Transfer learning and Curriculum learning

Here, with a **focus** on the **distance** between **tasks**

Defining a **geometry** of the space of learning tasks

Antoine Cornuéjols

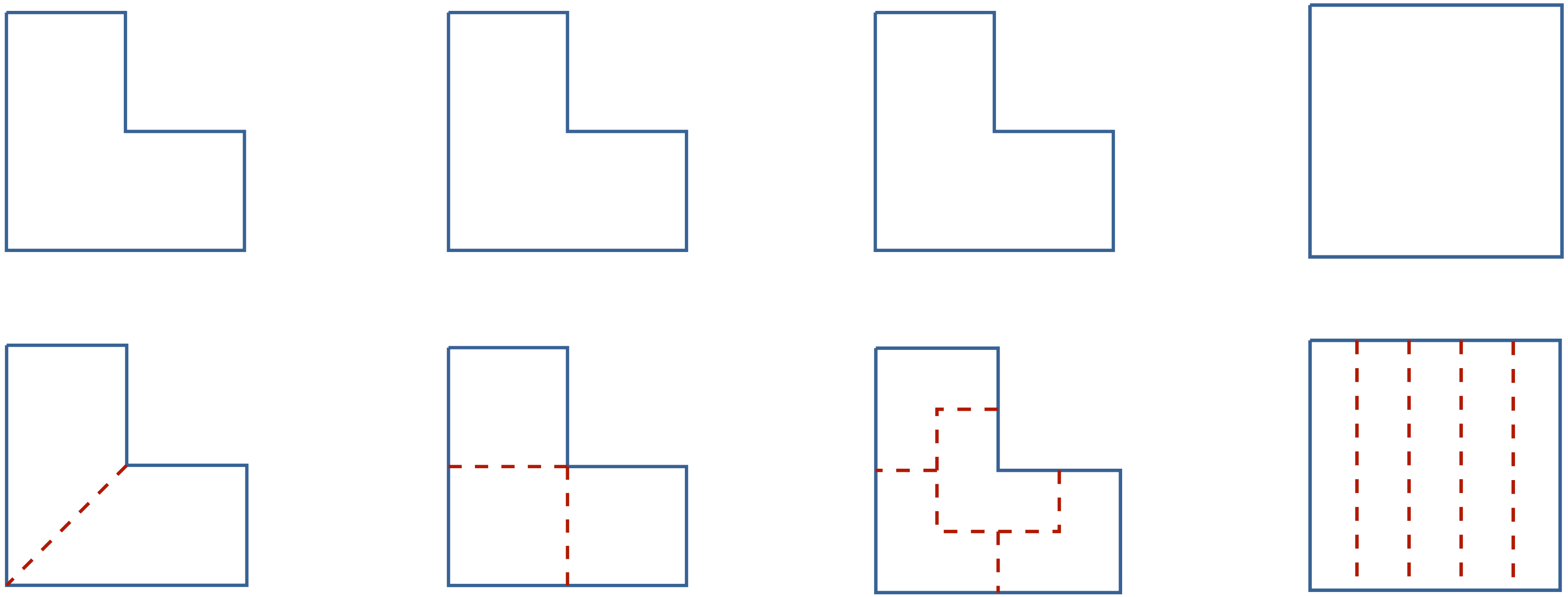
AgroParisTech – INRAE MIA Paris-Saclay

EKINOCS research group

Sequencing effects

- *Instruction*: cut the following figure in *n* **equal parts**

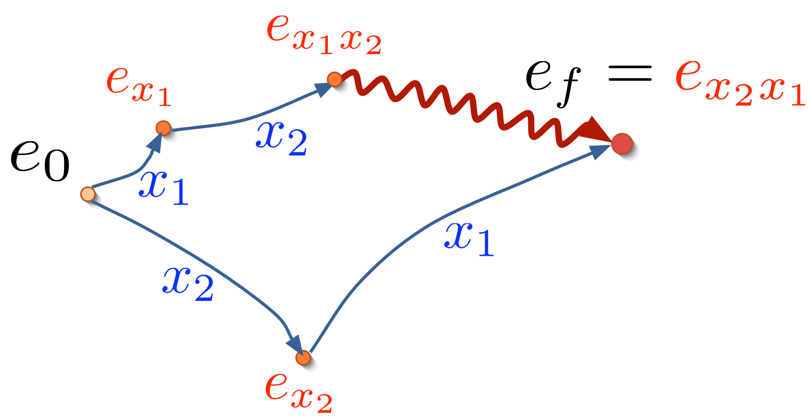
in 2 :   in 3 :   in 4 :   in 5 :



An example of ANTI-**curriculum**

Order effects

- How to **predict** them?
- How to **quantify** them?
- How to **formalize** them?
- How to **control** them?



Continual learning

- What?
  - Do **not retrain** for each new task
  - **Try to benefit** from what has been learned previously
- Why?
  - Often **too costly** to retrain for each new task
    - Lots of (labeled) training data is needed
  - A **good "source"** could provide a lot of useful information
- When?
  - Having a good source
    - **How** to **evaluate** this?
- How?
  - To **transfer** from one **source** to a **target**

Transfer learning

Transfer learning and **curriculum** learning

- An active and constructive viewpoint:
  - Training a system for a target task through **successive intermediate** learning **tasks**
  - Necessitates
    - To **identify** relevant intermediate subtasks
    - To **order** them

Curriculum learning

- **Transfer** learning
  - ability to **use** what has been learned **from a previous task** on a **new task**.

    The **difference with continual learning** is that transfer learning is not concerned about keeping the ability to solve previous tasks.

- **Curriculum** learning
  - a training process that proposes a **sequence of more and more difficult tasks** to a learning algorithm in order to make it able to **learn, at last**, a generally **harder task**.

    The sequence of tasks is designed in order to be able to learn the last one.

When $P_{Y|X}(\text{train}) \neq P_{Y|X}(\text{test})$

(and, not necessarily) $P_X(\text{train}) \neq P_X(\text{test})$

Concept shift

and sequences of concept shifts

Outline

1. Transfer learning: questions
2. Transfer learning in neural networks
3. TransBoost: an algorithm and what it tells on the role of the source
4. Curriculum learning and the geometry of the space of learning tasks
5. How to measure the difficulty of a training example
6. Conclusions

Transfer learning

Questions (more of them)

- What is a **“successful” transfer** learning situation?
  - How to **measure** “success”?
  - How can we **measure** the **performance** of transfer learning?
  - Is “**failure**” possible? Illustrations?

Remark:

if the **target** data set is **sufficiently large**,
transfer learning should not bring any advantage

Questions

- What are the conditions for a **successful transfer** learning?
- Should the proximity between the **source** and the **target** play a role?
  - How to **measure** this proximity?
    - Between the **input distributions** P_S and P_T?
    - Between the **underlying** true source and target **functions** f_S and f_T?
- What should intervene in the guarantees?
  - "**distance**" between source and target?
  - Size of the **target** training **data**?
  - Performance of the **source hypothesis**?

Questions

- **What** to transfer?
- **When** to transfer? Useful or not?
- **How** to transfer?

Bounds between the **real** risk and the **empirical** risk

By removing the “problematic” examples, you go

- From the **non realisable** case ($\mathcal{H}$ finite)

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m\left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2\,m}} \right] \;>\; 1-\delta$$

- To the **realisable** one ($\mathcal{H}$ finite)

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m\left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{m} \right] \;>\; 1-\delta$$

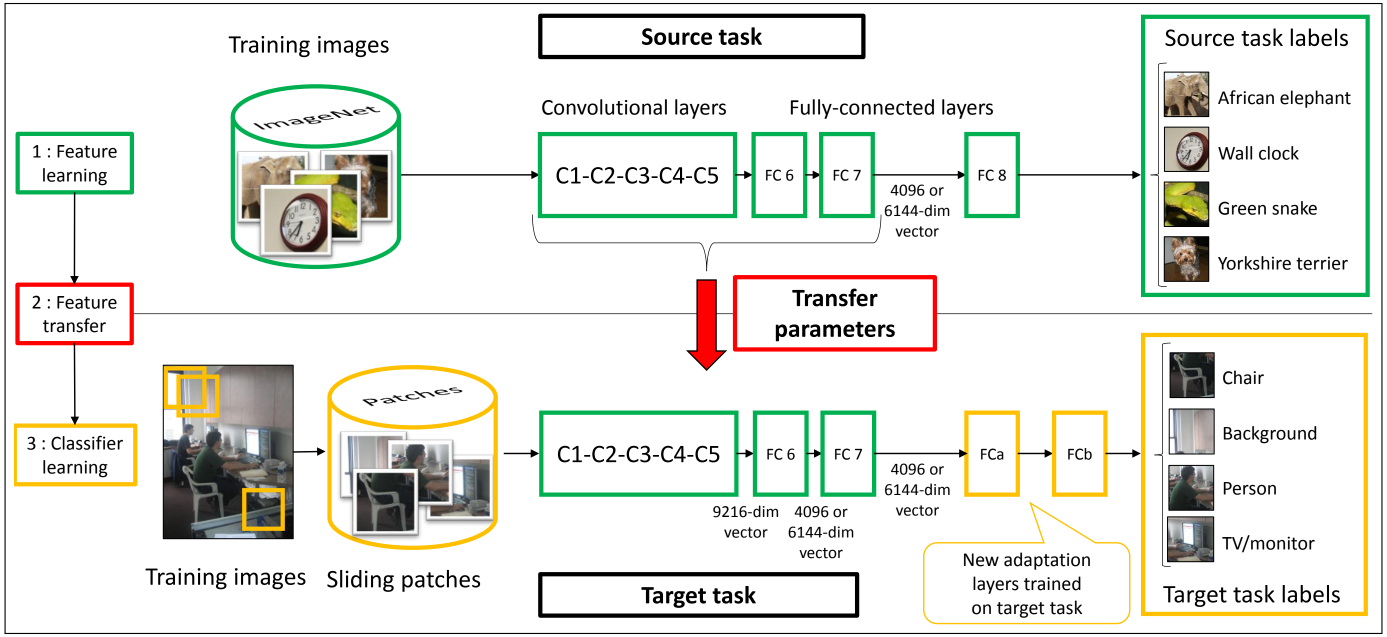
Which **link** between **training** and **testing**?

Transfer Learning

Which **link** between **training** and **testing**?

Transfer Learning

– Reuse the **latent space** learnt on the source data



From Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). **Learning and transferring mid-level image representations using convolutional neural networks**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717-1724).

Baldock, R., Maennel, H., & Neyshabur, B. (2021). **Deep learning through the lens of example difficulty**. *Advances in Neural Information Processing Systems, 34*.

Which **link** between **training** and **testing**?

Transfer Learning

– Reuse the **latent space** learnt on the source data

– Re-use the first layers of a NN trained on task **A**

– And fine-tune on task **B**

→ **Increases** the performance wrt. to training on task B alone

Transfer Learning

- Guarantees function of

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T

Transfer Learning

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T
  - A "**distance**" between the source task and the target one
    - The **smaller** the distance, the **better** the transfer

Transfer Learning

Really?

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T
  - A "**distance**" between the source task and the target one
    - The **smaller** the distance, the **better** the transfer
  - The size of the **target training data**
    - The **larger** the target training data set, the **useless** the transfer

Outline

1. Transfer learning: questions
2. Transfer learning in neural networks
3. TransBoost: an algorithm and what it tells on the role of the source
4. Curriculum learning and the geometry of the space of learning tasks
5. How to measure the difficulty of a training example
6. Conclusions

Transfer learning

for **neural networks**

Transfer learning for **deep neural networks**

- In practice, very few people train an entire Convolutional Network from scratch.
- Instead, it is common to **pretrain a ConvNet** on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories),
  - and then use the ConvNet either as an initialization
  - or a fixed feature extractor for the task of interest.
- Examples of pretrained networks
  - Oxford VGG Model
  - Google Inception Model
  - Microsoft ResNet model

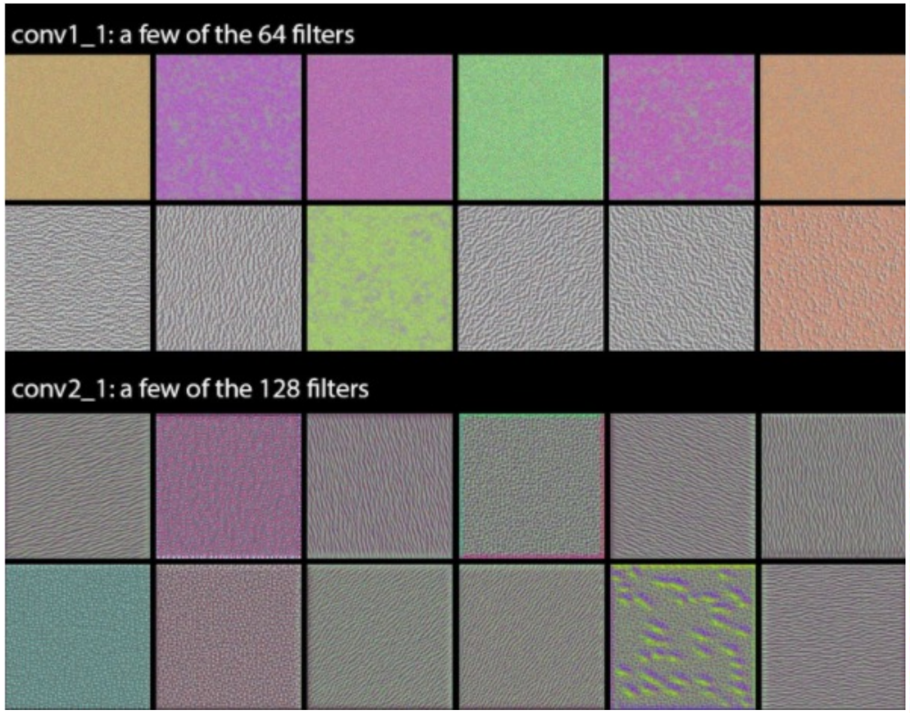
[Yosinski J, Clune J, Bengio Y, and Lipson H. ***How transferable are features in deep neural networks?*** In Advances in Neural Information Processing Systems 27 (NIPS '14), NIPS Foundation, 2014.]

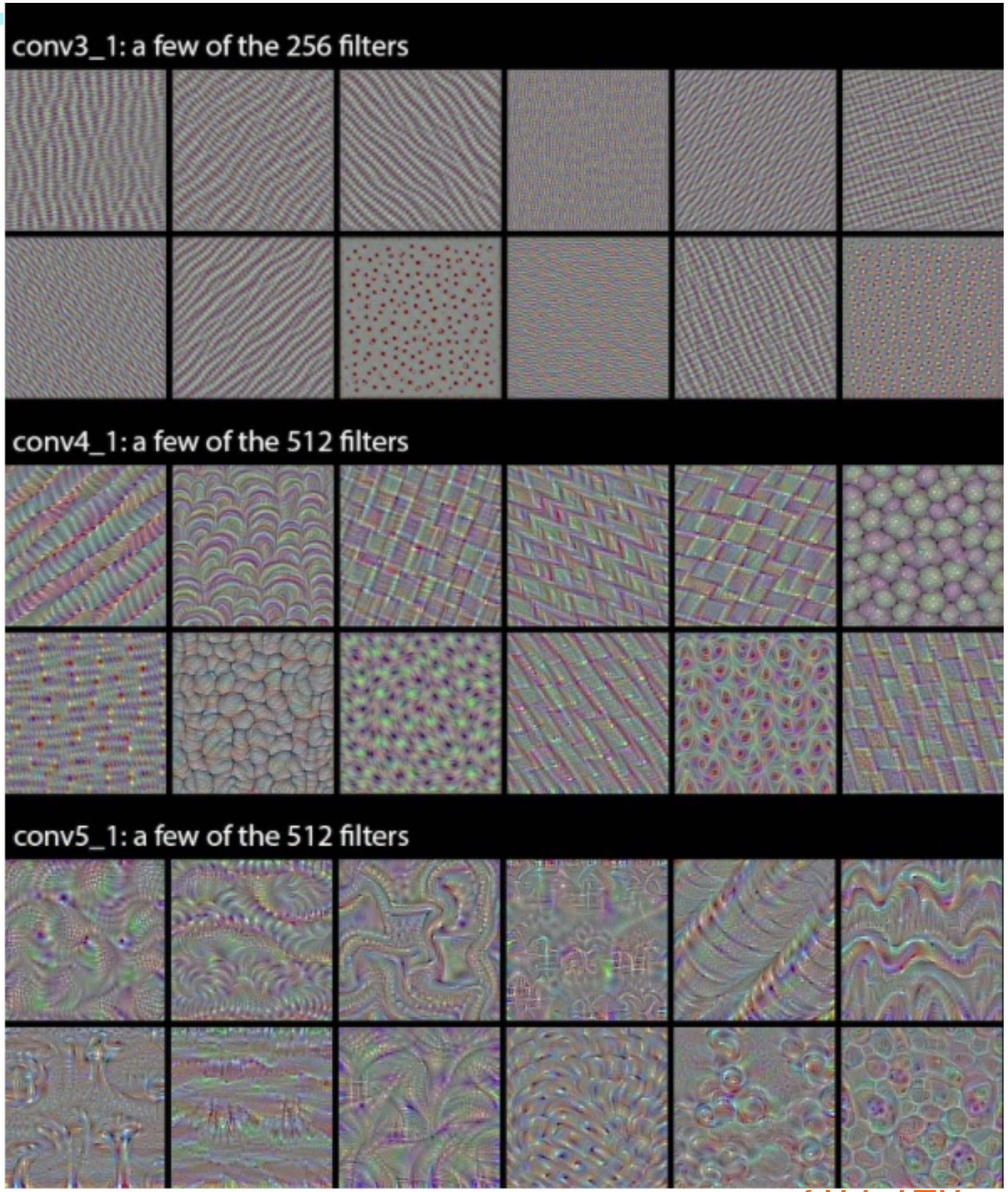
Transfer learning for **deep neural networks**

- The assumption:
  - the features learned for a task can be used almost as such for other, *related*, tasks

- Approach:
  - **Reuse** the first layers and **learn** the last ones
  - Same input spaces $X_S = X_T$, possibly $Y_S \neq Y_T$

Example: VGG 16 filters

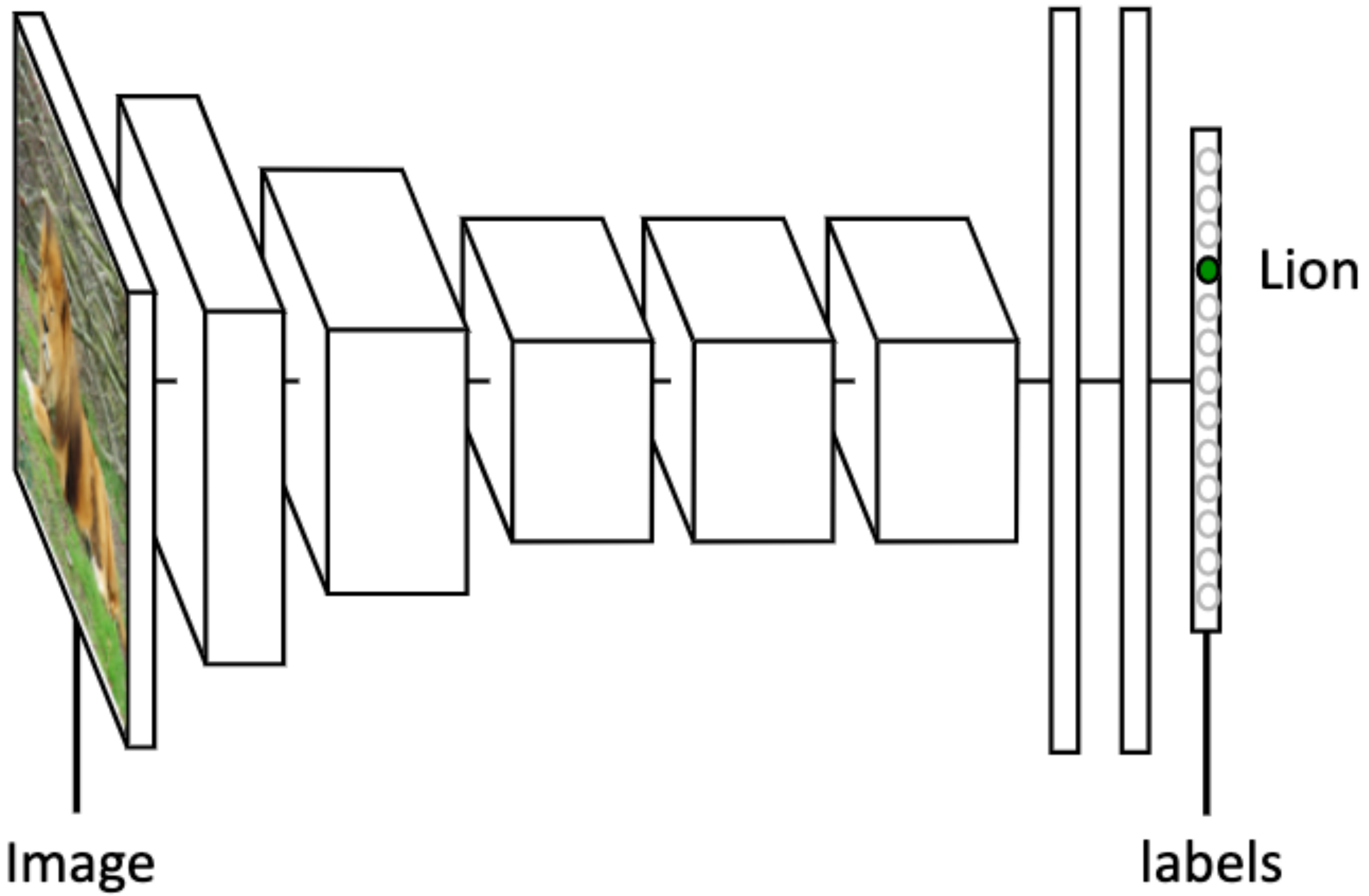




What the **successive layers** learn

...

