Maschinelles Lernen: Neuronale Netze

Ideen der Informatik

Kurt Mehlhorn

max planck institut informatik

SIC Saarland Informatics Campus

Juni 2021

Übersicht

- Stand der Kunst: Bilder verstehen
- Was ist ein Bild in Rohform?
- Biologische Inspiration: Das menschliche Sehsystem
- Künstliche neuronale Netze
- Trainieren von neuronalen Netzwerken
  - Prinzip
  - Buchstaben: Stand der Kunst in 1986
- Bilderkennung durch Deep Convolutional Networks
- Deep Neural Networks, 86 Mio Google Treffer am 26. 5. 2021; auch Spracherkennung, maschinelle Übersetzung, Spiele lernen, usw.
- Schwächen und Gefahren

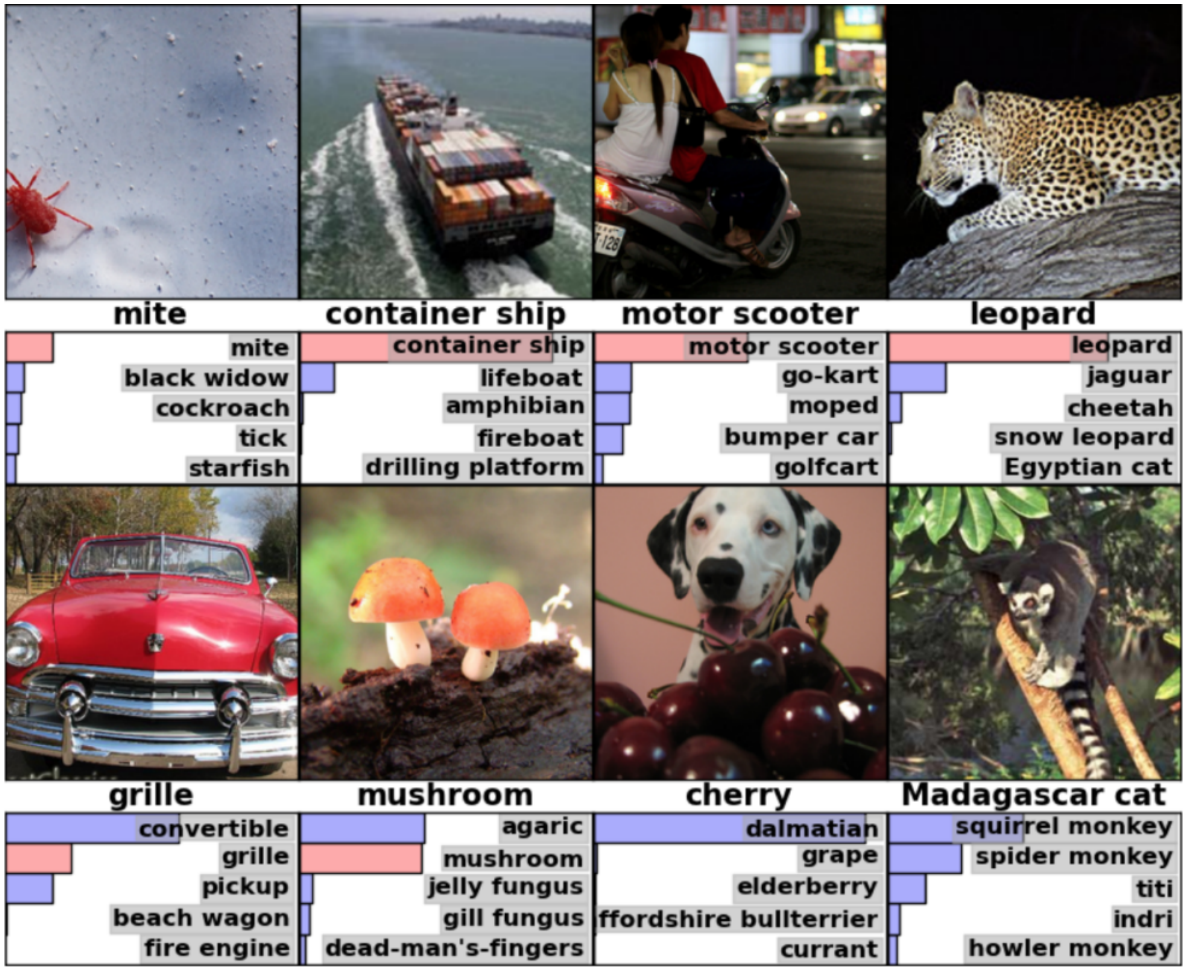
Turing-Preis 2018

Geoffrey Hinton

Yann Le Cun

Yoshua Bengio

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



Training: 1.2 Mio Bilder, eingeteilt in 1000 Klassen.

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

Hinweis: Leistung ist höchst beeindruckend, aber Bilder zeigen meist nur wenige Objekte.

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.

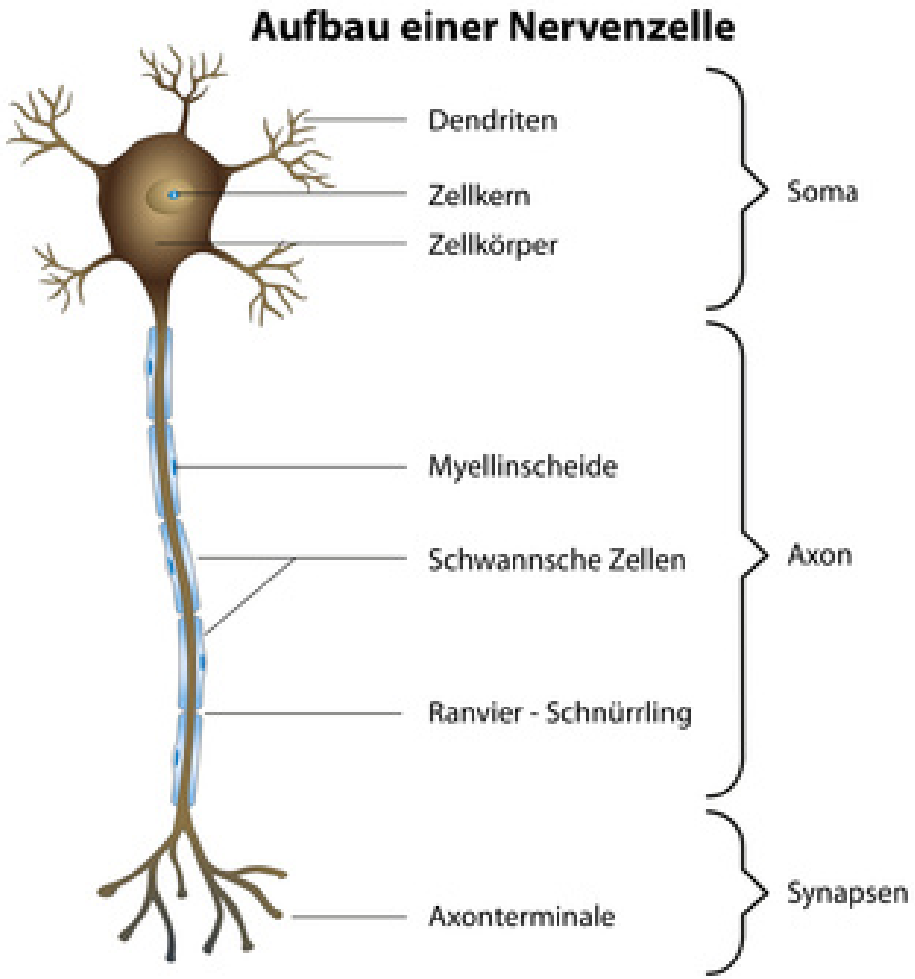
  1920 x 1080 Pixel bei HD-TV.

