Hochschule für Technik und Wirtschaft Dresden

Fakultät Informatik/Mathematik

Projektbericht

Thema:

Erstellen eines anpassbaren Systems zur Erzeugung von neuronalen Netzen für die Erkennung vordefinierter Merkmale in Satellitenbildern

Vorgelegt von: Sebastian Mischke
Dorfstraße 8, 01257 Dresden
geb. am 09.11.1995 in Dresden
Matrikelnummer: 37612

Studiengang: Medieninformatik

newwebtechnology

Unternehmen: New Web Technology GmbH

Adresse: Bahnhofstraße 35
40764 Langenfeld

Telefon: +49(0)2173-33 43 444
Telefax: +49(0)2173-33 43 100
E-Mail: kontakt@new-web-technology.de

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des Praxisprojektes des Studiengangs „Bachelor Medieninformatik“ an der HTW Dresden wurde vom 06.03. bis 31.05.2017 bei der Firma New Web Technology GmbH in Langenfeld eine Anwendung zur Satellitenbildanalyse mittels Deeplearning erstellt. Zuerst wurde ein Konzept basierend auf den Anforderungen der Firma angefertigt. Anschließend begann der Entwicklungsprozess bei welchem das System sukzessive um neue Funktionen erweitert wurde. Ziel des Projektes ist die Umsetzung eines Frameworks zur Extraktion von marketing-relevanten Informationen aus geographischen Daten.

2 Problemstellung

Die Kernaufgabe des Projektes war es, ein System zu entwickeln, welches neuronale Netze erstellt und trainiert. Bei den Eingabedaten des Netzes handelt es sich um Satellitenbilder von Häusern. Darauf basierend soll das Netz dann Merkmale des Hauses erkennen können. Welche Häuser dafür ausgewählt sind, wird in Form einer Adressliste vorgegeben.

Bei der Erstellung des Systems hatte die einfache Anpassbarkeit der einzelnen Komponenten eine hohe Priorität. Da die zu klassifizierenden Häuser über eine Liste in das System gegeben werden, ist das Hinzufügen und Entfernen eben dieser bereits möglich, allerdings sollen auch die Informationen, die den jeweiligen Häusern zugeordnet sind, verändert werden können. Die gewünschten Merkmale, die das System erkennen soll, müssen ohne großen Aufwand in ihrer Form veränderbar sein. Die Netzstruktur soll anpassbar sein, sich in seiner Ein- und Ausgabeschicht aber automatisch auf das Format der Ein- und Ausgabedaten einstellen, um Aufwand zu reduzieren. Außerdem soll es verschiedene Ergebnisvisualisierungen geben, welche je nach Merkmalsart zu- und abgeschaltet werden können.

Das System soll vollständig implementiert werden und dient als Grundlage für eine spätere Anwendung. Die Implementierung eines Nutzungs-Interfaces soll hierbei keine Rolle spielen.

3 Projektumsetzung

3.1 Programmiersprache

Eine fundamentale Entscheidungen vor dem Beginn eines Softwareprojektes ist die Wahl einer geeigneten Programmiersprache. Dafür waren zwei Kernpunkte zu betrachten. Zum einen sollte eine Sprache Verwendung finden, die bereits firmenintern genutzt wird, um die neue Software reibungslos in das bestehende Firmensystem einzufügen. Zum anderen muss die Sprache für die Programmierung von Deeplearning-Anwendungen nutzbar sein. Ein zusätzlicher Pluspunkt wäre es, wenn bereits Erfahrung mit der zu verwendenden Sprache vorliegt, da so die Einarbeitungszeit verkürzt würde. Unter Betrachtung dieser Punkte fiel die Wahl letztendlich auf die Programmiersprache „Python“.

3.2 Adressdatei laden

Die Adressliste liegt in Form einer CSV-Datei vor und beinhaltet neben einer Haus-ID, welche firmenintern für die eindeutige Zuordnung der Häuser verwendet wird,

noch die Adresse mit Straße, Hausnummer, Postleitzahl und Stadt, sowie die geographischen Koordinaten mit Längen- und Breitengrad. Zuerst ist die Funktionalität notwendig, CSV-Datei einzulesen und in ein Python-internes Format umzuwandeln, damit die darin enthaltenen Informationen im System verwendet werden können. Dazu wurde eine eigenständige Klasse implementiert, die sowohl CSV-Dateien lesen, als auch zweidimensionale Arrays in eine CSV-Datei schreiben kann.

3.3 Visualisierung der Daten

Um einen besseren Überblick über die vorliegende Adressliste zu bekommen, sollte diese ähnlich einer 2D-Karte visualisiert werden. Hierzu wurden für jeden Eintrag die geographischen Koordinaten aus der Adressliste ausgelesen und separat in einer zweiten Liste gespeichert. Diese zweite Liste wird anschließend genutzt, um mit der Python-Visualisierungs-Bibliothek „matplotlib“ ein Streudiagramm zu erstellen.

