Applied Deep Learning
Transfer Learning and Object Detection

Mark Schutera

Karlsruhe Institute of Technology | ZF Friedrichshafen AG | Hochschule Ravensburg-Weingarten

Course overview

1. Deep Learning Foundations

3. Transfer Learning and Object Detection

5. Segmentation Networks

8. Deep Reinforcement Learning

10. Generative Adversarial Networks

12. Recurrent Neural Networks

Course overview

Project proposal
One page on introduction, methods, dataset
Deadline 3. Lecture

Intermediate presentation
Ten minutes on achievements, problems, next steps
Due 7. Lecture

Final presentation
Code and results
Due 14. Lecture

Final documentation
Paper and code on github or jupyter notebook
Deadline 14. Lecture

Course features

Sli.do

Every question matters.

Get the app.

Ask questions (with slide number) or vote on other students' questions during the lecture.

And give direct feedback.

#TOBEDETERMINED

Questions will be covered immediately or in the next lecture in more depth.

Github

Find slides, tutorials, flashcards and references on Github.

https://github.com/schutera/DeepLearningLecture_Schutera

You found typos, additional material such as links, algorithms, papers, literature or want to contribute to the slides and lecture notes..

..Feel free to contribute, e-mail me.

Course features

Sli.do

Every question matters.

Get the app.

Ask questions (with slide number) or vote on other students' questions during the lecture.

#TOBEDETERMINED

Questions will be covered directly or in the next lecture in more depth.

Github

Typos, additional material such as links, algorithms, paper, literature, lecture notes..

..Feel free to contribute.

Grade Bonus .3

Prepare flashcards based on Ian Goodfellow's Deep Learning Book

- Commit to flashcard set by emailing me, first come first serve
- Must be comprehensive

This lecture in one slide

Object Detection with neural networks

Basic Architecture

Overview state-of-the-art approaches

Datasets and benchmarking

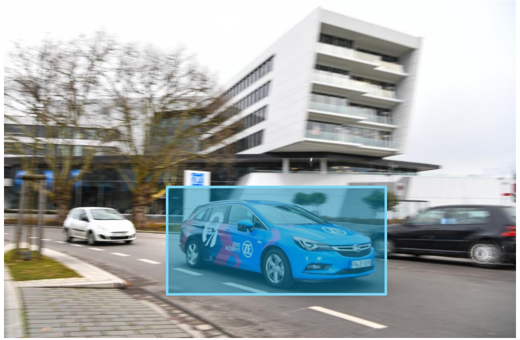
Introduction to Transfer Learning

Transfer Learning with neural networks

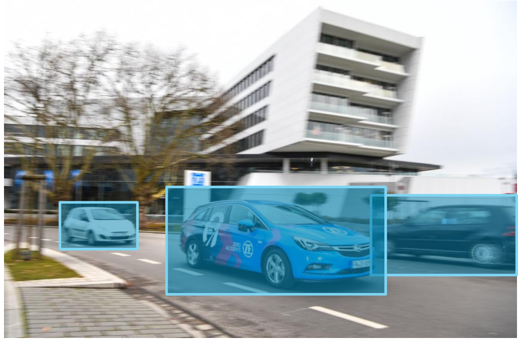
Introduction – Object Detection a problem statement

Classification

Classification
+ Localization

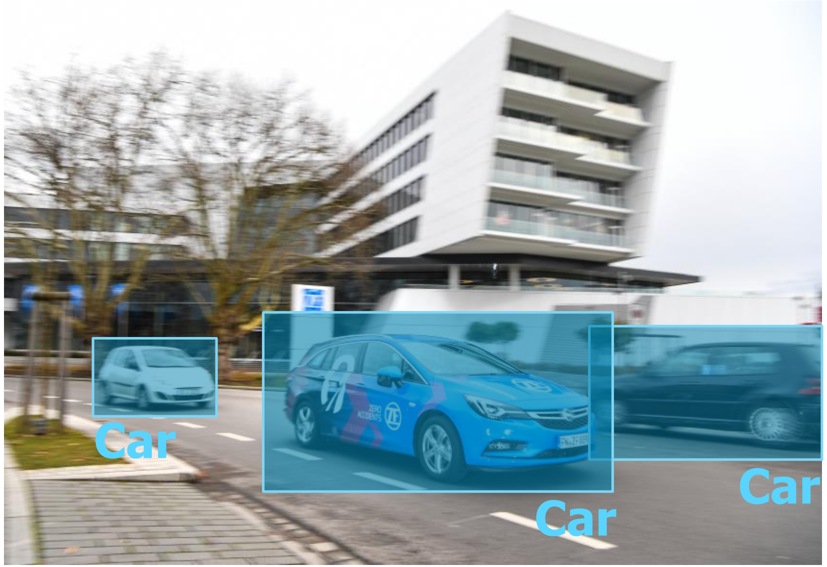


Object Detection



Introduction – Object Detection a problem statement

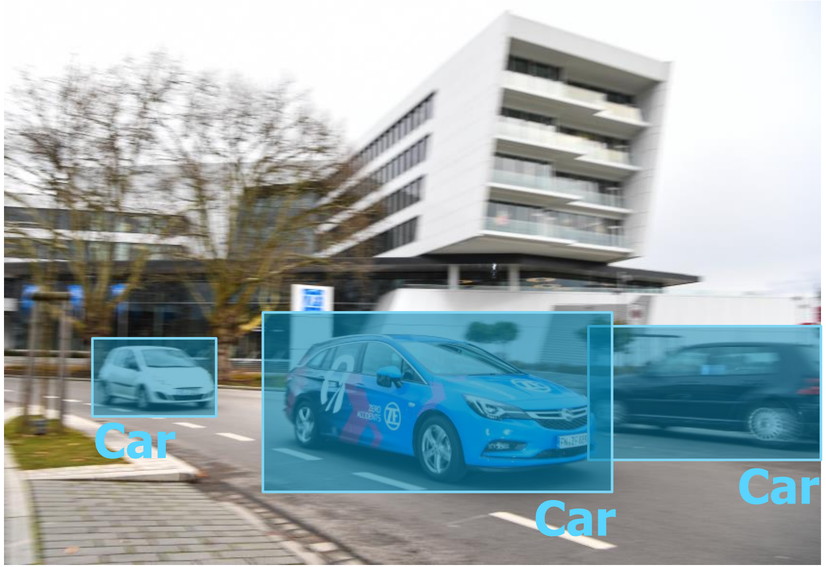
Object detection involves detection and localization of instances of objects from a particular class in a sensor measurement, such as a camera image.



Neural Network Object Detection - Basic structure

Object Detection Approach

1. Establish Bounding Box Hypotheses
2. Generate Features for each Bounding Box
3. Classify and give a confidence for the objects based on the Features

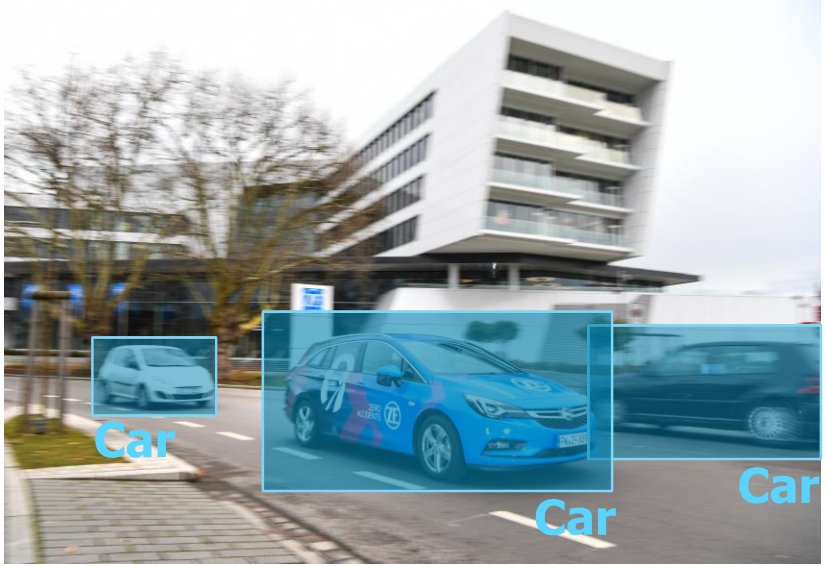


Neural Network Object Detection - Basic structure

Object Detection Approach

1. Establish Bounding Box Hypotheses
2. Generate Features for each Bounding Box
3. Classify and give a confidence for the objects based on the Features

This can also be done with conventional approaches, such as sliding windows combined with classification.



Neural Network Object Detection - Basic structure

YOLO: You Only Look Once

Discretization of the input image into a SxS grid.