Principle



Krizhevsky, Sutskever, Hinton — NIPS 2012

...

http://slideplayer.com/slide/8370683/

Transfer learning for deep neural networks

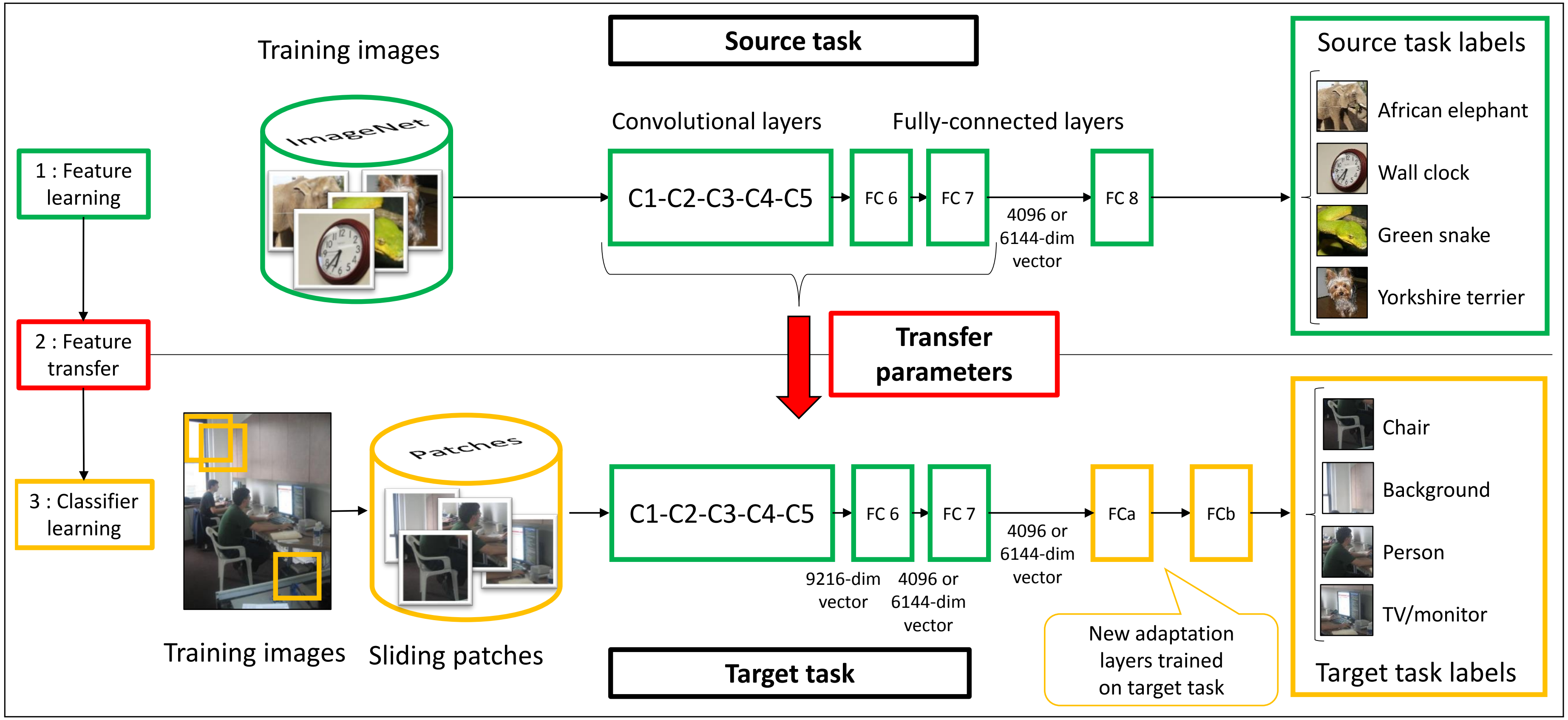
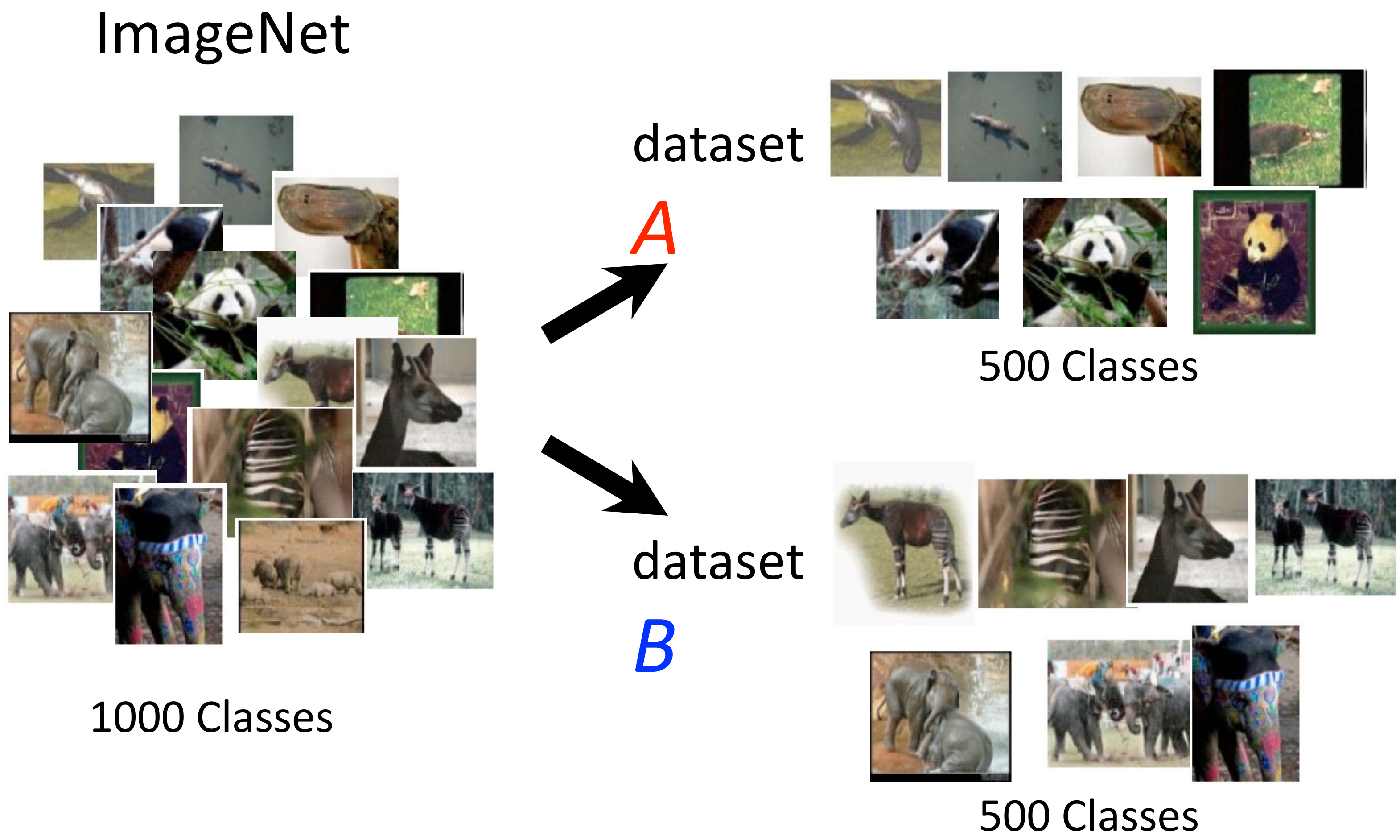


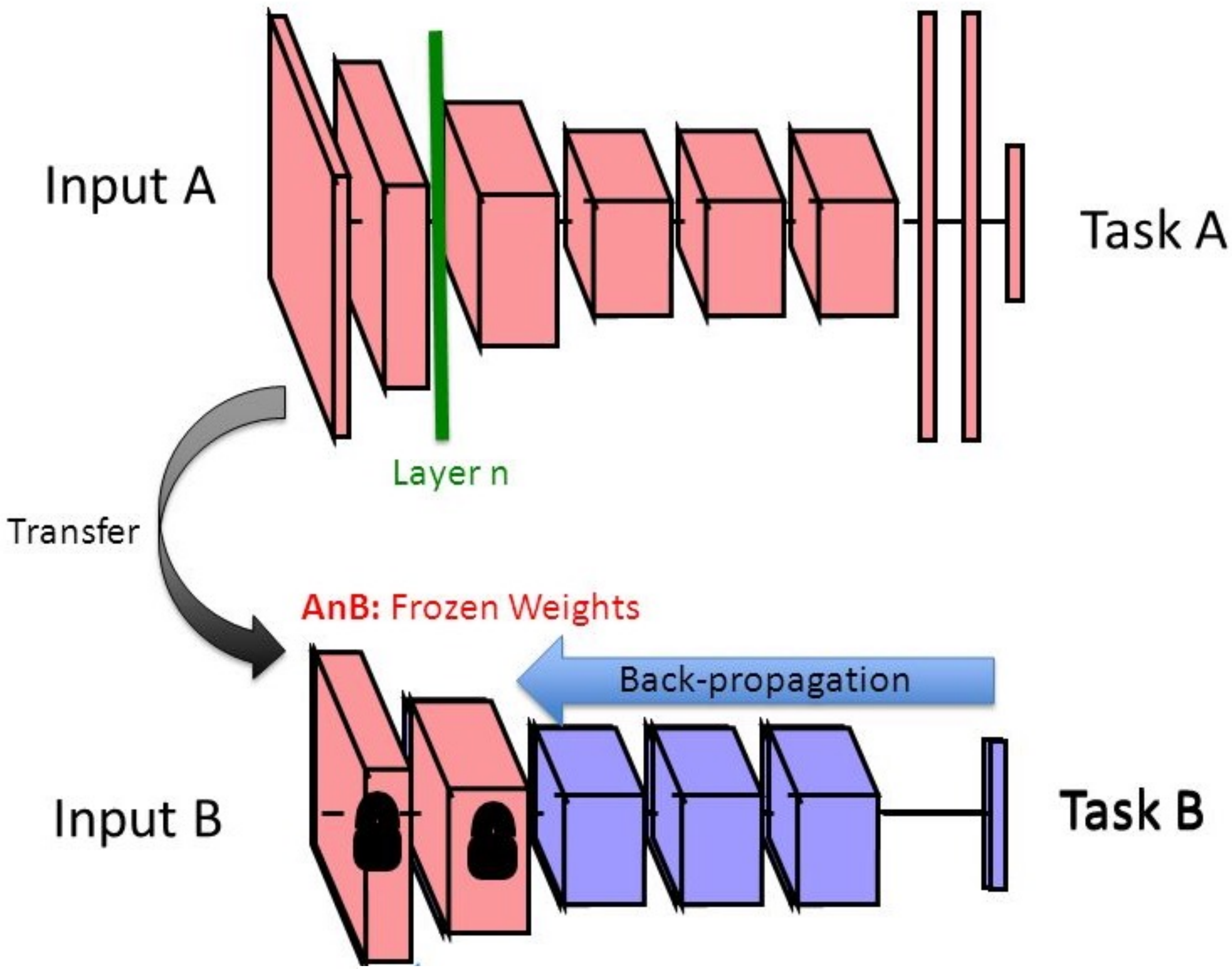
Figure 2: **Transferring parameters of a CNN.** First, the network is trained on the source task (ImageNet classification, top row) with a large amount of available labelled images. Pre-trained parameters of the internal layers of the network (C1-FC7) are then transferred to the target tasks (Pascal VOC object or action classification, bottom row). To compensate for the different image statistics (type of objects, typical viewpoints, imaging conditions) of the source and target data we add an adaptation layer (fully connected layers FCa and FCb) and train them on the labelled data of the target task.

From Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717-1724).

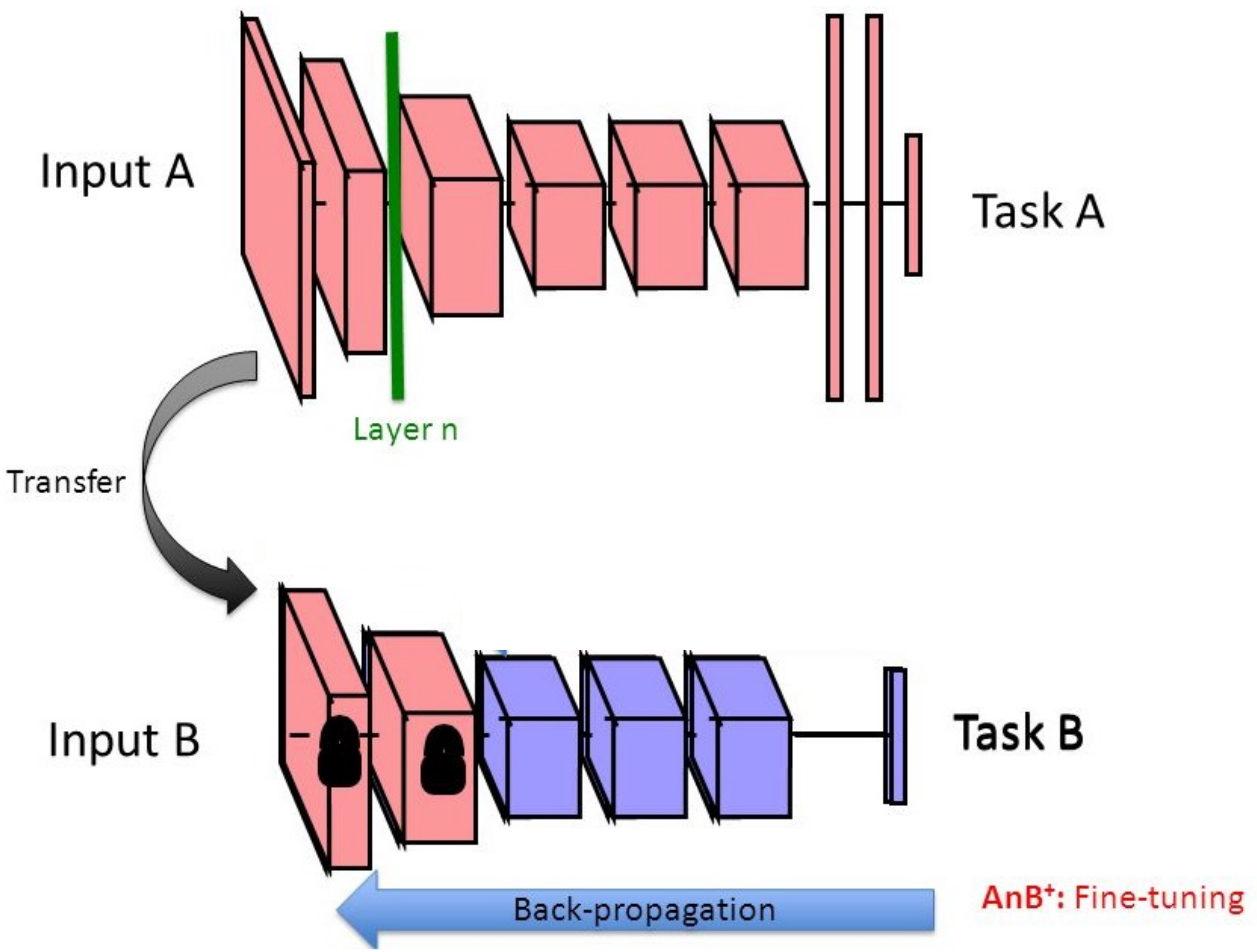
Experiments on two domains



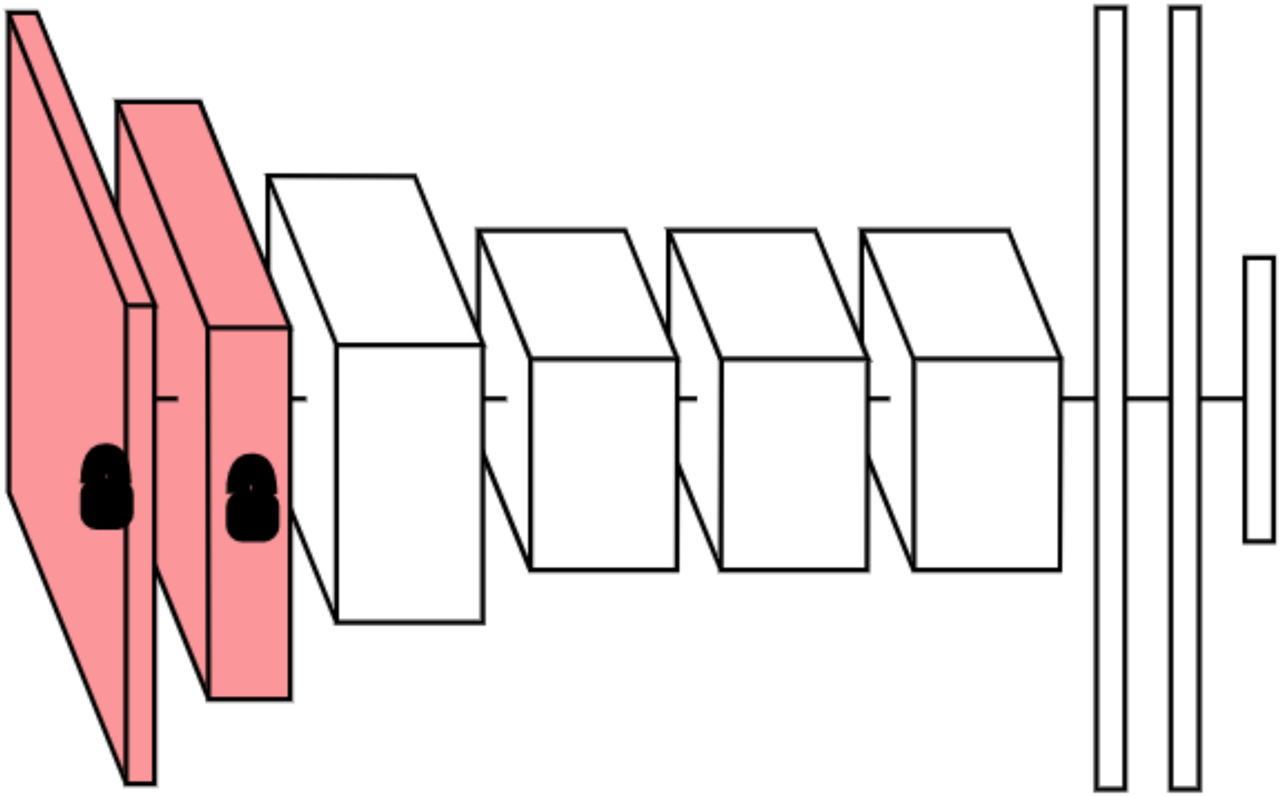
...



Input A
Task A
Layer n
Transfer
AnB: Frozen Weights
Back-propagation
Input B
Task B

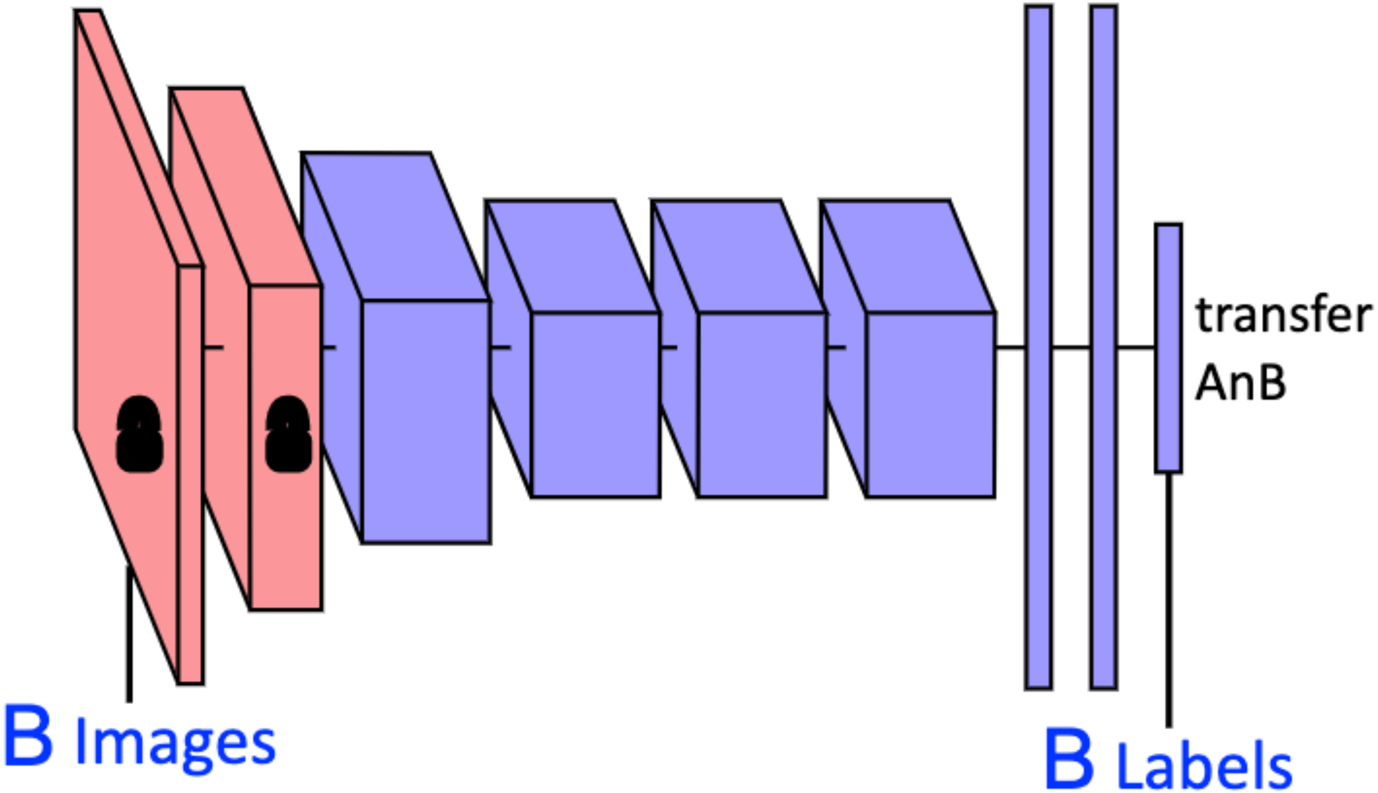


http://slideplayer.com/slide/8370683/



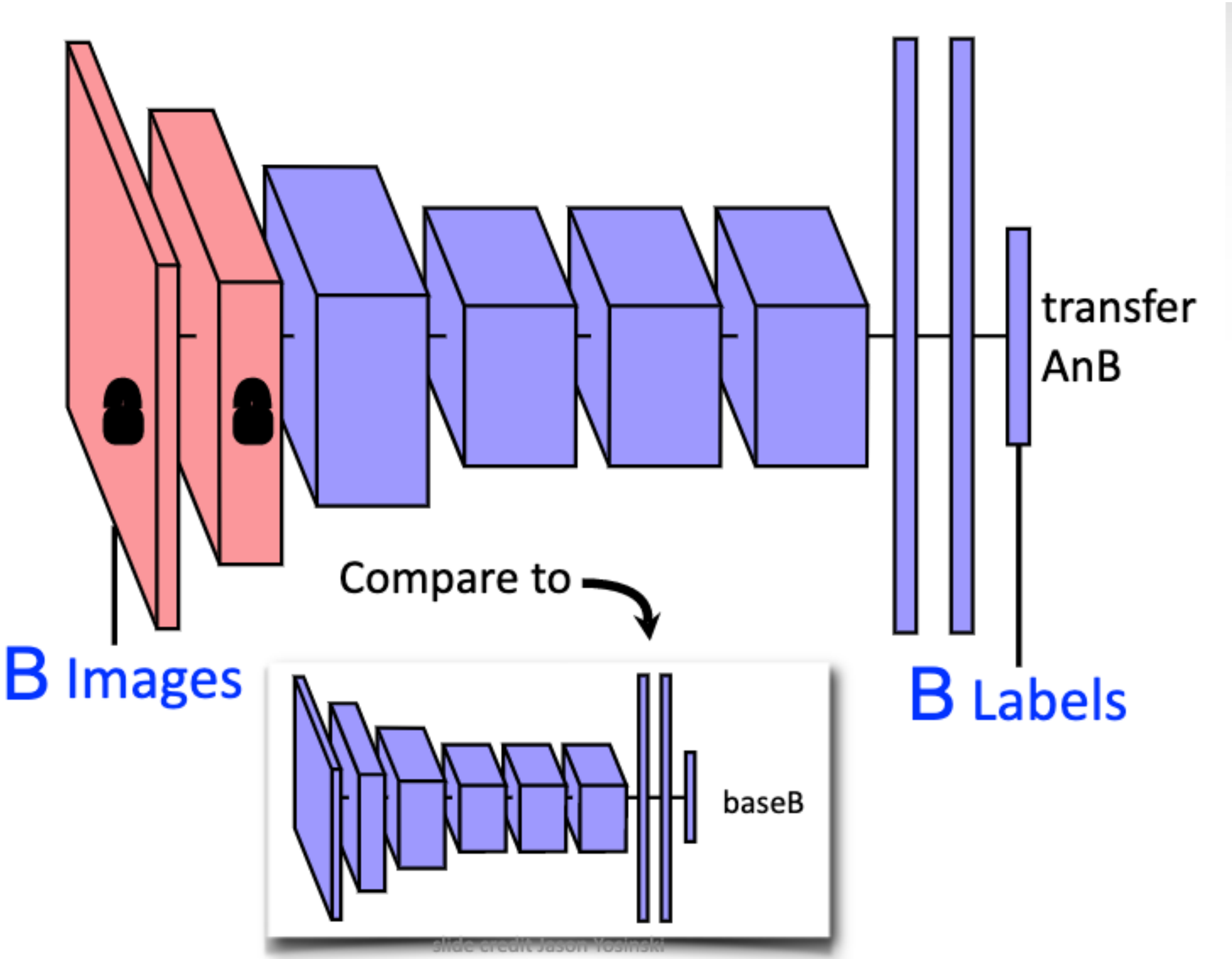
Hypothesis: If transferred features are specific to task A, performance on task B drops. **Otherwise** the performance should be the same.

http://slideplayer.com/slide/8370683/



...

http://slideplayer.com/slide/8370683/



...

http://slideplayer.com/slide/8370683/

- Comparisons between
  - **Base B** : a NN trained directly on database B (500 random classes)
  - **Selffer BnB** (self-transfer):
    - A number of the first layers are frozen, and re-training is done on the last ones
  - **Selffer BnB+** (self-transfer + retraining):
    - A number of the first layers are frozen, and re-training is done on all layers (a kind of initialization, but on the same task)
  - **Transfer AnB** (transfer + fine-tuning last layers only):
  - **Transfer AnB+** (transfer + retraining of all layers):

Results

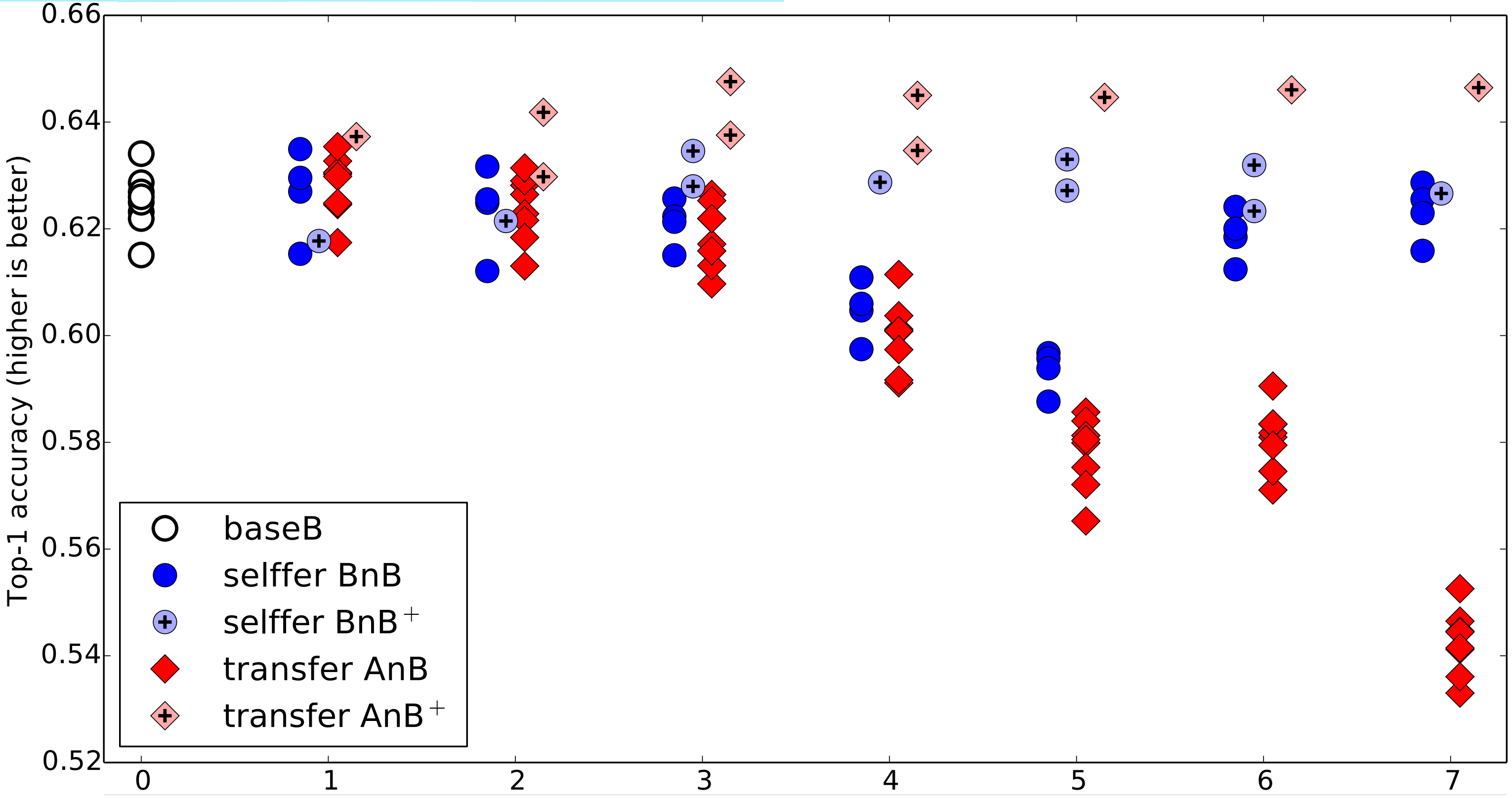


Figure 2: The results from this paper's main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B splits. Each dark blue dot represents a BnB network. Light blue points represent BnB$^+$ networks, or fine-tuned versions of BnB. Dark red diamonds are AnB networks, and light red diamonds are the fine-tuned AnB$^+$ versions. Points are shifted slightly left or right for visual clarity. *Bottom*: Lines connecting the means of each treatment. Numbered descriptions above each line refer to which interpretation from Section 4.1 applies.

