Neuronale Netze werden, wie auch in dieser Arbeit, häufig zur Bild- und Mustererkennung genutzt. Dabei steht dem Netzwerk eine zweidimensionale Matrix von Pixelinformation in Form von Farb- oder Graustufeninformationen als Eingabe zur Verfügung. Aus dieser Eingabe soll nun eine Ausgabe in Form einer Klassifizierung bestimmt werden. Eine solche Zuordnung kann mit den bereits beschriebenen klassischen Neuronalen Netzen erfolgen. Der Nachteil dieser Methode ist, dass diese Art von Neuronennetzen die innere Struktur der Bilder nicht berücksichtigen können. Beispielsweise stellt eine Verschiebung des Musters innerhalb des Eingangsbildes für solche Netze ein Problem bei der Erkennung dar. Auch eine Verkleinerung oder Vergrößerung führt zu Problemen. Solche Effekte müssen zur zuverlässigen Erkennung bereits in den Trainingsdaten berücksichtigt werden. Dies führt unweigerlich zu größeren Datensätzen, die für das Training eines zuverlässigen Netzes benötigt werden. Dies könnte durch Einbeziehung der inneren Struktur des Bildes verhindert werden. Dazu stellten Yann LeCun, Léon Bottou, Yoshua Bengio und Patrick Haffner in ihrerm Paper *Gradient-based learning applied to document recognition* von 1998 sogenannte *Convolutional Neural Networks* vor.

Die dort vorgestellte Netzstruktur erlaubt die Erkennung von Strukturen unabhängig von ihrer Position im Ausgangsbild. Im folgenden Abschnitt soll das grundlegende Prinzip von Convolutional Neural Networks (CNN) vorgestellt und erläutert werden.

0.1 Faltungsoperation

Convolutional Neural Networks beruhen auf der mathematischen Faltungsoperation (Convolution). Auf abstrakter Ebene wird des zweidimensionale Eingangsbild $X(i,j)$ mit einem zweidimensionalen gewichteten Fenster $W(k,l)$ gefaltet.

$$Y(i,j) = \sum_{p=0}^{n} \sum_{q=0}^{n} X(i-p+a, j-q+a) W(p,q) \tag{0.1}$$

Gleichung 0.1 zeigt die diskrete Faltung für den genannten Fall. Dabei bezeichnet n die Zeilen- bzw. Spaltenzahl der quadratischen Matrix W und a einen von n abhängigen Parameter, der das Fenster korrekt positioniert. Bildlich gesprochen wird hier ein gewichtetes Fenster sowohl in horizontaler als auch in vertikaler Richtung über das Ausgangsbild bewegt. An jeder Position werden die jeweiligen Werte des Ausgangsbildes mit den gewichten der Fenstermatrix multipliziert und zu einem Wert addiert. Dieser Wert fließt in Convolutional Neural Networks in ein Neuron der nächsten Netzwerkschicht.

