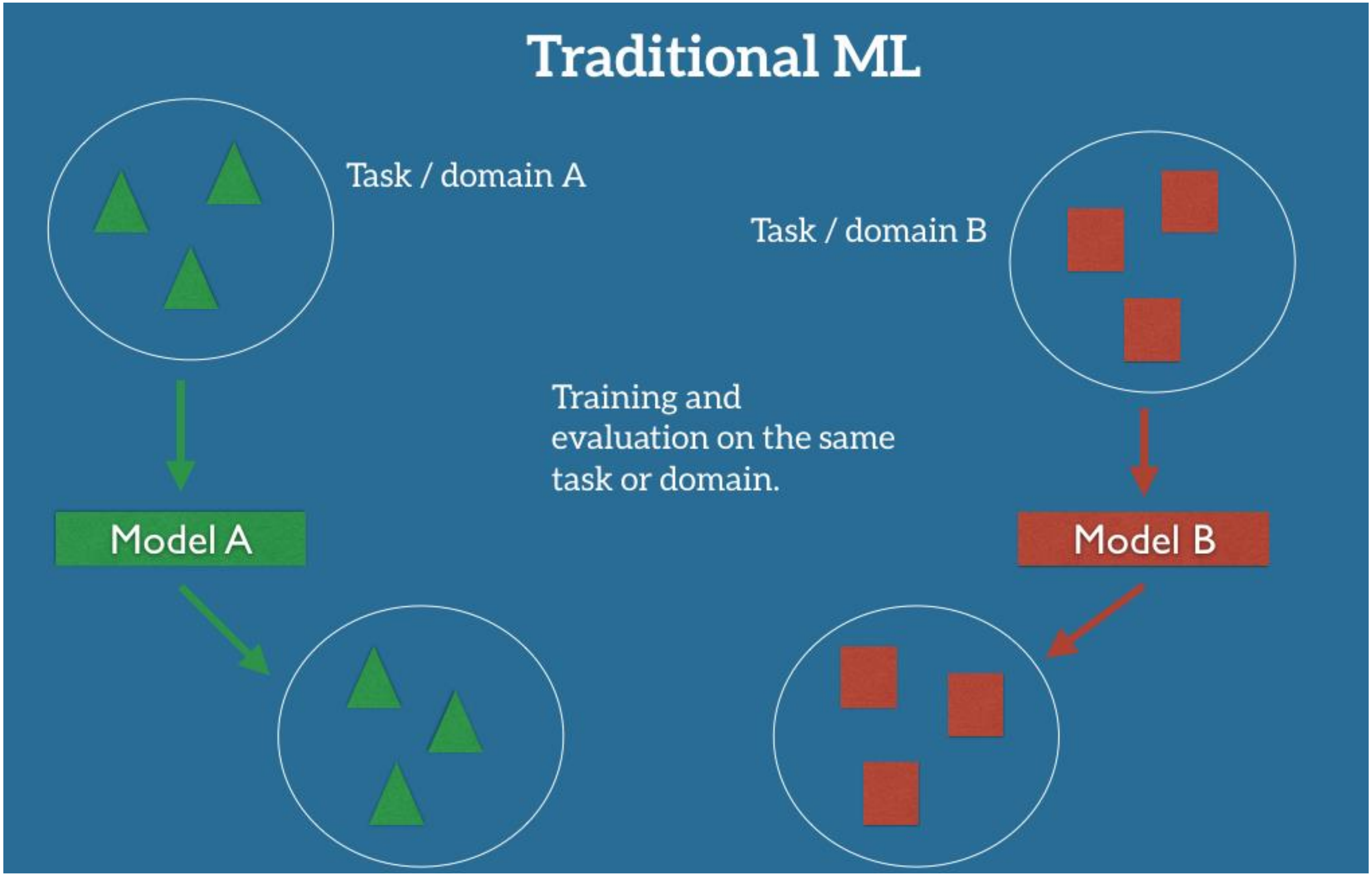
Transfer Learning and Domain Adaptation

Introduction [1]

- The ability of a system **to recognize and apply knowledge and skills** learned in previous tasks to novel tasks or new domains, which share some commonality.

- **It is motivated by human learning. People can often transfer knowledge learnt previously to novel situations**
  - **Chess -> Checkers**
  - **Mathematics -> Computer Science**
  - **Table Tennis -> Tennis**

The Traditional Supervised Learning Setup in ML [2]



The traditional supervised learning setup in ML [2]

- In the classic supervised learning scenario of machine learning, if we intend to train a model for some task and domain A, we assume that we are provided with labeled data for the same task and domain.

- We can see this clearly in Figure on previous slide, where the task and domain of the training and test data of our model A is the same.

- For the moment, let us assume that a task is the objective our model aims to perform, e.g. recognize objects in images, and a domain is where our data is coming from, e.g. images taken in San Francisco coffee shops.

The traditional supervised learning setup in ML [2]

- We can now train a model A on this dataset and expect it to perform well on unseen data of the same task and domain.

- On another occasion, when given data for some other task or domain B, we require again labelled data of the same task or domain that we can use to train a new model B so that we can expect it to perform well on this data.

The traditional supervised learning setup in ML [2]

- The **traditional supervised learning paradigm breaks down when we do not have sufficient labeled data** for the task or domain we care about to train a reliable model.

- **If we want to train a model to detect pedestrians on night-time images, we could apply a model that has been trained on a similar domain, e.g. on day-time images**.

- In practice, however, we often experience a **deterioration or collapse in performance** as the model has inherited the bias of its training data and does not know how to generalize to the new domain.

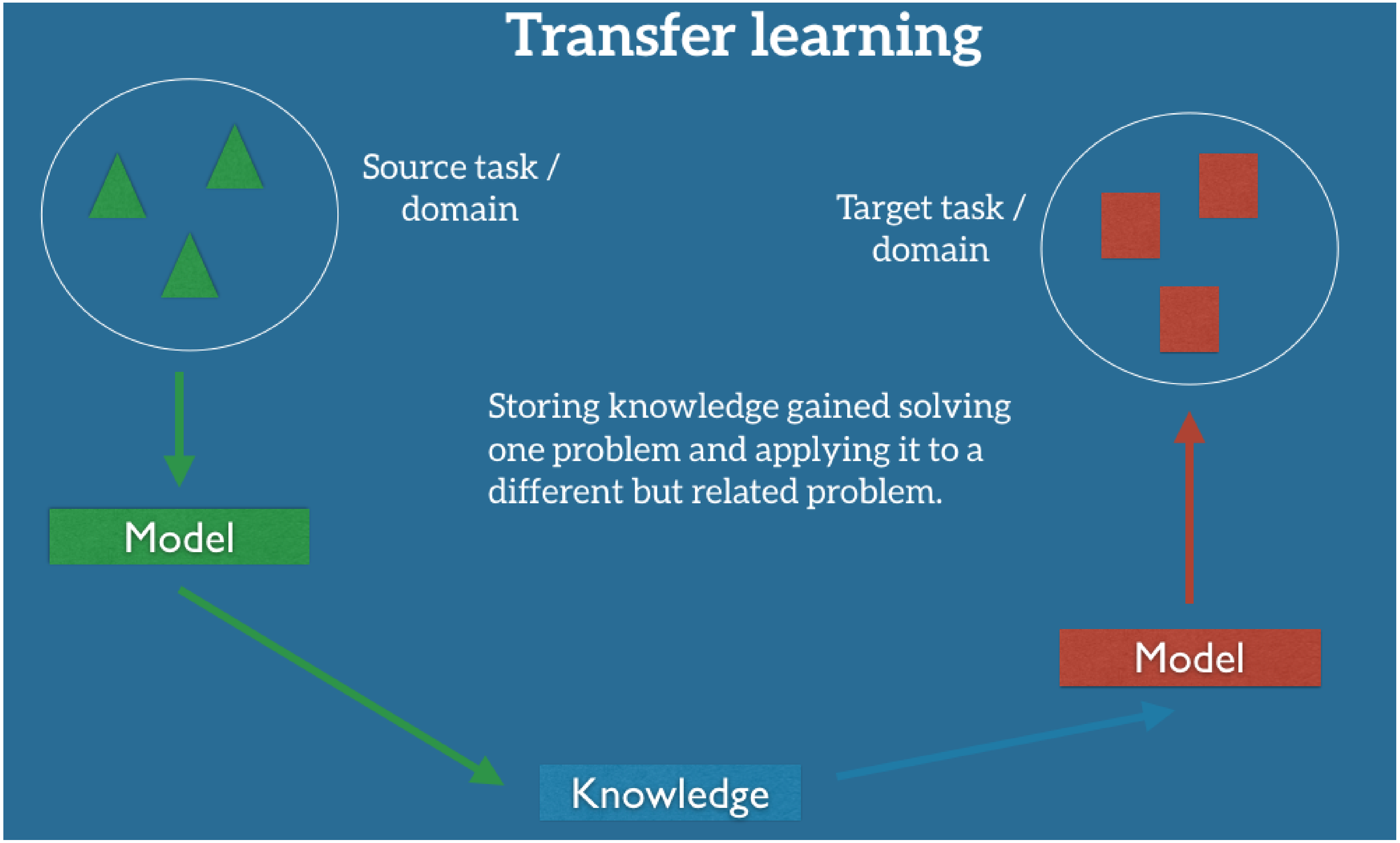
- If we want to train a model to perform **a new task, such as detecting bicyclists, we cannot even reuse an existing model, as the labels between the tasks differ.**

