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: 25. Juli 2017

Abgabetermin: 27. Juli 2017

Inhaltsverzeichnis

Symbolverzeichnis

API	Application programming interface Programmierschnittstelle zum Anbinden eines externen Systems an ein Programm.
Batchsize	Anzahl an Bildern, die für eine Anpassung der Gewichte des Netzes genutzt werden.
CNN	Convolutional Neural Network Netzstruktur eines künstlichen neuronalen Netzes, bestehend aus einer Aneinanderreihung von Konvolutionsschichten
CSV	Comma-separated values Textformat, in dem einzelne Daten einer Zeile durch Kommata oder Semikola getrennt sind.
Framework	Programmiergerüst, welches einen Entwicklungsrahmen für Softwareentwickler bereit stellt.
Konvolutionsschicht	Schichtart, bei der die Ausgabedaten durch eine Faltung der Eingabedaten erzeugt wird.
Pooling-Schicht	Schichtart, bei der die Ausgabemenge kleiner als die Eingabemenge ist, indem überflüssige Informationen verworfen werden.
URL	Uniform Resource Locator Adresse einer Webseite oder einer Datei im Internet
Überanpassung	Prozess, in dem ein neuronalen Netz die Klassifizierung der Trainingsdaten auswendig lernt, anstatt zu verallgemeinern.

Abbildungsverzeichnis

Algorithmenverzeichnis

Tabellenverzeichnis

Zusammenfassung

In dieser Arbeit wird ein System entwickelt, das sich basierend auf einer Adressliste automatisiert alle dazugehörigen Satellitenbilder herunterlädt, ein an die Ein- und Ausgabedaten angepasstes künstliches neuronales Netz erstellt und dieses anschließend für seine Aufgabe im Bereich der Klassifizierung beziehungsweise der Merkmalserkennung trainiert. Die Qualität der Parameter des Netzes werden dabei stetig evaluiert durch dem Netz bisher unbekannte Validierungsdaten und bei Verbesserungen für eine spätere Verwendung in Anwendungen inklusive Netzarchitektur lokal gespeichert. Abschließend wird das System in verschiedenen Experimenten für die Erkennung von geographischen Koordinaten, Postleitzahlen und Schularten verwendet und deren Ergebnisse ausgewertet.

1 Einleitung und Motivation

„Deeplearning bezeichnet Computermodelle, welche aus mehreren Verarbeitungsschichten bestehen und lernen, Daten auf verschieden Abstraktionsstufen darzustellen. Dieses Konzept brachte dramatische Verbesserungen in der Spracherkennung, der visuellen Objekterkennung und in anderen Fachgebieten, wie der Genomik. Deeplearning erkennt hochkomplexe Strukturen in große Datensätzen, indem es mittels dem Backpropagation-Lernalgorithmus erkennt, wie eine Maschine seine internen Parameter für die Abbildung der Schichten basierend auf den jeweils vorhergegangenen Schichten verändern muss. Convolutional Neural Networks erzielten Durchbrüche in der Verarbeitung von Bildern, Videos, Sprach- und anderen Audioaufnahmen, während rekurrente neuronales Netze eher sequentielle Daten wie Text oder Sprache beleuchten.“[LBH15]

Dieser Erfolg von Deeplearning wird vorangetrieben durch die ansteigende Rechnerleistung, welche die benötigte Zeit für den Trainingsvorgang verkürzt und größere Netzstrukturen benutzbar macht, sowie der stetig wachsenden Menge an verfügbaren Trainingsdaten. Ein Bereich der Bildverarbeitung, der sich bei zweiterem in den letzten Jahren enorm verbessert hat, ist die Auswertung von Satellitenbildern. Die Anzahl an Quellen für frei verfügbare qualitativ hochwertige Satellitenaufnahmen hat stark zugenommen.

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. Das Hauptkriterium dafür ist, dass sich das System schnell und einfach an unterschiedliche Eingabeformate der Daten anpassen lassen kann. Außerdem soll eine größtmögliche Automatisierung angestrebt werden, was bedeutet, dass der Aufwand bei der Benutzung des Systems minimal zu halten ist. Ebenso sollten die Resultate beziehungsweise das trainierte Netz sich mühelos in andere Anwendungen integrieren lassen.

3 Verwandte Arbeiten

3.1 ImageNet Large-Scale Visual Recognition Challenge

Die ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)[RDS+15] ist wohl einer der bekanntesten Wettbewerbe im Bereich der automatisierten Bildklassifizierung. Ihr Aufbau ist dabei an die Pascal Visual Object Classes Challenge[EVGW+10] 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[DDS+09].

Mit SuperVision[KSH12], 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[SZ14], 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 Multi-column Deep Neural Networks

Eine Alternative zum linearen neuronalen Netzwerk sind die mehrspaltigen neuronalen Netzwerke[CMS12]. Die Architektur besteht dabei aus mehreren herkömmlichen neuronalen Netzen, welche parallel zueinander angeordnet sind. Dabei ist anzumerken, dass diese mehreren Netze zunächst einzeln trainiert werden und erst für die spätere Anwendung zusammengeschlossen werden.

Auch dann noch verarbeiten die Teilnetze die Eingabedaten unabhängig voneinander. Für das abschließende Ergebnis des mehrspaltigen Netzes bildet man den Durchschnittswert der Einzelergebnisse der Teilnetze. Dies erbrachte in den durchgeführten Experimenten eine relative Verbesserung der Klassifizierungsgenauigkeit zwischen 26% und 72% im Vergleich zu den bisher besten Ergebnissen.

3.3 PlaNet

Im Projekt „PlaNet - Photo Geolocation with Convolutional Neural Networks“[WKP16] 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 Convolutional Neural Network mit einem Long short-term memory (LSTM)[HS97] kombiniert wurde, um anstatt eines einzelnen Bildes ein ganzes Bilderalbum zu klassifizieren.

3.4 isprs 2D Semantic Labeling Contest

Die Internationale Gesellschaft für Photogrammetrie und Fernerkundung trägt einen 2D Semantic Labeling Wettbewerb[fPuF] für Luftaufnahmen aus. Darin müssen anhand von Orthofotos, das heißt verzerrungsfreien und maßstabsgetreuen Abbildung der Erdoberfläche, welche durch Luft- und Satellitenaufnahmen erstellt wurden, sowie digitalen Höhenmodellen bestimmte Objekte und Kategorien erkannt werden. Bei diesen Kategorien handelt es sich um Flächenversiegelungen, also Straßen, Parkplätze und ähnliches, Gebäude, niedrige Vegetation, Bäume, Fahrzeuge und die gesonderte Klasse für Sonstiges, wozu Wasser oder andere Flächen gehören, die sich den bisherigen Gruppen nicht zuordnen ließen. Da es sich hier um die Klassifizierung größerer Bereiche handelt, werden die Klasseninformationen farbkodiert in Bildern gespeichert, die der identischen lokalen Zuordnung entsprechen, wie die dazugehörigen Orthofotos.

