entwickler magazin

Angriff auf verschlüsselte Mails >> S. 92

Deutschland € 8,50 Österreich € 9,00 Schweiz sFr 14,00

entwickler magazin

www.entwickler-magazin.de

November/Dezember 6.2018

GraphQL

Mit React Cheat Sheet

Forget (the) REST ?!

Kubernetes

Deep Dive ins Ressourcenmanagement

SERIE **Spieleklassiker**
Das eigene Space Invaders

SERIE **Web-Apps**
Modernes Webdesign mit CSS 3

4 194156 608503 06

Entwicklung einer Handschriftenerkennung auf Basis eines CNN

Was ist möglich mit Neuronalen Netzen?

Die Suche nach einem Ansatz zur Entwicklung einer Lösung für die Erkennung von Handschriften trieb ein ganzes Entwicklerteam um. In diesem Technologiebericht werden die Erfahrungen in diesem Projekt und die Gründe für die Nutzung eines Convolutional Neuronal Networks (CNN) wiedergegeben.

von Dr. Ralf-Thomas Pietsch

Seit dem 1. Januar 2018 müssen Banken und Sparkassen die Steueridentifikationsnummer (Steuer-ID) ihrer Kunden erfassen, wenn diese einen Konsumentenkredit in Höhe von 12 000 Euro und mehr in Anspruch nehmen. Für Finanzdienstleister bedeutet diese Gesetzesänderung unter anderem, dass sie die Steuer-IDs ihrer Bestandskunden anfordern und nachträglich in ihrem bestandsführenden System erfassen müssen. Im Zuge der Prozessimplementierung zur Nachforderung der Steuer-ID fragte ein langjähriger Kunde bei uns im Rahmen eines Scrum-Meetings an, wie die per Fax, E-Mail und Post eingehenden handschriftlich ausgefüllten Formulare automatisiert erfasst werden können. Zu diesem Zeitpunkt hatte man bereits Erfassungsformulare an seine Bestandskunden versendet und erhielt die ersten Rückläufer (**Abb. 1**). Um eine passende Lösung zu finden, sollten in einer Analysestory zunächst der Lösungsweg und die einzusetzenden Technologien geklärt werden.

Analysestory

In der Analysestory sondierten die Entwickler des Projekts den Markt nach möglichen Lösungen, die sich nahtlos in die bestehende Java-Anwendung integrieren ließen. Schnell war klar, dass OCR-Lösungen für die Erkennung von Handschriften nicht in Frage kamen, da sie nur gedruckten Text erkennen und umwandeln können. Als weiterer Ansatz wurde untersucht, inwiefern statistische Verfahren dazu geeignet sind, Handschriften zu erkennen. Auch dieser Ansatz erwies sich schnell als ungeeignet, da er zu aufwändig und zu ungenau ist. Weit größeren Erfolg versprach der Ansatz, eine Lösung auf Basis neuronaler Netze zu entwickeln.

Zwar hatten die in das Projekt involvierten Entwickler bis dato nur im Studium und auch dort nur wenige Berührungspunkte mit dieser KI-Technologie. Aber eine erste Literaturrecherche und die Beschreibungen der einschlägigen Open-Source-Bibliotheken versprachen, dass sich die Aufgabe auf diesem Wege lösen ließ. Aufgrund von Datenschutzbedenken schied die Verwendung von Cloud Services aus, eine Eigenentwicklung war daher unumgänglich.

Das Analyseergebnis stellten die Entwickler dem Kunden in einem weiteren Sprint-Meeting vor. Sie erhielten daraufhin das Mandat, einen Prototyp auf Basis eines geeigneten neuronalen Netzwerks zu bauen.

Bau des ersten Prototyps

Der erste Prototyp basierte auf einem einfachen Feed-Forward-Netzwerk. Da Java als Programmiersprache eingesetzt werden sollte, entschieden sich die Entwickler für die sehr gute und ausführlich dokumentierte Java-Bibliothek SNIPE [1]. Diese war nach Literaturwissen geeignet, ein schnelles und einfach benutzbares neuronales Netz zu implementieren. Aufgrund der geringen Einstiegshürde wurde SNIPE zur Umsetzung des Prototyps ausgewählt, auch wenn von Anfang an klar war, dass die Lizenz für den produktiven Einsatz nicht geeignet ist.