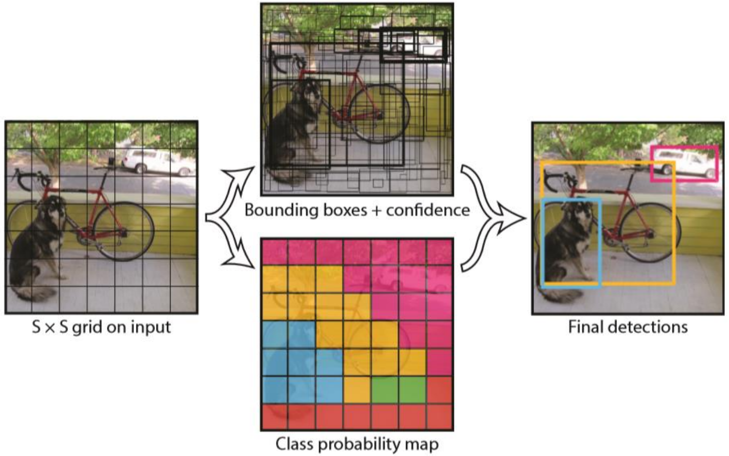
Fully Convolutional Network predicts B default bounding boxes per grid cell, with confidence.

Confidence is based on the Intersection over Union of the Bounding Boxes with each grid cell.

Closing a Non-Maxima Supression is used to prune the detections to the final detections.

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/



https://arxiv.org/abs/1506.02640

SSD: Single Shot MultiBox Detector

Discretization of the input image into a SxS grid.

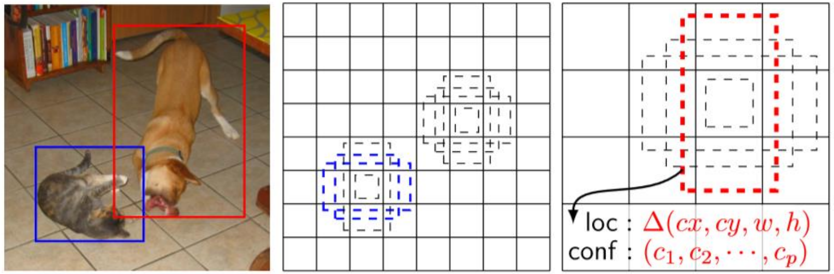
Fully Convolutional Network predicts B default bounding boxes per grid cell, with confidence.

Final detections are chosen with respect to their confidence

SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3], Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com, [4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

https://arxiv.org/abs/1512.02325

RFCN: Region-based Fully Convolutional Network

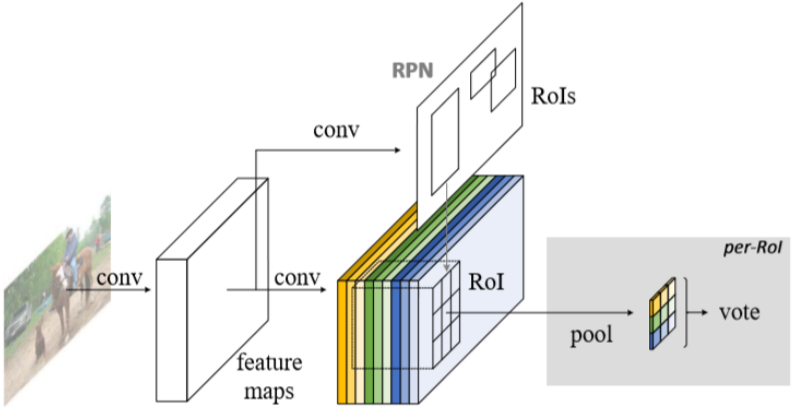
Two stage object detection.

First stage is Fully Convolutional Network, such as ResNet-101 for region proposals

Second stage does classification on the max-pooled proposed regions

R-FCN: Object Detection via Region-based Fully Convolutional Networks

Jifeng Dai Microsoft Research
Yi Li* Tsinghua University
Kaiming He Microsoft Research
Jian Sun Microsoft Research



https://arxiv.org/abs/1605.06409

Object Detection with neural networks

TensorFlow Model Zoo

Collection of detection models pre-trained on different object detection datasets.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Common Objects in Context

COCO-Detection has 200k images with bounding boxes or pixel-wise labels

Annotations

80 object categories (person, elephant, etc.), as well as captions.

Number of samples

200000 bounding box level annotations

Metric

Average Precision, AP across scales, Average Recall, AR across scales

https://github.com/nightrome/cocostuff

PASCAL Visual Object Classes

For each of twenty object classes predict the presence/absence of at least one object of that class in a test image.

Annotations

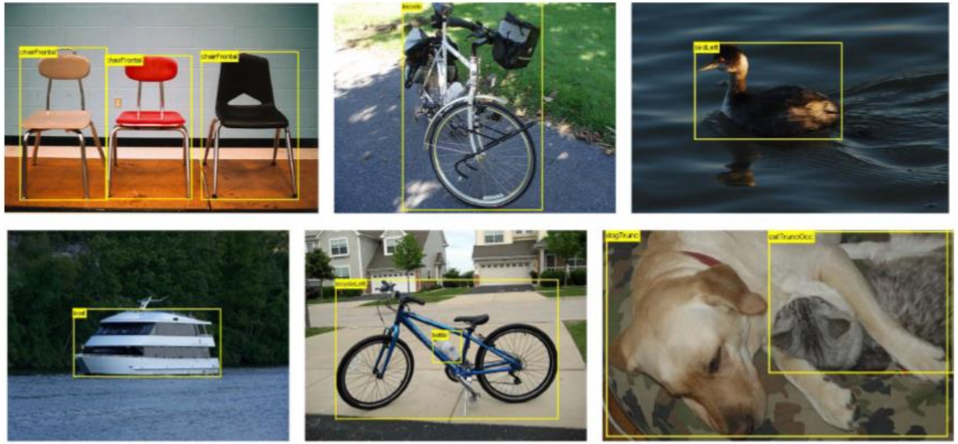
20 object classes (Person, Bicycle, etc.)

Number of samples

11540 bounding box level annotations

Metric

Average Precision, AP across scales,
Average Recall, AR across scales



http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham15.pdf

KITTI

We take advantage of our autonomous driving platform Annieway to develop novel challenging real-world computer vision benchmarks.

Annotations

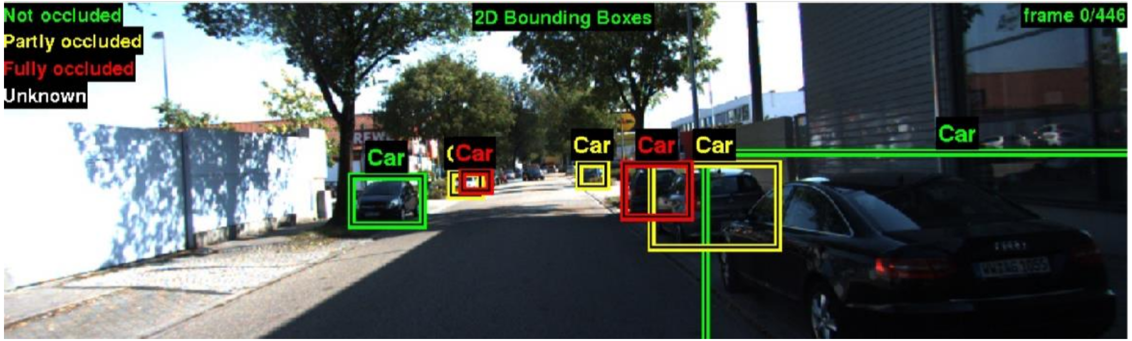
2D bounding box annotations with classes

