We Are Developers!

Herbst – 3/2023

Eine Themenbeilage der Heise Medien GmbH & Co. KG

> **KI-BILDGENERIERUNG**

Kreative Bilder mit Generative Adversarial Networks

> **ACCESSIBILITY**

Barrierefreie Websites entwickeln

> **LOSLEGEN MIT RUST**

Speichersicher und performant

> **QUEREINSTIEG IN DIE IT**

Dos und Don'ts beim Jobwechsel

Powered by iX

VOM 6-GELENK-ROBOTERARM ZUM SMART SPRAYING

„Als ich 11 Jahre alt war, kam der humanoide Roboter Asimo in meine Heimatstadt Luxemburg. Dieser Besuch hat Spuren hinterlassen – seitdem schlägt mein Herz für Robotik", erinnert sich Thomas Kinzig. Die Folge: ein Robotik-Studium in Edinburgh, ein Bachelor in Computer Engineering in Berlin und später ein Elektrotechnik-Master in Aachen. Immer lag sein Fokus auf der Entwicklung von Software.

Heute, 17 Jahre nach dem Besuch von Asimo, arbeitet Thomas als Embedded Software-Entwickler bei ITK Engineering. „Ich entwickle intelligente Software, die in der Landwirtschaft hilft, Pflanzen auf einem Feld zu identifizieren, um die richtigen Unkrautvernichter an den richtigen Stellen automatisiert auszubringen". Der Code, den Thomas schreibt, ist für Landwirte entscheidend, um auch künftig Felder effizient und umweltbewusst bewirtschaften zu können.

Neben der Chance, an bahnbrechenden Technologien von Morgen mitzuarbeiten, sieht er viele weitere Vorteile bei ITK Engineering. „Mir gefällt die Kultur hier sehr gut. Das Miteinander ist herzlich wie an der Uni und trotzdem sehr professionell. Jeder duzt jeden, alle arbeiten auf Augenhöhe und Hierarchien findet man nur im Organigramm."

Du willst künftig mit Thomas gemeinsam die digitale Welt von morgen gestalten? Dann lohnt sich ein Blick auf unsere offenen Stellen in den Bereichen Software Engineering, Embedded Systems, Cloud Computing, künstliche Intelligenz oder Cybersecurity. Besuche uns auf **www.itk-engineering.de!**

www.itk-engineering.de

ITK. The Art of Digital Engineering.

EDITORIAL

Von KI bis Quereinstieg

ChatGPT hat im vergangenen Jahr die volle Aufmerksamkeit im Bereich der KI auf sich gezogen. Daneben gab es in der KI-Forschung weitere spannende Entwicklungen, wie die Sprach-Bild-Synthese, bei der eine Künstliche Intelligenz Bilder aus Beschreibungen in natürlicher Sprache generiert. Unser Autor Timo Zander widmet sich in dieser Herbstausgabe von We Are Developers den Generative Adversarial Networks (GANs). Diese Netze erzeugen realistische Bilder, indem ein Generator Bilder erstellt und ein Discriminator zwischen echten und generierten Bildern unterscheidet. KI-generierte Bilder sind inzwischen sehr verbreitet, GANs bringen den Fortschritt immer mehr voran.

Ein weiteres aktuelles Thema dieser Ausgabe ist Accessibility. Im Juli 2021 trat in Deutschland das Barrierefreiheitsstärkungsgesetz (BFSG) in Kraft, das den European Accessibility Act (EAA) umsetzt. Es definiert die Anforderungen an die Barrierefreiheit von Produkten und Dienstleistungen, die nach dem 28. Juni 2025 auf den europäischen Markt kommen oder den Verbrauchern angeboten werden. Was also bis dahin nicht dem EAA entspricht, muss mit Konsequenzen wie erheblichen Bußgeldern rechnen. So weit muss es aber nicht kommen, wenn man sich gut vorbereitet. Maria Korneeva stellt in ihrem Artikel einige Tools und etablierte Techniken vor, die Entwicklerinnen und Entwickler auf ihrem Weg zu einer verbesserten Accessibility in ihren Webanwendungen unterstützen sollen – das Spektrum reicht von kostenlosen Browser-Plug-ins über Werkzeuge zur Testautomatisierung bis hin zur Integration in CI/CD-Pipelines.

Stefan Baumgartner eröffnet Rust-Interessierten mit seinem Artikel einen Einstieg in die immer populärer werdende Programmiersprache. Rust bietet nicht nur eine einzigartige, moderne Syntax für eleganten Code, sondern stellt mit dem Konzept der Traits eine Möglichkeit zur Abstraktion ohne Overhead zur Verfügung – Programme schreiben, ohne Geschwindigkeitsverlust. Darüber hinaus stellt die Programmiersprache über das Ownership-Konzept eine optimierte Speicherverwaltung zur Kompilierzeit sicher.

Blicken wir noch über den Tellerrand: Zunehmend ersetzen Coding-Bootcamps, IT-affine Hobbys und Zusatzausbildungen das klassische Informatikstudium. Ein erfolgreicher Quereinstieg erfordert Qualitäten wie Lernbereitschaft, klare Berufsziele, Teamarbeit und kontinuierliche Weiterentwicklung sowie Flexibilität bei der Standort- und Unternehmenswahl. Erfahrungen aus anderen Bereichen, Softskills und Leidenschaft sind gute Voraussetzungen. Eine Auswahl an Dos und Don'ts beim Quereinstieg in die IT liefert Hallie Barrows, selbst Quereinsteigerin.

Viel Spaß beim Lesen!

We Are Developers I Herbst 2023

INHALT

Young Professionals schreiben für Young Professionals

Unsere Beilage zu c't und iX basiert überwiegend auf einer Online-Artikelserie, die Young Professionals eine Bühne bietet für erste Fachartikel. Die Autoren und Autorinnen erhalten von der Heise-Redaktion Unterstützung beim Konzipieren und Schreiben.

Dein erster Fachartikel bei Heise

Die Serie ermutigt dazu, eigene Erfahrungen mitzuteilen, ein Projekt vorzustellen – oder wolltest du schon immer mal selbst einen Fachartikel schreiben und hast eine Idee?

Schreib uns: developer@heise.de

> Wenn Maschinen Kunst schaffen: KI-generierte Bilder mit GANs

Timo Zander

Moderne KI-Modelle versprechen das Unmögliche: Kreativität. Ob Schreiben von Poesie oder Generieren von Kunst: Dahinter stehen Generative Adversarial Networks.

Die Popularisierung von ChatGPT und die damit zusammenhängende gesellschaftliche Debatte hat einen großen Teilbereich der KI-Forschung überschattet: Nämlich die Sprach-Bild-Synthese, also das Erstellen von KI-generierten Bildern anhand natürlicher Spracheingabe. Angefangen haben solche Modelle mit schwammigen, niedrig aufgelösten Pixelmosaiken – doch Techniken wie DALL-E 2 von OpenAI oder Stable Diffusion von Stability AI beeindrucken durch ihre Echtheit und den Detailgehalt ihrer Ausgabe. Die Möglichkeiten gehen längst über Trivialitäten hinaus.

Nicht nur die Positionierung von Objekten, sondern auch Adjektive oder kulturelle Referenzen lassen sich angeben. Das Training derartiger Modelle erfordert Abermillionen von Bildern, um diese Präzision möglich zu machen. Doch die Ursprünge dessen sind deutlich greifbarer, als es die Komplexität der generierten Bilder vermuten lässt. Eine wichtige Grundlage schafft die KI-Generierung von zufälligen Inhalten. Solche Modelle können keine natürliche Sprache verarbeiten, sondern schlicht real aussehende, zufällige Bilder von Personen, Landschaften, Tieren und Weiterem erstellen. Das theoretische Fundament dahinter schafft das Konzept der sogenannten Generative Adversarial Networks.

In a Nutshell

- GANs (Generative Adversarial Networks) sind KI-Systeme, die kreative Bilder generieren können.
- Sie verwenden einen Wettbewerbsansatz zwischen einem Generator und einem Diskriminator, um hochwertige Bilder zu erstellen – gesteuert von natürlicher Sprache.
- Die Weiterentwicklung der GAN-Technik ermöglicht es den Nutzern, generierte Inhalte zu gestalten.

Adversarial Training: Nicht mit, sondern gegeneinander lernen

Das Ziel von Generative Adversarial Networks ist, ein Modell zu kreieren, das durch seine Struktur zufällige und abwechslungsreiche, aber ebenso sinnvolle Ausgaben erzeugen kann (Abbildung 1). Hierfür verwendete der KI-Forscher Ian Goodfellow Convolutional Neural Networks ("faltende neuronale Netze"). Das erste Netz, der Generator, erhält eine Zufallszahl als Eingabe und erstellt hieraus eine Ausgabe, also etwa ein Bild. Diese Zufallszahl ist ein stochastisches Rauschen, was dafür sorgen soll, dass die Ausgaben stets unterschiedlich und vielfältig sind. Da das eigentliche Ziel eines GAN die Generierung von Bildern ist, ist somit am Ende auch nur das Generator-Netz von Bedeutung.

Der Gegenspieler zu diesem Netz ist der Discriminator. Sein Zweck ist es, eine Eingabe (Bild) entgegenzunehmen und zu beurteilen, ob es real oder künstlich generiert ist. In der Praxis wird das Discriminator-Netz anhand bestimmter Objekte – Menschen, Wälder, Tiere oder Ähnliches – trainiert, sodass es eine Wahrscheinlichkeit ausgibt, mit der das Eingabebild echt ist. Für das Training ist das Netz daher unabdingbar, auch wenn es schlussendlich an Bedeutung verliert: Ist der Generator ausreichend trainiert, bedarf es keiner Analyse dessen Ausgaben mehr.

Das Training des gesamten Modells basiert auf der Idee des adversarialen (antagonistischen) Trainings. Statt also miteinander zu lernen, treten beide neuronalen Netze gegeneinander an und versuchen, in jedem Schritt über die Stärken

und Schwächen des Gegners hinauszuwachsen. Der Generator erstellt eine Ausgabe, die der Discriminator dann untersucht und bewertet. Ausgehend davon versucht der Generator im nächsten Schritt, die Ausgabe noch täuschend echter zu gestalten. Das initiale Problem ist jedoch, dass beide Netze zu Beginn untrainiert sind. Dieses Henne-Ei-Problem wird dank einer sogenannten Value-Funktion, manchmal auch Loss-Funktion, gelöst, die das korrekte Verhalten beider Netze formalisiert (Abbildung 2). Die originale Funktion von Goodfellow ist analog zu einem Klassifikationsproblem aufgebaut, schließlich versucht der Discriminator stets die Eingaben korrekt als real oder gefälscht zu charakterisieren.

Die Netze verfolgen unterschiedliche Ziele. Der Discriminator soll alle Eingaben korrekt als real oder falsch identifizieren und somit die Value-Funktion maximieren. Denn sie gibt die Wahrscheinlichkeiten an, ein reales Bild als korrekt zu erkennen, während gefälschte Bilder als Täuschung entlarvt werden. Dagegen versucht der Generator falsche Bilder real wirken zu lassen, und den zweiten Summanden der Funktion zu minimieren. Seine Ausgaben, die alle per Definition gefälscht sind, sollen real wirken. Das Prinzip dieser Value-Funktionen ist immer ähnlich, auch bei allen in der Literatur verfügbaren Alternativen zu Goodfellows Variante.

Methodisch wird auf das Gradientenverfahren gesetzt, das im Machine Learning der De-facto-Standard für Optimierungsprobleme ist. Das Prinzip hilft bei der Suche nach einem Minimum beziehungsweise Tal einer Funktion: In jedem Schritt folgt der Algorithmus der steilsten Richtung abwärts, bis dieser am Ende hoffentlich im Minimum endet. Aus Performancegründen wird dieser Gradient nicht ausgerechnet, sondern stochastisch approximiert.

Mithilfe des Gradienten lassen sich die Parameter beider neuronalen Netze anhand der Value-Funktion optimieren. Beide Netze lernen abwechselnd: Zunächst trainiert der Discriminator bis zu k-mal und aktualisiert seine Parameter entsprechend. Darauf folgt dasselbe Training mit dem Generator, dessen Gradient jedoch abweicht von dem des Diskriminators, da beide Netzwerke einen unterschiedlichen Einfluss auf die Value-Funktion haben. Das Wiederholen dieses Prozederes sorgt dann, so die Hoffnung, für Konvergenz.

