Universität Rostock

Traditio et Innovatio

Datenbankgestützte Erkennung von Mustern in trainierten gefalteten neuronalen Netzen

Verteidigung, 14. September 2022

Florian Baldauf
Universität Rostock, Institut für Mathematik

Ablauf

Ablauf

Einführung Mathematik und Künstlichen Intelligenz

- Neuronale Netze
- Erkennung von Mustern

Gefaltete neuronale Netze (CNN)

- Funktionsweise
- Backpropagation

Datenbankgestützte Umsetzung

- PArADISE-Projekt
- Aufzählungen
- Theorem / Beweis / andere Boxen

Details zum Uni Rostock Style

- Einbindung des Styles
- LaTeXund pdfLaTeX
- Farbschema
- Tabellen

Ablauf

Ablauf

Ablauf

Einführung Mathematik und Künstlichen Intelligenz

- Neuronale Netze
- Erkennung von Mustern

Gefaltete neuronale Netze (CNN)

- Funktionsweise
- Backpropagation

Datenbankgestützte Umsetzung

- PArADISE-Projekt
- Aufzählungen
- Theorem / Beweis / andere Boxen

Details zum Uni Rostock Style

- Einbindung des Styles
- LaTeXund pdfLaTeX
- Farbschema
- Tabellen

KI: Erfolg in vielen Disziplinen

Meine Problemstellungen

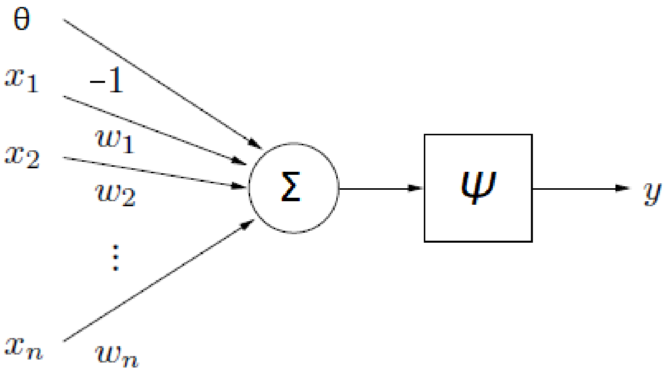
TODO

Erste Ansätze

Ideen von McCulloch und Pitts (1943):

- Entwicklung einer algorithmischen Beschreibung des Lernens
- menschliche Gehirn als Vorbild $\rightarrow$ das abstrakte Neuron

Das Neuron



Definition (Neuron)

Für eine gegebene Funktion $\phi : \mathbb{R} \to \mathbb{R}$, einen Vektor $w \in \mathbb{R}^n$ und ein Skalar $b \in \mathbb{R}$ wird die Funktion

$$\Phi : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto \phi(w^T x - b) =: y,$$

Neuron genannt.

Mögliche Aktivierungsfunktionen sind:

$$\begin{aligned} \text{Identität}: &\quad \phi(x) = x, \\ \text{Logistische Funktion}: &\quad \phi(x) = \frac{1}{1 + e^{-x}}, \\ \text{ReLU (rectified linear unit)}: &\quad \phi(x) = \max\{0, x\}. \end{aligned}$$

Definition (Neuron)

Für eine gegebene Funktion $\phi : \mathbb{R} \to \mathbb{R}$, einen Vektor $w \in \mathbb{R}^n$ und ein Skalar $b \in \mathbb{R}$ wird die Funktion

$$\Phi : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto \phi(w^T x - b) =: y,$$

Neuron genannt.

Mögliche Aktivierungsfunktionen sind:

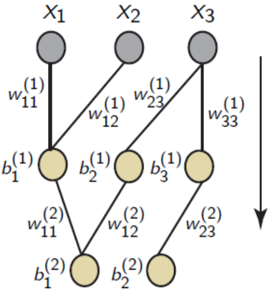
$$\begin{aligned} \text{Identität}: \quad & \phi(x) = x, \\ \text{Logistische Funktion}: \quad & \phi(x) = \frac{1}{1 + \mathrm{e}^{-x}}, \\ \text{ReLU (rectified linear unit)}: \quad & \phi(x) = \max\{0, x\}. \end{aligned}$$

$$\frac{\partial u}{\partial t} = \Delta u + f(u)$$

Die Verknüpfung von Neuronen zu Netzen führt zur Komposition von affin linearen Abbildungen und Aktivierungsfunktionen.

Ein Beispiel(Quelle Gitta Kutyniok [])

$$\Phi : \mathbb{R}^3 \to \mathbb{R}^2, \quad \Phi(x) = W^{(2)}\phi(W^{(1)}x - b^{(1)}) - b^{(2)}$$



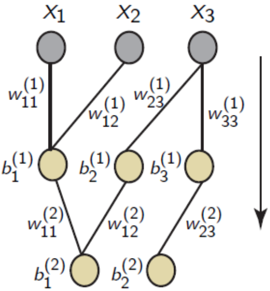
$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & 0 \\ 0 & 0 & w_{23}^{(1)} \\ 0 & 0 & w_{33}^{(1)} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(1)} & 0 \\ 0 & 0 & w_{23}^{(2)} \end{pmatrix}$$

Die Verknüpfung von Neuronen zu Netzen führt zur Komposition von affin linearen Abbildungen und Aktivierungsfunktionen.

Ein Beispiel(Quelle Gitta Kutyniok [])

$$\Phi : \mathbb{R}^3 \to \mathbb{R}^2, \quad \Phi(x) = W^{(2)}\phi(W^{(1)}x - b^{(1)}) - b^{(2)}$$



$$W^{(1)} = \begin{pmatrix} w^{(1)}_{11} & w^{(1)}_{12} & 0 \\ 0 & 0 & w^{(1)}_{23} \\ 0 & 0 & w^{(1)}_{33} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w^{(2)}_{11} & w^{(1)}_{12} & 0 \\ 0 & 0 & w^{(2)}_{23} \end{pmatrix}$$

Neuronale Netze

Definition (Vorwärtsgerichtetes neuronales Netz(FNN))

Seien

- $s_0 \in \mathbb{N}$: Dimension der Eingabeschicht,
- L: Anzahl der Schichten,
- ϕ: eine (nichtlineare) Aktivierungsfunktion,
- Neuronen:

  $T_\ell : \mathbb{R}^{s_{\ell-1}} \to \mathbb{R}^{s_\ell}, \ \ell = 1, \dots L,$ mit $T_\ell(x) = W^{(\ell)}x + b^{(\ell)}$

Dann ist $\Phi : \mathbb{R}^{s_0} \to \mathbb{R}^{s_L}$ durch die Komposition

$$\Phi(x) = T_L \circ \phi \circ T_{L-1} \circ \phi \circ \dots \circ \phi \circ T_1(x), \ x \in \mathbb{R}^{s_0}$$

erklärt. Der Vektor $y := \Phi(x)$ wird Ausgabe des (tiefen) FNN genannt.

Neuronale Netze

Definition (Vorwärtsgerichtetes neuronales Netz(FNN))

Seien

- $s_0 \in \mathbb{N}$: Dimension der Eingabeschicht,
- L: Anzahl der Schichten,
- ϕ: eine (nichtlineare) Aktivierungsfunktion,
- Neuronen:
  $T_\ell : \mathbb{R}^{s_{\ell-1}} \to \mathbb{R}^{s_\ell},\ \ell = 1, \dots L,$ mit $T_\ell(x) = W^{(\ell)}x + b^{(\ell)}$

Dann ist $\Phi : \mathbb{R}^{s_0} \to \mathbb{R}^{s_L}$ durch die Komposition

$$\Phi(x) = T_L \circ \phi \circ T_{L-1} \circ \phi \circ \dots \circ \phi \circ T_1(x),\ x \in \mathbb{R}^{s_0}$$

erklärt. Der Vektor $y := \Phi(x)$ wird Ausgabe des (tiefen) FNN genannt.

Veranschaulichung

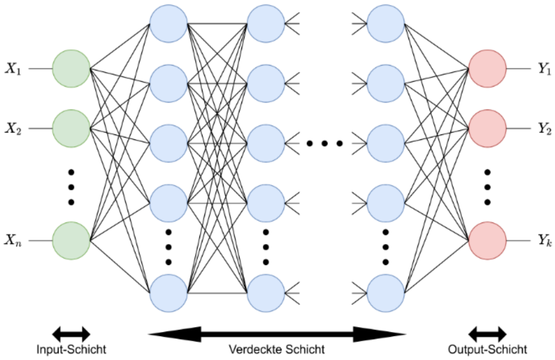


Abbildung: Die Abbildung ist aus Fenske[**?**,] entnommen.

Überwachtes Lernen

Klassifikation von Objekten mithilfe vonFFN

Gegeben:

- Menge $\mathcal{M} = \{x_i \in \mathbb{R}^n \ : \ 1 \leq i \leq m\}$,
- Funktion $f : \mathcal{M} \to \{1, \dots k\}$,
- also eine endliche Menge von Tupeln der Form $(x_i, f(x_i))_{i=1}^m$, auch Trainingsmenge $\mathcal{T}$ genannt.

