# Deep Learning

## Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2024-08 Dozent: Karsten Flügge

# Themen des Tages

## Tag 9

**Objekterkennung und Lokalisierung**
**Regional CNN**
**Regressionsprobleme?**

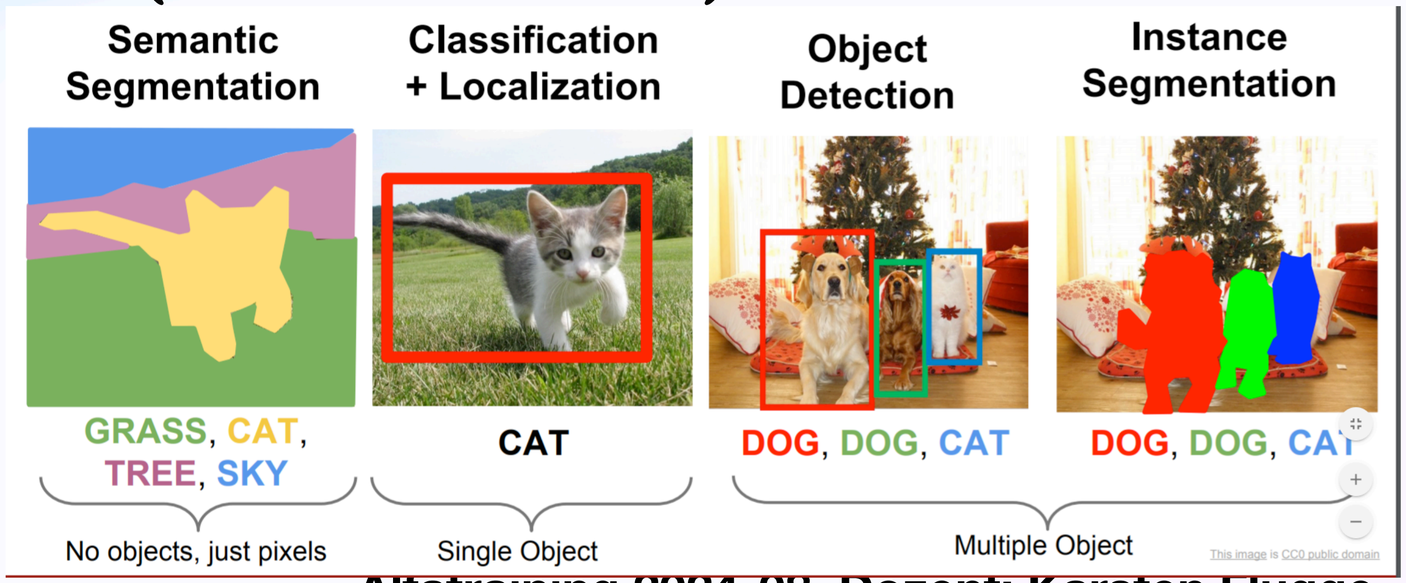
**Verzweigte neuronale Netze**

# Objekterkennung und Lokalisierung

Was ist da? Klassifizierung

Wo? Lokalisierung

- global
- 'boundary boxes'
- Segmentierung
- outline prediction (Silhouette)



# Objekterkennung und Lokalisierung

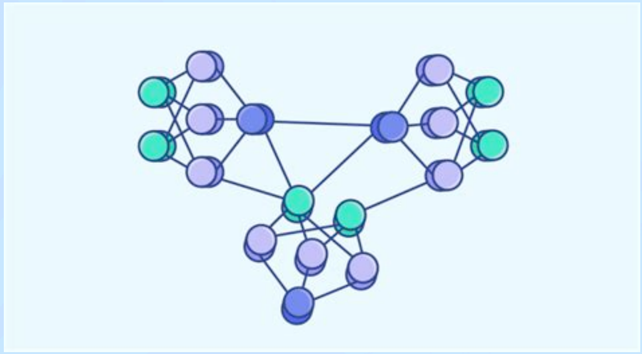
Was ist da? Klassifizierung

Wo? Lokalisierung

- global
- 'boundary boxes'
  Koordinaten (x,y,x2,y2)
  => Regression (Werte zwischen 0 und height/width)
- Segmentation:
  Klassifizierung per Pixel
  Instance Trennung?

