

Training von CNNs¹

Vorlesung 12, Artificial Intelligence

Dozenten:

Prof. Dr. M. O. Franz, Prof. Dr. O. Bittel, Prof. Dr. O. Dürr

HTWG Konstanz, Fakultät für Informatik

¹Folien basieren z.T. auf B. Sick, O. Dürr, DL Course.

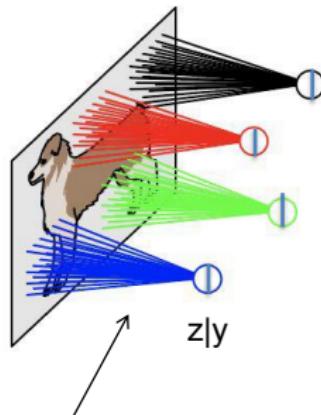
Übersicht

- 1 Tricks of the trade
- 2 Was tun bei kleineren Datensätzen?
- 3 Analyse trainierter Netzwerke

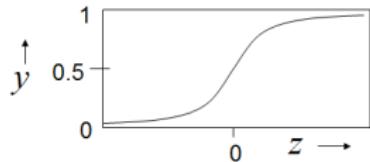
Übersicht

- 1 Tricks of the trade
- 2 Was tun bei kleineren Datensätzen?
- 3 Analyse trainierter Netzwerke

Gemeinsame Gewichte in der Faltungsschicht erfordern eine Standardisierung der Daten



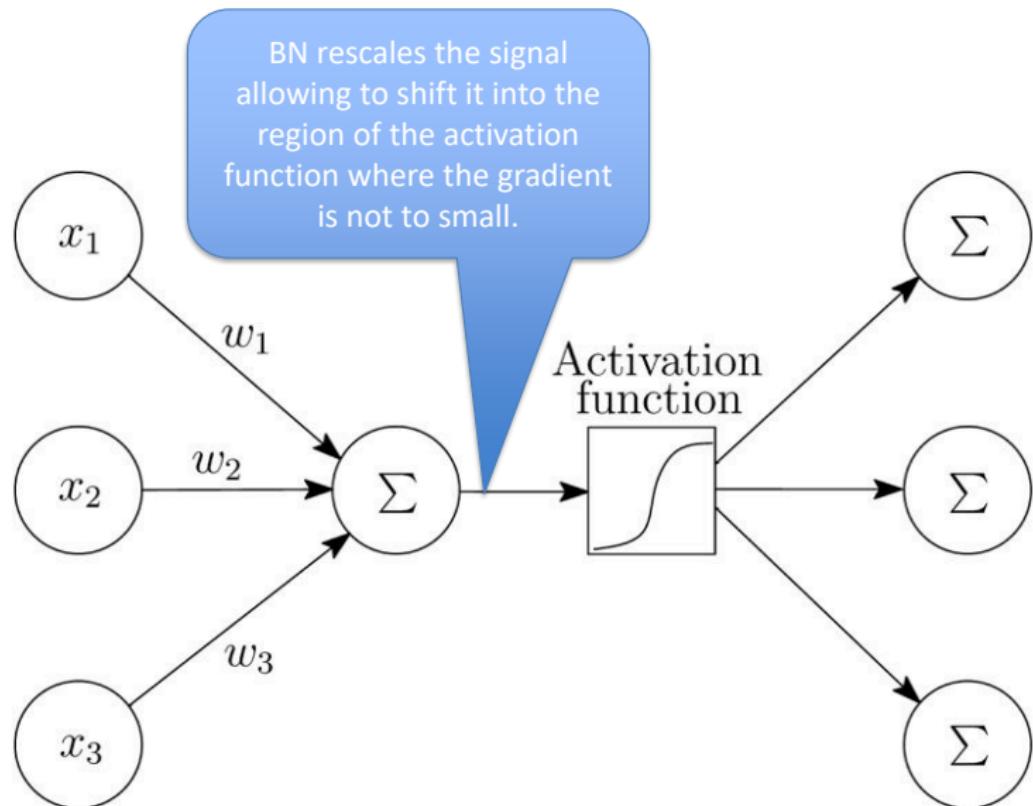
$$z = b + \sum_i x_i w_i$$



Alle Faltungsneuronen teilen sich denselben Gewichtssatz.

Der annähernd lineare Arbeitsbereich der Aktivierungsfunktion kann nur genutzt werden, wenn alle lokalen Bildausschnitte eine ähnliche Werteverteilung haben. Aufgrund der gemeinsamen Gewichte können sich die Faltungsneuronen nicht an lokal stark veränderliche Bedingungen anpassen.

Grundidee der Batch-Normierung (batch normalization)



Berechnung der Batch-Normierung

Eine Batch-Normierungsschicht wird vor eine Faltungsschicht geschaltet:

- ① Standardisierung:

$$x' = \frac{x - \text{mean}_{\text{batch}}(x)}{\text{stddev}_{\text{batch}}(x) + \epsilon}$$

x' hat nach diesem Schritt Mittelwert 0 und Standardabweichung 1; ϵ ist eine kleine Konstante zur Verhinderung einer Nulldivision bei konstanten Batches.

