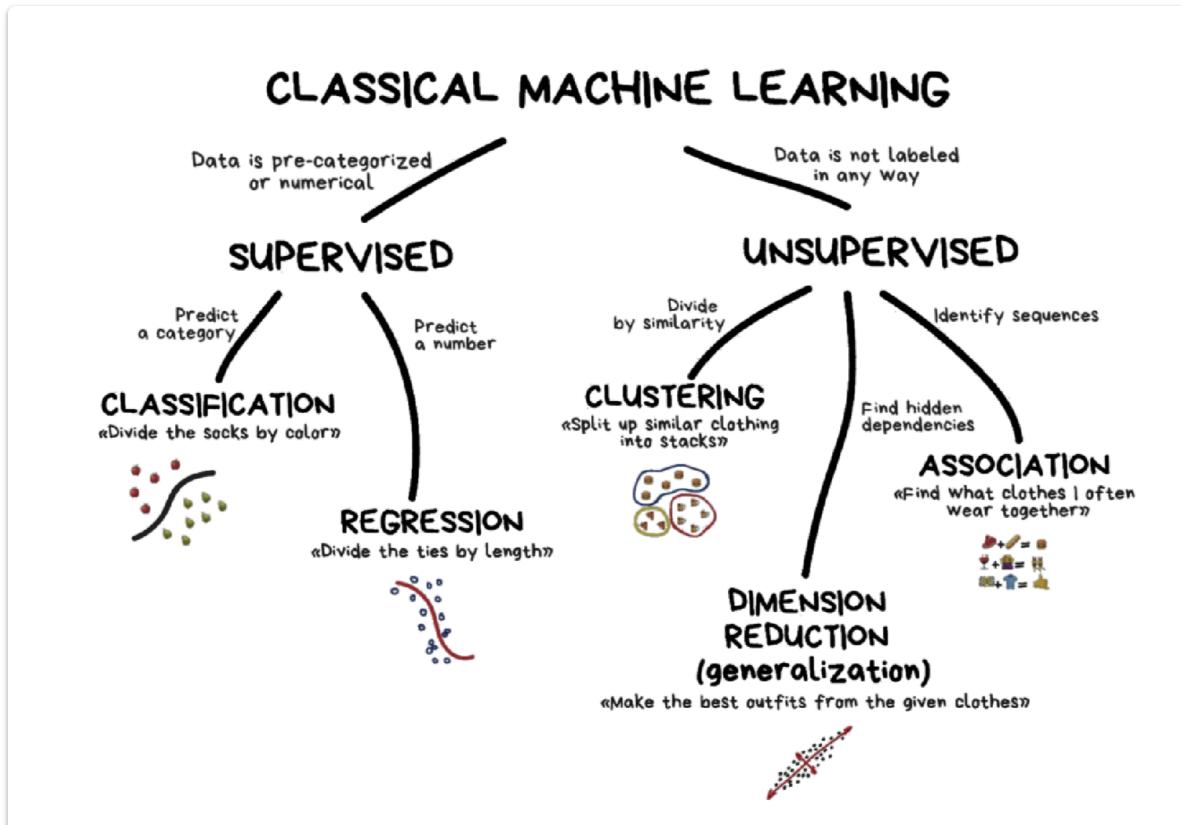


# 11. Deep Learning for Computer Vision

## Maschinelles Lernen

- **Definition:** Maschinelles Lernen (ML) ist ein Teilgebiet der künstlichen Intelligenz (KI).
- **Geschichte:** Etabliertes Forschungsgebiet, geprägt durch rasante technologische Fortschritte seit Jahrzehnten.
- **Begriff der KI:** Erstmals 1955 von Minsky, McCarthy, Newell und Simon definiert: Maschinen, die sich verhalten, als würden sie über eine Art menschliche Intelligenz verfügen.
- **Allgemeines Prinzip des ML:** ML umfasst Lernprozesse, um Zusammenhänge in bestehenden Datensätzen zu erkennen und darauf basierend Vorhersagen zu treffen.
- **Wesentliche Bedeutung:** Modelle ziehen aufgrund von selbstlernenden Algorithmen und existierenden Daten zukunftsrelevante Rückschlüsse, ohne explizit programmiert zu werden.
- **Grundlage:** Lernprozesse dienen als Dateneingabe, die durch eine vordefinierte Reihe an Attributen charakterisiert ist.
- **Kategorien des Maschinellen Lernens:**
  - Überwachtes Lernen (Supervised Learning)
  - Unüberwachtes Lernen (Unsupervised Learning)
  - Bestärkendes Lernen (Reinforcement Learning)



- **Überwachtes Lernen (Supervised Learning):**

- Daten sind vor-kategorisiert oder numerisch (Data is pre-categorized or numerical).
- Ziel: Vorhersage einer Kategorie (Klassifikation) oder einer Zahl (Regression).
- **Klassifikation:** Teilt die Daten nach Farben (Divide the socks by colors).
  - Beispiel: Vorhersage einer Kategorie.
- **Regression:** Passt eine Linie durch Datenpunkte (Predict a number).
  - Beispiel: Vorhersage eines Preises.

- **Unüberwachtes Lernen (Unsupervised Learning):**

- Daten sind in keiner Weise gelabelt (Data is not labeled in any way).
- Ziel: Struktur in den Daten finden.
- **Clustering:** Gruppert ähnliche Datenpunkte (Divide by similarity; spot similar clothing styles together).
  - Ziel: Gruppen ähnlicher Daten finden.
- **Assoziationsanalyse:** Findet häufige Abhängigkeiten (Find hidden dependencies; find what clothes I often wear together).
  - Ziel: Beziehungen zwischen Datenpunkten identifizieren.
- **Dimensionsreduktion (Generalisierung):** Reduziert die Anzahl der Merkmale (Create the best outfits from the given clothes).
  - Ziel: Wichtige Informationen extrahieren und Rauschen reduzieren.

## Überwachtes Lernen (Supervised Learning)

- **Zielvariable (dedizierte Ausgabewerte):** Wenn eine dedizierte Ausgabeveriable definiert ist, kann durch Supervised Learning ein Modell darauf trainiert werden, diese für bisher unbekannte Datensätze (Test Set) vorherzusagen.
- **Generelles Ziel:** Maximierung der Genauigkeit dieser Vorhersagen.
- **Anwendung:** Bei Datensätzen mit präzisen Ausgabewerten.
- **Lernprozess:** Algorithmus lernt die Beziehung zwischen Eingabewerten (Attributen) und den zugehörigen Ausgabewerten anhand des Trainingsdatensatzes.
- **Validierung:**
  - Der gesamte Datensatz wird in ein **Training Set** und ein **Test Set** aufgeteilt.
  - Der eigentliche Lernprozess zur Vorhersage der Zielvariable basiert auf dem **Training Set**.
  - Die Evaluation des trainierten Modells erfolgt mithilfe des **Testdatensatzes**.
  - Dadurch kann sichergestellt werden, dass die Bewertungsgrößen (Genauigkeit, Fehlerrate) anhand bisher unbekannter Daten bestimmt werden.

## Unüberwachtes Lernen (Unsupervised Learning)

- **Anwendung:** Vor allem dann, wenn ein Datensatz keine präzisen Ausgabewerte aufweist.
- **Ziel:** Durch Anwendung von Unsupervised Learning basierend auf den Eingabedaten bisher unbekannte Muster und Zusammenhänge abzuleiten.

## Bestärkendes Lernen (Reinforcement Learning)

- **Lernprozess:** Basiert auf Formen der Belohnung und Bestrafung.
- **Ziel:** Nutzen zu maximieren.
- **Hinweis:** Unsupervised und Reinforcement Learning werden im Folgenden nicht weiter behandelt.

## Ziel des überwachten Lernverfahrens

- Ausgabewerte anhand der vorhandenen Eingabewerte (den Attributen) mit möglichst hoher Genauigkeit vorherzusagen.

### Traditional Programming:



### Machine Learning:



## Klassifikation

- **Häufigste Anwendung in der Computer Vision:** Klassifikation anhand von Merkmalen (Features).
- **Merkmale und Klassenzuordnung:** Im Vorhinein bekannt, die Daten sind "gelabelt".
- **Beispiel (Obstbauer):**
  - Geerntete Äpfel sollen automatisch in die Handelsklassen A und B eingeteilt werden.
  - Sortieranlage misst für jeden Apfel zwei Merkmale (Features): Größe und Farbe.
  - Aufgabe: Entscheidung, zu welcher der beiden Klassen der Apfel gehört.
  - **Typische Klassifikationsaufgabe.**
  -
- **Klassifikatoren (Classifier):** Systeme, welche in der Lage sind, Merkmalsvektoren in eine endliche Anzahl von Klassen einzuteilen.
- **Trainingsdaten-Erstellung:**
  - Zur Einstellung der Maschine werden Äpfel von einem Fachmann händisch verlesen (klassifiziert).
  - Die Messwerte (Größe und Farbe) werden zusammen mit der Klasse in einer Tabelle eingetragen.
  - Größe: Durchmesser in Zentimetern.
  - Farbe: Zahlenwert zwischen 0 (grün) und 1 (rot).

- **Aufgabe beim maschinellen Lernen:** Aus den gesammelten klassifizierten Daten eine Funktion zu generieren, die für neue Äpfel aus den beiden Features (Größe und Farbe) den Wert der Klasse (A oder B) berechnet.

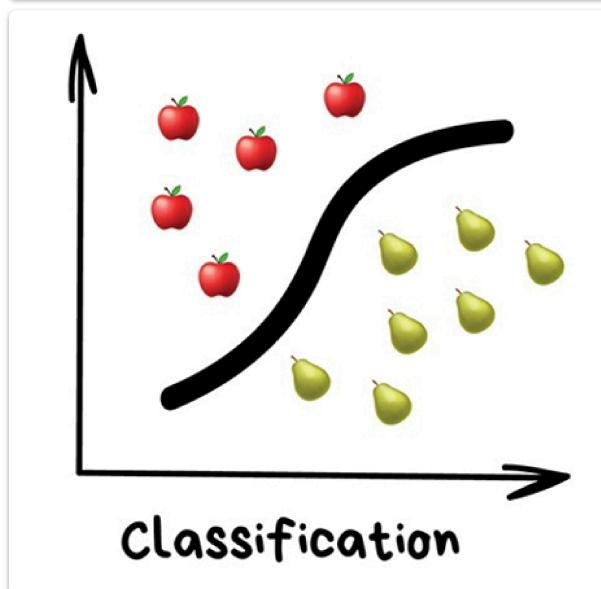
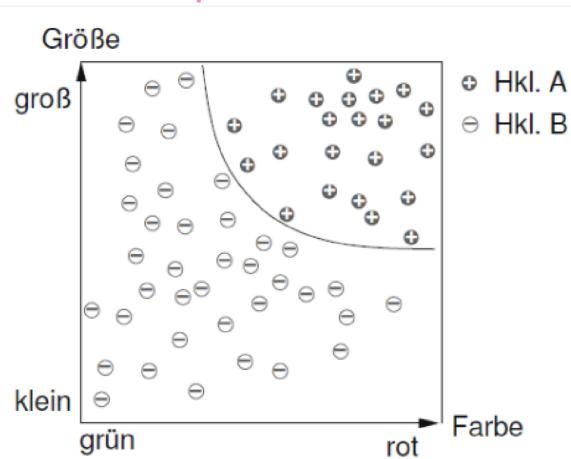
Größe [cm]	8	8	6	3	...
Farbe	0.1	0.3	0.9	0.8	...
Handelsklasse	B	A	A	B	...