Als Trainingsdatensatz nutzten die Entwickler zunächst den für die Handschriftenerkennung weit verbreiteten MNIST-Datensatz [2]. Mit der Erkenntnis, dass die US-amerikanischen MNIST-Daten in Deutschland wenig hilfreich sein würden, denn die Schreibweisen für die Ziffern 1 und 7 unterscheiden sich erheblich von der deutschen Schreibweise (**Abb. 2**).

Da für europäische Schreibweisen kein alternativer Datensatz gefunden wurde, bauten die Softwareingenieure einen komplett neuen eigenen Datensatz auf, mit dem das Netz trainiert und getestet werden sollte. Dazu baten sie zahlreiche Kollegen aus dem gesamten Unternehmen um Schriftproben. Diese kamen der Bitte gerne

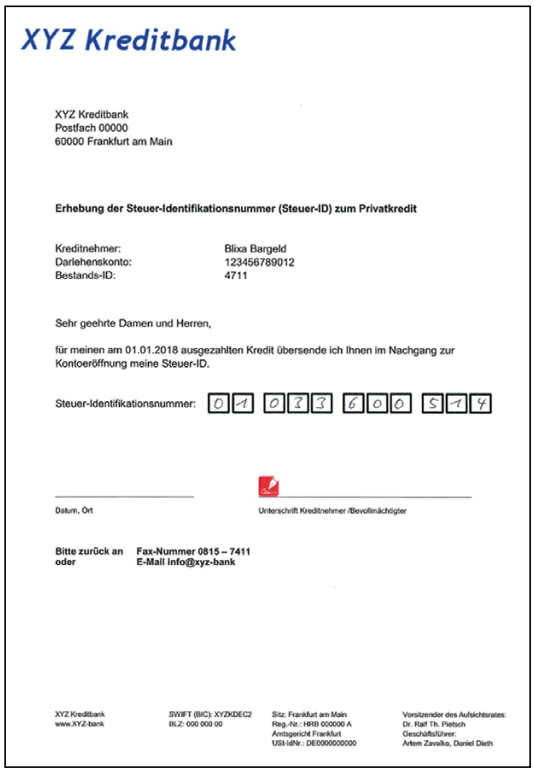

Abb. 1: Formularrückläufer

Abb. 2: Ziffern 1 und 7 machen den MNIST-Datensatz im europäischen Kontext unbrauchbar

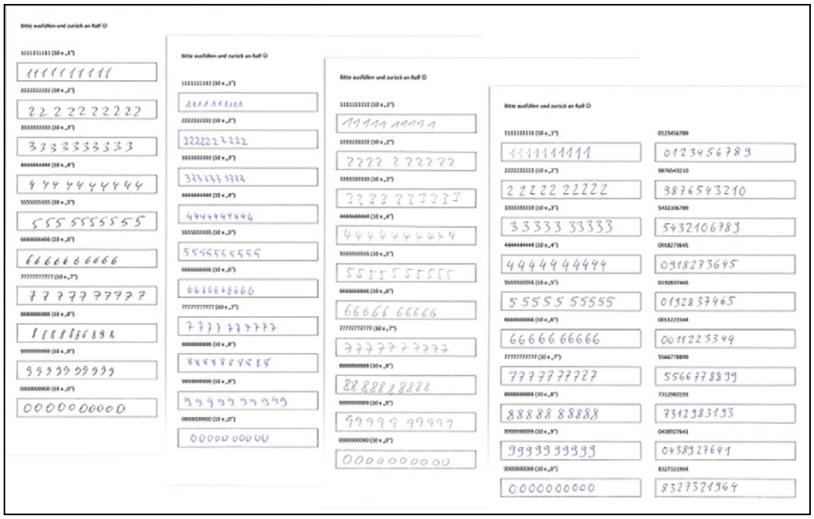

Abb. 3: Formulare zum Aufbau eines eigenen Trainings- und Testdatensatzes

nach (**Abb. 3**). Auf diese Weise wurde in kurzer Zeit ein großer Datensatz an Ziffern in europäischer Schreibweise aufgebaut, mit dem dann das Netz trainiert und getestet werden konnte.

Darüber hinaus bezogen die Entwickler auch einige Produktivdaten in das Training und Testen ein. Eine Herausforderung der Formulardaten war, dass diese in Kästchen eingetragen waren. Das mag optisch ansehnlich und zum Ausfüllen praktisch sein, ist aber für die Erkennung eine zusätzliche Herausforderung. Für das erste Training und Testen wurden die Ziffern aus den Formularen zunächst manuell separiert und extrahiert.

