Representation Learning

An Introduction

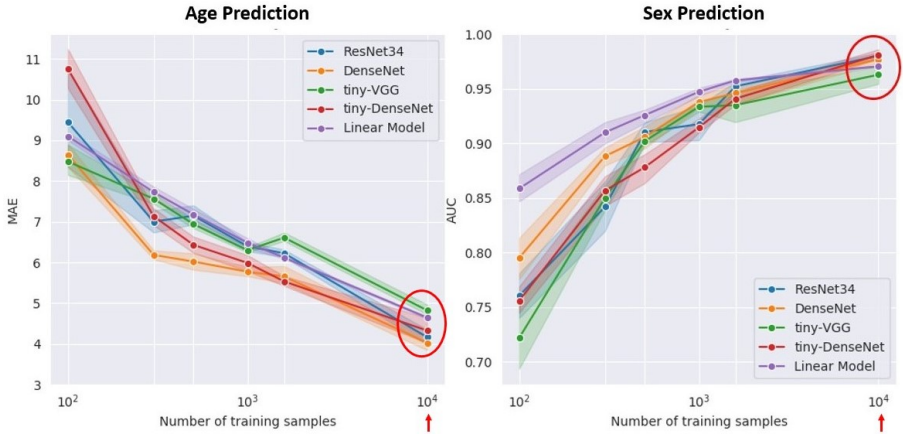
Pietro Gori

Associate Professor
Télécom Paris (IPParis)
Paris, France

Summary

Summary

- Deep learning (e.g., CNN or ViT) is a lazy and inefficient statistical method that needs millions if not billions of exemples to learn a precise task → **data hungry**



Introduction - Computer Vision

- Many specific tasks in Computer Vision, such as object detection[1] (e.g., YOLO), image classification[2] (e.g., ResNet-50), or semantic segmentation (e.g., U-Net), have reached astonishing results in the last years.
- Large and deep architectures (best performing) could be used mainly because:
  1. **large** ($N > 10^6$), **labeled** data-sets were *easily* accessible and *freely* available
  2. more computational power with hardware accelerators, such as **GPU** and TPU

[1] T.-Y. Lin et al. "Microsoft COCO: Common Objects in Context". In: *ECCV*. 2014.

[2] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR*. 2009.

- Mathematically, this comes from the fact that in machine (deep) learning, we usually follow the *empirical risk minimization principle*[3], which states that we should look for a model f, in the restricted space $\mathcal{F}$, that minimizes the *empirical risk* $\hat{R}(f)$, which is an approximation of the **risk function** $R(f)$ (or **generalization error**):

$$\arg\min_{f \in \mathcal{F}} R(f) = E_{(x,y)\sim p(X,Y)}[L(y, f(x))] \approx \hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) \tag{1}$$

- where we average the loss function L over the training set $\mathcal{D}_n = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$: n i.i.d.samples drawn from the fixed but unknown joint distribution $p(X, Y)$.
- We also assume that X is a real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$, and Y is a real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \ldots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression).

[3] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *NIPS*. 1991.

Empirical risk minimization

Let $\mathcal{F}$ be the restricted space of f:

- f^* : theoretical optimal model $\rightarrow f^* = \arg\min_f R(f)$
- $\tilde{f}$: theoretical best model in the restricted space $\mathcal{F} \rightarrow \tilde{f} = \arg\min_{f \in \mathcal{F}} R(f)$
- $\hat{f}$: best model in $\mathcal{F}$ based on the limited data $\mathcal{D}_n \rightarrow \hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}(f)$

We can notice that (without considering the optimization error):

$$\text{err}\{f^* - \hat{f}\} = \text{err}\{\underbrace{f^* - \tilde{f}}_{\text{approximation error}}\} + \text{err}\{\underbrace{\tilde{f} - \hat{f}}_{\text{estimation error}}\} \tag{2}$$

- The **approximation error** is due to the restriction of f to $\mathcal{F}$ since it is possible that $f^* \notin \mathcal{F} \rightarrow$ look for richer (many parameters) spaces $\mathcal{F}$ (e.g., deep learning)
- The **estimation error** is due to the limited training data. This tends to 0 when $n \rightarrow \infty$, (law of large numbers) $\rightarrow$ look for large, annotated data-sets

Empirical risk minimization

- Following the Vapnik's principle, we look for a model $\hat{f}$ that entails $\hat{R}(\hat{f}) \approx 0 \rightarrow$ this can be computed using the n training samples !
- However, we would like a model $\hat{f}$ which entails $R(\hat{f}) \approx 0$, but this can not be computed !
$\rightarrow$ under which conditions $R(\hat{f}) - \hat{R}(\hat{f}) \approx 0$?

[4] M. Mohri et al. *Foundations of Machine Learning*. MIT Press, 2019.

Empirical risk minimization

- Following the Vapnik's principle, we look for a model $\hat{f}$ that entails $\hat{R}(\hat{f}) \approx 0 \rightarrow$ this can be computed using the n training samples !
- However, we would like a model $\hat{f}$ which entails $R(\hat{f}) \approx 0$, but this can not be computed ! $\rightarrow$ under which conditions $R(\hat{f}) - \hat{R}(\hat{f}) \approx 0$?
- The Probably Approximately Correct (PAC) bound[4] states that, with probability $1 - \delta$ where $\delta > 0$, the following inequality holds:

$$\forall f \in \mathcal{F}, \qquad R(f) - \hat{R}(f) \leq \sqrt{\frac{1}{2n}\left(\log_2 |\mathcal{F}| + \log \frac{2}{\delta}\right)} \tag{3}$$

  - the greater n, the better $\hat{R}(f)$ approximates $R(f)$
  - the smaller the size of $\mathcal{F}$, the better $\hat{R}(f)$ approximates $R(f)$
  - Given n, it is better using a smaller and simpler $\mathcal{F} \rightarrow$ Occam's Razor principle (*"All other things being equal, a simpler model is better"*)

[4] M. Mohri et al. *Foundations of Machine Learning*. MIT Press, 2019.

Empirical risk minimization

- We can conclude that, if we use a rich DL model (many parameters, large $|\mathcal{F}|$):
  - the approximation error gets smaller (*lower bias*), but...
  - the estimation and generalization error may increase (*higher statistical complexity*) $\rightarrow$ it could perfectly adapt to the training samples and memorize them[5]
- We need to increase n to reduce the estimation and generalization error and not incur in the problem of **Overfitting**[6]
- This shows that we need:
  - large and labeled data-sets (big n)
  - powerful computational servers to train deep neural networks with millions of parameters (large $|\mathcal{F}|$)

[5] C. Zhang et al. "Understanding deep learning requires rethinking generalization". In: *ICLR*. 2017.

[6] Y. Abu-Mostafa et al. *Learning from data: A short course*. 2012.

Empirical risk minimization

• If we don't have much data, like in medical imaging where usual training data-sets have less than 1k images, what can we do to avoid Overfitting ?

- choose the **adequate hypothesis/model space** $\mathcal{F}$ (inductive bias problem)[7] $\rightarrow$ not always easy and very time-consuming. Need to try several models and hyper-parameters
- **Regularization** such as: weight decay, Dropout or early-stopping. $\rightarrow$ These are very important but they are not a miracle cure...
- **Data augmentation** (e.g., geometric and iconographic transformations) $\rightarrow$ can be very effective but one needs to find adequate transformations (not always easy for medical images)
- **Feature Engineering/Learning**
- **Transfer Learning**

[7] J. Baxter. "A Model of Inductive Bias Learning". In: *Journal of Artificial Intelligence Research* (2000).

Empirical risk minimization

- If we don't have much data, like in medical imaging where usual training data-sets have less than 1k images, what can we do to avoid Overfitting ?
  - choose the **adequate hypothesis/model space** $\mathcal{F}$ (inductive bias problem)[7] $\rightarrow$ not always easy and very time-consuming. Need to try several models and hyper-parameters
  - **Regularization** such as: weight decay, Dropout or early-stopping. $\rightarrow$ These are very important but they are not a miracle cure...
  - **Data augmentation** (e.g., geometric and iconographic transformations) $\rightarrow$ can be very effective but one needs to find adequate transformations (not always easy for medical images)
  - Feature Engineering/Learning
  - Transfer Learning

[7] J. Baxter. "A Model of Inductive Bias Learning". In: *Journal of Artificial Intelligence Research* (2000).

Summary

Summary

Feature Engineering

- The main idea behind Feature Selection/Engineering is to identify new features that :
  - are *not-redundant*
  - are possibly of *lower dimension* than the original features
  - *better describe* the original data
  - are *useful* for one (or more) downstream tasks
- Until the advent of Deep Learning[8], the feature selection process was mainly **manual**.
- Good results and representations are *interpretable* and *explainable* but ... it is usually tedious, time-consuming, labor-intensive and only for *experienced* practitioners

[8] A. Krizhevsky et al. "ImageNet classification with deep convolutional neural networks". In: *NIPS*. 2012.

Manual feature selection in neuroimaging

- During the last 30 years, there has been a lot of work about feature engineering in some applications of Medical Imaging, like anatomical brain imaging:

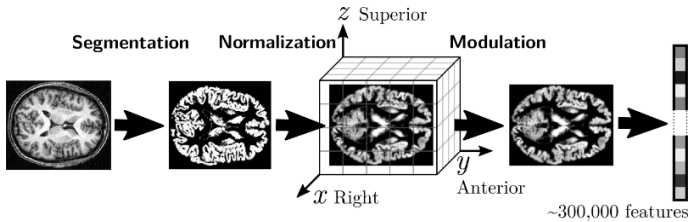


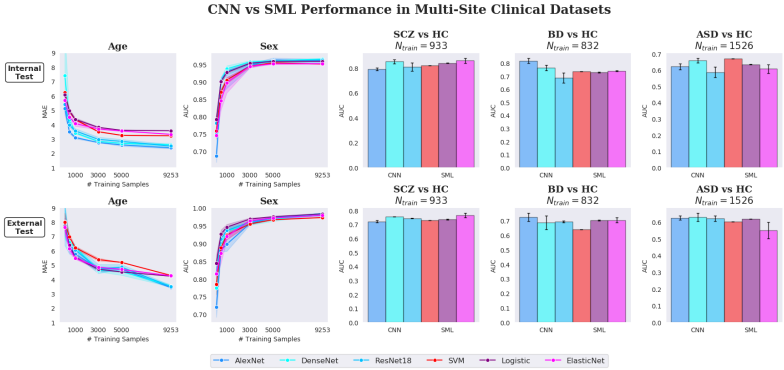
Figure: Feature extraction with VBM pipeline from MR T1-w images[9].

- where linear models applied to accurately extracted features (after a series of specifically designed pre-processing steps) can still compete with the most recent Deep learning algorithms[10] !

[9] J. Ashburner. "A fast diffeomorphic image registration algorithm". In: *NeuroImage* (2007).

[10] B. Dufumier. "Representation learning in neuroimaging". PhD thesis. 2022.

Manual feature selection in neuroimaging



- But... Brain Imaging has received a tremendous attention in the last 30 years (...even politicians are afraid of Alzheimer's disease...) and not all applications can benefit from the works of thousands of researchers ! $\rightarrow$ **Generic automatic feature learning algorithms** could be used in all applications without requiring a big expertise on the data

Summary

Feature/Representation Learning

Goal

Automatically learn discriminative, relevant and well-organized representations $f(x)$ of the original data x which are useful and generalizable to one or more downstream tasks

- The first and well-known methods for Automatic Representation Learning are the unsupervised dimensionality reduction methods, such as:
  - Principal Component Analysis (PCA)
  - Independent Componenent Analysis (ICA)
  - Non-negative Matrix factorization (NNMF)
- These methods can be very performing but are all based on strong assumptions (e.g., linearity, variance is interesting, statistical independence, positiveness)
- **Deep learning** gives a (more) generic way to automatically learn well-adapted representations, but...

... How should it be a "good" representation f ?[11]

1. **Smooth**
2. **Compact yet explanatory**
3. **Distributed**
4. **Hierarchical**
5. **Invariant**
6. **Disentangled**
7. **Generic, Well organized**

[11] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

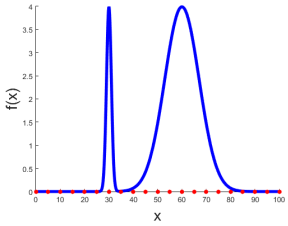
Summary

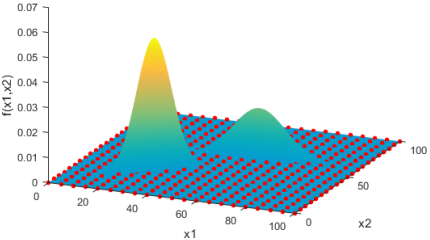
- **Smooth**: it implies that if $x \approx y$ then $f(x) \approx f(y)$. This prior is integrated in most unsupervised ML algorithms since prediction is usually achieved by a form of local interpolation between neighboring training examples[12] (e.g., kernel methods such as K-PCA, Isomap[13], LLE[14]).

- But... if the data are represented in a very large raw input space, due to the curse of dimensionality, we will need an exponentially growing number of samples to correctly reconstruct the entire functional landscape of f (and all its wrinkles)

[12] Y. Bengio et al. "Non-Local Manifold Tangent Learning". In: *NIPS*. 2004.
[13] J. B. Tenenbaum et al. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* (2000).
[14] S. T. Roweis et al. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* (2000).

Curse of dimensionality



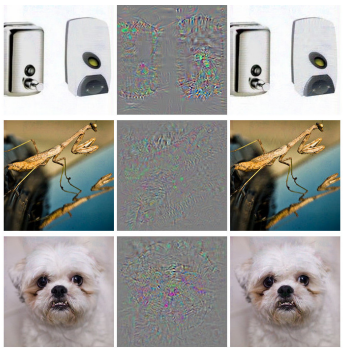


- *Curse of dimensionality*[15]: we assume that in 1D we need 21 samples to correctly retrieve the true latent function f. If the same process is transposed in 2D, we would need 21^2 samples, in 100D we will need 21^{100} samples! We have an exponentially growing number of samples to correctly sample the entire input space

[15] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Are all deep learning representations smooth ?

- Not necessarily... we can almost always find adversarial examples: generated images obtained by applying small perturbations to correctly classified input images, so that they are no longer classified correctly $\rightarrow$ no locally smooth, there are blind spots !



Adversarial examples[a] generated for AlexNet. (Left) is a correctly predicted sample, (center) difference between correct image and image predicted incorrectly, (right) adversarial example. All images in the right column are predicted to be an “ostrich, Struthio camelus”

[a] C. Szegedy et al. “Intriguing properties of neural networks”. In: *ICLR*. 2014.

Summary

Compact yet explanatory

- To avoid the curse of dimensionality, we usually also require the representation to be **compact** (or minimal)
- Mathematically, this mean learning a mapping: $f : \mathbb{R}^d \rightarrow \mathbb{R}^p$ where $p < d$
- Furthermore, the new features need to be **explanatory**, which means **expressive** enough to represent and capture most of the possible input configurations.
- The concept of **expressiveness** is very important in Representation Learning and several measures have been proposed to quantify it using, for instance:
  - Information theoretic measures
  - Supervised discriminative signals
  - Reconstruction error

Compact yet explanatory

- *Information theoretic measures*:
  - **InfoMax principle**: maximizing mutual information between inputs and representations $\max_\theta I(x; f_\theta(x))$[16] $\rightarrow$ MI is not necessarily a good predictor of the model performance[17]
  - The **information bottleneck** (IB) principle[18]: maximizing mutual information between representations and outputs while minimizing between inputs and representations $\max_\theta I(f_\theta(x); y) - \beta I(x; f_\theta(x)) \rightarrow$ seeks a minimal set of informative features for the chosen task. Low generalizability and transferability
  - **Maximal Coding Rate Reduction** (MCR2)[19]: learn a representation that discriminates between classes while being maximally diverse $\rightarrow$ task dependent

[16] A. J. Bell et al. "An Information-Maximization Approach to Blind Separation...". In: *Neural Computation* (1995).
[17] M. Tschannen et al. "On Mutual Information Maximization for Representation Learning". In: *ICLR*. 2020.
[18] N. Tishby et al. "Deep Learning and the Information Bottleneck Principle". In: *IEEE ITW*. 2015.
[19] Y. Yu et al. "Learning Diverse and Discriminative Representations...". In: *NeurIPS*. 2020.

