



Transfer Learning

Warum nicht jedes Modell von Grund auf neu erstellt werden sollte

Johannes Kauffmann, Leo Giesen
21.06.2021

Agenda

1. Einführung
2. Vorgehen
3. Hands-on Coding
4. Exkurs
5. Praxisbeispiel

Coding Challenge

Motivation Organisation Seminarthemen Ausblick

Umsetzung

- 1 **Vorlesungen:** Die Vorlesungen werden einige Inhalte des Buches aufgreifen und weiter vertiefen.
- 2 **Seminarbeiträge:** Pro Seminarthema soll ein entsprechender Vortrag (Folien, Code-Beispiele, ...) ausgearbeitet werden, welcher sich an den entsprechenden Abschnitten des Buches orientiert.
 - ▶ **Kleingruppen:** Bearbeitung in Kleingruppen (2-3 Teilnehmer)
 - ▶ **Vortrag:** Ca. 60 (2er Teams) bzw. 90 Minuten (3er Teams); auf Deutsch
 - ▶ **Inhalte:** Abschnitte des Buches sowie ggf. Sekundärliteratur
- 3 **Challenge:** Bearbeitung eines Deep-Learning-Problems in Kleingruppen
 - ▶ **Code + Flash Talk:** Entwicklung eines entsprechenden Deep-Learning-Ansatzes; kurze Präsentation
 - ▶ **Ausarbeitung:** 4 (2er Team) bzw. 6 Seiten (3er Teams); Deutsch oder Englisch
Vorlage: <https://www.acm.org/publications/proceedings-template>
Overleaf: <https://www.overleaf.com/latex/templates/association-for-computing-machinery-acm-sig-conference-proceedings-template/bmvfhcdnxfty>
- 4 **Mündliche Prüfung:** Abschließende Prüfung (individuell)
 - ▶ **Umfang:** 20 Minuten
 - ▶ **Inhalt:** Vorlesungen + alle Seminarvorträge + Deep-Learning-Challenge

Virtuelle Anwesenheit bei allen Veranstaltungen falls möglich :-).

Coding Challenge

3 Challenge: Bearbeitung eines Deep-Learning-Problems in Kleingruppen

- ▶ **Code + Flash Talk:** Entwicklung eines entsprechenden Deep-Learning-Ansatzes; kurze Präsentation
- ▶ **Ausarbeitung:** 4 (2er Team) bzw. 6 Seiten (3er Teams); Deutsch oder Englisch
Vorlage: <https://www.acm.org/publications/proceedings-template>
Overleaf: <https://www.overleaf.com/latex/templates/association-for-computing-machinery-acm-sig-conference-proceedings-template/bmvfhcdnxfty>

- Ziel: Binary Image Classifier → Unterscheidung Pferd vs. Mensch
- Problem: ~1000 Bilder = zu wenig Trainingsdaten 🤔
- Reguläres Trainieren: < 50% accuracy

💡 Habt ihr Ideen? 💡

Vortrainiertes Modell 🚀

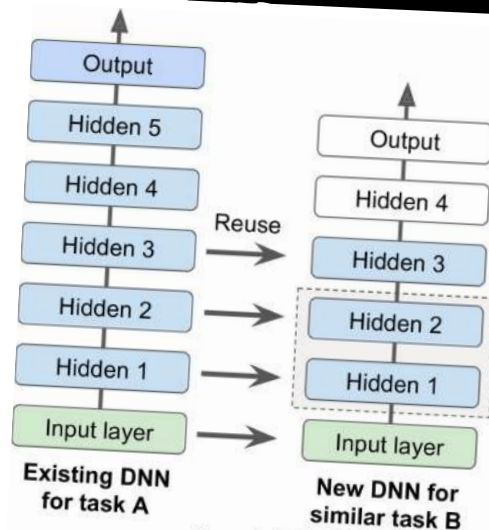
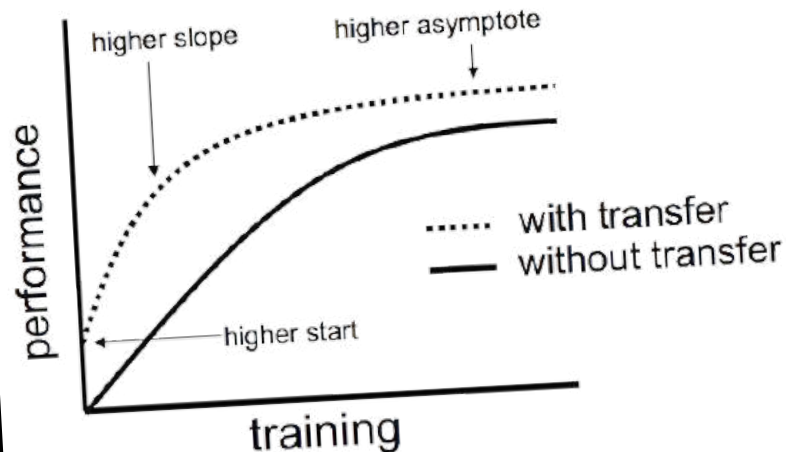