Gesucht: affin lineare Funktionen $(T_\ell)_{\ell=1}^L = (W^{(\ell)} \cdot + b^{(\ell)})_{\ell=1}^L$ (oft Netzeingabe, Input des Neurons) sodass das Problem

$$\min_{(W^{(\ell)}, b^{(\ell)})_\ell} \sum_{i=1}^m \mathcal{E}(\Phi(x_i), f(x_i)) + \lambda \mathcal{R}((W^{(\ell)}, b^{(\ell)})_l)$$

gelöst wird. Hier ist $\mathcal{E}(\mathcal{T}, \mathcal{W})$ eine wählbare Zielfunktion.

Überwachtes Lernen

Klassifikation von Objekten mithilfe vonFFN

Gegeben:

- Menge $\mathcal{M} = \{x_i \in \mathbb{R}^n \,:\, 1 \leq i \leq m\}$,
- Funktion $f : \mathcal{M} \to \{1, \ldots k\}$,
- also eine endliche Menge von Tupeln der Form $(x_i, f(x_i))_{i=1}^m$, auch Trainingsmenge $\mathcal{T}$ genannt.

Gesucht: affin lineare Funktionen $(T_\ell)_{\ell=1}^L = (W^{(\ell)} \cdot + b^{(\ell)})_{\ell=1}^L$ (oft Netzeingabe, Input des Neurons) sodass das Problem

$$\min_{(W^{(\ell)}, b^{(\ell)})_\ell} \sum_{i=1}^m \mathcal{E}(\Phi(x_i), f(x_i)) + \lambda \mathcal{R}((W^{(\ell)}, b^{(\ell)})_l)$$

gelöst wird. Hier ist $\mathcal{E}(\mathcal{T}, \mathcal{W})$ eine wählbare Zielfunktion.

Training von FFN

Training mithilfe des Gradientenverfahren als iteratives Verfahren mit dem langfristigen Ziel $\Phi(x_i) \approx f(x_i)$ für die Testdaten

$\rightarrow$ Backpropagation mit mehrdimensionaler Kettenregel

Sei $\mathcal{W} = \{(W^{(\ell)}, b^{(\ell)}) \,:\, 1 \leq \ell \leq L\}$ die Menge der Modellparameter.

Idee:

- Bestimme Gradient $\Delta_n = \nabla_{\mathcal{W}} \mathcal{E}(\mathcal{T}, \mathcal{W})$,
- Aktualisiere Parameter $\mathcal{W}_{n+1} = \mathcal{W}_n + \lambda \Delta_n$,
- solange ein Abbruchkriterium nicht erfüllt ist.

Training von FFN

Training mithilfe des Gradientenverfahren als iteratives Verfahren mit dem langfristigen Ziel $\Phi(x_i) \approx f(x_i)$ für die Testdaten

$\rightarrow$ Backpropagation mit mehrdimensionaler Kettenregel

Sei $\mathcal{W} = \{(W^{(\ell)}, b^{(\ell)}) \,:\, 1 \leq \ell \leq L\}$ die Menge der Modellparameter.

Idee:

- Bestimme Gradient $\Delta_n = \nabla_{\mathcal{W}} \mathcal{E}(\mathcal{T}, \mathcal{W})$,
- Aktualisiere Parameter $\mathcal{W}_{n+1} = \mathcal{W}_n + \lambda \Delta_n$,
- solange ein Abbruchkriterium nicht erfüllt ist.

Training von FFN

Training mithilfe des Gradientenverfahren als iteratives Verfahren mit dem langfristigen Ziel $\Phi(x_i) \approx f(x_i)$ für die Testdaten

$\to$ Backpropagation mit mehrdimensionaler Kettenregel

Sei $\mathcal{W} = \{(W^{(\ell)}, b^{(\ell)}) \, : \, 1 \le \ell \le L\}$ die Menge der Modellparameter.

Idee:

- Bestimme Gradient $\Delta_n = \nabla_{\mathcal{W}} \mathcal{E}(\mathcal{T}, \mathcal{W})$,
- Aktualisiere Parameter $\mathcal{W}_{n+1} = \mathcal{W}_n + \lambda \Delta_n$,
- solange ein Abbruchkriterium nicht erfüllt ist.

Training von FFN

Training mithilfe des Gradientenverfahren als iteratives Verfahren mit dem langfristigen Ziel $\Phi(x_i) \approx f(x_i)$ für die Testdaten

$\rightarrow$ Backpropagation mit mehrdimensionaler Kettenregel

Sei $\mathcal{W} = \{(W^{(\ell)}, b^{(\ell)}) : 1 \leq \ell \leq L\}$ die Menge der Modellparameter.

Idee:

- Bestimme Gradient $\Delta_n = \nabla_{\mathcal{W}} \mathcal{E}(\mathcal{T}, \mathcal{W})$,
- Aktualisiere Parameter $\mathcal{W}_{n+1} = \mathcal{W}_n + \lambda \Delta_n$,
- solange ein Abbruchkriterium nicht erfüllt ist.

Wähle Zielfunktion $\mathcal{E}(x, \mathcal{W}) = \frac{1}{2} \sum_{(x,c)\in\mathcal{T}} ||y - t||_2^2$ mit dem Zielvektor $t = t(x, c)$. Online-Backpropagation $\to$ $E_x = \frac{1}{2} \sum_{k=1}^{s_L} (y_k - t_k)^2$.

Satz (Online-Backpropagation)

Seien mit $w_{j,i}^{(\ell)}$ und $b_j^{(\ell)}$ lernbare Parameter der Schicht ℓ bezeichnet. Dann gilt

$$\Delta w_{j,i}^{(\ell)} = \frac{\partial E_x}{\partial w_{j,i}^{(\ell)}} = \delta_j^{(\ell)} z_i^{(\ell-1)}, \quad \Delta b_j^{(\ell)} = \frac{\partial E_x}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)}$$

mit

$$\delta_j^{(\ell)} = \begin{cases} (y_j - t_j)\phi'(T_j^{(L)}), & \text{wenn } j \text{ ein Ausgabeneuron ist,} \\ \left(\sum_k \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)}\right) \phi'(T_j^{(\ell)}), & \text{sonst.} \end{cases}$$

Wähle Zielfunktion $\mathcal{E}(x, \mathcal{W}) = \frac{1}{2} \sum_{(x,c)\in\mathcal{T}} ||y - t||_2^2$ mit dem Zielvektor $t = t(x, c)$. Online-Backpropagation $\to E_x = \frac{1}{2} \sum_{k=1}^{s_L} (y_k - t_k)^2$.

Satz (Online-Backpropagation)

Seien mit $w_{j,i}^{(\ell)}$ und $b_j^{(\ell)}$ lernbare Parameter der Schicht ℓ bezeichnet. Dann gilt

$$\Delta w_{j,i}^{(\ell)} = \frac{\partial E_x}{\partial w_{j,i}^{(\ell)}} = \delta_j^{(\ell)} z_i^{(\ell-1)}, \quad \Delta b_j^{(\ell)} = \frac{\partial E_x}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)}$$

mit

$$\delta_j^{(\ell)} = \begin{cases} (y_j - t_j)\phi'(T_j^{(L)}), & \text{wenn } j \text{ ein Ausgabeneuron ist,} \\ \left(\sum_k \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)}\right) \phi'(T_j^{(\ell)}), & \text{sonst.} \end{cases}$$

Lineare Algebra

Kompakte Darstellung:

$$\Delta W^{(\ell)} = \delta^{(\ell)} \cdot {z^{(\ell-1)}}^T, \quad \Delta b^{(\ell)} = \delta^{(\ell)}$$

mit

$$\delta^{(\ell)} = \begin{cases} (y - t) \odot \phi'(T^{(L)}), & \text{wenn } \ell = L, \\ \left({W^{(\ell+1)}}^T \delta^{(\ell+1)}\right) \odot \phi'(T^{(\ell)}), & \text{wenn } \ell \in \{1, \dots L-1\}, \end{cases}$$

und dem dyadischen Produkt, Matrixvektorprodukt sowie die elementweise Multiplikation.

Klassifikation von Grauwertbildern

Ein paar Probleme

Beispiel: Bild (1000×1000) und jedes Pixel als Merkmal

FFN mit 10 Ausgabeneuronen $\rightarrow 10^7 + 10$ freie Parameter

$\rightarrow$ Reduzierung durch Parameter Sharing, sparse connectivity

Korrelationen zwischen benachbarten Neuronen?

$\rightarrow$ Filter, zum Beispiel zur Kantendetektierung

Convolutional Neural Networks (CNN)

Unterschiede zu klassischen FFN:

- Berechnung der Netzeingabe.
- Mehrere Neuronen teilen sich Gewichte.
- Es liegt kein vollständiger Graph vor.

$\rightarrow$ Reduzierung der Parameteranzahl

$\rightarrow$ Anpassung der Vorwärts- und Rückwärtsrechnung.

