Generalisierung mittels Deep Learning (CNN) am Beispiel der Straßenextraktion aus GPS-Trajektorien

FRANK THIEMANN[1], SERCAN CAKIR[1] & FLORIAN POLITZ[1]

Zusammenfassung: Convolutional Neural Networks (CNN) sind in der Lage, wesentliche Merkmale der Eingabedaten zu erlernen. Mit dieser Arbeit wollten wir die Eignung von CNN zur Generalisierung am Beispiel der Extraktion von Straßenachsen aus GPS-Trajektorien testen. Als Eingangsdaten wurden aus den Trajektorien Dichtekarten abgeleitet. Als Referenzdaten haben wir das zugrundeliegende Straßennetz gerastert. In drei Experimenten wurden CNN auf simulierten Kfz- und realen Fahrrad-Trajektorien trainiert und dann auf die jeweiligen anderen Datensätze übertragen. Das Paper beschreibt die beobachteten Probleme.

1 Einleitung

GPS-Trajektorien werden an vielen Stellen erzeugt. Navigationssysteme, Smartphones, Sportuhren, etc. sind mit einfachen GPS-Empfängern ausgestattet, die die Position auf wenige Meter genau bestimmen können. Diese Positionsdaten werden als Zeitreihen mit wenigen Sekunden Auflösung aufgezeichnet und können auf diversen Internetplattformen gesammelt und geteilt werden. Frei verfügbare Trajektorien gibt es zum Beispiel im Openstreetmap-Projekt (OSM, openstreetmap.org). Sport- und Fitnessplattformen sammeln ebenfalls Trajektorien und bieten vielfältige Auswertungen an. Sehr beliebt sind dabei Heatmaps (Dichtekarten) aus den gesammelten Trajektorien. Unter Anderem ermöglichen Heatmaps häufig benutzte und damit vermeintlich beliebte Routen zu entdecken. Des Weiteren können sie auch Wegeverläufe sichtbar machen, die ansonsten nicht in Luftbildern und Karten erkennbar bzw. verzeichnet sind. So stellt die Fitnesstracking-Plattform Strava.com Heatmaps für OSM bereit, die als Grundlage zur Wegeerfassung dienen können.
In unserer Arbeit wollten wir testen, inwieweit sich die Extraktion des Straßennetzes aus Heatmaps mittels Deep Learning automatisieren lässt. Dabei fassen wir das Extraktionsproblem als Generalisierung auf, wobei die Straßenachsen die vereinfachte Repräsentation der Trajektorien bzw. der daraus abgeleiteten Heatmap darstellt.

2 Verwandte Arbeiten

Deep Convolutional Neural Networks (tiefe faltende neuronale Netze, CNN) werden seit Jahren erfolgreich für die semantische Klassifikation und Objektdetektion in Bildern eingesetzt (KRIZHEVSKY et al. 2012; GIRSHICK et al. 2013). In Form von Autoencodern (GOODFELLOW et al. 2016) können solche Netze die wesentlichen Merkmale der Eingangsdaten erlernen und diese zur Dimensionsreduktion bzw. zum Komprimieren der Daten nutzen. Dabei werden unwesentliche

[1] Leibniz Universität Hannover, Institut für Kartographie und Geoinformatik, Appelstraße 9 a, D-30167 Hannover, E-Mail: [Frank.Thiemann, Politz]@ikg.uni-hannover.de

Merkmale unterdrückt. Diese Eigenschaft von Autoencodern legt nahe, dass sie auch zur Generalisierung im kartographischen Sinne verwendet werden können.
SIMO-SERRA et al. (2016) haben bereits gezeigt, dass CNN zum Vereinfachen von groben Skizzen verwendet werden können. Die Autoren nutzen dabei ein CNN mit Autoencoder-Architektur. Anstatt, wie bei Autoencodern üblich, die Originalbilder sowohl als Eingangs- als auch als Referenzdaten zu nutzen, verwenden sie als Referenzdaten die vereinfachten Versionen der Eingangsbilder.