# Verzweigte neuronale Netze

## Branched Neural Networks



Da unsere Schichten verschiedene Eingaben und Ausgaben verwenden können, kann man jene auch **splitten und (re)kombinieren**

=> mehrere parallele Pfade

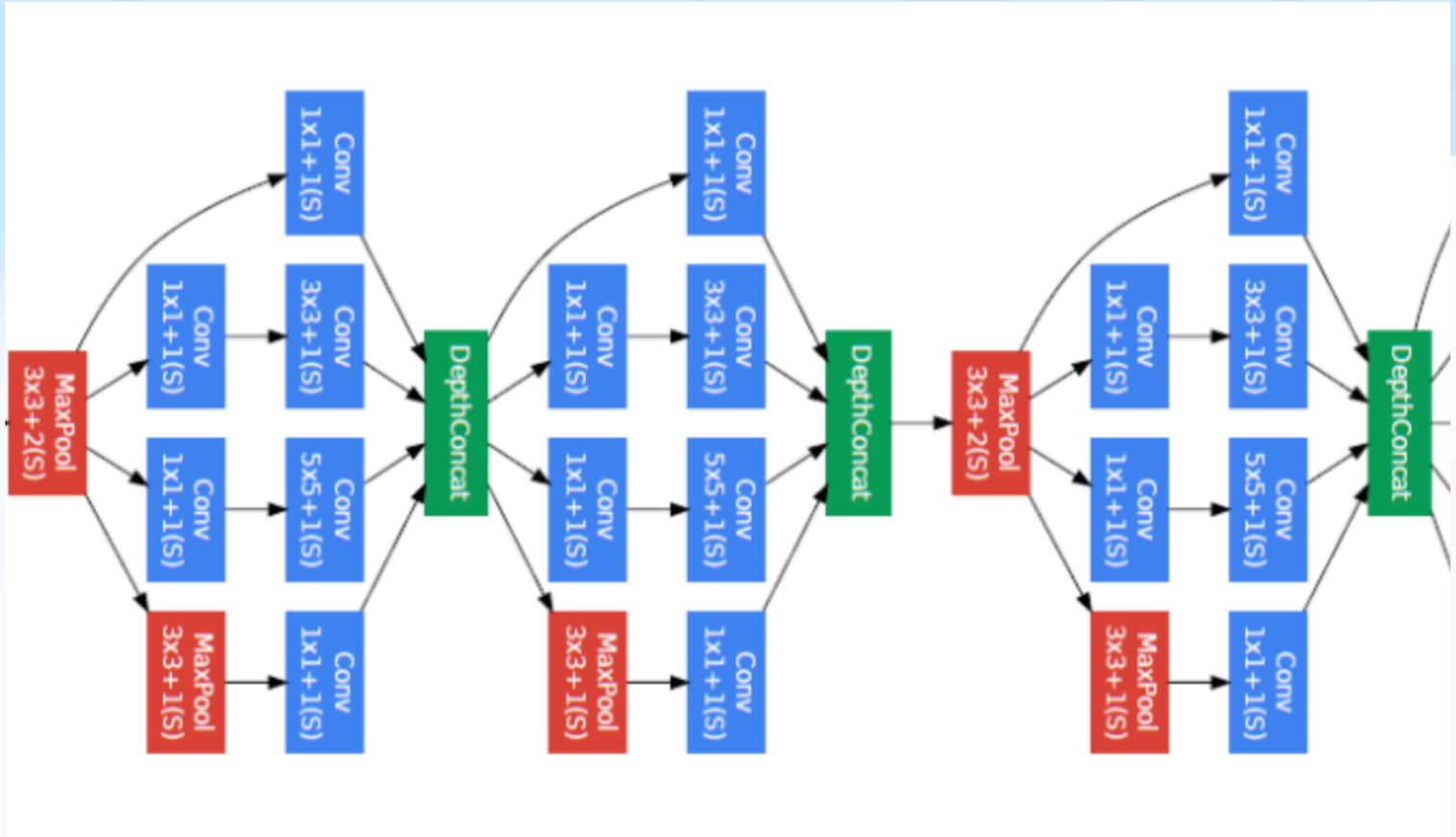
# Verzweigte neuronale Netze

- **Multiskalenanalyse** (z.B. Bildverarbeitung)
- Integration **spezialisierter Sub-Netze**
- auf bestimmte **Merkmale** optimiert

Wir kennen bereits:
Skip / Residual Connections (ResNet) / Dense Layer

# Verzweigte neuronale Netze

**Inception-Module:** Ein Ansatz von GoogLeNet (2014), der verschiedene Filtergrößen parallel anwendete und die Ergebnisse kombinierte, um eine tiefere und breitere Netzstruktur zu ermöglichen.