- Für jedes Pixel die Sättigung in den drei Grundfarben:

  rot, grün, blau $\in \{ 0, \ldots, 2^{16} - 1 \}$.

- Pixel sind so klein, dass das menschliche Auge sie nicht auflösen kann.

Biologische Inspiration I



Dendriten = Input.

Axonterminale = Output.

Zelle feuert, wenn Gesamterregung einen Schwellwert übersteigt.

Inputs können auch hemmend sein.

Biologische Inspiration II

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

Künstliches Neuron

- Neuron hat k eingehende Kanten.
- Eingabewerte x_1 bis x_k in 0 bis 1.
- $k + 1$ Gewichte (Parameter)

$$w_0, w_1, \ldots, w_k.$$

- w_0 heißt Grundgewicht (Bias); w_i ist die Wichtung von x_i.
- Ausgabe $= g(w_0 + w_1 x_1 + \ldots w_k x_k)$.
- $g =$ Sigmoid Funktion.
- Sigmoid Funktion ist differenzierbare Approximation einer Stufe von 0 nach 1 an der Stelle 0.

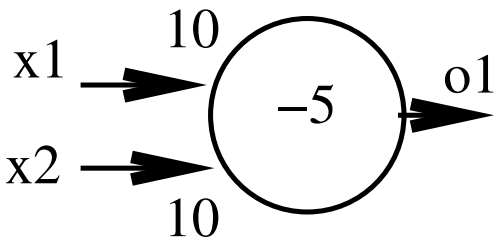
Sigmoid Funktion

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

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

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

Realisierung von Oder-Funktion und . . .

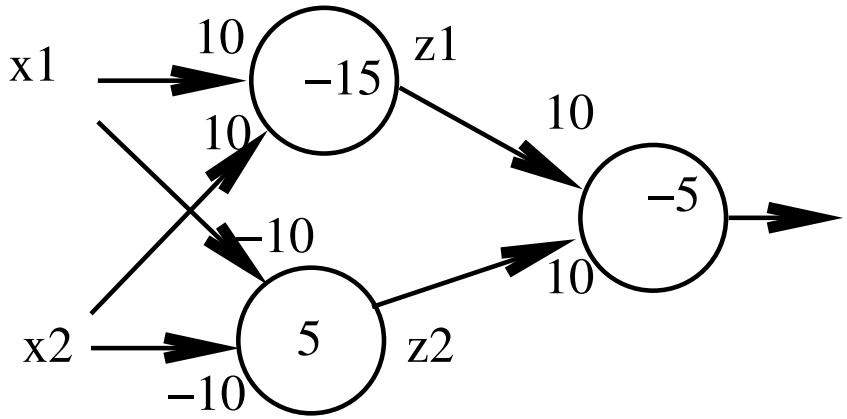


$$o_1 = g(-5 + 10x_1 + 10x_2) \approx x_1 \vee x_2$$
$$o_2 = g(15 - 10x_1 - 10x_2) \approx$$

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						

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

Realisierung einer Funktion durch ein Netz

Wir möchten eine gegebene Funktion mit einem Netz realisieren. Welche Struktur wählen wir und wie müssen wir die Parameter einstellen?

Realisierung einer Funktion durch ein Netz

Wir möchten eine gegebene Funktion mit einem Netz realisieren. Welche Struktur wählen wir und wie müssen wir die Parameter einstellen?

Im Prinzip ist das möglich.

Jede Funktion f mit n Argumenten in $[0, 1]$ und einem Ergebnis in $[0, 1]$ (jede Funktion $f : [0, 1]^m \to [0, 1]$) kann man durch ein künstliches neuronales Netz h beliebig gut approximieren.

Gut approximieren bedeutet: Für alle x, $|f(x) - h(x)| \leq \varepsilon$.

Beliebig gut bedeutet: Aussage gilt für alle positiven ε.

Das ist zunächst nur eine Existenzaussage. Sie besagt nicht, dass es einen Algorithmus gibt, die Struktur des Netzes und die Gewichte zu finden, geschweige denn einen effizienten.

Realisierung einer Funktion durch ein Netz

Wir möchten eine gegebene Funktion mit einem Netz realisieren. Welche Struktur wählen wir und wie müssen wir die Parameter einstellen?

Wahl der Struktur: Mehr dazu später.

Wahl der Gewichte bei gegebener Struktur

Die zu realisierende Funktion ist durch Beispiele gegeben. Jedes Trainingsbeispiel (für ein Netz mit n Eingaben und einem Ausgang) besteht aus einem Eingabevektor $x = (x_1, \ldots, x_n)$ und einer Ausgabe y.

Wir haben N Trainingsbeispiele $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$.

Gesucht: Einstellung der Parameter (Gewichte) des Netzes, so dass das Netz bei Eingabe $x^{(i)}$ als Ausgabe eine gute Approximation von $y^{(i)}$ liefert für $i = 1, 2, \ldots, N$.

Die Suche nach den Parametern heißt Training.

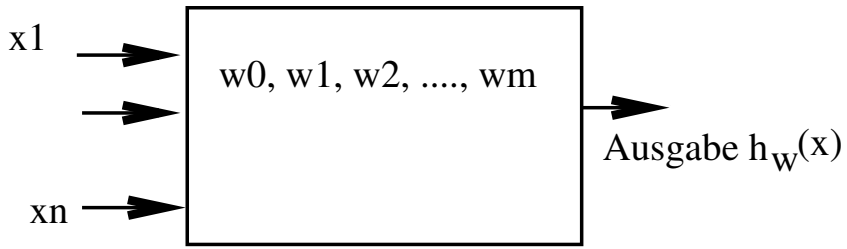
Präzisierung der Aufgabe I

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

Wir haben N Trainingsbeispiele $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$.

w = Vektor aller Parameter $w_0, \ldots, w_j, \ldots, w_m$.

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



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

Was heißt gut übereinstimmen?
Wie findet man Parametersatz?

Präzisierung der Aufgabe II

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.

Vorgehensweise: Gradientenabstiegs-Verfahren zur numerischen Bestimmung eines (lokalen) Minimums einer Funktion (in mehreren Variablen).

Lösung der Aufgabe I

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 Parameter w_j: Was passiert mit dem Fehler, wenn wir w_j geringfügig ändern? Geht er hoch oder nimmt er ab?

