Transfer Learning and representations

Transferring representations

Invariant Risk Minimization

Multi-task learning

Antoine Cornuéjols

AgroParisTech – INRAé MIA Paris

EKINOCS research group

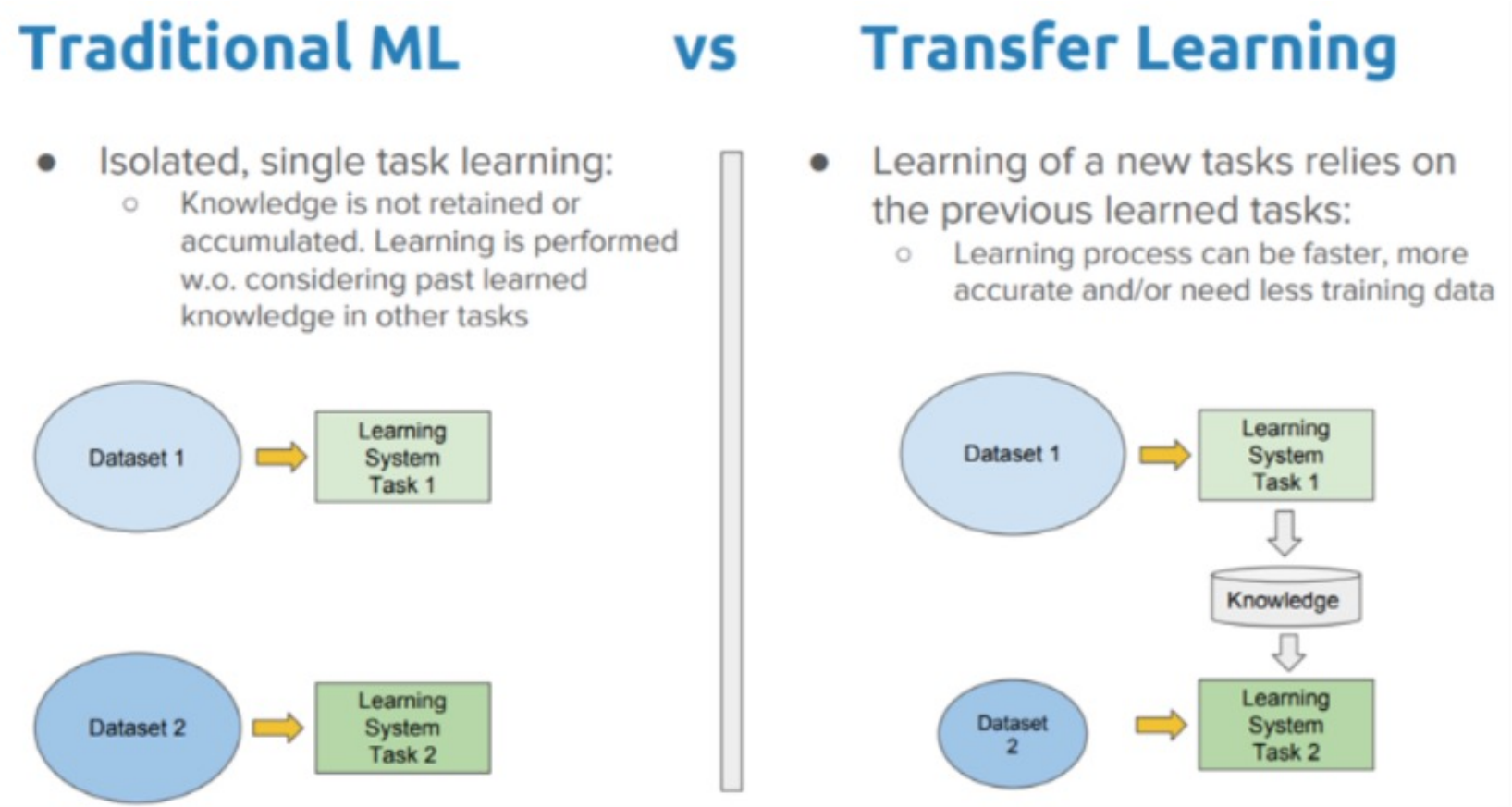
AgroParisTech

INRA SCIENCE & IMPACT

Outline

1. Transfer learning: reminder
2. Fine tuning: how transferable are features in deep NNs?
3. Are the features learned during pretraining of foundational models general enough to enable fine tuning on any task?
4. Multi-task learning
5. Conclusions

- Illustration



Notations

1. **Source** domain *S*
   - Source **training data** S_S
   - Source data **distribution** D_S
   - Source **hypothesis** h_S
2. **Target** domain *T*
   - Target **training data** S_T $(|S_T| << |S_S|)$
   - Target **data distribution** D_T
   - Target **hypothesis** h_T

What can we **transfer** from one task to another?

- In the following: a **strong assumption**

There is **something in common** between the source and the target

We will remove this assumption later on

What can we transfer

- What could be in **common**?

1. Look for a universal **representation**
2. Underlying supposedly **common regularities**
3. Learning a translation to a **common decision function**
4. **Others**

Outline

1. Transfer learning: reminder
2. Fine tuning: **how transferable** are features in deep NNs?
3. Are the features learned during pretraining of foundational models general enough to enable fine tuning on any task?
4. Multi-task learning
5. Conclusions

Universal representations ?

A **universal** representation for texts?

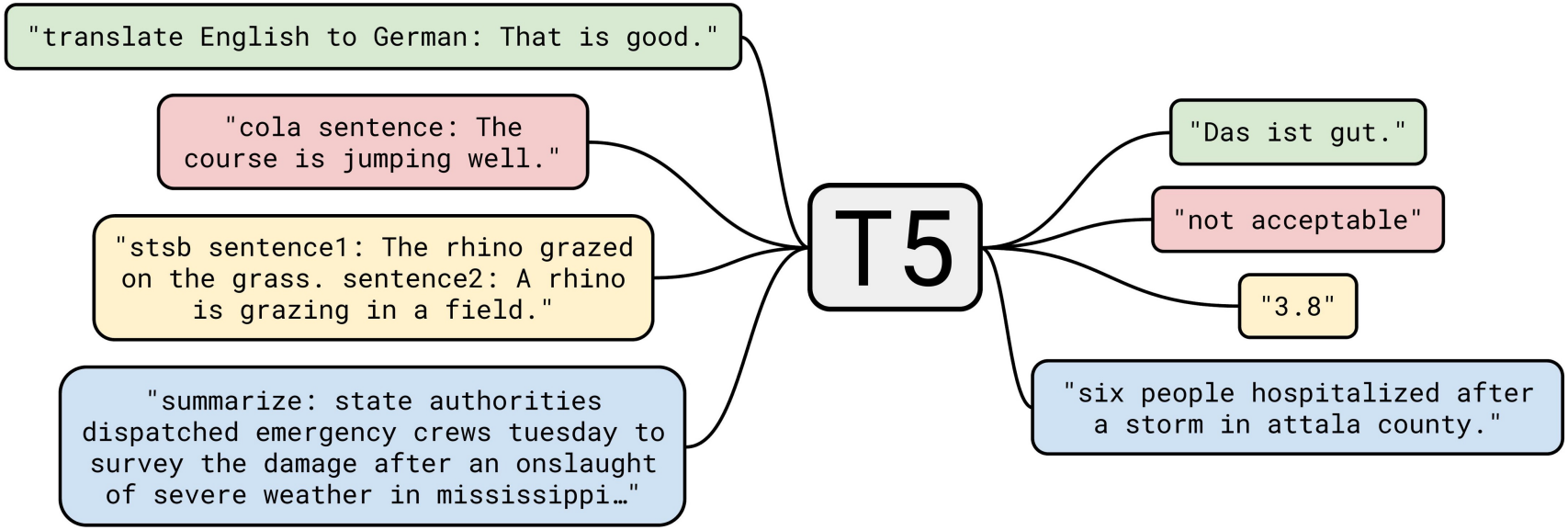


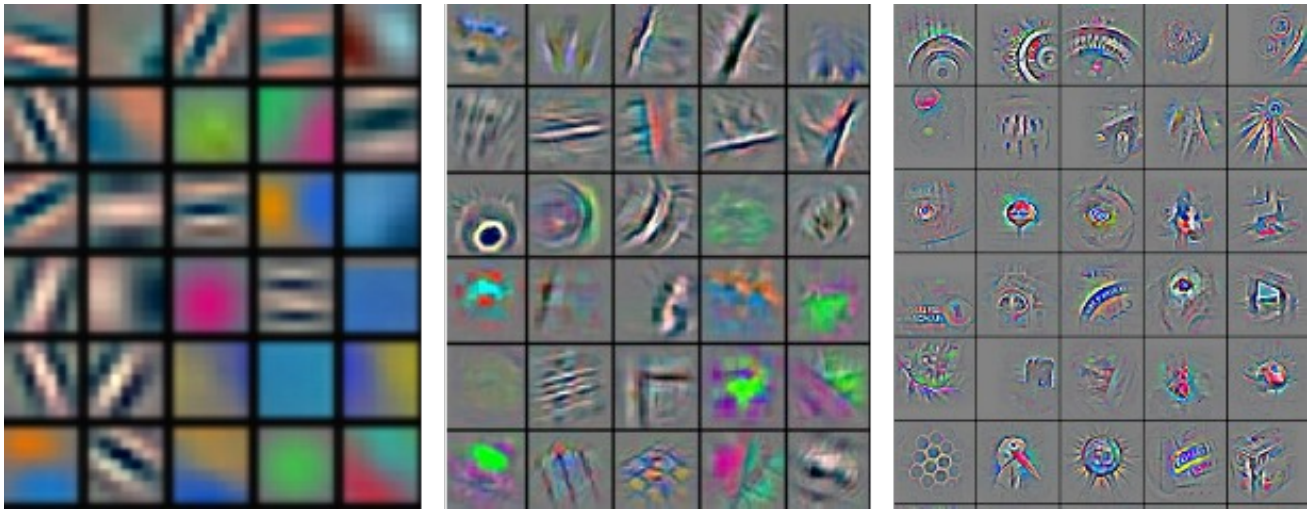
*Figure 15.36: Illustration of how the T5 model ("Text-to-text Transfer Transformer") can be used to perform multiple NLP tasks, such as translating English to German; determining if a sentence is linguistic valid or not (***CoLA** *stands for "Corpus of Linguistic Acceptability"); determining the degree of semantic similarity (***STSB** *stands for "Semantic Textual Similarity Benchmark"); and abstractive summarization. From Figure 1 of [Raf+20]. Used with kind permission of Colin Raffel.*

From [Kevin Murphy. **Probabilistic Machine Learning. An introduction**. MIT Press. (2022)], p.539

What can we transfer

1. Representations

   – E.g. for vision tasks



The idea of fine tuning

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**

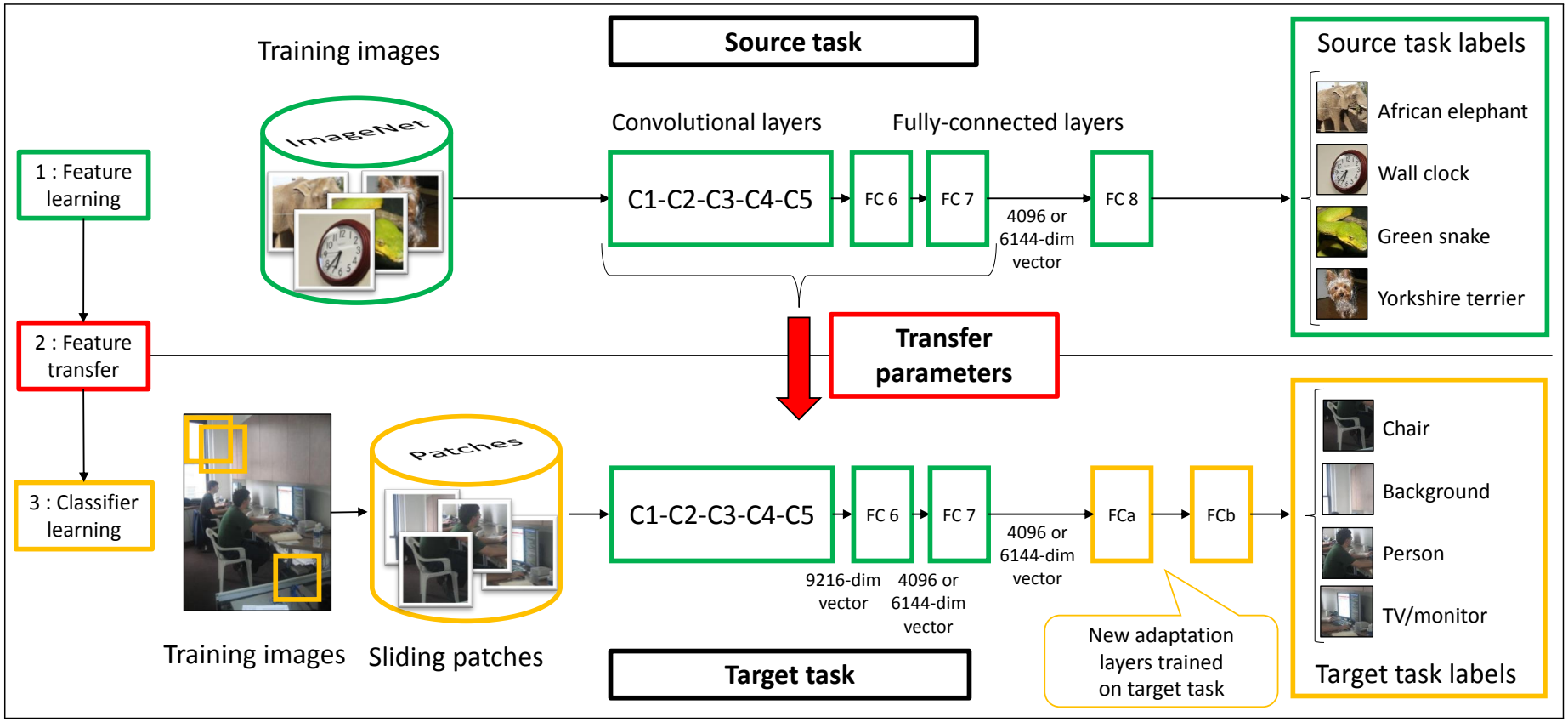
- The assumption:
  - the features learned **for a task** can be used almost as such for other, **related**, tasks
- Approach:
  - **Reuse** the first layers and **learn** the last ones
  - **Same input** spaces $X_S = X_T$, possibly $Y_S \neq Y_T$

Transfer learning

- 1st strategy
  1. Take a NN **pretrained** on the source data set,
  2. **remove** the last fully-connected layer
     (e.g. this layer's outputs are the 1000 class scores for a task like ImageNet),
  3. then treat the rest of the NN as a **fixed feature extractor**
     for the target dataset.

- 2nd strategy
  1. **Not only** replace and retrain the **classifier on top** of the NN
     on the taget dataset,
  2. but to also fine-tune the weights of the pretrained network by continuing
     the backpropagation

Transfer learning for **deep neural networks**



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)].

Transfer learning for **deep neural networks** *(case 1)*

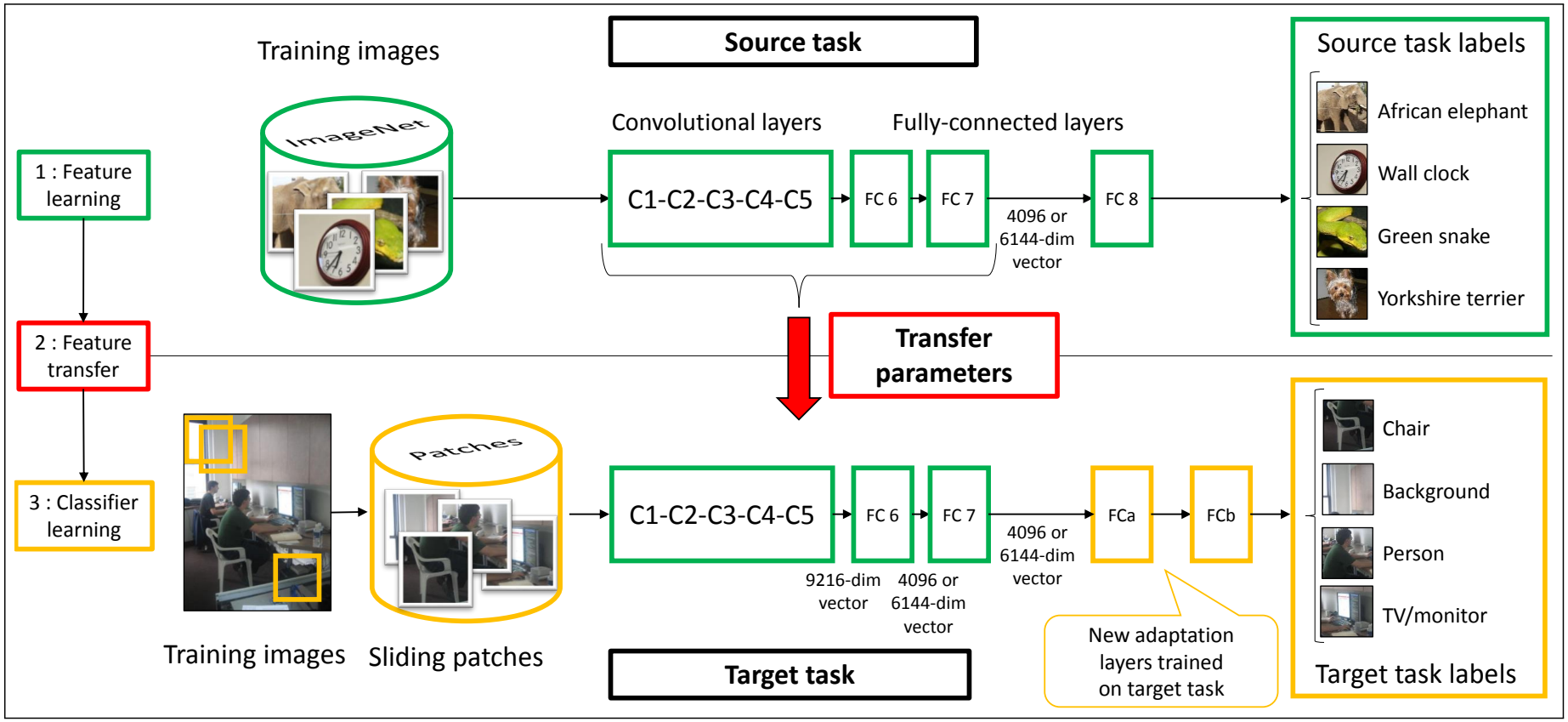


Figure 2: **Transferring parameters of a CNN.** First, the network is trained on the source task (ImageNet classification, top row) with a large amount of available labelled images. Pre-trained parameters of the internal layers of the network (C1-FC7) are then transferred to the target tasks (Pascal 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)].