Die Hürden beim Trainieren von GANs

Das Training, das Goodfellow in schlanke vier Schritte herunterbricht, ist in der Realität hochkomplex. Die Daten und Eingaben, mit denen gearbeitet wird, haben eine hohe In-

cewe

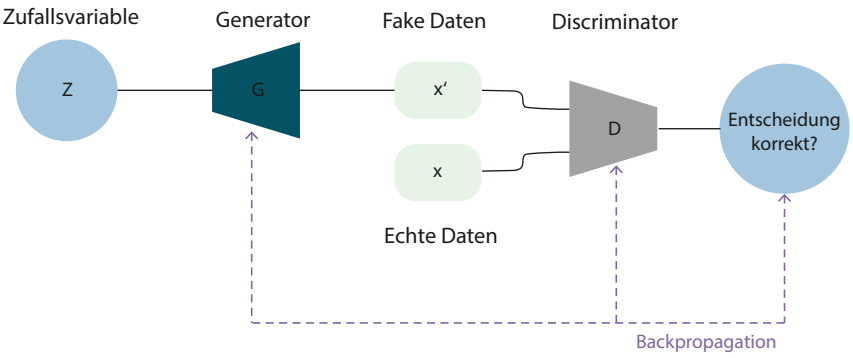
Wir von CEWE
gestalten mit Mut und frischen Ideen die Zukunft.

CEWE ist Europas führender Fotoservice und Markenhersteller im Fotofinishing. Mit rund 4.000 Mitarbeitenden sind wir in über 20 Ländern präsent und stellen für unsere Kund:innen jeden Tag einzigartige und ganz persönliche Fotoprodukte her.

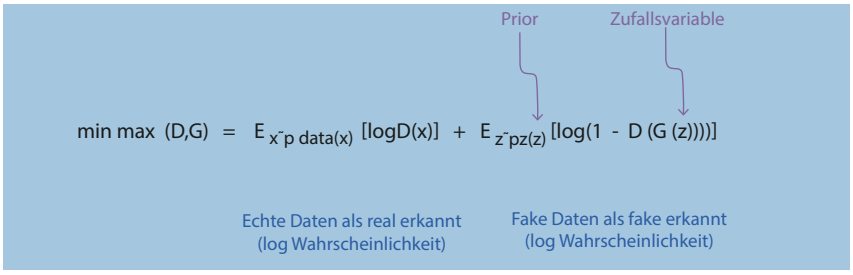
Neben unseren Produkten und Marken entwickeln wir unsere Software und Webapplikationen eigenständig. Wir forschen kontinuierlich nach neuer Technik und innovativen Funktionen für unsere Produktionsprozesse.

Verschaffen Sie sich auf unserer Karriereseite einen ersten Einblick von uns. Werden auch Sie Teil der CEWE Familie.
company.cewe.de/de/karriere

Wir freuen uns auf ein persönliches Kennenlernen.



>> Die Struktur eines Generative Adversarial Network (Abb. 1).



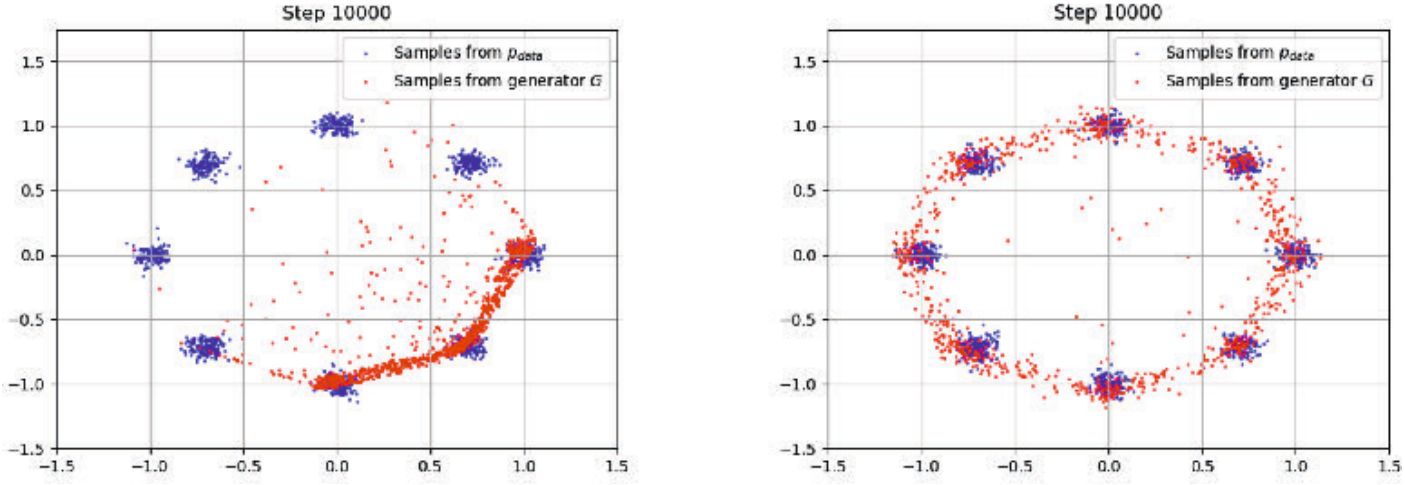
>> Die Value-Funktion von Ian Goodfellow beinhaltet die Wahrscheinlichkeiten, dass echte Bilder und Fakes korrekt als solche erkannt werden (Abb. 2).

formationsdichte und sind hochdimensional, sodass es häufig nicht einfach ist, heuristisch zu entscheiden, welche Methoden und Verfahren für spezifische Anwendungen die besten Resultate liefern. Die Forschung trifft daher auf immer neue Schwierigkeiten und Probleme. Eines der häufigsten ist das Phänomen des Mode Collapse.

Der Generator soll vielfältige und abwechslungsreiche Ausgaben erzeugen, doch oft beobachtet man, dass die Ausgaben sehr konzentriert auf einen Teilbereich aller Möglichkeiten sind. Statt zum Beispiel mannigfaltige Landschaften von tropischen Wäldern bis hin zu afrikanischen Savannen zu schaffen, erzeugt das Modell nur gräuliche Seelandschaften. Vom weiten Spektrum an zufälligen Eingabevariablen führen also viele zu einer kleinen Gruppe an Ausgaben. Daher spricht man im Englischen auch von Collapse – dem Zusammenbruch vieler Eingaben zu wenigen Ausgaben (Abbildung 3).

Während des Trainings entdeckt der Generator häufig eine Ausgabe, die den Discriminator besonders gut täuscht, sodass er immer weiter ähnliche Ausgaben produziert. Mit weiterem Training verschlimmert sich dieses Problem dann zunehmend, bis schlussendlich nur eine kleine Teilmenge aller Möglichkeiten übrig bleibt. Die Ursache liegt in einer falschen Optimierungsreihenfolge. Die Value-Funktion muss min-max optimiert werden; passiert das nicht, tritt häufig ein Mode Collapse auf. Das geschieht nicht bewusst, sondern durch das abwechselnde Training beider Netze. Hierdurch variiert oft die Lerngeschwindigkeit beider, was dazu führen kann, dass sich die Reihenfolge praktisch umdreht.

Die Bekämpfung des Problems bedient sich der Intuition der Nutzerinnen und Nutzer. Genauso wie Mode Collapse



>> Die roten Punkte im linken Bild formen keinen Kreis, sondern konzentrieren sich in der unteren rechten Ecke. Ein Symptom für Mode Collapse (Abb. 3).

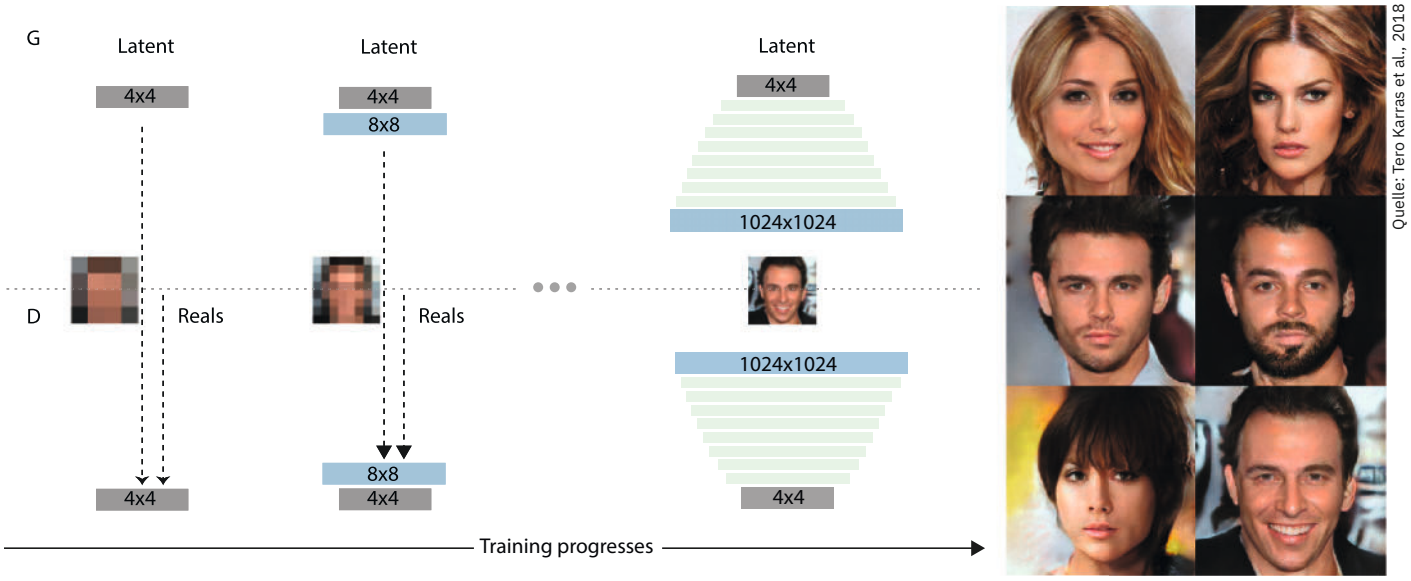
eingangs entdeckt wurde – alle Ausgaben des Netzes ähneln sich sehr stark –, wird dem auch entgegengewirkt. Der Discriminator muss einen Ähnlichkeits-Check implementieren. Das heißt, die Ähnlichkeit eines Output-Batches wird quantifiziert und dem Discriminator als Wert übergeben. Entdeckt dieser, dass alle Ausgaben quasi gleich aussehen, bestraft er den Generator hierfür, indem er die Ausgaben mit hoher Sicherheit als falsch klassifiziert.

Konvergenz: Hoffnung und Heuristiken

Neben Mode Collapse existiert auch eine viel fundamentalere Schwierigkeit: Das Modell kann schlicht auch nach tagelangem Training nur inhaltsloses Pixelrauschen erzeugen. Zu den vielfältigen möglichen Ursachen hierfür zählt eine Dysbalance im Training. Wenn eines der beiden Netze stark dominiert, kann der Gegner praktisch nicht mehr lernen und bleibt unbestimmt lange schlecht in seiner Performance – das offenbart die Erfolgsquote der Netze, oft Score genannt. Der Score eines Netzes schnellt bei Divergenz rasch zu 100 Prozent, während das andere Netz durchgehend versagt.

Das dominierende Netz nutzt die Schwächen im Gegner perfekt aus, sodass kein Raum für Verbesserung übrig bleibt. Ein guter Vergleich ist der Sport: Training mit einem hoffnungslos überlegenen Gegner führt selten zu einer maßgeblichen Verbesserung. Stattdessen lautet der allgemeine Rat, dass die Fähigkeiten des Gegners nur etwas über den eigenen liegen sollten, um das bestmögliche Trainingsergebnis für beide zu erzielen. Ganz wie bei der Divergenz führt ein zu großes Ungleichgewicht zu Stillstand. Diese Divergenz zu beheben ist in der Praxis deutlich schwieriger als der Kampf gegen Mode Collapse.

