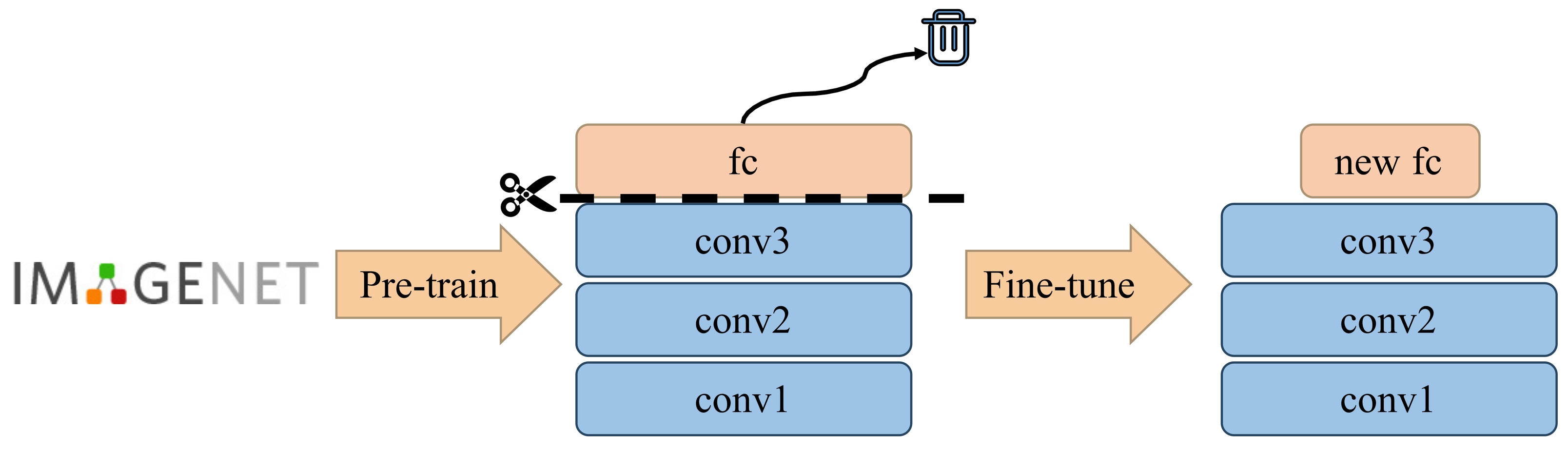
Co-Tuning for Transfer Learning

Kaichao You, Zhi Kou, Mingsheng Long, Jianmin Wang

Status Quo of Transfer Learning

- The naïve approach is still very popular

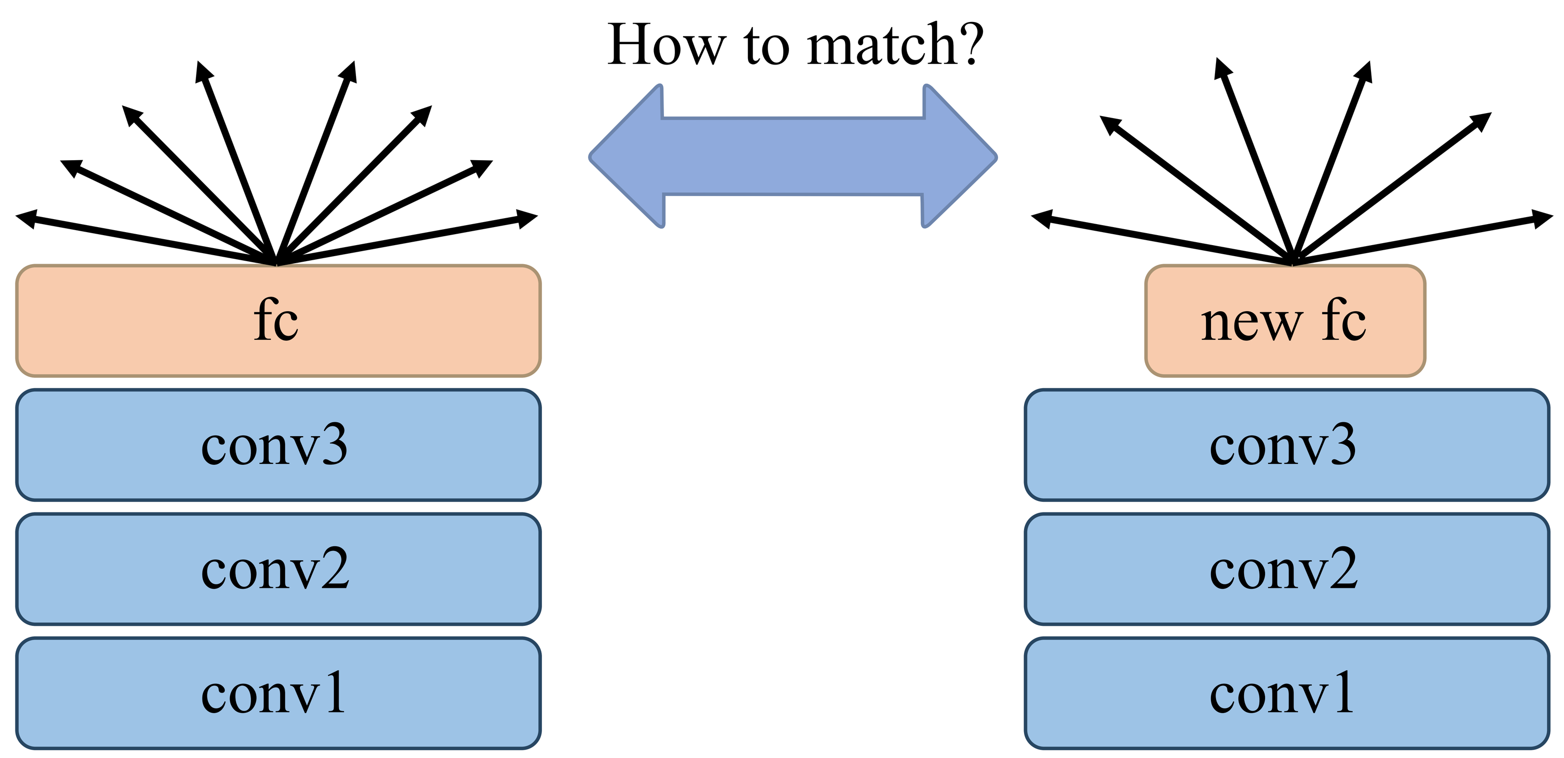