Transfer learning for **deep neural networks** *(case 2)*

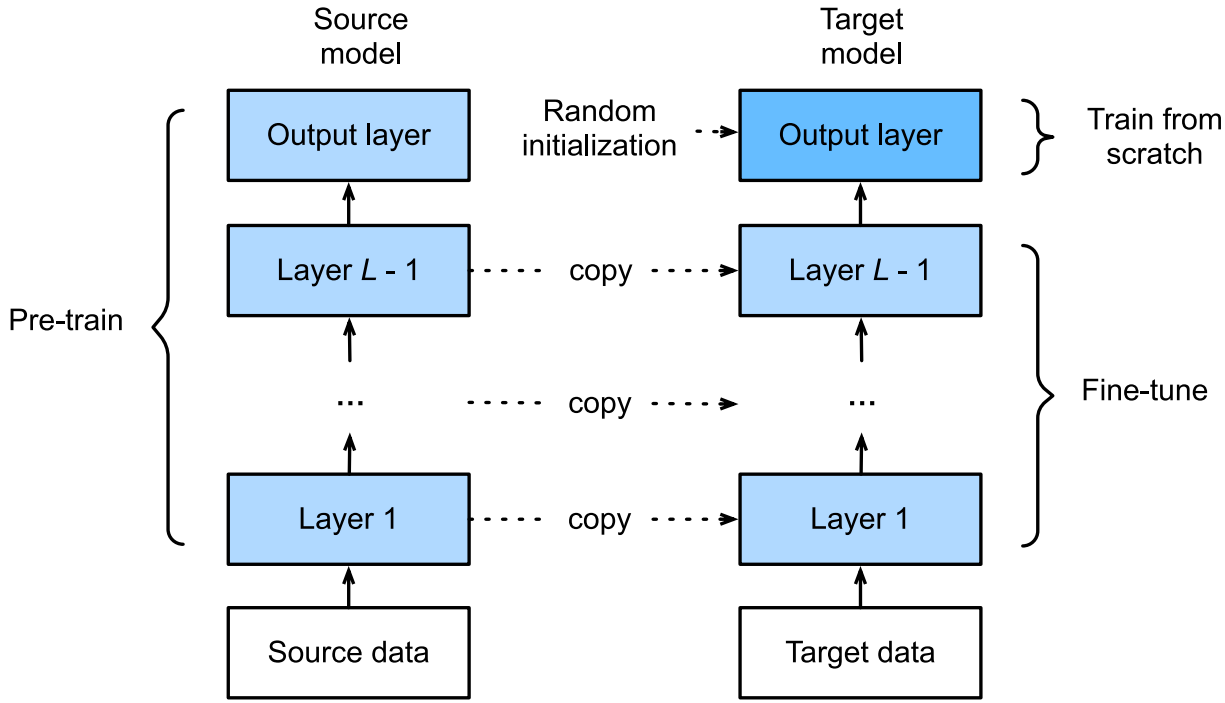


Figure 19.2: Illustration of fine-tuning a model on a new dataset. The final output layer is trained from scratch, since it might correspond to a different label set. The other layers are initialized at their previous parameters, and then optionally updated using a small learning rate. From Figure 13.2.1 of [Zha+20]. Used with kind permission of Aston Zhang.

From [Kevin Murphy. **Probabilistic Machine Learning. An introduction**. MIT Press. (2022)], p.625

(case 2)

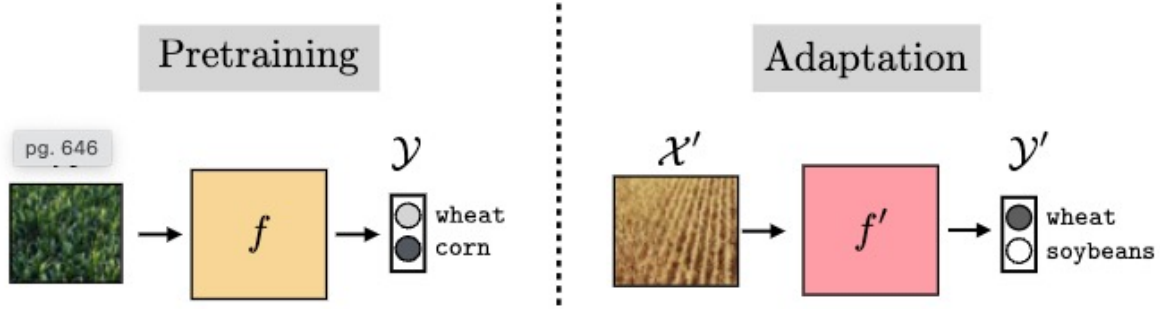


Figure 37.1: Transfer learning consists of two phases: first we pretrain a model on one task and then we adapt that model to perform a new task.

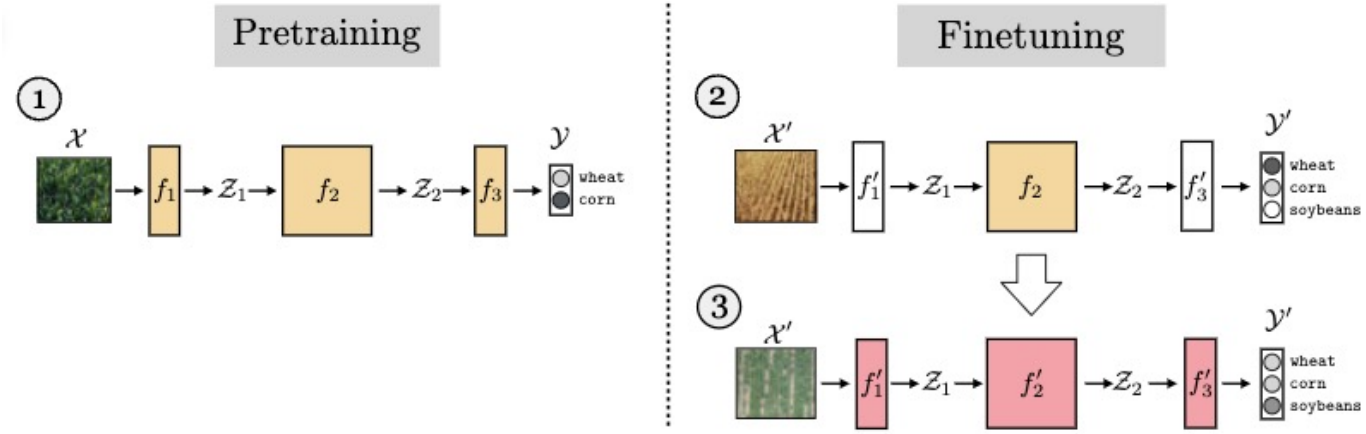


Figure 37.2: Finetuning diagram.

: Pretrained module
: Scratch module
: Finetuned module

f_1' and f_3' are trained from scratch **when the target input and output spaces are different** from the ones in the source task

From [Antonio Torralba et al. **Foundations of Computer Vision**. MIT Press, 2024, p.547]

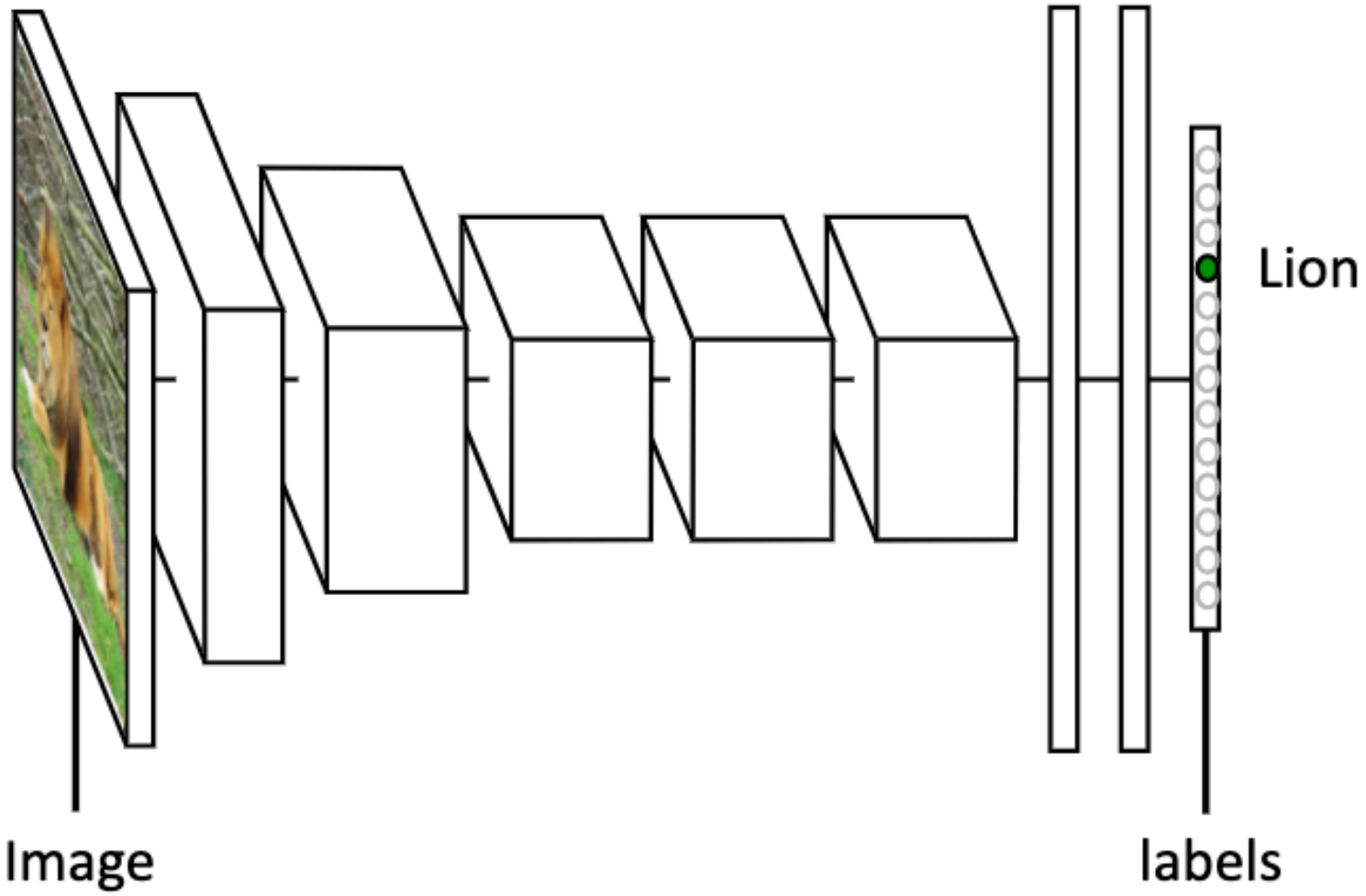
Recommandations

1. The target data set is small and **similar** to the source data set
   - **Train a linear classifier on top of the last layer** of pretrained NN
2. The target data set is **large** and **similar** to the source data set
   - **Fine-tune** the pretrained NN using the target data set
3. The target data set is small and very different from the source data set
   - Since the dataset is very different, it might **not be best to train the classifier from the top of the NN**, which contains more dataset-specific features.
     Instead, it might work better to **train a classifier** from **activations somewhere earlier** in the network.
4. The target data set is **large** and very different from the source data set
   - **Fine-tune** the pretrained NN using the target data set

But ...

... how **transferable** are **representations**?

Principle

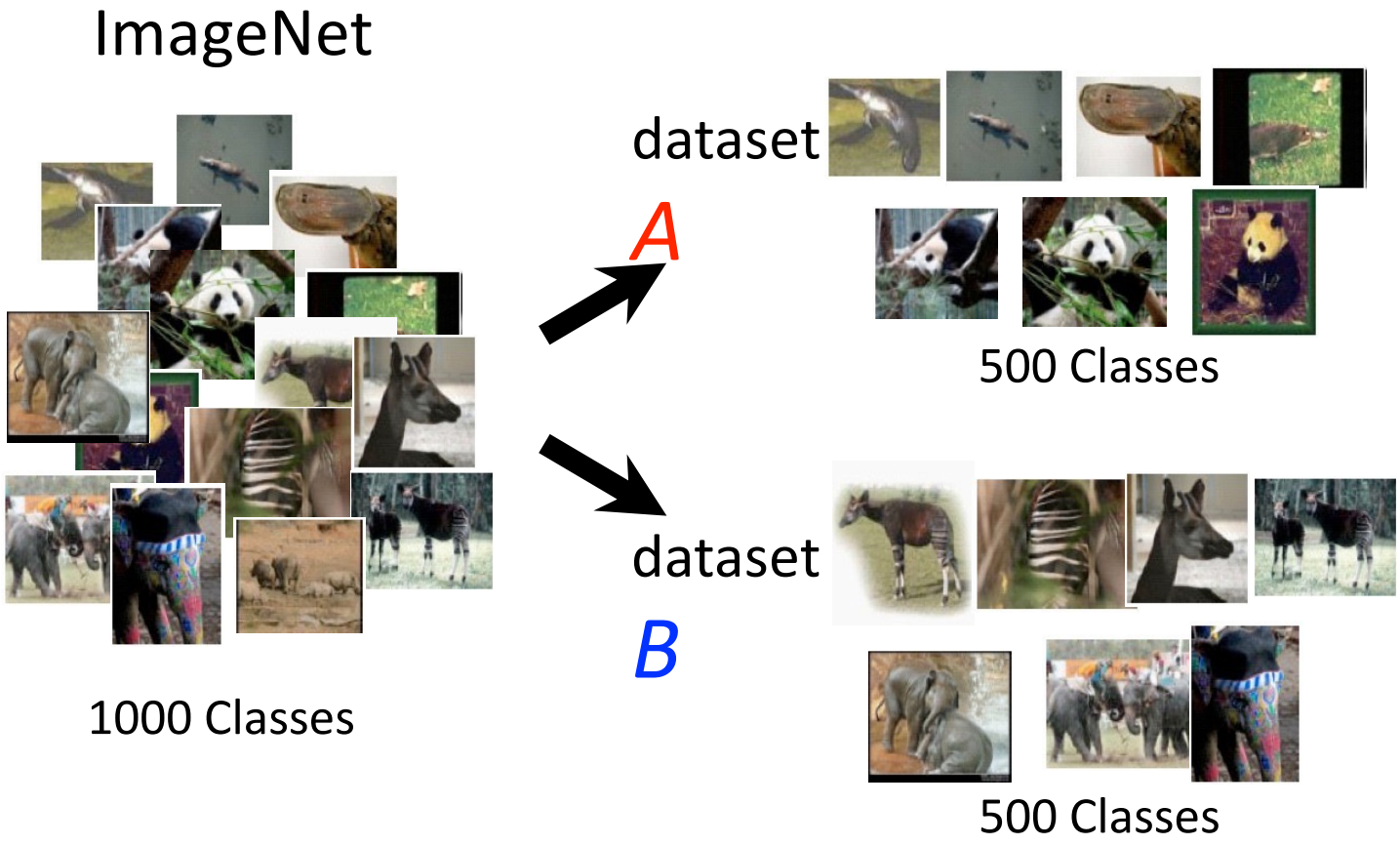


Krizhevsky, Sutskever, Hinton — NIPS 2012

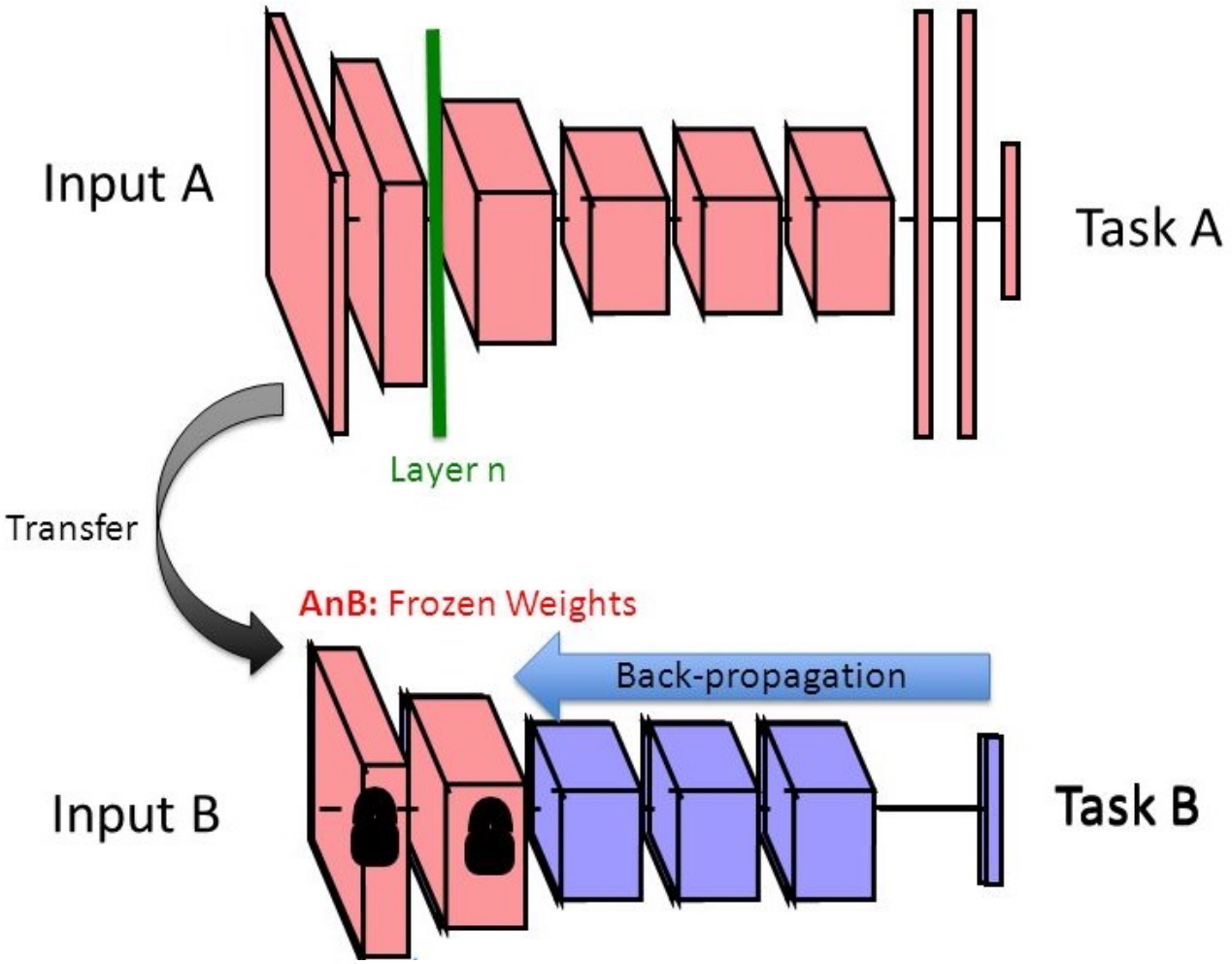
...