Zwar existieren viele hochspezifische Lösungsansätze, eine optimale Lösung für jeden Fall existiert aber nicht. Das Anpassen der Lernraten beider Netzwerke führt jedoch häufig zum gewünschten Erfolg. Konkret zeigten KI-Forscher in einem Paper aus 2017 (alle Links zum Artikel unter ix.de/zrx6), dass ein gezielt langsameres Training des Generators zu einer lokalen Konvergenz führen kann. Hierzu existieren spezifische Regeln verschiedener Autoren, um die genaue Geschwindigkeit zu wählen. Das Erreichen von Konvergenz darf allerdings nicht mit dessen Praktikabilität verwechselt werden – Konvergenz, die Jahre benötigt, ist in der Praxis wenig hilfreich. Denn häufig sind in der Praxis die Gradien-



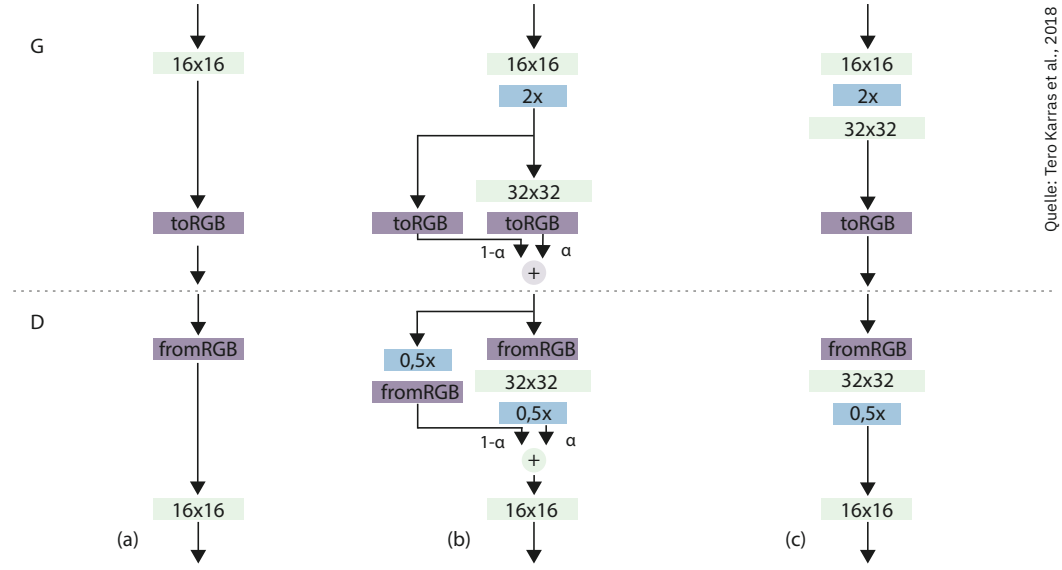
Quelle: Tero Karras et al., 2018

>> Der Trainingsprozess im Paper von Karras und Mitautoren zeichnet sich durch das Hinzufügen neuer, höherdimensionaler Ebenen aus (Abb. 4).

ten so klein, dass die Optimierung im Training effektiv in winzigen Schritten vonstattengeht.

Vanishing Gradients und Aktivierungsfunktionen

Neuronale Netze umfassen unzählige Ebenen, bestehend aus Neuronen, die mit der jeweils nächsten Ebene verbunden sind. Schließlich werden all diese Verbindungen mithilfe einer Aktivierungsfunktion aggregiert und zu einer Zahl kondensiert. Wählt man diese Funktion ungeschickt, kann es zu kleinen Gradienten – auch vanishing genannt, also verschwindende Gradienten – kommen. Die Sigmoidfunktion ist eine beliebte Wahl hierfür, da sie alle reellen Zahlen auf das Intervall von 0 bis 1 schrumpft. Die Ableitung des Sigmoidfunktion ist von besonderer Relevanz, da das Gradientenverfahren diese nutzt – und genau diese macht eine Sigmoidfunktion so ungeeignet. Funktionswerte der Ableitung sind alle numerisch sehr klein. Bestenfalls sind die Werte bei 0,2, sehr häufig liegen sie aber näher an 0. Das sorgt dafür, dass das Training deutlich ausgebremst wird, da die Schrittgröße im Gradientenverfahren schrumpft.



Quelle: Tero Karras et al., 2018

>> Die Architektur innerhalb der Progressive GANs, die die höheren Auflösungen langsam einführt (Abb. 5).

Eine beliebte Alternative und Lösung des Problems ist die Nutzung einer linearen Funktion, die bei negativen Eingaben 0 ausgibt. Die sogenannte Rectified-Linear-Unit-Funktion hat sich in der Praxis durch ihre Einfachheit und dennoch gute Performance bewährt und ist daher immer weiter verbreitet. Die Ableitung erzeugt numerisch größere Werte, sodass die Größenordnung des Gradienten steigt.

Datenmassen als Trainingsgrundlage

Nahezu alle Anwendungen des Machine Learnings erfordern Unmengen an Daten, dazu zählen auch Generative Adversarial Networks. So kommt es, dass sich viele Paper mit denselben Datensets befassen und diese auf verschiedene Arten analysieren und nutzen, um neue Architekturen oder Feinheiten zu erforschen. Eine sehr beliebte Sammlung ist CelebA, die Bilder von 200 000 Gesichtern von rund 10 000 Menschen in einer Auflösung von 178x218 Pixeln enthält und Labels wie Geschlecht, Alter oder Haarfarbe ergänzt. Allein auf GitHub finden sich über 2 000 Paper, die dieses Datenset in ihrer Forschung nutzen: Zu den Anwendungen zählen das Generieren von Bildern, Untersuchungen zu Deepfakes oder fortgeschrittene Dekompressionsmethoden.

Mithilfe dieses Datensatzes gefälschte Fotos von Menschen zu erzeugen, ist allerdings schwierig. Die schlechte Auflösung lässt die Ausgaben nicht gerade fotorealistisch wirken, besonders nicht in Zeiten von 4K-Handykameras. Das Problem mit höheren Auflösungen in Generative Adversarial Networks ist jedoch, dass die erhöhte Anzahl an Pixeln das Training erschwert. Der Discriminator kann einfacher Artefakte erkennen und ist im Allgemeinen präziser, schlicht aufgrund der erhöhten Menge an Informationen, die ihm vorliegen. Das verlangsamt das Training und benötigt zudem deutlich mehr Speicherplatz sowie Rechenpower. Die Paper, die dennoch auf das Generieren hochauflösender Fotos von Menschen abzielen, suchen daher nach speziellen GAN-Architekturen, um dieses Unterfangen zu vereinfachen.

Falsche Promis

Tero Karras und sein Forschungsteam haben genau das erreicht: Das Team hat den CelebA-Datensatz in einer höheren Auflösung neu generiert, um hiermit seine neue Architektur, sogenannte Progressive GANs, zu trainieren. Die Idee ist, die Auflösung im Laufe des Trainings kontinuierlich zu erhöhen. Angefangen mit 4x4-Pixel-Bildern, erhöhen die Autoren immer dann die Auflösung, sobald eine gewisse Stabilität im Training erreicht wurde. Dasselbe gilt für den Discriminator, der immer größere Fotos bewerten und klassifizieren kann (Abbildung 4). Der Kernpunkt der Idee ist, dass das Netz eine gewisse Stabilität aufweist, bevor man es mit einer höheren Auflösung konfrontiert. Auch der Übergang ist darauf ausgelegt, die Stabilität zu bewahren. Anstatt die neue Ebene für höhere Auflösungen schlicht einzufügen, wird diese langsam eingeführt (Abbildung 5).

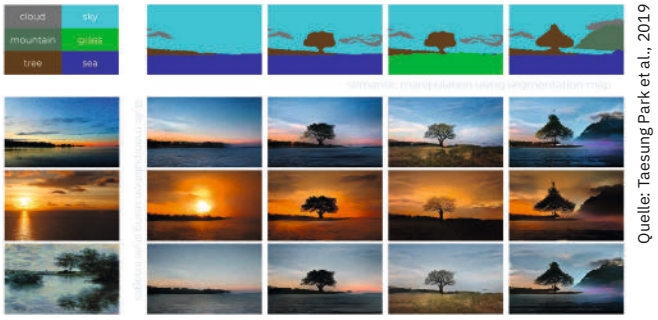
Das langsame statt abrupte Einführen vermeidet einen plötzlichen Schock und der vorhandene Trainingsfortschritt lässt sich nahezu ohne Verlust auf die neue Auflösung übertragen. Das geschieht mithilfe eines Bypasses: Das neuronale Netz erhält zwei Verbindungen ausgehend von der alten, niedrig aufgelösten Ebene. Eine durch die neue, größere Ebene hindurch und eine, die diese umgeht. Daraus bildet sich ein gewichtetes Mittel, das der Verbindung zur neuen Ebene kontinuierlich mehr Gewicht verleiht.

Zusätzlich wird nach jeder Ebene das Ausgabebild entweder mithilfe der Nearest-Neighbor-Methode hochskaliert oder mittels Averaging herunterskaliert. Das Umwandeln dieses verschwommenen Resultates in ein sinnhaftes Bild erfolgt dann in den darauffolgenden Ebenen. So gelingt es der Progressive GAN Architektur zunehmend realitätsnahe Fotos von nicht existierenden Personen zu schaffen. Dennoch sind die Ausgaben trotzdem häufig eher qualitativ minderwertig

software quality lab
Academy
QA method competence for
SOFTWARE ENGINEERS
REQUIREMENTS
IREB International Requirements Engineering Board
CPRE-FL Foundation Level
CPRE-AL Advanced Level
ARCHITECTURE
iSAQB
CPSA-F Foundation Level
CPSA-A Advanced Level
TESTING
ISTQB International Software Testing Qualifications Board
CTFL Foundation Level
CTAL Advanced Level
CT-TAE Test Automation Engineer
TÜV AUSTRIA ZERTIFIZIERT
ÖCERT
MORE THAN 70 SEMINARS
academy.software-quality-lab.com
QA training courses with over 2.000 participants every year!

Quelle: Tero Karras et al., 2018

>> Die Ergebnisse des Progressive GAN können sowohl sehr gut, als auch mangelhaft sein (Abb. 6).



Quelle: Taesung Park et al., 2019

>> Ein Beispiel verschiedener Eingaben und deren generierter Resultate mithilfe des GauGANs (Abb. 7).

und offenbaren deutlich, dass sie computergeneriert sind (Abbildung 6). Dasselbe Team um Karras ist auch für die Erschaffung der StyleGAN-Architektur verantwortlich, die ihre Prominenz vor allem durch die populäre Webseite thispersondoesnotexist.com erlangt. Die neuere Architektur bügelt bestehende Probleme in Progressive GAN aus.

Atemberaubende Landschaften per Knopfdruck

Die Resultate des Progressive GANs sind beeindruckend, jedoch bietet dieses Netzwerk den Nutzern keinerlei Möglichkeiten, die Ergebnisse zu beeinflussen. Ein Paper von Park und Co-Autoren aus 2019 ändert das, da die Nutzerinnen und Nutzer die dort generierten Landschaften gestalten können. Der GauGAN, benannt nach dem Maler Paul Gauguin, kann täuschend echte Landschaften erzeugen, die sich mithilfe zweier Werkzeuge beeinflussen lassen: der Segmentation Map und dem Style Image (Abbildung 7).

Die Segmentation Map fungiert wie eine Schablone und gibt an, wo sich welches Element auf dem finalen Bild befinden soll. Sechs Strukturen können unterschieden werden: Wolken, Himmel, Berge, Gras, Bäume und Wasser. Diese sind mit einer jeweils anderen Farbe in der Segmentation Map kodiert. Sowohl der Generator, als auch der Discriminator nutzen dieses Bild dann für ihre Zwecke. Der Discriminator braucht diese Information, um im Trainingsprozess dafür zu sorgen, dass der Generator die Schablone einhält. Eine unzureichende Übereinstimmung zwischen Segmentation Map und Generator-Ausgabe führt somit zur Bestrafung durch den Discriminator.

Das Style Image hingegen ist nur dafür nötig, die allgemeine Stimmung der Szene zu schaffen. Ist die Ausgabe eher ein rötlicher Sonnenuntergang oder eine blau-grüne Sumpflandschaft? Das Bild wird daher zu Beginn des Modells genutzt und in Mittelwert und Varianz kodiert. Mit den Parametern wird dann die Zufallszahl für den Generator generiert, sodass die Ausgabe eine stilistische Ähnlichkeit aufweist.

Aller Anfang muss nicht schwer sein

Die Idee von Generative Adversarial Networks basiert auf der Annahme, dass Wettbewerb ein Weg zur Verbesserung ist. Ian Goodfellow hat dies formalisiert und mit einer mathematischen Grundlage versehen, sodass weitere Forschungsteams die GANs in den letzten Jahren zunehmend verbessern, verfeinern und weiterentwickeln konnten. Nicht nur die frei wählbare Value-Funktion, sondern auch die Struktur und die genauen Architekturen der beiden Netze schaffen Spielraum für Optimierungen.

