Zschech, Patrick; Sager, Christoph; Siebers, Philipp; Pertermann, Maik

Mit Computer Vision zur automatisierten Qualitätssicherung in der industriellen Fertigung: Eine Fallstudie zur Klassifizierung von Fehlern in Solarzellen mittels Elektrolumineszenz-Bildern

HMD Praxis der Wirtschaftsinformatik

Provided in Cooperation with:
Springer Nature

Suggested Citation: Zschech, Patrick; Sager, Christoph; Siebers, Philipp; Pertermann, Maik (2020) : Mit Computer Vision zur automatisierten Qualitätssicherung in der industriellen Fertigung: Eine Fallstudie zur Klassifizierung von Fehlern in Solarzellen mittels Elektrolumineszenz-Bildern, HMD Praxis der Wirtschaftsinformatik, ISSN 2198-2775, Springer Fachmedien Wiesbaden, Wiesbaden, Vol. 58, Iss. 2, pp. 321-342, https://doi.org/10.1365/s40702-020-00641-8

HMD (2021) 58:321–342
https://doi.org/10.1365/s40702-020-00641-8

SCHWERPUNKT

Mit Computer Vision zur automatisierten Qualitätssicherung in der industriellen Fertigung: Eine Fallstudie zur Klassifizierung von Fehlern in Solarzellen mittels Elektrolumineszenz-Bildern

Patrick Zschech · Christoph Sager · Philipp Siebers · Maik Pertermann

Eingegangen: 26. Februar 2020 / Angenommen: 24. Juni 2020 / Online publiziert: 7. Juli 2020

Zusammenfassung Die Qualitätssicherung bei der Produktion von Solarzellen ist ein entscheidender Faktor, um langfristige Leistungsgarantien auf Solarpanels gewähren zu können. Die vorliegende Arbeit leistet hierzu einen Beitrag zur automatisierten Fehlererkennung auf Wafern, indem Elektrolumineszenz-Bilder eines realen Herstellungsszenarios mithilfe von verschiedenen Computer-Vision-Modellen klassifiziert werden. Die Herausforderung besteht hierbei nicht nur darin, defekte Wafer von funktionsfähigen zu separieren, sondern gleichzeitig auch zwischen spezifischen Fehlerarten zu unterscheiden, während geringe Inferenzzeiten sicherzustellen sind. Zu diesem Zweck werden neben einfachen statistischen Modellen verschiedene Deep-Learning-Architekturen auf Basis von Convolutional Neural Networks (CNNs) verprobt und miteinander vergleichen. Ziel der Arbeit ist es, verschiedene Klassifizierungsansätze unterschiedlicher Komplexität zu testen und auf ihre praktische Einsatzfähigkeit unter realen Bedingungen zu untersuchen. Die Fallstudie zeigt, dass je nach Situation unterschiedliche Modelle ihre Existenzberechtigung haben und in Kombination sehr gute Ergebnisse erzielen. So lassen sich bereits mit statistischen Modellen und einfachen CNN-Varianten zuverlässige Aussagen mit Genauigkeiten von über 99 % bei Fehlertypen einfacher bis mittlerer Erkennbarkeit

P. Zschech (✉) · C. Sager · P. Siebers
Lehrstuhl für Wirtschaftsinformatik, Business Intelligence Research, TU Dresden, 01062 Dresden, Deutschland
E-Mail: patrick.zschech@tu-dresden.de

C. Sager
E-Mail: christoph.sager@tu-dresden.de

P. Siebers
E-Mail: philipp.siebers@tu-dresden.de

M. Pertermann
Kontron AIS GmbH, Otto-Mohr-Straße 6, 01237 Dresden, Deutschland
E-Mail: maik.pertermann@kontron-ais.com

realisieren. Werden die Fehlerbilder demgegenüber diffuser und soll die Nachvollziehbarkeit der Ergebnisse durch positionsgenaue Lokalisierung von Fehlerobjekten gewährleistet werden, sind fortgeschrittenere Ansätze auf Basis sogenannter Region-Proposal-Netzwerke erforderlich, die allerdings auch mit einem erhöhten Labeling-Aufwand beim Annotieren der Fehlerobjekte einhergehen. Da die Umsetzung sämtlicher Modelle ausschließlich auf Open Source Tools wie zum Beispiel TensorFlow, Keras und OpenCV basiert, demonstriert die Fallstudie zudem, welche Möglichkeiten durch frei verfügbare Lösungen im Bereich von Computer Vision geboten werden.

Schlüsselwörter Deep Learning · Industrie · Maschinelle Bildverarbeitung · Objekterkennung · Photovoltaik · Qualitätssicherung

Automated Quality Assurance in Manufacturing Using Computer Vision: A Case Study on Classifying Defects in Solar Cells Based on Electroluminescence Images

Abstract Quality assurance in the production of solar cells is a decisive factor for long-term performance guarantees on solar panels. This work contributes to this area in developing computer vision models to automatically detect defects on wafers by classifying electroluminescence images from a real manufacturing scenario. The challenge is not only to separate defective wafers from flawless ones but also to distinguish between specific types of defects while ensuring low inference times. For this purpose, simple statistical models, as well as different kinds of deep learning architectures based on convolutional neural networks (CNNs), are tested and compared with each other. Therefore, this work aims to evaluate multiple classification approaches of varying complexity levels while examining their practical applicability under real industrial conditions. The case study shows that all models have their right to exist and achieve excellent results in combination. While statistical models and simple CNNs provide reliable statements with accuracies up to 99% for defect types of simple to medium detectability, more advanced approaches based on region proposal networks are required once the defect images become more diffuse. The more advanced approaches allow a precise object localization of defects; however, they are also associated with increased labeling effort when annotating wafer images. Since the implementation of all models is based exclusively on open source tools such as TensorFlow, Keras, and OpenCV, the case study also demonstrates the possibilities offered by freely accessible solutions in the field of computer vision.

Keywords Deep learning · Industry · Computer vision · Object detection · Photovoltaic · Quality assurance

1 Einleitung

Die technologischen Errungenschaften im Bereich von Computer Vision führen dazu, dass sich zahlreiche Wirtschaftszweige revolutionär verändern werden. So lassen

sich aufwendige manuelle Tätigkeiten wie das Zählen und Verfolgen von Objekten oder die Entdeckung anomaler Ereignisse zunehmend durch intelligente Systeme und maschinelle Bildverarbeitungsverfahren unterstützen (Szeliski 2010; Heinrich et al. 2019b). Erfolgsversprechende Anwendungen finden sich beispielsweise bereits im Bereich des autonomen Fahrens (Friederich and Zschech 2020), in der Biomedizin zur Erkennung auffälliger Zellstrukturen (Griebel et al. 2019) oder in der Agrarwirtschaft bei der automatisierten Ertragsprognose (Heinrich et al. 2019c).

Ein weiterer Bereich, der wesentlich von modernen, bildverarbeitenden Technologien profitieren kann, ist die industrielle Fertigung. Auf Basis umfangreichen Bildmaterials lassen sich z. B. Instandhaltungssysteme mit weiteren Informationen anreichern (Zschech 2018), Qualitätskontrollen können effizienter ausgestaltet werden (Trinks and Felden 2019) und Robotik-Systeme unterstützen bei logistischen Vorgängen (Thiel et al. 2018). Insbesondere in der Qualitätssicherung, wo das Ziel darin besteht, über die gesamte Produktionskette hinweg eine hohe Produktgüte zu gewährleisten, ergeben sich weitreichende Potenziale. Durch die Anwendung leistungsfähiger Algorithmen können Testbilder schneller und effizienter ausgewertet werden. Dadurch ergeben sich große Entlastungen für die prüfenden Mitarbeiter, Produktionsprozesse können stabiler betrieben werden und es lassen sich hohe Kosten und aufwendige Korrekturmaßnahmen aufgrund zu spät erkannter Qualitätsbeeinträchtigungen vermeiden.

Vor diesem Hintergrund stellt der vorliegende Artikel eine industrielle Fallstudie zur automatisierten Klassifizierung von Fehlern bei der Herstellung von Solarzellen vor. Die Herausforderung bestand dabei nicht nur darin, defekte Wafer von funktionsfähigen zu separieren, sondern gleichzeitig auch zwischen spezifischen Fehlerarten zu unterscheiden. Darüber hinaus galt es als weitere Nebenbedingung geringe Inferenzzeiten sicherzustellen, die sich aus einer hohen Taktfrequenz des zugrundeliegenden Produktionsprozesses ergeben. Zur Problemlösung werden verschiedene Methoden der maschinellen Bildverarbeitung auf Basis von Elektrolumineszenz-Bildern verprobt und prototypisch angewendet. Dazu kommen neben statistischen Verfahren (siehe Abschn. 5.1) insbesondere künstliche neuronale Netze unterschiedlicher Komplexitätsstufen zum Einsatz. Letztere reichen von einfachen *Convolutional Neural Networks* (CNNs) zur Bildklassifizierung (Abschn. 5.2) über *Region-based CNNs* zur Lokalisierung von Bildobjekten (Abschn. 5.3) bis hin zu Mask R-CNNs zur Objektsegmentierung (Abschn. 5.4). Das praxisgeleitete Forschungsziel der Arbeit besteht folglich darin, Klassifizierungsansätze unterschiedlicher Komplexität zu testen und auf ihre praktische Einsatzfähigkeit zu untersuchen. Die Umsetzung der einzelnen Computer-Vision-Verfahren erfolgt ausschließlich auf Basis von Open Source Software, wodurch das Leistungspotenzial frei zugänglicher Community-Lösungen für kritische Fragestellungen in industriellen Anwendungsszenarien demonstriert wird.