- **Abbildung 46 (Beispielhafte Daten):**

- Tabelle mit Spalten: Größe (cm), Farbe, Handelsklasse.
- Beispielhafte Datenpunkte zeigen verschiedene Kombinationen von Größe und Farbe mit der zugehörigen Handelsklasse (A oder B).

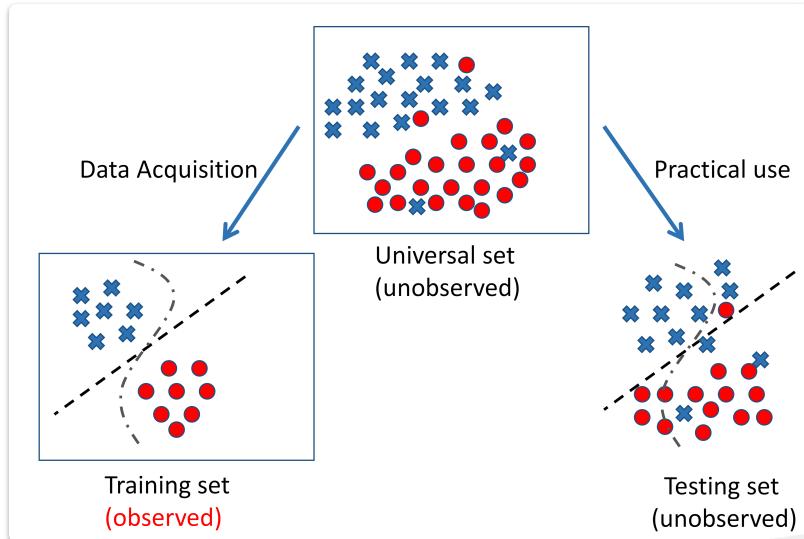
## Funktion zur Klassifikation

- Die in Abbildung 46 eingezeichnete Trennlinie repräsentiert eine Funktion, die die Klassifizierung vornimmt.
- **Klassenzuweisung:**
  - Äpfel mit Merkmalsvektor links unterhalb der Linie: Klasse B.
  - Alle anderen Äpfel: Klasse A.
- **Einfaches Beispiel:** Die Trennlinie ist in diesem Fall leicht zu finden.



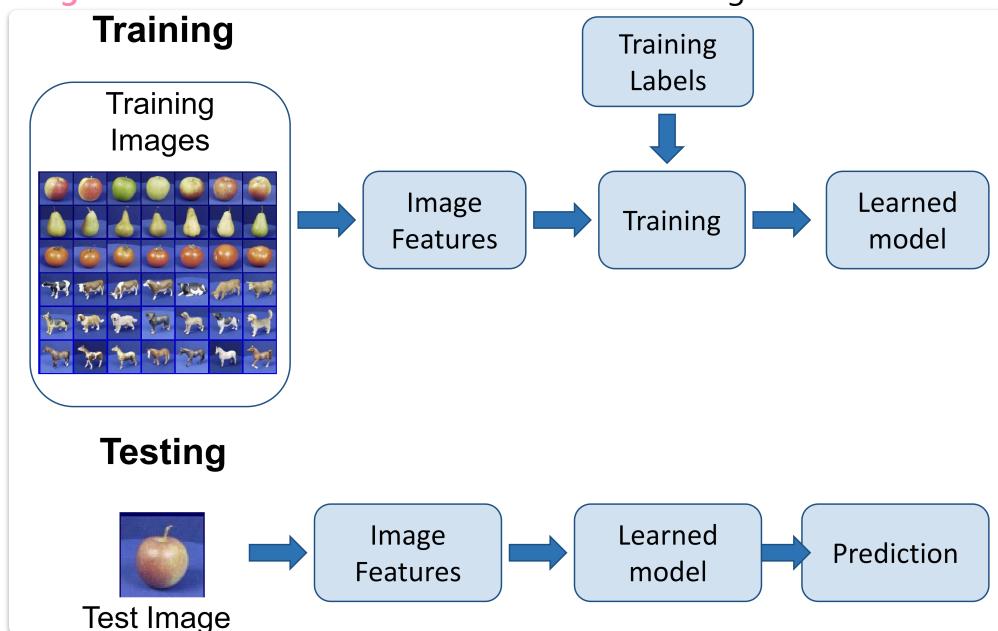
## Herausforderungen bei mehrdimensionalen Daten

- **Schwierigkeit bei vielen Merkmalen:** Die Aufgabe wird deutlich schwieriger und weniger anschaulich, wenn Objekte durch mehr als zwei Merkmale beschrieben werden.
- **Computer Vision:** Nutzt oft viele Merkmale, die aus Bilddaten abgeleitet werden.
- **n Merkmale:** Die Aufgabe besteht darin, im n-dimensionalen Merkmalsraum eine (n-1)-dimensionale Hyperfläche zu finden, welche die Klassen möglichst gut trennt.
- **"Gut" trennen:** Der relative Anteil falsch klassifizierter Objekte soll möglichst klein sein.



## Klassifikator als Abbildung

- Ein Klassifikator bildet einen Merkmalsvektor auf einen Klassenwert ab.
- **Feste Anzahl von Alternativen:** Meist eine kleine Anzahl möglicher Klassen.
- **Zielfunktion:** Diese Abbildung wird auch Zielfunktion genannt.
- **Aufgabe des Lernverfahrens:** Eine solche Abbildung zu lernen.

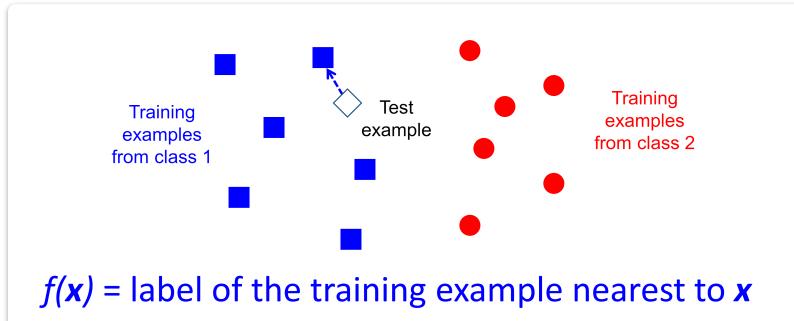


## Klassifikationsalgorithmen im Maschinellen Lernen

- **Vielfalt:** Es existiert eine Vielzahl von Klassifikationsalgorithmen im Bereich des maschinellen Lernens.
- **Überblick über geläufige überwachte Lernverfahren:**

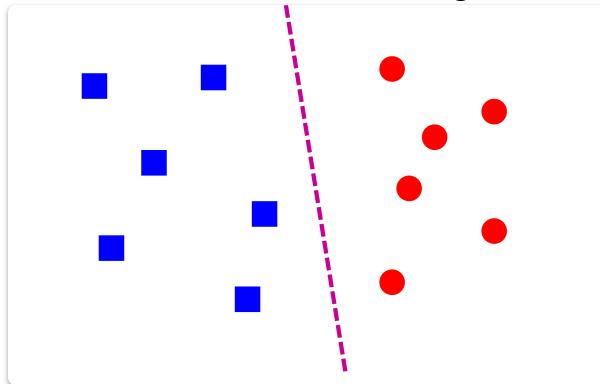
## Nearest Neighbor Algorithmus

- **Grundgedanke:** Ein Datenpunkt wird der Klasse zugeordnet, die durch den Mehrheitsentscheid seiner benachbarten Datenpunkte bestimmt wird.



## Lineare Gleichung

- Finden einer linearen Gleichung die die Klassen trennt



## Entscheidungsbäume (Decision Trees)

- **Klassenzuordnung:** Instanzen werden anhand ihres Pfades durch den Baum (Wurzelknoten und innere Knoten mit Entscheidungsregeln) einer Klasse zugewiesen.

## Random Forest Algorithmus

- **Erweiterung der Entscheidungsbäume.**
- **Klassenzuordnung:** Erfolgt durch den Mehrheitsbeschluss einer Vielzahl von unabhängigen Entscheidungsbäumen.

## Support Vector Machine (SVM)

- **Trennung der Klassen:** Verwendet eine Hyperebene, welche die Klassen mit dem möglichst größten Abstand voneinander trennt (Maximum-Margin-Klassifikator).

## Boosting-Verfahren (z.B. AdaBoost)

- **Ziel:** Steigerung der Gesamtleistung.
- **Funktionsweise:** Kombiniert eine Vielzahl von "schwachen" Klassifikatoren durch einen gewichteten Mehrheitsentscheid zu einem einzigen, stärkeren Klassifikator.

## Künstliche Neuronale Netze (NN)

- **Bedeutung:** Haben sich aufgrund der wachsenden Komplexität und Menge der Datensätze etabliert.
- **Weitere Behandlung:** Werden später noch genauer betrachtet.

## Generalisierung

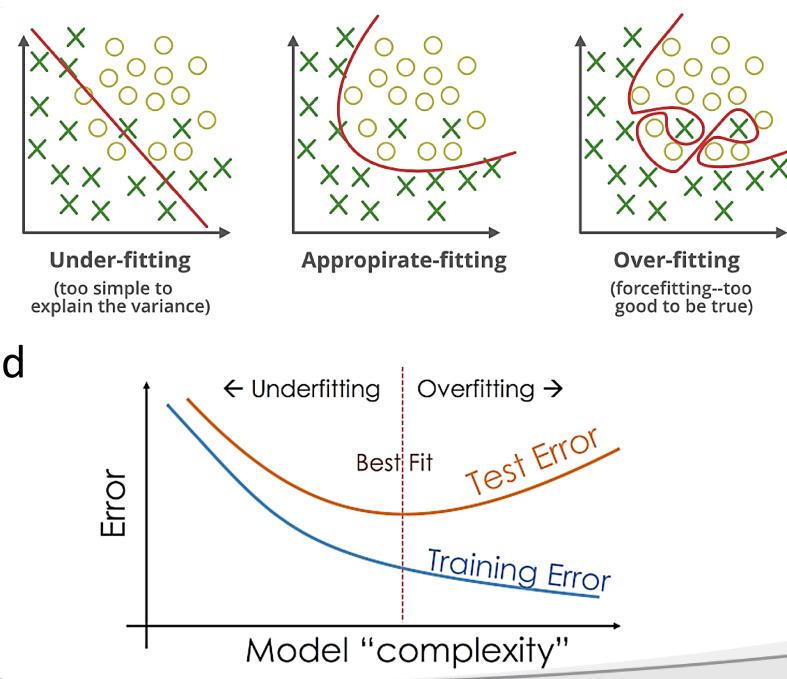


Training set (labels known)



Test set (labels unknown)

- **Überanpassung (Overfitting) - Auswirkungen:**
  - Das Modell lernt nicht die generalisierbaren Muster, sondern die spezifischen Details und das Rauschen der Trainingsdaten.
  - Führt zu einer hohen Genauigkeit auf den Trainingsdaten, aber einer geringen Genauigkeit auf neuen, ungesiehten Daten (Testdaten).
  - Die Leistung des Modells auf realen Anwendungen ist schlecht.



