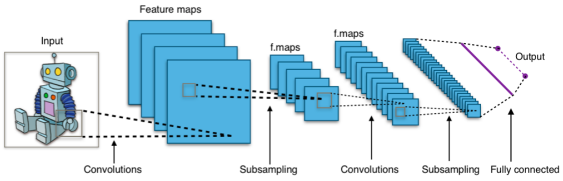
Convolutional Neural Networks - Handout

von Bastian Bertholdt und Ajit Parikh

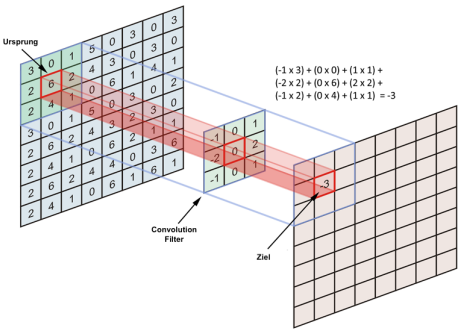
1 Topolgie

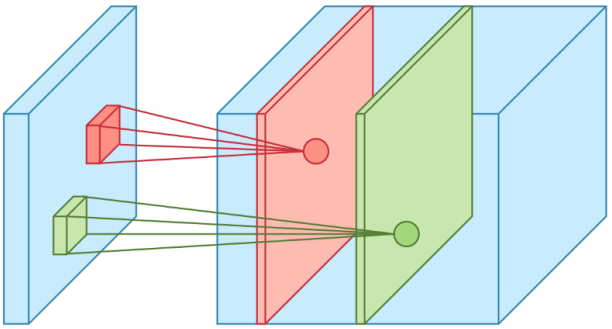
- Mehrschichtige Deep-Learning Architektur
- Jede Schicht generiert sogenannte feature maps, die zusammen die Eingabe der nächsten Schicht bilden
- Neuronen sind nicht vollständig, sondern nur lokal miteinander verknüpft
- Neuronenverbindungen haben teilweise die selbe Gewichtung
- Am Ende: Vektorisierung der feature-maps bildet Eingabe der vollständig-vernetzten Schichten



1.1 Convolutional Layer

- Führt Faltungen der Eingabe (I) mit mehreren Filtern (K) durch
- Faltung (convolution): $(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$
- Kreuzkorrelation (cross-correlation): $(I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$

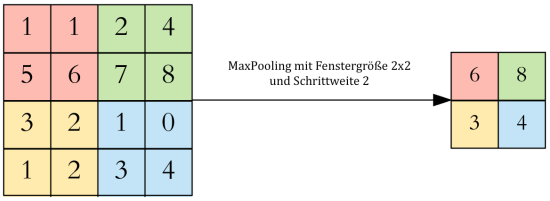




- Transformierung des Outputs durch Nicht-lineare Aktivierungsfunktion: ReLU $(f(x) = max(0,x))$

1.2 Pooling Layer

- Aggregiert feature map Werte, in einer lokal beschränkten Region zu einem Wert
- Reduzierung der Dimensionalität und Anzahl der Parameter



2 Backpropagation

- Grundidee: Fehlergradienten bzgl Filter und Eingabe lassen sich durch Convolutions berechnen
- Sei E der Fehler, X die Eingabe, F der Filter und O die Ausgabe mit $O = \text{Convolution}(X, F)$
- Des Weiteren sei $\delta F := \frac{\partial E}{\partial F}$, $\delta O := \frac{\partial E}{\partial O}$ und $\delta X := \frac{\partial E}{\partial X}$, dann

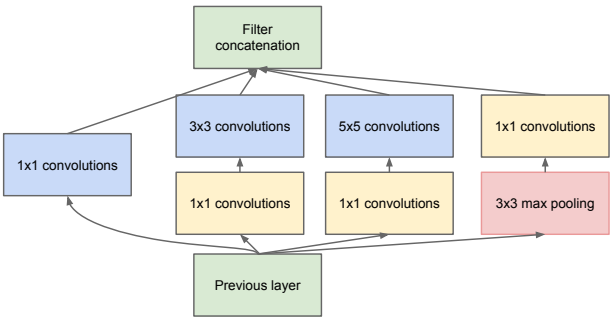
$$\begin{aligned} \delta F &= \textit{Convolution}\ (X, \delta O) \\ \delta X &= \ \textit{Full Convolution}\ (\delta O, \textit{Flip}\ (F)) \end{aligned}$$

3 Transpose Convolution (Deconvolution)[1]

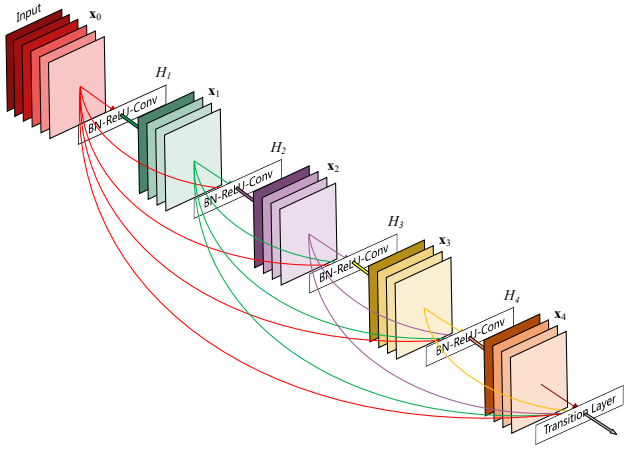
- Kann für Upsampling verwendet werden
- Darstellung der Faltung als einfache Matrixmultiplikation ($\vec{k}$...Kernel, $\vec{x}$...Input)
- Transponieung der Kernelmatrix K

$$K^T \vec{x} = \vec{k} *^T \vec{x}, \quad \begin{bmatrix} k_1 & 0 & 0 & 0 \\ k_2 & k_1 & 0 & 0 \\ k_3 & k_2 & k_1 & 0 \\ 0 & k_3 & k_2 & k_1 \\ 0 & 0 & k_3 & k_2 \\ 0 & 0 & 0 & k_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} k_1 x_1 \\ k_2 x_1 + k_1 x_2 \\ k_3 x_1 + k_2 x_2 + k_1 x_3 \\ k_3 x_2 + k_2 x_3 + k_1 x_4 \\ k_3 x_3 + k_2 x_4 \\ k_1 x_4 \end{bmatrix}$$

4 Varianten: Inception und DensNet



(a) Inception mit Bottleneck Layer: Alle Ausgänge der verschiedenen Operationen werden nach Feature Pooling (1x1) in einem Ausgang zusammen gefasst. Der Ausgang eines Blockes ist gleichzeitig der Eingang des nächsten Blocks. [3]



(b) Feature Propagation in einem DensNet Block: Jede Schicht ist zur Reduzierung der Parameter und für einen besseren Informationsfluss mit allen vorherigen Schichten verbunden. [2]

Literatur

[1] Serena Yeung Fei-Fei Li, Justin Johnson. Lecture 11: Detection and segmentation.

[2] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.