Compact yet explanatory

- *Supervised signals*
  - Minimizing the **cross-entropy** loss over a labeled training data-set (x, y) $\rightarrow$ highly task-dependent
- *Reconstruction error*
  - **Auto-encoders** $\rightarrow$ estimate a compact and informative representation that reconstructs the original images $\rightarrow$ results vary depending on metric (L2, L1, etc.), additive discriminative network (e.g., GAN), regularity terms, making it task-dependent

- Measuring the expressiveness or meaningfulness of a representation is an active research area.

- Most of these measures are task-dependent or based on a precise statistical measure (e.g., Mutual Information) $\rightarrow$ Need to quantify the **generalizability** of a representation to several, and possibly new, tasks

Summary

Distributed Representations

- **Local or non-distributed representations**: we create a feature for each possible input sample (or configuration). Thus, each feature represents a different input sample. This is also called one-hot feature representation.

- In a neural network, we would dedicate one neuron to every possible input sample → very simple, easy to code, easy to learn but.. highly **inefficient** !

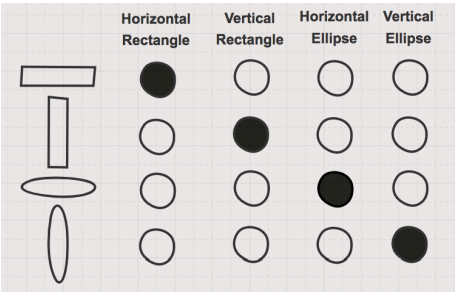
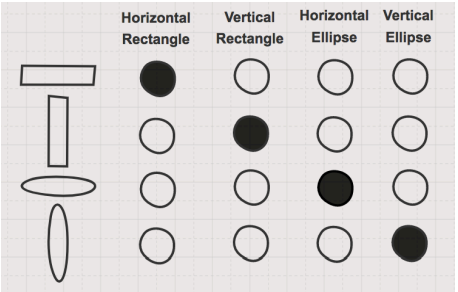


Figure: Each individual shape is represented by a single neuron. Image by Garrett Hoffman.

Distributed Representations

- Examples of one-hot feature representations are: clustering algorithms, mixtures, NN, decision trees, Gaussian SVM $\rightarrow$ they need $O(N)$ parameters/examples to distinguish $O(N)$ different input samples.[20]

- It is inefficient because it does not leverage possible relationships/similarities between input samples.



[20] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

Distributed Representations

- **Distributed representations**: features are related to general and high-level concepts that are shared between input samples and describe their intrinsic variability.

- It is highly efficient since we don't need one neuron per sample but we can describe many samples with few features.[21]

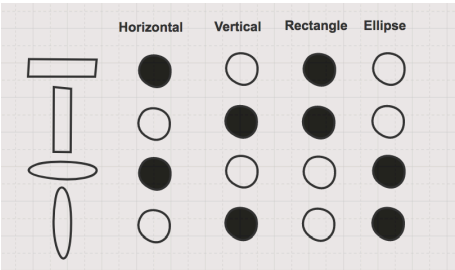


Figure: Each shape is represented by shared and high-level features. Image by Garrett Hoffman.

[21] Y. Bengio et al. "Non-Local Manifold Tangent Learning". In: *NIPS*. 2004.

Distributed Representations

- Distributed representations from neural network architectures (auto-encoders, multi-layer NN, CNN, RBMs, etc.) can all represent up to $O(2^N)$ input samples using only $O(N)$ parameters.[22]
- In a distributed representation, each input sample is represented by a certain number of active features, and each feature is involved in representing several input samples.

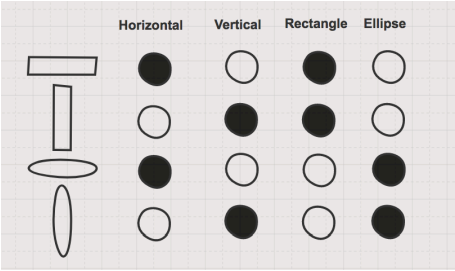


Figure: Each shape is represented by shared and high-level features. Image by Garrett Hoffman.

[22] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

Summary

Hierarchical Representations

- Deep architectures are highly used in representation learning because, even if they may be more difficult to train, they can:
  - learn multiple levels of features which constitute a hierarchy that goes from low-level to high-level/abstract features
  - promote the re-use of features, which explains the power of distributed representations.[23]

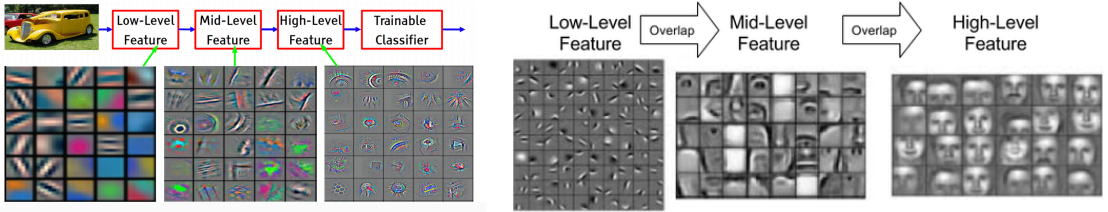


Figure: Visualization of input stimuli that excite individual feature maps at different layers.[24]

[23] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

[24] M. D. Zeiler et al. "Visualizing and Understanding Convolutional Networks". In: *ECCV*. 2014.

Hierarchical Representations

- In a hierarchical representation (e.g., CNN):
  - low-level features are related to contours, edges, angles, colors, texture $\rightarrow$ closer to raw pixels, retrieved at locale scale, more sensitive to noise and changes, less interpretable
  - high-level features describe semantically meaningful concepts like entire objects, faces, etc. $\rightarrow$ more abstract, retrieved at global scale (bigger receptive field), less sensitive to noise and changes (i.e., invariant), more interpretable

- Furthermore, deep and hierarchical architectures have many ways to **re-use** and **compose** the different features at different levels $\rightarrow$ the number of ways can grow exponentially with the depth of the network !

Summary

Invariant Representations

- A representation should be invariant to geometric/iconographic nuisance transformations $\rightarrow$ features should be *insensitive* to these nuisance variations that are uninformative for the aimed downstream tasks
- The high-level, abstract features corresponding to the animal "squirrel" should activate for both images
- In CNN: convolutions + non-linearities + pooling (subsampling) produce shift-invariant representations

Invariant Representations

- However, how to determine a priori which are the irrelevant/nuisance variations ? If the representations are destined to multiple tasks, they might have distinct relevant features !

- The most common answer to this question is to **preserve** as much as possible of the **information** in the data removing only the information that can be considered as irrelevant for all aimed tasks (e.g., the previous geometric transformations)

- Ultimately, the research postulate mostly followed by researchers is: "*the most robust approach to feature learning is to disentangle as many factors as possible, discarding as little information about the data as is practical.*"[25]

[25] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

Invariant Representations

- However, how to determine a priori which are the irrelevant/nuisance variations ? If the representations are destined to multiple tasks, they might have distinct relevant features !

- The most common answer to this question is to **preserve** as much as possible of the **information** in the data removing only the information that can be considered as irrelevant for all aimed tasks (e.g., the previous geometric transformations)

- Ultimately, the research postulate mostly followed by researchers is: "*the most robust approach to feature learning is to disentangle as many factors as possible, discarding as little information about the data as is practical.*"[25]

What does it mean disentangling factors of variation ?

[25] Y. Bengio et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).

Summary

Disentangled representations

- Data distributions can be usually described by the interaction of several *independent* (or conditionally independent) factors of variation.
- Between two samples, only few of them tend to change while the others stay fixed

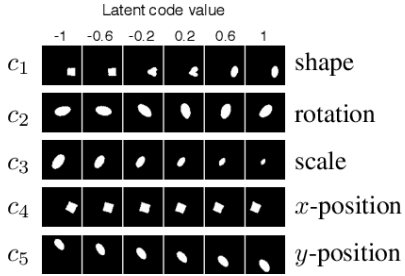


Figure: DSprite dataset where there are only 5 factors of variation (shape, rotation, scale, x-pos, ypos). Image representations have 5 latent codes (which capture the actual factors of variation). Each row shows how an image changes when traversing a single latent code.[26]

[26] Z. Lin et al. "InfoGAN-CR and ModelCentrality: Self-supervised Model ... for Disentangling GANs". In: *ICML*. 2020.

Disentangled representations

- However, representations of fully-connected networks or CNN are **not necessarily disentangled** ![27]

(a) Unit sensitive to lower round stroke.

Figure: Test images ($x \in \mathcal{T}$) that maximize the projection of the image representation $f(x)$ onto the i-th direction e_i of the last layer: $x' = \arg\max_{x \in \mathcal{T}} \langle f(x), e_i \rangle$

(a) Direction sensitive to upper straight stroke, or lower round stroke.

Figure: Test images ($x \in \mathcal{T}$) that maximize the projection of the image representation $f(x)$ onto a random direction v of the last layer: $x' = \arg\max_{x \in \mathcal{T}} \langle \phi(x), v \rangle$

- **Images within each row share similar semantic properties !** $\rightarrow$ This means that the entire representation space contains important **but entangled** semantic information. Source of variations are not disentangled among the individual units of the last layer.

[27] C. Szegedy et al. "Intriguing properties of neural networks". In: *ICLR*. 2014.

How can we encourage representations to be disentangled ?

- **Unsupervised disentanglement**: we would like to learn a disentangled representation purely from the observed data, without any form of supervision → Unfortunately, it is fundamentally impossible without inductive biases on both the models and the data ![28]
- That's why, researchers have mainly proposed two types of methods:
  - **Supervised disentanglement**: generative factors are (partly or entirely) known and we use this knowledge explicitly to guide the representation learning[29]
  - **Weakly-supervised disentanglement**: we leverage additional information (e.g., inductive biases, number of varying factors, labels) to disentangle the input data[30]
- We will discuss these methods in the next lecture !

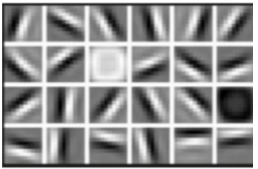
[28] F. Locatello et al. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled...". In: *ICML*. 2019.
[29] F. Locatello et al. "Disentangling Factors of Variation Using Few Labels". In: *ICLR*. 2020.
[30] F. Locatello et al. "Weakly-Supervised Disentanglement Without Compromises". In: *ICML*. 2020.

Summary

Generic Representations

- Ideally, we would like to learn a representation that is **generic** → it performs well for all possible tasks !

- Low-level features are closer to the pixel space → more useful for tasks such as image segmentation and specific (useful) to the employed imaging modality



Lines, Corners, Edges

- High-level features describe more high-level concepts → more useful for tasks such as image classification or object recognition and more specific to a task, namely if we change classes/objects they might not be so useful

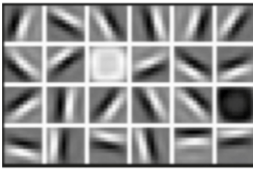
Meaningful Faces

We should train using several input modalities and/or different tasks

Generic Representations

- Ideally, we would like to learn a representation that is **generic** → it performs well for all possible tasks !

- Low-level features are closer to the pixel space → more useful for tasks such as image segmentation and specific (useful) to the employed imaging modality



Lines, Corners, Edges

- High-level features describe more high-level concepts → more useful for tasks such as image classification or object recognition and more specific to a task, namely if we change classes/objects they might not be so useful

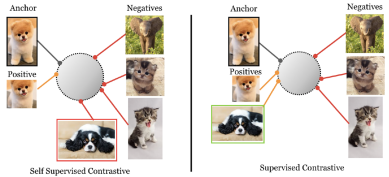
Meaningful Faces

Different learning paradigms exist to estimate a well organized and generic representation: Multi-task learning, Meta learning, Continual learning, ...

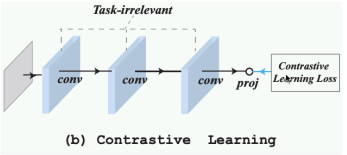
Semantically coherent Representations

- A generic and well-organized representation should also be **semantically coherent** → similar categorical concepts should be close to each other in the representation space[a]

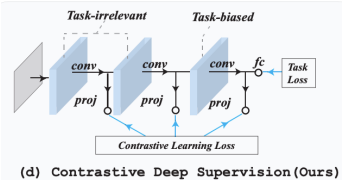
[a] P. Khosla et al. "Supervised Contrastive Learning". In: *NeurIPS*. 2020.



- **Contrastive Learning** (self-supervised strategy) uses this strategy to estimate the representation space → loss is computed using only the final layer



- **Deep (self) supervision** adds extra supervisions or self-supervision to the intermediate layers[ab]

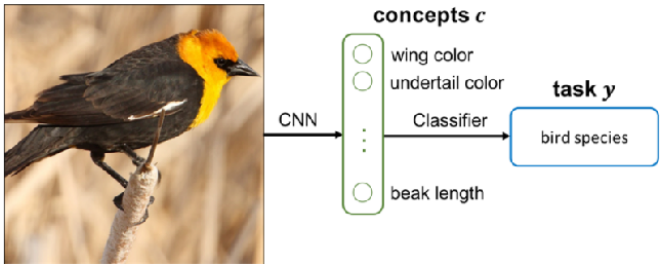


- More information in the next lecture !

[a] S. Ren et al. *DeepMIM: Deep Supervision for Masked Image Modeling*. 2023.
[b] L. Zhang et al. "Contrastive Deep Supervision". In: *ECCV*. 2022.

Interpretable Representations

- **Interpretability** is the degree to which a human can understand the reasoning behind the prediction of a model
- Very important in medical imaging. If you want to create a product you need to be able to explain "Why and How does your DL model work ?"
- And interpretable representations ? → **Concept-based models** make predictions based on human-understandable concepts (e.g.,[31,32] predicts c, then use c to predict y)



[31] P. W. Koh et al. "Concept Bottleneck Models". In: *ICML*. 2020.
[32] E. Kim et al. "Probabilistic Concept Bottleneck Models". In: *ICML*. 2023.

How can we evaluate the quality of a representation ?

- Unsupervised and self-supervised methods learn a representation without labels. To evaluate the quality of the representation, three methods are mainly used:
  - **Linear classification**: labeled train/test samples are passed through the frozen network and then linearly classified → the hypothesis is that classes should be *linearly* separable in the representation space[33]
  - **k-NN**[34]: labeled train/test samples are passed through the frozen network and then classified (non-linearly) using k-nearest neighbors (k-NN). Several values of k are tested. Best results reported.
  - **Transfer Learning**[35]: the network is fine-tuned on another labeled dataset and then evaluated.

[33] A. Kolesnikov et al. "Revisiting Self-Supervised Visual Representation Learning". In: *CVPR*. 2019.
[34] Z. Wu et al. "Unsupervised Feature Learning via Non-parametric Instance Discrimination". In: *CVPR*. 2018.
[35] P. Goyal et al. "Scaling and Benchmarking Self-Supervised Visual Representation Learning". In: *ICCV*. 2019.

Summary

Human learning Vs Machine Learning

- You have never seen (hopefully...) these animals, but you know that they don't exist.
- On the contrary, a ML algorithm would probably recognize them as animals but will struggle about the class.

Why is that ?

Human learning Vs Machine Learning

- You have never seen (hopefully...) these animals, but you know that they don't exist.
- On the contrary, a ML algorithm would probably recognize them as animals but will struggle about the class.

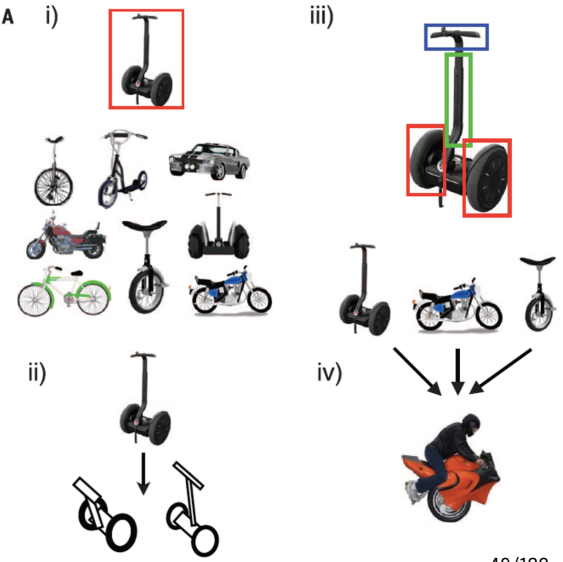
Why is that ?