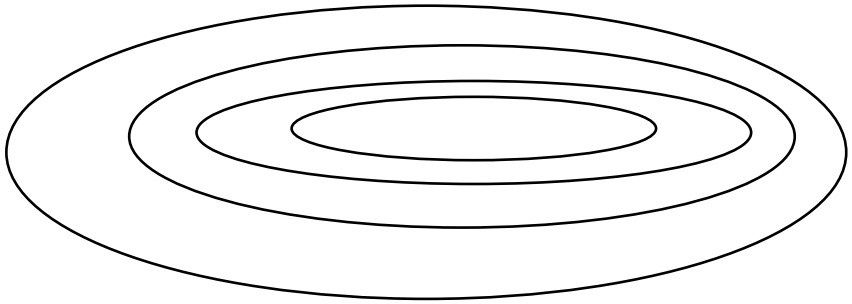
Dann ändern wir die w_j geringfügig, so dass der Gesamtfehler kleiner wird.

Das machen wir solange, bis wir zufrieden sind.

(Partielle) Ableitung nach w_j ist das richtige mathematische Konzept.

Partielle Ableitung $\partial E / \partial w_j$ von E nach w_j = relative Änderung von E bei kleiner Änderung von w_j. Betrachte E als Funktion nur in der Variablen w_j. Dann leite ab wie gelernt.

Gradientenabstieg zum Minimieren einer Funktion



Höhenlinien einer Funktion

Gradientenabstieg: Mache immer wieder einen kleinen Schritt senkrecht zur Höhenlinie. Wähle die Richtung nach unten.

Richtung senkrecht zur Höhenlinie heißt Gradient der Funktion. Gradient ist der Vektor aller partiellen Ableitungen $\partial E/\partial w_j$.

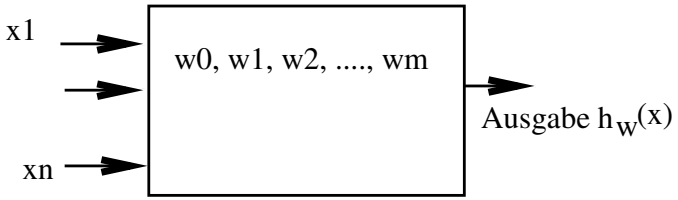
Gradientenabstiegs-Verfahren ist dann: mache einen kleinen Schritt (Schrittweite h) in Richtung des (negativen) Gradienten

$$w_j^{neu} = w_j^{alt} - h \cdot \frac{\partial E}{\partial w_j}(w^{alt}) \quad j = 1, \ldots, m.$$

Der Trainingsalgorithmus, Rummelhart/Hinton '86

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

$$w_j^{(neu)} = w_j - h \frac{\partial E}{\partial w_j}(w).$$

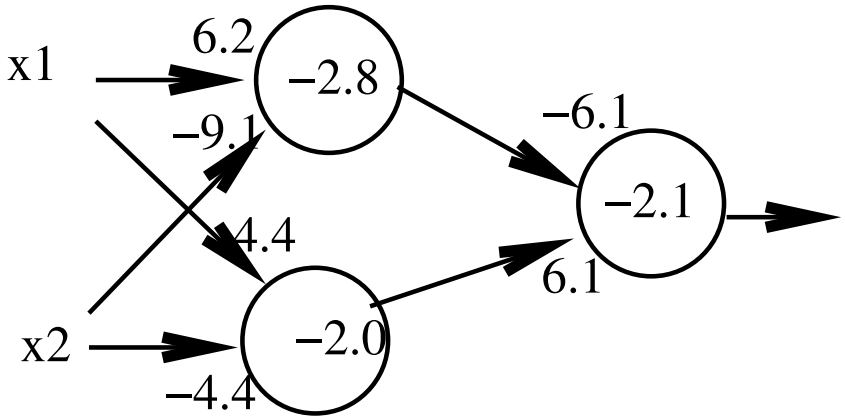
Bemerkung: Es gibt einen einfachen Alg (Back Propagation), um bei einem neuronalen Netz die partiellen Ableitungen nach den Parametern zu bestimmen.

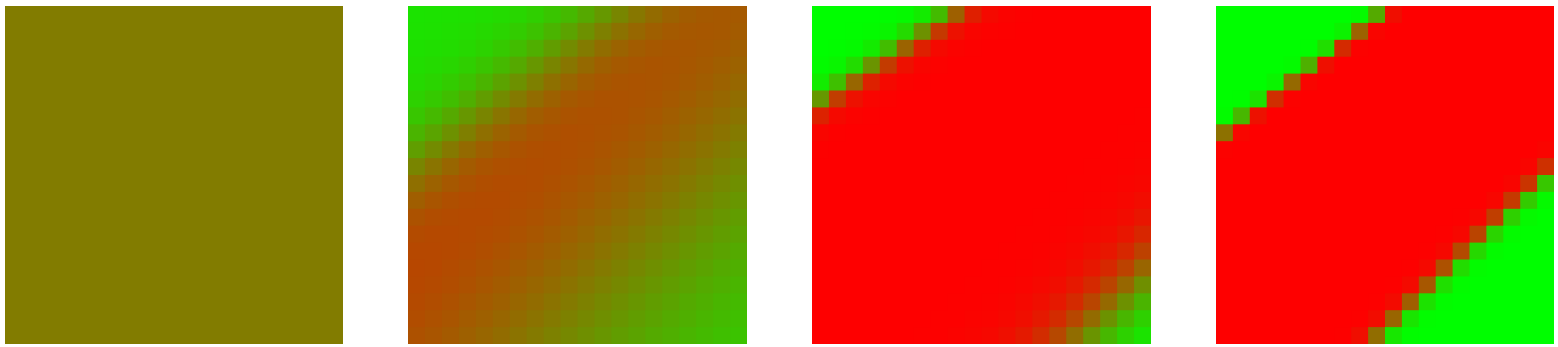
Beispiel für das Ergebnis eines Trainings

16 Trainingsbeispiele:
x_1 und x_2 in $\{0, 0.1, 0.9, 1\}$.

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

50000 Iterationen lieferten folgende Parameter:





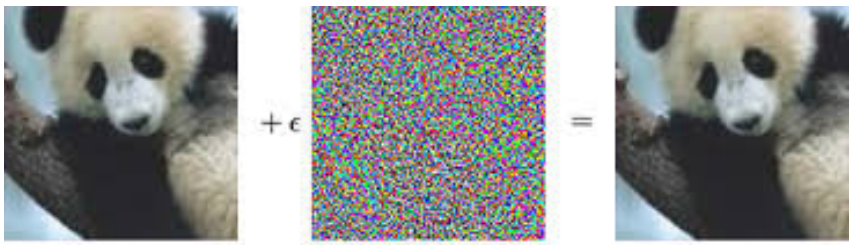
0 iterations 12500 25000 50000

green = 0, red = 1.

Verhalten auf Eingaben, die nicht trainiert wurden.

Hoffnung: Auch Eingaben, die Trainingsdaten ähneln, werden richtig klassifiziert. Man sagt, das Netz verallgemeinert.