- **Generalisierung - Das Ziel:**

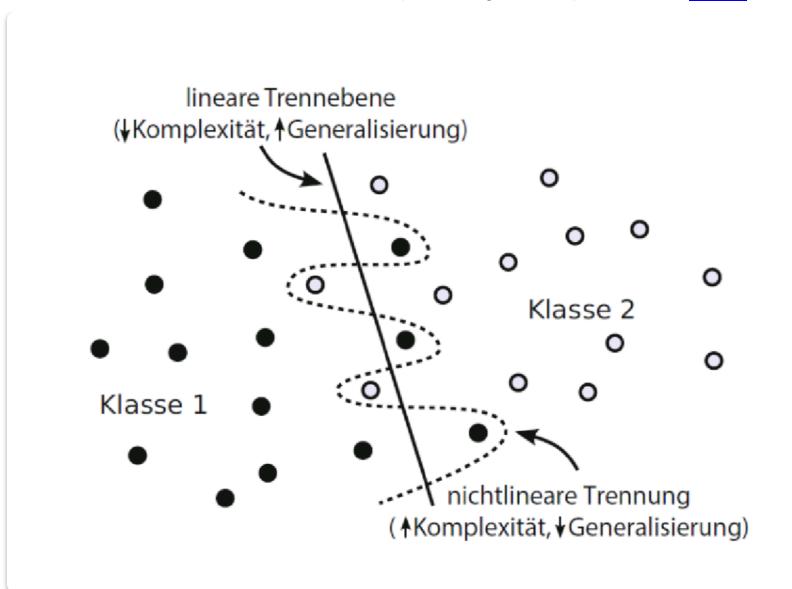
- Ein gutes Modell soll in der Lage sein, Muster zu erkennen, die in den Trainingsdaten vorhanden sind und diese Muster auch auf neue, unbekannte Daten anzuwenden.
- Eine hohe Generalisierungsfähigkeit ist das primäre Ziel beim Entwickeln von Machine-Learning-Modellen.

- **Modellkomplexität - Der Balanceakt:**

- **Zu einfache Modelle (Underfitting):** Können die zugrundeliegenden Zusammenhänge in den Daten nicht erfassen. Sowohl Trainings- als auch Testdaten werden schlecht vorhergesagt.
- **Zu komplexe Modelle (Overfitting):** Lernen die Trainingsdaten zu gut, inklusive Rauschen. Gute Leistung auf Trainingsdaten, schlechte Leistung auf Testdaten.
- **Die goldene Mitte:** Ein Modell mit der richtigen Komplexität, das die relevanten Muster erfasst, ohne sich zu stark an die Trainingsdaten anzupassen.

- **Praktische Implikationen:**

- Die Wahl der Modellarchitektur und der Hyperparameter beeinflusst die Modellkomplexität maßgeblich.
- Techniken wie Kreuzvalidierung (Cross-Validation) werden eingesetzt, um die Generalisierungsfähigkeit eines Modells auf unabhängigen Daten zu schätzen und Überanpassung zu erkennen.
- Regularisierungsmethoden können verwendet werden, um die Komplexität eines Modells zu reduzieren und die Generalisierung zu verbessern.



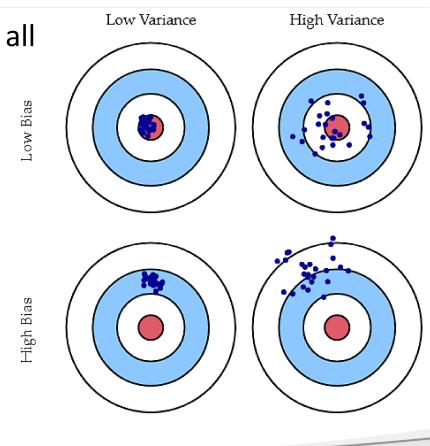
## Generalisierung - Bias-Variance Tradeoff

- **Bias (Verzerrung):**

- Maß für den Fehler aufgrund vereinfachter Annahmen im Modell.
- Hoher Bias bedeutet, dass das Modell die zugrundeliegenden Muster in den Daten nicht gut erfasst (Underfitting).
- Einfache Modelle haben tendenziell einen hohen Bias.
- Ein komplexes Modell hat einen niedrigen Bias, da es flexibler ist und sich besser an die Trainingsdaten anpassen kann.

• **Bias:** how much does the **average model** over all training sets **differ from the true model?**

• **Variance:** how much **models** estimated from different training sets **differ from each other**



- **Variance (Varianz):**

- Maß für die Sensitivität der Modellschätzung gegenüber Schwankungen in den Trainingsdaten.
- Hohe Varianz bedeutet, dass das Modell stark auf spezifische Details und Rauschen in den Trainingsdaten reagiert (Overfitting).
- Komplexe Modelle haben tendenziell eine hohe Varianz.
- Ein einfaches Modell hat eine niedrige Varianz, da seine Schätzungen weniger stark durch Änderungen in den Trainingsdaten beeinflusst werden.

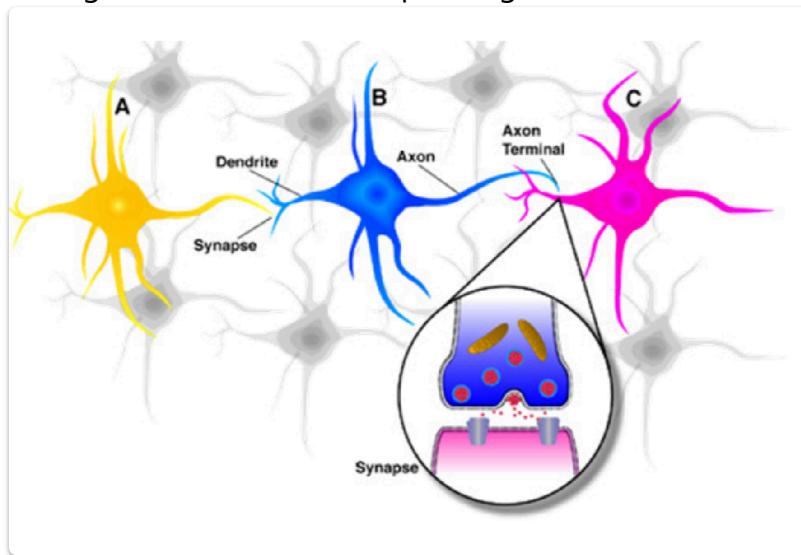
- **Bias-Variance Tradeoff:**

- Das Ziel ist es, ein Modell zu finden, das einen niedrigen Bias und eine niedrige Varianz aufweist, um eine gute Generalisierung zu erreichen.

- Es besteht ein Tradeoff, da das Reduzieren des Bias oft zu einer Erhöhung der Varianz führt und umgekehrt.
- **Gesamtfehler = Bias<sup>2</sup> + Varianz + Irreduzierbarer Fehler** (Der irreduzierbare Fehler ist durch die Daten selbst bedingt und kann durch das Modell nicht beeinflusst werden).
- **Komplexität und Bias-Variance:**
  - **Geringe Komplexität:** Hoher Bias, niedrige Varianz (Underfitting).
  - **Hohe Komplexität:** Niedriger Bias, hohe Varianz (Overfitting).
  - **Optimale Komplexität:** Findet ein Gleichgewicht zwischen Bias und Varianz, was zu einer guten Generalisierung führt.
- **Praktische Konsequenzen:**
  - Bei der Modellentwicklung müssen wir versuchen, die optimale Komplexität zu finden.
  - Die Leistung des Modells auf unabhängigen Validierungsdaten ist ein guter Indikator für das Vorliegen von Bias oder Varianz Problemen.
  - Techniken wie Regularisierung können helfen, die Varianz zu reduzieren, während komplexere Modelle den Bias reduzieren können (bis zu einem gewissen Punkt).

## Künstliche Neuronale Netze (NN) - Biologische Inspiration

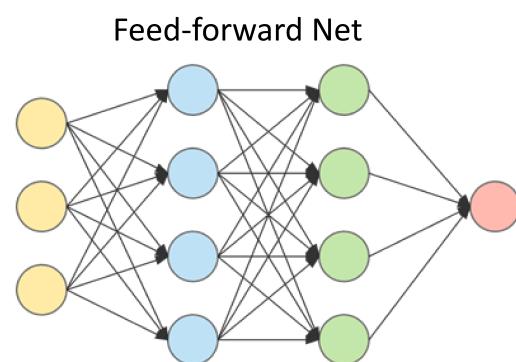
- **Biologisches Vorbild:** Neuronale Netze im Gehirn von Menschen und Tieren.
- **Komplexität des Gehirns:** Ca. 10 bis 100 Milliarden Nervenzellen im menschlichen Gehirn.
- **Grundlage für Intelligenz:** Komplexe Verschaltung und Adaptivität der Neuronen ermöglichen Lernen und Anpassung.



- **Struktur und Funktion eines biologischen Neurons (vereinfacht):**
  - **Zellkörper (A, B, C):** Enthält den Zellkern und andere Organellen. Fungiert als Speicher für kleine elektrische Spannungen (ähnlich Kondensator/Akku).
  - **Dendriten:** Verzweigte Fortsätze, die eingehende Signale von anderen Neuronen über Synapsen empfangen.