Den möglichen Anwendungsszenarien sind keine Grenzen gesetzt. Theoretisch lässt sich alles generieren, wofür genügend Trainingsdaten vorhanden sind – und nicht bloß menschliche Gesichter oder Landschaften. Die präsentierten Beispiele bieten einen ersten Überblick über die Vielfältigkeit der zugrundeliegenden Technik und wie Autorinnen und Autoren diese nutzen und anpassen. (mdo)

Quellen

Der hier gekürzte Beitrag ist in voller Länge auf heise Developer erschienen. Alle Links zum Artikel finden sich unter ix.de/zrx6.

Timo Zander

ist Senior Software Developer bei Senacor Technologies. Er interessiert sich für Open Source, das JavaScript-Universum und aufstrebende Technologien.

> Praktische Barrierefreiheit in der Webentwicklung

Maria Korneeva

Mit pragmatischen Tipps und Tricks lässt sich Barrierefreiheit in die tägliche Entwicklung integrieren – um so den Sprung vom Tooling zum Mindset zu schaffen.

Im Juli 2021 trat in Deutschland das Barrierefreiheitsstärkungsgesetz (BFSG) in Kraft, das den European Accessibility Act (EAA) umsetzt. Das BFSG definiert Anforderungen an die Barrierefreiheit von Produkten und Dienstleistungen, die nach dem 28. Juni 2025 auf den europäischen Markt kommen oder den Verbrauchern angeboten werden. Dazu gehören unter anderem der gesamte E-Commerce-Bereich, Hard- und Software, aber auch der nationale Personenverkehr oder Bankdienstleistungen. Das Gesetz betrifft demnach auch Webentwicklerinnen und Webentwickler, die Anwendungen für den Privatsektor implementieren. Auf ihrem Weg zu barrierefreien Anwendungen können sie auf eine Vielzahl unterstützender Tools und Techniken zurückgreifen.

Potenzielle Bußgelder bei EAA-Verstößen

Der European Accessibility Act ist nicht nur ein wichtiger und notwendiger Schritt, um das Internet für alle zugänglicher zu machen, sondern kann auch eine ernsthafte Gefahr darstellen, wenn er nicht eingehalten wird. Wenn ein Produkt oder eine Dienstleistung die Anforderungen an die Barrierefreiheit nicht erfüllt, können die deutschen Marktüberwachungsbehörden einen Rückruf oder die Einstellung anordnen. Darüber hinaus drohen Bußgelder von bis zu 100 000 Euro (weitere Informationen zum EAA unter ix.de/znvk). Ausnahmen gelten nur für kleine Unternehmen mit weniger als zehn Beschäftigten, einem Jahresumsatz von weniger als zwei Millionen Euro oder einer Jahresbilanzsumme von weniger als zwei Millionen Euro.

Der Stand der Web-Accessibility und der IT-Projekte kurz vor 2025 wird voraussichtlich wie folgt aussehen:

- Viele IT-Projekte laufen bereits oder stehen kurz vor ihrem Beginn, ohne dass ein spezielles Budget für Barrierefreiheit zur Verfügung steht. Das bedeutet, dass die Inklusivität des Produkts mit geringen Kosten erreicht werden soll – in Bezug auf Geld und geistigen Aufwand.
- Teams verfügen noch nicht über das nötige Accessibility-Know-how. Idealerweise sollte das Tooling daher auch Anfängern eine Anleitung bieten.

Accessibility sollte eine regelmäßige Praxis sein, keine Einmalaktion. „Man führt sie einmal ein, und dann ist man barrierefrei“ ist ein Mythos. Das bedeutet, dass Entwicklerinnen und Entwickler sich regelmäßig darum kümmern sollten, sowohl bei der Arbeit an neuen Features als auch bei der Wartung bestehender Features. Es gibt zwar verschiedene Accessibility-Checklisten, aber Selbstaudits funktionieren kaum dauerhaft, wenn sie nicht automatisiert oder gut in den Entwicklungsprozess integriert sind.

Darüber hinaus ist es derzeit eine unrealistische Erwartung, eine Website zu hundert Prozent barrierefrei zu gestalten. Web-Accessibility ist ein Kontinuum, weshalb Entwicklerinnen und Entwickler sie nicht mit einer „Alles oder

In a Nutshell

> In zwei Jahren drohen Webseiten, die mit dem European Accessibility Act (EAA) nicht konform sind, hohe Bußgelder.
> Ein Wandel vom Abarbeiten von Checklisten hin zum inklusiven Mindset ist notwendig, um Barrierefreiheit als kontinuierlichen Prozess zu etablieren.
> Mit zahlreichen Tools und Techniken lässt sich die Accessibility in den Webentwicklungsprozess integrieren.

nichts“-Haltung behandeln sollten. Stattdessen sollte der Schwerpunkt darauf liegen, ob Websites mehr oder weniger zugänglich sind. Es lohnt sich jeder einzelne Schritt, um eine Website inklusiver zu machen – und darüber zu sprechen. Auf dieser Grundlage bieten sich einige pragmatische Tipps und Tricks an, wie Webentwickler Accessibility-Features in ihre tägliche Programmierroutine integrieren können. Diese leicht durchführbaren Maßnahmen lassen sich mit sofortiger Wirkung und fast ohne Kosten umsetzen. Die folgenden Tools machen eine Website nicht vollständig barrierefrei, aber wie das Sprichwort sagt: Der Weg ist das Ziel.

Accessibility-Audits leicht gemacht durch Erweiterungen

Zwei Aspekte machen aus manuellem Accessibility-Testing ein teures und umständliches Unterfangen: Erstens benötigen die Testerinnen und Tester Kenntnisse über alle Anforderungen, und zweitens müssen sie die gesamte Website Zeile für Zeile nach Accessibility-Verstößen durchsuchen. Um diesen Aufwand deutlich zu reduzieren, stehen hilfreiche Browsererweiterungen bereit.

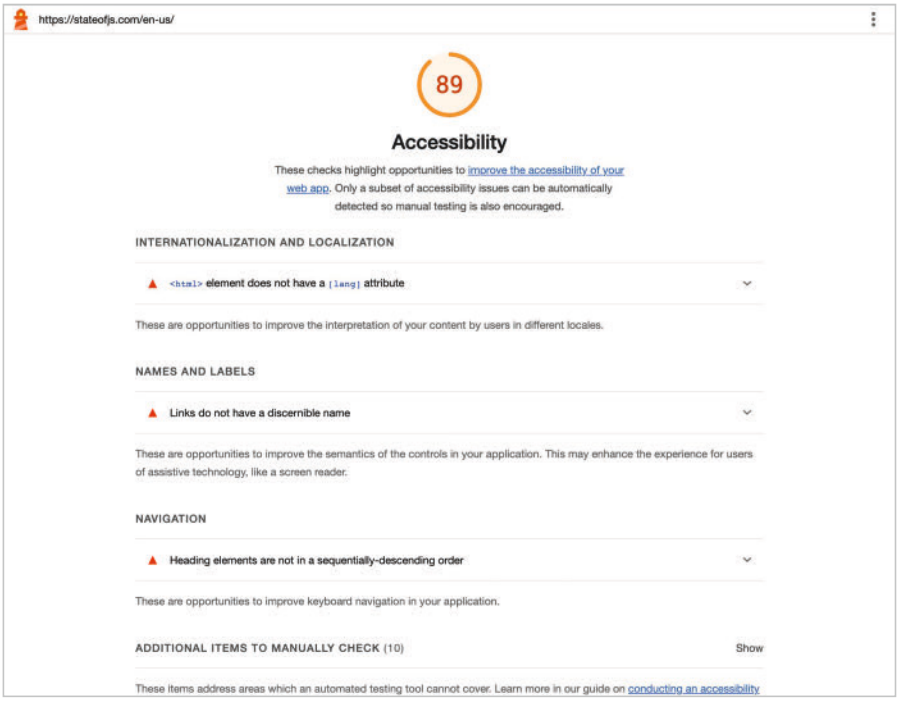
Eine dieser Erweiterungen ist Lighthouse, das vor allem für seine Performance-Analyse bekannt ist. Lighthouse lässt sich jedoch auch nutzen, um die Web-Accessibility zu bewerten und zu verbessern. Es ist direkt in Google Chrome integriert und ermöglicht es Entwicklerinnen und Entwicklern, ihre Websites im Hinblick auf potenzielle Accessibility-Probleme zu beurteilen und Einblicke in verbesserungsbedürftige Bereiche zu erhalten.

Nach Abschluss der Prüfung präsentiert Lighthouse einen umfassenden Bericht, der eine Bewertung der Barrierefreiheit, eine Liste der erfolgreich bestandenen Audits und spezifische Empfehlungen zum Beheben der festgestellten Probleme enthält (siehe Kasten „Lighthouse Accessibility Score“). Die Vorschläge basieren auf den Web Content Accessibility Guidelines (WCAG) (siehe Kasten „WCAG“) und sind entsprechend den Auswirkungen auf die User gewichtet. Zu den kritischen Regeln zählen:

- `[user-scalable="no"]` darf nicht im Element `<meta name="viewport">` vorhanden sein und das Attribut `[maximum-scale]` muss größer `5` sein. Wenn die Website gegen diese Regel verstößt, bedeutet das, dass sie entweder das Zoomen deaktiviert oder einschränkt, wie stark sich der Inhalt der Website vergrößern lässt. Das ist problematisch für Menschen mit Sehschwäche, die auf Bildschirmlupen angewiesen sind.
- `<video>`-Elemente enthalten ein `<track>`-Element mit dem Attribut `[kind="captions"]` zum Anzeigen von Untertiteln. Diese Prüfung ist für gehörlose Nutzerinnen

Lighthouse Accessibility Score

Lighthouse liefert einen Lighthouse Accessibility Score zwischen 0 und 100, basierend auf der Anzahl der bestandenen beziehungsweise nicht bestandenen Audits, wobei 100 den bestmöglichen Wert darstellt. Jede Barriere kann eine potenzielle Auswirkung auf die Nutzer haben, beispielsweise „kritisch“, „schwerwiegend“, „mäßig“ oder „geringfügig“, je nachdem, wie stark sie das Verwenden der Website beeinträchtigen könnte. Das Tool beurteilt die Auswirkungen auf die User, und Entwickler können diese Beurteilungen nutzen, um die Barrierefreiheit der Webseite zu verbessern, was wiederum den Lighthouse Accessibility Score positiv beeinflussen kann.



>> Lighthouse zeigt die Ergebnisse der Accessibility-Prüfung mit einer Punktzahl von 89 für die Website „State of JavaScript“ und hebt dabei folgende Probleme hervor: fehlendes lang-Attribut, nicht erkennbare Link-Namen und nichtsequenzielle Verwendung von Überschriftenelementen.

Wenn du mit Hightech Menschen verbindest.

Das ist FRITZ! Das ist AVM.

Jetzt bewerben auf avm.de/jobs

Web Content Accessibility Guidelines (WCAG)

Die Web Content Accessibility Guidelines (WCAG) 2.1 sehen drei Abstufungen vor, um eine Bandbreite von Nutzergruppen und Situationen abzudecken. Diese Stufen umfassen Stufe A (basic), Stufe AA (intermediate) und Stufe AAA (advanced). Jede höhere Stufe bedeutet die Zugänglichkeit für breitere Nutzergruppen und erfüllt demnach auch die Anforderungen der niedrigeren Stufen. Entspricht eine Webseite beispielsweise der Stufe AA, erfüllt sie automatisch die Standards sowohl von Stufe A als auch von Stufe AA. Für Stufe A müssen 25 Kriterien erfüllt sein, für Stufe AA sind 13 zusätzliche Kriterien erforderlich. Stufe AAA besitzt 23 weitere Kriterien.

und Nutzer wichtig, die sonst nur begrenzten oder keinen Zugang zu den Informationen in einem Video hätten. Personen, die eine Fremdsprache lernen, profitieren ebenfalls von Untertiteln.

Die Vorteile von Lighthouse sind, dass es in Google Chrome enthalten ist und messbare Einblicke liefert. Allerdings stützt es sich auf die Testregeln der axe-core-Bibliothek, einer etablierten Engine für das Accessibility-Testing von Web-User-Interfaces, und führt nur einen begrenzten Teil dieser Regeln aus. Für ein vollständiges Audit empfiehlt sich daher die Installation der Erweiterung axe DevTools, die auch für die Browser Mozilla Firefox und Microsoft Edge verfügbar ist. Zum Beispiel erreicht die Website „State of JavaScript“ in Lighthouse ein solides Ergebnis von 89 Punkten (Abbildung). Wenn man sie jedoch mit der Erweiterung axe DevTools testet, erscheinen 13 zusätzliche Empfehlungen im Vergleich zu Lighthouse.