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

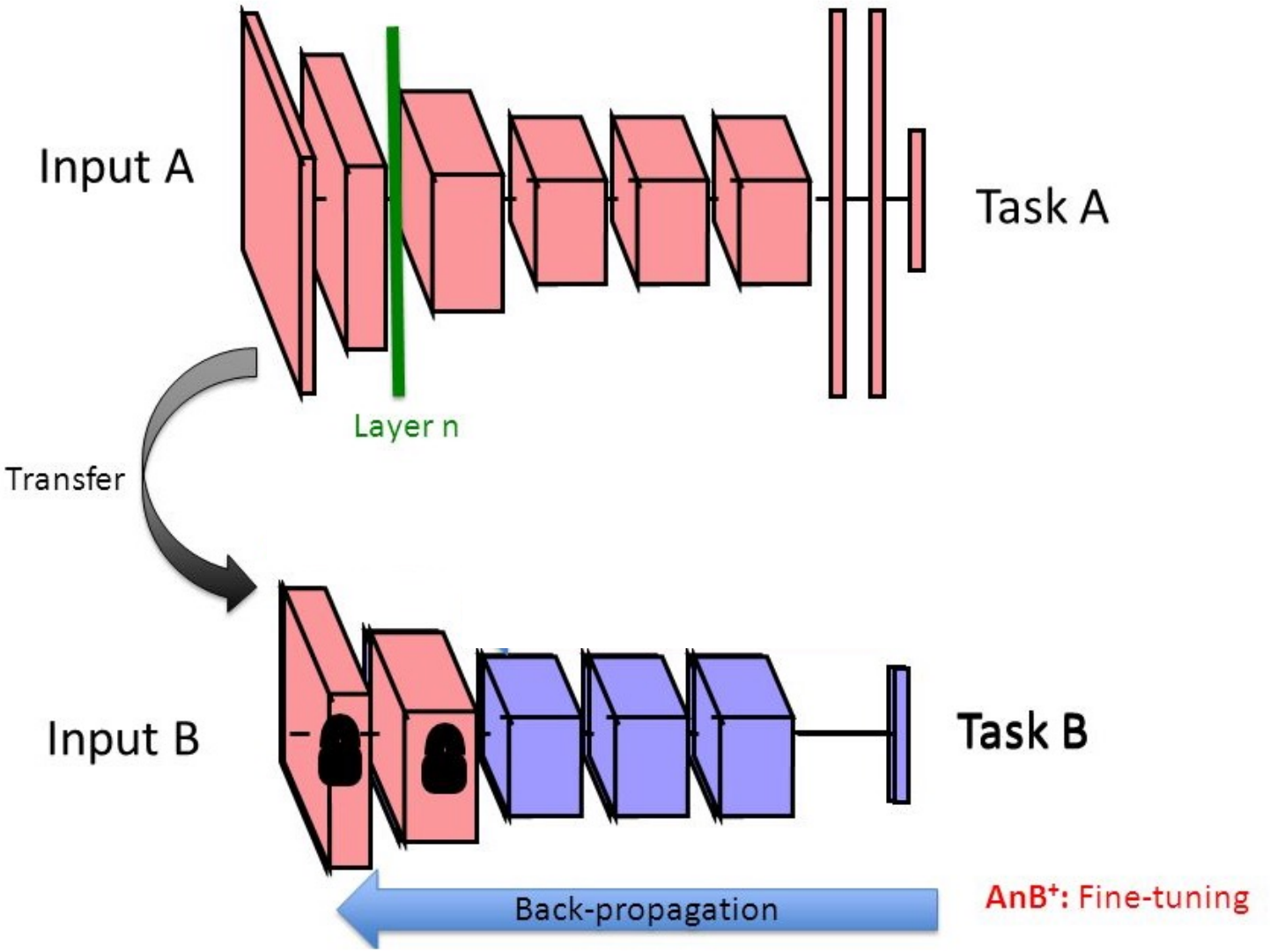
Experiments on two domains



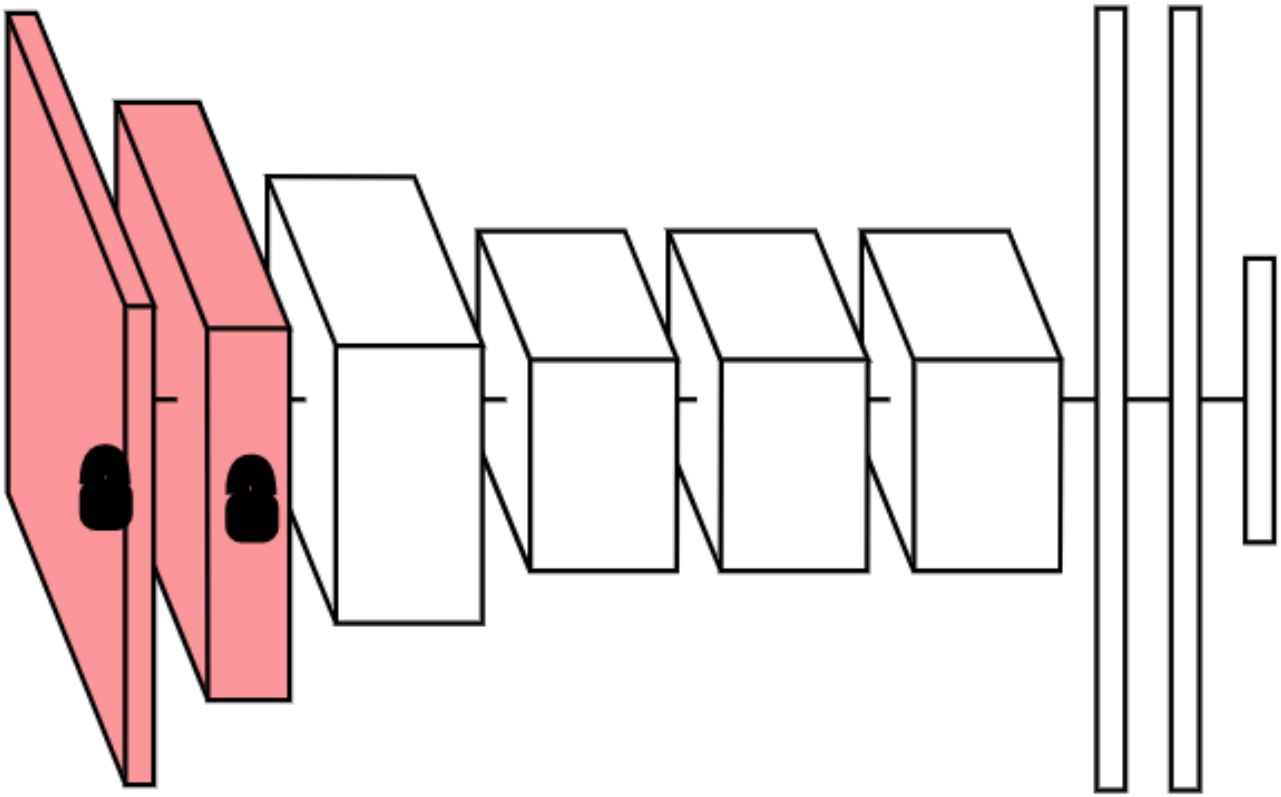
Randomly split the 1000 ImageNet classes into **two groups** each containing 500 classes and approximately half of the data, or about 645,000 examples each.



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

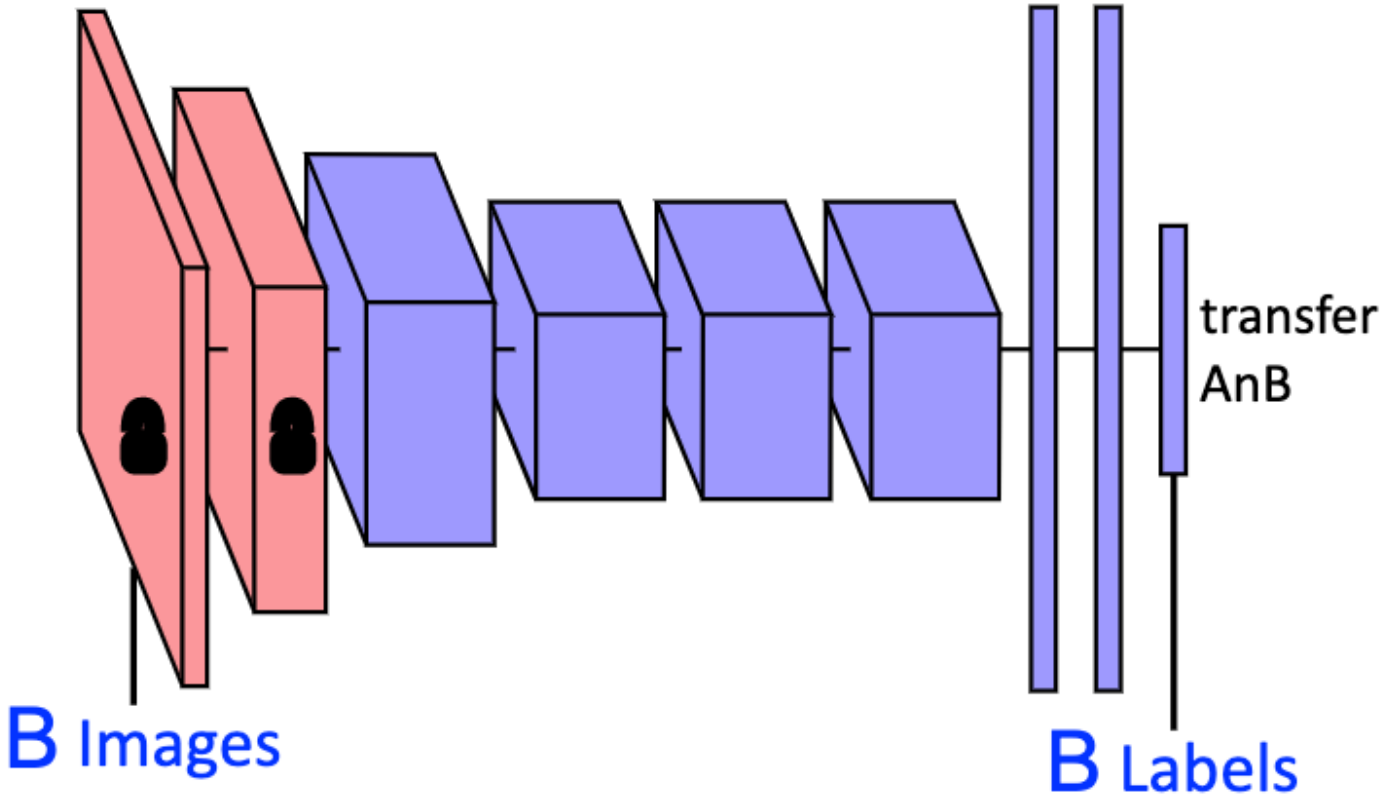


… http://slideplayer.com/slide/8370683/



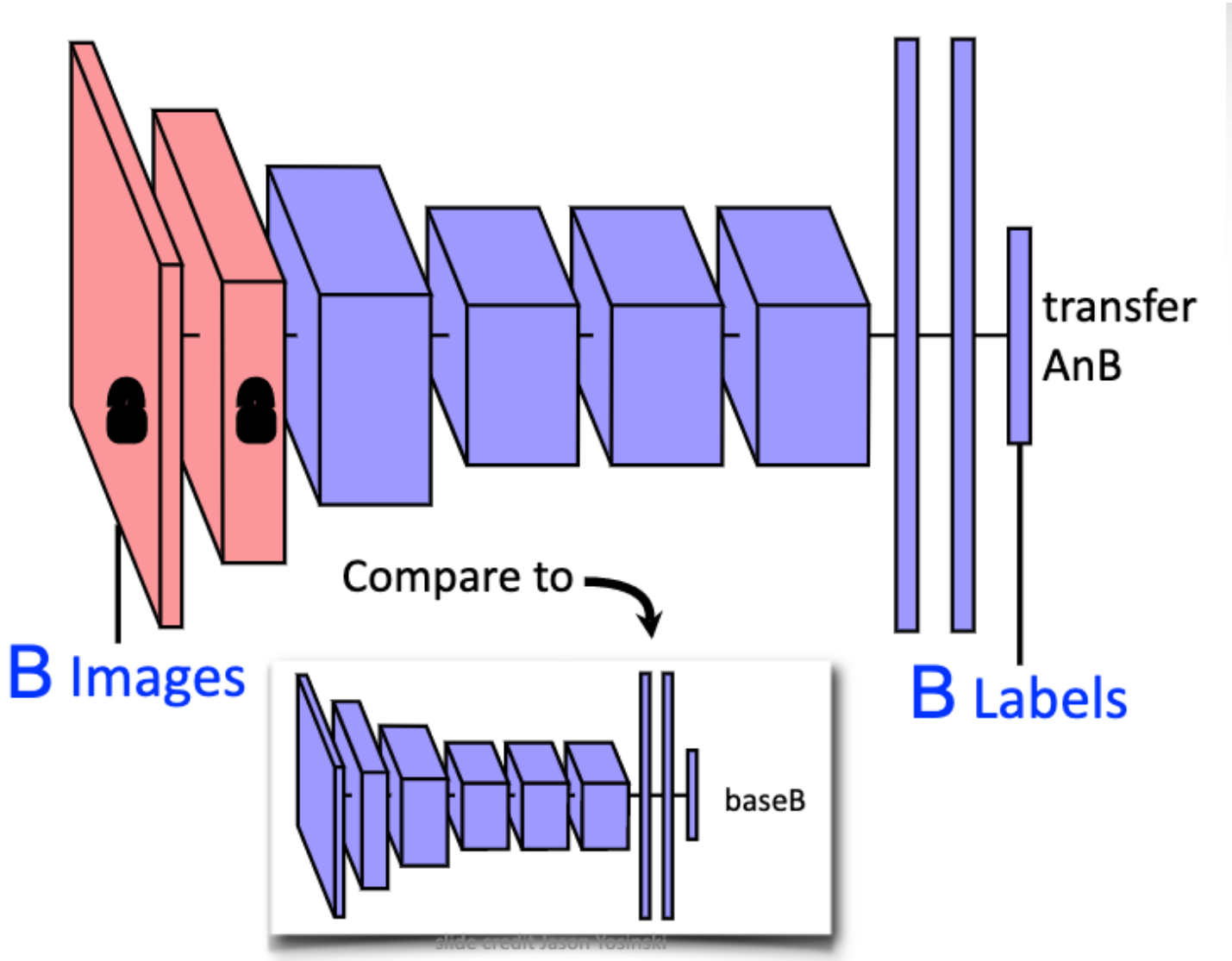
Hypothesis: If transferred features are specific to task A, performance on task B drops. **Otherwise** the performance should be the same.

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



...

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



...