The Transfer Learning Setup [2]

- **Transfer learning allows us to deal with these scenarios by leveraging the already existing labeled data of some related task or domain.**

- **We try to store this knowledge gained in solving the source task in the source domain and apply it to our problem of interest as can be seen in Figure on Next Slide.**

The Transfer Learning Setup [2]



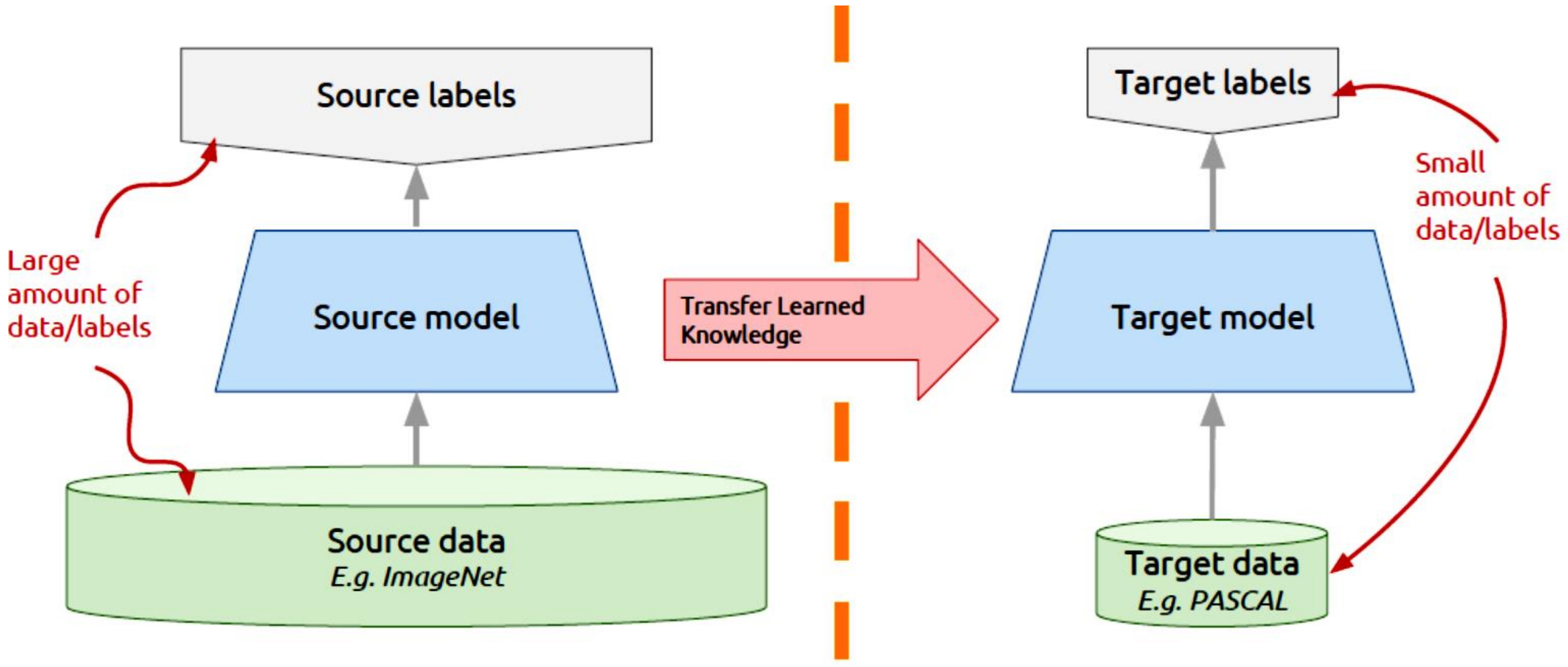
Storing knowledge gained solving one problem and applying it to a different but related problem.

Myth & Reality [3]

- Myth
  - you can't do deep learning unless you have a million labelled examples for your problem.
- Reality
  - You can transfer learned representations from a related task
  - You can train on a nearby surrogate objective for which it is easy to generate labels
  - You can learn useful representations from unlabelled data

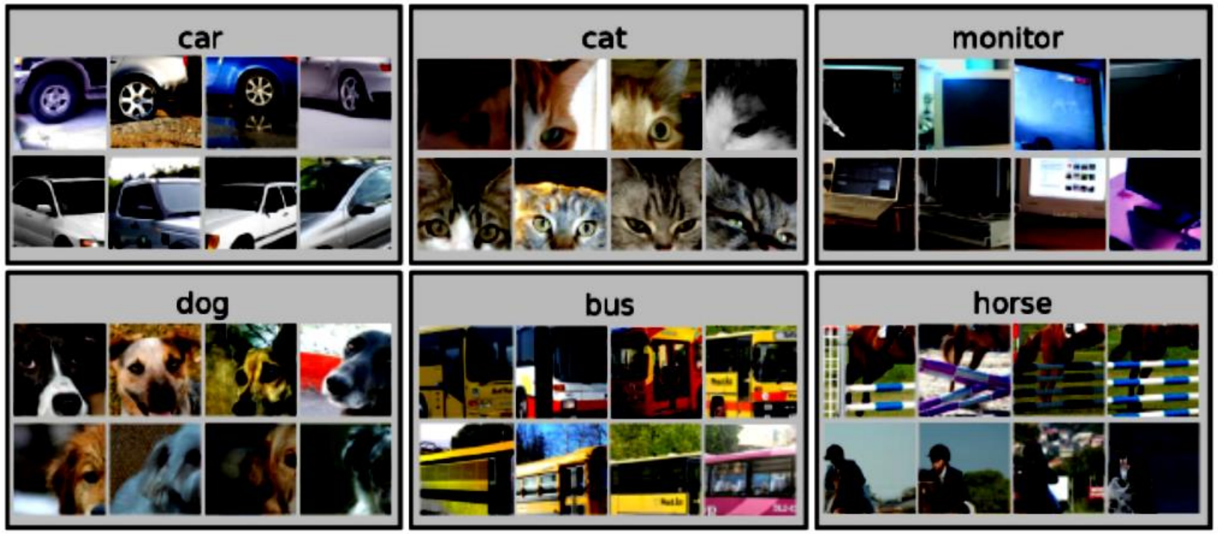
Transfer Learning: Idea [3]

- Instead of training a deep network from scratch for your task:
  - Take a network trained on a different domain for a different source task
  - Adapt it for your domain and your target task



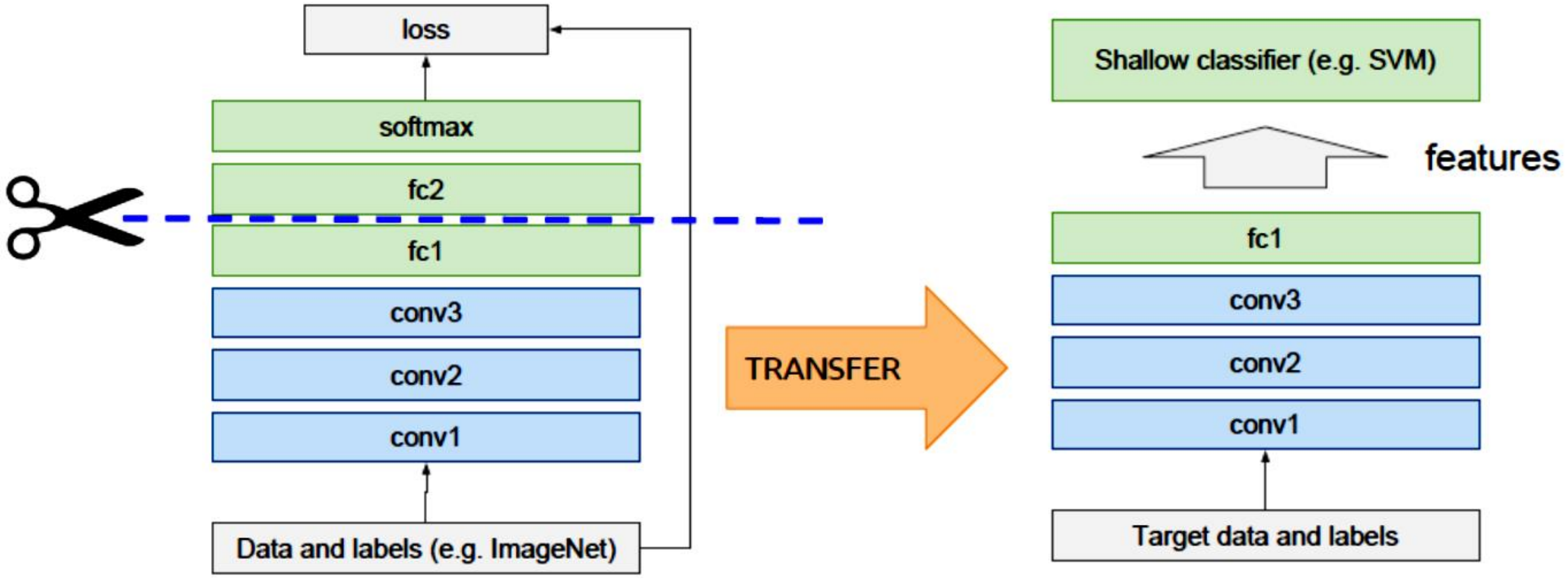
Example: PASCAL VOC 2007 [3]

- Standard classification benchmark, 20 classes, ~10K images, 50% train, 50% test
- Deep networks can have many parameters (e.g. 60M in Alexnet)
- Direct training (from scratch) using only 5K training images can be problematic. Model overfits.
- How can we use deep networks in this setting?