Figure 11-4. Reusing pretrained layers



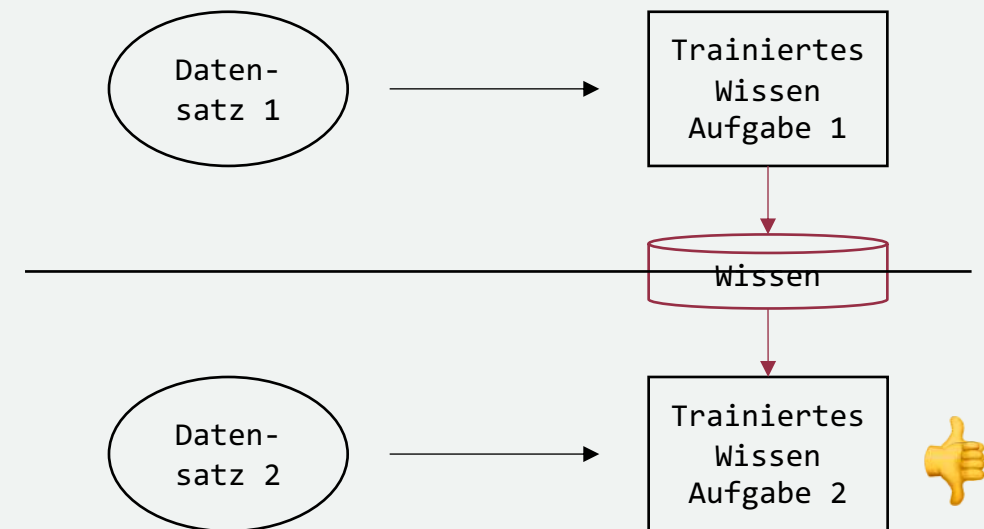
- Nutze Gelerntes von ähnlichem Modell
→ untere Layer übernehmen
(= grundlegendere Zusammenhänge)
- ↑ Ähnlichkeit → mehr Layer übernehmen
- ✗ neg. transfer → 📉 Performance
 - Vortrainiertes Modell nicht ähnlich
 - Kleine & dichte Modelle: aufgabenspezifische Muster

Wie vorgehen?

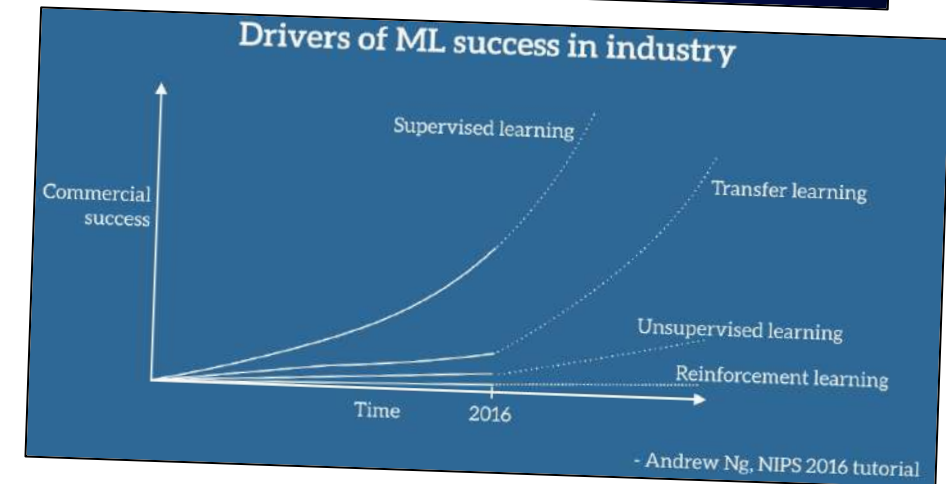
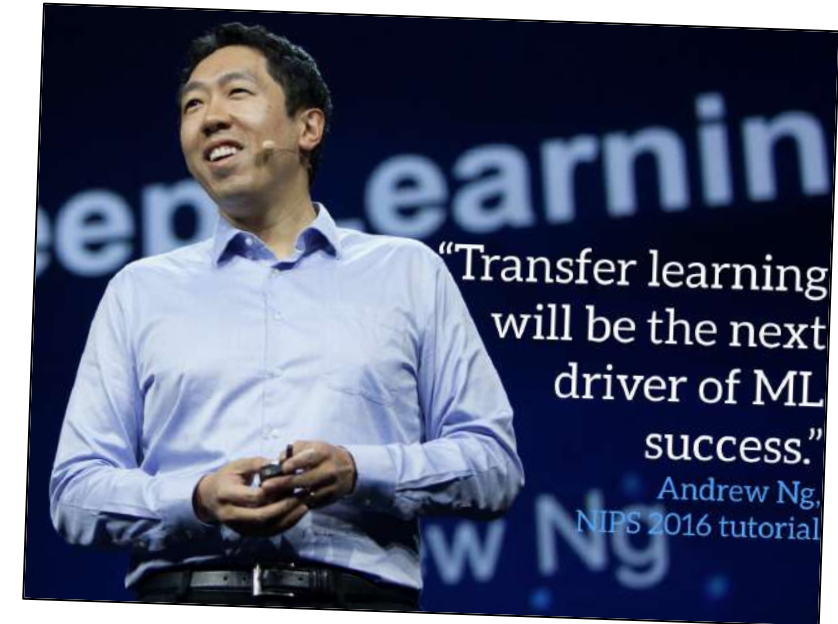
TRADITIONELLES ML

- Isoliertes Lernen von Einzelaufgaben
- Wissen nicht einbehalten
- Lernen erfolgt ohne Berücksichtigung von bereits gelerntem Wissen aus anderen Aufgaben

TRANSFER LEARNING



The next big thing? 🌟



Wie gehen wir vor? 🤔

- Dasselbe Input-Format wird benötigt
- Gelerntes Übertragen = untere Schichten inkl. Gewichte übernehmen
- Anderer Output
- Output Schicht hat zufällig initialisierte Gewichte
→ große Errors → alle Gewichte beeinflusst = vergisst Gelerntes 🤯