Convolutional Neural Networks (CNN)

Unterschiede zu klassischen FFN:

- Berechnung der Netzeingabe.
- Mehrere Neuronen teilen sich Gewichte.
- Es liegt kein vollständiger Graph vor.

$\rightarrow$ Reduzierung der Parameteranzahl

$\rightarrow$ Anpassung der Vorwärts- und Rückwärtsrechnung.

Convolutional Neural Networks (CNN)

Unterschiede zu klassischen FFN:

- Berechnung der Netzeingabe.
- Mehrere Neuronen teilen sich Gewichte.
- Es liegt kein vollständiger Graph vor.

$\rightarrow$ Reduzierung der Parameteranzahl

$\rightarrow$ Anpassung der Vorwärts- und Rückwärtsrechnung.

Convolutional Neural Networks (CNN)

Unterschiede zu klassischen FFN:

- Berechnung der Netzeingabe.
- Mehrere Neuronen teilen sich Gewichte.
- Es liegt kein vollständiger Graph vor.

$\rightarrow$ Reduzierung der Parameteranzahl

$\rightarrow$ Anpassung der Vorwärts- und Rückwärtsrechnung.

Convolutional Neural Networks (CNN)

Unterschiede zu klassischen FFN:

- Berechnung der Netzeingabe.
- Mehrere Neuronen teilen sich Gewichte.
- Es liegt kein vollständiger Graph vor.

$\rightarrow$ Reduzierung der Parameteranzahl

$\rightarrow$ Anpassung der Vorwärts- und Rückwärtsrechnung.

Convolutional Neural Networks (CNN)

vs. FFN

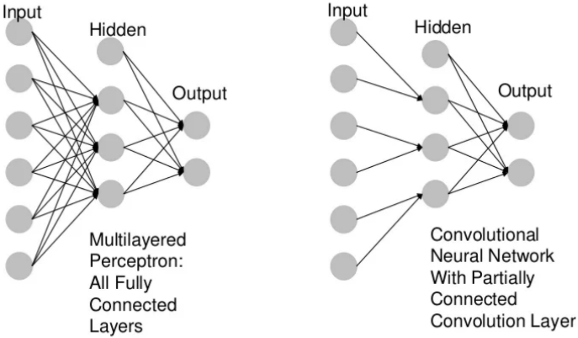


Abbildung: Architektur von FFN(links) und CNN(rechts). Die Abbildung wurde aus Bhandare[**?**] entnommen.

Convolutional Neural Networks (CNN)

Arbeitsweise

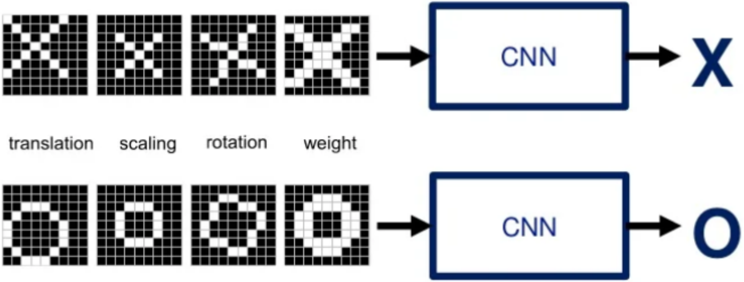


Abbildung: Klassifikation von Grauwertbildern durch CNN. Die Abbildung wurde aus Bhandare[?] entnommen.

Convolutional Neural Networks (CNN)

Aufbau

- Faltungsschicht: Für Neuronen, welche sich lokal in kleinen Regionen befinden, werden Skalarprodukte mit lernbaren Gewichten (Kernen) berechnet $\rightarrow$ Merkmalskarten.
- Aktivierungsschicht: Eine Aktivierungsfunktion, z.B. ReLU, wird elementweise auf die Neuronenaktivierungen der Karten angewendet.
- Pooling-Schicht: Durch sogenanntes downsampling wird die räumliche Dimensionen der Merkmalskarten verringert.
- FFN: Ein Vorwärtsgerichtetes Netz wird benutzt, um die Klassifikation zu berechnen.

Convolutional Neural Networks (CNN)

Aufbau

- Faltungsschicht: Für Neuronen, welche sich lokal in kleinen Regionen befinden, werden Skalarprodukte mit lernbaren Gewichten (Kernen) berechnet $\rightarrow$ Merkmalskarten.
- Aktivierungsschicht: Eine Aktivierungsfunktion, z.B. ReLU, wird elementweise auf die Neuronenaktivierungen der Karten angewendet.
- Pooling-Schicht: Durch sogenanntes downsampling wird die räumliche Dimensionen der Merkmalskarten verringert.
- FFN: Ein Vorwärtsgerichtetes Netz wird benutzt, um die Klassifikation zu berechnen.

Convolutional Neural Networks (CNN)

Aufbau

- Faltungsschicht: Für Neuronen, welche sich lokal in kleinen Regionen befinden, werden Skalarprodukte mit lernbaren Gewichten (Kernen) berechnet $\rightarrow$ Merkmalskarten.
- Aktivierungsschicht: Eine Aktivierungsfunktion, z.B. ReLU, wird elementweise auf die Neuronenaktivierungen der Karten angewendet.
- Pooling-Schicht: Durch sogenanntes downsampling wird die räumliche Dimensionen der Merkmalskarten verringert.
- FFN: Ein Vorwärtsgerichtetes Netz wird benutzt, um die Klassifikation zu berechnen.

Convolutional Neural Networks (CNN)

Aufbau

- Faltungsschicht: Für Neuronen, welche sich lokal in kleinen Regionen befinden, werden Skalarprodukte mit lernbaren Gewichten (Kernen) berechnet $\rightarrow$ Merkmalskarten.
- Aktivierungsschicht: Eine Aktivierungsfunktion, z.B. ReLU, wird elementweise auf die Neuronenaktivierungen der Karten angewendet.
- Pooling-Schicht: Durch sogenanntes downsampling wird die räumliche Dimensionen der Merkmalskarten verringert.
- FFN: Ein Vorwärtsgerichtetes Netz wird benutzt, um die Klassifikation zu berechnen.

Convolutional Neural Networks (CNN)

Aufbau

- Faltungsschicht: Für Neuronen, welche sich lokal in kleinen Regionen befinden, werden Skalarprodukte mit lernbaren Gewichten (Kernen) berechnet $\rightarrow$ Merkmalskarten.
- Aktivierungsschicht: Eine Aktivierungsfunktion, z.B. ReLU, wird elementweise auf die Neuronenaktivierungen der Karten angewendet.
- Pooling-Schicht: Durch sogenanntes downsampling wird die räumliche Dimensionen der Merkmalskarten verringert.
- FFN: Ein Vorwärtsgerichtetes Netz wird benutzt, um die Klassifikation zu berechnen.

Convolutional Neural Networks (CNN)

Faltung

Definition (Matrixfaltung)

Seien Matrizen $X \in \mathbb{R}^{h \times b}$ und $K \in \mathbb{R}^{k \times k}$ gegeben(k ungerade). Mit

$$Y_{i,j} = \sum_{u=-l}^{l} \sum_{v=-l}^{l} X_{i+u,j+v} K_{u,v}$$

und $X_{i,j} = 0$ für $i \notin [h]$ und $j \notin [b]$ wird die Matrixfaltung $Y = X * K \in \mathbb{R}^{h \times b}$ definiert. Dabei wird K speziell indiziert. Es ist $l = \lfloor k/2 \rfloor$.

Convolutional Neural Networks (CNN)

Kerne/Filter

Ist beispielsweise $k = 3$, so ist $K \in \mathbb{R}^{3\times 3}$ durch

$$K = \begin{pmatrix} K_{-1,-1} & K_{-1,0} & K_{-1,1} \\ K_{0,-1} & K_{0,0} & K_{0,1} \\ K_{1,-1} & K_{1,0} & K_{1,1} \end{pmatrix}$$

und K^{rot180} durch

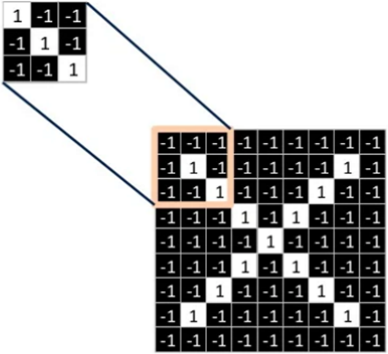
$$K^{rot180} = \begin{pmatrix} K_{1,1} & K_{1,0} & K_{1,-1} \\ K_{0,1} & K_{0,0} & K_{0,-1} \\ K_{-1,1} & K_{-1,0} & K_{-1,-1} \end{pmatrix}$$

gegeben.

Convolutional Neural Networks (CNN)

Veranschaulichung der Merkmalsextraktion

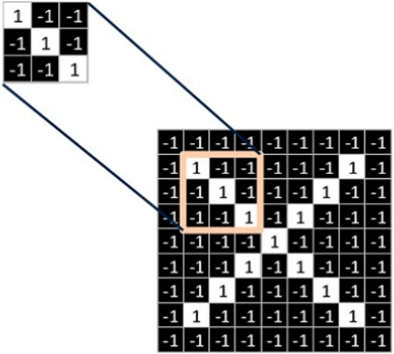
Strides $s_x = s_y = 1$, ohne zero padding, $k = 3$.