Zur Strukturierung der Fallstudie orientiert sich die Arbeit am klassischen Vorgehen datengetriebener Analyseprojekte mit Bezug auf die sechs Phasen (i) Domain Understanding, (ii) Data Understanding, (iii) Data Preparation, (iv) Modeling, (v) Evaluation und (vi) Deployment (Kurgan and Musilek 2006). Dieser Struktur folgend untergliedern sich die weiteren Ausführungen wie folgt: Nach der Darstellung der erforderlichen Grundlagen in Abschn. 2 wird zunächst ein Verständnis

über die Problemstellung innerhalb der Fallstudie in Abschn. 3 vorgestellt. Anschließend erfolgt in Abschn. 4 die Betrachtung der vorliegenden Datengrundlage. Abschn. 5 dient der Erläuterung der erforderlichen Vorverarbeitungsschritte und der Anwendung unterschiedlich komplexer Computer-Vision-Modelle, gefolgt von einer vergleichenden Evaluation der Ergebnisse in Abschn. 6. Abschließend wird in Abschn. 7 ein Fazit gezogen und ein Ausblick für weitere Arbeiten skizziert.

2 Grundlagen

Computer Vision ist ein interdisziplinäres Forschungsgebiet, das sich mit der Entwicklung von Modellen und Methoden zur maschinellen Erfassung, Verarbeitung und Auswertung von Bildmaterial oder anderweitig hochdimensionalen Daten beschäftigt (Szeliski 2010). Zu diesem Zweck bedient sich das Feld zunehmend Algorithmen des maschinellen Lernens, insbesondere aus dem Gebiet des sogenannten *Deep Learning*. Dabei handelt es sich um künstliche neuronale Netze mit komplexen, tief-verschachtelten Netzwerkarchitekturen, um interne Datenrepräsentationen über mehrere Abstraktionsebenen hinweg besser erkennen und verarbeiten zu können (LeCun et al. 2015).

Eine spezielle Architekturvariante, die in der Bilderkennung eine zentrale Rolle einnimmt, bildet die Klasse der sogenannten *Convolutional Neural Networks* (LeCun et al. 2015; Heinrich et al. 2019c). Hierbei wird jedes Eingabebild anhand seiner Pixel als Matrix der Dimension Höhe × Breite repräsentiert. Jede Zelle der Matrix enthält einen Pixelwert, der die Ausprägung des Bildpunktes widerspiegelt. Bei einem Graustufenbild entspricht der Pixelwert der Bestrahlungsstärke und nimmt Werte zwischen 0 (schwarz) und 255 (weiß) an. Bei Farbbildern, wie z. B. RGB-Bildern (Rot-Grün-Blau Farbsystem), steht der Pixelwert für die Intensität einer Farbe im Bild. Folglich gibt es für jede Farbe einen eigenen Farbkanal, der eine eigenständige Matrix darstellt. Durch Anwendung sogenannter Faltungsfunktionen werden Eingabematrizen dann in kleine Bereiche unterteilt und abgetastet, wodurch sich wichtige (visuelle) Merkmale (engl. *Features*) extrahieren lassen. In einem neuronalen Netz können prinzipiell verschiedene solcher Abtastfunktionen (engl. *Filter Kernel*) in mehreren Schichten (engl. *Convolutional Layer*) hintereinander angeordnet sein, um immer abstraktere Merkmale zu extrahieren und für die nachfolgenden Verarbeitungsschichten als eine Art Ansammlung (engl. *Feature Map*) zur Verfügung zu stehen. Einfache Merkmale der vorderen Schichten sind typischerweise Ecken, Kanten oder Rundungen, während nachgelagerte Schichten eher komplexere Formen abbilden können wie etwa Umrisse oder spezielle Merkmale eines Objektes (z. B. Reifen eines Fahrzeugs) (Zeiler and Fergus 2014). Da es sich meist um sehr viele hochdimensionale Feature Maps handelt, können zur nachträglichen Komplexitätsreduktion zwischen den einzelnen Convolutional Layern sogenannte *Pooling Layer* im Sinne von zusammenfassenden Schichten eingesetzt werden. Diese dienen der Komprimierung von extrahierten Bildinformationen, um den Berechnungsaufwand eines CNN zu reduzieren. Generell wird der erste Teil eines CNN auch als Feature-Extraktor bezeichnet, da er dafür verantwortlich ist, repräsentative Merkmale zur Beschreibung eines Bildes zu extrahieren. Der zweite Teil funktioniert

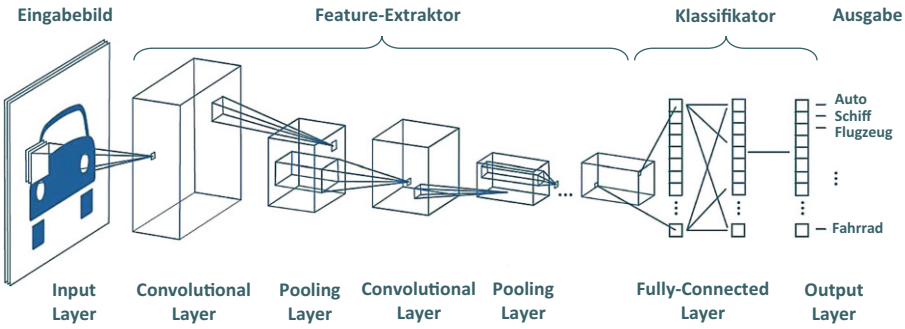

Abb. 1 Allgemeiner Aufbau eines CNN. (In Anlehnung an MathWorks 2017)

anschließend wie ein traditioneller Klassifikationsalgorithmus, bei dem die zuvor extrahierten Merkmale als Eingabegrößen für eine einfache Klassifikationsfunktion dienen. An dieser Stelle sorgt ein sogenannter *Fully-Connected Layer* dafür, dass die verarbeiteten Informationen der vorangegangenen Schichten wieder zusammengeführt werden. Die Anzahl der Neuronen in diesem Layer korrespondiert dann üblicherweise mit der Anzahl an Klassen, zwischen denen das Netz unterscheiden soll (LeCun et al. 2015). Der Aufbau eines klassischen CNN ist in Abb. 1 skizziert. Zudem sind CNNs je nach Architekturvariante mit verschiedenen Abtastfunktionen und Netzwerktiefen ausgestattet.

Auf dieser grundlegenden Funktionsweise aufbauend hat sich in den letzten Jahren eine Reihe an weiteren Netzwerkvarianten herausgebildet. Dazu gehört zum Beispiel der Ansatz der sogenannten Region-Based Convolutional Neural Networks (R-CNNs), in denen klassische CNNs mit sogenannten *Region Proposal Networks* (Girshick et al. 2014) kombiniert werden. Dadurch ist das Hybridnetzwerk neben einer reinen Klassifizierung auch in der Lage, eine Lokalisierung von Objekten durchzuführen. Zunächst werden Bildregionen ermittelt, die sich aufgrund ihrer Bildstruktur vom Hintergrund abheben und anschließend werden mittels CNN die Features dieser Bildregionen extrahiert und zur Objektbestimmung genutzt. Darauf aufbauend werden gleichklassifizierte Regionen zur algorithmischen Bestimmung der Positionen dieser Objekte genutzt. Eine recheneffiziente Implementierung dieses Konzepts bildet die sogenannte Faster-R-CNN-Architektur (Ren et al. 2017). Die Architektur ist mittlerweile weit verbreitet und war unter anderem auch die Grundlage vieler Wettbewerbsbeiträge, wie etwa dem ersten Platz bei dem ILSVRC- und COCO-Wettbewerb im Jahr 2015 (He et al. 2015).

Eine weitere State-of-the-Art-Modellarchitektur bilden die sogenannten Mask R-CNNs, die aktuell die am weitesten fortgeschrittene Stufe in der Entwicklung der R-CNNs darstellen (He et al. 2017). Ähnlich wie die zuvor beschriebene Architektur basieren sie auch auf dem Region-Proposal-Ansatz. Im Gegensatz zur bloßen Vorhersage von Bounding Boxen in einem Bild (engl. *Object Detection*) ermöglichen sie jedoch eine pixelgenaue Klassifizierung (engl. *Instance/Semantic Segmentation*). Dies setzt allerdings auch eine sehr präzise Annotation der Trainingsdaten voraus, was in der Regel durch aufwendige Kennzeichnungen in Form von Polygonen realisiert wird.

3 Vorstellung der Fallstudie

Hersteller von Photovoltaik-Wafern (Solarzellen) müssen hohen Qualitätsansprüchen gerecht werden. Einerseits muss eine gewisse Minimalleistung über einen festen Zeitraum garantiert werden. Dabei gewährt der Hersteller zumeist eine Produktgarantie zwischen fünf und zehn Jahren, in denen der Verbraucher ein Anrecht auf die Beseitigung von nicht selbstverschuldeten Mängeln hat. Oft wird auch eine sogenannte Leistungsgarantie gegeben, welche sich teilweise über 25 Jahre erstreckt (Tsai et al. 2013). Andererseits müssen Materialfehler, welche Sicherheitsrisiken verursachen könnten, rechtzeitig erkannt werden. So können sich Risse oder Druckpunkte bei großer Sonneneinstrahlung stark erhitzen und ausdehnen. Im schlimmsten Szenario bricht dabei das Sicherheitsglas und das Solarmodul wärmt sich so stark auf, dass es zu einem Brand kommt (Köntges et al. 2014). Im Sinne der Risikominimierung dürfen folglich nur Wafer mit hoher Qualität verbaut werden.

Darüber hinaus unterliegt der Herstellungsprozess kristalliner Solarzellen ständigen Weiterentwicklungen durch neuartige Strukturen und Beschichtungssysteme, mit denen sich die Effizienz der Zellen weiter steigern lässt. Neben der Effizienz im Sinne elektrischer Leistungsmerkmale spielen jedoch auch die optischen Eigenschaften bei der Bewertung einer Solarzelle eine wichtige Rolle, da das Endprodukt letztlich ein von vielen Kunden sichtbares Produkt darstellt.