Number of samples

7481 training images and
7518 test images

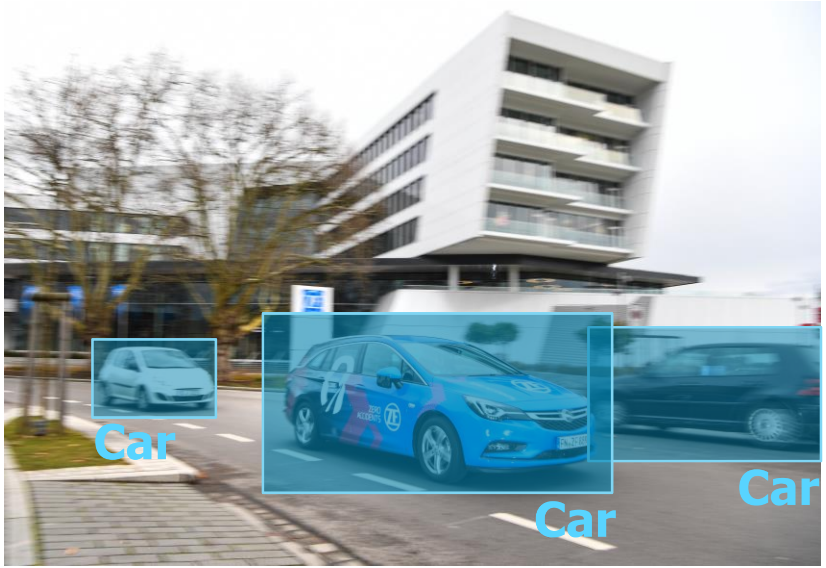
Metric

Average Precision,
Average Recall, both across scales



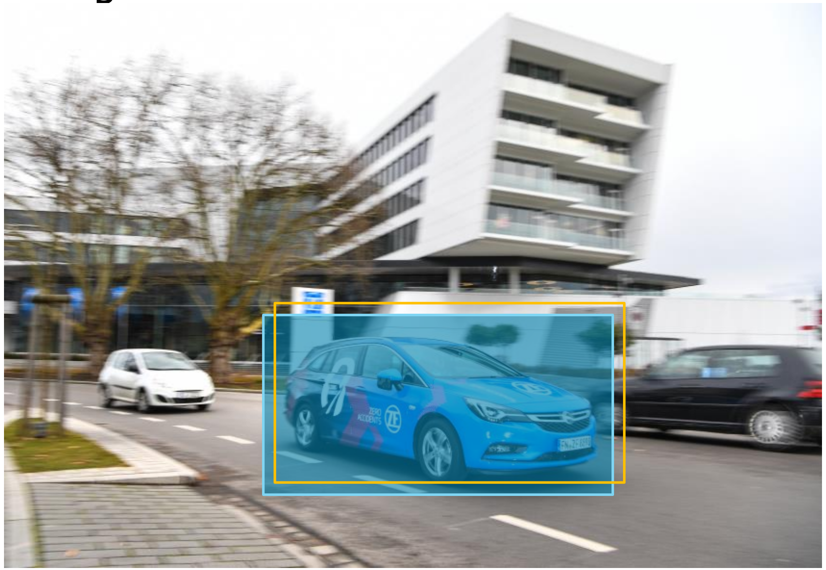
http://www.cvlibs.net/publications/Geiger2013IJRR.pdf

So how do we evaluate our Object Detectors?



So how do we evaluate our Object Detectors?

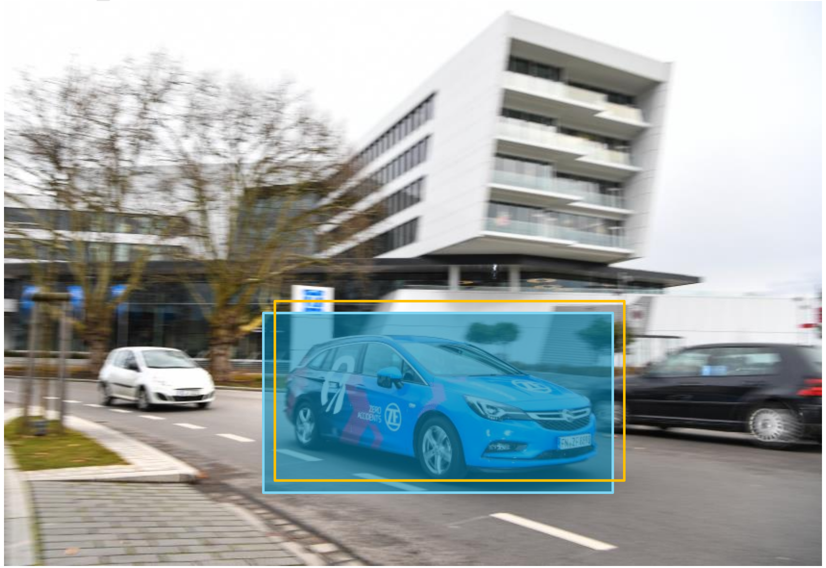
Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.



So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Filter predictions based on
the object detectors **confidence**



So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Assign true positives based on **Intersection over Union** with the ground truth

$$IoU(A,B) = \frac{A \cap B}{A \cup B}$$

$$IoU \geq p, p \in (0,1)$$

So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Assign true positives based on
Intersection over Union

```
prediction_prob = np.asarray([0.23, 0.37, 0.59, 0.84, 0.10, 0.92, 0.57, 0.49, 0.67, 0.51])
# Threshold for assignment
p = 0.5

# Define ground truth and true positives (1) and false negatives (0) of the prediction
groundtruth = np.asarray([1, 1, 1, 1, 1, 1, 1, 1, 0, 0])
prediction = (prediction_prob > p)*1
print(prediction)
# [0 0 1 1 0 1 1 0 1 1]
```

Precision

Detections over all Predictions

True Positives over True Positives and False Positives

$$mAP = \frac{1}{|c|} \sum_c \frac{TP(c)}{TP(c) + FP(c)}$$

```
# -------------------------------------------------------------------------------
# 1. Precision
P = sum(prediction*groundtruth) / sum(prediction)
print('Precision: ', P)
# Precision: 0.66

# -------------------------------------------------------------------------------
```

Recall

Detections over the number of Object Instances.
True Positives over True Positives and False Negatives.

$$mAR = \frac{1}{|c|} \sum_{c} \frac{TP(c)}{TP(c) + FN(c)}$$

```
# ----------------------------------------------------------------------------------------------------------
# 2. Recall
R = sum(prediction*groundtruth) / sum(groundtruth)
print('Recall: ', R)
# Recall: 0.5

# ----------------------------------------------------------------------------------------------------------
```

F1-Score

Harmonic mean between recall and precision
Harmonic mean lays more emphasis on the weaker metric, than the arithmetic would.

$$F1 = 2 \frac{mAP\, mAR}{mAP + mAR}$$

```
# ---------------------------------------------------------------------------------------------------
# 3. F1-Score
# Harmonic mean
F = 2*R*P / (R+P)
print('F1-Score: ', F)
# F1-Score:  0.57

meanPR = (R+P) / 2
print('Arithmetic mean: ', meanPR)
# Arithmetic mean:  0.58

# ---------------------------------------------------------------------------------------------------
```

Confusion Matrix

The comparison of the model's predictions and the correct values in the form of a contingency table

```
# ----------------------------------------------------------------------------------------------------------------
# 4. Confusion matrix
CM = tf.confusion_matrix(labels=groundtruth,
                         predictions=prediction)

with tf.Session() as sess:
    print('Confusion Matrix: ', sess.run(CM))
# Confusion Matrix:
# [ [0 2]
#   [4 4] ]
#
# [ [true_negatives  | false_positives]
#   [false_negatives | true_positives ] ]

# ----------------------------------------------------------------------------------------------------------------
```

Confusion Matrix

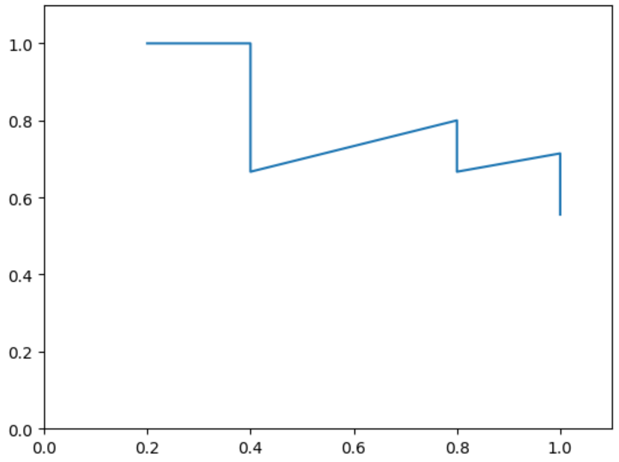
Which number in the matrix does never apply, when dealing with Object Detectors?

```
# ------------------------------------------------------------------------------------------
# 4. Confusion matrix
CM = tf.confusion_matrix(labels=groundtruth,
                         predictions=prediction)

with tf.Session() as sess:
    print('Confusion Matrix: ', sess.run(CM))
# Confusion Matrix:
# [ [0 2]
#   [4 4] ]
#
# [ [true_negatives  | false_positives]
#   [false_negatives | true_positives ] ]

# ------------------------------------------------------------------------------------------
```

Receiver Operating Characteristic (ROC)

The ROC curve displays the trade off between true-positive-rate and true-negative-rate.

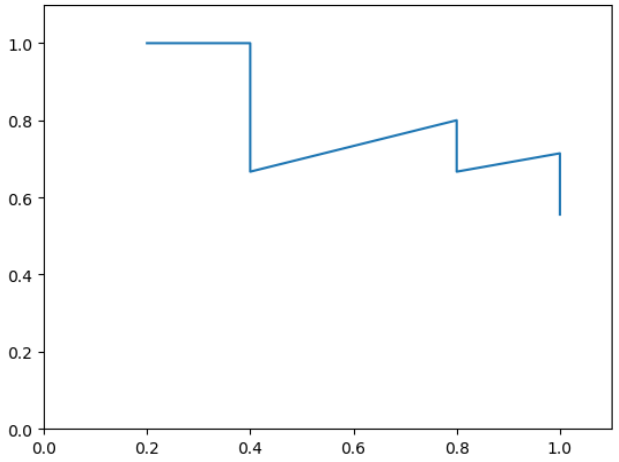
It can be seen as a matter of optimizing a threshold value with respect to a functional requirement.



Precision Recall Curve (PRC)

The PR curve displays the trade off between Precision and Recall.

It can be seen as a matter of optimizing a threshold value with respect to a functional requirement.