# Verzweigte neuronale Netze

```python
# Branch: Teilt den Output in mehrere Pfade auf
class Branch(nn.Module):
  def __init__(self, branches):
    super(Branch, self).__init__()
    self.branches = nn.ModuleList(branches) # Parameter-Registrierung

  def forward(self, x):
    outputs = [branch(x) for branch in self.branches]
    return outputs


# Join: Verbindet die Outputs der Pfade wieder
class Join(nn.Module):
  def forward(self, inputs, join_dim=1):
    return torch.cat(inputs, dim=join_dim)  # Concatenation entlang der Feature-Dimension
```

# Verzweigte neuronale Netze

Aufgabe:
a) Branch und Join selbst implementieren.
Gegebenenfalls spicken, aber möglichst viel selbst versuchen.

b) Einbau in ein einfaches Netz

c) andere Arten der Wiederzusammenfügung als Join?
- Max n Vektoren -> 1 Vektor (Elementweise Max)
- Max n Vektoren -> Skalar
- Add
- Mul
- Kein Zusammenfügen ;)

# Verzweigte neuronale Netze

Anwendungen

0. Inception: Multiskalen Bildanalyse
1. Memory Cells LSTM  (demnächst)
2. Transformer

3. **Multi-Task Learning**: Ein verzweigtes Netz könnte zwei getrennte Zweige haben: einen für die **Klassifikation** und einen anderen für die **Merkmalsanalyse**.
   Wenn man beispielsweise **nicht nur Ziffern klassifizieren** möchte, sondern gleichzeitig die **Dicke der Ziffernlinien** oder **Anzahl der Pixel** oder die **Symmetrie der Ziffern vorhersagen** möchte.

4. **Daten-Augmentierung mit mehreren Pfaden**: Ein Zweig könnte z.B. eine normale CNN-Verarbeitung des Bildes enthalten, während ein anderer Zweig mit transformierten (rotierten, skalierten) Versionen des Bildes arbeitet. Dies könnte das Netzwerk robuster gegenüber verschiedenen Transformationen machen.

# Pause

## Objekterkennung und Lokalisierung

**Ziel: spezifische Regionen in Bildern zu identifizieren, die Objekte enthalten**

Die **Ansätze und Techniken** zum erreichen dieses Zieles haben sich in den vergangenen Jahren **stark weiterentwickelt.** Wir geben einen **Überblick...**