1. Preprocessing: auf vortrainierten Modell-Input anpassen

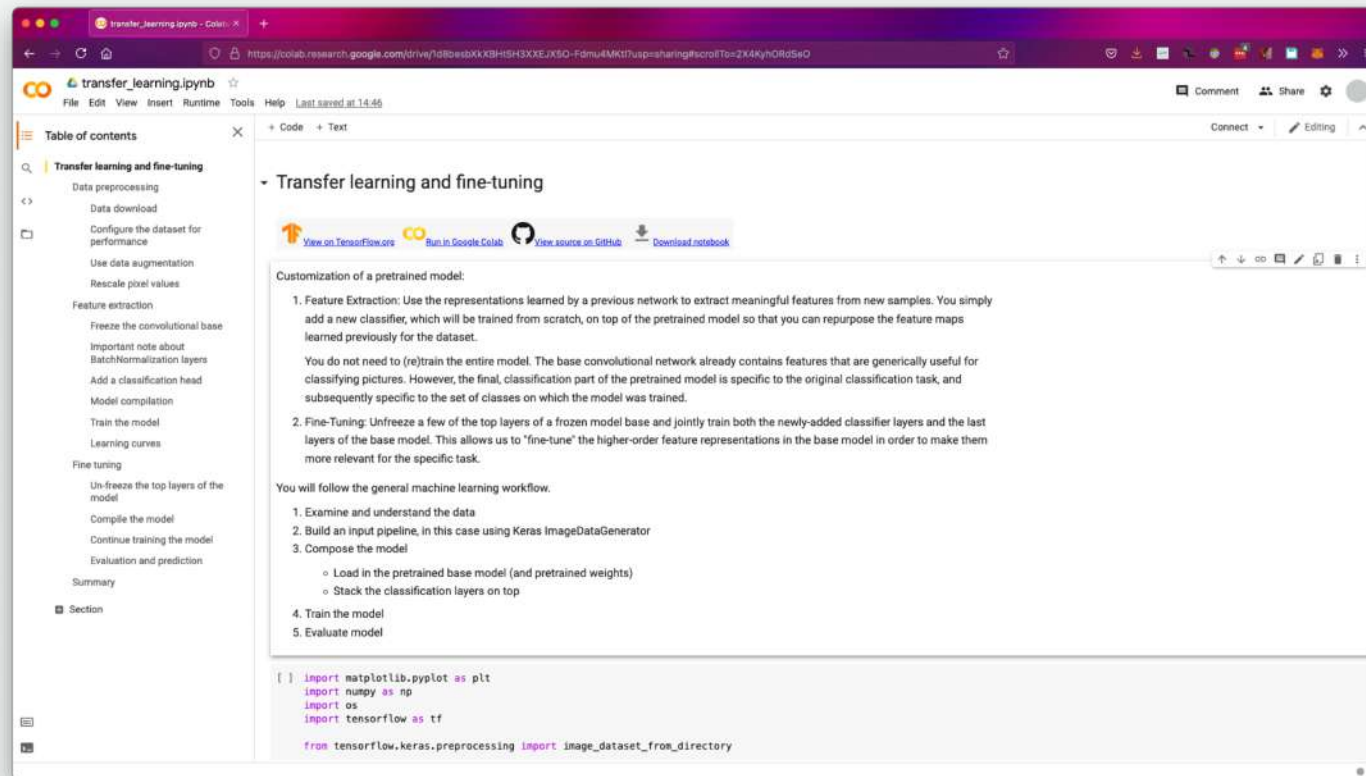
2. Feature extraction

1. Alle übernommenen Gewichte einfrieren
2. Output Schicht (= classifier head) hinzufügen
3. Deren Gewichte trainieren

3. Fine tuning

→ Bei schlechter Performance: ungünstige #Schichten übernommen? → weniger übernehmen




Wie sieht's im Code aus? 🧑💻



<https://colab.research.google.com/drive/1d8besbXkXBHT5H3XXEJX50-Fdmu4MKtI?usp=sharing>

Hands on Coding

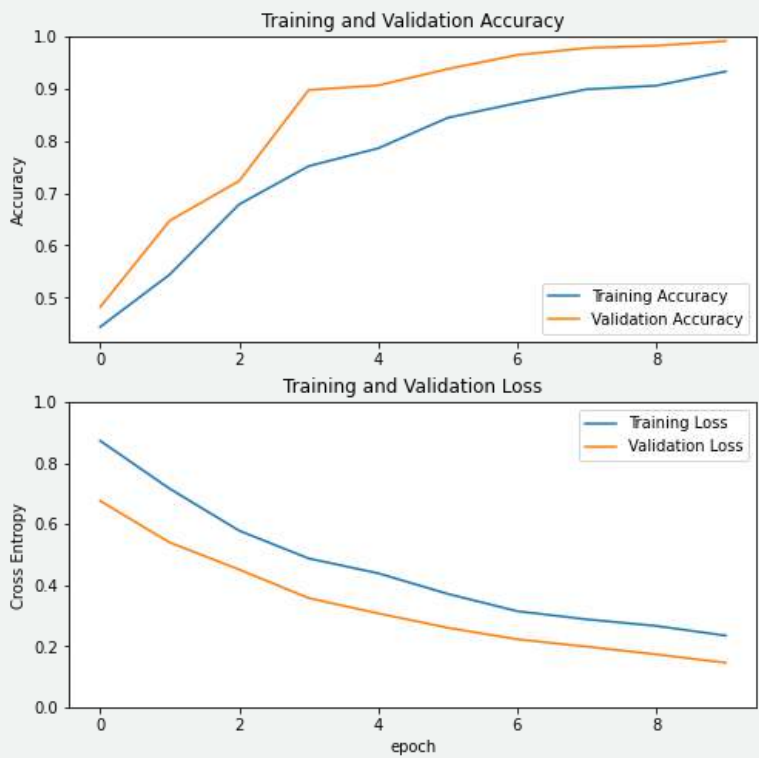
Jetzt seid ihr gefragt  Löst zusammen die Code-Challenge!

