UNIVERSITAT JAUME I

MACHINE LEARNING

University Master's Degree in Intelligent Systems

deep transfer learning

Ramón A. Mollineda Cárdenas

a quote

After supervised learning,
transfer learning will be the next driver of
ML commercial success.

Andrew Ng
data scientist

driver = clave, motor, fuerza impulsora...

a sentence in german

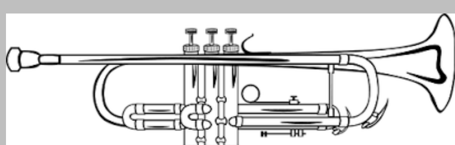
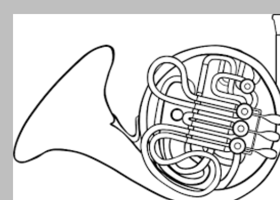
please, refrain german speakers!

Mein Gott ist gut

any hypotheses about its meaning?

natural knowledge transfer

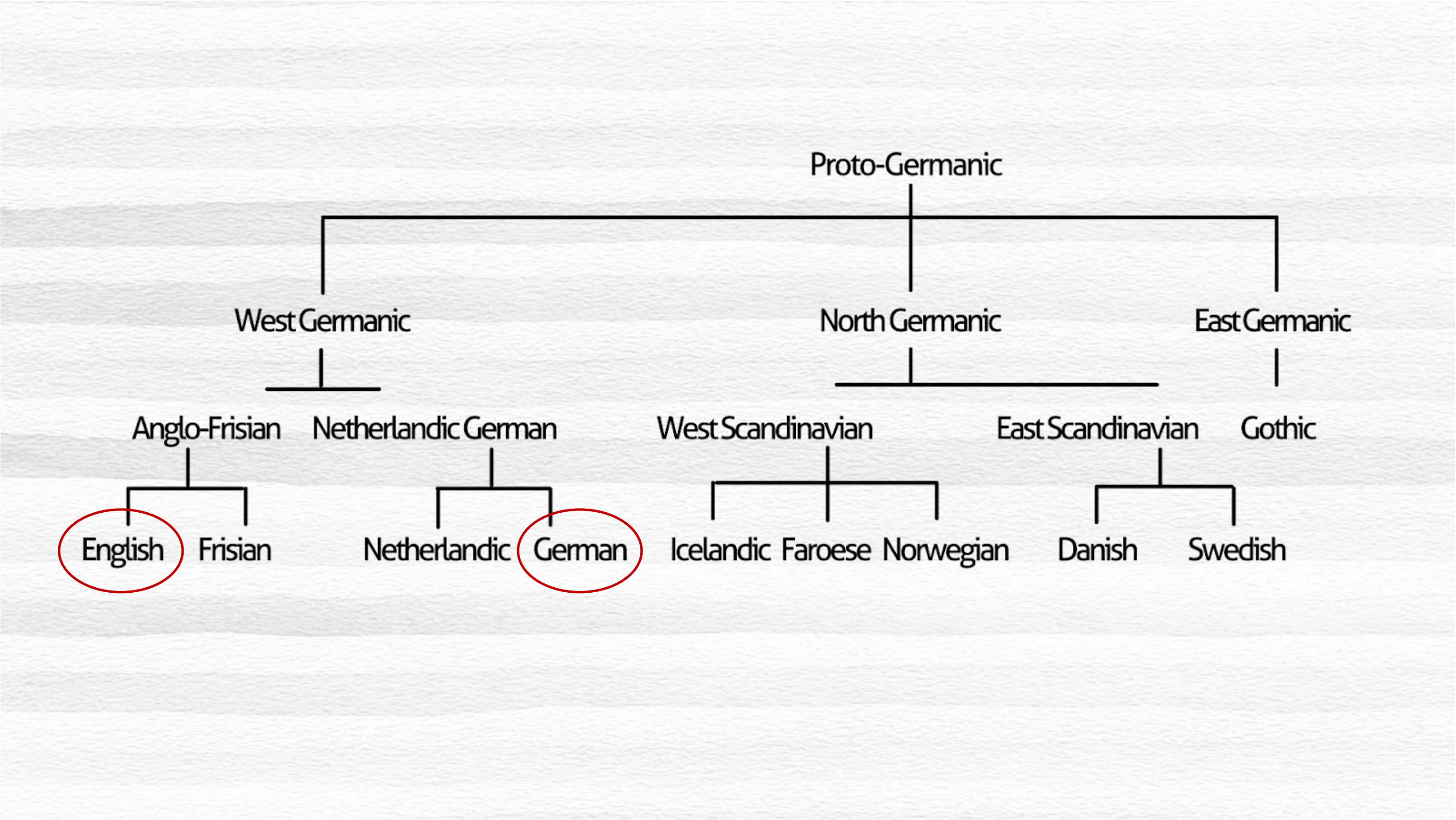
humans naturally take advantage of previously acquired knowledge and skills, to learn how to solve new related tasks.

if you know…	it'll be easier for you to learn…	relationship
I have… It is long… Where is my book? My name is Juan.	Ich habe… Es ist lang… Wo ist mein buch? Mein name ist Juan.	english and german are germanic languages
		trumpet and French horn are brass instruments

transfer learning takes advantage of the knowledge acquired in the solution of a task to solve other similar tasks; this way you avoid learning from scratch.

natural knowledge transfer

english & german



source: *https://www.europelanguagejobs.com/blog/german-english-cognates*

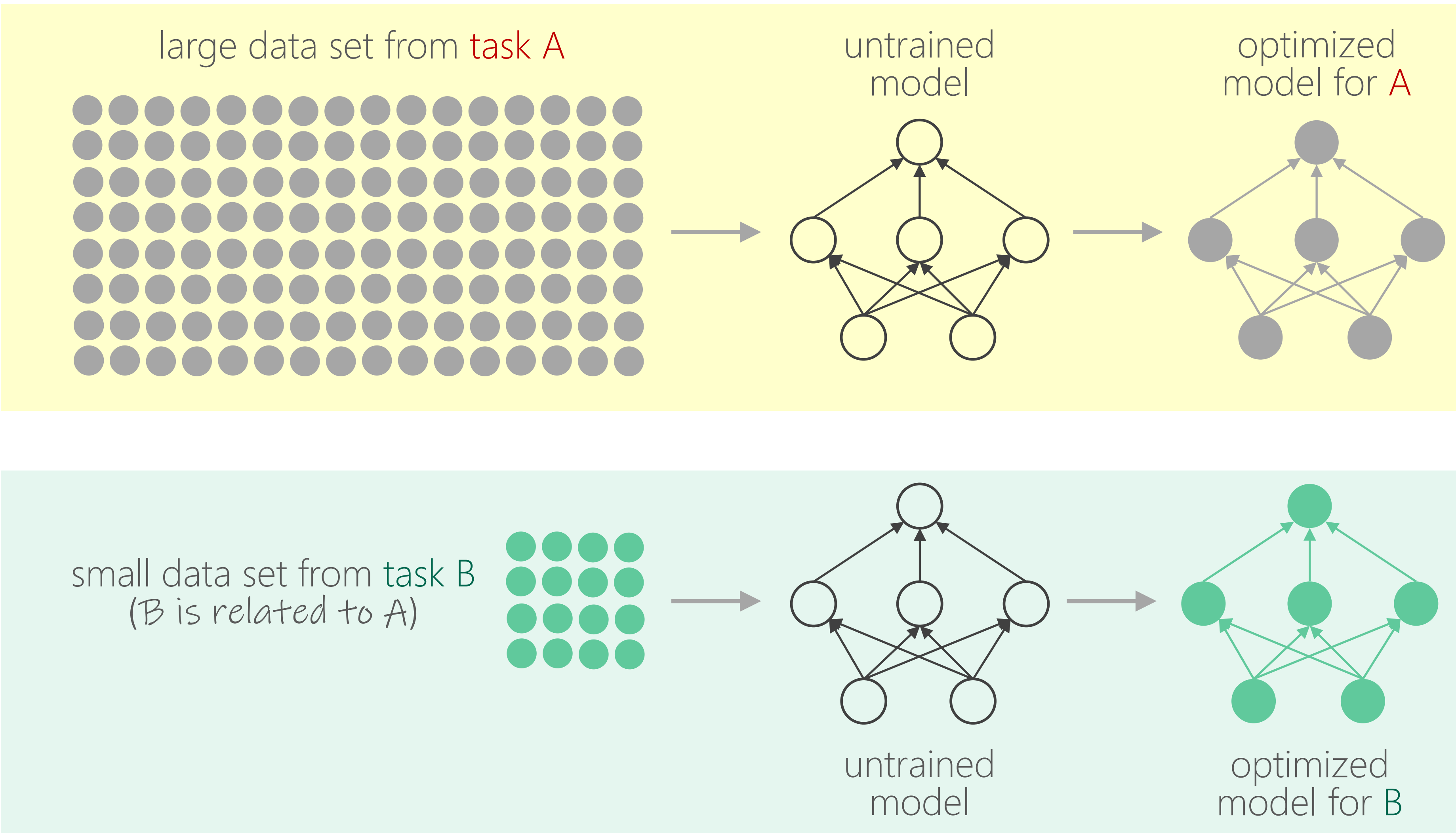
context

traditional machine learning learn from scratch:

starts with arbitrary parameter values and uses data from a target task to adjust them

context

tradicional machine learning



model of B does NOT take advantage of learning from A

context

deep learning

deep learning inherits this practice…

promise – deep networks are universal approximators, capable of solving <u>any</u> problem

fine print – they depend on millions of parameters to optimize!