Off-the-shelf Features [3]

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.



Off-the-shelf Features [3]

Works surprisingly well in practice!

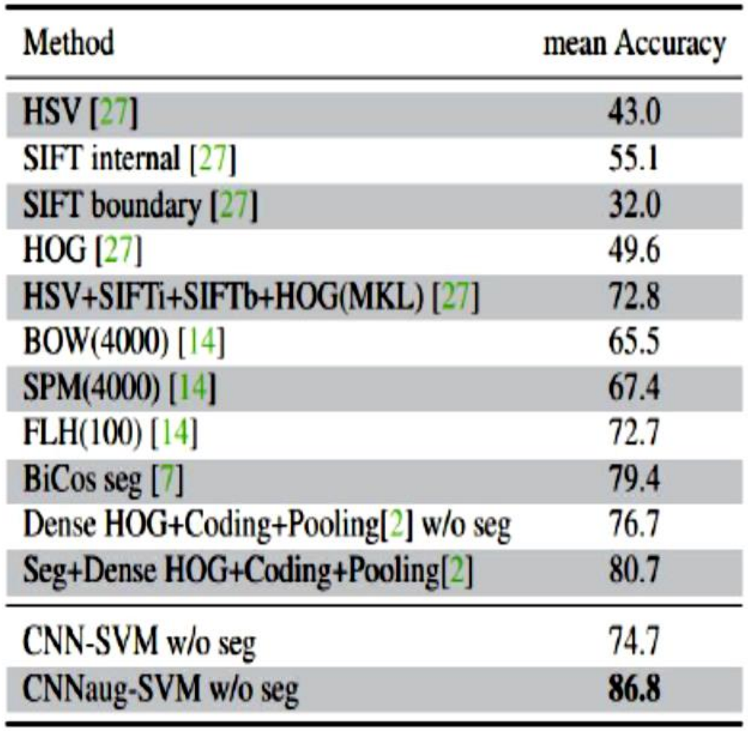
Surpassed or on par with state-of-the-art in several tasks in 2014

Image classification:

- PASCAL VOC 2007
- Oxford flowers
- CUB Bird dataset
- MIT indoors

Image retrieval:

- Paris 6k
- Holidays
- UKBench

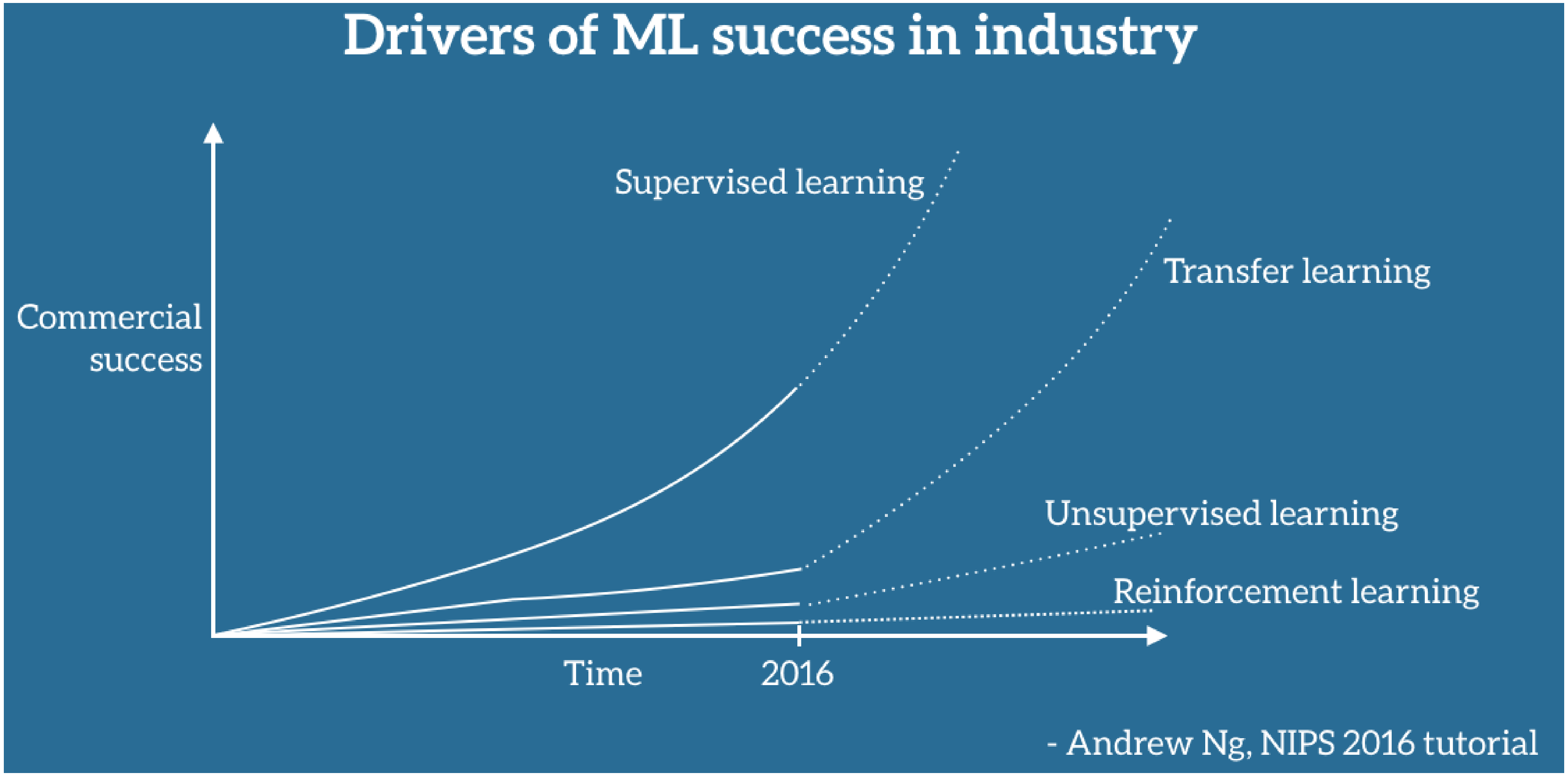


Method	mean Accuracy
HSV [27]	43.0
SIFT internal [27]	55.1
SIFT boundary [27]	32.0
HOG [27]	49.6
HSV+SIFTi+SIFTb+HOG(MKL) [27]	72.8
BOW(4000) [14]	65.5
SPM(4000) [14]	67.4
FLH(100) [14]	72.7
BiCos seg [7]	79.4
Dense HOG+Coding+Pooling[2] w/o seg	76.7
Seg+Dense HOG+Coding+Pooling[2]	80.7
CNN-SVM w/o seg	74.7
CNNaug-SVM w/o seg	**86.8**

Oxford 102 flowers dataset

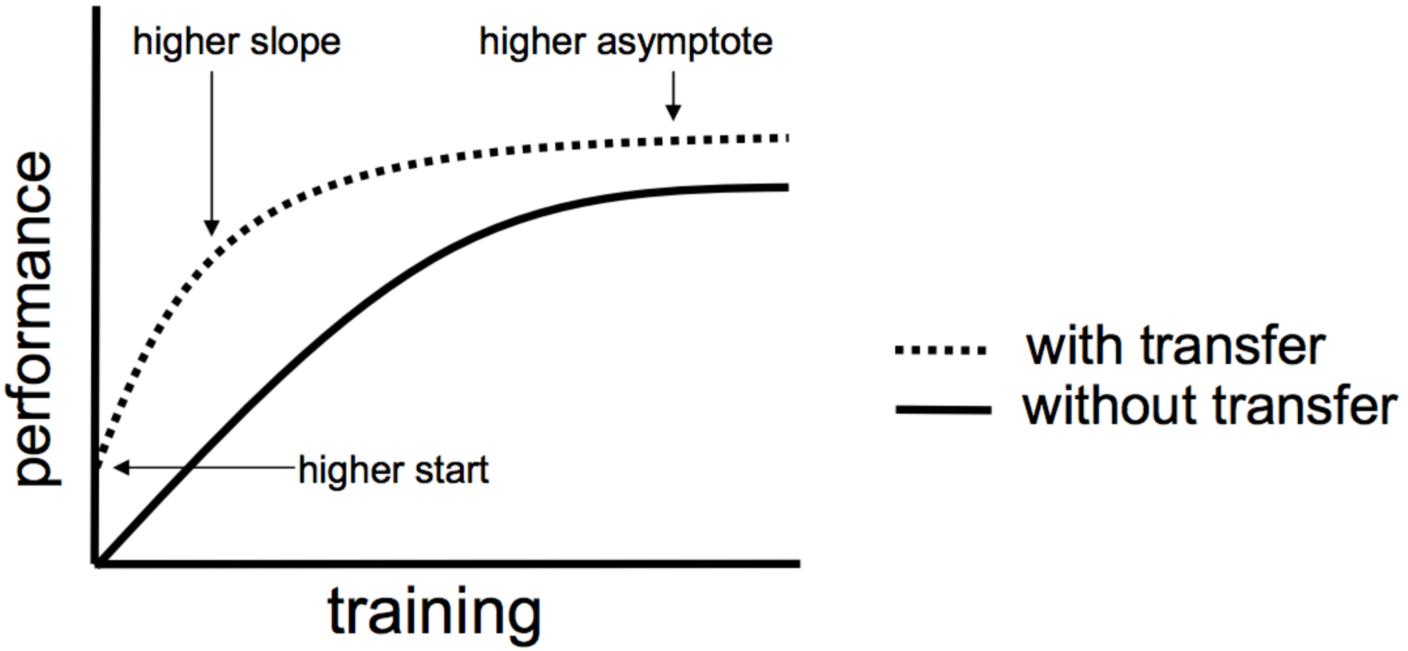
Razavian et al, **CNN Features off-the-shelf: an Astounding Baseline for Recognition,** CVPRW 2014 http://arxiv.org/abs/1403.6382