4 Verwendete Technik

4.1 Python

Als Programmiersprache wurde Python verwendet, da sie viele Vorteile gegenüber anderen Programmiersprachen besitzt. Sie 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 open-source Bibliotheken ermöglicht eine leichte Umsetzung zu nahezu jeder Problematik.

Außerdem besitzt Python die größte Anzahl an Deeplearning-Bibliotheken, und wird von großen Unternehmen und Einrichtungen in diesem Bereich unterstützt und vorangetrieben, wie unter anderem TensorFlow[ABC+16] von Google, CNTK[SA16] von Microsoft oder cuDNN[CWV+14] von NVIDIA.

Der primäre Nachteil von Python ist wohl der Geschwindigkeitsunterschied im Vergleich zu kompilierenden Sprachen wie C oder C++. Da in Python aber bereits Deeplearning-Implementierungen existieren, die Grafikkarten zum berechnen verwenden können, sollte dieser Makel nur eine untergeordnete Rolle spielen.

4.2 Tkinter

Python besitzt kein eigenes Interface. Deshalb muss zum Programmieren einer grafischen Anwendung eine zusätzliche GUI-Bibliothek genommen werden. Hierfür wurde Tkinter[Gra00] verwendet, da es in jeder Standardinstallation von Python enthalten ist. Dadurch werden für die Ausführung keine Zusatzmodule benötigt. Außerdem bietet es die Möglichkeit, die Anwendung auf unterschiedlichen Plattformen auszuführen, ohne dafür den Quellcode anpassen zu müssen.

Da Tkinter ein Schichtenkonzept verwendet und auf Tk[1] basiert, ist auch hier wieder einmal ein Nachteil die Problematik der Geschwindigkeit im Gegensatz zu nativen GUI-Bibliotheken.

4.3 Convolutional Neural Networks

Wie SuperVision[KSH12] und VGG16[SZ14] zeigen konnten, ist die Architektur eines Convolutional Neural Networks mit abschließenden vollvernetzten Schichten am besten geeignet, um Bilddaten auszuwerten. Und selbst die mehrspaltigen neuronalen Netze[CMS12] haben diese Grundstruktur intern beibehalten. Deshalb besitzt das Standardnetz des Systems einen ähnlichen Aufbau.

[1] http://www.tkdocs.com/

4.4 Keras

Als Deeplearning Framework wurde die Python Bibliotheken namens Keras[Cho15] verwendet. Im Gegensatz zu anderen Frameworks besitzt es die Möglichkeit als Grundlage entweder TensorFlow[ABC+16], CNTK[SA16] oder Theano[BBL+11] zu verwenden. Dies erlaubt es, die Implementierungen und damit auch die Vorteile aller drei zu nutzen. Außerdem ist Keras inzwischen als erste High-Level API in TensorFlow enthalten, was wiederum den Installationsaufwand verringert.

4.5 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 Tkinter verwenden.

5 Systemplan

Das System soll nach dem Konzept des Baukastenprinzips entworfen werden beziehungsweise die Anforderungen für Modularität aus [Mey88] erfüllen. Das bedeutet, das sich die verschiedenen Komponenten verändern lassen, ohne dafür den Rest des Systems verändern zu müssen. Der Datenfluss zwischen den Modulen ist in Abbildung 1 dargestellt.

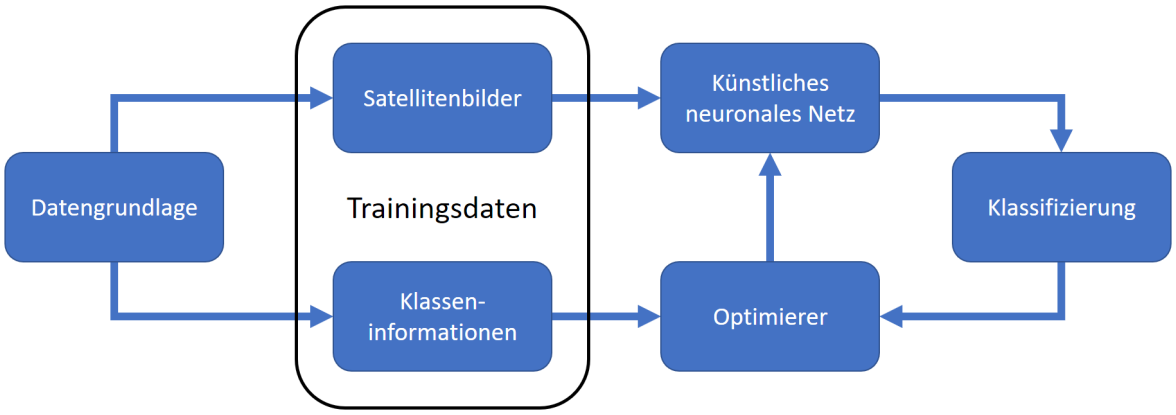

Abbildung 1: Datenflussdiagramm des Systems

Die erste Komponente ist die Datengrundlage, welche in Form einer CSV-Datei in das System gegeben wird. Sie kann geographische Koordinaten, Adressen oder andere Beschreibungen der gewünschten Eingabedaten enthalten, welche für den Trainings- und Validierungsprozess verwendet werden. Die zweite Komponente besteht aus den Satellitenbildern beziehungsweise dem Werkzeug zum Erzeugen der Satellitenbilder basierend auf der Datengrundlage. Bei der dritten Komponente handelt es sich um die Trainingsvorgaben. Sie bestehen aus den Klassenzuordnungen der Satellitenbilder und sind ebenfalls in der Datengrundlage enthalten. Sie repräsentieren die gewünschten Ausgabedaten des Netzes. Die letzte Komponente ist das künstliche neuronale Netz, welches für die Klassifizierungen zuständig ist und trainiert werden soll. Es bekommt die Trainingsdaten bestehend aus Satellitenbildern und Klasseninformationen und passt mittels eines Optimierers seine Gewichte so an, dass die Klassifizierung des Netzes möglichst mit der richtige Klassenzuordnung übereinstimmt. Dieser Hauptablaufplan des Systems befindet sich in Algorithmus 1.

5.1 Trainingsdaten

Die Trainingsdaten geben dem künstlichen neuronalen Netz vor, welche Aufgabe es zu erfüllen hat. Dabei bestehen die Trainingsdaten aus zwei Teilen.

Den Satellitenbildern, die die Eingabedaten des Netzes sind und welche auch für spätere Anwendungsfälle benötigt werden. Deren genauere Erzeugung wird in Abschnitt 5.2 erklärt.

Algorithmus 1 Hauptprozedur des Systems

```
# 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]
)
```

Der zweite Teil sind die Klasseninformationen der jeweiligen Bilder, welche die gewünschten Ausgabedaten des Netzes sind. Das Erzeugen der Klasseninformationen ist in der Regel ein aufwändiger Prozess und stark vom gewünschten zu erkennenden Merkmal abhängig. Es werden größere Mengen an Trainingsdaten benötigt, um ein Netz zu trainieren. Und wenn es für die gewünschte Aufgabe keine bereits existierende Datenquelle gibt, der man diese Menge an Informationen entnehmen kann, dann müssen sie notfalls eben händisch erzeugt werden. Dafür beinhaltet das System ein Interface, welches in Abbildung 2 zu sehen ist, und zeilenweise die Adressen einer CSV-Datei benutzt, um aus ihnen Satellitenbildern ebenfalls basierend auf Abschnitt 5.2 zu erzeugen, und diese anzeigt. Der Benutzer kann anschließend aus einer Auswahl an Klasseninformationen die dem Bild zugehörige auswählen. Seine Entscheidung wird danach zusammen mit der momentanen Adresse in einer neuen CSV-Datei abgespeichert.

