



**FOM Hochschule für Oekonomie & Management**

Hochschulzentrum Düsseldorf

**Master Thesis**

im Studiengang Big Data and Business Analytics

zur Erlangung des Grades eines

**Master of Science (M.Sc.)**

über das Thema

**Evaluierung und Visualisierung des Lernprozesses eines CNN zur  
Kategorisierung deutscher Straßenverkehrsschilder**

von

**Robin Maasjosthusmann**

Erstgutachter : Prof. Dr. Frank Lehrbass

Matrikelnummer : 498595

Abgabedatum : 28. Oktober 2020

## Inhaltsverzeichnis

<b>1 Theoretische Grundlagen</b>	<b>1</b>
1.1 Visualisierung der Entscheidung . . . . .	2
1.1.1 Activation Maximization . . . . .	2
1.1.2 Saliency Map . . . . .	3
1.1.3 Class Activation Mapping . . . . .	4
<b>Anhang</b>	<b>7</b>
<b>Literaturverzeichnis</b>	<b>8</b>

## **Abbildungsverzeichnis**

## **Tabellenverzeichnis**

## Abkürzungsverzeichnis

<b>AM</b>	Activation Maximization
<b>CAM</b>	Class Activation Mapping
<b>CNN</b>	Convolutional Neural Network
<b>NN</b>	Neural Network
<b>RGB</b>	Rot, Grün, Blau
<b>SM</b>	Saliency Map

# 1 Theoretische Grundlagen

In diesem Kapitel soll eine Übersicht über die wichtigsten theoretischen Grundlagen der Arbeiten gegeben werden. Die relevanten Fachbegriffe werden definiert und eine Herleitung der mathematischen Grundlagen der angewendeten Methoden wird dargestellt. Im folgenden Abschnitt wird auf Straßenverkehrsschilder und den verwendeten Datensatz eingegangen. Anschließend werden wichtige Aspekte von Neural Network (NN) und Convolutional Neural Network (CNN) erläutert und hergeleitet (Abschnitte ?? bzw. ??). Abschließend werden die verwendeten Algorithmen zur Visualisierung des Gelernten des CNN (Abschnitt 1.1) und die Python Bibliotheken die zum Einsatz kommen Abschnitt ?? vorgestellt.

## 1.1 Visualisierung der Entscheidung

### 1.1.1 Activation Maximization

Activation Maximization (AM) stellt eine Methode dar, um ein Bild  $x^*$  als Eingabe für das CNN zu erzeugen, welches die Ausgabe eines Neurons maximiert (siehe Gleichung 1, wobei  $\theta$  für die Parameter - Gewichte und Bias - des CNN steht). Dabei kann AM auf Neuronen aus jedem Layer eines CNN angewendet werden. Weiter verbreitet ist die Anwendung auf den letzten Convolutional Layer, in welcher jedes Neuron für eine Klasse steht. In diesem Fall erzeugt AM also ein Bild, welches die Ausgabe des Neurons für die Klasse  $k$  maximiert. Anhand dieses Bildes können Rückschlüsse gezogen werden, was das CNN gelernt hat um die Klasse  $k$  zu erkennen.

$$x^* = \underset{x}{\operatorname{argmax}} a_{i,l}(\theta, x) \quad (1)$$

Um das Bild zu generieren, wird ein iterativer Prozess verwendet. In diesem werden die Werte jedes Pixels im Bild in jedem Durchgang so angepasst, dass die Ausgabe des Zielneurons weiter maximiert wird. Der Prozess kann dabei in drei Schritte unterteilt werden:

1. Es wird ein Bild  $x = x_0$  mit zufälligen Werten für alle Pixel generiert (Rauschen). Dieses Bild wird dann in das CNN gegeben, um einen vorwärts Durchlauf durchzuführen.
2. Mithilfe von Backpropagation wird die partielle Ableitung des Zielneurons  $a_{i,l}$  im Bezug auf  $x$  durchgeführt ( $\frac{\partial a_{i,l}}{\partial x}$ ). Hierbei bleiben die Gewichte und Bias im CNN unverändert.
3. Mithilfe des berechneten Gradienten werden alle Pixeln in  $x$  angepasst, so dass die Aktivierung des Zielneurons weiter erhöht wird. Dabei wird mittels einer Schrittweite (auch Learning Rate,  $\eta$ ) gesteuert, wie stark die Anpassungen sind (Gleichung 2).

$$x \leftarrow x + \eta \cdot \frac{\partial a_{i,l}(\theta, x)}{\partial x} \quad (2)$$

Der Prozess wird beendet, wenn im erzeugten Bild kein Rauschen mehr vorhanden ist. Da dies ein Zustand ist, der bei CNN mit mehreren Layern selten erreicht wird, kann außerdem ein Schwellwert an Rauschen definiert werden, ab welchem der Prozess abbricht.