...

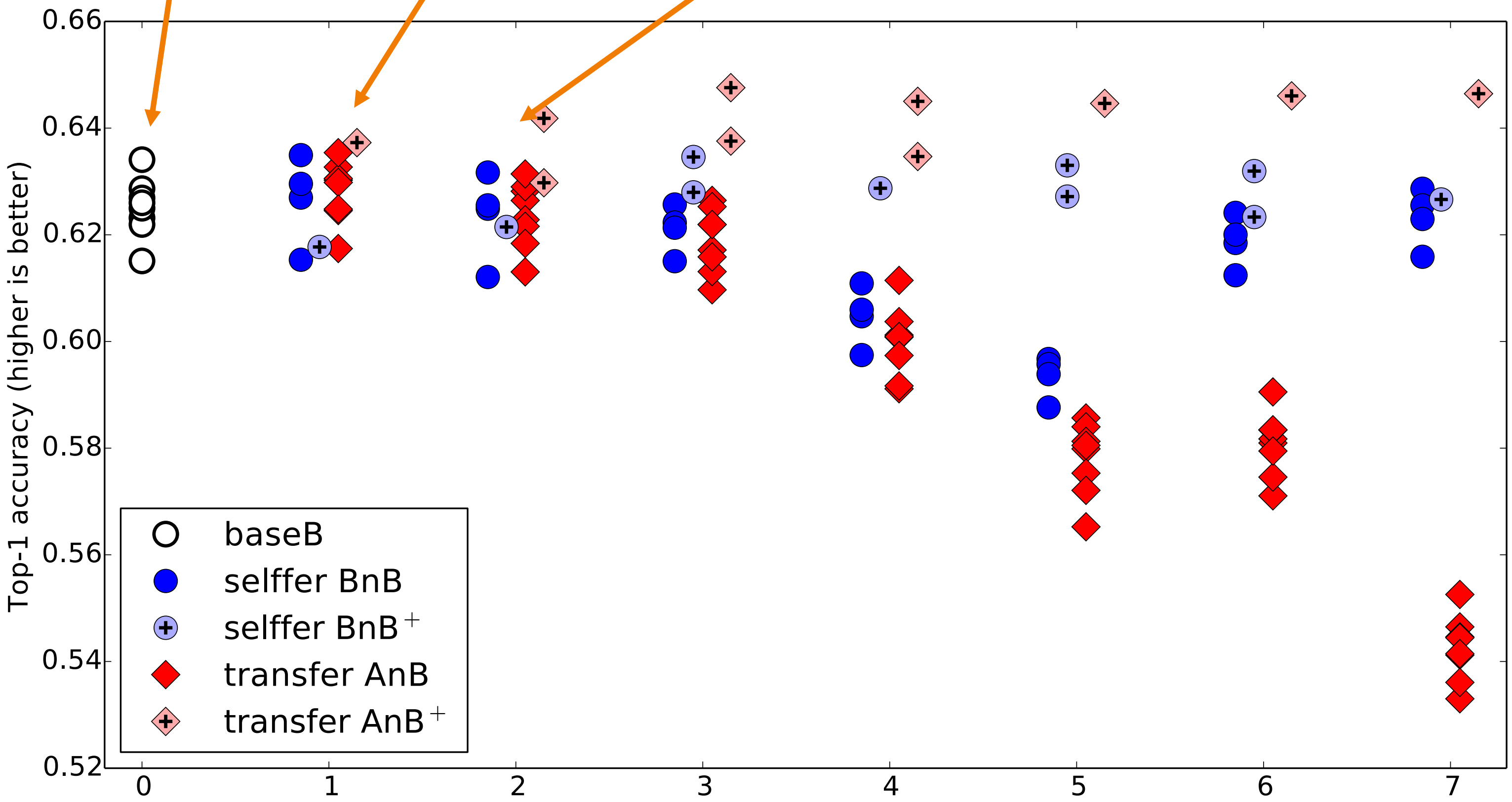
Results: what to think of them?

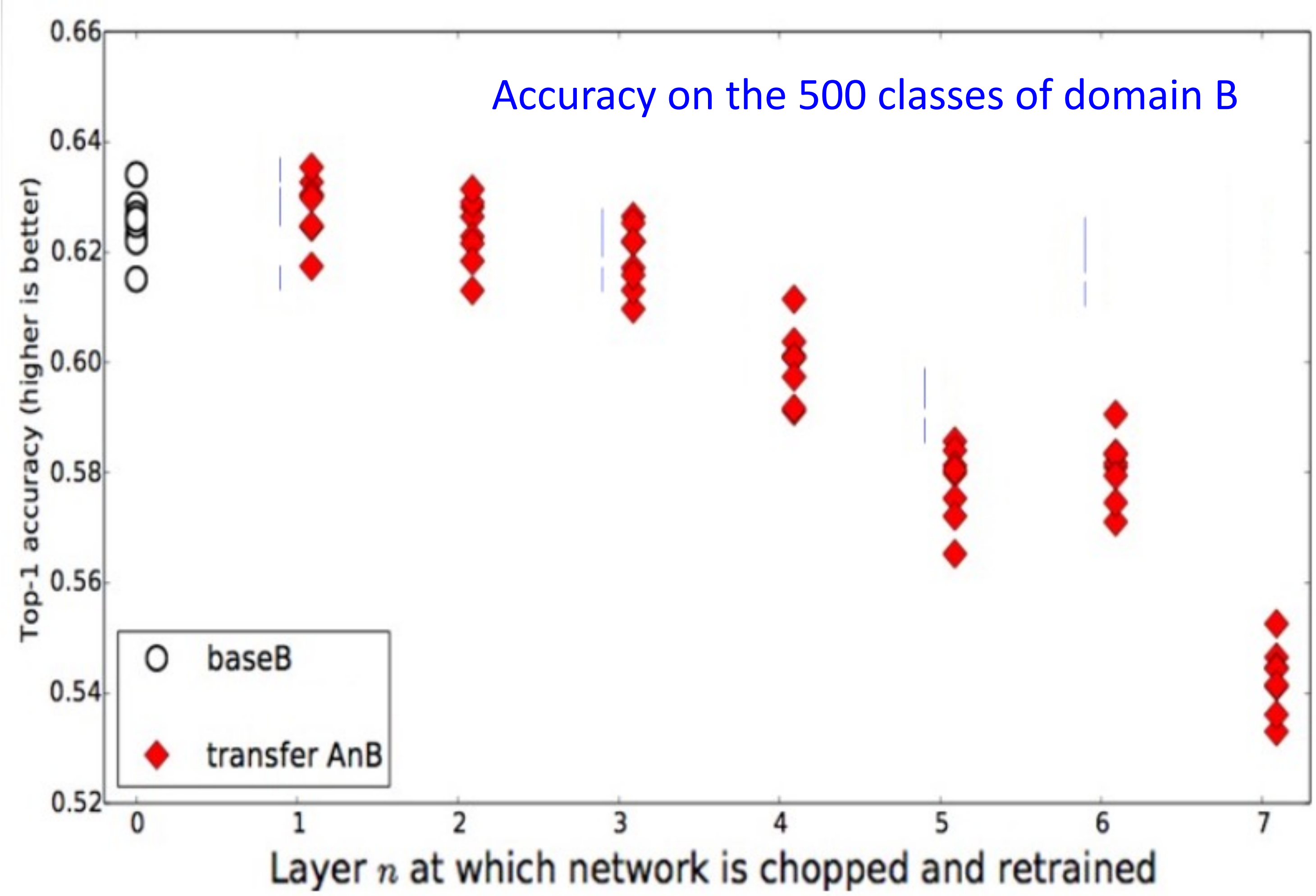
All layers **but the first** are retrained

NN (8 layers) trained from scratch

All layers **but the first two** are retrained

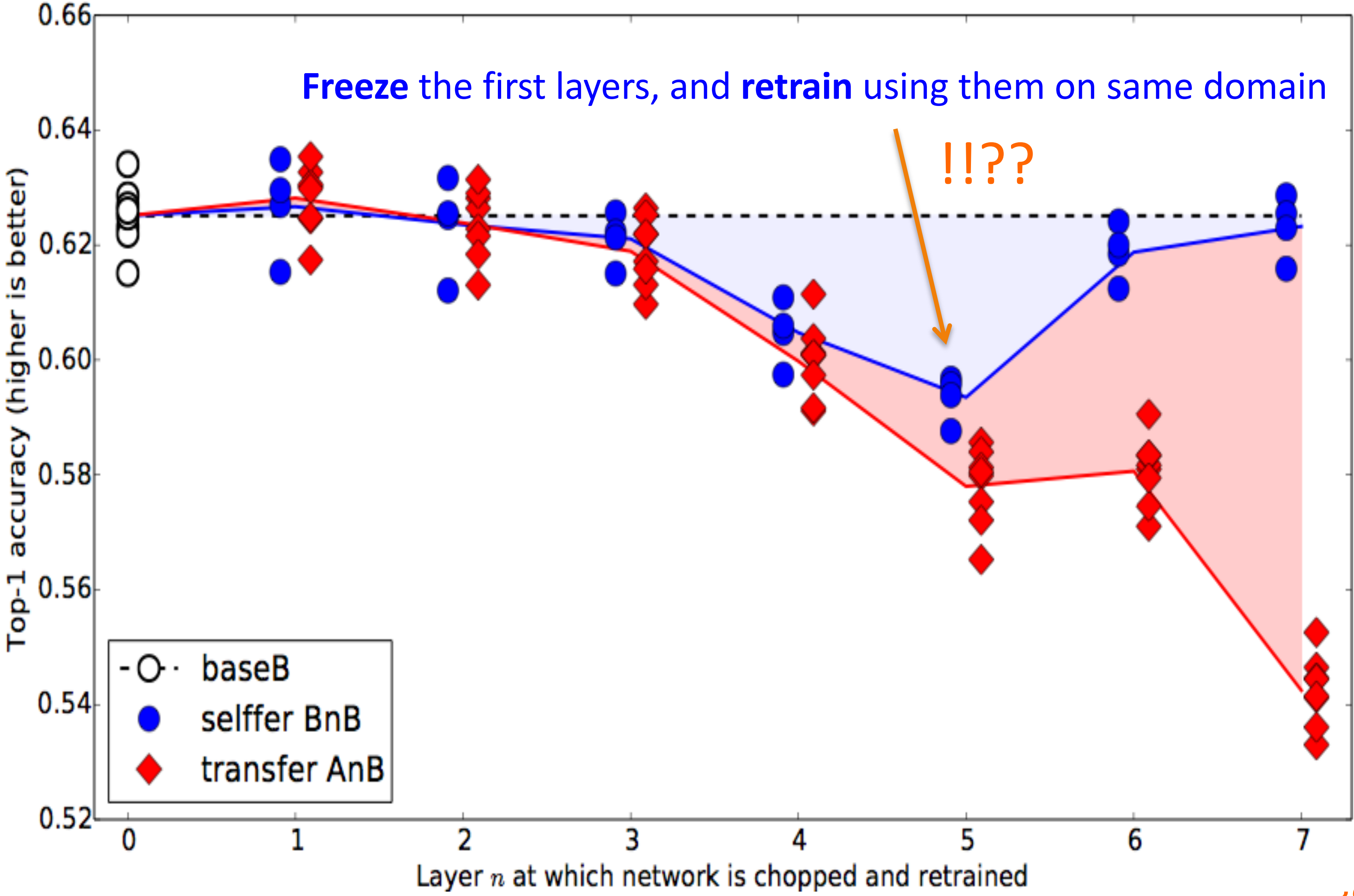
All layers but ... are retrained





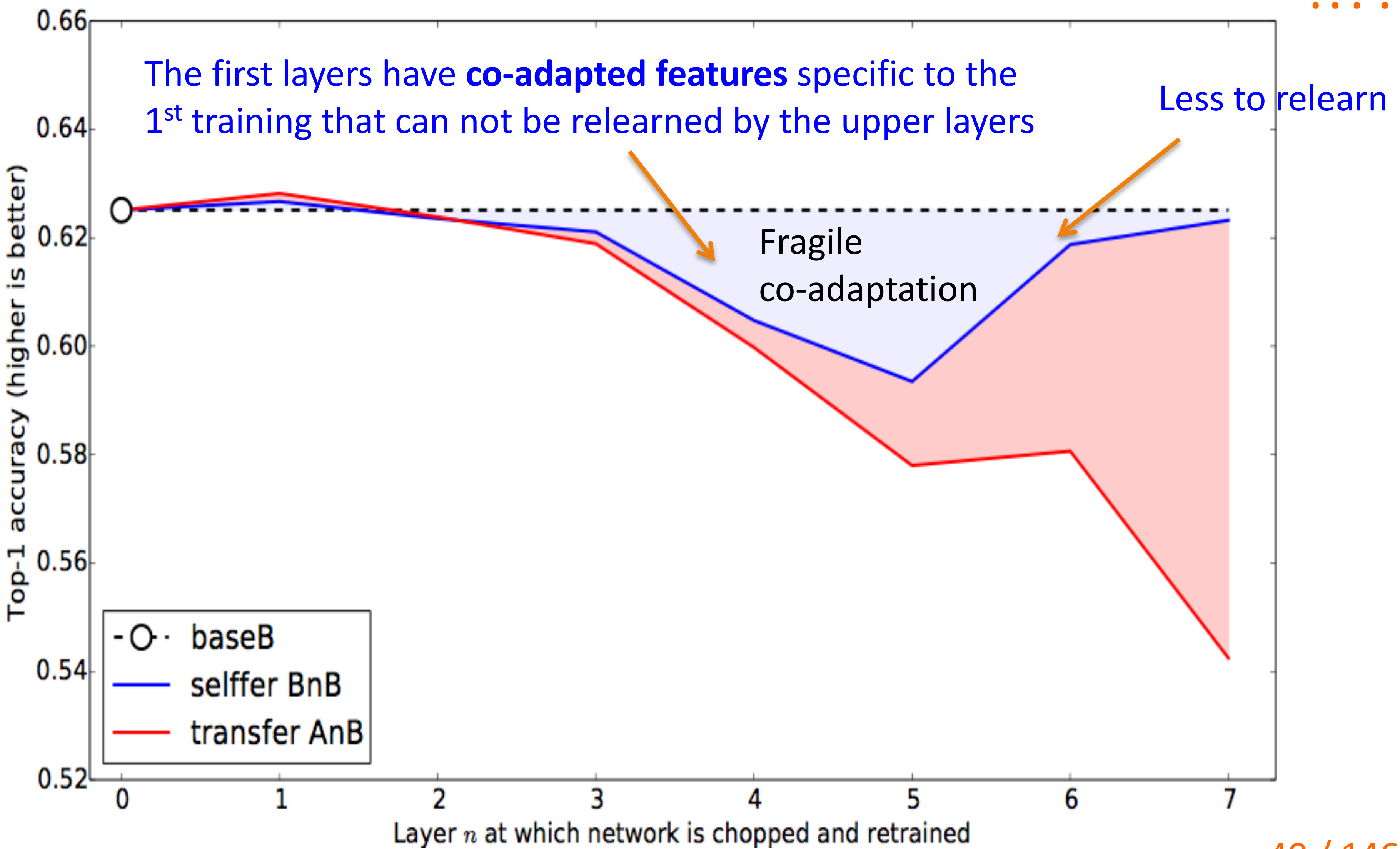
It is clear that the **higher** the layer, the **more specific** it is to task A

Interpretation



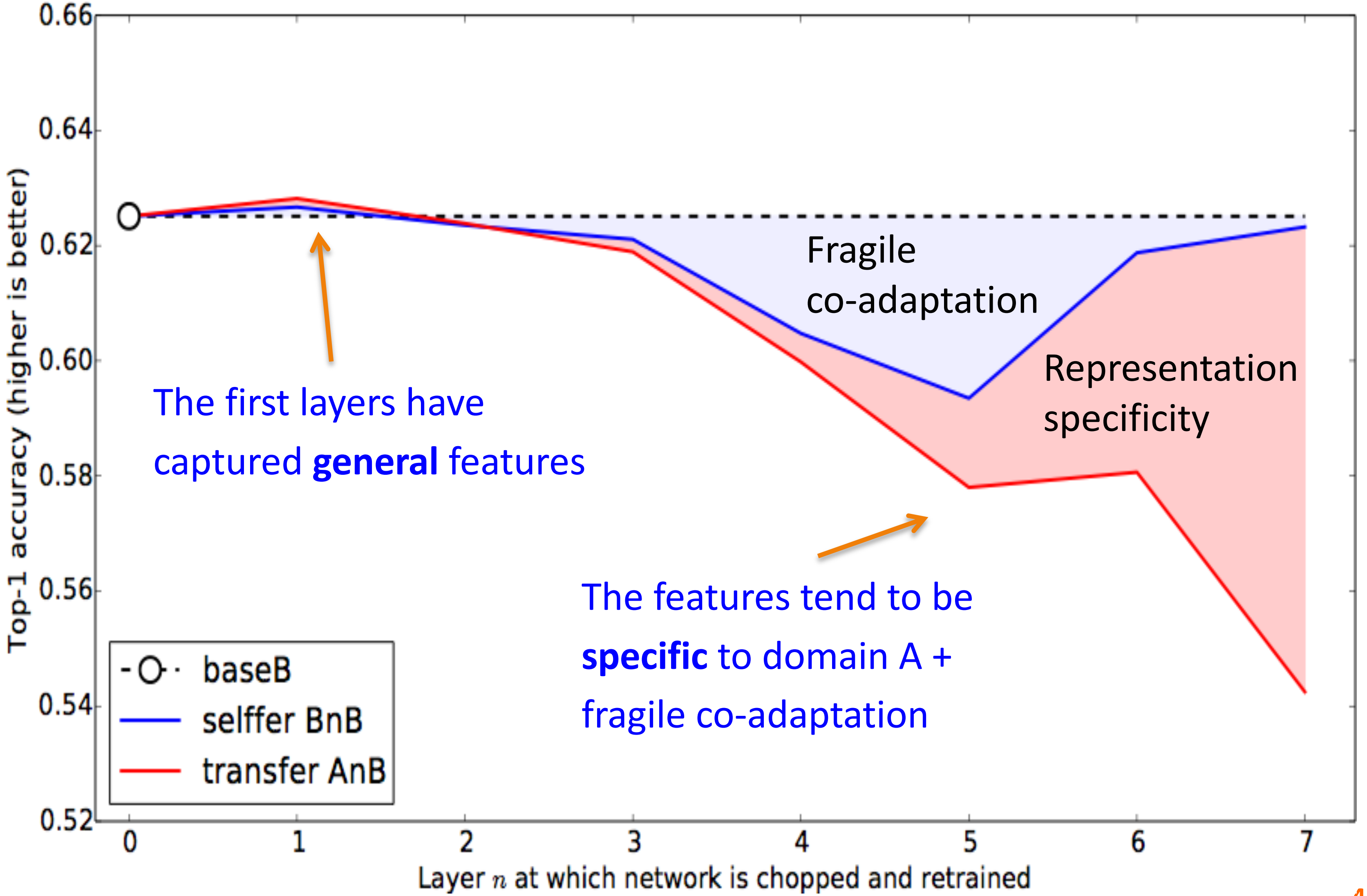
Interpretation

!!??



Interpretation

!!??



…

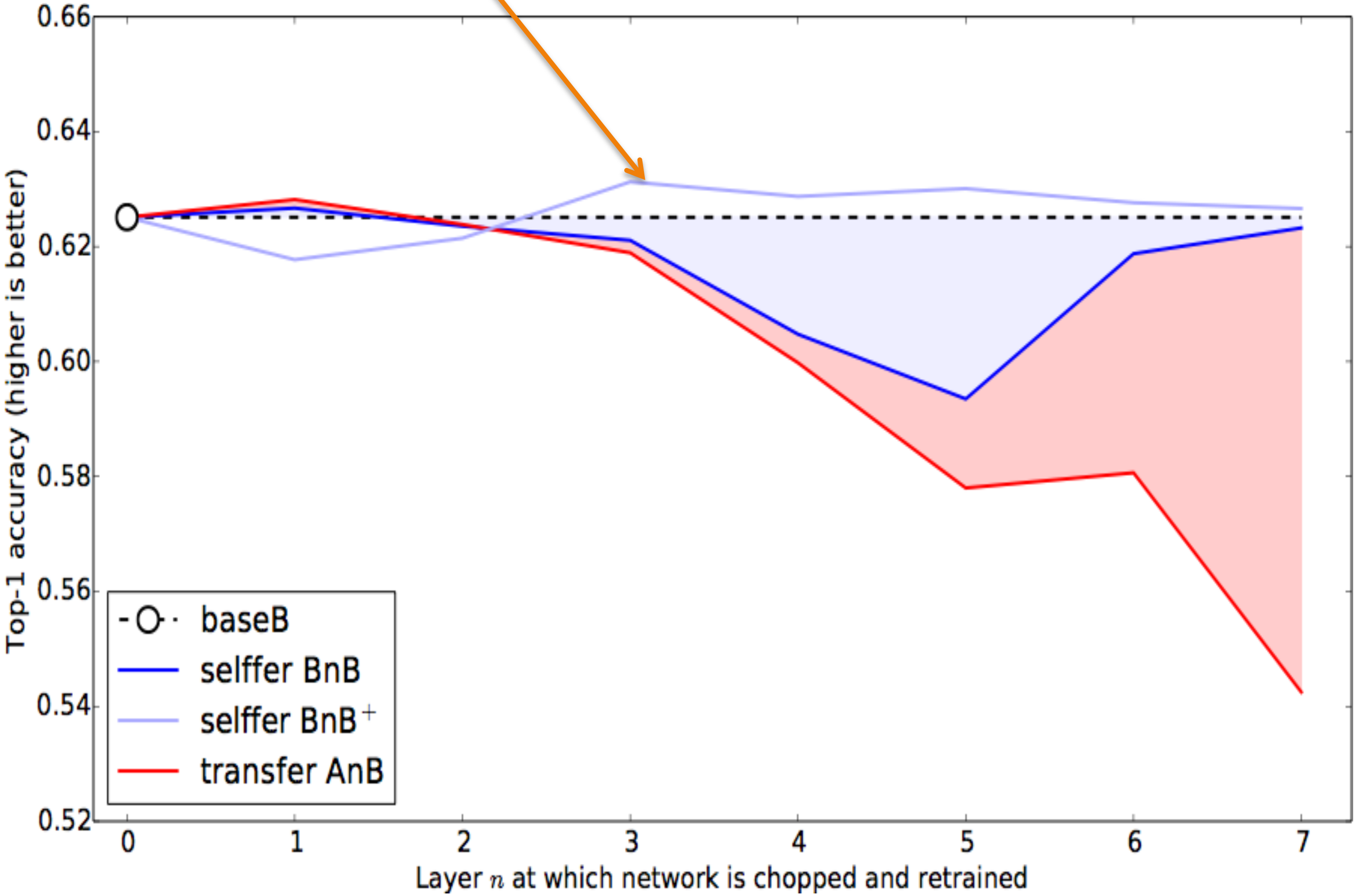
- Remark on the **scientific methodology**

It was essential to look at “*fragile co-adaptation*”

in order to assess the **true effect** of “*representation specificity*”