Um diese Qualitätsaspekte zu bewerten, kommen beim begleiteten Fallstudienpartner, einem international agierenden Hersteller von Photovoltaik-Anlagen, optische Systeme zum Einsatz, mit denen Testbilder erstellt werden können. Diese werden mittels *Elektrolumineszenz* (EL) gewonnen, wobei das Prinzip des photovoltaischen Effektes umgekehrt wird. Statt Strom durch eine photovoltaische Zelle zu erzeugen, werden die Zellen durch das Anlegen einer Spannung und entsprechenden resultierenden Stromfluss zum Leuchten angeregt. Bedingt durch die physikalischen Eigenschaften des verwendeten Halbleiters als Wafer-Material, speziell die Bandlücke von Silizium, erfolgt dieses Leuchten im infraroten Spektralbereich und wird von einer IR-empfindlichen-Kamera aufgezeichnet. Stellen, die den Strom gut leiten, leuchten dabei heller als andere Bereiche. Diese Eigenschaft ist nützlich, um markante Fehlerbilder zu erkennen, da fehlerhafte Stellen der Zelle wie z. B. Kratzer oder Risse den Strom schlechter leiten und somit dunkler erscheinen.

Eine solche Prüfung wurde bisher nur in Einzelfällen, vollständig manuell und zeitaufwendig ausgeführt. Es besteht jedoch die Vermutung, dass mit einer automatischen Erkennung von Fehlern während der finalen Qualitätskontrolle in der Zellfertigung eine höhere Qualität der ausgelieferten Produkte erreicht werden kann. Gleichzeitig kann die Analyse der Daten zu einer Verbesserung der Qualität in der Herstellung der Solarzellen führen. Die Aufgabe bestand demzufolge darin, die in der Produktion bereits für jede einzelne Solarzelle anfallenden EL-Bilder automatisiert zu klassifizieren und erkannte Fehler in die Sortierung der Zellen an der finalen Qualitätskontrolle einfließen zu lassen.

Als Ziele der Datenanalyse wurden hierfür verschiedene Meilensteine definiert. Das erste Ziel ist eine binäre Klassifizierung der Wafer in Zellen mit Fehler („defekt“) bzw. Zellen ohne Fehler („funktionsfähig“). Damit soll die Auslieferung von minderwertigen Zellen reduziert und somit die Qualität der Photovoltaik-Module

insgesamt gesteigert werden. Darüber hinausgehende Rückschlüsse auf Fehlerarten bzw. den Ort der Fehlerentstehung sind dabei nur über eine manuelle weitere Betrachtung der aussortierten Wafer möglich. Der zweite Meilenstein geht über diese Anforderungen hinaus und fordert eine Einteilung der aussortierten Wafer in eine von acht vorliegenden Fehlerklassen (vgl. Abschn. 4). Dadurch können gezieltere Rückschlüsse auf die Herkunft der Fehler gezogen werden und somit evtl. Produktionsschritte angepasst bzw. Warnungen frühzeitig ausgegeben werden.

Als wesentliche Nebenbedingung wurde zudem eine maximale Analysedauer bzw. Inferenzzeit von 2 s pro Wafer gefordert sowie eine Fehlertoleranz von 2 % als Orientierung vorgegeben. Die kurze Inferenzzeit resultiert aus der Taktrate der Produktion von ca. einer Sekunde pro Wafer und dem Abstand zwischen der EL-Kamera und dem nachgelagerten Sortierer. Die Qualitätsprüfung darf folglich nicht den Rest der Produktion verlangsamen.

4 Betrachtung der Datengrundlage

Bei der gegebenen Datengrundlage handelt es sich um EL-Bilder im TIFF-Dateiformat mit einer Auflösung von 1024 × 1024 Pixel. Sie liegen in Graustufen vor und kommen bei einem einzelnen Farbkanal auf eine Größe von jeweils 1,1 MB. Insgesamt stehen für Analysezwecke 6037 Testbilder aus einem Produktionszeitraum von circa acht Monaten zur Verfügung. Diese setzen sich aus 3786 fehlerhaften sowie 2251 fehlerfreien Wafern zusammen. Die markanten optischen Merkmale auf den Aufnahmen erlauben zudem die Bildung von charakteristischen (Fehler-)Klassen. Nach aktuellem Kenntnisstand der Fachexperten sind daraus die folgenden zehn Klassen ableitbar: (i) niedrige Helligkeit, (ii) Wafer auf Pin, (iii) Bruchstellen, (iv) Kratzer, (v) Punkte, (vi) Verschmutzung, (vii) Splitter im Tester, (viii) Flecken, (ix) schlechte Positionierung und (x) gute Wafer. Für eine prägnante Benennung wurden alle Klassen mit einer englischen Bezeichnung versehen. Tab. 1 fasst die

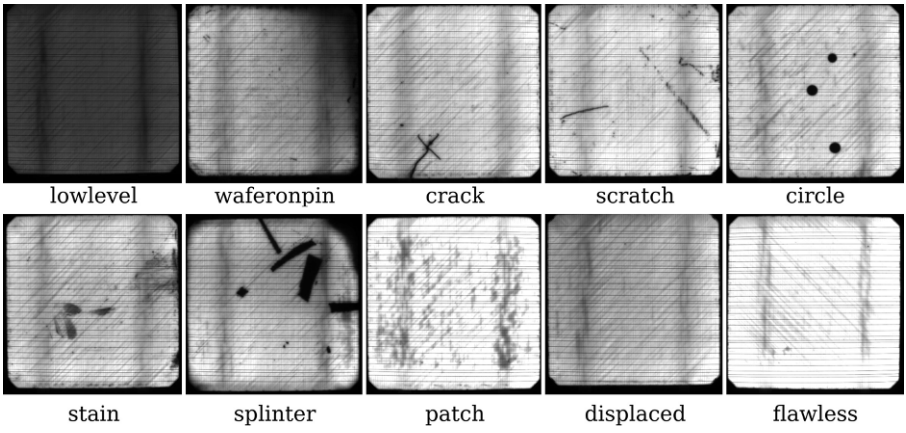

Abb. 2 Typische EL-Bilder von Wafern der jeweiligen (Fehler-)Klassen

Tab. 1 Beobachtbare (Fehler-)Klassen innerhalb des bereitgestellten Datenbestands

Klasse	Label	Beschreibung	Typ	Erkennbarkeit	Anzahl nach Labeling I	Anzahl nach Labeling II
Niedrige Helligkeit	*lowlevel*	Geringe Helligkeit im Gesamtbereich	Fehler	Hoch	420	–
Wafer auf Pin	*waferonpin*	Geringe Helligkeit im Eckbereich	Fehler	Hoch	256	–
Bruchstellen	*crack*	X-förmige Muster	Fehler	Mittel	577	587
Kratzer	*scratch*	Geradlinige Formen	Fehler	Mittel	579	837
Punkte	*circle*	Dichte, kreisrunde Verdunklungen	Fehler	Mittel	351	517
Verschmutzung	*stain*	Staubartige Flecken & Punktwolken	Fehler	Mittel	592	1062
Splitter im Tester	*splinter*	Verteilte, splitterartige Formen	Fehler	Mittel	79	79
Flecken	*patch*	Großflächige, schlierenartige Formen	Fehler	Niedrig	932	202
Schlechte Position	*displaced*	Verschobener Wafer-Ausschnitt	i. O.	–	993	–
Gute Wafer	*flawless*	Makellose Aufnahmen	i. O.	–	1258	–
Gesamt	–	–	–	–	6037	–

Eigenschaften der einzelnen Klassen zusammen. Zur Veranschaulichung wird in Abb. 2 ein repräsentatives EL-Bild je Klasse dargestellt.

Bei genauerer Betrachtung der einzelnen Klassen lassen sich detaillierte Eigenschaften feststellen. So kann beispielsweise bei den Fehlerklassen zwischen Fehlern des Wafers an sich (*lowlevel* und *waferonpin*) und Fehlern auf dem Wafer (*crack, scratch, circle, stain, splinter* und *patch*) differenziert werden. Fehler des Wafers an sich sind meist verhältnismäßig leicht zu erkennen. Diese sind dadurch gekennzeichnet, dass sie Bereiche geringer Helligkeit aufweisen und damit ineffiziente Wafer abbilden. Während sich diese Ausprägung bei der Klasse *lowlevel* über den Gesamtbereich erstreckt, bedingt durch einen geringeren Wirkungsgrad, tritt bei der Klasse *waferonpin* die Erscheinung nur in den Eckbereichen auf, da die Zellen in der Produktion fehlerhaft auf einem Bolzen auflagen.

Bei Fehlern auf dem Wafer lassen sich demgegenüber deutliche Unterschiede in der Erkennbarkeit feststellen. Beispielsweise umfasst die Klasse *stain* staubartige Flecken auf der Wafer-Oberfläche. Kleine Flecken ähneln dabei stark der Klasse *circle*, sind jedoch in der Regel weniger rund und nicht klar abgrenzbar. Auch die Größe des Fehlerbildes stellt eine Herausforderung dar. Im Gespräch mit den Fachexperten des Herstellers ergaben sich beispielsweise keine klaren Schwellwerte, ab wann ein *circle* nicht mehr als Fehler gewertet werden sollte. Das erschwert vor allem ein angemessenes Labeln von Bildern, da diesbezüglich mehrfach subjektive Entscheidungen getroffen werden müssen, die dann auch Einfluss auf die Modellergebnisse haben. Ein besonders herausforderndes Zuordnungsproblem besteht zudem

beim Auftreten der Fehlerklasse *patch*, da diese teilweise bereits für das menschliche Auge nur sehr schwer zu erkennen ist.

Bei den Bildern fehlerfreier Wafer kann demgegenüber zwischen makellosen Aufnahmen (*flawless*) und verschobenen Wafer-Ausschnitten (*displaced*) differenziert werden. Bei Letzteren handelt es sich um Testbilder, die zwar keine Fehler abbilden, bei denen sich jedoch die Wafer zum Zeitpunkt der Bildaufnahme nicht zentral auf den Förderbändern befanden. Aus diesem Grund werden sie von der Modellbildung ausgeschlossen.