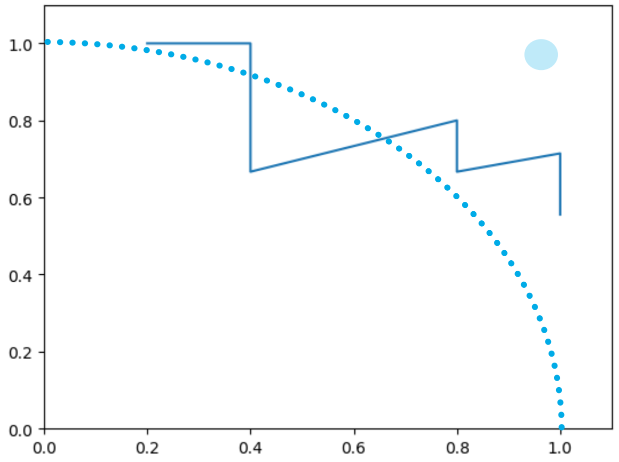


Receiver Operating Characteristic (ROC)

The ROC curve displays the trade off between Precision and Recall.

It can be seen as a matter of optimizing a **threshold** value with respect to a functional requirement.

- Intersection over Union
- Confidence



References

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *14th european conference on computer vision*, pages 21–37, 2016.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[9] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

This lecture in one slide

Object Detection with neural networks

Introduction to Transfer Learning

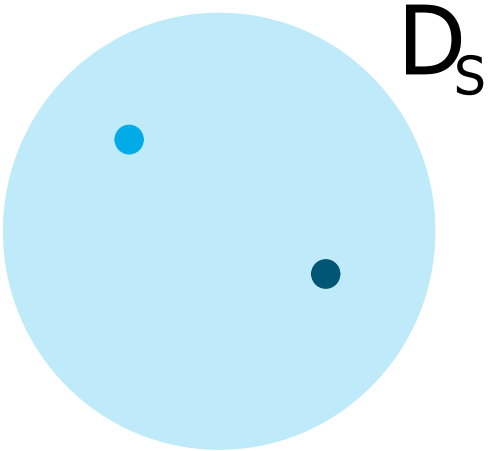
Transfer Learning a problem statement

Settings for Transfer Learning

Transfer Learning with neural networks

Introduction – Transfer Learning a problem statement

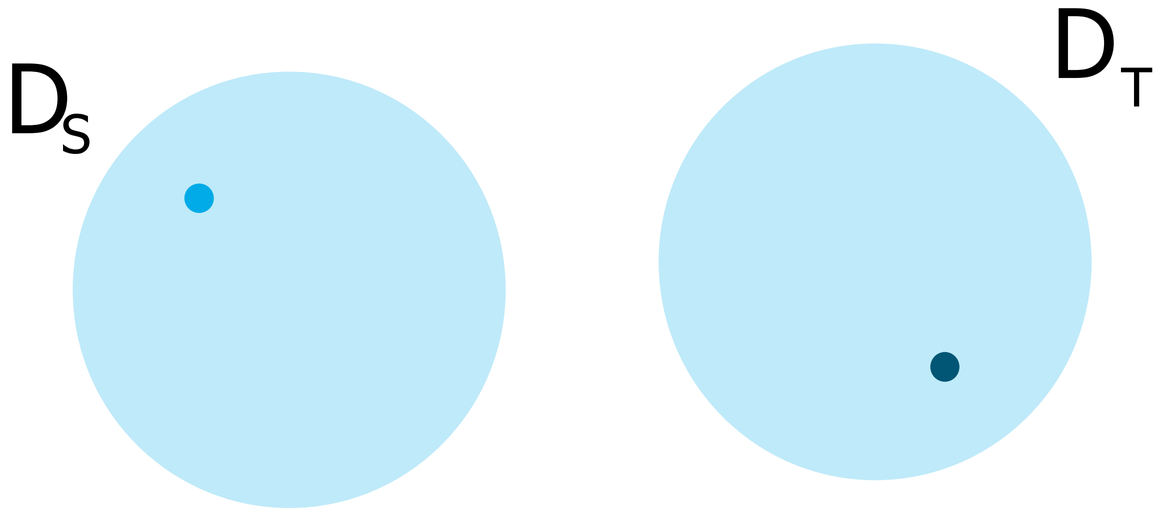
In machine learning it is assumed that the training data and the data received during inference are drawn from the same distribution and domain.



Introduction – Transfer Learning a problem statement

In machine learning it is assumed that the training data and the data received during inference are drawn from the same distribution and domain.

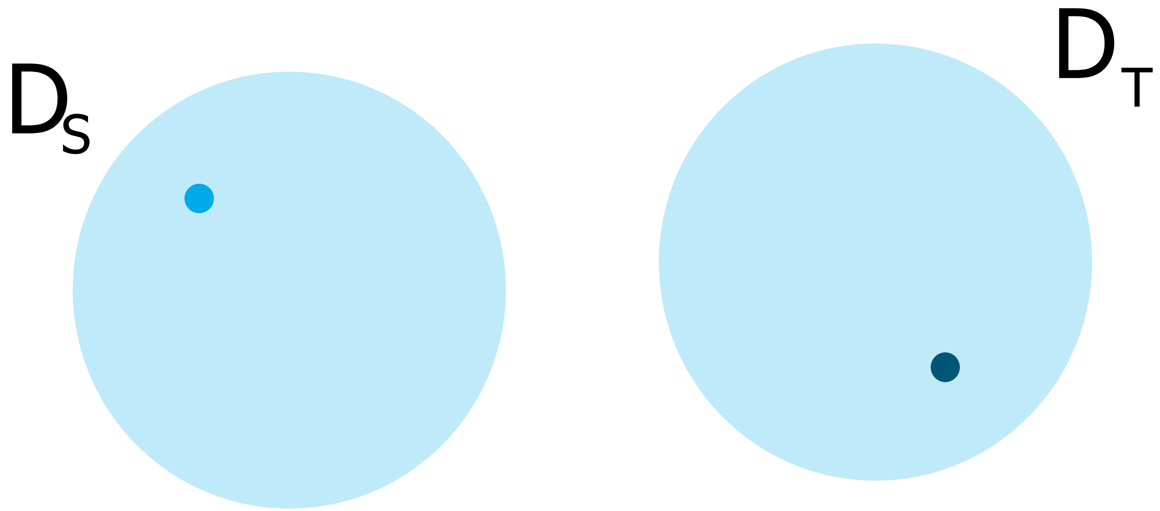
When facing real-world applications this assumption is regularly violated (e.g. day and night, new car models, etc.)



Introduction – Transfer Learning a problem statement

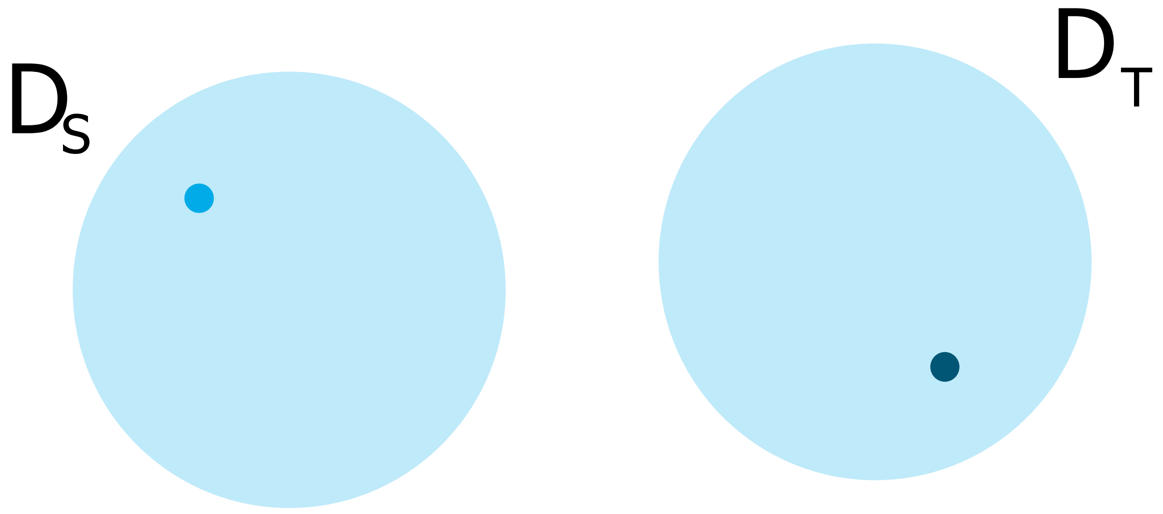
When facing real-world applications this assumption is regularly violated (e.g. day and night, new car models, etc.)

We distinguish between source and target domain.