Convolutional Neural Networks (CNN)

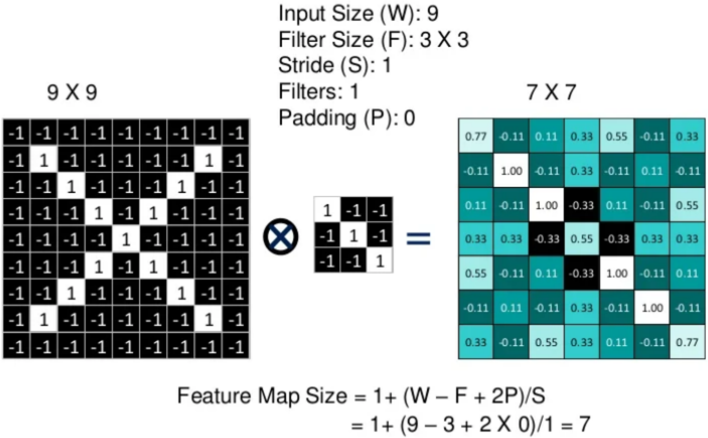
Veranschaulichung der Merkmalsextraktion

Strides $s_x = s_y = 1$, ohne zero padding, $k = 3$.



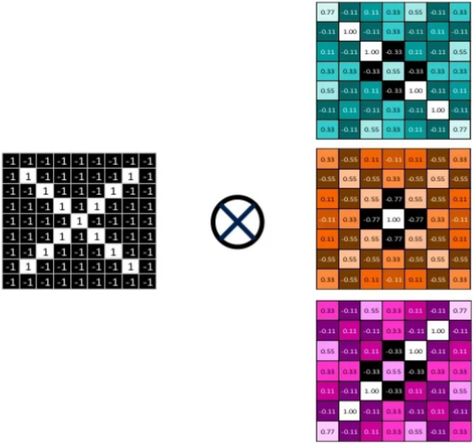
Convolutional Neural Networks (CNN)

Ergebnis einer Faltung



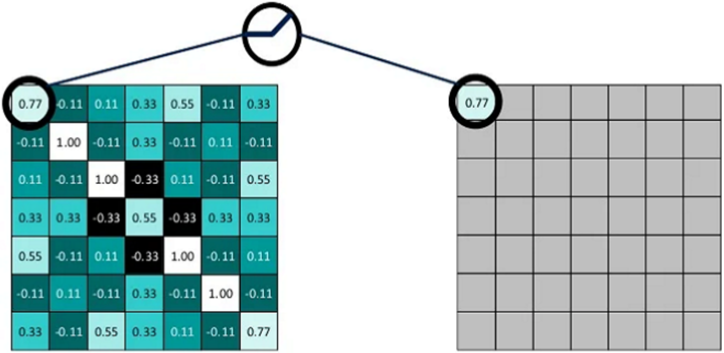
Convolutional Neural Networks (CNN)

Mehrere Filter



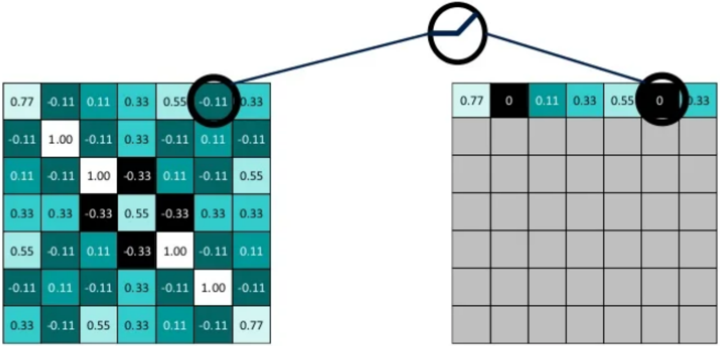
Convolutional Neural Networks (CNN)

Aktivierung



Convolutional Neural Networks (CNN)

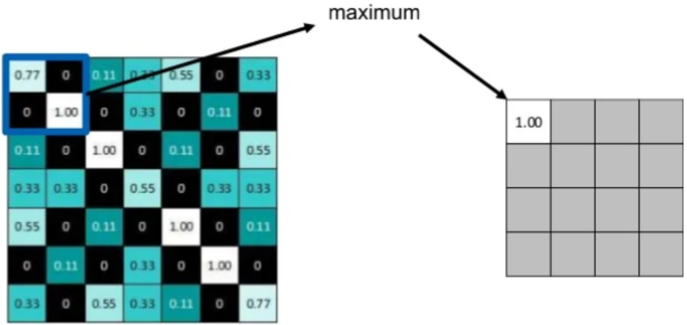
Aktivierung



Convolutional Neural Networks (CNN)

Pooling

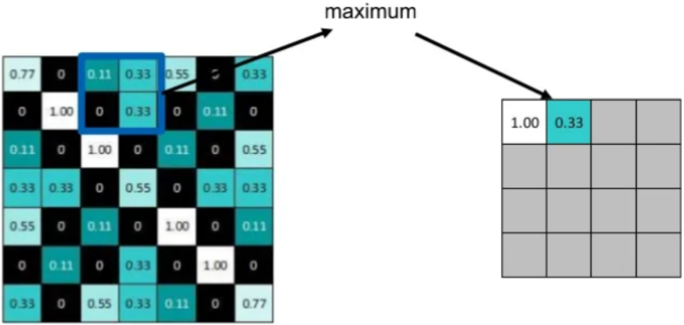
Pooling mit Schrittweiten $p_x = p_y = 2$, Stride $s_x = s_y = 2$



Convolutional Neural Networks (CNN)

Pooling

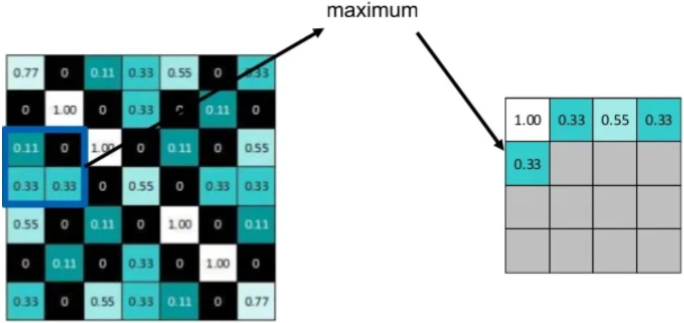
Pooling mit Schrittweiten $p_x = p_y = 2$, Stride $s_x = s_y = 2$



Convolutional Neural Networks (CNN)

Pooling

Pooling mit Schrittweiten $p_x = p_y = 2$, Stride $s_x = s_y = 2$



Convolutional Neural Networks (CNN)

Kombination von Schichten

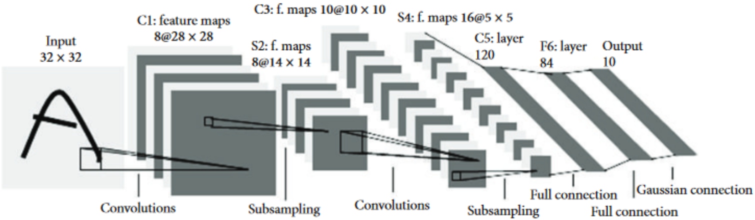


Abbildung: Abgebildet ist die LeNet-5-Architektur[**?**] zur Erkennung von Ziffern aus dem MNIST-Datensatz.

Convolutional Neural Networks (CNN)

Mathematische Beschreibung

Arraydarstellung: Grauwertbild $X \in \mathbb{R}^{h\times b\times 1}$, Kern $K \in \mathbb{R}^{z_{out}\times z_{in}\times k\times k}$

Definition (Gefaltete Netzeingabe)

Die Funktion

$$\Psi^{K,b}_{conv} : \mathbb{R}^{\cdot\times\cdot\times z_{in}} \to \mathbb{R}^{\cdot\times\cdot\times z_{out}}$$

mit

$$\Psi^{K,b}_{conv}(X)_{:,:,q} := \sum_{p=1}^{z_{in}} \alpha_{qp}\left(K_{q,p,:,:} * X_{:,:,p}\right) + b_q, \ \forall q \in [z_{out}]$$

wird gefaltete Netzeingabe bezeichnet.

Convolutional Neural Networks (CNN)

Mathematische Beschreibung

Arraydarstellung: Grauwertbild $X \in \mathbb{R}^{h \times b \times 1}$, Kern $K \in \mathbb{R}^{z_{out} \times z_{in} \times k \times k}$

Definition (Gefaltete Netzeingabe)

Die Funktion

$$\Psi_{conv}^{K,b} : \mathbb{R}^{\cdot \times \cdot \times z_{in}} \to \mathbb{R}^{\cdot \times \cdot \times z_{out}}$$

mit

$$\Psi_{conv}^{K,b}(X)_{:,:,q} := \sum_{p=1}^{z_{in}} \alpha_{qp} \left(K_{q,p,:,:} * X_{:,:,p}\right) + b_q, \ \forall q \in [z_{out}]$$

wird gefaltete Netzeingabe bezeichnet.

