

TECHNICAL APPLICATIONS AND DATA MANAGEMENT. SS 2023.

VORLESUNG 8.

15.05.2023
MÜNCHEN
STUDIENGANG
SUSTAINABILITY
MANAGEMENT.



AGENDA

1. Einführung Neuronale Netze/ Deep Learning
2. Convolutional Neural Networks
3. Transfer Learning
4. Case Study

GEPLANTE ROADMAP DER VORLESUNG.

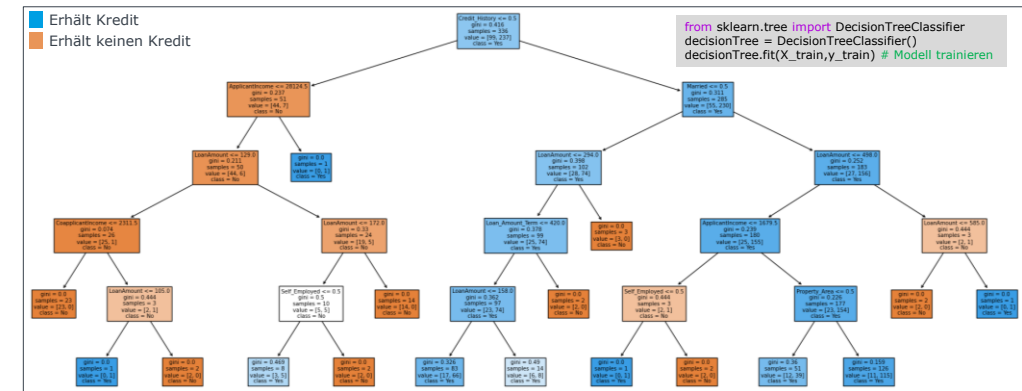
ROADMAP	WAS HABEN WIR VOR?
Vorlesung 1	Übersicht und Einführung
Vorlesung 2	Einführung Data Science und Data Science Workflow, Detaillierung Data Engineering
Vorlesung 3 und Vorlesung 4	Deskriptive und explorative Datenanalyse und Vertiefung anhand Case Study
	Vertiefung Datenanalyse anhand Case Study
Vorlesung 5	Aufgabenstellung Data Science, Übersicht und Einführung Machine Learning, unüberwachtes Lernen
Vorlesung 6 und Vorlesung 7	Überwachtes Lernen
	Vertiefung überwachtes Lernen anhand Case Study
Vorlesung 8 und Vorlesung 9	Neuronale Netze und Convolutional Neural Networks (CNN)
	Vertiefung CNN anhand Case Study, Aufgabenstellung AI
Vorlesung 10	Rekurrente Neuronale Netze
Vorlesung 11	Generative AI
Vorlesung 12	Ausblick
Vorlesung 13	„Fragestunde“

Folien der bisherigen Vorlesung verfügbar unter [Link](#)

1. NEURONALE NETZWERKE

MOTIVATION NEURONALE NETZE

- Wir haben in der letzten Vorlesung einige Machine Learning Verfahren kennengelernt.
- Diese Verfahren sind schnell einsetzbar und trainierbar.
- Und ihre Ergebnisse sind leicht verständlich – meistens....



Aber:

- sie haben Probleme mit großen Datenmengen.
- Sie liefern schlechte Performance mit komplexeren Daten wie Bildern, Texte oder Sprache aufgrund „statischer“ Sicht.

Neuronale Netze können mit sehr großen Datenmengen besser umgehen als die bisher betrachteten Verfahren. Dafür sind sie komplexer, benötigen länger zum Trainieren und die Erklärbarkeit der Ergebnisse ist nicht ganz einfach.

ÜBERSICHT NEURONALE NETZE.

Ziel: Wir suchen ein Modell, um für beliebige Eingaben X eine Zielgröße Y möglichst genau zu bestimmen oder anzunähern.

Ermöglicht:

- Regression: Vorhersage kontinuierlicher Wert für Y (bspw. natürliche oder reelle Zahl)
- Klassifikation: Vorhersage diskreter Wert für Y (bspw. Wahr/ Falsch, Tierart, Personenentdeckung,)
- Neu: Erkennen von Bildinhalten (CNN) und zeitabhängigen Daten (RNN)

Alles bisher
bekannt...

Was schauen wir uns?

- Convolutional Neural Networks
- Transfer Learning
- ~~Regression~~
- Rekurrente Neuronale Netze (übernächste Woche)

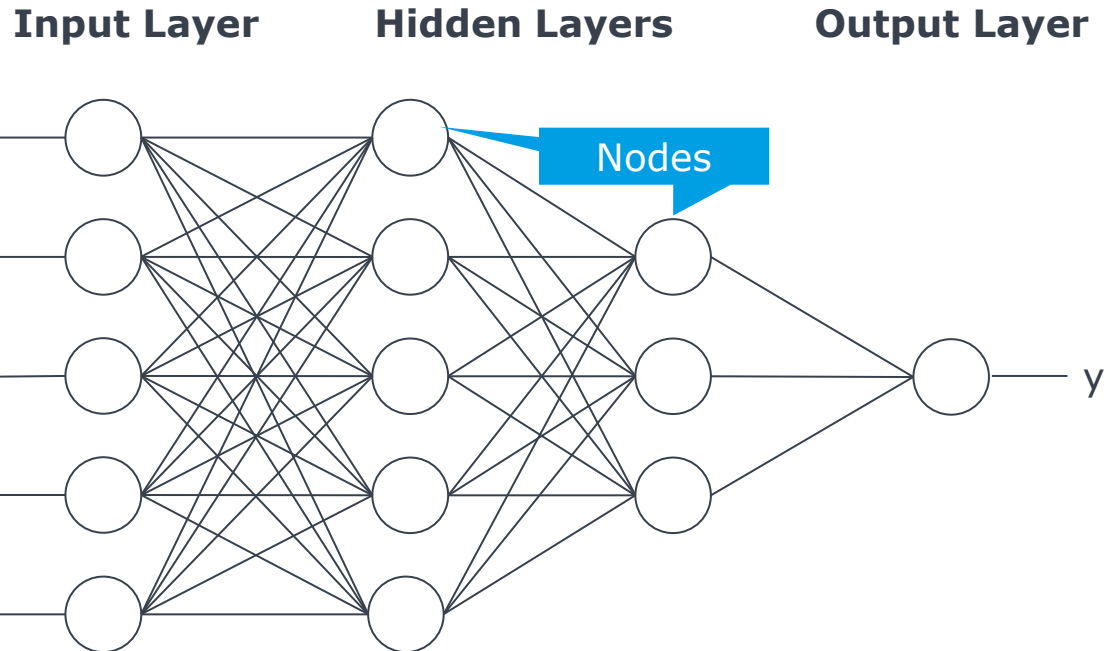
ARTEN NEURONALER NETZE.

- „simple“ neuronale Netzwerke für Regression/ Klassifikation.
- Convolutional Neural Networks (CNN)/ gefaltete neuronale Netzwerke: Top-Methode für Bilderkennung.
 - Simple CNN („Vanilla CNN“).
 - Transfer Learning: Übernahme und Anpassen vortrainierter CNN-Netzwerke.
- Rekurrente Neuronale Netzwerke: zeitabhängige Daten wie Sprache, Börsenkurse, ...

spätere Vorlesung

Alle drei Varianten haben prinzipiell die gleiche Struktur, CNN und RNN setzen auch „normale“ Neural Networks ein.

ÜBERSICHT EINFACHES NEURONALES NETZ (MULTILAYER PERCEPTRON).