- ... but very wasteful.
  - These task-specific layers take up many parameters in pre-trained models.

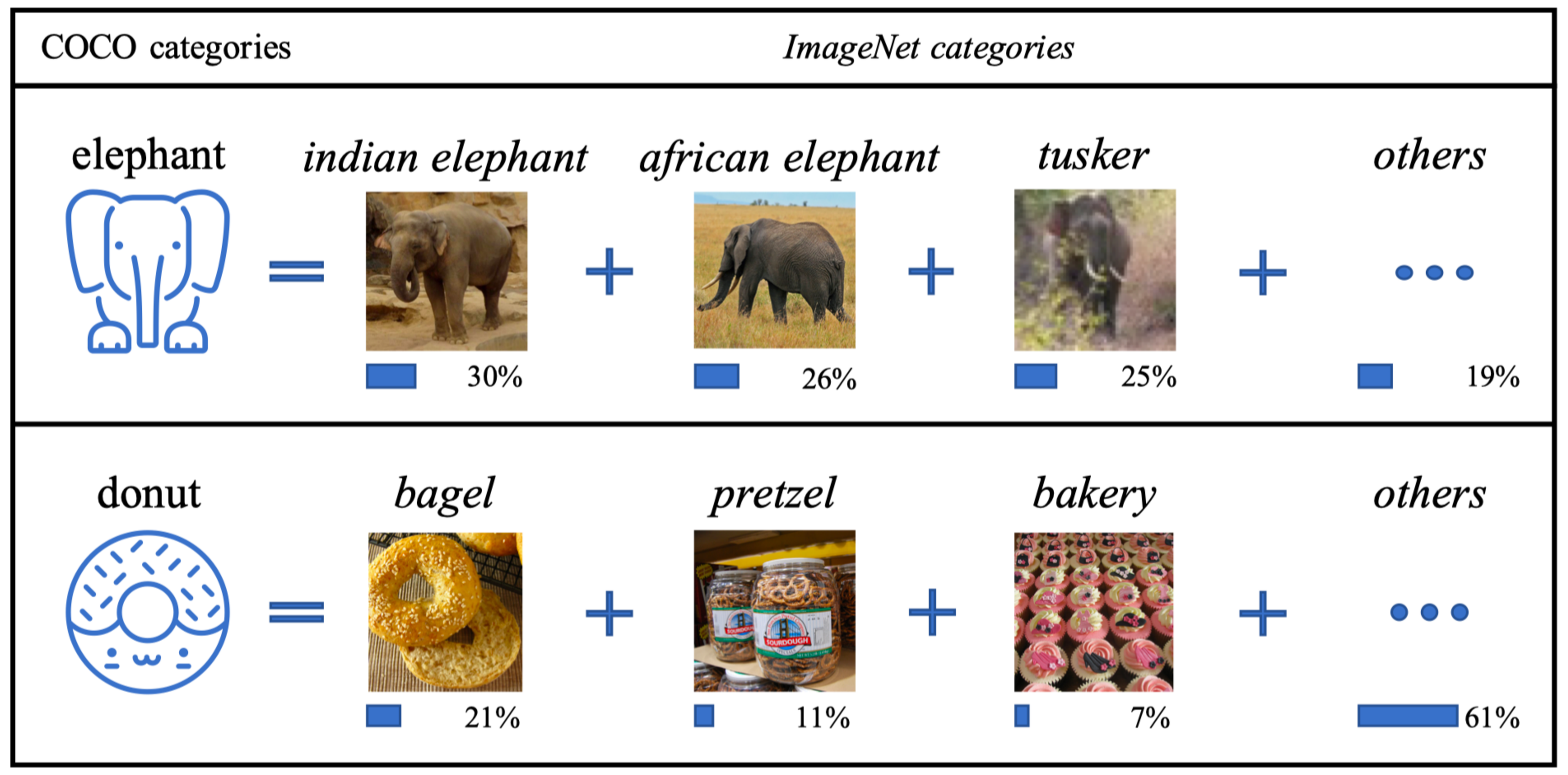
Parameter count in popular pre-trained models from torchvision and transformers.