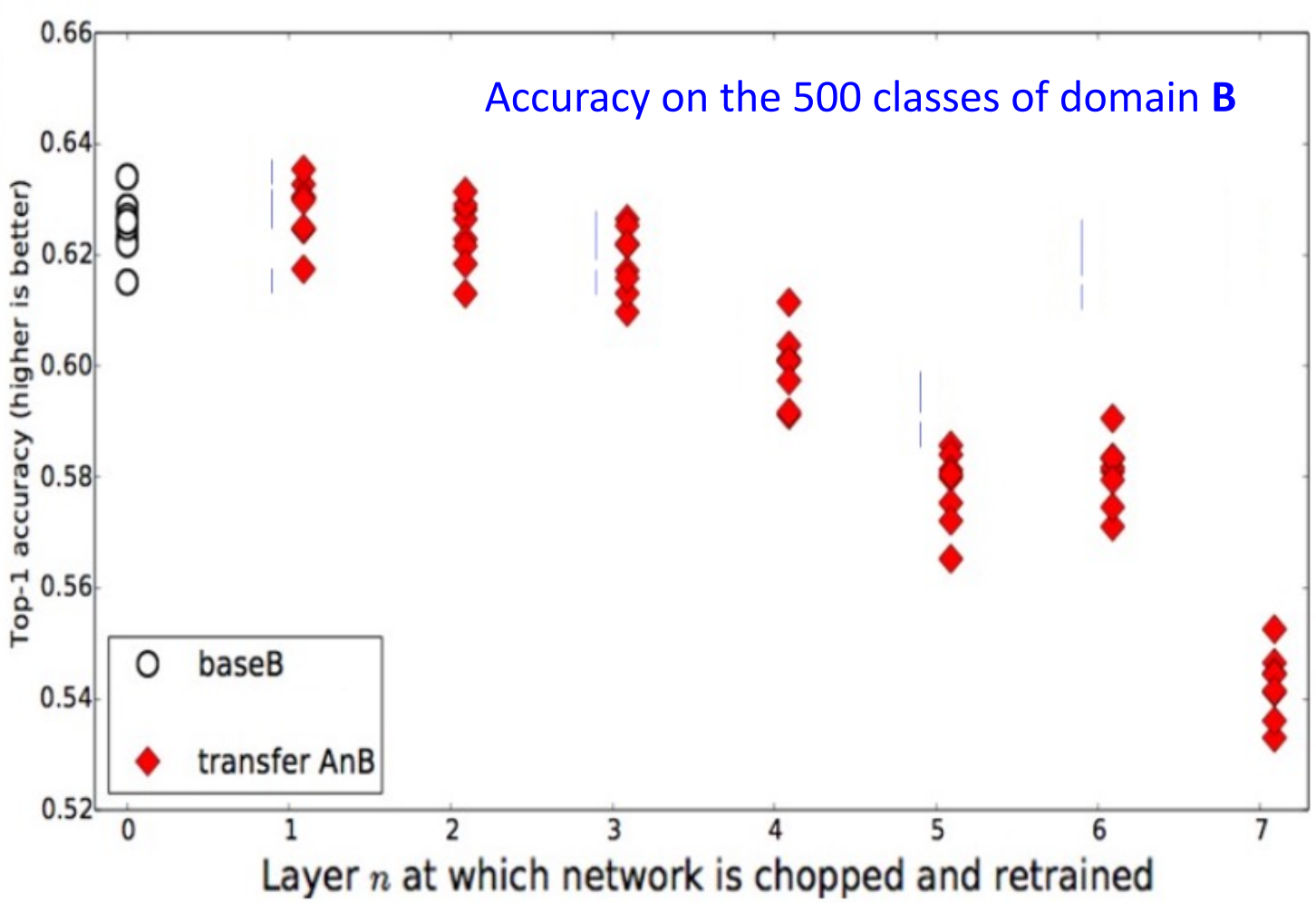
http://slideplayer.com/slide/8370683/

- Comparisons between

No transfer

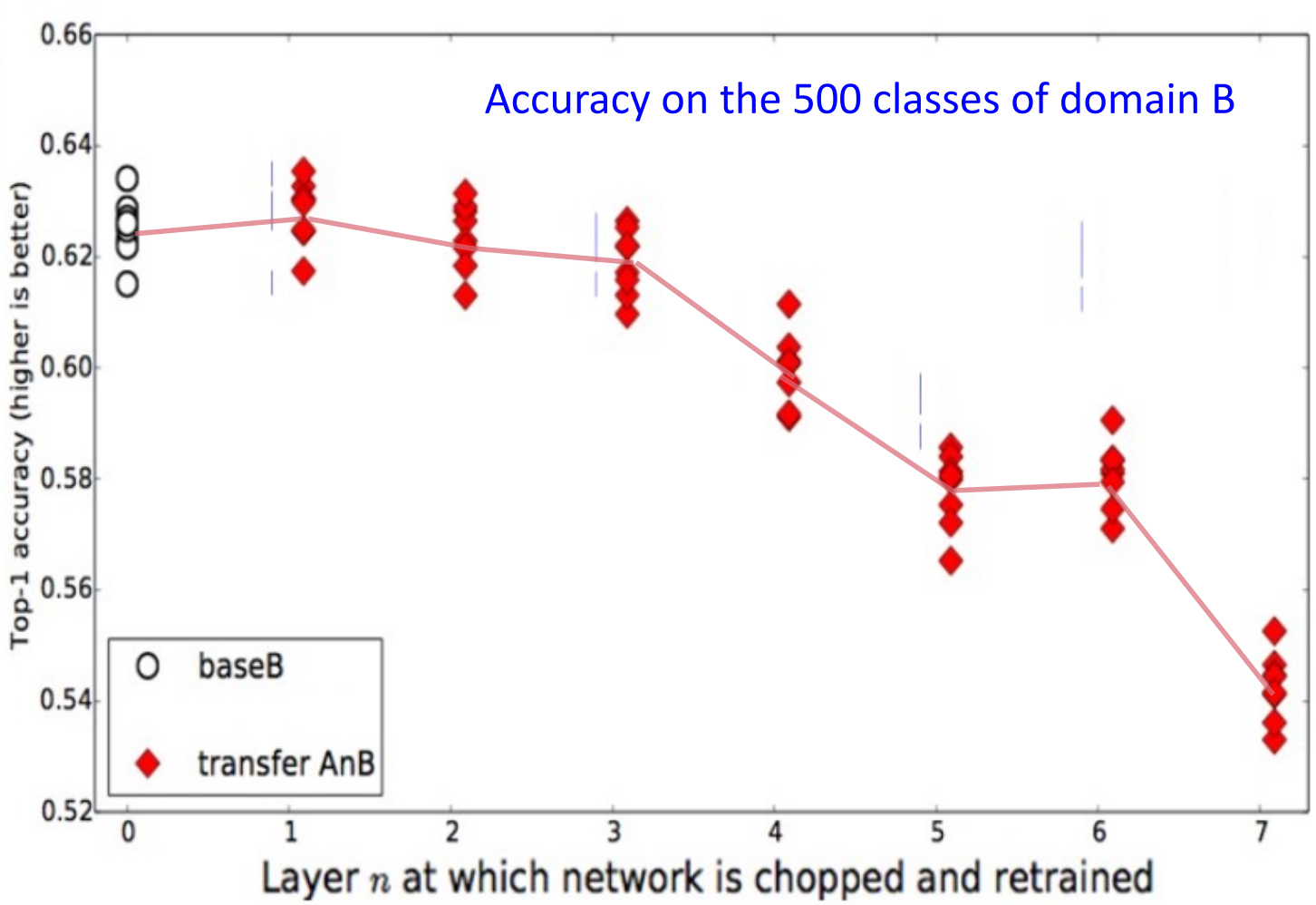
- Base **B** : a NN trained *directly on database* **B** (500 random classes)
- Selffer **BnB** (self-transfer):
  - A number of the first layers are frozen, and **re-training** is done on **the last ones**
- Selffer **BnB+** (self-transfer + retraining):
  - A number of the first layers are frozen, 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):



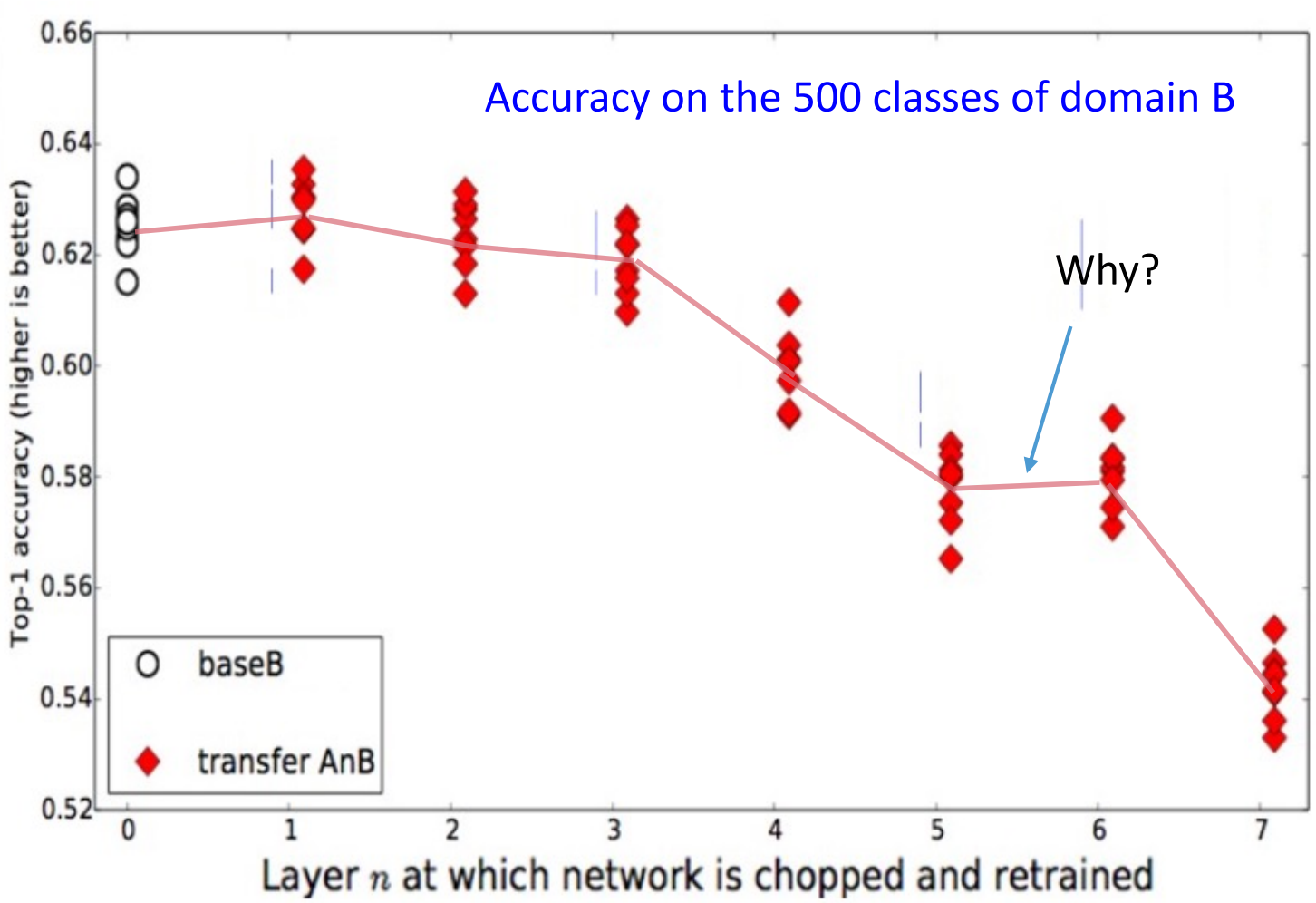
It is clear that the **higher** the layer, the **more specific** it is to task A

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



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

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



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

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

Results

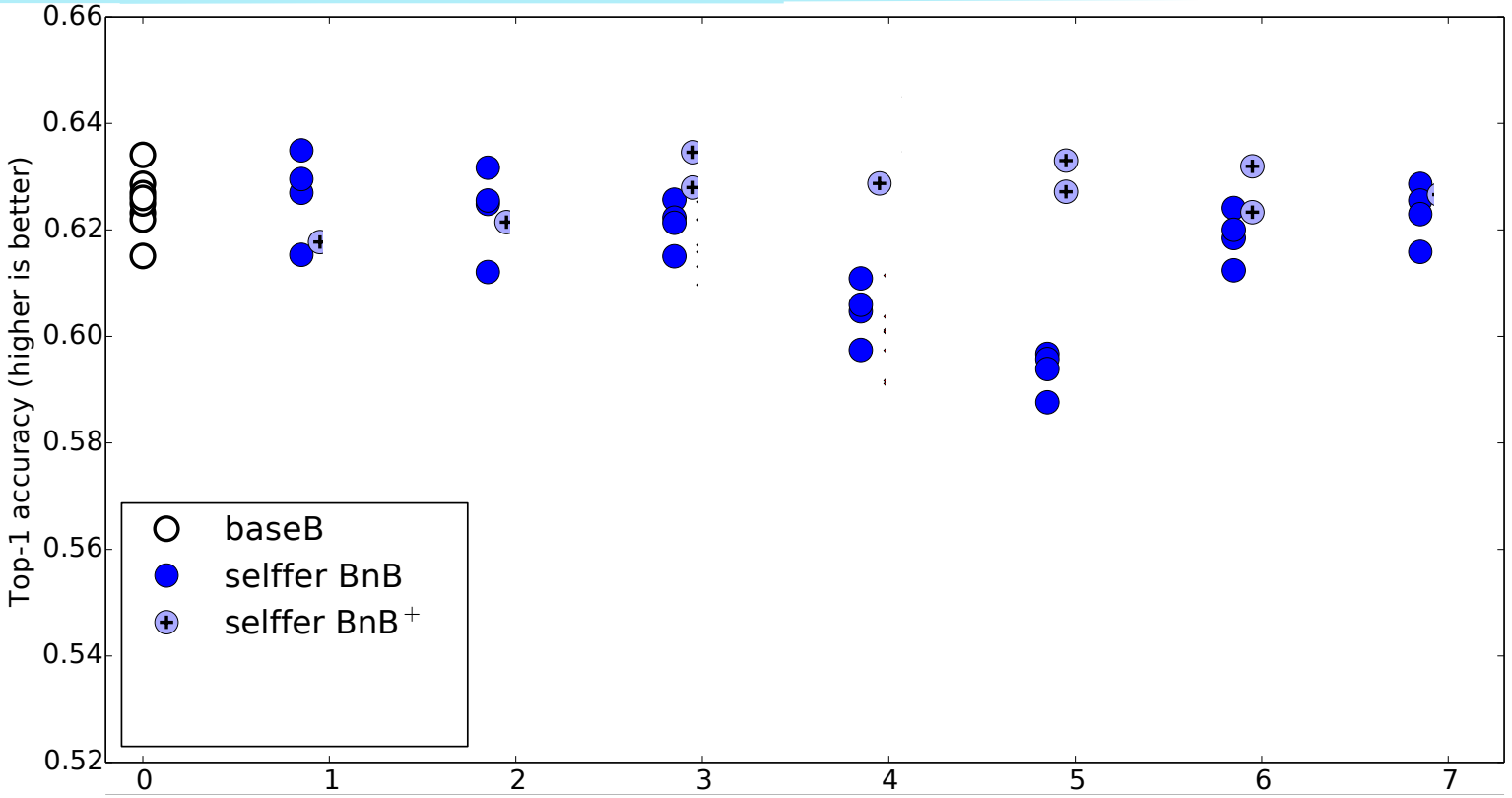


Figure 2: The results from this paper's main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B splits. Each dark blue dot represents a BnB network. Light blue points represent BnB$^+$ networks, or fine-tuned versions of BnB.

Results

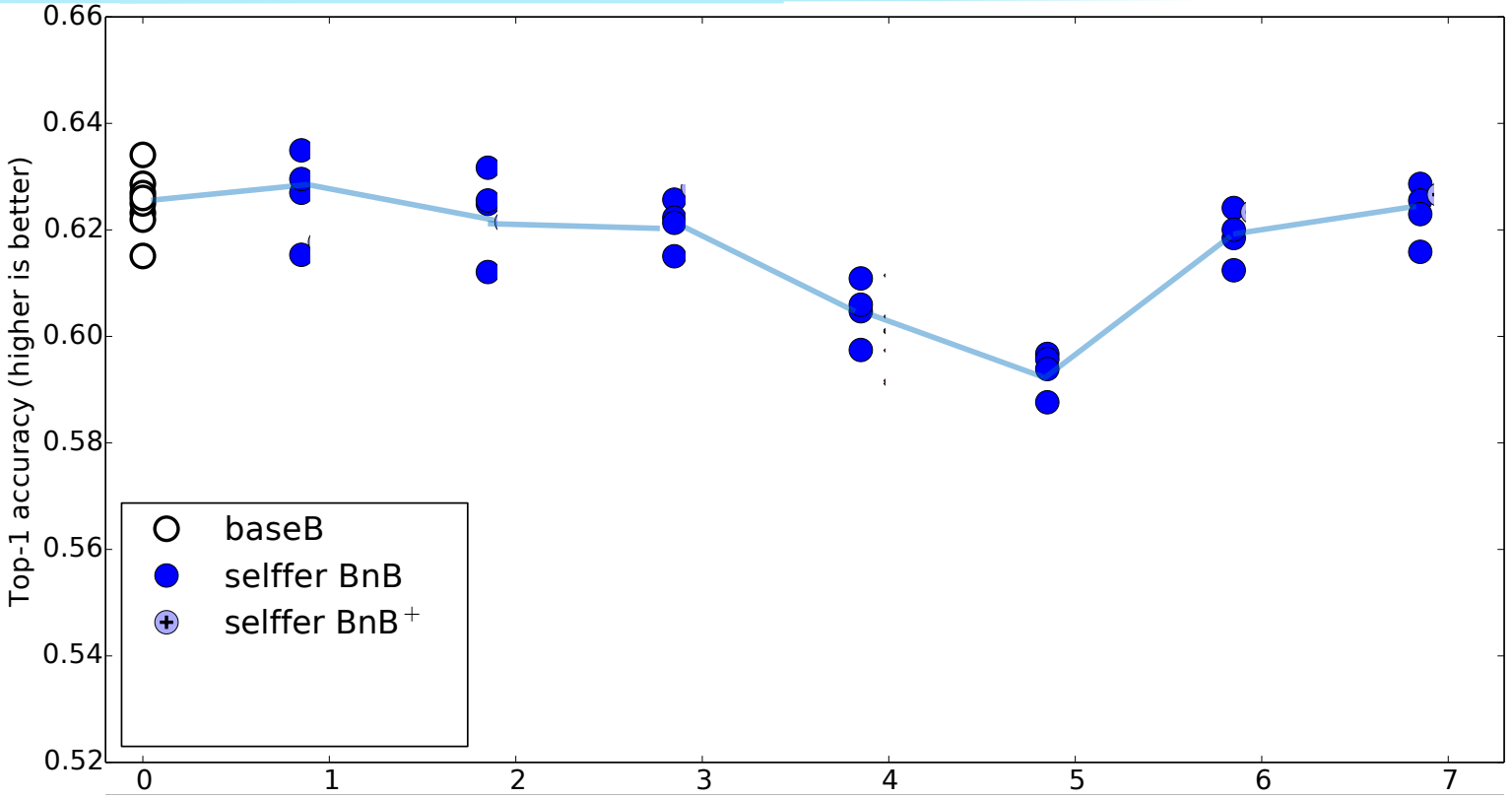


Figure 2: The results from this paper's main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B splits. Each dark blue dot represents a BnB network. Light blue points represent BnB$^+$ networks, or fine-tuned versions of BnB.