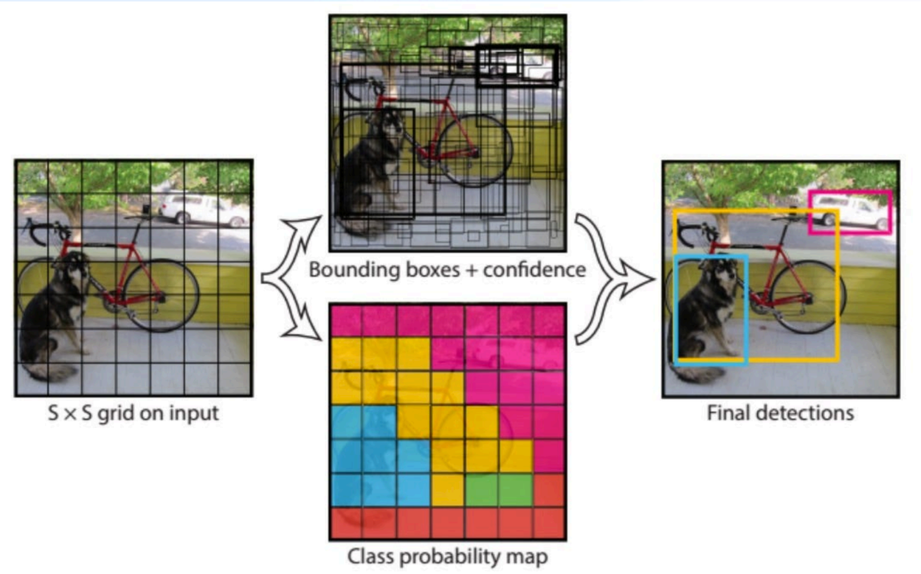
# Regional CNN

**Verschiedene historische R-CNN Varianten:**

- **R-CNN (2014):** Dies ist die ursprüngliche Version von R-CNN, die verwendet wurde, um in einem mehrstufigen Prozess **Regionen** zu identifizieren und zu **klassifizieren**.
- **Fast R-CNN (2015):** Diese Version verbessert die Geschwindigkeit und Genauigkeit durch eine vereinfachte Architektur, indem sie die Merkmalsextraktion pro Bild statt pro Region optimiert.
- **Faster R-CNN (2015):** Eine weitere Verbesserung durch die Einführung von Region Proposal Networks (RPNs), die die Regionenvorschläge integriert und so die Pipeline noch weiter beschleunigt.

https://en.wikipedia.org/wiki/Region_Based_Convolutional_Neural_Networks
"History" (dead)

Generelle Idee:
- wähle pro Segment jenes Feature mit der höchsten W-keit.
- selektiere "**Bounding boxes**" welche die Klasse umfassen.



# Regional CNN

1. **R-CNN Pipeline:**

   **Region Proposal:** Das Bild wird in mögliche Objektregionen unterteilt. Dies erfolgt typischerweise durch Algorithmen wie **Selective Search**.
   **Feature Extraction:** Ein CNN wird verwendet, um Merkmale (Features) aus den Regionen zu extrahieren.
   **Classification: Jede Region wird klassifiziert, um zu bestimmen, ob sie ein bestimmtes Objekt enthält oder nicht.**

   **Vorteile:**
   **Hohe Genauigkeit bei der Objekterkennung.**
   **Gut geeignet für Szenarien mit mehreren Objekten in komplexen Umgebungen.**
   **Nachteile:**
   **Ursprüngliche R-CNNs sind sehr langsam, da jedes Region Proposal separat verarbeitet wird.**
   **Hoher Speicher- und Rechenaufwand.**

3. **Fast R-CNN Verbesserungen:**

   **RoI Pooling (Region of Interest Pooling): Reduziert die Rechenlast, indem Merkmale pro Bild extrahiert werden und dann regionsweise klassifiziert werden.**
   **Multitask Loss: Optimiert Klassifizierung und Bounding Box Regression gleichzeitig.**

4. **Faster R-CNN Verbesserungen:**

   **Region Proposal Network (RPN): Ersetzt Selective Search durch ein eigenes Netzwerk, das in einem Schritt Regionen vorschlägt und klassifiziert, was die Geschwindigkeit enorm verbessert.**

## 1. Single Shot Detectors (SSD)

SSD-Modelle erkennen **Objekte direkt in einem Schritt**, ohne eine separate Phase der Regionenvorschläge. Sie teilen das Bild in verschiedene Bereiche auf und führen die Objekterkennung gleichzeitig für alle Regionen durch.

+ **Sehr schnell,** da das gesamte Bild auf einmal verarbeitet wird.
+ Geeignet für **Echtzeitanwendungen** wie Videos oder mobile Geräte.

Nachteile: Tendenziell weniger genau bei der Erkennung von kleineren Objekten.

Beispiel: SSD300, SSD512.

## 2. You Only Look Once (YOLO)

Das YOLO-Modell erkennt Objekte in einem einzigen Durchgang durch das Netz (**Single Pass**). Es teilt das Bild in ein **Gitter** auf und führt gleichzeitig Klassifizierung und Bounding-Box-Prädiktion für jedes Gittersegment durch.

Extrem schnell und effizient, ideal für Echtzeit-Anwendungen.

Eine gute Balance zwischen Geschwindigkeit und Genauigkeit.

Nachteile: Kann bei der genauen Lokalisierung von Objekten, insbesondere kleinerer, etwas weniger präzise sein als langsamere Modelle.

Beispiel: YOLOv3, YOLOv4, YOLOv5, YOLOv7 2022

### 3. EfficientDet

Beschreibung: EfficientDet kombiniert die Effizienz von EfficientNet als Backbone mit einer skalierbaren, top-down und bottom-up Feature-Fusion-Architektur, genannt BiFPN (Bi-directional Feature Pyramid Network). Dies ermöglicht eine verbesserte Balance zwischen Genauigkeit und Geschwindigkeit.

