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

Betreuender Prüfer: Prof. Dr. Marco Block-Berlitz

Zweitgutachter: Prof. Dr. Hans-Joachim Böhme

Externer Betreuer: Ann-Christin Storms
New Web Technology GmbH

Letzte Änderung: 20. Juli 2017

Abgabetermin: 27. Juli 2017

Inhaltsverzeichnis

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.

CNN Convolutional Neural Network
Ein Convolutional Neural Network (CNN oder ConvNet), zu Deutsch etwa „faltendes neurales Netzwerk", ist ein feedforward künstliches neuronales Netz. Es handelt sich um ein von biologischen Prozessen inspiriertes Konzept im Bereich des maschinellen Lernens (engl. machine learning). Convolutional Neural Networks finden Anwendung in zahlreichen, modernen Technologien der künstlichen Intelligenz, vornehmlich bei der maschinellen Verarbeitung von Bild- oder Audiodaten.

LSTM Long short-term memory
Long short-term memory (LSTM) bezeichnet im maschinellen Lernen einen Typ von rekurrenten neuronalen Netzen. Sie wurden 1997 von Sepp Hochreiter und Jürgen Schmidhuber in einer Veröffentlichung vorgestellt. Im Gegensatz zu traditionellen rekurrenten Netzen können LSTMs längere zeitlich verzögerte Effekte z. B. für Klassifizierungsaufgaben berücksichtigen und effektiv trainiert werden.

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

Ziel dieser Arbeit ist es, ein System zu entwickeln, mit dem Klassifikationen und Merkmalserkennungen in Satellitenbildern mittels Deeplearning leicht umzusetzen sind.

3 Verwandte Arbeiten

3.1 ImageNet Large-Scale Visual Recognition Challenge

Die ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)[1] ist wohl einer der bekanntesten Wettbewerbe im Bereich der automatisierten Bildklassifizierung. Ihr Aufbau ist dabei an die Pascal Visual Object Classes Challenge[2] angelehnt. Die Trainingsdaten bestehen aus Bilddaten und der Klasseninformation zu einem Objekt, welches jeweils im Bild vorhanden ist. Dies stammt alles aus der ImageNet-Datenbank[3].

Mit SuperVision[4], einem Deep Convolutional Neural Network, gewannen Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton die ILSVRC-2012. Ihr Netz beinhaltet fünf Konvolutionsschichten, manche gefolgt von einer Max-pooling Schicht, und abschließend drei vollvernetzten Schichten mit der finalen 1000-Neuronen Schicht, die das Ergebnis der Klassifizierungen darstellt. Damit konnte eine Top-5-Fehlerquote, die die relative Häufigkeit angibt, in der das korrekte Ergebnis nicht in den ersten fünf Prädiktionen liegt, von 16,4% erreicht werden. Die Fehlerquote für das korrekte Erkennen der richtigen Klasse lag dabei bei 38,1%.

VGG16 und VGG19[5], die Gewinner des ILSVRC-2014 im Bereich „Klassifikation und Lokalisation“ benutzten einen ähnlichen Aufbau von Konvolutions-, Max-pooling- und vollvernetzten Schichten, nur das sich die Tiefe des Netzes deutlich vergrößert hat. Statt fünf besitzt das Netz nun 13 beziehungsweise 16 Konvolutionsschichten. Damit konnte die Top-5-Fehlerquote auf 7,2% und die Top-1-Fehlerquote auf 24,4% gesenkt werden.

3.2 PlaNet

Im Projekt „PlaNet - Photo Geolocation with Convolutional Neural Networks“[6] wurde versucht, den Aufnahmeort eines Fotos lediglich anhand des Fotos zu erkennen. Dafür wurde die Weltkugel in mehrere Zellen eingeteilt und die Bilder jeweils der Zelle zugeordnet, in der ihr tatsächlicher Aufnahmeort liegt. Damit konnte in etwa 30% das korrekte Land erkannt werden und die Genauigkeitsrate für Städte lag bei rund 10%. Das reichte, um bisherige Modelle, welche noch auf handeingetragenen Merkmalen basierten, mit einen signifikanten Abstand zu überbieten. Außerdem war man in der Lage in einem Experiment beim Onlinespiel GeoGuessr gegen erfahrene Menschen zu gewinnen.

Das System wurde anschließend noch verbessert, indem das bestehende CNN mit einem LSTM kombiniert wurde, um anstatt eines einzelnen Bildes ein ganzes Bilderalbum zu klassifizieren.

3.3 isprs

Die Internationale Gesellschaft für Photogrammetrie und Fernerkundung hat in ihrem 2D Semantic Labeling Wettbewerb

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 und die große Anzahl an opensource Bibliotheken ermöglicht eine leichte Umsetzung zu nahezu jeder Problematik. Außerdem besitzt Python die größte Anzahl an Deeplearning-Bibliotheken, und wird auch von großen Unternehmen und Einrichtungen in diesem Bereich unterstützt 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[7] 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 in unserem Fall `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.