- Human beings can perform new tasks without training, or with few training samples → ML algorithms need thousands or millions of examples for a precise task !

Human learning Vs Machine Learning

- People can use previously learned concepts (from previous tasks) to[a]:
  1. recognize new objects with a single example (one-shot learning)
  2. create new similar examples
  3. parsing objects into parts and relations
  4. mix parts to create new objects

Can we do the same with ML algorithms ?



[a] B. M. Lake et al. "Human-level concept learning through probabilistic program induction". In: *Science* (2015).

Different kind of learning paradigms

- There are 7 main paradigms that deal with transferring knowledge between source (training) and target (test) tasks:
  - Transfer learning
  - Domain Adaptation
  - Multi-task learning
  - Meta-Learning (Learning to Learn)
  - Continual Learning (Lifelong Learning)
  - Online learning
  - Knowledge Distillation

Notation

- Adapting the notation of[36][37], we define for the training (source) and test (target) data:

Definition of Domain

A domain $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}, p(X, Y)\}$ consists of an input feature space $\mathcal{X} = \mathbb{R}^d$, an output space $\mathcal{Y}$ (e.g., $\mathcal{Y} = \{1, .., K\}$, $\mathcal{Y} = \{0, 1\}$, $\mathcal{Y} = \mathbb{R}$), and a joint distribution over the input and output space $p(X, Y)$, where X is a r.v. taking values $x \in \mathcal{X}$ and Y is a r.v. taking values $y \in \mathcal{Y}$

Definition of Task

A task $\mathcal{T} = f(\cdot)$ consists of the predictive function $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ which is the conditional distribution $p(Y|X)$. It thus depends on the output space $\mathcal{Y}$.

[36] S. J. Pan et al. "A Survey on Transfer Learning". In: *IEEE Trans. Knowl. Data Eng.* (2010).

[37] S. Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* (2010).

Different kind of learning paradigms

- The main differences between the 7 paradigms are:
  - **Number of source and target domains** $\rightarrow$ Example: we might have several source domains and one target domain

Different kind of learning paradigms

- The main differences between the 7 paradigms are:
  - **Number of source and target domains** $\rightarrow$ Example: we might have several source domains and one target domain
  - **Source domain** $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S, p_S(X, Y)\}$ **and target domains** $\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{Y}_T, p_T(X, Y)\}$ **can be different**. It means that one or more components are different.
    - $\mathcal{X}_S \neq \mathcal{X}_T \rightarrow$ different input feature space
    - $\mathcal{Y}_S \neq \mathcal{Y}_T \rightarrow$ different output space and thus different tasks $\mathcal{T}_S \neq \mathcal{T}_T$
    - $p_S(X, Y) \neq p_T(X, Y) \rightarrow$ differences in the components $p(X)$, $p(Y)$, $p(X|Y)$, $p(Y|X)$. At least one component of T needs to be related (e.g., equal, smooth changes) to the same component of S. **The two domains must be related**, need to transfer something[38].

[38] D. Shai Ben et al. "Impossibility Theorems for Domain Adaptation". In: *AISTATS*. 2010.

Different kind of learning paradigms

- The main differences between the 7 paradigms are:
  - **Number of source and target domains** $\rightarrow$ Example: we might have several source domains and one target domain
  - **Source domain** $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S, p_S(X, Y)\}$ **and target domains** $\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{Y}_T, p_T(X, Y)\}$ **can be different**. It means that one or more components are different.
    - $\mathcal{X}_S \neq \mathcal{X}_T \rightarrow$ different input feature space
    - $\mathcal{Y}_S \neq \mathcal{Y}_T \rightarrow$ different output space and thus different tasks $\mathcal{T}_S \neq \mathcal{T}_T$
    - $p_S(X, Y) \neq p_T(X, Y) \rightarrow$ differences in the components $p(X)$, $p(Y)$, $p(X|Y)$, $p(Y|X)$. At least one component of T needs to be related (e.g., equal, smooth changes) to the same component of S. **The two domains must be related**, need to transfer something[38].
  - **Source/Target data are annotated or not**

[38] D. Shai Ben et al. "Impossibility Theorems for Domain Adaptation". In: *AISTATS*. 2010.

Different kind of learning paradigms

- The main differences between the 7 paradigms are:
  - **Number of source and target domains** $\rightarrow$ Example: we might have several source domains and one target domain
  - **Source domain** $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S, p_S(X, Y)\}$ **and target domains** $\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{Y}_T, p_T(X, Y)\}$ **can be different**. It means that one or more components are different.
    - $\mathcal{X}_S \neq \mathcal{X}_T \rightarrow$ different input feature space
    - $\mathcal{Y}_S \neq \mathcal{Y}_T \rightarrow$ different output space and thus different tasks $\mathcal{T}_S \neq \mathcal{T}_T$
    - $p_S(X, Y) \neq p_T(X, Y) \rightarrow$ differences in the components $p(X)$, $p(Y)$, $p(X|Y)$, $p(Y|X)$. At least one component of T needs to be related (e.g., equal, smooth changes) to the same component of S. **The two domains must be related**, need to transfer something[38].
  - **Source/Target data are annotated or not**
  - Source/Target tasks are learned sequentially or simultaneously
  - Data from source domain comes gradually over time

[38] D. Shai Ben et al. "Impossibility Theorems for Domain Adaptation". In: *AISTATS*. 2010.

Summary

Definition of Transfer Learning

Given **one** source domain $\mathcal{D}_S$ and **one** target domain $\mathcal{D}_T$ entirely available at training/test time (no online setting), where $\mathcal{D}_S \neq \mathcal{D}_T$ **and/or** $\mathcal{T}_S \neq \mathcal{T}_T$, transfer learning aims to improve the learning of the target task $\mathcal{T}_T$ using the knowledge of $\mathcal{D}_S$ and $\mathcal{T}_S$

- Given this definition, we can divide Transfer Learning into two sub-categories[39][40]:
  1. **Inductive transfer learning**
  2. **Transductive Transfer learning** (domain adaptation)

[39] F. Zhuang et al. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE*. 2020.
[40] S. J. Pan et al. "A Survey on Transfer Learning". In: *IEEE Trans. Knowl. Data Eng.* (2010).

Transfer Learning

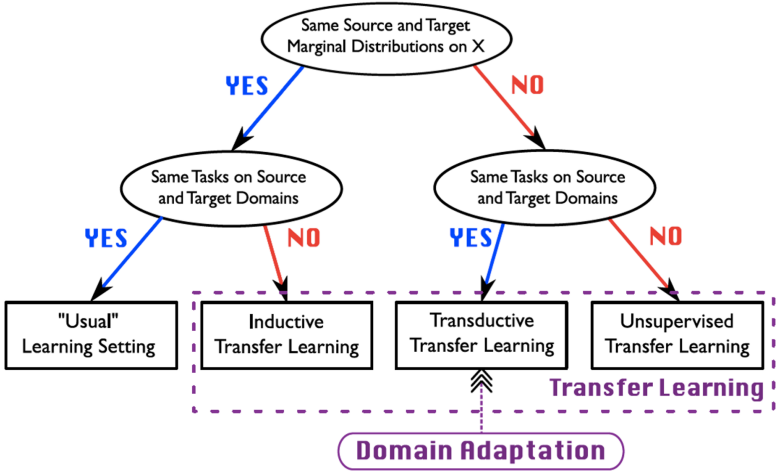


Figure: Fig. from[41]

[41] I. Redko et al. *Advances in Domain Adaptation Theory*. 2019.

- If we suppose that $p_S(X, Y) \neq p_T(X, Y)$, then, from Bayes' rule, we know that:

$$p(X, Y) = p(Y|X)p(X) \quad \text{and} \quad p(X, Y) = p(X|Y)p(Y)$$

- The four components are thus linked and one can study either the left or the right components (or both).
- Remember, that T and S should be related. Otherwise, training S data are of no use for predicting test T data.

- **Inductive transfer learning** ($\mathcal{X}_S = \mathcal{X}_T$ and $p_S(X) = p_T(X)$)
  - $\mathcal{Y}_S \neq \mathcal{Y}_T$: The two domains have different labels y
  - $\mathcal{Y}_S = \mathcal{Y}_T$ and $p_S(Y|X) \neq p_T(Y|X)$: S and T have the same labels but the conditional distributions are different (e.g., in regression, given the same feature vector, the predicted values are different)
- **Transductive Transfer learning** (domain adaptation).

- **Inductive transfer learning** ($\mathcal{X}_S = \mathcal{X}_T$ and $p_S(X) = p_T(X)$)
- **Transductive Transfer learning** (domain adaptation).

We assume $\mathcal{Y}_S = \mathcal{Y}_T$ and $\mathcal{X}_S = \mathcal{X}_T$ (not necessarily) and

- *covariate shift*: $p_S(Y|X) = p_T(Y|X)$ and $p_S(X) \neq p_T(X)$
- *concept shift*: $p_S(Y|X) \neq p_T(Y|X)$ and $p_S(X) = p_T(X)$
- *target shift*: $p_S(X|Y) = p_T(X|Y)$ and $p_S(Y) \neq p_T(Y)$
- *conditional shift*: $p_S(Y) = p_T(Y)$ and $p_S(X|Y) \neq p_T(X|Y)$
- *generalized target shift*: $p_S(X) \neq p_T(X)$, $p_S(Y|X) \neq p_T(Y|X)$, $p_S(Y) \neq p_T(Y)$ and $p_S(X|Y) \neq p_T(X|Y) \rightarrow$ need to use prior/inductive biases/constraints[42]
- *model shift*: $p_S(X) \neq p_T(X)$, $p_S(Y) \neq p_T(Y)$ and $p_S(Y|X) \neq p_T(Y|X)$[43]

[42] K. Zhang et al. "Domain Adaptation under Target and Conditional Shift". In: *ICML*. 2013.

[43] X. Wang et al. "Flexible Transfer Learning under Support and Model Shift". In: *NIPS*. 2014.

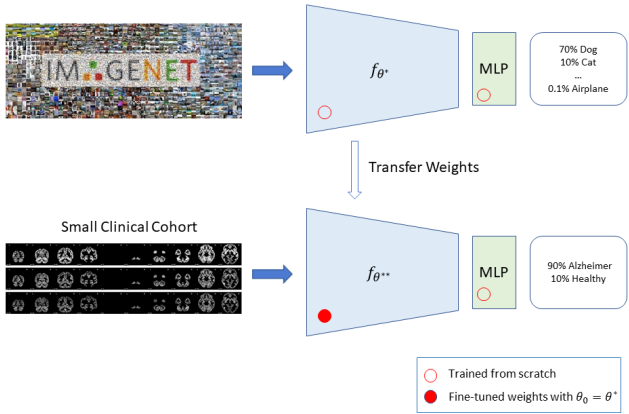
Transfer Learning

- Then, we can re-subdivide the two categories by considering whether **annotations** are available in the source S and/or target T domains

1. **Inductive transfer learning**
   - 1.1 *Supervised* $\rightarrow$ Large labeled source dataset and small labeled target dataset
   - 1.2 *Self-Supervised* $\rightarrow$ Large unlabeled source dataset and small labeled target dataset
   - 1.3 *Semi-Supervised* $\rightarrow$ Large unlabeled source dataset with few labeled source samples and small labeled target dataset
2. **Transductive Transfer learning** (domain adaptation)
   - 2.1 *Supervised* $\rightarrow$ Labeled source dataset and labeled target dataset
   - 2.2 *Unsupervised* $\rightarrow$ Labeled source dataset and unlabeled target dataset
   - 2.3 *Semi-supervised* $\rightarrow$ Labeled source dataset and unlabeled target dataset with few labeled target samples
   - 2.4 *Few-shot* $\rightarrow$ Labeled source dataset and few labeled target samples

Inductive Transfer Learning

- **Supervised transfer learning** from ImageNet is very common.[44][45][46]
- First train a network on Imagenet (source domain) and then transfer it to the target domain.
- Parameters θ are thus initialized with the ones from Imagenet instead than randomly.
- Then, either the last classification layer or the entire network are fine-tuned (small re-training).



[44] J. Donahue et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *ICML*. 2014.

[45] J. Yosinski et al. "How transferable are features in deep neural networks?" In: *NIPS*. 2014.

[46] K. Simonyan et al. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR*. 2015.

Domain gap/shift

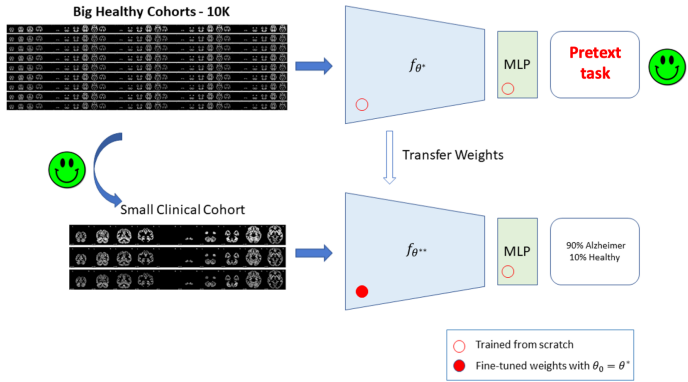
- The main assumption behind fine-tuning is that the features/representations learned from the source task are useful for the target task
- This kind of method fails when[47]:
  - Source and target domains are very **different** (not related)
  - The target task has very **few (or no)** labeled training data
- When $p_S(X) \neq p_T(X)$, we say that there is a **Domain gap or shift** between source and target domains $\rightarrow$ e.g., two different MRI machines/protocols between S and T but same modality, or two different imaging modalities
- To reduce the domain gap, one can use a different strategy...

[47] J. Yosinski et al. "How transferable are features in deep neural networks?" In: *NIPS*. 2014.

Inductive Transfer Learning

- **Self-supervised transfer learning** is a recent hot-topic[48].

- First pre-train a network on a large unlabeled source dataset leveraging a pretext task
- Then, transfer it to the target domain.
- Source and target domains are usually similar and thus highly related ! Small domain gap/shift.



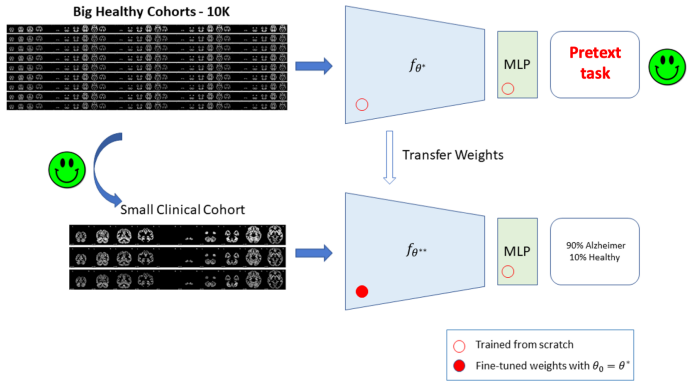
[48] T. Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*. 2020.

Inductive Transfer Learning

- **Self-supervised transfer learning** is a recent hot-topic[48].

- First pre-train a network on a large unlabeled source dataset leveraging a pretext task
- Then, transfer it to the target domain.
- Source and target domains are usually similar and thus highly related ! Small domain gap/shift.

→ Next lecture !



[48] T. Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*. 2020.