Convolutional Neural Networks (CNN)

Mathematische Beschreibung

Definition (Faltungsschicht, vgl. Grüning[?])

Ist $\Psi^{K,b}_{conv}$ eine gefaltete Übertragungsfunktion und ψ eine Aktivierungsfunktion, so wird das Paar $(\Psi^{K,b}_{conv}, \psi)$ als Faltungsschicht $\mathcal{S}_{conv}$ bezeichnet. Für eine sogenannte Eingabekarte $X \in \mathbb{R}^{\cdot \times \cdot \times z_{in}}$ ist die Ausgabe $Y \in \mathbb{R}^{\cdot \times \cdot \times z_{out}}$ der Schicht $\mathcal{S}_{conv}$ durch

$$Y = \psi \circ \Psi^{K,b}_{conv}(X) = \psi\left(\Psi^{K,b}_{conv}(X)\right)$$

gegeben. Die Matrizen $Y_{:,:,p}$ werden für $1 \leq p \leq z_{out}$ Merkmalskarten genannt. Weiter bezeichne $\Psi^{K,b,\psi}_{conv}$ die Faltungsschicht $\mathcal{S}_{conv}$ mit $\Psi^{K,b,\psi}_{conv}(X) := \psi\left(\Psi^{K,b}_{conv}(X)\right)$.

Convolutional Neural Networks (CNN)

Mathematische Beschreibung

Analog werden Pooling-Schichten mit symmetrischen Pooling-Funktionen definiert. Die Kombination mit einem FFN gelingt durch die Flatten-Operation:

Definition (Flatten-Schicht)

Sei $X \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Dann wird die Funktion $T_f : \mathbb{R}^{n_1 \times n_2 \times n_3} \to \mathbb{R}^{n_1 \cdot n_2 \cdot n_3}$ mit

$$T_f(X)_{(i-1)\cdot(n_2 \cdot n_3)+(j-1)\cdot n_3+k} := X_{i,j,k}, \quad \forall i \in [n_1],\ j \in [n_2],\ k \in [n_3]$$

Flatten-Funktion genannt. Die mehrdimensionale Eingabe wird also in einen Vektor umgewandelt.

Convolutional Neural Networks (CNN)

Kombination von Schichten

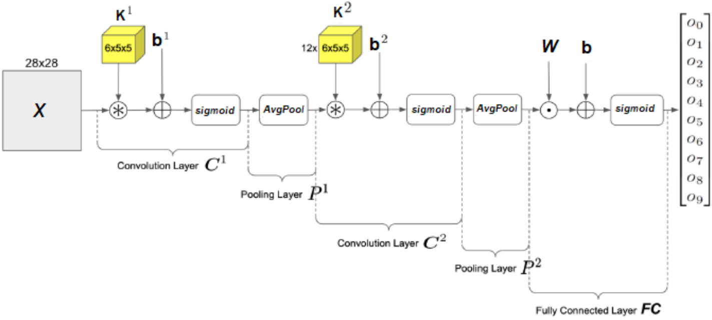


Abbildung: Zu sehen ist eine CNN-Architektur zur Erkennung von Ziffern.

Convolutional Neural Networks (CNN)

Backpropagation

Die Aktivierung einer Merkmalskarte j an der Stelle (x, y) lässt sich mit der Matrixfaltung komponentenweise als

$$y_j^{(\ell)}(x,y) = \psi\left(\sum_{i=1}^{z_{in}} \sum_{(u,v)\in F} y_i^{(\ell-1)}(x+u, y+u)\, w_{j,i}^{(\ell)}(u,v) + b_j^{(\ell)}\right)$$

schreiben, wobei $j \in [z_{out}]$. Beachte $K_{j,i,u,v}^{(\ell)} = w_{j,i}^{(\ell)}(u,v)$, $F = \{(u,v):\ -l \leq u,v \leq l\}$ und $y^{(\ell)} \in \mathbb{R}^{z_{out}\times\cdot\times\cdot}$.

Vergleich zu FFN:

$$y_j^{(\ell)}(1,1) = \psi\left(\sum_{i=1}^{s_{\ell-1}} y_i^{(\ell-1)}(1,1)\, w_{j,i}^{(\ell)} + b_j^{(\ell)}\right)$$

Convolutional Neural Networks (CNN)

Backpropagation

Die Aktivierung einer Merkmalskarte j an der Stelle (x, y) lässt sich mit der Matrixfaltung komponentenweise als

$$y_j^{(\ell)}(x,y) = \psi\left(\sum_{i=1}^{z_{in}} \sum_{(u,v)\in F} y_i^{(\ell-1)}(x+u, y+u)\, w_{j,i}^{(\ell)}(u,v) + b_j^{(\ell)}\right)$$

schreiben, wobei $j \in [z_{out}]$. Beachte $K_{j,i,u,v}^{(\ell)} = w_{j,i}^{(\ell)}(u,v)$, $F = \{(u,v) : \; -l \leq u, v \leq l\}$ und $y^{(\ell)} \in \mathbb{R}^{z_{out} \times \cdot \times \cdot}$.

Vergleich zu FFN:

$$y_j^{(\ell)}(1,1) = \psi\left(\sum_{i=1}^{s_{\ell-1}} y_i^{(\ell-1)}(1,1)\, w_{j,i}^{(\ell)} + b_j^{(\ell)}\right)$$

Convolutional Neural Networks (CNN)

Backpropagation

Der Gradient von $w_{j,i}^{(\ell)}$ ergibt sich als Summe über alle beteiligten Pixel (x, y) der Merkmalskarte j, also

$$\Delta w_{j,i}^{(\ell)}(u, v) = \sum_{(x,y)} \left(\delta_j^{(\ell)}(x, y) y_i^{(\ell-1)}(x + u, y + v) \right)$$

und

$$\Delta b_j^{(\ell)} = \sum_{(x,y)} \delta_j^{(\ell)}(x, y).$$

Convolutional Neural Networks (CNN)

Veranschaulichung

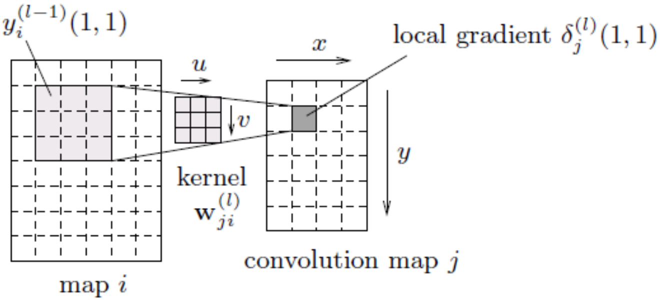


Abbildung: Es ist die Rückwärtsrechnung bei Faltungsschichten dargestellt. Die Abbildung wurde aus [?] entnommen.

Convolutional Neural Networks (CNN)

Vergleich Backpropagation

CNN:

$$\Delta w_{j,i}^{(\ell)}(u,v) = \sum_{(x,y)} \left(\delta_j^{(\ell)}(x,y) y_i^{(\ell-1)}(x+u, y+v) \right)$$

$$\Rightarrow \Delta W_{j,i} = y_i^{(\ell-1)} * \delta_j^{(\ell)}, \quad \Delta b_j^{(\ell)} = \sum_{(x,y)} \delta_j^{(\ell)}(x,y).$$

FFN:

$$\Delta w_{j,i}^{(\ell)} = z_i^{(\ell-1)} \delta_j^{(\ell)}$$

$$\Rightarrow \Delta W^{(\ell)} = \delta^{(\ell)} \cdot z^{(\ell-1)^T},$$

$$\Delta b_j^{(\ell)} = \delta_j^{(\ell)}$$

$$\Rightarrow \Delta b^{(\ell)} = \delta^{(\ell)}$$

Convolutional Neural Networks (CNN)

Vergleich Backpropagation

CNN:

$$\Delta w_{j,i}^{(\ell)}(u,v) = \sum_{(x,y)} \left(\delta_j^{(\ell)}(x,y) y_i^{(\ell-1)}(x+u, y+v) \right)$$

$$\Rightarrow \Delta W_{j,i} = y_i^{(\ell-1)} * \delta_j^{(\ell)}, \quad \Delta b_j^{(\ell)} = \sum_{(x,y)} \delta_j^{(\ell)}(x,y).$$

FFN:

$$\Delta w_{j,i}^{(\ell)} = z_i^{(\ell-1)} \delta_j^{(\ell)}$$

$$\Rightarrow \Delta W^{(\ell)} = \delta^{(\ell)} \cdot {z^{(\ell-1)}}^T,$$

$$\Delta b_j^{(\ell)} = \delta_j^{(\ell)}$$

$$\Rightarrow \Delta b^{(\ell)} = \delta^{(\ell)}$$

Convolutional Neural Networks (CNN)

lokale Fehler

Wie werden die lokalen Fehler $\delta_j^{(\ell)}(x,y) = \frac{\partial E_x}{\partial V_j^{(\ell)}(x,y)}$ mit

$$V_j^{(\ell)}(x,y) = \sum_k w_{j,k} y_k^{(\ell-1)}(x,y)$$

berechnet? Analog wie bei FFN, aber in Abhängigkeit von der Schicht $\ell + 1$.

