AI Spektrum 2019

AI Spektrum

Das Magazin für Künstliche Intelligenz

KÜNSTLICHE INTELLIGENZ IN DER PRAXIS

Die Suche nach dem passenden Algorithmus

Pragmatisch zum Einsatz: Machine Learning in der Cloud

Convolutional Neural Nets – Praxisbeispiel I

AUTOENCODER – EINE VIELSEITIG EINSETZBARE ARCHITEKTUR

Autoencoder sind generative neuronale Netzwerke, die äußerst vielseitig einsetzbar sind. Sie finden unter anderem beim maschinellen Übersetzen Anwendung, bei der Anomalieerkennung oder der Bildbearbeitung. Welche Spezialformen sind für die semantische Segmentierung von Bildern geeignet – wie es etwa beim autonomen Fahren unerlässlich ist?

Convolutional Neural Nets (CNN) spielen seit drei bis vier Jahren eine führende Rolle im Bereich der Computer Vision. Autonomes Fahren zum Beispiel kann nur dann funktionieren, wenn das Fahrzeug unterschiedliche Objekte auf Kamerabildern pixelgenau identifizieren kann. Diese Aufgabe ist deutlich schwerer als etwa eine Bildklassifikation oder eine Bounding Box-Lokalisation. Eine verwandte, speziellere Aufgabenklasse ist das Freistellen von Objekten etwa von Fahrzeugen auf Bildern, also die Trennung von Objekt und Hintergrund. Im Fokus dieses Artikels steht die Lösungsarchitektur für diese und andere Aufgabenklassen: der Autoencoder.

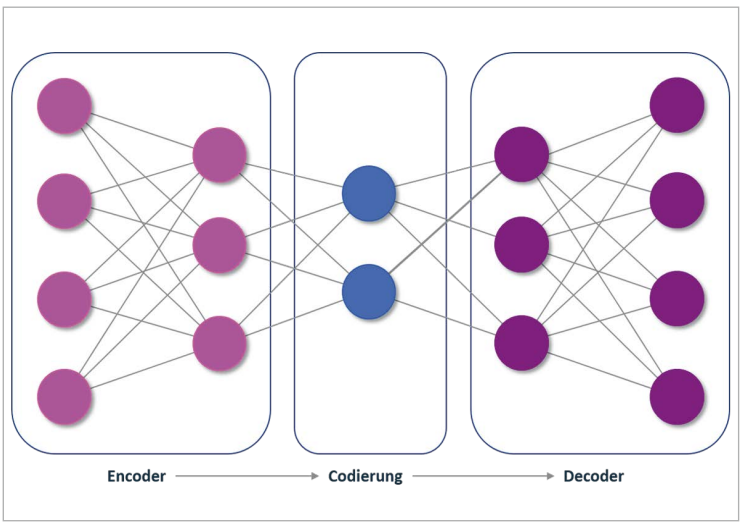

Abb. 1: Autoencoder führen im Prinzip drei Aufgaben durch

Autoencoder-Architektur

Autoencoder stellen eine spezielle Klasse neuronaler Netzwerke dar. Sie werden darauf trainiert, ihren Input zu rekonstruieren. Im Prinzip bestehen sie aus drei Teilen: dem Encoder, der Codierung und dem Decoder.

Ein einfacher Autoencoder kann als Hintereinanderausführung zweier Funktionen $\mathbf{h}$ (Encoder) und $\mathbf{g}$ (Decoder) modelliert werden: $\mathbf{g}\circ\mathbf{h}$. Im Fall des Feature Learnings ist man jedoch nicht an der Ausgabe des Decoders $\mathbf{g(h(x))}$ interessiert, sondern an der Codierung $\mathbf{h(x)}$. Eine Möglichkeit, die Abstraktion der Features zu fördern, besteht darin, eine kleinere Dimension für $\mathbf{h(x)}$ im Vergleich zu $\mathbf{x}$ wählen.

