

# Maschinelles Lernen: Neuronale Netze

Ideen der Informatik

Kurt Mehlhorn



Januar 2019



SIC Saarland  
Informatics Campus



- Stand der Kunst: Bilderverstehen
- Was ist ein Bild in Rohform?
- Biologische Inspiration: das menschliche Sehsystem
- Künstliche Neuronale Netze
  - Künstliche Neuronen,
  - Neuronale Netze
  - Realisierung von Und, Oder und Negation
- Trainieren von Neuronalen Netzwerken
  - Prinzip
  - Buchstaben: Stand der Kunst in 1986
- Bilderkennung heute (Deep Convolutional Networks)
- Deep Neural Networks erleben einen Boom (83 Mio Google Hits am 8.1.2020), auch Spracherkennung, maschinelle Übersetzung, Spiele lernen, usw.
- Schwächen und Gefahren



# Stand der Kunst: Klassifikation (Krizhevsky et al, 2012).



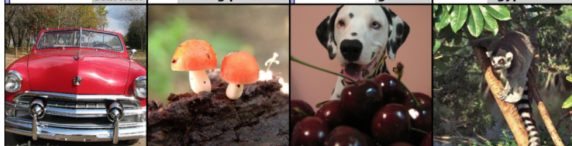
**mite**

**container ship**

**motor scooter**

**leopard**

mite black widow cockroach tick starfish	container ship lifeboat amphibian fireboat drilling platform	motor scooter go-kart moped bumper car golfcart	leopard jaguar cheetah snow leopard Egyptian cat



**grille**

**mushroom**

**cherry**

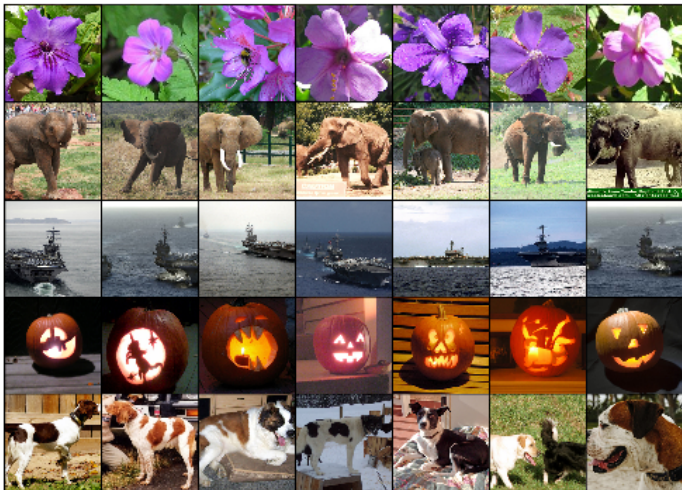
**Madagascar cat**

convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bullterrier currant	squirrel monkey spider monkey titi indri howler monkey

**Training:** 1.2 Mio Bilder, eingeteilt in 1000 Klassen.

**Klassifikation:**  
 neue Bilder.  
 System soll sagen, zu welcher Klasse das Bild gehört.

# Stand der Kunst: Suche (Krizhevsky et al, 2012), 1.2 Mio Bilder, 1000 Klassen



Frage = erste Spalte (neues Bild),

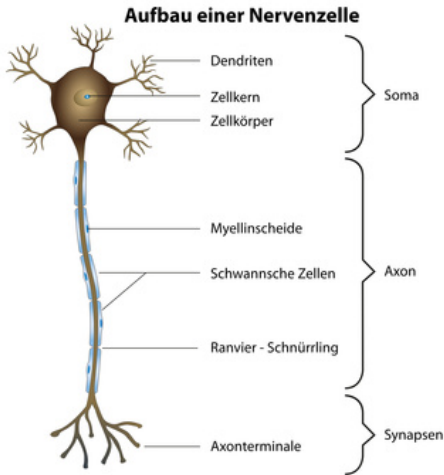
Antworten = andere Spalten  
(aus Trainingsmenge)



# Was ist ein Bild?

- Eine Matrix von Pixeln, z.B.  
1920 x 1080 Pixel bei HD-TV.
- Für jedes Pixel die Sättigung in den drei Grundfarben:  
rot, grün, blau  $\in \{0, \dots, 2^{16} - 1\}$ .
- Pixel sind so klein, dass das menschliche Auge sie nicht auflösen kann.