Unsupervised Domain Adaptation

- In the **unsupervised domain adaptation** setting, we assume that:
  - we only have labeled data in the source domain S but not in the target domain T
  - $\mathcal{Y}_S = \mathcal{Y}_T \rightarrow$ same labels (output space)
  - $\mathcal{X}_S = \mathcal{X}_T \rightarrow$ same features (input space)

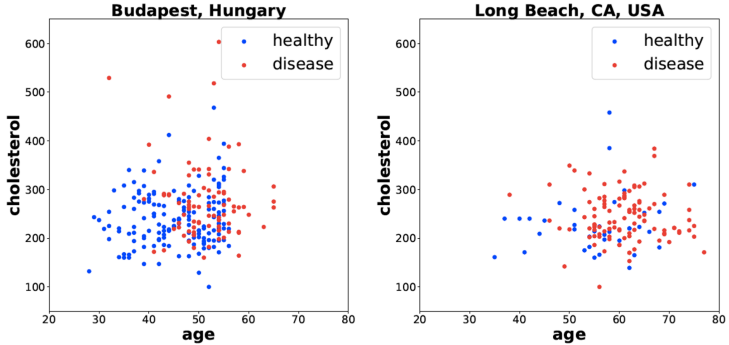


Figure: DA example about patients diagnosed with heart disease based on their age and cholesterol.(Left) source samples. (Right): target samples. Fig. from[49]

Unsupervised Domain Adaptation

- In the **unsupervised domain adaptation** setting, we assume that:
  - we only have labeled data in the source domain S but not in the target domain T
  - $\mathcal{Y}_S = \mathcal{Y}_T \rightarrow$ same labels (output space)
  - $\mathcal{X}_S = \mathcal{X}_T \rightarrow$ same features (input space)

- Considering a binary classification task (i.e., $f \in \mathcal{F}$), we define the error (or risk) on the source domain as:

$$e_S(f) = E_{(x,y)\sim p_S(X,Y)}[L(f(x), y)] = E_{(x,y)\sim p_S(X,Y)}[I(f(x) \neq y)] = Pr_{(x,y)\sim p_S(X,Y)}(f(x) \neq y)$$

and the empirical error as $\hat{e}_S(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)$. Similarly, we use $e_T(f)$ and $\hat{e}_T(f)$ for the target domain.

Question: under what conditions a model with a small error in the training set of the source domain $\hat{f}_S$ will produce a small error also in the (full) target domain $e_T(\hat{f}_S)$, where small means comparable to the best model for the target domain $e_T(\tilde{f}_T)$?

- In[49] Th.2, it has been shown that, if T has no labeled data ($\alpha = 0, \beta = 0$), we obtain:

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n) \tag{4}$$

- which holds with probability at least $1 - \delta$, for $\delta \in \{0, 1\}$

[49] J. Blitzer et al. "Learning Bounds for Domain Adaptation". In: *NIPS*. 2007.

Question: under what conditions a model with a small error in the training set of the source domain $\hat{f}_S$ will produce a small error also in the (full) target domain $e_T(\hat{f}_S)$, where small means comparable to the best model for the target domain $e_T(\tilde{f}_T)$?

- In[49] Th.2, it has been shown that, if T has no labeled data ($\alpha = 0, \beta = 0$), we obtain:

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n) \tag{4}$$

- which holds with probability at least $1 - \delta$, for $\delta \in \{0, 1\}$
- On the left, we have the cross-domain generalization error in the T domain, where :
  - $\hat{f}_S = \arg\min_{f \in \mathcal{F}} \hat{e}_S(f) \rightarrow$ best *empirical* model in $\mathcal{F}$ trained on n samples from S
  - $\tilde{f}_T = \arg\min_{f \in \mathcal{F}} e_T(f) \rightarrow$ best *theoretical* model in $\mathcal{F}$ for the T domain

[49] J. Blitzer et al. "Learning Bounds for Domain Adaptation". In: *NIPS*. 2007.

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n)$$

- $\lambda = e_S(f*) + e_T(f*)$ with $f* = \arg\min_{f \in \mathcal{F}} e_S(f) + e_T(f) \rightarrow$ the combined error of the ideal joint model (in $\mathcal{F}$) for both (full) domains.
- This shows the best performance that we can long for, if we had an infinite number of training/test samples in both domains.
- It also embodies the notion of *adaptability*. If $f*$ performs poorly, we cannot expect to learn a good target classifier by minimizing the source error.[50] On the other hand, if $\lambda = 0$, then the generalization error mostly depends on...

[50] S. Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* (2010).

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n)$$

- ... $\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T)$ the *empirical symmetric difference hypothesis* divergence (called $\mathcal{H}\Delta\mathcal{H}-$distance in[50]) which measures the distance between two unlabeled datasets $\mathcal{U}_S$, $\mathcal{U}_T$ of equal size n' drawn from $p_S(X)$ and $p_T(X)$, respectively.
- It approximates the theoretical $\mathcal{F}\Delta\mathcal{F}-$distance which **measures the discrepancy between** $p_S(X)$ and $p_T(X)$. It is used with binary classification problems and it takes two classifiers, looks at to what extent they disagree with each other on both domains and returns the value of the largest (*supremum*) difference[51].

$$\hat{d}_{\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) = 2 \sup_{f,f' \in \mathcal{F}} |\Pr_{\mathcal{U}_S}[f \neq f'] - \Pr_{\mathcal{U}_T}[f \neq f']| \quad \text{with } \Pr_{\mathcal{U}}[f \neq f'] = \frac{1}{|\mathcal{U}|} \sum_{x \in \mathcal{U}} I(f(x) \neq f'(x))$$

[50] S. Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* (2010).

[51] W. M. Kouw et al. *An introduction to domain adaptation and transfer learning*. Tech. rep. 2019.

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n)$$

- $\mathcal{C}(\mathcal{F}, \delta, n) \approx O\left(\sqrt{\frac{d\log(n+1)-\log(\delta)}{n}} + \sqrt{\frac{d\log(n')-\log(\delta)}{n'}}\right)$ describes the complexity of the model family $\mathcal{F}$ and it depends on:
  - $d \rightarrow$ the Vapnik-Chervonenkis (VC) dimension measuring the expressive power of $\mathcal{F}$
  - $n \rightarrow$ number of training samples in S used to estimate $\hat{f}_S$
  - $n' \rightarrow$ number of samples of $\mathcal{U}_S$ and $\mathcal{U}_T$
  - $\delta \rightarrow$ 1 minus the probability that this bound holds
- This shows that we would like to have a small d (not really DL...), a small δ and a big n and n'

$$e_T(\hat{f}_S) - e_T(\tilde{f}_T) \leq \lambda + \frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) + \mathcal{C}(\mathcal{F}, \delta, n)$$

- This bound[50] is the foundation of many works in Unsupervised Domain Adaptation and it states that:
  - $\lambda \to 0$: the model's family $\mathcal{F}$ needs to be **rich enough** to have a theoretical model $f*$ that can minimize both source S and target T errors
  - $\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{U}_S, \mathcal{U}_T) \to 0$: the two input spaces $p_S(X)$ and $p_T(X)$ must be similar (**small domain gap**), based on the chosen model family $\mathcal{F}$
  - $\mathcal{C}(\mathcal{F}, \delta, n) \to 0$: the VD dimension d should not be too big (trade-off with $\lambda \to 0$) and we should use a large training dataset (n) in the source S domain

[50] S. Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* (2010).

- The previous bound is rather generic. Further works have tried to find more tighter bounds by adding *constraints* and *assumptions*
- We are interested in minimizing the target error (i.e., risk) using data from the source distribution. How ca we relate $p_S(X, Y)$ and $e_T(f)$?

$$
\begin{aligned}
e_T(f) &= \sum_{y \in \mathcal{Y}} \int_{x \in \mathcal{X}} L(f(x), y) p_T(x, y) dx = \sum_{y \in \mathcal{Y}} \int_{x \in \mathcal{X}} L(f(x), y) \frac{p_T(x, y)}{p_S(x, y)} p_S(x, y) dx \\
&= \mathbb{E}_{(x,y) \sim p_S(x,y)} L(f(x), y) \frac{p_T(x, y)}{p_S(x, y)} \approx \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) \frac{p_T(x_i, y_i)}{p_S(x_i, y_i)}
\end{aligned}
\tag{5}
$$

- where samples (x_i, y_i) are drawn from the source S domain $p_S(x, y)$! Here, $p_T(x_i, y_i)$ refers to the probability of those samples under the target distribution.[51]

[51] W. M. Kouw et al. *An introduction to domain adaptation and transfer learning*. Tech. rep. 2019.

- The empirical generalization error on the target T domain is thus:

$$\hat{e}_T(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{p_T(x_i, y_i)}{p_S(x_i, y_i)} \tag{6}$$

- As previously seen, joint distributions can be decomposed in two ways. We can thus make assumptions on these four components:

$$p(x, y) = p(x|y)p(y) \quad \text{or} \quad p(x, y) = p(y|x)p(x)$$

- The three main studied data shifts are:
  - *covariate shift*: $p_S(Y|X) = p_T(Y|X)$ and $p_S(X) \neq p_T(X)$
  - *target (prior) shift*: $p_S(X|Y) = p_T(X|Y)$ and $p_S(Y) \neq p_T(Y)$
  - *concept shift*: $p_S(Y|X) \neq p_T(Y|X)$ and $p_S(X) = p_T(X)$

Domain Adaptation - Covariate Shift

- The covariate shift assumptions, $p_S(Y|X) = p_T(Y|X)$ and $p_S(X) \neq p_T(X)$, is probably the most studied case
- This is mainly due to the **sampling selection bias** : samples in S are not drawn randomly. Some samples are more likely to be included or a subset of samples is excluded from the sampling (i.e., missing samples). Target samples are instead assumed to be unbiased.

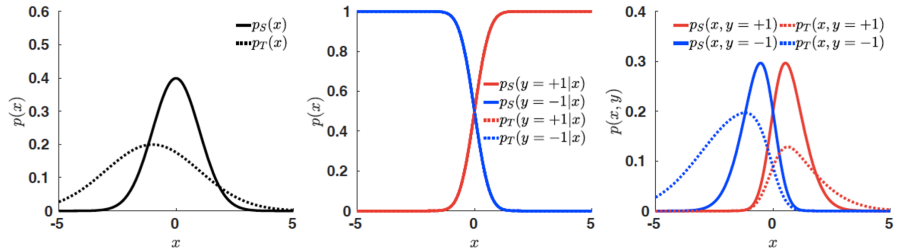


Figure: Image from[52]

[52] W. M. Kouw et al. *An introduction to domain adaptation and transfer learning.* Tech. rep. 2019.

Domain Adaptation - Covariate Shift

- Under the covariate shift assumptions, $p_S(Y|X) = p_T(Y|X)$ and $p_S(X) \neq p_T(X)$, the empirical generalization target error can be rewritten as:

$$\hat{e}_T(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{p_T(x_i, y_i)}{p_S(x_i, y_i)} = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{\cancel{p_T(y_i|x_i)}p_T(x_i)}{\cancel{p_S(y_i|x_i)}p_S(x_i)} = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\underbrace{\frac{p_T(x_i)}{p_S(x_i)}}_{w(x_i)}$$

- The importance weights $w(x) = \frac{p_T(x)}{p_S(x)}$[53] indicates how the probability of a (biased) source sample should be *corrected* to reflect the (unbiased) probability under the target distribution

- The weights influence a classification model by increasing the loss for certain samples and decreasing the loss for other

[53] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

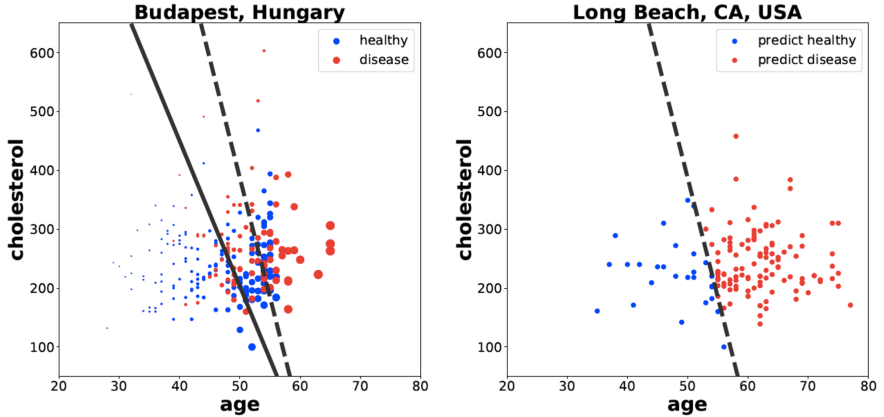


Figure: Effect of importance-weighting. (Left) Weighted source samples (large dot = large weight) with $w(x)$. (Right) Target samples. Black line: original samples classifier. Dashed line: importance-weighted classifier. Fig. from[54]

[54] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

- What about the generalization error of a given classifier f? The difference between the true target T error $e_T(f)$ and the empirical weighted source WS error $\hat{e}_{WS}(f)$ is[55]:

$$e_T(f) - \hat{e}_{WS}(f) \leq \sqrt{d_2(p_T(X)|p_S(X))}\mathcal{C}(c, n, \delta)$$

- where $d_2(p_T(X)|p_S(X)) = \sum_{x \in X} \frac{p_T^2(x)}{p_S(x)} = \mathbb{E}_{x \sim p_S(x)}[w(x)^2]$ is the Rényi divergence between the two marginal distributions and it is directly related to $w(x)$...
- ... and $\mathcal{C}(c, n, \delta) \approx O(-p \log(pn) n^{-3/8})$ where $p \geq 0$ is the pseudo-dimension of the model space $\mathcal{F}$ (a generalization of VC-dimension)
- Similarly to before, this bound tells us that the model family should not be too complex ($p \to 0$), the number of samples big ($n \to \infty$) and the two domains not so different ($d_2(p_T(X)|p_S(X)) \to 0$)

[55] C. Cortes et al. "Learning Bounds for Importance Weighting". In: *NIPS*. 2010.

Domain Adaptation - Covariate Shift

$$e_T(f) - \hat{e}_{WS}(f) \leq \sqrt{d_2(p_T(X)|p_S(X))}\mathcal{C}(c, n, \delta)$$

- Since $d_2(p_T(X)|p_S(X)) = \mathbb{E}_{x\sim p_S(x)}[w(x)^2]$, we know that we should have $\mathbb{E}_{x\sim p_S(x)}[w(x)^2] < \infty$ and thus that w should not be too big
- Based on this bound, one can also show that if we use the *correct* model family $\mathcal{F}$ and we have *enough* samples n, an unweighted model will also work ![56]
- However, we usually don't know the correct $\mathcal{F}$ and we don't have an infinit number of samples n, so **how do we choose the appropriate weights** $w(x)$ **?**

[56] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

To compute $w(x)$ one can[57]:

- **Estimate the distributions** $p_S(X)$ and $p_T(X)$
  - *parametrically*: both pdf follow a parametric function, like: $\hat{w}(x_i) = \frac{\mathcal{N}(x_i|\hat{\mu}_T, \hat{\Sigma}_T)}{\mathcal{N}(x_i|\hat{\mu}_S, \hat{\Sigma}_S)}$
  - *non-parametrically*: using kernel density estimators (KDE): $\hat{w}(x_i) = \frac{m^{-1}\sum_{j=1}^{m} k_{\sigma_T}(x_i - x_j)}{n^{-1}\sum_{t=1}^{n} k_{\sigma_S}(x_i - x_t)}$, where k is a kernel, σ is the hyper-parameter (one for S, one for T), m and n are the total samples in T and S respectively.[58]
- **Direct estimate of** w **via optimization**
- **Direct estimate of** w **without optimization**

[57] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

[58] H. Shimodaira. "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *JSPI* (2000).