- **Axon:** Langer Fortsatz, der Ausgangssignale (Spannungsimpulse) zu anderen Neuronen über Synapsen weiterleitet.
- **Feuern des Neurons:** Wenn die im Zellkörper gespeicherte Spannung einen bestimmten Schwellwert überschreitet, entlädt sich das Neuron und sendet einen Spannungsimpuls über das Axon.
- **Synapsen:** Kontaktstellen zwischen dem Axon eines Neurons und den Dendriten eines anderen Neurons. Hier wird das Signal übertragen.
- **Lernen im biologischen neuronalen Netz:**
  - **Adaptivität der Synapsen:** Nicht die Neuronen selbst, sondern die Verbindungen (Synapsen) sind adaptiv.
  - **Veränderung der Verbindungsstärke:** Die Stärke der synaptischen Verbindungen kann sich verändern. Dies ermöglicht Lernen und Anpassung.
- **Mathematisches Modell der NN:**

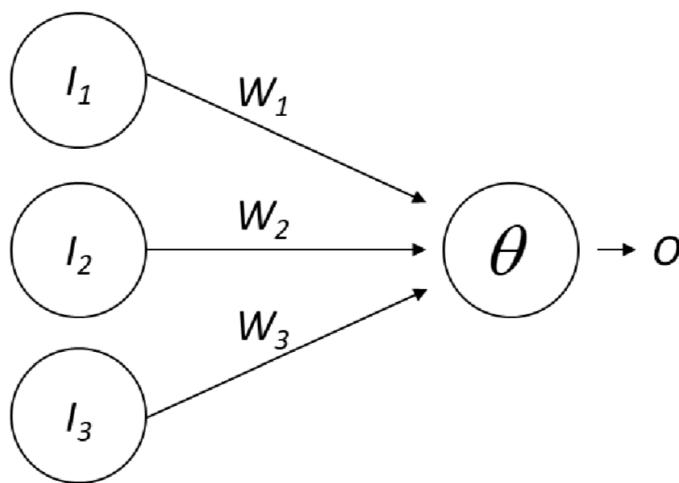
- In short: Neural Networks (NN)
- Multi-layer fully-connected NN
- Elements:
  - Neurons (nodes)
  - Synapses (weights)
- Consist of:
  - Input layer,
  - Multiple hidden layers,
  - Output layer.
- Every node in one layer is connected to every other node in the next layer.



- **Inspiration:** Das biologische Modell diente als Vorbild.
- **McCulloch und Pitts (1943):** Stellten das erste mathematische Modell eines Neurons als grundlegendes Schaltelement vor.
- **Grundlage für NN:** Dieses Modell war die Basis für den Bau künstlicher neuronaler Netze.
- **Bestandteile des mathematischen Modells:**
  - **Neuronen (Knoten):** Entsprechen den Nervenzellen.
  - **Synapsen (Kanten):** Verbindungen zwischen den Neuronen.
  - **Gewichte:** Den Kanten (Synapsen) werden Gewichte zugewiesen.
- **Adaption:** Durch die Anpassung der Gewichte kann das "Verhalten" des künstlichen Neurons verändert werden (entspricht der Adaptivität der biologischen Synapsen).

## Das Perceptron - Ein frühes künstliches neuronales Netz

- **Entwicklung:** 1958 von Frank Rosenblatt vorgestellt.
- **Struktur:** Besteht aus einem einzelnen künstlichen Neuron und einer Reihe von Eingängen ( $I_1$  bis  $I_k$ ).



- **Analogie zum biologischen Neuron:**
  - **Eingänge ( $I_1$  bis  $I_k$ ):** Entsprechen den Dendriten.
  - **Gewichte ( $w_1$  bis  $w_k$ ):** Werden mit den Eingangssignalen multipliziert. Entsprechen der Verstärkung oder Abschwächung natürlicher Signale durch Synapsen.
- **Gewichtete Summe der Eingaben:**  $\Phi(x) = \sum_i w_i I_i$
- **Aktivierungsfunktion ( $f$ ):** Wird auf die gewichtete Summe angewendet, um die Ausgabe des Neurons zu bestimmen:  $O = f(\sum_i w_i I_i)$ .
- **Ausgabe ( $O$ ):** Wird über "synaptische Gewichte" an nachgeschaltete Neuronen weitergegeben.

## Aktivierungsfunktionen

- **Vielfalt:** Es existieren verschiedene Möglichkeiten für die Aktivierungsfunktion  $f$ .
- **Einfachste Form: Identität**
  - $f(x) = x$
  - Das Neuron gibt lediglich die gewichtete Summe der Eingabewerte weiter.
  - **Problem:** Führt zu Konvergenzproblemen, da die Funktion nicht beschränkt ist und die Funktionswerte unbegrenzt wachsen können.

## Schwellwertfunktion im Perceptron

- **Motivation:** Um die unbegrenzte Ausgabe der Identitätsfunktion zu vermeiden, wird eine Schwellwertfunktion eingesetzt.
- **Funktionsweise:** Die gewichtete Summe der Eingangssignale  $\Phi(x) = \sum_i w_i I_i$  wird mit einem Schwellwert  $\theta$  verglichen.
- **Ausgabe ( $O$ ):**

$$O = \begin{cases} 1 & \text{falls } \sum_i w_i I_i + \theta \geq 0 \\ 0 & \text{sonst} \end{cases}$$

- **Abhängigkeit der Ausgabe:** Von den Eingangssignalen ( $I$ ), den zu lernenden Gewichten ( $w$ ) und dem Schwellwert ( $\theta$ ).

- **Ziel des Perceptrons (Klassifikation):** Trennung von Daten, die zu zwei unterschiedlichen Klassen gehören.
- **Lernaufgabe:** Anpassung der Gewichte  $w_i$ , sodass das Neuron bei Daten der ersten Klasse 0 und bei Daten der zweiten Klasse 1 ausgibt (oder umgekehrt).

## Perceptron-Lernalgorithmus ( $\delta$ -Regel / Widrow-Hoff-Regel)

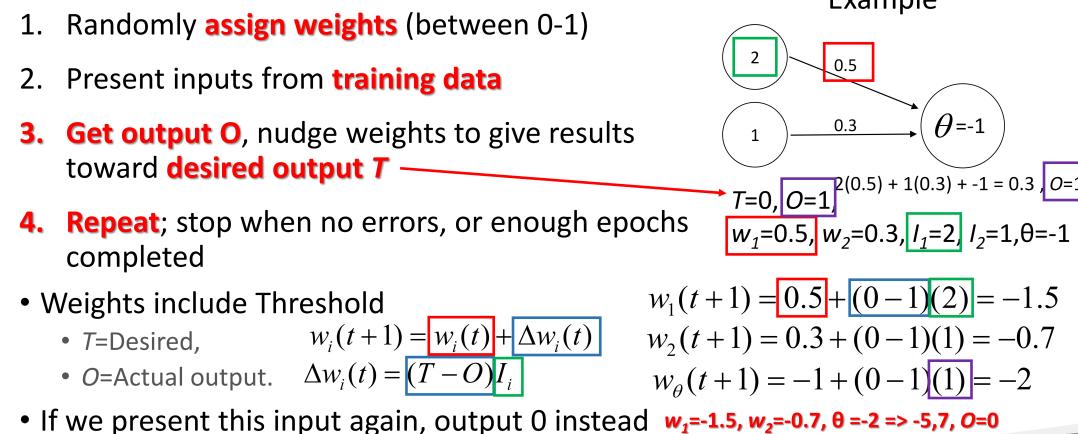
- **Ziel:** Anpassung der Gewichte, um die gewünschte Ausgabe zu erzielen.
- **$\delta$ -Regel:** Die Gewichtsänderung  $\Delta w_i(t)$  zum Zeitpunkt  $t$  wird basierend auf der Differenz zwischen der gewünschten Ausgabe ( $T$ ) und der tatsächlichen Ausgabe ( $O$ ) berechnet:

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\Delta w_i(t) = (T - O)I_i$$

- **Zwei Schlüsselkonzepte der  $\delta$ -Regel:**

1. **Fehlerbasierte Anpassung:** Die Gewichtsanpassung basiert auf dem Fehler ( $T - O$ ).
2. **Eingabeabhängigkeit:** Die Änderung hängt auch von der Eingabe  $I_i$  (der Ausgabe des vorherigen "Neurons") ab.



## Konvergenz des Perceptron-Lernalgorithmus

- **Rosenblatts Theorem:** Der Lernalgorithmus des Perceptrons konvergiert in endlicher Zeit, wenn die Daten linear separierbar sind.
- **Implikation:** Das Perceptron kann alles lernen, was es repräsentieren kann, in endlicher Zeit.

## Beschränkungen des Perceptrons

- **Nicht-lineare Separierbarkeit:** Ein einschichtiges Perceptron kann keine Funktionen lernen, die nicht linear separierbar sind.
- **Lineare Separierbarkeit:** Die Eigenschaft, dass Daten im Raum durch Hyperebenen (in 2D: Geraden, in 3D: Ebenen) voneinander getrennt werden können.

## Mehrschichtige Perceptrons

- **Zweischichtige Perceptrons:** Können konvexe Mengen separieren.

- **Mehrschichtige Perceptron-Netze:** Können beliebige Mengen trennen. Dies ist ein wichtiger Schritt zur Überwindung der Beschränkungen des einfachen Perceptrons.

## Aktivierungsfunktionen für stetige Neuronen

- **Problem der Stufenfunktion:** Für binäre Neuronen (Ausgabe 0 oder 1) sinnvoll, erzeugt aber eine Unstetigkeit bei stetigen Neuronen (Aktivierungen zwischen 0 und 1).
- **Lösung: Sigmoid-Funktion:** Eine Möglichkeit, die Unstetigkeit zu glätten.
- **Beispiel einer Sigmoid-Funktion:**

$$O = \frac{1}{1 + e^{-(\Phi(x)+\theta)}}$$

wobei  $\Phi(x) = \sum_i w_i I_i$  die gewichtete Summe der Eingaben und  $\theta$  der Schwellwert ist.

- **Eigenschaften der Sigmoid-Funktion:**
  - **Annähernd linear:** Verhält sich in der Nähe des kritischen Bereichs um den Schwellwert  $\theta$  annähernd linear.
  - **Asymptotisch beschränkt:** Die Ausgabe liegt immer zwischen 0 und 1. Dies verhindert das Problem des unbegrenzten Wachstums der Aktivierungen.
- **Vorteil gegenüber der Stufenfunktion:** Ermöglicht differenzierbare Ausgaben, was für viele Lernalgorithmen in neuronalen Netzen (insbesondere für das Training mit Gradientenabstieg) entscheidend ist.

