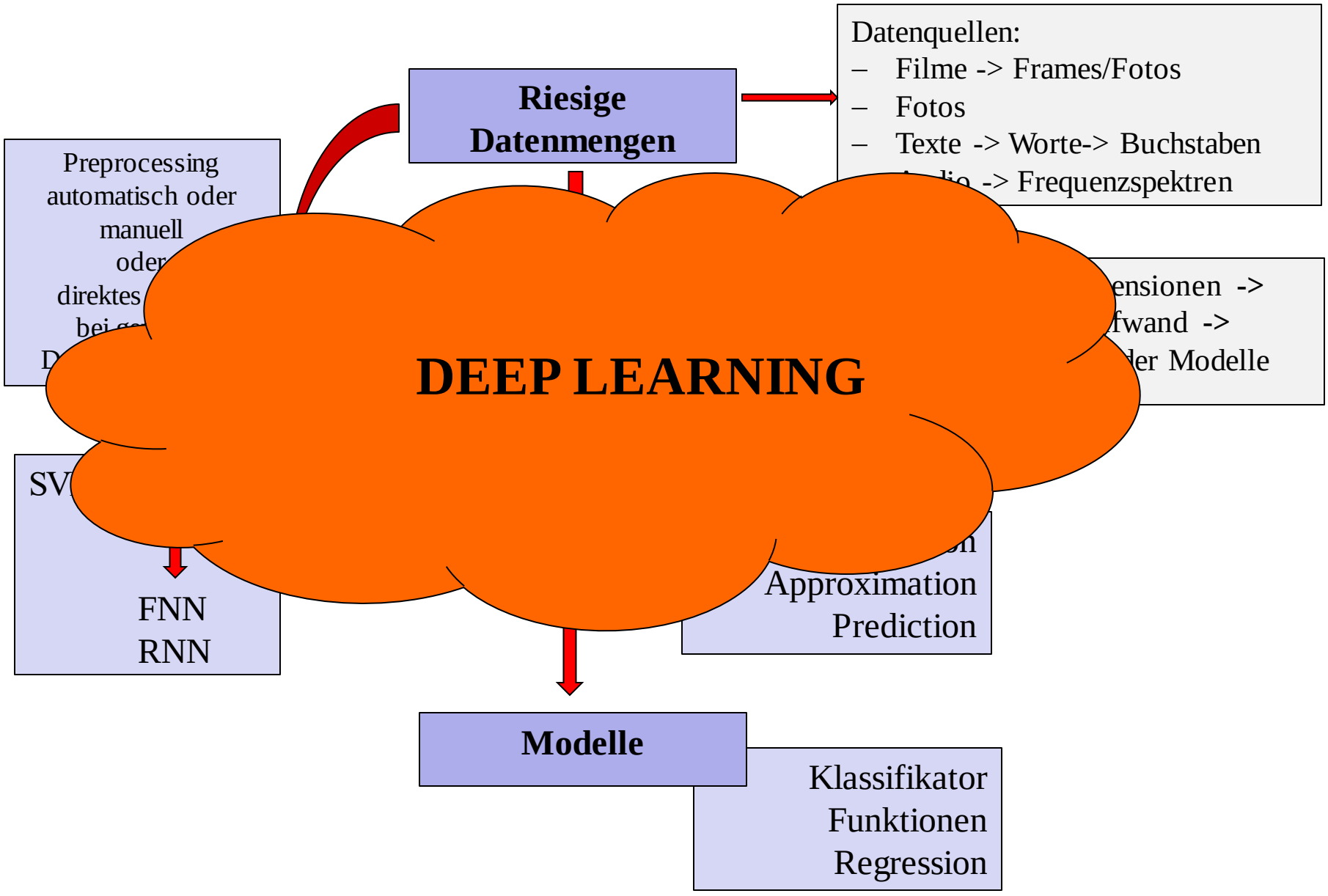
Deep Learning
Teil 1 - CNN

Übersicht

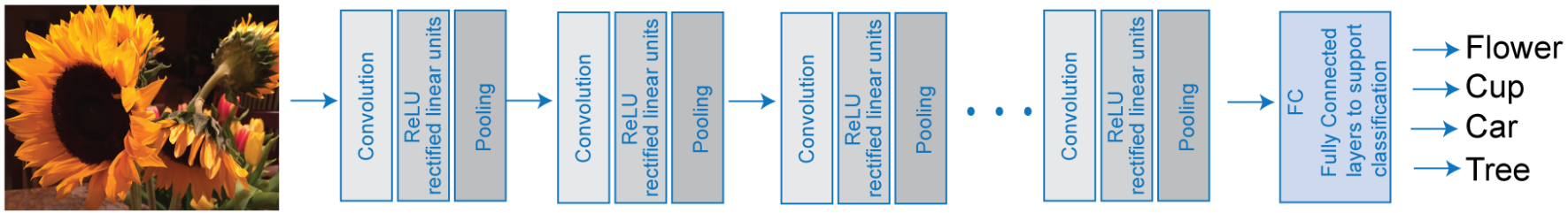
- Einführung und Motivation
- Convolutional Neural Networks (CNNs)
- Weitere Einsatzmöglichkeiten CNNs
- Berühmte Repräsentanten CNNs
- Kritikpunkte Deep Learning allgemein

Machine Learning vs. Datenmengen



Was ist Deep Learning?

- Deep Learning ist eine Art maschinelles Lernen, bei dem ein Modell meistens Klassifizierungsaufgaben direkt aus Bildern, Text oder Sound auszuführen lernt
- Deep Learning wird normalerweise unter Verwendung neuronaler Netzwerkarchitekturen implementiert
- Herkömmliche neuronale Netzwerke enthalten nur 2 oder 3 „hidden layers“, während tiefe Netzwerke über 100 enthalten können. Je mehr versteckte Ebenen, desto tiefer (deeper) das Netzwerk.
- Neuartige Netzwerkarchitekturen werden eingeführt:
  - Convolutional Networks
  - LSTM’s (Long Short Term Memory) als Erweiterung zu rekurrenten Netzwerken

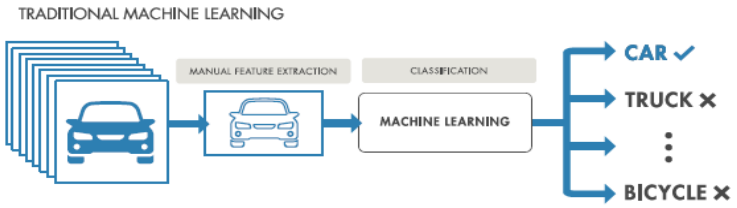


convolutional network

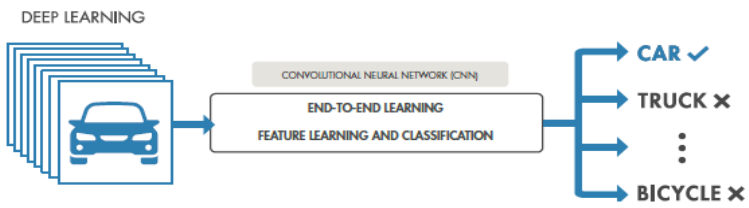
Unterschied zwischen Deep Learning und Machine Learning

- Deep Learning ist eine Unterart des maschinellen Lernens.
  - Beispiel: Beim maschinellen Lernen werden relevante Merkmale eines Bildes manuell extrahiert.
- Mit Deep Learning werden unbearbeitete Bilder direkt in ein tiefes neuronales Netzwerk eingespeist, das die Features automatisch lernt.
- Deep Learning erfordert oft Hunderttausende oder Millionen von Bildern, um gute Ergebnisse zu erzielen.
  - Rechenintensiv, leistungsstarke GPUs erforderlich

Unterschied zwischen Deep Learning und Machine Learning



Manuelles oder automatisches Festlegen der Features: Typ, Anzahl



Features werden automatisch gelernt

Machine Learning	Deep Learning
+ Good results with small data sets	— Requires very large data sets
+ Quick to train a model	— Computationally intensive
— Need to try different features and classifiers to achieve best results	+ Learns features and classifiers automatically
— Accuracy plateaus	+ Accuracy is unlimited

Deep Learning – warum gerade jetzt?

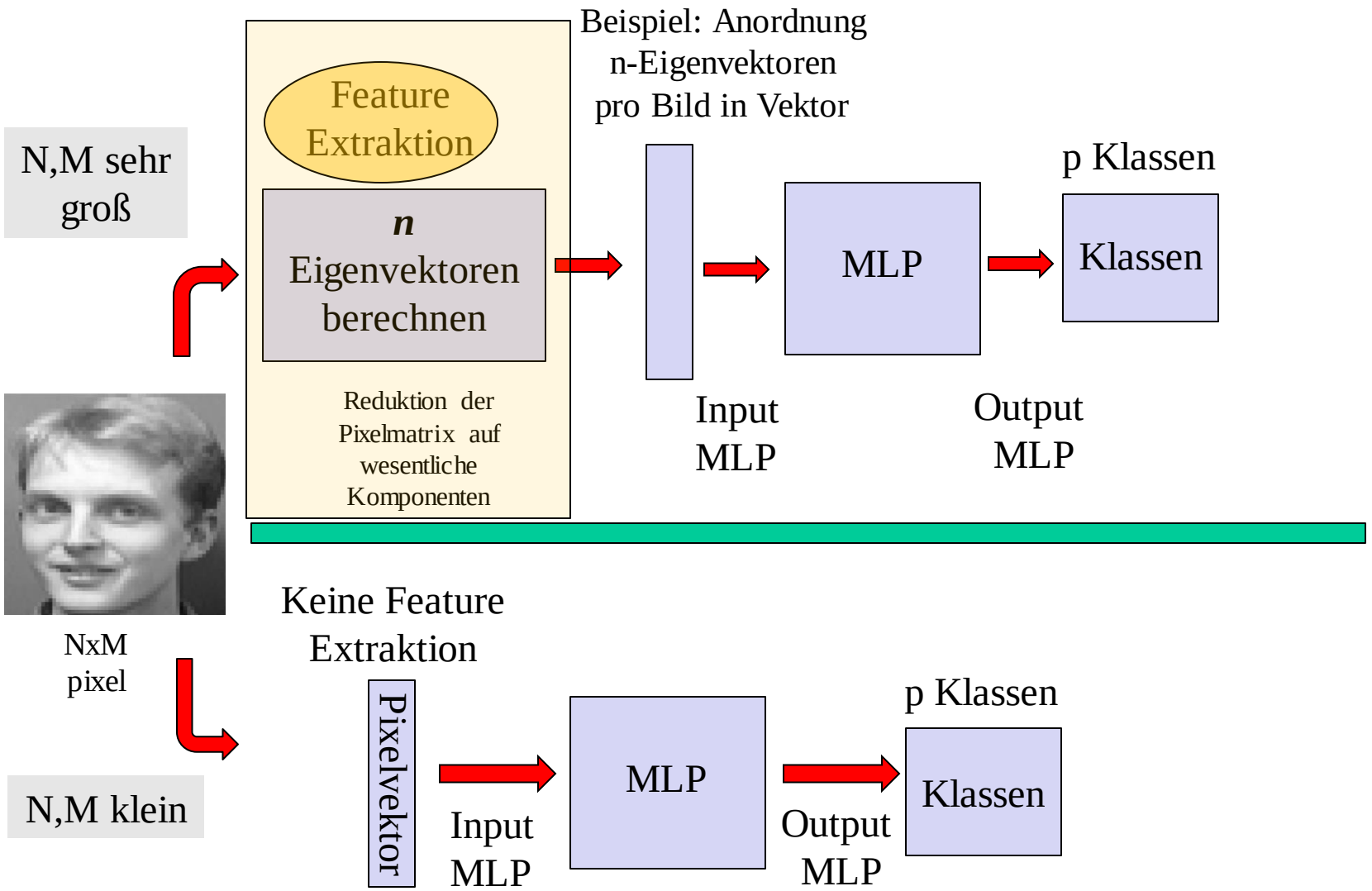
- **Große, qualitativ hochwertige Datensätze vorhanden**
  - Daten selbst erklären wahrscheinlich die meisten Verbesserungen
- **Massive Parallelverarbeitung mit GPUs**
  - neuronale Netze eigentlich nur ein Bündel von Gleitkommaberechnungen, die parallel mit GPUs durchgeführt werden können
  - Übergang von CPU-basiertem Training zu GPU-basiertem hat zu enormen Geschwindigkeitsverbesserungen für die Berechnung dieser Modelle geführt
- **Backpropagation-freundliche Aktivierungsfunktionen**
  - Übergang weg von Aktivierungsfunktionen wie **tanh** und **sigmoid** zu neunen Aktivierungsfunktionen wie **ReLU** und **softmax** haben das verschwindende Gradientenproblem gemildert
- **Verbesserte Architekturen**
  - CNNs und LSTMs halten die Gradienten im Fluss und erhöhen die Tiefe und Flexibilität des Netzwerks
- **Software Plattformen**
  - Frameworks wie Tensorflow oder Theano bieten automatisches Parallelisierung für GPU-Computing und machen Prototyping schneller und weniger fehleranfällig
- **Neue Regularisierungstechniken**
  - Techniken wie *dropout*, *batch normalization* und *Datenerweiterung* ermöglichen große und größere Netzwerke ohne (oder mit weniger) Overfitting zu trainieren

Kritikpunkte Deep Learning

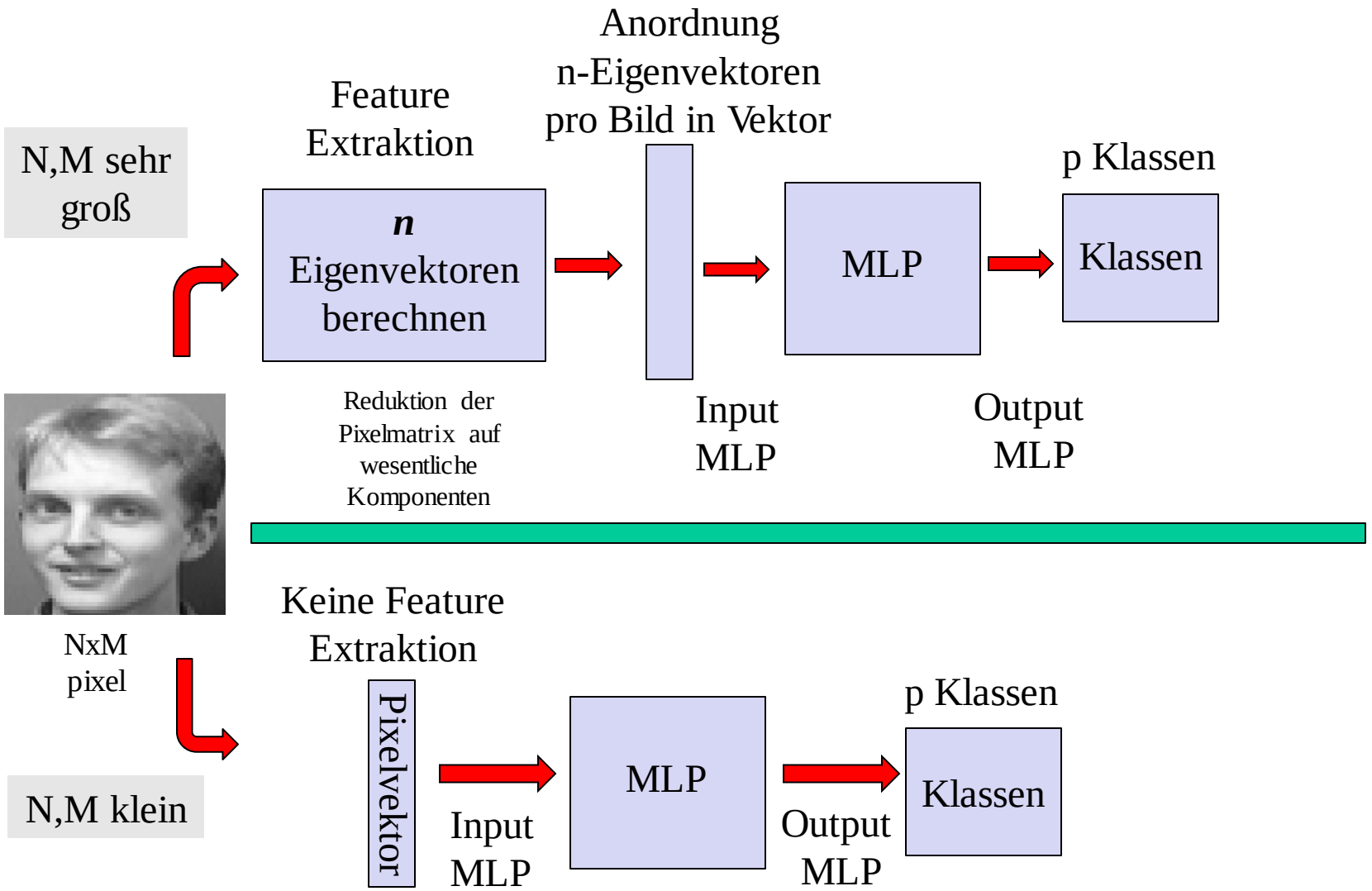
- **DL ist datenhungrig.** Anders als Menschen benötigt DL Millionen von Trainingsbeispielen, um richtige Klassifizierungen durchführen zu können, sonst Overfitting
- **DL-Modellen fehlen starke Transferlernfähigkeiten**: Übertragen der Resultate eines Trainingslaufes einer Aufgabe auf eine andere Aufgabe beschränkt möglich.
- **Einbinden von Vorwissen in DL-Modellen ist extrem schwierig:** menschliche Gehirne verfügen über Instinkte, die ihnen helfen, schnell zu lernen. Forscher sind besessen davon, DL-Modelle direkt aus Daten ohne Einbinden von Vorwissen lernen zu lassen.
- **DL Modelle basieren auf „supervised Learning**: viele AI Aufgaben wären „unsupervised“.