3 Methodik

Ziel unserer Arbeit ist es, die Eignung von CNN zur Generalisierung zu testen. Dazu adaptieren wir den Ansatz SIMO-SERRA et al. (2016) auf die Extraktion der Straßenachsen aus Heatmaps. Als Eingangsdaten dienen Dichtekarten von sowohl simulierten als auch realen GPS-Trajektorien. Als Referenzdaten werden die zugrundeliegenden Fahrbahn- bzw. Fahrwegachsen genutzt.
Für das Training des Netzes werden weitgehend fehlerfreie Referenzdaten benötigt. Eine direkte Verwendung von bestehenden Straßendatensätzen ist nur bedingt möglich, da das Straßennetz nicht überall ausreichend oft befahren wird, um eine Mindestanzahl an Trajektorien je Straße zu gewährleisten. Somit müssen zunächst nicht befahrene Straßenteile aus den Referenzdaten gelöscht werden. Auf Grund der geometrischen Ungenauigkeit, sowohl der Trajektorien und der Straßendatensätze, ist eine automatische Zuordnung der befahrenen Straßen nicht immer und nur mit großem Aufwand möglich.
Um die Korrektheit der Trainingsdaten zu gewährleisten, haben wir zunächst Trajektorien auf dem aus Openstreetmap extrahierten Straßennetz von Hannover simuliert. Des Weiteren haben wir versucht, Referenzdaten zu realen Fahrrad-Trajektorien zu generieren, was uns wegen der beschriebenen Probleme jedoch nicht in der benötigten Genauigkeit gelang.
Mit beiden Datensätzen haben wir ein CNN trainiert und dieses dann auf dem jeweils anderen Datensatz getestet. In einem weiteren Test haben wir die gelernten Modelle auf einen Datensatz aus realen Kfz-Trajektorien angewendet und die Ergebnisse visuell beurteilt.

3.1 Eingangs- und Referenzdatensätze

Als Quelle für die Referenzdaten haben wir das Straßennetz aus Openstreetmap verwendet. Aus den Heatmaps von Fahrrad-Trajektorien von Strava (heatmaps.strava.com) haben wir Dichtekarten generiert. Des Weiteren haben wir zeitlich hochaufgelöste Kfz-Trajektorien von der Stadt Chicago verwendet (AHMED et al., 2015).
Die Erstellung eines Referenzdatensatzes gestaltet sich für größere Straßendatensätze äußerst aufwendig. Um einen fehlerfreien Referenzdatensatz zu erhalten, muss das Straßennetz passend zu den Trajektorien vektorisiert werden. Selbst bei der Verwendung von existierenden Straßendatensätzen wie z. B. ATKIS oder OSM, ist eine erhebliche manuelle Nachbearbeitung notwendig. Straßen und Straßenteile, für die keine Trajektorien vorliegen, müssen entfernt werden, da sie nicht prädiziert werden können. Ebenso müssen fehlende Straßen oder Fahrspuren ergänzt und Geometrien ggf. korrigiert werden. Insbesondere für die Fahrrad-Trajektorien gestaltete sich die Zuordnung zum Wegenetz schwierig, da alternative Spuren wie z. B. Radweg und rechter

Fahrbahnrand oft sehr eng nebeneinanderliegen. Zudem ist die Vollständigkeit der Radwege in OSM deutlich geringer als die der Fahrbahnachsen.
Um die Vollständigkeit und geometrische Korrektheit der Referenzdaten zu gewährleisten, haben wir Trajektorien auf einem ausgewählten Straßendatensatz simuliert. Dazu wurde zunächst das für Kraftfahrzeuge nutzbare Straßennetz von Hannover extrahiert. Im Folgenden wurden zufällig Knoten im Straßennetz selektiert und ein Routing zwischen den Knoten durchgeführt. Die daraus entstandene Route wurde dann in regelmäßigen Abständen abgetastet. Auf die so erhaltenen Punktkoordinaten wurde anschließend ein zufälliges Rauschen (σ = 0,6 m) und eine zufällige, systematische Verschiebung je Trajektorie (σ = 5 m) addiert.
Zur Verwendung eines CNN müssen die Trajektorien und das Referenzstraßennetz in Bild- bzw. Rasterdaten überführt werden. Aus den Trajektorien wurde dazu mit einem Gaußkern die Dichte berechnet. Die Straßenachsen wurden direkt gerastert (siehe Abb. 1). Als Pixelgröße wurden 3 Meter gewählt, da dies etwa der Genauigkeit der einzelnen Trajektorien entspricht. Bei der gewählten Auflösung sind parallele Fahrbahnachsen im Raster noch unterscheidbar. Zum Rastern eines Radwegenetzes ist diese Auflösung jedoch nicht mehr ausreichend. Hier beginnen einige Achsen, wie z. B. die Achsen von straßenparallelen Radwegen mit der Fahrbahnachse, zu verschmelzen (siehe Abb. 2).