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Neuronenschicht mit K Neuronen:

$$\delta_j^{(\ell)}(x,y) = \sum_{k=1}^{K} \left(\sum_{(x,y)} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)}(x,y) \right) \psi'(V_j^{(\ell)}),$$

beachte $x, y > 1$ möglich.

FFN:

$$\delta_j^{(\ell)} = \left(\sum_{k=1}^{s_{l+1}} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)} \right) \phi'(T_j^{(\ell)})$$

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Neuronenschicht mit K Neuronen:

$$\delta_j^{(\ell)}(x,y) = \sum_{k=1}^{K} \left(\sum_{(x,y)} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)}(x,y) \right) \psi'(V_j^{(\ell)}),$$

beachte $x, y > 1$ möglich.

FFN:

$$\delta_j^{(\ell)} = \left(\sum_{k=1}^{s_{l+1}} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)} \right) \phi'(T_j^{(\ell)})$$

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell+1$ Faltungsschicht mit z_{out} Karten

$$\delta_j^{(\ell)}(x,y) = \sum_{k=1}^{z_{out}} \left(\sum_{(u,v)\in F} \delta_k^{(\ell+1)}(x-u,y-v) w_{k,j}^{(\ell+1)}(u,v) \right) \psi'(V_j^{(\ell)})$$

$$\Rightarrow \left(\delta_j^{(\ell)} = \sum_{k=1}^{z_{out}} \delta_k^{(\ell+1)} * w_{k,j,rot180}^{(\ell+1)} \right) \psi'(V_j^{(\ell)})$$

beachte wieder $w_{k,j}^{(\ell+1)} = K_{k,j,:,:}^{(\ell+1)}$ und $K_{rot180}(-u,-v) = K(u,v)$.

FFN:

$$\delta_j^{(\ell)} = \left(\sum_{k=1}^{s_{l+1}} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)} \right) \phi'(T_j^{(\ell)})$$

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Faltungsschicht mit z_{out} Karten

$$\delta_j^{(\ell)}(x, y) = \sum_{k=1}^{z_{out}} \left(\sum_{(u,v)\in F} \delta_k^{(\ell+1)}(x-u, y-v) w_{k,j}^{(\ell+1)}(u, v) \right) \psi'(V_j^{(\ell)})$$

$$\Rightarrow \left(\delta_j^{(\ell)} = \sum_{k=1}^{z_{out}} \delta_k^{(\ell+1)} * w_{k,j,rot180}^{(\ell+1)} \right) \psi'(V_j^{(\ell)})$$

beachte wieder $w_{k,j}^{(\ell+1)} = K_{k,j,:,:}^{(\ell+1)}$ und $K_{rot180}(-u, -v) = K(u, v)$.

FFN:

$$\delta_j^{(\ell)} = \left(\sum_{k=1}^{s_{l+1}} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)} \right) \phi'(T_j^{(\ell)})$$

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Faltungsschicht mit z_{out} Karten

$$\delta_j^{(\ell)}(x,y) = \sum_{k=1}^{z_{out}} \left(\sum_{(u,v)\in F} \delta_k^{(\ell+1)}(x-u, y-v) w_{k,j}^{(\ell+1)}(u,v) \right) \psi'(V_j^{(\ell)})$$

$$\Rightarrow \left(\delta_j^{(\ell)} = \sum_{k=1}^{z_{out}} \delta_k^{(\ell+1)} * w_{k,j,rot180}^{(\ell+1)} \right) \psi'(V_j^{(\ell)})$$

beachte wieder $w_{k,j}^{(\ell+1)} = K_{k,j,:,:}^{(\ell+1)}$ und $K_{rot180}(-u,-v) = K(u,v)$.

FFN:

$$\delta_j^{(\ell)} = \left(\sum_{k=1}^{s_{l+1}} \delta_k^{(\ell+1)} w_{k,j}^{(\ell+1)} \right) \phi'(T_j^{(\ell)})$$

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell+1$ Pooling-Schicht mit Schrittweiten p_x, p_y und Funktion T:

$$\delta_j^{(l)}(x,y) = \delta_j^{(l+1)}(\lceil x/p_x \rceil, \lceil y/p_y \rceil) \nabla T. \tag{1}$$

Dabei wird der Gradient ∇T jeweils an der zu (x,y) korrespondierenden Stelle ausgewertet.

Ist $l+1$ eine Flatten-Schicht mit einer Flatten-Funktion T_f, so gilt

$$\delta_j^{(l)} = T_f^{-1}(\delta_j^{(l+1)}). \tag{2}$$

Die Berechnung in Gleichung (1) und Gleichung (2) wird Upsampling genannt.

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Pooling-Schicht mit Schrittweiten p_x, p_y und Funktion T:

$$\delta_j^{(l)}(x, y) = \delta_j^{(l+1)}(\lceil x/p_x \rceil, \lceil y/p_y \rceil) \nabla T. \qquad (1)$$

Dabei wird der Gradient ∇T jeweils an der zu (x, y) korrespondierenden Stelle ausgewertet.

Ist $l + 1$ eine Flatten-Schicht mit einer Flatten-Funktion T_f, so gilt

$$\delta_j^{(l)} = T_f^{-1}(\delta_j^{(l+1)}). \qquad (2)$$

Die Berechnung in Gleichung (1) und Gleichung (2) wird Upsampling genannt.

Convolutional Neural Networks (CNN)

lokale Fehler

$\ell + 1$ Pooling-Schicht mit Schrittweiten p_x, p_y und Funktion T:

$$\delta_j^{(l)}(x,y) = \delta_j^{(l+1)}(\lceil x/p_x \rceil, \lceil y/p_y \rceil)\nabla T. \tag{1}$$

Dabei wird der Gradient ∇T jeweils an der zu (x,y) korrespondierenden Stelle ausgewertet.

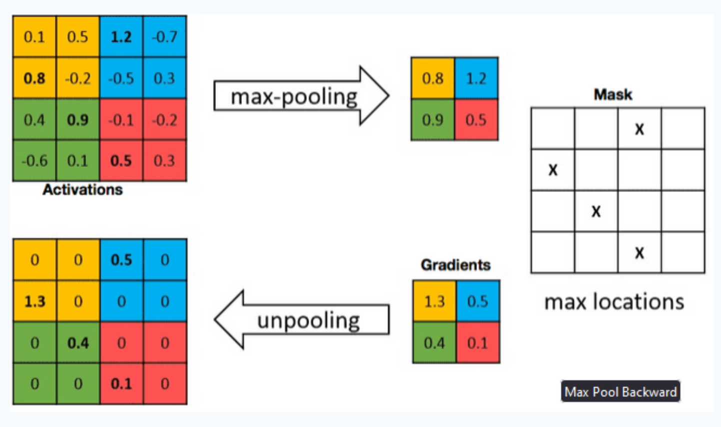
Ist $l+1$ eine Flatten-Schicht mit einer Flatten-Funktion T_f, so gilt

$$\delta_j^{(l)} = T_f^{-1}(\delta_j^{(l+1)}). \tag{2}$$

Die Berechnung in Gleichung (1) und Gleichung (2) wird Upsampling genannt.

Convolutional Neural Networks (CNN)

Pooling: lokaler Fehler, vgl. Rathi(**?**)



Convolutional Neural Networks (CNN)

Abschließende Bemerkungen

- Auch für die Upsampling-Verfahren gibt es Darstellungen die beispielsweise das Kronecker-Produkt nutzen.
- Die Faltungsoperation muss performant implementiert werden.
- Es sind beliebige Kombinationen möglich.
- Bei der Bildklassifikation werden von CNN oft bessere Generalisierungsraten im Vergleich zu FFN erzielt.

Convolutional Neural Networks (CNN)

Abschließende Bemerkungen

- Auch für die Upsampling-Verfahren gibt es Darstellungen die beispielsweise das Kronecker-Produkt nutzen.
- Die Faltungsoperation muss performant implementiert werden.
- Es sind beliebige Kombinationen möglich.
- Bei der Bildklassifikation werden von CNN oft bessere Generalisierungsraten im Vergleich zu FFN erzielt.

Convolutional Neural Networks (CNN)

Abschließende Bemerkungen

- Auch für die Upsampling-Verfahren gibt es Darstellungen die beispielsweise das Kronecker-Produkt nutzen.
- Die Faltungsoperation muss performant implementiert werden.
- Es sind beliebige Kombinationen möglich.
- Bei der Bildklassifikation werden von CNN oft bessere Generalisierungsraten im Vergleich zu FFN erzielt.