Deep Learning: A Critical Appraisal

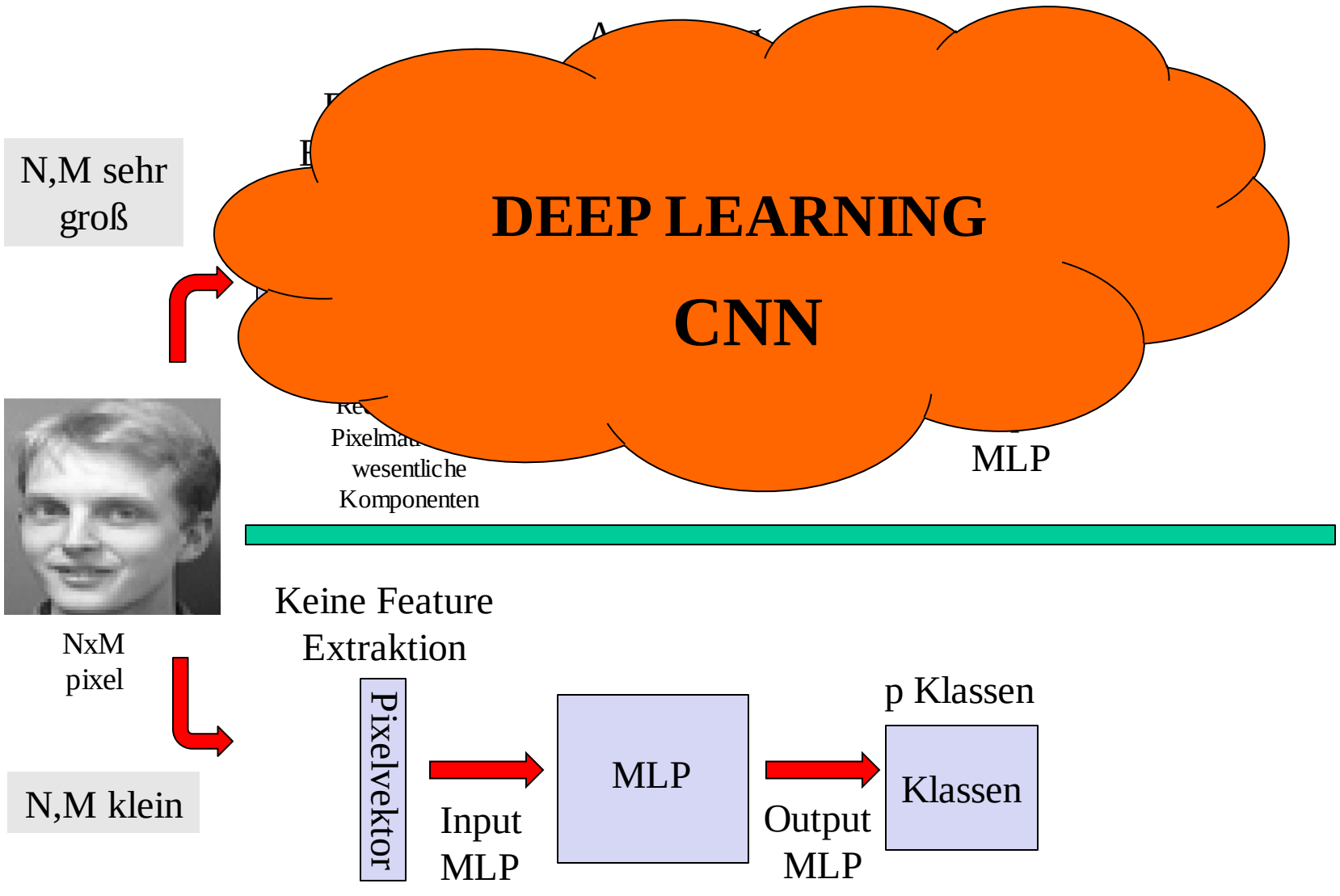
Beispiel Face Recognition



Beispiel Face Recognition



Beispiel Face Recognition



Beispiel: Deep Learning zur Objekterkennung in Fotos

Beispiel:

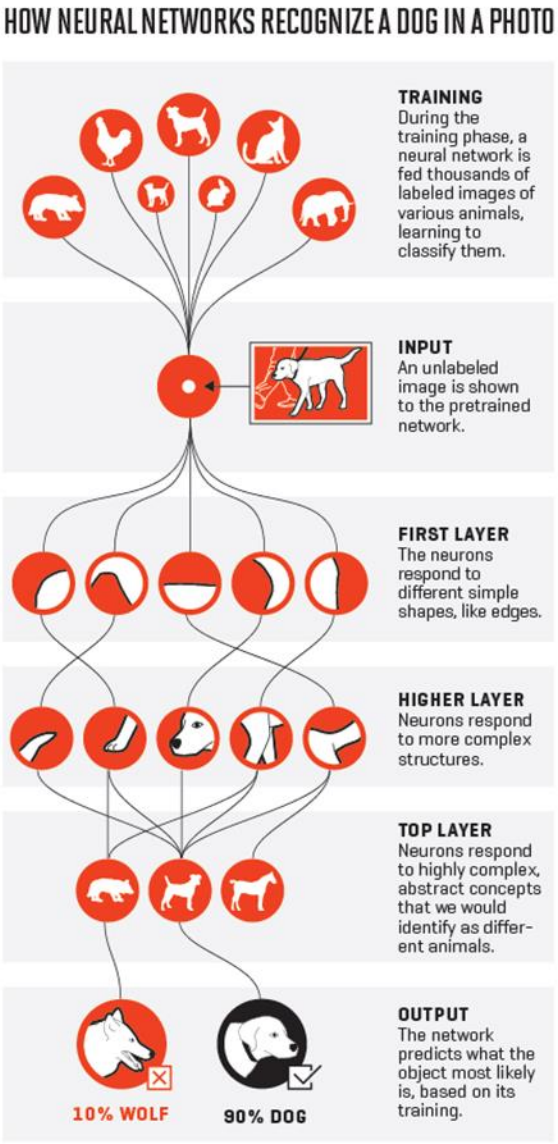
Finde 200 Objektklassen (Stuhl, Tisch, Person, Fahrrad,..) in Fotos mit ca. 500 x 400 Pixel Auflösung und 3 Farbkanälen (RGB)

Möglicher Ansatz mit klassischem MLP:

- Konvertiere Bildmatrix mit drei Farbkanälen in Vektordarstellung ohne Featureextraktion
- MLP mit **600.000** Neuronen am Inputlayer und **200** Neuronen im Outputlayer, ergibt **120 Millionen** Gewichte, die beim Training anzupassen sind.
- Gewichte der Neuronen der Hidden Layer sind nicht miteingerechnet!

CNN - Convolutional Network (Faltungsnetzwerk):

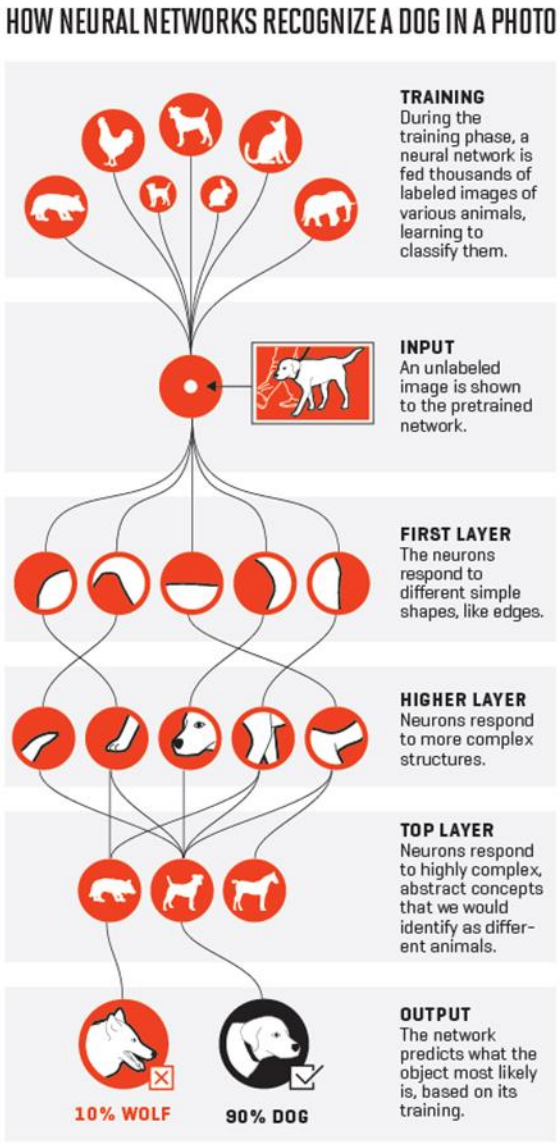
- Sonderform des MLPs, state-of-the-art-Methode für zahlreiche Anwendungen im Bereich der Klassifizierung von Objekten auf Fotos.
- Lernt Features zur Objekterkennung während des Trainings selbstständig
- Grundlage ist die Annahme, dass die Features eines Objektes viele räumliche Beziehungen zwischen benachbarten Pixeln besitzen, die unabhängig von ihrer Lage auf dem Foto sind. Es wird nur die Nachbarschaft der Features zueinander gelernt (spatial invariance).
- Nach dem Training werden die Features eines Objekts erkannt, unabhänigig von deren Position, Drehung, Stauchung oder Dehnung – automatische Featureextraktion
- Extrahierte Features für MLP -Klassifikation benutzen



Convolutional Network:
Feature and Pattern Recognition Learning

Feature Learning:

- Feature Learning Algorithmen lernen Muster, die wichtig für die Unterscheidung der einzelnen Objekt-Klassen sind.
- Convolutional Networks finden während des Trainings diskriminiernde Features eines Objekts, die von einem Convolutional Layer zum nächsten immer komplexer werden.
- Beispiel: Kurven/Kanten -> Nase, Augen, Wange, Mund -> Gesicht
- Der letzte Layer (Fully Connected) des Convolutional Networks verwendet die gefundenen Features für die Klassifikation.
- Ein Convolutional Network verwendet zum Training Backpropagtion.

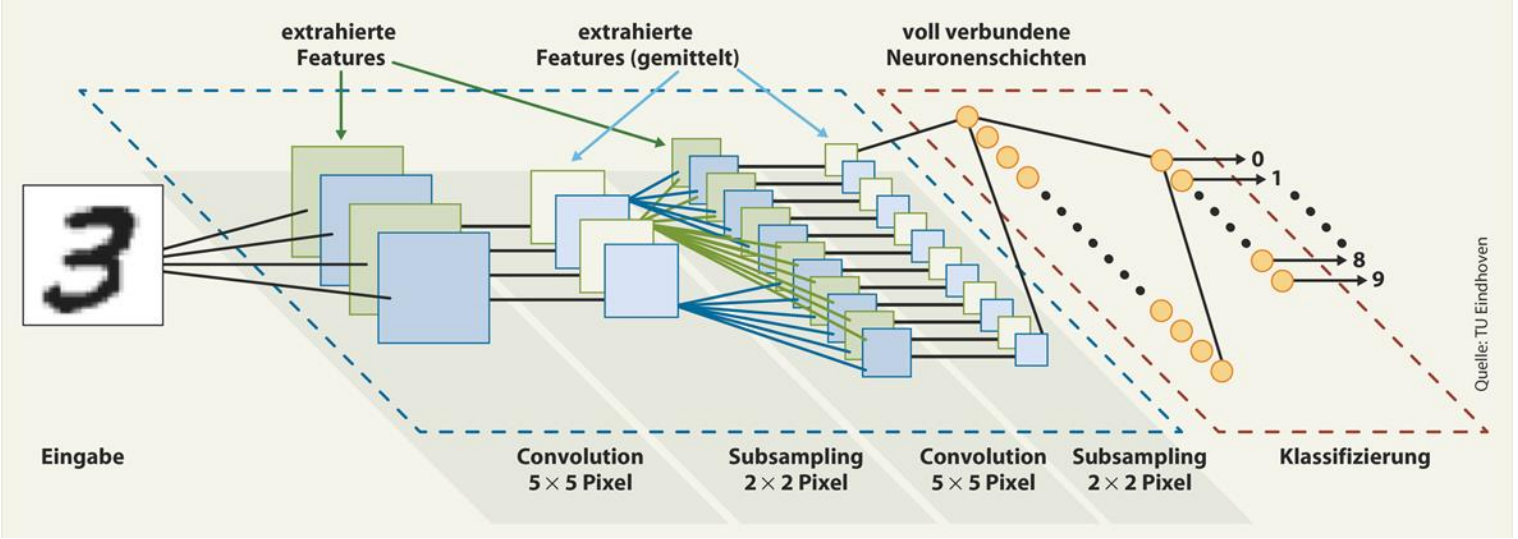


Struktur eines Convolutional Neural Network (CNN oder ConvNet)

- Convolutional Layer(s)
- Pooling Layer(s) (Subsampling)

} werden paarweise verwendet, können beliebig oft vorkommen

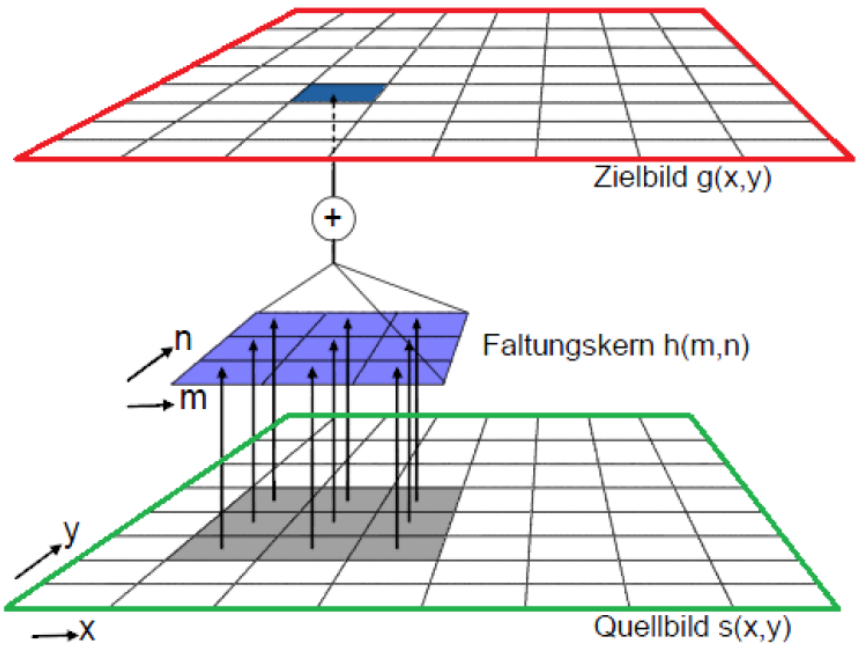
- Fully connected layer (immer letzter Layer, MLP mit einem oder mehrerer Hidden Layers für Klassifikation des Inputs)



Basis Convolutional Network: Bildbearbeitung: Faltung (Convolution)

- Eine Filtermatrix besteht meist aus einer quadratischen Matrix (Fenster, Maske, Kernel, Filter), die auf einer Bildkoordinate (x, y) zentriert und dann von Pixel zu Pixel über das ganze Bild verschoben werden kann.
- Jeder Zielbildwert wird aus dem entsprechenden Quellbildausschnitt durch Multiplikation und Summation berechnet.