Sehr flexibel, da es auf verschiedenen Skalierungsstufen trainiert werden kann.

Effizient im Ressourcenverbrauch bei hoher Genauigkeit.

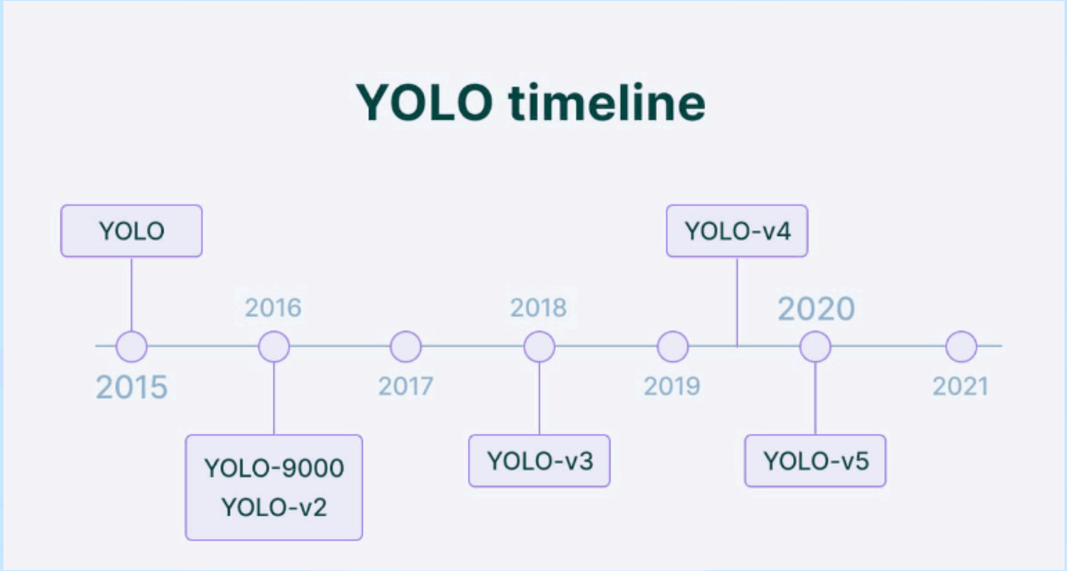
Nachteile: Komplexer als einfachere Modelle wie SSD oder YOLO.

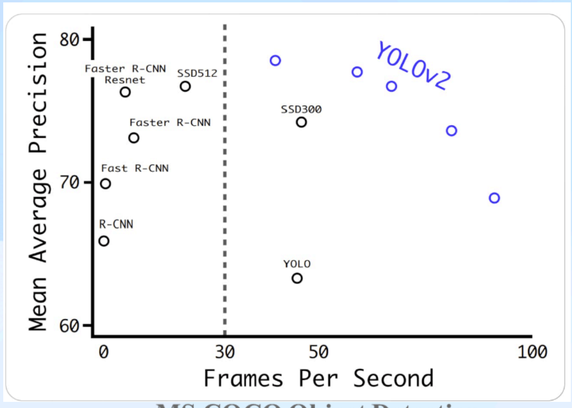
Beispiel: EfficientDet-D0 bis D7.