5.2 Satellitenbilder

Die Satellitenbilder bilden die Grundlage der Eingabedaten 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[Goo] 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 geographischen 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[2] 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.

Algorithmus 2 Erzeugen der URL für die Google Static Maps API

```
1  url = "http://maps.google.com/maps/api/staticmap"
2  if centermode == "xy":
3      # Latitude and Longitude
4      url += "?center=" + y + "," + x
5  elif centermode == "address":
6      # Address
7      url += "?center=" + address
8  url += "&size=" + w + "x" + h
9  url += "&zoom=" + zoom
10 url += "&maptype=" + maptype
```

Die mittels dieser URL erzeugten Bilder werden lokal gespeichert und basierend auf ihren Parametern benannt. Dies verhindert ein mehrfaches Herunterladen des gleichen Bildes. Zusätzlich können die Bilder dadurch vor dem Trainingsprozess noch zugeschnitten oder auf eine andere Größe skaliert werden. Außerdem erleichtert es die Umwandlung der Bilder in drei 2D-Arrays für den Rot-, Grün- und Blaukanal. Das ist notwendig, da das Netz keine Farbbilder als Eingabe akzeptieren kann.

5.3 Künstliche Datenmengenvergrößerung

Sollte die Menge an Trainingsdaten für die gewünschte Aufgabe des Netzes zu klein sein, ist das Risiko einer Überanpassung und der damit einhergehenden schlechten Genauigkeitsrate für die Validierungsdaten sehr hoch.

Eine sehr einfache Variante um die Menge der Trainingsdaten künstlich zu erhöhen, ist das geringfügige Verändern der Satellitenbilder bei gleichbleibender Klasseninformation. Die genaue Abwandlung eines Bildes ist dabei zufällig, aber durch gewisse Rahmenbedingungen eingeschränkt. Wie in Algorithmus 3 gezeigt, ist dabei eine Vielzahl von Operationen, wie das Spiegeln, Drehen, Verschieben, Strecken oder Vergrößern des Bildes möglich.

Die Bilddaten werden hierbei aber nicht mehr über eine externen Datei geladen, die sämtliche Adressen der Trainingsdaten enthält, sondern direkt in Form von Bilddateien aus einem Ordner, wie in Algorithmus 4 zu erkennen.

Für das Erzeugen der Bilddateien ist dafür aber immer noch eine seperate Adressdatei notwendig und erfolgt wie in Abschnitt 5.2 beschrieben. Die Klasseninformation

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

Algorithmus 3 ImageDataGenerator

```
from keras.preprocessing.image import ImageDataGenerator

# Verändern der Bilddaten
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rescale=1. / 255,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='reflect'
)
```

Algorithmus 4 flow_from_directory

```
# Erzeugen der Trainingsdaten
generator = datagen.flow_from_directory(
    'data/train_data',
    target_size=(width, height),
    batch_size=batchsize,
    class_mode=classmode,
    shuffle=True
)
```

werden nun in Form von Unterordnern übermittelt. Jeder Ordner steht dabei für eine eigenständige Klasse. Deshalb ist diese Methode auch nur für Versuche mit klaren Klassenzuordnungen anwendbar.

Diese Variante der Datenmengenvergrößerung wurde auch in „Building powerful image classification models using very little data“[Cho16] verwendet.

5.4 Erstellen des künstlichen neuronalen Netzes

Die Grundstruktur des Netzes ist angelehnt an die Vorgaben von VGG16[SZ14] und besteht aus zehn Konvolutionsschichten, vier Max-pooling Schichten sowie drei vollvernetzten Schichten inklusive der abschließenden Schicht für die Repräsentation der Ausgabedaten. Die Funktion zum Erstellen des Netzes befindet sich in Algorithmus 5. Die Größe der Eingabeschicht wird automatisch durch die Größe der Bilder in den Trainingsdaten bestimmt.

Für die vollvernetzten Schichten wird das Prinzip des Dropout[SHK+14] angewendet. Damit werden nach jeder Trainingsepoche ein vorher eingetragener Prozentsatz der Vernetzungen entfernt beziehungsweise auf eine Stärke von 0 zurückgesetzt. Dies soll das Risiko einer Überanpassung weiter verringern.

Die Ausgabeschicht des Netzes muss sowohl in der Anzahl der Neuronen als auch in der Aktivierungsfunktion an das Format der Ausgabedaten angepasst werden. So beinhaltet bei einem Klassifizierungsproblem diese Schicht so viele Neuronen, wie es zu unterscheidende Klassen gibt. Außerdem wählt man die Softmax-Aktivierungsfunktion, siehe Gleichung (1). Das heißt, die Summe der Aktivierungen der Neuronen beträgt 1. Das Netz entscheidet also seine Klassenzuordnung darüber, welches Neuron am stärksten aktiviert wird. Die Stärke der Aktivierung sagt zudem aus, wie sicher sich das Netz bei dieser Entscheidung ist.

$$softmax_i(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_i}} \tag{1}$$

Bei einem Zwei-Klassen-Problem nutzt man lediglich ein Neuron mit einer Sigmoidfunktion, siehe Gleichung (2), für die Aktivierung als Ausgabeschicht.

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Für eine Regression wird ebenso eine Ausgabeschicht mit einem Neuronen verwendet. Aber da das gewünschte Format der Ausgabedaten ein Zahlenwert ist, handelt es sich hier bei der Aktivierung um eine identische Abbildung, siehe Gleichung (3).

$$id(x) = x \tag{3}$$

5.5 Training des Netzes

Nachdem nun die Trainingsdaten geladen und das neuronale Netz erstellt wurde, müssen die Gewichte des Netzes an das gewünschte Ziel angepasst werden. Dafür wird eine vorher festgelegt Anzahl, auch Batchsize genannt, an Satellitenbildern aus den Trainingsdaten genommen und vom Netz klassifiziert. Da sämtliche Berechnungen für den Trainingsprozess auf der Grafikkarte stattfinden sollen und der Grafikkartenspeicher durch die Größe der Bilddaten schnell ausgereizt ist, muss die Batchsize relativ klein gehalten werden. Die Ergebnisse der Klassifizierung werden nun mit den Trainingsvorgaben verglichen. Basierend auf diesen Abweichungen werden anschließend die Gewichte des Netzes angepasst. Bei einer niedrigen Batchsize ist ein Stochastischer Gradientenabstieg[Bot10] dafür am besten geeignet. Anschließend werden die nächsten Bilder genommen und der Anpassungvorgang beginnt von vorne. Man spricht von einer Trainingsepoche, wenn einmal alle Trainingsdaten verwendet wurden.