- Anwendungsbeispiele:
  - Bildglättung
  - Bildschärfung
  - Kantenfilter



Beispiel Filter

Faltung (Convolution) Beispiel

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

image

W_{11}	W_{12}
W_{21}	W_{22}

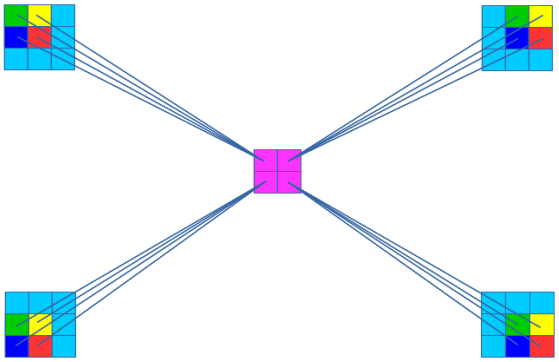
filter

h_{11}	h_{12}
h_{21}	h_{22}

feature map

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

W_{11}	W_{12}
W_{21}	W_{22}



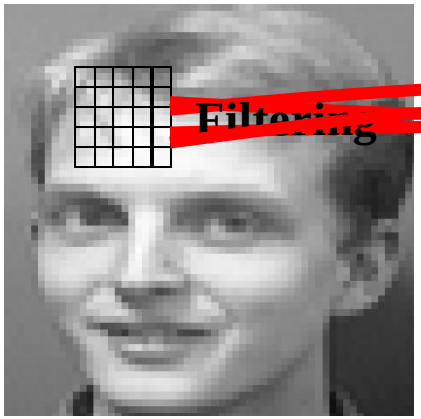
$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Basis Convolutional Network:
Bildbearbeitung: Faltung



gaussian

contrast

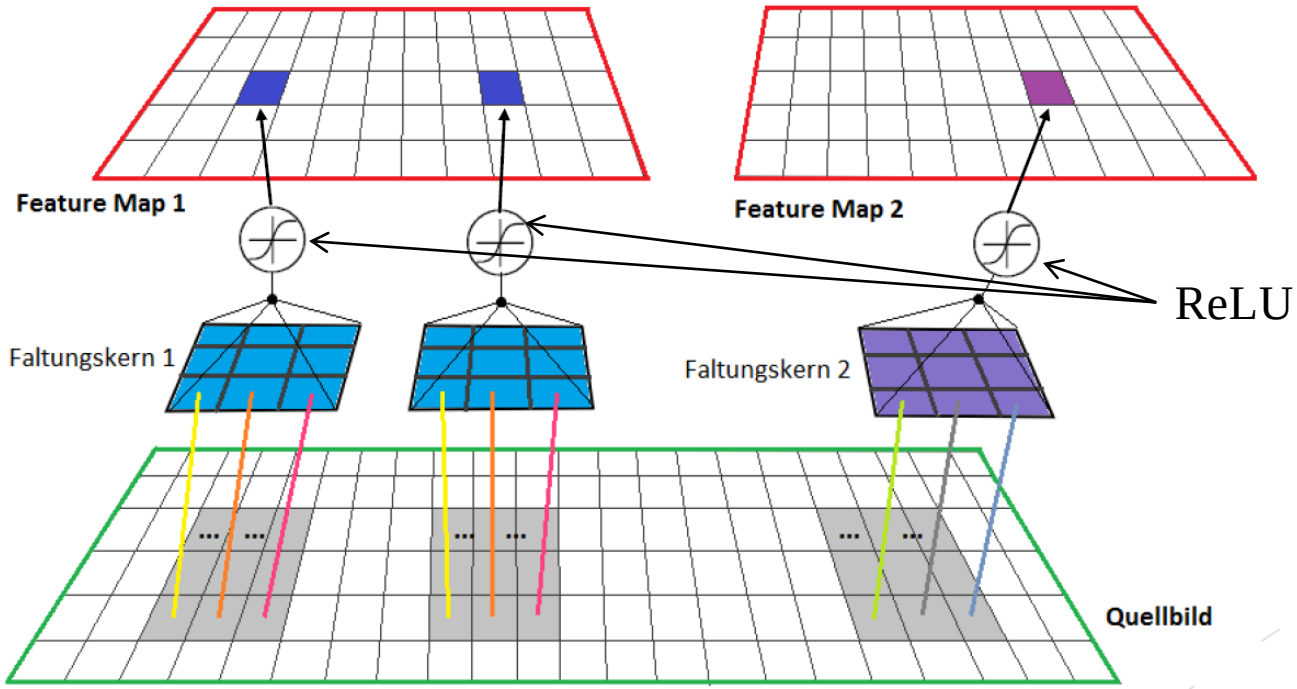
laplacian

prewitt

Image after activation of convolution layer

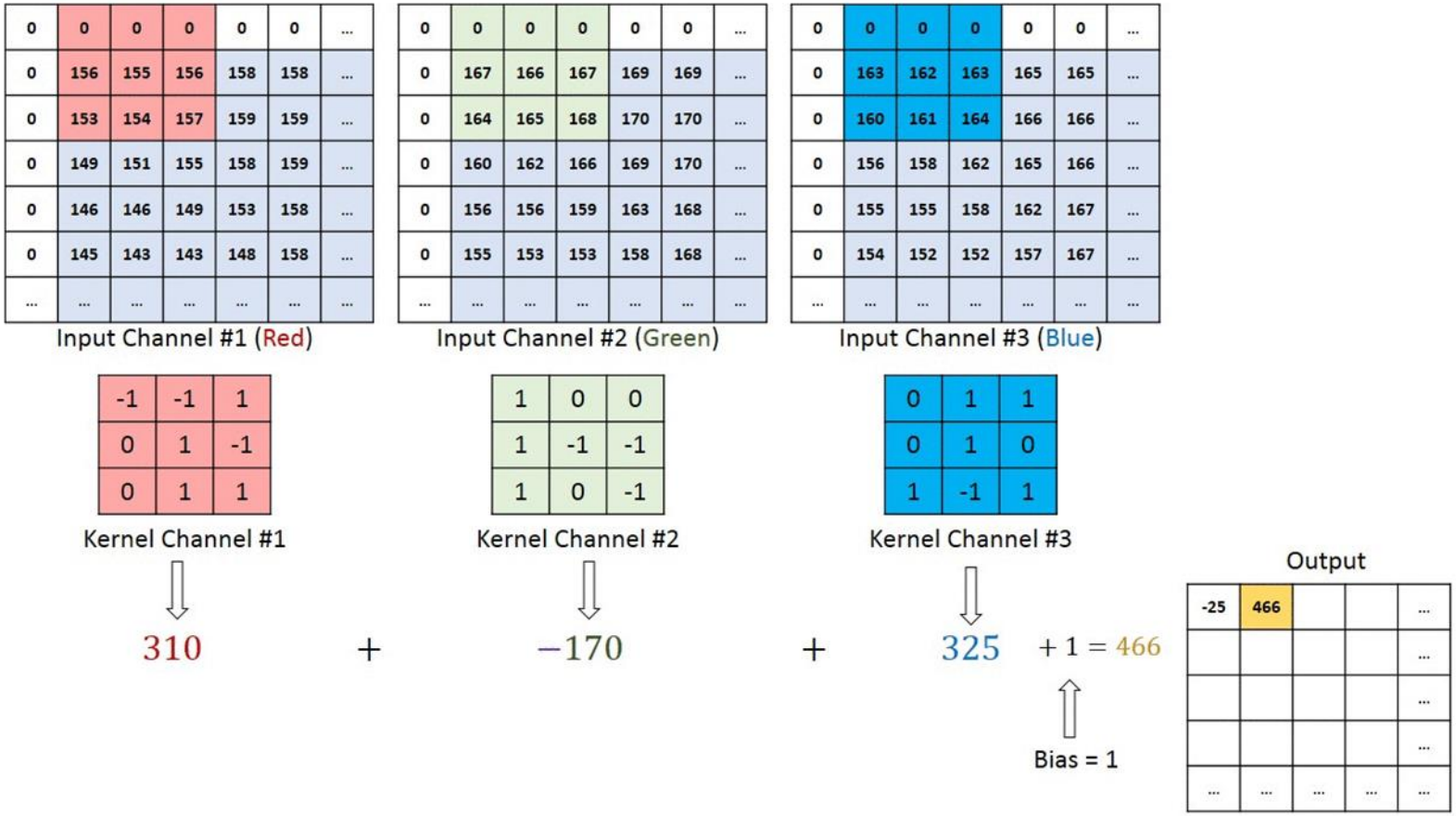
Convolutional Layer:
Faltung: Input Bild mit Graustufen

Meist werden mehrere unterschiedlich initialisierte Kernel verwendet, die unterschiedliche Outputs erzeugen. Diese Output Bilder werden auch als **Feature Maps** oder **Activation Maps** bezeichnet.



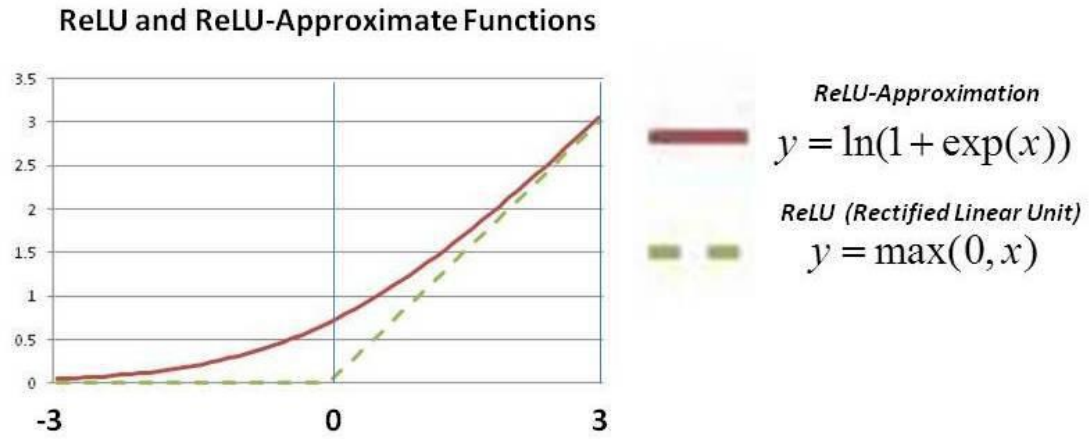
convolutional network – Feature Maps

Convolutional Layer:
Faltung: Input Bild RGB



Convolutional Layer: Faltung im Neuronalen Netz

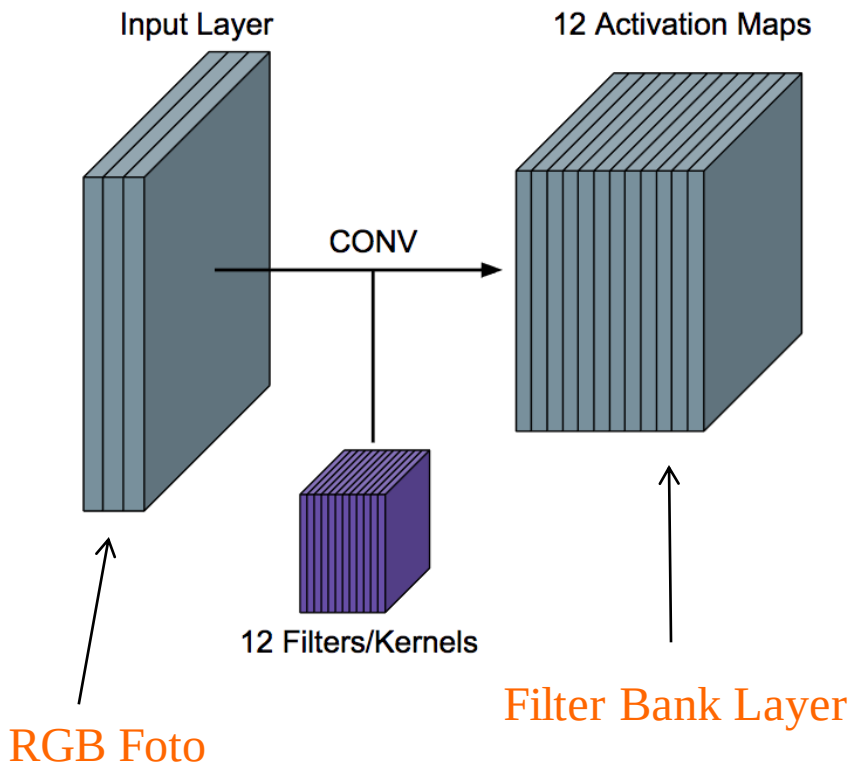
- Input entweder als zwei- oder dreidimensionale Matrix (Graubild oder RGB-Bild bzw. vergleichbarer Farbraum) bestehend aus Pixeln gegeben
- Aktivität jedes Neurons wird über eine diskrete Faltung (Kernel) berechnet, der Kernel wird über das ganze Bild bewegt. Jede Position des Kernels entspricht einem Neuron und berechnet sich als inneres Produkt des Kernels mit dem aktuell unterliegenden Bildausschnitt.
- Die Werte der Kernelmatrix werden mit Zufallszahlen initialisiert und beim Training erlernt. Die Gewichte der Neuronen sind pro Kernelmatrix konstant (gemeinsame Gewichte, englisch: shared weights)
- Der ermittelte Input eines jeden Neurons wird nun von einer Aktivierungsfunktion, bei CNNs üblicherweise ReLu in den Output verwandelt. Da Backpropagation die Berechnung der Gradienten verlangt, wird in der Praxis eine differenzierbare Approximation von ReLu benutzt.