Zu den zahlreichen weiteren Browsererweiterungen gehören WAVE, Siteimprove Accessibility Checker, Accessibility Insights und Tenon. Diese Tools bieten den Vorteil, dass sie verwertbare Erkenntnisse liefern, ohne umfangreiche Kenntnisse über Barrierefreiheit vorauszusetzen. Eine etablierte Option ist axe-core, aber ein Entwicklungsteam muss selbst entscheiden, welches Tool seinen Bedürfnissen am ehesten entspricht. Verlässt man sich jedoch ausschließlich auf Browsererweiterungen für Accessibility-Tests, kann das Einschränkungen für den Entwicklungs-Workflow mit sich bringen.

Wenn Browsererweiterungen nur während der Testphase und nicht während der gesamten Feature-Implementierung zum Einsatz kommen, können Accessibility-Probleme bis zu späteren Phasen des Entwicklungsprozesses unentdeckt bleiben. Browsererweiterungen testen in der Regel Seite für Seite, was sich bei der Arbeit an umfangreichen oder komplexen Websites als zeitaufwändig und umständlich erweisen kann. Außerdem bieten die Erweiterungen unter Umständen nur eingeschränkte Konfigurationsmöglichkeiten

oder testen nur nach einem vorgegebenen Regelwerk, wodurch sie projektspezifische Anforderungen übersehen können. Automatisierte Unit- und End-to-End-Tests können einigen dieser Einschränkungen entgegenwirken.

Automatisiertes Accessibility-Testing

Automatisierte Tests sind programmierte Evaluationen einzelner Komponenten und kompletter Anwendungs-Workflows, die zum Gewährleisten konsistenter und effizienter Tests ohne manuelle Eingriffe ablaufen. Sie lassen sich in bestehende Test-Frameworks und Versionsverwaltungssysteme integrieren. Allgemeine Testbibliotheken lassen sich auch für Accessibility-Tests verwenden, etwa um sicherzustellen, dass eine Schaltfläche sowohl auf Klick- als auch auf Eingabetasten-Ereignisse reagiert und dass User per Tastatur dorthin navigieren können (WCAG 2.1 Keyboard Accessible). Anstatt das Rad neu zu erfinden und sich die notwendigen Testfälle auszudenken, kann man stattdessen auf die Vorteile spezieller Accessibility-Bibliotheken setzen, die sich auf gängige Accessibility-Anforderungen beziehen. Die folgenden Beispiele zeigen verschiedene Bibliotheken, Test-Engines und -Tools, und wie sie den Testing-Prozess unterstützen.

Testing Library mit Erweiterung

Testing Library bietet drei Funktionen für Accessibility-Testing: `getRoles()`, `logRoles()` und `isInaccessible()`. Es setzt auf Tests, die der realen Nutzung von Webseiten sehr nahe kommen, und legt den Fokus auf Queries, die visuelle/Maus-Benutzer ebenso berücksichtigen wie jene, die assistive Technologien verwenden, beispielsweise `getByRole()`, `getByLabelText()` und `getByPlaceholderText()`. Zusätzlich lässt sich eine Erweiterung für das Keyboard-Testing verwenden, um das Verhalten von Usern zu simulieren, die ausschließlich per Tastatur arbeiten.

Accessibility Testing Engines

Accessibility-Regelsätze sind das Herzstück mehrerer Testwerkzeuge. Die Grundlage dafür bilden Standards wie WCAG und Accessible Rich Internet Applications (ARIA). Am bekanntesten ist axe-core von Deque Systems, eine weit verbreitete Accessibility Engine für das Testen von Web-User-Interfaces. axe-core lässt sich mit verschiedenen Test Runnern und Frameworks wie Jest (jest-axe), Ember (ember-a11y-testing), React (axe-core/react) und Vue (vue-axe) integrieren. Das Verwenden von axe-core ist auch in End-to-End-Tests mit Tools wie Cypress (cypress-axe), Selenium (axe-core-selenium) und Playwright (axe-core/playwright) möglich.

Für den Bereich **Informationstechnik** suchen wir für unseren Standort **Darmstadt** Sie als

Softwareentwickler .NET (m/w/d)

Aufgaben

- Entwicklung und kontinuierliche Verbesserung und Standardisierung von Anwendungen beim TÜV Hessen mit dem Schwerpunkt Auto Service
- Entwicklung von Front- und Backend der entsprechenden Anwendungen
- Abstimmung von Anforderungen der unterschiedliche internen und externen Stakeholder-Spezifizierung
- Third-Level-Support für die Anwendungen des Fachbereich Auto Service

Qualifikationen

- Abgeschlossenes Studium der Fachrichtung Informatik, Wirtschaftsinformatik oder vergleichbare Ausbildung im Bereich Anwendungsentwicklung
- Gute Programmierkenntnisse im Microsoft .Net Umfeld (C#, WPF), Kenntnisse mit relationalen Datenbanken (MySQL)
- Erfahrung im Umgang mit Webservice (REST), mit agilen Methoden wie Scrum sind wünschenswert
- Ausgeprägtes Know-how von Methoden, Vorgehensmodellen und Tools zur Softwareentwicklung
- Analytische Fähigkeiten, Kommunikationsstärke, Teamfähigkeit und Proaktivität
- Flexibilität, Einsatzfreude und Belastbarkeit verbunden mit einer selbstständigen Arbeitsweise

Wir bieten Ihnen

- Eine anspruchsvolle Tätigkeit in einem erfolgsorientierten Team
- Selbstständiges Arbeiten in einem interessanten, weitgefächerten Aufgabengebiet
- Flexible Arbeitszeiten unter Berücksichtigung der individuellen Work-Life-Balance
- Vielfältige Leistungen rund um „Familie und Beruf“ sowie einen sicheren Arbeitsplatz

Geben Sie Ihrer Zukunft ein Stück Gewissheit …
Wir freuen uns auf Sie!

TÜV Technische Überwachung Hessen GmbH

Bitte haben Sie Verständnis dafür, dass wir Bewerbungen nur online entgegennehmen.

www.tuev-hessen.de/jobs

Aus Gründen der Lesbarkeit wird auf geschlechtsspezifische Formulierungen verzichtet. Personenbezogene Bezeichnungen beziehen sich auf Männer, Frauen und Divers in gleicher Weise.

Die Tests profitieren von einer umfangreichen und anpassbaren Regelsammlung und sind für bestimmte WCAG-Versionen und die Konformitätsstufen A, AA oder AAA konfigurierbar (siehe Infokasten „WCAG“). Zu den weiteren Accessibility Testing Engines und Accessibility-Regelsätzen zählen die Accessibility Checker Engine von IBM, die WAVE Stand-alone API and Testing Engine von WebAIM, die Tenon/Access Engine von Level Access, ARC von TGPi und der Alfa Accessibility Checker der Softwarefirma Siteimprove.

Screenreader-Treiber

Guidepup ist ein Screenreader-Treiber für VoiceOver (macOS) und NVDA (NonVisual Desktop Access, Windows). Er ist darauf ausgelegt, Entwicklern dabei zu helfen, mit Screenreadern so zu navigieren, wie es ein User tun würde und hilft, auf automatisierte Weise festzustellen, was Nutzer von Screenreadern wirklich hören. Weitere Tools in dieser Gruppe sind JAWS Inspect, die Screen Reader Testing Library für NVDA und Assistive-Webdriver (JAWS und NVDA). Das korrekte Schreiben von Testaussagen setzt allerdings ein gewisses Verständnis der Funktionsweise von Screenreadern voraus. Die größte Einschränkung der genannten Tests und Browsererweiterungen besteht darin, dass Entwickler die Audits manuell auslösen müssen. Mit anderen Worten: Der Code ist in Ordnung, solange ihn niemand testet. Linter können dieses Defizit beheben.

Mit Accessibility Lintern zu einem inklusiveren Web – Zeile für Zeile

Ein Linter ist ein statisches Tool zur Codeanalyse, das Entwicklerinnen und Entwicklern hilft, potenzielle Probleme in ihrem Code zu erkennen, indem es ihn anhand einer Reihe vordefinierter Regeln oder Best Practices überprüft. Linter lassen sich in den Entwicklungs-Workflow integrieren, wo sie automatisch beim Speichern von Änderungen oder als Teil einer Continuous-Integration-(CI)-Pipeline ausgeführt werden.

Es gibt mehrere Accessibility Linter, von denen jeder seinen eigenen Regeln und Best Practices folgt. Einige Beispiele sind:

- Mehrere Plug-ins für ESLint auf der Grundlage des axe-core-Regelsatzes: für JSX (eslint-plugin-jsx-a11y), Vue (eslint-plugin-vuejs-accessibility), Lit (eslint-plugin-lit-a11y) und CoffeeScript (eslint-plugin-coffee)
- ASLint-Validierungsdienstprogramm auf der Grundlage des Tenon/Access-Engine-Regelsatzes
- ESLint mit angular-eslint/eslint-plugin: ein Accessibility-Checker für Angular-Anwendungen mit eigenem Regelwerk
- ESLint mit ember-template-lint: ein Accessibility-Checker für Ember- oder Handlebars-Vorlagen
- ESLint mit dem HTML-Linter: ein allgemeines Regelwerk für HTML-Linting, einschließlich Best Practices, SEO, Barrierefreiheit und Styles
- Stylelint mit dem Plug-in stylelint-a11y: Stylelint ist ein beliebter CSS-Linter, der sich durch Plug-ins erweitern lässt. Das stylelint-a11y-Plug-in konzentriert sich auf Barrierefreiheitsprobleme in CSS-Code.
- storybook-addon-a11y: Dieses Add-on ist streng genommen kein Linter, sondern integriert Accessibility-Checks unmittelbar in die Komponentenbibliothek und bietet visuelles Feedback innerhalb der Storybook-Oberfläche.

Ein Blick auf die Regelsätze der Linter wirft ein Licht auf die bewährten Verfahren für barrierefreie Websites. Mehrere Linter, die auf unterschiedlichen Regelsätzen basieren, haben sich zum Beispiel für die Implementierung einer `no-positive-tabindex`-Regel entschieden. Diese Regel verlangt, dass alle Elemente, die die Eigenschaft `tabindex` verwenden, diese auf `0` oder negative Werte setzen. Das ist deshalb wichtig, weil Elemente mit einem positiven `tabindex` die ersten Elemente sind, die Benutzer auf einer Webseite ansteuern. In diesem Fall würden sich die Tastaturnavigation und die Screenreader-Ansagen von der visuellen (und logischen) Reihenfolge der Elemente unterscheiden, was Benutzer verwirren könnte.

Eine weitere, in mehreren Lintern implementierte Regel betrifft die Verwendung von Autofokus. Die automatische Fokussierung eines Formularsteuerelements kann sehbeeinträchtigte Menschen, die Screenreader verwenden, und Menschen mit kognitiven Beeinträchtigungen verwirren. Wenn der Autofokus zugewiesen ist, „teleportieren“ Screenreader ihre Benutzer auf das Formularsteuerelement, ohne sie zuvor zu warnen.

Zudem kontrollieren viele Regeln die Implementierung von Accessible Rich Internet Applications (ARIA), wie `aria-roles`, `aria-props` und `aria-unsupported-elements`. Diese Regeln kontrollieren die Verwendung von ARIA-Attributen, indem sie sicherstellen, dass die richtigen Werte und Wertetypen vorhanden sind.

Linter können Entwicklerinnen und Entwickler zwar vor Accessibility-Problemen warnen, aber sie können nicht verhindern, dass sie diese Warnungen ignorieren und den Code dennoch in Produktion geben. Der nächste Schritt, um eine barrierefreie Implementierung zu gewährleisten, wäre, das Linting in den Pre-Commit-Hook zu integrieren und damit Commits mit erkannten Problemen zu blockieren. Task-Runner und npm-Skripte können die gleiche Funktion erfüllen. Es gibt mehrere Plug-ins für Gulp und Grunt, die auf den Tenon- und axe-core-Regelsätzen basieren.

Während ein Entwicklungsteam selbst entscheiden könnte, ob es Browsererweiterungen und Linter verwenden möchte,

erfordert der nächste Schritt eine erste kleine Änderung in der Priorisierung von Accessibility-Problemen. Können sie ein Grund für einen Aufschub des Release sein? Wenn ja, mit welchen Tools lässt sich das durchsetzen?

Continuous Integration und Accessibility

Bei der Continuous Integration (CI) werden Codeänderungen regelmäßig in ein gemeinsames Repository integriert, um sicherzustellen, dass der Code stets in einem veröffentlichungsfähigen Zustand ist. Die Integration von Barrierefreiheitsprüfungen in die CI-Pipeline trägt dazu bei, Accessibility-Probleme frühzeitig zu erkennen und zu verhindern, dass sie die Produktion erreichen. Je nach verwendeten Tools und Diensten gibt es verschiedene Möglichkeiten, Barrierefreiheitsprüfungen in eine CI-Pipeline einzubinden.