5.6 Bewertung des Netzes

Nach jeder Trainingsepoche werden die Validierungsdaten vom Netz klassifiziert und die Ergebnisse mit der Trainingsvorlage verglichen. Der sich daraus ergebende Genau-

igkeitswert 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 eine Überanpassung des Netzes vermieden werden.

Die Methode, nach der dieser Genauigkeitswert berechnet wird, hängt dabei vom Format der Ausgabedaten, und damit auch von der Aktivierungsart der letzten Schicht des Netzes ab. Bei Mehrklassenproblemen nimmt man die Häufigkeit, in der das Netz die richtige Klasse erkannt hat. Bei linearen Problemen hingegen werden der mittlere absolute Fehler oder der mittlere quadratischer Fehler verwendet.

5.7 Visualisierung von Klassifizierungen

Für die Darstellung der Ergebnisse der Klassifizierung des Netzes gibt es verschiedene Möglichkeiten. Eine Variante ist die Darstellung basierend auf geographischen Koordinaten. Dabei wird mit matplotlib, siehe Abschnitt 4.5, ein Streudiagramm erstellt. Die x- und y-Werte bestehen dabei aus den Längen- und Breitengraden der Koordinaten. Die Ergebnisse der Klassifizierung werden dann in der Farbe der Punkte kodiert. Ein Beispiel dafür ist in Abbildung 7 dargestellt.

Alternativ lassen sich die Satellitenbilder direkt mit dem korrekten Vorgabewerte sowie dem Ergebnis der Klassifizierung in einem eigenen Interface anzeigen. Dabei könnte besonders Wert auf Falschklassifizierungen gelegt werden, da an ihnen die Probleme des Netzes am besten erkennbar sind.

6 Experimente

Um die in Abschnitt 2 geforderten Kriterien für das System zu testen, wurden verschiedene Experimente durchgeführt. Diese unterscheiden sich in ihren Datengrundlagen für die Trainingsdaten, in der Form der Datengrundlagen, sowie im Format beziehungsweise Typ der Ausgabedaten. Ebenso werden darin die verschiedenen Arten der Ergebnisvisualisierung verwendet.

6.1 Geographische Koordinaten

Der erste Versuch besteht daraus, die geographischen 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"[3], deren Verteilung in Abbildung 3 dargestellt ist.

Bei einer Trainings-Validierungs-Aufteilung von 10% ergibt dies eine Anzahl von 5703 Trainingsdaten und 634 Validierungsdaten.

Die mittleren absoluten Fehler der Trainings- und Validierungsdaten lagen bei 2,602 und 1,534. Der Verlauf dieses Fehlers während der Trainingsphase des Netzes ist in Abbildung 4 dargestellt.

Die genauere Verteilung der Ergebnisse ist Abbildung 5 und Tabelle 1 zu entnehmen.

	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: Abweichungen der Netzklassikationen 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.

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

Die Trainingsdaten bestehen aus 64390 Häusern in Dresden, deren räumliche Anordnung sowie Zuordnung zu einer Postleitzahl in Abbildung 6 dargestellt ist.

6.3 Schule oder Wohnhaus

Das dritte Experiment wird die Klassifizierung von Häusern beinhalten. Dafür soll ein Netz trainiert werden, welches zwischen Wohnhaus und Schule unterscheiden kann.

Die Datengrundlage für die Schulen besteht aus einer Liste mit 204 Einträgen, veröffentlicht unter der Sektion Schulen in Dresden[4] vom Amt für Presse- und Öffentlichkeitsarbeit der Landeshauptstadt Dresden. Ein Auszug daraus befindet sich in Tabelle 3. Um die Trainingsdaten in beiden Klassen möglichst vergleichbar zu halten, enthält die Liste der Wohnhäuser ebenfalls 204 Einträge von zufällig aus dem Raum Dresden ausgewählten Adressen.

6.4 Schularten

Das letzte Experiment beinhaltet die Klassifizierung von Schulen in Sachsen nach den drei Klassen Grundschule, Oberschule und Gymnasium. Die dafür verwendeten Trainingsdaten basieren auf der offiziellen Sächsischen Schuldatenbank[5], herausgegeben vom Sächsischen Staatsministerium für Kultus. Ein Auszug daraus ist in Tabelle 4 zu sehen. Da hier weder geographische Koordinaten noch Adressen vorhanden sind, wird stattdessen der Name der Schule als Zentrum des Satellitenbildes in der Google Static Maps API[Goo] verwendet.

Die Klasseninformationen werden in Form eines Kategorievektors in das Netz gegeben. Dafür werden zunächst alle möglichen Klassen mit Null beginnend durchnummeriert. Anschließend werden die Kategorievektoren erstellt, wobei die Nummer der Klasse die Stelle angibt, in der im Vektor eine 1 eingetragen wird. An allen anderen Positionen befindet sich eine 0. Für dieses Experiment lautet die Zuordnung der Vektoren wie in Tabelle 2 abgebildet.

Klasse	Nummer	Vektor
Grundschule	0	(1 0 0)
Oberschule	1	(0 1 0)
Gymnasium	2	(0 0 1)

Tabelle 2: Zuordnung der Klassen zu ihren Kategorievektoren

[4]https://www.dresden.de/de/leben/schulen/schulen-in-dresden.php
[5]https://schuldatenbank.sachsen.de/index.php?id=2

7 Ergebnisse

Da sowohl Eingabe-, als auch Ausgabedaten in Form einer CSV-Datei in das System gegeben werden, muss das System lediglich an das Format der Ausgabedaten angepasst werden. Der Aufwand für das Wechseln der Datengrundlage oder des Klassifikationsziels ist damit gering gehalten. Die benötigten Satellitenbilder werden automatisch basierend auf der Datengrundlage heruntergeladen und das Netz passt seine Eingabeschicht selbstständig an die Größe der Trainingsdaten an. Dies sind die Kernpunkte für das Einhalten des Ziels der größtmöglichen Automatisierung. Außerdem befindet sich die Architektur und die Gewichte des Netzes nach dem Trainingsvorgang in einer seperaten Datei. Dadurch kann das Netz unabhängig vom Rest des Systems in einer Anwendung verwendet werden. Das Kriterium der leichten Integration in externe Anwendungen wurde damit erfüllt.

Insgesamt erfüllt das System alle in Abschnitt 2 geforderten Kriterien in gewissen Maßen und konnte deren Umsetzung sowie seine Funktionsfähigkeit in den verschiedenen Experimenten von Abschnitt 6 zeigen.

8 Ausblick

Die wohl größte Verbesserung für das erzeugte System wäre die Umsetzung einer Anwendung, die ein trainiertes Netz automatisch als Grundlage für die Klassifizierung von Satellitenbildern nimmt. Die Satellitenbilder könnten ebenfalls auf Bedarf mittels der Google Static Maps API[Goo] erzeugt werden und auf vom Nutzer eingegebenen Adressen basieren. Dadurch wäre der Schritt einer Integration in eine seperate Anwendung nicht mehr notwendig und ein trainiertes Netz könnte direkt nach Abschluss des Trainingsvorgangs verwendet werden.