myth – therefore, you will not be able to create a DL solution unless you have sufficient amounts of data from the target task

context

deep learning

deep learning inherits this practice...

promise – deep networks are universal approximators, capable of solving <u>any</u> problem

fine print – they depend on millions of parameters to optimize!

myth – therefore, you will not be able to create a DL solution unless you have sufficient amounts of data from the target task

context

deep learning

deep learning inherits this practice...

promise – deep networks are universal approximators, capable of solving <u>any</u> problem

fine print – they depend on millions of parameters to optimize!

truth – it's possible to...

- transfer learned representations to related tasks
- learn good representations from unlabeled data
- learn representations common to different domains

context

deep learning

traditional scenario

learn from scratch with <u>insufficient</u> data => *overfitting*

(cheap solution, but generally useless)

context

deep learning

fuente: flickr

UNICORN scenario

learn from scratch with unlimited resources...

- unlimited labeled data
- unlimited computing power
- unlimited time

(optimal solution at a prohibitive cost)

motivation

aim

good generalization at a low cost with limited amounts of data

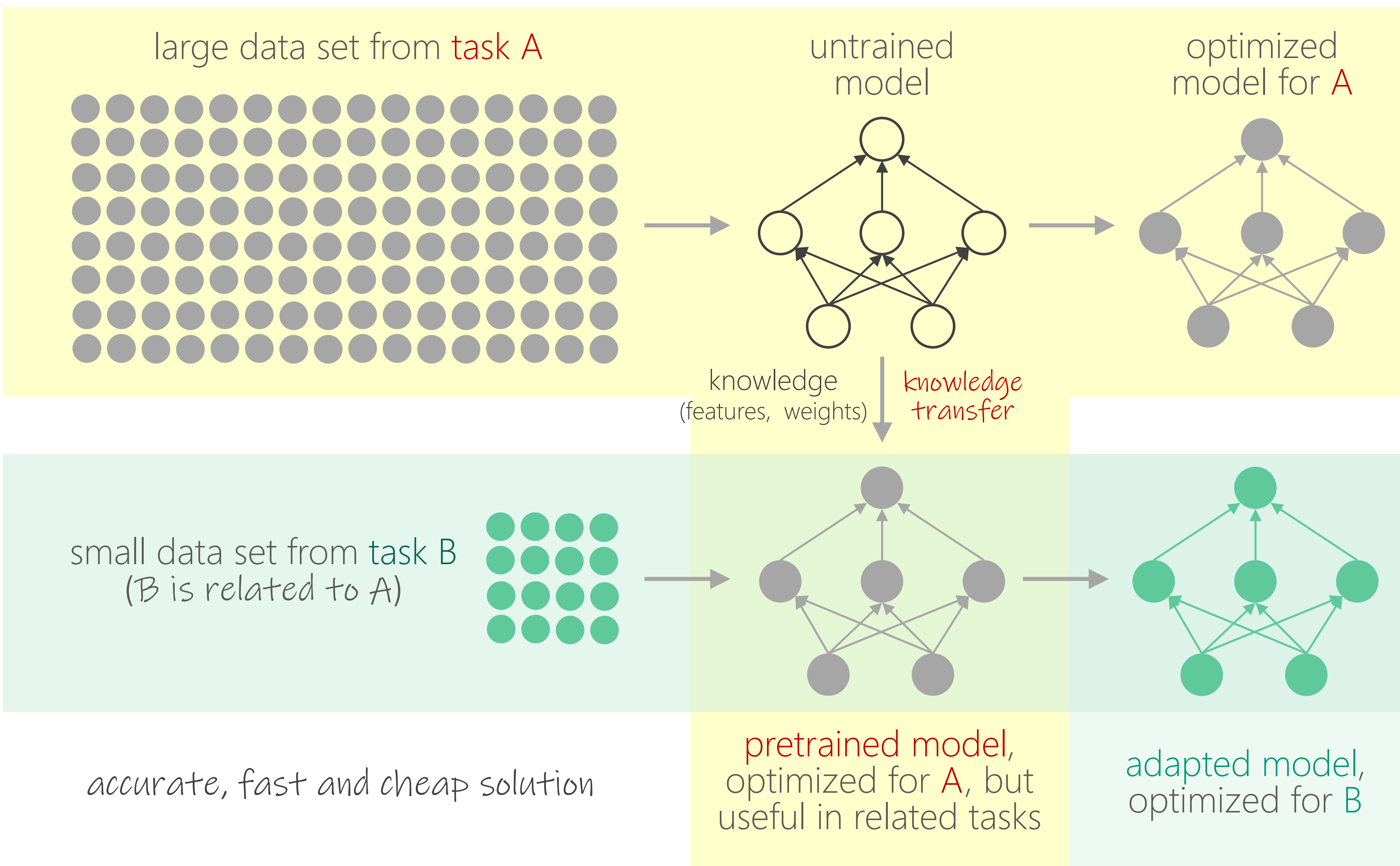
(good and affordable solution)

definition

Transfer learning … refers to the situation where what has been learned in one setting … is exploited to improve generalization in another setting.

I. Goodfellow, Y. Bengio & A. Courville. Deep Learning, 2016.

transfer learning – popular strategy



another quote...

Deep Learning on Steroids

with the Power of Knowledge Transfer!

Dipanjan Sarkar

author of the book "Hands-On Transfer Learning with Python" (link)

formal definition

domain and task

domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$

$\mathcal{X}$, feature space

$P(X)$, marginal probability distribution

$X = \{x_1, \ldots, x_n\}, x_i \in \mathcal{X}$

example: document classification (bag-of-words)

- $\mathcal{X}$, space for all possible documents
- X, training document set, $X \subset \mathcal{X}$
- x_i, document representation i

task $\mathcal{T} = \{\mathcal{Y}, P(y|x)\}$

$\mathcal{Y}$, label space

$P(y|x)$, conditional probability distribution (prediction)

$P(y|x)$ learns from $\{(x_i, y_i)\},\ x_i \in X,\ y_i \in \mathcal{Y}$

example: document classification (bag-of-words)

- in sentiment analysis, $\mathcal{Y} = \{\text{Positive, Negative, Neutral}\}$

Source: Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359 (link).

formal definition

domain and task

given

- $(\mathcal{D}_s, \mathcal{T}_s)$, source domain and task
- $(\mathcal{D}_t, \mathcal{T}_t)$, target domain and task

transfer learning aims at…

- learning $P_t(y|x)$ using knowledge from $\mathcal{D}_s$ and/or $\mathcal{T}_s$
- cases: $\mathcal{D}_s \neq \mathcal{D}_t$ xor $\mathcal{T}_s \neq \mathcal{T}_t$
- generally, the number of labeled samples from $\mathcal{T}_t$ is significantly less than the number of labeled samples from $\mathcal{T}_s$

Source: Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359 (link).

formal definition

scenarios

scenarios

- domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$
  - $\mathcal{X}_s \neq \mathcal{X}_t$
    e.g., documents written in different languages
  - $P(X_s) \neq P(X_t)$
    e.g., documents discuss different topics (politics, science…); it is known as *domain adaptation*.
- task $\mathcal{T} = \{\mathcal{Y}, P(y|x)\}$
  - $\mathcal{Y}_s \neq \mathcal{Y}_t$
    e.g., documents are assigned to different classes/labels; it usually occurs together with the following scenario.
  - $P_s(y|x) \neq P_t(y|x)$
    e.g., different distributions of documents among classes

Source: Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359 (link).