Axonterminale = Output

Dendriten = Input

Zelle feuert, wenn Gesamterregung einen Schwellwert übersteigt

Inputs können auch hemmend sein.

Visueller Kortex ist schichtenweise aufgebaut; 6 Schichten.

Neuronen der ersten Schicht bekommen Input von einem kleinen Feld von Sehzellen, Neuronen höherer Schichten von einem kleinen Feld der davorliegenden Schicht.

One-Learning Algorithm Hypothese, Mausexperiment



- Neuron hat  $k$  eingehende Kanten.
- Eingabewerte  $x_1$  bis  $x_k$  in 0 bis 1.
- $k + 1$  Gewichte (Parameter)  
 $w_0, w_1, \dots, w_k$ .
- $w_0$  heißt Grundgewicht (Bias);  $w_i$  ist die Wichtung von  $x_i$ .
- Ausgabe =  $g(w_0 + w_1 x_1 + \dots + w_k x_k)$ .
- $g$  = Sigmoid Funktion.
- Sigmoid Funktion ist differenzierbare Approximation einer Stufe von 0 nach 1 an der Stelle 0.



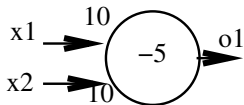
$$g(z) = \frac{1}{1 + e^{-z}}$$

- $g(0) = 1/2$ .
- $g(1) = 0.73$
- $g(4) = 0.95, g(10) = 0.99995$
- symmetrisch zu  $(0, 1/2)$ .
- differenzierbare Approximation einer Stufe von 0 nach 1 an der Stelle 0.
- wenn man  $e^{-z}$  durch  $e^{-10z}$  ersetzt, wird Flanke steiler.

$$g(z) + g(-z) = 1$$

$$g'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = g(z)(1 - g(z))$$

# Realisierung von Oder und ...



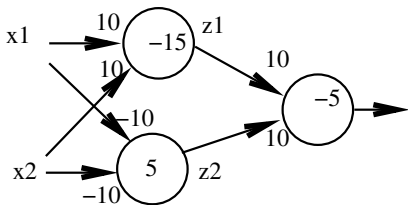
$$o_1 = g(-5 + 10x_1 + 10x_2) \approx x_1 \vee x_2$$

$$o_2 = g(15 + 10x_1 - 10x_2) \approx$$

$x_1$	$x_2$	$o_1 =$	$o_1 \approx$	$o_2 =$	$o_2 \approx$
0	0	$g(-5)$	0		
0	1				
1	0				
1	1				

## Komplexeres Beispiel

Welche Boolesche Funktion wird durch dieses Neuronale Netz berechnet?



$x_1$	$x_2$	$z_1 =$	$z_1 \approx$	$z_2 =$	$z_2 \approx$	$o =$	$o \approx$
0	0	$g(-15)$	0				
0	1						
1	0						
1	1						

$$o = g(-5 + 10 \cdot g(-15 + 10x_1 + 10x_2) + 10 \cdot g(5 - 10x_1 - 10x_2))$$

Bisher: Wir haben bestimmt, welche Funktion durch ein Netz mit gegebenen Parameterwerten ausgerechnet wird.

Nun: Wir möchten, dass das Netz eine bestimmte Funktion realisiert. Wie müssen wir die Parameter einstellen?



Bisher: Wir haben bestimmt, welche Funktion durch ein Netz mit gegebenen Parameterwerten ausgerechnet wird.

Nun: Wir möchten, dass das Netz eine bestimmte Funktion realisiert. Wie müssen wir die Parameter einstellen?

Im Prinzip ist das möglich.

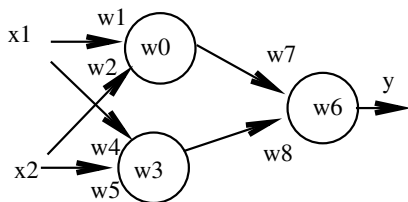
Wenn man das Netz nur groß genug macht, dann kann man jede Funktion  $f$  mit  $n$  Eingängen in  $[0, 1]$  und einem Ausgang in  $[0, 1]$  beliebig gut approximieren.