Eine andere Erweiterung wäre das Trainieren eines Netzes mit einer großen Anzahl an Klassen, ähnlich dem Konzept von VGG16[SZ14] mit den Gewichten für ImageNet[RDS+15]. An dieses Netz könnten dann weitere Schichten angehangen werden, die das tatsächliche Format der gewünschten Ausgabedaten für ein Klassifizierungsproblem beinhalten, entsprechend dem Ansatz, welcher in „Building powerful image classification models using very little data“[Cho16] verwendet wird. Da dadurch ein Großteil des Netzes nicht zufällig initialisiert wird, sondern bereits vortrainierte Gewichte besitzt, ließe sich die Dauer zukünftiger Trainingsvorgänge enorm verkürzen.

Ein Problem, welches sich in den Abschnitten 6.3 und 6.4 ergeben hat, ist die fehlende Möglichkeit der Darstellung der Trainingsdaten sowie der Ergebnisse in Form eines Streudiagramms, sollte die Datengrundlage keine geographischen Koordinaten beinhalten. Dieses Problem könnte man mithilfe der Google Maps Geocoding API[6] lösen, indem man über die Adresse, oder im Fall von Abschnitt 6.4 über den Schulnamen, die Längen- und Breitengrade für jedes Element abfragt und in der jeweiligen Zeile der CSV-Datei einfügt.

[6] https://developers.google.com/maps/documentation/geocoding/start

Literatur

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[BBL+11] James Bergstra, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, Ian Goodfellow, Arnaud Bergeron, Yoshua Bengio, and Pack Kaelbling. Theano: Deep learning on gpus with python, 2011.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Cho15] Francois Chollet. Keras. https://github.com/fchollet/keras, 2015.

[Cho16] Francois Chollet. Building powerful image classification models using very little data. *Retrieved December*, 13:2016, 2016.

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[CWV+14] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[fPuF] Internationale Gesellschaft für Photogrammetrie und Fernerkundung. 2d semantic labeling contest. http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html.

[Goo] Google static maps api. https://developers.google.com/maps/documentation/static-maps/?hl=de.

[Gra00] John E Grayson. *Python and Tkinter programming*. Manning, 2000.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Mey88] Bertrand Meyer. *Object-oriented software construction*, volume 2. Prentice hall New York, 1988.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[SA16] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM, 2016.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[WKP16] 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.

Anlagen

Thesen zur Bachelorarbeit

Für die automatisierte Merkmalserkennung im Bereich der Bildverarbeitung findet Deeplearning zunehmend Anwendung. Über die genaue Netzstruktur für verschiedene Arten von Bilddaten gibt es eine Reihe von Arbeiten, doch im Aufbau und Ablauf des restlichen System unterscheiden sie sich dabei voneinander. Die vorliegende Arbeit verfolgt deshalb das Ziel, ein allgemeingültiges System zu entwickeln, mit dem die Erstellung eines künstlichen neuronalen Netzes sowie dessen Trainingsvorgang für Merkmalserkennungen in Satellitenbilder durchgeführt werden kann.

1. Merkmalserkennung mittels Deeplearning befindet sich in den letzten Jahren in stetem Wachstum.

2. Es gibt bisher kein einheitliches System zur Entwicklung von Klassifizierungen von Satellitenaufnahmen.

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Dresden, 25. Juli 2017

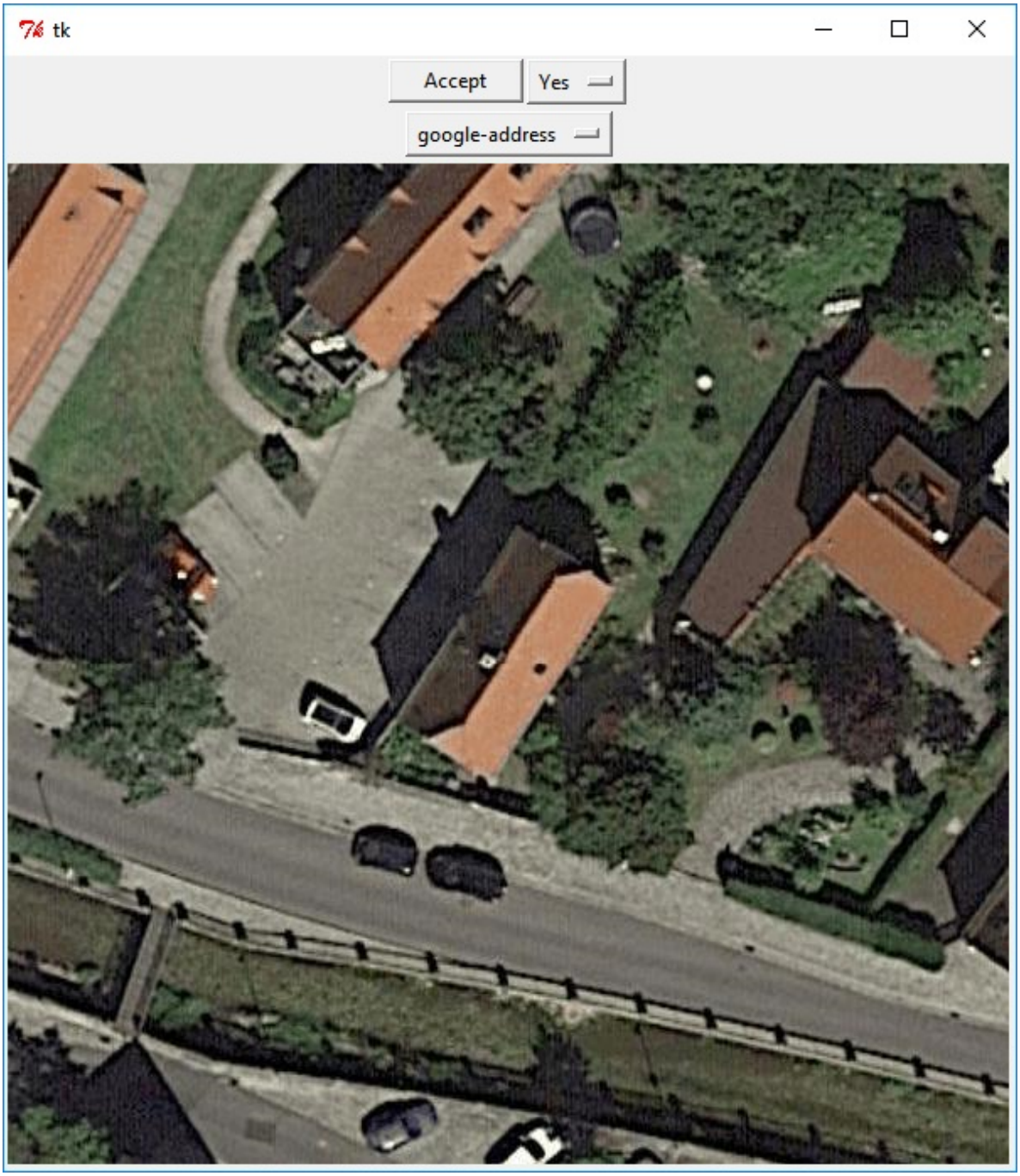

Abbildung 2: Anwendung zum händischen Erzeugen von Trainingsdaten

Algorithmus 5 Funktion zum Erstellen des künstlichen neuronalen Netzes

```
def create_net(X_train, Y_train):
    from keras.models import Sequential
    from keras.layers import (Flatten, Dense, Convolution2D,
        MaxPooling2D, ZeroPadding2D, Dropout)

    model = Sequential()
    model.add(ZeroPadding2D((1, 1), input_shape=X_train[0].shape[1:]))
    model.add(Convolution2D(16, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(16, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(32, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(ZeroPadding2D((1, 1)))
    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(Y_train.shape[1], activation='linear'))

    from keras.optimizers import SGD
    SGD = SGD(lr=0.0001, momentum=0.0, decay=0.0, nesterov=False)
    model.compile(loss='mean_absolute_error', optimizer=SGD, metrics=['
        mse'])

    return model
```