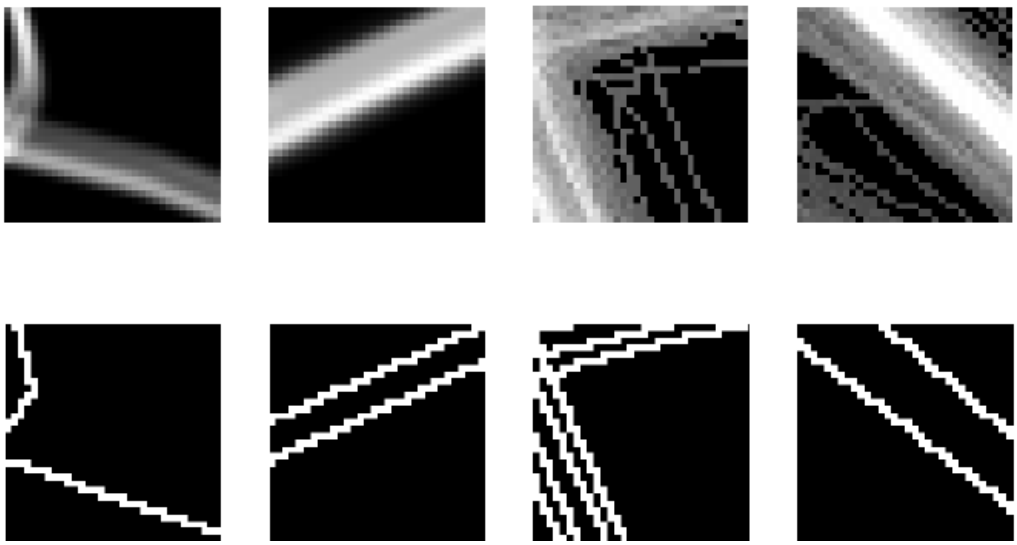

Abb. 1: Simulierte Dichtekarten (oben) und dazugehörige Referenzdaten (unten)

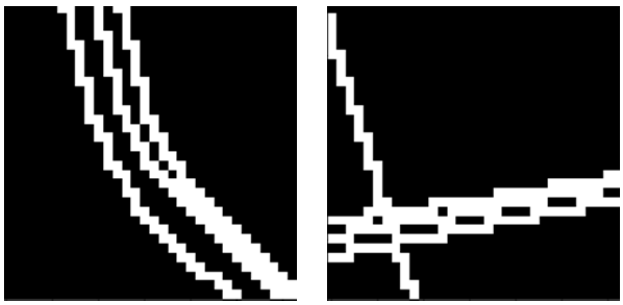

Abb. 2: Radwege- und Fahrbahnachsen verschmelzen beim Rastern mit 3 m Auflösung

3.2 Convolutional Neural Network

Das für unsere Arbeit genutzte Convolutional Neural Network wurde von dem Modell von SIMO-SERRA et al. (2016) durch Reduktion der Parameter abgeleitet. Das vollständige CNN ist in Abb. 3 dargestellt. Die Größe der Eingangsbilder ist dabei frei wählbar, da das Modell auf voll vernetzte Layer verzichtet. Wir haben quadratische Bildpatches mit einer Größe von 32 x 32 Pixel verwendet. Der erste Teil des Netzwerkes fungiert als Encoder und soll die Eingabebilder komprimieren, indem er signifikante Merkmale extrahiert. Statt einer Pooling-Funktion wird durch zwei Down-Convolutions mit einer Schrittweite von 2 das Eingabebild auf 1/16 seiner ursprünglichen Größe reduziert. Der Informationsverlust wird durch Erhöhung der Filterzahl ausgeglichen.
Der mittlere Teil des Netzwerkes extrahiert die grundlegende Information der Straßenachsen mit konstanter Bildgröße. Hier wird die Zahl der Filter bis zur Mitte des Netzes systematisch erhöht und anschließend wieder reduziert. Das CNN schließt mit einem dekodierenden Teil, der das nun generalisierte Bild wieder in die Ausgangsgröße zurückführt. Die Up-Convolutions sind als Up-Sampling gefolgt von einer Faltung mit einer 5 x 5 großen Filtermatrix implementiert.
Die Ergebnisse jeder Faltung werden mit einer nichtlinearen Funktion aktiviert. Dabei wird mit Ausnahme der letzten Schicht eine Rectified Linear Unit-Funktion (ReLU-Funktion) verwendet (NAIR & HINTON, 2010). Um die Ausgabe in das Intervall [0,1] zu normieren wird beim letzten Layer die Sigmoid-Funktion verwendet. Zur Vermeidung von Überanpassung werden die Parameter mit der L2-Norm regularisiert und es wird nach jedem Layer ein Dropout-Layer (SRIVASTAVA et al., 2014) mit einer Dropout-Rate von 10 % eingefügt.