Das ist zunächst eine Existenzaussage. Es sagt noch nicht, ob es einen effizienten Algorithmus gibt, die Gewichte zu finden.

## Training (Rummelhart/Hinton 86)

Bisher: Wir haben bestimmt, welche Funktion durch ein Netz mit gegebenen Parameterwerten ausgerechnet wird.

Nun: Wir möchten, dass das Netz eine bestimmte Funktion realisiert. Wie müssen wir die Parameter einstellen?



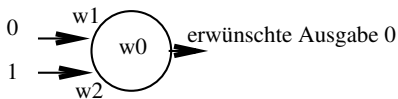
$x_1$	$x_2$	$y$
0	0	1
0	1	0
1	0	0
1	1	1

$$y = h_w(x) = g(w_6 + w_7 \cdot g(w_0 + w_1 x_1 + w_2 x_2) + w_8 \cdot g(w_3 + w_4 x_1 + w_5 x_2))$$

Aufgabe: finde  $w_0$  bis  $w_8$ , so dass die durch die Tabelle gegebene Funktion berechnet wird.

Man sagt: Das Netz wird **trainiert**. Das Netz **lernt**.

# Intuition für Trainingsschritt



Parameterwerte:  $w_0 = 0.2$ ,  $w_1 = 0.1$  und  $w_2 = -0.2$ .

Trainingsbeispiel: an Eingabe  $(0, 1)$  soll die Ausgabe  $0$  sein. Im allgemeinen gibt es viele Trainingsbeispiele.

Das Netz gibt aus:

$$h_w(0, 1) = g(0.2 + 0.1 \cdot 0 - 0.2 \cdot 1) = g(0) = 1/2.$$

Um den Fehler zu reduzieren, sollten wir  $h_w(0, 1)$  verringern. Dazu müssen wir

$$z = w_0 + 0 \cdot w_1 + 1 \cdot w_2$$

verringern. Also  $w_0$  und  $w_2$  etwas verringern und  $w_1$  gleich lassen.

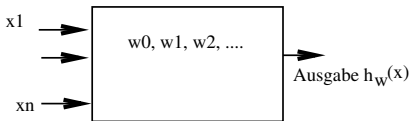
## Training II

Jedes Trainingsbeispiel (für ein Netz mit  $n$  Eingaben und einem Ausgang) besteht aus einem Eingabevektor  $x = (x_1, \dots, x_n)$  und einer Ausgabe  $y$ .

Wir haben  $N$  Trainingsbeispiele  $(x^{(1)}, y_1), \dots, (x^{(N)}, y_N)$ .

$w$  = Vektor aller Parameter (alle  $w_i$ 's).

$h_w(x)$  = Ausgabe (Hypothese) des Netzes mit Parametersatz  $w$  bei Eingabe  $x = (x_1, \dots, x_n)$



Training soll Parametersatz  $w$  finden, so dass Hypothesen des Netzes und korrekte Ausgaben möglichst gut übereinstimmen.

Was heißt gut übereinstimmen?  
Wie findet man Parametersatz?



Fehler am Trainingsbeispiel  $(x, y)$ :  $(y - h_w(x))^2$

Gesamtfehler  $E$  = Summe der Einzelfehler über alle Trainingsbeispiele

$$E = E(w) = \sum_{i=1}^N \left( y_i - h_w(x^{(i)}) \right)^2.$$

Beachte: Der Gesamtfehler ist eine Funktion der Parameter  $w$ .

Die Paare  $(x^{(i)}, y_i)$ ,  $1 \leq i \leq N$ , sind die Trainingsbeispiele. Sie sind gegeben und fest.

Präzisierung des Ziels des Trainings: Bestimme einen Parametersatz, der den Gesamtfehler minimiert.

Gesamtfehler  $E$  = Summe der Einzelfehler über alle Trainingsbeispiele

$$E = E(w) = \sum_{i=1}^N \left( y_i - h_w(x^{(i)}) \right)^2.$$

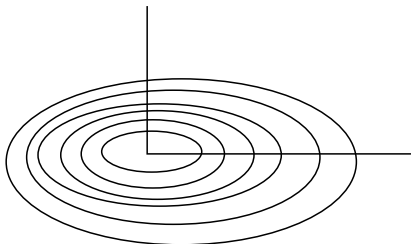
Beachte: Der Gesamtfehler ist eine Funktion der Parameter  $w$ .