- Im Beispiel haben wir nur mit Eingaben in der Nähe der Ecken trainiert. Das Netz ist sich vollkommen sicher über den Funktionswert in der Mitte des Quadrats. Das ist kühn.
- Ähneln als Bild $\neq$ ähneln im Pixelraum $\mathbb{R}^{\text{Anzahl der Pixel}}$.



- 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. Im $\mathbb{R}^{1000}$ liegt über 99,99% des Volumens in den äußeren 1% einer Kugel.

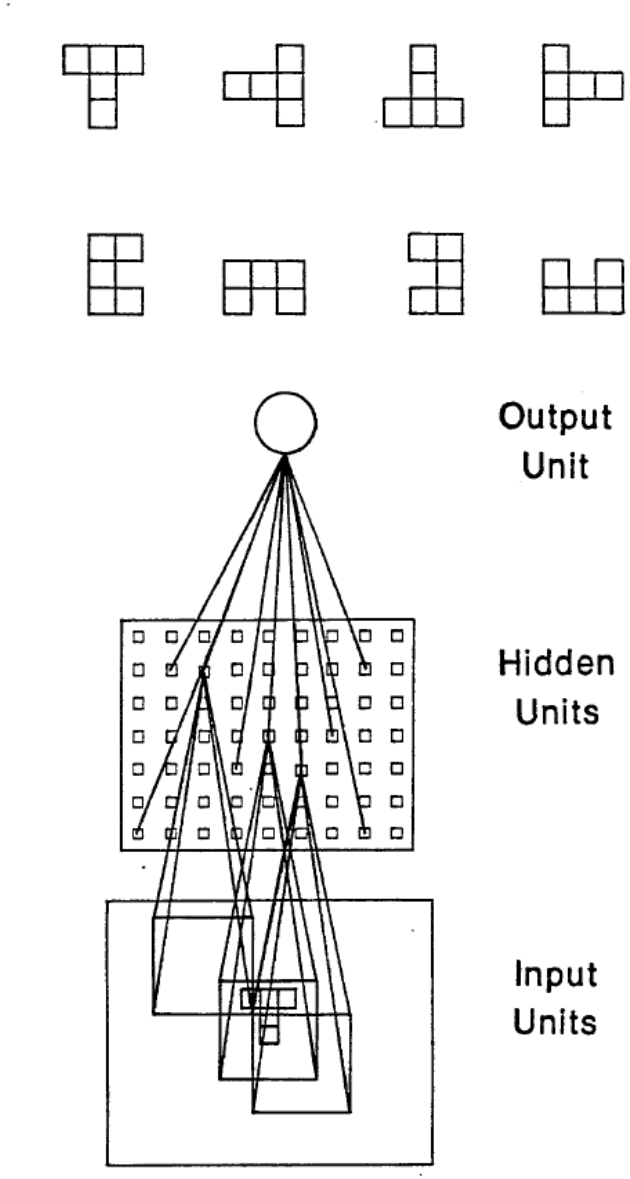
Unterscheide T und C (Convolutional Networks)

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.

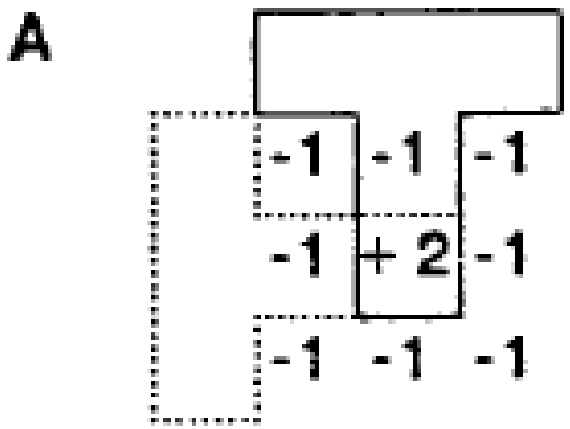
einfach: alle Neuronen der ersten Schicht schauen sich eine 3 x 3 Matrix an und sind identisch, d.h. haben die gleichen Parameter.
Ausgabeneuron hängt von allen Neuronen der ersten Schicht ab.

Anzahl der Parameter $= 9 + 1 + (25 - 2)^2 + 1$.



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

Eine Lösung (Das Bild zeigt die Gewichte)



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

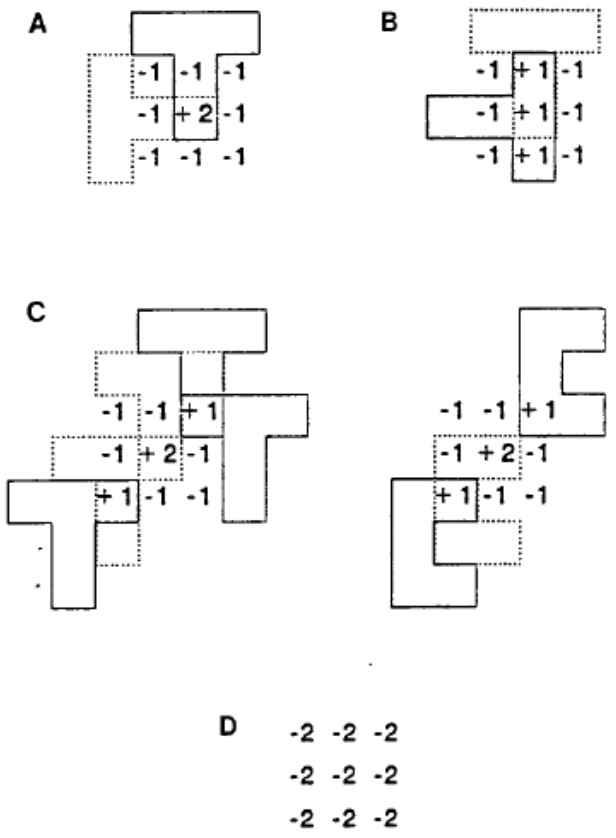
Bei einem C erzeugt der Filter immer einen Wert kleiner gleich 0.

Neuron der ersten Schicht =

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

Ausgabeneuron = Oder aller Neuronen der ersten Schicht.

Lösungen durch Trainieren (Rumelhart/Hinton/Williams, 86)



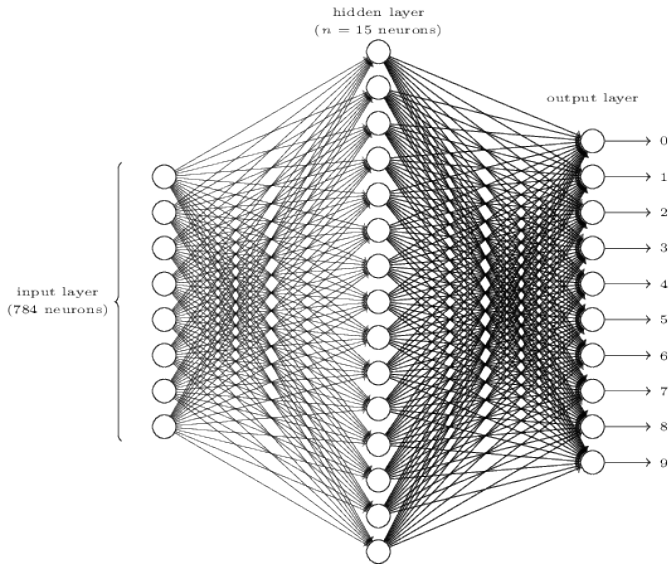
Training fand vier verschiedene Lösungen.