Domain Adaptation - Covariate Shift

To compute $w(x)$ one can[57]:

- **Estimate the distributions** $p_S(X)$ and $p_T(X)$
- **Direct estimate of** w **via optimization** by minimizing a discrepancy measure D between $p_T(X)$ and $w(x)p_S(x)$ and fulfilling two constraints about the non-negative values of w and $w(x)p_S(x)$ should be a valid p.d.f.:

$$\hat{w} = \arg\min_w D(w, p_S(X), p_T(X)) \quad s.t. w \geq 0 \text{ and } \int_X p_T(x)dx = \int_X w(x)p_S(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} w(x_i) = 1$$

where D can be the Maximum Mean Discrepancy (MMD) or the Kullback-Leibler divergence

- **Direct estimate of** w **without optimization**

[57] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

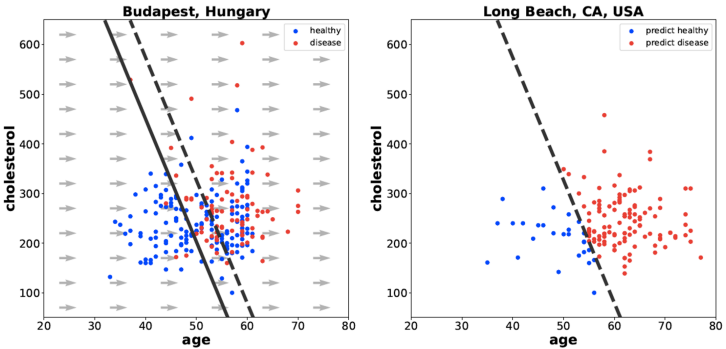
To compute $w(x)$ one can[57]:

- **Estimate the distributions** $p_S(X)$ and $p_T(X)$
- **Direct estimate of** w **via optimization**
- **Direct estimate of** w **without optimization**
  - use logistic regression to discriminate between S and T and then use the estimated posterior probabilities as weights
  - Divide the feature space into regions (Voronoi cells) and then use Nearest-Neighbour algorithm
  - jointly optimize the weight and the final classifier

[57] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

- In addition to reweighting methods, there are other methods that look for transformations t to match the two marginal distributions.
- one can look for a transformation t such that $p_S(t(X)) \approx p_T(X)$[58] $\rightarrow$ However, domains can be very different and high-dimensional. A too complex t might bring to overfitting.



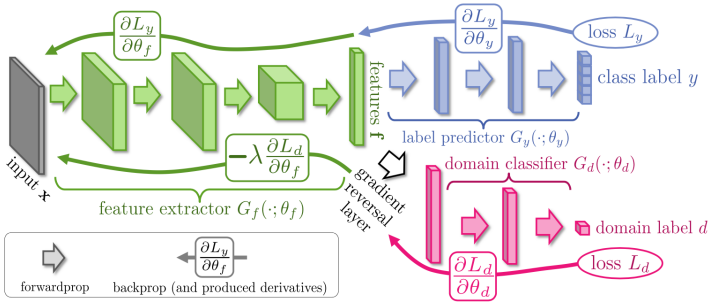
[58] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Covariate Shift

- A solution could be mapping the data onto common subspaces and then apply simple, linear transformations to align them[59]. This should thus reduce the discrepancy between the marginals:
  - First compute PCA on S and T keeping the first d eigenvectors: U_S and U_T
  - Compute (linear) transformation matrix M that aligns U_S to U_T: $M* = \arg\min_M ||U_S M - U_T||_F^2$, which results $M* = U_S^T U_T$.
  - Project source data X_S onto U_S ($X_S U_S$) and then transform it: $X_S U_S M*$
  - Project target data X_T onto U_T : $X_T U_T$
  - Train Classifier on modified source data ($X_S U_S M*$) to predict projected target data ($X_T U_T$)
- **Limitation**:here we use a simple linear mapping (i.e., PCA) to estimate a new representation. Why not using DL to estimate a domain-invariant representation (same for S and T)?

[59] B. Fernando et al. "Unsupervised Visual Domain Adaptation Using Subspace Alignment". In: *ICCV*. 2013.

Domain Adaptation - DANN method



- In[60], authors combine domain adaptation and deep feature learning using:
  - feature extractor (green): learns a relevant mapping $f = G_f(x;\theta_f)$
  - label predictor (blue): predicts the label $\hat{y} = G_y(f(x);\theta_y)$
  - domain classifier (pink): predicts the domain $\hat{d} = G_d(f(x);\theta_d)$

[60] Y. Ganin et al. "Unsupervised Domain Adaptation by Backpropagation". In: *ICML*. 2015.

Domain Adaptation - DANN method

- During training, we want to correctly predict the class $y \in \mathcal{Y}$ of the labeled S samples:

$$\arg\min_{\theta_f, \theta_y} E_S(\theta_f, \theta_y) = \sum_{(x_i, y_i) \in S} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \tag{7}$$

- ... at the same time, we want to make the features $f(x)$ domain-invariant, that is: $S(f) \approx T(f)$, where $S(f) = \{f(x) | x \in S\}$ and $T(f) = \{f(x) | x \in T\}$.

- The two distributions $S(f)$ and $T(f)$ are high-dimensional, complex and changing during training. How can we compute their dissimilarity ?

Domain Adaptation - DANN method

- During training, we want to correctly predict the class $y \in \mathcal{Y}$ of the labeled S samples:

$$\arg\min_{\theta_f, \theta_y} E_S(\theta_f, \theta_y) = \sum_{(x_i, y_i) \in S} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \tag{7}$$

- ... at the same time, we want to make the features $f(x)$ domain-invariant, that is: $S(f) \approx T(f)$, where $S(f) = \{f(x) | x \in S\}$ and $T(f) = \{f(x) | x \in T\}$.

- The two distributions $S(f)$ and $T(f)$ are high-dimensional, complex and changing during training. How can we compute their dissimilarity ? $\rightarrow$ using a (trained) **domain classifier** !

- If the domain classifier can not distinguish between the two domains (probability = 0.5 $\forall\, x$), it means that the features f are **domain invariant** !

Domain Adaptation - DANN method

- During training, we want to correctly predict the class $y \in \mathcal{Y}$ of the labeled S samples:

$$\arg\min_{\theta_f,\theta_y} E_S(\theta_f,\theta_y) = \sum_{(x_i,y_i)\in S} L_y(G_y(G_f(x_i;\theta_f);\theta_y),y_i) \tag{7}$$

- ... at the same time, we want to make the features $f(x)$ domain-invariant, that is: $S(f) \approx T(f)$, where $S(f) = \{f(x)|x \in S\}$ and $T(f) = \{f(x)|x \in T\}$.

- The two distributions $S(f)$ and $T(f)$ are high-dimensional, complex and changing during training. How can we compute their dissimilarity ? $\rightarrow$ using a (trained) **domain classifier** !

- If the domain classifier can not distinguish between the two domains (probability = 0.5 $\forall\, x$), it means that the features f are **domain invariant** !

- **Problem**: we don't have a trained domain classifier...

- **Idea**: Train domain classifier G_d to distinguish between S ($d = 0$) and T ($d = 1$) and **at the same time** train feature extractor G_f to fool the discriminator $\rightarrow$ Adversarial, two-player minimax training as in GAN[61] !

$$\arg\max_{\theta_f} \arg\min_{\theta_d} E_D(\theta_f, \theta_d) = \sum_{x_j \in S,T} L_d(G_d(G_f(x_j; \theta_f); \theta_d), d_j)$$

- The final optimization problem becomes:

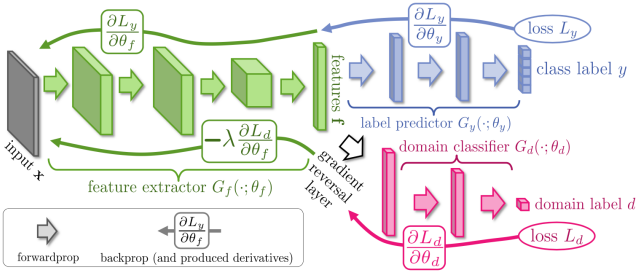
$$\begin{aligned} (\hat{\theta}_f, \hat{\theta}_y) &= \arg\min_{\theta_f, \theta_y} E_S(\theta_f, \theta_y) - \lambda E_D(\theta_f, \theta_d) \\ \hat{\theta}_d &= \arg\min_{\theta_d} \lambda E_D(\theta_f, \theta_d) \end{aligned} \tag{8}$$

- where $\lambda > 0$ is a hyper-parameter that balances the two losses

[61] I. Goodfellow et al. "Generative Adversarial Nets". In: *NIPS*. 2014.

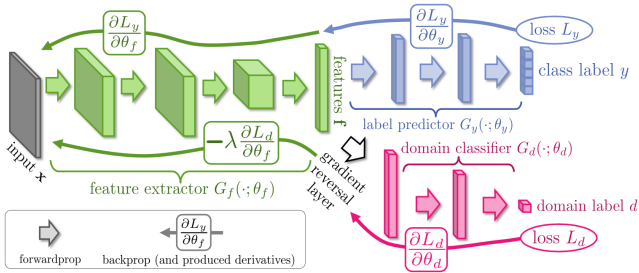
Domain Adaptation - DANN method

- Using a gradient descent scheme, the updates of all parameters are straightforward. For θ_f, we obtain: $\theta_f \leftarrow \theta_f - \mu(\frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f})$, where $\mu > 0$ is the learning rate
- To implement such update, authors introduce the gradient reversal layer (GRL):
  - *During forward*: GRL acts as an identity transform
  - *During back-propagation*: GRL takes the gradient from the subsequent level, multiplies it by $-\lambda$ and passes to the preceding layer



Domain Adaptation - DANN method

- At test time, the network can predict labels y from both domains.
- A more in-depth theoretical analysis and in particular a link between the discriminator G_d and the $\mathcal{F}\Delta\mathcal{F}$-distance can be found in[62]



[62] Y. Ganin et al. "Domain-Adversarial Training of Neural Networks". In: *JMLR* (2017).

Domain Adaptation - MDD method

- Another interesting, and currently still SOTA method, is based on a new divergence measure between domains: **Margin Disparity Discrepancy (MDD)**[63]
- Similar generalization bounds as in[64] but based on margin theory:

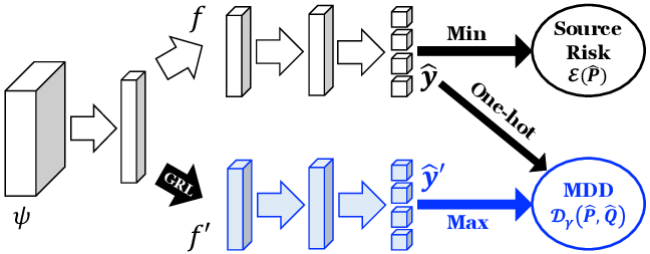
$$e_T(f) \leq \hat{e}_S(f) + d_f(\hat{S}, \hat{T}) + \lambda + C$$

- Expected error on target domain $e_T(f)$ is bounded by the sum of the empirical margin error on the source domain $\hat{e}_S(f)$, empirical MDD $d_f(\hat{S}, \hat{T})$ between domains, ideal margin error λ and complexity terms C.
- $\lambda \to 0$ if $\mathcal{F}$ is rich enough, thus authors propose to look for an $f \in \mathcal{F}$ that minimize $\hat{e}_S(f)$ and the MDD $d_f(\hat{S}, \hat{T})$

[63] Y. Zhang et al. "Bridging Theory and Algorithm for Domain Adaptation". In: *ICML*. 2019.

[64] S. Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* (2010).

Domain Adaptation - MDD method



- Similarly to DANN, authors propose to use a feature extractor ψ and an auxiliary classifier $f' \in \mathcal{F}$ optimized in an adversarial way:

$$\min_{f,\psi} \hat{e}_S(f) + \eta d_f(\hat{S}, \hat{T}) \quad \text{and} \quad \max_{f'} d_f(\hat{S}, \hat{T})$$

- where $\eta > 0$ and $\hat{e}_S(f)$ and $d_f(\hat{S}, \hat{T})$ are computed using cross-entropy losses.[65]

[65] Y. Zhang et al. "Bridging Theory and Algorithm for Domain Adaptation". In: *ICML*. 2019.

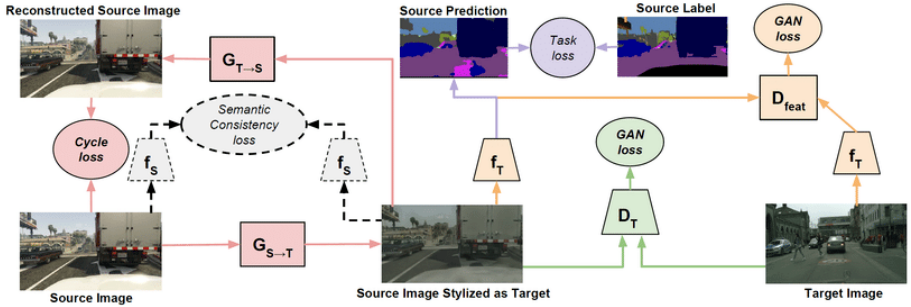
Domain Adaptation - domain-invariant

- Both DANN and MDD methods (as well as all domain-invariant algorithms) try to minimize a discrepancy between features extracted from S and T using a *common* feature extractor
- **Limitations**
  - there is no guarantee that matching $p_S(X)$ with $p_T(X)$ will also imply $p_S(Y|X) \approx p_T(Y|X)$. In other words, aligning marginal distributions does not enforce semantic consistency. Source features of class A may be aligned to Target features of class $B \rightarrow$ interesting solution proposed in[66]
  - feature-level alignment (i.e., at the representation level) can ignore low-level details $\rightarrow$ need to also work at the pixel-level, translating source data S to the "style" of the target domain T[67] Next method

[66] K. Saito et al. "Maximum Classifier Discrepancy for Unsupervised Domain Adaptation". In: *CVPR*. 2018.

[67] J. Hoffman et al. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *ICML*. 2018.

Domain Adaptation - CyCADA method



- **Pixel-level adaptation**: pink and green. Cycle-consistency as in[68].
- **Feature-level adaptation**: orange and purple. GAN loss[69].

[68] J.-Y. Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *ICCV*. 2017.
[69] I. Goodfellow et al. "Generative Adversarial Nets". In: *NIPS*. 2014.

- **Pixel-level adaptation**:
  - Mappings $G_{S\to T}$ and $G_{T\to S}$ are trained to produce realistic samples that fool adversarial discriminators D_T and D_S respectively (e.g., $G_{S\to T}$ is trained so that $D_T(G_{S\to T}(x_S)) = 1$). The discriminators D_S and D_T are also trained on real data so that they output 1 when their inputs come from S and T respectively.

$$\min_{G_{S\to T}, G_{T\to S}} \max_{D_T, D_S} \mathcal{L}_{GAN} = \mathbb{E}_{x_T\in T} \log D_T(x_T) + \mathbb{E}_{x_S\in S} \log(1 - D_T(G_{S\to T}(x_S))) + \mathbb{E}_{x_S\in S} \log D_S(x_S) + \mathbb{E}_{x_T\in T} \log(1 - D_S(G_{T\to S}(x_T)))$$

  - How to guarantee that $G_{S\to T}(x_S)$ preserves structure and content of x_S ? $\to$ **cycle-consistency constraint** based on a $L1-$penalty (sharper reconstruction)

$$\min_{G_{S\to T}, G_{T\to S}} \mathcal{L}_{cyc} = \mathbb{E}_{x_S\in S} ||G_{T\to S}(G_{S\to T}(x_S))) - x_S||_1 + \mathbb{E}_{x_T\in T} ||G_{S\to T}(G_{T\to S}(x_T))) - x_T||_1$$

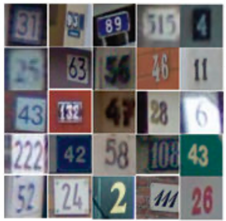
Domain Adaptation - CyCADA method

- **Feature-level adaptation**:
  - train a source model f_S (i.e., feature extractor + label predictor) on S minimizing cross-entropy (CE) loss and a target model f_T; using $G_{S \to T}$; minimizing the CE loss between $f_T(G_{S \to T}(x_S))$ and y_s
  - use a GAN loss at the feature level to align representations. Mappings $G_{S \to T}$ and a new feature-discriminator D_{feat}

$$\min_{G_{S \to T}} \max_{D_{feat}} \mathbb{E}_{x_T \in T} \log D_{feat}(f_T(x_T)) + \mathbb{E}_{x_S \in S} \log(1 - D_{feat}(f_T(G_{S \to T}(x_S))))$$

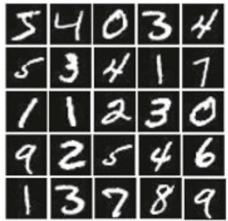
- Authors also propose a *semantic consistency loss* so that the content is preserved before and after translation. They leverage the pre-trained f_S and encourage all images, from both S and T, to be classified in the same way before and after translation by minimizing the CE loss between $f_S(G_{T \to S}(x_T))$ and $\arg\max f_S(X_T)$ as well as the CE loss between $f_S(G_{S \to T}(x_S))$ and $\arg\max f_S(X_T)$.