Wir fragen uns für jeden der Parameter  $w_i$ : Was passiert mit dem Fehler, wenn wir  $w_i$  geringfügig ändern? Geht er hoch oder nimmt er ab?

Dann ändern wir  $w_i$  geringfügig, so dass der Gesamtfehler kleiner wird.

Das machen wir solange, bis wir zufrieden sind.

# Newton Iteration zum Minimieren einer Funktion



**Newton Iteration:** Mache immer wieder einen kleinen Schritt senkrecht zur Höhenlinie. Gehe nach unten.

Richtung senkrecht zur Höhenlinie heißt **Gradient der Funktion**; geschrieben  $\nabla E(w)$ . Newton Verfahren ist dann: mache einen kleinen Schritt in Richtung des (negativen) Gradienten

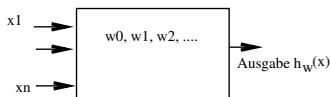
$$w^{neu} = w^{alt} - h \cdot \nabla E(w^{alt})$$

Formel für den Gradienten = Vektor der partiellen Ableitungen.

$$\nabla E(w) = \begin{pmatrix} \frac{\partial E(w)}{\partial w_0} \\ \dots \\ \frac{\partial E(w)}{\partial w_k} \end{pmatrix}$$

# Der Trainingsalgorithmus (Back Propagation)

- 1) Initialisiere die Parameter mit kleinen zufälligen Werten
- 2) Solange nicht zufrieden, d.h. Gesamtfehler  $E$  zu groß:



$$E = E(w) = \sum_{i=1}^N (y_i - h_w(x^{(i)}))^2$$

Mache einen kleinen Schritt in Richtung des negativen Gradienten, d.h., falls  $w$  der augenblickliche Wert des Parametersatzes ist, dann sind die neuen Werte

$$w_i^{(neu)} = w_i - h \frac{\partial E(w)}{\partial w_i} \quad w^{(neu)} = w - h \cdot \nabla E(w).$$

**Bemerkung:** es gibt einen einfachen Alg, um alle partiellen Ableitungen zu bestimmen.

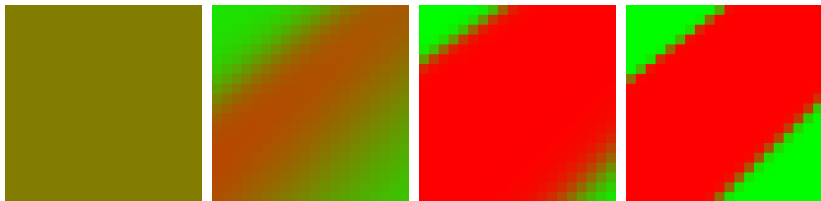
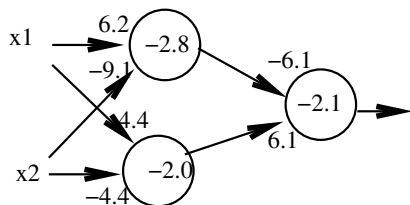
## Beispiel für das Ergebnis eines Trainings

16 Trainingsbeispiele:

$x_1$  und  $x_2$  in  $\{0, 0.1, 0.9, 1\}$

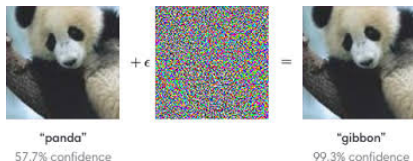
$y = 1$  genau wenn  $x_1 \approx x_2$

In 50000 Iterationen wurden folgende Parameter bestimmt.



## Verhalten auf Eingaben, die nicht trainiert wurden

Die Hoffnung ist, dass auch Eingaben, die Trainingsdaten ähneln, richtig klassifiziert werden, aber



Ähneln als Bild  $\neq$  ähneln als Vektor im  $\mathbb{R}^{\text{Anzahl der Pixel}}$

Im Beispiel haben wir nur mit Eingaben in der Nähe der Ecken trainiert. Das Netz ist sich vollkommen sicher über den Funktionswert an Mitte des Quadrats.