Pre-trained model	ResNet-50	DenseNet-121	Inception-V3	BERT-base
Task-specific parameters / Million	2.0	1.0	2.0	22.9
Total parameters / Million	25.6	8.0	27.2	108.9
Percentage / %	7.8	12.5	7.4	**21.0**

- The challenge of reusing task-specific pre-trained Layer(s)
  - How to automatically map categories across datasets.

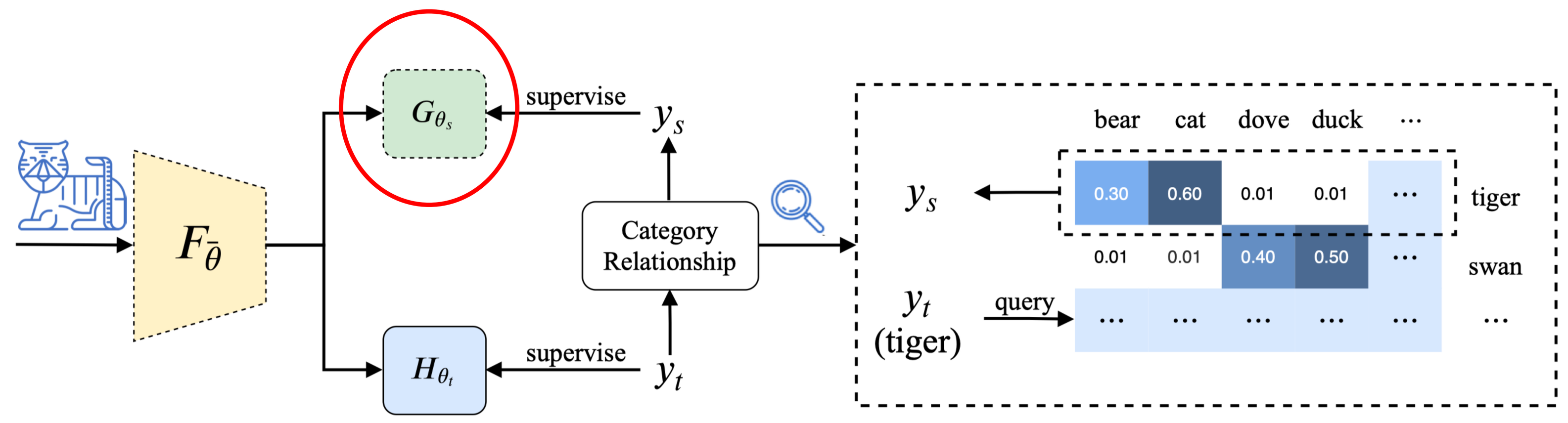


- ... can be solved if we can figure out the relationship of categories.
  - For example, when it learns a new category like "elephant", it can automatically learn that elephants can be represented by several kinds of elephants in ImageNet.