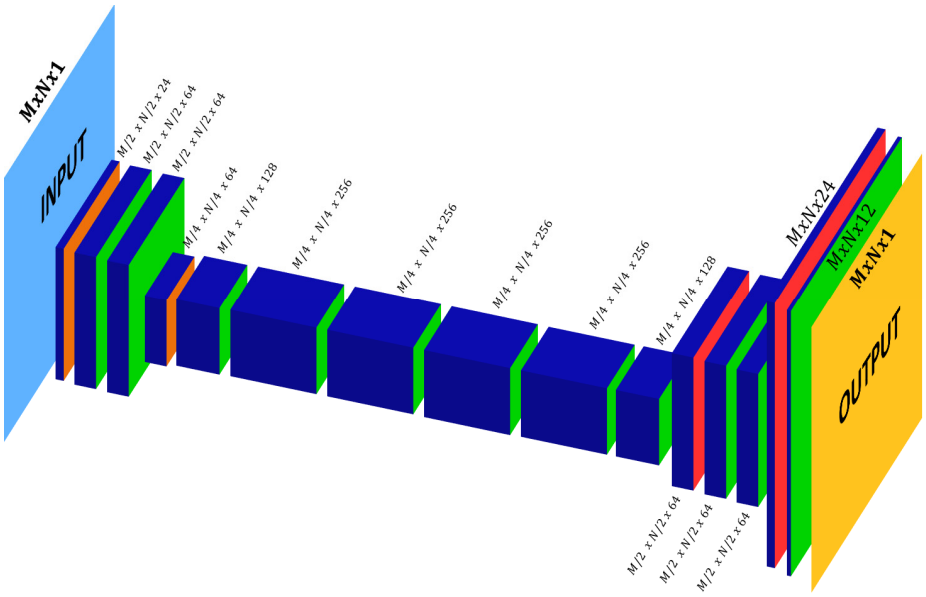

Abb. 3: Verwendetes CNN (Zeilen x Spalten x Kanäle)

Das Modell wird mit den Heatmaps als Eingabe und den gerasterten Straßenachsen als Referenzdaten trainiert. Als Kostenfunktion wurde die binäre Kreuz-Entropie verwendet. Diese ist besonderes für binäre Klassifikationen geeignet. Als weitere Metrik wurde die Kovarianz von Ausgabe- und Referenzbild berechnet. Um auch geringe Lageabweichungen der prädizierten

Straßen zu erlauben, wurden nur die inneren 30 × 30 Pixel als gleitendes Fenster über das 32 × 32 Pixel Referenzbild geschoben und die maximale Kovarianz verwendet.
Um Probleme mit der Einstellung der Lernrate zu umgehen, wurde die Optimierung mit dem Adadelta-Verfahren (ZEILER 2012) durchgeführt.

Tab. 1: CNN-Struktur. Die Farben wurden analog zu Abb. 3 gewählt.

Art der Faltung	Größe (Zeilen x Spalten x Kanäle)	Schrittweite	Filtergröße
Eingabe	M x N x 1		-
down-conv-1	M/2 x N/2 x 24	2 x 2	5 x 5
flat-conv-1	M/2 x N/2 x 64	1 x 1	3 x 3
flat-conv-2	M/2 x N/2 x 64	1 x 1	3 x 3
down-conv-2	M/4 x N/4 x 64	2 x 2	3 x 3
flat-conv-3	M/4 x N/4 x 128	1 x 1	3 x 3
flat-conv-4	M/4 x N/4 x 256	1 x 1	3 x 3
flat-conv-5	M/4 x N/4 x 256	1 x 1	3 x 3
flat-conv-6	M/4 x N/4 x 256	1 x 1	3 x 3
flat-conv-7	M/4 x N/4 x 256	1 x 1	3 x 3
flat-conv-8	M/4 x N/4 x 128	1 x 1	3 x 3
up-conv-1	M/2 x N/2 x 64	2 x 2	3 x 3
flat-conv-9	M/2 x N/2 x 64	1 x 1	3 x 3
flat-conv-10	M/2 x N/2 x 64	1 x 1	3 x 3
up-conv-2	M x N x 24	2 x 2	3 x 3
flat-conv-11	M x N x 12	1 x 1	3 x 3
flat-conv-12	M x N x 1	1 x 1	3 x 3