Convolutional Neural Networks (CNN)

Abschließende Bemerkungen

- Auch für die Upsampling-Verfahren gibt es Darstellungen die beispielsweise das Kronecker-Produkt nutzen.
- Die Faltungsoperation muss performant implementiert werden.
- Es sind beliebige Kombinationen möglich.
- Bei der Bildklassifikation werden von CNN oft bessere Generalisierungsraten im Vergleich zu FFN erzielt.

Convolutional Neural Networks (CNN)

Abschließende Bemerkungen

- Auch für die Upsampling-Verfahren gibt es Darstellungen die beispielsweise das Kronecker-Produkt nutzen.
- Die Faltungsoperation muss performant implementiert werden.
- Es sind beliebige Kombinationen möglich.
- Bei der Bildklassifikation werden von CNN oft bessere Generalisierungsraten im Vergleich zu FFN erzielt.

Datenbankgestützte Umsetzung

PArADISE-Projekt

- Design von Assistenzsystemen, Unterstützung der Entwickler solcher Systeme
- Entwicklungsphase: Datenwisenschaftler sammeln Sensordaten, um Benutzeraktivitäten zu erkennen und vorherzusagen.
- Nutzung von ML-Algorithmen, um Aktivitätsmodelle zu lernen $\rightarrow$ PaMeLA

Datenbankgestützte Umsetzung

PArADISE-Projekt

- Design von Assistenzsystemen, Unterstützung der Entwickler solcher Systeme
- Entwicklungsphase: Datenwisenschaftler sammeln Sensordaten, um Benutzeraktivitäten zu erkennen und vorherzusagen.
- Nutzung von ML-Algorithmen, um Aktivitätsmodelle zu lernen $\rightarrow$ PaMeLA

Datenbankgestützte Umsetzung

PArADISE-Projekt

- Design von Assistenzsystemen, Unterstützung der Entwickler solcher Systeme
- Entwicklungsphase: Datenwisenschaftler sammeln Sensordaten, um Benutzeraktivitäten zu erkennen und vorherzusagen.
- Nutzung von ML-Algorithmen, um Aktivitätsmodelle zu lernen $\rightarrow$ PaMeLA

Datenbankgestützte Umsetzung

PArADISE-Projekt

- Design von Assistenzsystemen, Unterstützung der Entwickler solcher Systeme
- Entwicklungsphase: Datenwisenschaftler sammeln Sensordaten, um Benutzeraktivitäten zu erkennen und vorherzusagen.
- Nutzung von ML-Algorithmen, um Aktivitätsmodelle zu lernen $\rightarrow$ PaMeLA

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [?]
- andere Verfahren? $\rightarrow$ FNN und CNN

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [?]
- andere Verfahren? $\rightarrow$ FNN und CNN

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [?]
- andere Verfahren? $\rightarrow$ FNN und CNN

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [?]
- andere Verfahren? → FNN und CNN

$$\frac{\partial u}{\partial t} = \Delta u + f(u)$$

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [?]
- andere Verfahren? $\rightarrow$ FNN und CNN

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [**?**]
- andere Verfahren? → FNN und CNN

Datenbankgestützte Umsetzung

PaMeLA: Parallelization of Machine Learning Algorithms

- Anfallen großer Datenmengen, Problem: mangelnde Rechenleistung
- transparente Datenbankunterstützung für Big Data Analytics
- Ziel: Transformation von ML-Algorithmen in parallele SQL-Systeme
- besonders interessant: Parallelisierung von Operationen der linearen Algebra
- Beispiel: Hidden-Markow-Modelle, siehe Marten et. al. [**?**]
- andere Verfahren? $\rightarrow$ FNN und CNN

Datenbankgestützte Umsetzung

Datenbanken und SQL

Beispiel: zwei Relationen **Angestellte** und **Projekt** als Tabellen

Angestellte				
ID	Name	Spezialisierung	Projektnummer	Gehalt
1	Martin	Elektrotechnik	3	2300
2	Lennardt	Informatik	1	1500
3	Johann	Informatik	3	1800
4	Jana	Maschinenbau	3	2100
5	Antonia	Buchhaltung	2	2000

Tabelle: Abgebildet ist die Beispielrelation Angestellte mit den Attributen ID, Name, Spezialisierung, Projektnummer und Gehalt.

Datenbankgestützte Umsetzung

Datenbanken und SQL

Projekt			
Projektnummer	Projektname	Budget	Ort
1	Datenbank 2.0	50000	Rostock
2	Verwaltung	25000	Rostock
3	Forschungsabteilung	40000	Schwerin

Tabelle: Abgebildet ist die Beispielrelation Projekt mit den Attributen Projektnummer, Projektname, Budget und Ort.

Datenbankgestützte Umsetzung

Datenbanken und SQL

Eine Anfrage:

```
SELECT A.Name, P.Projektname
FROM Angestellte A JOIN Projekt P
ON A.Projektnummer = P.Projektnummer
WHERE P.Ort = 'Rostock'
```

Ergebnis	
A.Name	P.Projektname
Lennardt	Datenbanken 2.0
Antonia	Verwaltung

Tabelle: Zu sehen ist die Ergebnisrelation der zuvor beschriebenen Anfrage.

Datenbankgestützte Umsetzung

Datenbanken und SQL

Eine Anfrage:

SELECT A.Name, P.Projektname
FROM Angestellte A JOIN Projekt P
ON A.Projektnummer = P.Projektnummer
WHERE P.Ort = 'Rostock'

Ergebnis	
A.Name	P.Projektname
Lennardt	Datenbanken 2.0
Antonia	Verwaltung

Tabelle: Zu sehen ist die Ergebnisrelation der zuvor beschriebenen Anfrage.

Datenbankgestützte Umsetzung

Lineare Algebra und SQL

Ist

$$A = \begin{pmatrix} 1 & 2 \\ -5 & 2 \\ 0 & 7 \end{pmatrix} \in \mathbb{R}^{3\times 2}$$

gegeben, so ergibt sich Coordinate-Schema wie in Tabelle 4.

A		
i	j	v
1	1	1
1	2	2
2	1	-5
2	2	2
3	1	0
3	2	7

Datenbankgestützte Umsetzung

Lineare Algebra und SQL

Ist

$$A = \begin{pmatrix} 1 & 2 \\ -5 & 2 \\ 0 & 7 \end{pmatrix} \in \mathbb{R}^{3\times 2}$$

gegeben, so ergibt sich Coordinate-Schema wie in Tabelle 4.

A		
i	j	v
1	1	1
1	2	2
2	1	-5
2	2	2
3	1	0
3	2	7

Datenbankgestützte Umsetzung

Lineare Algebra und SQL

Matrixvektormultiplikation $Ax \in \mathbb{R}^m$ einer Matrix $A \in \mathbb{R}^{m\times n}$ und eines Vektors $x \in \mathbb{R}^n$ als SQL-Anfrage

$$\begin{aligned}&\text{SELECT } A.i \text{ AS } i,\ \text{SUM}(A.v * x.v) \text{ AS } v\\ &\text{FROM } A \text{ JOIN } x \text{ ON } A.j = x.i\\ &\text{GROUP BY } A.i.\end{aligned}$$

Dabei tauchen Aggregatfunktionen und Gruppierungen auf. Andere Basisoperationen wie Skalarprodukte, Matrixmatrixmultiplikation etc. auch möglich.

Datenbankgestützte Umsetzung

Lineare Algebra und SQL

Matrixvektormultiplikation $Ax \in \mathbb{R}^m$ einer Matrix $A \in \mathbb{R}^{m \times n}$ und eines Vektors $x \in \mathbb{R}^n$ als SQL-Anfrage

$$\begin{aligned}
&\text{SELECT } A.i \text{ AS } i,\ \text{SUM}(A.v * x.v) \text{ AS } v \\
&\text{FROM } A \text{ JOIN } x \text{ ON } A.j = x.i \\
&\text{GROUP BY } A.i.
\end{aligned}$$

Dabei tauchen Aggregatfunktionen und Gruppierungen auf. Andere Basisoperationen wie Skalarprodukte, Matrixmatrixmultiplikation etc. auch möglich.

Datenbankgestützte Umsetzung

Aggregate und Gruppierungen

Sei

$$A = \begin{pmatrix} 3 & 1 \\ 4 & 1 \end{pmatrix}.$$

Die Spaltensumme von A kann mit der Gruppierungsoperation

$$\gamma_{j,\,\mathrm{SUM}(v)\to v}(\mathrm{A}) \Rightarrow$$

j	v
1	7
2	2

berechnet werden.

$$\begin{aligned} &\mathrm{SELECT}\ A.j\ \mathrm{AS}\ j,\ \mathrm{SUM}(A.v)\ \mathrm{AS}\ v \\ &\mathrm{FROM}\ A \\ &\mathrm{GROUP\ BY}\ A.j. \end{aligned}$$

Datenbankgestützte Umsetzung

Dünn besetzte Matrizen

Darstellung im Compressed-Sparse-Column-Schema ist durch

$$A_{i,j} = \mathrm{val}[k] \Leftrightarrow (\mathrm{rowInd}[k] = i) \wedge (\mathrm{colPtr}[j] \leq k < \mathrm{colPtr}[j+1])$$

gegeben. Sei

$$A = \begin{pmatrix} a_{11} & 0 & 0 & a_{14} \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}.$$

- **val**$=[a_{11}, a_{21}, a_{22}, a_{42}, a_{33}, a_{14}, a_{44}]$,
- **rowInd**$=[1, 2, 2, 4, 3, 1, 4]$,
- **colPtr**$=[1, 3, 5, 6, 8]$.