The Co-Tuning Framework

- Learn the category relationship $p(y_s|y_t)$.
- Pre-trained task-specific layers can be retained during training, supervised by source labels y_s translated from y_t.
- After training, task-specific layers will be removed so that Co-Tuning improves fine-tuning without additional inference cost.



How to Learn $p(y_s|y_t)$

- The direct approach: average over source predictions for each target category.

$$p(y_s|y_t = y) \approx |\mathcal{D}_t^y|^{-1}\Sigma_{(x,y_t)\in\mathcal{D}_t^y} f_0(x), \quad \mathcal{D}_t^y = \{(x, y_t) \in \mathcal{D}_t | y_t = y\}$$

- The reverse approach: (1) learn $y_s \to y_t$ mapping (maps probabilistic source predictions to target labels); (2) recover $p(y_s|y_t)$ by Bayes's rule.

Algorithm 1 Category relationship learning (the reverse approach)

Input: f_0, source validation data $\mathcal{D}_s^v = \{(x_s^i, y_s^i)\}_{i=1}^{m_v}$, target training data $\mathcal{D}_t = \{(x_t^i, y_t^i)\}_{i=1}^{m_t}$
Output: Category relationship $p(y_s|y_t)$
Call Alg. 2 to calibrate f_0 with $\mathcal{D}_s^v$, which returns the calibrated deep model $\tilde{f}_0$
Construct $\tilde{\mathcal{D}}_t = \{(\tilde{f}_0(x_t^i), y_t^i)\}_{i=1}^{m_t}$, further split it into training set $\tilde{\mathcal{D}}_t^{train}$ and validation set $\tilde{\mathcal{D}}_t^v$
Learn a neural network g from $\tilde{\mathcal{D}}_t^{train}$ to map calibrated source predictions to target labels
Call Alg. 2 to calibrate g with $\tilde{\mathcal{D}}_t^v$, which returns $p(y_t|y_s) \approx \tilde{g}(y_s)$
Compute marginal probability $p(y_s)$ and $p(y_t)$ from $\tilde{\mathcal{D}}_t$
Compute $p(y_s|y_t)$ by Bayes's rule: $p(y_s = i|y_t = j) = \frac{p(y_s=i)}{p(y_t=j)} p(y_t = j|y_s = i)$
Return $p(y_s|y_t)$

- In practice, the direct approach is simple and straightforward, while the reverse one is more effective.

(Optional) Calibration of Pre-trained Networks

- We want $f_0(x)$ to reflect the probability of source categories with high fidelity.
- Without calibration, DNNs can be over-confident.
- Calibration can be done by minimizing negative log-likelihood (NLL) on validation data through adjusting a single temperature.

Algorithm 2 Neural network calibration

Input: DNN f that outputs uncalibrated logits, validation data $\mathcal{D} = \{x^i, y^i\}_{i=1}^{m}$
Output: A neural network $\tilde{f}$ that outputs **calibrated** logits
Compute the scaling parameter $t^* = \arg\min_{t>0} \sum_{i=1}^{m} \texttt{cross\_entropy}(\texttt{softmax}(f(x^i)/t), y^i)$
Return $\tilde{f}$, where $\tilde{f}(x) = f(x)/t^*$

- We advocate that pre-trained model providers release pre-trained models and their calibrated version.

Experimental Results

Co-Tuning is empirically evaluated in several dimensions:

- Task: 4 visual classification tasks and one NLP task (named entity recognition).
- Dataset scale: medium-scale dataset ($\approx$ 100 samples per class) and large-scale dataset ($\approx$ 1000 samples per class). We also explore different sampling rates (the proportion of images used for training) to compare the performance among a wide spectrum of dataset scales.
- Pre-trained model: ResNet-50, DenseNet-121 and BERT-base.

Code is available at https://github.com/thuml/CoTuning
Main experimental results are shown in the following tables.

Table 2: Classification accuracy in medium-scale classification datasets (Pre-trained ResNet-50).