3.4 Deeplearning Framework

Da der Aufwand zur Erstellung eines eigenen Deeplearning-Frameworks zu hoch ist, soll ein bereits existierendes Framework verwendet werden. Deshalb wurde zuerst eine Liste von möglichen Frameworks sowie deren Vor- und Nachteile zusammengestellt. Nach anschließendem Abgleich mit den gegebenen Anforderungen, fiel die Wahl auf das Framework „Keras”.

3.5 Einarbeitung in Keras

Um Erfahrung mit der Verwendung von „Keras“ zu sammeln, wurde für die Einarbeitung zunächst mit der Erstellung eines einfachen neuronalen Netzes begonnen. Dieses sollte anhand der geographischen Koordinaten die Postleitzahl eines Hauses erkennen. Die dafür benötigten Informationen gingen aus der Adressliste hervor.

Nach dem Training des Netzes wurde der komplette Eingabedatensatz vom Netz klassifiziert und die Ergebnisse zusammen mit den Eingabedaten in einer neuen Liste gespeichert. Die Visualisierung dieser Liste geschah anschließend wieder als Streudiagramm. Die Klassifizierungsergebnisse wurden dabei jeweils verschiedenen Farbwert zugeordnet.

3.6 Satellitenbilder laden

Es ist nicht immer möglich, auf einen vollständigen Bilddatensatz von Häusern zugreifen zu können. Deshalb sollte eine eigene Lösung entwickelt werden, um das benötigte Bildmaterial zu erzeugen. Hierfür wurden verschiedene Online-Kartendienste miteinander verglichen, die Entscheidung fiel auf Google Maps. Nach der Einarbeitung in die API von Google Maps, erfolgte die Implementierung einer Funktion, mit deren Hilfe die benötigten Informationen korrekt einsetzt werden können. Die Funktion für jeden Eintrag der Adressliste ausgeführt um die geladenen Bilder für die spätere Verwendung lokal zu speichern.

3.7 Neuronales Netz

Neben dem Satellitenbild können zu Häusern noch zusätzliche Meta-Daten existieren. Diese bieten die Möglichkeit der Verbesserung der Ergebnisgenauigkeit des Netzes. Deshalb muss die Netzstruktur besonders an das Eingabeformat der Da-

ten angepasst werden. Dazu wird das Netz aus mehreren Teilnetzen zusammengesetzt. Dies ermöglicht mehrere Eingabeschichten und damit die Nutzung verschiedener Eingabeformate. Das erste Teilnetz soll die Bilddaten analysieren, weshalb es die typischen Struktur eines Convolutional Neural Network, bestehend aus Faltungsschichten am Anfang und vollvernetzte Schichten am Ende, erhält. Das zweite Teilnetz soll die Meta-Daten analysieren und besteht somit ausschließlich aus vollvernetzte Schichten. Das abschließende Teilnetz soll die beiden Eingabe-Netze verbinden und basierend auf deren Ergebnissen die Ausgabeschicht berechnen.

3.8 Trainingsprozess

Um den Ablauf des Trainingsprozesses des Netzes besser analysieren zu können, ist eine Visualisierungen der Trainingsstatistiken, wie zum Beispiel die Klassifizierungsgenauigkeit, notwendig. Dafür werden die benötigten Informationen bereits während des Trainingsprozesses gespeichert. Im Anschluss daran werden alle auf diese Weise erfassten Daten in eine eigens dafür angefertigte Funktion gegeben, welche daraus abermals mit Hilfe von matplotlib ein Kurvendiagramm über die Trainingsepochen erstellt.

3.9 Speichern und Laden von trainierten Netzen

Der Trainingsvorgang eines neuronalen Netzes ist sehr zeitaufwändig. Um also trainierte Netze zu verwenden oder einen Trainingsprozess unterbrechen und fortsetzen zu können, muss das Speichern und Laden eines Netzes möglich sein. Dafür werden die Struktur des Netzes und die Gewichte aller Schichten in zwei separaten Dateien gespeichert. Dies ermöglicht es, ein vollständiges trainiertes Netz zu laden, aber auch, eine Netzstruktur für einen anderen Trainingsprozess wiederzuverwenden.

3.10 Anpassen der Bilddaten

Die vorliegenden Bilddaten entsprechen nicht immer den Anforderungen des Trainingsprozesses. Deshalb soll es möglich sein, die Bilder vor Beginn des Trainingsprozesses zu bearbeiten. Dazu wurde eine Funktion erstellt, welche ein lokal vorliegendes Bild lädt und nach besonderen Parametern editiert. So kann ein Bild skaliert, beschnitten oder in seinem Seitenverhältnis geändert werden. Das Speichern des Ergebnisbildes zur Wiederverwendung ist möglich, aber sehr speicheraufwändig, da aus jeder Parameterkombination ein neues Bild hervorgeht. Darum wird das Ergebnisbild direkt in den Trainingsprozess gegeben, wodurch ein langfristiges Speichern nicht notwendig ist.