Nach der Sichtung des gesamten Datenmaterials wurden die einzelnen Bilder gemeinsam mit einem Fachexperten einem ersten manuellen Labeling-Vorgang (I) unterzogen. Gemäß des ersten Meilensteins bestand das Ziel hierbei zunächst darin, zwischen EL-Bildern fehlerfreier und fehlerbehafteter Wafer zu unterscheiden. Gleichzeitig sollten alle Fehlerbilder bereits ihren jeweiligen Unterklassen zugeordnet werden, wobei es häufiger vorkam, dass sich auf einem Wafer mehrere Fehler verschiedener Klassen befanden. Bei der Anwendung einer ausdifferenzierten Annotation hätte sich somit der manuelle Labeling-Aufwand wesentlich erhöht. Stattdessen wurde jedes einzelne EL-Bild in diejenige Klasse sortiert, in der die charakteristischen Erkennungsmerkmale am stärksten ausgeprägt erschienen. Auf diese Weise wurde die Datengrundlage initial mit einer Dauer von 4–6 s pro Testbild einer der zehn Klassen zugeordnet.

Die Anwendung dieser Annotation ist zwar schneller durchführbar, sie bringt jedoch die Einschränkung mit sich, dass das Training eines Modells zur Erkennung von zwei oder mehreren Fehlerklassen nicht möglich ist, sobald mehrere der zu trainierenden Fehlerklassen auf einem Wafer auftauchen. Aus diesem Grund wurde für einige Fehlerklassen im späteren Verlauf ein weitaus differenzierterer Labeling-Vorgang (II) durchgeführt (siehe Abschn. 5.1).

Tab. 1 liefert einen Überblick über die einzelnen Klassenverteilungen. Der Umfang der Fehlerklassen ist insgesamt positiv zu bewerten. Einzige Ausnahme bildet die Fehlerklasse *splinter* mit nur 79 Bildern, die allerdings dennoch ausreichend vielfältige Ausprägungen enthält, um mithilfe von einfachen Data-Augmentation-Verfahren noch mit erfolgreichen Modellergebnissen rechnen zu können. Darüber hinaus konnten einige seltene Fehlerbilder identifiziert werden, für die bisher keine bekannte Klasse existierte (z. B. halbmondförmige Kratzer). Tatsächlich war die Menge an Fehlerbildern jedoch zu gering, um sie für ein Modelltraining zu berücksichtigen.

5 Vorverarbeitung und Modellerstellung

Zur automatischen Fehlererkennung lassen sich eine Reihe von unterschiedlichen Verfahren heranziehen, wobei im vorliegenden Fall neben Aspekten der Genauigkeit auch Restriktionen bezüglich der Verarbeitungszeit und Ressourcennutzung zu beachten sind. Insbesondere bei der Unterscheidung der verschiedenen Fehlertypen mit ihren individuellen Herausforderungen der Erkennbarkeit, gilt es unterschiedlich komplexe Verfahren im praktischen Einsatz zu verproben. Aus diesem Grund wird versucht, die einzelnen Fehlerklassen stufenweise mit möglichst kleinen und

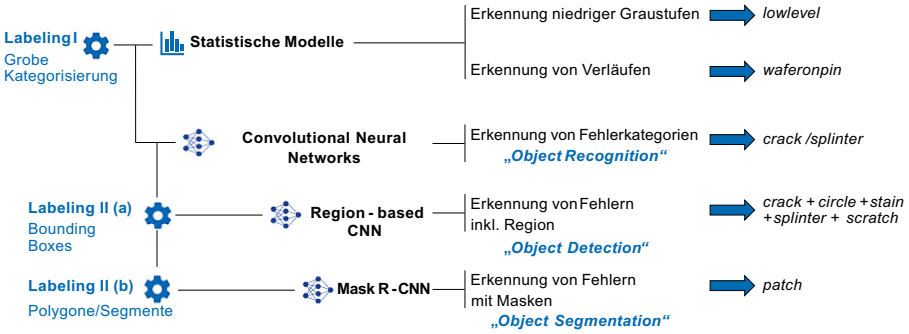

Abb. 3 Gesamtvorgehen zur Modellentwicklung

schnellen Modellen zu klassifizieren. Dafür werden zunächst einfache statistische Modelle entwickelt, um anschließend komplexere Deep-Learning-Modelle nur für schwieriger zu erkennende Fehler einsetzen zu müssen (siehe Abb. 3). Dieses stufenweise Vorgehen ermöglicht zudem eine Parallelisierung der Modellberechnung und ein vorzeitiges Abbrechen weiterer Verfahren, sobald ein Fehler sicher erkannt wurde. Die einzelnen Verfahren sowie die erforderlichen Vorverarbeitungsschritte werden in den nachfolgenden Abschnitten detailliert beschrieben. Die Implementierung sämtlicher Schritte erfolgt in der Programmiersprache Python unter Verwendung einschlägiger Machine Learning Frameworks, die als Open Source Tools frei zugänglich zur Verfügung stehen (Chollet 2018).

5.1 Datenvorverarbeitung und erweitertes Labeling

Die unterschiedlichen Verfahren erfordern jeweils verschiedene Arten der Datenvorverarbeitung. Für die statistischen Modelle und einfachen CNNs werden die Bilder mittels *OpenCV*[1] ausgelesen und anschließend in einem speicheroptimierten *Numpy-Array* abgelegt. Das für die CNN-Erstellung verwendete Framework *Keras*[2] (Chollet 2018) erwartet daraufhin noch eine Normalisierung der Grauwerte zwischen 0 und 1.

Für die Entwicklung von R-CNN-Modellen wurden die Bilder ausgewählter Fehlerklassen zudem neu mit Positionsdaten in einem zweiten Labeling-Vorgang annotiert. Dazu wurde das Open Source Tool *LabelImg*[3] verwendet, um die einzelnen Fehlerobjekte auf den EL-Bildern mit positionsgebenden Rahmen (engl. *Bounding Boxes*) zu kennzeichnen. Das Labeling hat im Durchschnitt circa 15 s pro Bild gedauert. Teilweise wurde auch auf ein semi-automatisiertes Annotieren (engl. *Semi-Automated Labeling*) zurückgegriffen, bei dem nur noch die erzeugten Labels eines vortrainierten Modells durch einen Menschen überprüft werden mussten. Die durch-

[1] OpenCV ist ein freie Bibliothek mit verschiedenen Methoden für Computer-Vision-Anwendungen. Mehr Informationen sind auf der Projektseite zu finden (https://opencv.org/).

[2] Keras ist eine freie Python-Bibliothek, die eine einheitliche Schnittstelle für verschiedene Deep Learning Backends anbietet, wie z. B. für TensorFlow, Microsoft Cognitive Toolkit und Theano (https://keras.io).

[3] LabelImg ist ein freies Tool für das Annotieren von Bildern (https://github.com/tzutalin/labelImg).

schnittliche Annotationsdauer konnte damit um mehr als 50 % auf mittlere Labeling-Zeiten zwischen 5 und 7 s reduziert werden. Die Nutzung der *TensorFlow*[4] *Object Detection API* machte darüber hinaus die Umwandlung der Bild- und Annotationsdaten in ein spezielles, integriertes Dateiformat erforderlich.

Für das Training der Mask-R-CNN-Modelle war es demgegenüber erforderlich, anstelle der groben Bounding Boxen die genauen Umrisse eines Fehlerobjektes in Form von Polygonen zu markieren. Das Labeling dieser Masken ist sehr zeitaufwendig, da die genaue Kontur des Fehlers manuell definiert werden muss. Pro Aufnahme wurden je nach Komplexität des Fehlerbildes durchschnittlich circa 45 s benötigt.

5.2 Statistische Modelle

Die Fehlerklassen *lowlevel* und *waferonpin* betreffen jeweils große Bereiche einer Solarzelle. Damit sind bereits schnell Unterschiede in aggregierten Statistiken der Bildgrauwerte im Vergleich zu funktionstüchtigen Zellen feststellbar. Für die Identifizierung von *lowlevel-Wafern* wurde mit der mittleren Helligkeit eine Kenngröße entwickelt, die den Durchschnitt aller Pixelgrauwerte eines Zellbildes angibt. Nach einer Optimierung auf 420 Fehlerbildern und 1258 guten Zellen lassen sich die beiden Klassen anhand eines Grauwertes von 139 mit einer Accuracy von 99,94 % fast vollständig separieren. Funktionstüchtige Solarzellen haben dabei eine fast vierfach höhere mittlere Helligkeit als Fehlerbilder der Klasse *lowlevel*. Allerdings existiert eine Überschneidung mit anderen Fehlerklassen, die aufgrund ihrer Fehlerbildes ebenfalls geringe Helligkeitswerte aufweisen. Dies führt bei einem Test auf allen Daten zu einer etwas niedrigeren Accuracy (96,81 %). Demnach ist eine Kombination mit weiteren fehlertypspezifischen Modellen erforderlich, um eine korrekte Fehlerzuordnung zu bestimmen.

Bei Fehlern des Typs *waferonpin* handelt es sich um Zellen, die während des Säurebads auf einem Bolzen der Ablage lagen. Dadurch zeichnet sich ein von einer Ecke ausgehender Gradient im Helligkeitsverlauf ab. Die 256 Bilder der Klasse enthalten 274 solcher Ecken, mit denen sich der typische Helligkeitsverlauf einer fehlerhaften

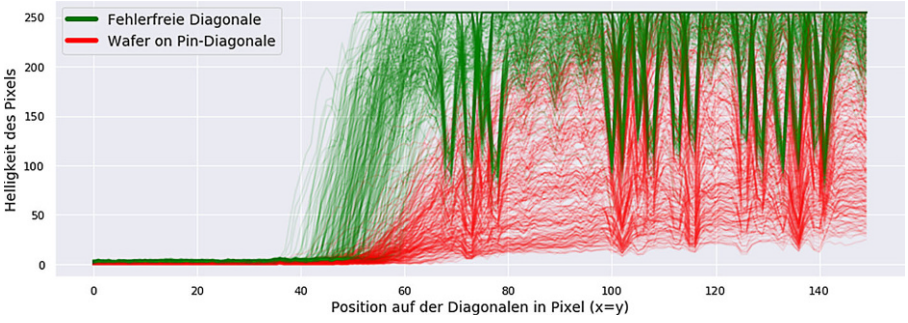

Abb. 4 Vergleich der Diagonalengrauwerte bei guten Wafern und Wafer on Pin

[4] TensorFlow ist eine freie Bibliothek zur datenstromorientierten Programmierung, die häufig zur Entwicklung von neuronalen Netzen bei Rechenoperationen auf mehrdimensionalen Datenfeldern verwendet wird (https://www.tensorflow.org/).