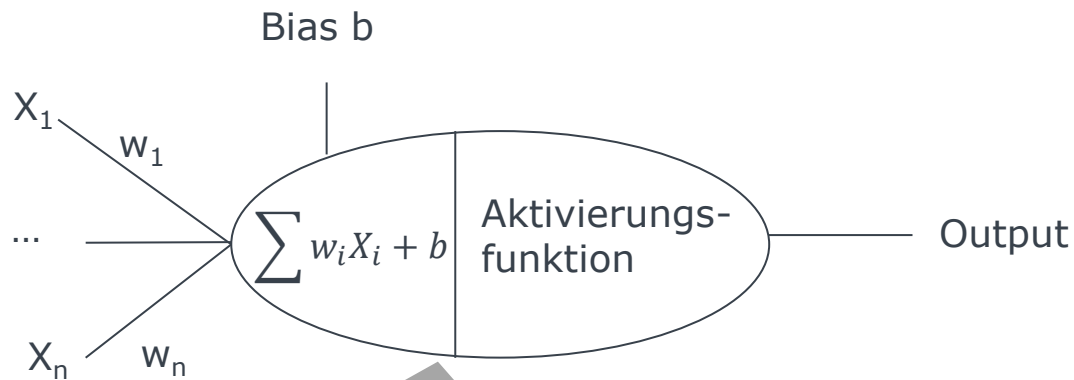


Der Input Layer wird bei der Anzahl der Layer meist nicht mitgezählt. Ein neuronales Netz mit mindestens 1 Hidden Layer wird **Deep Learning Network** genannt.

- **Input Layer:** Erhält Eingaben **X** und reicht diese direkt an Hidden Layer weiter. Keine Datenumwandlung/-bearbeitung! Der Input Layer skaliert mit/ bildet die Eingabegröße ab (bspw. ein Input je Anzahl Pixel eines Bildes, ...).
- **Hidden Layer:** hier erfolgt das Lernen innerhalb der einzelnen **Nodes** sowie in **Kombination der nachgelagerten Layers**. Durch Verwenden mindestens eines Hidden Layers können beliebig komplexe Funktionen abgebildet/ angenähert werden¹.
- **Connections:** Verbindungen zwischen einzelnen Elementen der Layer. Ist ein Layer mit allen Elementen des vorherigen und nachherigen Layer verbunden, spricht man von einem **fully connected layer**.
- **Output Layer:** letzte Layer. Generiert das finale Ergebnis **y**. Anzahl entspricht dem zu prädizierenden Ergebnis (bspw. 1 für Erkennen Hund/ Katze oder 10 bei Ziffern).

Eingaben X für den Input Layer werden durch die Nodes der Hidden Layers sowie des Output Layer so umgewandelt, daß die Zielgröße y möglichst genau bestimmt oder angenähert wird (Fallbeispiel Malaria: Input-Bild wird klassifiziert)

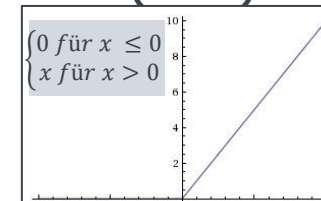
DETAILLIERUNG NODE/ PERCEPTRON.



Der Name neuronales Netz stammt von der Ähnlichkeit eines Perceptrons zu einem Neuron im Gehirn. Beide funktionieren aber unterschiedlich.

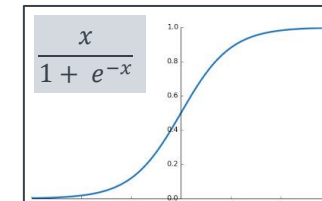
- **Weights w_i :** jeder Input hat ein Gewicht, das angibt, wie stark der Einfluß des Inputs auf den Node ist.
- **Bias b_i :** stellt Aktivierung Node auch bei Inputs = 0 sicher. (vgl. Geradengleichung $mx + t$ für $x=0$).
- **Aktivierungsfunktion¹:** verarbeitet die Gewichte w , Eingaben X_i und Bias b . Ermöglicht Abbilden komplexer Beziehungen zwischen Input und Output. Meist setzen wir eine der folgenden Funktionen ein:

Rectified Linear Unit (ReLU)²



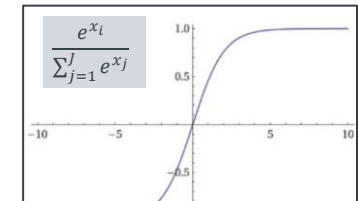
Empfehlung: Einsatz für alle Schichten (bis auf letzten Layer)

Sigmoid



Einsatz im letzten Layer f. binäre Klassifikation

SoftMax



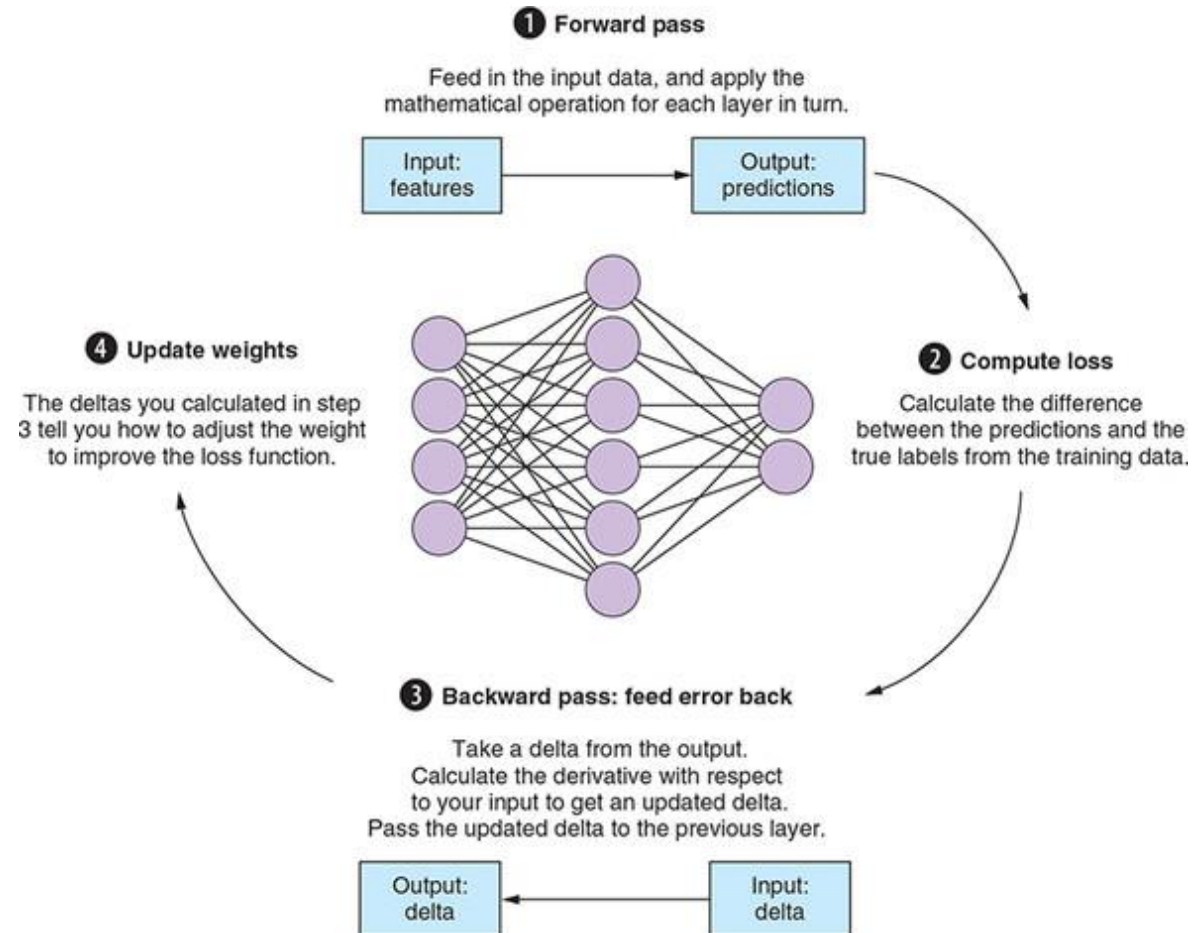
Einsatz im letzten Layer für Klassifik. vieler Zielklassen

Weights und Bias sind die Parameter, die im Training gelernt werden, um die Zielgröße/n anzunähern.