Ein weiteres Problem von komplexen CNN mit vielen Layern ist, dass die Muster in den späteren Layern für den Menschen immer weniger interpretierbar werden. Um diese Interpretierbarkeit zu erhöhen, können verschiedene Methoden zur Regularisierung angewendet werden. Hierbei wird der Gleichung ein Term  $-\lambda(x)$  hinzugefügt, der als Bias beim Erstellen des Bildes  $x^*$  wirkt (siehe Gleichung 3).

$$x^* = \operatorname{argmax}_x (a_{i,l}(\theta, x) - \lambda(x)) \quad (3)$$

Für  $\lambda(x)$  können verschiedene Methoden verwendet werden, bekannte Implementationen sind zum Beispiel  *$l_2$  decay*, *Gaussian blur*, oder *mean image initialization*.<sup>1,2,3</sup> Jede dieser Methoden hat seine eigenen Effekte,  *$l_2$  decay* verhindert zum Beispiel, dass einige extreme Pixelwerte das erzeugte Bild dominieren. Die verschiedenen Methoden können auch kombiniert werden, um die verschiedenen Effekte nutzen zu können.<sup>4</sup>

### 1.1.2 Saliency Map

Während AM den Fokus auf die Visualisierung der Aktivierung einzelner Neuronen legt, betrachtet Saliency Map (SM) das gesamte CNN. Die Grundidee hinter SM ist es, die Pixel zu ermitteln, welche den größten Einfluss auf die Zuordnung in eine bestimmte Klasse haben.

Nimmt man ein Bild  $I_0$ , eine Klasse  $c$  und ein trainiertes CNN mit der Bewertungsfunktion  $S_c(I)$ , ist das Ziel alle Pixel nach ihrem Einfluss auf den Wert  $S_c(I_0)$  zu sortieren. Im einfachen Fall einer linearen Bewertungsfunktion (siehe Gleichung 4, mit  $I$  als eindimensionaler Vektor des Bildes und  $w_c$  bzw.  $b_c$  als Parametervektoren des CNN) gibt bereits die Größe der Gewichte  $w$  an, wie wichtig die dazugehörigen Pixel für die Zuordnung in Klasse  $c$  sind.

$$S_c(I) = w_c^T I + b_c \quad (4)$$

Die Bewertungsfunktion innerhalb eines CNN ist allerdings eine nicht lineare Funktion, so dass dieses Prinzip nicht direkt übernommen werden kann. Geht man allerdings von dem Bild  $I_0$  aus, kann sich dem Wert von  $S_c(I)$  mit einer linearen Funktion genähert werden.

<sup>1</sup> Vgl. Simonyan, K., Vedaldi, A., Zisserman, A., 2014, S. 1–8.

<sup>2</sup> Vgl. Yosinski, J. et al., 2015, S. 1–12.

<sup>3</sup> Vgl. Wei, D. et al., 2015, S. 1–7.

<sup>4</sup> Vgl. Erhan, D. et al., 2009, S. 1–14.



Hierfür wird eine Taylorreihe erster Ordnung berechnet (siehe Gleichung 5, mit  $w$  gleich  $\frac{\partial S_c}{\partial I} \Big|_{I_0}$ ).

$$S_c(I) \approx w^T I + b \quad (5)$$

Eine mögliche Interpretation von  $\frac{\partial S_c}{\partial I} \Big|_{I_0}$  ist, dass die Größe der Ableitung anzeigt, welche Pixel am wenigsten verändert werden müssen, um das Ergebnis der Bewertungsfunktion am stärksten zu verändern. Es wird davon ausgegangen, dass diese Pixel mit der Position des Objekt der bewerteten Klasse im Bild korrespondieren.

