseinandersetzen. Die gesamte Entwicklung mit einem KI-basierten Embedded Vision System findet in einer einzigen Softwareplattform in der Cloud statt. Der Anwender kann sich damit voll und ganz auf die Aufgabe der Anwendung konzentrieren, ohne auf ein physisches Gerät zugreifen zu müssen. Sowohl für die App-Erstellung als auch für erste Tests kann man auf Bilder oder ganze Datensätze zugreifen, die auch für das Training der neuronalen Netze verwendet werden. Auf diese Weise ist die Funktionalität einer App vollständig simulierbar. Natürlich sollten Tests auf physischer Hardware vor dem Einsatz erfolgen, jedoch können spezielle Situationen mit geeigneten Testdatensätzen auch bereits in der Cloud durchgespielt und Schritt für Schritt debuggt werden, bevor die App in der Kamera zum Einsatz kommt, wenn man nur den reinen Bildverarbeitungsteil betrachtet.

Beschreiben statt Programmieren

Auch bei der Programmierung eines Anwendungsablaufs gibt es Vereinfachungspotenzial. Durch einen mehr „anwendungsorientierten" Ansatz kann ein Ziel einerseits schneller und einfacher erreicht werden. Andererseits helfen mehr Übersicht und Wissen über die Gerätefunktionen auch, mehr Freiraum bei der Gestaltung des Prozesses zu schaffen. Es gilt also auch hier, das Für und Wider gegeneinander abzuwägen. Oder einfach zwei unterschiedliche Ansätze zu schaffen.

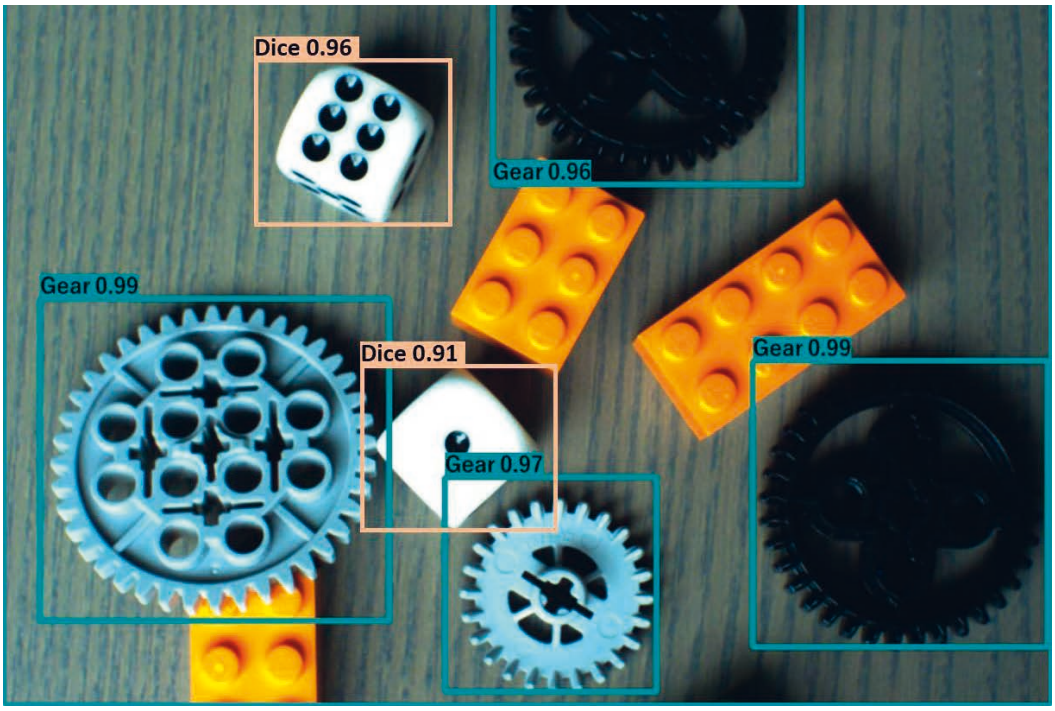

Bild 1: Einfaches Beispiel: Erkennung von Zahnrädern und Würfeln im selben Bild mit zwei neuronalen Netzen