- Vorgehen: 2 Gruppen mit 
- Gruppe **A**: vortrainiertem Modell mit IMAGENET 
- Gruppe **B**: zufällige Gewichten
- Code-Lücken ausfüllen [~ 10 min]
- Vergleich der Ergebnisse 



Vergleich der Ergebnisse

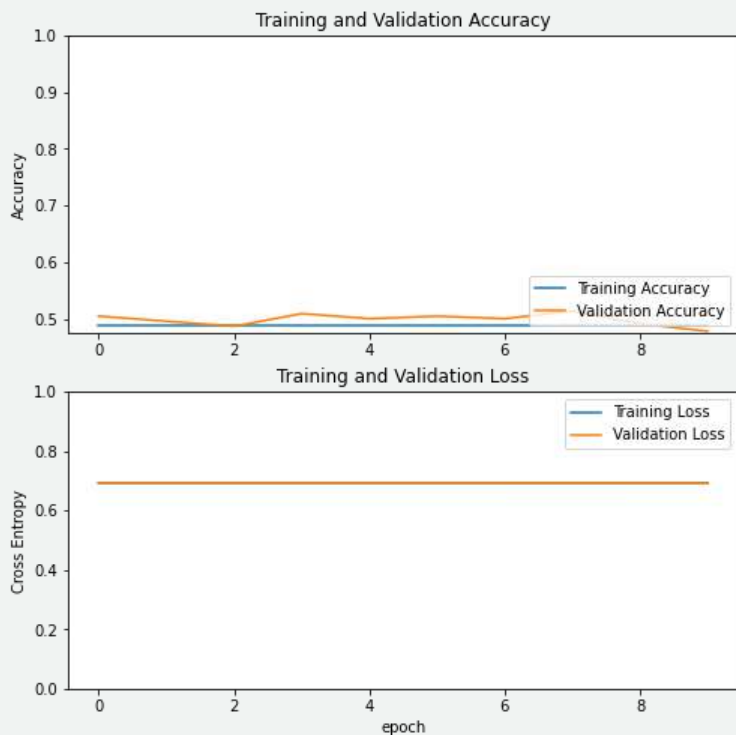
Gruppe A: ImageNet-Gewichte



T: 93,28%
V: 99,11%

T: 23,45%
V: 14,54%

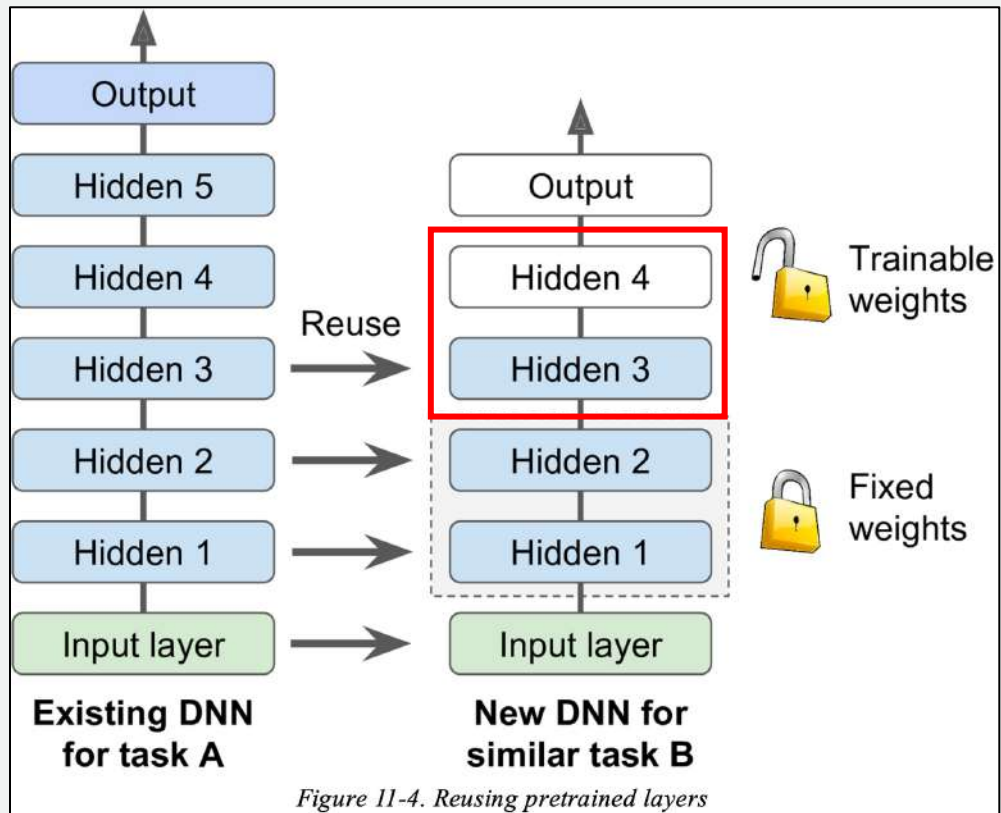
Gruppe B: zufällige Gewichte



T: 46,69%
V: 47,77%

T: 69,31%
V: 69,31%

Fine Tuning



Nicht nur Klassifikator (oberste Schicht) trainieren, sondern mehrere obere Schichten des vorab trainierten Modells

→ Nur wenn vorherige Schritte bereits durchgeführt wurden

→ Keine tiefen Schichten

→ Lernrate verringern

Fine Tuning

```
base_model.trainable = True

# Let's take a look to see how many layers are in the base model
print("#layers in base model:", len(base_model.layers))

#Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all layers before the fine_tune_at layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

# #layers in base model: 154
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.RMSprop(lr=base_learning_rate/10),
              metrics=['accuracy'])
```

Vortrainierte Modelle



```
!pip install --upgrade tensorflow_hub  
  
import tensorflow_hub as hub  
  
model = hub.KerasLayer("https://tfhub.dev/google/nnlm-en-dim128/2")  
embeddings = model(["The rain", "in Spain", "falls", "In the plain!"])  
  
print(embeddings.shape) #(4,128)
```



Interoperabilität durch ONNX

- Open Neural Network Exchange (ONNX): **offenes Format**
- Modelle nicht mehr fest an bestimmtes Framework gebunden
- **Modelle können leicht zwischen Frameworks verschoben werden,**
z.B. bei Marktveränderungen