Introduction – Transfer Learning a problem statement

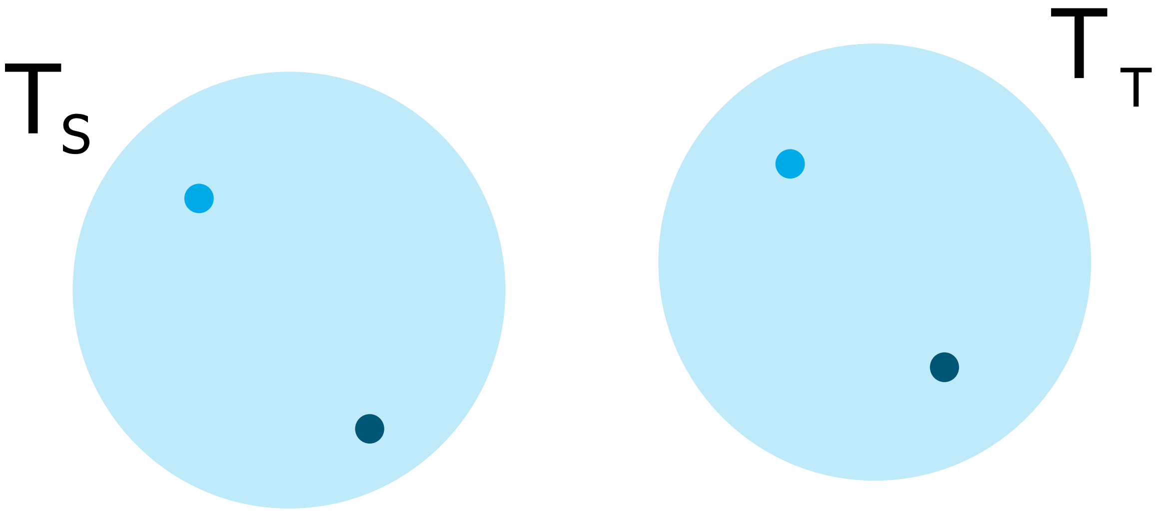
Definition 1.1 ***(Domain).*** *A domain D consists of a feature space $\mathscr{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathscr{X}$. The source domain data is denoted as $D_S = \{(x_{S_1}, y_{S_1}), \ldots, (x_{S_n}, y_{S_n})\}$. The target domain data is denoted as $D_T = \{(x_{T_1}, y_{T_1}), \ldots, (x_{T_n}, y_{T_n})\}$ [19].*



Introduction – Transfer Learning a problem statement

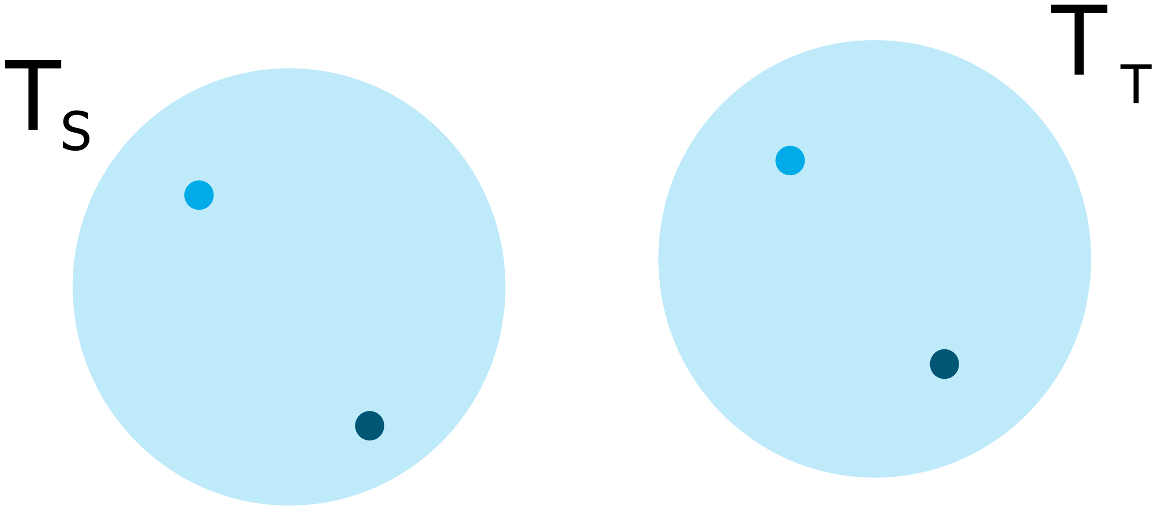
Machine learning models are trained with respect to a task T such as object detection, segmentation, etc.

These can also vary between source and target.



Introduction – Transfer Learning a problem statement

Definition 1.2 ***(Task).*** *A task T consists of a label space $\mathscr{Y}$ and an objective prediction function $f(.)$, denoted by $T = \{\mathscr{Y}, f(.)\}$. $f(x)$ can be written as $P(y|x)$ [19].*



Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

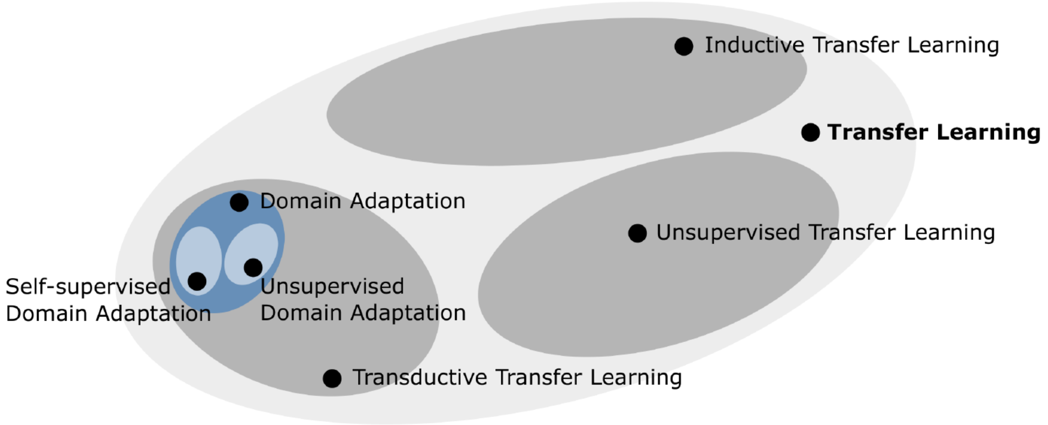


Figure 1.5: Overview of the different transfer learning settings: Inductive transfer learning, unsupervised transfer learning, and transductive transfer learning.

Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

- **Inductive** Transfer Learning
- **Unsupervised** Transfer Learning
- **Transductive** Transfer Learning

Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

Inductive Transfer Learning

Labels are available in the source and target domain.
The target task is different from the source task. Can be understood as task adaptation.

Unsupervised Transfer Learning
Transductive Transfer Learning

Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

Inductive Transfer Learning

Unsupervised Transfer Learning

There are no labels available. And the tasks in source and target are different.
Think clustering approaches.

Transductive Transfer Learning

Introduction – Transfer Learning a problem statement

Transfer Learning is to be understood with respect to the concerned domains and their combination with a task setting

Inductive Transfer Learning
Unsupervised Transfer Learning

Transductive Transfer Learning

There are only labels in the source domain and none in the target domain. The tasks are the same in both domains. Can be understood as domain adaptation.

References

[1] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

This lecture in one slide

Object Detection with neural networks
Introduction to Transfer Learning

Transfer Learning with neural networks

Fine Tuning
Freezing Weights
Shared Latent Feature Spaces
Pseudo Labels
Overview state-of-the-art approaches
Datasets and Benchmarking
Deep dive Day2Night

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a **generic model of the visual world**.

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world.

You can then take advantage of these learned feature maps without having to start from scratch training a large model on a large dataset.

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

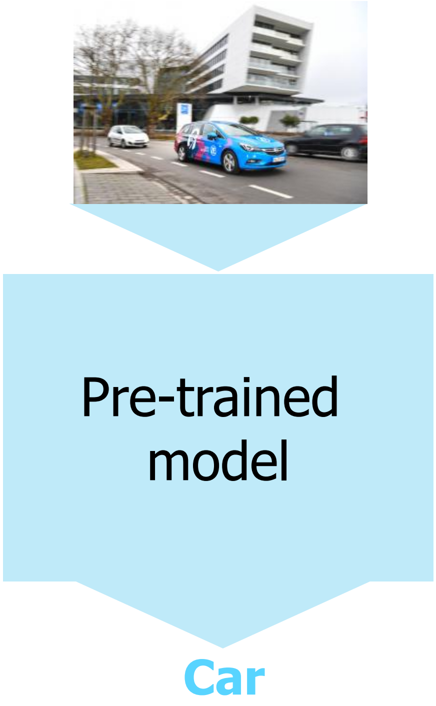
Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on source domain for a source task.