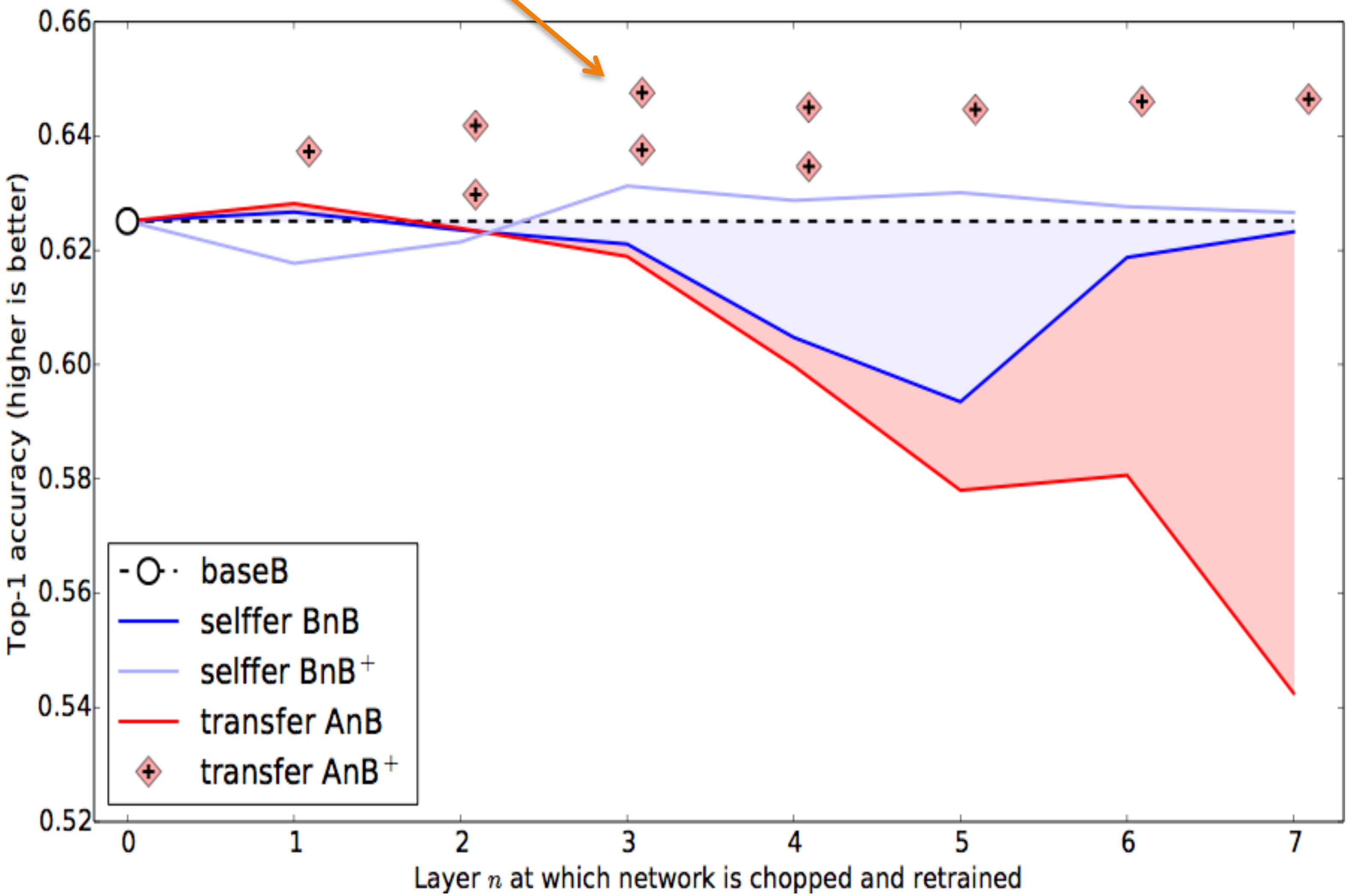
Interpretation

Retrain on all layers (fine-tuning) on domain B

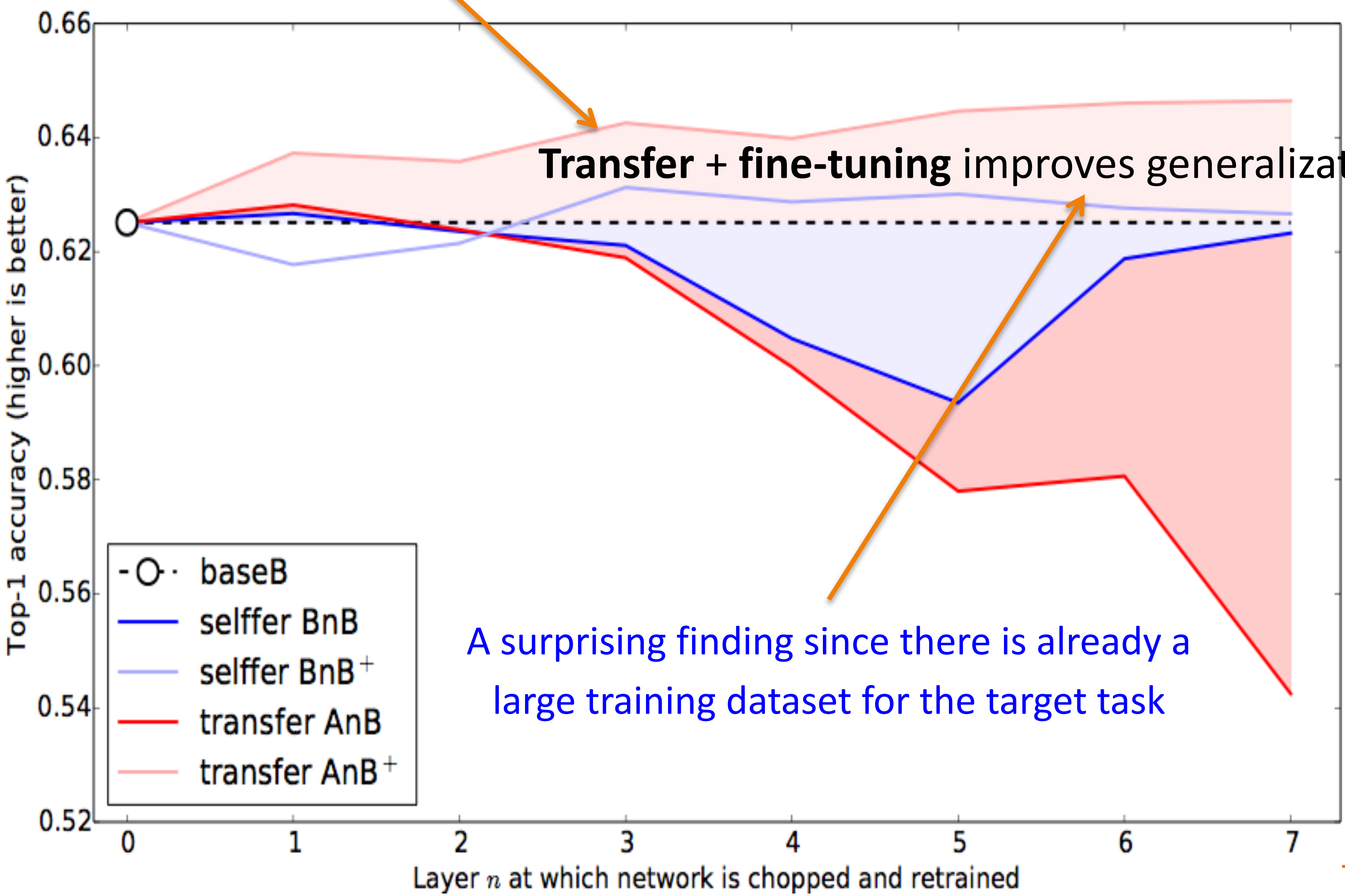


Interpretation

Retrain on all layers (fine-tuning) on domain B **after transfer** from domain A



Retrain on all layers (fine-tuning) on domain B **after transfer** from domain A



Transfer + **fine-tuning** improves generalization

A surprising finding since there is already a large training dataset for the target task

Conclusions of the paper

1. Be **careful** to separate effects
   - Fragile **co-adapted** first layers
   - **Specialization** of higher layers

2. The transferability gap grows as the **distance** between tasks increases

3. But even **features transfered** from distant tasks **are better** than random weights

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). **How transferable are features in deep neural networks?.** *Advances in neural information processing systems, 27.*

- ImageNet has many categories

Dataset A: random

gecko

fire truck

baseball

panther

rabbit

gorilla

Dataset B: random

garbage truck

toucan

radiator

binoculars

lion

bookshop

http://slideplayer.com/slide/8370683/

- ImageNet has many categories

Dataset A: man-made

fire truck

radiator

baseball

binoculars

bookshop

Dissimilar

Dataset B: natural

gorilla

gecko

toucan

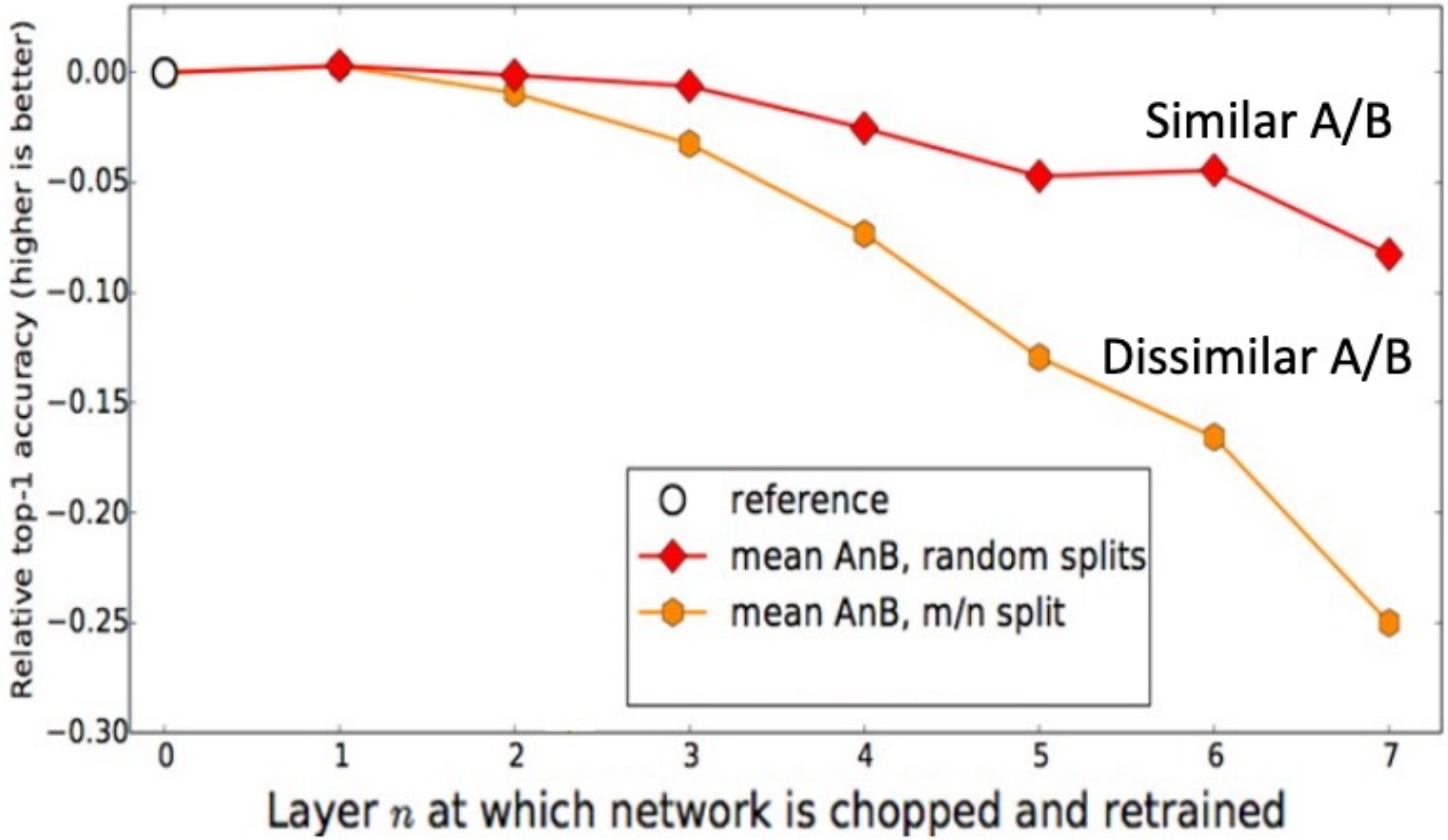
rabbit

panther

lion

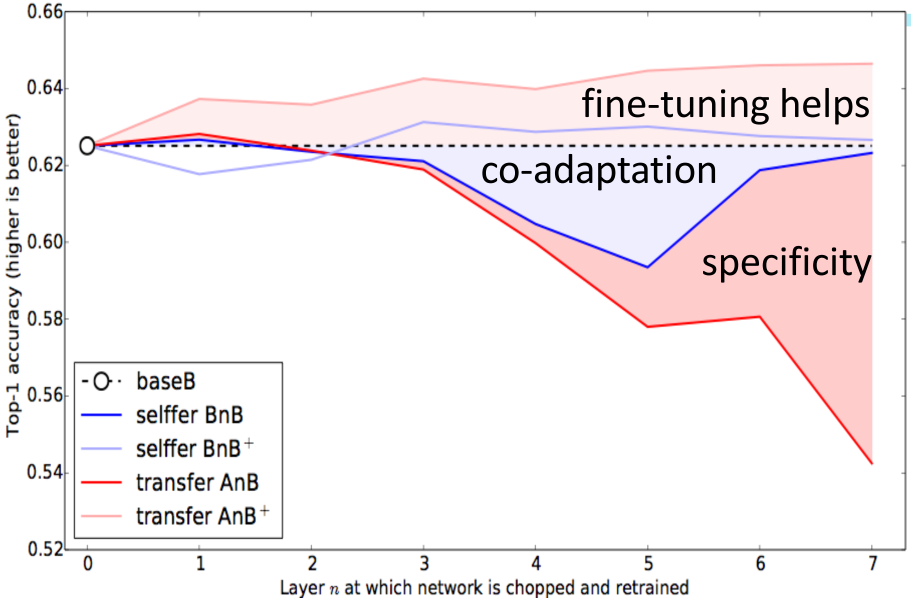
http://slideplayer.com/slide/8370683/

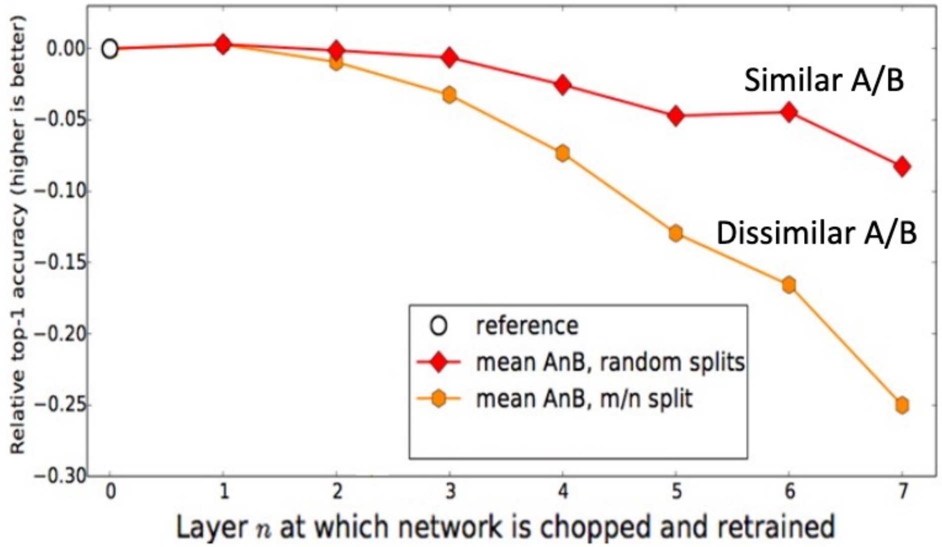
- Comparison



http://slideplayer.com/slide/8370683/

Conclusions





- Transferability governed by:
  - **lost** co-adaptations
  - **specificity**
  - **difference** between base and target dataset
- Fine-tuning helps even on large target dataset

Transfer learning with **language data**

- For texts in different
  - **Domains** (e.g. finance, politics, society, ...)
  - **Media** (e.g. journals, blogs, ...)
- A word embedding is used
  - A **mapping** of the words to a **high-dimensional** (e.g. 500) **continuous vector space** where different words with similar meanings have a similar vector representation
- There exit **pre-trained models** trained on very large corpus of text documents
  - Google word2vec
  - Stanford Glove model

Outline

1. Transfer learning: questions
2. Transfer learning in neural networks
3. TransBoost: an algorithm and what it tells on the role of the source
4. Curriculum learning and the geometry of the space of learning tasks
5. How to measure the difficulty of a training example
6. Conclusions

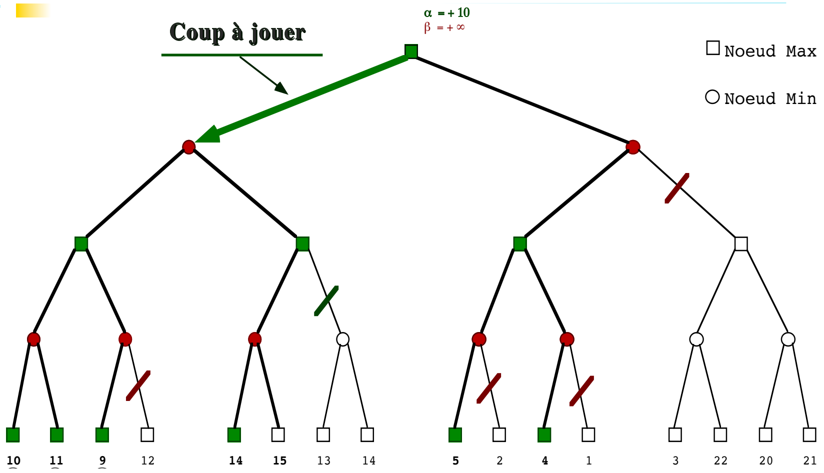
TransBoost: an algorithm for **transfer learning**

And what it tells about the **role of the source**

Cornuéjols, A. (2024). **Some thoughts about Transfer learning. What role for the source domain**. *International journal of Approximate Reasoning (IJAR)*, vol. 166, p.109107. Elsevier.

A **LUPI** type of algorithm for **transfer** learning

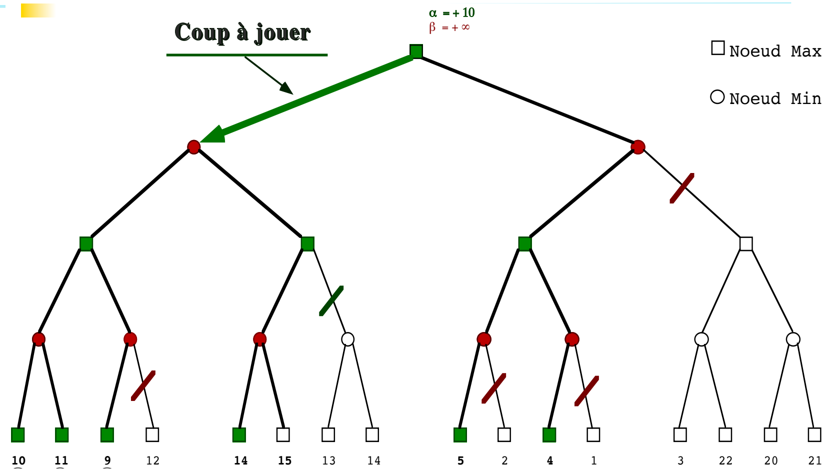
Taking decision when the current information is **incomplete**



…

Algorithms for games

Taking decision when the current information is **incomplete**



- Which move to play?

  The evaluation function is **insufficiently informed** at the root (current situation)

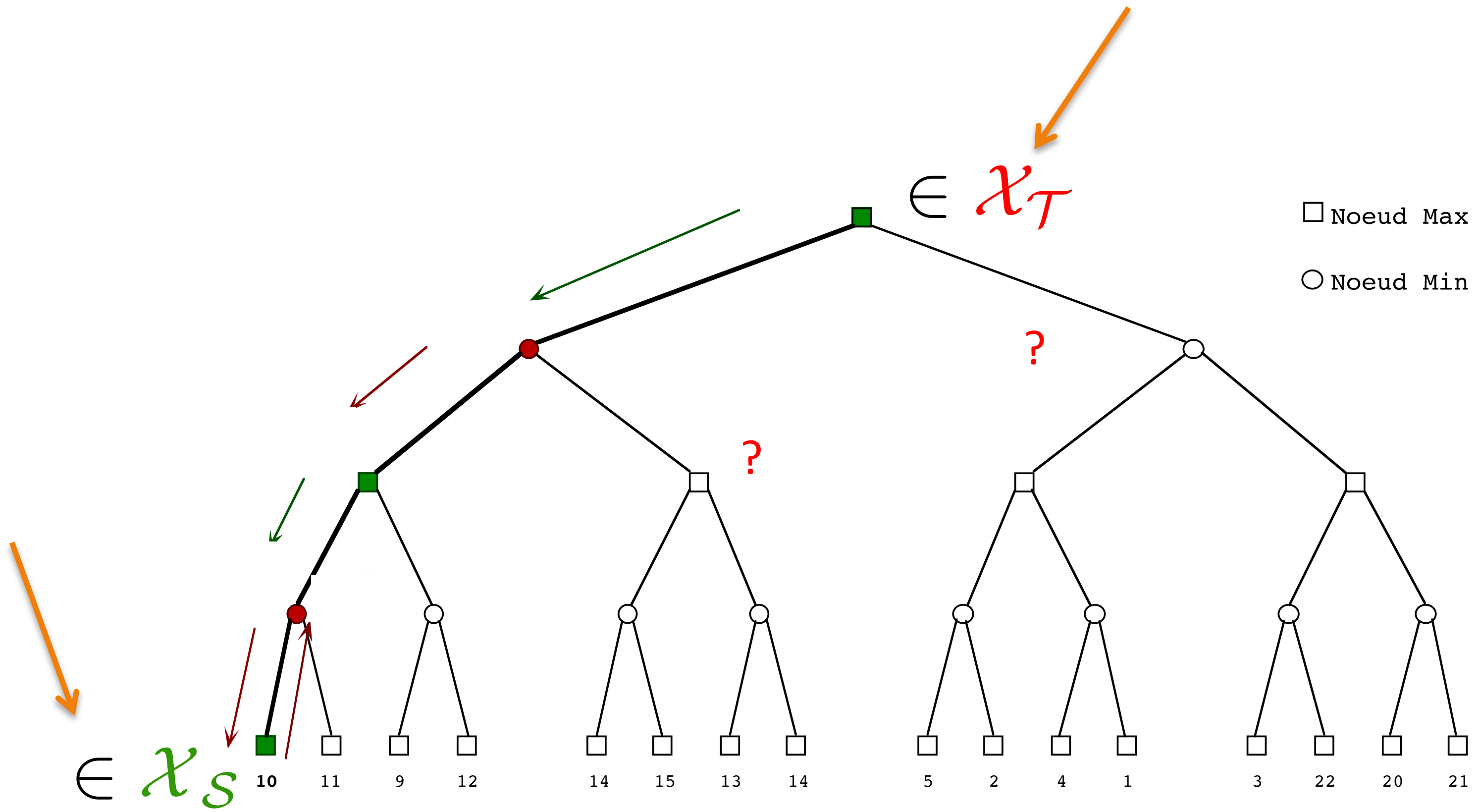
  1. **Query experts** that have more information about potential outcomes

  2. **Combination** of the estimates through MinMax

*"Experts" may live in **input spaces** that are **different***

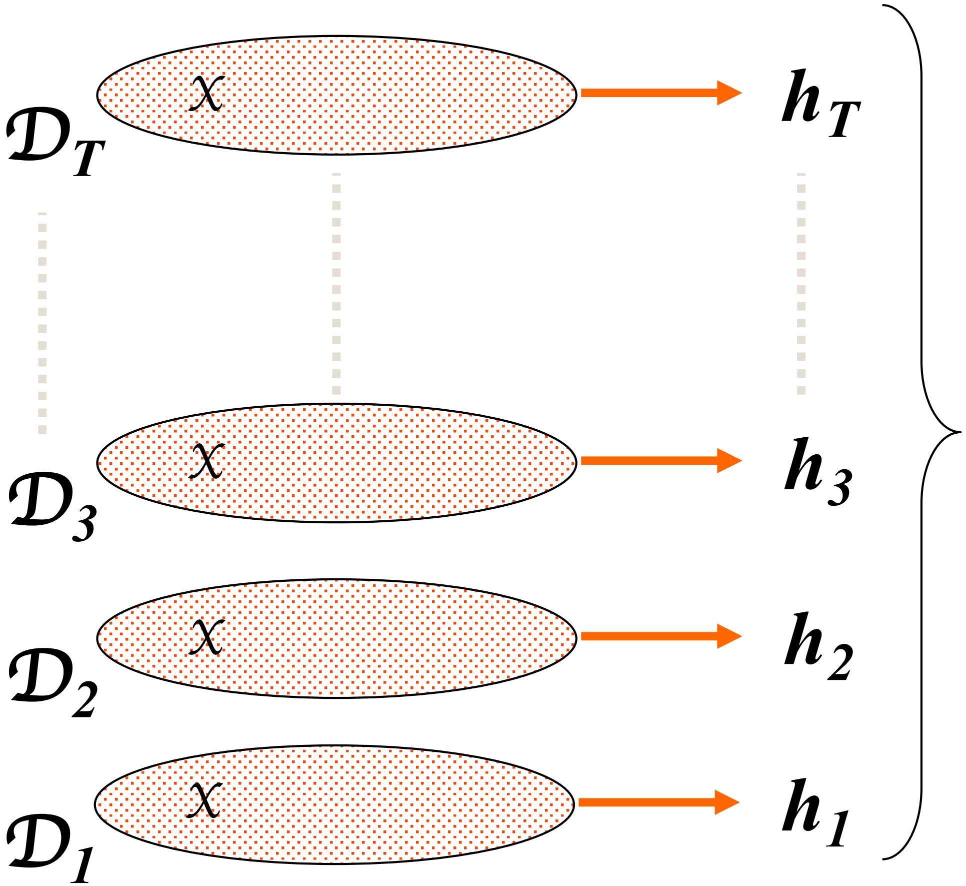
Algorithms for games and transfer learning

...