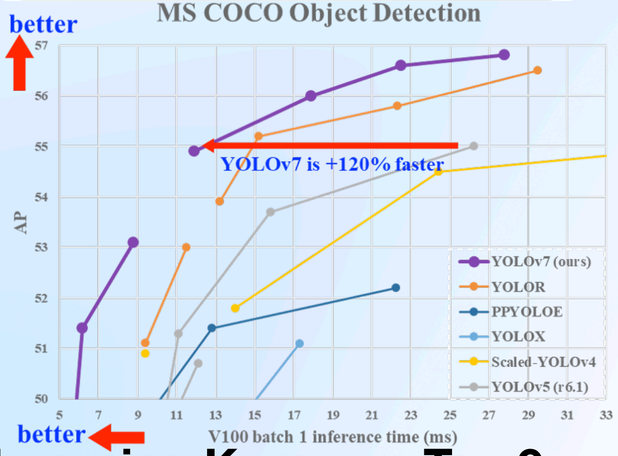
# 4. Transformer based …

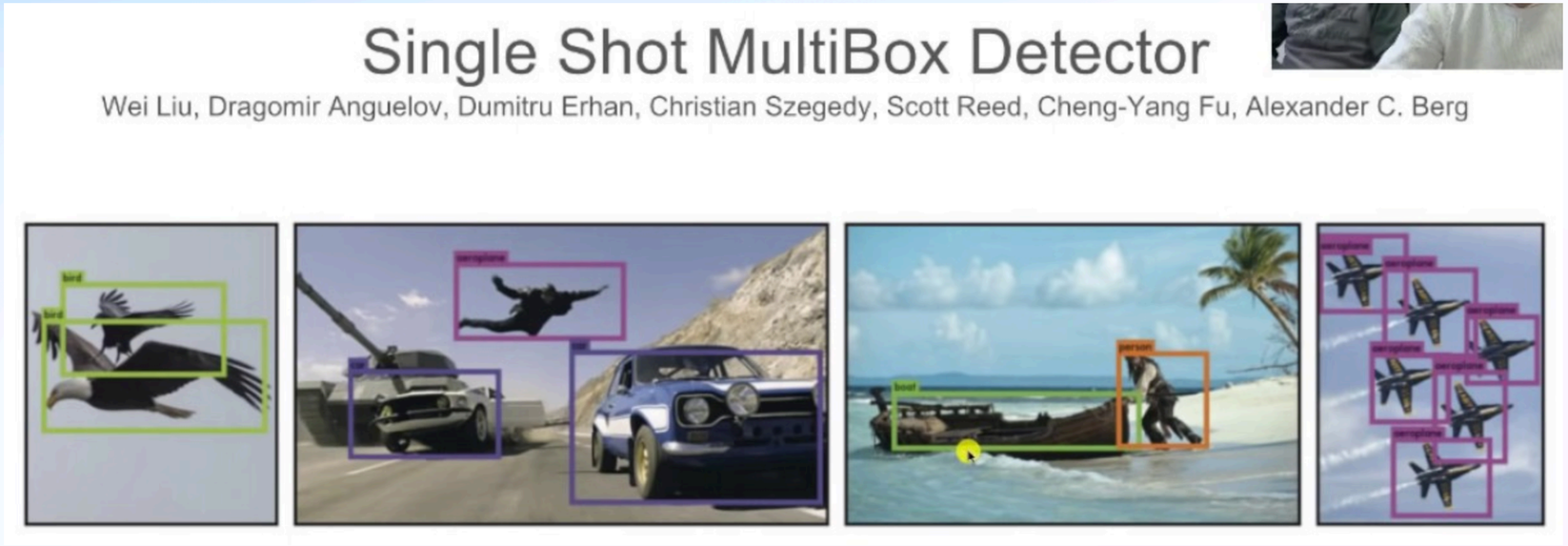
# modernen Alternativen

1. Single Shot Detectors (SSD)
2. You Only Look Once (YOLO)
3. EfficientDet
4. Transformer based …



# modernen Alternativen

**Transformers für Objekterkennung:**

## 4. DETR (DEtection TRansformer)

**DETR verwendet Transformer-Architekturen, um Objekterkennung als eine direkte Zuordnungsaufgabe zu behandeln. Es entfernt die Notwendigkeit von Ankerboxen oder region proposal networks (RPN), wie sie in traditionellen CNN-basierten Modellen verwendet werden.**

**Vorteile:**

- **Vereinfacht die Pipeline der Objekterkennung durch die direkte Zuordnung.**
- **Keine speziellen Region-Vorschläge erforderlich.**

**Nachteile: Langsamer bei der Konvergenz im Training im Vergleich zu anderen Architekturen.**

## 5. Vision Transformers (ViT)

**Beschreibung: Obwohl sie primär für Bildklassifikation entwickelt wurden, haben Vision Transformers auch Anwendung in der Objekterkennung gefunden. Sie sind besonders stark in Kombination mit CNNs oder anderen Attention-Mechanismen.**

**Vorteile:**

- **Können kontextuelle Informationen über das gesamte Bild erfassen.**
- **Gute Leistung bei großen Datensätzen und hoher Rechenleistung.**