- AccessLint und axe Linter für GitHub Actions überprüfen Codeänderungen in Pull Requests und hinterlassen einen Kommentar bei neu entdeckten Accessibility-Problemen, die auf den axe-core Assertions basieren. Wird das Auflösen von Kommentaren zu einem Pflichtkriterium für den Merge, ist die (teilweise) Automatisierung der Barrierefreiheit geboren! Einziger Nachteil: Es ist nur für persönliche Open-Source-Projekte kostenlos verfügbar.
- Im Allgemeinen lassen sich GitHub Actions, GitLab CI, Jenkins und weitere CI/CD-Dienste nutzen, um einen benutzerdefinierten Workflow zu erstellen, der Accessibility-Prüfungen mit Tools wie Lighthouse CI, axe-linter-action, accessibility-insights-scan oder Pa11y beinhaltet. Die in den vorangegangenen Abschnitten erwähnten Linter und automatisierten Testwerkzeuge können ebenfalls Teil dieses Workflows sein. Ein umfassendes Beispiel findet sich in Adrián Bolonios Artikel „Automating the accessibility tests of your source code with GitHub Actions“ (siehe ix.de/znvk).

Die Integration von Accessibility-Checks in die CI/CD-Pipeline ist entscheidend für das frühzeitige Erkennen von Problemen und das Aufrechterhalten einer inklusiven User Experience. Es ist jedoch wichtig, die potenziellen Auswirkungen auf die Performance zu berücksichtigen, etwa die Zeit vom Commit bis zur Veröffentlichung (Commit to Publish, C2P). Anfängliche Implementierungen automatisierter Accessibility-Tests können sich negativ auf die C2P auswirken, sodass sich die Ausführungszeiten einiger Tests mehr als verdoppeln.

Um die Auswirkungen auf die Leistung zu minimieren, sollte man Assertions bewusst einsetzen und sie auf bestimmte Bildschirmsegmente beschränken, wobei der Schwerpunkt auf wichtigen und oft benutzten Transaktionspfaden liegt. Es ist wichtig zu bedenken, dass eine hundertprozentige Abdeckung von Accessibility-Tests nicht das Ziel ist; Tests sind nur ein Werkzeug im Werkzeugkasten der Barrierefreiheit.

Optimierte Konfigurationseinstellungen, das Weglassen leistungsschwacher Regeln mit geringer Auswirkung und verteilte Testjobs können die Leistung erheblich verbessern. Alternativ können dedizierte Jobs Assertions parallel ausführen, um die C2P-Zeit niedrig zu halten und eine reibungslose CI/CD-Pipeline sicherzustellen.

Von Verständnis zu Praxis

Das eingangs erwähnte fehlende Know-how lässt sich mit einer Vielzahl an kostenlosen Ressourcen beheben, etwa Workshops, Kursen oder E-Learnings. Zu den bekanntesten gehören:

- Die MDN Accessibility Learning Area
- Learn Accessibility von Google Developers
- Digital Accessibility Foundations von der W3C Web Accessibility Initiative

Browsererweiterungen und Linter bieten erste Anhaltspunkte für die Barrierefreiheit im Web, aber die richtige Interpretation der Ergebnisse ist entscheidend. Diese Tools zeichnen sich durch eine hervorragende Syntaxprüfung aus, verfolgen aber einen „Einheitsansatz“, der für alles gilt. So sollten beispielsweise `alt`-Attribute nicht leer sein, aber nicht alle Bilder benötigen Textalternativen. Dekorative Bilder tragen nicht zum Inhalt bei und sollten daher nicht angekündigt werden. Das sture Befolgen von Audit-Aktionselementen kann die Benutzerfreundlichkeit von Screenreadern verschlechtern. Das Verständnis von ARIA-Rollen, -Bezeichnungen und -Attributen ist für die Barrierefreiheit entscheidend. Im ARIA Authoring Practices Guide (APG) heißt es: „Kein ARIA ist besser als schlechtes ARIA.“ Ein falscher Gebrauch von ARIA kann die Nutzerfreundlichkeit stark beeinträchtigen, etwa wenn man unsachgemäß einem `<div>`-Tag eine Rolle hinzufügt, ohne die erwarteten Tastaturinteraktionen zu implementieren. Mangelndes Verständnis bedeutet auch die Unkenntnis über wesentliche Funktionen wie „zum Hauptinhalt springen“. Damit können Benutzer, die auf die Tastaturnavigation angewiesen sind, sich wiederholende Navigationselemente umgehen und direkt auf den Hauptinhalt einer Seite zugreifen. Darüber hinaus ist es sehr hilfreich zu wissen, wie die WCAG zu navigieren und zu interpretieren sind, da sie weithin als Goldstandard für die Barrierefreiheit im Web gelten. Neben den Anforderungen bieten die WCAG auch wertvolle Techniken und Best Practices, die die Compliance erleichtern.

Die Schulung eines Entwicklungsteams ist ein nachhaltigerer Ansatz als die Installation von Lintern und die Integration von Barrierefreiheit in eine CI/CD-Pipeline, aber sie ist nicht der letzte Schritt. Die Einführung eines inklusiven Designprozesses in all seinen Facetten erfordert ein Umdenken in Unternehmen und hat eine enorme transformative Kraft. Das würde jedoch den Rahmen dieses Artikels sprengen.

Annehmen eines Accessibility-Mindsets

Der Weg zu einem barrierefreieren Web erfordert konsequente Anstrengungen und Aufmerksamkeit. Eine Reihe von Tools und Techniken ermöglichen das Schaffen inklusiverer Online-Erfahrungen, jedoch können automatische Barrierefreiheitstests nur 30 bis 50 Prozent aller Accessibility-Probleme identifizieren. Es ist von entscheidender Bedeutung, ein tieferes Verständnis für Barrierefreiheitsanforderungen zu entwickeln und eine Haltung anzunehmen, die Barrierefreiheit als Praxis und nicht als einmaliges Feature betrachtet, da wir uns den durch das Barrierefreiheitsstärkungsgesetz und den European Accessibility Act festgelegten Fristen nähern. Die in diesem Artikel vorgestellten Tools und Strategien dienen als pragmatische Ausgangspunkte auf diesem wichtigen Weg, aber man sollte sich mit Ressourcen wie dem Accessibility Automation Tracker und der Web Accessibility Evaluation Tools List über Aktualisierungen auf dem Laufenden halten.

Jeder Schritt auf dem Weg zu einem barrierefreieren Web ist ein Schritt, der es wert ist, gefeiert zu werden, und gemeinsam können wir die digitale Welt für alle zugänglicher machen. (mai)

Quellen

Links zu den genannten Gesetzen, Tools und Browsererweiterungen finden sich hier: ix.de/znvk

Maria Korneeva

ist Frontend Technology Lead und Google Developer Expert mit Fokus auf Angular. Sie schreibt für ng-conf und spricht auf Tech-Meetups und Konferenzen.

> Quereinstieg in die IT

Hallie Barrows

Karrierewege sind oft nicht geradlinig.
Viele entdecken erst später ihre Leidenschaft für IT.
Quereinsteigerin Hallie Barrows gibt Tipps, wie
der Quereinstieg in die IT gelingt.

Der klassische Weg in die IT führt über ein Informatikstudium. Das Statista Research Department veröffentlichte im August 2023 eine Statistik, die die Anzahl der Studierenden im Fach Informatik in Deutschland nach Geschlecht vom Wintersemester 1998/1999 bis 2022/2023 anzeigt. Im Wintersemester 2022/2023 studierten rund 143 582 Personen Informatik an deutschen Universitäten oder Hochschulen (alle Links zum Artikel unter ix.de/zxq8). Doch die jährlichen Absolventinnen und Absolventen decken bei Weitem nicht den Bedarf an Fachkräften im IT-Bereich – derzeit fehlen in Deutschland rund 137 000 Informatikerinnen und Informatiker (Stand November 2022). Der Grund: Durch die Digitalisierung und die damit einhergehenden neuen Fachgebiete entwickelt sich die Branche in einem rasanten Tempo. Dadurch wird das Thema Informationstechnik für so gut wie jedes Unternehmen wichtig.

In a Nutshell

- Der herkömmliche Weg in die IT über ein Informatikstudium wird zunehmend durch alternative Wege wie Ausbildungen, Coding-Bootcamps und IT-affine Hobbys ersetzt.
- Erfahrungen aus anderen Bereichen, Softskills und Leidenschaft sind gute Voraussetzungen für einen Quereinstieg.
- Ein erfolgreicher Quereinstieg erfordert Qualitäten wie Lernbereitschaft, klare Berufsziele, Teamarbeit und kontinuierliche Weiterentwicklung sowie Flexibilität bei der Standort- und Unternehmenswahl.

Studium ist nicht immer notwendig

Muss jede Person, die in der IT tätig ist, ein Studium in diesem Bereich abgeschlossen haben? Nein, denn Fachkräfte können auch eine entsprechende Ausbildung absolviert haben oder ursprünglich aus einer völlig anderen Branche stammen, aber ein paar Jahre relevante Berufserfahrung im IT-Umfeld gesammelt haben. Eine Ausbildung zum Fachinformatiker dauert in der Regel drei Jahre und setzt mindestens einen mittleren Schulabschluss voraus. Kann man gute Kenntnisse in Mathematik oder Informatik nachweisen, ist das ein großes Plus in der Bewerbung. Die Ausbildung vermittelt sowohl Praxis- als auch Theoriewissen.

Nach erfolgreichem Abschluss sind diverse Weiterbildungen möglich. So können sich Fachinformatiker noch weiter spezialisieren. Neben der Ausbildung ist auch der erfolgreiche Abschluss sogenannter Coding-Bootcamps gern gesehen. In diesen lernen die Teilnehmenden in einer praxisnahen Ausbildung alle wichtigen Programmierfähigkeiten. Das Ganze erfolgt im Schnelldurchlauf, denn das Bootcamp dauert, je nach Anbieter, in der Regel zwischen zwei und zwölf Monaten. Oftmals werden in dieser Zeit auch umfangreiche Praxisprojekte und gegebenenfalls ein Praktikum absolviert. Doch auch ein IT-nahes Hobby wie Gaming kann als Eingangstor in die Branche dienen.

Do #1: Wichtige Qualitäten erkennen und leben

Wie in jeder Branche gilt auch in der IT: Es ist noch keine Meisterin und kein Meister vom Himmel gefallen. Selbstverständlich ist es wichtig, sich beim Programmieren lernen realistische Ziele zu setzen. Bei einem Quereinstieg ist die Zeit zum nebenberuflichen Programmieren jedoch sehr begrenzt – die fünf Jahre, die das Informatikstudium im Bachelor plus Master mindestens dauert, stehen nicht zur Verfügung. Das liegt nicht nur daran, dass Quereinsteiger schnellstmöglich anfangen wollen, in dem neuen Bereich Fuß zu fassen, sondern auch, dass sie in der Regel parallel zum Lernprozess einen anderen Job ausüben. Deshalb ist es für den gelungenen Quereinstieg zwingend notwendig, wichtige Tugenden zu erkennen und auch zu leben. Neben Lernbereitschaft und Offenheit für die schnelle Entwicklung der Branche ist es auch wichtig, sein analytisches Denken zu fördern. Schließlich geht es in dem Beruf häufig darum, Lösungen für komplexe Probleme zu finden.

Ein Quereinstieg in die IT ist also nicht ausgeschlossen. Tatsächlich beschäftigen selbst große Unternehmen wie Zalando und Otto IT-Spezialistinnen und -Spezialisten mit nicht geradlinigem Werdegang. Kein Wunder, denn worüber sich viele nicht bewusst sind: Gesammelte Erfahrungen aus anderen Branchen vermitteln oftmals Softskills, durch die die Arbeit in der IT leichter von der Hand geht. Zudem sind Leute, die sich für einen Quereinstieg in der IT entscheiden, häufig mit Leidenschaft bei der Arbeit. Schließlich haben sie sich

Do #2: Details sind aufwendig

IT ist nicht gleich IT! Softwareentwicklung, IT-Projektmanagement oder IT-Support – Berufsfelder gibt es viele. Für einen gelungenen Quereinstieg müssen die Basics der Informationstechnik natürlich sitzen. Tief gehende Expertise brauchen Neueinsteiger jedoch nur auf dem Gebiet, das für ihre spätere Arbeit relevant ist. Daher gilt: Wer sich frühzeitig mit der Frage auseinandersetzt, welches Berufsziel er oder sie in der IT verfolgt, ist klar im Vorteil. Ist dieser Schritt gegangen, sollten sich alle folgenden Bemühungen, Übungssessions und mögliche Praktika nach dem neuen Karriereziel richten.