Aufgabe: Finden Sie heraus, wie B und C funktionieren.

D ist besonders interessant (Die Autoren kannten diese Lösung **nicht**): es funktioniert, weil
ein C 20 Rezeptoren überlappt und
ein T 21 Rezeptoren.

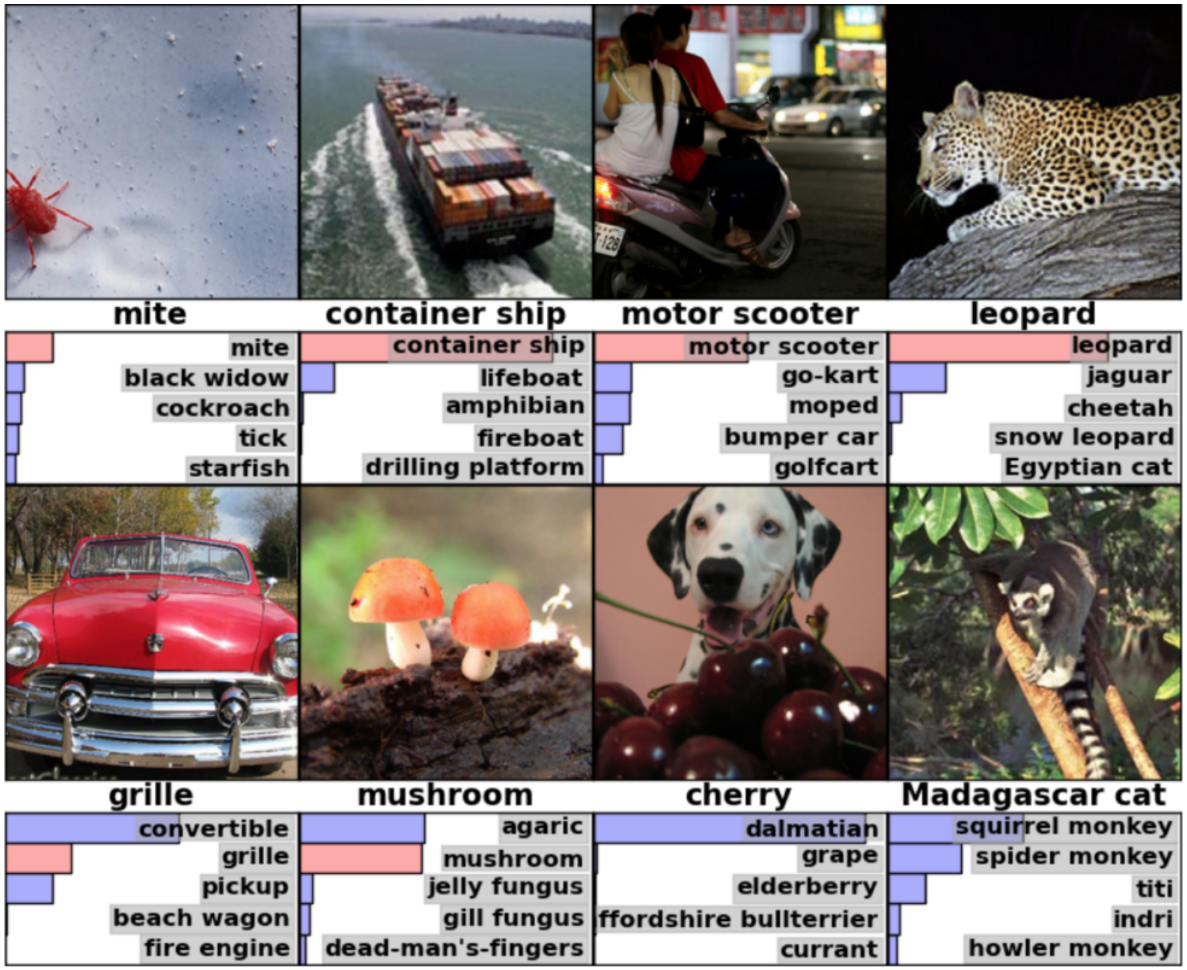
0	0	0	0	0		0	0	0	0	0
0	1	1	0	0		0	1	1	1	0
0	1	0	0	0		0	0	1	0	0
0	1	1	0	0		0	0	1	0	0
0	0	0	0	0		0	0	0	0	0

Ziffernerkennung (Michael Nielsen)



- MNIST Datensatz: 60,000 Bilder 28 x 28 Pixel; für jedes Pixel: Helligkeit, 0 oder 1.
- 10 Ausgabeneuronen, 15 Neuronen in der Mittelschicht, 784 = 28 x 28 Eingabewerte.
- Jedes Neuron der Mittelschicht hat 784 Eingaben. Jedes Ausgabeneuron hat 15 Eingaben.
- Anzahl der Parameter = 15 x 785 + 10 x 16 = 11935.
- Fehlerrate: unter 3 Prozent.
- Zahlen sind schön zentriert.

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

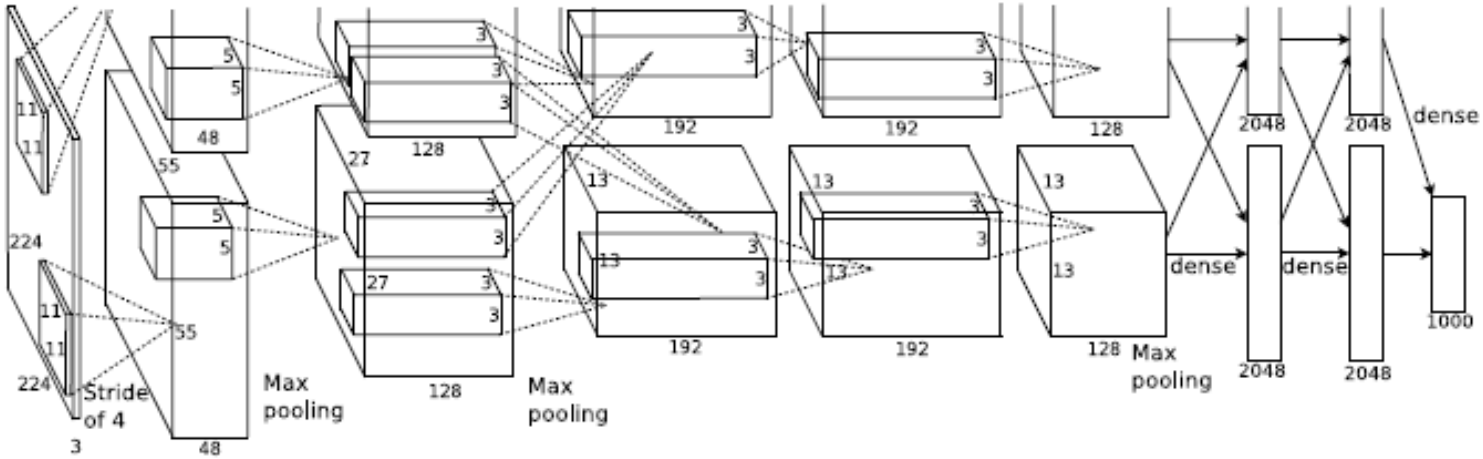


Training: 1.2 Mio Bilder, eingeteilt in 1000 Klassen.