**Nachteile: Hohe Rechenkosten, insbesondere bei kleineren Datensätzen.**

# Bild Segmentierung

Eine Verfeinerung des groben Bounding-Box Ansatzes ist (semantische) **Bild Segmentierung:**

**Hier werden einzelne <u>Pixel</u> verschiedenen (Teil)Objekten zugewiesen.**

# Bild Segmentierung

**Segmentierung:**

**Architektur**

**Input (width, height, color) => Magie => Output (width, height, class)  per Pixel!**

**Magie** heisst bei uns:
**Zwischenschichten** Sammeln per **Convolution**:
**Scannen** von **Mustern** mit convolutional Filtern verschiedener Größe (NICHT pro pixel)
**Komplexe** Zusammenhänge in mittleren Schichten!
Nachdem **Zusammenhänge** erkannt wurden kann wieder per Pixel **zugeordnet** werden
(dieser Weisse Punkt gehört zum Fell)

# Bild Segmentierung

1) Bestandteile **nebeneinander** ( Hut + Jacke = Uniform )
2) Bestandteile übereinander
   **one-hot (eine Klasse) => multi-hot** (hierarchisch mehrere **Klassen)**
   **Auge ist Teil vom Gesicht (überlagert)**
3) getrennte Teile ( usb stick + kappe weit weg)
   'fernassoziation'? Ja, prinzipiell in foundation Models (LLM wie GPT)

**Multimodal**: Lerne **Bild**erkennung **zusammen** mit **Text**.

# Bild Segmentierung

## Anwendungsgebiete:

- **Bildverarbeitung in der Medizin:**
  - Segmentierung von **Organen** oder Tumoren in medizinischen Bildern.
    **Diagnose**, Operationen (Roboter)! Gewebe**arten/grenzen** (Unterstützung für Chirurgen)
- **Autonomes Fahren:** Identifikation von **Fahrspuren**, Fußgänger(wege)n und Hindernissen.
- **Umweltschutz:** Überwachung von Landflächen, Vegetation und Wasserflächen in Satellitenbildern.
- **Open Street Map: S**atellitenbild zu Karte mit spezifischen Objekten (POI...)
  - Illegale Plantagen, Schätzung von CO2 Aufnahme, …
- **Object tracking:** Erkannt => Pfad vorhersagen => Verfolgen
  - Verkehrsüberwachung, Sky Net
- **Background/Objekt removal** (=> Generierung morgen)
- **Intern** bei fortgeschrittenen Aufgaben (GPT Bilderklärung etc, Gesichtfinder)
- …

# Bild Segmentierung

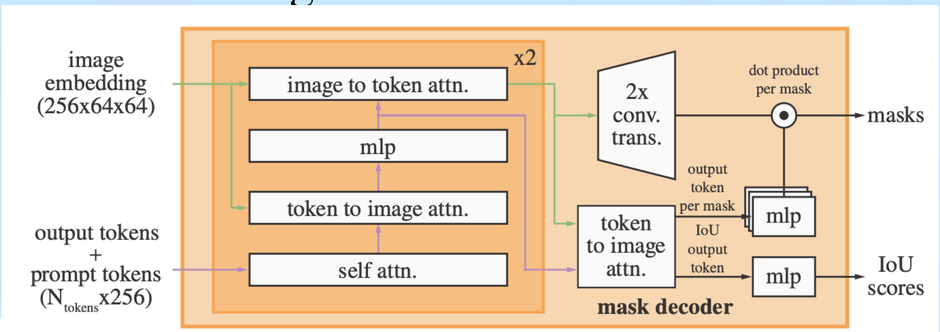
Bei **Bild Segmentierung** werden einzelne Pixel verschiedenen Objekten zugewiesen.