Möchte man SM auf ein Farbbild (z.B. im Rot, Grün, Blau (RGB) Farbraum, wobei jede Farbe ein eigener Layer im Eingabebild ist) anwenden, müssen zunächst die Ableitungen  $w \left( \frac{\partial S_c}{\partial I} \Big|_{I_0} \right)$  mittels Backpropagation berechnet werden. Anschließend wird der maximale Wert der drei Layer als Wert für den Pixel in der SM Matrix  $M$  verwendet (siehe Gleichung 6, mit  $w_{h(i,j,c)}$  als Index eines Pixel an der Position  $i, j$  im Layer  $c$ ).<sup>5</sup>

$$M_{ij} = \max_c |w_{h(i,j,c)}| \quad (6)$$

### 1.1.3 Class Activation Mapping

Class Activation Mapping (CAM) wurde 2016 von Zhou et al. vorgestellt und wird auf bereits trainierte CNN angewendet. Das Ziel von CAM ist es, innerhalb eines Bildes die relevanten Bereiche für die Zuordnung zu einer Klasse zu markieren. Damit CAM verwendet werden kann, muss das CNN einen bestimmten Aufbau haben. Nach dem letzten Convolutional Layer muss es einen *Global Average Pooling Layer* vor dem *fully-connected Layer* geben.

Der letzte Convolutional Layer erzeugt dabei  $k$  Feature Maps, für jeden Kernel eine. Damit hat die Ausgabe des Convolutional Layers drei Dimensionen. Die Höhe  $v$ , die Breite  $u$  und die Anzahl von Feature Maps  $k$ . Pro Feature Map wird Global Average Pooling angewendet, das heißt es wird der Mittelwert über alle Pixel in einer Feature Maps  $A^k$  gebildet (siehe Gleichung 7).

$$avgpool_k = \frac{1}{z} \sum_{i=1}^u \sum_{j=1}^v A_{ij}^k \quad (7)$$

<sup>5</sup> Vgl. Simonyan, K., Vedaldi, A., Zisserman, A., 2014, S. 3f.

Nachdem Global Average Pooling auf alle Feature Maps angewendet wurde, bleibt ein Vektor mit  $k$  Nummer über. Diese werden in einen fully-connected Layer gegeben. Die Gewichte  $w$  zwischen dem Ergebnis des Global Average Pooling und dem fully-connected Layer, werden dabei während des Trainingprozesses des Netzwerkes gelernt.

Zur Erzeugung der Ergebnis Matrix - bei visueller Darstellung auch Heatmap genannt - von CAM, werden die gelernten Gewichte  $w$  mit den Feature Maps aus der Aktivierung des letzten Convolutional Layers  $A_k$  multipliziert. Zu beachten ist hierbei, dass CAM die Aktivierung für das aktuelle Bild und auf eine angegebene Klasse darstellt. Gleichung 8 zeigt die Formel zur Berechnung von CAM für die Klasse  $c$ .

$$cam_c = \sum_k w_k^c A^k \quad (8)$$

Das Ergebnis von CAM ist somit eine Matrix mit den Maßen der Feature Maps des letzten Convolutional Layer. Häufig wird dieses hochskaliert, so dass die Heatmap über das Originalbild gelegt werden kann.<sup>6</sup>

Um die Beschränkung der Architektur des CNN durch CAM aufzuheben, haben Selvaraju et al. 2016 mit Grad-CAM die Verallgemeinerung von CAM vorgeschlagen.

Bei dieser Verallgemeinerung benötigt das CNN keinen Global Average Pooling Layer nach dem letzten Convolutional Layer mehr. Es kann eine beliebige Anzahl von Layern nach dem Convolutional Layer vorhanden sein, die einzige Bedingung für diese ist, dass sie differenzierbar sind. Laut den Autoren von Grad-CAM wird dabei der letzte Convolutional Layer als Grundlage der Visualisierung verwendet, da zu erwarten ist, dass dieser das beste Gleichgewicht zwischen Interpretierbarkeit und räumlicher Auflösung bietet.

Wie bei CAM, werden alle  $k$  Feature Maps gewichtet addiert um die Ergebnismatrix zu erhalten. Der Unterschied liegt in der Berechnung der Gewichte. Während CAM die Gewichte  $w$  des CNN verwendet, berechnet Grad-CAM eigene Gewichte  $\alpha$ , welche auf den Gradienten basieren.

Als Grundlage für die  $\alpha$  Werte verwendet Grad-CAM die Ausgabewerte des CNN, bevor diese in den Softmax Layer gegeben werden ( $y^c$ ). Der eigentliche Prozess von Grad-CAM kann in drei Schritte unterteilt werden. Zunächst wird der Gradient von  $y^c$  mit Bezug auf die Feature Maps  $A^k$  berechnet ( $\frac{\partial y^c}{\partial A^k}$ ). Dieser ist abhängig von dem aktuellen Eingabebild, da dieses die Grundlage für die Feature Maps und  $y^c$  liefert. Wie bei CAM ergibt ein zweidimensionales Eingabebild ein dreidimensionalen Gradienten ( $v, u, k$ ). Im zweiten

<sup>6</sup> Vgl. Zhou, B. et al., 2016, S. 2-4.