Convolutional Layer: Feature Maps

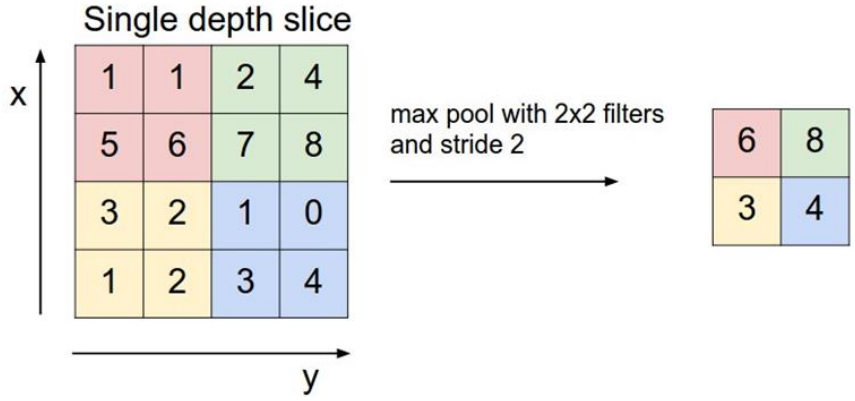
Filter Bank Layer:

- Das Netzwerk lernt eine Menge von L Faltungskernen
- Diese werden auf die überlappenden Quellbild-ausschnitte angewendet
- Dadurch entstehen L Feature Maps (=Activation Maps)
- Jede Feature Map besteht aus Neuronen, die:
  - dieselben Parameter (Gewichte und Bias) haben
  - dazu dienen, auf ein und dasselbe Feature zu reagieren
  - und zwar unabhängig von der Position im Quellbild!
- L Feature Maps bilden den Filter Bank Layer



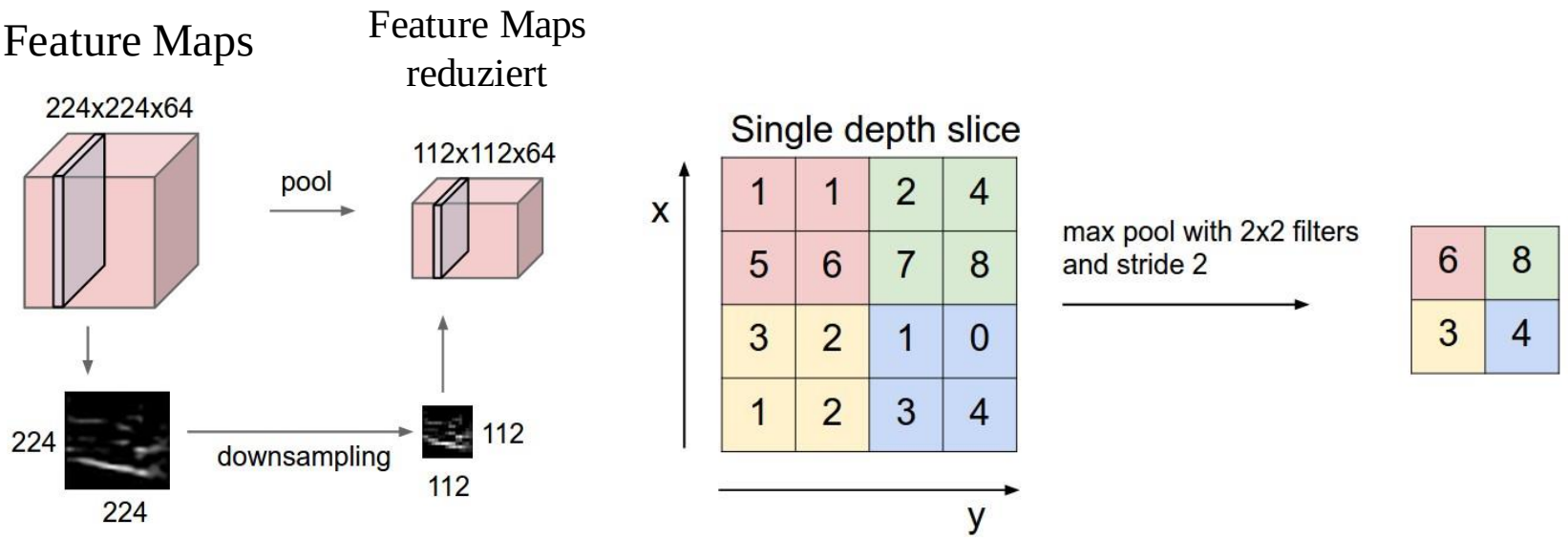
Pooling Layer: Datenreduktion

- Im nächsten Layer, dem Pooling, werden überflüssige Informationen verworfen.
- Zur Objekterkennung in Bildern ist die exakte Position einer Kante im Bild von vernachlässigbarem Interesse.
- Es gibt verschiedene Arten des Poolings. Mit Abstand am stärksten verbreitet ist das Max-Pooling, wobei aus jedem 2 × 2 Quadrat aus Werten der Feature Maps nur die Aktivität des aktivsten (daher "Max") für die weiteren Berechnungsschritte beibehalten wird, Großteil der Aktivität der Feature Maps wird verworfen.
- Trotz der Datenreduktion (im Beispiel 75 %) verringert sich in der Regel die Performance des Convolutional Networks nicht.



Max-Pooling mit einem 2×2-Filter und Schrittgröße = 2

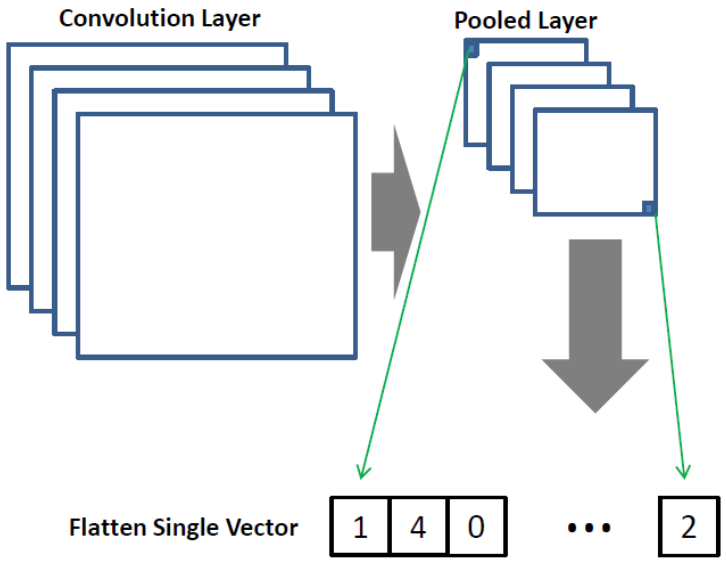
Pooling Layer
Datenreduktion



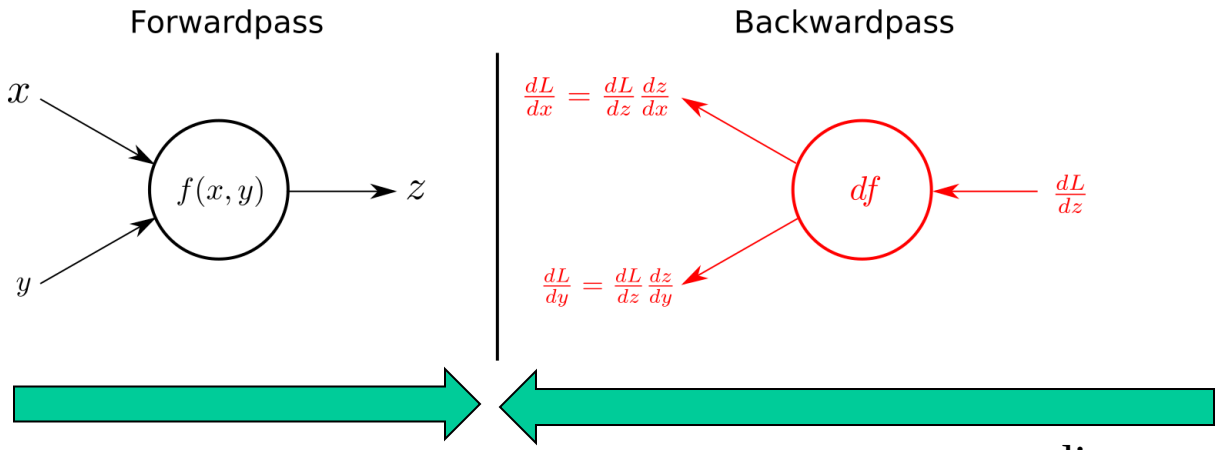
- Pooling wird auch als **„*Subsampling*** oder ***Downsampling*“** bezeichnet
- Jede Feature Map wird nun separat betrachtet
- Man berechnet typischerweise den mittleren oder maximalen Wert der Nachbarschaft, es werden keine Gewichte bzw. Neuronen verwendet
- Ergebnis = neue Feature Map
  - mit einer kleineren Auflösung
  - Robust zu kleinen Lageverschiebungen der Features

Fully Connected Layer: MLP

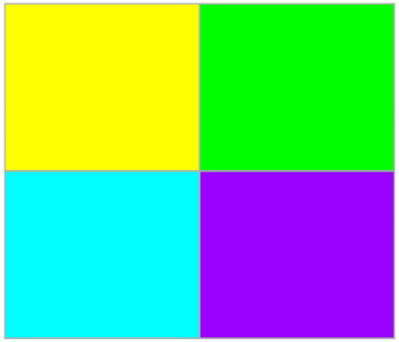
- Nach einigen sich wiederholenden Einheiten bestehend aus Convolutional und Pooling Layers wird das CNN mit einem (oder mehreren) **Fully-connected Layer(s) (=hidden layer)** entsprechend der Architektur eines MLPs abgeschlossen.
- Der letzte Pooling Layer wird in einen einzelnen Vektor umgewandelt, der als Input für das MLP dient (flattening), anschließend können weitere hidden Layer folgen.
- Die Anzahl der Neuronen im letzten Layer korrespondiert dann üblicherweise zu der Anzahl an Klassen, die das Netz unterscheiden soll. (One-Hot-encoding mit Softmax Output Funktion)



Convolutional Layer: Backpropagation



X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}



gradients

∂h_{11}	∂h_{12}
∂h_{21}	∂h_{22}

$$\partial W_{11} = X_{11}\partial h_{11} + X_{12}\partial h_{12} + X_{21}\partial h_{21} + X_{22}\partial h_{22}$$

$$\partial W_{12} = X_{12}\partial h_{11} + X_{13}\partial h_{12} + X_{22}\partial h_{21} + X_{23}\partial h_{22}$$

$$\partial W_{21} = X_{21}\partial h_{11} + X_{22}\partial h_{12} + X_{31}\partial h_{21} + X_{32}\partial h_{22}$$

$$\partial W_{22} = X_{22}\partial h_{11} + X_{23}\partial h_{12} + X_{32}\partial h_{21} + X_{33}\partial h_{22}$$

Pooling Layer: Backpropagation

Max-Pooling

Der Fehler wird nur dem Ursprung zugewiesen - der „Gewinnereinheit“, da andere Einheiten in den Pooling-Blöcken der vorherigen Ebene nicht dazu beigetragen haben, und daher alle anderen zugewiesenen Werte Null

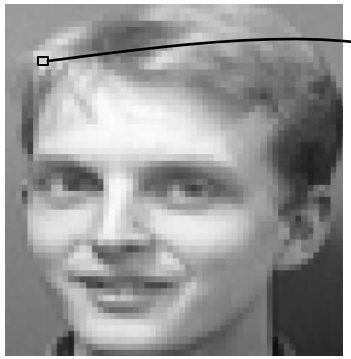
Average-Pooling

Der Fehler wird mit $1/N \times N$ *(N Pooling-block Dimension)* multipliziert und dem gesamten Pooling-Block zugeordnet (alle Einheiten erhalten denselben Wert).

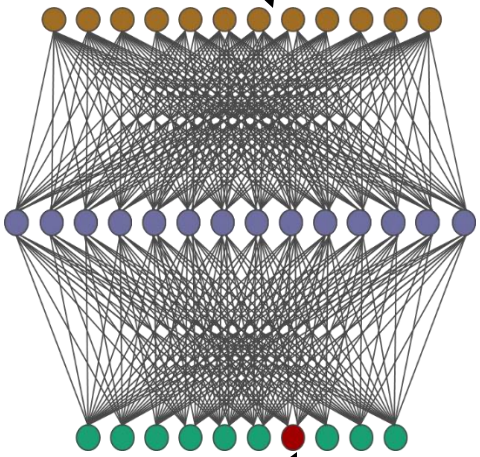
Beispiel:
CNN Definition Keras - Beispielcode

```
% Define CNN Model with keras
model = Sequential()

% Convolutional Layers

model.add(Conv2D(32, (3, 3), input_shape=(32, 32,3), activation='relu',
         padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

% FC Layers
model.add(Flatten())
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dense(num_classes, activation='softmax'))

%compile model
model.compile(loss='categorical_crossentropy', optimizer=adam,
            metrics=['accuracy'])

% train model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50,
                batch_size=64)
```

Beispiel:
MLP (mit 1 deep layer) zur Personenerkennung



pixel



0	…	1	0	…	0	0	0	0	0
…	…	…	..	…	…	…	…	…	…

Person-Klasse

Bild n*p pixel
zB. n=28 p=28

N =n*p Anzahl der Eingabeparameter
N=784

P Hidden Layers-Anzahl
K(i) Anzahl Neuronen pro Hidden Layer
zB P=1, K=50;