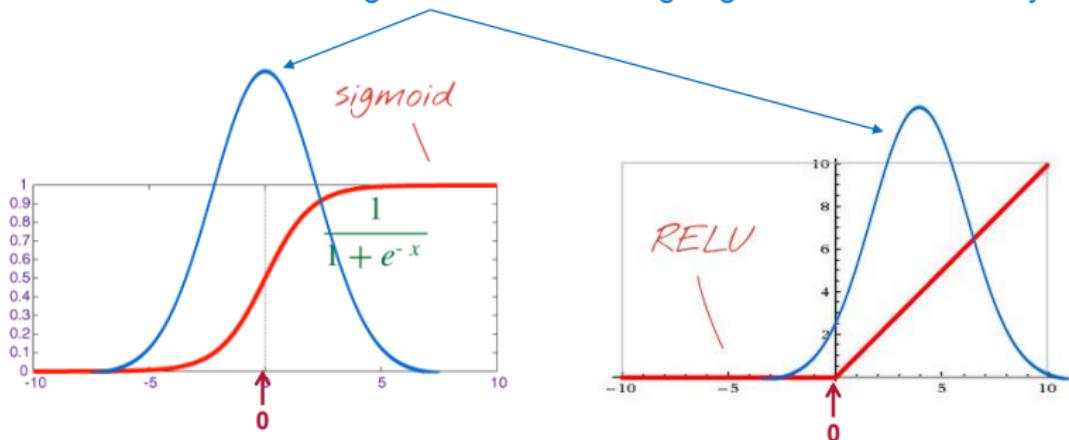
- ② Anpassung von Mittelwert und Standardabweichung:

$$\text{BN}(x') = \alpha x' + \beta$$

Die Parameter α und β werden während des Trainings gelernt. Mit ihnen können Mittelwert und Standardabweichung an den Arbeitsbereich der Aktivierungsfunktion angepasst werden.

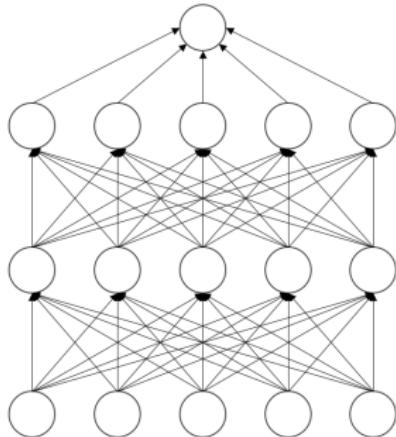
Verteilung des gewichteten Inputs nach der Batch-Normierung

Observed distributions of signal after BN before going into the activation layer.

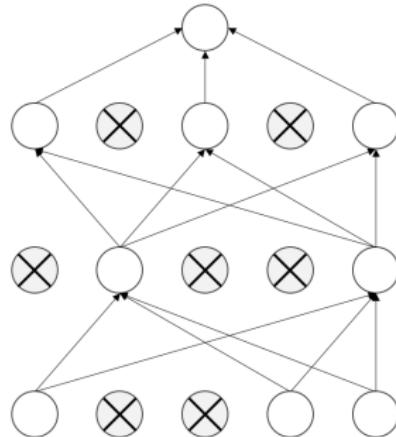


- Batch-Normierung beschleunigt häufig den Trainingsprozess. Es können höhere Lernraten verwendet werden.
- Biasterm kann in der anschließenden Schicht weggelassen werden (β hat denselben Effekt)
- Bei ReLU kann die Anpassung von α entfallen, da der Arbeitsbereich nicht nach oben begrenzt ist.

Dropout - Training



Standard Neural Net

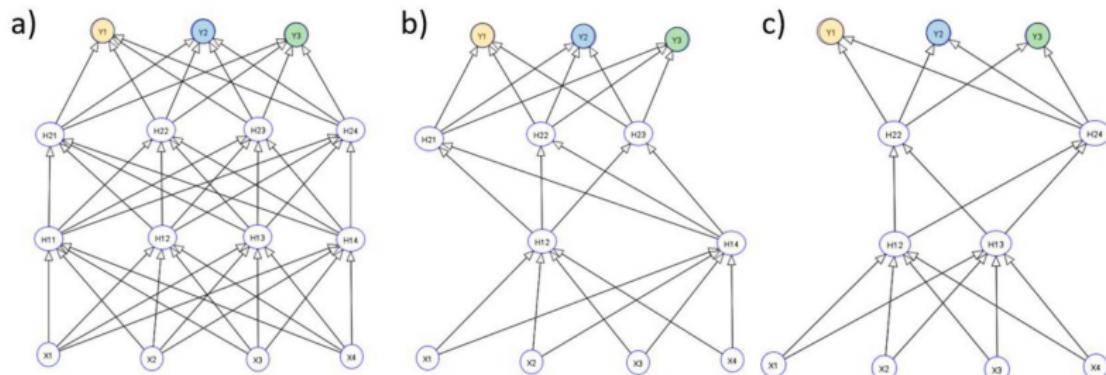


After applying dropout

[Deep Learning for Computer Vision]

Dropout: bei jedem Update wird ein bestimmter Prozentsatz p von zufällig gewählten Neuronen abgeschaltet. Dadurch wird in jedem Schritt ein leicht anderes neuronales Netz trainiert.

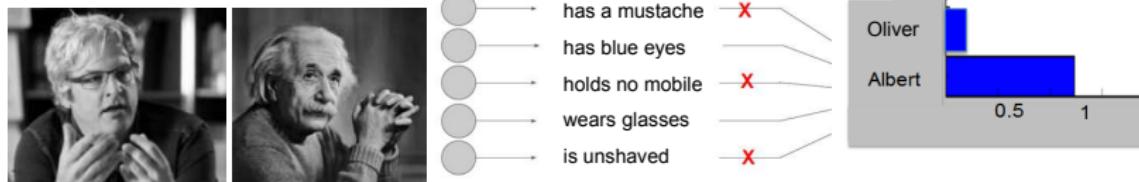
Dropout - fertig trainiertes Netzwerk



Der Output des fertig trainierten Netzwerks ist der Mittelwert aus den einzelnen Unternetzwerken der Trainingsphase. Da im Training nur p Prozent der Neuronen aktiv waren, wird der Gesamtoutput in der Testphase um den Faktor $1/p$ zu hoch sein. Um den gleichen Output wie im Training zu erhalten, muss also der Output mit p multipliziert werden.