Results

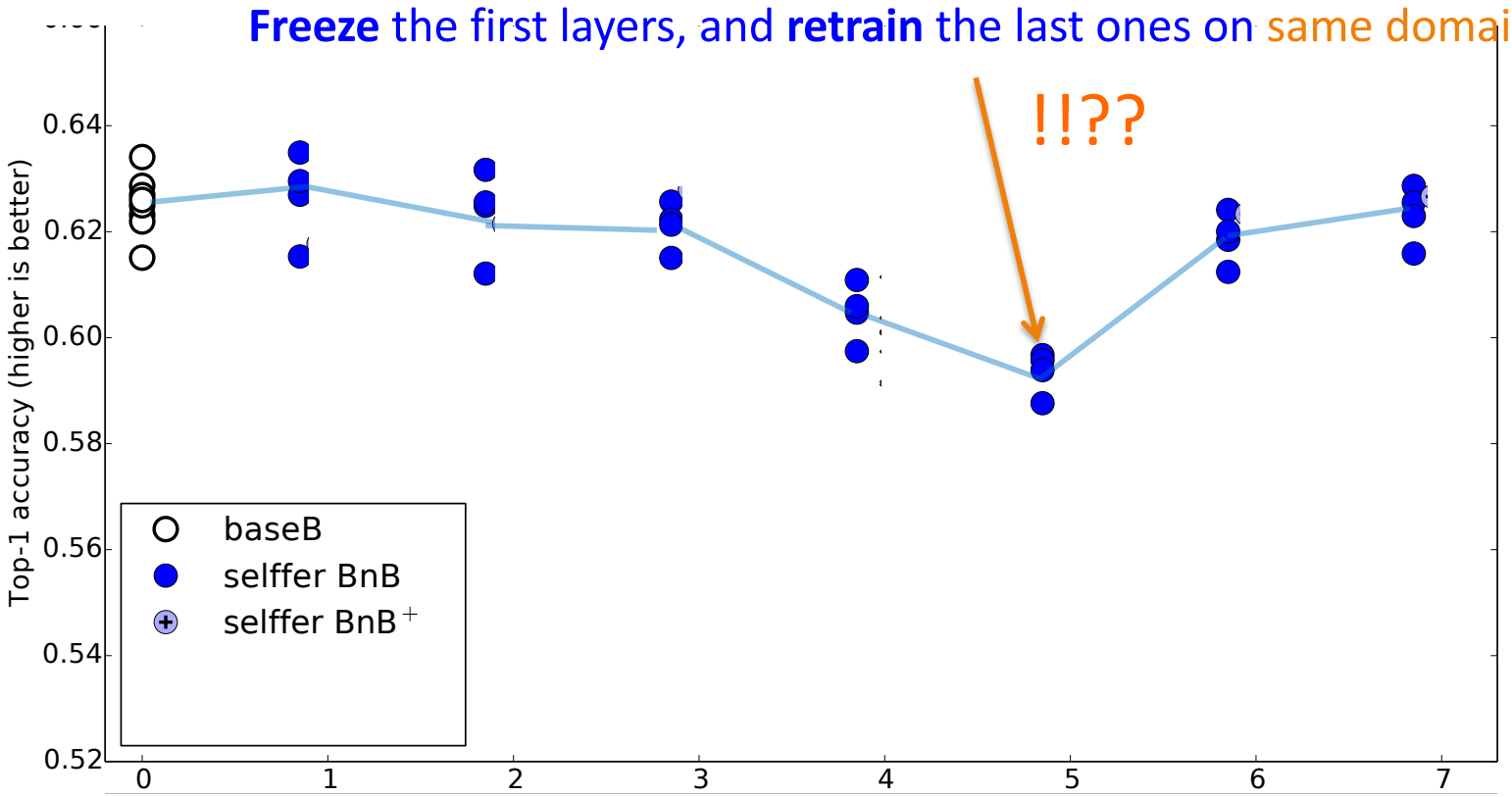


Figure 2: The results from this paper's main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B splits. Each dark blue dot represents a BnB network. Light blue points represent BnB$^+$ networks, or fine-tuned versions of BnB.

1

...

1

...

Results

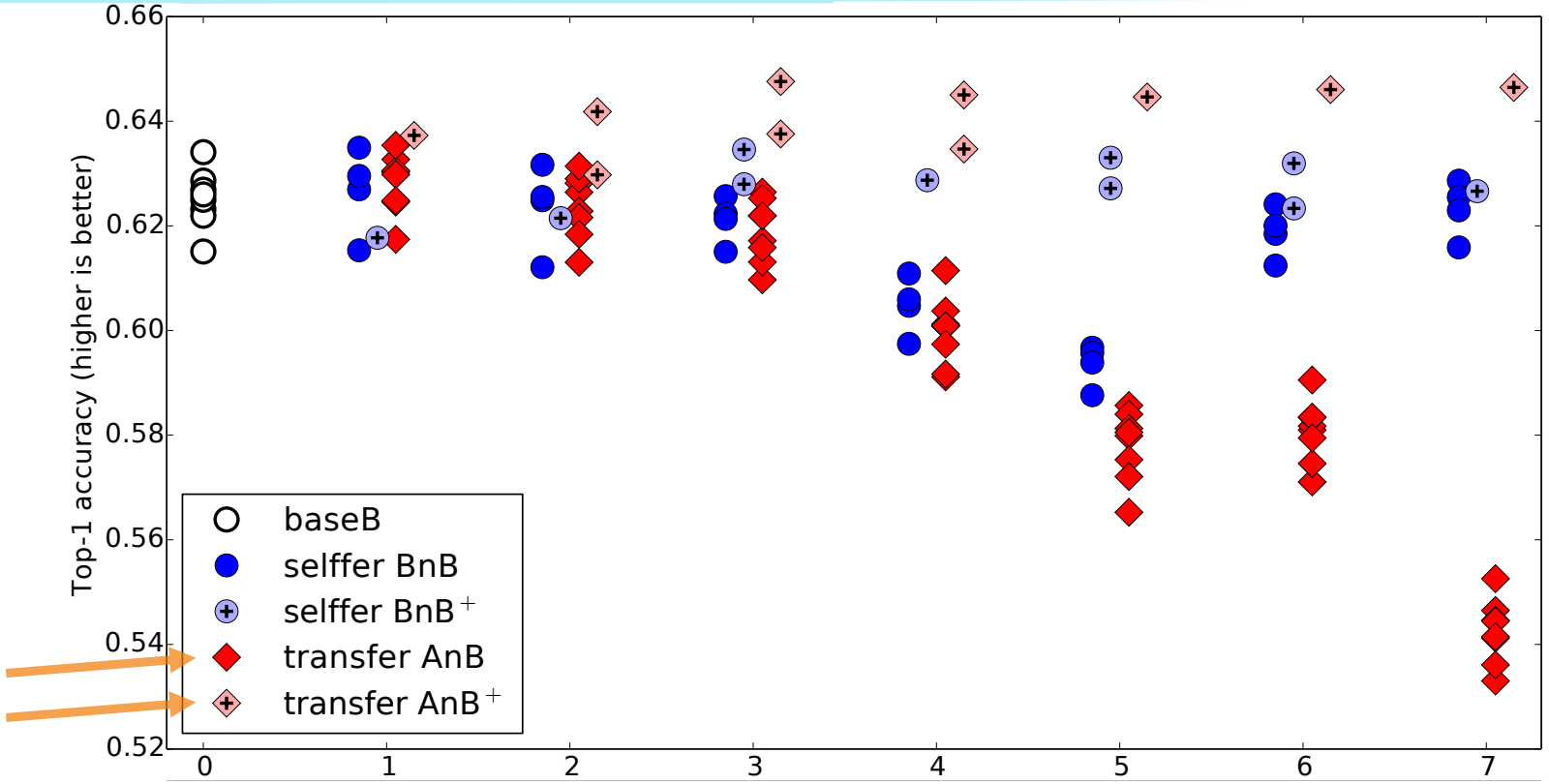
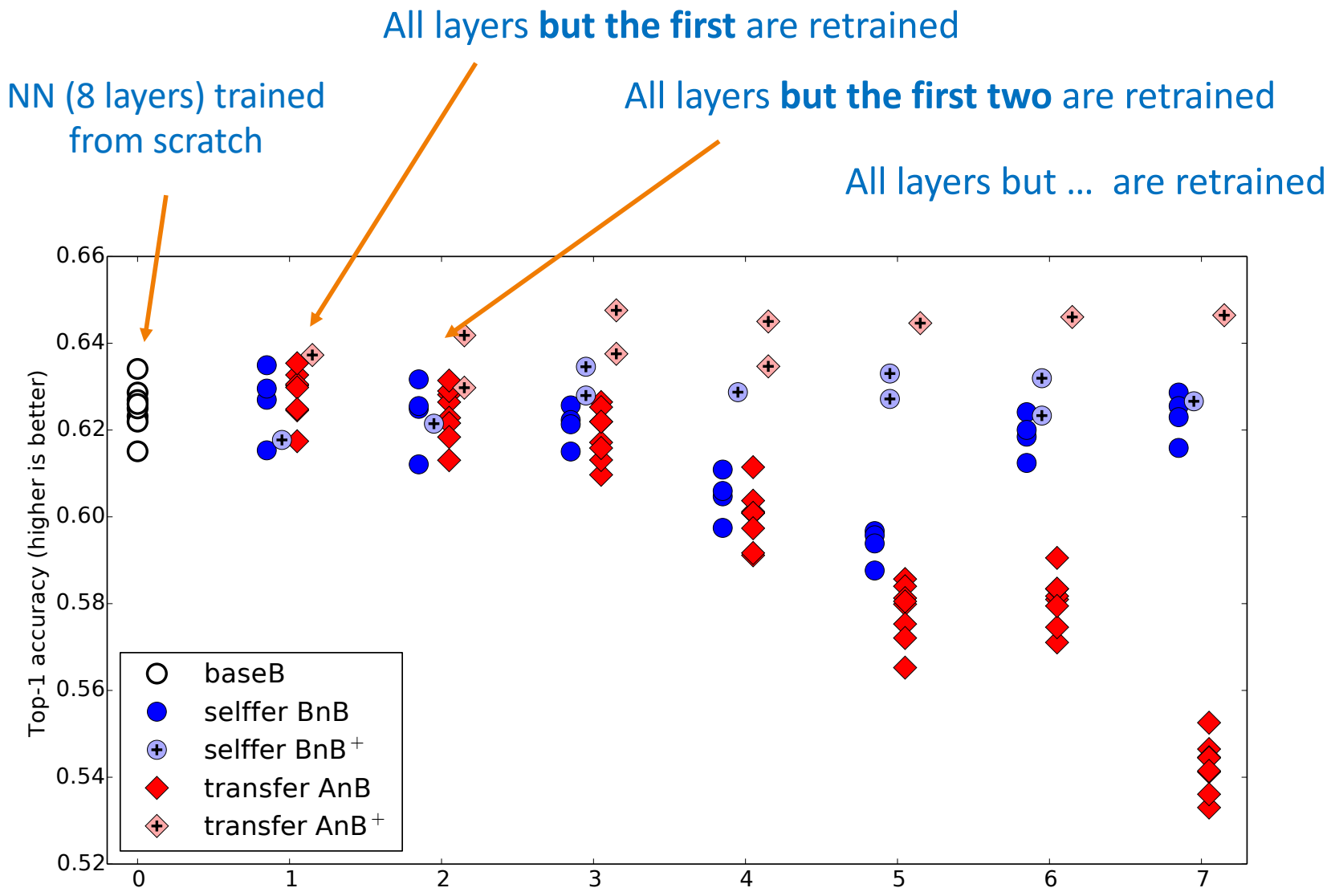
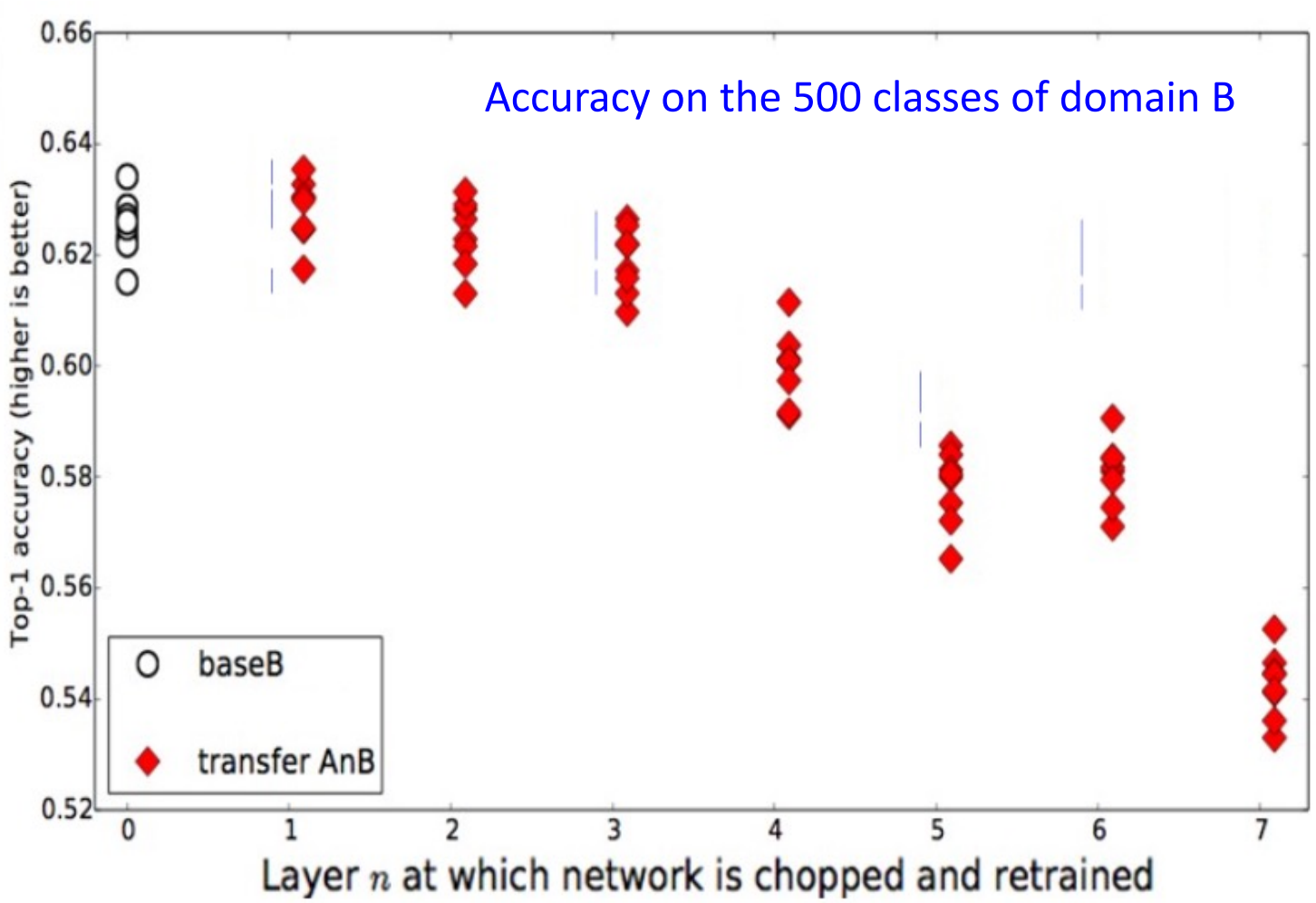


Figure 2: The results from this paper's main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B 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.

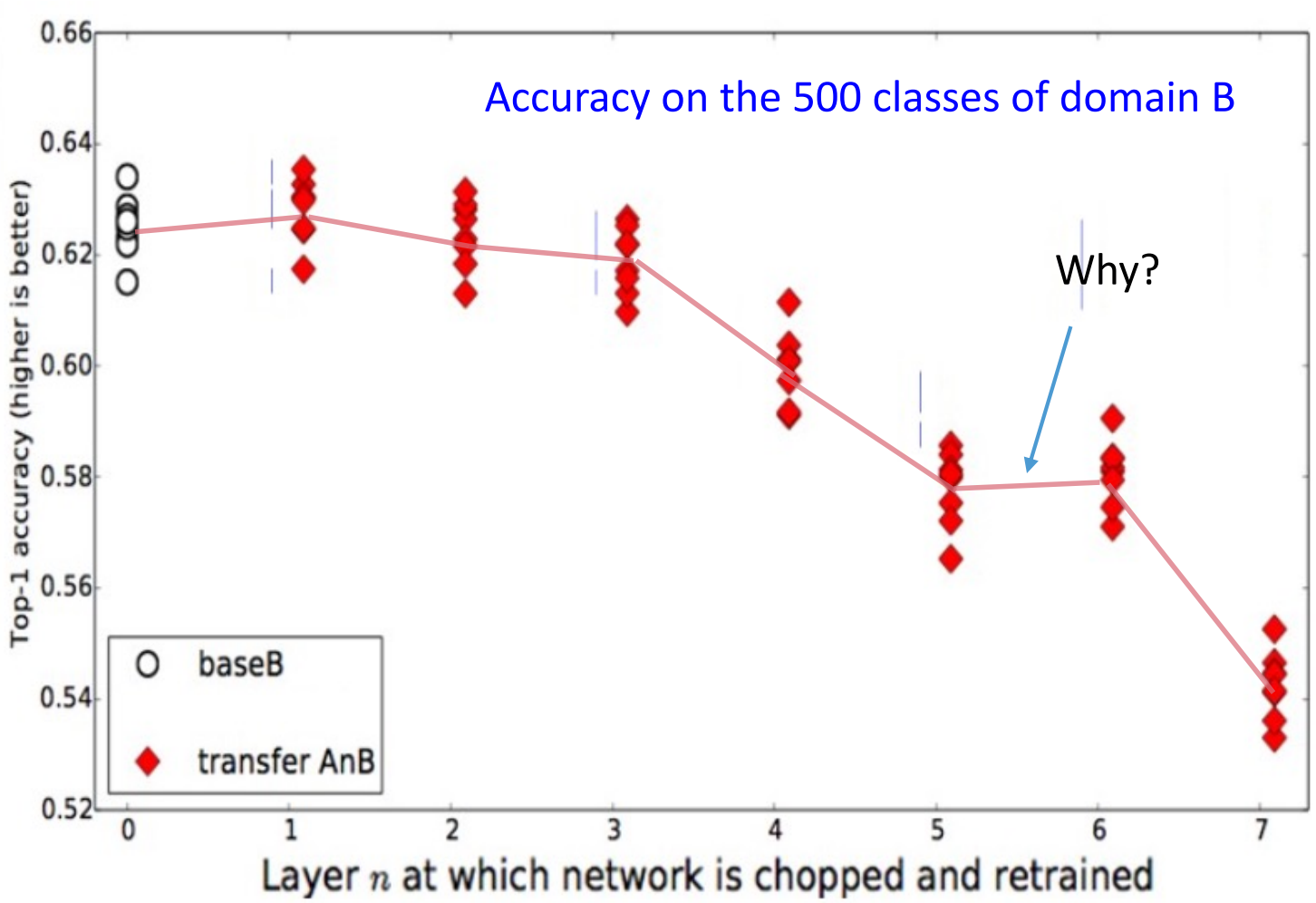
Results: what to think of them?