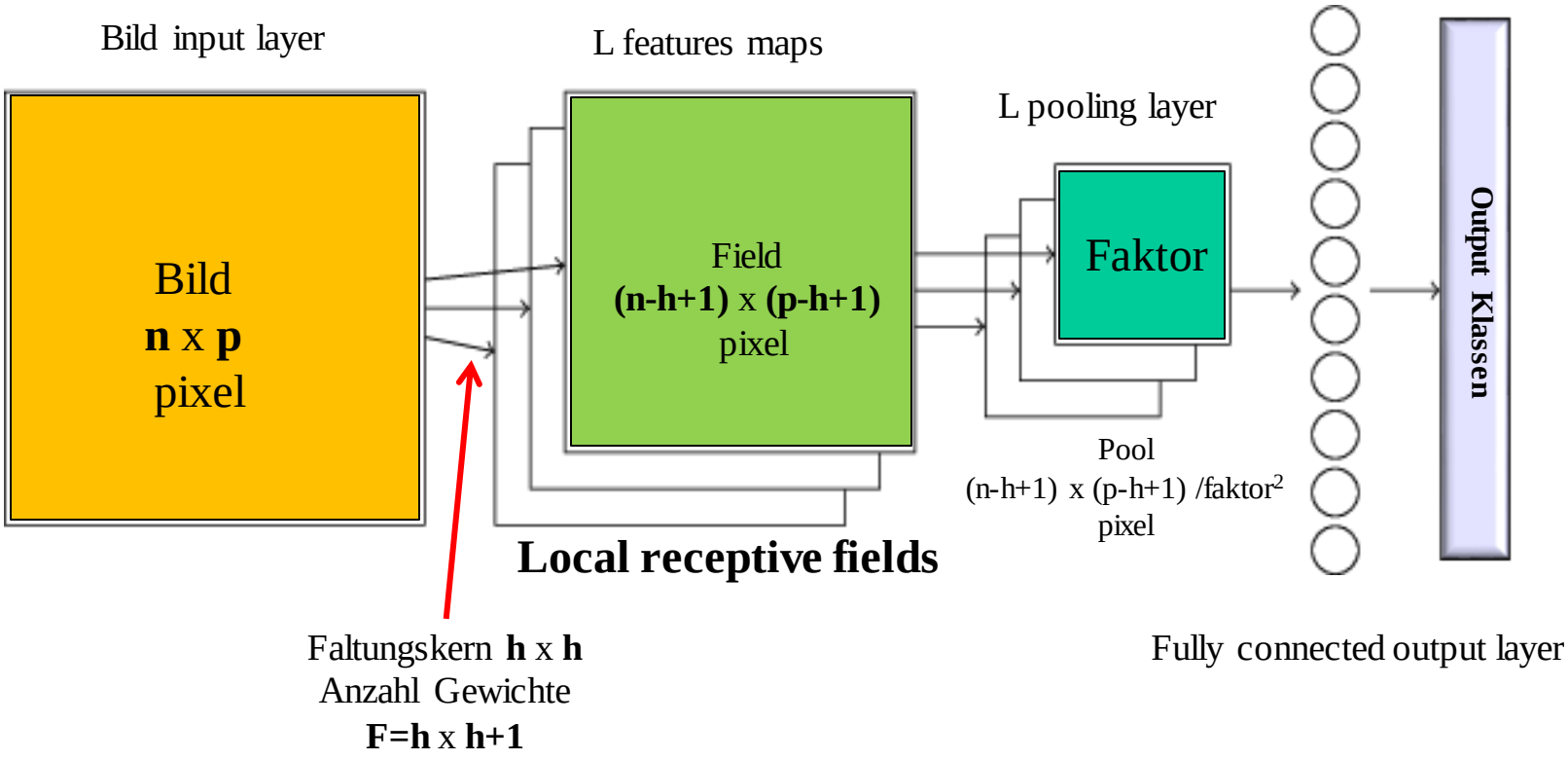
M Anzahl der Person-Klassen
zB. M=40

Anzahl der Gewichte
W= n*p*K+K*M + M +K

W= 41 290

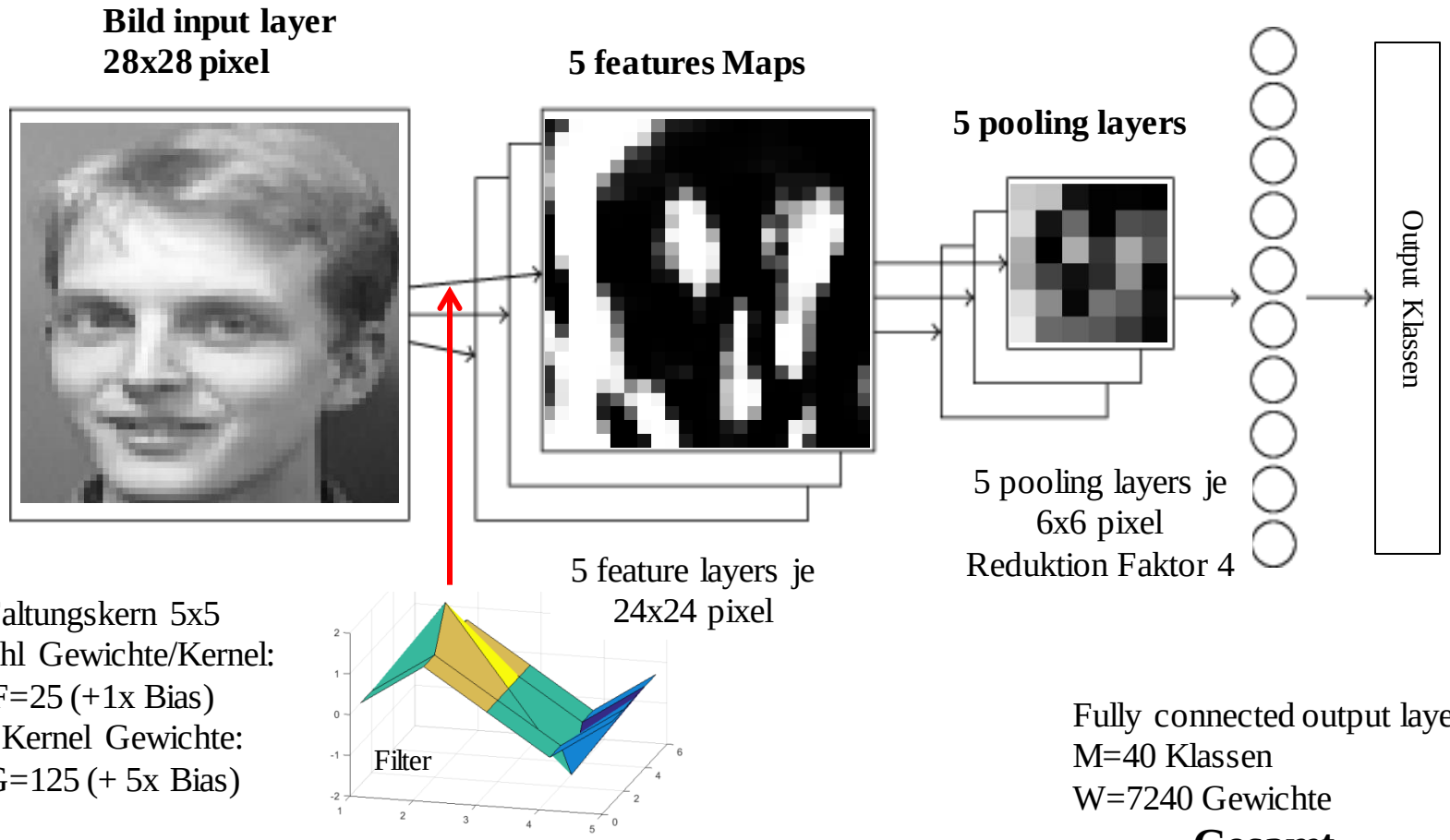
Data: 400 Bilder von 40 Personen

Beispiel:
CNN (mit 1 deep layer) zur Personenerkennung



convolutional network – 1 deep level

Beispiel:
CNN (mit 1 deep layer) zur Personenerkennung



Faltungskern 5x5
Anzahl Gewichte/Kernel:
F=25 (+1x Bias)
5x Kernel Gewichte:
FG=125 (+ 5x Bias)

Fully connected output layer
M=40 Klassen
W=7240 Gewichte

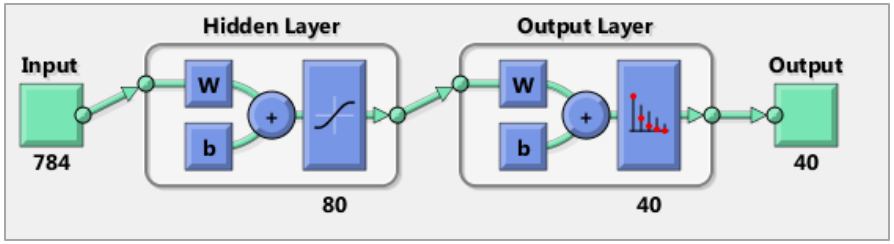
Gesamt
W= 7370
Reduktion ~ 80 %

convolutional network – 1 deep level

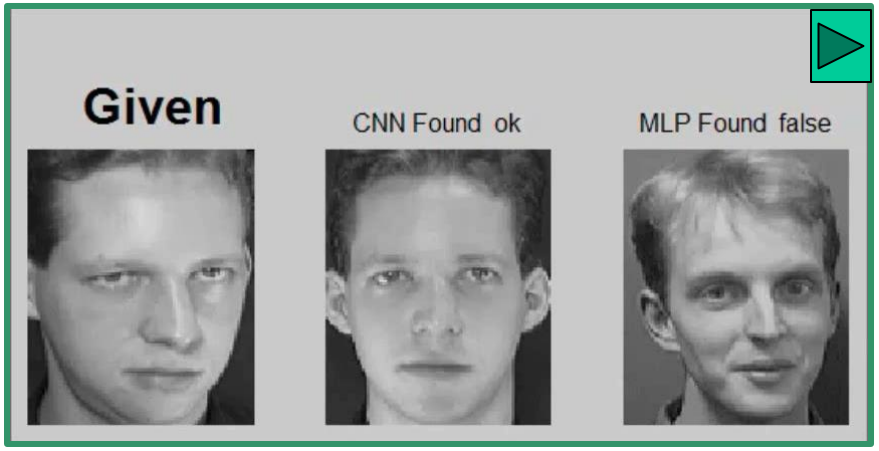
Personenerkennung: CNN vs. MLP

Data: 400 Bilder von 40 Personen

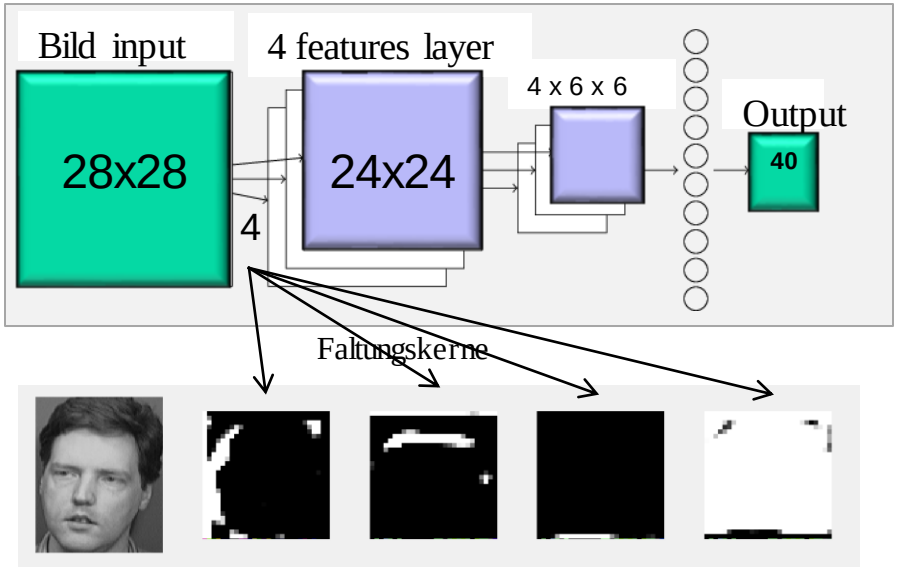
MLP



Multilayer Perceptron – Recognition Error = ~ 10%



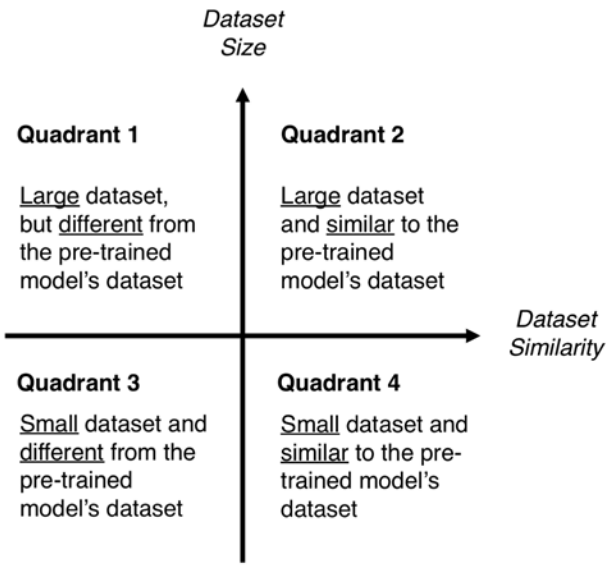
CNN

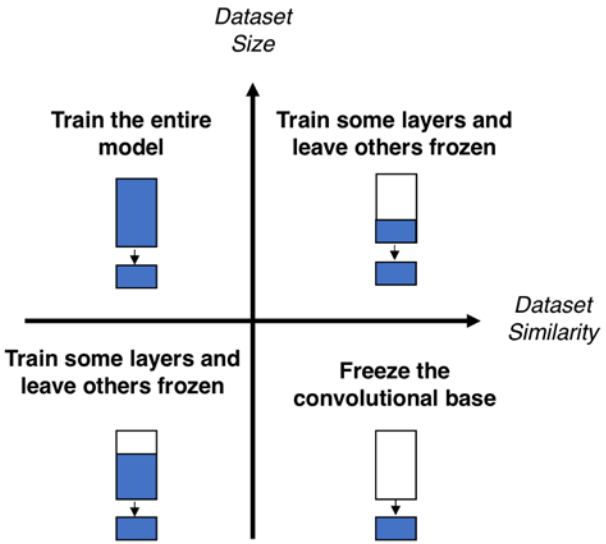


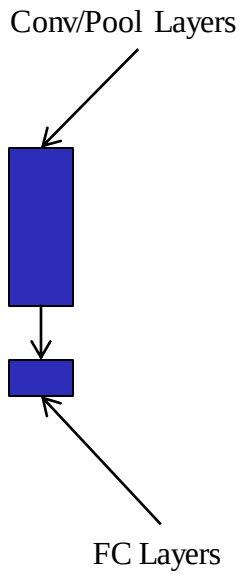
Convolutional Network – Recognition Error = ~1.5 %

Weiterer Einsatz für CNNs: Transfer Learning

- Transfer-Lernen ist eine Technik des maschinellen Lernens, bei der ein für eine Aufgabe trainiertes Modell für eine zweite verwandte Aufgabe erneut verwendet werden kann.
- Das bedeutet, dass Teile des Modells wieder verwendet werden, abhängig von der verwendeten Modellierungstechnik.
- Transfer Learning mit CNN-Modellen:



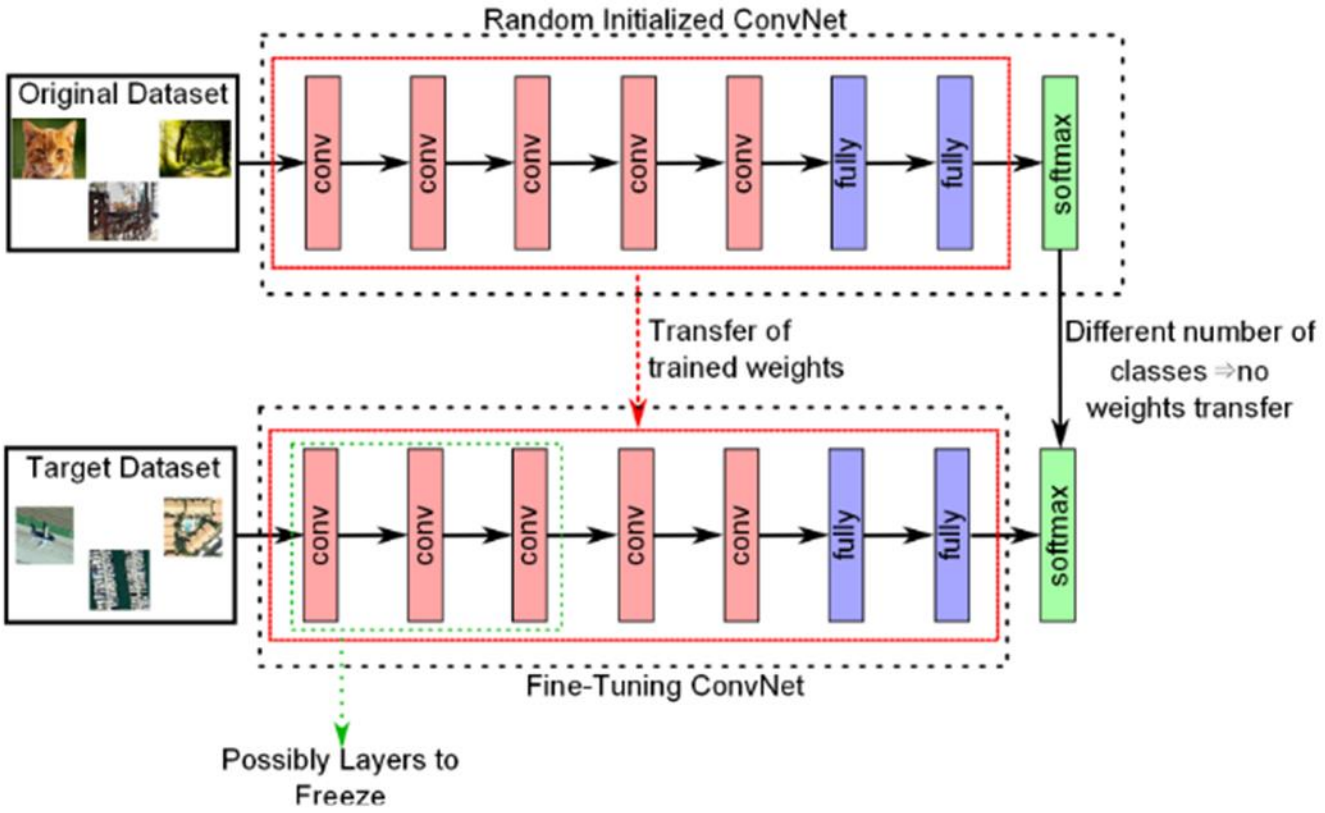




Weiterer Einsatz für CNNs: Transfer Learning - Beispiel

Fully trained CNN

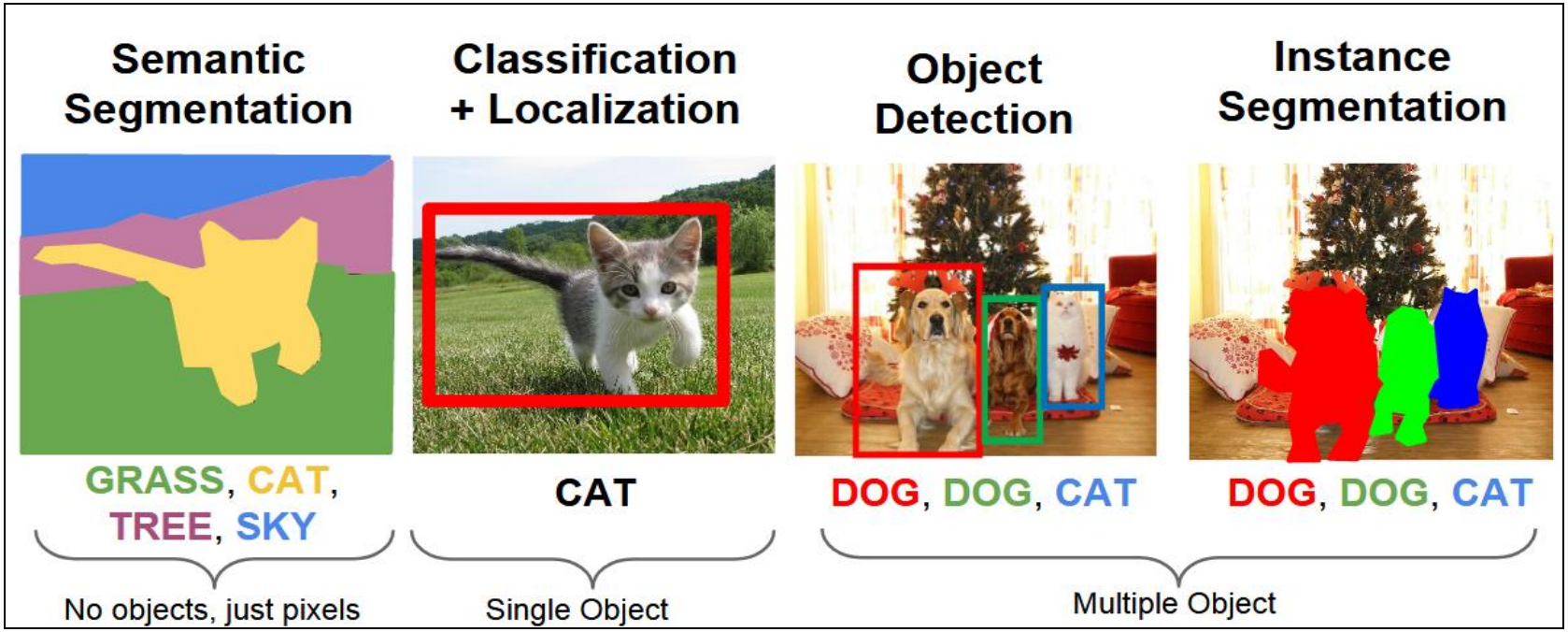
Train some layers and leave others frozen
Replace output layer to number of new classes



Frozen Layers Keras: Keras Layers haben das Attribut `trainable`.
Für jeden frozen Layer kann `trainable=false` gesetzt werden

Transfer Learning see: https://arxiv.org/pdf/1602.01517.pdf

Weiterer Einsatz für CNNs
R-CNN (Region Based CNN)



A Non-Technical Survey on Deep Convolutional Neural Network Architectures

Region Based CNNs (R-CNN - 2013, Fast R-CNN - 2015, Faster R-CNN - 2015)

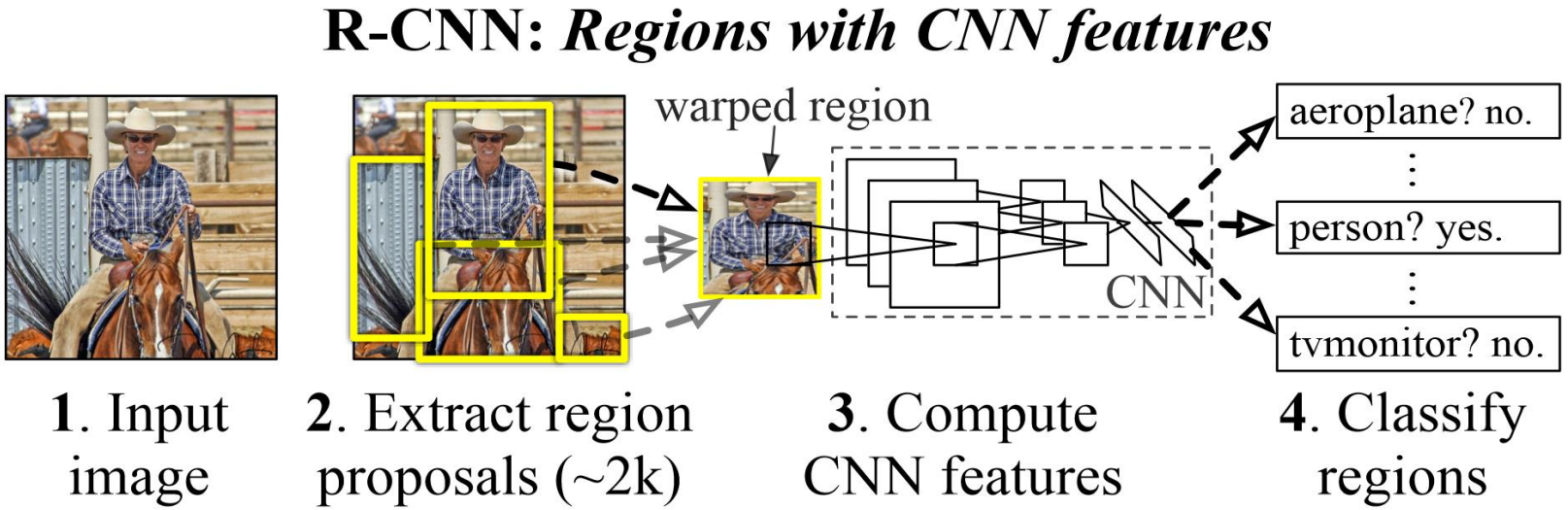
Mask R-CNN (Mask R-CNN - 2018):

GluonCV: a Deep Learning Toolkit for Computer Vision
See: https://gluon-cv.mxnet.io

R-CNN (Region Based CNN)

R-CNN besteht aus drei Modulen.

- Die erste generiert kategorienunabhängige Regionsvorschläge (region proposals).
- Das zweite Modul ist ein großes neuronales CNN Netzwerk, das aus jeder Region einen Feature-vektor fester Länge extrahiert.
- Das dritte Modul besteht aus einer Reihe klassenspezifischer linearer SVMs (Support Vector Machine).

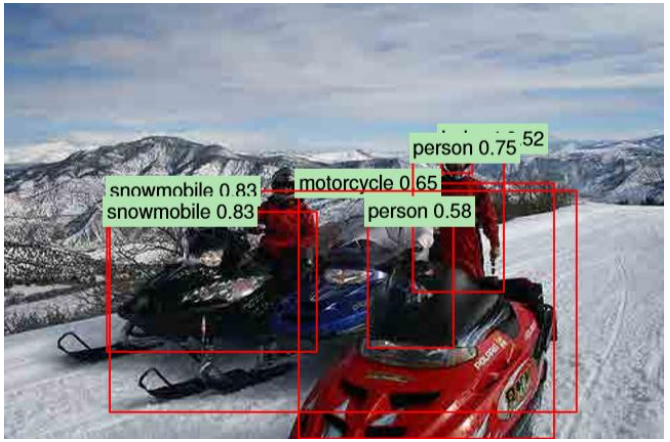


R-CNN (Region Based CNN)

Region Vorschläge. Verschiedene Image-Segmentierungsbasierte Systeme bieten Methoden zur Erstellung von kategorienunabhängigen Regionsvorschlägen.
In der Regel wird die selektive Suchmethode verwendet.

Selektive Suche:

1. Generieren Sie eine anfängliche Bildsegmentierung (viele Kandidatenregionen)
2. Verwenden Sie den „greedy"-Algorithmus, um ähnliche Nachbar-Regionen mithilfe der Regions-Ähnlichkeitsfunktion (similarity → colour, texture, size, fill) rekursiv zu größeren Regionen zu kombinieren
3. Verwenden Sie die angegebene Anzahl generierter Regionen, um die endgültigen Vorschläge für Kandidatenregionen zu erstellen



R-CNN (Region Based CNN)

Feature Extraktion.
aus jedem Regionsvorschlag wird einen N-dimensionalen Feature-vektor unter Verwendung großes neuronales CNN Netzwerk extrahiert.
Features werden berechnet, indem ein RGB-Regionsbild in Vorwärtsrichtung durch fünf Faltungsschichten CNN und zwei vollständig verbundene Schichten weitergeleitet wird. Um Features für einen Regionsvorschlag zu berechnen, müssen zuerst die Bilddaten in dieser Region in eine Form konvertieren werden, die mit dem CNN kompatibel sind.

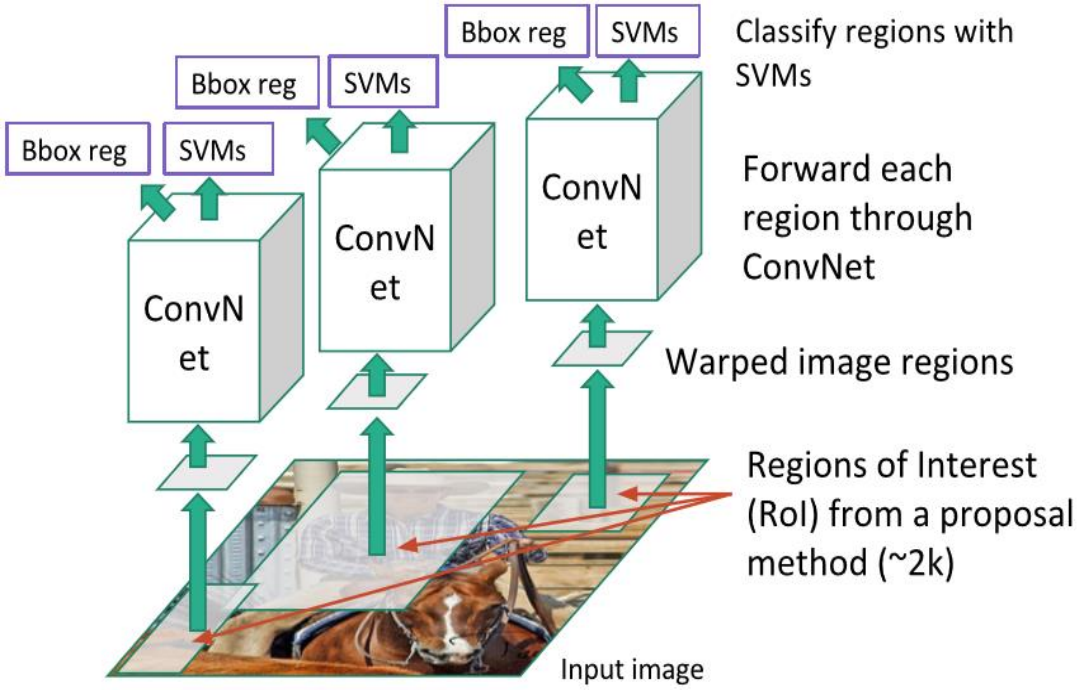
Verschiedene Transformationen für Regionvorschläge. (A) der ursprüngliche Objektvorschlag in seinem tatsächlichen Maßstab relativ zu den transformierten CNN-Eingaben; (B) engstes Quadrat mit Kontext; (C) engstes Quadrat ohne Kontext; (D) Verziehen.

R-CNN (Region Based CNN)

Training

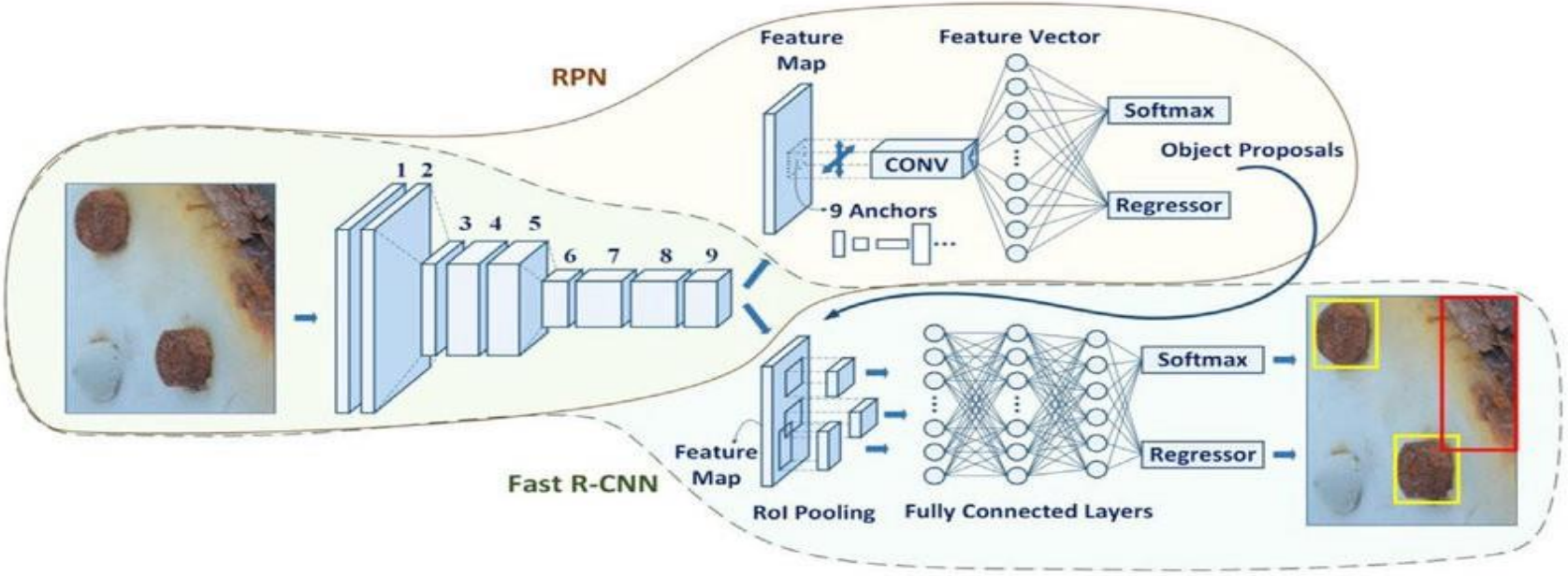
Dann wird für jede Klasse die Bewertung für jeden extrahierten Feature-Vektor unter Verwendung der für diese Klasse trainierten SVM berechnet. Bei allen bewerteten Regionen in einem Bild wird eine nicht maximale Unterdrückung (für jede Klasse unabhängig) angewendet, die eine Region zurückweist, wenn sie eine Überschneidung über Vereinigung (IoU) mit einer ausgewählten Region mit höherer Bewertung aufweist, die größer als ein erlernter Schwellenwert ist.