Can we do the “same” for transfer learning?

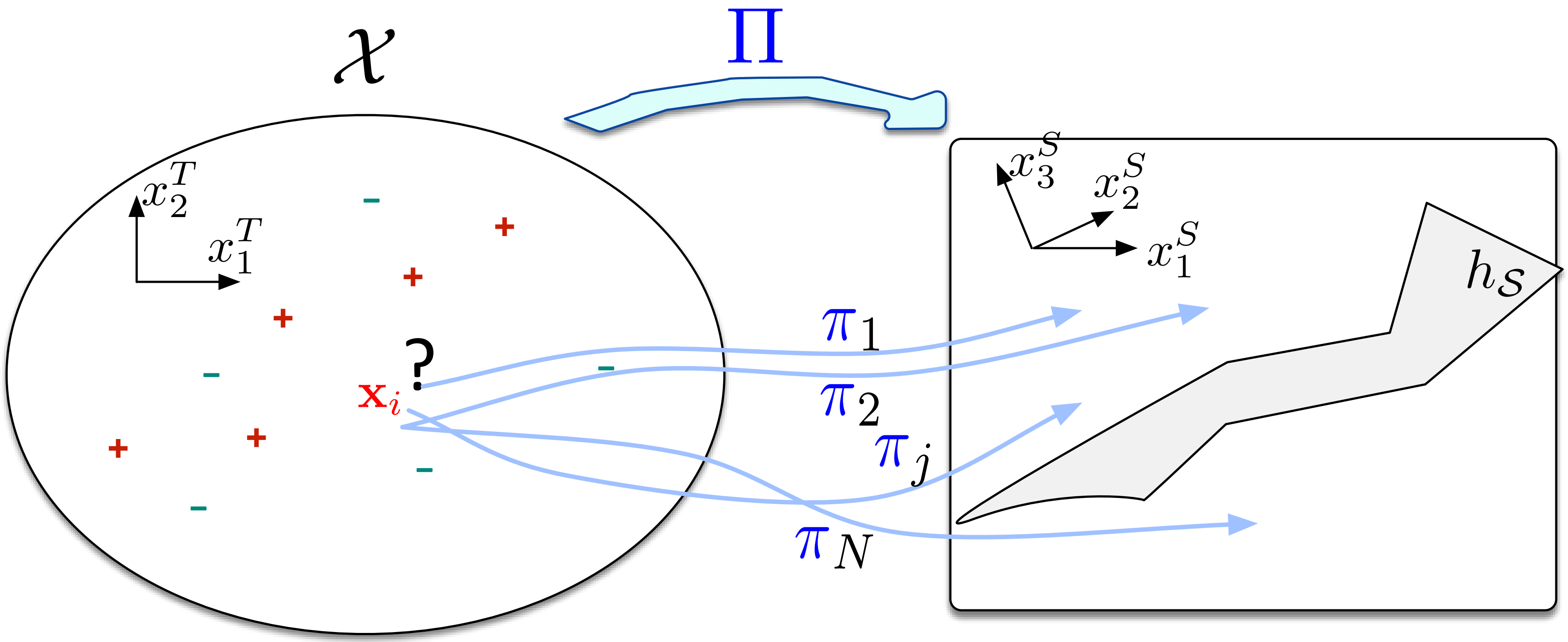
Boosting



$$H(\mathbf{x}) \;=\; \mathrm{sign}\left[\sum_{t=1}^{T} \alpha_t \, h_t(\mathbf{x})\right]$$

- How to compute $\mathcal{D}_t$ from $\mathcal{D}_{t-1}$ and thus h_t?
- How to compute the α_t ?

TransBoost



***Target** Domain*

***Source** Domain*

$$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \text{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\big(\pi_n(\mathbf{x}^{\mathcal{T}})\big)\right\}$$

TransBoost

- Principle:
  - Learn "*weak projections*": $\pi_i : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$
    - Using the target training data: $S_{\mathcal{T}} = \{(\mathbf{x}_i^{\mathcal{T}}, y_i^{\mathcal{T}})\}_{1 \leq i \leq m}$
  - With boosting
    - **Projection** π_n such that : $\varepsilon_n \doteq \mathbf{P}_{i \sim D_n}[h_{\mathcal{S}}(\pi_n(\mathbf{x}_i)) \neq y_i] < 0.5$
    - **Re-weight** the training time series and loop until termination
  - Result

$$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \operatorname{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}(\pi_n(\mathbf{x}^{\mathcal{T}}))\right\}$$

TransBoost

Algorithm 1: Transfer learning by boosting

Input: $h_{\mathcal{S}} : \mathcal{X}_{\mathcal{S}} \rightarrow \mathcal{Y}_{\mathcal{S}}$ the source hypothesis
$\mathcal{S}_{\mathcal{T}} = \{(\mathbf{x}_i^{\mathcal{T}}, y_i^{\mathcal{T}}\}_{1 \leq i \leq m}$: the target training set

Initialization of the distribution on the training set: $D_1(i) = 1/m$ for $i = 1, \ldots, m$;

for $n = 1, \ldots, N$ **do**

Find a projection $\pi_i : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$ st. $h_{\mathcal{S}}(\pi_i(\cdot))$ performs better than random on $D_n(\mathcal{S}_{\mathcal{T}})$;

Let ε_n be the error rate of $h_{\mathcal{S}}(\pi_i(\cdot))$ on $D_n(\mathcal{S}_{\mathcal{T}})$: $\varepsilon_n \doteq \mathbf{P}_{i \sim D_n}[h_{\mathcal{S}}(\pi_n(\mathbf{x}_i)) \neq y_i]$ (with $\varepsilon_n < 0.5$) ;

Computes $\alpha_i = \frac{1}{2} \log_2\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$;

Update, for $i = 1 \ldots, m$:

$$D_{n+1}(i) = \frac{D_n(i)}{Z_n} \times \begin{cases} e^{-\alpha_n} & \text{if } h_{\mathcal{S}}\big(\pi_n(\mathbf{x}_i^{\mathcal{T}})\big) = y_i^{\mathcal{T}} \\ e^{\alpha_n} & \text{if } h_{\mathcal{S}}\big(\pi_n(\mathbf{x}_i^{\mathcal{T}})\big) \neq y_i^{\mathcal{T}} \end{cases}$$
$$= \frac{D_n(i) \exp\big(-\alpha_n\, y_i^{(\mathcal{T})}\, h_{\mathcal{S}}(\pi_n(\mathbf{x}_i^{(\mathcal{T})}))\big)}{Z_n}$$

where Z_n is a normalization factor chosen so that D_{n+1} be a distribution on $\mathcal{S}_{\mathcal{T}}$;

end

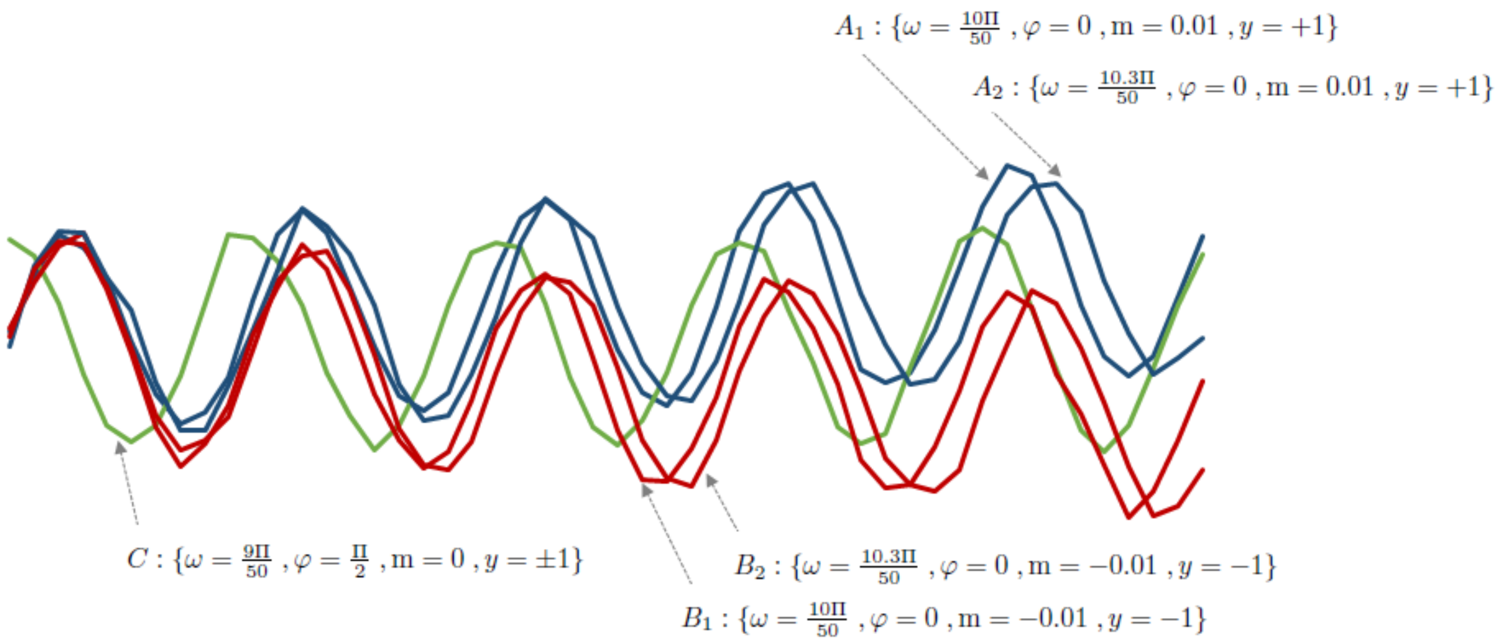
Output: the final target hypothesis $H_{\mathcal{T}} : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{Y}_{\mathcal{T}}$:

$$\boxed{H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \operatorname{sign}\Big\{\sum_{n=1}^{N} \alpha_n\, h_{\mathcal{S}}\big(\pi_n(\mathbf{x}^{\mathcal{T}})\big)\Big\}} \tag{2}$$

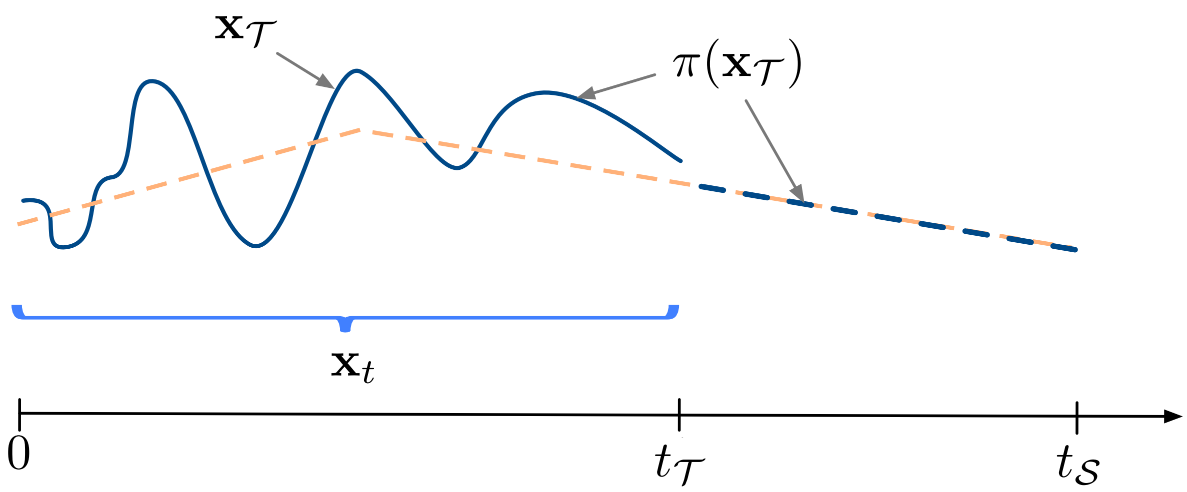
Controlled data

- The **slope** to distinguish between **classes**
- The **shapes** of time series within each class: variety
- The **noise level**

$$\mathbf{x}_t = \underbrace{t \times \text{slope} \times \text{class}}_{\text{information gain}} + \underbrace{\mathrm{x}_{max} \sin(\omega_i \times t + \varphi_j)}_{\text{sub shape within class}} + \underbrace{\eta(t)}_{\text{noise factor}}$$



The set of projections



Example of a projection π (a hinge function with three parameters):

- the first slope,
- the second one
- and the time of the hinge) that is adjusted to the target exemple x_T by least square.

The resulting projection $\pi(x_T)$ is the concatenation of x_T and the remaining part of the adjusted hinge function.

Results

Learning from **target data only**

TransBoost

Naïve transfer

First a projection from X_T to X_S by SVR then using h_S

Very little information in the source

Increasing information in the source

Increasing level of noise

slope, noise, $t_{\mathcal{T}}$	SVM (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)	SVR+SVM (test)
0.001, 0.001, **20**	0.50 ± 0.08	0.08 ± 0.03	**0.08** ± 0.02	0.49 ± 0.01
0.005, 0.001, **20**	0.49 ± 0.01	0.01 ± 0.01	**0.01** ± 0.01	0.45 ± 0.01
0.005, 0.002, **20**	0.49 ± 0.03	0.03 ± 0.02	**0.04** ± 0.02	0.43 ± 0.01
0.005, 0.020, **20**	0.48 ± 0.03	0.09 ± 0.01	**0.10** ± 0.01	0.47 ± 0.01
0.001, 0.200, **20**	0.50 ± 0.01	0.46 ± 0.02	0.51 ± 0.02	0.49 ± 0.01
0.010, 0.200, **20**	0.47 ± 0.03	0.34 ± 0.02	0.35 ± 0.02	0.35 ± 0.01
0.001, 0.001, **50**	0.50 ± 0.01	0.08 ± 0.03	**0.08** ± 0.02	0.41 ± 0.01
0.005, 0.001, **50**	0.28 ± 0.09	0.01 ± 0.01	**0.01** ± 0.01	0.28 ± 0.01
0.005, 0.002, **50**	0.30 ± 0.08	0.02 ± 0.01	**0.02** ± 0.01	0.28 ± 0.01
0.005, 0.020, **50**	0.30 ± 0.08	0.04 ± 0.01	**0.04** ± 0.01	0.31 ± 0.01
0.001, 0.200, **50**	0.50 ± 0.01	0.38 ± 0.03	0.44 ± 0.02	0.43 ± 0.01
0.010, 0.200, **50**	0.12 ± 0.04	0.10 ± 0.02	0.11 ± 0.02	0.15 ± 0.02
0.001, 0.001, **100**	0.47 ± 0.03	0.07 ± 0.02	**0.07** ± 0.02	0.23 ± 0.01
0.005, 0.001, **100**	0.07 ± 0.03	0.01 ± 0.01	**0.01** ± 0.01	0.07 ± 0.02
0.005, 0.002, **100**	0.10 ± 0.04	0.02 ± 0.01	**0.02** ± 0.01	0.07 ± 0.01
0.005, 0.020, **100**	0.09 ± 0.03	0.02 ± 0.01	**0.03** ± 0.01	0.07 ± 0.01
0.001, 0.200, **100**	0.46 ± 0.02	0.28 ± 0.02	0.31 ± 0.01	0.31 ± 0.01
0.010, 0.200, **100**	0.05 ± 0.02	0.04 ± 0.01	0.05 ± 0.01	0.05 ± 0.01

Lots of information in the source and lots of noise

Results

Learning from **target data only**

TransBoost

On the source domain

Naïve transfert

High noise level

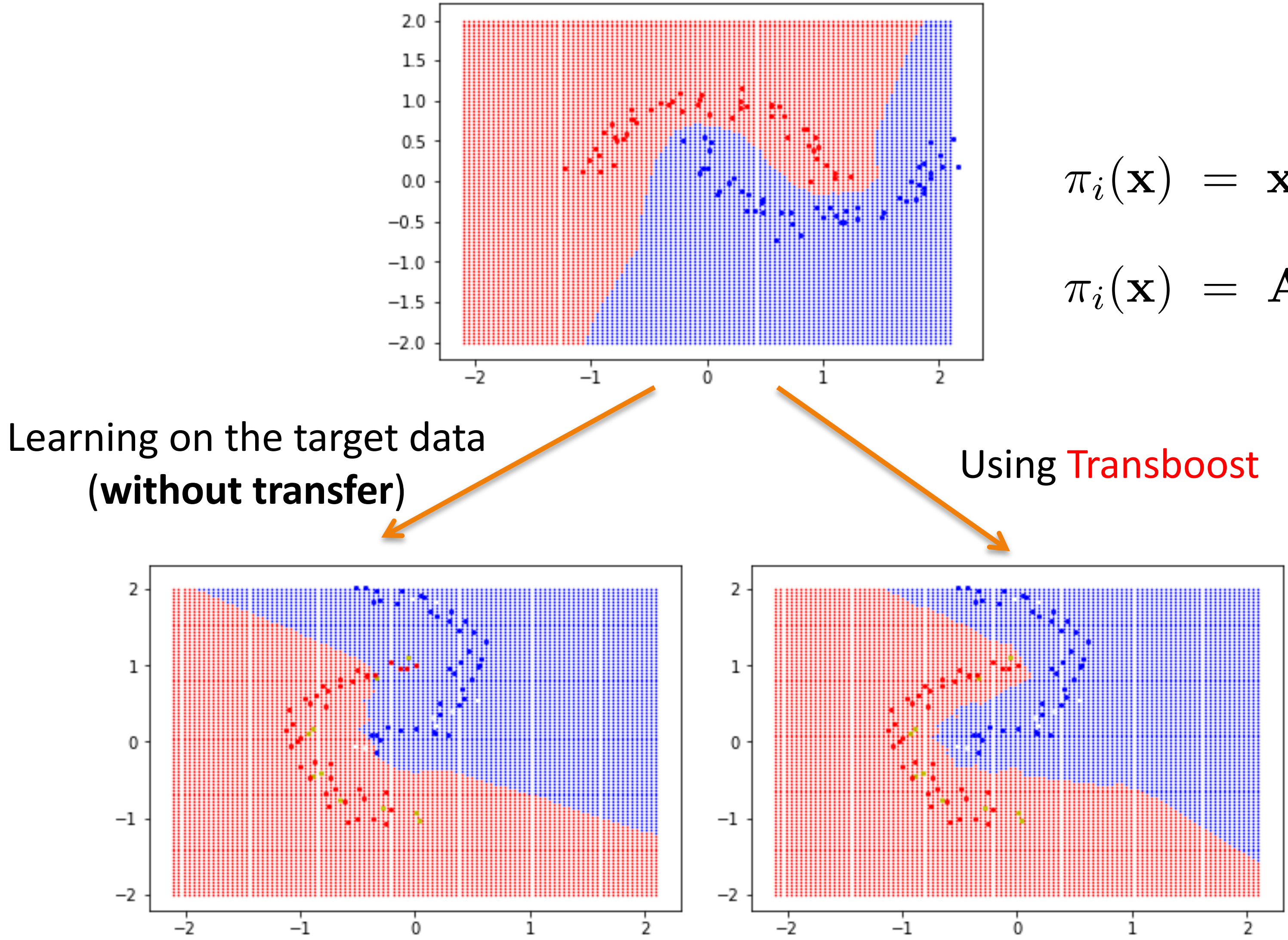
Easy

Large slope

slope, noise, $t_{\mathcal{T}}$	$h_{\mathcal{T}}$ (train)	$h_{\mathcal{T}}$ (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)	$h_{\mathcal{S}}$ (test)	$H'_{\mathcal{T}}$ (test)
0.001, 0.001, 20	0.46 ± 0.02	0.50 ± 0.08	0.08 ± 0.03	$\mathbf{0.08} \pm 0.02$	0.05	0.49 ± 0.01
0.005, 0.001, 20	0.46 ± 0.02	0.49 ± 0.01	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.45 ± 0.01
0.005, 0.002, 20	0.46 ± 0.02	0.49 ± 0.03	0.03 ± 0.02	$\mathbf{0.04} \pm 0.02$	0.02	0.43 ± 0.01
0.005, 0.02, 20	0.44 ± 0.02	0.48 ± 0.03	0.09 ± 0.01	$\mathbf{0.10} \pm 0.01$	0.01	0.47 ± 0.01
0.001, 0.2, 20	0.46 ± 0.02	0.50 ± 0.01	0.46 ± 0.02	0.51 ± 0.02	0.11	0.49 ± 0.01
0.01, 0.2, 20	0.42 ± 0.03	0.47 ± 0.03	0.34 ± 0.02	0.35 ± 0.02	0.02	0.35 ± 0.01
0.001, 0.001, 50	0.46 ± 0.02	0.50 ± 0.01	0.08 ± 0.03	$\mathbf{0.08} \pm 0.02$	0.06	0.41 ± 0.01
0.005, 0.001, 50	0.25 ± 0.07	0.28 ± 0.09	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.28 ± 0.01
0.005, 0.002, 50	0.27 ± 0.07	0.30 ± 0.08	0.02 ± 0.01	$\mathbf{0.02} \pm 0.01$	0.02	0.28 ± 0.01
0.005, 0.02, 50	0.26 ± 0.07	0.30 ± 0.08	0.04 ± 0.01	$\mathbf{0.04} \pm 0.01$	0.01	0.31 ± 0.01
0.001, 0.2, 50	0.44 ± 0.02	0.50 ± 0.01	0.38 ± 0.03	0.44 ± 0.02	0.15	0.43 ± 0.01
0.01, 0.2, 50	0.10 ± 0.03	0.12 ± 0.04	0.10 ± 0.02	0.11 ± 0.02	0.03	0.15 ± 0.02
0.001, 0.001, 100	0.43 ± 0.03	0.47 ± 0.03	0.07 ± 0.02	$\mathbf{0.07} \pm 0.02$	0.02	0.23 ± 0.01
0.005, 0.001, 100	0.06 ± 0.03	0.07 ± 0.03	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.07 ± 0.02
0.005, 0.002, 100	0.08 ± 0.03	0.10 ± 0.04	0.02 ± 0.01	$\mathbf{0.02} \pm 0.01$	0.02	0.07 ± 0.01
0.005, 0.02, 100	0.08 ± 0.03	0.09 ± 0.03	0.02 ± 0.01	$\mathbf{0.03} \pm 0.01$	0.01	0.07 ± 0.01
0.001, 0.2, 100	0.04 ± 0.03	0.46 ± 0.02	0.28 ± 0.02	0.31 ± 0.01	0.16	0.31 ± 0.01
0.01, 0.2, 100	0.03 ± 0.01	0.05 ± 0.02	0.04 ± 0.01	0.05 ± 0.01	0.02	0.05 ± 0.01

Table 1: Comparison of learning directly in the target domain (columns $h_{\mathcal{T}}$ (train) and $h_{\mathcal{T}}$ (test)), using `TransBoost` (columns $H_{\mathcal{T}}$ (train) and $H_{\mathcal{T}}$ (test)), learning in the source domain (column $h_{\mathcal{S}}$ (test)) and, finally, completing the time series with a SVR regression and using $h_{\mathcal{S}}$ (naïve transfer). Test errors are highlighted in the orange columns. Bold numbers indicates where `TransBoost` significantly dominates both learning without transfer and learning with naïve transfer.

Transfer learning using Transboost

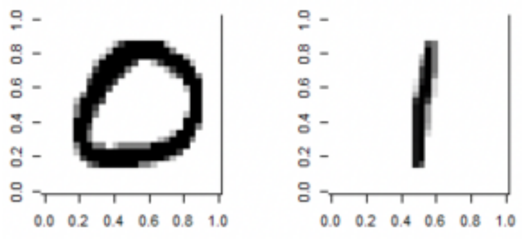


$$\pi_i(\mathbf{x}) \;=\; \mathbf{x} \;+\; \mathbf{v}_i$$

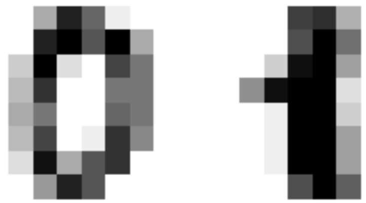
$$\pi_i(\mathbf{x}) \;=\; \mathbf{A}_i \cdot \mathbf{x} \;+\; \mathbf{v}_i$$

Transfer learning using Transboost

- Illustrations

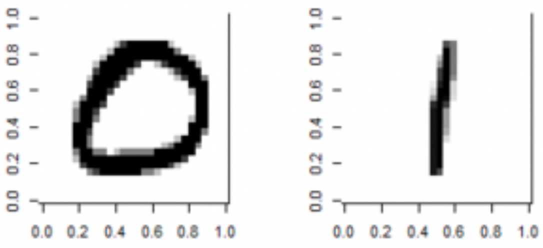


(a) Is it a zero or a one ?