Think **Object Detection Model Zoo**

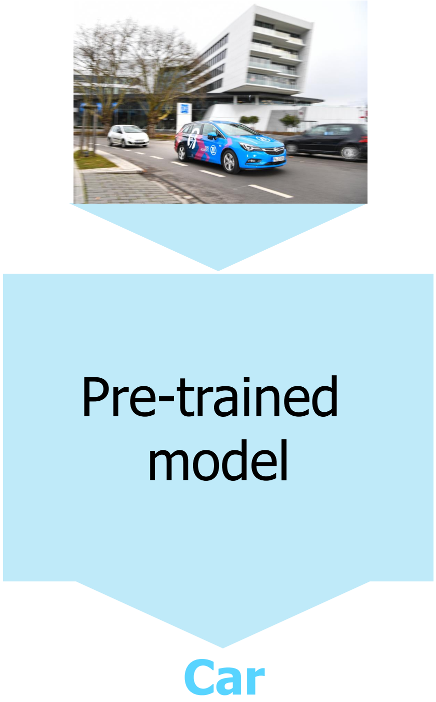


https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

The source domain is usually a **large dataset**.



Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

The source domain is usually a **large dataset**.

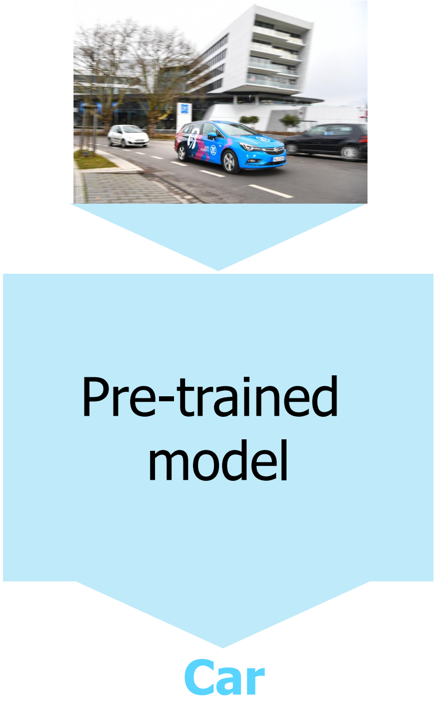
Think **COCO**, **ImageNet**, **PASCAL VOC**

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

You either use a **pre-trained model as it is**.



Transfer Learning with Neural Networks - Fine Tuning

You either use a pre-trained model as it is.

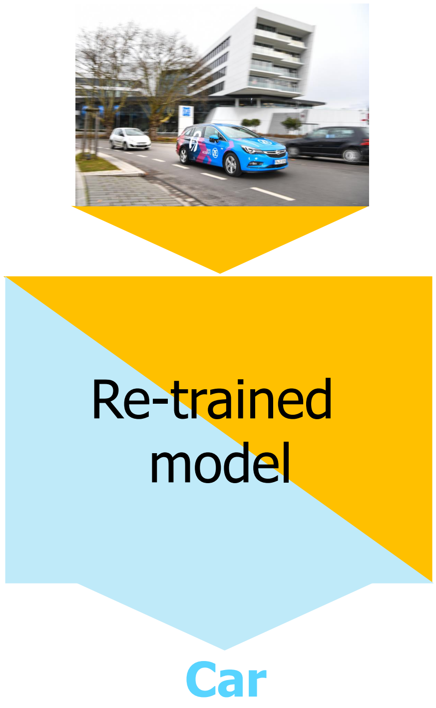
- No additional training
- Usually **performance drops**

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

You can **fine-tune** a **pre-trained model** by retraining on additional data.



Transfer Learning with Neural Networks - Fine Tuning

You can fine-tune a pre-trained model by retraining on additional data.

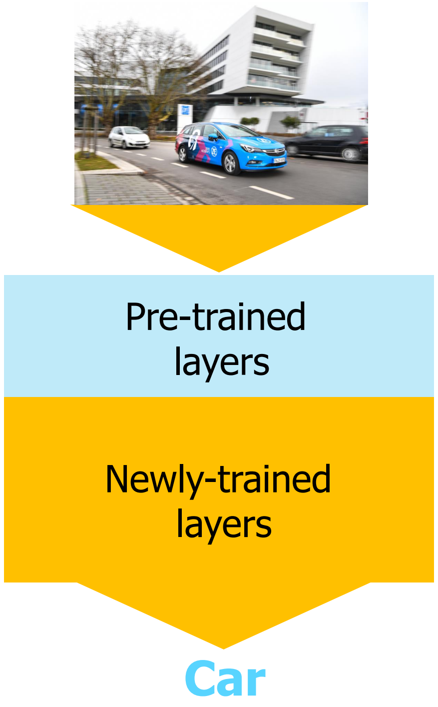
- Additional work to set up retraining pipeline
- Usually performance drops in the source domain
- **Performance improvement** in the target domain

Car

Transfer Learning with Neural Networks - Feature Extraction

You can **repurpose** the **pre-trained layers** as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

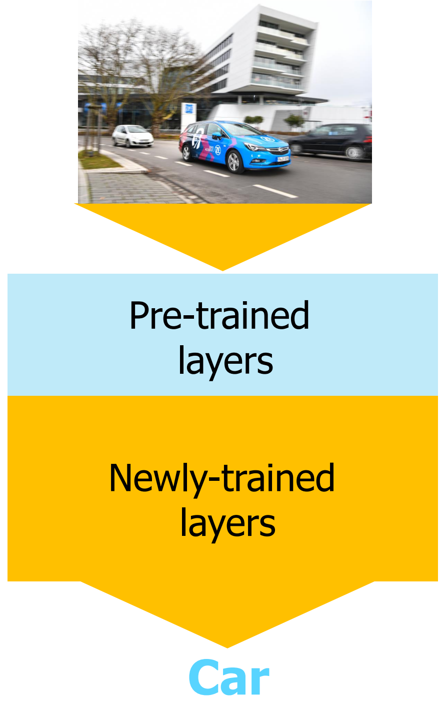


Transfer Learning with Neural Networks - Feature Extraction

You can repurpose the pre-trained layers as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

- Additional work to assemble layer structure
- **Regularization** effects
- **Performance improvement** in the target domain
- **Output** layer becomes **adjustable** (such as for adding a class)



Transfer Learning with Neural Networks - Shared Latent Feature Space

So far the transfer was **supervised**, meaning we were in possession of labels to deploy in the target domain.

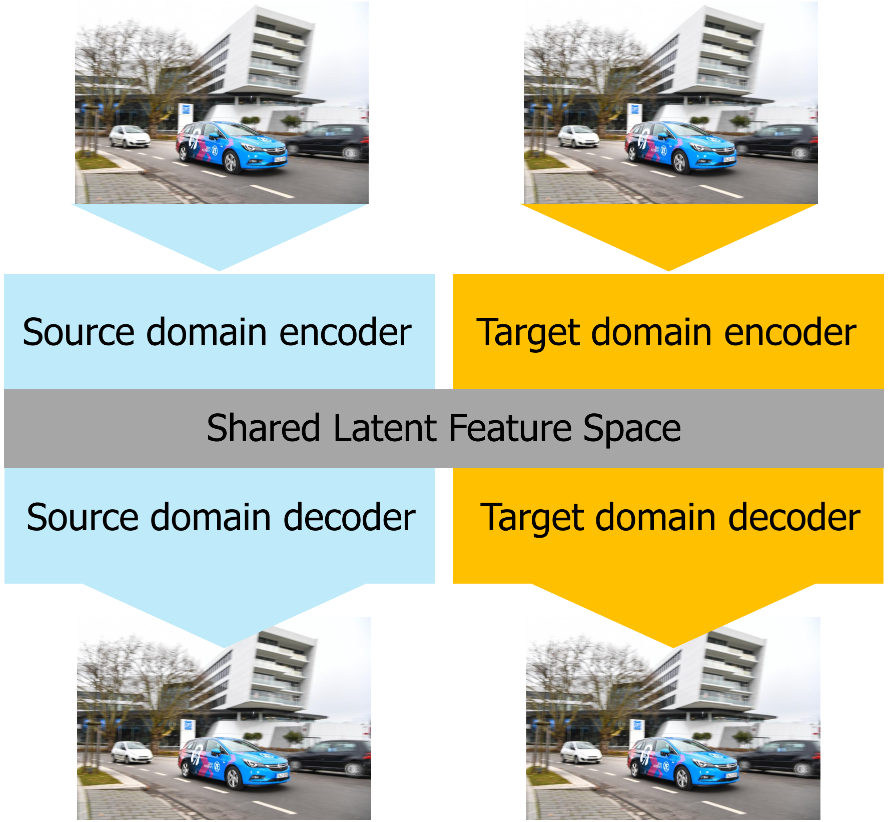
Transfer Learning with Neural Networks - Shared Latent Feature Space

So far the transfer was supervised, meaning we were in possession of labels to deploy in the target domain.

Next up are methods that do not require target domain ground truth – **unsupervised transfer learning** approaches.