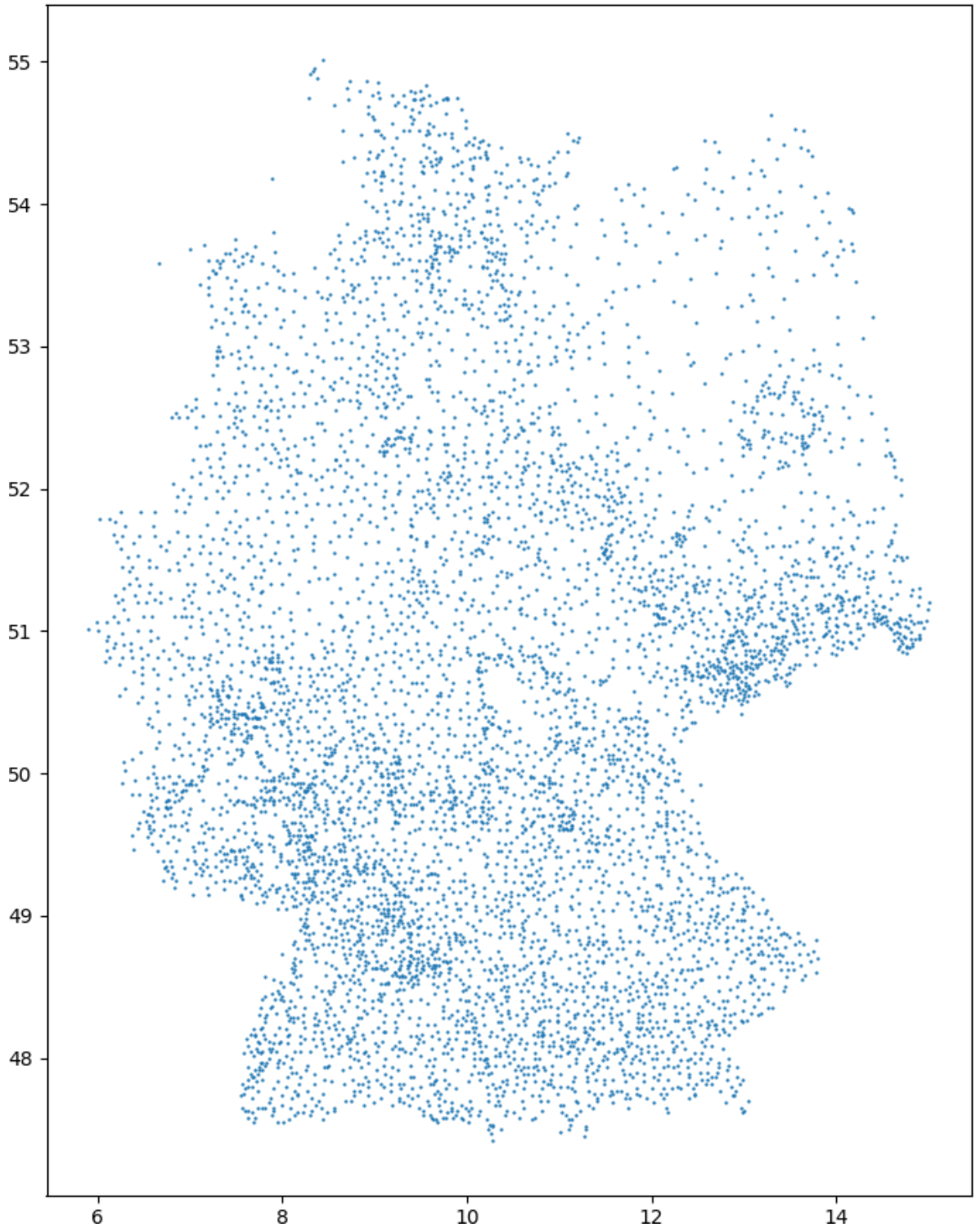

Abbildung 3: Deutsche Städte und Gemeinden

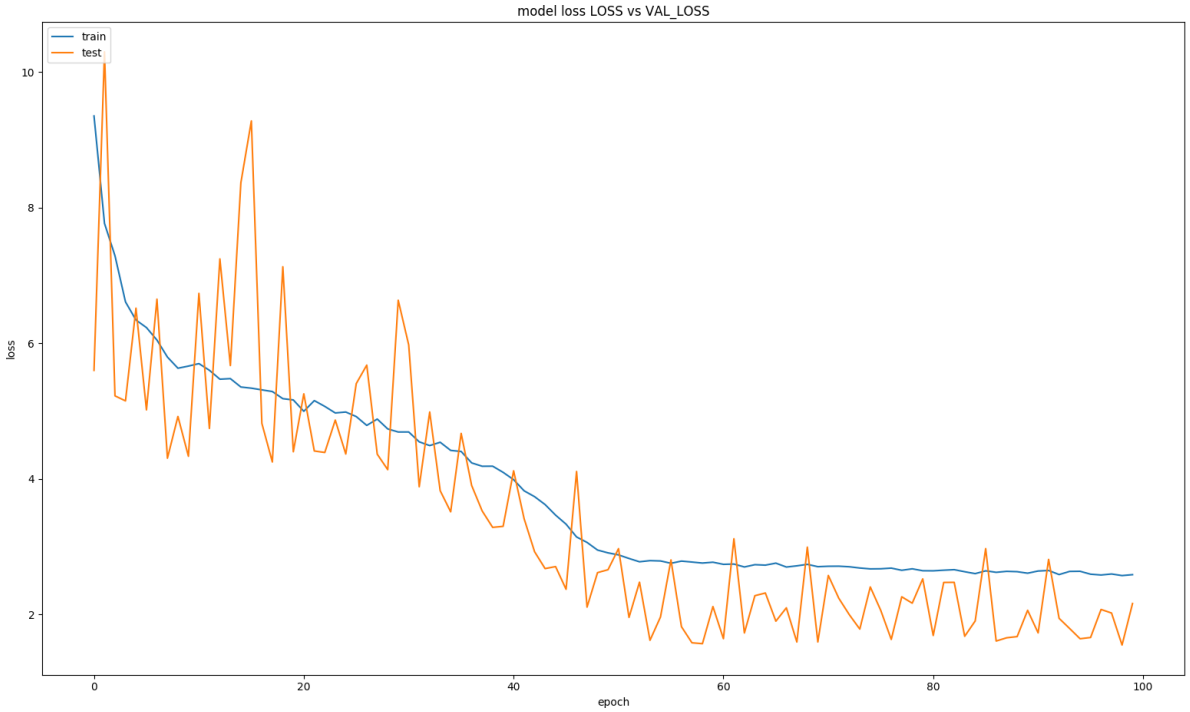

Abbildung 4: Mittlerer absoluter Fehler während der Trainingsepochen

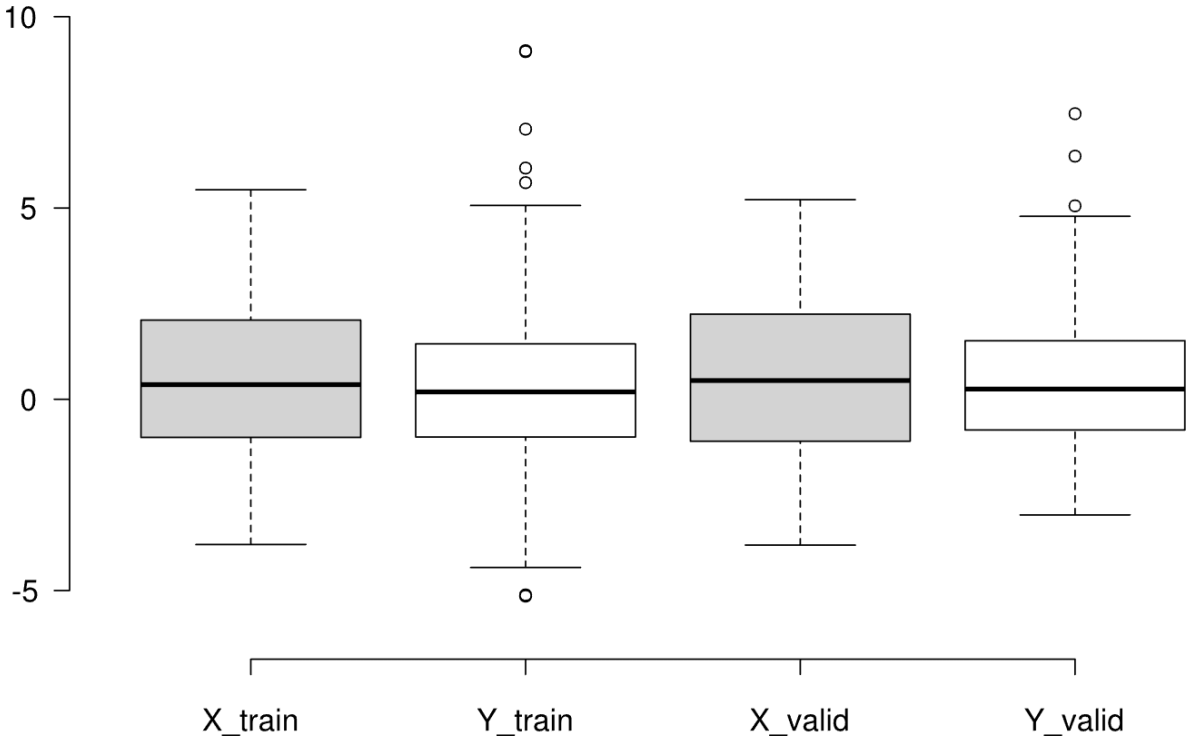

Abbildung 5: Abweichungen der Netzvorhersagen vom Vorgabewert

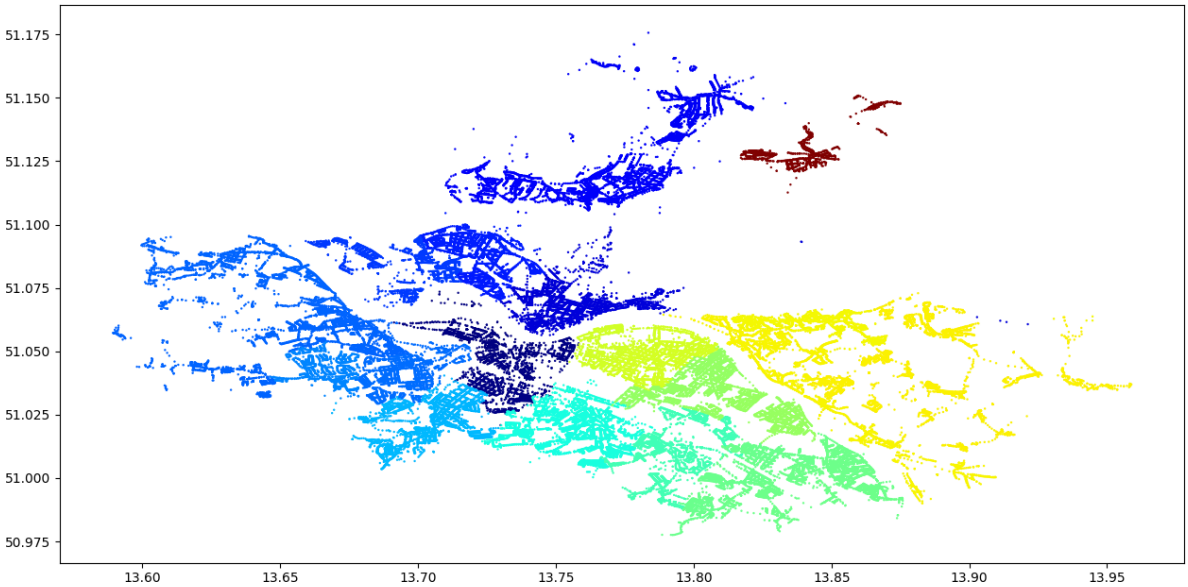

Abbildung 6: Postleitzahlenbereiche in Dresden

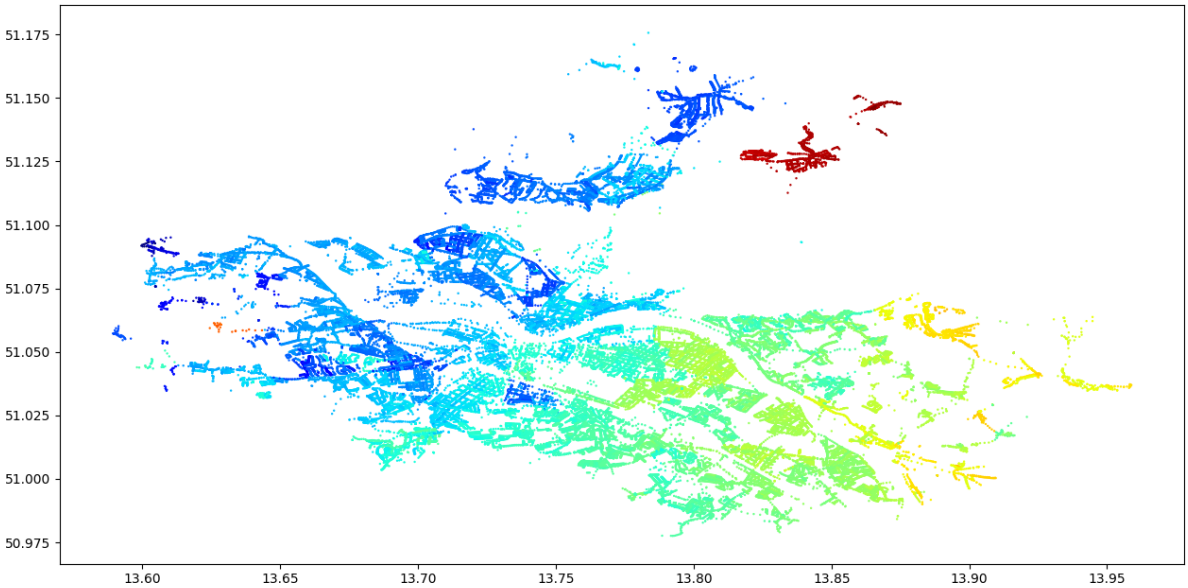

Abbildung 7: Ergebnisse der Klassifizierung des Netzes

Name	Straße	Stadtteil	Schultyp
10. Grundschule	Struvestraße 11	Seevorstadt	Grundschule
10. Oberschule	Messering 3	Friedrichstadt	Oberschule
101. Oberschule	Pfotenhauerstraße 42	Johannstadt	Oberschule
102. Grundschule	Pfotenhauerstraße 40	Johannstadt	Grundschule
103. Grundschule	Hohnsteiner Straße 8	Radeberger Vorstadt	Grundschule
106. Grundschule	Großenhainer Straße 187	Pieschen/Trachau	Grundschule
107. Oberschule	Hepkestraße 26	Striesen/Gruna	Oberschule
108. Grundschule	Hepkestraße 28	Gruna	Grundschule
113. Grundschule	Georg-Nerlich-Straße 1	Johannstadt	Grundschule
116. Oberschule	Feuerbachstraße 5	Leubnitz-Neuostra	Oberschule
117. Grundschule	Reichenbachstraße 12	Südvorstadt	Grundschule
12. Grundschule	Hebbelstraße 20	Cotta	Grundschule
120. Grundschule	Trattendorfer Straße 1	Prohlis	Grundschule
121. Oberschule	Gamigstraße 28	Prohlis	Oberschule
122. Grundschule	Gamigstraße 30	Prohlis	Grundschule
128. Oberschule	Rudolf-Bergander-Ring 3	Strehlen	Oberschule
129. Grundschule	Otto-Dix-Ring 57	Strehlen	Grundschule
135. Grundschule	Amalie-Dietrich-Platz 10	Gorbitz	Grundschule