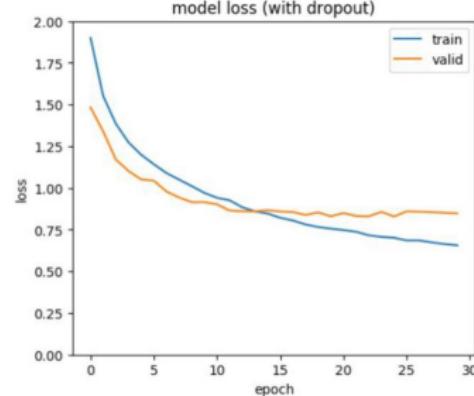
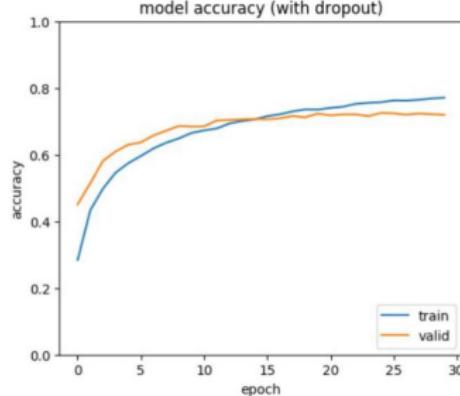
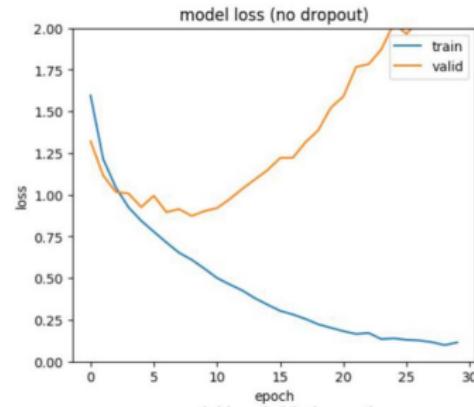
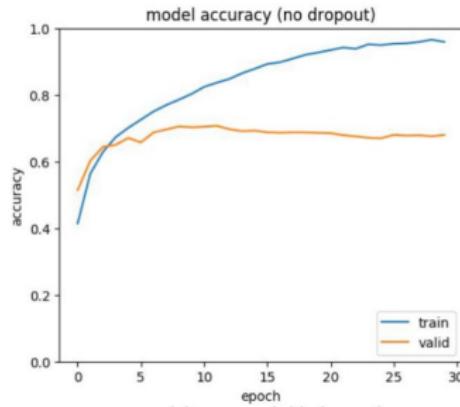
Warum hilft Dropout?

- Netzoutput ist Mittelwert über ein Ensemble. Dadurch wird die Genauigkeit und die Robustheit gegenüber Ausreißern erhöht.
- Die Unternetzwerke haben eine einfachere Struktur, was die Gefahr des Overfittings erniedrigt.
- Dropout zwingt das Netzwerk zur Benutzung von redundanten und unabhängigen Merkmalen.



[B.Sick]

Beispiel: Weniger Overfitting bei CIFAR10 durch Dropout



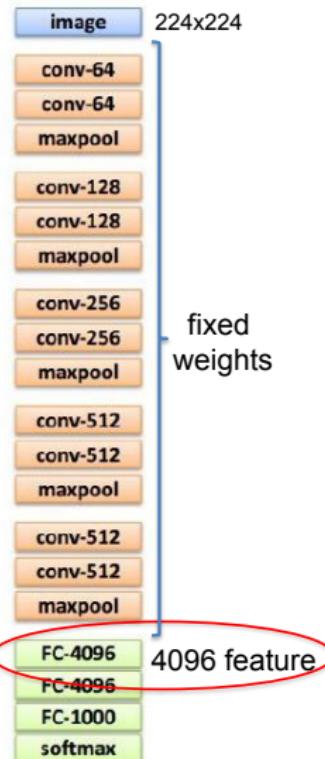
[Demo]

Übersicht

- 1 Tricks of the trade
- 2 Was tun bei kleineren Datensätzen?

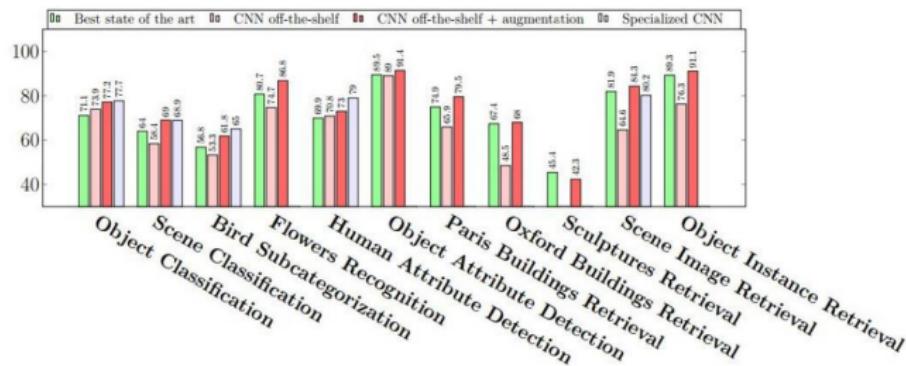
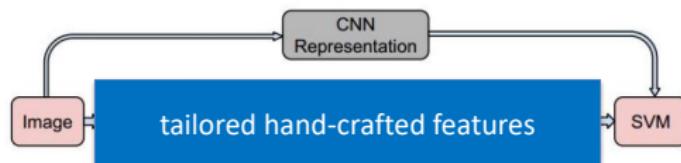
- 3 Analyse trainierter Netzwerke