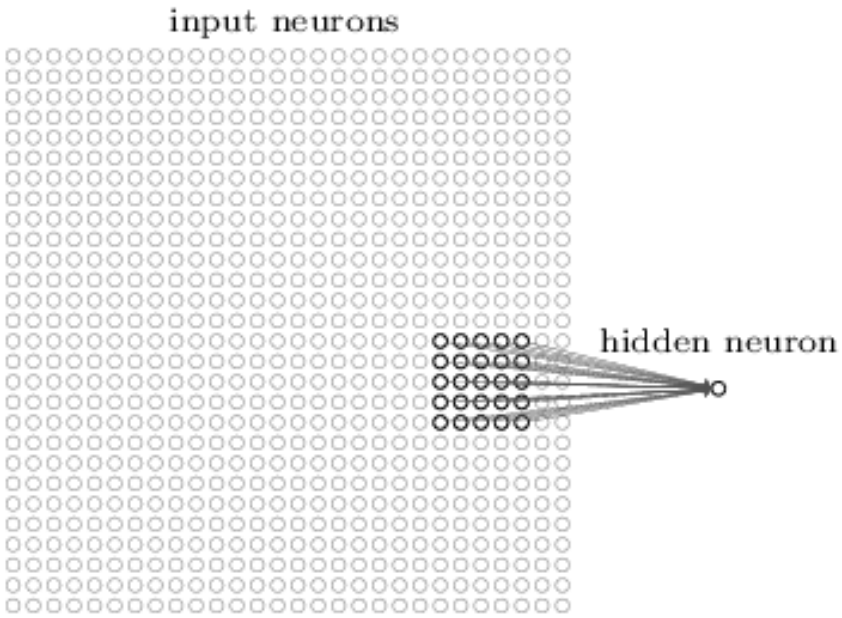

Figure 0.1: Berechnung eines Neurons aus einem Faltungskern (vgl. **NNADL_PIC_CONV_1**)

In Abbildung 0.1 ist der Vorgang für den Faltungskern an einer Position dargestellt. Dabei werden Neuronen aus dem Eingangsbild über den Faltungskern mit den entsprechenden Gewichten auf ein Hidden Neuron abgebildet.

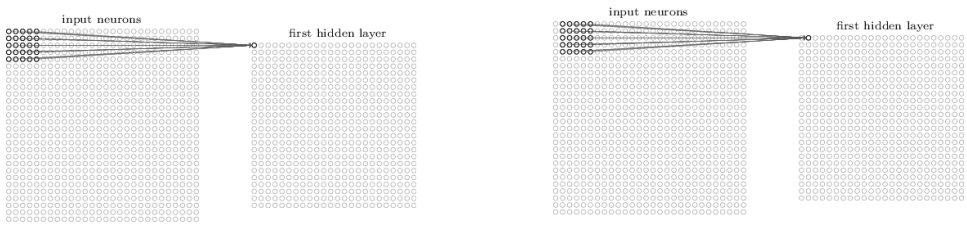

Figure 0.2: Faltungsoperation für zwei Ausgangsneuronen (vgl. **NNADL_PIC_CONV_2**)

Führt man nun die Faltung weiter, erhält man als Ergebnis erneut ein zweidimensionales Ausgangsbild mit geringerer Seitenlänge. Abbildung 0.2 zeigt diesen Schritt. Für ein 28×28-Eingangsbild und einen 5×5-Faltungskern erhält man bei einer Schrittweite von 1 beispielsweise ein 24×24-Ausgangsbild.

Bei dieser Faltungsoperation werden logischerweise für jedes Neuron die gleichen Gewichte verwendet. Im Fall des hier gezeigten Neuronalen Netzwerkes teilen sich die Ausgangsneuronen dieser Operation auch die Biases. Mit der Aktivierungsfunktion σ bestimmt sich der Output eines Neurons damit nach Gleichung **??**.

$$output = \sigma(b + \sum_{l} \sum_{m} w_{l,m} a_{j+l,k+m}) \tag{0.2}$$

Mit einem Satz geteilter Gewichte und Biases ist das Netz nun in der Lage, ein Strukturelement an beliebigen Positionen im Bild zu erkennen. Um es dem Netz nun zu gestatten, mehrere verschiedene Strukturen zu erkennen, vervielfacht man diesen Vorgang mit je eigenen Gewichten und Biases. Es entstehen sogenannte Featuremaps. Dieser Vorgang ist in Abbildung 0.3 dargestellt. Das Netzwerk ist somit in der Lage verschiedene Strukturelemente an beliebigen Positionen zu erkennen. Erweitert man das Netz um weitere Schichten, kann es mit entsprechendem Training auch lernen, welche Kombinationen dieser Elemente zu einem Mustertyp gehören.