Transfer Learning with Neural Networks - Shared Latent Feature Space

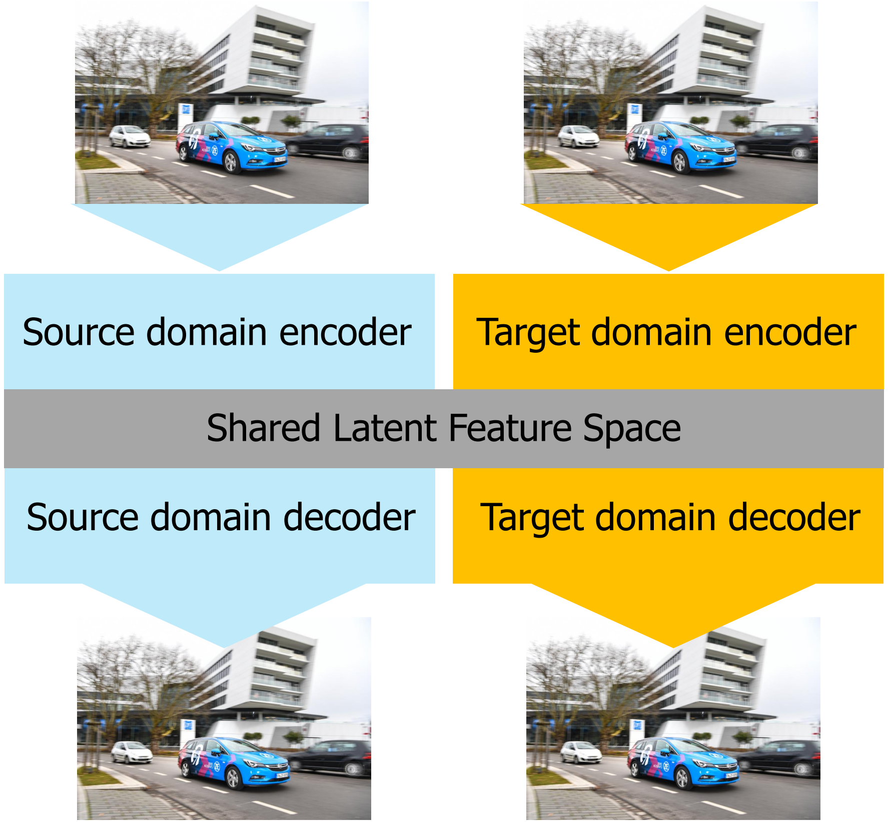
We can introduce a **shared layer** which models a shared latent feature space within an encoder decoder architecture.



Transfer Learning with Neural Networks - Shared Latent Feature Space

We can introduce a shared layer which models a shared latent feature space within an encoder decoder architecture.

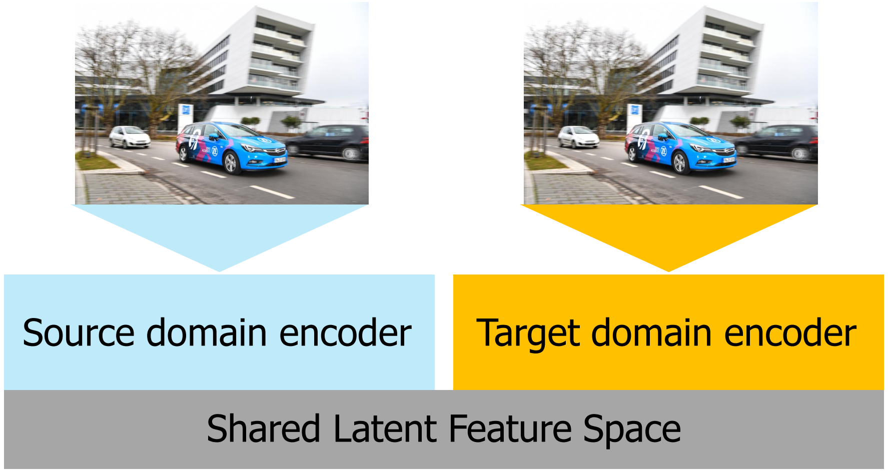
This layer **shares** the **parameters across** the two architectures



Transfer Learning with Neural Networks - Shared Latent Feature Space

We can introduce a shared layer which models a shared latent feature space within an encoder decoder architecture.

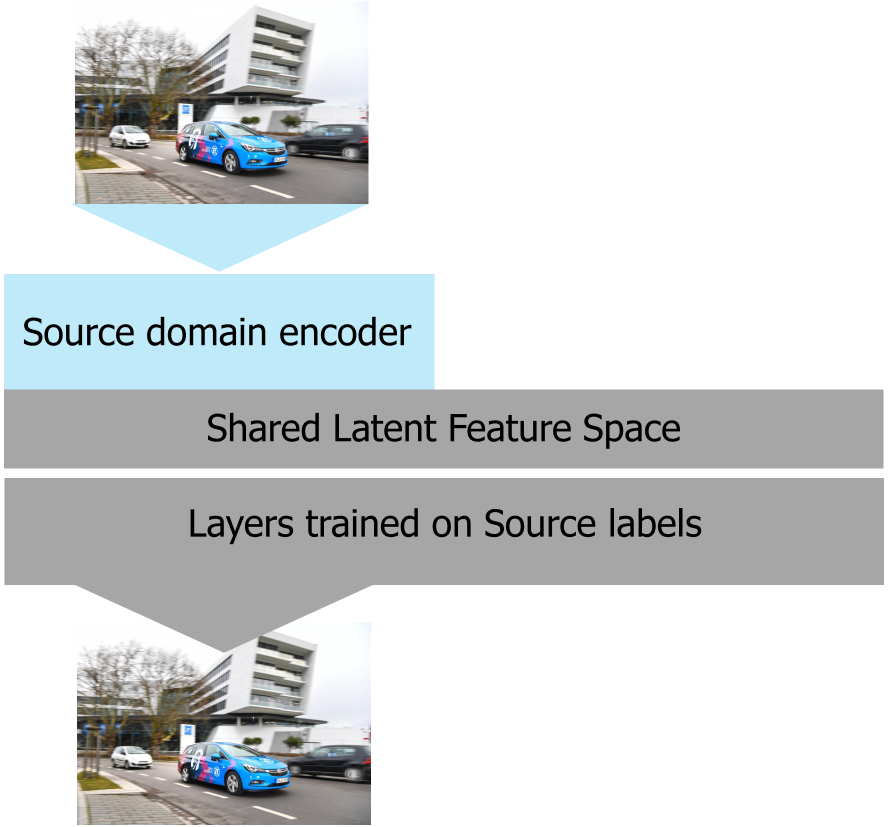
This layer **shares** the **parameters across** the two architectures and thus develops a shared **feature representation** for both domains.



Transfer Learning with Neural Networks - Shared Latent Feature Space

We can introduce a shared layer which models a shared latent feature space within an encoder decoder architecture.

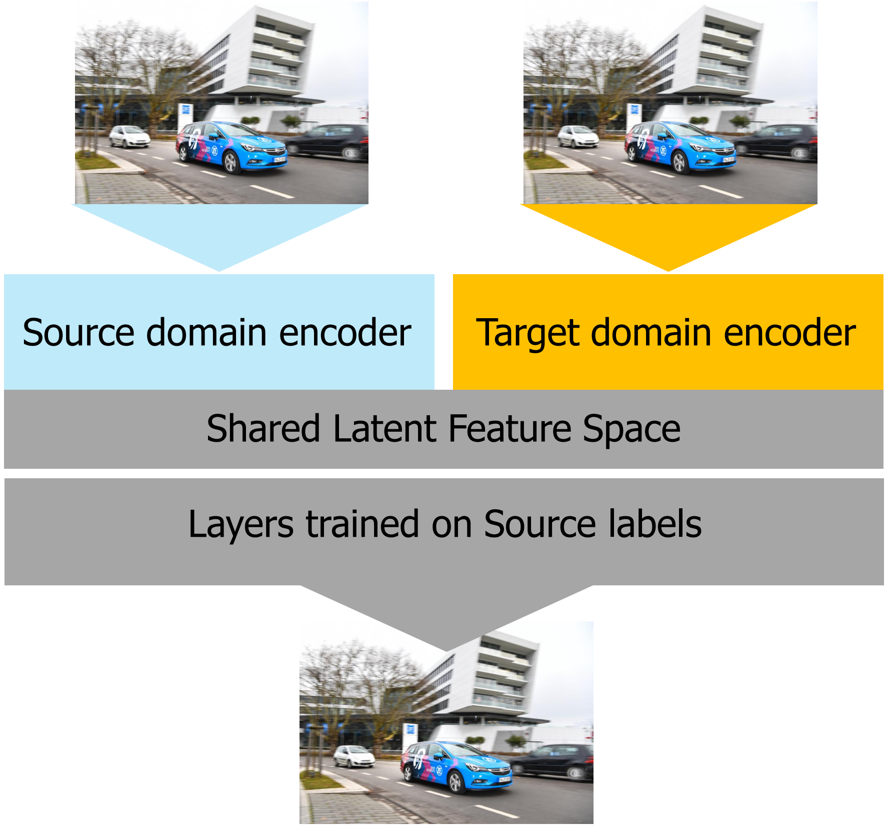
The shared **feature representation** is used as a base for training the task on source domain labels.