¹ Funktion sollte:

- Kont. Differenzierbar/ keine Lücken haben, da sonst Gradient Descent nicht funktioniert
 - begrenzte Wertemenge (meist zwischen 0 und 1) haben, damit Gradient Descent stabiler ist
 - nicht linear sein, um komplexere Funktionen als Geraden abzubilden. Aber: ReLU funktioniert sehr gut!
- Quellen: 2 Nair, V. & Hinton, G.: "Rectified linear units improve restricted Boltzmann machines". ICML, 2010.

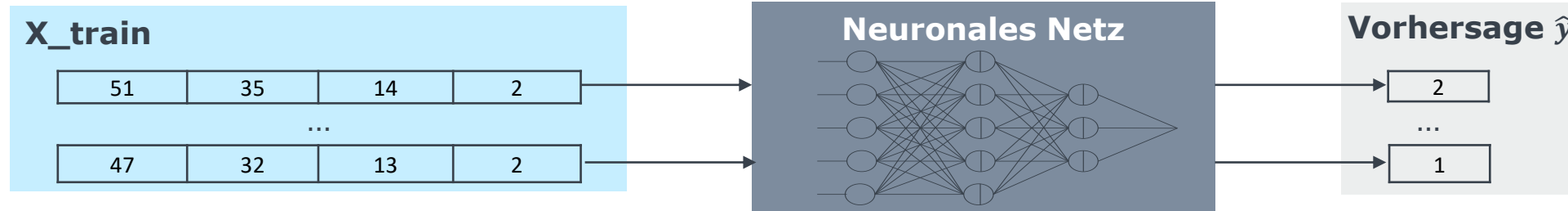
ÜBERSICHT TRAININGS-PROZESS NEURAL NETWORK.



Durchführen Schritte 1-4 inkl. Update Modellparameter für gesamtes Trainingsset heißt **Epoch**, für Teile Trainingsset **Batch**. Anzahl Durchläufe je Batch/ Epoch erfahrungsabhängig inkl. Abbruchkriterien (ausreichende oder stagnierende Güte)

TRAINING VON NEURONALEN NETZEN

1. Feed Forward: Vorhersage Wert je **Sample** (individuelle Zeile Datensatz).



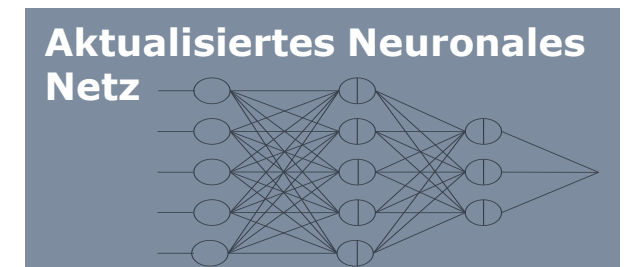
2. Berechnen Modellgüte und Fehler des Modells

- **Loss Function:** Berechnen Delta zwischen realem und vorhergesagtem Wert.
- **Cost Function:** Durchschnitt der Losses für jedes Sample des Trainingsset.



3. Minimierung Fehler und Anpassen Modellparameter:

- Aufstellen Gleichung für Minimierung Cost Function.
- Lösen Gleichung für die Modellparameter per Gradient Descent¹.
- Aktualisierung Modellparameter per **Backpropagation**².

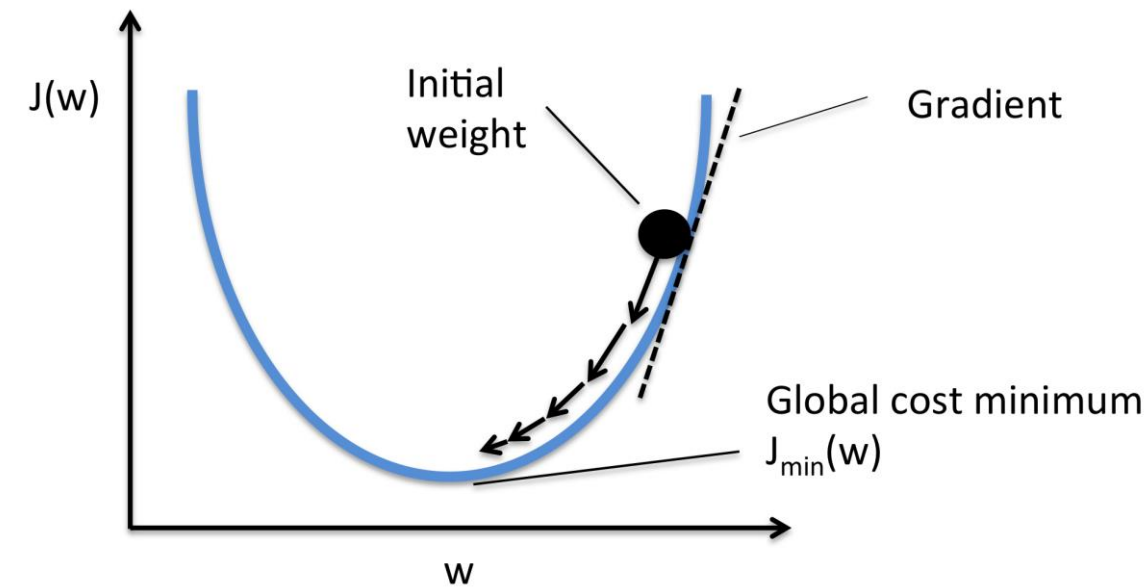


Durchführen Schritte 1-3 inkl. Update Modellparameter für gesamtes Trainingsset heißt **Epoch**, für Teile Trainingsset **Batch**. Anzahl Durchläufe je Batch/ Epoch erfahrungsabhängig inkl. Abbruchkriterien (ausreichende oder stagnierende Güte)

DIE ANPASSUNG DER PARAMETER DES NEURONALEN NETZES ERFOLGT DURCH GRADIENT DESCENT.

2 Möglichkeiten für Berechnen globales Minimum¹:

1. Lösen Gleichung Cost-Fkt $J(w) = 0$ für alle Gewichte/ Bias
2. Iteratives Verfahren Gradient Descent



Ablauf Gradient Descent Algorithmus:

1. Initialisiere zufällig gewählte Gewichte w sowie Bias b .
2. Iteriere über alle Parameter:
 1. Feed-Forward: Berechne \hat{y} (prädizierter Wert).
 2. Berechne Loss-Funktion $J(w)$ für w und b .
 3. Berechne Gradienten, d.h. ziehe von w und b ein kleines Delta $\eta * w$ und $\eta * b$ ab → **Graduelles Absteigen!**
Dieses η ist die **Lernrate**, ein wichtiger **Hyperparameter**.
3. Gehe zu 2 solange bis sich Loss nicht mehr signifikant ändert.