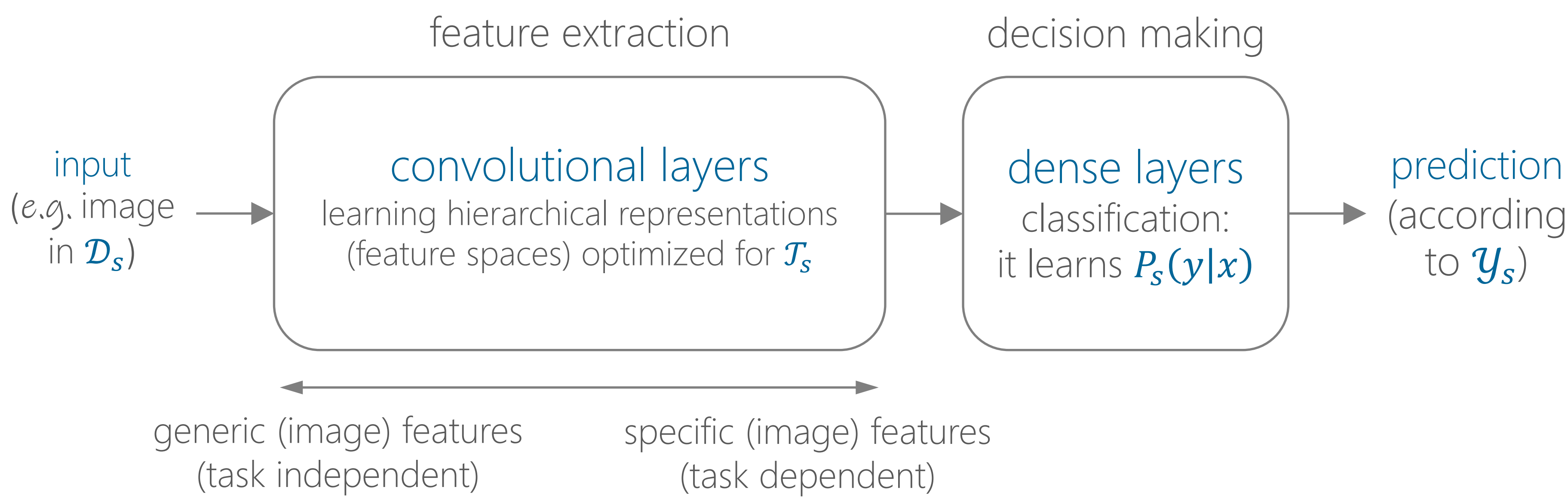
TL methods in deep learning

- transfer of pretrained features, when $\mathcal{Y}_s \neq \mathcal{Y}_t$
  - simple and popular method
  - it takes advantage of the convolutional base trained in a related task $\mathcal{T}_s$
- domain adaptation, when $P(X_s) \neq P(X_t)$
  - it learns common representation to $\mathcal{D}_s$ y $\mathcal{D}_t$
  - it minimizes classification error on $\mathcal{T}_s$
  - it maximizes confusion between $\mathcal{D}_s$ y $\mathcal{D}_t$ (on common representation)

pre-trained CNN features

CNN architecture

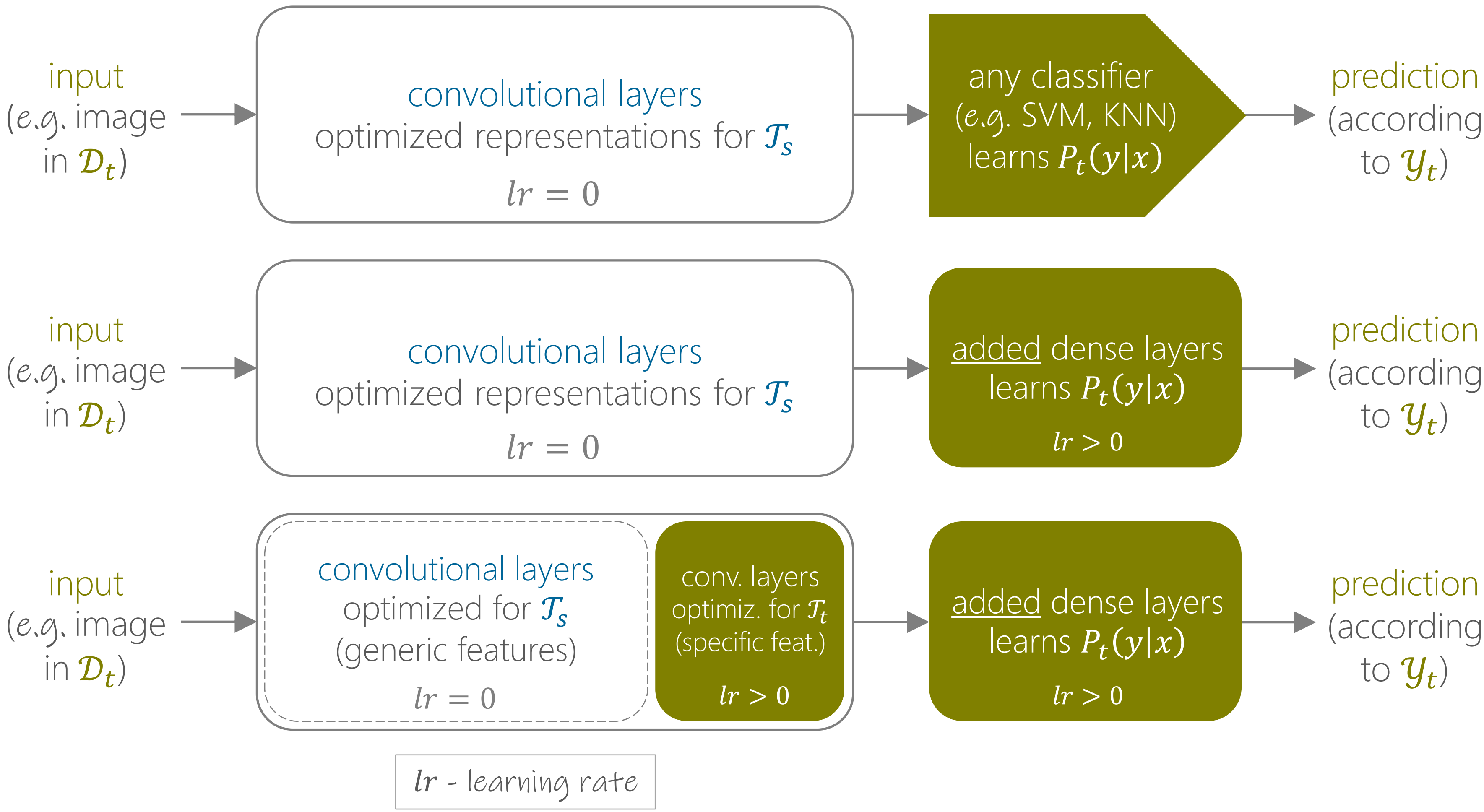
learning processes from $(\mathcal{D}_s, \mathcal{T}_s)$ => pre-trained models



pre-trained CNN features

strategies

exploitation on $(\mathcal{D}_t, \mathcal{T}_t)$ / scenario: $\mathcal{D}_s \approx \mathcal{D}_t, \mathcal{Y}_s \neq \mathcal{Y}_t$



pre-trained CNN features

typical workflow

- There is a base model previously optimized for a source task.
- Choose a subsequence of convolutional layers from the start (conv. base).
- Freeze weights of previous layers to avoid destroying learned knowledge.
- Add new layers (new subnet) connected to the output of the conv. base.
- Train new layers to adapt old features in useful predictions for a target task.
- Optionally perform a fine tuning of the entire model:
  - unfreeze convolutional base layers (usually all)
  - retrain the entire model with data from the new target task with a very small lr
  - …for a small number of epochs: danger of overfitting!

pre-trained CNN features

typical workflow in Keras

```
# instantiate a base model with pre-trained weights
base_model = keras.applications.Xception(
    weights='imagenet',  # load weights pre-trained on ImageNet.
    input_shape=(150, 150, 3),
    include_top=False)  # do not include the ImageNet classifier at the top.

# freeze the base model
base_model.trainable = False

# create a new model on top
inputs = keras.Input(shape=(150, 150, 3))
# make sure that base_model is running in inference mode by passing `training=False`
x = base_model(inputs, training=False)
# convert features to vectors
x = keras.layers.GlobalAveragePooling2D()(x)
# a Dense classifier with a single unit (binary classification)
outputs = keras.layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

# train the model on new data
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=[keras.metrics.BinaryAccuracy()])
model.fit(new_dataset, epochs=20, callbacks=..., validation_data=...)
```

Transfer learning & fine-tuning, fcollet, 2020.

pre-trained CNN features

typical workflow in Keras

```
# optional fine-tuning of the whole base model

# unfreeze the base model
base_model.trainable = True

# it's important to recompile your model after you make any changes to the `trainable` attribute of any inner
layer, so that your changes are take into account
model.compile(optimizer=keras.optimizers.Adam(1e-5),  # very low learning rate
            loss=keras.losses.BinaryCrossentropy(from_logits=True),
            metrics=[keras.metrics.BinaryAccuracy()])

# train end-to-end; be careful to stop before you overfit!
model.fit(new_dataset, epochs=10, callbacks=..., validation_data=...)
```

Transfer learning & fine-tuning, fcollet, 2020.

pre-trained CNN features

examples of reusable models

pretrained models for computer vision tasks

- VGG16 (+)
- VGG19 (+)
- Inception V3 (+)
- Xception (+)
- ResNet-50 (+)
- EfficientNet (+)

pretrained models for NLP tasks (sentiment analysis, name entity recognition, machine translation, text summarization, natural language generation, speech recognition, QA systems, etc.)