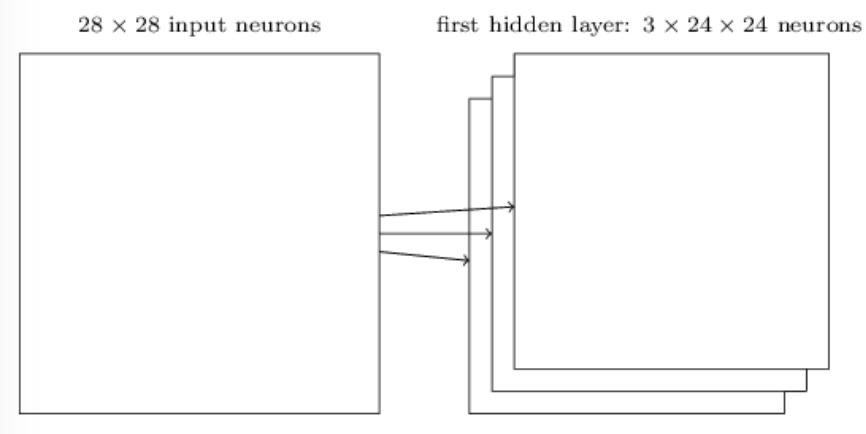

Figure 0.3: Darstellung verschiedener Featuremaps (vgl. **NNADL_PIC_CONV_FEATURES**)

Damit erweitert sich der eingangs genannte Vorteil von CNNs. Nicht nur die Verschiebung ganzer Muster innerhalb eines Bildes kann erkannt werden, sondern auch wiederkehrende Strukturelemente können für die Erkennung anderer Muster durch entsprechende Kombination in weiteren Schichten genutzt werden.

0.2 Pooling

Die Erzeugung von vielen Featuremaps aus einer einzelnen Eingabe gehört zu den grundlegenden Eigenschaften der im vorherigen Abschnitt beschriebenen Faltungsoperation. Dies führt allerdings zu einer erheblichen Vergrößerung der Datenmenge, die von nachfolgenden Layern bearbeitet werden muss. Um das Ausmaß dieser Vergrößerung einschätzen zu können, wird an dieser Stelle die Größe des Outputs einer Faltung in Abhängigkeit von der Eingabegröße und weiterer Eigenschaften der Faltungsschicht untersucht. In den nachfolgenden Gleichungen steht S für die Größe einer einzelnen Ein- bzw. Ausgabe. Die Variablen x, y und f stehen für die Dimensionen der Ein- bzw. Ausgabe. Die Größe des Filterkernels ist durch F_x und F_y bestimmt, für die

Schrittweite wird der Wert 1 angenommen.

$$\begin{aligned} S_{in} &= x_{in} y_{in} f_{in} \\ S_{out} &= x_{out} y_{out} f_{out} \\ x_{out} &= x_{in} - F_x + 1 \\ y_{out} &= y_{in} - F_y + 1 \end{aligned} \tag{0.3}$$

Unter der Bedingung, dass $x_{in} >> F_x$ und $y_{in} >> F_y$ ergibt sich die Näherung $\frac{S_{out}}{S_{in}} = \frac{f_{out}}{f_{in}}$. Die Anzahl der Neuronen einer Faltungsschicht gegenüber der des vorherigen Layers steigt bei der Faltungsoperation demzufolge annähernd proportional zur Anzahl der Featuremaps. Mit der Datenmenge im Ausgang der Faltungsschicht steigt auch die Zahl der Aktivierungen, die in den nachfolgenden Schichten bearbeitet werden müssen. Dies stellt einen beträchtlichen Mehraufwand dar und kann zu einer signifikanten Steigerung der Rechenzeit und des Speicherbedarfs führen. Um diesen Mehraufwand zu reduzieren, sind die Faltungsschichten in den meisten Implementierungen von CNNs von sogenannten Pooling-Schichten gefolgt. Beim Pooling wird jede Featuremap in rechteckige Abschnitte unterteilt, die dann jeweils auf einen neuen Aktivierungswert abgebildet werden. Für diese Abbildung gibt es mehrere Methoden, einige davon werden im folgenden Abschnitt näher erläutert.

0.2.1 Average pooling

TODO: Inhalt von **paperMixedPooling** kann auf mehrere Arten zitiert werden [**paperMixedPooling**]

0.2.2 Max pooling

0.2.3 Vergleich

0.3 Anpassungen der Backpropagation