Eine exakte Rekonstruktion, das heißt $\mathbf{g(h(x)) = x}$ ist zudem oftmals nicht das Ziel: Ein traditioneller Autoencoder minimiert beim Training den Loss $\mathbf{L(x,g(h(x)))}$. Bei ausreichender Kapazität begünstigt dies das Lernen der Identitätsfunktion. Ein Denoising-Autoencoder hingegen minimiert nun $\mathbf{L(x,g(h(x^*)))}$, wobei $\mathbf{x^*}$ eine korrumpierte Version von $\mathbf{x}$ ist. Der Autoencoder muss also lernen, die Korrumpierung rückgängig zu machen.

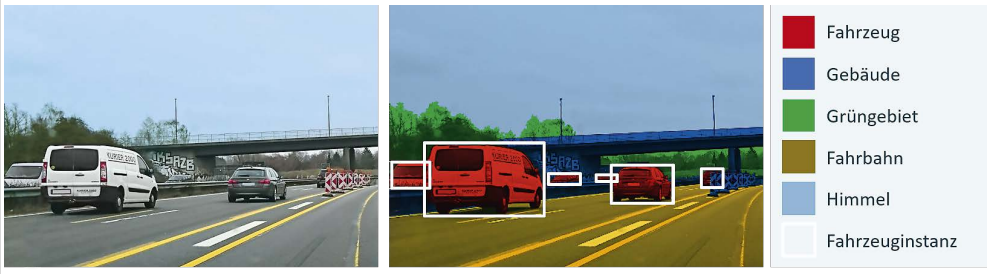

Abb. 2: Semantic (Instance) Segmentation klassifiziert Bildinhalte beim autonomen Fahren

Autoencoder kommen traditionell bei der Dimensionsreduktion oder beim Feature Learning zum Einsatz. In jüngerer Zeit hingegen werden sie häufig auch als generative Methode genutzt – insbesondere sind hier die Variational Autoencoder im Kontext des maschinellen Übersetzens sowie die Semantic Segmentation zu nennen.

Semantic Segmentation beim Autonomen Fahren

Fahrzeuge benötigen ein Konzept ihrer Umgebung, um selbstständig navigieren zu können. Dazu analysieren sie vor allem auch die Umgebungsbilder der Außenkameras und müssen auf diesen Aufnahmen Objekte voneinander unterscheiden können. Eine Möglichkeit besteht darin, jedem Pixel eine eigene Klasse zuzuordnen (zum Beispiel „Fahrzeug“, „Fußgänger“ oder „Verkehrszeichen“). Der zugehörige Problemtyp heißt Semantic Segmentation. Werden die klassengleichen Bereiche zudem in identifizierbare Objekte unterteilt, spricht man von Instance Segmentation. Autonomes Fahren benötigt diese Unterscheidbarkeit, da etwa bei Straßenschildern die Instanzen entscheidend sind, um richtig agieren zu können. Für den Anwendungsfall der Objektfreistellung ist die klassenweise Semantic Segmentation hinreichend.

Aufgrund ihrer generativen Eigenschaft sind Autoencoder für Semantic Segmentation gut geeignet – wenn dabei das Trainingsziel angepasst wird: Es geht nicht darum, das Ausgangsbild zu reproduzieren, sondern eine Segmentierung zu erzeugen. Allgemein bestehen die Samples aus dem Ausgangsbild (Input) und einem Zielbild (Target). Das Target ordnet jedem Pixel eine Klasse zu. Jede Klasse entspricht also genau einer Farbe (vgl. Abbildung 2).

Das Unet (vgl. [UNET17]) ist eine Autoencoder-Architektur aus dem medizinischen Bereich, deren Fokus auf der effektiven Nutzung des Trainingssets und der Anwendung von Daten-Augmentierung liegt. Diese spezielle Architektur gehört zu den Fully Convolutional Networks und wird häufig zur Semantic Segmentation und insbesondere zur Binary Segmentation verwendet.