Transfer Learning with Neural Networks - Shared Latent Feature Space

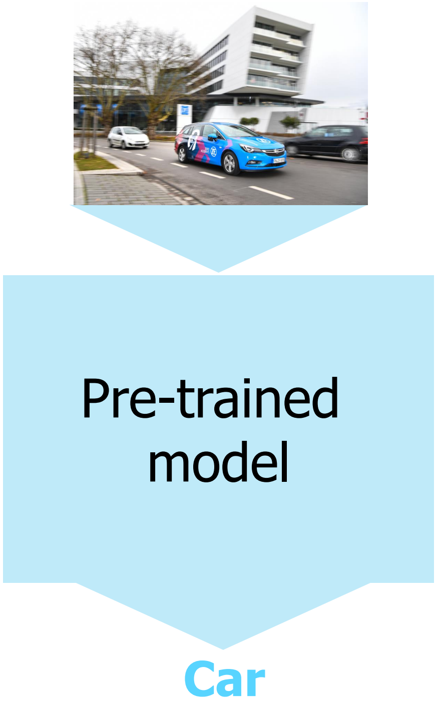
We can introduce a shared layer which models a shared latent feature space within an encoder decoder architecture.

During **inference** the prediction is done **independent** of the sample's **domain**.



Transfer Learning with Neural Networks - Self-supervised

In case the target domain is **overlapping** with the source domain or can be brought in a **continuous composition** the model can transfer the knowledge **self-supervised**.



In case the target domain is overlapping with the source domain or can be brought in a continuous composition the model can transfer the knowledge self-supervised.

This is realized by introducing **pseudo labels** on the target domain data, which are predicted by the pre-trained model.

$$y_{P,\, i} = \begin{cases} 1, & \text{if } i = \arg\max_i \; \theta(x_T). \\ 0, & \text{else.} \end{cases}$$

Pre-trained model

Pseudo Label

Transfer Learning with Neural Networks - Self-supervised

In case the target domain is overlapping with the source domain or can be brought in a continuous composition the model can transfer the knowledge self-supervised.

This is realized by introducing **pseudo labels** on the target domain data, which are predicted by the pre-trained model.

$$y_{P,\,i} = \begin{cases} 1, & \text{if } i = \arg\max_i \; \theta(x_T). \\ 0, & \text{else.} \end{cases}$$

Pre-trained model

Pseudo Label

Re-trained model

Car

Semi-Supervised Self-Training of Object Detection Models

Chuck Rosenberg
Google, Inc.
Mountain View, CA 94043
chuck@google.com

Martial Hebert
Carnegie Mellon University
Pittsburgh, PA 15213
hebert@ri.cmu.edu

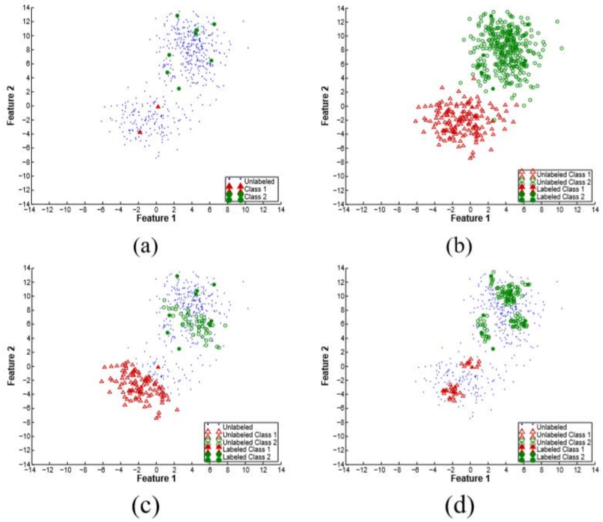
Henry Schneiderman
Carnegie Mellon University
Pittsburgh, PA 15213
hws@ri.cmu.edu

Semi-supervised self-training of object detection models

Train a model on the source domain (supervised).

Generate Pseudo-Labels on target domain.

Add samples to the training set for fine-tuning with respect to a prediction confidence threshold.



https://www.ri.cmu.edu/pub_files/pub4/rosenberg_charles_2005_1/rosenberg_charles_2005_1.pdf

Learning without Forgetting

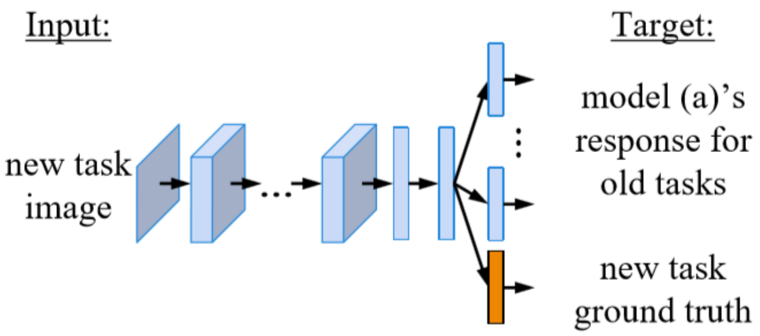
Train a model on a source task (supervised).

Generate pseudo-labels for the source task on the target domain.

Add nodes for target task and jointly train for both tasks.

Learning without Forgetting

Zhizhong Li, Derek Hoiem, *Member, IEEE*



https://arxiv.org/pdf/1606.09282.pdf

Unsupervised image-to-image translation network UNIT

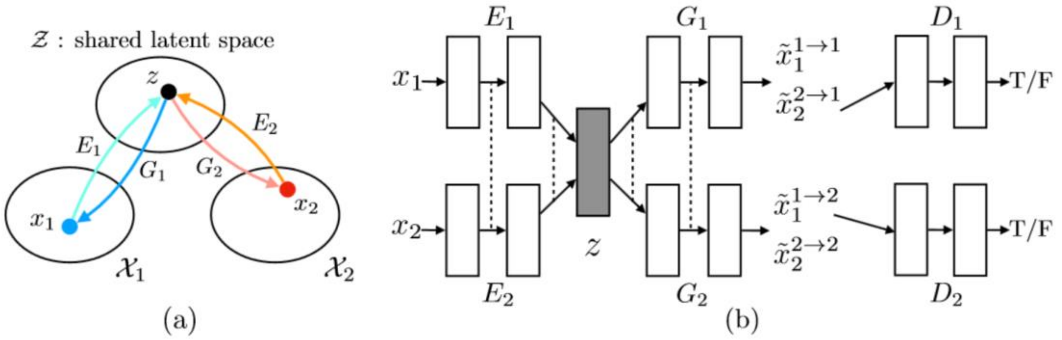
Map samples from different domains (source and target) into a shared feature space.

The mapping is learned by a joint encoding and decoding as well as an adversarial loss of two discriminators.

Later samples from one domain can be translated into the other domain and respectively.

Unsupervised Image-to-Image Translation Networks

Ming-Yu Liu, Thomas Breuel, Jan Kautz
NVIDIA
{mingyul,tbreuel,jkautz}@nvidia.com



https://arxiv.org/pdf/1703.00848.pdf

Model-specific

- Task-specific performance metrics are considered for source domain, target domain, and intermediate domains
- Catastrophic forgetting depicting the performance decrease on the source domain.
- Domain gap depicting the performance difference between source and target domain.

Training-specific

Model-specific

Training-specific

- Model size efficiency as in memory size increase of the model
- Sample size efficiency as in memory size when samples need to be stored for retraining and replay strategies
- Computational efficiency as in operations needed to adapt a model to the next domain

References

[10] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1*, volume 1, pages 29–36. IEEE, 2005.

[11] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, Dec 2018.

[12] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 700–708. Curran Associates, Inc., 2017.

[13] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

[14] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

Transfer Learning - Take it from here

Codebase

Journal Paper

Tutorial

jupyter TL-TUT Last Checkpoint: 08/13/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted | tensorflow

Run Markdown

Applied Deep Learning Tutorial

Transfer Learning for Object Classification

Introduction

In this tutorial, you will attempt to benefit from a model that has been pretrained for the same task but on a different dataset. You will deploy the first layers and their feature extraction capabilities of a converged network. This process is known as transfer learning.

Fig. 1: Cat and dog in an image

Core idea

A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task, such as ImageNet, and COCO. We can either use the pretrained model as it is for inference on the task it has been trained on or we can do transfer learning using the pretrained convents for further training on a new dataset with a possibly new output space.

The intuition behind transfer learning is that if this model trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world and the semantic features present in the visual world and shared between all visual tasks. We can leverage these learned feature maps without having to train a large model on a large dataset by using these models as the basis of our own model specific to our task. There are 2 scenarios of transfer learning using a pretrained model:

- Fine Tuning or Retraining: Unfreezing a few of the top layers of a frozen model base used for feature extraction, and jointly training both the newly added classifier layers as well as the last layers of the frozen model. This allows us to "fine tune" the higher order feature representations in addition to our final classifier in order to make them more relevant for the specific task involved.
- Feature Extraction: Use the representations learned by a previous model to extract meaningful features from new samples. We simply add a new output layer, which will be trained from scratch, on top of the pretrained model so that we can repurpose the feature maps learned previously for our dataset and our new output space.

Thanks for your time
Questions?

Contact! [Subject]

mark.schutera@kit.edu [ADL]