Bildlich veranschaulicht: Stellen Sie sich vor, daß Sie vom Gipfel eines Berges zum tiefsten Punkt laufen wollen.

Ziel Algorithmus: iteratives Wählen von Werten für Gewichte und Bias, um Cost-Funktion so weit wie möglich zu minimieren. Ein iteratives Verfahren ist weniger rechenaufwendig als die Alternative Einsetzen aller Parameter.

FALLBEISPIEL KLASSIFIKATION ANHAND GROSSER DATENSÄTZE ZEIGT VORTEILE NEURONALER NETZE.

Zufällig generiertes Dataset mit 50'000 Zeilen und 20 Features:

Algorithmus	Genauigkeit	Dauer Training (ohne GPU)
Logistische Regression	0.938	296 ms
XGBoost	0.966	5.86 s
Neuronales Netz (nur 20 Epochen)	0.968	1 min 11 s

Zufällig generiertes Dataset mit 500'000 Zeilen und 30 Features:

Algorithmus	Genauigkeit	Dauer Training ohne GPU/ mit GPU
Logistische Regression	0.90	2,4 s/ 2,14 s
XGBoost	0.913	1,46 min/ 1,33 min
Neuronales Netz (nur 20 Epochen)	0.955	14 min/ 22 min

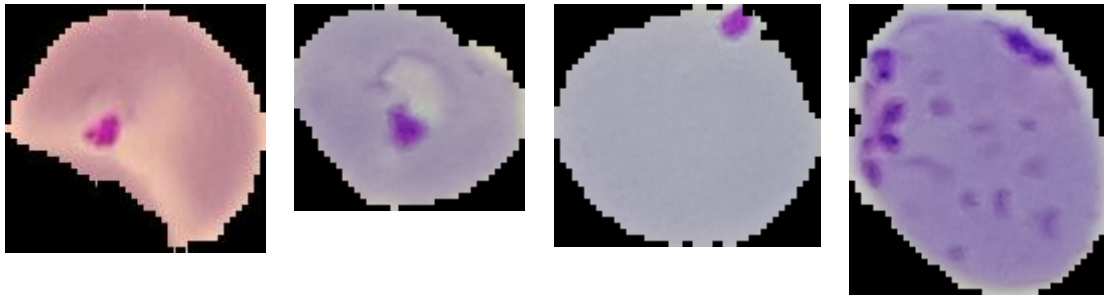
Bei großen bis sehr großen Datenmengen bietet neuronales Netz sehr gute Performance bei hoher Trainingszeit



2 GEFALTETE NEURONALE NETZWERKE (CNN)

CONVOLUTIONAL NEURAL NETWORK: MOTIVATION.

Bei einer Klassifikation von Bildern haben wir 2 grundlegende Probleme:



Gleiche Objekte, aber verschiedene Darstellungen.
Die bisher vorgestellten Verfahren würden die Positionen einzelner Pixel lernen, was nicht funktionieren würde.



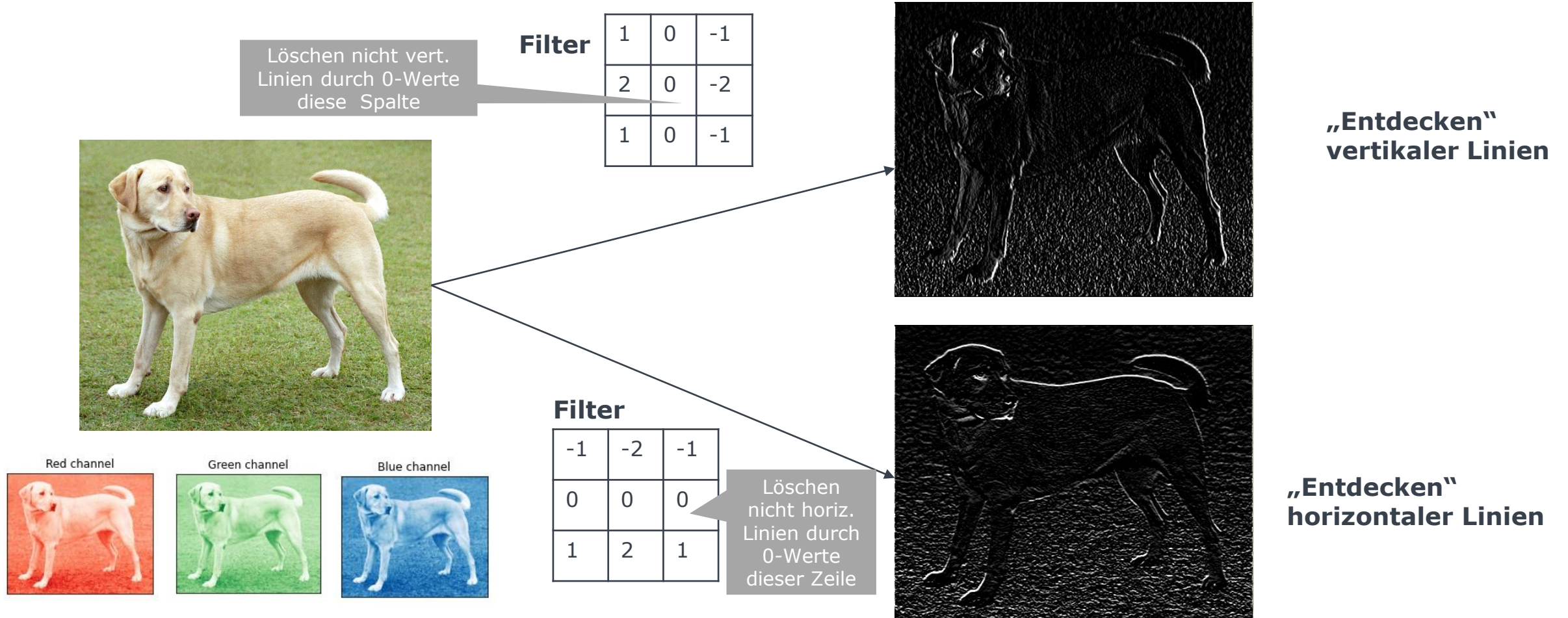
1. Bilder können sich stark unterscheiden:
 - Größe
 - Ausrichtung: Quer- vs. Hochformat, gedrehtes Bild
 - Farben
 - Belichtung
 - ...
2. Bilder sind enorm groß verglichen mit Text und Zahlen:
Kamerabilder haben 16 Mio. Punkte!

Convolutional neural networks lernen Formen und Strukturen (Features) in Bildern.

CONVOLUTIONAL NEURAL NETWORKS – ZUM EINSTIEG LIVE DEMOS IM BROWSER.

- <https://poloclub.github.io/cnn-explainer/>
- <http://cs231n.stanford.edu/2022/>
- <https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html>
- <https://playground.tensorflow.org/>

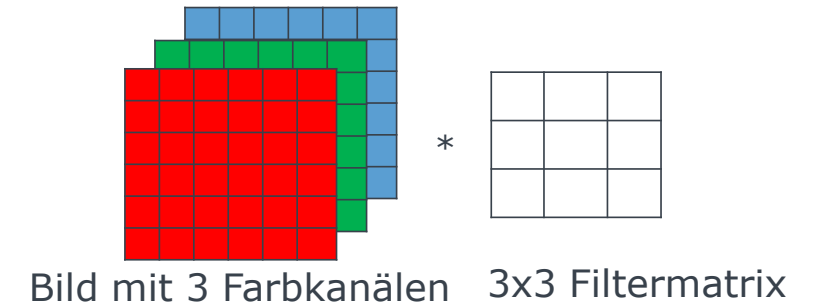
CNN SETZT FALTUNGEN FÜR ERKENNEN VON STRUKTUREN IN BILDERN EIN.