Vereinfachung des Arbeitsablaufs

Ein Beispiel für die Vereinfachung des Arbeitsablaufs ist die Einführung einer Art „Interview-Modus", der den Nutzer zu Beginn eines Projekts unterstützt. Dem Nutzer werden dabei eine Auswahl an vorgefertigten Anwendungsfällen, einfache Abfragen sowie Tipps gezeigt, die ihn durch den gesamten Prozess der App-Erstellung führen. Ohne konkret wissen zu müssen, welche Art von neuronalem Netz bzw. welche Machine Learning Methode (ML) eingesetzt werden soll, stellt ein Assistent im Hintergrund die richtigen Komponenten zusammen, um Objekte zu finden, zu zählen und die Ergebnisse später über die gewünschten Geräteschnittstellen zur Verfügung zu stellen. Danach wird dem Nutzer mitgeteilt, welche Art von Bildern er je nach Anwendungsfall hochladen und wie er sie beschriften soll. Der „Anwendungsassistent" erstellt dann aus den Bildern geeignete Datensätze und trainiert ein neuronales Netz. In nur wenigen Schritten wird so eine komplette Embedded Vision App in der Cloud fertiggestellt, die der Nutzer nur noch hochladen, aktivieren und direkt auf seiner Kamera ausführen muss. Mit so einem Ansatz können viele einfache Anwendungsfälle vollständig abgebildet werden.

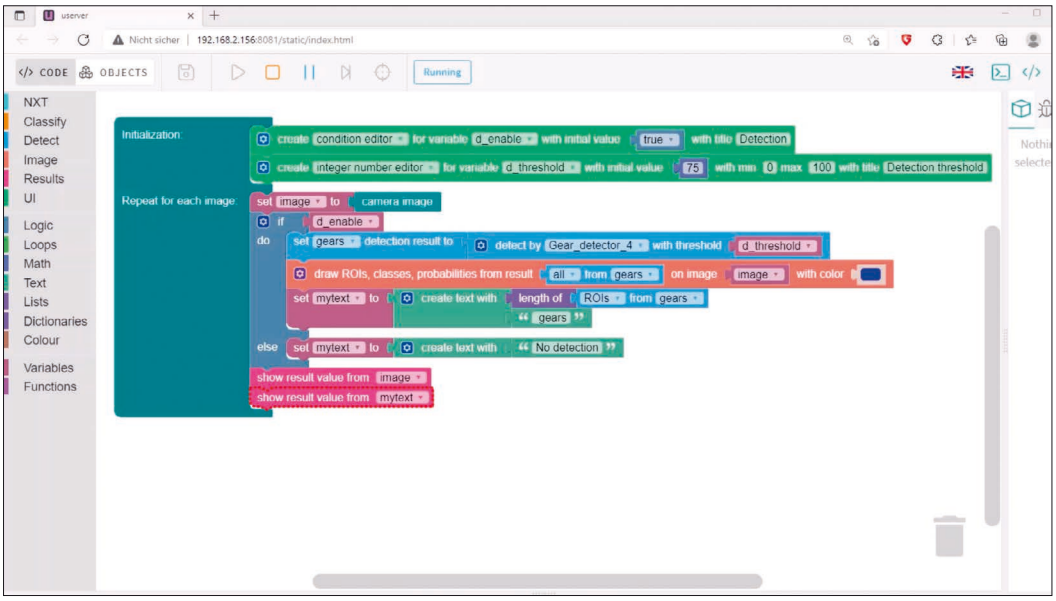

Bild 2: Blockbasierter Code für das Beispiel zur Erkennung von Zahnrädern und Würfeln im selben Bild mit zwei neuronalen Netzen