3.11 Erzeugen von Trainingsvorgaben

Innerhalb der Trainingsdaten müssen die gewünschten Merkmale für jedes Haus als Trainingsvorgabe vorliegen. Sollte es sich dabei um Merkmale handeln, zu denen noch keine Trainingsvorgaben vorliegen, müssen diese händisch erzeugt werden. Dafür wurde eine eigenständige Anwendung entwickelt, welche dem Nutzer das Erzeugen der Trainingsvorgaben erleichtert. Um bereits implementierte Funktionen wiederverwenden zu können, wurde die Anwendung in Python mit dem GUI-Toolkit „Tk“ umgesetzt.

In der Anwendung wird ein Satellitenbild, sowie eine Liste von möglichen Merkmalen angezeigt. Der Nutzer kann die Merkmale auswählen, die für das gesamte Bild beziehungsweise das gezeigte Haus im Bild zutreffen. Diese Zuordnung wird dann in einer CSV-Datei gespeichert. Ist der Nutzer mit der Merkmalszuordnung für ein Bild fertig, wird der nächste Adresslisteneintrag gezeigt.

3.12 GPS-Koordinaten oder Adressen als Positionseingabe

Da es von Vorteil wäre, wenn die Liste von Häusern nicht zwingend deren geographischen Koordinaten enthalten muss, soll es nun noch zusätzlich die Möglichkeit geben, stattdessen die Adressen der Häuser zu benutzen. Dafür müssen die Adresstücke, welche in der Adressliste in Straße, Hausnummer, Postleitzahl und Stadt aufgeteilt sind, zusammengesetzt und in ein Format gebracht werden, welches von der Google Maps API genutzt werden kann. Dadurch können alle folgenden Schritte unabhängig von der Art der Positionsangabe wie bisher erfolgen.

Sollten sich in der Adressliste sowohl Koordinaten als auch die Adresse der Häuser befinden, so kann der Funktion, welche die Bilder herunterlädt, die zu benutzende Positionsangabe als Parameter übergeben werden.

3.13 Zusammenfügen von Bildern

Teilweise besteht der Bilddatensatz aus Bildern, die nur eine kleine Fläche abdecken. Dies könnte bei der Analyse von größeren Häusern oder Grundstücken ein Problem darstellen. Außerdem erhält das Netz in diesem Fall keine Informationen über die nähere Umgebung, wodurch Ungenauigkeiten hinsichtlich der Klassifizierungsquote zustande kommen können. Deshalb wurde eine Funktion implementiert, welche Satellitenbilder zusammensetzt, um damit ein Bild zu erzeugen, welches eine größere Oberfläche zeigt. Das Ursprungsbild, welches direkt die Hausadresse oder -koordinate beinhaltet, befindet sich dabei im Zentrum, damit das zu betrachtende Haus immer mittig im Ergebnisbild liegt.

3.14 Speichern und Laden von Trainingsdaten in SQL

Da sich das Speichern von Adresslisten inklusive der zugehörigen Meta-Daten als CSV-Datei langfristig betrachtet als unpraktisch erweisen könnte, sollen die benötigten Informationen aus einer SQL-Datenbank geladen werden. Dafür wurde eine Klasse erstellt, welche SQL-Tabellen und CSV-Dateien ineinander konvertieren kann. Somit bleibt die Option bestehen, CSV-Dateien weiterhin als Dateneingabe für das Netz zu verwenden.

Zudem können die Klassifizierungsergebnisse des Netzes, welche bisher ebenfalls in Form einer CSV-Datei erzeugt wurden, in der Datenbank gespeichert werden. Wird die Arbeit mit mehrerer Computern gleichzeitig durchgeführt, können so verschiedene Bilder parallel klassifiziert und in der Datenbank zusammengeführt werden.

4 Fazit

Während meines Praxisprojektes wurde ein Systems zur Erzeugung von neuronalen Netzen für die Analyse von Satellitenbildern entwickelt. Dabei erfüllt es alle mit der Firma erarbeiteten Vorgaben und ist bereit für eine Integration in das firmeninterne System.

Aufgrund der Modularität und der daraus folgenden Anpassungsfähigkeit des Systems wäre es auch in nicht-marketing spezifischen Anwendungen einsetzbar. So könnte im Bereich der Stadtplanung die automatische Erkennung von Häusermerkmalen die Darstellung des Ist-Zustandes einer Stadt verbessern und vereinfachen. Außerdem können die zu erkennenden Merkmale auch häuser-unabhängig sein, wodurch auch ein Einsatz in der Landschaftsanalyse möglich wäre.