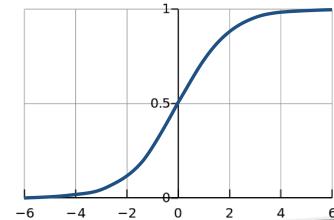
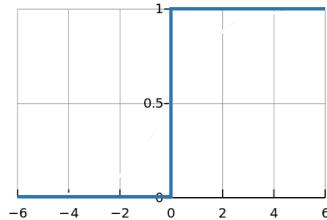
$$\Delta w_k = c I_k (T_j - O_j) f'(\text{ActivationFunction})$$

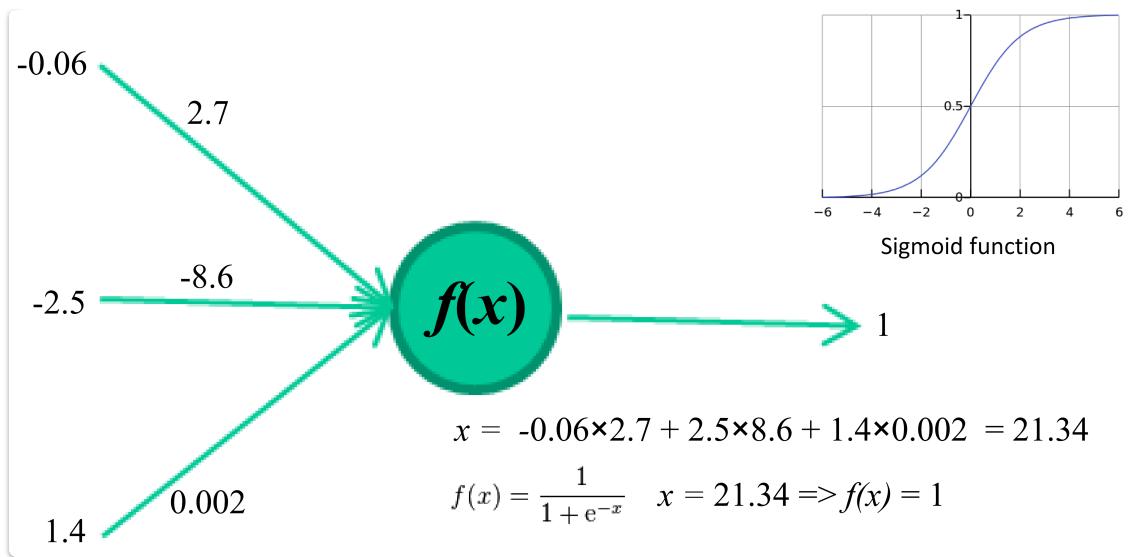
Old:

$$O = \begin{cases} 1: \sum_i w_i I_i + \theta > 0 \\ 0: \text{otherwise} \end{cases}$$

New:

$$O = \frac{1}{1 + e^{-\sum_i w_i I_i + \Theta}}$$

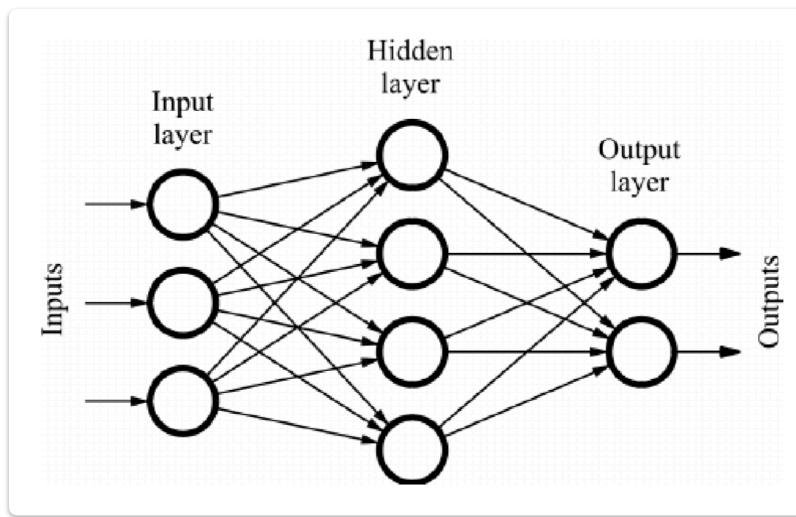




Ein animiertes Beispiel findet man in den Slides: [EVC-CV11-Deep Learning\\_2025S\\_Slides](#), p.42

## 11.6 Multilayer Perceptron (MLP)

- **Typischer Aufbau:** Besteht aus mehreren Schichten von Verarbeitungseinheiten.
- **Beispiel (Abbildung 49):**
  - **Eingangsschicht (Input Layer):** Empfängt die Eingabedaten.
  - **Verbogene Schicht (Hidden Layer):** Eine oder mehrere Zwischenschichten, die komplexe Muster in den Daten verarbeiten.
  - **Ausgabeschicht (Output Layer):** Liefert das Ergebnis der Verarbeitung (z.B. Klassifikation oder Regression).
- **Verbindungen:** Zwischen den Schichten befinden sich Lagen von Verbindungen, die mit Gewichten versehen sind.



- **Variationen der Struktur:**
  - **Shortcut-Verbindungen:** Direkte Verbindungen zwischen der Eingangsschicht und der Ausgabeschicht.
  - **Mehrere verbogene Schichten (Deep Neural Networks):** Ermöglichen die Modellierung noch komplexerer Beziehungen in den Daten.
- **Arbeitsweise: Vorwärtsvermittlung (Feed-forward Network):**
  - Die Aktivierung erfolgt schichtweise von der Eingabe- zur Ausgabeschicht.

- **Keine Rückkopplungen:** Signale fließen nur in eine Richtung.

## Training von Multilayer Perceptron (MLP)

- **Überwachter Lernvorgang:** Die Gewichtsfaktoren zwischen den Verarbeitungseinheiten werden während des Trainings angepasst.
- **Lernzyklus (Epoche):** Alle Muster des Trainingsdatensatzes werden vom Netz klassifiziert (oder für Regression verwendet).
- **Fehlerberechnung:** Die Ergebnisse des Netzes werden mit den Soll-Werten (Labels) des Trainingsdatensatzes verglichen.
- **Gewichtsanpassung:** Die Gewichtsfaktoren werden so angepasst, dass der Fehler zwischen der Netzwerkausgabe und dem Soll-Wert minimiert wird.
- **Ziel:** Nach einer Reihe von Trainingszyklen soll der Trainingsdatensatz mit einer vorgegebenen Genauigkeit reproduziert werden.

## Backpropagation-Training (Fehler-Rückvermittlung)

- **Bekanntestes Trainingsverfahren für MLPs.**
- **Prinzip:** In jedem Lernschritt werden die Gewichte in Richtung des abnehmenden Fehlers verändert (Gradientenabstieg).
- **Ablauf eines Lernschritts:**
  1. **Vorwärtslauf (Forward Pass):** Ein Eingangsmuster wird an das Netz angelegt und die Aktivierungen werden schichtweise bis zur Ausgabeschicht berechnet.
  2. **Vergleich mit Soll-Output:** Der Zustand der Ausgabeschicht wird mit dem bekannten Soll-Output für dieses Muster verglichen.
  3. **Fehlerberechnung:** Die Abweichung (der Fehler) zwischen der tatsächlichen und der gewünschten Ausgabe wird berechnet.
  4. **Rückwärtslauf (Backward Pass):** Der Fehler wird vom Ende des Netzes (Ausgabeschicht) zurück zu den vorherigen Schichten propagiert.
  5. **Gewichtsaktualisierung:** Die Gewichte jeder Verbindung im Netz werden proportional zum Beitrag dieser Verbindung zum Fehler angepasst.

## Generalisierte Delta-Regel und Fehlerrückvermittlung

- **Problem bei mehrschichtigen Netzen:** Für die verborgenen Schichten ist kein direkter Soll-Zustand bekannt. Die einfache Delta-Regel ist daher nicht direkt anwendbar.
- **Lösung:** Die Generalisierte Delta-Regel mit Fehlerrückvermittlung (Backpropagation).
- **Kernidee:** Der Fehler der Ausgabeschicht wird verwendet, um einen "virtuellen Fehler" für die Neuronen in den verborgenen Schichten zu berechnen. Dieser virtuelle Fehler gibt an, wie stark die Aktivität eines verborgenen Neurons zum Fehler in der Ausgabeschicht beigetragen hat.
- **Anwendung:** Mithilfe dieses virtuellen Fehlers können dann auch die Gewichte der Verbindungen, die in die verborgenen Neuronen führen, angepasst werden.

## Zielfunktion des Backpropagation-Algorithmus

- **Ziel:** Minimierung der Summe der quadratischen Fehler (Least Mean Squares, LMS) bei der Klassifikation aller Trainingsmuster.
- **Fehlerfunktion (LMS):**

$$Distance(LMS) = \frac{1}{n} \sum_{p=1}^n (T_p - O_p)^2$$

- $T_p$ : Soll-Ausgangswert für das  $p$ -te von  $n$  Trainingsbeispielen.
- $O_p$ : Tatsächlicher Ausgangswert des Netzes für das  $p$ -te Trainingsbeispiel.
- **Gesucht:** Das globale Minimum dieser Gesamtfehlerfunktion.

## Gradientenabstieg

- **Prinzip der Gewichtsanpassung:** Die Gewichte werden stets in Richtung des abnehmenden Fehlers verändert, d.h., entgegen dem Gradienten des Fehlers.
- **Ursprüngliche Delta-Regel:** Eine frühe Form dieser Idee.

## Kernidee des Backpropagation-Verfahrens

- **Fehler als Funktion der Gewichte:** Der Ausgangswert der Ausgabeeinheiten (und damit der Fehler  $F$ ) ist eine Funktion ihrer Eingabewerte. Diese Eingabewerte sind wiederum Funktionen der Ausgänge der vorhergehenden (verdeckten) Schicht und der Gewichte zwischen diesen Schichten.
- **Kettenregel:** Die Ausgänge der Zwischenschicht sind eine Funktion der Gewichte zwischen Eingangs- und Zwischenschicht. Daraus folgt, dass der Fehler  $F$  letztendlich auch eine Funktion dieser Gewichte ist.
- **Partielle Differentiation:** Durch Anwendung der Kettenregel ist es möglich, auch für die Zwischenschichten einen Fehler zu definieren.
- **Gradient für Zwischenschichten:** Aus diesem Fehler kann ein Gradient berechnet werden, um die Gewichte zwischen den Schichten anzupassen.

## Fehlerrückvermittlung (Error Backpropagation)

- **Berechnung des Fehlerwerts:** Zuerst für die Ausgabeeinheiten, dann für die verdeckten Einheiten.
- **"Backpropagation":** Die Fehleroptimierung pflanzt sich von hinten (Ausgabeschicht) nach vorne durch das Netz.
- **Implikation:** Die Information, wie die einzelnen Gewichte zum Fehler der Ausgabeschicht beigetragen haben, wird zurück durch das Netzwerk geleitet, um eine sinnvolle Anpassung der Gewichte in allen Schichten zu ermöglichen.

# Deep Learning

---

## 1. What exactly is Deep Learning?

- Deep Learning means using a neural network with several layers of nodes between input and output

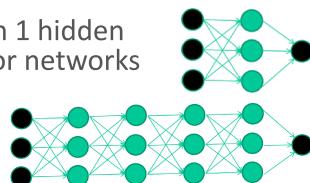
## 2. Why is it generally better than other methods?

- The series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.

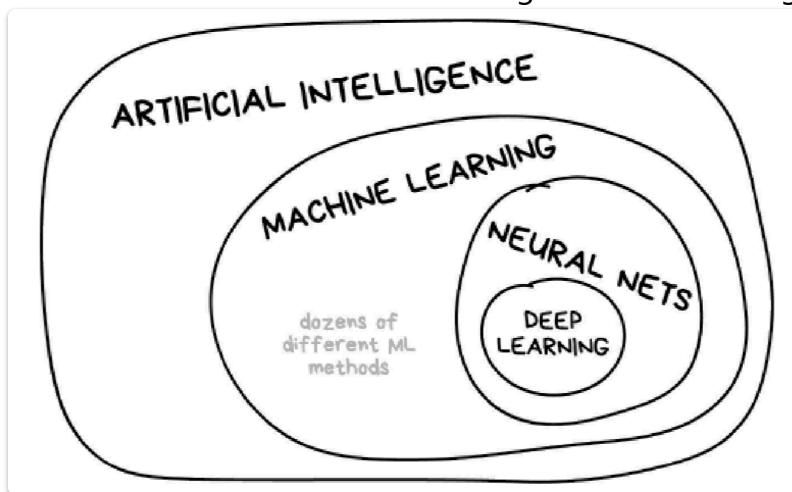
## 3. Multilayer neural networks have been around for 30 years.