3.3 Experimente und Ergebnisse

Damit sich Trainings- und Validierungsdaten nicht überlappen und möglichst ähnliche Straßennetzstrukturen abdecken, wurde das Gebiet in adjazente, nicht überlappende Kacheln von 32 x 32 Pixeln aufgeteilt. Kachelpaare, bei denen Eingabe- oder Referenzbild keine Straßenpixel enthielten, wurden vor dem Training aus den Datensätzen entfernt. Etwa 80 % der Daten wurden fürs Training und die restlichen 20 % zur Validierung verwendet.

3.3.1 Training mit simulierten Trajektorien

Für das erste Experiment wurde das Netz mit den simulierten Trajektorien trainiert. Wir haben 3772 Trainings- und 1000 Validierungsbilder verwendet. Das Training erfolgte über 250 Epochen. Die Kostenfunktion auf den Trainingsdaten fällt dabei stetig, d. h. die Optimierung konvergiert. Die Kosten im Validierungsdatensatz erreichen hingegen bereits nach ca. 15 Epochen ihr Minimum und beginnen dann wieder zu steigen. Die Kovarianzen steigen sowohl auf dem Trainings- als auf dem Validierungsdatensatz stetig, wobei der Wert für den Validierungsdaten-

satz deutlich unterhalb des Wertes des Trainingsdatensatzes verbleit. Die Werte für den Validierungsdatensatz lassen sich durch Variation der Dropout-Rate und der Gewichtung der L2-Regularisierung nicht wesentlich verbessern. Die Richtigkeit und die Vollständigkeit der prädizierten Straßenachsen auf dem gesamten Datensatz betragen rund 90 % (siehe Tab. 2). Der vermeintlich gute Wert ist jedoch auf eine Überanpassung an die Trainingsdaten zurückzuführen.

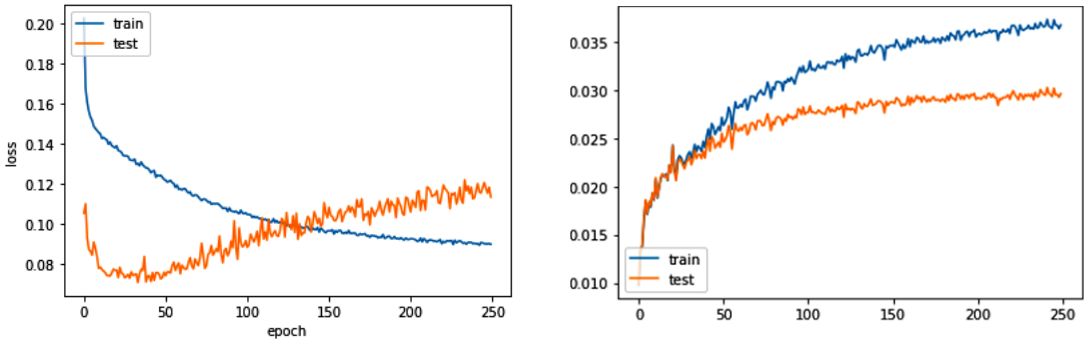

Abb. 4: Verlauf der Kostenfunktion (Kreuz-Entropie) (links) und der Kovarianz (rechts) über die 250 Trainingsepochen. Die absoluten Werte der Entropie von Trainings- und Testdaten lassen sich nicht vergleichen, da beim Training ein Dropout von 10 % verwendet wurde, welches beim Validieren nicht zur Anwendung kommt.

3.3.2 Training mit Fahrrad-Heatmap

Im zweiten Experiment haben wir als Eingabedaten die Heatmaps von Strava verwendet und als Referenzdaten das zugrundeliegende Radwegenetz aus OSM herangezogen. Es wurden die selben Hyperparameter wie in Kap. 3.3.1 verwendet. Auch hier konvergiert die Optimierung. Die Kovarianz von Prädiktion und Referenzdaten fällt jedoch geringer als beim Training mit den simulierten Trajektorien aus. Die Kovarianzen von Trainings- und Validierungsdaten unterscheiden sich weniger (siehe Abb. 5).