Domain Adaptation - CyCADA method



SVHN

USPS



MNIST

Model	MNIST → USPS	USPS → MNIST	SVHN → MNIST
Source only	82.2 ± 0.8	69.6 ± 3.8	67.1 ± 0.6
DANN (Ganin et al., 2016)	-	-	73.6
DTN (Taigman et al., 2017a)	-	-	84.4
CoGAN (Liu & Tuzel, 2016a)	91.2	89.1	-
ADDA (Tzeng et al., 2017)	89.4 ± 0.2	90.1 ± 0.8	76.0 ± 1.8
CyCADA pixel only	**95.6 ± 0.2**	96.4 ± 0.1	70.3 ± 0.2
CyCADA pixel+feat	**95.6 ± 0.2**	**96.5 ± 0.1**	**90.4 ± 0.4**
Target only	96.3 ± 0.1	99.2 ± 0.1	99.2 ±0.1

Domain Adaptation - CyCADA method

(a) Without Semantic Loss (b) Without Cycle Loss

Figure: Ablation study removing the semantic consistency loss (left) or the cycle consistency loss (right). Each triple contains the SVHN image (left), the image translated into MNIST style (middle) and the image reconstructed back into SVHN (right).[70]

- An interesting follow-up method based on the conditional adversarial mechanisms is[71]

[70] J. Hoffman et al. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *ICML*. 2018.

[71] M. Long et al. "Conditional Adversarial Domain Adaptation". In: *NeurIPS*. 2018.

Domain Adaptation - Covariate Shift

- Many other methods exist, based on
  - Optimal Transport[72]
  - Separate Style (i.e., textures) from Content[73,74]
  - Self-labeling, Self-training[75] and Co-training[76]
  - Mix-up approach with intermediate domains[77]
- There are also interesting theoretical works[78] and surveys, such as[79,80]

[72] N. Courty et al. "Optimal Transport for Domain Adaptation". In: *IEEE TPAMI* (2016).
[73] H. Nam et al. "Reducing Domain Gap by Reducing Style Bias". In: *CVPR*. 2021.
[74] R. Geirhos et al. "ImageNet-trained CNNs are biased towards texture ...". In: *ICLR*. 2019.
[75] V. Prabhu et al. "SENTRY: Selective Entropy Optimization via Committee Consistency for UDA". In: *ICCV*. 2021.
[76] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).
[77] J. Na et al. "FixBi: Bridging Domain Spaces for Unsupervised Domain Adaptation". In: *CVPR*. 2021.
[78] A. Mehra et al. "Understanding the Limits of Unsupervised Domain Adaptation via Data Poisoning". In: *NeurIPS*. 2021.
[79] I. Redko et al. *Advances in Domain Adaptation Theory*. 2019.
[80] X. Liu et al. "Deep Unsupervised Domain Adaptation: A Review of Recent Advances and Perspectives". In: *APSIPA* (2022).

- The *target (prior) shift* assumptions are $p_S(X|Y) = p_T(X|Y)$ and $p_S(Y) \neq p_T(Y)$ refer in particular to class imbalance and cost-sensitive learning.

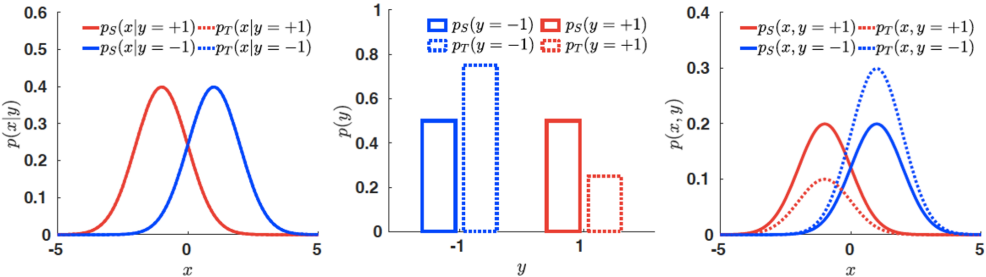


Figure: Fig. from[81]

[81] W. M. Kouw et al. *An introduction to domain adaptation and transfer learning.* Tech. rep. 2019.

Domain Adaptation - Target Shift

- Under the target shift assumptions, $p_S(X|Y) = p_T(X|Y)$ and $p_S(Y) \neq p_T(Y)$, the empirical generalization target error can be rewritten as:

$$\hat{e}_T(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{p_T(x_i, y_i)}{p_S(x_i, y_i)} = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{\cancel{p_T(x_i|y_i)}p_T(y_i)}{\cancel{p_S(x_i|y_i)}p_S(y_i)} = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\underbrace{\frac{p_T(y_i)}{p_S(y_i)}}_{w(y_i)}$$

- The importance weights $w(x) = \frac{p_T(y)}{p_S(y)}$ correct for the change in class priors[82]
- If target labels are available one can under- or over-sample data points from one class. **But what if target labels (unsupervised) are not available ?**

[82] W. M. Kouw et al. "A Review of Domain Adaptation without Target Labels". In: *IEEE TPAMI* (2021).

Domain Adaptation - Target Shift

- A possible method is **Black Box Shift Estimation (BBSE)**[83][84]:
  - train a model $f \in \mathcal{F}$ on the train split of source S domain
  - compute the confusion matrix $C_{f,y}$ on a validation split of S
  - make predictions for the target data T: $f(x \in T)$
  - compute the empirical target prior for label/class y as the proportion of target samples classified by f to class y: $\hat{p}_T(f(x) = y) = \frac{1}{m} \sum_{j=1}^{m} I(f(x_j) = y)$
  - weights are computed as the product of the inverse confusion matrix and predicted empirical target priors: $\hat{w} = C_{f,y}^{-1} \hat{p}_T(f(x) = y)$, which is a vector of size $[\#classes, 1]$
  - Estimate the final model f' on S by using $\max(\hat{w}, 0)$ as weights. Practically, you weight the training samples $(x_i, y_i = k)$ by $w[k]$

[83] Z. C. Lipton et al. "Detecting and Correcting for Label Shift with Black Box Predictors". In: *ICML*. 2018.

[84] S. Rabanser et al. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *NeurIPS*. 2019.

Domain Adaptation - Concept Shift

- The *concept shift* assumptions, $p_S(Y|X) \neq p_T(Y|X)$ and $p_S(X) = p_T(X)$, refers to the case where source S and target T domains share the same feature and labels/classes but they have different tasks.

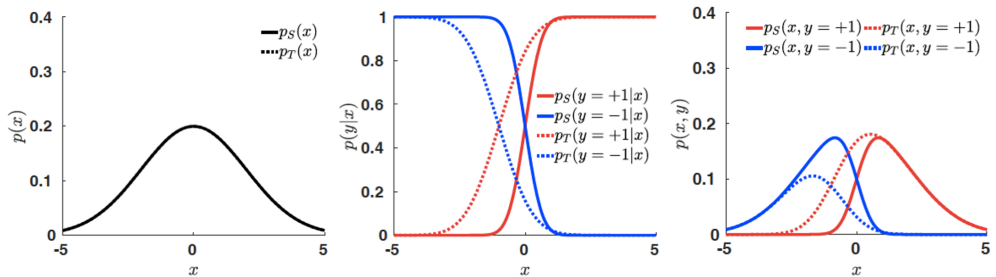


Figure: Fig. from[85]

[85] W. M. Kouw et al. *An introduction to domain adaptation and transfer learning.* Tech. rep. 2019.

Domain Adaptation - Concept Shift

- Under the *concept shift* assumptions, $p_S(Y|X) \neq p_T(Y|X)$ and $p_S(X) = p_T(X)$, the empirical generalization target error can be rewritten as:

$$\hat{e}_T(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{p_T(x_i, y_i)}{p_S(x_i, y_i)} = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\frac{p_T(y_i|x_i)\cancel{p_T(x_i)}}{p_S(y_i|x_i)\cancel{p_S(x_i)}}$$

- However, in this case, adapting the source classifier to the target domain requires **labeled** data from both domains !

- Concept shift is also related to *data drift* in non-stationary environments, like time-series. At each time step, the posterior distribution can change but if one supposes small changes, then the drift can be modeled and the next time step predicted (here we use a smooth change assumption $\rightarrow$ induction bias)

Summary

Multitask learning (MTL)

Multitask learning (MTL) aims to improve generalization by jointly learning **multiple** and **related** tasks $\mathcal{T}$. Source and target tasks are different, labeled and learned simultaneously. Domains usually have the same $\mathcal{X}$ but different $p(X)$

- As humans can leverage knowledge or experience acquired while learning a task A to better perform a related task B, MTL[86][87] leverages the knowledge acquired about a task A to improve B.
- Since all tasks are jointly learned, the goal is to simultaneously improve the generalization performance of the main or all tasks[88][89][90]

[86] R. Caruana. "Multitask Learning". In: *Machine Learning* (1997).
[87] J. Baxter. "A Model of Inductive Bias Learning". In: *Journal of Artificial Intelligence Research* (2000).
[88] A. Maurer et al. "The Benefit of Multitask Representation Learning". In: *JMLR* (2016).
[89] S. Vandenhende et al. "Multi-Task Learning for Dense Prediction Tasks: A Survey". In: *IEEE TPAMI* (2022).
[90] Y. Zhang et al. "A Survey on Multi-Task Learning". In: *IEEE TKDE*. 2022.

Multitask Learning

- All ML/DL methods have an **inductive/learning bias** which can be defined as *"anything (e.g., assumptions, architecture constraints, optimization choices) that makes the algorithm prefer some hypotheses/patterns over others to predict its output"*[91]
- MTL uses the training signals of related tasks as an inductive bias to improve generalization $\rightarrow$ This makes the algorithm prefer patterns that are useful for all tasks.[92]
- The goal of MTL is to leverage the additional sources of information to improve the performance of learning a main task (or all tasks). Inductive transfer from the related tasks may improve:
  - generalization accuracy (less overfitting)
  - speed of learning
  - less parameters to tune
  - less training data needed

[91] T. M. Mitchell. "The Need for Biases in Learning Generalizations". In: *Readings in Machine Learning*. 1980.

[92] R. Caruana. "Multitask Learning". In: *Machine Learning* (1997).

The no-free-lunch theorem

- In Machine Learning and optimization, the **no-free-lunch theorem**[93] states that some preferences (or inductive bias) are necessary for the algorithm to generalize → there is no completely general-purpose learning algorithm, **all algorithms will generalize well on some data and worse on other**

- Most ML methods use assumptions, constraints, regularizations to predict and generalize. This means that they prefer one representation (set of parameters) over another → the algorithm thus *prioritize solutions with certain properties*

- An inductive bias in neural network is the translation invariance property (even if we translate the object in the image, the output should not change). It can be obtained by:
  - replacing matrix multiplication by convolutions and pooling (i.e., CNN)
  - averaging the network predictions over transformations of the input
  - using data augmentation

[93] D. Wolpert et al. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1 (1997).

The no-free-lunch theorem

- AI researchers aiming at human-level performance need thus to identify inductive biases that are most relevant to the human perspective on the world
- There are many inductive biases in deep learning algorithms. Here it is a list from[94]:

inductive bias	corresponding property
distributed representations	patterns of features
convolution	group equivariance (usually over space)
deep architectures	complicated functions = composition of simpler ones
graph neural networks	equivariance over entities and relations
recurrent nets	equivariance over time
soft attention	equivariance over permutations
self-supervised pre-training	$P(X)$ is informative about $P(Y\|X)$

[94] A. Goyal et al. "Inductive biases for deep learning of higher-level cognition". In: *Proc. R. Soc. A.* (2022).

Multitask Learning

- MTL improves generalization by training tasks in parallel while using a **shared representation** → Mostly two ways: *Hard* or *Soft* parameter sharing

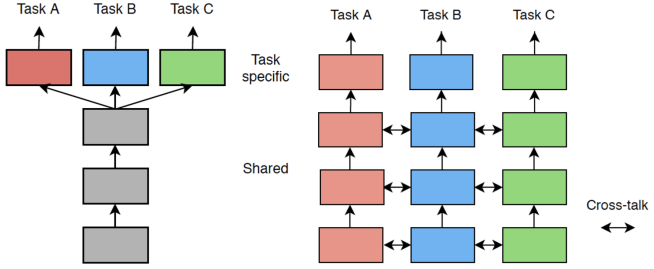


Figure: Image from[95]. **Hard Sharing**: A shared encoder followed by (small) task-specific decoding heads. **Soft sharing**: each task has its own network but the input to each layer is a combination (e.g., linear[96]) of the outputs of the previous layers from every network.

[95] S. Vandenhende et al. "Multi-Task Learning for Dense Prediction Tasks: A Survey". In: *IEEE TPAMI* (2022).

[96] I. Misra et al. "Cross-Stitch Networks for Multi-task Learning". In: *CVPR*. 2016.

Which are the hidden mechanisms of MTL ?[97]:

- **Statistical Data Amplification**: if the training signals of two (or more) tasks are noisy, then it will be better to average the parameters update coming from multiple losses (i.e., tasks) instead than from a single loss (i.e., task).

[97] R. Caruana. "Multitask Learning". In: *Machine Learning* (1997).

Which are the hidden mechanisms of MTL ?[97]:

- **Statistical Data Amplification**: if the training signals of two (or more) tasks are noisy, then it will be better to average the parameters update coming from multiple losses (i.e., tasks) instead than from a single loss (i.e., task).
- **Mutual help**: if a shared encoder (or hidden layer) F is useful to multiple tasks, but it's difficult to learn with task T_A, it will be better to learn the parameters of F using the other tasks than simply using T_A. Furthermore, the net will leverage F to better perform in T_A.

[97] R. Caruana. "Multitask Learning". In: *Machine Learning* (1997).

Which are the hidden mechanisms of MTL ?[97]:

- **Statistical Data Amplification**: if the training signals of two (or more) tasks are noisy, then it will be better to average the parameters update coming from multiple losses (i.e., tasks) instead than from a single loss (i.e., task).
- **Mutual help**: if a shared encoder (or hidden layer) F is useful to multiple tasks, but it's difficult to learn with task T_A, it will be better to learn the parameters of F using the other tasks than simply using T_A. Furthermore, the net will leverage F to better perform in T_A.
- **Common representations**: if two or more tasks share one local minima for the parameters of the shared encoder (or hidden layer) F, then it's likely that the MTL will fall into that representation. Similarly, if one representation is not convenient for a task, it's likely that MTL will avoid that.

[97] R. Caruana. "Multitask Learning". In: *Machine Learning* (1997).

- We assume T tasks and for each task t we have a training set $\{(x_{ti} \in \mathbb{R}^d, y_{ti} \in \mathbb{R})\}_{i=1}^N$, where N and d are the same for all tasks.
- $\{(x_{ti}, y_{ti})\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} P_t$, where P_t is the joint pdf on $X \times Y$. All P_t are different but *related*.
- **Goal**: estimate the T functions $f_t : \mathbb{R}^d \to \mathbb{R}$ by minimizing the empirical loss:

$$\underset{f_t \in \mathcal{F}\ \forall t}{\arg\min} \sum_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_t L_t(y_{ti}, f_t(x_{ti}))$$

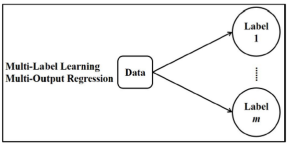
- where the weights w_t are usually considered as hyper-parameters tuned by the user or automatically estimated as in[98][99] . Please note that this optimization problem can also be formulated as a multi-objective optimization as in[100]

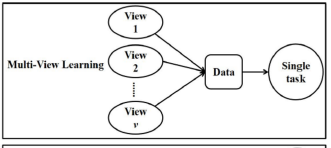
[98] A. Kendall et al. "Multi-task Learning Using Uncertainty to Weigh Losses for Scene ...". In: *CVPR*. 2018.
[99] S. Liu et al. "End-To-End Multi-Task Learning With Attention". In: *CVPR*. 2019.
[100] O. Sener et al. "Multi-Task Learning as Multi-Objective Optimization". In: *NeurIPS*. 2018.

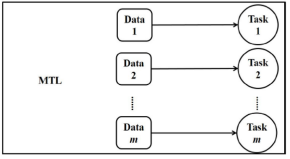
$$\underset{f_t \in \mathcal{F}\ \forall t}{\arg\min} \sum_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_t L_t(y_{ti}, f_t(x_{ti}))$$

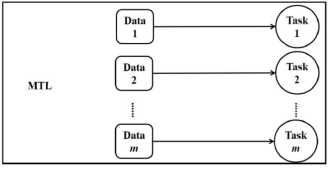
- Please note that one might also have a different number of samples N_t and features dimension d_t per task t. But tasks must be related.