Do #3: Den Weg gemeinsam gehen

In der IT, besonders wenn es um das Coding geht, ist Feedback essenziell! Viele Leute haben bereits die Expertise, über die ein IT-Neuling gerne verfügen würde, und eignen sich hervorragend als Mentorinnen und Mentor. Hier ist es wichtig, sich von jemandem Unterstützung zu holen, bei dem auch das Zwischenmenschliche passt. Schließlich sollten Herausforderungen und auch mögliche Ängste auf dem Karriereweg offen angesprochen werden können. Doch nicht nur Kontakt zu Profis ist für den weiteren Lernprozess essenziell. Coding ist auch eine Teamaufgabe. Es wird immer Momente geben, an denen der Code nicht das Ergebnis liefert, für das er geschrieben wurde. Gemeinsam findet man die Fehlerquelle meist am schnellsten.

Don't #1: Vorhandene Skills unterschätzen

Wer quer einsteigt, hat einen entscheidenden Vorteil, der nicht missachtet werden sollte: Im bereits absolvierten Studium und in früheren Jobs wurden wertvolle Skills erlernt, die das Arbeiten in der IT erleichtern. So scheint zum Beispiel auf den ersten Blick ein Job im Kindergarten sehr weit vom Coden entfernt. Auf den zweiten Blick wird aber klar, dass die Zusammenarbeit mit Kindern einen vor allem in zwei Dingen schult: Geduld und Frustrationstoleranz. Kinder machen nicht immer, was man ihnen sagt, und das ist beim Coden genau dasselbe. Wenn man schon vor der Arbeit in der IT seine Resilienz gestärkt hat, bringen einen selbst hartnäckige Fehler im Code nicht aus der Ruhe.

Don't #2: Arbeitsort und Unternehmen vorab festlegen

In einigen Regionen Deutschlands sitzen besonders viele Firmen, die IT-Jobs anbieten. Hamburg führt die Liste an. Angehende Developer sollten sich jedoch immer bewusst sein, dass die deutschen Großstädte auch beliebte Orte zum Wohnen sind. Die Folge: Die Wohnungssuche gestaltet sich schwierig und die Konkurrenz ist besonders hoch. Wenn angehende IT-Profis sich für ihren Job jedoch nicht auf eine einzelne Stadt festlegen, bieten sich ihnen viel mehr Möglichkeiten für einen gelungenen Einstieg in die Branche. Informatik-Fähigkeiten sind fast überall gefragt und in manchen Bundesländern, wie etwa Mecklenburg-Vorpommern und Sachsen-Anhalt ist der Anteil an sozialversicherungspflichtigen IT-Fachkräften am geringsten. Hier lässt sich bei geringerer Konkurrenz wertvolle praktische Erfahrung sammeln.

bewusst für den neuen Werdegang entschieden und sich die notwendigen Fertigkeiten angeeignet. Doch damit der Übergang in die IT-Branche reibungslos funktioniert, gilt es einiges zu beachten.

Schnelle Anpassung nötig

Der Traum von einem auf Umwegen ergatterten Job in der IT-Branche muss kein Traum bleiben. Gerade wer beruflich zwischen zwei Stationen steht, kann sich über den Bildungsgutschein der Bundesagentur für Arbeit weiterbilden lassen. Es gibt beispielsweise den 14-monatigen Digital-Career-Institute-Kurs und weitere vom Bildungsgutschein abgedeckte Kurse, die auf der Website der Bundesagentur für Arbeit zu finden sind.

Für einen gelungenen Start müssen Quereinsteigende jedoch zeigen, dass sie diverse Soft- und Hardskills beherrschen. Dazu zählen unter anderen die Fähigkeit, komplexe Sachverhalte möglichst unkompliziert zu erläutern, in Teams an den Lösungen für Herausforderungen zu arbeiten und sich dem rasanten Wandel der Branche anzupassen. Mit dem richtigen Mindset sollte das jedoch nicht abschrecken. Schließlich offenbart die IT-Branche ein großes Potential für Technisch-Begeisterte, die sich persönlich und beruflich entfalten wollen und an spannenden Lernprozessen interessiert sind. (mdo)

Quellen

Alle Quellen zum Artikel finden sich unter ix.de/zxq8

Don’t #3: Keine Zeit zur Weiterentwicklung einplanen

Dieser Hinweis scheint erst einmal im Gegensatz zum ersten Do zu stehen. Jedoch ist die Quintessenz klar: Auch wenn es mit dem ersten Job in der IT geklappt hat, ist man noch lange kein Profi. Für angehende IT-Profis ist es deshalb essenziell, außerhalb der Arbeit Zeit zum Coden zu finden. Hierbei ist es beispielsweise hilfreich, sich von seinem Team Zusatzaufgaben mit nach Hause geben zu lassen, die einen weder über- noch unterfordern und an denen man in der Freizeit rumtüfteln kann. Optional können Quereinsteiger den Arbeitgeber vor Vertragsabschluss um einen Weiterentwicklungstag pro Woche bitten.

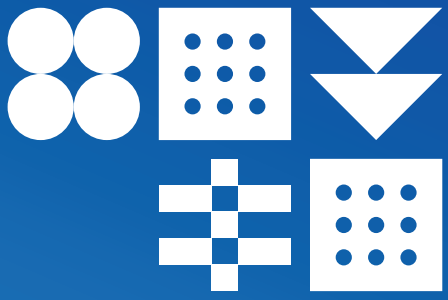
Hallie Barrows

ist Full Stack Web Developer bei LiveEO. Zuvor arbeitete sie bei MTV, studierte Applied Cultural Analysis, war Englischlehrerin und Office-Manager in einem Tech-Unternehmen.

> Rust für Neugierige

Stefan Baumgartner

Die Programmiersprache Rust ist stark im Aufwind. Rust-Neulinge lernen in diesem Artikel die Gründe dafür sowie Rusts einzigartige Fähigkeiten kennen.



Die Programmiersprache Rust wurde in diesem Jahr zum achten Mal in Folge zur beliebtesten in der jährlichen Stack-Overflow-Entwicklerumfrage gewählt und auch große Unternehmen wie Microsoft, AWS und Google schenken ihr Beachtung. Was macht diese Sprache so beliebt?

Eine moderne Programmiersprache

Rust wurde Anfang der 2010er Jahre von Graydon Hoare bei Mozilla im Rahmen der experimentellen Browser-Plattform Servo entwickelt. Das grundlegende Ziel war es, das Erstellen neuer Browserfunktionen zu ermöglichen und im gleichen Zug die Probleme der Speicherverwaltung und Speicherzuweisung von C und C++ zu beseitigen. Ähnlich wie die Programmiersprache Go sollte Rust performante Software ermöglichen und obendrein sollte die Entwicklung Spaß machen.

Dabei weist Rust Einflüsse aus vielen anderen Programmiersprachen auf. Die Syntax ähnelt stark C, aber es sind auch Spuren funktionaler Programmiersprachen vorhanden, insbesondere OCaml, worin die erste Version des Rust-Compilers geschrieben wurde. Listing 1 zeigt eine Rust-Funktion, die die Punkte für ein Wort in einem Scrabble-Spiel berechnet. Interessanterweise ist dieser Codeausschnitt genauso lesbar wie entsprechender Python-Code, enthält aber auch Bequemlichkeitsfunktionen wie Ausdrücke. Diese ermöglichen es, einen `match`-Ausdruck direkt neben der Addition von zwei Zahlen zu haben. Der `match`-Ausdruck selbst ermöglicht Pattern Matching über eine Reihe von gültigen Werten. In diesem Fall geht das Pattern Matching alle möglichen Werte des `char`-Werteraums durch. Das Schlüsselwort `match` verlangt von der Anwendung, alle möglichen Werte zu behandeln. Da es mehr Zeichen als die des Alphabets gibt, kann ein Standardfall mit dem Unterstrichzeichen alle verbleibenden Werte abfangen.

Um zu verstehen, warum Big-Tech-Unternehmen Rust verwenden, hilft ein genauerer Blick auf drei wesentliche Funktionen.

In a Nutshell

- Rust bietet eine einzigartige, moderne Syntax für eleganten Code.
- Die Programmiersprache setzt das Ownership-Konzept ein, um eine optimierte Speicherverwaltung zur Kompilierzeit sicherzustellen.
- Beispiele zeigen, wie Rust sowohl Zero-Cost-Abstraktionen als auch Ownership verwendet, um Datenkonflikte zu verhindern.

Speichersicherheit ohne Garbage Collection

Jahrzehntelang waren die am häufigsten verwendeten Programmiersprachen in zwei Lager unterteilt: Zum einen war es möglich, Software in Programmiersprachen mit automatischem Speichermanagement zu schreiben. Dazu gehörten Skriptsprachen wie Ruby, Python oder JavaScript, Virtual-Machine-basierte Sprachen wie C# oder Java und einige nativ kompilierte Sprachen wie Go. Zum anderen konnten Entwickler Speicher selbst verwalten, so wie es in C oder C++ möglich ist. Das erlaubte die Entwicklung sehr performanter Software, da keine Laufzeitumgebung erforderlich war, um über

Von der Vision zur Realität

Entwickeln Sie **Zukunft**

Mit modernen Technologien zur IT-Lösung.

Als international agierende Unternehmensgruppe mit weltweit mehr als 10.000 Mitarbeitenden bieten wir ausgezeichnete Karrierechancen in der Softwareentwicklung und IT-Beratung. Wir unterstützen Sie kontinuierlich beim Ausbau Ihrer Qualifikationen. Denn unser gemeinsamer Erfolg ist die Basis Ihres persönlichen Fortschritts. Überzeugen Sie sich selbst. Steigen Sie ein bei msg und zeigen Sie uns, was Sie können!

- Flexible Arbeitszeiten
- Möglichkeit für Homeoffice & mobiles Arbeiten
- Umfangreiche Weiterbildungs- & Gesundheitsangebote

karriere.msg.group

value – inspired by people

.msg

Listing 1: Berechnung eines Scrabble-Scores

```
pub fn score(word: &str) -> u64 {
    let mut score = 0;
    for ch in word.to_lowercase().chars() {
        score = score + match ch {
            'a' | 'e' | 'i' | 'o' | 'u' | 'l' | 'n' | 'r' | 's' | 't' => 1,
            'd' | 'g' => 2,
            'b' | 'c' | 'm' | 'p' => 3,
            'f' | 'h' | 'v' | 'w' | 'y' => 4,
            'k' => 5,
            'j' | 'x' => 8,
            'q' | 'z' => 10,
            _ => 0
        }
    }
    score
}
```

Listing 2: Ein Wert, auf den zwei Variablen zeigen. Dieses Beispiel wird nicht kompiliert.

```
fn main() {
    let numbers = vec![1, 2, 3, 4];
    let other_numbers = numbers;
    println!("{:?}", numbers);
}
```

Listing 3: Ein funktionierendes Beispiel mit Referenz

```
fn main() {
    let numbers = vec![1, 2, 3, 4];
    let other_numbers = &numbers;
    println!("{:?}", numbers);
}
```

Listing 4: Veränderlicher Zugriff auf numbers zur Änderung der Daten

```
fn append(vec: &mut Vec<u32>) {
    vec.push(5);
}
fn main() {
    let mut numbers = vec![1, 2, 3, 4];
    append(&mut numbers);
    println!("{:?}", numbers);
}
```

Garbage Collection den Speicher zu verwalten. Rust verfolgt jedoch eine andere Strategie: Es setzt auf eine Speicherzuweisung zur Kompilierzeit, die auf einem strengen Regelwerk basiert, dem Entwicklerinnen und Entwickler folgen müssen. Das erfordert etwas Lernaufwand, führt aber zu hundertprozentig speichersicheren Anwendungen. Und Speichersicherheit ist entscheidend: Die Verwendung von Speicher nach dessen Freigabe, doppelte Speicherfreigaben oder Pufferüberläufe und -überschreibungen führen zu Sicherheitslücken, die sich leicht ausnutzen lassen. Microsoft und Google haben in Windows und im Chrome-Projekt jeweils 70 Prozent aller schweren Sicherheitsprobleme als Speicherprobleme entlarvt (weitere Informationen dazu unter ix.de/zquk). Im Dezember 2022 waren bereits 20 Prozent des nativen Codes von Android in Rust geschrieben, und bis heute sind im Rust-Code von Android keine Speichersicherheitsprobleme aufgetreten. Rust erreicht die Speichersicherheit durch sein „Ownership and Borrowing“-System. Dessen grundlegendes Prinzip besagt, dass es nur einen Besitzer (Owner) von Daten geben kann. Wenn ein Wert einer Variable zugewiesen wird, wird diese Variable zum Besitzer. Wenn der Besitzer den Gültigkeitsbereich verlässt, wird der Speicher freigegeben. Die Eigentümerschaft (Ownership) lässt sich jedoch übertragen. Das Beispiel in Listing 2 zeigt ein typisches Szenario, das in einer Programmiersprache wie Java oder JavaScript kein Problem darstellen würde, aber in Rust zu Kompilierfehlern führt. In dem Moment, in dem der Variablen `other_numbers` der Wert von `numbers` zugewiesen wird, hat die Variable `numbers` keinen Wert mehr. `other_numbers` besitzt die Daten, und der Compiler gestattet Entwicklern nicht, das ursprüngliche Binding weiter zu verwenden. Allerdings bietet der Rust-Compiler genaue Informationen darüber, was geschehen ist, und bietet in der Regel auch Lösungsvorschläge.