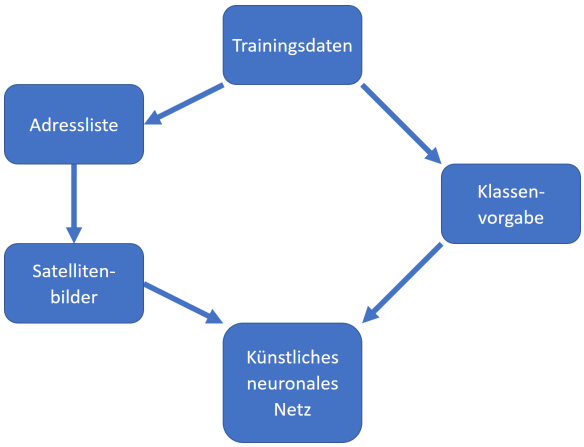

Abbildung 1: Datenflussdiagramm

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 wird eine URL mittels Algorithmus 2 erzeugt, die die Einstellungen des zu erzeugenden Bildes beinhaltet. Zum einen der Mittelpunkt in Form einer Adresse oder einer GPS-Koordinate. Dies ist die Kerninformation und das sich später einzige ändernde Merkmal innerhalb einer Bilderreihe für einen Trainingsprozess. Dann die Größe des Bildes angegeben in Pixeln. Hierbei liegt das Maximum bei 640x640, deswegen wird dies auch die Standardeinstellung. Anschließend die Zoom-Stufe, die Größe eines Objektes innerhalb eines Bildes angibt. Die offizielle Google Maps API Dokumentation[1] gibt eine Tabelle für die Detailebenen an, die man basierend auf diesem Parameter erwarten kann. Außerdem gibt es den Parameter des Kartentypen. Da es lediglich um die Informationsgewinnung aus Satellitenaufnahmen geht, ist hier immer satellite ausgewählt.

Die mittels dieser URL erzeugten Bilder werden lokal gespeichert und basierend auf ihren Parametern benannt. Dadurch könnten sie vor dem Trainingsprozess noch zugeschnitten oder auf eine andere Größe skaliert werden. Außerdem verhindert es ein mehrfaches Herunterladen des gleichen Bildes.

[1] https://developers.google.com/maps/documentation/javascript/tutorial#vergrerungsstufen

Algorithmus 1 Hauptprozedur

```
# Load data
images, teach = load_csv("data.csv")

# Create net
model = create_net(images, teach)

# Create checkpointer
cp = ModelCheckpoint(
    filepath='model.hdf5'
)

# Train the net
model.fit(images, teach,
    batch_size=batchsize,
    epochs=epochs,
    validation_split=split,
    callbacks=[cp]
)
```

Algorithmus 2 Erzeugen der URL für die Google Maps API

```
url = "http://maps.google.com/maps/
    api/staticmap"
if centermode == "xy":
    # Latitude and Longitude
    url += "?center=" + y + "," + x
elif centermode == "address":
    # Address
    url += "?center=" + address
url += "&size=" + w + "x" + h
url += "&zoom=" + zoom
url += "&maptype=" + maptype
```

5.2 Trainingsdaten

Trainingsdaten geben dem Netz den gewünschten Output vor und werden dementsprechend zu dessen 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

Die Grundstruktur des Netzes basiert auf den Vorgaben von SuperVision[4] und VGG16[5]

Abbildung 2: Beispiel eines Satellitenbildes

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.

Building powerful image classification models using very little data[8]

5.4 Training des Netzes

Da das Netz keine Farbbilder als Eingabe akzeptieren kann, müssen die Bilder vorher in drei 2D-Arrays für den Rot-, Grün- und Blaukanal umgewandelt werden.

Nachdem nun die Trainingsdaten geladen und das neuronale Netz erstellt wurde, müssen die Gewichte des Netzes an das gewünschte Ziel angepasst werden. Dies geschieht, indem ein Optimierer, in unserem Fall ein Stochastischer Gradientenabstieg[9], 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 gemerkt. Dadurch soll ein Overfitting des Netzes vermieden werden.

5.5 Bewertung des Netzes

NOTE: mittlere absolute Fehler, binäre crossentropie etc.

6 Experimente

6.1 GPS-Koordinaten

Der erste Versuch besteht daraus, die GPS-Koordinaten, das heißt sowohl Längen-, als auch Breitengrad, eines Satellitenbildes zu bestimmen. Dafür sollen keine zusätzliche Metadaten über das Bild bekannt sein. Die verfügbaren Daten zum Trainieren des Netzes basieren dabei auf den 6337 Einträgen der Datenbank "Geografische Längen- und Breitengrade deutscher Städte und Gemeinden"[2], deren Verteilung in Abbildung 3 dargestellt ist.