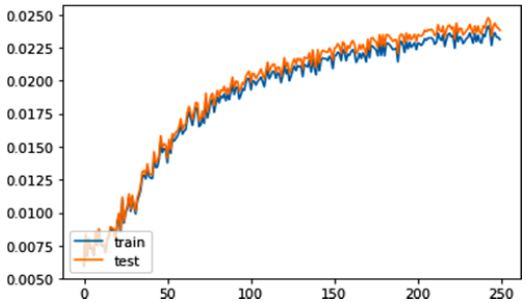

Abb. 5: Verlauf der Kovarianz über 250 Trainingsepochen beim Training mit der Fahrrad-Heatmap.

Die erzielte Richtigkeit und Vollständigkeit auf dem Strava-Datensatz (Trainings- und Testdaten zusammen) beträgt lediglich 38 % und 33 %. Dies kann auf die geringere Genauigkeit der Referenzdaten zurückzuführen sein, denn bereits beim Erstellen der Referenzdaten fiel auf, dass die OSM-Wegeachsen und die dazugehörigen Stellen in der Heatmap oft nicht gut zusammenpassen. Beim Testen des trainierten Netzwerkes auf den simulierten Daten wird lediglich eine Richtig-

keit von 17 % und eine Vollständigkeit von 10 % erreicht. Das auf Fahrradtrajektorien trainierte CNN lässt sich folglich nicht auf die simulierten Daten übertragen.

3.3.3 Training mit simulierten und realen Daten

Im dritten Experiment haben wir beide Datensätze, also die simulierten Trajektorien sowie die realen Strava-Heatmaps, gemeinsam als Trainingsdaten für das CNN verwendet. Es standen insgesamt rund 7000 Eingangsbilder für das Training zur Verfügung. Die erzielte Richtigkeit beträgt hier 53 %, und die Vollständigkeit 48 %. Die Kovarianz der Validierungsdaten ist in diesem Experiment nur geringfügig besser als beim zweiten. Auch hier bleibt der Wert für den Validierungsdatensatz deutlich kleiner als für den Trainingsdatensatz, was auf eine Überanpassung an den Trainingsdatensatz hindeutet.

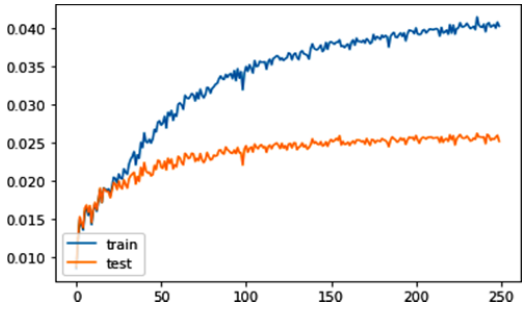

Abb. 6: Verlauf der Kovarianz über 250 Trainingsepochen mit kombiniertem Datensatz

3.3.4 Vergleich der Ergebnisse

Die trainierten Netzwerke wurden jeweils auf die einzelnen und den kombinierten Datensatz angewendet, um die Gesamtleistung zu messen. Die besten Ergebnisse wurden mit den simulierten Daten auf dem CNN mit den simulierten Trajektorien erzielt. Die Struktur der Trajektorien ist hier am einfachsten und die Genauigkeit der Referenzdaten am höchsten. Das so trainierte Netzwerk lässt sich jedoch nicht auf die realen Fahrradtrajektorien übertragen. Eine Ursache hierfür könnte sein, dass Fahrradfahrer ein anderes Spurwahlverhalten haben als Kraftfahrzeuge. Radfahrer fahren in der Regel immer am rechten Fahrbahnrand oder rechts neben der Fahrbahn auf dem Radweg. Auf einigen Plätzen können Radfahrer ihre Fahrspuren vollkommen frei wählen. Dieses Verhalten ist in den simulierten Trajektorien jedoch nicht abgebildet, wodurch das so trainierte CNN dieses nicht lernen konnte.
Das Training auf den realen Fahrrad-Heatmaps liefert mit 38 % Richtigkeit und 33 % Vollständigkeit nur schlechte Ergebnisse. Der Referenzstraßensatz passt hier vermutlich zu schlecht zur Heatmap. Das so trainierte Netzwerk konnte ebenfalls nicht auf den anderen Datensatz übertragen werden. Beim Training auf gemischten Trainingsdaten ergeben sich für die Fehlermaße geringfügig schlechtere Ergebnisse als beim Übertragen des CNN mit den simulierten Trajektorien. Visuell betrachtet wirkte das Ausgabebild jedoch plausibler.