Zusätzlich ein lineares Bounding-Box-Regressionsmodell wird trainiert, um neue Erkennungsfenster unter Berücksichtigung der letzten Pool-Features für einen Vorschlag für einen selektiven Suchbereich vorherzusagen.



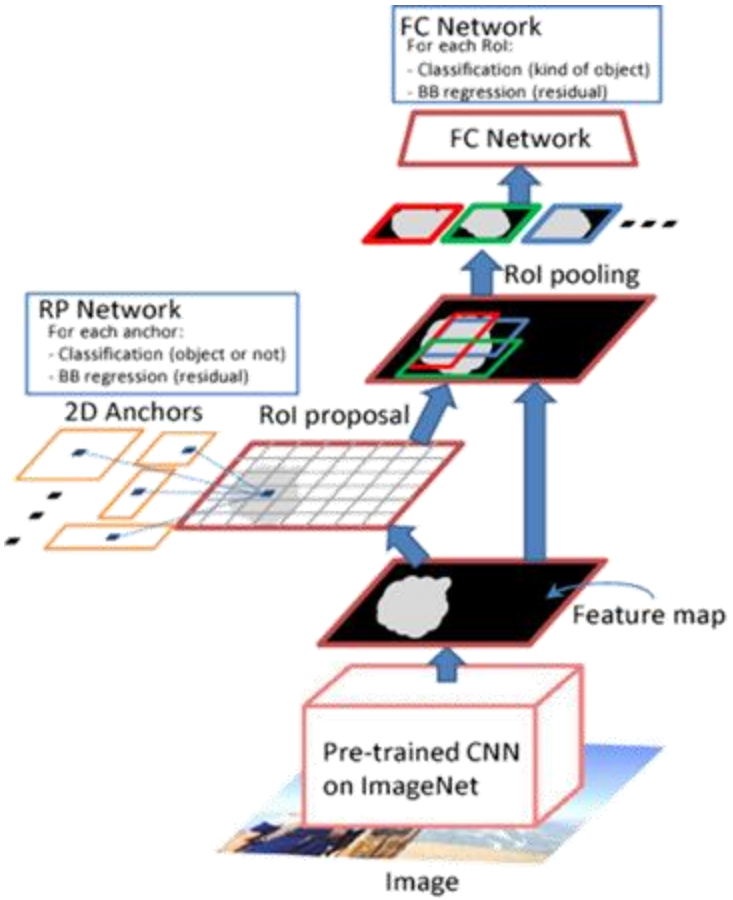
Faster R-CNN (Faster RCNN)

R-CNN und Fast R-CNN verwenden eine selektive Suche, um die Regionsvorschläge herauszufinden. Die selektive Suche ist ein langsamer und zeitaufwendiger Prozess, der sich auf die Leistung des Netzwerks auswirkt.
Ähnlich wie bei R-CNN wird das Bild als Eingabe für ein CNN bereitgestellt, das eine Feature-Map bereitstellt. Anstatt einen selektiven Suchalgorithmus auf der Feature-Map zu verwenden, um die Regionsvorschläge zu identifizieren, wird ein separates Netzwerk verwendet, um die Regionsvorschläge vorherzusagen. Die vorhergesagten Regionsvorschläge werden dann unter Verwendung einer ROI-Pooling-Ebene umgeformt, die dann verwendet wird, um das Bild innerhalb der vorgeschlagenen Region zu klassifizieren und die Versatzwerte für Bounding-Box vorherzusagen.

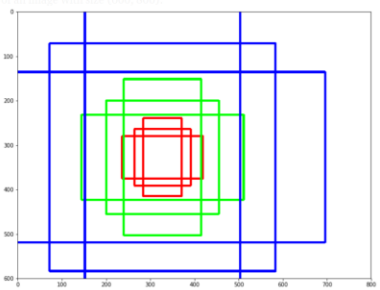


Faster R-CNN (Faster RCNN)

Faster R-CNN verfügt über zwei Netzwerke: Region Proposal Network (RPN) zum Generieren von Regionsvorschlägen und ein Netzwerk, das diese Vorschläge zum Erkennen von Objekten verwendet. Der Hauptunterschied bei R-CNN besteht darin, dass der R-CNN die selektive Suche verwendet, um Regionsvorschläge zu generieren. Der Zeitaufwand für die Erstellung von Regionsvorschlägen ist bei der RPN viel geringer als bei der selektiven Suche, wenn die RPN die meisten Berechnungen mit dem Objekterkennungsnetzwerk gemeinsam nutzt. RPN ordnet Regionsfelder (sogenannte Anker) und schlägt diejenigen vor, die höchstwahrscheinlich Objekte enthalten. Ein Anker ist ein Box.



Die Ausgabe Region Proposal Network (RPN) besteht aus einer Reihe von Boxes-Vorschlägen, die von einem Klassifizierer und einem Regressor geprüft werden, um schließlich das Vorkommen von Objekten zu überprüfen.



Anker

Faster R-CNN (Faster RCNN)

Jede Position in der Feature-Map hat 9 Anker und jeder Anker hat zwei mögliche Beschriftungen (Hintergrund, Vordergrund). Wenn Sie die Tiefe der Feature-Map auf 18 (9 Anker x 2 Beschriftungen) festlegen, wird für jeden Anker ein Vektor mit zwei Werten (normalerweise Logit genannt) erstellt, der den Vordergrund und den Hintergrund darstellt. Wenn wir das Logit in eine Aktivierungsfunktion für Softmax / Logistische Regression einspeisen, werden die Labels vorhergesagt.

ROI-Pooling

Nach RPN erhalten wir vorgeschlagene Regionen mit unterschiedlichen Größen. Region of Interest Pooling kann das Problem vereinfachen, indem die Feature-Maps auf dieselbe Größe reduziert werden. Das ROI-Pooling teilt die Eingabe-Feature-Map in eine feste Anzahl von ungefähr gleichen Regionen auf und wendet dann das Max-Pooling auf jede Region an. Daher ist die Ausgabe von ROI-Pooling unabhängig von der Größe der Eingabe immer fest.

Ein Eingabebild und mehrere ROIs werden in ein vollständig faltungsorientiertes Netzwerk eingegeben. Jeder ROI wird in einer Feature-Map fester Größe zusammengefasst und dann durch vollständig verbundene Layer (FCs) einem Feature-Vektor zugeordnet. Das Netzwerk hat zwei Ausgangsvektoren pro RoI: Softmax-Wahrscheinlichkeiten und Bounding-Box-Regressionsoffsets pro Klasse.

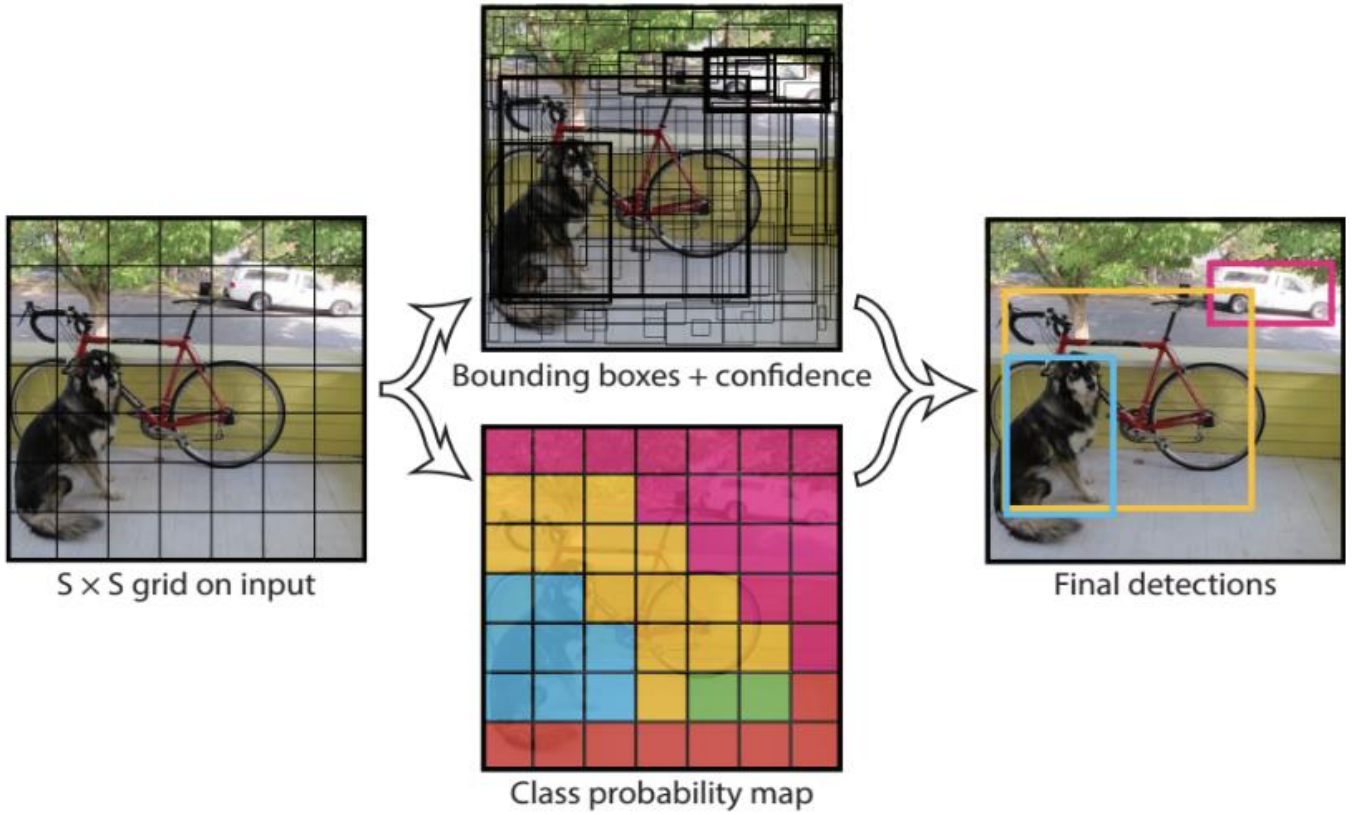
R-CNN: YOLO (You Only Look Once)

Beide oben genannten Algorithmen (R-CNN und Fast R-CNN) verwenden eine selektive Suche, um die Regionsvorschläge herauszufinden. Die selektive Suche ist ein langsamer und zeitaufwendiger Prozess, der sich auf die Leistung des Netzwerks auswirkt.

YOLO System unterteilt das Eingabebild in ein S × S -Gitter.
Wenn die Mitte eines Objekts in eine Gitterzelle fällt, ist diese Gitterzelle für die Erkennung dieses Objekts verantwortlich.
Jedes Gitter sagt B Bounding Boxes und Confidence Scores für Boxes voraus. Diese Konfidenz zeigt, wie sicher das Modell ist, dass die Box ein Objekt enthält und auch, wie genau sie zeigt, dass die Box den Vorhersagen entspricht.

Wenn in dieser Zelle kein Objekt vorhanden ist, sollten die Konfidenzwerte Null sein. Jeder Bouding Box besteht aus 5 Parameter: x, y, w, h und Vertrauen. Die (x, y) - Koordinaten stellen die Mitte der Box relativ zu den Grenzen der Gitterzelle dar. Die Breite und Höhe sind für das gesamte Bild vorhersehbar. Schließlich repräsentiert die Konfidenzvorhersage die IoU zwischen der vorhergesagten Box und einer beliebigen Grundwahrheitsbox.
Jede Gitterzelle sagt auch C-bedingte Klassenwahrscheinlichkeiten voraus. Diese Wahrscheinlichkeiten hängen von der Gitterzelle ab, die ein Objekt enthält.

R-CNN: YOLO (You Only Look Once)



Input image
3 x H x W

Divide image into grid
7 x 7

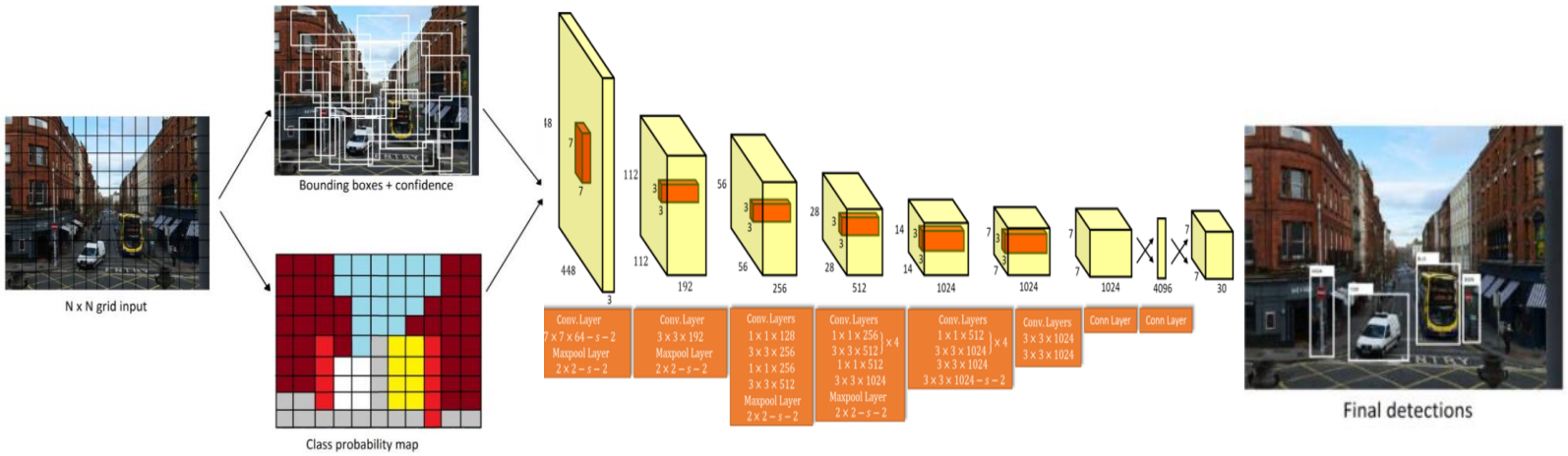
Image a set of **base boxes** centered at each grid cell
Here B = 3

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

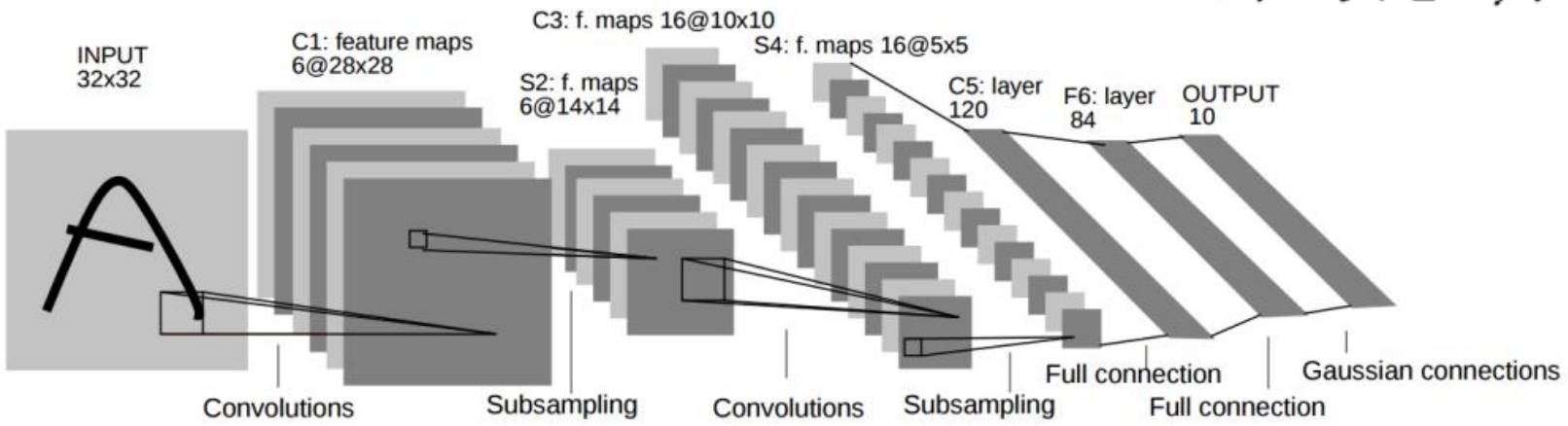
R-CNN: YOLO (You Only Look Once)

YOLO hat einige Ähnlichkeiten mit Faster R-CNN. Jede Gitterzelle schlägt potenzielle Begrenzungsrahmen (Bouding Box) vor und bewertet diese mithilfe von Faltungs-features. System stellt jedoch räumliche Einschränkungen für die Gitterzellenvorschläge bereit, wodurch mehrere Erkennungen desselben Objekts verringert werden. Das System schlägt auch weit weniger Begrenzungsrahmen vor, nur ~ 100 pro Bild im Vergleich zu ungefähr 2000 aus der selektiven Suche in R-CNN. Schließlich kombiniert System diese einzelnen Komponenten zu einem einzigen, gemeinsam optimierten Modell.