Geometrie im  $\mathbb{R}^2$  ist keine gute Intuition für  $\mathbb{R}^{1000}$ : Bruchteil des Volumens einer Kugel, der nahe der Oberfläche ist, steigt mit der Dimension.

# Unterscheide T und C (Convolutional Networks)

Aufgabe: in einem 25 x 25 Pixelbild befindet sich ein T oder C in einer der vier möglichen Orientierungen; fünf Pixel sind Eins, die anderen sind Null.

Bestimme ein neuronales Netz (einfacher Architektur), das T und C unterscheidet.

**einfach:** alle Neuronen der ersten Schicht schauen sich eine 3 x 3 Matrix an und sind identisch, d.h. haben die gleichen Parameter.

Ausgabeneuron hängt von allen Neuronen der ersten Schicht ab.

Anzahl der Parameter =  $10 + (25 - 2)^2 + 1$ .

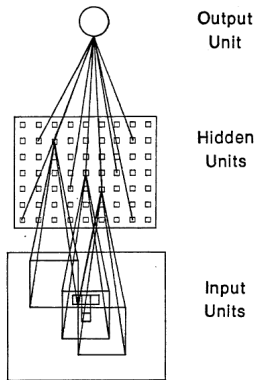
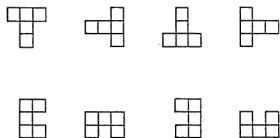
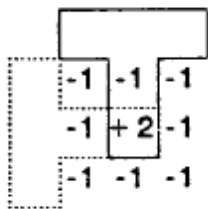


FIGURE 14. The network for solving the T-C problem. See text for explanation.

## Eine Lösung (Das Bild zeigt die Gewichte)

A



Wenn die Mitte des Filter auf dem Fuß des T's liegt, liefert der Filter eine +1.

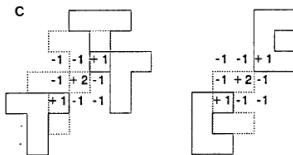
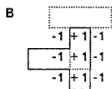
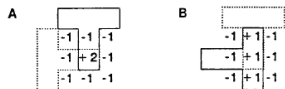
Bei einem C erzeugt der Filter immer einen Wert kleiner gleich 0.

Neuron der ersten Schicht =

$$g(-5 + 20 \cdot \text{Mittelpixel} - 10 \cdot \text{Summe der Randpixel})$$

Ausgabeneuron = Oder aller Neuronen der ersten Schicht.





**D**

-2 -2 -2  
 -2 -2 -2  
 -2 -2 -2

Training fand vier verschiedene Lösungen.

Aufgabe: finde heraus, wie B und C funktionieren.

D ist besonders interessant: es funktioniert weil ein C 20 Rezeptoren überlappt und ein T 21 Rezeptoren.

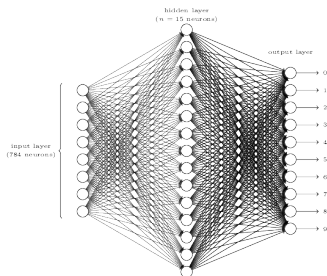
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	1	1	0
0	1	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