Abbildung 3 zeigt das Unet. Der Encoder bildet in mehreren Schritten den Feature-Raum aus. Mit jedem Schritt vergrößert sich die Anzahl an Features, während sich die Größe der Eingabe verkleinert. Der Decoder bildet darauffolgend den Feature-Raum auf die ursprüngliche Größe der Eingabe zurück.

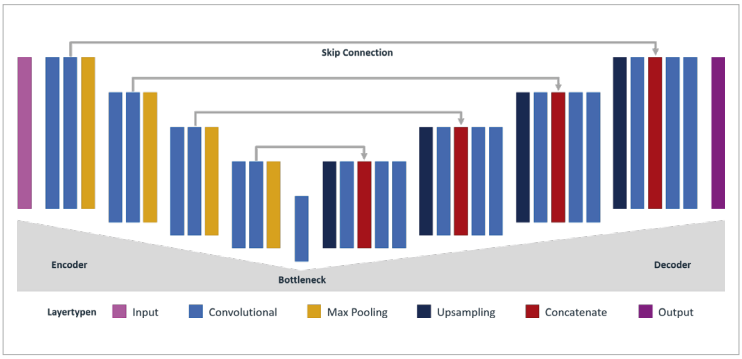

Abb. 3: Die Autoencoder-Architektur Unet

Encoder und Decoder sind mit Long Range Skip Connections verbunden. Diese erlauben es dem Decoder, auf Attribute des Encoders zuzugreifen und damit durch das Downsampling verloren gegangene, räumliche Information wiederherzustellen. Diese sind insbesondere wichtig, um eine pixelscharfe Rekonstruktion des Inputs zu erhalten. Encoder und Decoder sind durch das Bottleneck verbunden. Das Unet schließt mit

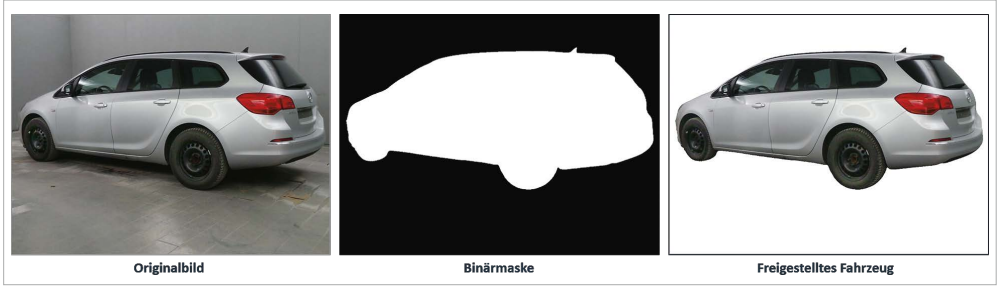

Abb. 4: Originalbild, Binärmaske und freigestelltes Fahrzeug mittels Binary Segmentation