Block für Block

Komplexere Analysen mit mehreren Prüfkriterien, wie z. B. eine Qualitätssicherung in der Lebensmittelindustrie, ist mit dem Einsatz eines neuronalen Netzes nur schwer möglich. Der Trainingsprozess wäre viel zu aufwändig. Jedoch auch für eine zweistufige Kontrolle nach beispielsweise Fruchtart und -qualität, besteht keine Notwendigkeit, sofort auf C++ oder eine andere textbasierte Programmiersprache umzusteigen und den Komfort der Cloud-Entwicklung zu verlassen. Visuelle Editoren haben sich für diesen Ansatz schon immer gut geeignet. Ein Beispiel dafür ist die Verwendung eines blockbasierten Code-Editors, der die „Blockly"-Bibliothek von Google nutzt. Hier können Prozesse wie Puzzlestücke per Drag & Drop in Sequenzen unterschiedlicher Komplexität kombiniert werden. Das eröffnet eine wesentlich höhere Flexibilität der Anwendungsbeschreibung und macht den Prozess gleichzeitig leicht nachvollziehbar. Mit Hilfe einer intuitiven Benutzeroberfläche können so auch Anfänger schnell zu erfolgreichen Ergebnissen kommen.

Bild 1 zeigt ein einfaches Beispiel: es geht um die Erkennung von Zahnrädern und Würfeln im selben Bild mit zwei neuronalen Netzen. Bild 2 zeigt den Code dazu.

Blockbasierte Programmierung

Der Vorteil der blockbasierten Programmierung gegenüber einem „Interview- bzw. Assistenten Modus" ist die Möglichkeit, individuelle Abläufe zu erstellen. Variablen, Parameter und KI-Ergebnisse können leicht durch logische Verknüpfungen mit mathematischen Berechnungen, bedingten if/else-Anweisungen, wiederkehrenden Aktionen durch Schleifen oder sogar die Verwendung mehrerer neuronaler Netze verbunden werden, wie beispielsweise eine zweistufige Analyse nach Fruchtart und -qualität:

Ein Objektdetektor sorgt dabei für eine grundlegende Vorsortierung der verschiedenen Teile, gefolgt von einer detaillierten Fehleranalyse durch einen zweiten Klassifikator, um die Teile noch genauer zu kategorisieren. Solche Verfahren sind sonst nur mit einer komplexeren Entwicklungsumgebung und Kenntnissen einer geeigneten textbasierten Programmiersprache möglich. Wesentlich einfacher wäre, wenn diese Art Anwendung zunächst im „Interview-Modus" mit den notwendigen Komponenten zusammengesetzt werden könnte und man jederzeit mit einem blockbasierten Editor tiefer in den Workflow einsteigen könnte, wenn die Anwendung aufgrund der Komplexität des Problems etwas mehr Anpassungsaufwand erfordert (Bild 3).

Instant Apps

Ein weiterer erwähnenswerter Nebeneffekt dieser vereinfachten Anwendungserstellung ist, dass die so erstellten „Vision Apps" problemlos dynamisch veränderbar bleiben. Denn die Anwendungsbasis enthält dazu bereits alle verfügbaren Gerätefunktionalitäten in einer vollständig plattformspezifischen, vorkompilierten Bibliothek als Baukasten und hat darüber hinaus eine universelle und plattformunabhängige Benutzerschnittstelle. Damit kann der eigentliche Workflow jederzeit dynamisch verändert oder neu beschrieben werden, auch während die Anwendung ausgeführt wird. Zeitaufwändiges Cross-Debugging und Neukompilieren des geänderten Ablaufs ist nicht mehr notwendig. Das bedeutet, dass die Weiterentwicklung bei Bedarf auch direkt in der Kamera an der Maschine erfolgen kann.

Und der Gewinner ist...

... der Anwender. Denn mit so einem cloudbasierten KI-Vision System gibt es für jede Nutzergruppe mit unterschiedlichen Vorkenntnissen einen geeigneten Einstieg in die Anwendungsentwicklung mit KI-basierter Bildverarbeitung. Dadurch werden ganz neue Zielgruppen in die Entwicklung von Anwendungen (mit)einbezogen. Zum Beispiel kann ein Maschinenbediener jederzeit kleine Anpassungen vornehmen und so einen Prozess kontinuierlich optimieren. Die Wartung einer Anwendung wird einfacher, die Qualität kann leichter und mit weniger Aufwand gesteigert werden. Und das Wichtigste: Mit diesem Ansatz von Entwicklungswerkzeugen kann die Projektzeit, vom Konzept bis zur laufenden Anwendung, stark reduziert werden.