# Ziffernerkennung (Michael Nielsen)

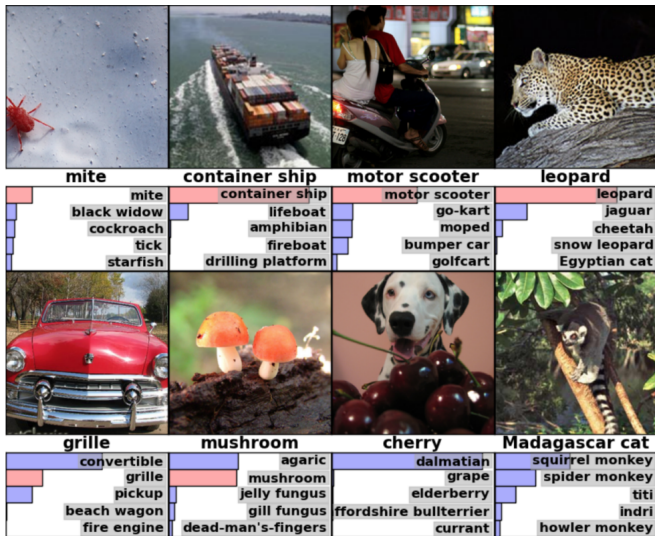


0 0 0 0 0 0 0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9 9 9 9 9 9 9

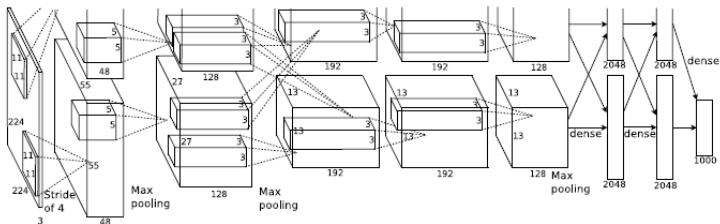
- MNIST Datensatz: 60,000 Bilder  
28 x 28 Pixel; für jedes Pixel:  
Helligkeit.
- 10 Ausgabeneuronen,  
15 Neuronen in der Mittelschicht,  
 $784 = 28 \times 28$  Eingabewerte.
- Jedes Neuron der Mittelschicht  
hat 784 Eingaben. Jedes  
Ausgabeneuron hat 15 Eingaben.
- Anzahl der Parameter =  $15 \times 785$   
 $+ 10 \times 16 = 11,935$ .
- Fehlerrate: unter 3 Prozent.
- Zahlen sind schön zentriert.



# Krishevsky et al., 2012: Netzwerk klassifiziert Bilder nach 1000 Kategorien

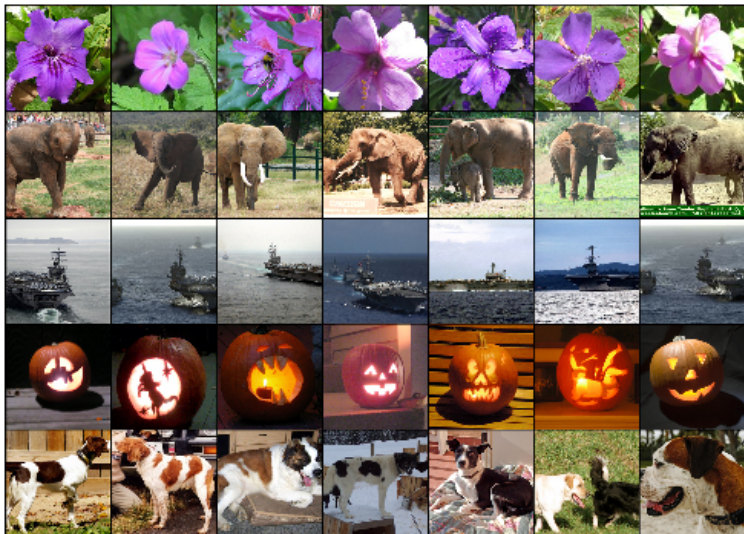


- Bilder haben 224 x 224 Pixel jeweils mit 3 Farbwerten (0 bis 255)
- Netzwerk hat 8 Schichten, 650000 Neuronen, 60 Millionen Parameter
- Ausgabeschicht hat 1000 Neuronen, eins pro Klasse
- 1.2 Millionen Trainingsbeispiele aus 1000 Klassen.
- Training dauerte 1 Woche



- Eingabe: 224 x 224 Pixel mit jeweils 3 Farbwerten
- Schicht 1: 96 verschiedene Neuronen in 55 x 55 Matrix; jedes Neuron sieht 11 x 11 Feld der Eingabe; Shift von 4.
- Schicht 2: 256 verschiedene Neuronen in 27 x 27 Matrix; jedes Neuron sieht 5 x 5 Feld der Schicht 1, also 31 x 31 Feld der Eingabe.
- Schicht 3, 4, 5: ähnlich
- Schichten 6, 7: 4096 Neuronen, sehen ganze vorherige Schicht
- Schicht 8, Ausgangsschicht: 1000 Neuronen

# Suche: Suchbild in Spalte 1, Trainingsbilder mit ähnlichster Erregung der Ausgabeneuronen



## Wie funktioniert das?

Neuronen der ersten Schicht entdecken Kanten, Linien, Bögen in 11 x 11 Feldern der Eingabe.