→ Bessere Nutzung von Hard- und Softwareressourcen

→ Kürzere Entwicklungszeiten

Konvertierung: Keras → ONNX

```
import onnxmltools
from keras.models import load_model

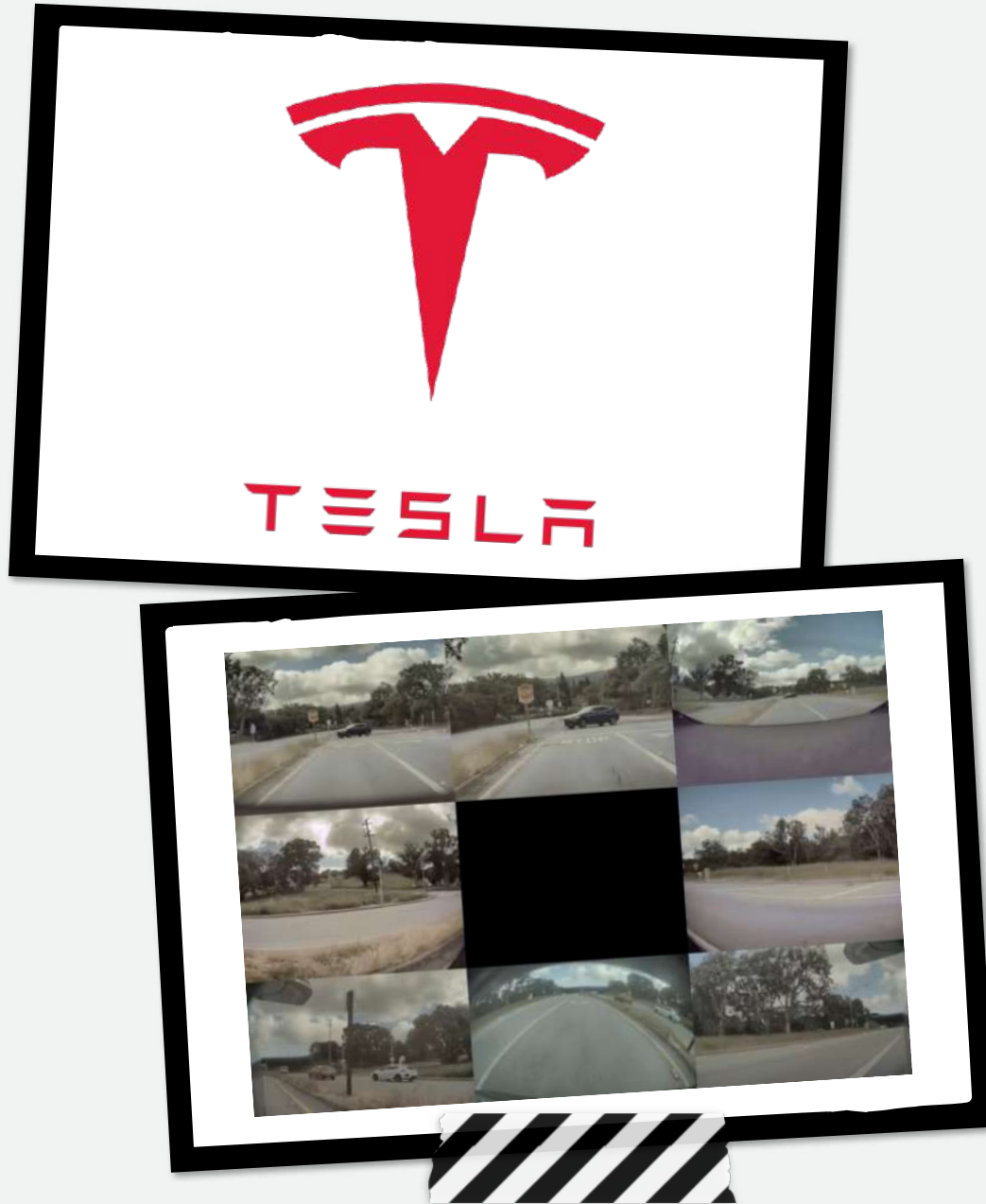
# Update the input name and path for your Keras model
input_keras_model = 'model.h5'

# Change this path to the output name and path for the ONNX model
output_onnx_model = 'model.onnx'

# Load your Keras model
keras_model = load_model(input_keras_model)

# Convert the Keras model into ONNX
onnx_model = onnxmltools.convert_keras(keras_model)

# Save as protobuf
onnxmltools.utils.save_model(onnx_model, output_onnx_model)
```