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

Hinweis: Leistung ist höchst beeindruckend, aber Bilder zeigen meist nur wenige Objekt.

Stand der Kunst (Krishevsky et al., 2012)



- 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, Ausgabeschicht: 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.

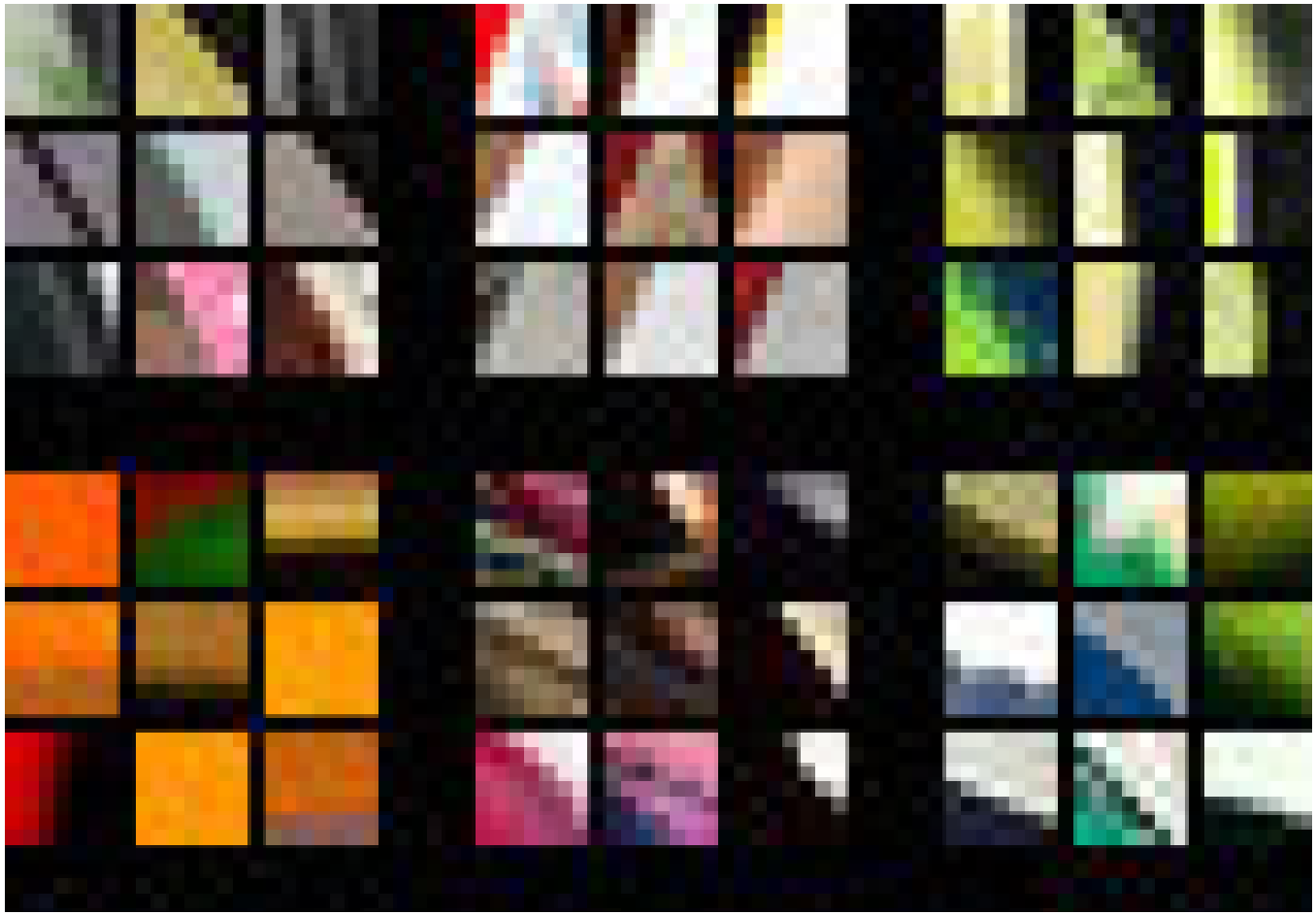


Abbildung zeigt Eingaben, bei denen 6 ausgewählte Neuronen der ersten Schicht besonders stark ansprechen.

Wie funktioniert das?