Transferlernen: vortrainiertes CNN als Merkmalsdetektor



- ① Ausgangspunkt ist ein bereits auf einer ähnlichen Aufgabe trainiertes Netzwerk.
- ② Die Eingangsbilder für die neue Aufgabe müssen auf die Eingangsgröße des CNN skaliert werden (z.B. 224x224 bei VGG16)
- ③ Für alle neuen Trainingsbeispiele wird ein Vorwärtstlauf durchgeführt.
- ④ Die Ausgangswerte der ersten vollverbundenen Schicht werden als Merkmalsvektoren abgespeichert.
- ⑤ Mithilfe dieser Merkmalsvektoren wird eine einfachere Lernmaschine auf die neue Aufgabe trainiert, z.B. ein vollverbundenes neuronales Netz, Supportvektormaschine, Random Forest etc.

Vergleich von vortrainierten und speziell von Hand angepassten Merkmalen

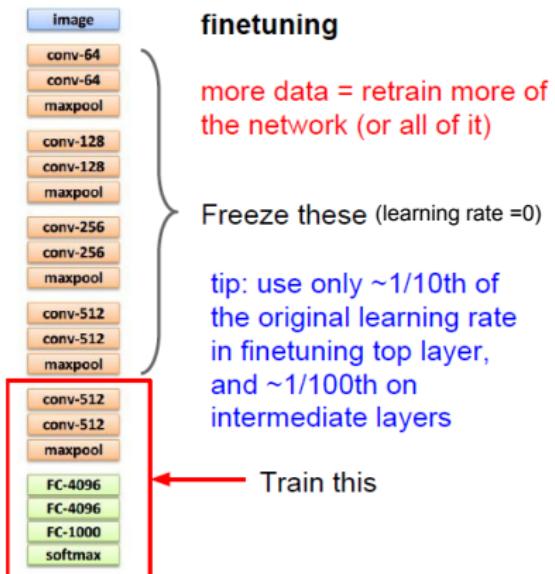


"Astonishingly, we report consistent superior results compared to the highly tuned state-of-the-art systems in all the visual classification tasks on various datasets."

CNN Features off-the-shelf: an Astounding Baseline for Recognition [Razavian et al, 2014]

Feineinstellung beim Transferlernen

e.g. medium data set (<1M images)

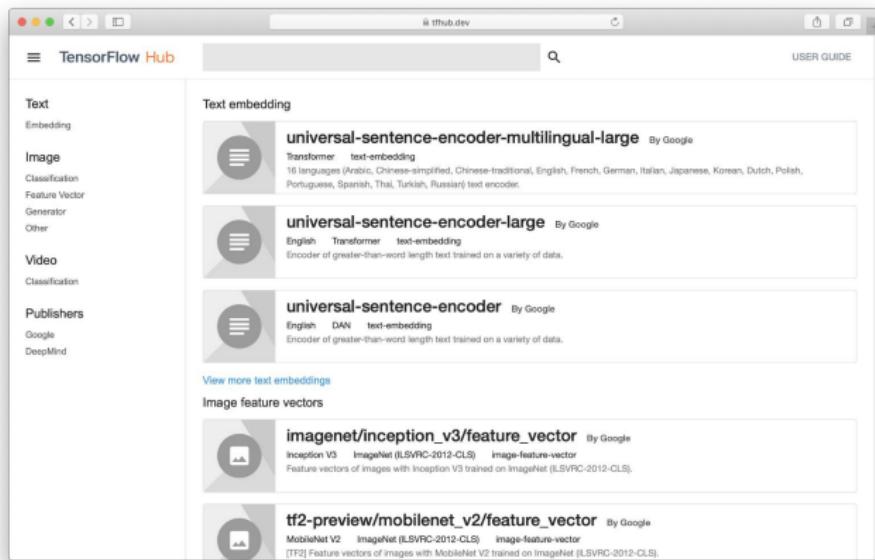


	Ähnliche Lernaufgabe	Stark abweichende Lernaufgabe
wenig Daten	Merkmalsvektoren aus den oberen vollverbundenen Schichten, Training eines externen Klassifikators	problematisch, evtl. hilft versuchsweises Training mit Merkmalen aus verschiedenen Schichten
viele Daten	Feineinstellung der hinteren Schichten, incl. einiger Faltungsschichten	Feineinstellung vieler Schichten.

Oft ist es sinnvoll, zunächst nur die vollverbundenen Schichten zu trainieren, und erst danach alle hinteren Schichten incl. einiger Faltungsschichten.

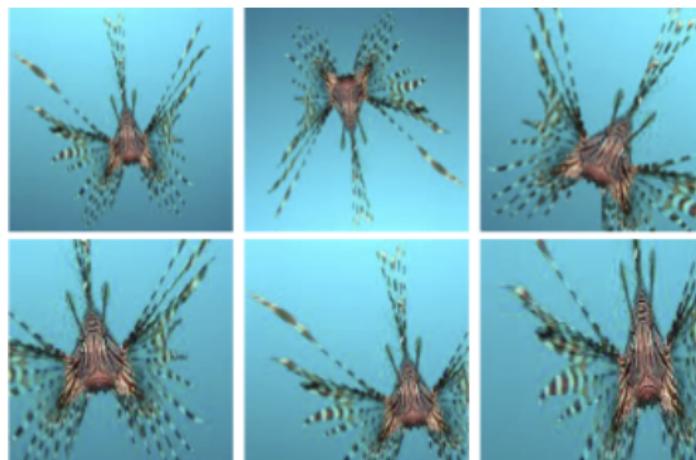
Wo gibt es vortrainierte Netze?