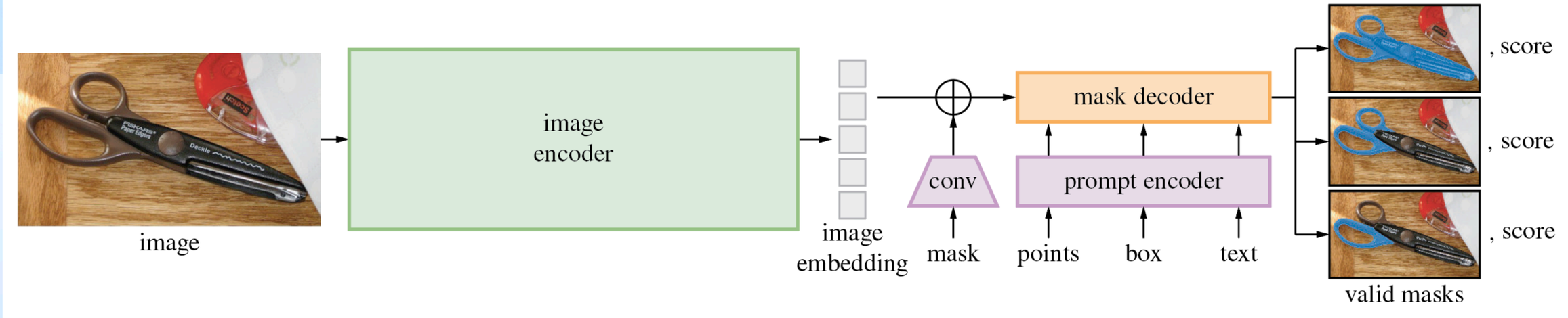
Leider ist dies **sehr rechenaufwändig** so dass wir auf existierende Bibliotheken zurückgreifen müssen:

**Segment Anything**

**https://segment-anything.com/demo (SAM)**

**https://sam2.metademolab.com/demo (SAM2 video segmentation)**




⚠️ Tatsächlich besteht bei modernen Architekturen keine Hoffnung mehr, selbst zu trainieren

Zumindest können wir glücklich sein dass die **Inferenz** bei uns in endlicher Zeit abläuft.

# Foundation Models

**Foundation Models** sind große vortrainierte KI-Modelle, die als Grundlage für eine Vielzahl von Aufgaben verwendet werden können. Sie zeichnen sich durch ihre Fähigkeit aus, allgemeine Muster und Repräsentationen aus riesigen Datenmengen gelernt zu haben, die dann auf spezifischere Aufgaben angepasst (**fine-tuned**) werden können.

**Multimodal Foundation Models**

**LLMs**

**ChatGPT kann man finetunen aber auf deren Servern**

**Open Source LLMs finetunen oder Kontext mitgeben**

**One Super Model**

# Modell Zoo

Für praktische Aufgaben gibt es einen so genannten **Modell Zoo.**

Das ist eine **Sammlung von alten und State of the Art Modellen** zusammen mit Gewichten. Dieses lassen sich typischerweise über **'hub' Bibliotheken** herunterladen.

zB:

**PyTorch Hub** https://pytorch.org/hub/

**Hugging Face Hub** https://huggingface.co/models

**ONNX Model Zoo** https://github.com/onnx/models

**TensorFlow Hub** https://tfhub.dev/

**TensorFlow Model Garden** https://www.tensorflow.org/tfmodels

⚠️💡 Es gibt und überschaubar viele Modelle. Am besten man schaut auf die aktuellen **Benchmarks** um ein gutes auszuwählen.

Papers With Code https://paperswithcode.com

# Modell Zoo

https://paperswithcode.com/task/semantic-segmentation

1. **DeepLabV3**: dilatierte Convolutions und ein Spatial Pyramid Pooling, um die Segmentierungsgenauigkeit zu verbessern.

**2. Segmenter (Vision Transformers)** nutzt ein **Encoder-Decoder-Design,** um globale Bildmerkmale besser zu erfassen. Dies löst einige der Einschränkungen konvolutioneller Netze in Bezug auf lokale Merkmale.

**3. SAM 2 (Segment Anything Model 2)** Mit einer **speicherbasierten Architektur** kann es **Videodaten in Echtzeit** verarbeiten
und liefert dabei **deutlich bessere Ergebnisse als SAM 1**

# Modell Zoo

Segmentierung mit Objektbenennung:

**1. MMsegmentation (OpenMMLab)** umfassende Bibliothek für semantische Segmentierung, die auf PyTorch basiert. Sie unterstützt viele fortschrittliche Modelle wie DeepLab, SegFormer und MaskFormer.

**PaddleSeg (PaddlePaddle)**

**Detectron2** (von Facebook AI Research): Bietet umfassende Funktionen für semantische Segmentierung und Objektbenennung, basierend auf State-of-the-Art-Modellen wie Mask R-CNN.

# LLMs !!