Zelle anhand der Werte der Bilddiagonalen modellieren lässt. Gleichzeitig existieren mit den guten Zellen 5032 Wafer-Viertel, an denen geprüft werden kann, dass keine Bilder falsch klassifiziert werden. Zur Klassifizierung werden die Grauwerte der Diagonalen ausgelesen und anschließend mit einer vorher berechneten Grenzkurve verglichen, welche genau zwischen den Diagonalwerten beider Klassen liegt (siehe Abb. 4). Dabei wird geprüft, wie viele der Grauwerte unterhalb bzw. oberhalb der Grenzkurve liegen, um einer Klasse zugeordnet zu werden. Die teils starken Einbrüche in den hinteren Pixelregionen lassen sich durch die durchscheinenden Förderbänder der Produktionsanlage erklären. Nach einer Optimierung der Grenzkurve auf den gegebenen Wafer-Vierteln kann das Modell mit einer Accuracy von 99,46 % *waferonpin*-Wafer von guten Exemplaren unterscheiden.

5.3 Convolutional Neural Networks

Zur Erkennung der übrigen sechs Fehlerklassen wird mithilfe von einfachen CNN-Architekturen der erste Deep-Learning-Ansatz auf Basis der Annotationen von Labeling-Vorgang I realisiert. Zum Vergleich werden sowohl individuell trainierte Modelle betrachtet als auch Varianten auf Basis aller Fehlerklassen. Für das Modelltraining erfolgt jeweils ein Split der Datenbasis in 75 % Trainings-, 15 % Validierungs- und 10 % Testdaten, um die Stabilität der Ergebnisse an Out-of-Sample-Daten zu überprüfen (Raykar and Saha 2015). Zudem werden die Modelle mit der gleichen Anzahl an fehlerhaften wie guten Wafer-Bildern trainiert, um einen Bias durch unausgeglichene Klassen zu vermeiden. Bei der Konfiguration der Netzwerkarchitekturen werden verschiedene Hyperparameter stufenweise variiert (z. B. Anzahl und Art der Layer, Filtergrößen, Batchsize, Steps, Lernrate). Weiterhin kommen verschiedene Data-Augmentation-Techniken zum Einsatz (z. B. Bildrotation, horizontale/vertikale Spiegelung), um insbesondere im Fall der Klasse *splinter* die Anzahl an Trainingsdaten zu erhöhen. Als Bewertungsgrundlage zur Modelleinschätzung werden die Metriken *Accuracy*[5], *Recall*[6] und *Precision*[7] für alle drei Datenpartitionen herangezogen.

Die Ergebnisse zeigen, dass CNN-Modelle, die gleichzeitig auf allen Fehlerklassen trainiert werden, nicht über Accuracy-Werte von 93 % auf den Validierungsdaten kommen. Die fehlerspezifischen Modelle erzielen demgegenüber zum Teil weitaus bessere Ergebnisse. Tab. 2 listet die besten CNN-Modelle für die Betrachtung der einzelnen Fehlerklassen auf. Bis auf das Modell für die schwer zu erkennende Fehlerklasse *patch* weisen alle übrigen CNN-Modelle hohe Accuracy-Werte von über 90 % auf den Validierungs- und Testdaten auf. Insbesondere die beiden Modelle für die Klassen *splinter* und *crack* heben sich in ihrer Güte von den restlichen Modellen über sämtliche Performance-Metriken ab. Die Modelle für die Klassen

[5] Accuracy gibt den Anteil aller Objekte (sprich Fehlerobjekte und Nicht-Fehlerobjekte) an, die korrekt klassifiziert werden.

[6] Recall gibt den Anteil der korrekt als Fehler klassifizierten Objekte an der Gesamtheit der tatsächlichen Fehlerobjekte an.

[7] Precision gibt den Anteil der korrekt als Fehler klassifizierten Objekte an der Gesamtheit der als Fehler klassifizierten Objekte an.

Tab. 2 Ergebnisse der besten CNN-Modelle je Fehlerklasse (Angaben in Prozent)

	Modell	Accuracy			Precision			Recall		
		Train	Val	Test	Train	Val	Test	Train	Val	Test
1	*cnn_splinter*	100	100	100	100	100	100	100	100	100
2	*cnn_crack*	98,84	99,42	100	97,95	98,75	100	99,76	100	100
3	*cnn_stain*	97,97	98,87	96,64	97,16	97,4	98,25	98,84	100	95,16
4	*cnn_scratch*	100	92,95	94,78	100	96,19	100	100	90,16	89,66
5	*cnn_circle*	96,39	95,24	90,14	99,63	98,08	96,88	93,05	92,45	84,62
6	*cnn_patch*	97,85	88,17	86,1	97,68	86,9	82,29	98,02	89,55	90,11

stain und *scratch* weisen wiederum hohe Precision-Werte auf, während die Recall-Werte etwas abfallen, insbesondere für die Klasse *scratch*. Demnach klassifizieren die Modelle fehlerhafte Wafer öfter als fehlerfrei, als dass der umgekehrte Fall eintritt. Somit kommt es zwar zu weniger Fehlalarmen, jedoch werden einige Fehlerausprägungen nicht erkannt, was aufgrund der geforderten Risikominimierung im vorliegenden Anwendungskontext nicht mehr im akzeptablen Toleranzbereich liegt. Auch für die Fehlerklassen *circle* und *patch* ist der hier realisierte Erkennungsansatz aufgrund inakzeptabler Gütemaße nicht mehr geeignet, weshalb im weiteren Verlauf noch weitere Architekturvarianten verprobt werden.

5.4 Region-based Convolutional Neural Networks

Wie zuvor beschrieben sind R-CNN in der Lage, neben der Klassifizierung von Bildern auch eine Lokalisierung von Objekten auf Basis von Bounding Boxen durchzuführen. Die Motivation zur Nutzung dieser Architekturvariante besteht darin, durch die Fehlerregion weitere Informationen zu erhalten (z. B. Größe und Häufigkeit) und dadurch die Nachvollziehbarkeit für den Anwender zu erhöhen. Weiterhin besteht auch die Vermutung, dass ein R-CNN eine bessere Leistung bei einem Multi-Klassen-Problem erbringt als ein vergleichbares CNN, da in den Trainingsdaten die genaue Position des Fehlers übergeben wird.

Zur Realisierung wird auf eine Faster-R-CNN-Architektur zurückgegriffen. Die Implementierung erfolgt mittels *TensorFlow Object Detection API*. Dabei handelt es sich um eine Open-Source-Bibliothek für die Entwicklung von Modellen zur Objekterkennung, welche auch die Nutzung von Transfer Learning ermöglicht. Transfer Learning nutzt bereits vorhandene Modelle, welche zumeist auf mehreren hunderttausend Bildern trainiert wurden, um die Gewichte des neuen Modells zu initiieren. Dadurch ist es bereits mit wenigen Bildern und relativ kurzem Training möglich, gute Ergebnisse zu erzielen (Oquab et al. 2014). TensorFlow bietet darüber hinaus eine Auswahl von den in Wissenschaft und Praxis verbreiteten Modellen zur direkten Integration in den Trainingsprozess an. Diese werden im sogenannten „Modell Zoo“ mit Metriken zur Genauigkeit sowie zur Geschwindigkeit in Millisekunden pro Inferenz gelistet. Für den vorliegenden Anwendungsfall wurde eine Auswahl von

COCO-Modellen[8] getroffen. Darunter befinden sich neben einem Faster-R-CNN-Modell mit Inception Layer auch ein *SSD-Modell* und eine *MobileNet*[9], welche allgemein eine geringere Inferenzzeit aufweisen und damit für den Anwendungsfall als relevant erscheinen.

Für das Modelltraining wurden initial nur fehlerhafte Wafer verwendet, später jedoch auch gute Wafer mit leeren Annotationen berücksichtigt, um die Stabilität zu überprüfen. Weiterhin wurde die vorherige Split-Strategie beibehalten (75:15:10) und es wurde eine Reihe von Modellparametern angepasst und ausgetestet. Dazu gehören neben der Anzahl der gleichzeitig zu erkennenden Fehlerklassen unter anderem der Feature-Extractor-Typ, die Lernrate, die Batchsize sowie die maximale Skalierung der Input-Bilder. Letzteres hat großen Einfluss auf die Größe und Geschwindigkeit des Modells (Huang et al. 2017). Die Modelle vom Typ *MobileNet* legen hierbei eine maximale Bildgröße von 300 × 300 Pixel fest und sind damit deutlich schneller aber gleichzeitig auch ungenauer als die klassischen Modelle für Bilder bis 1024 Pixel. Zudem wurden erneut verschiedene Data-Augmentation-Techniken angewendet, um die Trainingsbilder durch verschiedene Transformationen künstlich zu verzerren und die Variantenvielfalt zu erhöhen. Das Training der Modelle erfolgte an dieser Stelle aufgrund der hohen Ressourcenanforderungen auf speziellen virtuellen Maschinen eines Cloud-Anbieters. Dabei standen 56 GB Arbeitsspeicher und eine Nvidia K80 Grafikkarte mit 24 GB GDDR5-Speicher zur Verfügung, was zu einer Trainingsdauer jeweils zwischen 12 und 24 h führte. Waren keine Fortschritte in den Metriken *Accuracy* bzw. *Total Loss* mehr auszumachen, wurde das Training vorzeitig abgebrochen.