Wer schreibt:

IDS Imaging Development Systems GmbH ist ein führender Hersteller und Pionier in der industriellen Bildverarbeitung. IDS entwickelt leistungsstarke und vielseitig einsetzbare USB, GigE und 3D Kameras sowie Modelle mit künstlicher Intelligenz. Das Anwendungsspektrum erstreckt sich über industrielle sowie nicht-industrielle Branchen. Das KI-Vision System IDS NXT eröffnet zudem neue Einsatzgebiete und mit visionpier betreibt IDS einen Online-Marktplatz, der Anbieter von Bildverarbeitungslösungen und Endkunden zusammenbringt. ◄

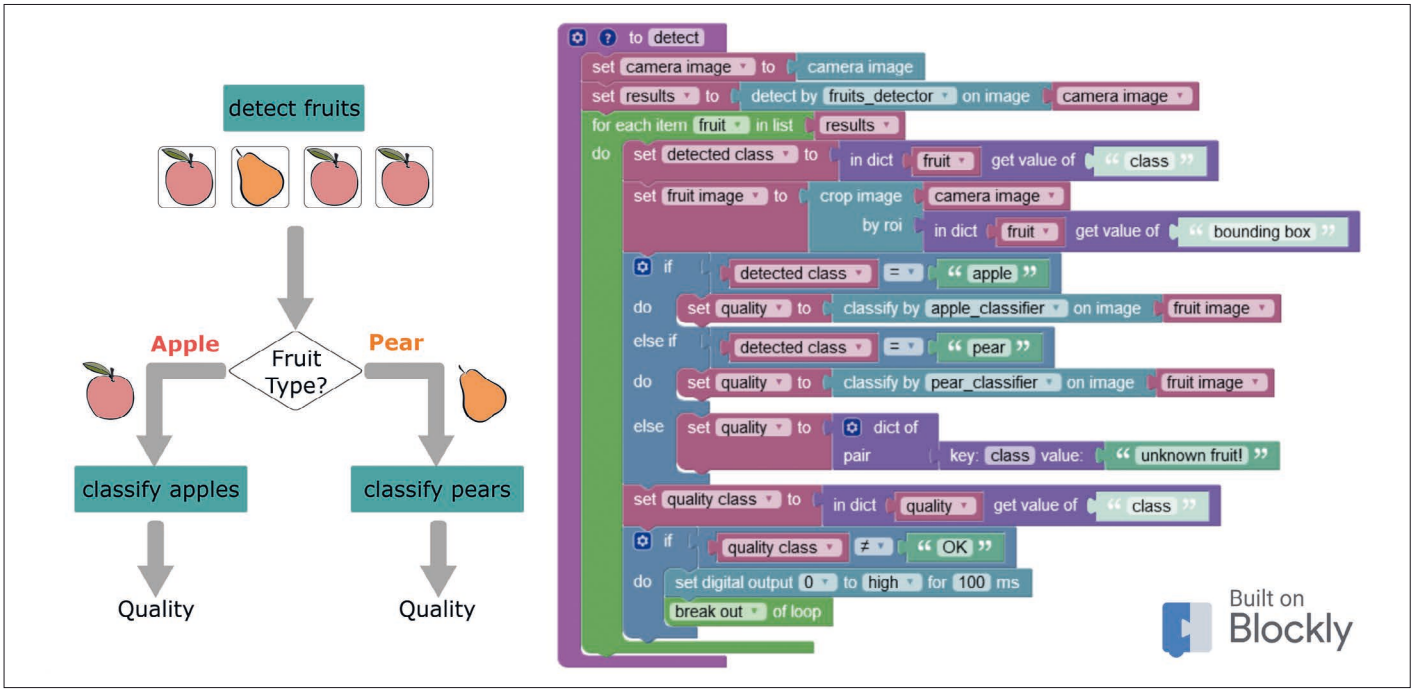

Bild 3: Eine kombinierte 2-stufige Analyse mit zwei unterschiedlich trainierten CNNs kann mit dem blockbasierten Editor von IDS NXT einfach realisiert werden
© 2022 IDS Imaging Development Systems GmbH