Die Faltung geschieht indem rollierend Zeile für Zeile jeweils mit einem Filter multipliziert werden

CONVOLUTIONAL NEURAL NETWORK. WIE FUNKTIONIERT DAS LERNEN?

Bild mit seinen 3 Farbkanälen Rot, Grün, Blau wird mit Filter multipliziert (Faltung).
Dimension Filter ist ungerade, um Symmetrien zu vermeiden.



Filter durchläuft analog Lupe Zeile für Zeile des Bildes.
Ergebnis wird in separater, verkleinerter Matrix gespeichert

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

Ziel ist es, durch mehrere kombinierte Faltungen nacheinander das Bild auf die wichtigsten Informationen zu reduzieren.

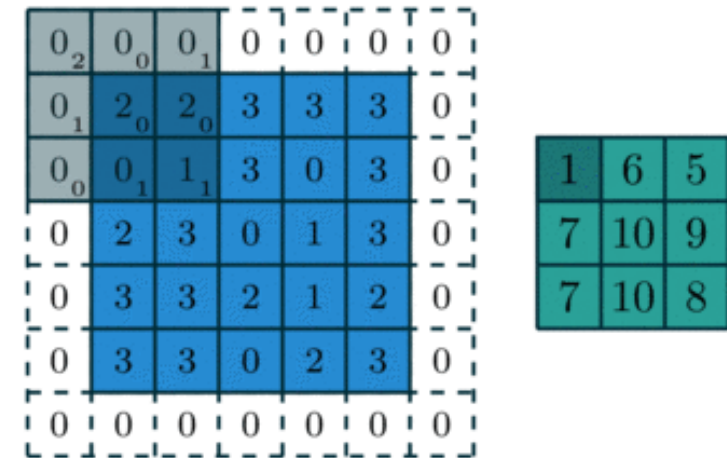
CONVOLUTIONAL NEURAL NETWORK. PADDING UND STRIDING

Stride: bestimmt die Schrittweite des Filters je Faltung.
Ermöglicht Reduzierung Daten (Kompression).

Padding: ist die Breite oder Höhe des Bildes nicht genau durch die Höhe oder Breite des Filters inklusive Schrittweite teilbar, bleibt Rest über.
Padding ermöglicht:

- **Padding = 0:** Rest abzuschneiden, Rand wird „ignoriert“ vom Filter.
- **Padding = n:** jeweils n Zeilen/ Spalten mit Zahl 0 werden an den Rand hinzugefügt, damit Rand berücksichtigt wird.

Frage: welche Schrittweite im Bild?
Welches Padding?



Padding mit 1x1 Grenzzone und Stride = 2

Padding und Stride sind Bestandteil jeder Faltung und helfen gesamthaft mit, ein Bild auf die wichtigsten Infos zu reduzieren.
Für die meisten Fälle genügen für Padding und Striding die Default-Werte.

CONVOLUTIONAL NEURAL NETWORKS: ZUSAMMENSPIEL DER FALTUNGSOPERATOREN.

<https://poloclub.github.io/cnn-explainer/>

CONVOLUTIONAL NEURAL NETWORK. POOLING.

1	4	3	12
8	9	7	4
3	9	2	1
15	13	3	7

Max-Pooling mit 2x2 Filter
und Stride = 2



9	?
?	7

Frage: was kommt
hier rein?

Pooling ermöglicht Reduzierung Daten, Entfernen von Rauschen und verhindert Overfitting.

CONVOLUTIONAL NEURAL NETWORK. DROPOUT.

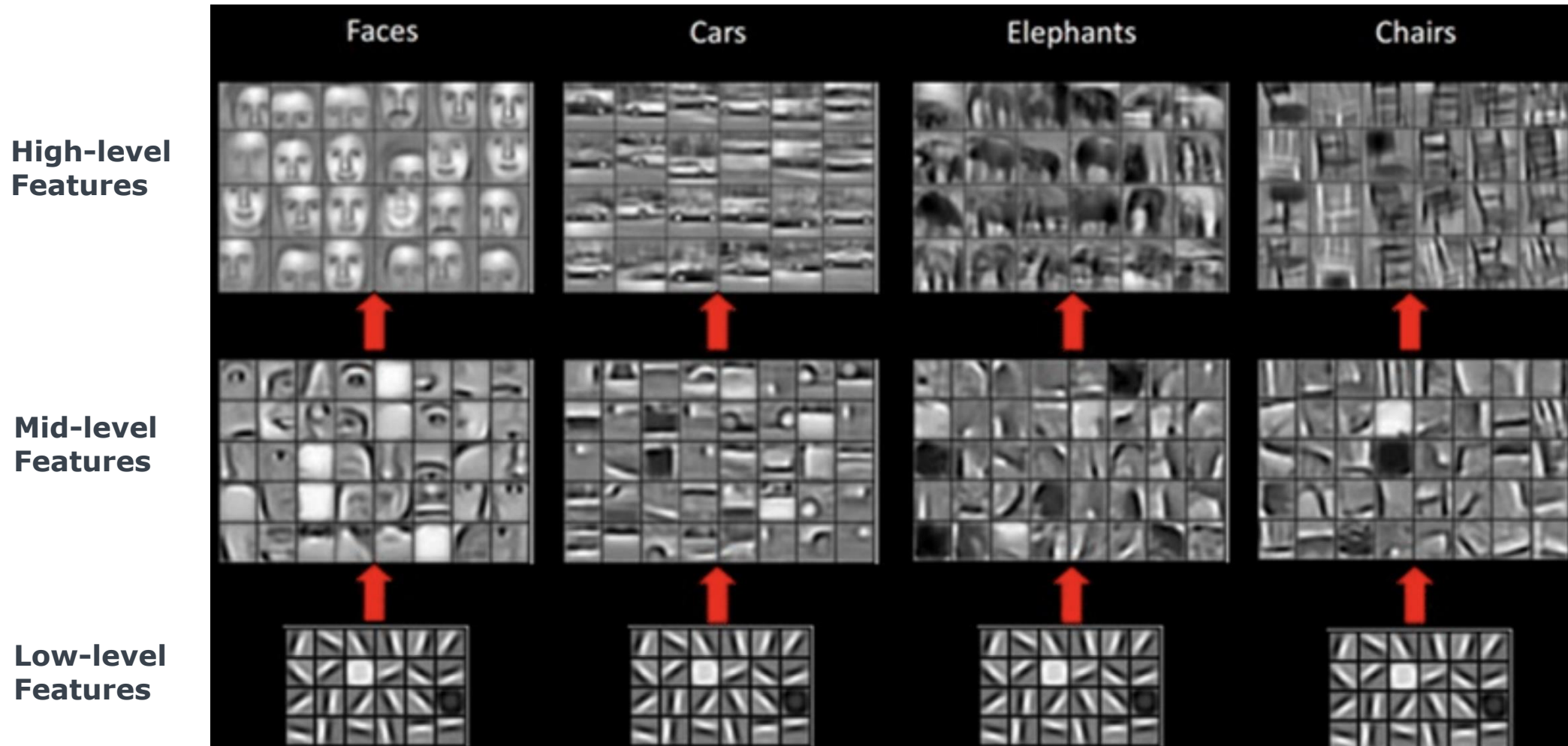


Fully-connected ANN und **CNN** neigen aufgrund Vielfalt an Parametern zu Overfitting