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

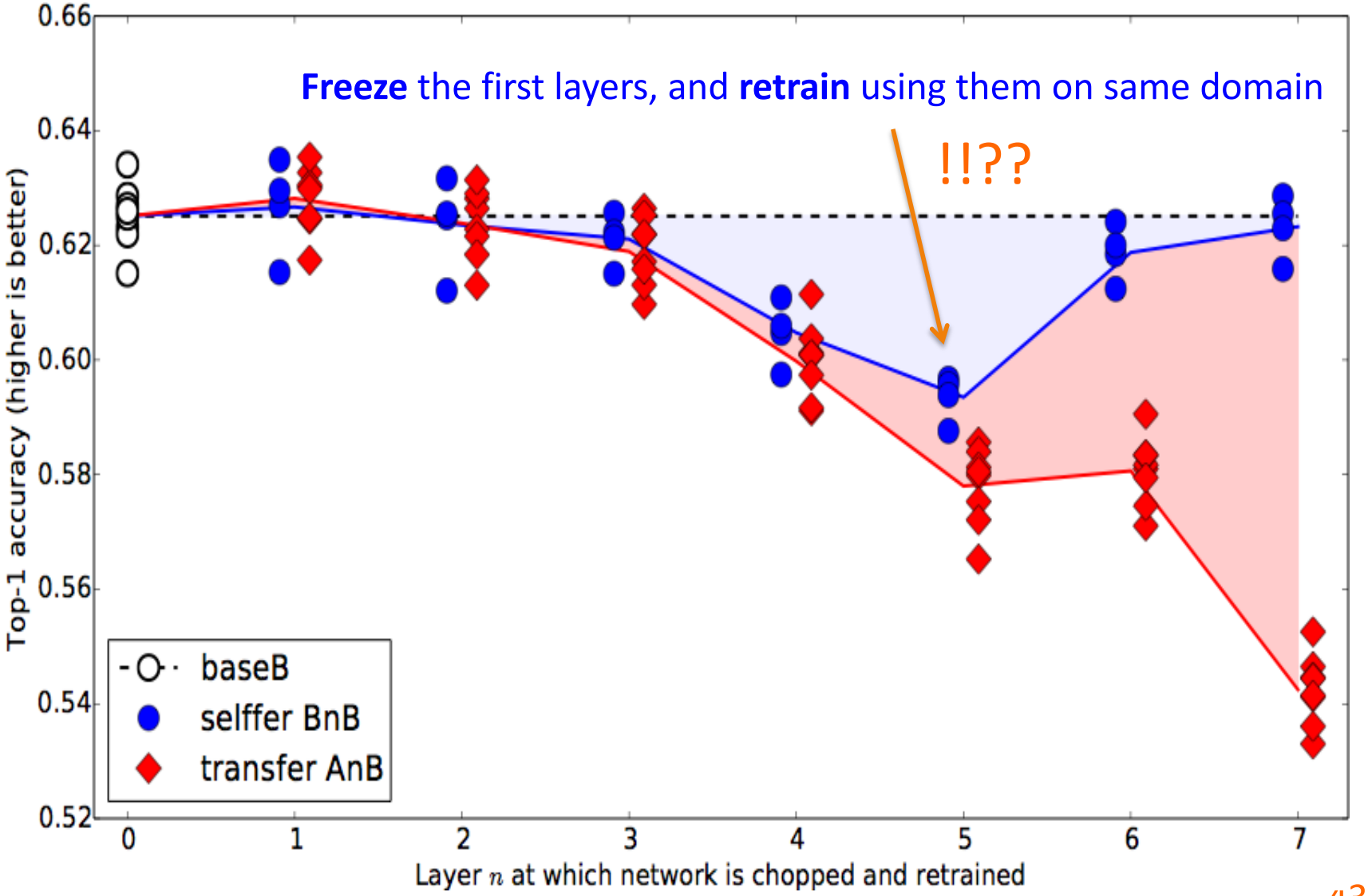
http://slideplayer.com/slide/8370683/



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

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

Interpretation



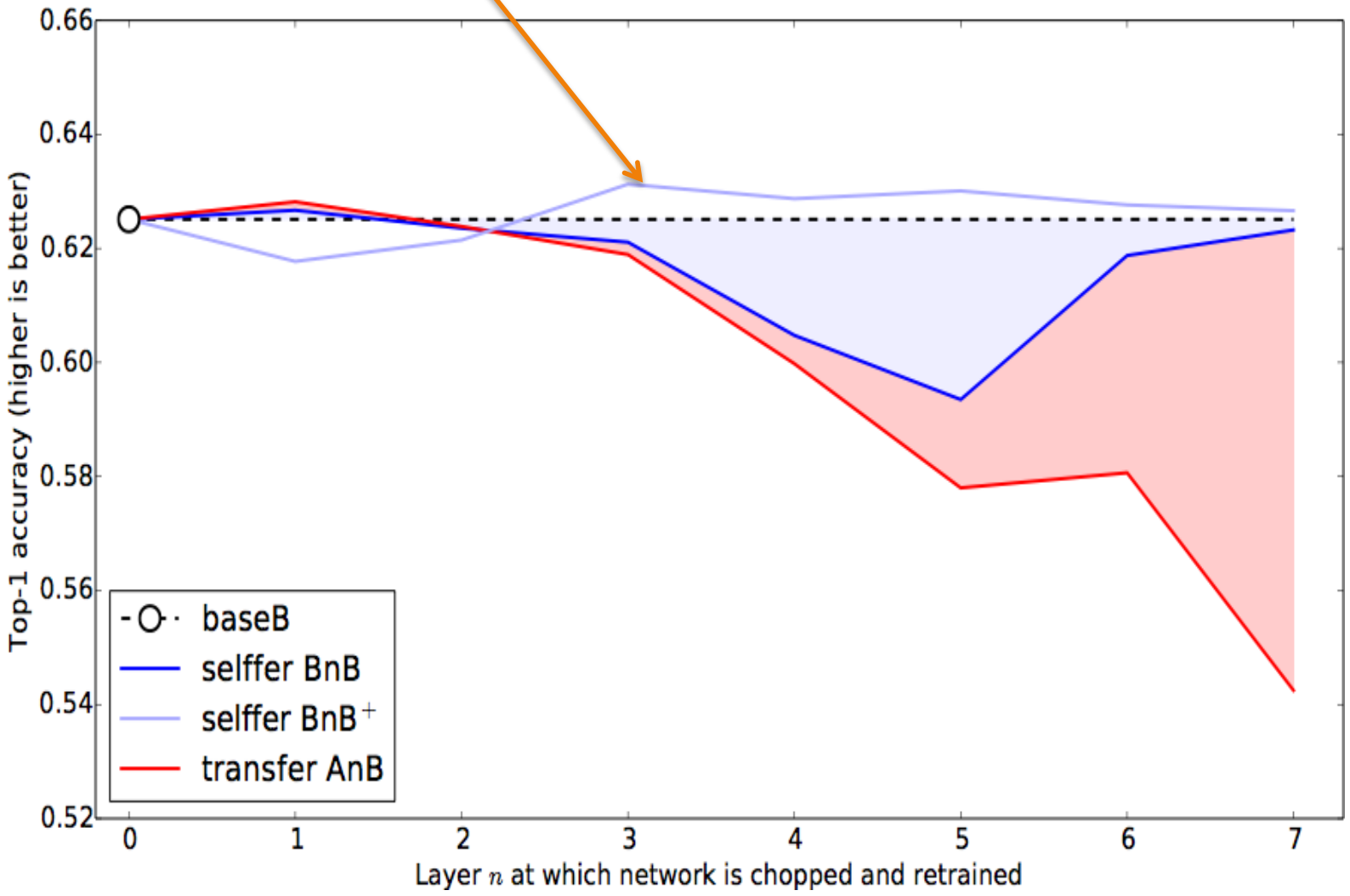
- Remark on the **scientific methodology**

It was essential to look at "*fragile co-adaptation*"

in order to assess the **true effect** of "*representation specificity*"

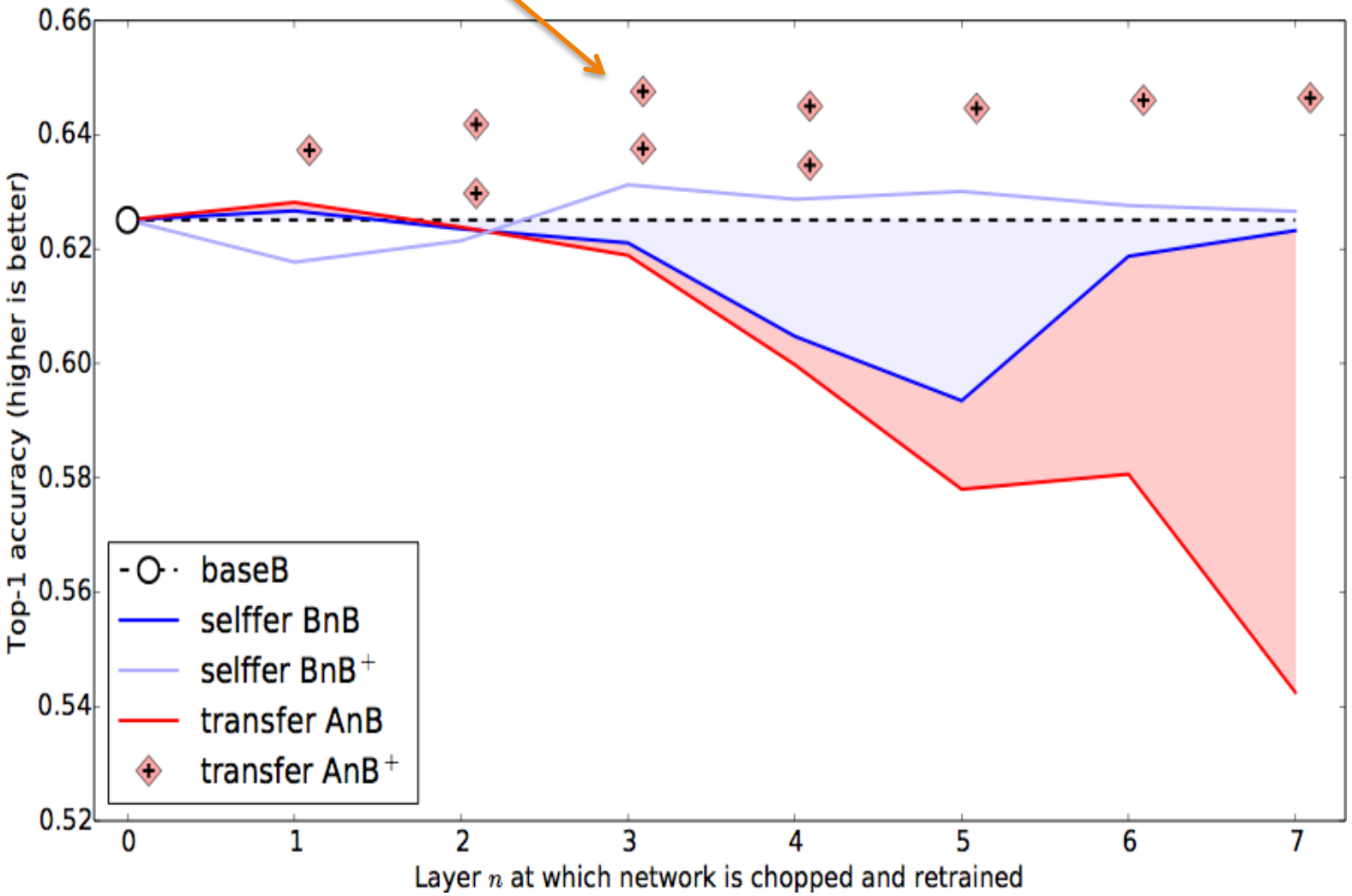
Interpretation

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



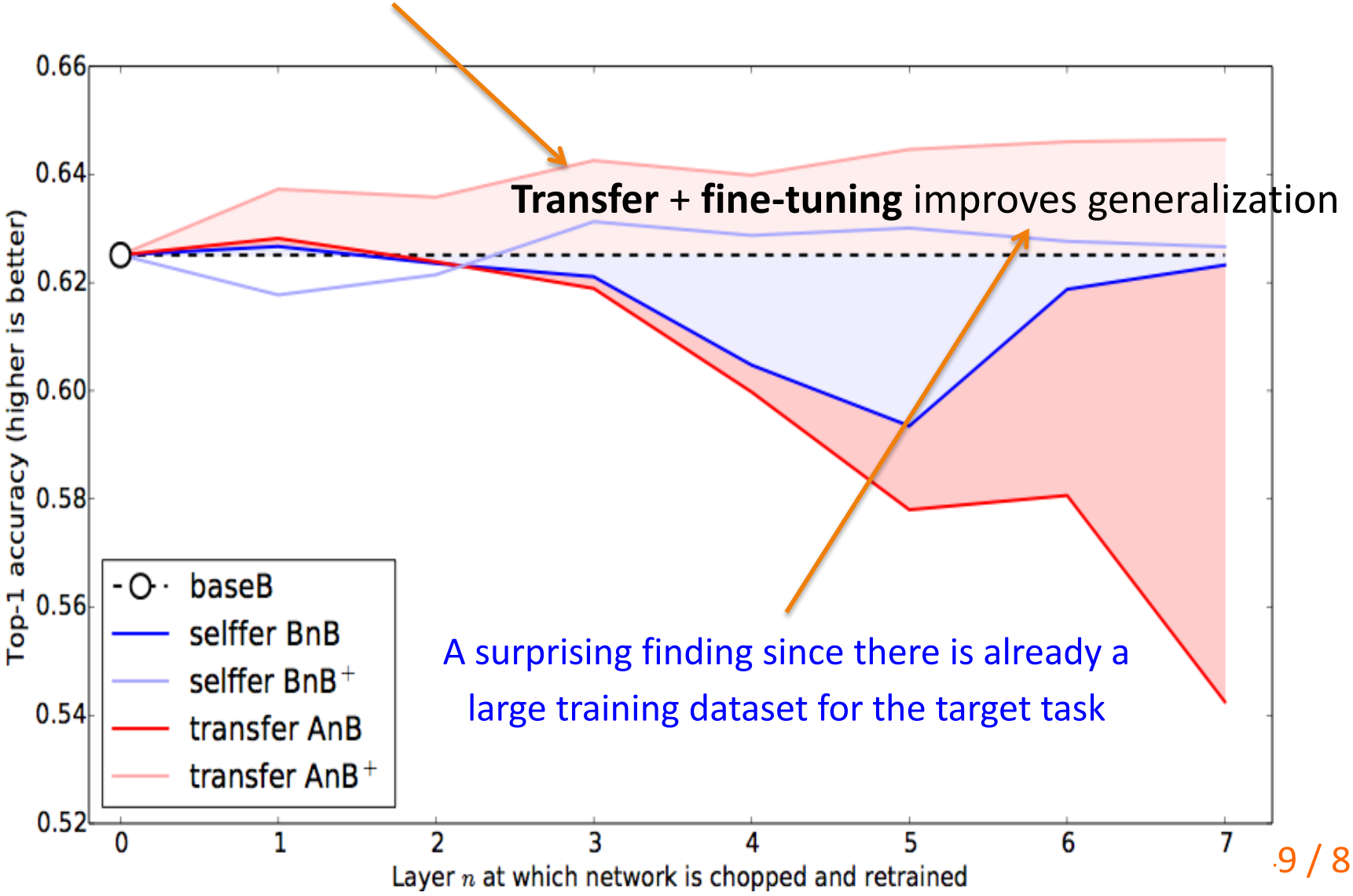
Interpretation

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



Interpretation

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



Transfer + **fine-tuning** improves generalization

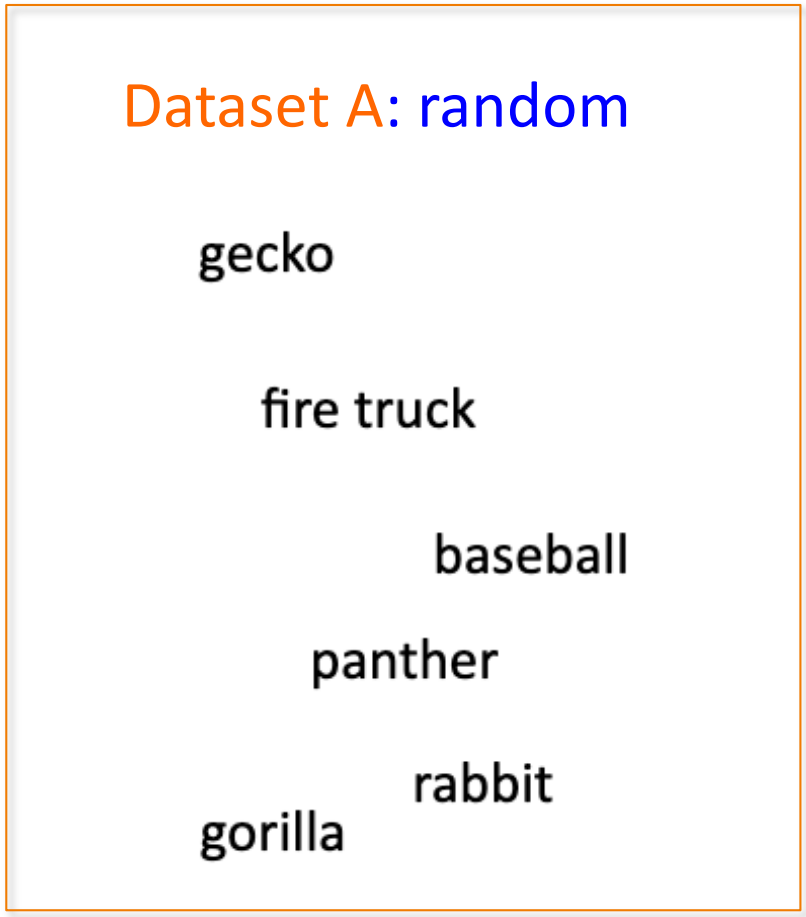
A surprising finding since there is already a large training dataset for the target task

