Hochschule für Technik und Wirtschaft Dresden

Fakultät Informatik/Mathematik

Bachelorarbeit

Thema:

Merkmalserkennung von Gebäuden und Grundstücken in Satellitenbildern mittels Deeplearning

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

Studiengang: Medieninformatik

Externer Betreuer: Ann-Christin Storms
New Web Technology GmbH

Betreuender Prüfer: Prof. Dr. Marco Block-Berlitz

Zweitgutachter: Prof. Dr. Hans-Joachim Böhme

Letzte Änderung: 17. Juli 2017

Abgabetermin: 27. Juli 2017

Inhaltsverzeichnis

Verzeichnis verwendeter Abkürzungen

Nomenclature

API Eine Programmierschnittstelle, genauer Schnittstelle zur Anwendungsprogrammierung, häufig nur kurz API genannt (englisch application programming interface, wörtlich ‚Anwendungsprogrammierschnittstelle'), ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.

MLP A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that is not linearly separable.

Abbildungsverzeichnis

Zusammenfassung

Inhalt der Arbeit

1 Einleitung und Motivation

„Ein Bild sagt mehr als tausend Worte“. Doch gilt dies für alle Bilder? Und was sind das für Worte? Mit dem Aufkommen von Deeplearning und der hochwertige Möglichkeiten der Bildanalyse eines Convolutional Neural Networks ist die Wissenschaft der Feature-Detection in den letzten Jahren weit vorangeschritten.

2 Konkretisierung der Aufgabenstellung

Das Ziel der Arbeit soll es sein, die Möglichkeiten und Grenzen der Merkmalserkennung in Satelittenbildern zu erforschen.

3 Stand der Technik

Im Projekt „PlaNet - Photo Geolocation with Convolutional Neural Networks“ konnte bereits gezeigt werden, dass ein Convolutional Neural Network in der Lage ist, den Aufnahmeort eines Fotos auf durchschnittlich 1200km genau zu bestimmen.

4 Verwendete Technik

Die Möglichkeiten der Technik sind vielzählig. Von der Programmiersprache, über die verwendete Netzstruktur, die Bibliotheken und Module, bis hin zu Interfacespezifika und Visualisierungen.

4.1 Python

Die verwendete Programmiersprache ist Python.

Python besitzt viele Vorteile gegenüber anderen Programmiersprachen. Es ist leicht zu lesen, wodurch man die Logik und den Ablauf eines Programmes schneller erkennt. Es ist unabhängig vom Betriebssystem. Die große Anzahl an opensource Bibliotheken ermöglicht eine leichte Umsetzung von nahezu jedem Programm. Außerdem besitzt Python die größte Anzahl an Deeplearning-Bibliotheken, und wird auch von großen Unternehmen und Einrichtungen in diesem Bereich untersützt und vorangetrieben, wie `TensorFlow` von Google, `CNTK` von Microsoft oder `cuDNN` von NVIDIA.

4.2 ConvNet

4.3 Keras - theano

Als Deeplearning Framework wurde Keras verwendet. Es benötigt als Grundlage entweder `TensorFlow`, `CNTK` oder `Theano`. Dadurch ist die Installation aufwändiger als bei Bibliotheken, die keine zusätzlichen Voraussetzungen besitzen, jedoch erlaubt es aber auch die Implementierung und damit die Vorteile von allen dreien zu benutzen.

4.4 matplotlib

Nahezu alle Visualisierungen wurden mit matplotlib, einer Python Bibliothek für graphische Darstellungen, gemacht. Damit lassen sich nicht nur Grafiken erzeugen, direkt anzeigen oder abspeichern, sondern auch in anderen GUI-Bibliotheken wie `GTK+`, `Qt`, `wxWidgets` oder `Tk` verwenden.

4.5 Tk

Da Python keine eigene GUI besitzt, muss zum Programmieren eines Interfaces eine zusätzliche GUI-Bibliothek wie Tk verwendet werden.

5 Gesamtplan

Das System zur Merkmalserkennung besteht aus drei Kernbausteinen. (siehe Abbildung 1)

- Den Satellitenbildern, die als Informationsgrundlage und Input-Daten für den Klassifizierer genutzt werden
- Dem künstlichen neuronalen Netz, welches die Input-Daten analysiert und klassifiziert
- Der Trainingsvorgabe, welche zum Trainieren des Netzes benutzt wird.