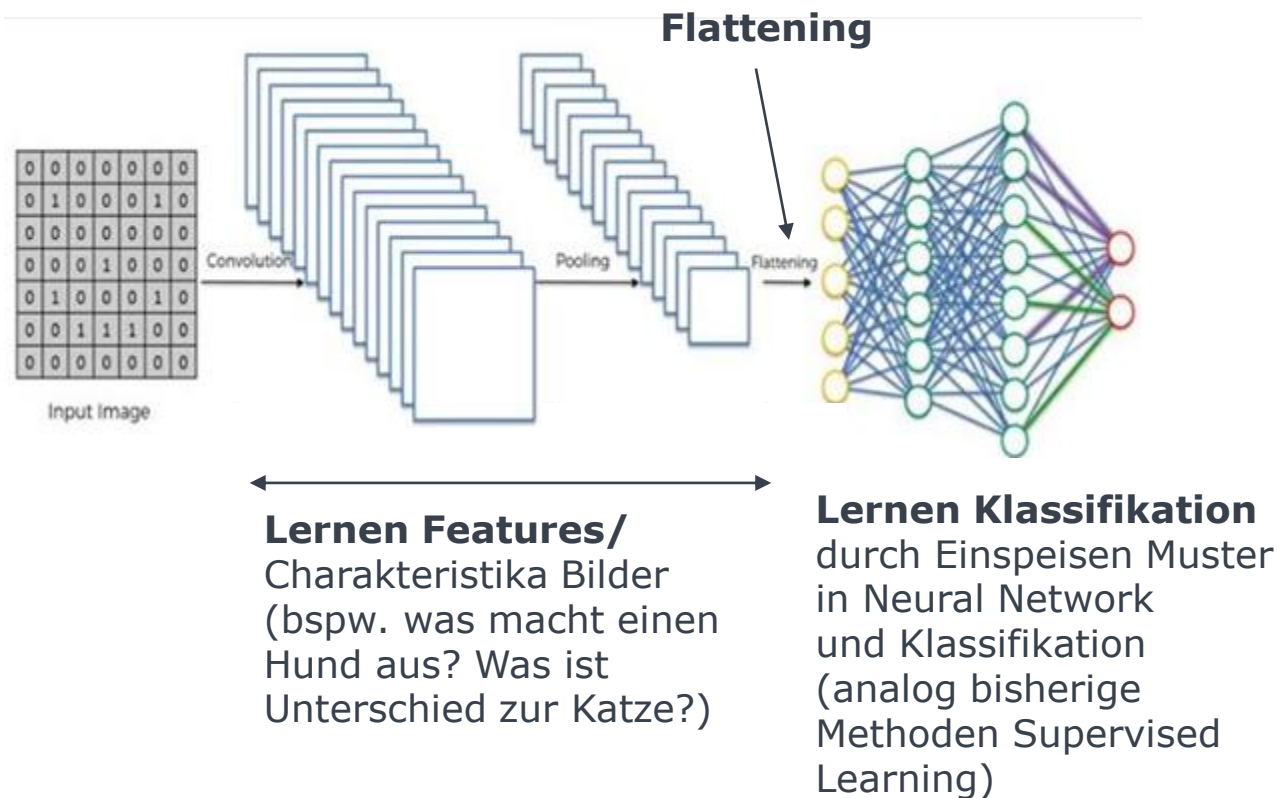
Dropout schaltet jeden Knoten mit Wahrscheinlichkeit $1-p$ als „passiv“. In jedem Trainingsdurchlauf werden andere Knoten passiv geschaltet (hier: 50% Knoten).

Dropout reduziert Overfitting (und beschleunigt das Training).

CONVOLUTIONAL NEURAL NETWORKS: KOMBINATION FALTUNGEN UND FEATURES ERMÖGLICHT LERNEN KOMPLEXER FEATURES.

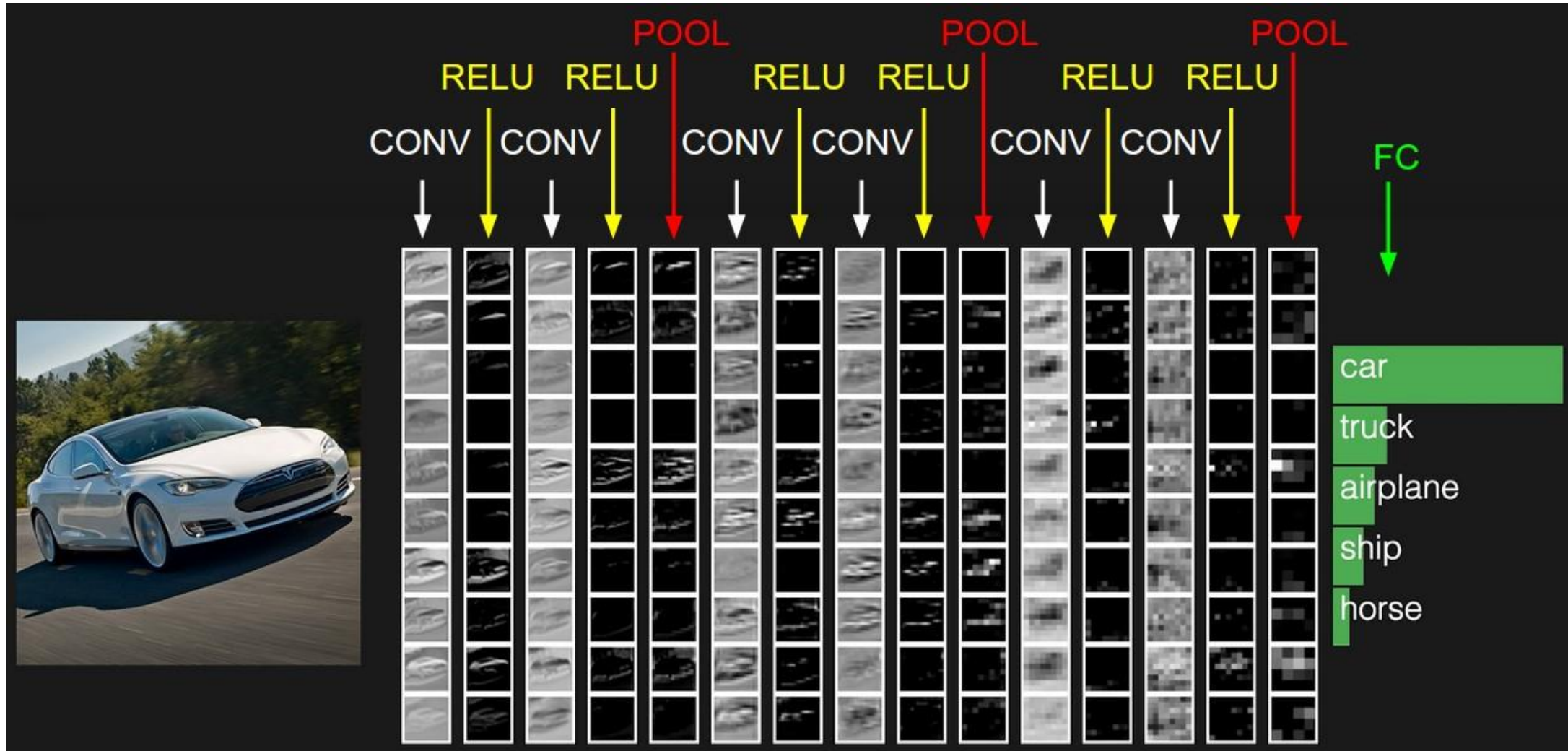


KOPPLUNG CNN MIT NEURONALEN NETZEN.