Die Bewertung dieser Objekterkennungsmodelle unterscheidet sich von den vorherigen CNNs in der Form, dass neben der korrekten Klassifizierung auch die Position des Fehlers berücksichtigt wird. In der Forschung wird dafür eine kombinierte Metrik aus *Mean Average Precision (mAP)* und *Intersection over Union (IoU)* eingesetzt. Die mAP ist der Mittelwert der Average Precision aller Klassen. Die Average Precision berechnet die durchschnittliche Precision für verschiedene Recall-Werte zwischen 0 und 1, also die Fläche unter der Precision-Recall-Kurve (Padilla 2019). Der IoU beschreibt, wie genau die berechnete Bounding Box zur tatsächlichen Boun-

Tab. 3 Ausgewählte Ergebnisse verschiedener R-CNN-Architekturen und deren Parameter

	Modell	Anzahl Fehlerklassen	Batchsize	Data Augmentation	Steps	mAP (IoU = 50 %)
1	*faster_rcnn_inception_v2*	5	20	Ja	35 k	78,1
2	*faster_rcnn_inception_v2*	1	1	Nein	69 k	77,7
3	*faster_rcnn_inception_v2*	4	1	Nein	200 k	73,9
4	*ssd_mobilenet_fpn_460*	3	1	Nein	24 k	70,2
5	*ssd_inception_v2*	3	1	Nein	6 k	66,1

[8] Dabei handelt es sich um Modelle, die auf dem COCO-Datensatz (https://cocodataset.org) trainiert wurden.

[9] SSD-Modelle (*Single Shot Detection*) klassifizieren Bilder in einem Schritt, ohne zuvor Regionen zu identifizieren. MobileNets sind speziell für den Einsatz auf Smartphones entwickelt und damit besonders klein und schnell (Huang et al. 2017).

ding Box (engl. *Ground Truth*) positioniert ist. Dafür wird die Überschneidungsfläche der beiden Rechtecke durch die kombinierte Fläche geteilt. Der IoU-Grenzwert gibt anschließend an, ab wann eine Prediction als True Positive gilt. In der Literatur sind die Grenzwerte 50 % und 75 % verbreitet (Everingham et al. 2010). Für den vorliegenden Fall wird die etwas weniger strenge Definition von 50 % verwendet, da einer exakten Positionierung keine hohe Priorität im Anwendungskontext zukommt. Wichtiger ist, dass ein Modell insgesamt die vorhandenen Fehler einer Zelle korrekt erkennt und klassifiziert. Zur Berechnung dieser Metrik wird die Implementierung der *COCO Detection Evaluation Metrics* verwendet.

Auf Basis der zuvor beschriebenen Konfigurationsmöglichkeiten wurde eine Vielzahl an Modellen verprobt, um die Accuracy bzw. die Geschwindigkeit stetig zu verbessern. Aus Gründen der Übersichtlichkeit wird an dieser Stelle jedoch nur auf eine Auswahl an Konfigurationen und Ergebnissen eingegangen (siehe Tab. 3).

Die Ergebnisse bestätigen die Vermutung, dass sowohl die SSD-Architektur als auch das MobileNet zwar niedrigere Inferenzzeiten aufweisen (ca. 5–6 s vs. 7–9 s für Faster-R-CNNs), aber gleichzeitig auch deutlich schlechtere mAP-Werte erreichen. Mit Modell 2 wurde ein binäres Modell mit einer generischen „fault"-Klasse für alle Fehlerklassen trainiert und untersucht, ob die Fehlerklassen möglicherweise untereinander schwer zu unterscheiden sind. Der Gewinn an Präzision (4 % im Vergleich zu Modell 3) wird allerdings durch den Verlust an Nützlichkeit relativiert, da hierdurch keine genaue Identifikation der Fehlerart möglich ist. Der mAP-Wert von 78,1 des finalen Modells (1) konnte schließlich über die Erhöhung der Batchsize und den Einsatz von Data-Augmentation-Techniken erreicht werden. Dabei erkennt das Modell alle fünf Fehlerklassen mittlerer Komplexität (d. h. *crack, scratch, circle, splinter* und *stain*). Um die erreichte Genauigkeit in Kontext zu den vorherigen CNN-Modellen zu setzen, wurde eine manuelle Berechnung mit 165 Testbildern (ca. 1/3 gute und 2/3 fehlerhafte Wafer mit beliebigen Fehlern) durchgeführt. Nach Anwendung des Modells wurden hierfür die Pseudo-Metriken für eine rein binäre Klassifizierung ermittelt („Wurde eine Zelle korrekt als insgesamt fehlerhaft oder gut klassifiziert?"). Dabei konnte eine für die Qualitätssicherung akzeptable Accuracy von 98,78 % erreicht werden. Da durch dieses Modell jedoch noch nicht die schwer zu erkennende Klasse *patch* adressiert werden konnte, wurde im nächsten Schritt eine noch leistungsfähigere CNN-Architekturvariante verprobt.

5.5 Mask R-CNN

Während bereits sieben von acht Fehlerklassen mit den bisherigen Modellen gut erkannt werden können, stellt die letzte Klasse *patch* eine besondere Herausforderung dar. So weisen deren Fehlerbilder eine hohe Varianz in Form und Größe auf, wodurch sie selbst mit dem menschlichen Auge nur schwer vom restlichen Bild abzugrenzen sind. Die bisherigen Modelle stoßen hierbei an ihre Grenzen, da sie den Fehler nicht adäquat repräsentieren können. Fehlerfreie Bildabschnitte durchbrechen häufig das Muster des Fehlerbildes. Eine Bounding Box kann diesen Sachverhalt nicht darstellen, da sie nur eine äußere Umrahmung des Fehlers zulässt. Dadurch hat es den Anschein, der Fehler erstrecke sich über einen deutlich größeren Bereich, als es tatsächlich der Fall ist (siehe Abb. 5). Diese Eigenschaften erfordern den Einsatz

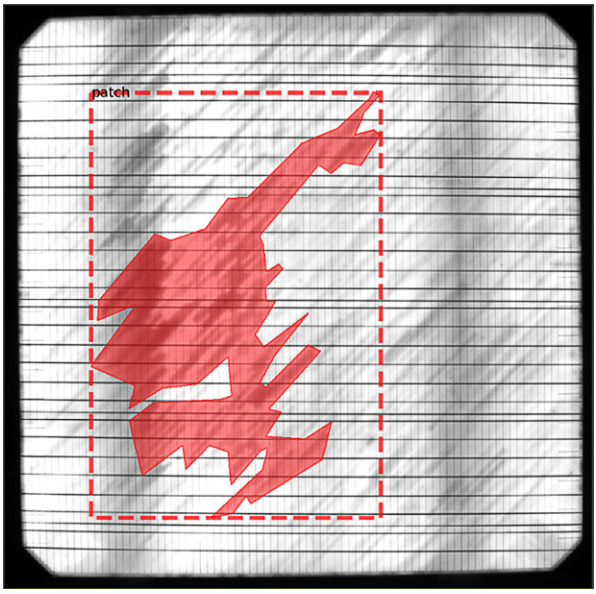

Abb. 5 Annotation der Fehlerklasse „patch“ mittels Bounding Box (Rechteck) vs. pixelgenaue Segmentierung (Polygon)

von Modellen, die eine pixel- bzw. umrissgenaue Segmentierung zulassen. Nur so kann im Nachhinein das genaue Fehlerausmaß bei der Bewertung sich überlappender Flächen nachvollzogen werden.

Zur Adressierung dieser Problemstellung wird der Einsatz einer Mask-R-CNN-Architektur verprobt. Sie wird jedoch ausschließlich zur Erkennung der Fehlerklasse *patch* implementiert, da die Grenzen der übrigen Fehlerklassen klar erkennbar sind und folglich die Kontur der Fehler nur eine geringe Rolle spielt.

Für die Implementierung wird die auf Github frei verfügbare *Matterport-Bibliothek*[10] verwendet und auf den spezifischen Wafer-Datensatz angepasst. Bei der Konfiguration der Modellarchitektur werden erneut verschiedene Parameter getestet und stufenweise angepasst. Wesentliche Unterschiede lassen sich insbesondere bei der Anzahl an Schritten feststellen, die beim Trainieren verschiedener Architekturbereiche angesetzt werden. In Analogie zum vorherigen Ansatz wird auch bei den Mask R-CNNs mittels Transfer Learning auf ein vortrainiertes Modell zurückgegriffen, wobei erneut die Initialgewichte des COCO-Datensatzes zum Einsatz kommen. Das Training erstreckt sich dabei über zwei Durchläufe mit einer festzusetzenden Anzahl an Trainingsschritten (*Steps:Heads, Steps:ResNet3+*). Diese beinhalten zunächst nur das Training des Region Proposal Networks und der Klassifizierungsschichten, während im zweiten Durchlauf Feinjustierungen einzelner Layer eines vortrainierten ResNet-Modells vorgenommen werden.

Als Datenbasis stehen insgesamt 932 Fehlerbilder der Klasse *patch* zur Verfügung. Aufgrund des aufwendigen Labelns der Masken (siehe Abschn. 5.1) wurde der Umfang jedoch auf 202 Fehlerbilder beschränkt. Zusätzlich wurden die Fehler-

[10] https://github.com/matterport/Mask_RCNN.