### What's actually new?

- We had good algorithms for learning weights in networks with 1 hidden layer, but these algorithms are not good at learning weights for networks with more hidden layers.
- What's new is: algorithms for training many-layer networks



- **Bisherige Ansätze (Machine Learning):** Lernalgorithmen können aus Trainingsdaten komplexe Klassifikationsaufgaben lernen.
- **Manuelle Merkmalsgenerierung:** Die zur Klassifikation verwendeten Merkmale (z.B. Farbe, Kanten, SIFT) mussten bisher manuell von einem "Wissensingenieur" ausgewählt und generiert werden.
  - Ziel: Einen sinnvollen Satz von möglichst wenigen, aussagekräftigen Merkmalen finden.
  - Diese Merkmale dienen dann als Eingabe für die Lernalgorithmen.



## Herausforderung: Direkte Verwendung von Sensordaten

- **Idee:** Warum nicht direkt alle verfügbaren Sensordaten verwenden (z.B. Pixel eines Bildes)?
- **Beispiel (RGB-Bild):** Ein Foto mit 10 Millionen Pixeln hätte einen Eingabevektor der Länge 30 Millionen (10 Mio. Pixel x 3 Farbwerte).
- **Problem: Skalierung mit der Dimension der Eingabedaten:**
  - **Trainingszeiten:** Wachsen sehr schnell, oft exponentiell, mit der Dimension der Eingabedaten.
  - **Rechenaufwand:** Um die Rechenzeiten in Grenzen zu halten, müssen die Eingabedaten zuerst auf kurze Merkmalsvektoren abgebildet werden.

## Traditionelle Merkmalsgewinnung

- **Frühe Merkmalsbestimmung:** Merkmale werden frühzeitig anhand manuell definierter Formeln bestimmt.
- **Grundlage:** Meist Expertenwissen, orientiert an der menschlichen Wahrnehmung ("Wie würde der Mensch die Klassifikationsaufgabe lösen?").

## Deep Learning: End-to-End-Learning

- **Herausforderung der manuellen Merkmalsgenerierung:** Für viele komplexe Anwendungen (z.B. Objektklassifikation in Bildern) ist es sehr schwierig bis unmöglich, manuell gute Merkmale zu definieren.
- **Prinzip von Deep Learning:** End-to-End-Learning.
  - **Simultanes Lernen:** Nicht nur die optimale Verarbeitung der Merkmale wird gelernt, sondern auch gleichzeitig die optimale Herleitung dieser Merkmale aus den Rohdaten.
  - **Automatisierte Merkmalsentwicklung:** Das Netzwerk lernt selbstständig, welche Merkmale für die jeweilige Aufgabe relevant sind.

## Deep Learning Architekturen

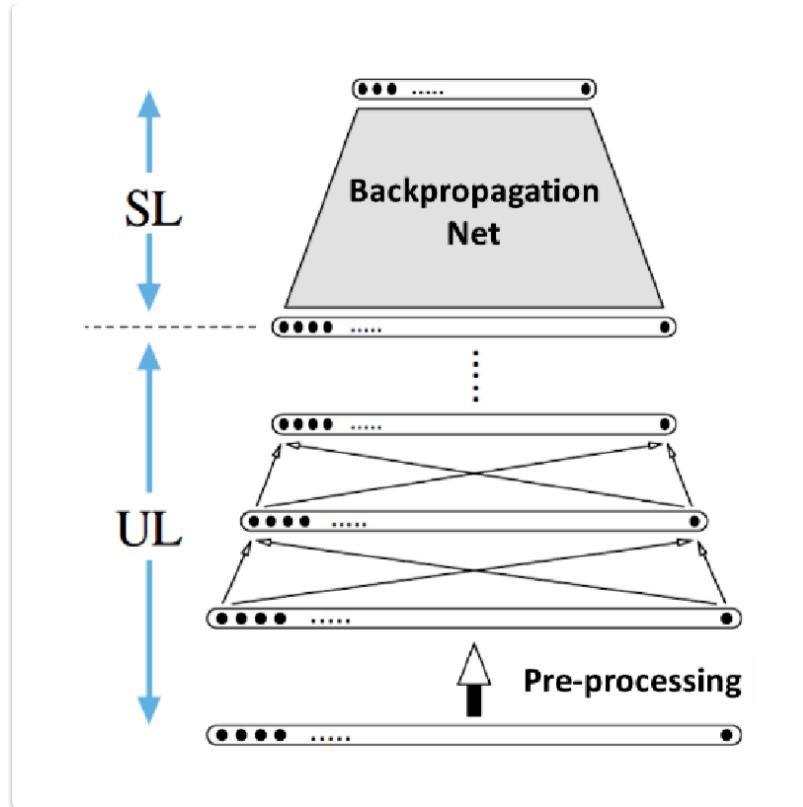
- **Computer Vision:**
  - **Convolutional Neural Networks (CNNs):** Kommen in der Regel zum Einsatz.
  - **Filterkoeffizienten:** Werden gelernt, um die Bilder zu Beginn zu falten (convolved). Diese Faltungsschritte extrahieren automatisch hierarchische Merkmale aus den Bildern (z.B. Kanten, Texturen, Objektteile).
- **Andere Deep Learning Verfahren:**
  - **Recurrent Neural Networks (RNNs):** Gut geeignet für sequenzielle Daten (z.B. Text, Zeitreihen).
  - **Generative Adversarial Networks (GANs):** Werden für generative Aufgaben eingesetzt (z.B. Erzeugung neuer Bilder, Texte).
  - **Variationen dieser Architekturen.**

## Komplexität tiefer neuronaler Netze

- **Anzahl der Schichten:** Architekturen mit bis zu fünfzig oder mehr Schichten sind möglich.
- **Hohe Komplexität:** Die genauen Architekturen sind oft sehr komplex und können hier nicht im Detail dargestellt werden.
- **Ressource für weitere Informationen:** Gute Einführungen finden sich auf [ufldl.stanford.edu/tutorial/](http://ufldl.stanford.edu/tutorial/).

## Deep Learning mit vielen Schichten

- **Aktueller Erfolg:** Erfolgreiche Deep-Learning-Lösungen verwenden viele Schichten von Neuronen.
- **Netzwerkstruktur (Abbildung 50):** Oft in zwei Teile aufgespalten.

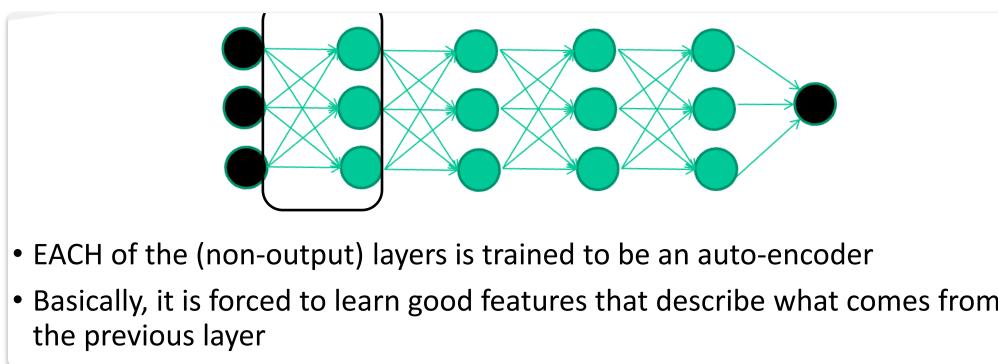
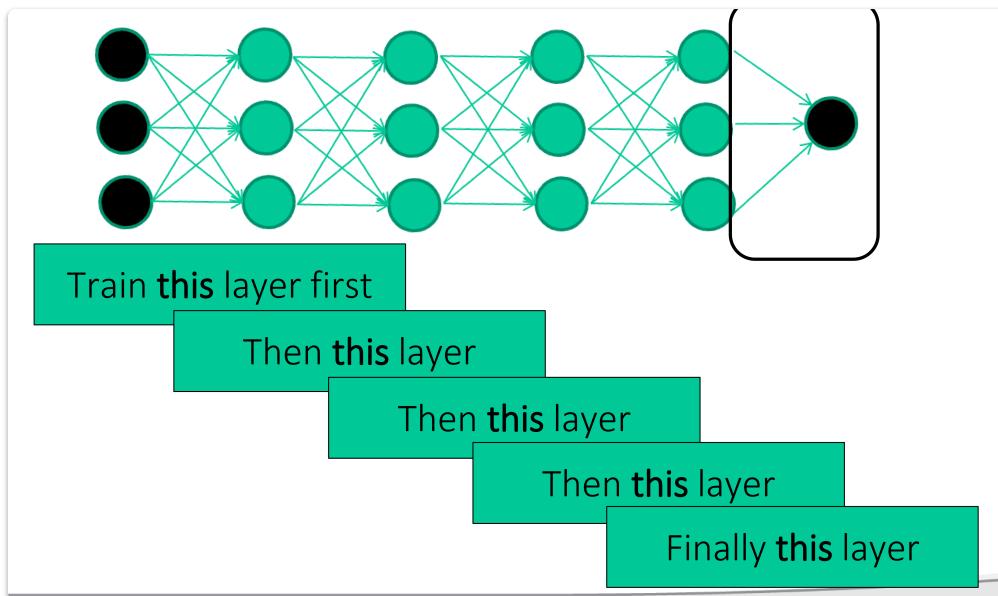


- **Unsupervised Learning (UL) zur Vorverarbeitung:**
  - Nach einer Vorverarbeitungsschicht folgen mehrere Lagen, die mit unüberwachtem Lernen vorgenommen werden.
  - **Merkalsrepräsentation:** Jede Lage im UL-Netz repräsentiert Merkmale des Eingabemusters.
  - **Hierarchische Merkmale:**
    - **Tiefe Lagen:** Repräsentieren einfache Merkmale (z.B. Kanten oder Linien in verschiedenen Ausrichtungen bei der Objekterkennung in Bildern).
    - **Höhere Lagen:** Können komplexe Merkmale bilden (z.B. das Vorhandensein eines Gesichts).
- **Supervised Learning (SL) zur Klassifikation/Regression:**
  - An das UL-Netz schließt sich ein klassisches überwachtes Lernen an.
  - **Training:** Kann beispielsweise mit Backpropagation trainiert werden.
- **Ablauf des Lernens:**
  1. **UL-Training:** Vortrainieren der Gewichte aller Merkmalschichten (Extraktion der Merkmale).
  2. **SL-Training:** Training des gesamten Netzes (einschließlich der vorgenommenen Schichten) mit überwachten Daten (z.B. Gradientenabstieg).