- Tensorflow/Keras: <https://tfhub.dev/>



- PyTorch: <https://pytorch.org/vision/stable/models.html>
- Caffe: <https://github.com/BVLC/caffe/wiki/Model-Zoo>

Data Augmentation ("Datenvermehrung")



Sehr häufig werden aus den ursprünglichen Trainingsdaten durch Transformationen künstliche neue Daten erzeugt, z.B.

- Rotationen
- Skalierungen
- Verschiebungen
- Achsenspiegelungen
- Helligkeits- und Kontraständerungen

Wenn die Transformationen gut an die tatsächlich im Problem vorkommenden Variationen angepasst sind, kann die Klassifikationsleistung erheblich verbessert werden.

Data Augmentation in Keras

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    #zoom_range=[0.1,0.1]
)

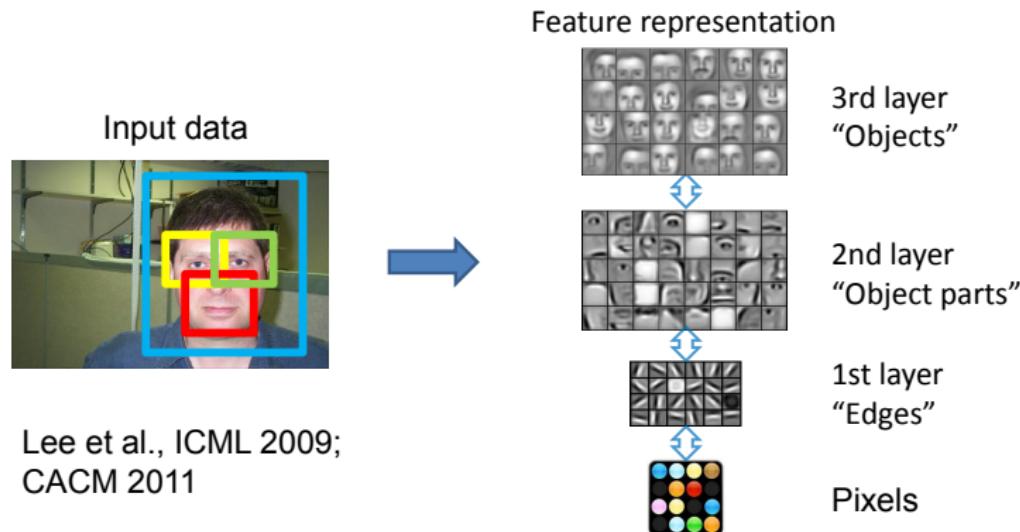
train_generator = datagen.flow(
    x = X_train_new,
    y = Y_train,
    batch_size = 128,
    shuffle = True)