- Wir haben gesehen, wie Features in Bildern gelernt werden.
- Aber: für **Klassifizieren** Bilder fehlt noch **Output-Vektor**. Dies geschieht durch Verknüpfen der gelernten Features mit einem „einfachem“ neuronalen Netz.
- Mittels des **Flatten**-Operator werden die gelernten Features des Bildes als 1-dimensionaler Vektor abgespeichert.
- Weitere Vorgehensweise analog „einfachem“ neuronalen Netz oder den bisher gelernten Verfahren.

CONVOLUTIONAL NEURAL NETWORK: ÜBERSICHT GESAMTABLAUF.





3. TRANSFER LEARNING.

TRANSFER LEARNING.

Transfer Learning bedeutet **Übernahme** und ggf. **Anpassen** eines **bereits trainierten Modells**.

Vorteile:

- Schnell einsetzbar.
- Modelle wurden lang auf sehr großen Daten trainiert und haben somit validierte, gute Performance.

Nachteile:

- Nur für ähnliche Probleme einsetzbar.
- Anpassungen am Netz mit Vorsicht durchzuführen (vorgelernte Features können geändert/ gestört werden).

Einsatzgebiete:

- Bilderkennung
- Spracherkennung
- Übersetzung

ÜBERSICHT NETZE UND IHRER PERFORMANCE.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	0.790	0.945	22,910,480	126	109.42	8.06
VGG16	528	0.713	0.901	138,357,544	23	69.50	4.16
VGG19	549	0.713	0.900	143,667,240	26	84.75	4.38
ResNet50	98	0.749	0.921	25,636,712	-	58.20	4.55
ResNet101	171	0.764	0.928	44,707,176	-	89.59	5.19
ResNet152	232	0.766	0.931	60,419,944	-	127.43	6.54
ResNet50V2	98	0.760	0.930	25,613,800	-	45.63	4.42
ResNet101V2	171	0.772	0.938	44,675,560	-	72.73	5.43
ResNet152V2	232	0.780	0.942	60,380,648	-	107.50	6.64
InceptionV3	92	0.779	0.937	23,851,784	159	42.25	6.86
InceptionResNetV2	215	0.803	0.953	55,873,736	572	130.19	10.02
MobileNet	16	0.704	0.895	4,253,864	88	22.60	3.44
MobileNetV2	14	0.713	0.901	3,538,984	88	25.90	3.83
DenseNet121	33	0.750	0.923	8,062,504	121	77.14	5.38
DenseNet169	57	0.762	0.932	14,307,880	169	96.40	6.28
DenseNet201	80	0.773	0.936	20,242,984	201	127.24	6.67
NASNetMobile	23	0.744	0.919	5,326,716	-	27.04	6.70
NASNetLarge	343	0.825	0.960	88,949,818	-	344.51	19.96
EfficientNetB0	29	-	-	5,330,571	-	46.00	4.91
EfficientNetB1	31	-	-	7,856,239	-	60.20	5.55
EfficientNetB2	36	-	-	9,177,569	-	80.79	6.50
EfficientNetB3	48	-	-	12,320,535	-	139.97	8.77
EfficientNetB4	75	-	-	19,466,823	-	308.33	15.12
EfficientNetB5	118	-	-	30,562,527	-	579.18	25.29
EfficientNetB6	166	-	-	43,265,143	-	958.12	40.45
EfficientNetB7	256	-	-	66,658,687	-	1578.90	61.62

Erklärung Parameter:

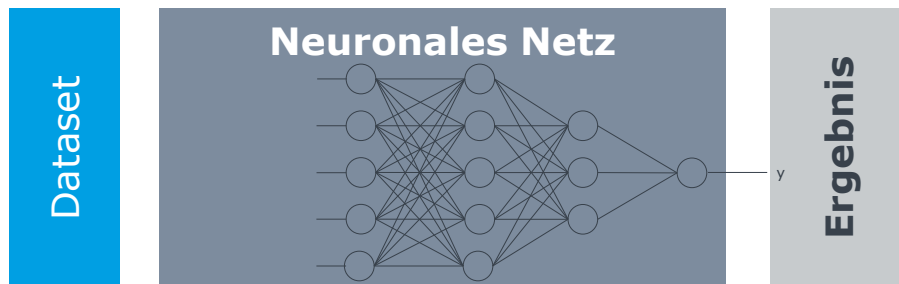
- **Size:** Größe des zu speichernden Modells. Relevant für den Einsatz auf Geräten mit beschränktem Speicherplatz wie z.B. Handys oder IoT-Geräten.
- **Top-1 Accuracy:** gibt an, wie oft auf Daten des ImageNet-Datensatz genau die richtige Klasse erkannt wurde.
- **Top-5 Accuracy:** gibt an wie oft auf Daten des ImageNet-Datensatz eine von 5 gleichzeitigen Vorhersagen des Modells die richtige war.
- **Parameters:** Anzahl Parameter. Relevant falls das Modell angepaßt wird, da dies ein Indikator für die benötigte Rechenleistung und somit Dauer des Trainings ist.
- **Depth:** Tiefe des Netzes mit allen seinen Schichten.
- **Time per inference step:** wie lang Netz für die Inferenz braucht (also Auswertung Input in Netz).

ImageNet-Wettbewerb ist
Benchmark für Bildererkennung

Kein Modell ist perfekt, Wahl ist abhängig vom jeweiligen Einsatzgebiet und Ressourcen

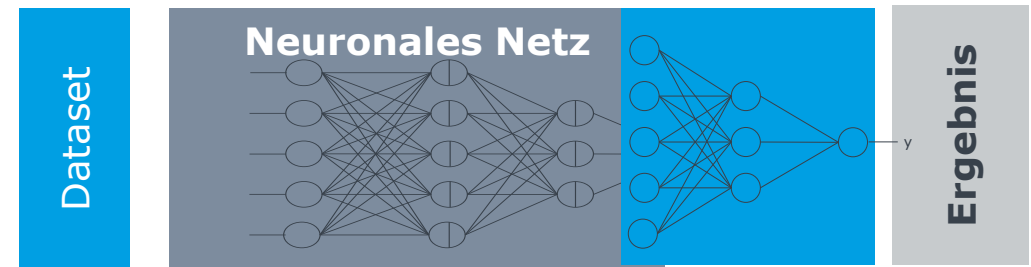
TRANSFER LEARNING: WIE SETZEN WIR ES EIN?