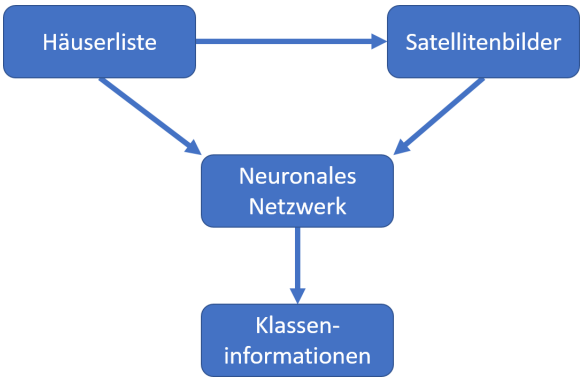

Abbildung 1: Datenflussdiagramm

5.1 Satellitenbilder

Die Satellitenbilder bilden die Grundlage der Input-Daten für das Netz. Da eine eigenständige Erzeugung einen viel zu hohen Aufwand bedeuten würde, werden stattdessen Bilder verwendet, die mittels der Google Static Maps API erzeugt wurden. Dafür sind entweder die GPS-Koordinaten oder die Adresse des Grundstückes notwendig.

Die Bilder werden lokal gespeichert und später zum Verarbeiten durch das Netz in 2D-Arrays für den Rot-, Blau, und Grün-Channel umgewandelt.

5.2 Trainingsdaten

Trainingsdaten geben dem Netz den gewünschten Output vor und werden dementsprechend zu dessem Training verwendet.

Das Erzeugen der Trainingsdaten ist in der Regel ein aufwändiger Prozess und stark vom gewünschten zu erkennenden Merkmal abhängig. Neben der großen Menge an Trainingsdaten, die zum Trainieren eines Netzes benötigt werden, sind manche Merkmale nicht leicht mit dem menschlichen Auge zu erkennen.

5.3 Künstliches neuronales Netz

Das neuronale Netz besteht es drei Teilen. Einem CNN, welches die Satellitenbilder als Input-Daten bekommt und diese auswertet, einem MLP, welches zusätzliche Metadaten des Gebäudes bzw. Grundstückes als Input-Daten bekommt, und einem MLP, welches die Output-Daten des CNN und des MLP als Input-Daten bekommt.

[5][2]

```
1 def main():
```

Abbildung 2: Beispiel eines Satellitenbildes

```
2     # Load csv
3     X_train, Y_train = load_csv("
          data.csv")
4     # Create net
5     model = create_net(X_train,
          Y_train)
6     # Train net
7     history = model.fit(x=X_train, y
          =Y_train)
8     # Save net
9     save_model(model, "structure.
          json", "weights.h5")
```

5.4 Training des Netzes

Nachdem die Trainingsdaten geladen und das neuronale Netz erstellt wurden, müssen die Gewichte des Netzes an das gewünschte Ziel angepasst werden. Dies geschieht, indem ein Optimierer, in unserem Fall ein Stochastischer Gradientenabstieg[1], in mehreren Epochen die Klassifizierung des Netzes an die Trainingsvorgabe anpasst.

Nach jeder Trainingsepoche werden die Validierungsdaten vom Netz klassifiziert und die Ergebnisse mit der Trainingsvorlage verglichen. Der sich daraus ergebende Genauigkeitswert wird dann mit dem bisherigen Bestwert verglichen. Sollte die neue Klassifizierung besser sein als der bisherige Spitzenwert, so werden die Gewichte des Netzes gespeichert und der neue Bestwert wird sich ge-

merkt. Dadurch soll ein Overfitting des Netzes vermieden werden.

6 Experimente

6.1 x-, y-Koordinaten

In Anlehnung an das Projekt von PlaNet[6] besteht der erste Versuch daraus, die GPS-Koordinaten, das heißt sowohl Längen-, als auch Breitengrad, eines Satellitenbildes zu bestimmen, ohne dafür zusätzliche Metadaten für das Bild zu besitzen.