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

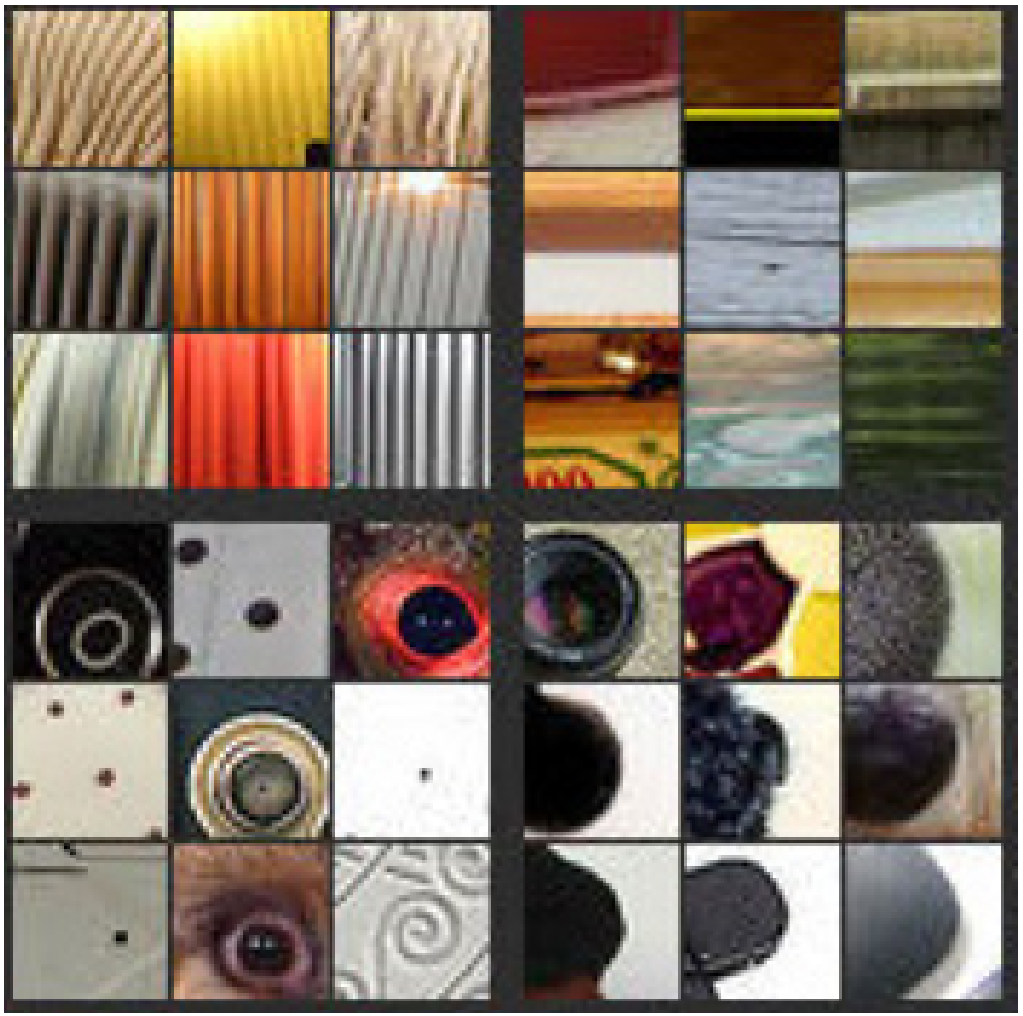


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der zweiten Schicht besonders stark ansprechen.

Wie funktioniert das?

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

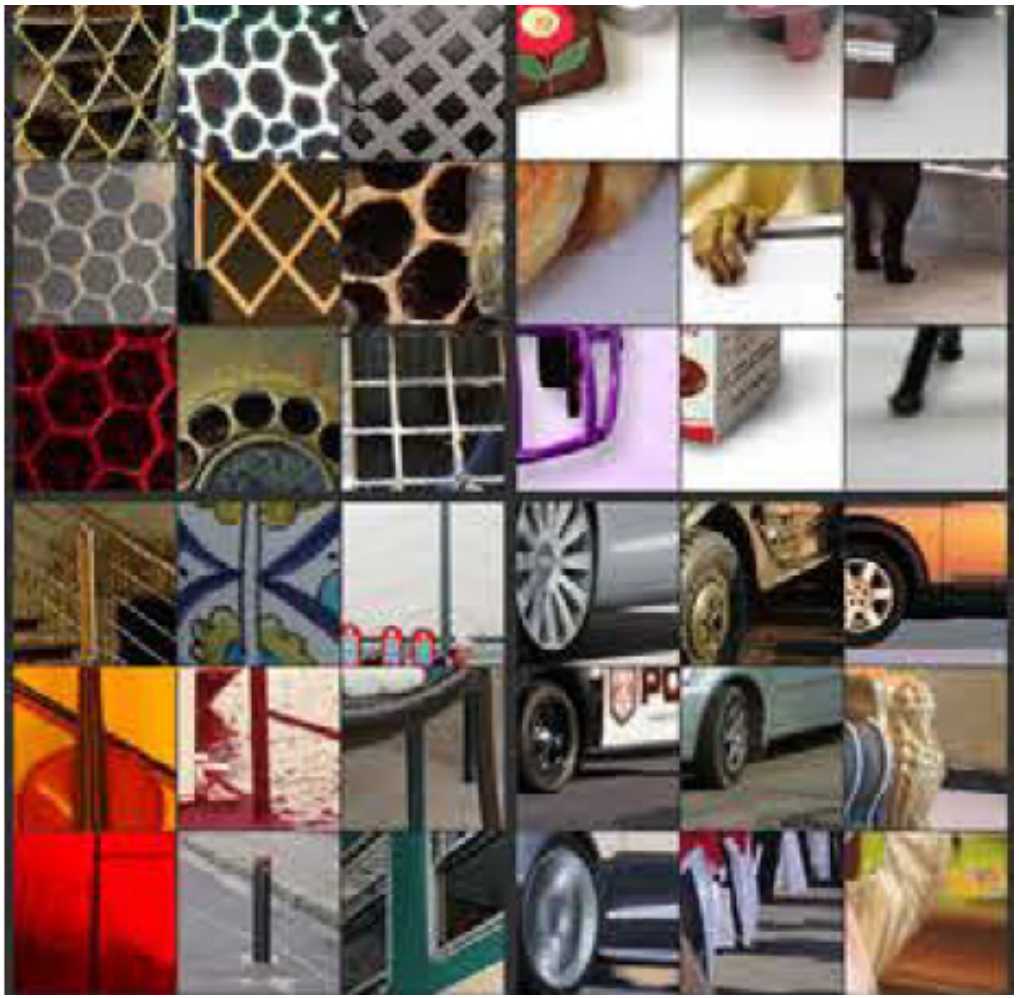


Abbildung zeigt Eingaben, bei denen 4 ausgewählte Neuronen der dritten Schicht besonders stark ansprechen.

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.