Auf Basis des neuen Datensatzes konnte die grundsätzliche Funktion des Ansatzes bestätigt werden. Mit der gesicherten Erkenntnis, einen Weg gefunden zu haben, mit der die Aufgabe zu bewältigen sei, gingen die Softwareexperten in das nächste Sprint-Meeting. Dabei berichteten sie aber auch, dass die Vorverarbeitung, also das Extrahieren der Ziffern, noch ungeklärt sei und voraussichtlich einen großen Teil der Aufwände verursachen würde.

Implementierung

In den nächsten Sprint wurde in Abstimmung mit dem Kunden eine erste Implementierungsstory eingeplant, die im Wesentlichen der Automatisierung der Vorverarbeitung und dem Test mit Daten aus der Produktion diente. Damit die Ziffern aus den Formularen nicht mehr manuell aus den Kästchen herausgeschnitten werden mussten, schrieben die Softwareexperten einen Algorithmus. Dieser arbeitet in mehreren Stufen. In einem ersten Schritt führt er eine Weichzeichnung des Eingangsbilds mittels Gauß-Filter [3] durch (**Abb. 4**, Schritt 2). Dieser Schritt ist erforderlich, damit das oft stark auftretende Rauschen bei unterschiedlichen Quellen gedämpft wird. Im nächsten Schritt werden mittels Canny-Edge-Algorithmus [4] die Kanten des Formulars extrahiert (**Abb. 4**, Schritt 3). In einem weiteren Bearbeitungsschritt wird das Bild ausgehend von den Eckpunkten einheitlich eingefärbt. Das Resultat ist ein Bild, in dem die Kästchen des Formulars eindeutig hervortreten (**Abb. 4**, Schritt 4). Durch die Berechnung eines horizontalen und vertikalen Histogramms können in diesem Bild die Positionen der Kästchen sehr genau bestimmt werden (**Abb. 4**, Schritt 5). Das ist die Voraussetzung, um die Ziffer anschließend extrahieren zu können.

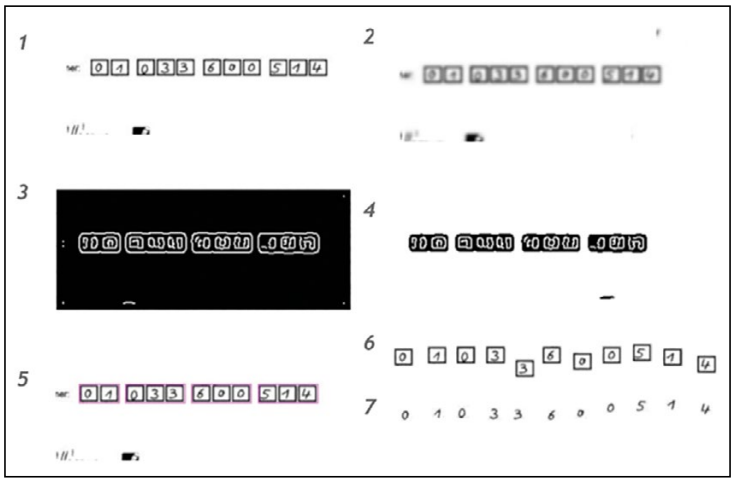

Abb. 4: Extraktion der Ziffern

Abb. 5: Erfassungsmaske für die Steuer-ID mit Vorbelegung der vom Netz erkannten Steuer-ID

Eine weitere Herausforderung lag darin, die Ausrichtung der Formulare zu korrigieren. Daher wurden die Ziffern zunächst mit Rahmen extrahiert (**Abb. 4**, Schritt 6) und in einem weiteren Schritt wurde der Rahmen in den Bildausschnitten entfernt (**Abb. 4**, Schritt 7). Damit waren die Voraussetzungen geschaffen, um das neuronale Netz automatisch mit den Echtdaten aus dem kundenspezifischen Formular zu versorgen.

Zur Erweiterung des selbst aufgebauten Datensatzes gaben weitere Kollegen aus dem Team der Lösungsfinder Schriftproben ab. Dazu nutzten sie diesmal Formulare, die den realen Gegebenheiten des kundenspezifischen Rückantwortformulars weitgehend entsprachen. Denn der Bearbeitungsvorgang und das Training des Netzes sollten unter realistischen Bedingungen erfolgen.