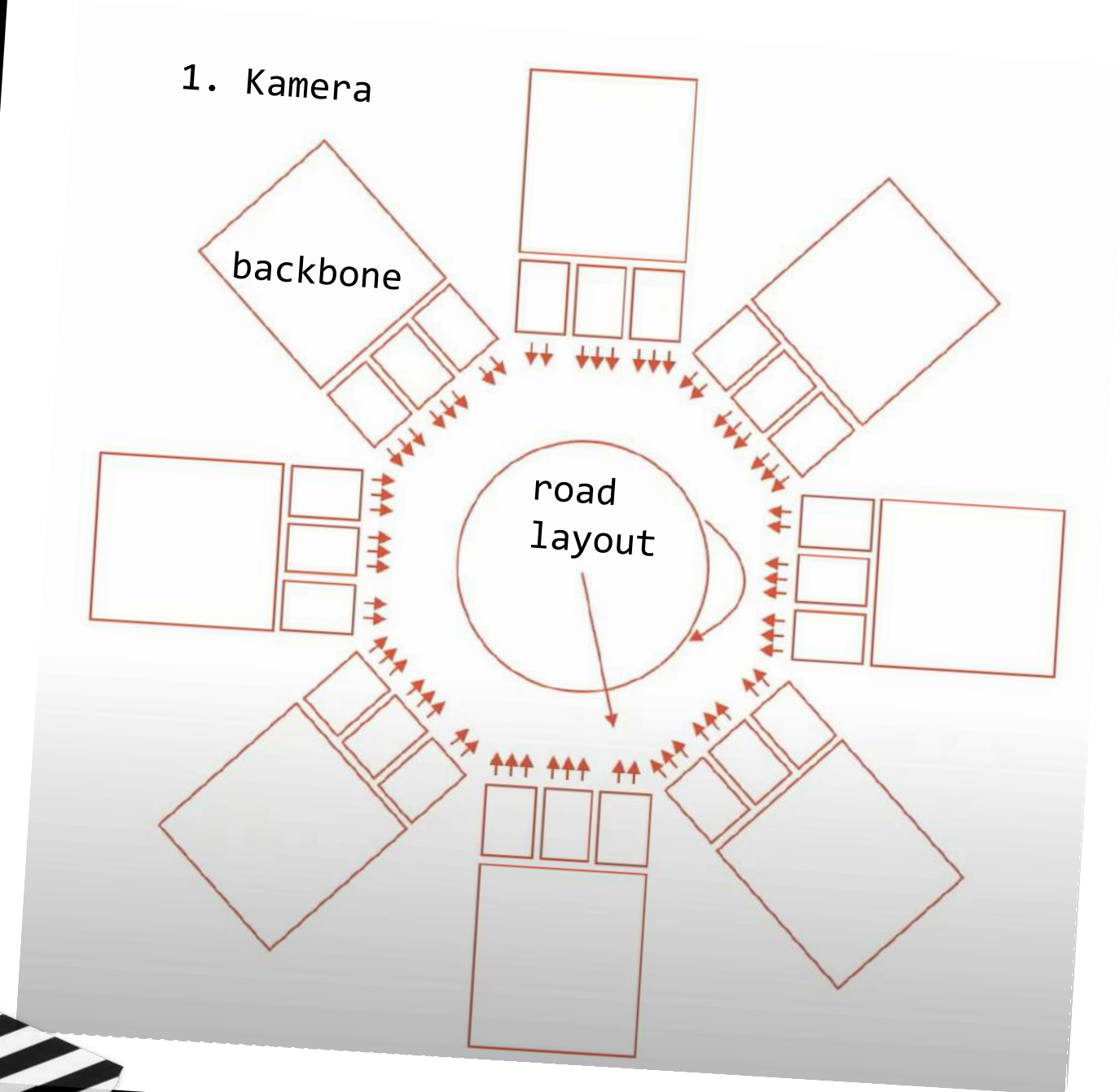
Autonomes Fahren 🚗

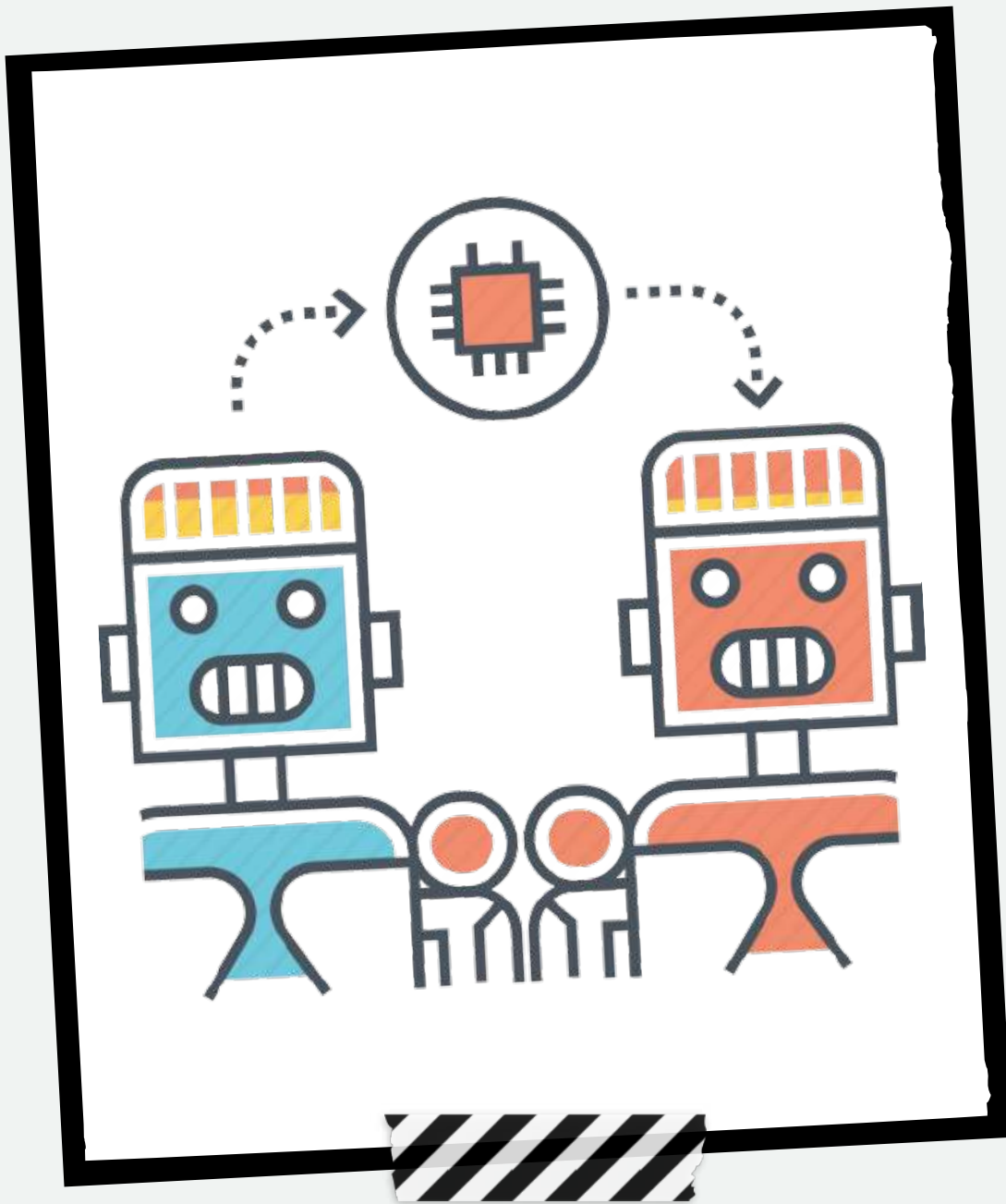
- 8 Kameras
- Rekurrentes neuronales Netz mit 16 Zeitschritten
- Jeder Forward pass: 4096 Bilder