Dataset	Method	Sampling Rates			
		15%	30%	50%	100%
CUB-200-2011	Fine-tune (baseline)	45.25 ± 0.12	59.68 ± 0.21	70.12 ± 0.29	78.01 ± 0.16
	L^2-SP (Li et al., 2018)	45.08 ± 0.19	57.78 ± 0.24	69.47 ± 0.29	78.44 ± 0.17
	DELTA (Li et al., 2019)	46.83 ± 0.21	60.37 ± 0.25	71.38 ± 0.20	78.63 ± 0.18
	BSS (Chen et al., 2019)	47.74 ± 0.23	63.38 ± 0.29	72.56 ± 0.17	78.85 ± 0.31
	Co-Tuning	$\mathbf{52.58} \pm 0.53$	$\mathbf{66.47} \pm 0.17$	$\mathbf{74.64} \pm 0.36$	$\mathbf{81.24} \pm 0.14$
Stanford Cars	Fine-tune (baseline)	36.77 ± 0.12	60.63 ± 0.18	75.10 ± 0.21	87.20 ± 0.19
	L^2-SP (Li et al., 2018)	36.10 ± 0.30	60.30 ± 0.28	75.48 ± 0.22	86.58 ± 0.26
	DELTA (Li et al., 2019)	39.37 ± 0.34	63.28 ± 0.27	76.53 ± 0.24	86.32 ± 0.20
	BSS (Chen et al., 2019)	40.57 ± 0.12	64.13 ± 0.18	76.78 ± 0.21	87.63 ± 0.27
	Co-Tuning	$\mathbf{46.02} \pm 0.18$	$\mathbf{69.09} \pm 0.10$	$\mathbf{80.66} \pm 0.25$	$\mathbf{89.53} \pm 0.09$
FGVC Aircraft	Fine-tune (baseline)	39.57 ± 0.20	57.46 ± 0.12	67.93 ± 0.28	81.13 ± 0.21
	L^2-SP (Li et al., 2018)	39.27 ± 0.24	57.12 ± 0.27	67.46 ± 0.26	80.98 ± 0.29
	DELTA (Li et al., 2019)	42.16 ± 0.21	58.60 ± 0.29	68.51 ± 0.25	80.44 ± 0.20
	BSS (Chen et al., 2019)	40.41 ± 0.12	59.23 ± 0.31	69.19 ± 0.13	81.48 ± 0.18
	Co-Tuning	$\mathbf{44.09} \pm 0.67$	$\mathbf{61.65} \pm 0.32$	$\mathbf{72.73} \pm 0.08$	$\mathbf{83.87} \pm 0.09$

Table 3: Classification accuracy in large-scale COCO-70 dataset (Pre-trained DenseNet-121).

Method	Sampling Rates			
	15%	30%	50%	100%
Fine-tune (baseline)	76.60 ± 0.04	80.15 ± 0.25	82.50 ± 0.43	84.41 ± 0.22
L^2-SP (Li et al., 2018)	77.53 ± 0.47	80.67 ± 0.29	83.07 ± 0.39	84.78 ± 0.16
DELTA (Li et al., 2019)	76.94 ± 0.37	79.72 ± 0.24	82.00 ± 0.52	84.66 ± 0.08
BSS (Chen et al., 2019)	77.39 ± 0.15	80.74 ± 0.22	82.75 ± 0.59	84.71 ± 0.13
Co-Tuning	$\mathbf{77.64} \pm 0.23$	$\mathbf{81.19} \pm 0.18$	83.43 ± 0.22	$\mathbf{85.65} \pm 0.11$

Case Study in CUB

- Take two similar bird species "Crested Auklet" and "Parakeet Auklet".
- Top 3 similar ImageNet classes are in the below table to roughly represent their source distributions $p(y_s|y_t)$.

CUB Class	Top 3 Similar ImageNet Class		
Crested Auklet	black swan	oystercatcher	black grouse
Parakeet Auklet	black grouse	oystercatcher	junco

- Their distributions are similar (both have "black grouse" and "oystercatcher") but still differ (one has "black swan" while the other has "junco").
- Co-Tuning works by finding meaningful category relationship $p(y_s|y_t)$ as long as the pre-trained dataset is diverse enough.

School of Software - Tsinghua University - China
BNRist, Research Center for Big Data
Mail: youkaichao@gmail.com