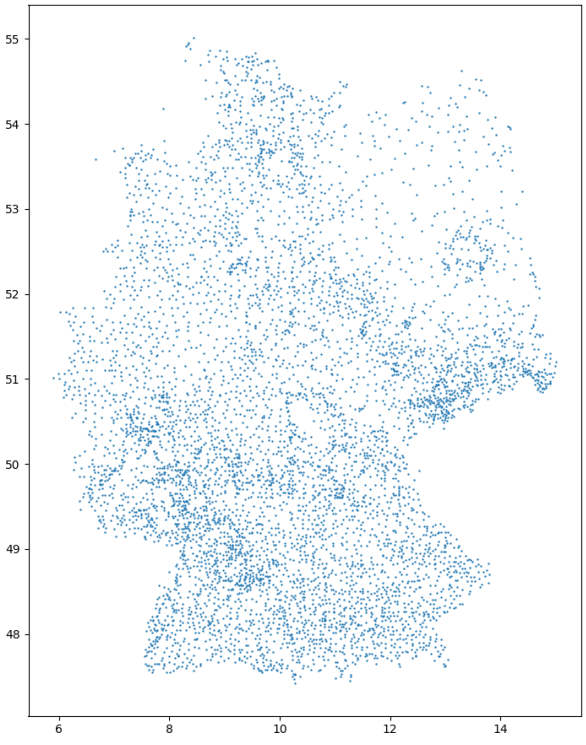

Abbildung 3: Deutsche Städte und Gemeinden

[2]http://www.fwiegleb.de/geodat.htm

Bei einer Trainings-Validierungs-Teilung von 10% ergibt das 5703 Trainingsdaten und 634 Validierungsdaten.

Die mittleren absoluten Fehler der Trainings- und Validierungsdaten lagen bei *2,6* und *1,5*.

Die genauere Verteilung der Ergebnisse ist Abbildung 4 und Tabelle 1 zu entnehmen.

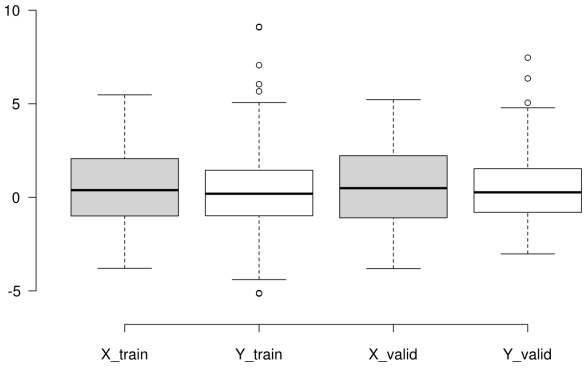

Abbildung 4: Abweichungen der Netzvorhersagen vom Vorgabewert

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.

Die Trainingsdaten bestehen aus 64390 Häusern in Dresden, deren räumliche Anordnung sowie Zuordnung zu einer Postleitzahl in Abbildung 5 dargestellt ist.

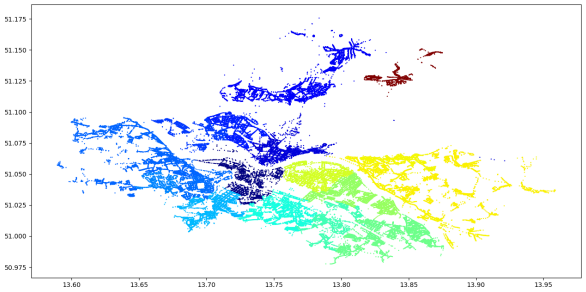

Abbildung 5: Postleitzahlenbereiche in Dresden

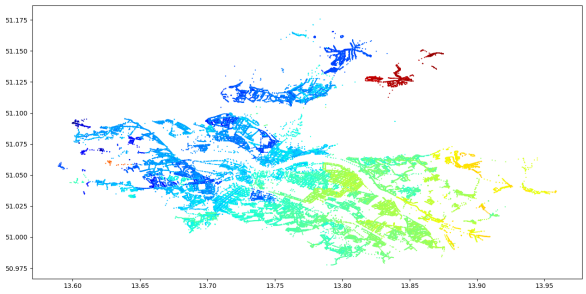

Abbildung 6: Ergebnisse der Klassifizierung des Netzes

6.3 Schulen - Wohnhäuser

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.

6.4 Schularten

7 Ergebnisse

8 Ausblick

9 Danksagung

[3] https://www.dresden.de/de/leben/schulen/schulen-in-dresden.php

Literatur

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec 2015. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

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

[6] T. Weyand, I. Kostrikov, and J. Philbin, "Planet-photo geolocation with convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 37–55.

[7] F. Chollet, "Keras," https://github. com/fchollet/keras, 2015.

[8] ——, "Building powerful image classification models using very little data," *Retrieved December*, vol. 13, p. 2016, 2016.

[9] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

	X_train	Y_train	X_valid	Y_valid
Maximum	5.48	5.07	5.22	4.78
Oberes Quartil	2.07	1.45	2.22	1.53
Median	0.38	0.19	0.49	0.27
Unteres Quartil	-1.00	-0.99	-1.10	-0.80
Minimum	-3.80	-4.40	-3.81	-3.02
Anzahl Datenpunkte	5703	5703	634	634

Tabelle 1: Ergebnisse des x-y-Koordinatenversuchs