Tab. 2: Vollständigkeit und Richtigkeit der Straßenpixel im Überblick

Validierung mit:	Training mit:	simuliert	Fahrrad	kombiniert
simulierten Trajektorien	Richtigkeit	0,90	0,18	0,54
	Vollständigkeit	0,89	0,37	0,63
realen Fahrrad-Heatmaps	Richtigkeit	0,17	0,38	0,28
	Vollständigkeit	0,10	0,33	0,22
kombinierten Datensätzen	Richtigkeit			0,53
	Vollständigkeit			0,48

Auf dem Datensatz von Chicago, welches auf Kfz-Trajektorien basiert, wurde zudem ein Test mit visueller Begutachtung durchgeführt (siehe Abb. 7). Das auf den simulierten Trajektorien trainierte CNN gibt nur ein unvollständiges Straßennetz als Ergebnis aus. Das lässt sich damit erklären, dass für das Training in unserem Referenzdatensatz Straßen, denen nur wenige Trajektorien zuzuordnen waren, gelöscht wurden. Nach dem Training mit realen Fahrrad-Heatmaps ist das Ausgabebild vollständiger. Es werden jedoch auf breiten Straßen mehrere Achsen prädiziert. Nach dem Training mit dem kombinierten Datensatz reduziert sich dieser Effekt. Es ergibt sich hier visuell das beste Ergebnis.