history = model.fit_generator(
            train_generator,
            samples_per_epoch = X_train_new.shape[0],
            epochs = 400,
            validation_data = (X_valid_new, Y_valid),
            verbose = 2, callbacks=[checkpointer])
```

Übersicht

- 1 Tricks of the trade
- 2 Was tun bei kleineren Datensätzen?
- 3 Analyse trainierter Netzwerke

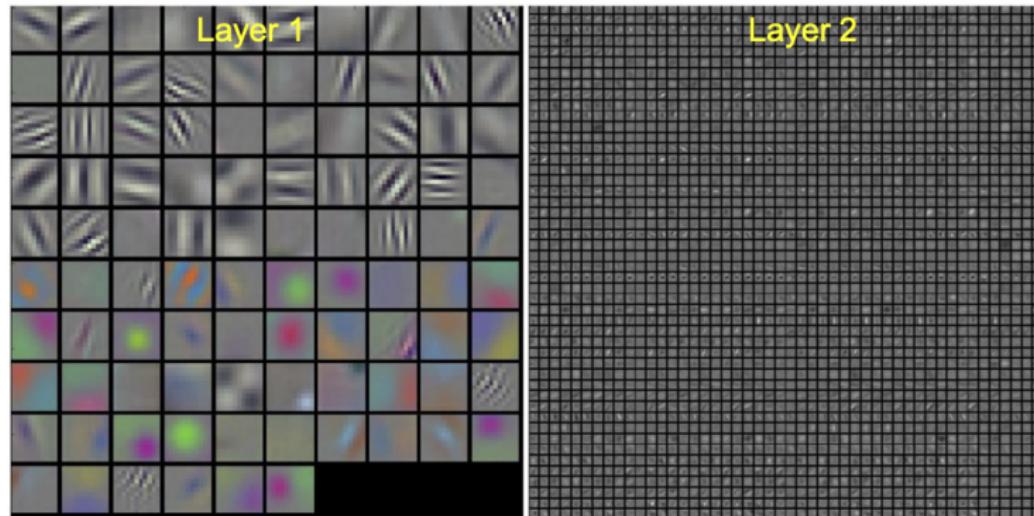
Das Ideal: eine gelernte Merkmalshierarchie



Lee et al., ICML 2009;
CACM 2011

- **Verteilte Repräsentation:** Merkmale in Zwischenschichten werden vielfach wiederverwendet.
- **Grundidee:** eine Schicht extrahiert Merkmale aus der Vorgängerschicht.

Bildliche Darstellung der gelernten Filtermasken eines CNNs

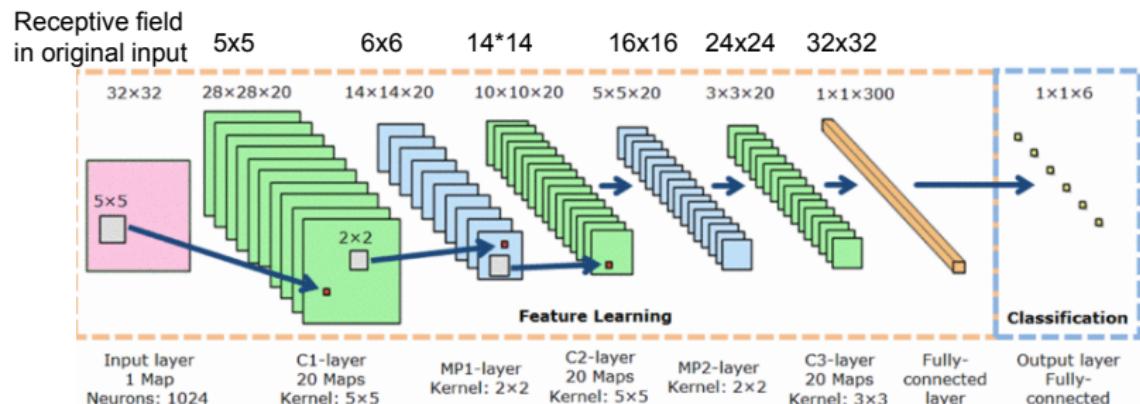


[<http://cs231n.github.io/understanding-cnn/>]

Meist findet man nur in der Eingangsschicht interpretierbare Muster (Kanten , Blobs, etc.). In den hinteren Schichten lässt sich nur überprüfen, ob die Filtermasken stark verrauscht sind, was auf zu kurzes Training oder zu wenig Regularisierung hindeutet.

Rezeptives Feld

Rezeptives Feld eines Neurons: Der Bereich des Eingangsbildes, der den Ausgangswert des Neurons beeinflussen kann, d.h. der vom Neuron gesehen wird.



[<http://pubs.sciepub.com/ajme/2/7/9/figs>]

Die Neuronen der hinteren Schichten haben meist größere rezeptive Felder als die Neuronen der vorderen Schichten.

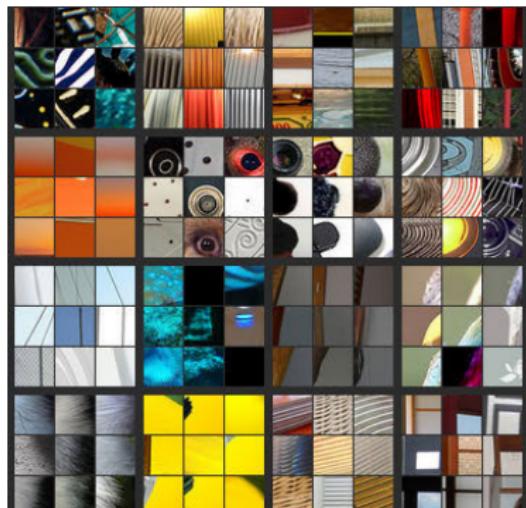
Code zur Größenbestimmung des rezeptiven Feldes:

<http://stackoverflow.com/questions/35582521/how-to-calculate-receptive-field-size>

Welche Bildausschnitte im rezeptiven Feld führen zur maximalen Aktivierung eines Neurons? (1)



Erste Schicht



Zweite Schicht

[Zeiler & Fergus, 2013]

Welche Bildausschnitte im rezeptiven Feld führen zur maximalen Aktivierung eines Neurons? (2)



Dritte Schicht

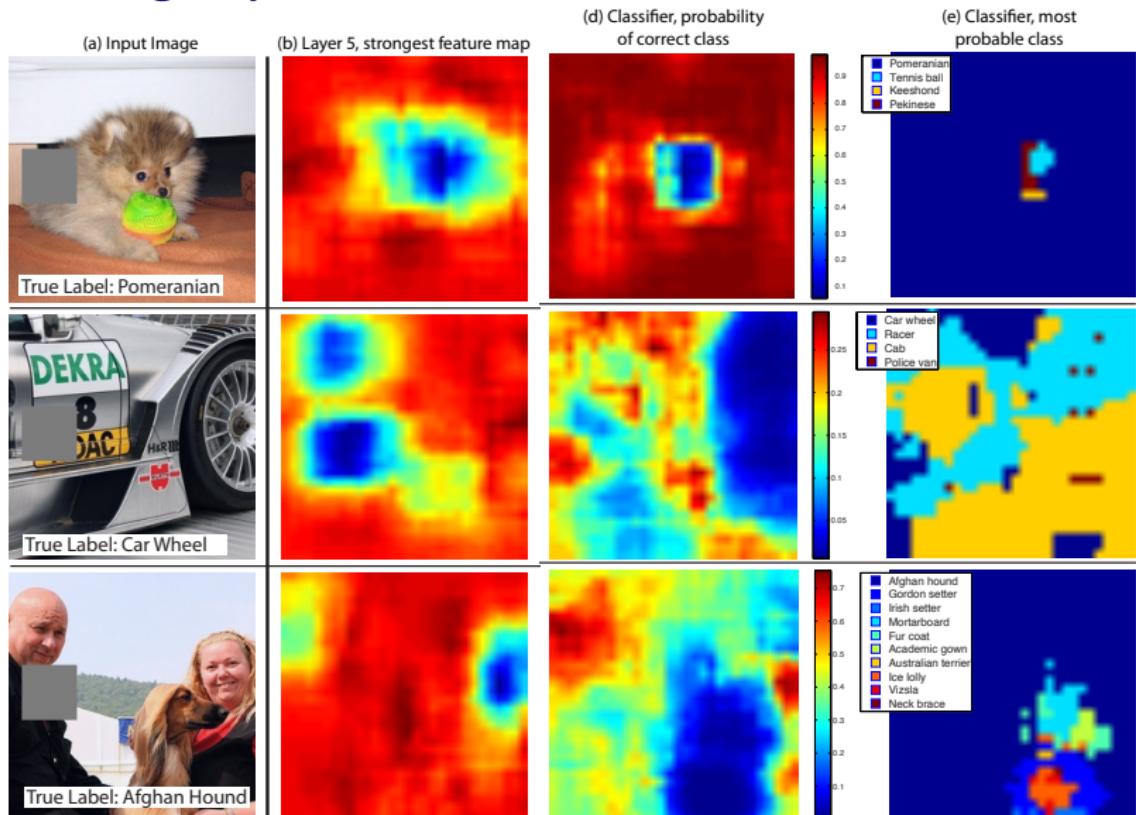


Vierte Schicht

[Zeiler & Fergus, 2013]

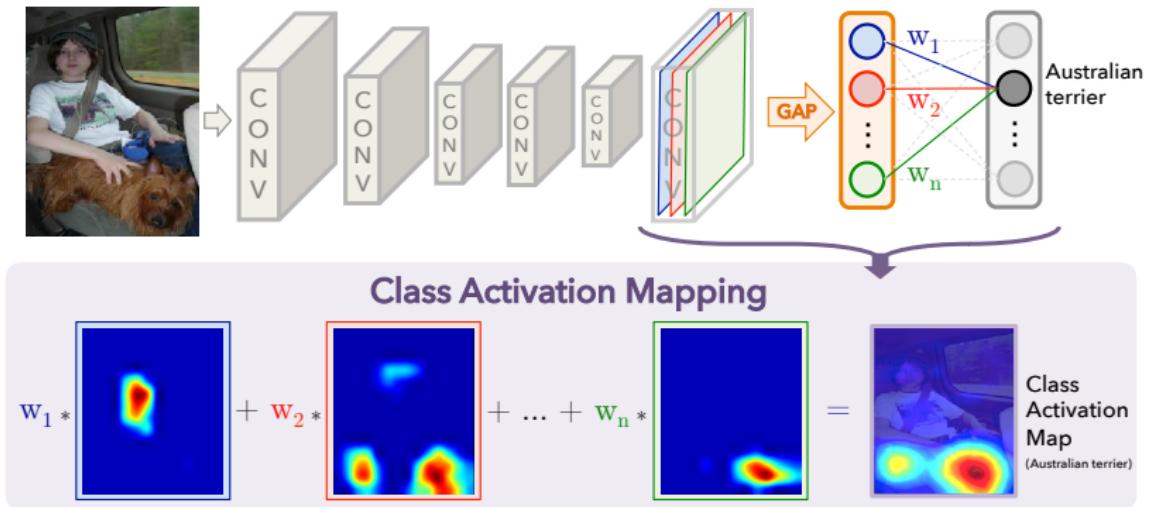
⇒ Die gewünschte Merkmalshierarchie ist tatsächlich vorhanden, aber nicht in den Filtermasken der höheren Schichten sichtbar. Grund: die Merkmale, auf die ein Neuron reagiert, ergeben sich als Kombination aller Vorgängerschichten.

Verdeckungsexperimente



[Zeiler & Fergus, 2013]

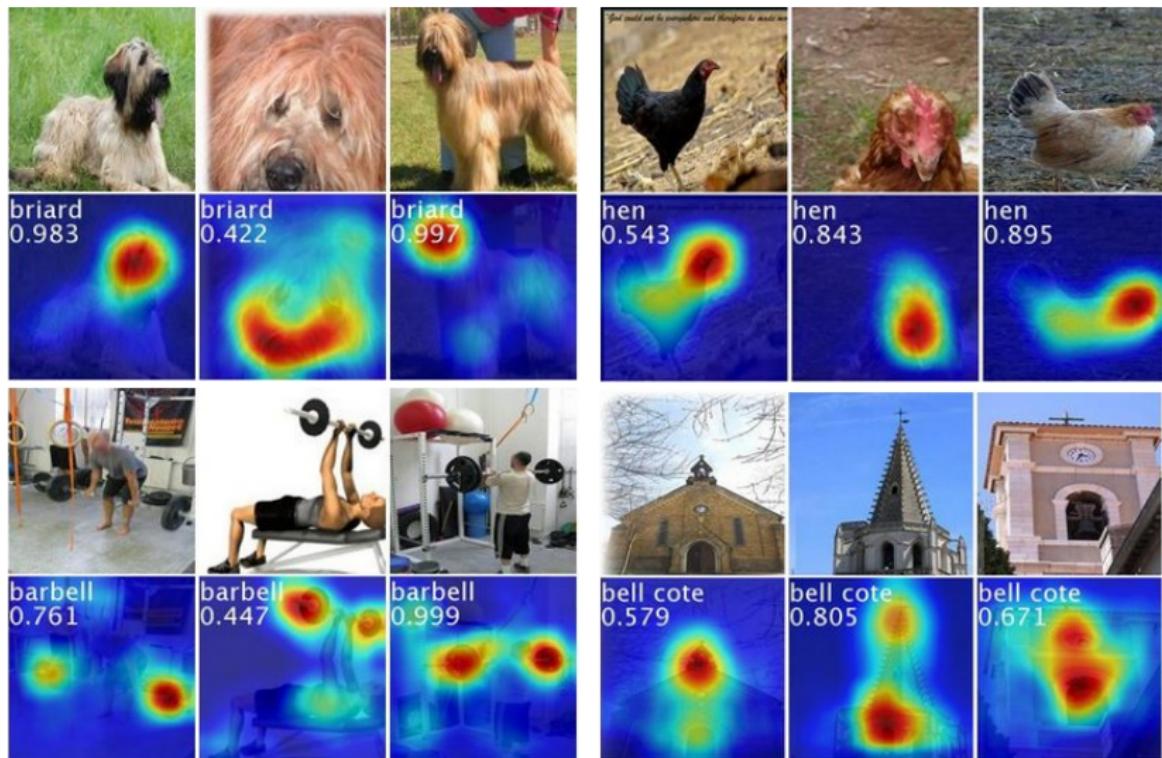
Class Activation Mapping (CAM)



[<https://arxiv.org/abs/1512.04150>]

Die Aktivierungskarten der letzten Faltungsschicht werden gewichtet aufsummiert (globales Average Pooling) und auf die Größe des Eingangsbildes skaliert. Die Gewichte w_i sind der letzten vollverbundenen Schicht entnommen. CAM funktioniert nur, wenn nach den Faltungsschichten eine einzige vollverbundene Schicht folgt.

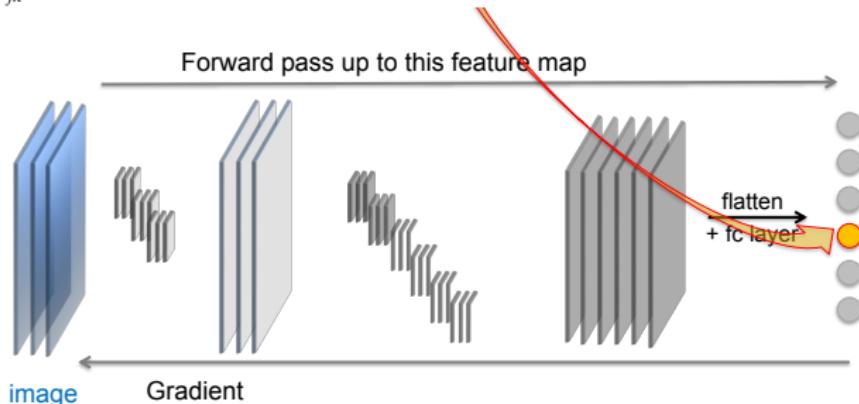
Beispiele: Class Activation Mapping



[<https://arxiv.org/abs/1512.04150>]

Gradient Backpropagation

Welche Pixel würden zu keiner Aktivierung des Neurons führen, wenn sie andere Werte hätten? Für welche Inputpixel x_{jk} ist der Gradient der neuronalen Aktivierung $\frac{\partial a_j}{\partial x_{jk}}$ groß?