Datenbankgestützte Umsetzung

Dünn besetzte Matrizen

Darstellung im Compressed-Sparse-Column-Schema ist durch

$$A_{i,j} = \mathrm{val}[k] \Leftrightarrow (\mathrm{rowInd}[k] = i) \wedge (\mathrm{colPtr}[j] \leq k < \mathrm{colPtr}[j+1])$$

gegeben. Sei

$$A = \begin{pmatrix} a_{11} & 0 & 0 & a_{14} \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}.$$

- **val**=$[a_{11}, a_{21}, a_{22}, a_{42}, a_{33}, a_{14}, a_{44}]$,
- **rowInd**=$[1, 2, 2, 4, 3, 1, 4]$,
- **colPtr**=$[1, 3, 5, 6, 8]$.

Datenbankgestützte Umsetzung

Dünn besetzte Matrizen

Definition (Spaltenkompression)

Eine dünn besetzte Matrix $A \in \mathbb{R}^{m \times n}$ wird als Relation

$$\begin{aligned}\mathrm{A}(&i \text{ int ARRAY},\\ &\underline{j} \text{ int},\\ &v \text{ double ARRAY})\end{aligned}$$

gespeichert. Diese Repräsentation wird Spaltenkompression genannt und ist insbesondere für die Matrixvektormultiplikation nützlich.

Beachte: Die Array-Darstellung ist nicht in allen Systemen unterstützt.

Datenbankgestützte Umsetzung

Dünn besetzte Matrizen

Definition (Spaltenkompression)

Eine dünn besetzte Matrix $A \in \mathbb{R}^{m \times n}$ wird als Relation

$$\begin{aligned} \mathrm{A}(&i \text{ int ARRAY}, \\ &\underline{j} \text{ int}, \\ &v \text{ double ARRAY}) \end{aligned}$$

gespeichert. Diese Repräsentation wird Spaltenkompression genannt und ist insbesondere für die Matrixvektormultiplikation nützlich.

Beachte: Die Array-Darstellung ist nicht in allen Systemen unterstützt.

Auflistungen und Aufzählungen

Diese Folie hat zur Abwechslung mal keinen Untertitel, dafür ist sie aber zweispaltig.

- erster Auflistungspunkt
  - nächste Ebene
  - nächste Ebene
    - tiefste Ebene
    - tiefste Ebene
  - nächste Ebene
- zweiter Auflistungspunkt

1. mit Aufzählungen
   1.1 geht das natürlich
      1.1.1 ebenso
      1.1.2 wie mit
   1.2 Auflistungen

Theorem / Beweis / andere Boxen I

Theorem

Diese Box ist schön.

Beweis.

Die CD-Vorlage ist insgesamt schick, ergo muss jedes Teil hiervon dekorativ sein, folglich also auch die obige Theorem-Box. □

Beispiel

Diese Box ist auch ein nettes Beispiel für schicke Boxen.

Theorem / Beweis / andere Boxen II

Blocktitel

Ein Block mit dem Titel Blocktitel

Alertblocktitel

Ein Alertblock.

Beispielblocktitel

Ein Beispielblock.

Einbindung des Styles

Alle für das Style notwendigen Dateien liegen im Unterordner `./unirostock`. Das Stylefile selbst kann mittels

```
\usepackage[mnf,footuni]{./unirostock/beamerthemeRostock}
```

eingebunden werden. Der erste Parameter ist das Fakultätskürzel, das das Farbschema vorgibt. Mögliche Werte sind `uni`, `inf`, `msf`, `ief`, `mnf`, `mef`, `juf`, `wsf`, `auf`, `thf`, `phf`.

Der zweite Parameter steuert das Verhalten der Fußzeile:

`footuni`	Fußzeile wie im CD-Handbuch (Standard)
`foottitle`	Autor und Kurztitel in der Fußzeile
`footheadings`	Abschnitt und Unterabschnitt in der Fußzeile
`footuniheadings`	Author und Uni sowie Abschnitt und Unterabschnitt

LaTeXund pdfLaTeX

Da zur Erstellung der Foliendekoration das (sowieso mit der `beamer`-class eingebundene) Paket `pgfgraphics` zur Anwendung kam, können die mit diesem Style erstellten Vorträge sowohl mittels

```
pdflatex beamer_sample.tex
```

als auch mit

```
latex beamer_sample.tex
dvips beamer_sample.dvi
ps2pdf beamer_sample.ps
```

übersetzt werden. Es gelten natürlich die üblichen Regeln bezüglich der erlaubten Grafikformate bei Verwendung von pdfLaTeXbzw.LaTeX.

Farbschema

Tolle Automatik

Das Farbschema wird komplett durch den Fakultätsparameter bei der Einbindung des Styles bestimmt. Es empfiehlt sich, bei Grafiken, Tabellen ein entsprechend passendes Schema zu wählen.

Das Hintergrundbild auf der Titelseite wird auch automatisch eingebunden. Es ist jedoch nicht vorgeschrieben und kann (und sollte) nach eigenem Bedarf ersetzt werden. Hierzu dient der Befehle `\titleimage{..}`. Weitere Details hierzu finden sich im Quellcode des Beispieldokumentes und auf der übernächsten Folie. Bei manueller Wahl eines Hintergrundbildes empfiehlt sich, auf einen guten Pass zum Farbschema zu achten.

Tabellenfarben

Dei Befehle `\zfA`, `\zfB`, `\zfC`, `\zfD` stellen einen effektiven Weg dar, um die Zeilenfarbe in Tabellen anzupassen. Sie werden einfach vor die Tabellenzeile gestzt, also zum Beispiel

```
\zfA Spalte1 & spalte2 & ..
```

Intern wird mit dem Befehl `\rowcolor{..}` des Paketes `colortbl` gearbeitet. Es empfiehlt sich ein Blick in dessen Dokumentation für fortgeschrittene Anwendungen.

Zeilenfarbe A	gewählt mit dem Befehl	`\zfA`
Zeilenfarbe B	gewählt mit dem Befehl	`\zfB`
Zeilenfarbe C	gewählt mit dem Befehl	`\zfC`
Zeilenfarbe D	gewählt mit dem Befehl	`\zfD`

Befehle für die Titelseite I

- Vortragstitel:

```
\title[Kurztitel (f\"ur die Fu{\ss}zeile)]{Langer Titel}
```

- Untertitel:

```
\subtitle{Untertitel}
```

- Autor(en):

```
\author{Name}
```

- Einrichtung / Institut / Universität:

```
\institute{Universit\"at Rostock, Institut f\"ur Physik}
```

Befehle für die Titelseite II

- Zusätzlich gibt es Platz für eigene Einträge, ein Logo oder ähnliches mittels

  ```
  \titlegraphic{Zusatztext}
  ```

  Dabei kann beliebiger LaTeX-Code übergeben werden, also auch `\includegraphics{..}`, `\centering` etc.

- Ein eigenes Titelbild kann mit

  ```
  \titleimage{dateiname.xyz}
  ```

  eingebunden werden. Die Skalierung und das Abschneiden der oberen rechten Ecke erfolgen automatisch. Wichtig ist, ein sinnvolles Seitenverhältnis der Originalgraphik zu wählen, um

Kopf- und Fußzeile

- Der Institutsname für die Fußzeile wird mit

  ```
  \footinstitute{Fakult\"at, Institut}
  ```

  angepasst. Er wird nur angezeigt, wenn bei Paketeinbindung der Parameter `footuni` angegeben ist.

- Der Logobereich oben rechts kann mittels

  ```
  \renewcommand{\mylogo}{\includegraphics[
                          width=18.5mm]{institutslogo}}
  ```

  angepasst werden. Dies ist der in diesem Dokument benutzte Beispielcode. Erlaubt sind alle Minipage-verträglichen LaTeX-Befehle.

Universität Rostock

Traditio et Innovatio

Allgemeine Bemerkungen

Hinweise zum Design

Diese Beispielpräsentation ist reichlich überladen, um alle Features zu demonstrieren. Es ist empfehlenswert, die eigene Präsentation kritisch zu hinterfragen in Bezug auf die Fülle der Folien, ihre Anzahl und Gestaltung sowie die Einhaltung der allgemeinen Regeln für Präsentationen.

Wirklich letzter Hinweis

Viele Fragen lassen sich beim Blick in den Quellcode dieser Beispielpräsentation klären. Insbesondere die (zugegebenermassen sparsam verteilten) Kommentare könnten hilfreich sein. Ebenso ist ein Blick in `beamerusersguide.pdf` immer zu empfehlen. Und ein zweiter Blick auch ;-)