Die Trainingsdaten sind 6337 Städte und Orte in Deutschland.[7]

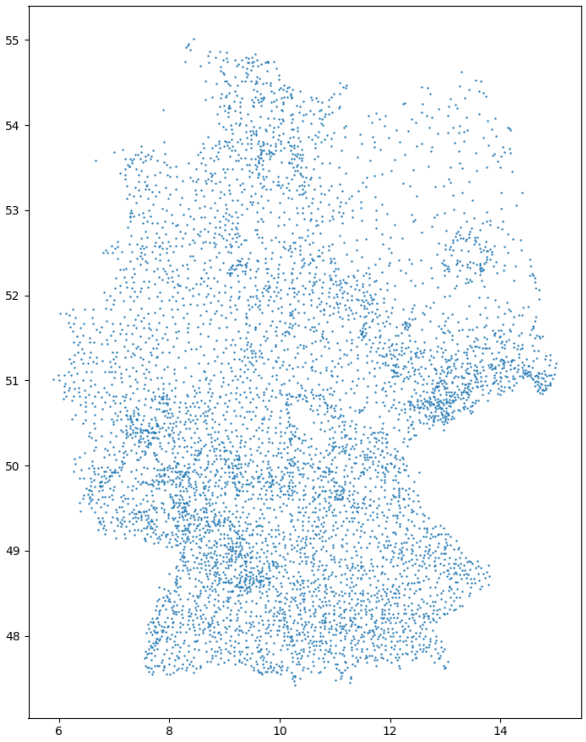

Abbildung 3: Städte und Orte in Deutschland

6.2 Postleitzahl

Im Gegensatz zu Längen- und Breitengrad, sind Postleitzahlen in Deutschland nur bedingt geordnet. Es gilt zwar das Prinzip, dass zwei zahlenmäßig nah beieinander liegende Postleitzahlen sich häufig auch örtlich nah sind, jedoch ist diese Zuordnung nicht konsistent. Deswegen sollte es für ein Netz auch entsprechend schwerer sein, nur anhand eines Satellitenbildes eines Hauses die Postleitzahlen von eben diesem zu bestimmen.

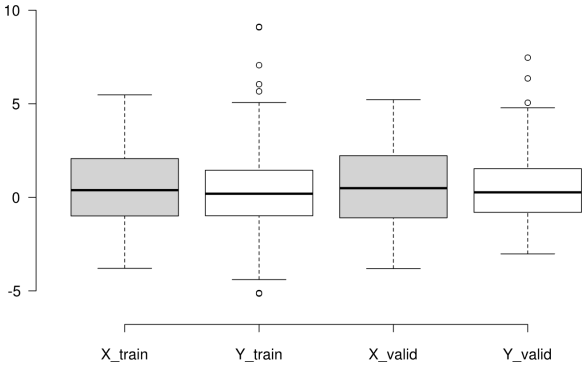

Abbildung 4: Abweichungen der Netzvorhersagen vom Vorgabewert

Die Trainingdaten bestehen aus 64390 Häusern in Dresden.

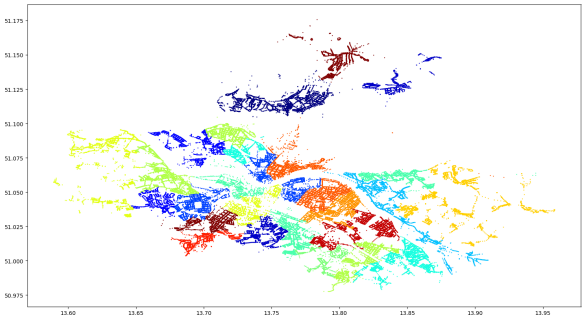

Abbildung 5: Postleitzahlenbereiche in Dresden

6.3 Solaranlagen

6.4 Schulen

Das dritte Experiment wird die Klassifizierung von Häusern beinhalten. Dafür soll ein Netz trainiert werden, welches zwischen Wohnhaus und Schule unterscheiden kann. Als Datengrundlage wird hierfür eine Liste mit allen 204 Schulen in Dresden [3], sowie 204 zufällig ausgewählte Wohnhäusern, ebenfalls aus dem Raum Dresden.

7 Ergebnisse

8 Ausblick

9 Danksagung

Literatur

[1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[2] Francois Chollet. Building powerful image classification models using very little data. *Retrieved December*, 13:2016, 2016.

[3] Landeshauptstadt Dresden. Schulen in dresden.

[4] Sächsisches Staatsministerium für Kultus. Sächsische schuldatenbank.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[7] Dipl.-Med. Frank Wiegleb. Geografische längen- und breitengrade deutscher städte und gemeinden.

	X_train	Y_train	X_valid	Y_valid
Upper whisker	5.48	5.07	5.22	4.78
3rd quartile	2.07	1.45	2.22	1.53
Median	0.38	0.19	0.49	0.27
1st quartile	-1.00	-0.99	-1.10	-0.80
Lower whisker	-3.80	-4.40	-3.81	-3.02
Nr. of data points	5703	5703	634	634

Tabelle 1: Ergebnisse des x-y-Koordinatenversuchs