Why Transfer Learning? [2]



Benefits to look for when using Transfer Learning [4]

- **Higher start.** The initial skill (before refining the model) on the source model is higher than it otherwise would be.
- **Higher slope.** The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
- **Higher asymptote.** The converged skill of the trained model is better than it otherwise would be.



Benefits to look for when using Transfer Learning [4]

- **Ideally, you would see all three benefits from a successful application of transfer learning.**

- **It is an approach to try if you can identify a related task with abundant data and you have the resources to develop a model for that task and reuse it on your own problem, or there is a pre-trained model available that you can use as a starting point for your own model.**

- **On some problems where you may not have very much data, transfer learning can enable you to develop skilful models that you simply could not develop in the absence of transfer learning.**

- **The choice of source data or source model is an open problem and may require domain expertise and/or intuition developed via experience.**

Major Transfer Learning Scenarios [5]

- In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size.

- Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

Major Transfer Learning Scenarios [5]

- The Two Major Transfer Learning scenarios look as follows:
  - **ConvNet as fixed feature extractor**
  - **Fine-tuning the ConvNet**

Major Transfer Learning Scenarios [5]

- The Two Major Transfer Learning scenarios look as follows:
  - ConvNet as fixed feature extractor
    - Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer’s outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset.
    - In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes.
    - It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case).
    - Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

Major Transfer Learning Scenarios [5]

- The Two Major Transfer Learning scenarios look as follows:
  - Fine-tuning the ConvNet
    - The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation.
    - **It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network.**
    - **This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset.**

When and How to Fine Tune? [5]

- **How do you decide what type of transfer learning you should perform on a new dataset?**

- **This is a function of several factors, but the two most important ones are the size of the new dataset (small or big), and its similarity to the original dataset** (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images).

- **Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:**
  - **New dataset is small and similar to original dataset**
  - **New dataset is large and similar to the original dataset**
  - **New dataset is small but very different from the original dataset**
  - **New dataset is large and very different from the original dataset**

When and How to Fine Tune? [5]

- New dataset is small and similar to original dataset
  - Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns.
  - Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well.
  - Hence, the best idea might be to train a linear classifier on the CNN codes.

When and How to Fine Tune? [5]

- New dataset is large and similar to the original dataset
    - Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.

When and How to Fine Tune? [5]

- New dataset is small but very different from the original dataset
  - Since the data is small, it is likely best to only train a linear classifier.
  - Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features.
  - Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.

When and How to Fine Tune? [5]

- New dataset is large and very different from the original dataset
  - Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch.
  - However, in practice it is very often still beneficial to initialize with weights from a pretrained model.
  - In this case, we would have enough data and confidence to fine-tune through the entire network.

Practical Advice [5]

- There are a few additional things to keep in mind when performing Transfer Learning:
  - **Constraints from pretrained models.**
    - Note that **if you wish to use a pretrained network**, you may be **slightly constrained in terms of the architecture** you can use for your new dataset.
    - For example, you can't arbitrarily take out Conv layers from the pretrained network.
    - However, some changes are straight-forward: Due to parameter sharing, **you can easily run a pretrained network on images of different spatial size.**
    - This is clearly evident in the case of Conv/Pool layers because their forward function is independent of the input volume spatial size (**as long as the strides "fit"**).

Applications of Transfer Learning [2, 6]

- Transfer Learning with Image Data
- Transfer Learning with Language Data
- Learning from simulations
- Transferring knowledge across languages
- Adapting to new domains

Applications of Transfer Learning [6]

- **Transfer Learning with Image Data**
  - It is common to perform transfer learning with predictive modeling problems that use image data as input.
  - This may be a prediction task that takes photographs or video data as input.
  - **For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.**
  - The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware.

Applications of Transfer Learning [6]

- Transfer Learning with Image Data
  - These models can be downloaded and incorporated directly into new models that expect image data as input.
  - Three examples of models of this type include:
    - Oxford VGG Model
    - Google Inception Model
    - Microsoft ResNet Model
  - For more examples, see the Caffe Model Zoo where more pre-trained models are shared.
  - **This approach is effective because the images were trained on a large corpus of photographs and require the model to make predictions on a relatively large number of classes, in turn, requiring that the model efficiently learn to extract features from photographs in order to perform well on the problem.**

Applications of Transfer Learning [6]

- **Transfer Learning with Language Data**
  - It is common to perform transfer learning with natural language processing problems that use text as input or output.
  - **For these types of problems, a word embedding is used that is a mapping of words to a high-dimensional continuous vector space where different words with a similar meaning have a similar vector representation.**
  - Efficient algorithms exist to learn these distributed word representations and it is common for research organizations to release **pre-trained models trained on very large corpa of text documents** under a permissive license.

Applications of Transfer Learning [6]

- Transfer Learning with Language Data
  - **Two examples of models of this type include:**
    - **Google's word2vec Model**
    - **Stanford's GloVe Model**
  - These distributed word representation models can be downloaded and incorporated into deep learning language models in either the interpretation of words as input or the generation of words as output from the model.

Applications of Transfer Learning [2]

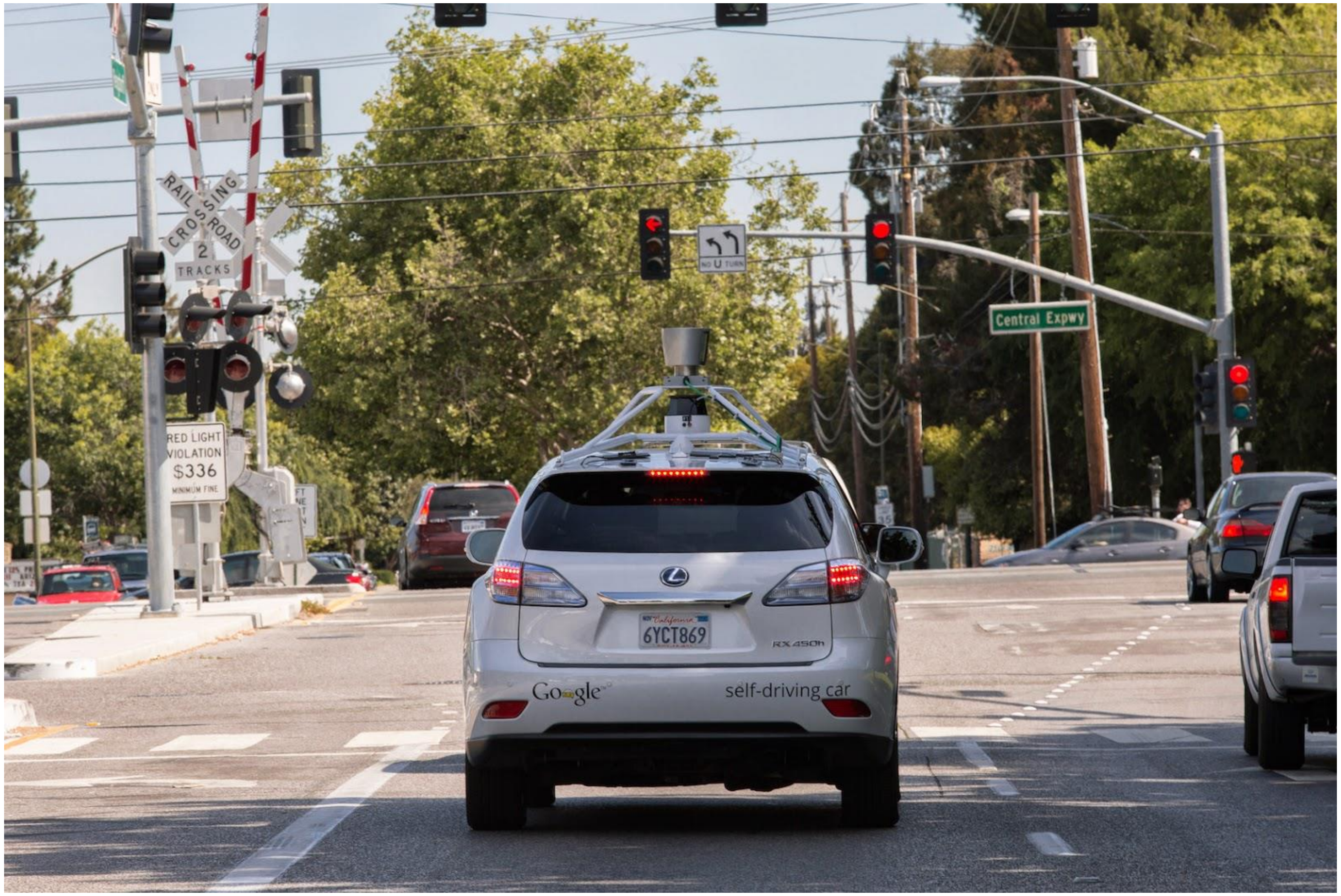
- Learning from simulations
  - For many machine learning applications that rely on hardware for interaction, gathering data and training a model in the real world is **either expensive, time-consuming, or simply too dangerous**.
  - **It is thus advisable to gather data in some other, less risky way.**
  - **Simulation is the preferred tool for this and is used towards enabling many advanced ML systems in the real world.**