Möglichkeit A: Direktes Verwenden Netz



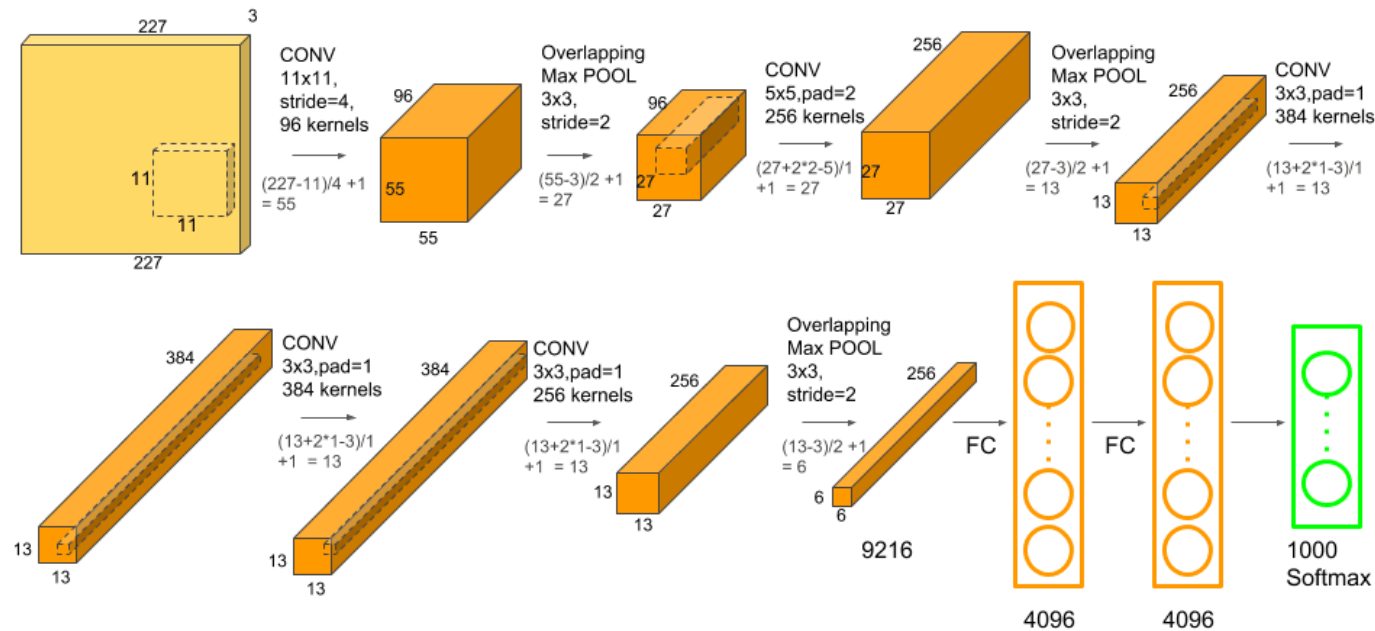
Einsatz für Standardanwendungsfall,
keine Änderungen notwendig.

Möglichkeit B: Anpassen/ Ersetzen spezifischer Netzumfänge



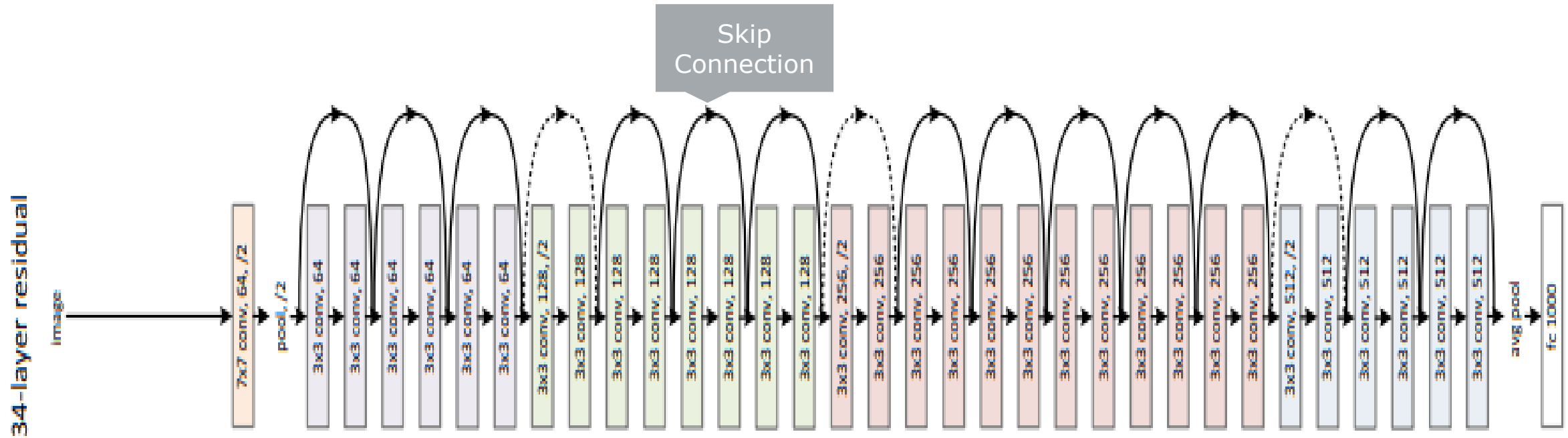
Änderungen notwendig, bspw. aufgrund geänderter Zielklassen.
Angepasst werden die letzten Schichten des Modells.

BEISPIEL FÜR TRANSFER LEARNING: ALEXNET IN 2012.



- Eines der einflußreichsten Paper in Computer Vision, ca. 70 000 mal zitiert.
- „Startschuß“ für verstärkten Einsatz GPU.
- Aber: Dan-Net nutzte vorher schon GPU, und war mit seinen Weiterentwicklungen auch besser als AlexNet. Es ist aber weniger bekannt...

BEISPIEL FÜR TRANSFER LEARNING: RESNET- 2015.



In neuronal Netzen kann sich für tiefe Layer die Trainingsgenauigkeit wenig verbessern oder gar verschlechtern. **Skip Connections** ermöglichen ein „Überspringen“ von Layern und ermöglichen dem Modell das Lernen einer „Identitätsfunktion“, sodaß höhere Layer nicht schlechter performen als tiefere Layer.



4. CASE STUDY

CASE STUDY IN GRUPPENARBEIT

1. Erstellen Sie ein Notebook in Google Colab.
2. Laden Sie die Datei CNN mit Fashion MNIST-v2.ipynb (liegt auf Github).
3. Probieren Sie verschiedene CNN-Strukturen aus: wie ändert sich dabei die Genauigkeit des Modells?
 - Passen Sie die bestehenden Conv2D-Faltungen an bzgl. Anzahl Filter, Kernel, Padding, Aktivierungsfunktionen.
 - Bauen Sie eine/ mehrere neue Faltungen (Conv2D) ein.
 - MaxPooling2D: Experimentieren Sie mit der Größe des Pooling.
 - Dropout: Experimentieren Sie mit dem Prozentwert.
 - Passen Sie die bestehende Dense-Layer an.
 - Bauen Sie neue Dense-Layer ein.
4. Probieren Sie verschiedene Hyperparameter aus: wie ändert sich dabei die Genauigkeit des Modells?
 - BATCH_SIZE
 - EPOCHS
 - LEARNING_RATE
 - Optimizer: Adam, RMSProp, Adagrad, Stochastic Gradient Descent.
5. Stellen Sie Ihre Ergebnisse, Hypothesen und Plots vor.

30 – 45 Minuten

LITERATUR UND WEITERE QUELLEN (AUSZUG).

Künstliche Intelligenz:

- Gröner, Heinecke: Kollege KI
- Burkov: The Hundred-Page Machine Learning Book, online verfügbar unter [Link](#)
- Nielsen: Neural Networks and Deep Learning, online verfügbar unter [Link](#)
- Russel, Norvig: Artificial Intelligence – a modern approach
- Pumperla, Ferguson: Deep Learning and the Game of Go
- Produktentwicklung mit AI:
 - Ameisen: Building Machine Learning Powered Applications: Going from Idea to Product
 - Ng: Machine Learning Yearning, online verfügbar unter [Link](#)

Kostenfreie Online-Kurse (bei Interesse):

- Python-Kurse
 - Python for Everybody ([Link](#))
 - Udacity Python Course ([Link](#))
 - **Coursera Course Deep Learning** ([Link](#))
 - FAST AI ([Link](#))

Web-Links:

- Dive into Deep Learning: <https://d2l.ai/>
- CNN: <https://cs231n.github.io/convolutional-networks/>
- CNN: <https://medium.com/machine-learning-researcher/convlutional-neural-network-cnn-2fc4faa7bb63>
- CNN: <https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491>
- CNN: <https://www.wandb.com/articles/fundamentals-of-neural-networks>