Als neuronales Netz setzten die Entwickler zunächst das Fast-Forward-Netz aus der Analysephase ein. Bei diesem Netztyp sind alle Neuronen untereinander verbunden, was der ältesten bekannten Struktur für neuronale Netze entspricht. Allerdings stellte sich bei diesem Netztyp auch nach der Optimierung der Vorverarbeitung kein Erfolg ein. Im Vergleich zu den ersten Tests gingen die Erkennungsraten bei der Auswertung der Echtdaten mit vergrößertem Datensatz stark nach unten, statt wie erwartet nach oben. Eine tiefergehende Analyse zeigte, dass das eingesetzte neuronale Netz nicht tolerant genug gegenüber Verschiebungen und leichten Verzerrungen der Ziffern war. Zwar funktionierte die Erkennung identisch positionierter Ziffern sehr gut, Lageänderungen innerhalb des Kästchens ließen allerdings die Erkennungsraten inakzeptabel schlecht werden. Die Ursache dafür liegt in der Struktur dieses neuronalen Netzes, sodass nach einem alternativen Ansatz gesucht wurde.

Listing 1: Ausschnitt aus der Maven-Konfiguration: Nur die benötigten Code-Binarys werden hinzugefügt.

```
<properties>
  <nd4j.version>0.9.1</nd4j.version>
  <dl4j.version>0.9.1</dl4j.version>

  <javacpp.platform.linux-x86_64>linux-x86_64</javacpp.platform.linux-x86_64>
  <javacpp.platform.windows-x86_64>windows-x86_64</javacpp.platform.windows-x86_64>
</properties>
...
<dependencies>
  ...
  <dependency>
    <groupId>org.nd4j</groupId>
    <artifactId>nd4j-native-platform</artifactId>
    <version>${nd4j.version}</version>
    <classifier>${javacpp.platform.windows-x86_64}</classifier>
  </dependency>

  <dependency>
    <groupId>org.nd4j</groupId>
    <artifactId>nd4j-native-platform</artifactId>
    <version>${nd4j.version}</version>
    <classifier>${javacpp.platform.linux-x86_64}</classifier>
  </dependency>
  ...
</dependencies>
```

Einsatz eines Convolutional Neuronal Networks

Nach einer erneuten Literaturrecherche und weiterer Vertiefung in das Thema entschieden sich die Entwickler für den Einsatz eines CNNs. Dieses für das Verarbeiten von Bildern entwickelte Netzwerk hat seine Stärken in der Erkennung von Mustern und kann daher auch unterschiedlich positionierte Ziffern zuverlässig erkennen. Allerdings schied die Standard-Library zur Implementierung von CNN, Tensorflow [5], aus: Sie basiert auf Python und war mit der Java-Prämisse nicht zu vereinbaren. Daher entschieden sich die Entwickler zunächst für den Einsatz des nativ in Java implementierten CNNs, Java CNN [6]. Diese Library bietet jedoch nur eine eingeschränkte Flexibilität bei der Konfiguration der Netzstruktur. Das führte dazu, dass die Entwickler trotz diverser Optimierungsbemühungen wie etwa einer Verfeinerung der Pixelauflösung nicht über Erkennungsraten in Höhe von 85 Prozent hinauskamen. Das klingt zwar recht gut, ist aber für den produktiven Einsatz bei Weitem nicht ausreichend.

Größere Flexibilität bei der Konfiguration der Netzstruktur versprach die plattformübergreifende Java-Programmbibliothek für künstliche Intelligenz, Deeplearning4j [7]. Diese wählten die Entwickler für einen weiteren Ansatz. Die Bibliothek war zwar aufwendiger zu konfigurieren, zeigte aber bereits nach verhältnismäßig wenigen strukturellen Änderungen des Netzes eine vielversprechende Erkennungsrate von bereits 88 Prozent. Hier eine weitere Verbesserung zu erzielen, schien auf Basis der bisherigen Erfahrungen und des inzwischen angesammelten Wissens sehr wahrscheinlich.

Im weiteren Sprintverlauf konnte die Erkennungsrate auf 93 Prozent gesteigert werden – ein Wert, der immer noch zu wenig für den vollautomatischen produktiven Einsatz ist. Aber er reichte aus, um den Sachbearbeitern ein unterstützendes Tool an die Hand zu geben,

das ihre Arbeit erheblich erleichtert. Daher wurde mit dem Kunden vereinbart, dieses Netz zunächst als Supportfunktion einzusetzen. Dazu wird das Eingabefeld für die Erfassung der Steuer-ID mit der durch das CNN erkannten Steuer-ID vorbelegt. Der Anwender kann dann direkt am Monitor überprüfen, ob die Erkennung korrekt durchgeführt wurde oder ob ein Nachbearbeitungsbedarf besteht (**Abb. 5**).