Applications of Transfer Learning [2]

- Learning from simulations
  - Learning from a simulation and applying the acquired knowledge to the real world is an instance of transfer learning, as the feature spaces between source and target domain are the same (both generally rely on pixels), but the marginal probability distributions between simulation and reality are different, i.e. objects in the simulation and the source look different, although this difference diminishes as simulations get more realistic.
  - At the same time, the conditional probability distributions between simulation and real world might be different as the simulation is not able to fully replicate all reactions in the real world.
  - Learning from simulations has the benefit of making data gathering easy as objects can be easily bounded and analyzed, while simultaneously enabling fast training, as learning can be parallelized across multiple instances.

Applications of Transfer Learning [2]

- Learning from simulations
  - Consequently, it is a prerequisite for large-scale machine learning projects that need to interact with the real world, such as self-driving cars (Figure Below)



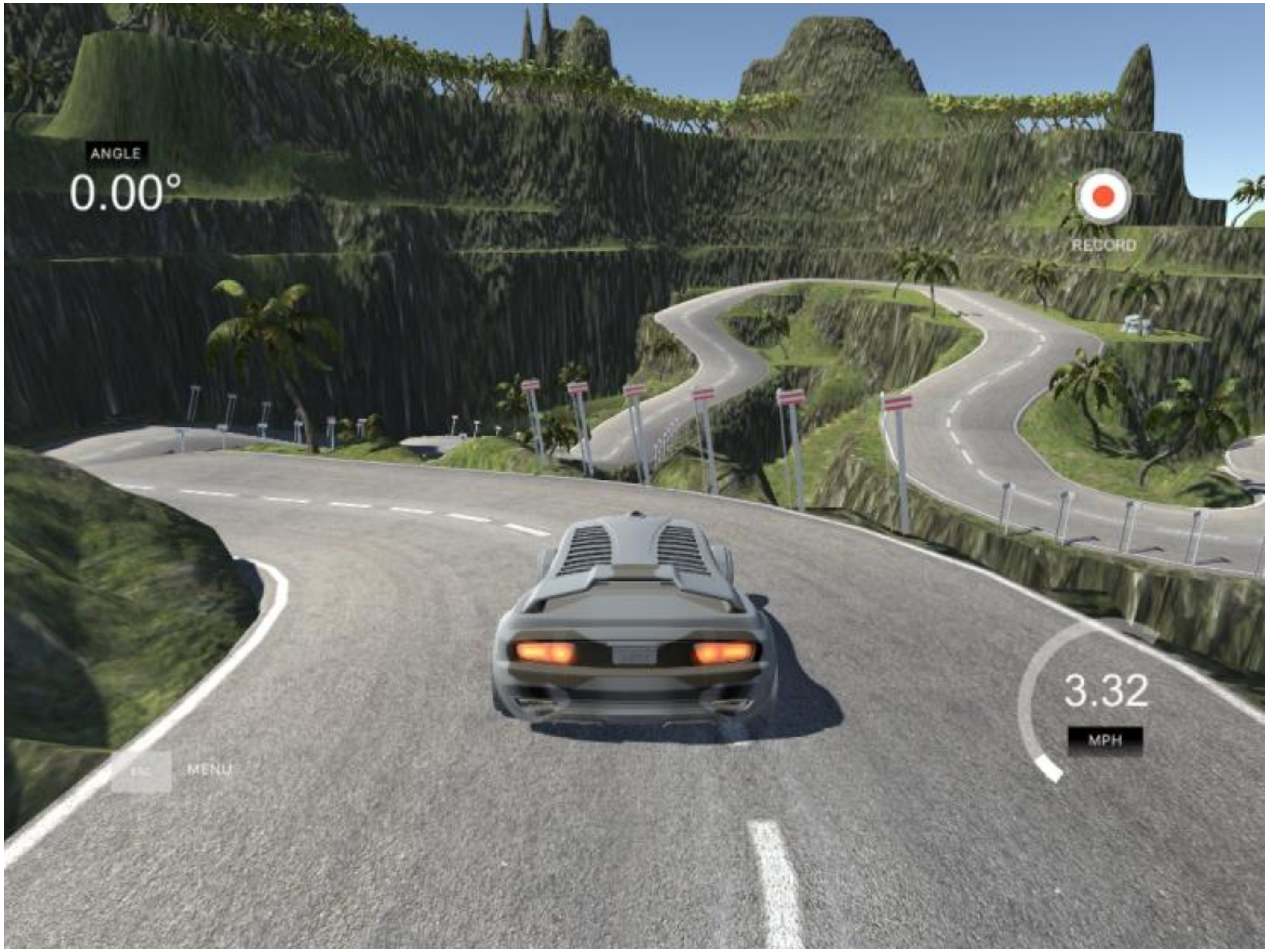
A Google self-driving car (source: Google Research blog)

Applications of Transfer Learning [2]

- Learning from simulations
  - **According to Zhaoyin Jia, Google's self-driving car tech lead, "Simulation is essential if you really want to do a self-driving car".**
  - **Udacity has open-sourced the simulator it uses for teaching its self-driving car engineer nanodegree, which can be seen in Figure on the next slide.**

Applications of Transfer Learning [2]

- Learning from simulations



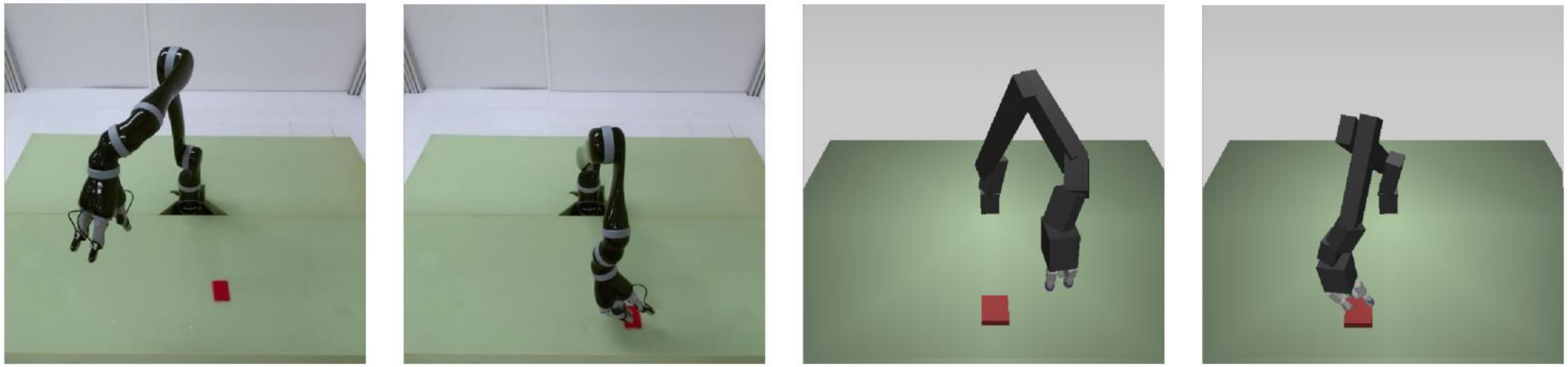
Udacity's self-driving car simulator (source: TechCrunch)

Applications of Transfer Learning [2]

- Learning from simulations
  - **Another area where learning from simulations is key is robotics: Training models on a real robot is too slow and robots are expensive to train.**
  - Learning from a simulation and transferring the knowledge to real-world robot alleviates this problem and has recently been garnering additional interest.
  - An example of a data manipulation task in the real world and in a simulation can be seen in Figure on the next slide.