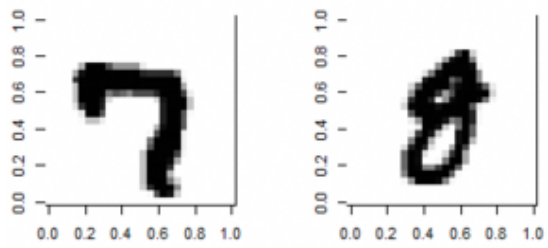


(b) Is it a zero or a one ?

Figure 15: Transfer learning of the source model 0/1 mnist so that it can distinguish 0/1 sklearn digits



(a) Is it a zero or a one ?



(b) Is it an eight or a seven ?

Transfer learning using Transboost

- Illustrations

FIGURE 1: Trained model on the data source : is it a picture of a dog or a cat ?

Task A

$$\mathcal{X}_A \neq \mathcal{X}_B$$

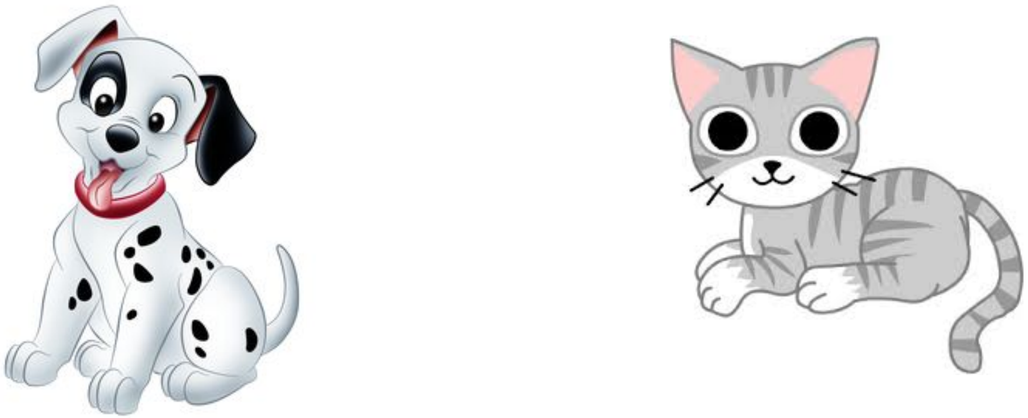
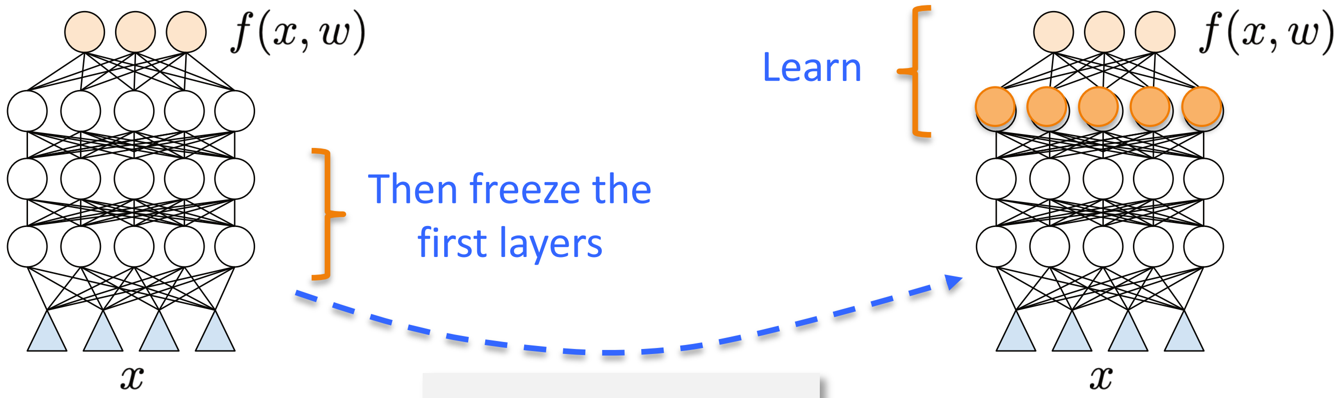


FIGURE 2: Model source transferred on the data target : is it a clip-art of a dog or a cat ?

Task B

Standard Transfer with NNs

...



Transferring the **features**

Learn NN on task A

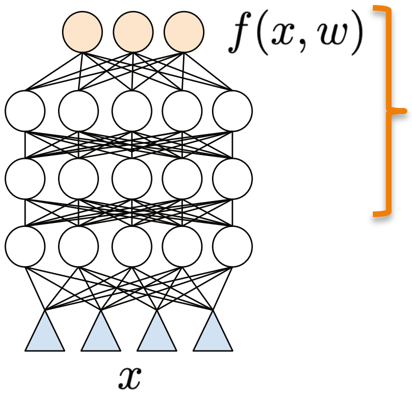
Learn the **last** layers on task B

Same input space $\mathcal{X}_A = \mathcal{X}_B$

From Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). **Learning and transferring mid-level image representations using convolutional neural networks**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717-1724).

TransBoost with NNs

...

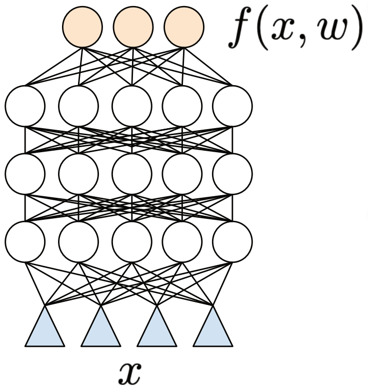


Then **freeze all layers** except the first (and second)

Learn NN on task A

TransBoost with NNs

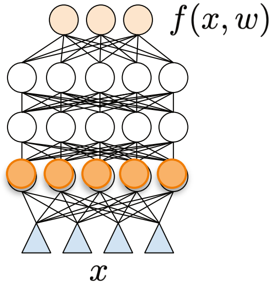
...



Transferring the **decision function**

Then **freeze all layers** except the first (and second)

Learn projection π_1

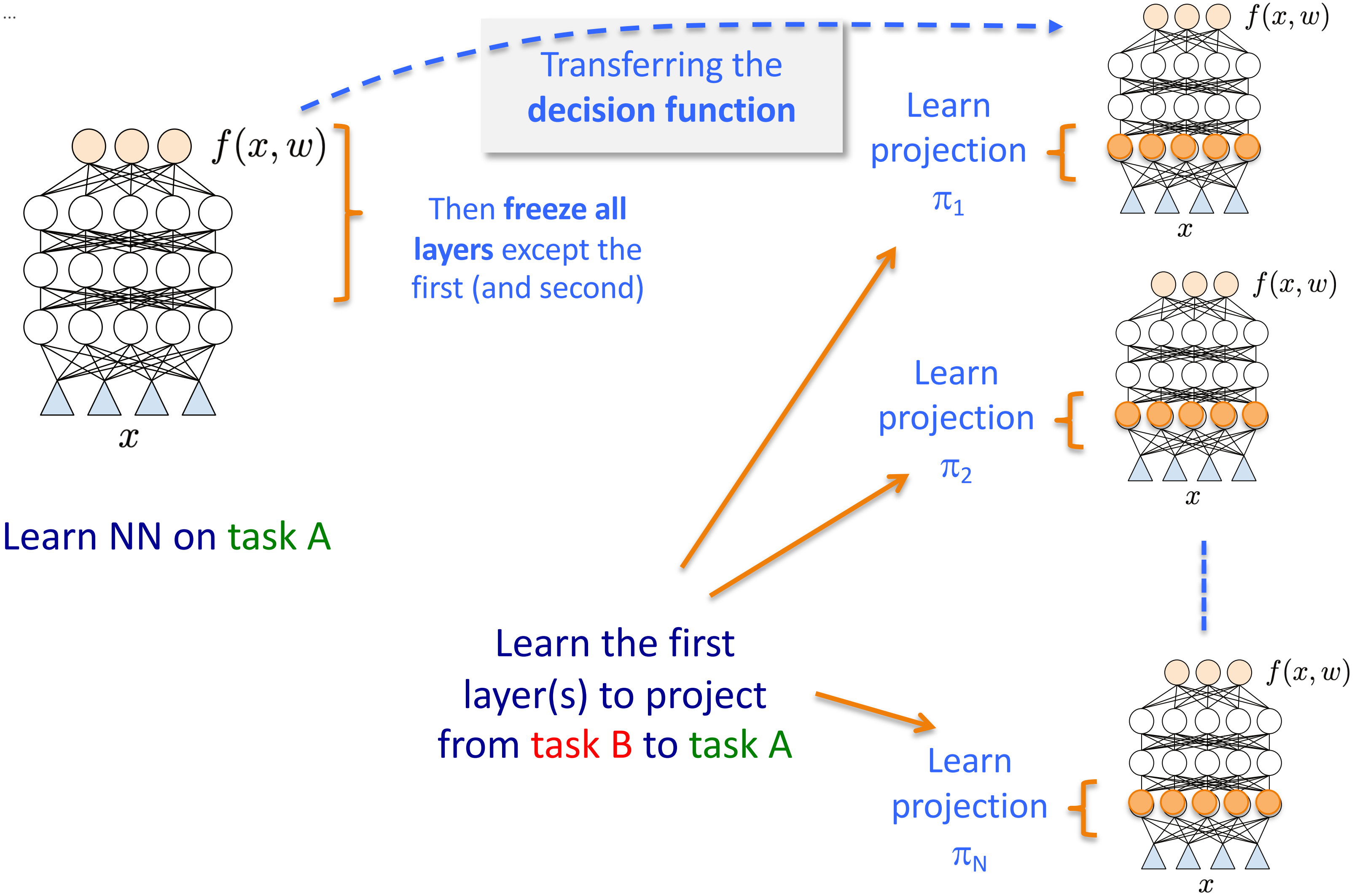


Learn NN on task A

Learn the first layer(s) to project from task B to task A

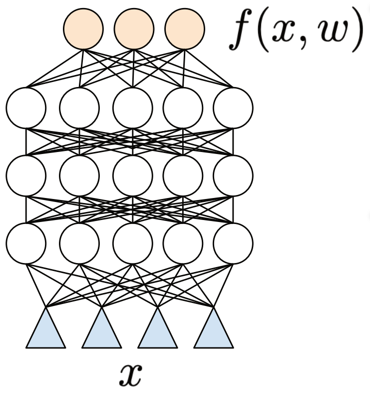
TransBoost with NNs

...



TransBoost with NNs

...

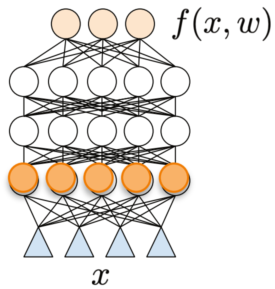


Transferring the **decision function**

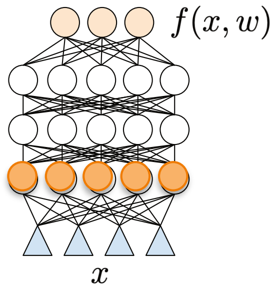
Then **freeze all layers** except the first (and second)

Learn NN on task A

Learn projection π_1

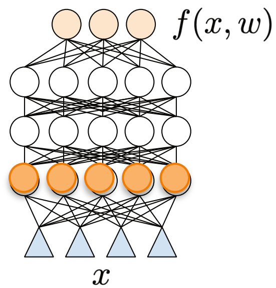


Learn projection π_2



Learn the first layer(s) to project from task B to task A

Learn projection π_N



Different input spaces

$\mathcal{X}_A \neq \mathcal{X}_B$

$$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \text{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\big(\pi_n(\mathbf{x}^{\mathcal{T}})\big)\right\}$$

Does the quality of h_S plays a **role**?

What if ...

Source hypothesis a priori **without relation** to the target task

Learning **from target data only** | TransBoost with "**irrelevant**" source hypothesis

	slope, noise, $t_{\mathcal{T}}$	$h_{\mathcal{T}}$ (train)	$h_{\mathcal{T}}$ (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)
	0.001, 0.001, 70	0.44 ± 0.02	0.48 ± 0.02	0.06 ± 0.02	$\mathbf{0.06} \pm 0.02$
	0.005, 0.005, 70	0.11 ± 0.04	0.13 ± 0.05	0.02 ± 0.01	$\mathbf{0.02} \pm 0.02$
	0.005, 0.005, 70	0.10 ± 0.04	0.11 ± 0.05	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$
	0.005, 0.05, 70	0.11 ± 0.04	0.12 ± 0.05	0.04 ± 0.02	$\mathbf{0.03} \pm 0.01$
Hard	0.001, 0.001, 70	0.42 ± 0.03	0.48 ± 0.02	0.33 ± 0.02	$\mathbf{0.37} \pm 0.02$
	0.01, 0.1, 70	0.06 ± 0.03	0.08 ± 0.03	0.08 ± 0.02	0.08 ± 0.02

Very good results!!

h_S **randomly chosen** on the source task $\widehat{R}(h_{\mathcal{S}}) \approx 0.5$

Does the quality of h_S plays a **role**? NO!!

What is the **role** of h_S??

Analysis

- The **quality of the source hypothesis** on the source data?
  - Plays no role

- The **proximity of the source and target** distributions P_X and P_Y?
  - Plays no role

But… !?

=> *No condition on the source!??*

Still some transfer learning problems

appear to us **more easy than others**???

Interpretation

Transfer acts as a bias and h_S is a strong part of this bias

- **If** the **source hypothesis** is **well chosen**: the **bias** is well informed
    - Which **does not mean** that h_S must be good on the source task
- **Otherwise**: Learning is **badly directed**

  or there is **over-fitting** if the capacity of $h_{\mathcal{S}} \circ \pi$ is too large

Lessons

- The learning problem now becomes the problem of **choosing** a good set of (weak) projections

- Theoretical guarantees exist

Analysis

- The **generalization properties** of TransBoost

  can be imported from the ones for **boosting**

$$\mathcal{H}_{\mathcal{T}} \;=\; \left\{ \mathtt{sign}\left[\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}} \circ \pi_n\right] \middle| \alpha_n \in \mathbb{R}, \pi_n \in \Pi, n \in [1, N] \right\}$$

$$d_{\mathtt{VC}}(\mathcal{H}_{\mathcal{T}}) \;\leq\; 2(d_{h_{\mathcal{S}} \circ \Pi} + 1)(N+1) \log_2\big((N+1)\, e\big)$$

$$R(h) \;\leq\; \widehat{R}(h) \;+\; \mathcal{O}\left(\sqrt{\frac{d_{h_{\mathcal{S}} \circ \Pi} \;\ln(m_{\mathcal{T}} / d_{h_{\mathcal{S}} \circ \Pi}) + \ln(1/\delta)}{m_{\mathcal{T}}}}\right)$$

Theory for HTL

$$h(\mathbf{x}) := \langle \hat{\boldsymbol{w}}, \mathbf{x} \rangle$$

$$\hat{\boldsymbol{w}} = \underset{\mathbf{w}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \frac{1}{m}\sum_{i=1}^{m}(\langle \hat{\boldsymbol{w}}, \mathbf{x}_i\rangle - y_i)^2 + \lambda\|\mathbf{w} - \sum_{j=1}^{n}\beta_j \mathbf{w}_{\text{src}}^j\|_2^2 \right\}$$

THEOREM 7.3 ([KUZ 17]).– *Let $h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}$ a hypothesis output by a regularized ERM algorithm from a m-sized training set T i.i.d. from the target domain $\mathcal{T}$, n source hypotheses $\{h^i_{src} : \|h^i_{src}\|_\infty \leq 1\}_{i=1}^n$, any source weights $\boldsymbol{\beta}$ obeying $\Omega(\boldsymbol{\beta}) \leq \rho$ and $\lambda \in \mathbb{R}_+$. Assume that the loss is bounded by M: $\ell(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}(\mathbf{x}), y) \leq M$ for any $(\mathbf{x}, y)$ and any training set. Then, denote $\kappa = \frac{H}{\sigma}$ and assuming that $\lambda \leq \kappa$ with probability at least $1 - e^{-\eta}$, $\forall \eta \geq 0$:*

$$\mathrm{R}_{\mathcal{T}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) \leq \mathrm{R}_{\hat{\mathcal{T}}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{\mathrm{R}^{src}_{\mathcal{T}}\kappa}{\sqrt{m}\lambda} + \sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}}\rho\kappa^2}{m\lambda}} + \frac{M\eta}{m\log\left(1 + \sqrt{\frac{M\eta}{u^{src}}}\right)} \right)$$

$$\leq \mathrm{R}_{\hat{\mathcal{T}}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{\kappa}{\sqrt{m}}\left(\frac{\mathrm{R}^{src}_{\mathcal{T}}}{\lambda} + \sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}}\rho}{\lambda}} \right) + \frac{\kappa}{m}\left(\frac{\sqrt{\mathrm{R}^{src}_{\mathcal{T}}M\eta}}{\lambda} + \sqrt{\frac{\rho}{\lambda}} \right) \right),$$

where $u^{src} = \mathrm{R}^{src}_{\mathcal{T}}\left(m + \frac{\kappa\sqrt{m}}{\lambda}\right) + \kappa\sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}}m\rho}{\lambda}}$ and $\mathrm{R}^{src}_{\mathcal{T}} = \mathrm{R}_{\mathcal{T}}(h^{\boldsymbol{\beta}}_{src})$ is the risk of the source hypothesis combination.

Analysis

- The **generalization properties** of TransBoost can be imported from the ones for **boosting**

$$\mathcal{H}_{\mathcal{T}} \;=\; \left\{ \mathtt{sign}\left[\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}} \circ \pi_n\right] \middle| \alpha_n \in \mathbb{R}, \pi_n \in \Pi, n \in [1, N] \right\}$$

$$d_{\mathtt{VC}}(\mathcal{H}_{\mathcal{T}}) \;\leq\; 2(d_{h_{\mathcal{S}} \circ \Pi} + 1)(N+1) \log_2\big((N+1)\, e\big)$$

$$R(h) \;\leq\; \widehat{R}(h) \;+\; \mathcal{O}\left(\sqrt{\frac{d_{h_{\mathcal{S}} \circ \Pi} \;\ln(m_{\mathcal{T}} / d_{h_{\mathcal{S}} \circ \Pi}) + \ln(1/\delta)}{m_{\mathcal{T}}}}\right)$$

"Authors also present some theory, but at the moment, again, it is essentially a trivial extension of boosting theory. **TL bounds should incorporate the quality of the source hypothesis**, e.g. the risk of the source on \mathcal{D}_T."

Theoretical guarantees

$$\forall\, \widehat{h}_{\mathcal{S}} \in \mathcal{H}_{\mathcal{S}} : \quad \underset{\pi \in \Pi}{\text{Min}}\, R_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi) \;\leq\; \omega\big(R_{\mathcal{S}}(h_{\mathcal{S}})\big) \qquad (2)$$

where $\omega : \mathbb{R} \rightarrow \mathbb{R}$ is a non-decreasing function.

Theorem 1. *Let $\omega : \mathbb{R} \rightarrow \mathbb{R}$ be a non-decreasing function. Suppose that $P_{\mathcal{S}}$, $P_{\mathcal{T}}$, $h_{\mathcal{S}}$, $h_{\mathcal{T}} = \widehat{h}_{\mathcal{S}} \circ \pi (\pi \in \Pi)$, $\widehat{h}_{\mathcal{S}}$ and Π have the property given by Equation (2). Let $\widehat{\pi} := \text{ArgMin}_{\pi \in \Pi}\, \widehat{R}_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi)$, be the best apparent projection.*

Then, with probability at least $1 - \delta \quad (\delta \in (0,1))$ over pairs of training sets for tasks $\mathcal{S}$ and $\mathcal{T}$:

$$\begin{aligned} R_{\mathcal{T}}(\widehat{h}_{\mathcal{T}}) \;\leq\; \omega\big(\widehat{R}_{\mathcal{S}}(\widehat{h}_{\mathcal{S}})\big) \;+\; 2\sqrt{\frac{2\, d_{\mathcal{H}_{\mathcal{S}}} \log(2e m_{\mathcal{S}}/d_{\mathcal{H}_{\mathcal{S}}}) \;+\; 2\log(8/\delta)}{m_{\mathcal{S}}}} \\ +\; 4\sqrt{\frac{2\, d_{h_{\mathcal{S}} \circ \Pi} \log(2e m_{\mathcal{T}}/d_{h_{\mathcal{S}} \circ \Pi}) \;+\; 2\log(8/\delta)}{m_{\mathcal{T}}}} \end{aligned} \qquad (3)$$

[Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*".

Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.]

Theoretical guarantees

$$\forall\, \widehat{h}_{\mathcal{S}} \in \mathcal{H}_{\mathcal{S}} : \quad \underset{\pi \in \Pi}{\mathrm{Min}}\, R_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi) \;\leq\; \omega\big(R_{\mathcal{S}}(h_{\mathcal{S}})\big) \qquad (2)$$

Ridiculous

where $\omega : \mathbb{R} \rightarrow \mathbb{R}$ is a non-decreasing function.

Irrelevant

$$R_{\mathcal{T}}(\widehat{h}_{\mathcal{T}}) \;\leq\; \omega\big(\widehat{R}_{\mathcal{S}}(\widehat{h}_{\mathcal{S}})\big) \;+\; 2\,\sqrt{\frac{2\, d_{\mathcal{H}_{\mathcal{S}}}\, \log(2e m_{\mathcal{S}}/d_{\mathcal{H}_{\mathcal{S}}}) \;+\; 2\log(8/\delta)}{m_{\mathcal{S}}}}$$

$$+\; 4\,\sqrt{\frac{2\, d_{h_{\mathcal{S}} \circ \Pi}\, \log(2e m_{\mathcal{T}}/d_{h_{\mathcal{S}} \circ \Pi}) \;+\; 2\log(8/\delta)}{m_{\mathcal{T}}}}$$

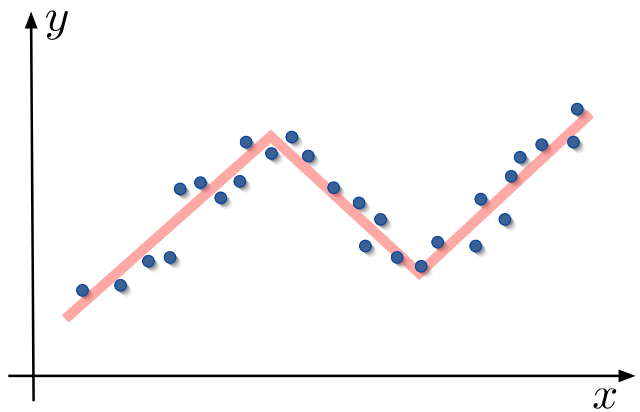
[Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.]

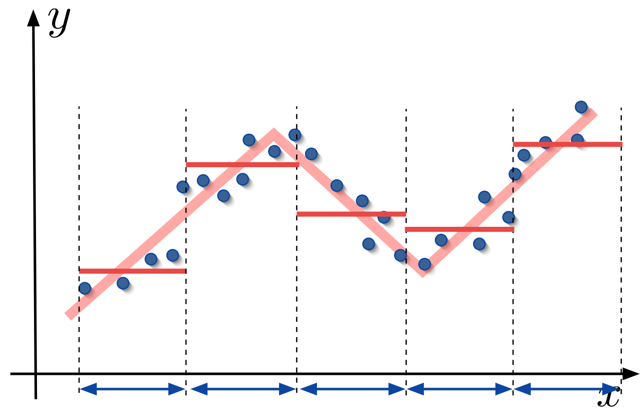
A relationship with **tracking**?

Tracking

Instead of **learning a complex function** over the whole of $\mathcal{X}$

- **If** you know that the task is slowly evolving with time
- Learn a simple **local** function

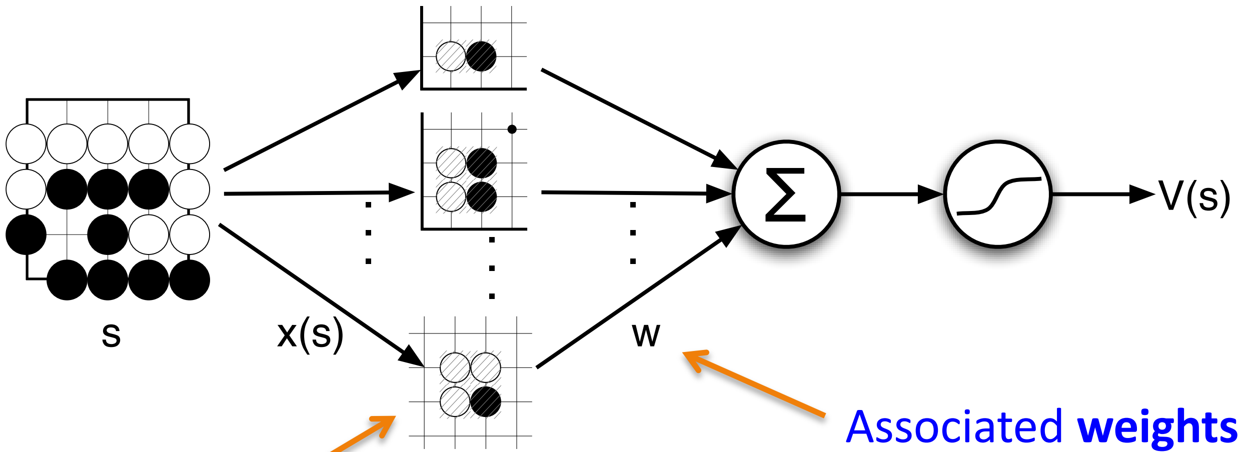




R. Sutton and A. Koop and D. Silver (2007) *"On the role of tracking in stationary environments"* (ICML-07) Proceedings of the 24th international conference on Machine learning, ACM, pp.871-878, 2007.