- Another typical simplification is to consider the same input for all tasks: $x_{tj} = x_{t'j}\ \forall t, t' \rightarrow$ this is also called Multi-label learning or Multi-output regression.[a]









[a] Y. Zhang et al. "A Survey on Multi-Task Learning". In: *IEEE TKDE*. 2022.

Multitask Learning

- In MTL, tasks need to be related to each other to be of help (or related to the task/s of interest) $\rightarrow$ related $\neq$ correlated output signals ! Similar imaging patterns that are exploited by similar internal representations of the network.

[101] M. Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020.

[102] S. Ben-David et al. "Exploiting Task Relatedness for Multiple Task Learning". In: *COLT*. 2003.

Multitask Learning

- In MTL, tasks need to be related to each other to be of help (or related to the task/s of interest) $\rightarrow$ related $\neq$ correlated output signals ! Similar imaging patterns that are exploited by similar internal representations of the network.
- We thus need to **minimize the negative transfer** or destructive interference $\rightarrow$ tasks that are not related may have conflicting needs. Increasing the performance on one task may hurt the performance of another, unrelated, task[101]
- Almost all methods propose a strategy to separate the common from the task-specific parameters/features $\rightarrow$ this results in a well organized and generalizable representation

[101] M. Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020.
[102] S. Ben-David et al. "Exploiting Task Relatedness for Multiple Task Learning". In: *COLT*. 2003.

Multitask Learning

- In MTL, tasks need to be related to each other to be of help (or related to the task/s of interest) $\rightarrow$ related $\neq$ correlated output signals ! Similar imaging patterns that are exploited by similar internal representations of the network.

- We thus need to **minimize the negative transfer** or destructive interference $\rightarrow$ tasks that are not related may have conflicting needs. Increasing the performance on one task may hurt the performance of another, unrelated, task[101]

- Almost all methods propose a strategy to separate the common from the task-specific parameters/features $\rightarrow$ this results in a well organized and generalizable representation

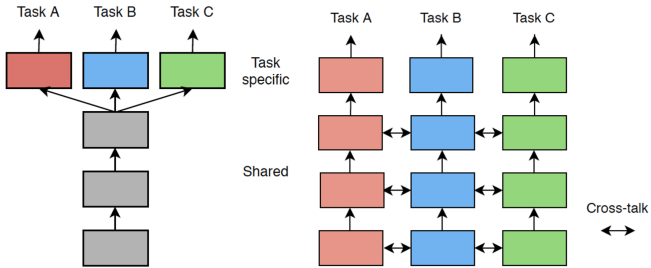
- How to understand if two tasks are related *a priori* ? $\rightarrow$ very hard question ! Usually one uses heuristic/assumptions. For instance, two training sets of two different tasks, that can be transformed one into the other, can be defined as "related"[102]

[101] M. Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020.

[102] S. Ben-David et al. "Exploiting Task Relatedness for Multiple Task Learning". In: *COLT*. 2003.

Multitask Learning

- Most of the works are based on one of these two architectures.
- **What about the actual neural architecture present in our brain ?** We are very efficient multi-task learners (at least some of us...) ! $\rightarrow$ can we mimic it ?



(a) Hard parameter sharing (b) Soft parameter sharing

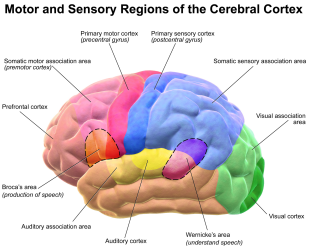
AI and cognitive neuroscience

Figure: Images created using `https://dream.ai/create`

- Many AI research algorithms try to mimic the brain mechanisms and take inspiration from cognitive science and neuroscience → **Synergy** between domains !
- At the same time, AI models can drive new insights into the neural mechanisms underlying conscious processing, instantiating a virtuous circle.[103]

[103] A. Goyal et al. "Inductive biases for deep learning of higher-level cognition". In: *Proc. R. Soc. A.* (2022).

AI and cognitive neuroscience



- The human brain cortex contains multiple regions with distinct, specialized functions that are are selectively activated by a specific perceptual or cognitive task $\rightarrow$ **Why did evolution make this choice ?**

- Three possible answers are:[104]
  - accident of evolution: it can more easily add modules to solve new problems than redesign an entire system from scratch
  - functional specialization allows mental processes to be selectively modulated
  - computational reasons: distinct brain regions arise only for tasks that cannot be solved using a more generic machinery based on the entire brain

[104] K. Dobs et al. "Brain-like functional specialization emerges spontaneously in deep NN". In: *Science Advances* (2022).

Can we study DL to infer about our brain ?

- Is face recognition functionally segregated in the brain because more domain-general visual representations simply do not suffice ? Test it with CNNs

[105] K. Dobs et al. "Brain-like functional specialization emerges spontaneously in deep NN". In: *Science Advances* (2022).

Can we study DL to infer about our brain ?

- Is face recognition functionally segregated in the brain because more domain-general visual representations simply do not suffice ? Test it with CNNs
- Authors in[105] measure face and object recognition performance in CNNs trained to classify faces, objects, or both. Their findings are:
  - networks trained only on objects perform poorly on face recognition and much worse than face-trained networks
  - face discrimination spontaneously segregates from object recognition in networks trained on both tasks despite the lack of built-in face-specific inductive biases.
  - spontaneous segregation happens not only for faces but also for other categories

[105] K. Dobs et al. "Brain-like functional specialization emerges spontaneously in deep NN". In: *Science Advances* (2022).

Can we study DL to infer about our brain ?

- Is face recognition functionally segregated in the brain because more domain-general visual representations simply do not suffice ? Test it with CNNs
- Authors in[105] measure face and object recognition performance in CNNs trained to classify faces, objects, or both. Their findings are:
  - networks trained only on objects perform poorly on face recognition and much worse than face-trained networks
  - face discrimination spontaneously segregates from object recognition in networks trained on both tasks despite the lack of built-in face-specific inductive biases.
  - spontaneous segregation happens not only for faces but also for other categories
- Tendency for task segregation in networks $\rightarrow$ **Functional segregation in brains is a natural consequence of optimization to solve multiple tasks**?

[105] K. Dobs et al. "Brain-like functional specialization emerges spontaneously in deep NN". In: *Science Advances* (2022).

Summary

Can we distill knowledge from an ensemble of models or from a large and cumbersome model to a small and fast one ?

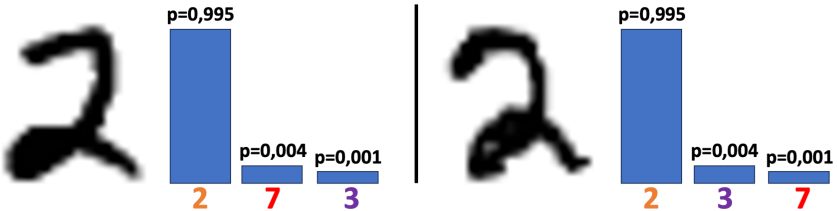
- Yes ! First Caruana et al.[106] and then Hinton et al.[107] propose a very simple way to do that, called **Knowledge Distillation**
- It has originally been designed for classification problems and it can be seen as a form of "*model compression*": a big model is compressed into a small one.
- It is also related to *privileged information*[108,109], where a Student learns from a Teacher that has "more information and knowledge".

Better than a thousand days of diligent study is one day with a great teacher

[106] C. Buciluǎ et al. "Model compression". In: *KDD*. 2006.
[107] G. Hinton et al. "Distilling the Knowledge in a Neural Network". In: *NIPS Workshop*. 2014.
[108] V. Vapnik et al. "Learning Using Privileged Information: Similarity Control and Knowledge Transfer". In: *JMLR* (2015).
[109] D. Lopez-Paz et al. "Unifying distillation and privileged information". In: *ICLR*. 2016.

What is knowledge in a trained (classification) model ?

- We usually identify it with the learned model's parameters → how can we change the model but keep the same knowledge ? Difficult... Change definition !
- **Knowledge** is about the learned mapping from an input sample to the output probabilities. All probabilities (for all classes and not only the most probable one) are important and, in particular, the **relative difference between probabilities**.



- Little *p* tell us that there are two kinds of digit 2, one more similar to 3 and one more similar to 7....

- ... but they have little influence on the cross-entropy loss ! Calling z_i (resp. v_i) the logit of class i for the small model (resp. big model) and q_i (resp. p_i) its probability (after the softmax), we obtain:

$$C(p,q) = -\sum_i p_i \log(q_i) = -\sum_i \left[\frac{\exp(v_i)}{\sum_j \exp(v_j)} \log\left(\frac{\exp(z_i)}{\sum_j \exp(z_j)} \right) \right] =$$
$$-0,995 \log\left(\frac{\exp(z_1)}{\sum_j \exp(z_j)} \right) - 0,004 \log\left(\frac{\exp(z_2)}{\sum_j \exp(z_j)} \right) - 0,001 \log\left(\frac{\exp(z_3)}{\sum_j \exp(z_j)} \right) \approx$$
$$-0,995 \log\left(\frac{\exp(z_1)}{\sum_j \exp(z_j)} \right)$$

- where we have used the previous MNIST example with 3 classes.
- This means that using **hard probabilities** within the Cross-entropy loss won't be useful to transfer knowledge between models !

Knowledge Distillation

- Use **soft probabilities** by dividing each logit by a temperature $T \rightarrow p_i = \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)}$

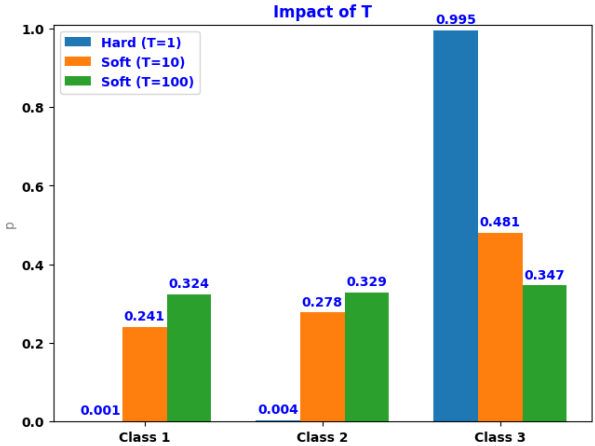


Figure: Effect of dividing the logits by a temperature before the softmax function.

Knowledge Distillation

- The higher is T, the softer is the probability distribution over classes, namely low-probability classes have a greater importance
- Be careful ! low-probable classes might carry important information but also noise. Finding the right temperature T is quite important (and tricky) !
- High temperature T values increase the importance of low-probable classes $\rightarrow$ if T is too big, then all classes have the same importance

- The higher is T, the softer is the probability distribution over classes, namely low-probability classes have a greater importance
- Be careful ! low-probable classes might carry important information but also noise. Finding the right temperature T is quite important (and tricky) !
- High temperature T values increase the importance of low-probable classes $\rightarrow$ if T is too big, then all classes have the same importance

So, how do we use Knowledge Distillation in practice ?

Knowledge Distillation

- We first train the big, Teacher model $f_T \in \mathcal{F}_T$ using a training dataset $\{x_i, y_i\}_{i=1}^N$, by minimizing:

$$f_T = \arg\min_{f \in \mathcal{F}_T} \frac{1}{N} \sum_i \mathcal{L}(y_i, \sigma(f(x_i))) + \Omega(f)$$

- where $\sigma()$ is the softmax function, $\mathcal{L}$ and Ω are a data-term (e.g., cross-entropy) and regularization loss.

- Then, we train the small, student model $f_s \in \mathcal{F}_S$, by minimizing:

$$f_T = \arg\min_{f \in \mathcal{F}_S} \frac{1}{N} \sum_i \left[\mathcal{L}(\sigma(\hat{f}_T(x_i)/T), \sigma(f(x_i)/T)) + \lambda \mathcal{L}(y_i, \sigma(f(x_i))) \right]$$

- where $\hat{f}_T$ is fixed and λ is a trade-off hyper-parameter between learning from real (hard) target y and from the (soft) probabilities of the Teacher[110]

[110] G. Hinton et al. "Distilling the Knowledge in a Neural Network". In: *NIPS Workshop*. 2014.

Knowledge Distillation

- Hinton et al.[111] propose to also multiply the gradient of the soft target by T^2 (i.e., $T^2\mathcal{L}(\sigma(\hat{f}_T(x_i)/T), \sigma(f(x_i)/T))$) so that the contribution of hard and soft targets remains roughly unchanged if changing the value of $T \rightarrow$ important when looking for the best T value
- Vapnik et al.[112] suggest not to soften the probability distribution of the Student f_S classes when comparing with the soft targets of the Teacher, that is:

$$\mathcal{L}(\sigma(\hat{f}_T(x_i)/T), \sigma(f(x_i))$$

- In practice, it highly depends on the application, data distribution and number of training samples training.

[111] G. Hinton et al. "Distilling the Knowledge in a Neural Network". In: *NIPS Workshop*. 2014.
[112] D. Lopez-Paz et al. "Unifying distillation and privileged information". In: *ICLR*. 2016.

Knowledge Distillation - Classification

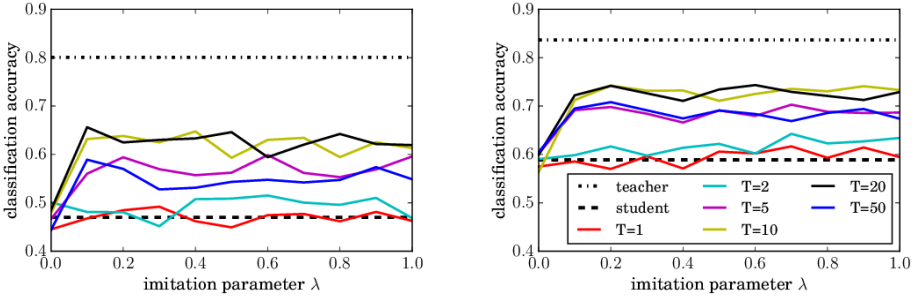
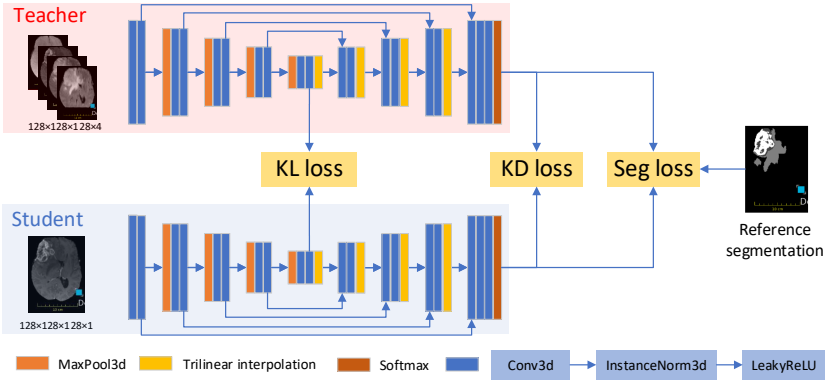


Figure: Results on MNIST for 300 samples (left) and 500 samples (right).

- As shown in[113], the benefits of distillation diminished as we further increased the sample size N. This means that the student model can learn alone... but having a Teacher can accelerate the convergence of the training.

[113] D. Lopez-Paz et al. "Unifying distillation and privileged information". In: *ICLR*. 2016.

Knowledge Distillation - Segmentation



- Knowledge distillation can also be used to distill knowledge from a Multi-modal teacher network to a uni-modal student network.[114]

[114] M. Hu et al. "Knowledge Distillation from Multi-modal to Mono-modal Segmentation Networks". In: *MICCAI*. 2020.

Summary

Meta-learning or Learning to learn

- In Transfer Learning, we have seen that using the parameters of a pre-trained model on a large labeled dataset can be a good **initialization**
- **Limitation**:
  - works well only if the domain gap between source and target domains is small
  - does not work when target data-set has few labeled samples
  - one must use the same architecture for source and target domain

[115] S. Thrun. "Lifelong Learning Algorithms". In: *Learning to Learn*. 1998.