In der Programmierung ist es oft notwendig, mehrere Zugriffe auf die gleichen Daten zu gestatten, weshalb es einschränkend sein kann, immer nur einen Besitzer zu haben. Daher erlaubt Rust das Ausleihen (Borrowing) von Daten durch das Erstellen von Referenzen, die auf die gleichen Daten zeigen. Das ursprüngliche Binding behält dabei weiterhin die Eigentümerschaft. Listing 3 zeigt, wie man Referenzen erstellt.

Das kaufmännische Und (&) gehört zum Typ. Es zeigt exakt, was die Funktion erwartet, beispielsweise wenn sie in Funktionssignaturen Referenzen deklariert. Dann überprüft der Rust-Compiler, ob die Variablen, die im Besitz der Daten sind, lange genug leben und nicht außerhalb des Gültigkeitsbereichs verschwinden, bevor die Referenzen dies tun. Wird ein Verweis verwendet, nachdem der Besitzer verworfen wurde, lässt Rust den Code nicht kompilieren. Das Verändern von Daten erfordert das Deklarieren einer Bindung als veränderlich (mutable, kurz `mut`). Außerdem sind veränderliche Referenzen zu erstellen, wenn andere Teile des Codes die Daten des Besitzers ändern sollen, wie in Listing 4 gezeigt. Es können mehrere geteilte Referenzen vorhanden sein, aber nur eine veränderliche Referenz. Der Grund liegt darin, dass bei der Änderung von Daten durch die veränderliche Referenz eine neue Allokation entstehen könnte, woduch die geteilten Referenzen auf einen falschen Wertebereich verweisen würden. Rust stellt sicher, dass kein Zeiger auf Daten zeigt, die ungültig werden könnten. Der wichtigste Aspekt von Ownership und Borrowing ist, dass im Code deutlich wird, was geschieht. Die Typen zei-

gen, was Entwickler erwarten können, und der Compiler versteht, wie der Speicher verwaltet werden muss.

Abstraktion ohne Overhead

Eine der Maximen von Rust ist es, Abstraktionen ohne Overhead zu ermöglichen. Das bedeutet, dass sich Programme elegant schreiben lassen, ohne dafür Geschwindigkeitsverluste hinzunehmen. Rust erreicht dieses Ziel durch Traits. Sie ähneln Interfaces aus anderen Programmiersprachen, da sie ebenfalls gemeinsames Verhalten über Typen hinweg definieren und abstrahieren. Anders als Interfaces lassen sich Traits auch für Typen implementieren, die nicht in der eigenen Projektstruktur beheimatet sind. Dadurch können Bibliotheken eine Kompatibilität mit dem breiteren Rust-Ökosystem von crates.io garantieren. Listing 5 zeigt eine Implementierung für Fibonacci-Zahlen im 128 Bit breiten vorzeichenlosen Integer-Typ. Die Struktur speichert eine aktuelle und eine nächste Zahl, und die Berechnung der Fibonacci-Zahlen erfolgt mithilfe eines Iterator-Trait. Die Definition des zugehörigen Typs als `Item` zeigt an, welche Ergebnisse zu erwarten sind. Mit jedem Aufruf von `next()` wird die nächste Fibonacci-Zahl berechnet und die aktuelle Zahl der Folge ausgegeben. Die Methode `next` gibt eine Option zurück, die ein Enum mit zwei möglichen Werten enthält: entweder einen Wert oder keinen Wert (`None`). Rust verlangt, dass Anwendungen beide möglichen Ergebnisse behandeln. Dazu dienen einerseits die `match`-Operation (siehe oben) und andererseits eingebaute Mechanismen wie der Iterator, bei dem `None` ein Zeichen zum Beenden des Iterationsvorgangs ist. Dadurch sind Iteratoren kompatibel mit `for`-Schleifen. Diese rufen die `next`-Methode auf, bis die Methode `None` zurückgibt. Eine Anwendung legt demnach fest: „Iteriere über diese Sammlung.“ Die Methode `checked_add` verhindert Überlauf. Sie gibt ebenfalls eine Option zurück, jedoch kommt hier eine weitere Funktion von Rust ins Spiel: der Fragezeichen-Operator. Mit ihm lässt sich wesentlich kürzerer Code schreiben, da der „schlechte“ Fall – `None` – sofort zurückgegeben und der „gute“ Fall – `Some` – entpackt wird. Durch diese Operation, als „Bubbling“ bezeichnet, muss man sich beim Programmieren keine Gedanken über Fehler oder fehlende Werte machen.

Der Rust-Compiler entfernt alle Abstraktionen und erstellt den bestmöglich optimierten Code. Wenn es das Szenario erlaubt, wird der Fibonacci-Iterator zur Kompilierzeit ausgeführt und das Ergebnis einfach in die Binärdatei geschrieben.

Furchtlose Nebenläufigkeit

In einer nebenläufigen Umgebung verhindern Rusts Ownership- und Borrowing-System sowie die Abstraktionen in Form von Traits und Typen das Auftreten von Data Races. Voraussetzung dafür sind zwei oder mehr Zeiger, die gleichzeitig auf die gleichen Daten zugreifen, wobei einer der Zeiger Schreibzugriff hat. Wenn keine Mechanismen vorhanden sind, um den Zugriff auf Daten zu synchronisieren, wird sich das Programm in manchen Situationen mit hoher Wahrscheinlichkeit undefiniert verhalten. In nebenläufigen Szenarien greifen üblicherweise mehrere Threads auf die gleichen Daten zu. Listing 6 erstellt einen Mutex für eine ganze Zahl, die von fünf parallelen Threads inkrementiert wird. Dieses Beispiel ist ebenfalls nicht kompilierbar, da der Be-

Listing 5: Erstellen von Fibonacci-Zahlen durch Implementierung des Iterator-Trait

```
struct Fibonacci {
    curr: u128,
    next: u128,
}
impl Iterator for Fibonacci {
    type Item = u128;
    fn next(&mut self) -> Option<Self::Item> {
        let new_next = self.curr.checked_add(self.next)?;
        self.curr = self.next;
        self.next = new_next;
        Some(self.curr)
    }
}
```

Listing 6: Der Besitz von counter wurde in der ersten Iteration übertragen.

```
fn main() {
    let counter = Mutex::new(0);
    let mut handles = vec![];
    for _ in 0..5 {
        let handle = thread::spawn(move || {
            let mut num = counter.lock().unwrap();
            *num += 1;
        });
        handles.push(handle);
    }

    for handle in handles {
        handle.join().unwrap();
    }
    println!("Result: {}", *counter.lock().unwrap());
}
```

Listing 7: Mit Atomic Reference Counters lässt sich gemeinsamer Zugriff auf den Mutex erlangen.

```
let counter = Arc::new(Mutex::new(0));
let mut handles = vec![];
for _ in 0..5 {
    let counter = Arc::clone(&amp;counter);
    let handle = thread::spawn(move || {
        let mut num = counter.lock().unwrap();
        *num += 1;
    });
    handles.push(handle);
}
```

sitz des Mutex an den ersten gestarteten Thread übertragen wurde. In der nächsten Iteration gibt es keine Zählvariable, die sich übernehmen lässt. In diesem Fall ist sicherzustellen, dass jeder Thread geteilten Zugriff auf den Mutex hat. Über eine Thread-sichere Kontrollstruktur `Arc`, kurz für Atomic Reference Counter, lassen sich mehrere Referenzen auf den Mutex erzeugen, die allerdings in einer Datenstruktur sind, die in den Besitz der Threads übergehen kann. Die `clone`-Methode erhöht den Referenzzähler um Eins. Sobald der Klon den Gültigkeitsbereich verlässt, wird der Zähler wieder niedriger. Ist der Zähler auf `0`, wird der Speicher freigegeben. Das ist eine sehr simple Form der Garbage Collection (Listing 7).

Sicher und performant

Rust bietet nicht nur eine moderne Syntax und Abstraktionen ohne Overhead, sondern glänzt vor allem durch sein Ownership- und Borrowing-System, das Speichersicherheit ohne Garbage Collection gewährleistet. Dieses System verhindert Speicherfehler wie Data Races und Pufferüberläufe und bietet gleichzeitig die Möglichkeit zur furchtlosen Nebenläufigkeit. Daher ist Rust für Entwicklerinnen und Entwickler, die sicheren und performanten Code schreiben möchten, eine ernsthafte Überlegung wert. (mai)

Quellen

Alle Links und die ungekürzte Fassung des Artikels unter: ix.de/zquk

Stefan Baumgartner

ist Softwarearchitekt und Entwickler. Er schreibt Artikel, Tutorials und Guides zu TypeScript, Rust, React und Softwareengineering und bietet über oida.dev Schulungen an.

Impressum We Are Developers!

Redaktion: iX
Telefon: 0511 5232-387, **E-Mail:** post@ix.de

Herausgeber: Ansgar Heise

Chefredakteur (verantwortlich für den Textteil): Dr. Oliver Diedrich

Konzeption und redaktionelle Leitung: Maika Möbus (mai@ix.de) -589

Redaktion: Madeleine Domogalla (mdo@ix.de) -590

Autoren dieser Ausgabe:
Hallie Barrows, Stefan Baumgartner, Maria Korneeva, Timo Zander

DTP-Produktion:
Lisa Hemmerling, heise medienwerk, Rostock

Korrektorat:
Lara Bögner, Marei Stade, heise medienwerk, Rostock

Titelbild:
Lisa Hemmerling, heise medienwerk, freepik

Verlag:
Heise Medien GmbH & Co. KG,
Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover;
Telefon: 0511 5352-0, Telefax: 0511 5352-129

Geschäftsführer:
Ansgar Heise, Beate Gerold

Mitglieder der Geschäftsleitung:
Jörg Mühle, Falko Ossmann

Anzeigenleitung (verantwortlich für den Anzeigenteil):
Michael Hanke (-167), E-Mail: michael.hanke@heise.de,
www.heise.de/mediadaten/ix

Leiter Vetrieb und Marketing:
André Lux

Druck:
Dierichs Druck + Media GmbH & Co. KG,
Frankfurter Straße 168, 34121 Kassel

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Kein Teil dieser Publikation darf ohne ausdrückliche schriftliche Genehmigung des Verlages verbreitet werden; das schließt ausdrücklich auch die Veröffentlichung auf Websites ein.

Printed in Germany

© Copyright by Heise Medien GmbH & Co. KG

Inserenten


Imagine The Future.
And Make Incredible Happen.
AKKODIS
Stell Dir vor, Du könntest jedes
Netzwerk zu einer smarten Umgebung machen.
Und dann tu es.
Bewirb Dich jetzt unter
hello-incredible.akkodis.com

IN AGILEN WORKSTREAMS DIE CLOUD-LÖSUNGEN VON MORGEN ENTWICKELN.

DARUM SIND WIR BEI DATEV.

Gemeinsam sichere Cloud-Lösungen und innovative Apps realisieren: Als Cloud-Entwicklerin oder -Entwickler erwarten dich bei DATEV vielfältige Aufgaben in einer agilen Innovations-Kultur. Informiere dich über freie Stellen und spannende Projekte bei einem der führenden IT-Dienstleister in Europa.

Valeria und Dominik, Cloud-Entwicklerin und -Entwickler bei DATEV

DATEV.DE/KARRIERE

DATEV

Zukunft gestalten. Gemeinsam.