## Unsupervised Learning der Gewichte und Merkmalsextraktion

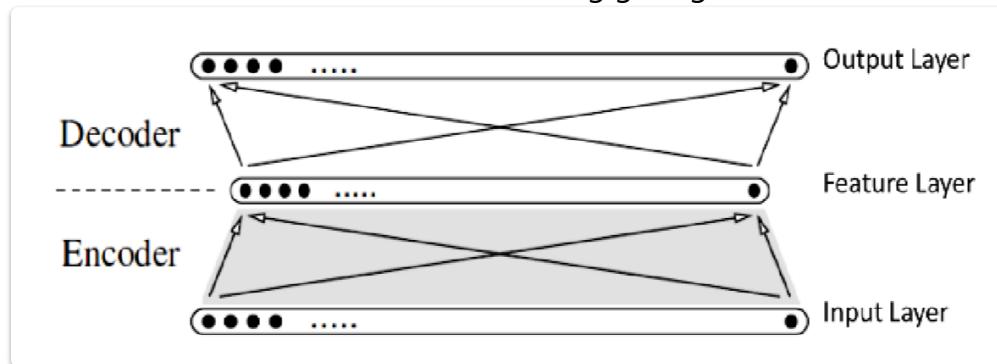
- **Ziel des UL-Trainings:** Bildung der Merkmale durch das Lernen der Gewichte zu den Merkmalschichten. Hier findet die eigentliche Merkmalsextraktion statt.
- **Eigenschaft der Merkmalsextraktion:** Die Eingabedaten sollen in einen niedrigerdimensionalen Raum abgebildet werden, möglichst ohne (zu viel) Informationsverlust.

Dies reduziert die Dimensionalität des Problems für die nachfolgende SL-Phase und kann zu robusteren und effizienteren Modellen führen.



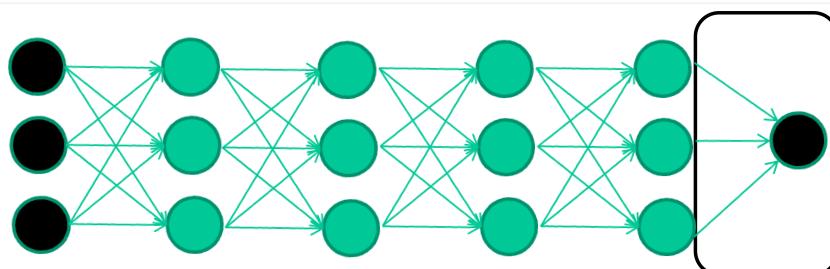
## Autoencoder

- **Bestimmung der Gewichte der Merkmalschichten (UL):** Erfolgt bei sogenannten Autoencodern nach dem in der Abbildung gezeigten Verfahren.



- **Training der ersten verdeckten Merkmalschicht:**
  - Ein Autoencoder wird mit unüberwachtem Lernen (UL) trainiert.
  - **Ziel des Autoencoders:** Die identische Abbildung aller Eingabevektoren  $x$  auf sich selbst zu lernen (Identitätsfunktion).
  - **Merkmalschicht:** Dient hier als die erste verdeckte Lage (Feature Layer).
- **Nach dem Training des ersten Autoencoders:**

- Die nicht benötigte Decoderlage (die zweite Lage von Gewichten) wird gelöscht.
- Die erste Lage der Gewichte (Encoder) wird eingefroren und für die Berechnung der Merkmale in der ersten Lage des UL-Netzes übernommen.
- **Training der zweiten Merkmalschicht:**
  - Mit dieser festen ersten Schicht von Gewichten wird die zweite Merkmalschicht wieder mit dem Autoencoder-Verfahren trainiert.
  - Die Gewichte der Encoder-Seite des zweiten Autoencoders werden eingefroren.
- **Iterativer Prozess:** Dieser Vorgang wird so weitergeführt bis zur letzten Merkmalschicht.
- **Abschluss des unüberwachten Teils des Lernens:** Sobald alle Merkmalschichten auf diese Weise vortrainiert wurden, ist der unüberwachte Teil des Lernens beendet.



- Is trained to predict class based on outputs from previous layers

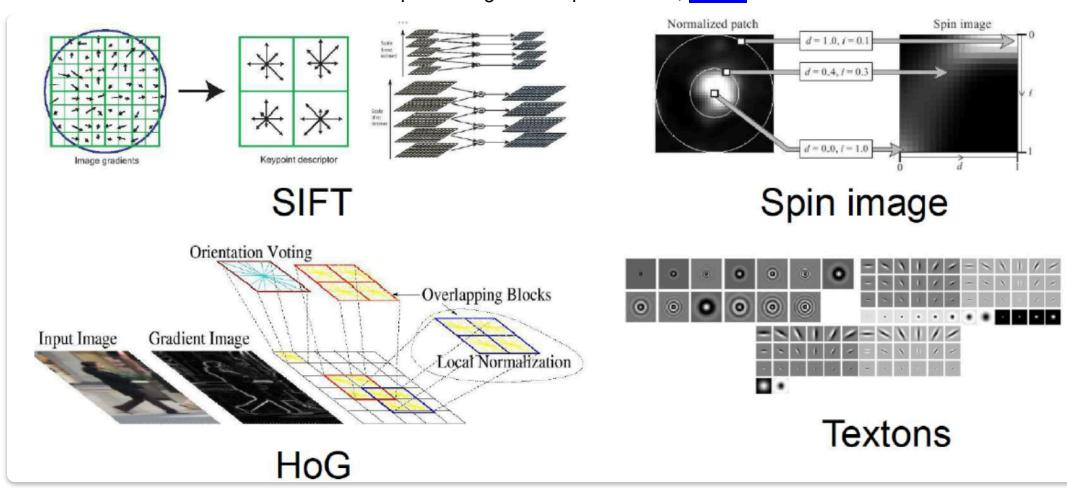
## Pros / Cons von Deeplearning

• Pros	• Cons
<ul style="list-style-type: none"> <li>• Not-domain specific</li> <li>• Supervised / Semi-supervised / Unsupervised</li> <li>• Classification / regression in last layer</li> <li>• Simple math</li> <li>• Hip</li> </ul>	<ul style="list-style-type: none"> <li>• Lots of meta-parameters</li> <li>• Needs a lot of data</li> <li>• Very computational intensive</li> <li>• Hip</li> </ul>

## Weitere VO-Slides:

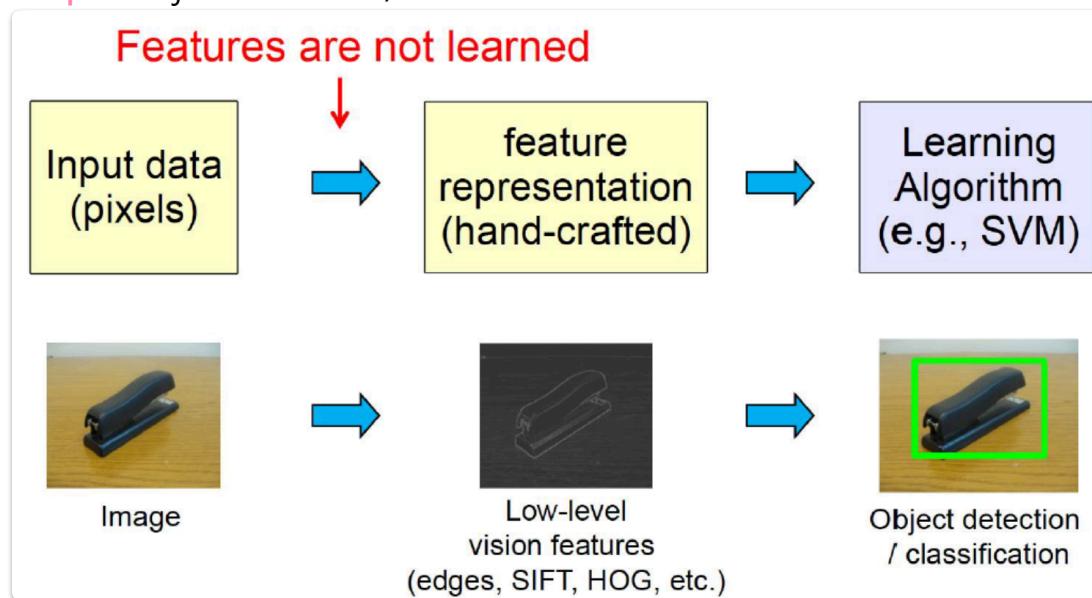
### Traditional Computer Vision Features

- **Beispiele für handgefertigte Features:**
  - SIFT (Scale-Invariant Feature Transform)
  - Spin image
  - HoG (Histogram of Oriented Gradients)
  - Textons
  - Viele andere wie SURF, MSER, LBP, Color SIFT, Color Histogram, GLOH, ...



## Traditional Recognition Approach

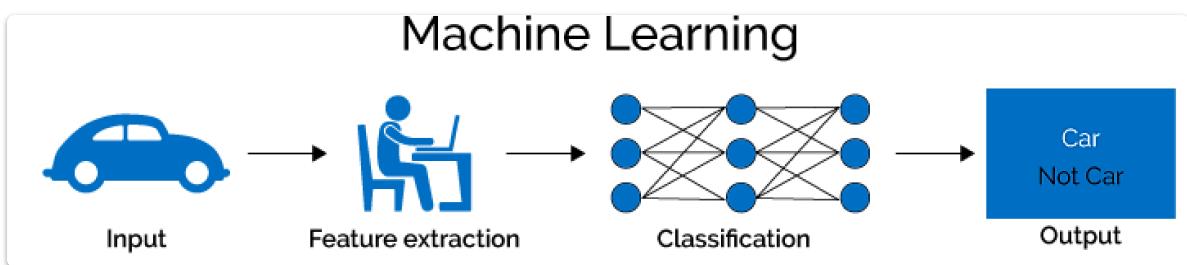
- Merkmale werden nicht gelernt (Features are not learned).
- Ablauf:
  1. **Input data (pixels):** Rohdaten, z.B. ein Bild.
  2. **Feature representation (hand-crafted):** Manuell entworfene Merkmalsrepräsentation (Low-level vision features wie edges, SIFT, HoG, etc.).
  3. **Learning Algorithm (e.g., SVM):** Ein Lernalgorithmus (z.B. Support Vector Machine) wird auf den extrahierten Merkmalen trainiert.
  4. **Output:** Objekt-Detektion / Klassifikation.