Meta-learning or Learning to learn

- In Transfer Learning, we have seen that using the parameters of a pre-trained model on a large labeled dataset can be a good **initialization**
- **Limitation**:
  - works well only if the domain gap between source and target domains is small
  - does not work when target data-set has few labeled samples
  - one must use the same architecture for source and target domain
- **Goal of Meta-learning**: can we learn a meta-learning model from *several* data-sets (different domains and tasks) so that it will generalize well to new data-sets ?[115] $\rightarrow$ This is also called **learning to learn** !
- When all data-sets have few (or just one) images per class we talk about **few-shot (one-shot) learning**

[115] S. Thrun. "Lifelong Learning Algorithms". In: *Learning to Learn*. 1998.

Meta-learning or Learning to learn

- In Meta-learning one considers datasets as “samples”. The different training sets constitutes the *support set*

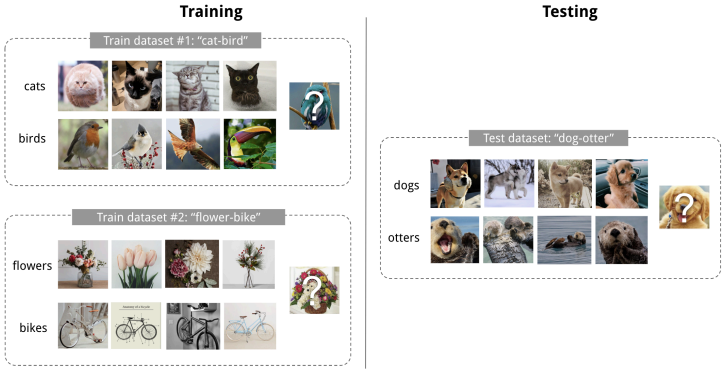
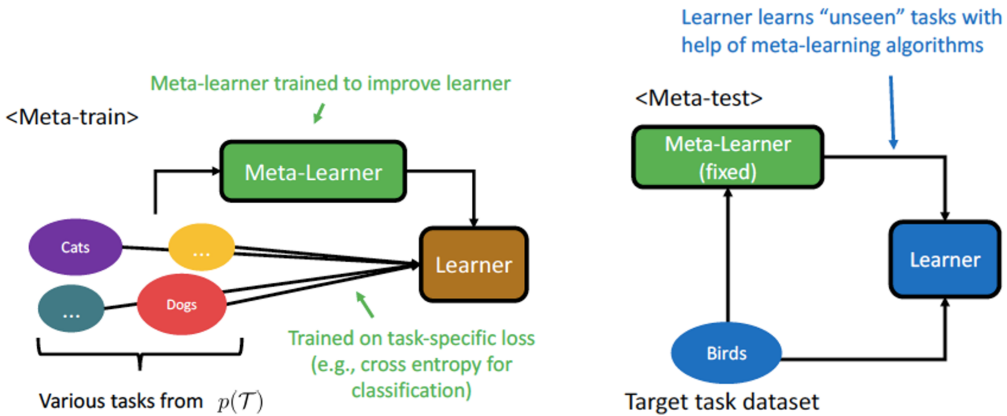


Figure: Example of 3-shot (3 images per class) learning with also 3 classes. Source [116]

[116] https://lilianweng.github.io/posts/2018-11-30-meta-learning

Meta-learning or Learning to learn

- The goal is to learn a **meta-learner model** (i.e., algorithm/strategy/optimizer) so that the learner/model can learn patterns across training tasks and generalize well to unseen tasks. Source [117]



[117] https://alinlab.kaist.ac.kr/resource/Lec17_Meta_learning.pdf

Meta-learning or Learning to learn

- Meta-learning has usually two levels of learning. Source[118]:
  - Inner loop: Fix parameters ϕ of meta-learner and optimize learner/model f parameters θ for each task t, minimizing $\mathcal{L}_{io}(\theta|\phi)$
  - Outer loop: Optimize parameters ϕ of a meta-learner, *which learns how to modify/update the parameters* θ of f , by minimizing $\mathcal{L}_{mo}(\theta, \phi)$, loss to evaluate learner f on new task.

Algorithm 1 Common meta-learning algorithm

1: **while** not done **do**
2: **for** $t = 1, \cdots, T$ **do**
3: Optimize parameters θ of learner f_θ
4: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \nabla_{\theta^{(t)}} \mathcal{L}_{\text{io}}$
5: **end for**
6: Optimize meta-parameters ϕ
7: $\phi \leftarrow \phi - \nabla_{\phi} \mathcal{L}_{\text{mo}}$
8: **end while**

Inner loop

Outer loop

[118] https://alinlab.kaist.ac.kr/resource/Lec17_Meta_learning.pdf

Meta-learning or Learning to learn

- There are three common approaches to meta-learning:
  - **Model-based** $\rightarrow$ find model parameters that work well for various tasks and that can be easily and fast adapted when fine-tuned on new tasks (i.e., good initialization for fine-tuning)
  - **Metric-based** $\rightarrow$ need to define a metric/similarity measure between samples and support samples. Based on that, one evaluate the probability over a set of known labels from the support set (similar to K-NN).
  - **Optimization-based** $\rightarrow$ usual optimizers (e.g., SGD, ADAM) are defined for large, labeled data-sets and can converge in many epochs. Find new strategies to adjust the optimization algorithm so that the model can learn with few samples and in few steps.

Foundation Models

- All these methods are very interesting but... today researchers use **Foundation Models** (e.g., DINOV2[119], CLIP[120]) which are large models pre-trained in a self-supervised way on massive data-sets and many different tasks → zero-shot capabilities ! strong out-of-distribution performance ! Models can be used without fine-tuning.

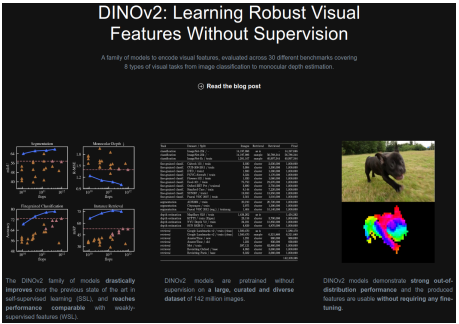


Figure: DINOV2, an example of Foundation Model.

[119] https://dinov2.metademolab.com/

Foundation Models

- All these methods are very interesting but... today researchers use **Foundation Models** (e.g., DINOV2[119], CLIP[120]) which are large models pre-trained in a self-supervised way on massive data-sets and many different tasks → zero-shot capabilities ! strong out-of-distribution performance ! Models can be used without fine-tuning.

Next Lecture

[119] https://dinov2.metademolab.com/
[120] https://openai.com/research/clip

Further reading

- I. Sucholutsky et al. “Getting aligned on representational alignment”. In: *Arxiv* (2023)

References

Abu-Mostafa, Y. et al. *Learning from data: A short course*. 2012.
Ashburner, J. "A fast diffeomorphic image registration algorithm". In: *NeuroImage* (2007).
Baxter, J. "A Model of Inductive Bias Learning". In: *Journal of Artificial Intelligence Research* (2000).
Bell, A. J. et al. "An Information-Maximization Approach to Blind Separation...". In: *Neural Computation* (1995).
Bellman, R. *Dynamic Programming*. Princeton University Press, 1957.
Ben-David, S. et al. "A theory of learning from different domains". In: *Machine Learning* (2010).
Ben-David, S. et al. "Exploiting Task Relatedness for Multiple Task Learning". In: *COLT*. 2003.
Bengio, Y. et al. "Non-Local Manifold Tangent Learning". In: *NIPS*. 2004.
Bengio, Y. et al. "Representation Learning: A Review and New Perspectives". In: *IEEE TPAMI* (2013).
Blitzer, J. et al. "Learning Bounds for Domain Adaptation". In: *NIPS*. 2007.
Buciluǎ, C. et al. "Model compression". In: *KDD*. 2006.
Caruana, R. "Multitask Learning". In: *Machine Learning* (1997).
Chen, T. et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*. 2020.
Cortes, C. et al. "Learning Bounds for Importance Weighting". In: *NIPS*. 2010.
Courty, N. et al. "Optimal Transport for Domain Adaptation". In: *IEEE TPAMI* (2016).
Crawshaw, M. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020.
Deng, J. et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR*. 2009.

References

Dobs, K. et al. “Brain-like functional specialization emerges spontaneously in deep NN”. In: *Science Advances* (2022).

Donahue, J. et al. “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”. In: *ICML*. 2014.

Dufumier, B. “Representation learning in neuroimaging”. PhD thesis. 2022.

Fernando, B. et al. “Unsupervised Visual Domain Adaptation Using Subspace Alignment”. In: *ICCV*. 2013.

Ganin, Y. et al. “Domain-Adversarial Training of Neural Networks”. In: *JMLR* (2017).

Ganin, Y. et al. “Unsupervised Domain Adaptation by Backpropagation”. In: *ICML*. 2015.

Geirhos, R. et al. “ImageNet-trained CNNs are biased towards texture ...”. In: *ICLR*. 2019.

Goodfellow, I. et al. “Generative Adversarial Nets”. In: *NIPS*. 2014.

Goyal, A. et al. “Inductive biases for deep learning of higher-level cognition”. In: *Proc. R. Soc. A.* (2022).

Goyal, P. et al. “Scaling and Benchmarking Self-Supervised Visual Representation Learning”. In: *ICCV*. 2019.

Hinton, G. et al. “Distilling the Knowledge in a Neural Network”. In: *NIPS Workshop*. 2014.

Hoffman, J. et al. “CyCADA: Cycle-Consistent Adversarial Domain Adaptation”. In: *ICML*. 2018.

Hu, M. et al. “Knowledge Distillation from Multi-modal to Mono-modal Segmentation Networks”. In: *MICCAI*. 2020.

Kendall, A. et al. “Multi-task Learning Using Uncertainty to Weigh Losses for Scene ...”. In: *CVPR*. 2018.

References

Khosla, P. et al. “Supervised Contrastive Learning”. In: *NeurIPS*. 2020.

Kim, E. et al. “Probabilistic Concept Bottleneck Models”. In: *ICML*. 2023.

Koh, P. W. et al. “Concept Bottleneck Models”. In: *ICML*. 2020.

Kolesnikov, A. et al. “Revisiting Self-Supervised Visual Representation Learning”. In: *CVPR*. 2019.

Kouw, W. M. et al. “A Review of Domain Adaptation without Target Labels”. In: *IEEE TPAMI* (2021).

– *.An introduction to domain adaptation and transfer learning*. Tech. rep. 2019.

Krizhevsky, A. et al. “ImageNet classification with deep convolutional neural networks”. In: *NIPS*. 2012.

Lake, B. M. et al. “Human-level concept learning through probabilistic program induction”. In: *Science* (2015).

Lin, T.-Y. et al. “Microsoft COCO: Common Objects in Context”. In: *ECCV*. 2014.

Lin, Z. et al. “InfoGAN-CR and ModelCentrality: Self-supervised Model ... for Disentangling GANs”. In: *ICML*. 2020.

Lipton, Z. C. et al. “Detecting and Correcting for Label Shift with Black Box Predictors”. In: *ICML*. 2018.

Liu, S. et al. “End-To-End Multi-Task Learning With Attention”. In: *CVPR*. 2019.

Liu, X. et al. “Deep Unsupervised Domain Adaptation: A Review of Recent Advances and Perspectives”. In: *APSIPA* (2022).

Locatello, F. et al. “Challenging Common Assumptions in the Unsupervised Learning of Disentangled...”. In: *ICML*. 2019.

Locatello, F. et al. “Disentangling Factors of Variation Using Few Labels”. In: *ICLR*. 2020.

References

Locatello, F. et al. “Weakly-Supervised Disentanglement Without Compromises”. In: *ICML*. 2020.

Long, M. et al. “Conditional Adversarial Domain Adaptation”. In: *NeurIPS*. 2018.

Lopez-Paz, D. et al. “Unifying distillation and privileged information”. In: *ICLR*. 2016.

Maurer, A. et al. “The Benefit of Multitask Representation Learning”. In: *JMLR* (2016).

Mehra, A. et al. “Understanding the Limits of Unsupervised Domain Adaptation via Data Poisoning”. In: *NeurIPS*. 2021.

Misra, I. et al. “Cross-Stitch Networks for Multi-task Learning”. In: *CVPR*. 2016.

Mitchell, T. M. “The Need for Biases in Learning Generalizations”. In: *Readings in Machine Learning*. 1980.

Mohri, M. et al. *Foundations of Machine Learning*. MIT Press, 2019.

Na, J. et al. “FixBi: Bridging Domain Spaces for Unsupervised Domain Adaptation”. In: *CVPR*. 2021.

Nam, H. et al. “Reducing Domain Gap by Reducing Style Bias”. In: *CVPR*. 2021.

Pan, S. J. et al. “A Survey on Transfer Learning”. In: *IEEE Trans. Knowl. Data Eng.* (2010).

Prabhu, V. et al. “SENTRY: Selective Entropy Optimization via Committee Consistency for UDA”. In: *ICCV*. 2021.

Rabanser, S. et al. “Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift”. In: *NeurIPS*. 2019.

Redko, I. et al. *Advances in Domain Adaptation Theory*. 2019.

Ren, S. et al. *DeepMIM: Deep Supervision for Masked Image Modeling*. 2023.

References

Roweis, S. T. et al. “Nonlinear Dimensionality Reduction by Locally Linear Embedding”. In: *Science* (2000).

Saito, K. et al. “Maximum Classifier Discrepancy for Unsupervised Domain Adaptation”. In: *CVPR*. 2018.

Sener, O. et al. “Multi-Task Learning as Multi-Objective Optimization”. In: *NeurIPS*. 2018.

Shai Ben, D. et al. “Impossibility Theorems for Domain Adaptation”. In: *AISTATS*. 2010.

Shimodaira, H. “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: *JSPI* (2000).

Simonyan, K. et al. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *ICLR*. 2015.

Sucholutsky, I. et al. “Getting aligned on representational alignment”. In: *Arxiv* (2023).

Szegedy, C. et al. “Intriguing properties of neural networks”. In: *ICLR*. 2014.

Tenenbaum, J. B. et al. “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. In: *Science* (2000).

Thrun, S. “Lifelong Learning Algorithms”. In: *Learning to Learn*. 1998.

Tishby, N. et al. “Deep Learning and the Information Bottleneck Principle”. In: *IEEE ITW*. 2015.

Tschannen, M. et al. “On Mutual Information Maximization for Representation Learning”. In: *ICLR*. 2020.

Vandenhende, S. et al. “Multi-Task Learning for Dense Prediction Tasks: A Survey”. In: *IEEE TPAMI* (2022).

Vapnik, V. “Principles of Risk Minimization for Learning Theory”. In: *NIPS*. 1991.

References

Vapnik, V. et al. “Learning Using Privileged Information: Similarity Control and Knowledge Transfer”. In: *JMLR* (2015).

Wang, X. et al. “Flexible Transfer Learning under Support and Model Shift”. In: *NIPS*. 2014.

Wolpert, D. et al. “No free lunch theorems for optimization”. In: *IEEE Transactions on Evolutionary Computation* 1 (1997).

Wu, Z. et al. “Unsupervised Feature Learning via Non-parametric Instance Discrimination”. In: *CVPR*. 2018.

Yosinski, J. et al. “How transferable are features in deep neural networks?” In: *NIPS*. 2014.

Yu, Y. et al. “Learning Diverse and Discriminative Representations...”. In: *NeurIPS*. 2020.

Zeiler, M. D. et al. “Visualizing and Understanding Convolutional Networks”. In: *ECCV*. 2014.

Zhang, C. et al. “Understanding deep learning requires rethinking generalization”. In: *ICLR*. 2017.

Zhang, K. et al. “Domain Adaptation under Target and Conditional Shift”. In: *ICML*. 2013.

Zhang, L. et al. “Contrastive Deep Supervision”. In: *ECCV*. 2022.

Zhang, Y. et al. “A Survey on Multi-Task Learning”. In: *IEEE TKDE*. 2022.

Zhang, Y. et al. “Bridging Theory and Algorithm for Domain Adaptation”. In: *ICML*. 2019.

Zhu, J.-Y. et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *ICCV*. 2017.

Zhuang, F. et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE*. 2020.