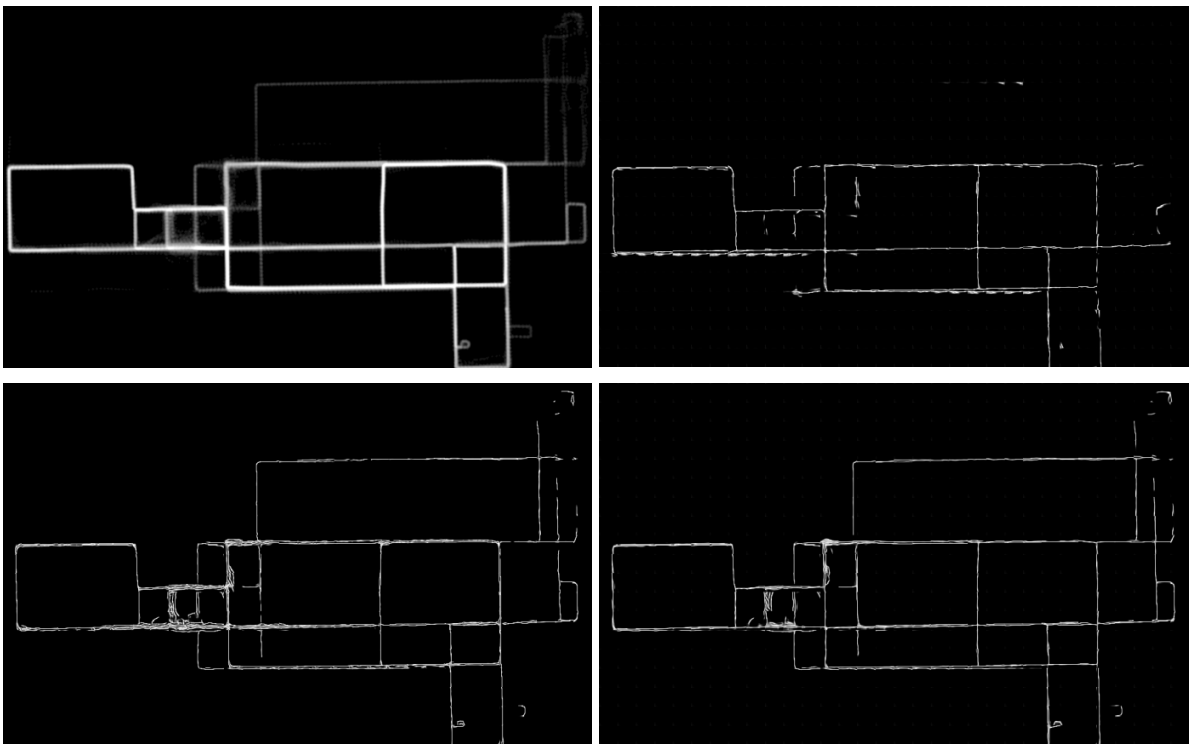

Abb. 7: Eingangsdaten: Heatmap von Chicago (a), Prädiktion (Wahrscheinlichkeit) der Straßenachsen: nach Training mit simulierten Trajektorien (b), mit realer Fahrrad-Heatmap (c) und mit kombiniertem Trainingsdatensätzen (d).

4 Fazit & Ausblick

Die mit dem Ansatz erzielten Ergebnisse sind bisher noch nicht zufriedenstellend. Das Erstellen der Referenzdaten stellte sich als sehr aufwendig heraus. Die Verwendung von bestehenden Straßendatensätzen als Referenz erfordert viel manuelle Nacharbeitung, um fehlende, nicht benutzte oder ungenaue Straßen zu korrigieren. Die Simulation von Trajektorien bedarf aufwendigerer Algorithmen, um ein realistisches Spurwahlverhalten auch auf breiteren Fahrbahnen nachzubilden.
Veränderungen am Modell haben auf die Ergebnisse nur wenig Auswirkung gezeigt. Die Kostenfunktion (Kreuz-Entropie) erreicht auf dem Validierungsdatensatz bereits nach ca. 15 Iterationen ihr Minimum und beginnt dann wieder zu steigen. Das Ausgabebild ist nach 15 Iterationen noch unscharf und wird erst mit größerer Epochenzahl schärfer. Mit zunehmender Schärfe liegt die Prädiktion aufgrund der Lageungenauigkeiten der Trajektorien und des Referenzdatensatz oft geringfügig neben der Referenz. Jegliche Lageabweichung ($\geq$ 1 Pixel) führt bei der Kostenfunktion jedoch zu einem starken Anstieg. Dem kann nur begegnet werden, wenn entweder die Referenzdaten optimal an die Trainingsdaten angepasst werden oder eine tolerantere Kostenfunktion genutzt wird, die kleine Lageabweichungen weniger bestraft. Die vorgestellte gleitende Kovarianz, ist dazu ein erster Ansatz, wenngleich sie nur auf kleinen Patches sinnvolle Ergebnisse liefern kann.
Der Überanpassung könnte künftig durch mehr Trainingsdaten begegnet werden. Eine stärkere Regularisierung und höhere Dropout-Raten führten lediglich zu unschärferen Ausgaben, was einer unsichereren Prädiktion gleichkommt.
Generell stellt der benutzte Ansatz ein interessantes Werkzeug für weitere Probleme zur Ableitung von Achsen bereit. So wollen wir künftig die Anwendbarkeit des Ansatzes zur Extraktion von Gewässer- und Wegeachsen aus digitalen Geländemodellen testen.

5 Literaturverzeichnis

AHMED, M., KARAGIORGOU, S., PFOSER, D. & WENK, C., 2015: A Comparison and Evaluation of Map Construction Algorithms. GeoInformatica, **19**(3), 601-632. Chicago Datensatz unter http://mapconstruction.org

GIRSHICK, R., DONAHUE, J., DARRELL, T. & MALIK, J., 2013: Rich features hierarchies for accurate object detection and semantic segmentation. Extended version of the published conference paper at CVPR 2014. arXiv: 1311.2524v5.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A., 2016: Deep Learning. MIT Press 2016, Kapitel 8. http://www.deeplearningbook.org

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G., 2012: ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information (NIPS), **2012**(1), 1097-1105.

NAIR, V. & HINTON, G. E. 2010: Rectified Linear Units Improve Restricted Boltzmann Machines. In: ICML'10 Proceedings of the 27th International Conference on Machine Learning, 807-814.

SIMO-SERRA, E., IIZUKA, S., SASAKI, K. & ISHIKAWA, H., 2016: Learning to simplify: fully convolutional networks for rough sketch cleanup. In: ACM Trans. Graph, **35**(4), Artikel 121.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R., 2014: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: Journal of Machine Learning Research, **15**, 1929-1958.

ZEILER, M. D., 2012: ADADELTA: An Adaptive Learning Rate Method. arXiv: 1212.5701v1.