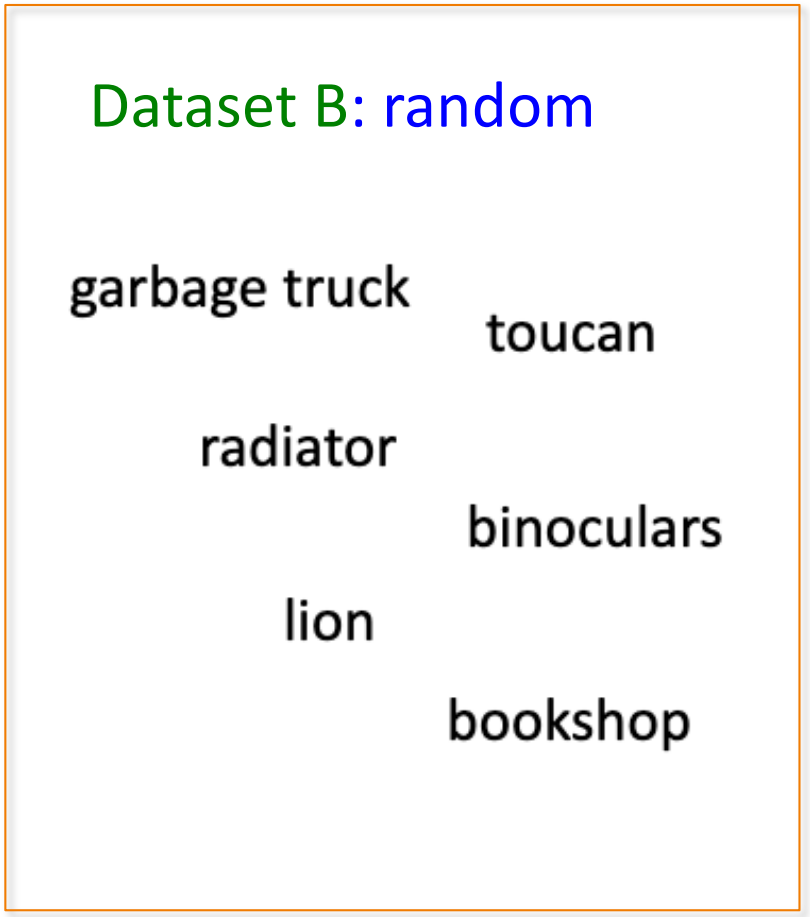
Conclusions of the paper

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





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

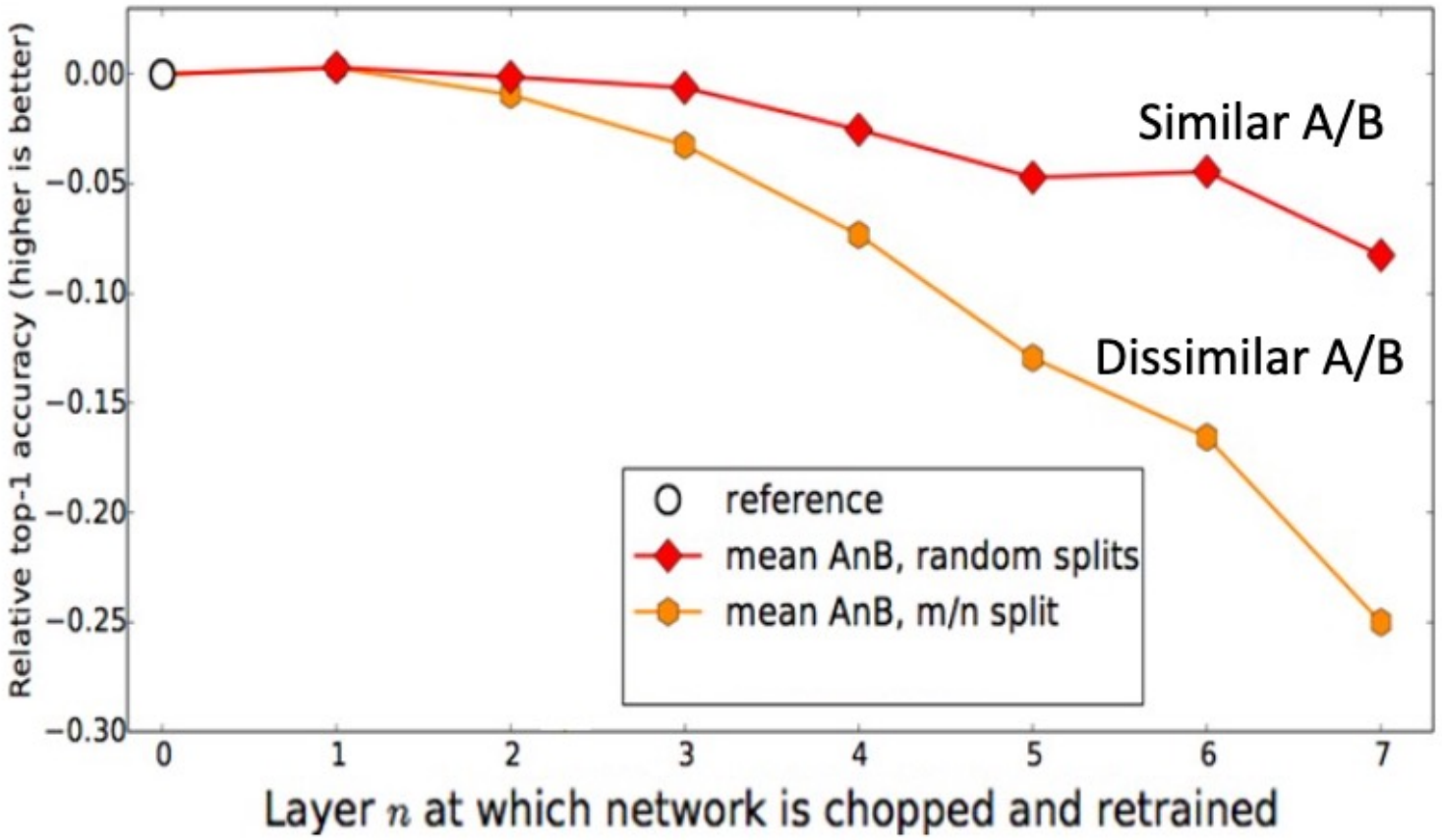
rabbit

panther

lion

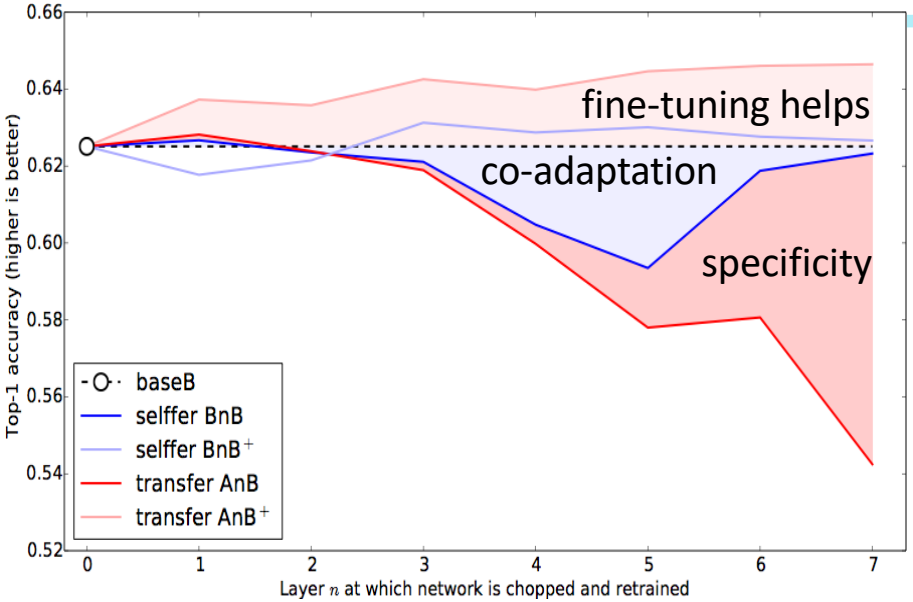
http://slideplayer.com/slide/8370683/

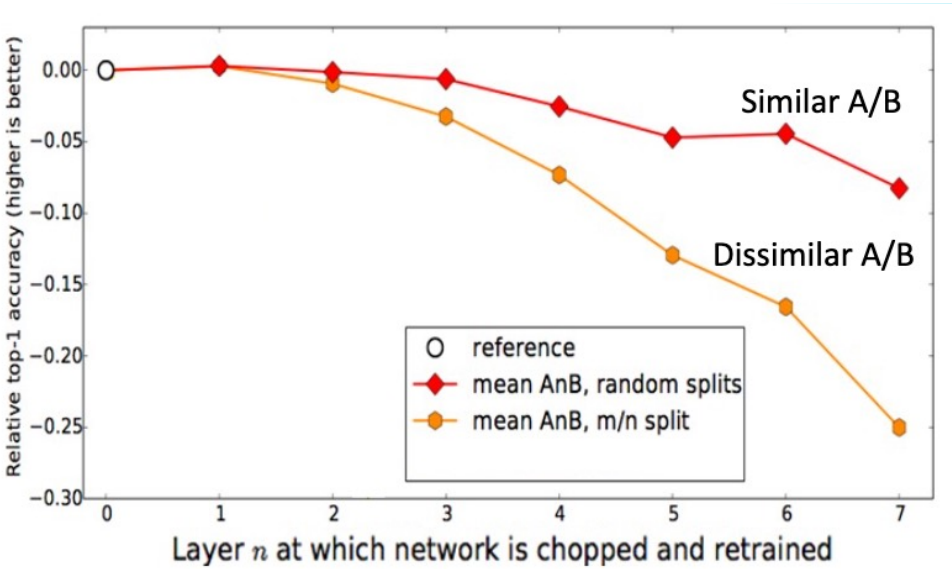
- Comparison



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: reminder
2. Fine tuning: how transferable are features in deep NNs?
3. **Are the features** learned during pretraining of foundational models **general enough** to enable fine tuning on any task?
4. Multi-task learning
5. Conclusions

Fine tuning: the scenario

1. **Pre-training** using a large and diverse training dataset composed of cheap examples that are somehow related to the task of interest

2. The network is **adapted** (fine-tuned) using a much smaller dataset composed of examples that are directly related to the task of interest

   – Dataset **too small** to enable direct training from scratch

Fine tuning: the **challenge**

- Ensure that transferred features are **sufficient** to handle new, unseen datasets.
  - A pretrained network that is **missing crucial feature information** may not perform on par with direct learning if there was enough data to learn the target task
- The folklore vision
  - By pretraining these enormous models on "**everything**", they would **learn all the features** we would ever want for any task

Fine tuning: the **question**

Should we invest into building ever **larger all-purpose** foundational models, or into collecting tasks-specific datasets for training **smaller specialized** models?

How should we approach this issue?

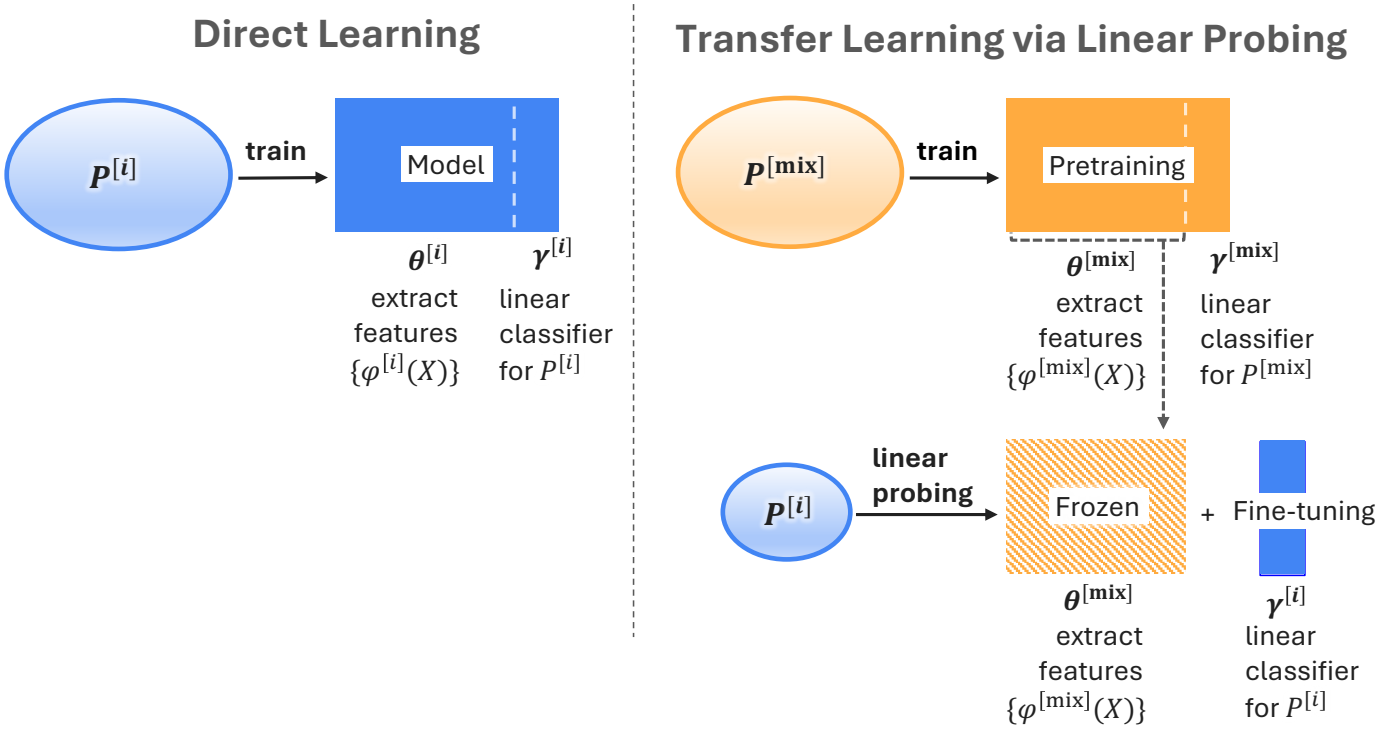
How should we approach this issue?

Yang, Xingyu Alice, Jianyu Zhang, and Léon Bottou. "**These are Not All the Features You are Looking For: A Fundamental Bottleneck In Supervised Pretraining**."
arXiv preprint arXiv:2506.18221 (2025).

The general idea

1. Make sure that the distribution of examples for the **target task** is included in the distribution of examples used for training the foundation model (a very favorable assumption)

2. **Check** whether, under this very favorable scenario, the fine-tuned foundation model can equal the performance of a directly trained model on the target data



Question: Do features pretrained using $P^{[mix]}$ work nearly as well as features learned directly for target $P^{[i]}$?

- A neural network *f* has the form:

$$f(X;\theta,\gamma) = \sum_{\ell} \underbrace{\varphi_\ell(X;\theta)}_{\text{feature extractor}} \cdot \underbrace{\gamma_\ell}_{\text{classifier}}$$

  - Training transforms the raw input *X* into **real-valued features** using parameter θ
  - The linear classification layer combines the extracted features to produce the final prediction.

- Directly training a neural network on the target data yields the trained model $f(\cdot\,; \theta^{[i]}, \gamma^{[i]})$
  - The model learns both **feature parameters** $\theta^{[i]}$ and **classifier weights** $\gamma^{[i]}$ optimized for the target distribution $P^{[i]}$

- Pretraining on $P^{[\mathrm{mix}]}$ yields feature parameters $\theta^{[\mathrm{mix}]}$ which are frozen and only the linear classifier weights γ_ℓ are fine-tuned on $P^{[i]}$

Question

- Do the **solutions** $f(\cdot\,; \theta^{[i]}, \gamma^{[i]})$ optimized for the target distribution $P^{[i]}$ can be matched or approximated by **linearly combining** features learned by pretraining $\varphi_\ell(X; \theta^{[\mathrm{mix}]})$

Potential problem

- **Stochastic gradient descent** in deep learning networks tend to favor **sparse solutions**, removing features deemed redundant
- The remaining features **depend** on the *order* of the training examples and the *initialization*

Do the **remaining features** contain ones that are **useful** for **task *i***?

Simple counter-example

- Imagine a feature extractor which can learn only **two feature extraction functions**.
  - Depending on parameters $\Phi(X) = \big(\varphi_1(X), \varphi_2(X)\big)$, the space spanned by the selected features can have 0, 1 or 2 dimensions.