Schritt werden die  $\alpha$  Werte berechnet. Jede Feature Map  $k$  erhält dabei einen Wert für die gewählte Klasse  $c$  ( $\alpha_k^c$ ). Für diesen Wert wird über die Höhe und Breite der Feature Map gelaufen um Global Average Pooling durchzuführen (siehe Gleichung 9). Dies bedeutet, dass am Ende dieses Schrittes eine Matrix mit der Form  $[1,1,k]$  oder vereinfacht der Vektor  $[k]$  vorliegt. Jede Feature Map hat somit einen  $\alpha$  Wert, der als Gewicht verwendet werden kann.

$$\alpha_k^c = \frac{1}{Z} \sum_{i=0}^u \sum_{j=0}^v \frac{\partial y^c}{\partial A_{ij}^k} \quad (9)$$

Im dritten Schritt wird nun die Ergebnismatrix von Grad-CAM berechnet. Diese ist die lineare Kombination der gewichteten Feature Maps, wobei das Ergebnis in die ReLU Funktion gegeben wird (siehe Gleichung 10). Die ReLU Funktion sorgt dabei dafür, dass nur positive Werte einbezogen werden, da alle negativen Werte auf 0 gesetzt werden.

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right) \quad (10)$$

Wie bei CAM hat die Ergebnismatrix/Heatmap von Grad-CAM die Form  $u \times x$ , sie muss also hoch skaliert werden, damit sie über das Eingangsbild gelegt werden kann.<sup>7</sup>

<sup>7</sup> Vgl. Selvaraju, R. R. et al., 2020, S. 4-6.

## Anhang

### Anhang 1: Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut sein kann.







#### Anhang 1.1: Weitere Gliederungsebene

Auch eine zweite Gliederungsebene ist möglich.

### Anhang 2: Bilder

Auch mit Bildern. Diese tauchen nicht im Abbildungsverzeichnis auf.

#### Abbildung 1: Beispielbild

Name	Änderungsdatum	Typ	Größe
 abbildungen	29.08.2013 01:25	Dateiordner	
 kapitel	29.08.2013 00:55	Dateiordner	
 literatur	31.08.2013 18:17	Dateiordner	
 skripte	01.09.2013 00:10	Dateiordner	
 compile.bat	31.08.2013 20:11	Windows-Batchda...	1 KB
 thesis_main.tex	01.09.2013 00:25	LaTeX Document	5 KB

## Literaturverzeichnis

- Erhan, Dumitru, Bengio, Y., Courville, Aaron, Vincent, Pascal* (2009): Visualizing Higher-Layer Features of a Deep Network, in: Technical Report, Univeristé de Montréal (2009), Nr. 1341, S. 4–6, [Zugriff: 2020-10-22]
- Selvaraju, Ramprasaath R., Cogswell, Michael, Das, Abhishek, Vedantam, Ramakrishna, Parikh, Devi, Batra, Dhruv* (2020): Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in: *Int J Comput Vis*, 128 (2020), Nr. 2, S. 336–359, [Zugriff: 2020-06-11]
- Simonyan, Karen, Vedaldi, Andrea, Zisserman, Andrew* (2014): Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, in: *Bengio, Yoshua, LeCun, Yann* (Hrsg.), 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings, International Conference on Learning Representations, Banff, AB, Canada, 2014, S. 1–8
- Wei, Donglai, Zhou, Bolei, Torralba, Antonio, Freeman, William T.* (2015): Understanding Intra-Class Knowledge inside CNN, in: *CoRR*, abs/1507.02379 (2015), [Zugriff: 2020-10-26]
- Yosinski, Jason, Clune, Jeff, Nguyen, Anh Mai, Fuchs, Thomas J., Lipson, Hod* (2015): Understanding Neural Networks through Deep Visualization, in: *CoRR*, abs/1506.06579 (2015), [Zugriff: 2020-10-26]
- Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, Torralba, Antonio* (2016): Learning Deep Features for Discriminative Localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 2016-06, S. 2921–2929, [Zugriff: 2020-06-16]

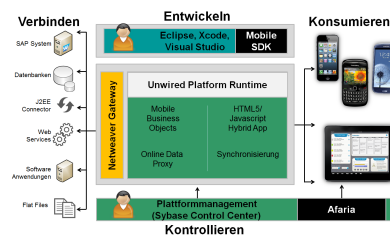
---

## Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit **einverstanden/nicht einverstanden**, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Düsseldorf, 28.10.2020

(Ort, Datum)



(Eigenhändige Unterschrift)