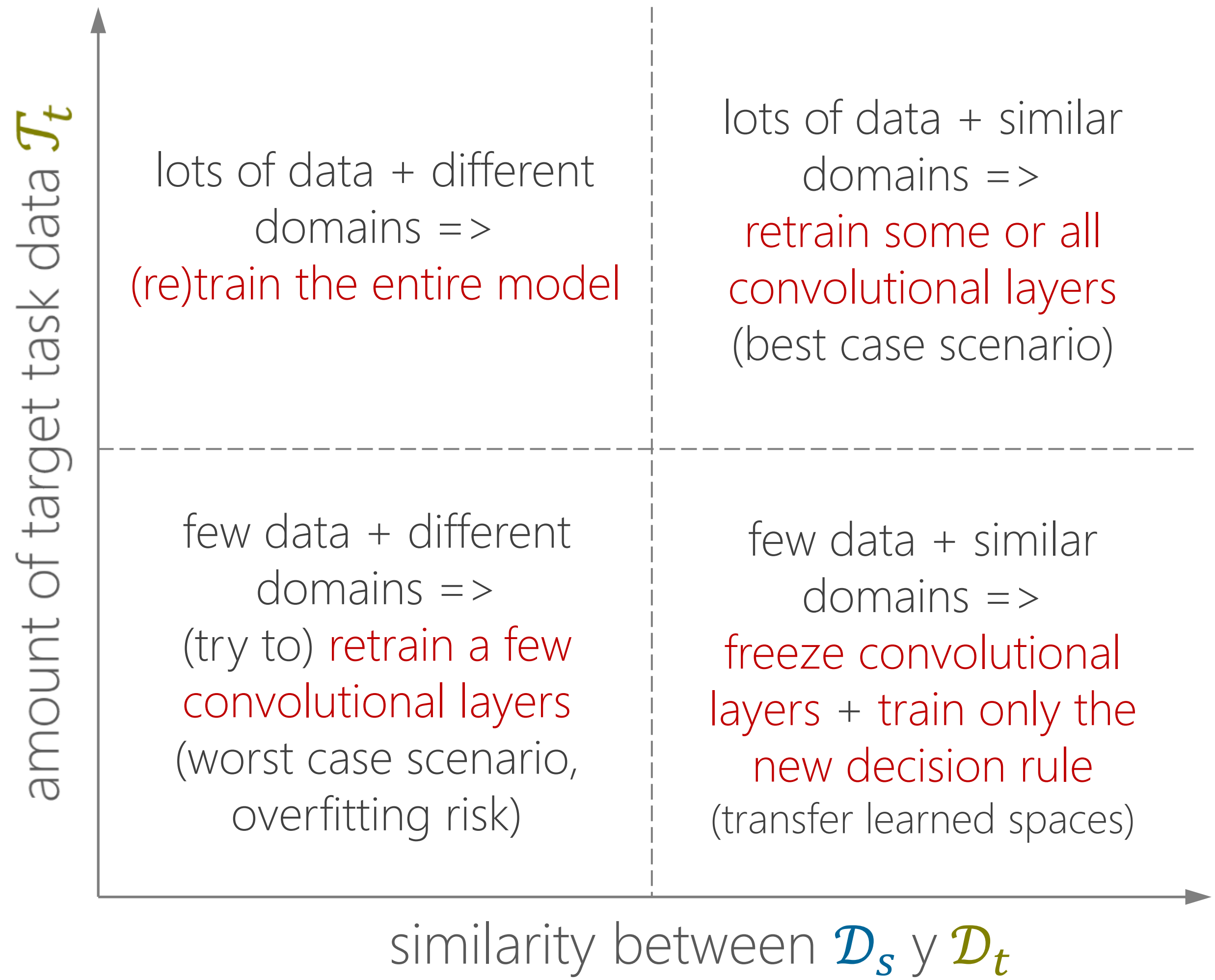
- BERT by Google (+)
- RoBERTa by Facebook (+)
- GPT-3 by OpenAI (+)
- Huggingface transformers (+)

Pre-trained models can be easily loaded from libraries such as PyTorch, Tensorflow, etc.

Dipanjan (DJ) Sarkar, A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning, Medium, 2018 (link).

pre-trained CNN features

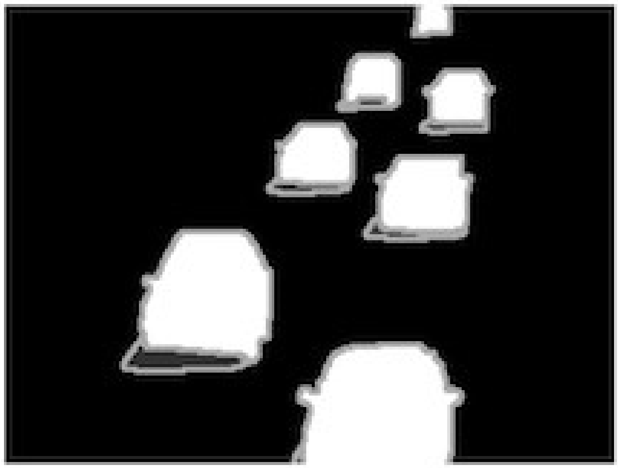
strategies



pre-trained CNN features

case study

target task ($\mathcal{T}_t$): car segmentation on a highway



objective: to create a CNN model capable of solving the segmentation task

restriction: we only have 25 training samples (image pairs)!

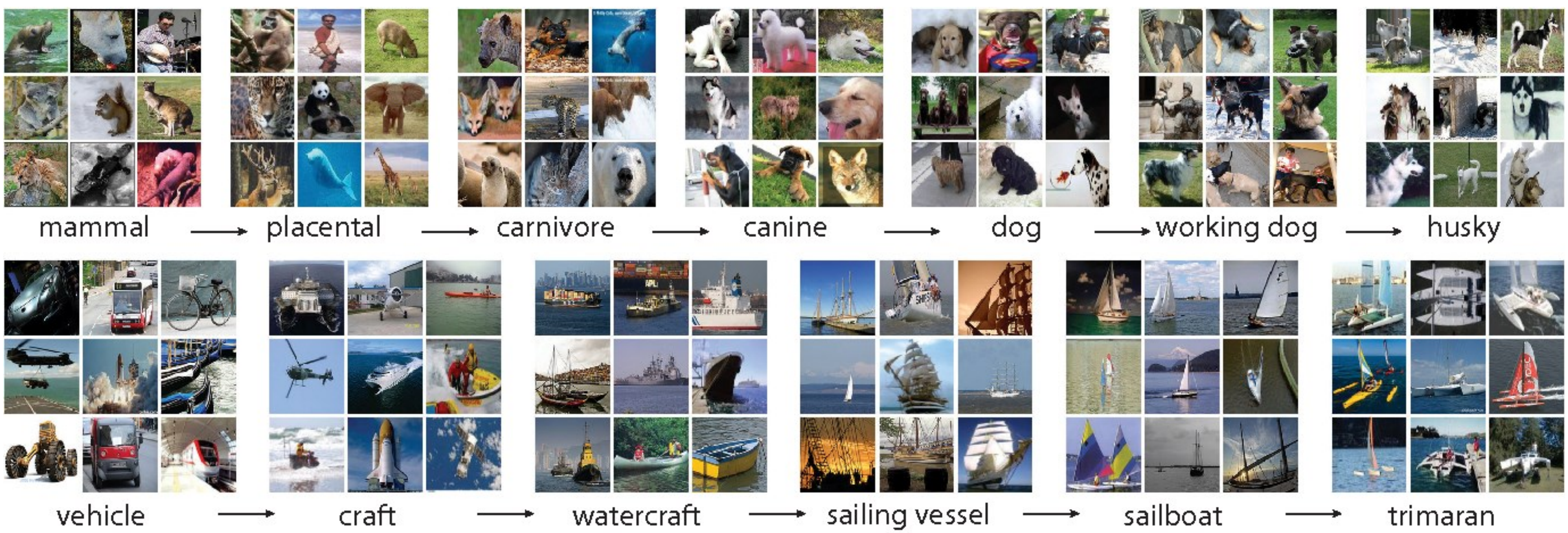
Long Ang, A comprehensive guide on how to fine-tune deep neural networks using Keras on Google Colab (Free GPU),
https://towardsdatascience.com/a-comprehensive-guide-on-how-to-fine-tune-deep-neural-networks-using-keras-on-google-colab-free-daaaa0aced8f

pre-trained CNN features

case study

source task ($\mathcal{T}_s$): image classification (ImageNet)