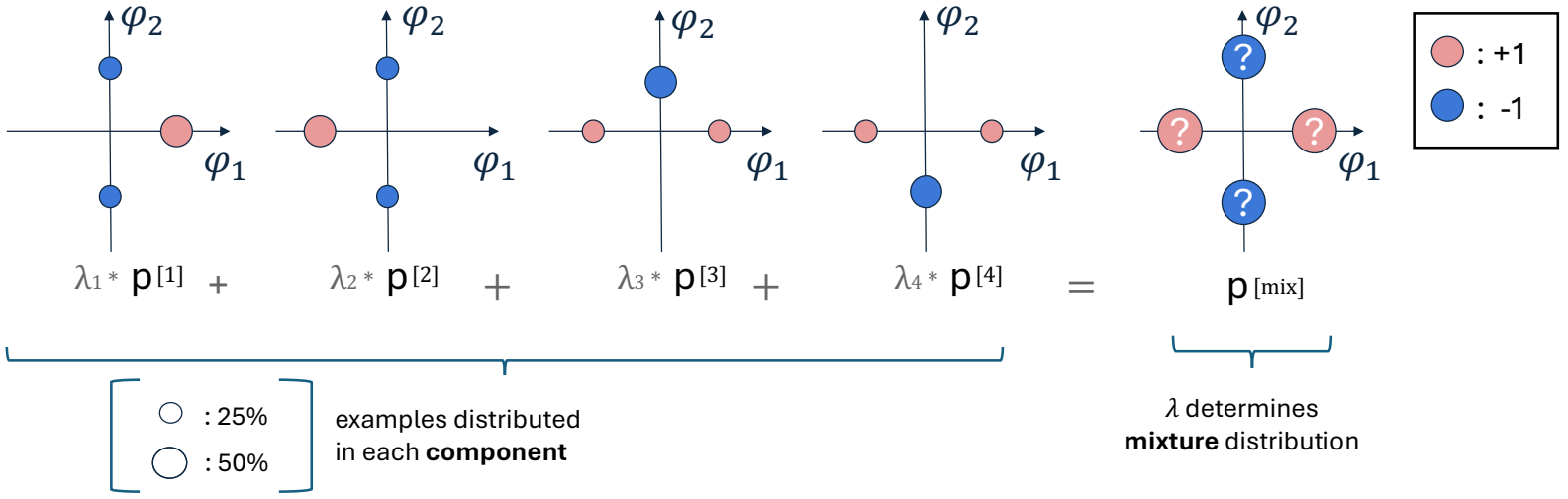
- Consider four **individual distributions** in this space

- $P^{[1]}[\,\Phi(X){=}(+1,0),\ Y{=}+1\,] = \frac{1}{2}$ and $P^{[1]}[\,\Phi(X){=}(0,\pm1),\ Y{=}-1\,] = \frac{1}{4}$.
- $P^{[2]}[\,\Phi(X){=}(-1,0),\ Y{=}+1\,] = \frac{1}{2}$ and $P^{[2]}[\,\Phi(X){=}(0,\pm1),\ Y{=}-1\,] = \frac{1}{4}$.
- $P^{[3]}[\,\Phi(X){=}(0,+1),\ Y{=}-1\,] = \frac{1}{2}$ and $P^{[3]}[\,\Phi(X){=}(\pm1,0),\ Y{=}+1\,] = \frac{1}{4}$.
- $P^{[4]}[\,\Phi(X){=}(0,-1),\ Y{=}-1\,] = \frac{1}{2}$ and $P^{[4]}[\,\Phi(X){=}(\pm1,0),\ Y{=}+1\,] = \frac{1}{4}$.

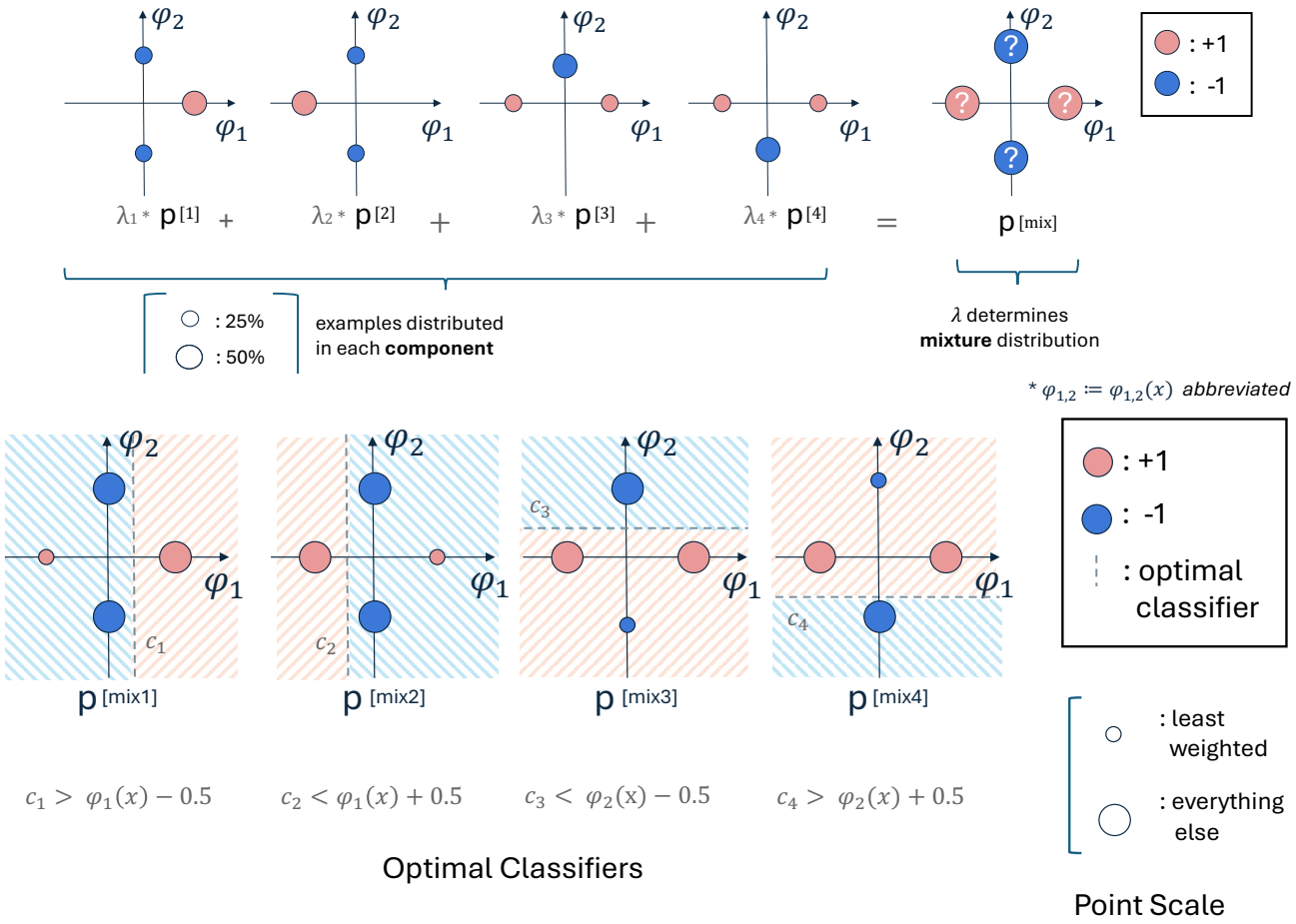
$$P^{[\mathrm{mix}]} = \lambda_1 \cdot P^{[1]} + \lambda_2 \cdot P^{[2]} + \lambda_3 \cdot P^{[3]} + \lambda_4 \cdot P^{[4]} \quad \text{where } \lambda_i > 0,\ \sum_{i=1} \lambda_i = 1$$

- $P^{[1]}[\,\Phi(X)=(+1,0),\ Y=+1\,]=\frac{1}{2}$ and $P^{[1]}[\,\Phi(X)=(0,\pm 1),\ Y=-1\,]=\frac{1}{4}$.
- $P^{[2]}[\,\Phi(X)=(-1,0),\ Y=+1\,]=\frac{1}{2}$ and $P^{[2]}[\,\Phi(X)=(0,\pm 1),\ Y=-1\,]=\frac{1}{4}$.
- $P^{[3]}[\,\Phi(X)=(0,+1),\ Y=-1\,]=\frac{1}{2}$ and $P^{[3]}[\,\Phi(X)=(\pm 1,0),\ Y=+1\,]=\frac{1}{4}$.
- $P^{[4]}[\,\Phi(X)=(0,-1),\ Y=-1\,]=\frac{1}{2}$ and $P^{[4]}[\,\Phi(X)=(\pm 1,0),\ Y=+1\,]=\frac{1}{4}$.

$$P^{[\mathrm{mix}]}=\lambda_1\cdot P^{[1]}+\lambda_2\cdot P^{[2]}+\lambda_3\cdot P^{[3]}+\lambda_4\cdot P^{[4]}\quad\text{where }\lambda_i>0,\ \sum_{i=1}^{}\lambda_i=1$$



...



- For mixtures **mix1** and **mix2**, feature $\varphi_1(X)$ is enough
  - But problems 3 and 4 cannot be solved
- For mixtures **mix3** and **mix4**, feature $\varphi_2(X)$ is enough
  - But problems 1 and 2 cannot be solved

Analysis

The authors identify an *information saturation bottleneck*

- Deep NNs **struggle to learn new features** when they have encoded similar competing features during training
- Because of their sparsity bias, they **may permanently lose critical feature** needed for effective transfer

In practice on genomic tasks

- Comparison of
  - **transfer learning** from Genomic Foundation Models (GFM)
  - and **direct** supervised learning

	Model	Model Size	Pretraining Base-Pairs	Average Score ↑	Average Rank ↓	Mean %Imp.↑	Median %Imp.↑
Foundation Models	Enformer	252M	4B	0.569	11.86	27.73	27.91
	NT-1000G (500M)	500M	20.5T	0.625	10.52	33.48	36.74
	NT-1000G (2.5B)	2.5B	20.5T	0.656	7.0	36.58	40.86
	NT-Multispecies (500M)	500M	174B	0.700	3.81	40.76	45.07
	NT-Multispecies (2.5B)	2.5B	174B	0.697	4.08	40.51	45.52
	DNABERT-2	117M	32.5B	0.680	6.88	38.65	43.59
	HyenaDNA-1K	1.6M	3.2B	0.708	6.92	41.2	43.36
	HyenaDNA-32K	1.6M	3.2B	0.630	10.22	33.96	36.93
	Caduceus-PS	1.9M	35B	0.689	6.69	39.08	41.38
	Caduceus-PH	1.9M	35B	0.725	4.69	42.63	45.01
Supervised Models	Wide ResNet	2.0M	0	0.694	6.83	37.16	43.08
	UNet	4.5M	0	0.68	7.78	38.67	42.69
	DASHA (our workflow)	10.5M	0	**0.761**	**3.69**	**46.33**	**49.08**

Transferring regularities in common

Outline

1. Transfer learning: reminder
2. Fine tuning: how transferable are features in deep NNs?
3. Are the features learned during pretraining of foundational models general enough to enable fine tuning on any task?
4. **Multi-task learning**
5. Conclusions

What is Multi-Task learning (MTL)?

- As soon you try to optimize more than one loss function

  - E.g. From someone's **picture**, trying to guess both
    - The **gender**
    - The **age**
    - The **emotion**

Why Multi-Task learning (MTL)?

- (IF) The tasks at hand are **not unrelated**
  - E.g. From someone's picture, trying to guess both
    - The **gender**
    - The **age**
    - The **emotion**

- It may help to consider them all together:

  better performance with **less** computing resources
  - E.g. guessing the *gender* may help recognize the *emotion* and vice-versa

Rk: There are links with the LUPI framework

Assumption behind MTL

- The **combined learning** of multiple related tasks **can outperform learning each task in isolation**

- MTL allows for **common information** shared between the tasks to be used in the learning process, which leads to better generalization if the tasks are related

- E.g. Learning to **predict the ratings** for several different critics (in different countries) can lead to better performances for **each separate task** (predict the restaurant ratings for a specific critic)

- Learning to **recognize a face** and the **expression** (fear, disgust, anger, …)

- **Multi modality learning**: e.g. vision and proprioception

Possible **relations** between tasks

- All functions to be learn are **close** to each other **in some norm**
  - E.g. functions capturing preferences in users' modeling problems

- Tasks that share a **common underlying representation**
  - E.g. in *human vision*, all tasks use the **same set of features** learnt in the first stages of the visual system (e.g. local filters similar to wavelets)
  - Users may also *prefer* different types of things (e.g. books, movies, music) based on the **same set of features** or **score** functions

Question

How do we choose to

model the **shared information** between the tasks?

- Idea: Some shared underlying constraints
  - E.g. a **low dimensional representation** shared across multiple related tasks
    - By way of a **shared hidden layer** in a neural network
    - By explicitly constraining the **dimensionality of a shared representation**

An approach for the **linear** case: minimizing the distance with a shared weight vector

- T binary classification tasks defined over $\mathcal{X} \times \mathcal{Y}$

$$\mathcal{S} = \{\{(\mathbf{x}_{11}, y_{11}), (\mathbf{x}_{21}, y_{21}), \ldots, (\mathbf{x}_{m1}, y_{m1})\}, \ldots, \{(\mathbf{x}_{1T}, y_{1T}), (\mathbf{x}_{2T}, y_{2T}), \ldots, (\mathbf{x}_{mT}, y_{mT})\}\}$$

$$h_j(\mathbf{x}) = \mathbf{w}_j \cdot \mathbf{x}$$ **Linear** hypotheses

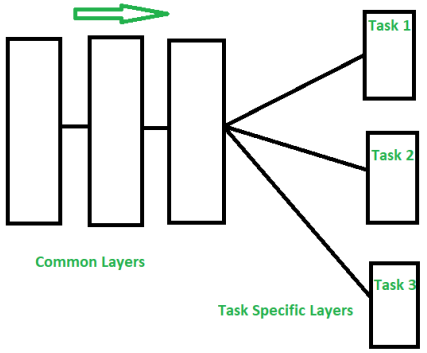
That share a weight vector $\mathbf{w}_j = \mathbf{w}_0 + \mathbf{v}_j$

$$h_1^\star, \ldots, h_T^\star = \underset{\mathbf{w}_0, \mathbf{v}_j, \xi_{ij}}{\text{Argmin}} \left\{ \sum_{j=1}^{T} \sum_{i=1}^{m} \xi_{ij} + \frac{\lambda_1}{T} \sum_{j=1}^{T} ||\mathbf{v}_j||^2 + \lambda_2 ||\mathbf{w}_0||^2 \right\}$$

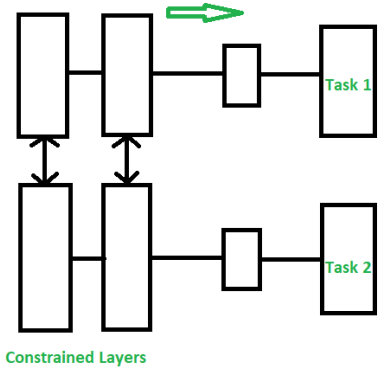
MTL with deep neural networks

- Approaches

1. Sharing **features** (first layers) and have multiple task-specific heads



1. **Soft**-features or parameters sharing



- Multi-Task Learning induces a bias that prefers hypotheses that can “explain” all tasks

- Beware:
  - Can lead to **worse** performance if the tasks are **unrelated** or **adversarially** related

- Question: *how to measure the **relatedness** of learning tasks*?

...

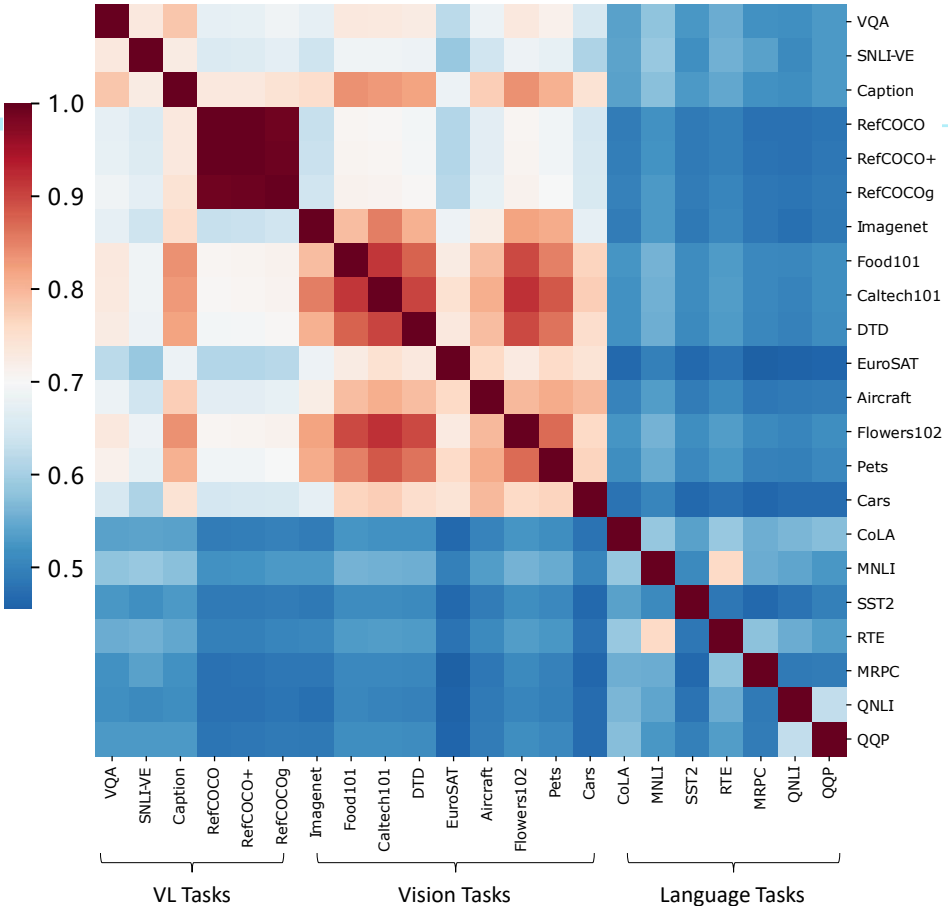


Figure 1. Heatmap of the predicted task similarities, composed of both unimodal and multimodal tasks. Vision-language tasks are more similar to vision tasks compared to language tasks. Best viewed in color.

WU, Chengyue, WANG, Teng, GE, Yixiao, et al. **$\pi $-Tuning: Transferring Multimodal Foundation Models with Optimal Multi-task Interpolation**. In *International Conf. on Machine Learning* (ICML). PMLR, 2023. p. 37713-37727.

Outline

1. Transfer learning: definition
2. Transfering representations
3. IRM: Invariant Risk Minimization
4. Multi-task learning
5. Conclusions

- Transferring representations is not that straightforward
  - The **relevant hidden layer** depends on the conditions
  - **Necessary features** can be **missing**