138. Oberschule	Omsewitzer Ring 2	Gorbitz	Oberschule
139. Grundschule	Omsewitzer Ring 4	Gorbitz	Grundschule
14. Grundschule	Schweizer Straße 7	Südvorstadt	Grundschule
144. Grundschule	Micktner Straße 10	Pieschen	Grundschule
15. Grundschule	Görlitzer Straße 8	Äußere Neustadt	Grundschule
16. Grundschule	Josephinenstraße 6	Wilsdruffer/Seevorstadt	Grundschule
19. Grundschule	Am Jägerpark 5	Radeberger Vorstadt	Grundschule
25. Grundschule	Pohlandstraße 40	Striesen	Grundschule
25. Oberschule	Pohlandstraße 40	Striesen	Oberschule
26. Grundschule	Osterbergstraße 22	Pieschen	Grundschule
30. Grundschule	Hechtstraße 55	Leipziger Vorstadt	Grundschule
30. Oberschule	Unterer Kreuzweg 4	Innere Neustadt	Oberschule
32. Grundschule	Hofmannstraße 34	Blasewitz	Grundschule
32. Oberschule	Hofmannstraße 34	Blasewitz	Oberschule
33. Grundschule	Marienberger Straße 5	Seidnitz	Grundschule
35. Grundschule	Bünaustraße 12	Löbtau	Grundschule
35. Oberschule	Clara-Zetkin-Straße 20	Löbtau	Oberschule
36. Oberschule	Emil-Ueberall-Straße 34	Löbtau	Oberschule
37. Grundschule	Stollestraße 43	Löbtau	Grundschule
39. Grundschule	Schleiermacherstraße 8	Plauen	Grundschule
4. Grundschule	Löwenstraße 2	Innere Neustadt	Grundschule

Tabelle 3: Auszug aus Schulen in Dresden

Name	Schultyp
ABC Grundschule Neschwitz	Grundschule
Achatschule St. Egidien-Oberschule mit Berufsorientierung	Oberschule
Adam-Friedrich-Oeser-Schule - Grundschule der Stadt Leipzig	Grundschule
Adam-Ries-Schule-Grundschule	Grundschule
Adolf-Traugott-von-Gersdorf- Oberschule	Oberschule
Adolph-Diesterweg-Gymnasium	Gymnasium