Tab. 4 Ausgewählte Ergebnisse verschiedener Mask-R-CNN-Architekturen und deren Parameter

	Modell	Gemischtes Training	Batch-size	Data Augmen-tation	Steps: Heads	Steps: Res-Net3+	mAP (IoU = 50 %)	Pseudo Accura-cy
1	*mrcnn_patchonly*	Nein	2	Ja	1,5 k	1,5 k	72,01	54,05
2	*mrcnn_mixed*	Ja	2	Ja	12 k	0	25,62	96,72
3	*mrcnn_mixed*	Ja	2	Ja	12 k	40,4 k	27,45	81,15
4	*mrcnn_mixed*	Ja	2	Ja	12 k	48,5 k	26,92	81,97
5	*mrcnn_mixed*	Ja	2	Ja	23,8 k	0	49,57	95,90

bilder einschließlich ihrer Annotationen gespiegelt und es fand eine Aufteilung in Trainings- und Validierungsdaten im Verhältnis 85:15 statt, um im Rahmen des Modelltrainings auf mehr Daten zugreifen zu können. Auf dieser Basis konnte ein erstes Mask-R-CNN entwickelt werden (Modell 1), das sich insgesamt mit einem mAP-Wert von 72,01 % in die Ergebnisse der vorherigen Faster-R-CNN-Modelle einreiht. Bei der Berechnung der Pseudo-Accuracy anhand von Out-of-Sample-Daten, die sich zu gleichen Teilen aus fehlerhaften und fehlerfreien Wafer-Bildern zusammensetzten, konnte jedoch nur eine unzureichende Güte von 54,05 % erreicht werden (siehe Tab. 4). Aus diesem Grund wurden in einem zweiten Schritt zusätzlich 404 einwandfreie EL-Bilder der Klasse *flawless* einem gemischten Trainingsprozess hinzugefügt. Dem neuronalen Netz wird damit die Aufgabe gestellt, relevante Fehlerstrukturen von irrelevantem Rauschen zu unterscheiden und damit einem Overfitting vorzubeugen. Im Ergebnis zeigten sich bei diesem Ansatz solide Accuracy-Werte bis zu 96,72 %, was eine wesentliche Steigerung im Vergleich zur einfachen CNN-Architektur darstellt (86,1 %). Gleichzeitig können jedoch aufgrund der gemischten Trainingsgrundlage wesentliche Einbrüche der mAP-Werte festgestellt werden. Diese resultieren aus den teils starken Abweichungen zwischen den vorhergesagten und den tatsächlichen Fehlerstrukturen, die sich zwar nicht in der Bewertung der Objektklassifizierung widerspiegeln, allerdings bei deren Lokalisierung ins Gewicht fallen. Durch längeres Modelltraining (Anzahl Steps) in den verschiedenen Architekturschichten ließ sich der Effekt zum Teil etwas eindämmen. Aufgrund der Komplexität des Fehlerbildes war es jedoch nicht möglich, weitaus höhere mAP-Werte jenseits von 50 % zu erreichen.

6 Evaluation

Abschließend werden die besten Modelle der einzelnen Modelltypen einer zusammenfassenden Evaluation unterworfen. Dabei gilt es ihre industrielle Eignung gemäß den vorgegebenen Anforderungen zu prüfen. Dies umfasst insbesondere eine maximale Fehlertoleranz von 2 % sowie eine maximale Inferenzzeit von 2 s. Tab. 5 liefert eine Zusammenfassung über die jeweiligen Modellausprägungen.

Die statistischen Modelle erkennen die einfachen Fehlerklassen *lowlevel* und *waferonpin* zu 99,94 % bzw. 99,42 %. Da sie auf einer großen Datengrundlage über einen langen Produktionszeitraum basieren, können sie auch gut auf andere Wafer-Bilder übertragen werden und qualifizieren sich für den Einsatz in der Produktion.

Tab. 5 Zusammenfassung über Modellgenauigkeiten und Inferenzzeiten

Modell	Accuracy	Precision	Recall	mAP (IoU = 50 %)	Inferenzzeit Lokal (CPU) (in s)	Inferenzzeit Online (GPU) (in s)
stat_lowlevel	99,94	100	99,76	–	0,2	–
stat_waferonpin	99,42	95,91	94,16	–	0,2	–
cnn_splinter	100	100	100	–	0,2	–
cnn_crack	100	100	100	–	0,2	–
faster_rcnn_inception_v2 (*crack, scratch, circle, splinter, stain*)	98,78	99,11	99,12	78,1	7	1,8
mrcnn_mixed (*patch*)	95,90	92,06	100	49,57	8,5	0,8

Demgegenüber müssen sämtliche Änderungen im Herstellungs- als auch im Bildaufnahmeprozess beobachtet werden, da sie einen Einfluss auf die Helligkeit der Wafer-Bilder haben könnten, wodurch eine Anpassung der Grenzwerte erforderlich wird.

Die genauesten Modelle unter den CNNs sind *cnn_splinter* und *cnn_crack*. Beide Modelle erkennen sämtliche Fehler auf den Out-of-Sample-Daten. Während sich hierbei der Einsatz des *crack*-Modells für den Produktionsbetrieb aufgrund der angemessenen Datenbasis bereits als zuverlässig erweist, muss das *splinter*-Modell noch weiteren Untersuchungen unterzogen werden. Hierbei standen ursprünglich nur 79 (bzw. nach entsprechender Spiegelung 158) Wafer-Bilder zur Verfügung, weshalb die Ergebnisse auf zusätzlichen Fehlerbildern zu verproben sind.

Bei den R-CNN-Modellen liefert das Modell *faster_rcnn_inception_v2* eine mAP von 78,1 %. Durch die Berechnung einer „Pseudo-Accuracy“ für binäre Klassifizierung lässt sich das Modell auch mit den statistischen Modellen und einfachen CNNs vergleichen. Hier erreicht das Modell in allen drei Metriken Werte von über 98,78 %, obwohl es nicht nur für einen spezifischen Fehlertyp, sondern insgesamt fünf verschiedene Fehlerklassen trainiert wurde. Damit zeigt das Modell eine hohe Praxistauglichkeit, da es gleichzeitig für mehrere Erkennungsprobleme eingesetzt werden kann und eine hohe Güte aufweist. Darüber hinaus bietet das Modell den Vorteil, dass es eine genaue Lokalisierung der Fehler zulässt, worauf zukünftig weitere Analysen aufbauen können (z. B. Bestimmung hoch frequentierter Zellenbereiche für speziellen Fehlertypen).

Die entwickelten Mask-R-CNN-Modelle für die Klasse *patch* weisen demgegenüber noch einige Schwächen auf. Zwar konnten beim Blick auf die ermittelte Pseudo-Accuracy mithilfe der umrissgenauen Fehlereingrenzung wesentliche Performance-Steigerungen im Vergleich zur einfachen CNN-Architektur gewonnen werden, die maximale Fehlertoleranz von 2 % ließ sich jedoch nicht realisieren. Zudem zeigen die bisher entwickelten Modelle schlechte mAP-Werte von unter 50 %, was darauf hindeutet, dass die Netze starke Probleme haben, die charakteristischen Fehlerstrukturen korrekt zu erkennen bzw. eindeutig abzugrenzen. Da dies allerdings auch für das ungeschulte menschliche Auge nur schwer zu realisieren ist, sind laut Fachexperten derartige Genauigkeitswerte durchaus verkraftbar. Bevor das Modell

im Produktionsbetrieb zum Einsatz kommt, sollen noch weitere Modellparameter verprobt werden, die zur Steigerung der Genauigkeit beitragen könnten.

Die Bewertung der Inferenzzeiten erfolgte auf Basis eines lokalen Consumer-Notebook mittels CPU sowie einer virtuellen Maschine eines Cloud-Anbieters unter Nutzung von GPUs (siehe Abschn. 5.3). Aufgrund der einfachen Berechnungen bieten die statistischen Modelle mit ca. 0,2 s pro Bild die schnellsten Klassifizierungsansätze. Jedoch ordnen sich auch die Inferenzzeiten der einfachen CNN-Modelle in diesem Größenbereich ein, weshalb sich beide Varianten ohne größere Restriktionen für den Produktiveinsatz eignen. Es können sogar bis zu 10 dieser Modelle in Reihe geschaltet werden, ohne die geforderten 2 s zu überschreiten. Somit ist bei der Anwendung der einfachen CNNs auch der Einsatz fehlerklassenspezifischer Modelle anstelle eines fehlerübergreifenden Modells ohne Parallelisierung plausibel. Da die Ergebnisse der fehlerklassenspezifischen CNNs besser sind als die eines Gesamtmodells, kann mit der Inkaufnahme einer langsameren Inferenz pro Bild (0,2 s × Modellanzahl) besser klassifiziert werden. Die R-CNN-Modelle überschreiten indes deutlich den Zeitrahmen auf der lokalen CPU. Mithilfe einer GPU, mit ihren parallelen Architekturen und expliziten Eigenschaft für hohe Datendurchsätze, ist es jedoch möglich, die Inferenzzeiten beider Modelle zu reduzieren. Dadurch können auch die beiden leistungsfähigsten Modelle bei mittleren bzw. komplexen Fehlerbildern mit einer geeigneten Hardware im Produktionsbetrieb angewendet werden.

7 Fazit und Ausblick

Das Gebiet um Computer Vision ist mittlerweile ein ausgereiftes Forschungsfeld, in dem sich zahlreiche Studien mit der Entwicklung und der Evaluation von Methoden, Modellen und Systemen beschäftigen. Häufig werden dabei fortwährend neue Architekturvarianten von neuronalen Netzen entwickelt und auf Basis breit angelegter Benchmark-Datensätze hinsichtlich ihrer Genauigkeit, ihrer Geschwindigkeit oder etwa ihrer Komplexität verprobt und miteinander verglichen (Voulodimos et al. 2018; Heinrich et al. 2019a; Zhao et al. 2019). Demgegenüber fehlt es jedoch mitunter an Reflektionen aus der industriellen Praxis, wie sich solche Ansätze tatsächlich unter realen Bedingungen anwenden lassen und verhalten. Hierzu liefert der vorliegende Beitrag Einblicke in eine Fallstudie zur automatisierten Qualitätssicherung bei der Fertigung von Solarzellen unter Verwendung unterschiedlich komplexer Modelle auf Basis von Open Source Software.

Die Ergebnisse zeigen, dass nicht für jeden Anwendungsfall die fortgeschrittensten Deep-Learning-Architekturen erforderlich sind, da bereits mit statistischen Modellen und einfachen CNNs zuverlässige Aussagen mit Genauigkeiten von über 99 % bei Fehlertypen einfacher bis mittlerer Erkennbarkeit erreicht werden können. Werden die Fehlerbilder demgegenüber diffuser und soll die Nachvollziehbarkeit der Ergebnisse bspw. durch positionsgenaue Lokalisierung von Fehlerobjekten gewährleistet werden, sind schon fortgeschrittenere Ansätze erforderlich, die beispielsweise auf den vorgestellten Region-Proposal-Ansätzen basieren. Allerdings gehen solche Modelle auch mit einem beträchtlich höheren Aufwand für Vorverarbeitungsschritte und die Annotation der Fehlerausprägungen einher. Dies gilt insbesondere für

das umrissgenaue Labeling am Beispiel der Mask-R-CNN-Architektur, weshalb im vorliegenden Fall der Einsatz auf die am schwierigsten zu erkennende Fehlerklasse beschränkt wurde. Es muss daher abgewogen werden, inwiefern die Schätzung einer genauen Fehlerkontur für einzelne Klassen einen Mehrwert gegenüber einer groben Umrahmung generieren kann.