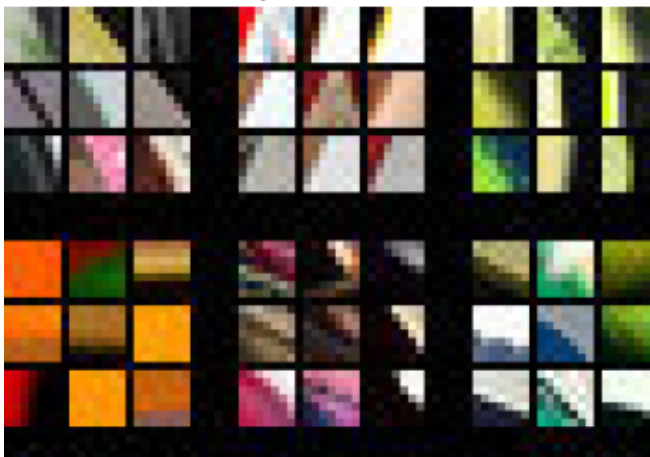


Abbildung zeigt Eingaben, bei denen 6 ausgewählte Neuronen der ersten Schicht besonders stark ansprechen.

## Wie funktioniert das?

Neuronen der zweiten Schicht entdecken komplexere Merkmale in 31 x 31 Feldern der Eingabe.

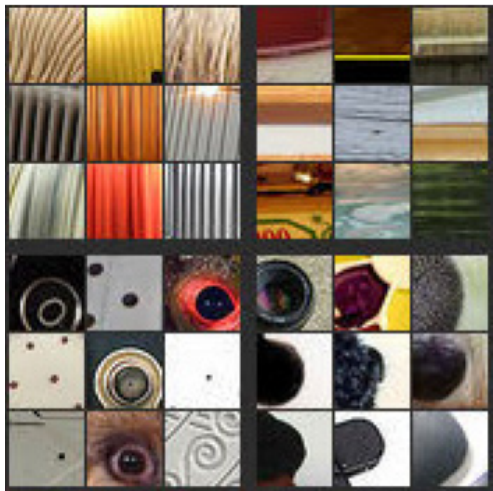


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der zweiten Schicht besonders stark ansprechen.



## Wie funktioniert das?

Neuronen der dritten Schicht entdecken noch komplexere Merkmale in wiederum größeren Feldern der Eingabe.

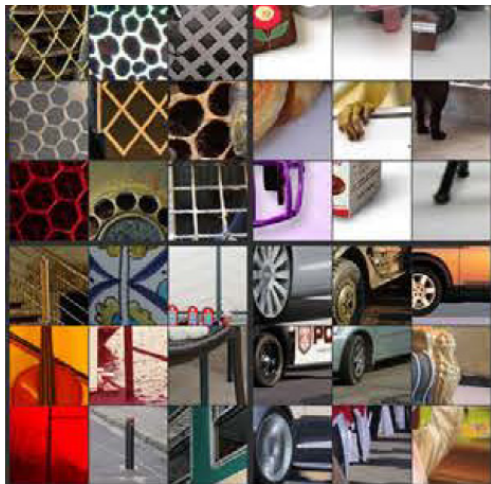


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der dritten Schicht besonders stark ansprechen.

# Wie funktioniert das?

Neuronen der vierten Schicht entdecken noch komplexere Merkmale in wiederum größeren Feldern der Eingabe.

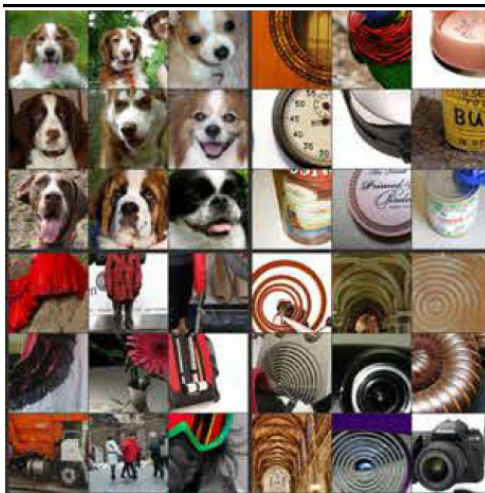


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der vierten Schicht besonders stark ansprechen.