Afra-Grundschule Meißen	Grundschule
AHFGrundschule Leipzig	Grundschule
Aktive Schule Dresden	Grundschule
Aktive Schule Dresden Oberschule des epharisto e.V.	Oberschule
Aktive Schule Leipzig e.V.	Grundschule
Albert-Einstein-Grundschule	Grundschule
Albert-Schweitzer-Gymnasium	Gymnasium
Albert-Schweitzer-Oberschule	Oberschule
Alexander-von-Humboldt-Gymnasium	Gymnasium
Alfred-Kästner-Schule - Grundschule der Stadt Leipzig	Grundschule
Altstädter Schule - Grundschule	Grundschule
Altstadtschule Stollberg, Oberschule	Oberschule
Andert-Oberschule Ebersbach	Oberschule
Anna-Magdalena-Bach-Schule	Grundschule
Annenschule -Grundschule	Grundschule
Annenschule, Oberschule	Oberschule
Anton-Philipp-Reclam-Schule - Gymnasium des	Gymnasium
Anton-Semjonowitsch-Makarenko-Grundschule	Grundschule
Apollonia-von-Wiedebach-Schule - Oberschule der Stadt Leipzig	Oberschule
Arthur-Kießling-Oberschule Königsbrück	Oberschule
Artur-Becker-Oberschule Delitzsch	Oberschule
Astrid-Lindgren-Grundschule	Grundschule
Astrid-Lindgren-Grundschule Heidenau	Grundschule
Astrid-Lindgren-Schule - Grundschule der Stadt Leipzig	Grundschule
August Moritz Böttcher Grundschule	Grundschule
August-Bebel-Schule - Grundschule der Stadt Leipzig	Grundschule
Augustum-Annen-Gymnasium Görlitz	Gymnasium
Basaltus-Grundschule Stolpen	Grundschule
Baumgartengrundschule	Grundschule
Bebelschule-Grundschule	Grundschule
Benjamin-Geißler-Grundschule Liebstadt	Grundschule
Bergschule St. Egidien Grundschule	Grundschule
Bernhard-von-Cotta-Gymnasium	Gymnasium

Tabelle 4: Auszug aus der Sächsischen Schuldatenbank