Applications of Transfer Learning [2]

- Learning from simulations



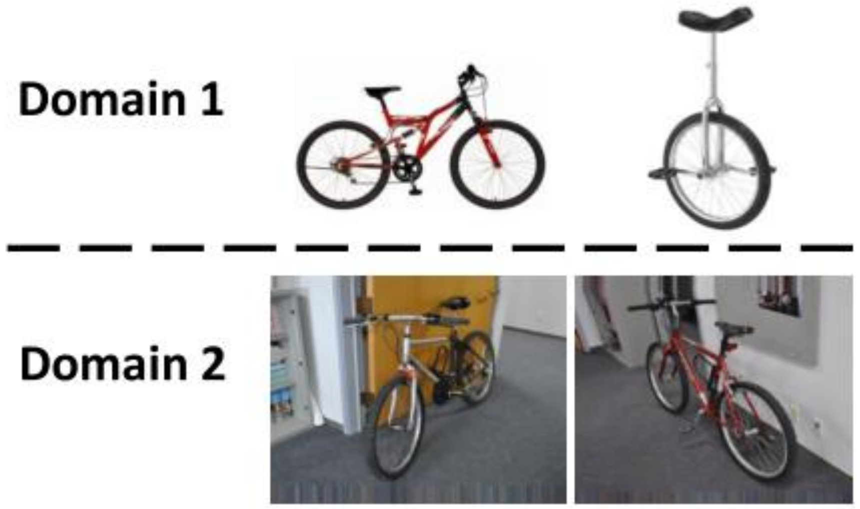
Robot and simulation images [7]

Applications of Transfer Learning [2]

- Transferring Knowledge across Languages
  - **Learning from one language and applying our knowledge to another language is another killer application of transfer learning, in the context of cross-lingual embedding models.**
  - Reliable cross-lingual adaptation methods would allow to leverage the vast amounts of labeled data we have in English and apply them to any language, particularly underserved and truly low-resource languages.
  - **Given the current state-of-the-art, this still seems utopian, but recent advances such as zero-shot translation [8] promise rapid progress in this area.**

Applications of Transfer Learning [2]

- Adapting to New Domains
  - 1.
    - **While learning from simulations is a particular instance of domain adaptation, it is worth outlining some other examples of domain adaptation.**



Different visual domains [9]

- Domain adaptation is a common requirement in vision as often the data where labelled information is easily accessible and the data that we actually care about are different, whether this pertains to identifying bikes as in Figure above or some other objects in the wild.

Applications of Transfer Learning [2]

- Adapting to New Domains
    - Even if the training and the the test data look the same, the training data may still contain a bias that is imperceptible to humans but which the model will exploit to overfit on the training data.

  - 2.

    - Another common domain adaptation scenario pertains to adapting to different text types: **Standard NLP tools such as part-of-speech taggers or parsers are typically trained on news data such as the Wall Street Journal**, which has historically been used to evaluate these models.

    - **Models trained on news data, however, have difficulty coping with more novel text forms such as social media messages and the challenges they present.**

Applications of Transfer Learning [2]

- Adapting to New Domains
  - 3.
    - **Even within one domain such as product reviews, people employ different words and phrases to express the same opinion.**
    - **A model trained on one type of review should thus be able to separate the general and domain-specific opinion words that people use in order not to be confused by the shift in domain.**

Applications of Transfer Learning [2]

- Adapting to New Domains
  - 4.
    - Finally, while the above challenges deal with general text or image types, problems are amplified if we look at domains that pertain to individual or groups of users: Consider the case of **automatic speech recognition (ASR).**
    - **Speech is poised to become the next big platform, with 50% of all our searches predicted to be performed by voice by 2020.**
    - **Most ASR systems are evaluated traditionally on the Switchboard dataset, which comprises 500 speakers.**
    - **Most people with a standard accent are thus fortunate, while immigrants, people with non-standard accents, people with a speech impediment, or children have trouble being understood.**
    - **Now more than ever do we need systems that are able to adapt to individual users and minorities to ensure that everyone's voice is heard.**

Negative Transfer [11]

- **Negative transfer happens when source domain data and task contribute to reduced performance of learning in the target domain.**

- **Causes:**
  - **Domains are too dissimilar.**
  - **Tasks are not well-related**

Domain Adaptation [10]

- Domain Adaptation is a field associated with machine learning and transfer learning.
- **Learning a discriminative classifier or other predictor in the presence of a shift between training and test distributions is known as domain adaptation (DA).**
- This scenario arises when we aim at learning from a source data distribution a well performing model on a different (but related) target data distribution.
- **For instance, one of the tasks of the common spam filtering problem consists in adapting a model from one user (the source distribution) to a new one who receives significantly different emails (the target distribution).**
- Note that, when more than one source distribution is available the problem is referred to as multi-source domain adaptation.

Types of Domain Adaptation [10]

- There are several contexts of domain adaptation. They differ in the information considered for the target task.

  - **The unsupervised domain adaptation:** the learning sample contains a set of labelled source examples, a set of unlabelled source examples and an unlabelled set of target examples.

  - **The semi-supervised domain adaptation:** in this situation, we also consider a "small" set of labelled target examples.

  - **The supervised domain adaptation:** all the examples considered are supposed to be labelled.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Learning a discriminative classifier or other predictor in the presence of a shift between training and test distributions is known as domain adaptation (DA).

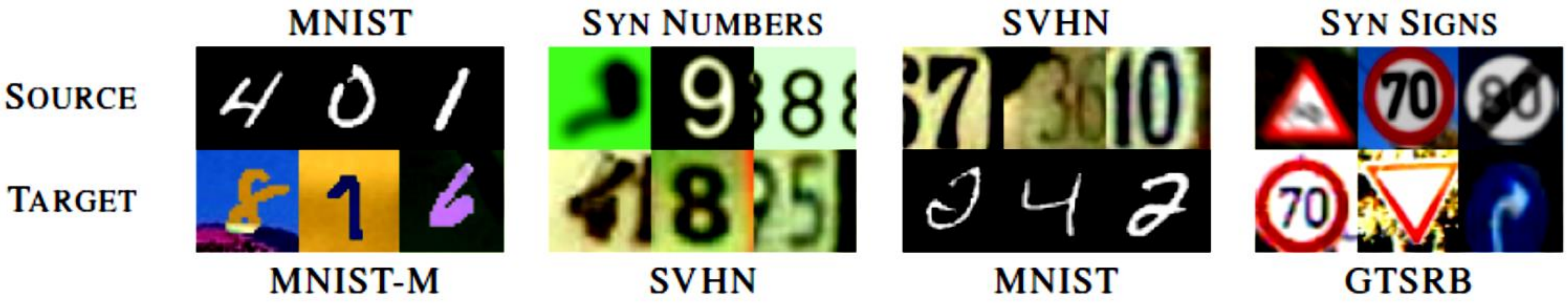


Figure 2. Examples of domain pairs used in the experiments. See Section 4.1 for details.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

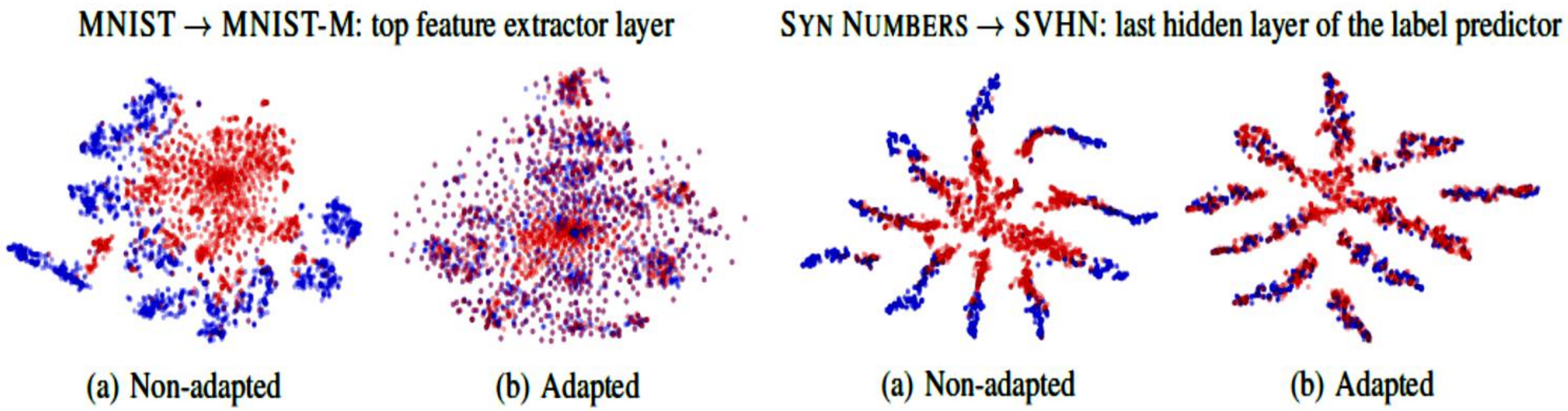


Figure 3. The effect of adaptation on the distribution of the extracted features (best viewed in color). The figure shows t-SNE (van der Maaten, 2013) visualizations of the CNN's activations **(a)** in case when no adaptation was performed and **(b)** in case when our adaptation procedure was incorporated into training. *Blue* points correspond to the source domain examples, while *red* ones correspond to the target domain. In all cases, the adaptation in our method makes the two distributions of features much closer.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