→ Schaffen **nicht einmal 2 GPUs!**

Teslas Lösung 💡

- Architektur: HydraNets
 - 48 neuronal network heads
- Tiefe Schichten (backbones) werden geteilt





Take home message 🏠

- Transfer learning...

... bezeichnet das Wiederverwenden von Schichten vortrainierter Modelle bei einem ähnlichen Problem

- + Spart Trainingszeit
- + Erhöht Performance
- + Ermöglicht performante neuronale Netzwerke schon mit geringer Menge an Trainingsdaten

Abbildungsverzeichnis

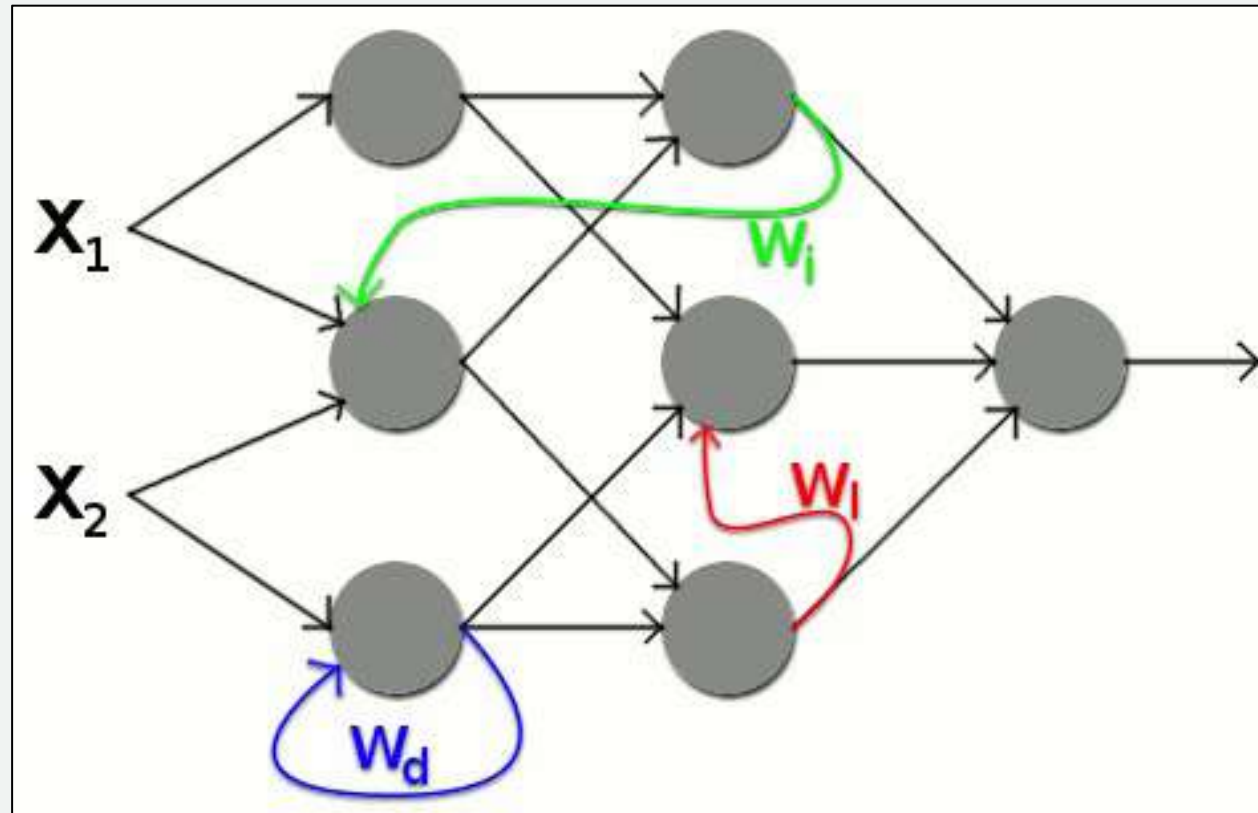
- Bottle PR. (2020). *School Absorbing GIF*. GIPHY.
<https://giphy.com/gifs/wearebottle-school-knowledge-scooping-KCq04k31TnkC2pT5LY>
- Ruder, S. (2017). *Transfer Learning - Machine Learning's Next Frontier*.
<https://ruder.io/transfer-learning/index.html#whytransferlearningnow>
- Dilmegani, C. (2021). *Transfer Learning in 2021: What It Is & How It Works*. AI Multiple. <https://research.aimultiple.com/transfer-learning/>
- Sarkar, D. (2019). *Guide to Transfer Learning with Real-World Applications in Deep Learning*. Third Eye Data. <https://thirdeyeddata.io/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning/>