IMAGENET **



** *Deng, Jia et al. "ImageNet: A large-scale hierarchical image database." 2009 IEEE Conf. on Comp. Vision and Pattern Recognition (2009): 248-255.*

pre-trained CNN features

case study

source task ($\mathcal{T}_s$): image classification (ImageNet)

training:

- 1.200.000 labeled images
- 1.000 class/labels (*ground truth*)
- one label per image: identifies the main object

validation and test:

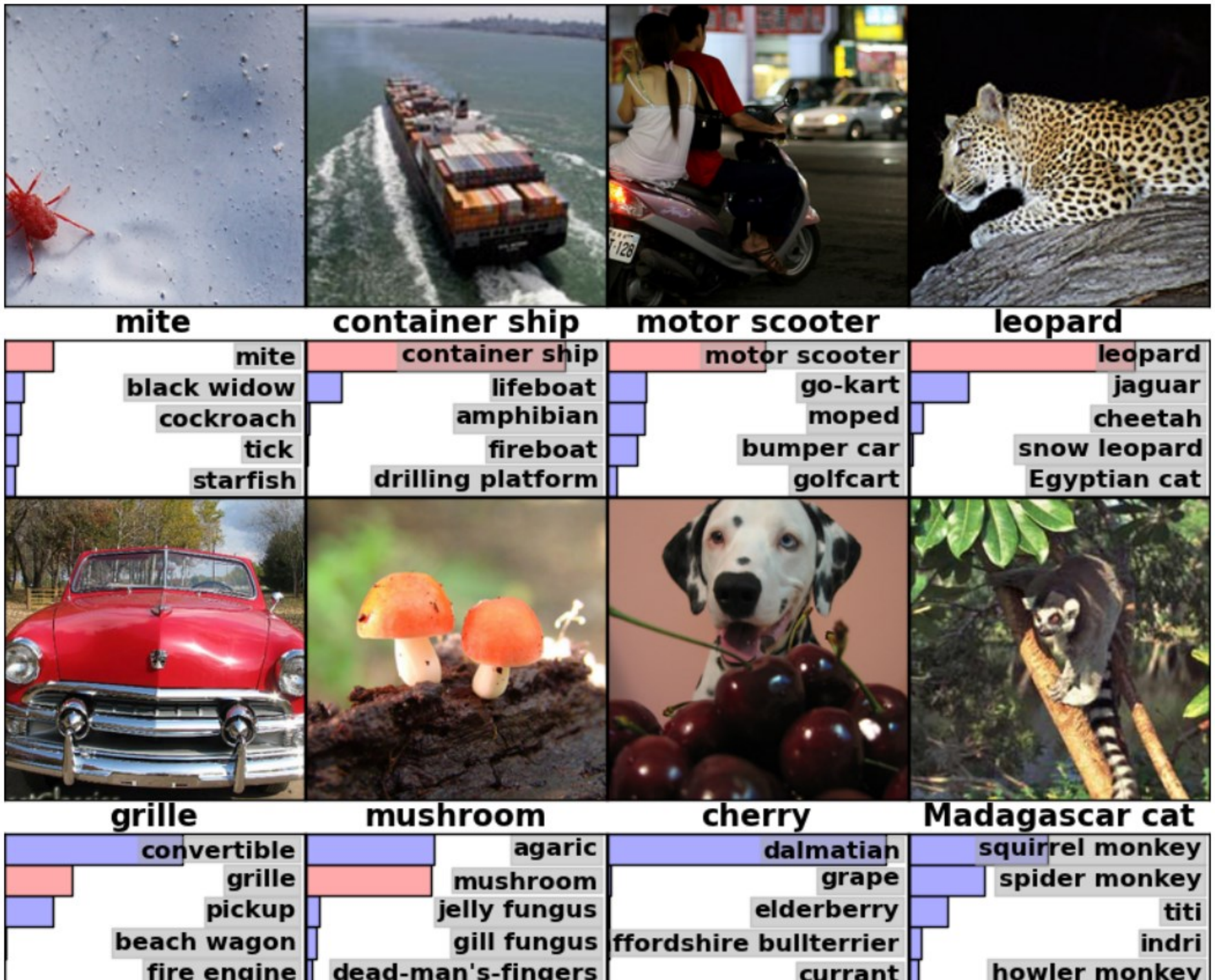
- 150.000 real world photos (obtained through search engines)
- 50.000 for validation purposes
- 100.000 for testing purposes
- typical output: the 5 most likely categories/classes

success/hit: the ground truth label is one of these 5 categories (*top-5 error*)

pre-trained CNN features

case study

examples of classification with AlexNet network (link)

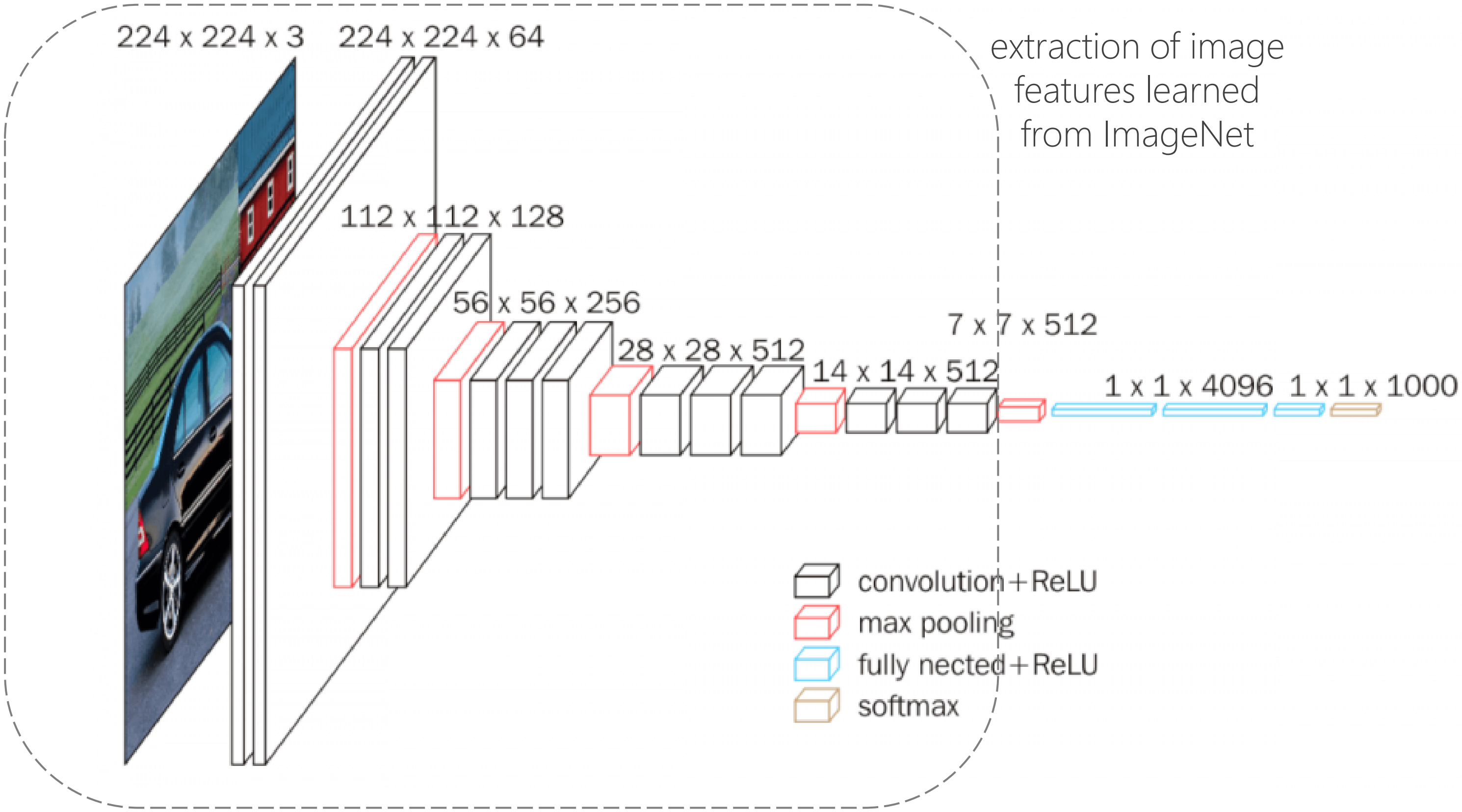


Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

pre-trained CNN features

case study

VGG16 **: source task solution ($\mathcal{T}_s$, ImageNet)