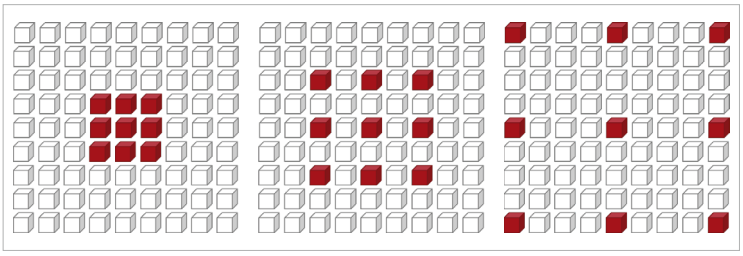

Abb. 5: 3x3-Filter mit den Ausdehnungsraten 1, 2 und 4

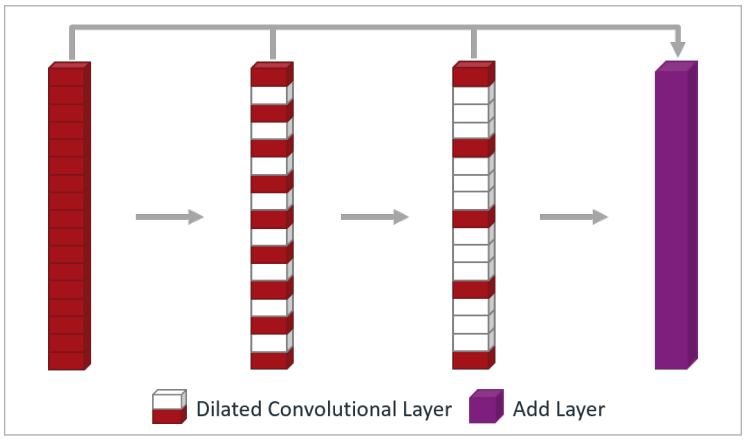

Abb. 6: Bottleneck des Dilated Unet mit kaskadierten Dilated Convolutional Layern

einem Convolutional Layer ab, dessen 1x1-Convolution der pixelweisen Klassifikation dient.

Spezialfall Binary Segmentation

Ein Spezialfall der Semantic Segmentation stellt die sogenannte Binary Segmentation dar, bei der nur genau zwei Klassen existieren. Ein typisches Einsatzgebiet ist die Segmentierung eines Bildes in Objekt und Hintergrund, also das Lernen einer entsprechenden Binärmaske.

Die Architektur des im Folgenden behandelten CNN orientiert sich am bereits vorgestellten Unet. Den Bottleneck haben wir im Vergleich zum klassischen Unet durch abgestufte Dilated Convolutions (vgl. [MSCA15]) ausgebaut (vgl. Abb. 5).

Es handelt sich hier um eine Sequenz von Convolutional Layern, welche mit einer zunehmenden Ausdehnungsrate (Dilation Rate) versehen sind. Diese Rate ist also ein Maß für die Granularität der Features. Der Decoder verwendet diese verschiedenen "Sichten" für die Rekonstruktion beziehungsweise die Generierung der Maske. Encoder, Decoder, Classification Layer und Skip Connections orientieren sich wieder an der klassischen Unet-Architektur.

Trainingsdaten und Augmentierung

Das im hier geschilderten Praxisbeispiel gewählte Trainingsset ist mit rund 5.000 Fahrzeugen und etwa 17.500 Samples recht klein. Zudem weist es nur wenige, unterschiedliche Fahrzeugperspektiven und -lagen auf. Auch der Bildhintergrund variiert nur schwach. Diese Effekte stellen einen Bias im Trainingsset dar, der durch dessen Erstellung begründet ist und motiviert den Einsatz von Daten-Augmentierung, das heißt der Anreicherung des Trainingssets um künstlich erzeugte Samples.

Dem Bias bei der Fahrzeugdarstellung wird mit Standard-Augmentierungen wie Skalierung und Translation auf dem Trainingsset begegnet. Der Bildhintergrund benötigt ein anderes Vorgehen: Während des Trainings ersetzen hier realistische oder synthetische Bilder den Originalhintergrund. Wir benötigen also Wissen darüber, welcher Teil eines Inputs den Hintergrund ausmacht. Aufgrund der Natur der Aufgabe enthält das gewählte Trainingsset dieses Wissen.

Abbildung 7 zeigt die Pipeline zur Batch-Generierung. Standard-Augmentierungen werden von den Keras-Generatoren ausgeführt und können mit der Hintergrund-Augmentierung kombiniert werden. Die synthetischen Hintergründe werden mithilfe eines parametrischen Frameworks generiert.

Die Hintergrund-Augmentierung zielt weniger auf die Accuracy der Modelle als vielmehr auf die Generalisierungsfähigkeit und damit die Praxistauglichkeit. Dieser Ansatz hat sich im Verlauf der Experimente als sehr wirksam herausgestellt.