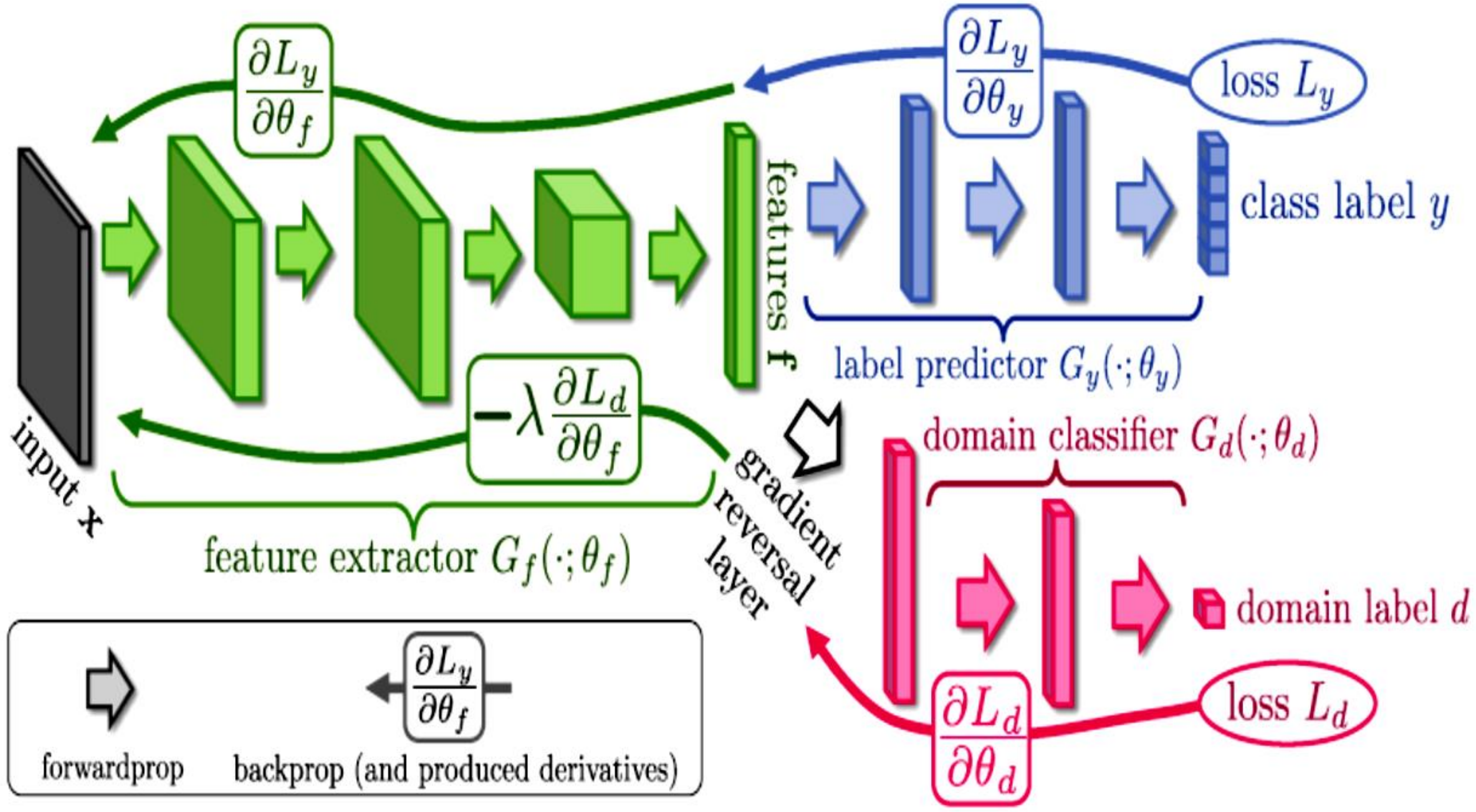
- Problem Statement
  - Given a labelled source domain and an unlabelled target domain, we would like to train a classifier or a predictor which would give accurate predictions on the target domain.

- Assumptions
  - Probability distribution of source domain is not equal to the probability distribution of target domain.
  - Source dataset is labelled. Target dataset is unlabelled.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Goal
  - Perform some transformation on the source and target domain and bring the transformed distributions closer. Then, Train the classifier on the transformed source distribution and since both the transformed distributions are now similar, The model will achieve better accuracy on the target domain during test time.
  - In-order to perform the transformation, a neural network is used.

Unsupervised Domain Adaptation by Backpropagation [11, 12]



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features **f**
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 1. The **proposed architecture** includes a deep *feature extractor* (green) and a deep *label predictor* (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a *domain classifier* (red) connected to the feature extractor via a *gradient reversal layer* that multiplies the gradient by a certain negative constant during the backpropagation-based training. Otherwise, the training proceeds in a standard way and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- **Feature Extractor:**
  - This is a neural network that will learn to perform the transformation on the source and target distribution.
- **Label Classifier:**
  - This is a neural network that will learn to perform classification on the transformed source distribution. Since, source domain is labelled.
- **Domain Classifier:**
  - This is a neural network that will be predicting whether the output of the Feature Extractor is from the source or the target distribution.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- **Idea:**
  - At training time, in order to obtain domain-invariant features, we seek the parameters θ_f of the feature mapping that maximize the loss of the domain classifier (by making the two feature distributions as similar as possible (mind the word feature distribution and not source distribution or target distribution)), while simultaneously seeking the parameters θ_d of the domain classifier that minimize the loss of the domain classifier.
  - In addition, we seek (the parameters θ_y) to minimize the loss of the label predictor.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- **Intuition:**
  - Basically, the feature extractor will try to perform some transformation on the source and target instances such that the transformed instances appear as if it is coming from the same distribution and domain classifier will not be able to classify the domain of the transformed instances.
  - This is achieved by training both the feature extractor and domain classifier in such a way that feature extractor will be trained to maximize the domain classification loss, while domain classifier will try to minimize the domain classification loss.
  - So, this is similar to adversarial training wherein the feature extractor is trying to confuse domain classifier by bringing the two distributions closer.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- **Intuition:**
  - For the transformed source instances the label predictor will be trained for predicting the labels of the source instances.
  - The feature extractor will therefore be trained to minimize classification loss of the label predictor and maximize classification loss of domain predictor.
  - The label predictor and domain predictor will be trained to minimize there respective classification loss.
  - Thus using the above three components, The Feature extractor will learn to produce discriminative and domain-invariant features.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Optimization with Backpropagation

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

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Optimization with Backpropagation

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f}\right) \tag{4}$$

$$\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \tag{5}$$

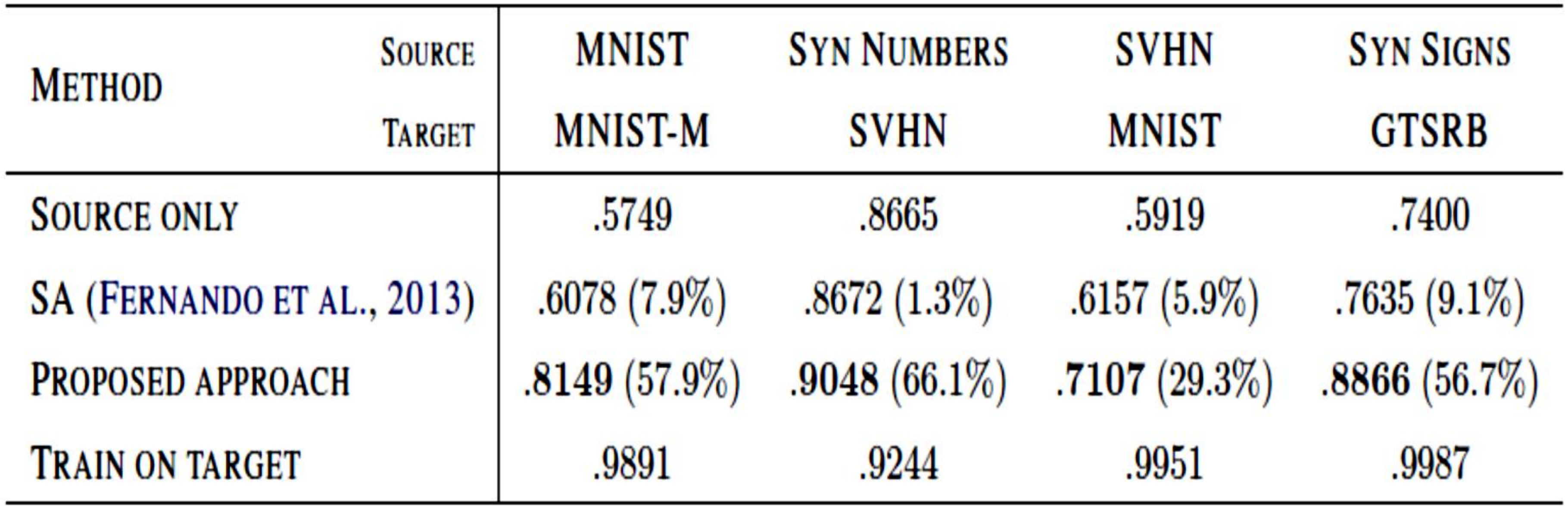
$$\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \tag{6}$$

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Gradient Reversal Layer
  - For training the feature extractor in order to maximize the classification loss of domain predictor, Gradient Reversal layer was place between Feature extractor and domain classifier.
  - The Gradient Reversal Layer basically acts as an identity function (outputs is same as input) during forward propagation but during back propagation it multiplies its input by $-\lambda$.
  - Intuitively, During back propagation the output of GRL is basically leading to the opposite of Gradient descent that is performing gradient ascent on the feature extractor with respect to the classification loss of Domain predictor.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