Eine Überraschung bot Deeplearning4j kurz vor dem ersten Produktionseinsatz dann allerdings dennoch: Die Programmbibliothek enthält den nativen Code für zahlreiche Plattformen. Das macht die Software zwar universell einsetzbar, erhöht jedoch den Speicherbedarf drastisch. Da man aber zur Entwicklung den Windows-Code und für alle anderen Systeme einschließlich des Produktivsystems nur den Linux-Code benötigte, entfernten die Entwickler vor Produktivstart durch Konfiguration des Java-Build-Tools Maven die überflüssigen Code-Binarys (Listing 1). Damit war die ausgelieferte WAR-Datei wieder in akzeptablen Größenordnungen.

Finale Story: Weiteres Training und Erhöhung der Erkennungsrate

Damit die Lösung im vollautomatischen Einsatz ihre Feuertaufe bestehen kann, musste das CNN weiter trainiert werden. Um eingehende Steuer-IDs korrekt erkennen zu können, war das Ziel, eine Erkennungsrate von über 95 Prozent in Produktion zu erreichen. Hierzu wurde vor allem ein noch größerer Trainingsdatensatz notwendig. Den Entwicklern stand inzwischen ein Testdatensatz mit knapp 60 000 eigenen Samples zur Verfügung, zudem fanden sie eine Möglichkeit, den MNIST-Datensatz doch noch zum Training einsetzen zu können. Durch einen selbstentwickelten Algorithmus wurden Modifikationen an den Ziffern 1 und 7 durchgeführt, die bei diesen die fehlenden Querstriche ergänzten und so die Ziffern in die deutsche Schreibweise transformierten (**Abb. 6**). Damit wurde der MNIST-Datensatz nutzbar und vergrößerte den Testdatensatz um weitere rund 68 000 Ziffern. Abgerundet wurde der Trainingsdatensatz noch mit etwa 20 000 Formularfeldausschnitten aus der Produktion.

Gerade die Daten aus der Produktion enthielten immer wieder Überraschungen, etwa schiefe Kästchen oder auch Handyfotos der Steueridentifikationsnummer als Grundlage für die Erkennung. Bei den Trainingsdurchläufen wurden daher immer wieder Anpassungen an der Vorverarbeitung zur Extraktion der Ziffern vorgenommen, mit denen die neu aufkommenden Herausforderungen gelöst wurden. Im Kern ging es aber auch darum, die optimale Struktur des CNNs zu finden – mit dem Erfolg, dass die Erkennungsrate inzwischen bei hervorragenden 98 Prozent liegt.

Fazit

Für die Erkennung von Handschriften sind CNNs eine sehr gute Basis. Allerdings bedarf es hierzulande eines

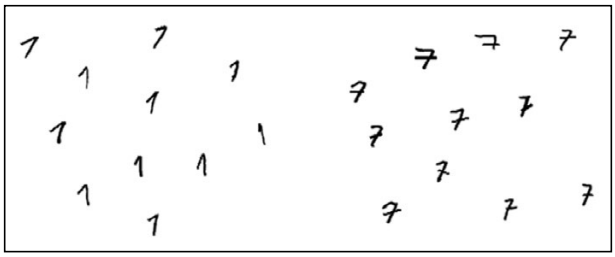

Abb. 6: Europäisierte Ziffern 1 und 7 aus dem MNIST-Datensatz

geeigneten Datensatzes mit europäischen Schriften. Der MNIST-Datensatz eignet sich aufgrund der unterschiedlichen Schreibweisen der Ziffern 1 und 7 nicht zur Erkennung in europäischer Schreibweise.

Die Leistung von neuronalen Netzwerken steht und fällt mit der Menge an Trainingsdaten, aber auch mit der Qualität der Eingangsdaten. Daher ist auch eine gute Vorverarbeitung der Daten wesentlich, um das neuronale Netz optimal anzusteuern. Spezifische Herausforderungen, wie in diesem Fall die Anlieferung von Ziffern in Kästchen, müssen sauber gelöst werden, bevor ein geeignetes Netz trainiert wird.

Ein weiteres Learning ist aber auch, dass neuronale Netzwerke kein Hexenwerk sind. Letztendlich leben sie nur in dem Raum, der beim Training vorgegeben wird. Das hier beschriebene Netz lebt in einer Welt von ausschließlich zehn Ziffern und wird jedes dem Netz vorgelegte Bild in eine dieser zehn Kategorien einteilen. Beim Konfigurieren und Trainieren des Netzes gilt es daher auch, stets den Einsatzzweck im Auge zu behalten.