# Wie funktioniert das?

Neuronen der fünften Schicht entdecken noch komplexere Merkmale in wiederum größeren Feldern der Eingabe.

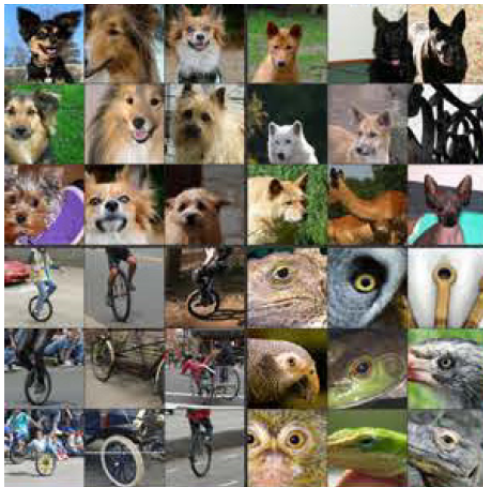


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der fünften Schicht besonders stark ansprechen.

## Classification Results (CLS)

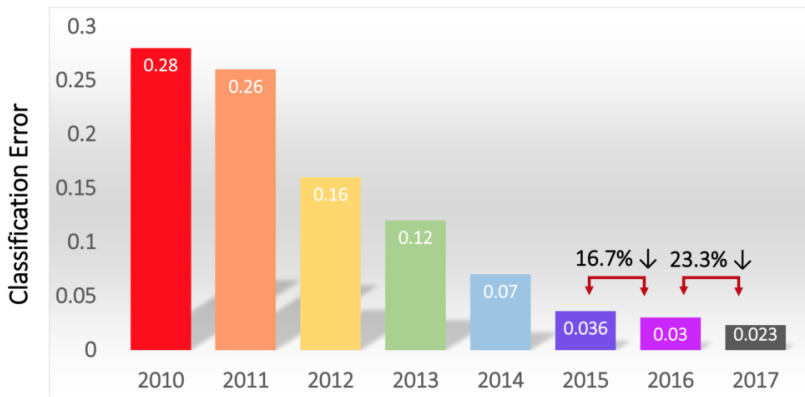


Image von imagenet.org.

Neuronale Netze mit vielen Schichten (deep networks) haben Durchbruch in Bilderkennung, Handschriftenerkennung, und Spracherkennung geschafft.

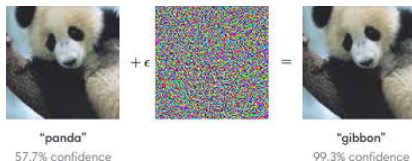
Je höher die Schicht, desto komplexere Merkmale werden erkannt; Merkmale auf einer Schicht sind Kombinationen von Merkmalen auf der vorherigen Schicht.

Training braucht sehr große Datensätze, die seit einigen Jahren durch soziale Netzwerke und Crowdsourcing zur Verfügung stehen.

Training ist aufwendig und dauert sehr lange, aber das ist auch bei Menschen so (meine Enkel waren 18 Monate alt als sie das Wort Elefant mit einem Bild eines Elefanten verknüpften).

Daher Durchbruch erst jetzt, obwohl Technik seit mehr als 25 Jahren bekannt.

- schlägt Weltranglistenersten 3:0
- trainiert durch Spiele gegen sich selbst
  - Anfangswissen = Regeln von Go
  - nach 40 Tagen Weltmeisterniveau
- hat Muster und Spielzüge gefunden, die Menschen bisher verborgen blieben



- Mangelnde Robustheit.
- Kann nicht: Rotation der Bilder, Skalierung, ...
- Ergebnis nicht besser als Daten.
  - Unvollständige Trainingsdaten, z.B. nur Affen und Menschen mit weißer Hautfarbe.
  - Gezielte Irreführung: Twitter-Nutzer machen IBM-Chatbot Tay zur Rassistin.
  - Vorurteile werden festgeschrieben.
- Fragwürdige Anwendungen: Vorhersage von Rückfällen bei Straftätern.
- Schwarzer Kasten (black box): Entscheidungen sind nicht nachvollziehbar.