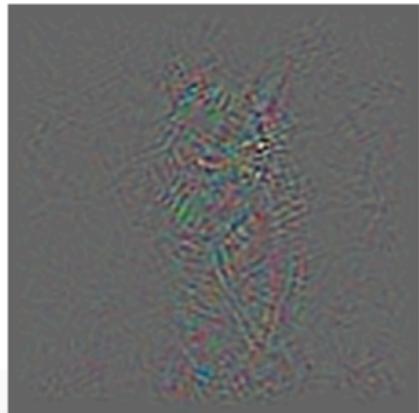
- 1 Für ein gegebenes Eingangsbild wird der Vorwärtsdurchlauf des Netzwerks berechnet.
- 2 Das Ausgangssignal a_j von Neuron j wird auf 1 gesetzt, alle anderen Ausgangssignale auf 0.
- 3 Mit diesem Ausgangssignal (statt wie sonst mit dem Gradienten der Kostenfunktion) wird der Rückwärtstlauf mit Backpropagation berechnet.

Typischer Gradient eines Ausgangsneurons

Input image

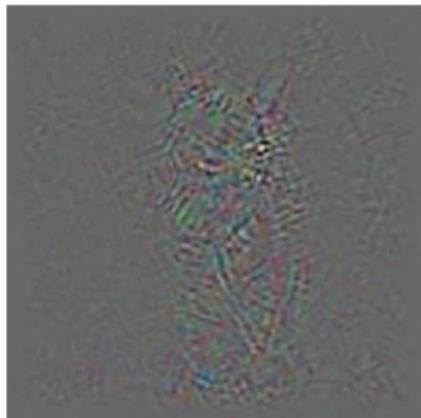


Backpropagation



Außerhalb des Objekts ist der Gradient wie erhofft 0, allerdings sind die Bereiche innerhalb des Objekts schwer interpretierbar. Grund: negative Gradienten unterdrücken die Weiterverarbeitung, wenn dort Aktivität ist. Diese hemmenden Bereiche haben selten eine interpretierbare Beziehung zum Objekt.

Guided Backpropagation



Backprop



Guided Backprop

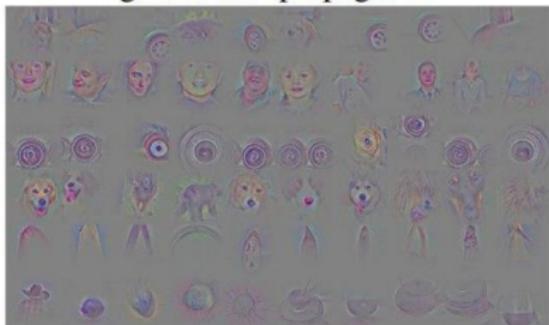
Grundidee: Neuronen funktionieren als Detektoren für bestimmte Bildmerkmale. Uns interessiert eher, welche Merkmale das Neuron detektiert, nicht welche Merkmale seine Aktivität unterdrücken. Daher werden alle negativen Gradienten beim Rückwärtslauf auf 0 gesetzt.

Beispiele: Guided Backprop

guided backpropagation



guided backpropagation



corresponding image crops



corresponding image crops



Springerberg et al, Striving for Simplicity: The All Convolutional Net (ICLR 2015 workshops)