Bei der Verwendung von KI-Tools im produktiven Umfeld mussten von den Entwicklern Herausforderungen gemeistert werden, die in einem KI-Tutorial nicht behandelt werden. Die Handschrifterkennung im Kundenauftrag zu entwickeln, war ein spannendes und lehrreiches Unterfangen, bei dem alle Beteiligten mit großer Begeisterung dabei waren.

Dr. Ralf-Thomas Pietsch ist Softwarearchitekt bei der Finanz Informatik Solutions Plus GmbH.

Links & Literatur

[1] http://www.dkriesel.com/en/tech/snipe

[2] http://yann.lecun.com/exdb/mnist/

[3] https://de.wikipedia.org/wiki/Weichzeichnen#Gau%C3%9Fscher_Weichzeichner

[4] https://de.wikipedia.org/wiki/Canny-Algorithmus

[5] https://www.tensorflow.org/

[6] https://github.com/ratopi/JavaCNN

[7] https://deeplearning4j.org/

finanz informatik solutions plus

Wir suchen ab sofort für Frankfurt, Fellbach und Berlin eine/n

Software Ingenieur/in Java SE/EE

Sie übernehmen

+ die professionelle Umsetzung von kundenindividuellen Anforderungen in Projekten über alle Phasen des Entwicklungsprozesses hinweg
+ die Entwicklung und Integration von modernen IT-Anwendungen

Sie haben

+ ein abgeschlossenes (Fach-)Hochschulstudium mit Fachrichtung (Wirtschafts-) Informatik, Mathematik oder Physik studiert und bereits umfangreiche Berufs- und Projekterfahrung in der Anwendungsentwicklung gemacht
+ sehr gute Kenntnisse in der Entwicklung mit aktuellen Java SE/EE Technologien
+ fundierte und praktische Kenntnisse moderner Frameworks und Werkzeuge (z. B. JSF, Spring, AngularJS, JPA, Hibernate, JSON, XML, Eclipse/IntelliJ)
+ gute Kenntnis gängiger Anwendungsarchitekturen (C/S, Web, n-Tier)
+ sicheren Umgang in der Anwendung von bekannten Vorgehensmodellen / Methoden (z. B. Agile, Scrum, UML, Patterns, Refactoring)
+ Kenntnisse im Einsatz von Applikationsservern, Datenbanksystemen oder Middleware-Produkten wie z. B. IBM DB2, Oracle, Jboss oder IBM WebSphere
+ idealerweise Erfahrungen mit der Konzeption und Implementierung von Schnittstellen zur Integration neuer und bestehender IT-Systeme in Systemlandschaften (z.B. SOA, ESB, EAI, Messaging)
+ analytisches Denkvermögen
+ ausgeprägte Kunden- und Serviceorientierung
+ Teamfähigkeit und Kommunikationsstärke

Wir bieten

+ teamorientiertes Arbeiten, in das sich jeder optimal einbringen kann
+ Campus-Mentalität in Kombination mit der Sicherheit des Sparkassen-Verbunds
+ individuelle Lösungen für die Bedürfnisse unserer Mitarbeiterinnen und Mitarbeiter
+ vielfältige Karrieremöglichkeiten und persönliche Weiterentwicklung auf Basis der eigenen Fähigkeiten und Wünsche
+ ein attraktives Gehalt und Zusatzleistungen wie Betriebsrestaurants und kostenfreie Getränke (Tee, Kaffeespezialitäten)

Wir sind

seit über 20 Jahren erfolgreich – denn wir wachsen kontinuierlich. Unsere Passion: durch innovative Softwarelösungen Digitalisierung in der Sparkassen-Finanzgruppe zu gestalten. Dafür suchen wir Mitglieder für unser „Team der Lösungsfinder“, die Lust auf herausfordernde IT-Projekte, Eigenverantwortung und eine individuelle Weiterentwicklung haben.

Mehr Informationen zu uns, unserem Unternehmen und weitere Stellenangebote finden Sie auf:

www.team-der-loesungsfinder.de

Ihre Bewerbung

übermitteln Sie bitte per E-Mail an *entwickler@team-der-loesungsfinder.de* oder über unser Karriereportal unter *www.team-der-loesungsfinder.de/karriere*. Wir freuen uns darauf!

Finanz Informatik Solutions Plus GmbH
Theodor-Heuss-Allee 90
60486 Frankfurt a. M.
(069) 829749-0