Fortschritt in den letzten Jahren

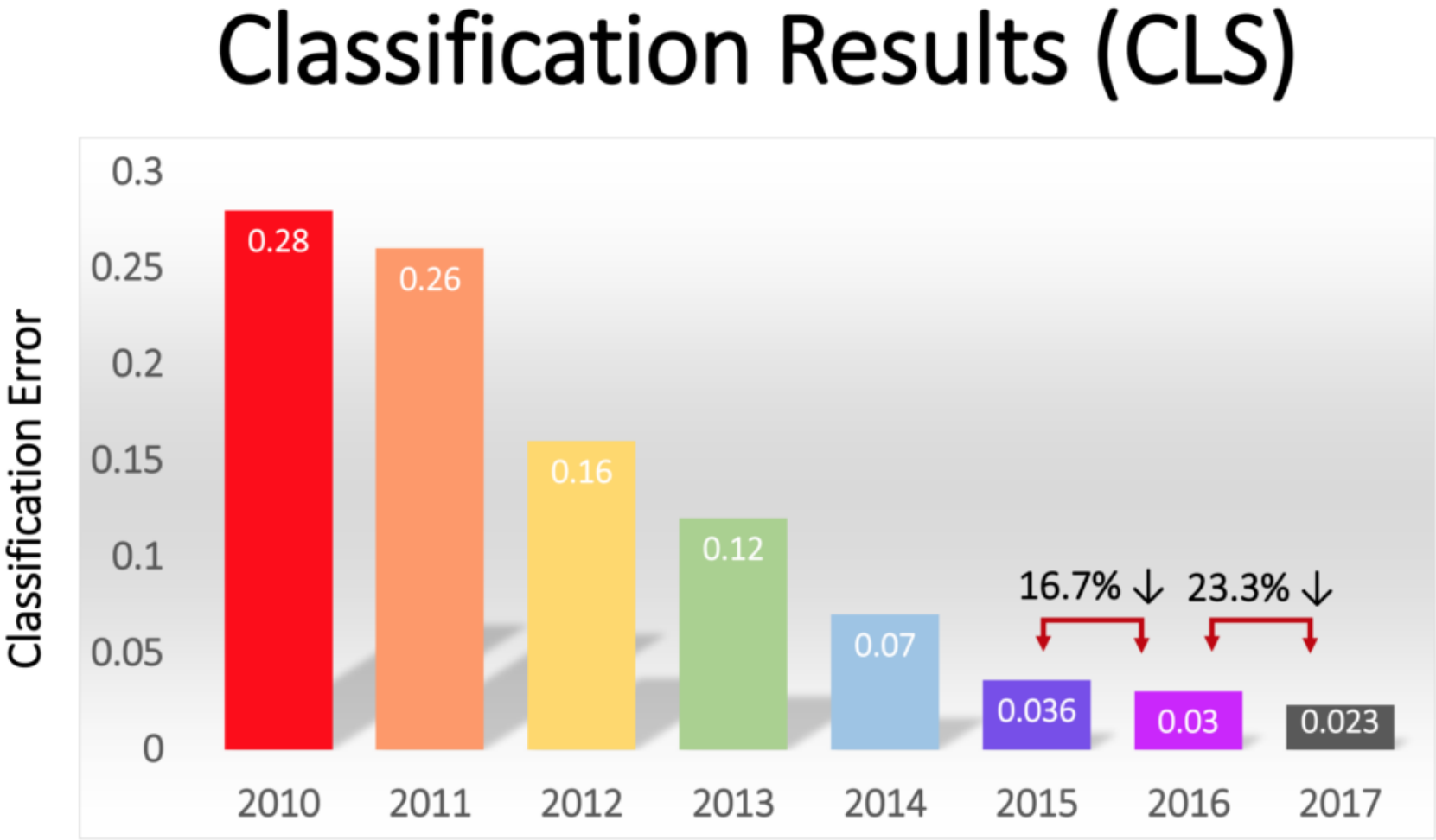


Image von imagenet.org.

Zusammenfassung

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

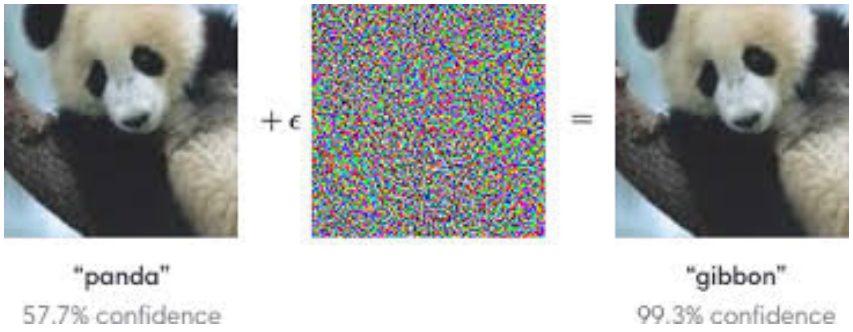
Je höher die Schicht, desto komplexere Merkmale werden erkannt; Merkmale auf einer Schicht sind Kombinationen von Merkmalen auf der vorherigen Schicht.

Training braucht sehr große Datensätze, die seit einigen Jahren durch soziale Netzwerke und Crowdsourcing zur Verfügung stehen, und sehr leistungsfähige Rechner (graphic processing units (gpu)), die es wegen der Computerspiele gibt.

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).

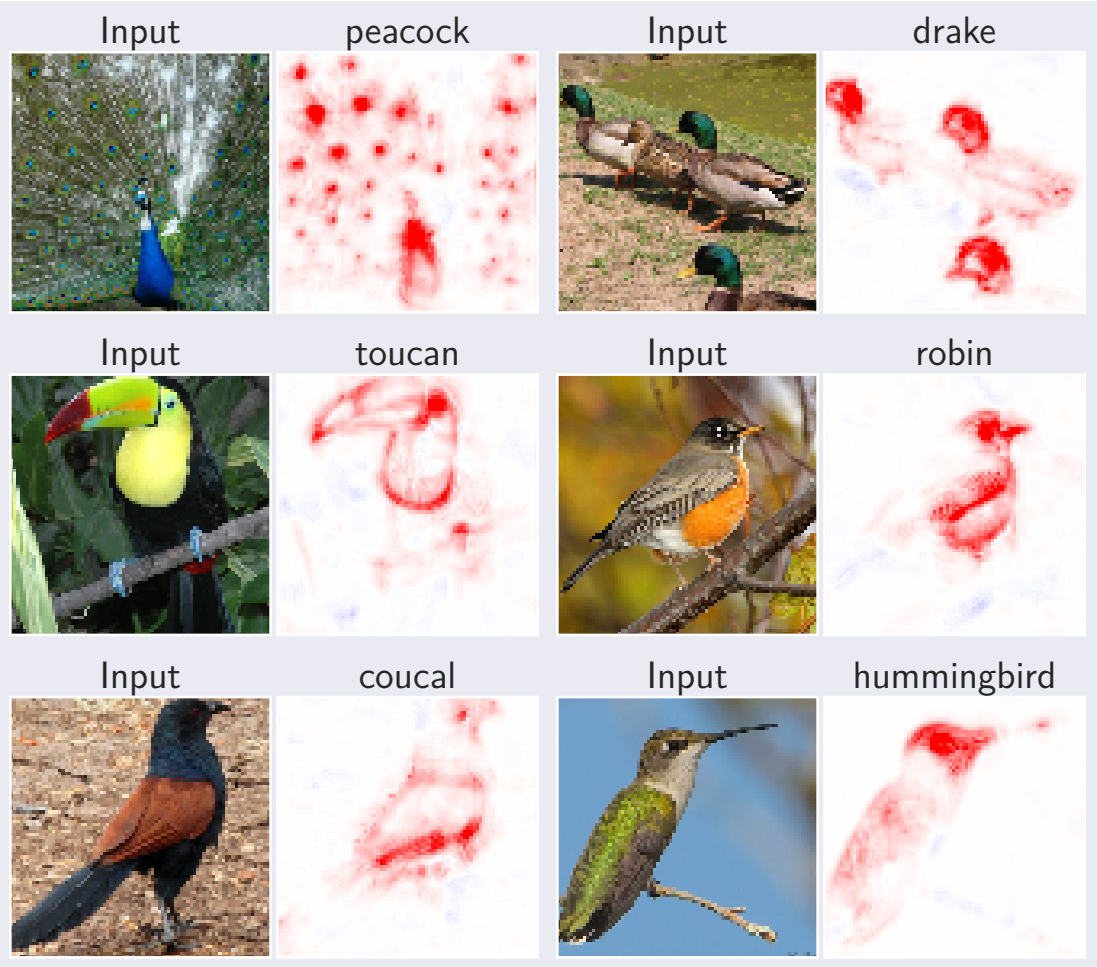
Daher Durchbruch erst jetzt, obwohl Technik seit mehr als 25 Jahren bekannt.

Schwächen und Gefahren



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

Partielle Nachvollziehbarkeit der Entscheidungen



Rote Pixel: wichtig für die Entscheidung

Blaue Pixel: unwichtig

Weiße Pixel: neutral

M. Böhle, M. Fritz, B. Schiele (MPI-INF und CISPA, 2021)