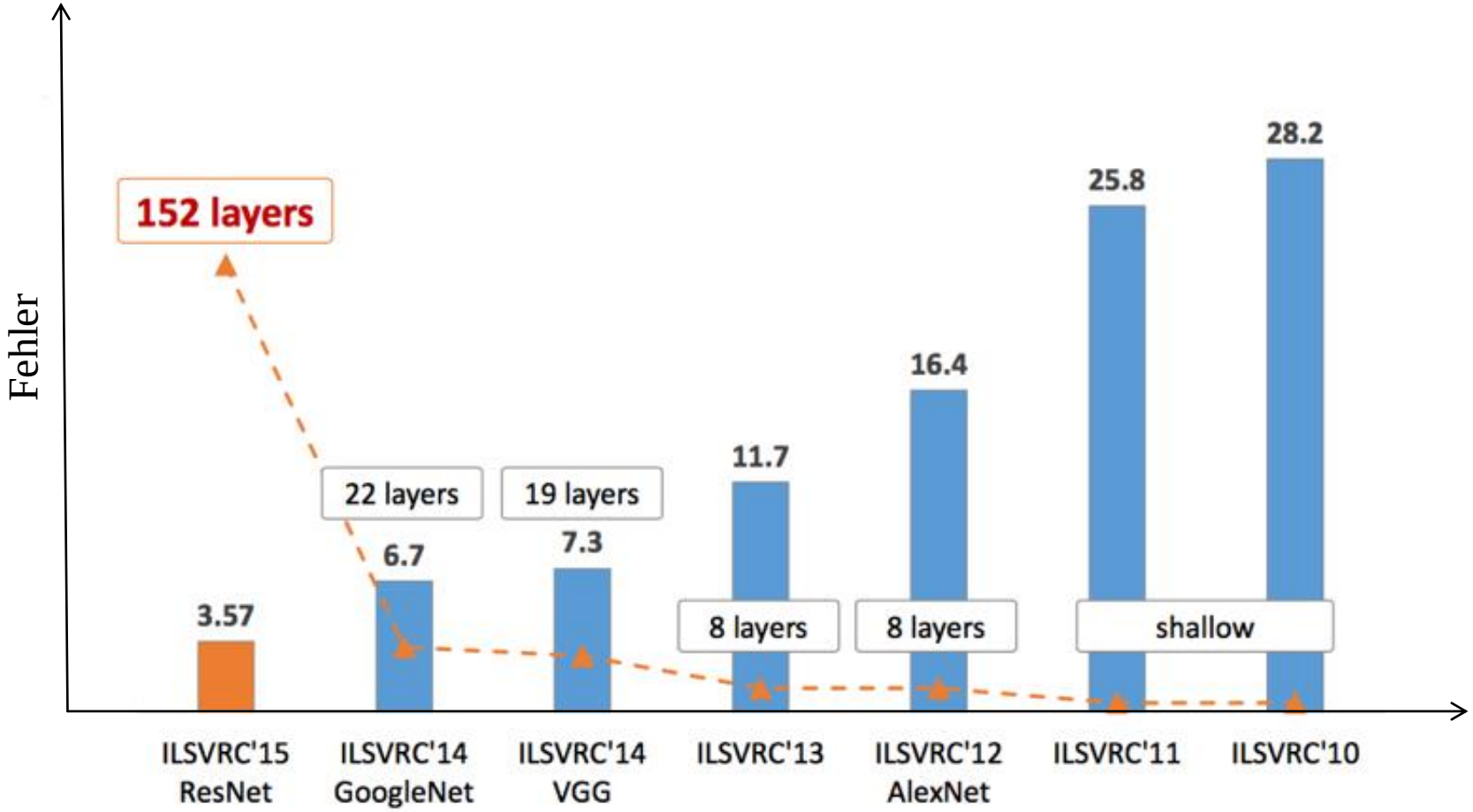


LeNet5 (Professor LeCun et al., 1998)
Director of AI Research at Facebook

- Erstes publiziertes CNN
- Entwickelt für die Klassifikation handgeschriebener Zeichen
- Bilder mit der Auflösung von 32x32 Pixel wurden klassifiziert
- Erkennungsrate beträgt 99,0%



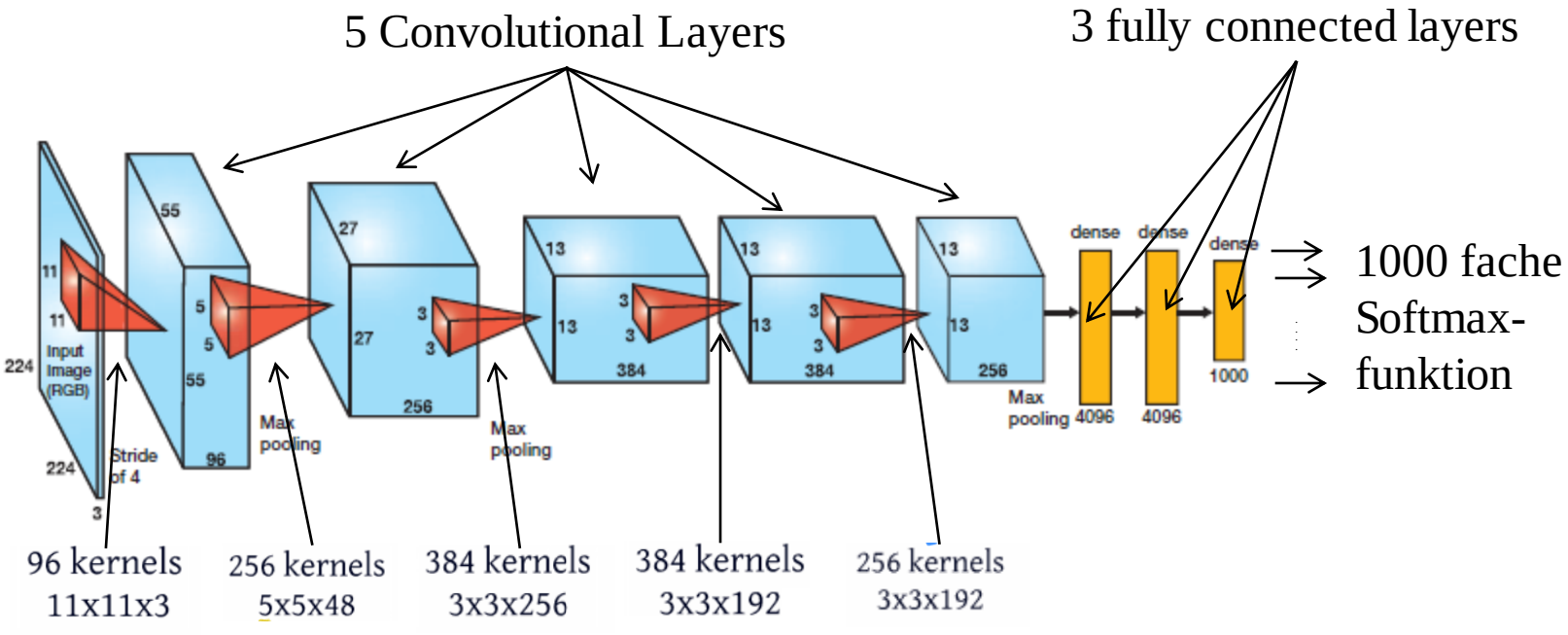
ILSVRC
ImageNet Large Scale Visual Recognition Challenge



Klassifikationsfehler Mensch: ca. 5%

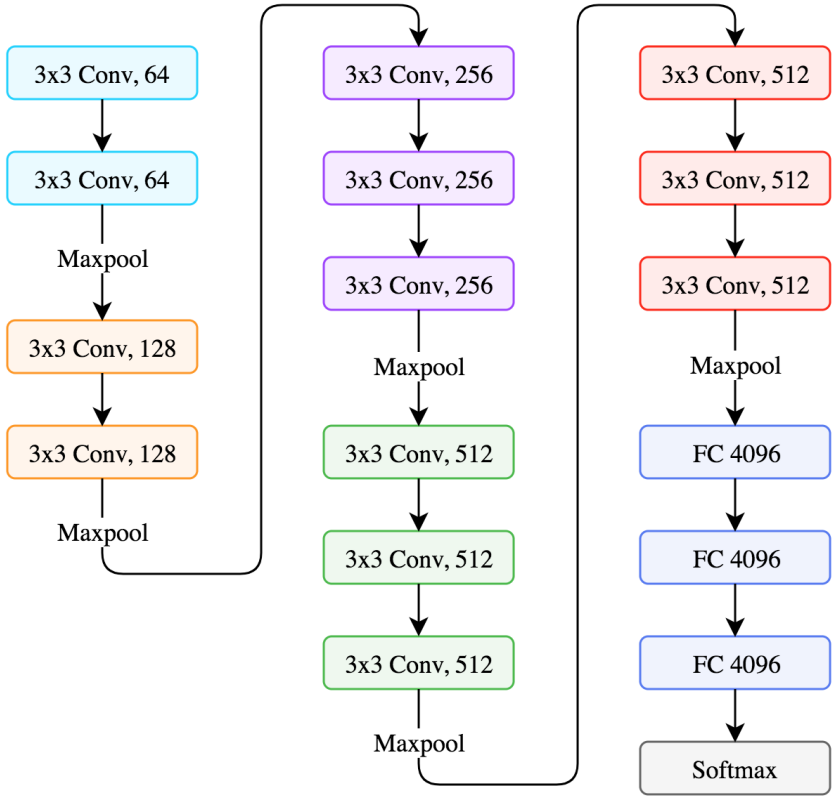
AlexNet (von Alex Krizhevsky 2012)

- Über 15 Millionen klassifizierte hochauflösende Fotos (227x227x3) verwendet
- Mit ca. 1 000 verschiedenen Klassen
- Fotos im Internet gesammelt und händisch klassifiziert
- Training mit 2 GPUs 6 Tage lang



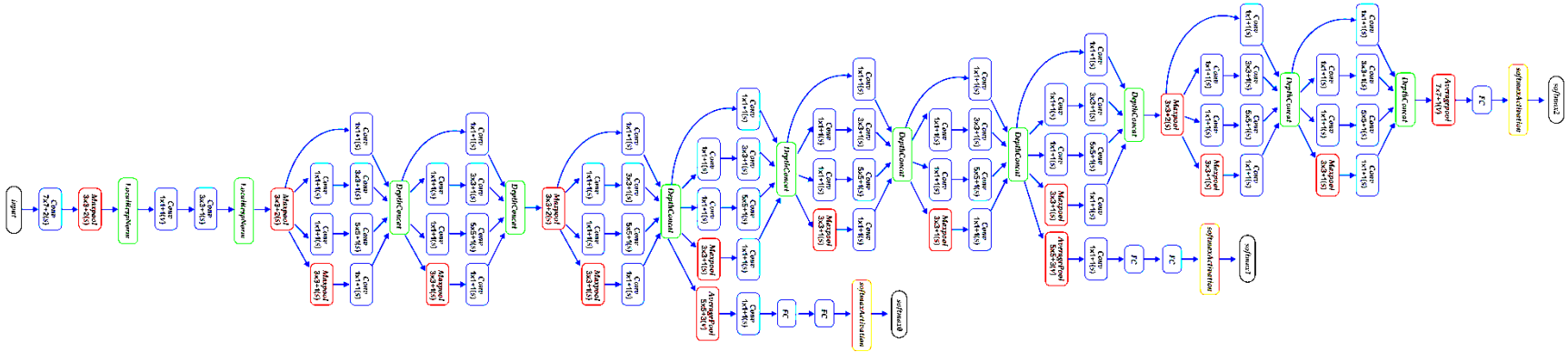
VGG-16 bzw. VGG-19 (Oxford's Visual Geometry Group, 2014)

- gleiche Aufgabenstellung wie bei AlexNet
- Nur 3x3 dimensionale Kernels verwendet, dafür mehrere Convolutional Layers hintereinander geschaltet
- 16 - 19 layers (VGG16/VGG19)
- Training mit 4 GPUs 3 Wochen lang



GoogleNet 2015

- gleiche Aufgabenstellung wie bei AlexNet
- 22 Layers
- Optimierungen durch Änderungen an der Topologie
- 12x weniger Parameter als AlexNet
- Trainiert mit „einigen high-end GPUs" in einer Woche
- 6.7% Fehler



ResNet, Microsoft 2015

- Seit AlexNet gehen aktuelle CNN-Architekturen immer mehr in die Tiefe.
- Während AlexNet nur 5 Convolutional Layers aufwies, hatten VGG19 und GoogleNet schon 19 bzw. 22 Layer
- Microsoft präsentierte 2015 ResNet mit 152 Layern
- Erhöhung der Netzwerktiefe funktioniert NICHT, indem einfach Layer übereinander gestapelt werden:
  - Tiefe Netzwerke sind wegen des berüchtigten Problems des **verschwindenden Gradienten** schwer zu trainieren: Wenn der Gradient auf frühere Layer zurück propagiert wird, kann die wiederholte Multiplikation den Gradienten unendlich klein machen.
  - Wenn das Netzwerk zu tief wird, ändern sich durch das **Verschwinden des Gradienten** die Gewichte nicht mehr und das Netzwerk hört auf zu lernen !

Abhilfe mit sogenannten **Residual Networks** siehe:
https://wiki.tum.de/display/lfdv/Deep+Residual+Networks

Geoffrey Hinton thinks that we need to actually start all over ……..(2017)

„My view is throw it all away and start again. "

Geoffrey Hinton (Universität Toronto und Google) untersucht die Anwendung von künstlichen neuronalen Netzen in den Bereichen Lernen, Gedächtnis, Wahrnehmung und Symbolverarbeitung.
Er gehörte zu den Wissenschaftlern, die den **Backpropagation-Algorithmus** einführten ….