Transfer Learning und Fine Tuning

In unserem Projekt konnten wir Transfer Learning und Fine Tuning (siehe Kasten) erfolgreich

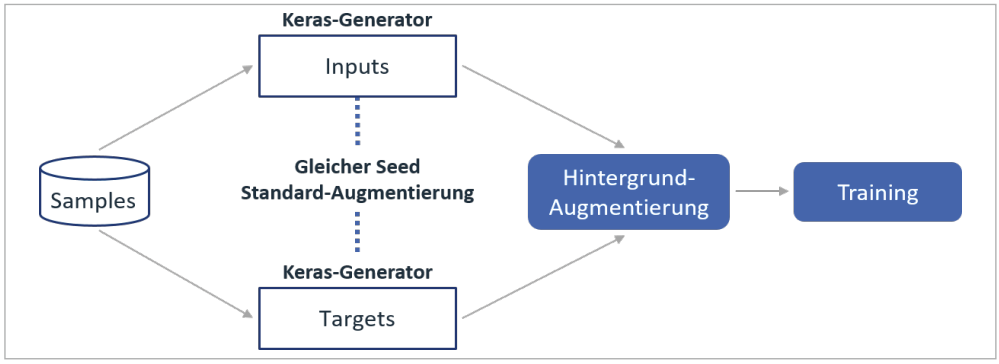

Abb. 7: Im Praxisbeispiel verwendete Pipeline

Transfer Learning und Fine Tuning

Transfer Learning steht für die Übertragung bereits gelernten Wissens auf eine verwandte oder sogar neue Domäne und/oder Aufgabenklasse. Technisch geschieht das in der Regel in zwei Schritten:

1. Architektur-Adaption

2. Adaption der lernbaren Parameter durch fokussiertes Training

Das populäre "ResNet50" (vgl. [DERES15]) zum Beispiel ist als auf dem ImageNet-Datensatz vortrainierter CNN-Klassifikator frei verfügbar und kennt 1.000 Klassen aus dem Bereich Tiere, Natur und Alltagsgegenstände. Ein solches Netz hat also bereits einen “alltagstauglichen visuellen Cortex" ausgebildet.

Angenommen, die neue Domäne wäre die Klassifikation 200 verschiedener amerikanischer Automarken, dann bestünde die Architektur-Adaption im Austausch des ResNet50-Klassifikators durch einen neuen, untrainierten Klassifikator mit 200 Output-Neuronen. Im zweiten Schritt würde dann das neu entstandene CNN zum Beispiel mit dem Stanford-Cars-Datensatz re-trainiert.

Dabei können dem Grunde nach zwei verschiedene Techniken angewandt werden:

- "Einfrieren" der bereits gelernten CNN-Filter (Gewichte/Biases) des ResNet50-Cortex, so dass nur der neue Klassifikator trainiert wird. Dann spricht man im engeren Sinne von Transfer Learning.
- Verwenden der bereits gelernten CNN-Filter lediglich im Sinne einer Gewichts-Initialisierung für das Re-Trainieren. Dieser Ansatz wird meist als Fine Tuning bezeichnet.

Natürlich sind auch (Layer-weise) Abstufungen und Mischformen über mehrere Trainingsstufen hinweg anwendbar.

Die wesentlichen Vorteile von Transfer Learning/Fine Tuning sind:

- Das Training benötigt in der Regel wesentlich weniger Daten und damit auch Rechenzeit, sofern die Zieldomäne genügend nah an der Transferbasis liegt.
- Die resultierenden Systeme abstrahieren/generalisieren in der Regel wesentlich besser als solche, die dem Problem der geringen Trainingsdaten allein durch Augmentierung begegnen (siehe oben).

im Bereich der Autoencoder-Architekturen einsetzen. Die Architektur-Adaption besteht dabei darin, den Encoder des Dilated Unet beispielsweise durch eine vortrainierte Instanz des ResNet50 (ohne Klassifikator-Kopf) zu ersetzen und geeignete Long Range Skip Connections zwischen den Residual Blocks des neuen Encoders und dem Decoder zu etablieren.