Bezüglich der Anwendungstauglichkeit spielen neben der Genauigkeit noch weitere Aspekte eine Rolle. So ist bei den statistischen Modellen aufgrund der einfachen Schwellwertbetrachtungen eine Anpassung an veränderte Produktionsumgebungen schnell möglich. Zudem können die Modelle durch ihre Geschwindigkeit und ihren geringen Speicher- und Rechenbedarf auf einem einfachen lokalen System sowie online ohne zeitaufwendige Einrichtung angewendet werden. Die CNN-Modelle sind demgegenüber etwas unflexibler. Zwar benötigen auch sie wenig Speicherplatz und Rechenressourcen und sind zudem schnell, doch Änderungen erfordern es, das gesamte Modell neu zu trainieren. Die Dauer des Neutrainings ist jedoch vertretbar kurz, womit die CNNs immer noch eine gute Anpassbarkeit aufweisen. Die am wenigsten flexiblen Modelle sind auch gleichzeitig die komplexesten und größten. So steigt der Bedarf an Speicher- sowie Rechenressourcen bei den vorgestellten R-CNNs wesentlich an. Auch verlangsamt die erhöhte Modelltiefe die Inferenz, wodurch der Einsatz auf einer CPU im Betrieb aufgrund der langen Inferenzzeiten ausgeschlossen werden kann. Schließlich bedarf es wie bei den CNNs eines erneuten Trainings bei Veränderungen, auch wenn auf den vorherigen Modellen aufgebaut werden kann. Im Vergleich zu den deutlich kleineren CNN-Modellen nimmt das Training der R-CNNs aber eine längere Zeit in Anspruch. Auch der Aufwand des Labelings neuer Bilddaten, sowie komplexer Strukturen, sollte dabei nicht unterschätzt werden.

Darüber hinaus erfordern Modellarchitekturen mit zunehmender Komplexität und Tiefe auch eine adäquate Auswahl und Konfiguration der zugrundeliegenden Modellparameter. Während in der vorliegenden Arbeit einzelne Parameter stufenweise verprobt und angepasst wurden, ist eine weitaus feingliedrigere Hyperparameteroptimierung denkbar, um einerseits gezielt die Auswirkungen einzelner Parameter zu bestimmen und andererseits die Modellergebnisse noch weiter zu verbessern. Dies erfordert in der Regel jedoch viel Zeit, Rechenressourcen und technisches Vorwissen, weshalb in der Praxis die Erstellung komplexerer Modelle vorzugsweise gemeinsam mit Data-Science-Experten durchgeführt werden sollte, um den Parameterbereich einzuschränken und Rechenkapazitäten sinnvoll auszunutzen. Weitere Untersuchungsaspekte in diesem Zusammenhang betreffen zum Beispiel die Verwendung weiterer Data-Augmentation-Techniken oder den Einsatz alternativer Transfer-Learning-Modelle aus anderen Datengrundlagen. Die entwickelten Transfer-Learning-Modelle basieren hierbei hauptsächlich auf dem COCO-Datensatz. Dahingehend wäre es interessant zu untersuchen, ob ein Transfer von Modellen aus einem industrieähnlichen Kontext möglicherweise zu besseren Ergebnissen führen könnte. Zudem erfolgte die Bewertung der Modelle bisher ohne eine Priorisierung der einzelnen Fehlerklassen. Demnach ist in Folgearbeiten geplant, die Kritikalität und die Opportunitätskosten der Erkennung einzelner Fehlerklassen mit ins Kalkül aufzunehmen. Darüber lassen sich Aussagen treffen, ab wann sich ein höherer La-

beling- sowie Modellkonfigurationsaufwand lohnen könnte bzw. explizit vermieden werden sollte.

Abschließend gilt festzuhalten, dass das Angebot an frei zugänglicher Open Source Software in den Bereichen Deep Learning und Computer Vision eine große Chance für die betriebliche Unternehmenspraxis darstellt. Mit den exemplarisch aufgezeigten Tools wie Keras, TensorFlow, LabelImg und OpenCV werden mittlerweile umfangreiche und ausgereifte Community-Lösungen angeboten, die es Unternehmen erlauben, ohne große Investitionen und zusätzlichen Programmieraufwand wissensintensive KI-Ressourcen wiederzuverwenden und für die eigenen Geschäftstätigkeiten zu adaptieren. Die aktive Beteiligung von Forschern und Entwicklern aus vielfältigen Disziplinen sorgt zudem dafür, dass neueste Fortschritte kontinuierlich das Feld erweitern und in Form von frei zugänglichen Lösungen für die breite Community zur Verfügung gestellt werden, wie es etwa beim verprobten Mask-R-CNN-Modell der Fall ist. Gleichzeitig birgt die Schnelllebigkeit des Gebietes jedoch die Gefahr, dass verwendete Bibliotheken und Frameworks schnell veralten, was zu Herausforderungen beim Betrieb und der Wartung des Programmcodes führt. Darüber hinaus müssen auch Aspekte wie Nachvollziehbarkeit und Zuverlässigkeit der Lösungsangebote berücksichtigt werden, denn trotz offener Quellsysteme handelt es sich bei Deep-Learning-Modellen meist um Black-Box-Modelle, deren Verhalten nicht vollständig verstanden und nachvollzogen werden kann (Heinrich et al. 2020). Vor derartigen Hürden sollte die Unternehmenspraxis jedoch nicht zurückschrecken, wenn es um die Realisierung vielversprechender Nutzenpotenziale geht. Vielmehr ist es Aufgabe von Forschung und Praxis, an diesen Punkten anzuknüpfen, um die nachhaltige und sichere Anwendbarkeit von Computer-Vision-Technologien für betriebliche Anwendungsfelder sicherzustellen.

Funding Open Access funding provided by Projekt DEAL.

Literatur

Chollet F (2018) Deep learning with Python. Manning, Shelter Island, New York

Everingham M, Van Gool L, Williams CKI et al (2010) The Pascal Visual Object Classes (VOC) challenge. Int J Comput Vis 88:303–338. https://doi.org/10.1007/s11263-009-0275-4

Friederich J, Zschech P (2020) Review and systematization of solutions for 3D object detection. In: Wirtschaftsinformatik 2020 Proceedings

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, S 580–587

Griebel M, Dürr A, Stein N (2019) Applied image recognition: guidelines for using deep learning models in practice. In: Wirtschaftsinformatik 2019 Proceedings

He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. ArXiv151203385 Cs

He K, Gkioxari G, Dollár P, Girshick R (2017) "Mask R-CNN,". in: 2017 IEEE International Conference on Computer Vision (ICCV), October, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

Heinrich K, Janiesch C, Möller B, Zschech P (2019a) Is bigger always better? Lessons learnt from the evolution of deep learning architectures for image classification. In: Pre-ICIS SIGDSA Symposium Munich, Germany

Heinrich K, Roth A, Zschech P (2019b) Everything counts: a taxonomy of deep learning approaches for object counting. In: European Conference on Information Systems Stockholm-Uppsala, Sweden

Heinrich K, Zschech P, Möller B et al (2019c) Objekterkennung im Weinanbau – Eine Fallstudie zur Unterstützung von Winzertätigkeiten mithilfe von Deep Learning. HMD Prax Wirtsch 56:964–985. https://doi.org/10.1365/s40702-019-00514-9

Heinrich K, Graf J, Chen J et al (2020) Fool me once, shame on you, fool me twice, shame on me: a taxonomy of attack and defense patterns for AI security. In: Proceedings of the 28th European Conference on Information Systems (ECIS) Marrakesh, Morocco

Huang J, Rathod V, Sun C et al (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Honolulu, HI, S 3296–3297

Köntges M, Kurtz S, Packard C et al (2014) Performance and reliability of photovoltaic systems: subtask 3.2: Review of failures of photovoltaic modules: IEA PVPS task 13: external final report IEA-PVPS. International Energy Agency, Photovoltaic Power Systems Programme, Sankt Ursen

Kurgan LA, Musilek P (2006) A survey of knowledge discovery and data mining process models. Knowl Eng Rev 21:1–24. https://doi.org/10.1017/S0269888906000737

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444. https://doi.org/10.1038/nature14539

MathWorks (2017) Introduction to deep learning: what are convolutional neural networks? https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html. Zugegriffen: 16. Juni 2020

Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Columbus, OH, USA, S 1717–1724

Padilla R (2019) Object detection metrics release v0.2 https://doi.org/10.5281/zenodo.2554189

Raykar VC, Saha A (2015) Data split strategies for evolving predictive models. In: Appice A, Rodrigues PP, Santos Costa V et al (Hrsg) Machine learning and knowledge discovery in databases. Springer, Cham, S 3–19

Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Szeliski R (2010) Computer vision: algorithms and applications. Springer, Heidelberg, Berlin, New York

Thiel M, Hinckeldeyn J, Kreutzfeldt J (2018) Deep-Learning-Verfahren zur 3D-Objekterkennung in der Logistik. Logist J Proc. https://doi.org/10.2195/lj_Proc_thiel_de_201811_01

Trinks S, Felden C (2019) Smart Factory – Konzeption und Prototyp zum Image Mining und zur Fehlererkennung in der Produktion. HMD Prax Wirtsch 56:1017–1040. https://doi.org/10.1365/s40702-019-00529-2

Tsai D-M, Wu S-C, Chiu W-Y (2013) Defect detection in solar modules using ICA basis images. IEEE Trans Ind Inform 9:122–131. https://doi.org/10.1109/TII.2012.2209663

Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep learning for computer vision: a brief review. Comput Intell Neurosci. https://doi.org/10.1155/2018/7068349

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (Hrsg) Computer vision—ECCV 2014. Springer, Cham, S 818–833

Zhao Z-Q, Zheng P, Xu S-T, Wu X (2019) Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst 30:3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

Zschech P (2018) A taxonomy of recurring data analysis problems in maintenance analytics. In: Proceedings of the 26th European Conference on Information Systems (ECIS) Portsmouth, UK