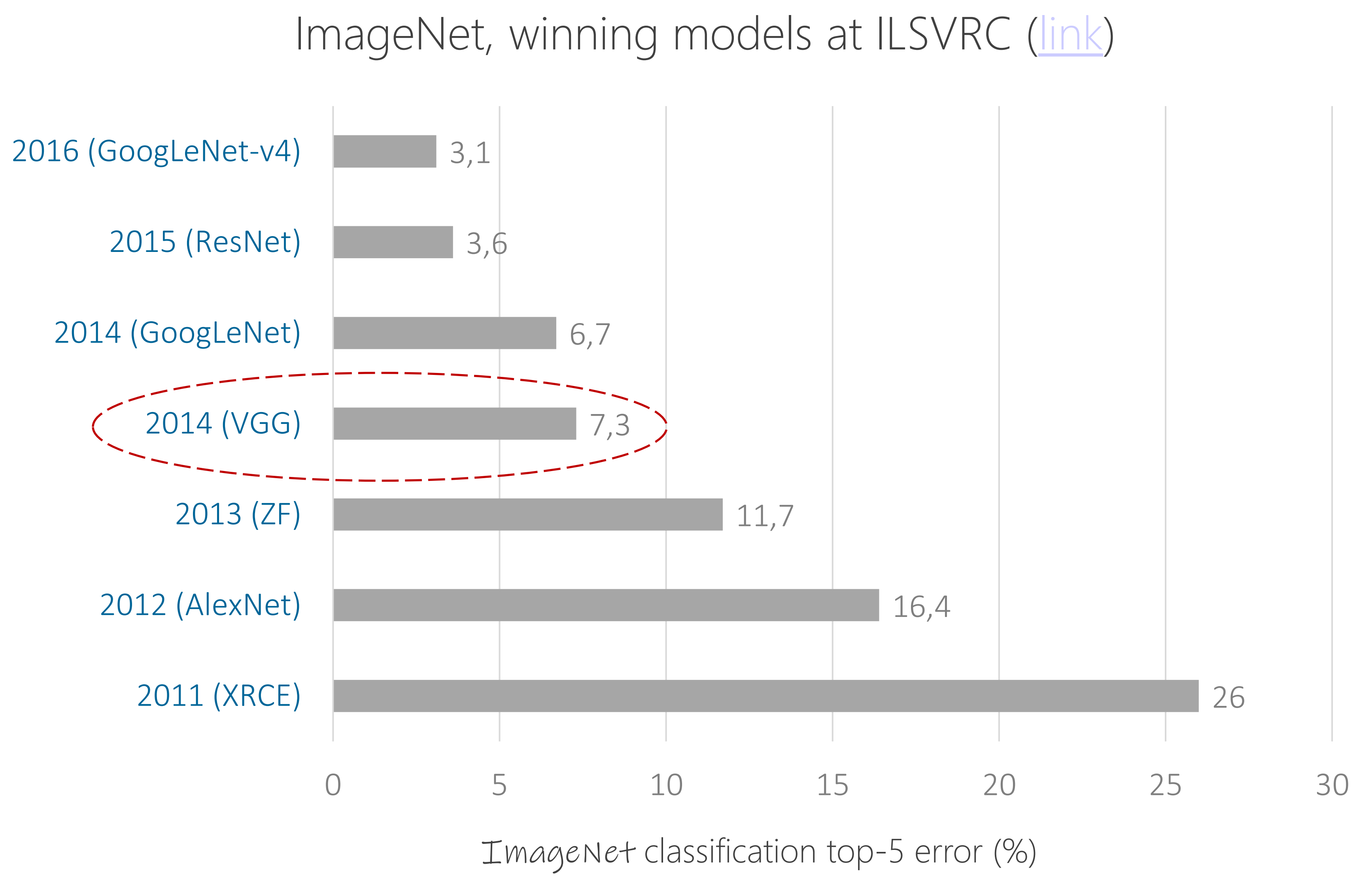


extraction of image features learned from ImageNet

** Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". ArXiv:1409.1556 (2014).

pre-trained CNN features

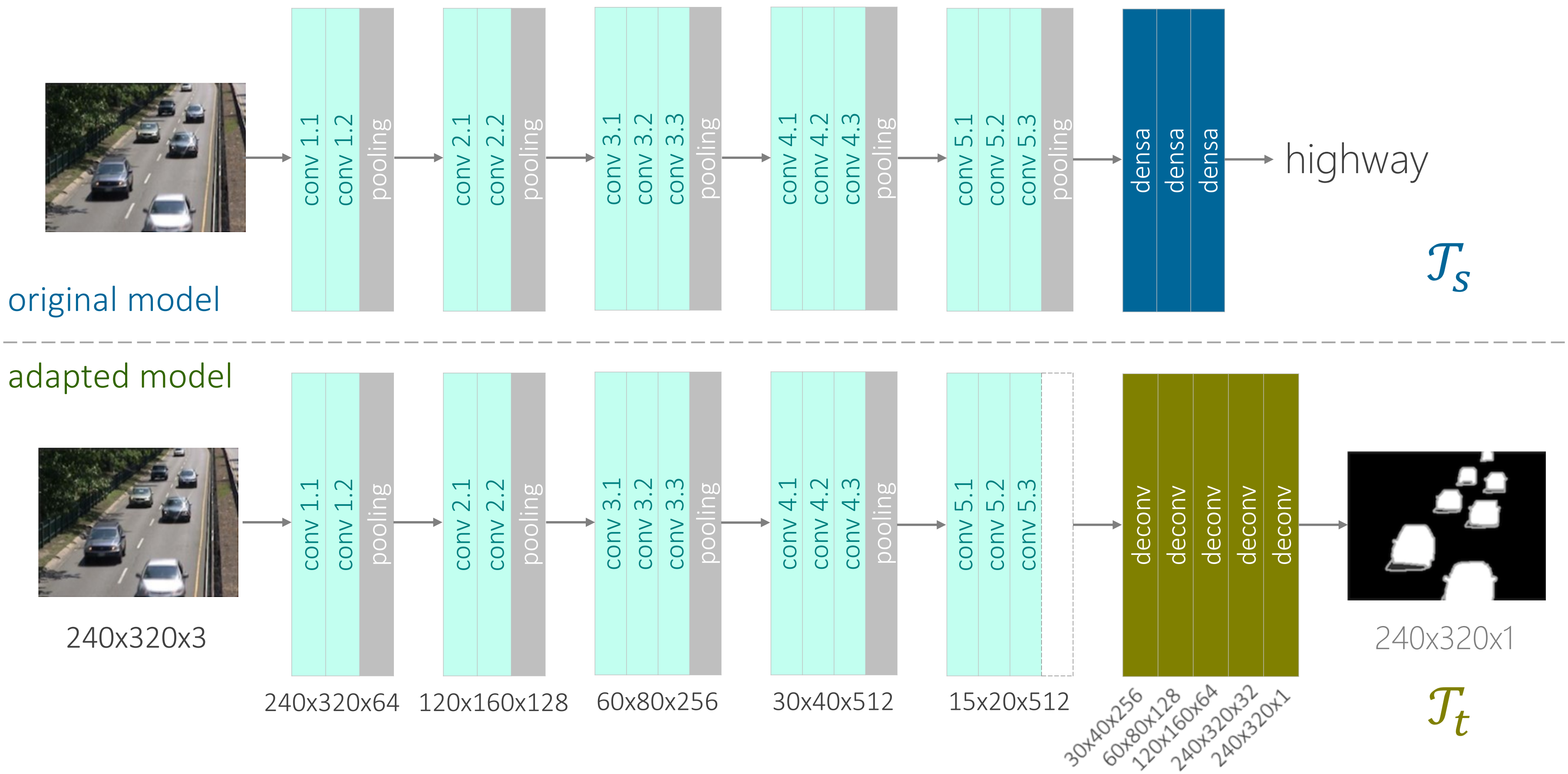
case study



pre-trained CNN features

case study

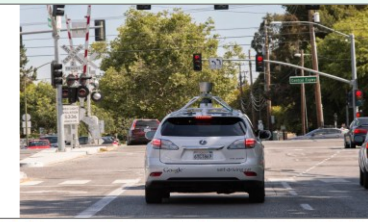
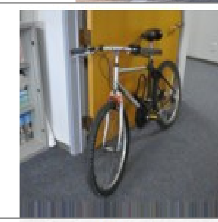
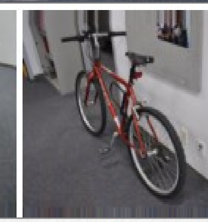
architecture adaptation of VGG16 (convolutional base is transferred)