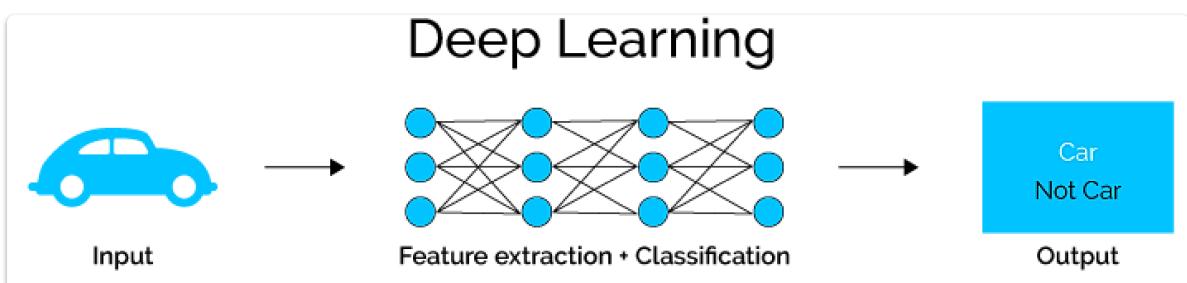
## Difference of Deep Learning to Classic Machine Learning

- **Machine Learning (Klassisch):**
  - Computes features.
  - Has simple and complex features.
  - **Ablauf:**
    1. **Input:** Rohdaten (z.B. ein Auto-Bild).
    2. **Feature extraction:** Manuelle oder algorithmische Extraktion von Merkmalen.

3. **Classification:** Ein separater Klassifikator lernt, die extrahierten Merkmale den Klassen zuzuordnen.
4. **Output:** Klassifikation (z.B. "Car" oder "Not Car").



- **Deep Learning:**
  - Does not need human interaction for feature design.
  - **Ablauf:**
    1. **Input:** Rohdaten (z.B. ein Auto-Bild).
    2. **Feature extraction + Classification:** Die Merkmalsextraktion und die Klassifikation erfolgen gemeinsam und werden end-to-end gelernt in einem tiefen neuronalen Netzwerk.
    3. **Output:** Klassifikation (z.B. "Car" oder "Not Car").

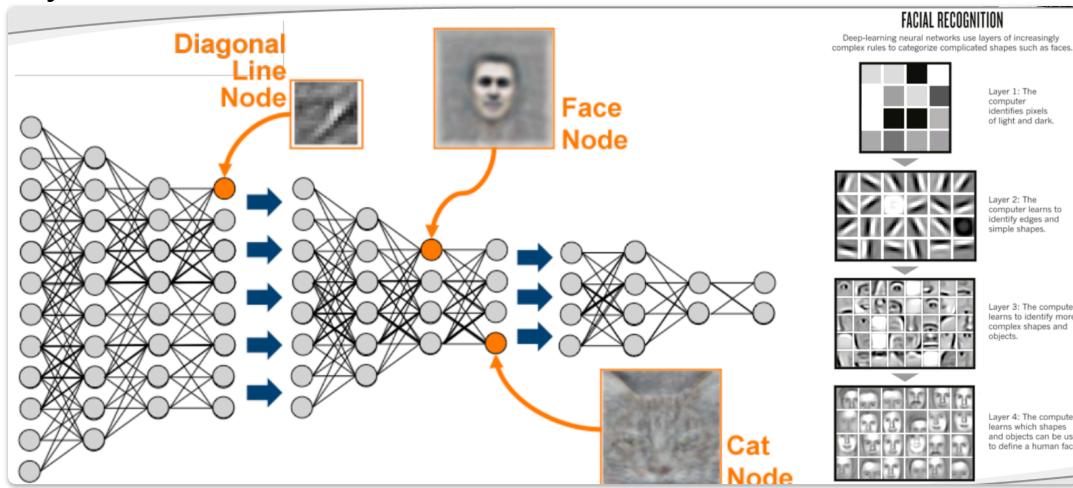


**Zusammenfassend:** Der Hauptunterschied besteht darin, dass traditionelles Machine Learning auf handgefertigten Merkmalen basiert, während Deep Learning die relevanten Merkmale direkt aus den Rohdaten lernt, was die Notwendigkeit menschlicher Expertise im Feature Engineering reduziert und oft zu besseren Ergebnissen bei komplexen Aufgaben führt.

## CNN Features

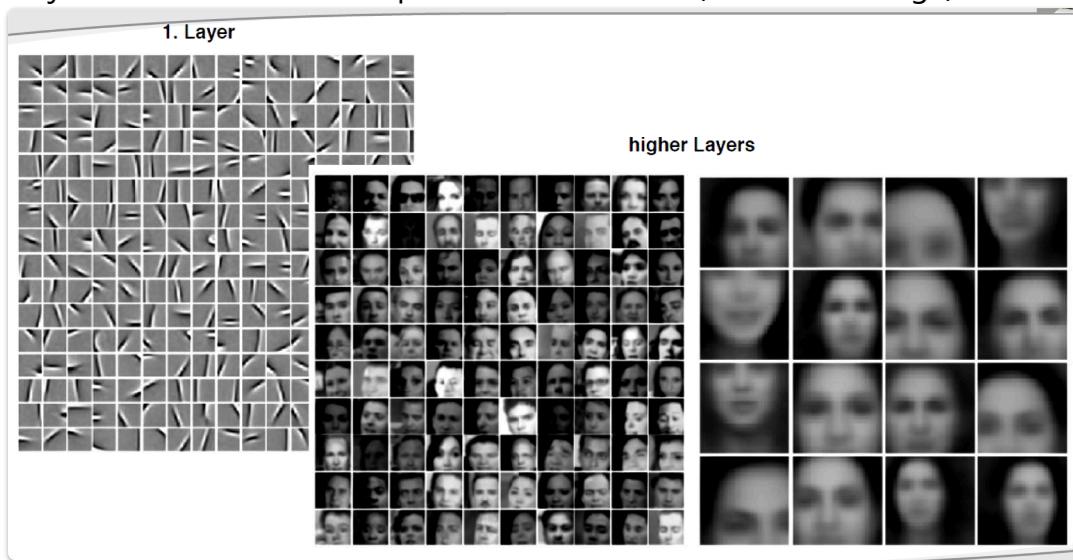
- **Hierarchische Merkmalsrepräsentation:** Convolutional Neural Networks (CNNs) lernen hierarchische Merkmale.
- **Beispiel (Gesichtserkennung):**
  - **Diagonale Kanten (frühe Schichten):** Neuronen in frühen Schichten können auf einfache Merkmale wie diagonale Kanten reagieren.
  - **Gesichtsteile (mittlere Schichten):** Spätere Schichten kombinieren diese einfachen Merkmale, um komplexe Muster wie Augen, Nasen oder Münden zu erkennen.
  - **Gesicht (späte Schichten):** Noch höhere Schichten können das gesamte Gesicht als abstraktes Merkmal repräsentieren.

- **Katze (anderes Beispiel):** Ähnliche hierarchische Merkmalsentwicklung für andere Objekte.



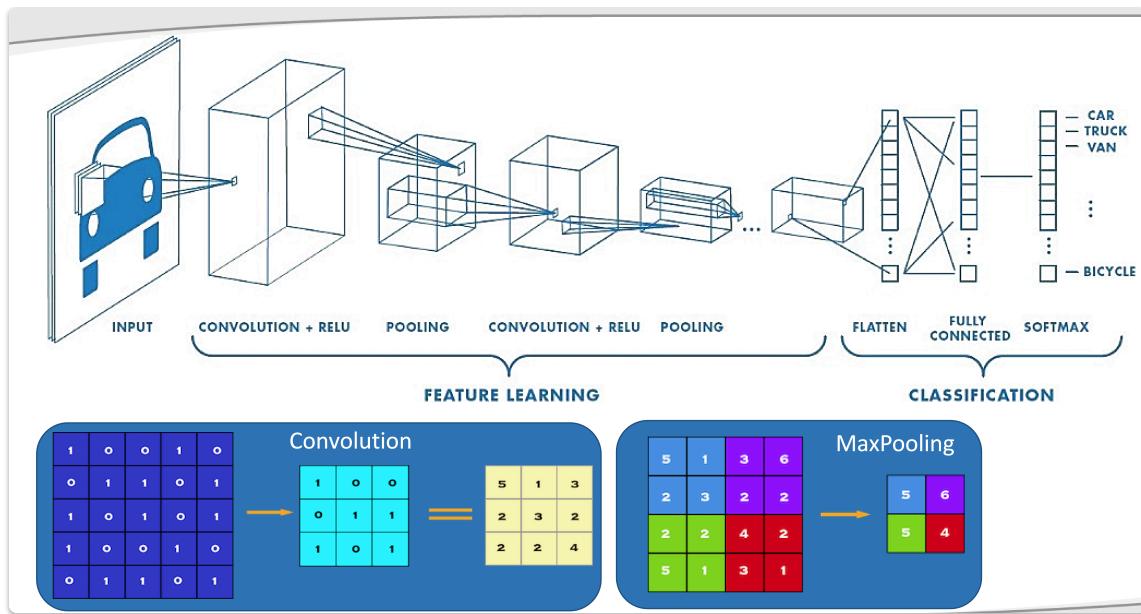
## CNN Visualization

- **Visualisierung gelernter Filter:** Die Abbildung zeigt beispielhafte Filter, die von CNNs in verschiedenen Schichten gelernt wurden.
  - **1. Layer:** Lernt oft einfache, lokale Muster wie Kanten und Orientierungen.
  - **Höhere Layer:** Lernen zunehmend komplexere und globalere Muster, die spezifische Objekte oder Teile davon repräsentieren können (z.B. Gesichtszüge).



## Inception Topologies

- **Beispiel einer komplexeren CNN-Architektur:** Die Inception-Architektur (hier vereinfacht dargestellt) verwendet verschiedene Filtergrößen und Operationen parallel, um Merkmale unterschiedlicher Skalen zu erfassen.
- **Feature Learning Pfad:** Zeigt typische Operationen wie Convolution und Pooling zur Merkmalsgewinnung.
- **Classification Pfad:** Führt die gelernten Merkmale zu einer Klassifikationsschicht (z.B. Fully Connected Layer mit Softmax).



## Deep Learning - Why does it work?

- **Can cope with vast amounts of data:** Tiefe Netze können von großen Datenmengen profitieren, um komplexe Muster zu lernen.
- **Learns small invariances:** Sie lernen, invariant gegenüber kleinen Variationen in den Eingabedaten zu sein (z.B. leichte Verschiebungen, Skalierungen, Rotationen).
- **Over-complete, sparse representations:** Können redundante und gleichzeitig spezialisierte Repräsentationen lernen.
- **Learn embedding:** Lernen, Eingabedaten in einen semantisch sinnvollen, niedrigerdimensionalen Raum einzubetten.
- **Lots of data:** Die Verfügbarkeit großer Trainingsdatensätze ist entscheidend für den Erfolg von Deep Learning.
- **Recent advance: it is actually computable!**: Fortschritte in der Hardware (z.B. GPUs) und in den Algorithmen haben das Training tiefer Netze in praktikabler Zeit ermöglicht.

## Weitere vo-slides zu ImageNet

### CNN Success Story: ILSVRC

- **ImageNet database:**
  - 14 million labeled images.
  - 20,000 categories.

### ILSVRC: Classification

- **Computer Vision: International Large-Scale Visual Recognition Challenge (ILSVRC).**
- **Ziel:** Bildklassifizierung auf sehr großem Maßstab.
- **Beispielbilder aus dem ILSVRC Datensatz.**