Tracking to play Go

- 5 x 5 Go
  - More than 5 x 10^{10} unique positions
- Usual approach: learn a **general** evaluation function V(*s*) valid $\forall s$

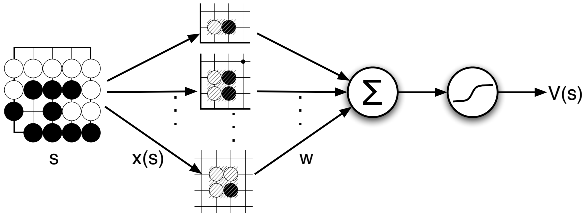


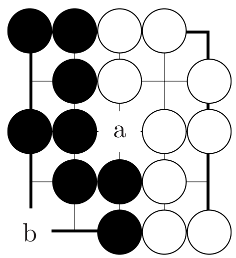
Associated **weights** (learnt)

Features describing the situation

- **Tracking approach**: learn an **evaluation** function V(*s*)

  local to the current *s*

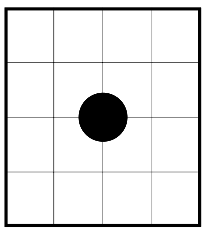




In general, playing (a) (center) is better than playing (b)

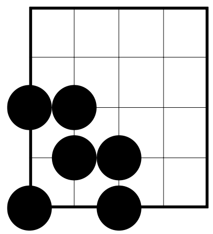
BUT

In this situation, playing (b) is better than playing (a)



More weight

More weight



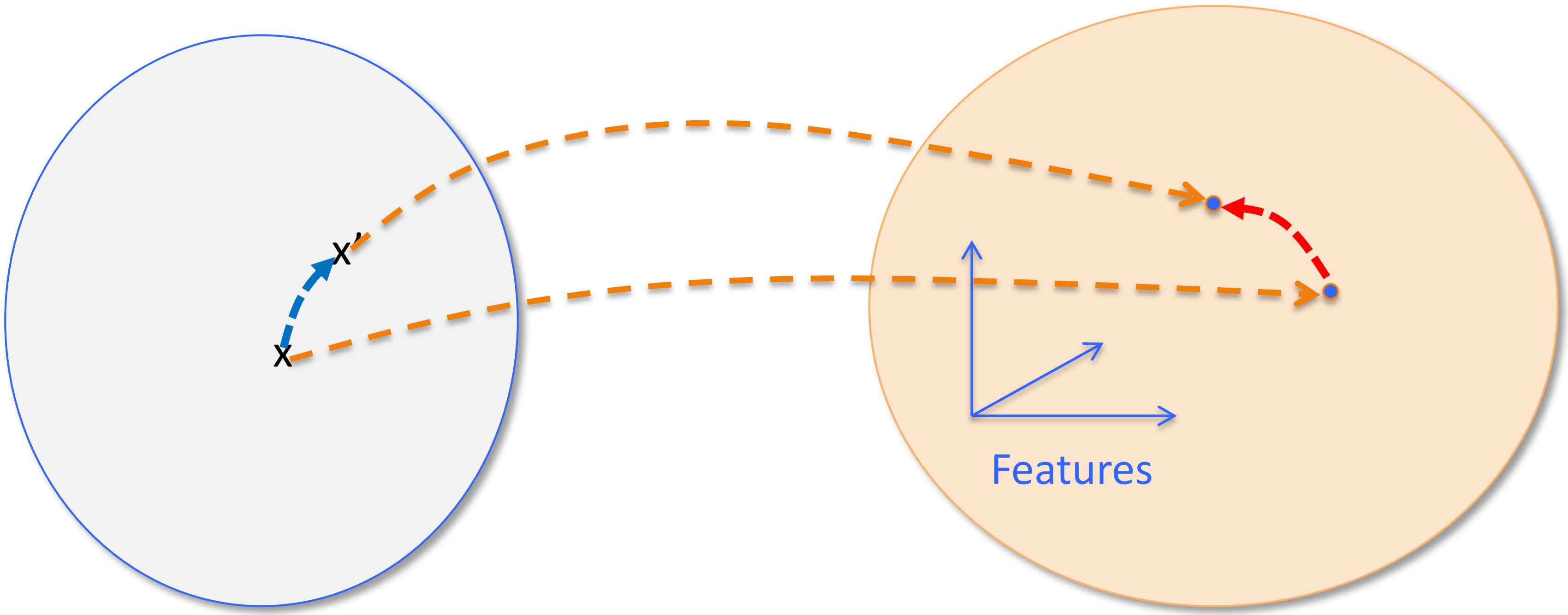
Tracking as **local changes** of representation

Embedding

…

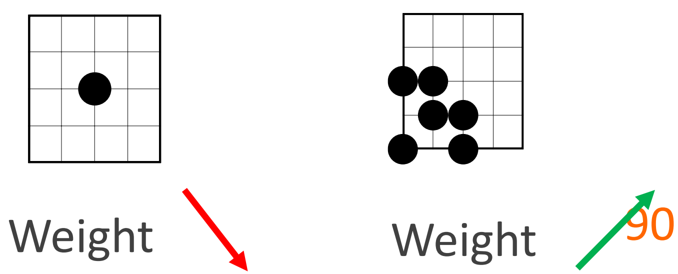
Space of go **positions**

Space of **representations**



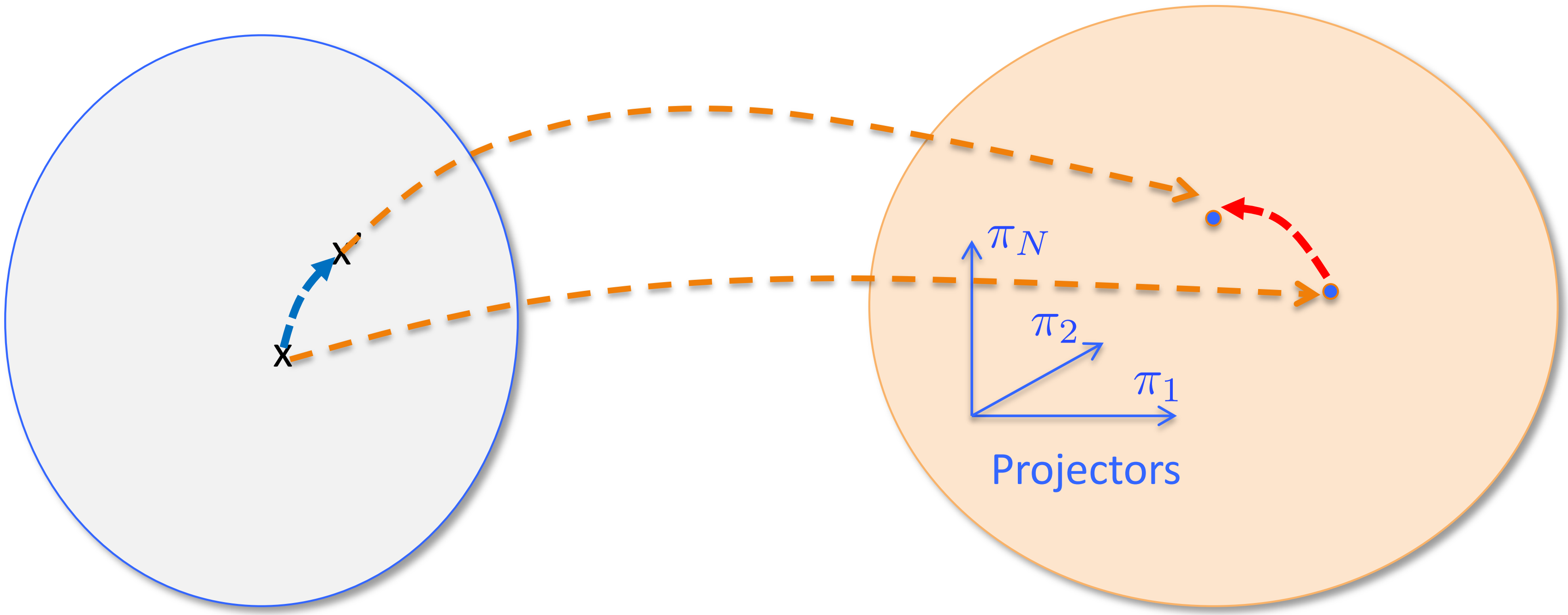
*The **weights** of the features*

***change** with the evolving position*



Transboost as **local changes** of representation

...



Space of learning tasks

Target training sets

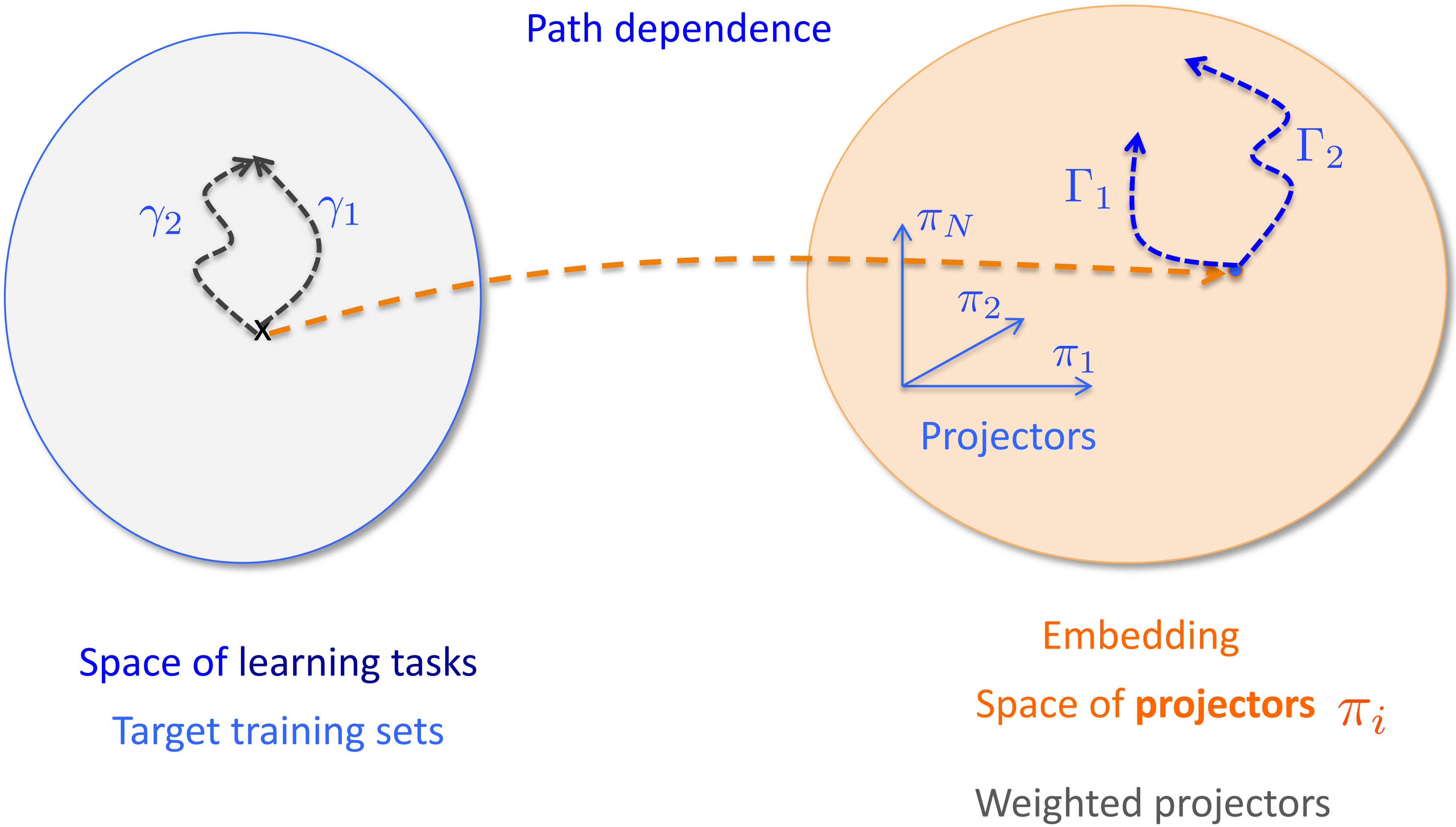
Embedding

Space of **projectors** π_i

Weighted projectors

Transboost as **local changes** of representation

...



Outline

1. Transfer learning: questions
2. Transfer learning in neural networks
3. TransBoost: an algorithm and what it tells on the role of the source
4. Curriculum learning and the geometry of the space of learning tasks
5. How to measure the difficulty of a training example
6. Conclusions

Curriculum building

And the **geometry** of the space of **learning tasks**

Sequencing effects

A fundamental question

Outline

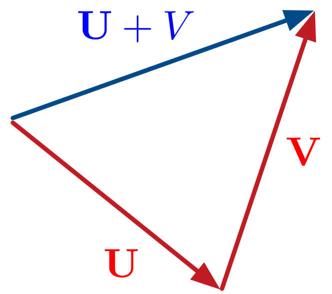
1. Supervised induction: the classical setting
2. What about Out Of Distribution learning (OOD)?
3. Parallel transport, covariant derivative and transfer learning
   - What they are
   - ... and in Machine Learning
4. A way to deal with different spaces of tasks
5. Conclusions

Parallel Transport

and Covariant Derivative

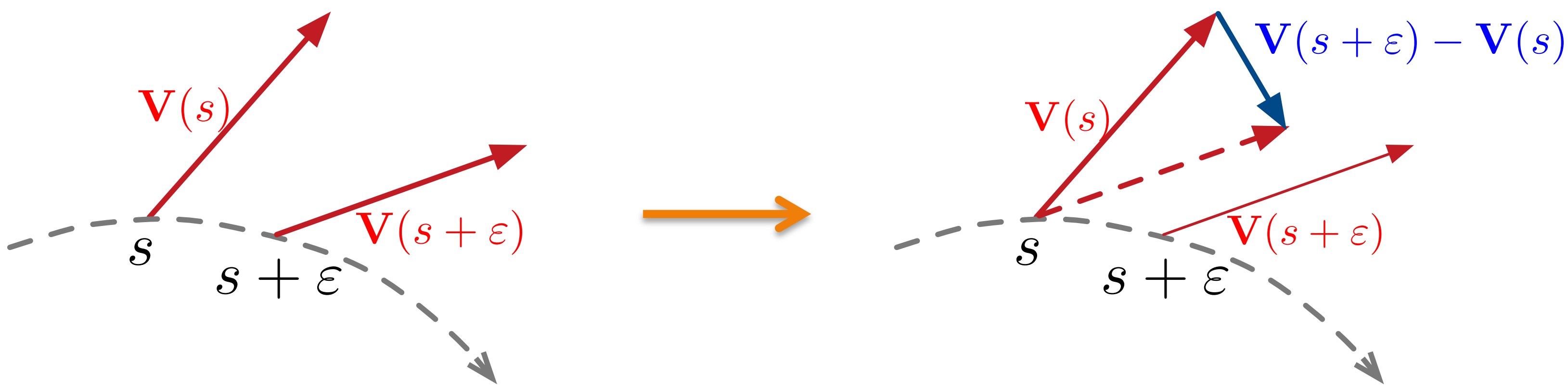
Euclidian geometry

- **Addition** of vectors



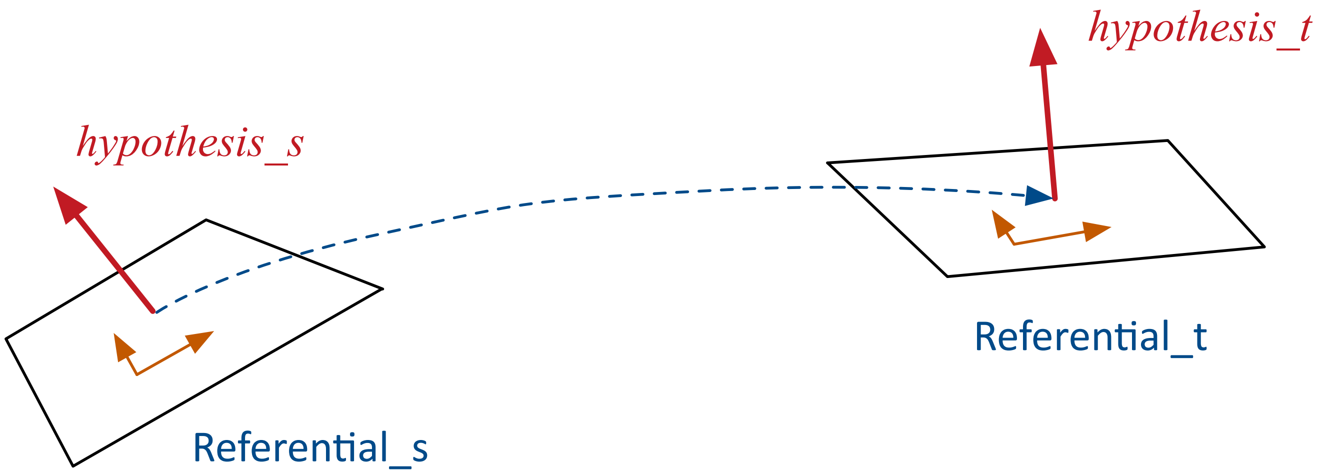
- **Substraction** of vectors and **derivative**

$$\frac{\mathrm{d}\mathbf{V}}{\mathrm{d}s}=\lim_{\varepsilon\to 0}\frac{\mathbf{V}(s+\varepsilon)-\mathbf{V}(s)}{\varepsilon}$$



Non Euclidian geometry

- Substraction of vectors and **derivative**



We can **no** longer **directly compare** vectors (or tensors)

Necessity of the covariant derivative

Parallel transport

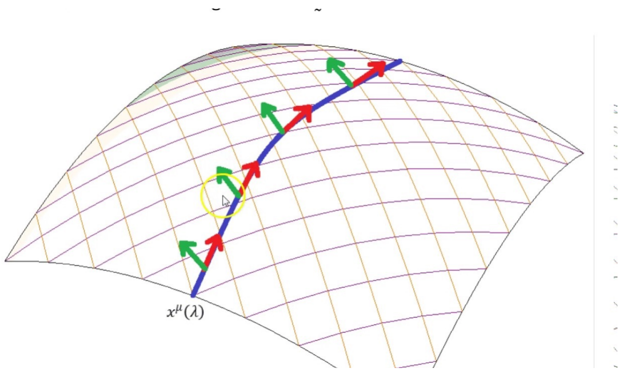
- **Transport** a vector (or a tensor) **parallel to itself** along a curve

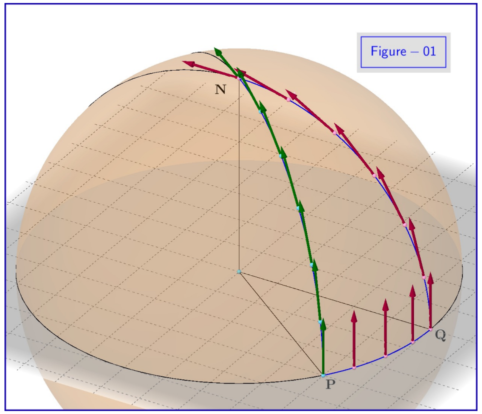
Covariant derivative = 0

Kronecker symbol

$$(\partial_k V^i)^{\text{covariant}} = \partial_k V^i + \Gamma^i_{jk} V^j$$

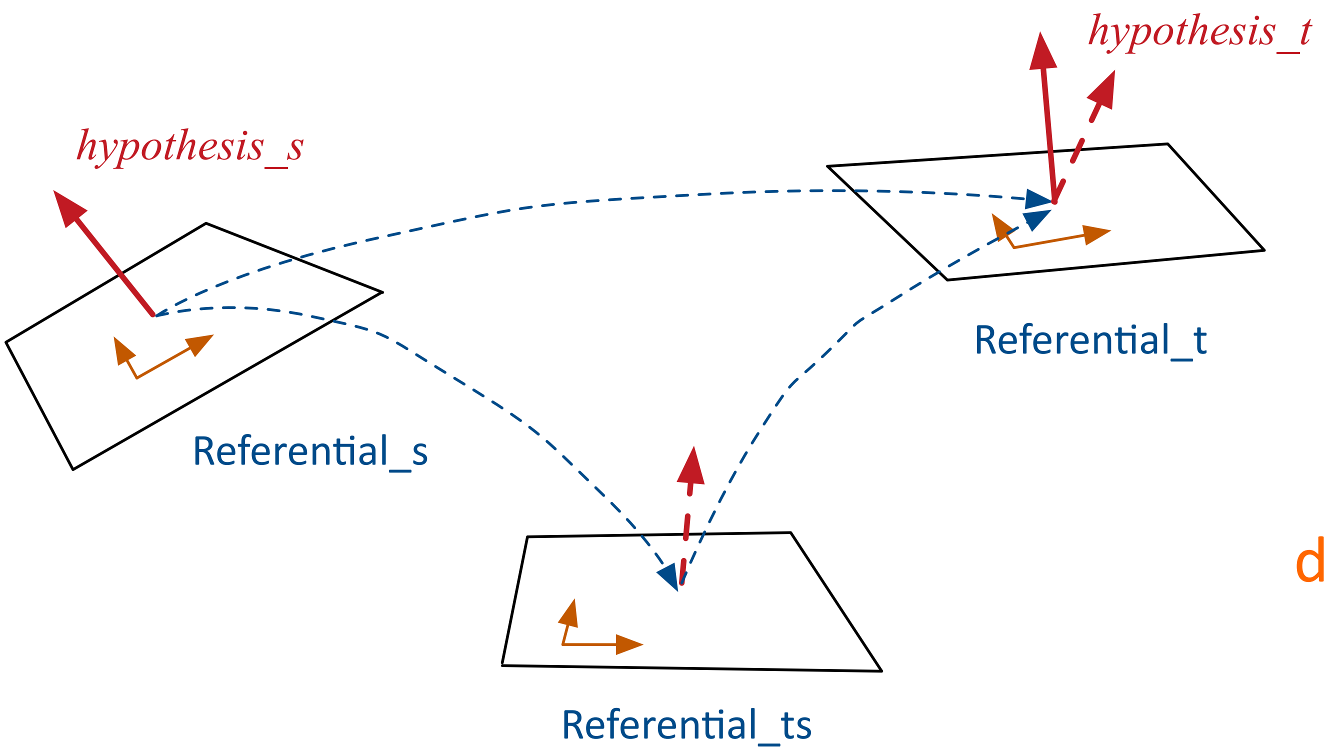
$$V^i(x^k)^{\text{parallel transported}} = V^i(x^k) + \Gamma^i_{jk} V^j \Delta x^k$$





Path **dependent!**

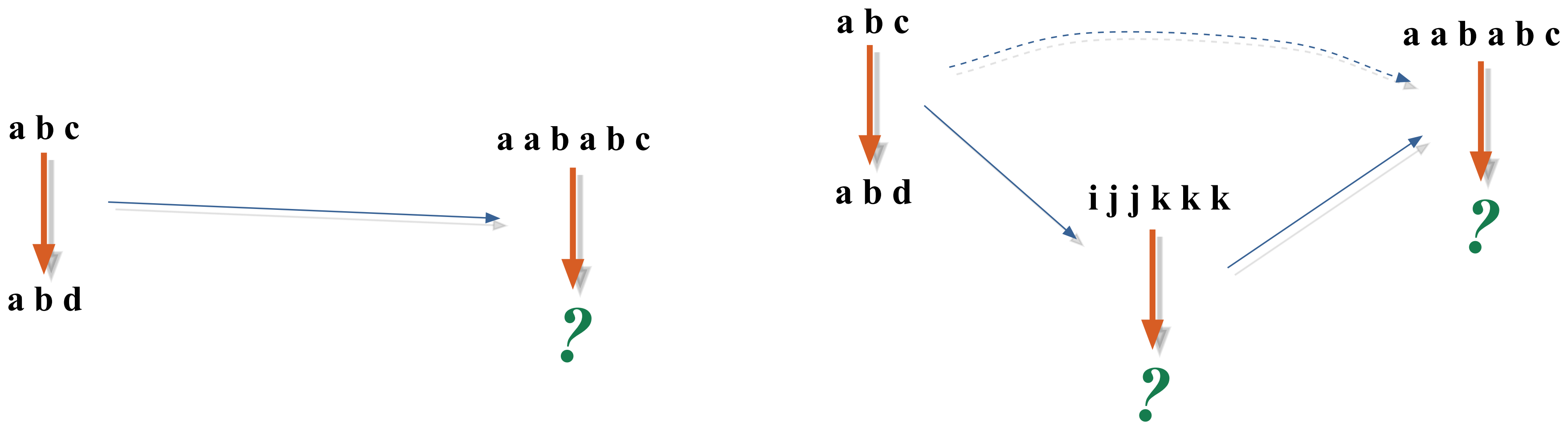
Transfer and **path dependence**



Path dependence

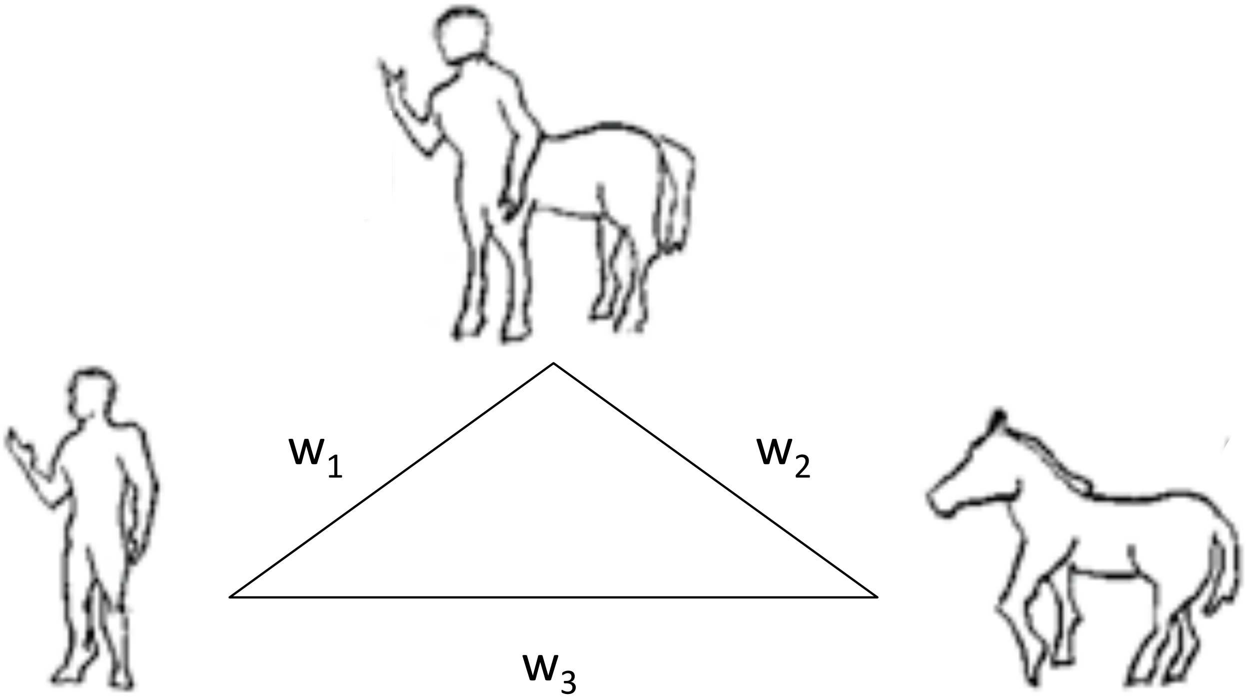
Transfer $\stackrel{?}{=}$ **Parallel transport** of hypothesis from source to target

Transfer and **path dependence**



…

Need for non-symetrical similarity



$$w_3 > w_1 + w_2$$

Adapted from: D.W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classication with non-metric distances: Image retrieval and class representation. PAMI 2000.