domain adaptation

introduction

$$\mathcal{D}_s \neq \mathcal{D}_t, \mathcal{Y}_s = \mathcal{Y}_t$$

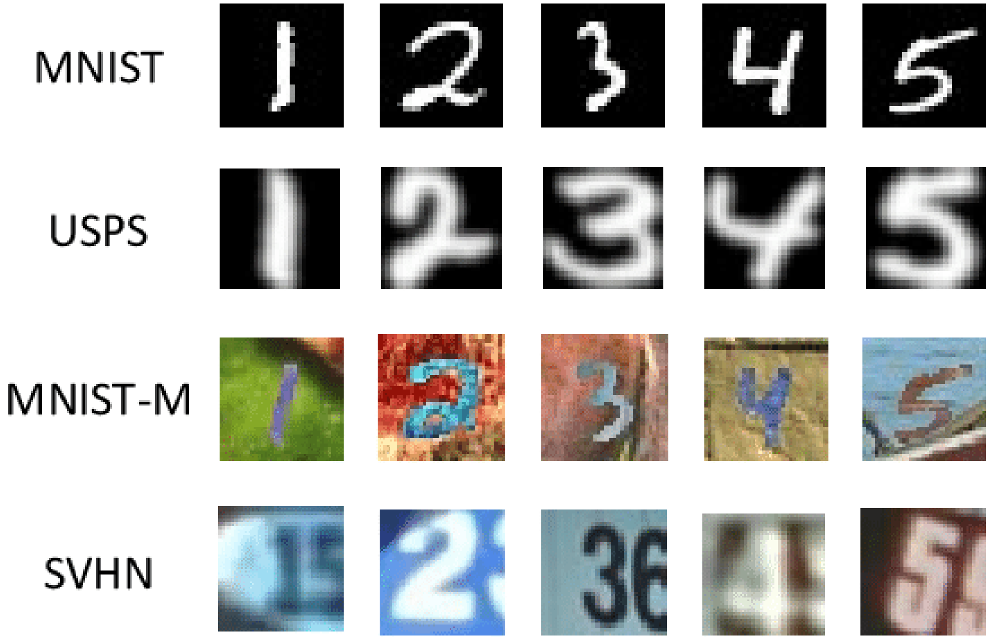
source domain ($\mathcal{D}_s$)	target domain ($\mathcal{D}_t$)
self-driving car simulator source: TechCrunch 	Google self-driving car source: Google Research blog 
contextless objects source: Sun et al., 2016  	objects in the wild source: Sun et al., 2016  
NLP, news docs source: pxhere.com 	social media msg source: flickr.com 
ASR systems trained with standard accents 	ASR systems used by people with non-standard accents, speech difficulties, dyslexics, etc.

NLP = Natural Language Processing
ASR = Automatic Speech Recognition

Source: Sebastian Ruder (2017). Transfer Learning - Machine Learning's Next Frontier, post (link).

domain adaptation

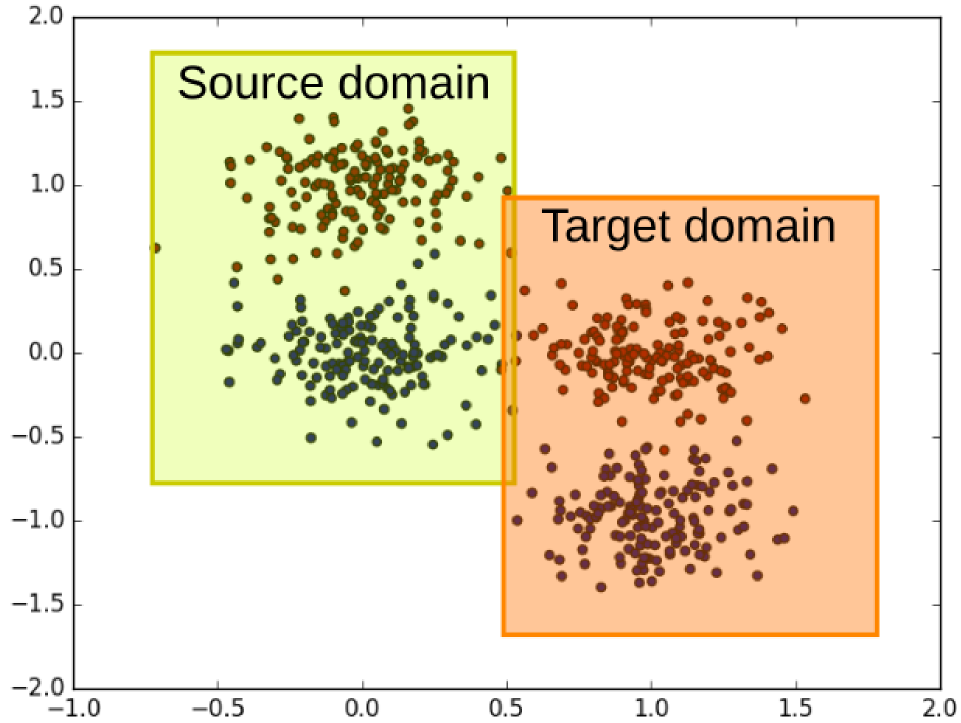
introduction



$$\mathcal{D}_s \neq \mathcal{D}_t, \mathcal{Y}_s = \mathcal{Y}_t$$

domain adaptation

domain shift



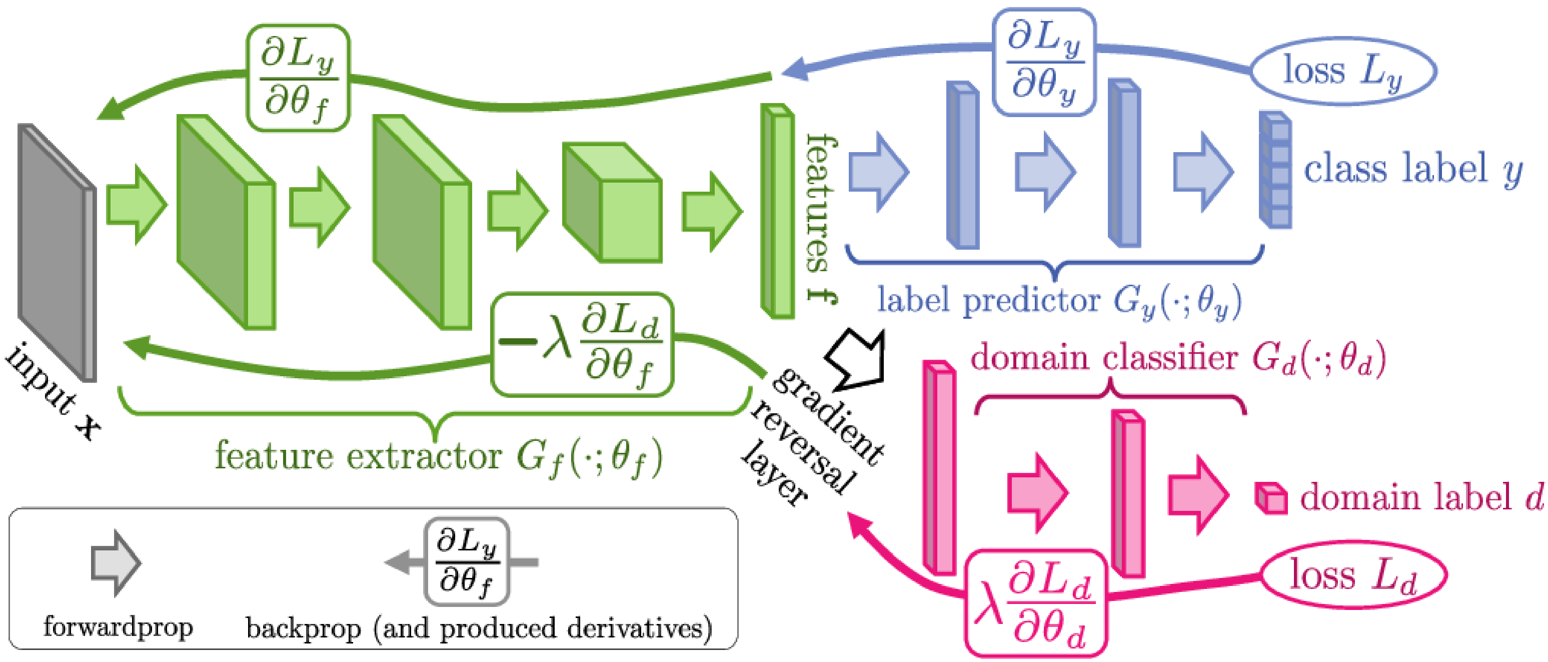
it is assumed...