Literaturverzeichnis

- Asgarian, A. (2018). *An Introduction to Transfer Learning*. Medium. <https://medium.com/georgian-impact-blog/transfer-learning-part-1-ed0c174ad6e7>
- Code mit Flow. (2020). *Was Ist das Transfer Learning? | Künstliche Intelligenz*. YouTube. https://www.youtube.com/watch?v=K_csnXsNN5Q
- Cohen, J. (2020). *Computer Vision at Tesla*. Heartbeat. <https://heartbeat.fritz.ai/computer-vision-at-tesla-cd5e88074376>
- Dilmegani, C. (2021). *Transfer Learning in 2021: What It Is & How It Works*. AI Multiple. <https://research.aimultiple.com/transfer-learning/>
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media, Inc.
- Kaggle. (2018). *Transfer Learning | Kaggle*. YouTube. <https://www.youtube.com/watch?v=mPFq5KMxKVw>
- Luber, S., & Litzel, N. (2019). *Was ist ONNX (Open Neural Network Exchange)?* <https://www.bigdata-insider.de/was-ist-onnx-open-neural-network-exchange-a-851510/>
- Persson, A. (2020). *TensorFlow Tutorial 11 - Transfer Learning, Fine Tuning and TensorFlow Hub*. YouTube. <https://www.youtube.com/watch?v=WJZoywOG1cs>
- Ruder, S. (2017). *Transfer Learning - Machine Learning's Next Frontier*. <https://ruder.io/transfer-learning/index.html#whytransferlearningnow>
- Sarkar, D. (2019). *Guide to Transfer Learning with Real-World Applications in Deep Learning*. Third Eye Data. <https://thirdeyedata.io/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning/>
- TensorFlow. (2021a). *TensorFlow Hub*. <https://www.tensorflow.org/hub>
- TensorFlow. (2021b). *Transfer Learning and Fine-Tuning*. TensorFlow Advanced. https://www.tensorflow.org/tutorials/images/transfer_learning

Backlog

Rekurrentes neuronales Netz

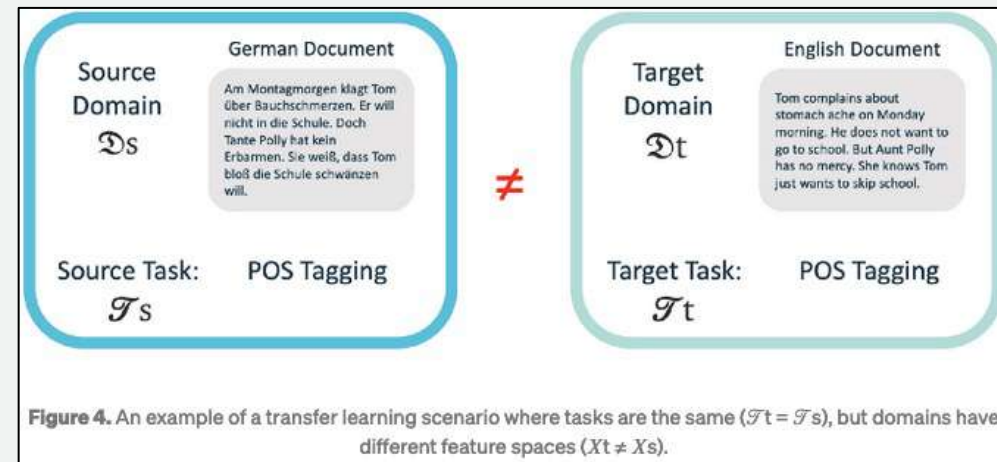


Notationen & Definitionen

- Anwendungsbereich/Domain $\mathcal{D} = \{X, P(X)\}$
 - + Feature space (= Merkmalsraum) X
 - + Marginal probability distribution $P(X)$
where $X = \{x_1, x_2, x_3, \dots, x_n\} \in X$
- Unterschiedliche domains liegen vor, wenn:
 - + $X_t \neq X_s$
 - + Randvt.: $P(X_t) \neq P(X_s)$
- Task $\mathcal{T} = \{Y, f(\cdot)\} = \{Y, P(Y|X)\}$
 - + Label space (= Beschriftungsraum) Y
 - + Training data $\{(x_i, y_i) \mid i \in \{1, 2, \dots, N\}\}$
 - + Predictive function $f(\cdot)$: $f(x_i) = p(y_i|x_i)$
- Unterschiedliche tasks liegen vor, wenn:
 - + $Y_t \neq Y_s$
 - + Wahrscheinlichkeitsvt.:
($P(Y_t|X_t) \neq P(Y_s|X_s)$)

Notationen & Definitionen

- “Given a source domain \mathcal{D}_s and corresponding learning task \mathcal{T}_s , a target domain \mathcal{D}_t and learning task \mathcal{T}_t , **transfer learning** aims to improve the learning of the conditional probability distribution $P(Y_t|X_t)$ in \mathcal{D}_t with the information gained from \mathcal{D}_s and \mathcal{T}_s , where $\mathcal{D}_t \neq \mathcal{D}_s$ or $\mathcal{T}_t \neq \mathcal{T}_s$ ” (Asgarian, 2018)
- Entweder $\mathcal{D}_t \neq \mathcal{D}_s$ oder $\mathcal{T}_t \neq \mathcal{T}_s \rightarrow$ [4 Szenarien](#) (vgl. Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1). doi:10.1186/s40537-016-0043-6)



Kategorien

HOMOGENES TL

- $X_t = X_s$, $Y_t = Y_s$ -> Lücke in Datenverteilung überbrücken:
 $P(X_t) \neq P(X_s)$ und/oder $P(Y_t|X_t) \neq P(Y_s|X_s)$
- Lösungsansätze:
 - + Verbinde Randverteilungsunterschiede:
 $P(X_t) \neq P(X_s)$
 - + Korrigiere bedingte Verteilungsdifferenz:
 $P(Y_t|X_t) \neq P(Y_s|X_s)$
 - + Korrigiere beides

HETEROGENES TL

- $X_t \neq X_s$ (generell ohne Überschneidungen) und/oder $Y_t \neq Y_s$
-> Lücke in Merkmalsraum überbrücken, Problem auf Homogenität reduzieren