Parallel transport in **ML works**

Transfer = parallel transport of the source hypothesis

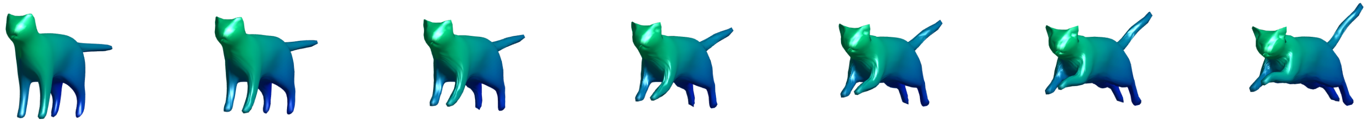
1. Tracking

2. Computer vision

3. Curriculum learning

Computer vision

...



Bauer, M., Klassen, E., Preston, S. C., & Su, Z. (2018). **A diffeomorphism-invariant metric on the space of vector-valued one-forms**. arXiv preprint arXiv:1812.10867.

Parallel transport in computer vision

Problem:

- the **convolution** operator used in standard neural network for vision assumes an **Euclidian space**
  - Translation invariance (in particular)

- But this is **not true** for general forms
- We want a convolution **operator** that changes with the position

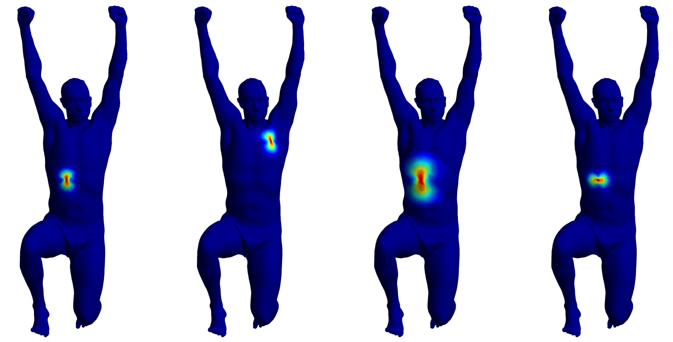


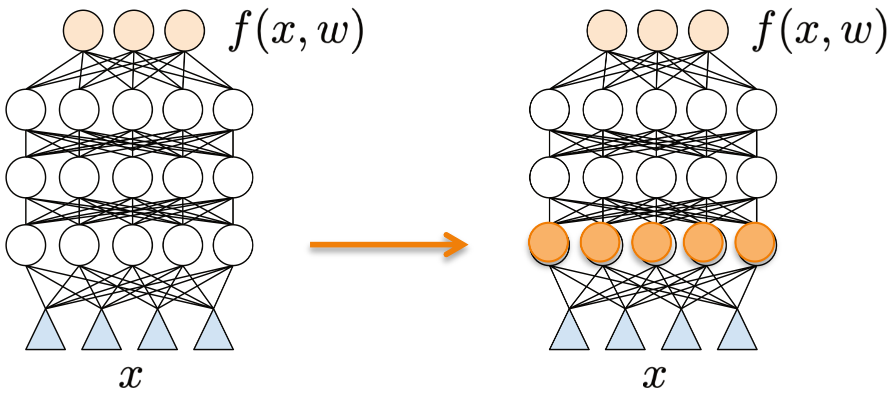
Figure 1: A compactly supported kernel (a) is transported on a manifold from the FAUST data set [2] through translation (b), translation + dilation (c) and translation + rotation (d).

Question: what **convolution operations** to use then?

Schonsheck, S. C., Dong, B., & Lai, R. (2018). **Parallel transport convolution: A new tool for convolutional neural networks on manifolds**. arXiv preprint arXiv:1805.07857.

Parallel transport in computer vision

...



Standard CNN

PTCNet

Parallel Transported Convolution layer

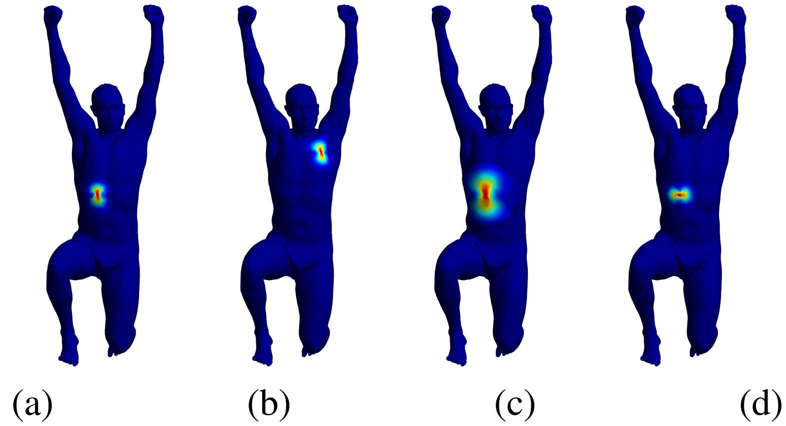


Figure 1: A compactly supported kernel (a) is transported on a manifold from the FAUST data set [2] through translation (b), translation + dilation (c) and translation + rotation (d).

The crucial idea of PTC is to define a kernel function k(x, y) which is able to encode x − y using a **parallel transportation** that naturally incorporates the manifold structure

Schonsheck, S. C., Dong, B., & Lai, R. (2018). **Parallel transport convolution: A new tool for convolutional neural networks on manifolds**. arXiv preprint arXiv:1805.07857.

Outline

1. Reminders from the past classes
2. Sequencing effects
3. Parallel transport, covariant derivative and transfer learning
4. Curriculum building
5. Can we find a role for the source task in solving a target one?
6. Conclusions

Curriculum building

Sequencing effects

- How to eliminate them?

NO!

- How to organize them and **guide** learning?
- How to build a **curriculum** for machines?

YES!

- "… Unlike (statistical) machine learning, in human learning supervision is often accompanied by a curriculum. Thus the order of presented examples is rarely random when a human teacher teaches another human.
- Likewise, the task may be divided by the teacher into smaller sub-tasks, a process sometimes called shaping (Krueger & Dayan, 2009) and typically studied in the context of reinforcement learning (e.g. Graves et al., 2017).
- Although it remained for the most part in the fringes of machine learning research, **curriculum learning** has been identified as **a key challenge** for machine learning throughout."

[Daphna Weinshall et al. (2018) « **Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks** ». ICML-2018.]

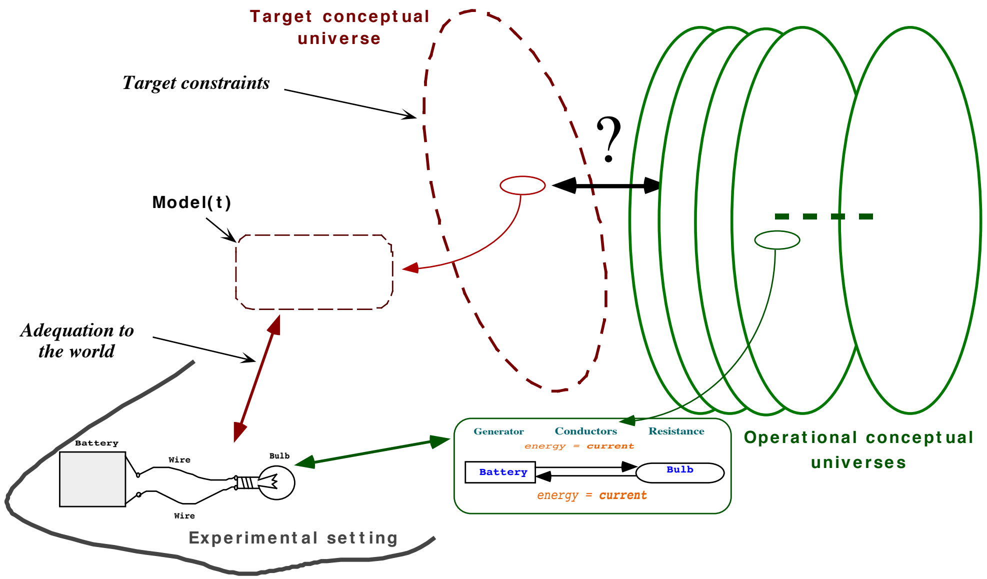
Curriculum learning

- "Humans need about two decades to be trained as fully functional adults of our society.
- That training is highly organized, based on **an education system** and a curriculum which introduces different concepts at different times, exploiting previously learned concepts to ease the learning of new abstractions.
- By choosing which examples to present and in which order to present them to the learning system, one can guide training and remarkably increase the speed at which learning can occur."

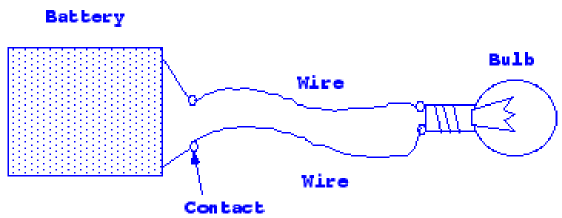
[Joshua Bengio (2018) « **Learning deep architectures for AI** ». Now Publishers Inc, 2009.]

Cognitive tunnel effect

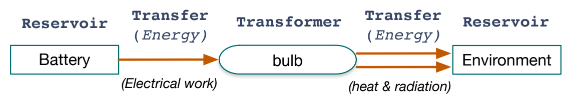
[A. Cornuéjols, A. Tiberghien, G. Collet. *Tunnel Effects in Cognition: A new Mechanism for Scientific Discovery and Education*. Arxiv-1707.04903- Tue, 18 Jul 2017 00:00:00 GMT]



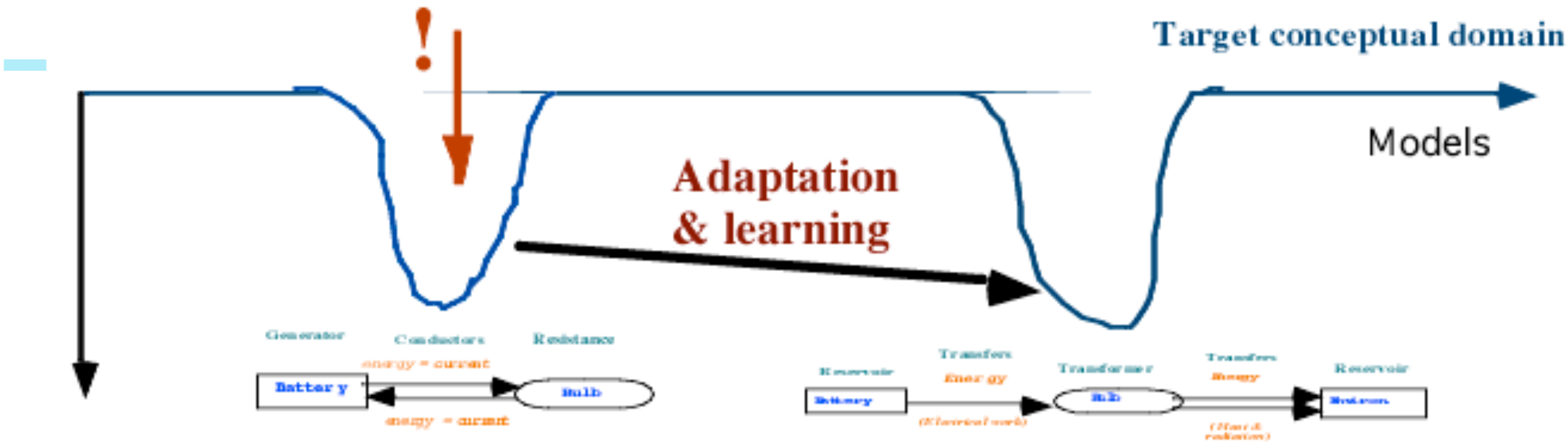
Experimental setting

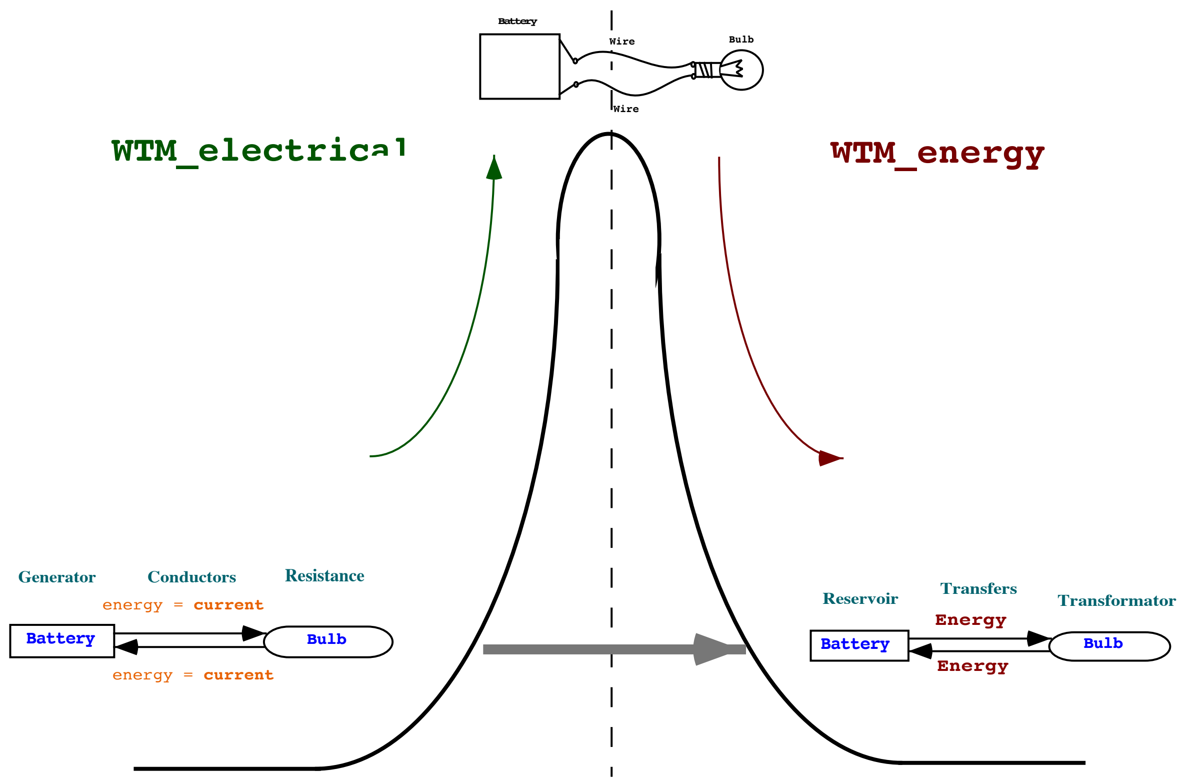


Conceptual interpretation in terms of **energy chain**



Cognitive tunnel effect





...

- We expect that transfer is easy when source and target tasks are “close”
- And it may be difficult to transfer across tasks that are “far away”

But **how to measure** “*closeness*”

and “*far away*” for learning tasks?

Define a geometry over the space of tasks

Geometry of the space of tasks

- Desiderata

  1. Should **incorporate the hypothesis space**,

     and not only the “distance” between the inputs (as is usually done)

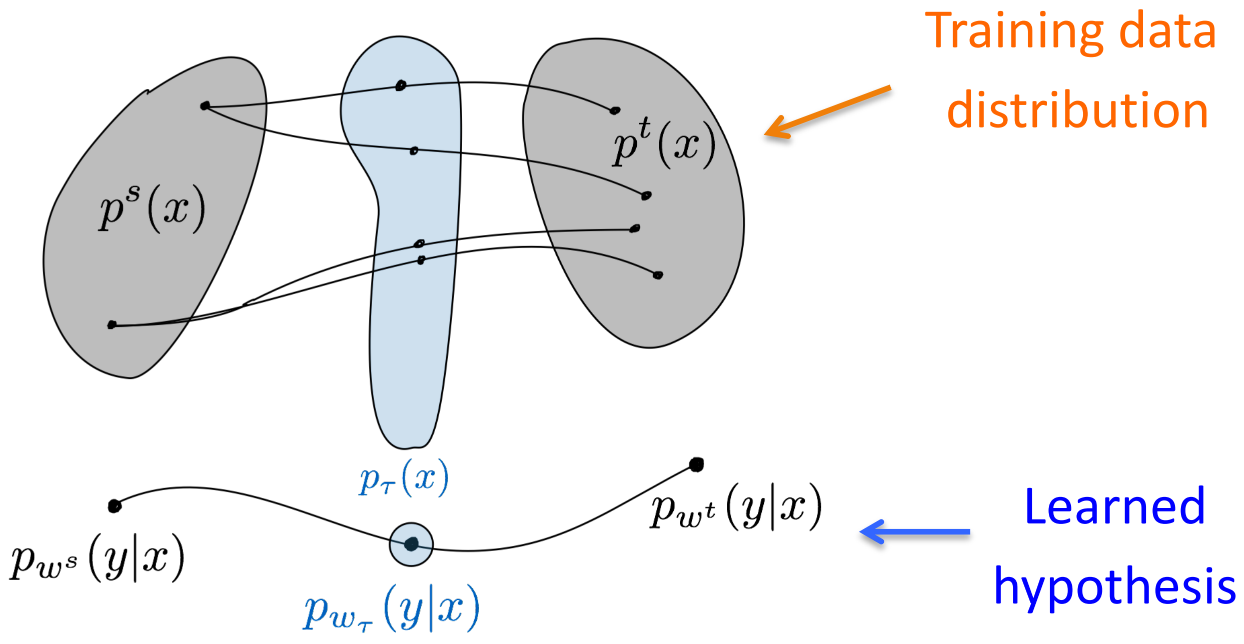
     - For instance, it is often observed that *transferring larger models is easier*.

       The geometry should reflect this.

  2. The distance between tasks is **not symmetrical**

Gao, Y., & Chaudhari, P. (2021, July). **An information-geometric distance on the space of tasks**.
In *International Conference on Machine Learning* (pp. 3553-3563). PMLR.

Idea



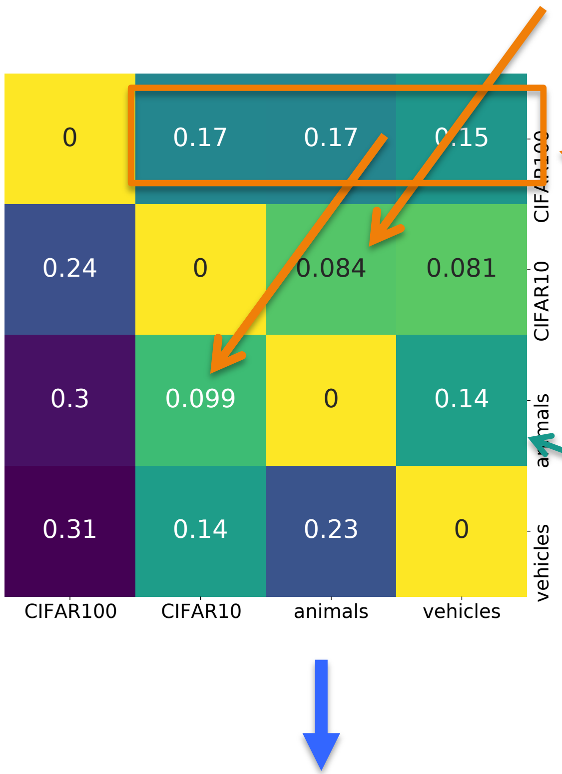
Modify **conjointly** the training data distribution and the **learned hypothesis**

Compute iteratively the intermediate training sets such that

- at each step τ the new task is close to
- what can be learned by **the current learner** (characterized by its **current hypothesis**)

Experimental results

- Using an **8-layer convolutional NN** (ReLU, dropout, batch-normalization) with a final fully connected layer



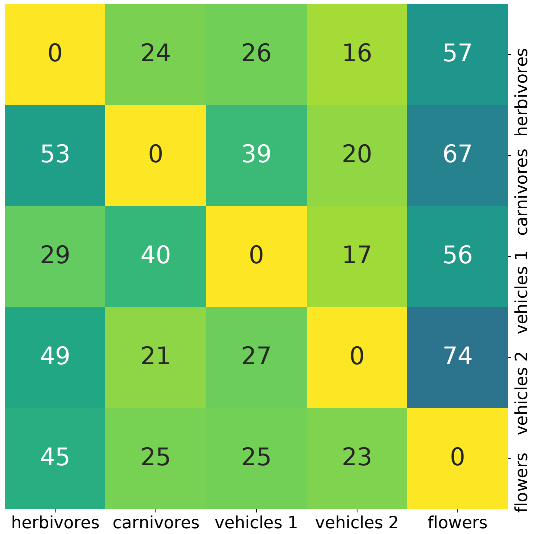
Distance is asymmetrical

- CIFAR-10 to animals < animals to CIFAR-10
- CIFAR-100 to any other is much easier than the reverse

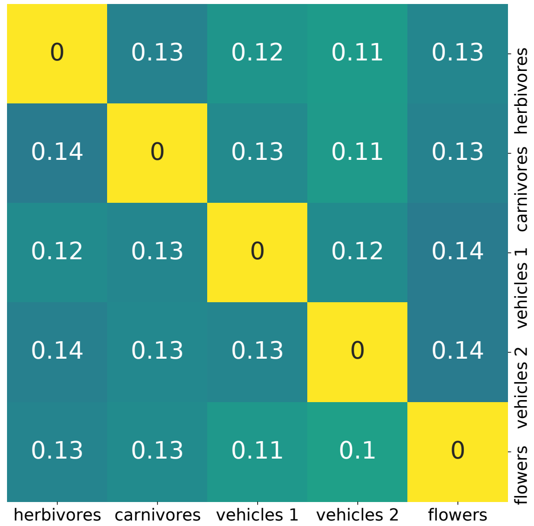
Estimated task distances

Experimental results

- Using an **8-layer convolutional NN**



- And a **wide residual network** (WRN-16-4): larger capacity



Distance is much **reduced** using a **larger capacity** model

Conclusions

- **Interesting** work
  - New definition of **distance** between tasks
    - **Asymmetrical**
    - Depends on the **capacity** of the learning system
  - New way to build a **curriculum**

Conclusions

- Interesting work
  - New definition of **distance** between tasks
    - **Asymmetrical**
    - Depends on the **capacity** of the learning system
  - New way to build a **curriculum**

- **Limits**
  - Still a **crude** way to build intermediate tasks
  - **Same** input-output source and target domains!!!
  - **Same hypothesis space** in both source and target domains!!!

Conclusions

- Interesting work
  - New definition of **distance** between tasks
    - **Asymmetrical**
    - Depends on the **capacity** of the learning system
  - New way to build a **curriculum**

- Limits
  - Still a **crude** way to build intermediate tasks
  - **Same** input-output source and target domains!!!
  - **Same hypothesis space** in both source and target domains!!!

Not **general** transfer learning

What if the space of tasks is not continuous?