- classification task on $\mathcal{X} \otimes \mathcal{Y}$
- there are two distributions $P_s(X_s)$ y $P_t(X_t)$
- $P_s(X_s)$ is shifted with respect to $P_t(X_t)$ en $\mathcal{X}$ (domain shift)
- objective: learn $P_t(y|x)$
- we have: $(x_i, y_i) \in X_s \otimes \mathcal{Y}$, $x_j \in X_t$

source link

domain adaptation neural network

[Ganin et al. 2015]



Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In Proceedings of the 32nd International Conference on Machine Learning. (Vol. 37).

domain adaptation neural network

use of data in training and testing

training/learning phase

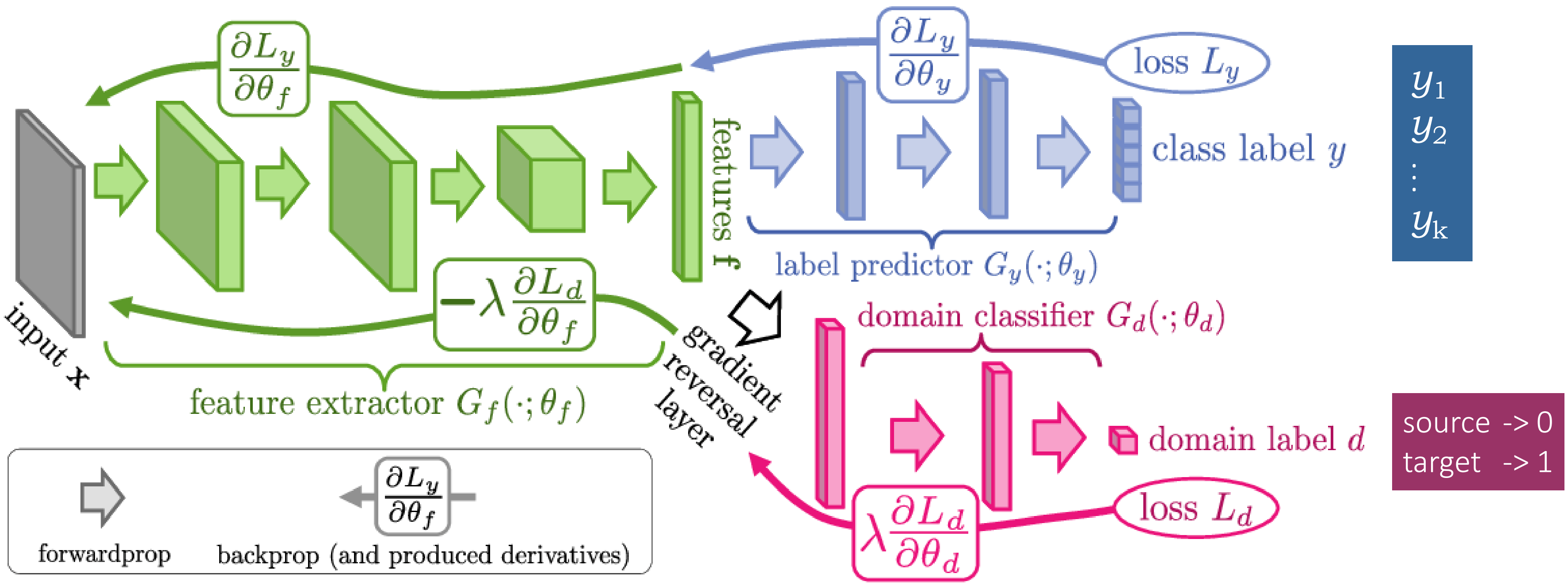
- $(x_i, y_i) \in X_s \otimes \mathcal{Y}$: labeled samples of the source domain
- $x_j \in X_t$: unlabeled samples of the target domain

test/exploitation phase

- goal: predict labels $y_j \in \mathcal{Y}$ for $x_j \in X_t$

domain adaptation neural network

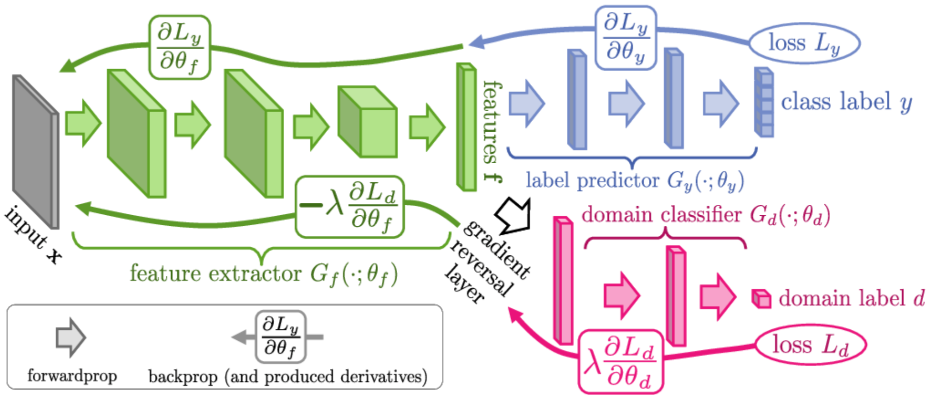
one feature space, two classification tasks



Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In Proceedings of the 32nd International Conference on Machine Learning. (Vol. 37).

domain adaptation neural network

loss function

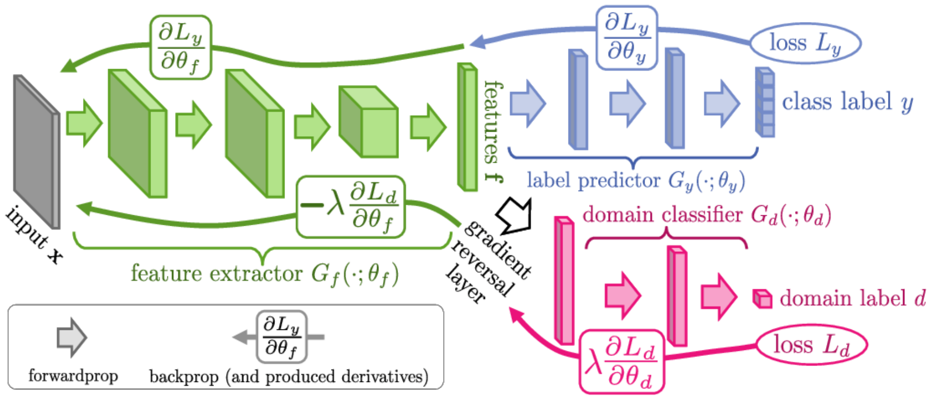


$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) - \lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), d_i\right)$$

Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In Proceedings of the 32nd International Conference on Machine Learning. (Vol. 37).

domain adaptation neural network

optimization



$$\theta_f \longleftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right)$$

$$\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y}$$

$$\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}$$

$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y \left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i \right) - \lambda \sum_{i=1..N} L_d \left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), d_i \right)$$

Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In Proceedings of the 32nd International Conference on Machine Learning. (Vol. 37).

domain adaptation neural network

optimization

$$\theta_f \longleftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right)$$

$$\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y}$$

$$\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}$$

```
(Xs,Ys), Xt <- training_data()
model, model_domain, model_source <- models(λ)
do_label = array([0]*batch_size + [1]*batch_size)
do_label_adv = array([1]*batch_size + [0]*batch_size)
for each update:
    Xs_b, Ys_b <- next(batch_generator(Xs, Ys))
    Xt_b <- next(batch_generator(Xt))
    save(θ_f)
    model_domain.train([Xs_b, Xt_b], do_label)
    restore(θ_f)
    save(θ_d)
    model.train([Xs_b, Xt_b], [Ys_b, zeros_like(Ys_b)], do_label_adv)
    restore(θ_d)
```

domain adaptation neural network

[Ganin et al. 2015]

METHOD — SOURCE / TARGET	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY	.5225	.8674	.5490	.7900
SA (FERNANDO ET AL., 2013)	.5690 (4.1%)	.8644 (−5.5%)	.5932 (9.9%)	.8165 (12.7%)
PROPOSED APPROACH	**.7666** (52.9%)	**.9109** (79.7%)	**.7385** (42.6%)	**.8865** (46.4%)
TRAIN ON TARGET	.9596	.9220	.9942	.9980

Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation.
In Proceedings of the 32nd International Conference on Machine Learning. (Vol. 37).

transfer learning

expected benefits