Neben Architekturvariationen haben wir verschiedene Experimente mit abgestuftem Transfer Learning und Fine Tuning durchgeführt. Als vielversprechend für unser Szenario hat sich dabei ein mehrstufiges Verfahren herausgestellt, bei dem der neue Encoder mit kleiner Learning Rate und sehr breiter Augmentierung zunächst für wenige Epochen antrainiert wird, bevor die Encoder-Layer eingefroren werden. Transfer Learning mit eingefrorenem Encoder gleich in der ersten Stufe anzuwenden, hat sich für unseren Kontext als nicht-optimal herausgestellt, zumal die vortrainierten ResNet50-Filter offenbar nicht genau genug auf unsere Zieldomäne passen.

Stichproben mit frei gewählten Testdaten aus dem Internet zeigen bessere Generalisierungseigenschaften der mittels Transfer Learning und Fine Tuning erzeugten Modelle als Modelle ohne diesen Ansatz.

Fazit

Für den hier vorgestellten Anwendungsfall, die Freistellung von Objekten auf Fotos, hat sich die binäre semantische Segmentierung als geeignetes Problem-Framing herausgestellt. Und eine Autoencoder-Architektur bietet einen dazu passenden Lösungsansatz. Dilated Unets und Derivate sind in diesem Bereich durchaus State-of-the-Art, wie unter anderem Kaggle-Challenges zeigen. Mit dem Transfer Learning-Ansatz, Hintergrund-Augmentierung und Ensembling verbesserte sich die Accuracy leicht und die Generalisierungsfähigkeit deutlich. Das finale Ensemble besitzt eine Accuracy von 99,4 Prozent auf dem Testdatensatz des Beispiels. Zudem sind die Ergebnisse von Bildfreistellungen im realen Betrieb sehr vielversprechend. Um das Modell weiter zu entwickeln, werden wir deshalb mehr und besser verteilte Trainingsdaten akquirieren sowie eine (automatisierte) Hyperparameter-Optimierung durchführen.

Literatur & Links

[UNET17] Ronneberger, O., Fischer, P. & Brox, T. (2017). U-Net: Convolutional Networks for Biomedical Image Segmentation. Bildverarbeitung für die Medizin.

[MSCA15] Yu, F. & Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions. *CoRR, abs/1511.07122*.

[DERES15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). Deep Residual Learning for Image Recognition.

Autoren

Thomas Stahl
(thomas.stahl@bmiag.de) ist Informatiker/Mathematiker und arbeitet als Lead Architect/CTO bei der b+m Informatik AG. Er beschäftigt sich seit dem Studium intensiv mit AI und Machine Learning.

Christian Koch
(christiankochjj@gmx.de) arbeitet als Data Scientist bei einer Bank in Hamburg. Er ist für das Modellieren von hochdimensionalen Daten und Automatisieren von komplexen Prozessen zuständig.

Stephan Wersig
(stephan.wersig@bmiag.de) arbeitet als Softwarearchitekt bei der b+m Informatik AG. Seine Schwerpunkte sind Machine Learning und Software-Engineering.

b+m Informatik

Innovation. Intelligent. Individuell.

ARTIFICIAL INTELLIGENCE AS A SERVICE

Mehr Innovation für Ihr Business

Nutzen Sie unsere AI-Kompetenz:

- Risikoprognosen (Klassifikation/Regression)
- Computer-Vision (Bildanalyse und -verarbeitung)
- Semantische Textanalyse
- Anomalieerkennung (z. B. Fraud oder IT-Infrastruktur)
- Data Mining

Erfahren Sie mehr auf **bmiag.de**

b+m Informatik