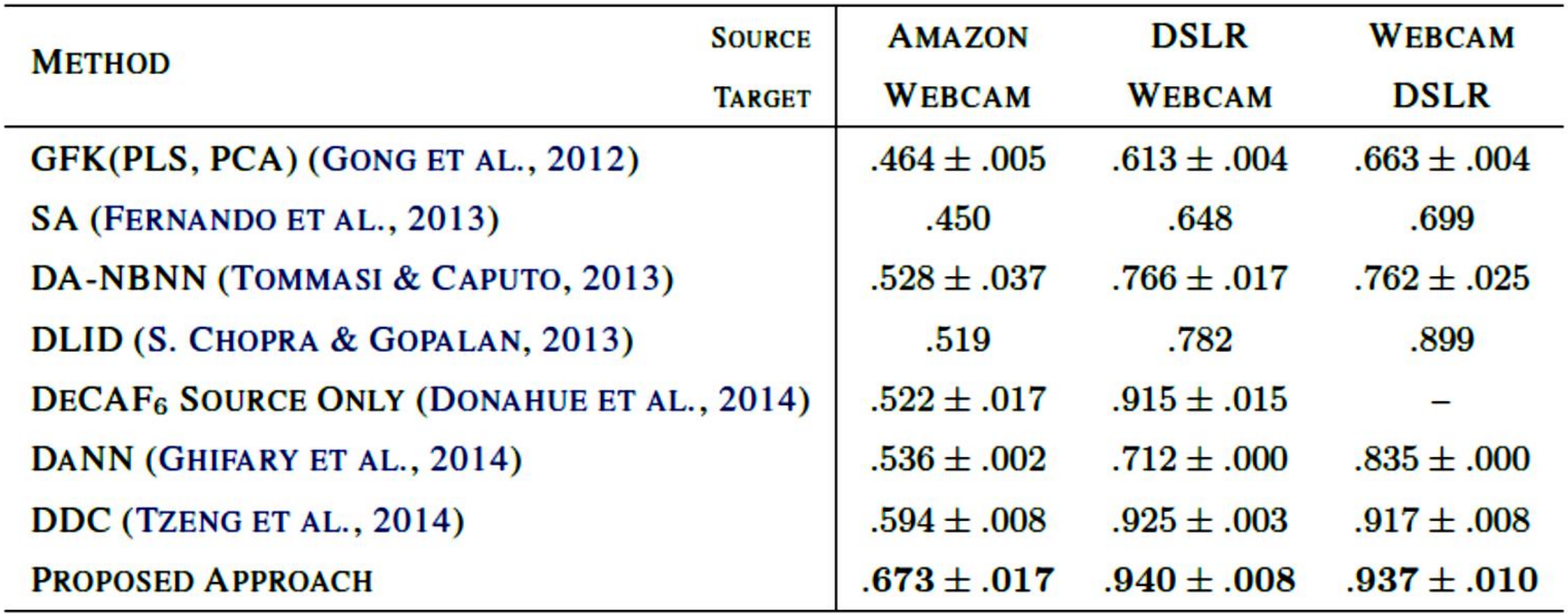
- Results



Method	Source	MNIST	Syn Numbers	SVHN	Syn Signs
	Target	MNIST-M	SVHN	MNIST	GTSRB
Source only		.5749	.8665	.5919	.7400
SA (Fernando et al., 2013)		.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
Proposed approach		.8149 (57.9%)	.9048 (66.1%)	.7107 (29.3%)	.8866 (56.7%)
Train on target		.9891	.9244	.9951	.9987

Unsupervised Domain Adaptation by Backpropagation [11, 12]

➢ Results



METHOD	SOURCE	AMAZON	DSLR	WEBCAM
	TARGET	WEBCAM	WEBCAM	DSLR
GFK(PLS, PCA) (GONG ET AL., 2012)		$.464 \pm .005$	$.613 \pm .004$	$.663 \pm .004$
SA (FERNANDO ET AL., 2013)		.450	.648	.699
DA-NBNN (TOMMASI & CAPUTO, 2013)		$.528 \pm .037$	$.766 \pm .017$	$.762 \pm .025$
DLID (S. CHOPRA & GOPALAN, 2013)		.519	.782	.899
DECAF$_6$ SOURCE ONLY (DONAHUE ET AL., 2014)		$.522 \pm .017$	$.915 \pm .015$	–
DANN (GHIFARY ET AL., 2014)		$.536 \pm .002$	$.712 \pm .000$	$.835 \pm .000$
DDC (TZENG ET AL., 2014)		$.594 \pm .008$	$.925 \pm .003$	$.917 \pm .008$
PROPOSED APPROACH		$.673 \pm .017$	$.940 \pm .008$	$.937 \pm .010$

Accuracy evaluation of different DA approaches on the standard OFFICE (Saenko et al., 2010) dataset. Prposed method (last row) outperforms competitors setting the new state-of-the-art.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Results

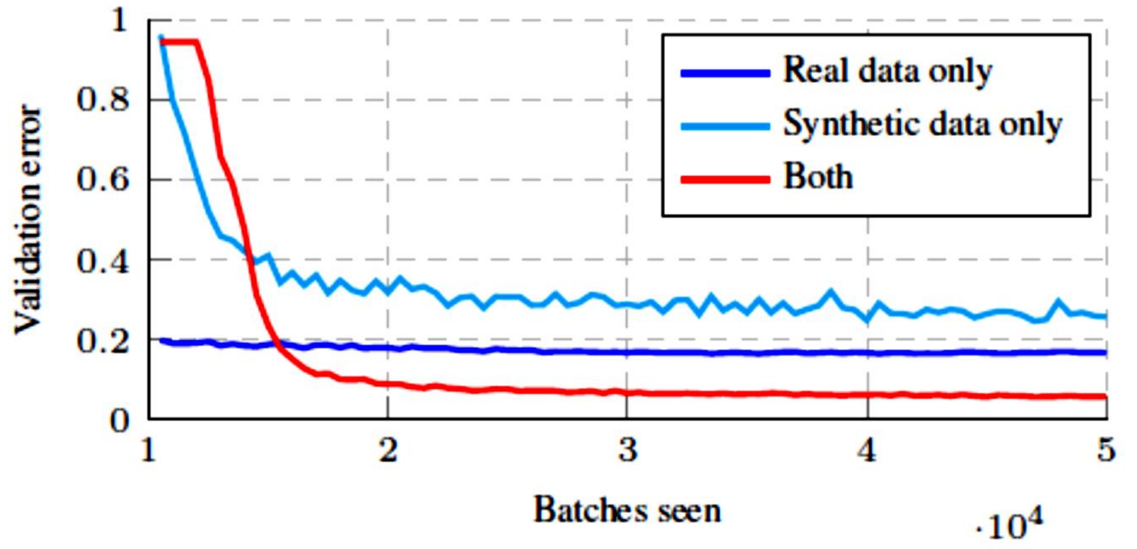


Figure 4. Semi-supervised domain adaptation for the traffic signs. As labeled target domain data are shown to the method, it achieves significantly lower error than the model trained on target domain data only or on source domain data only.

As an additional experiment, they also evaluated the proposed algorithm for semi-supervised domain adaptation, i.e. when one is additionally provided with a small amount of labeled target data. For that purpose we split GTSRB into the train set (1280 random samples with labels, original size > 50K images) and the validation set (the rest of the dataset). The validation part is used solely for the evaluation and does not participate in the adaptation. The training procedure changes slightly as the label predictor is now exposed to the target data.

Unsupervised Domain Adaptation by Backpropagation [11, 12]

- Pros
  - Brings both domains closer after performing some transformation
  - Learns discriminative and domain invariant features

- Cons
  - Gradient Reversal Layers leads to vanishing gradient problem once the domain predictor has achieved good accuracy

References

1. www.ntu.edu.sg/home/sinnopan/.../A%20Survey%20on%20Transfer%20Learning.ppt

2. http://ruder.io/transfer-learning/

3. http://imatge-upc.github.io/telecombcn-2016-dlcv/slides/D2L5-transfer.pdf

4. Olivas, Emilio Soria, ed. Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques. IGI Global, 2009.

5. http://cs231n.github.io/transfer-learning/

References

6. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

7. Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., & Hadsell, R. (2016). Sim-to-Real Robot Learning from Pixels with Progressive Nets. arXiv Preprint arXiv:1610.04286.

8. Johnson, M., Schuster, M., Le, Q. V, Krikun, M., Wu, Y., Chen, Z., … Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation.

9. Sun, B., Feng, J., & Saenko, K. (2016). Return of Frustratingly Easy Domain Adaptation. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

References

10. https://en.wikipedia.org/wiki/Domain_adaptation

11. https://pdfs.semanticscholar.org/presentation/f4af/de757b9dfc697d149e95cb193aa4749530e2.pdf

12. Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." arXiv preprint arXiv:1409.7495 (2014).

13. https://medium.com/@2017csm1006/unsupervised-domain-adaptation-by-backpropagation-da730a190fd2

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.