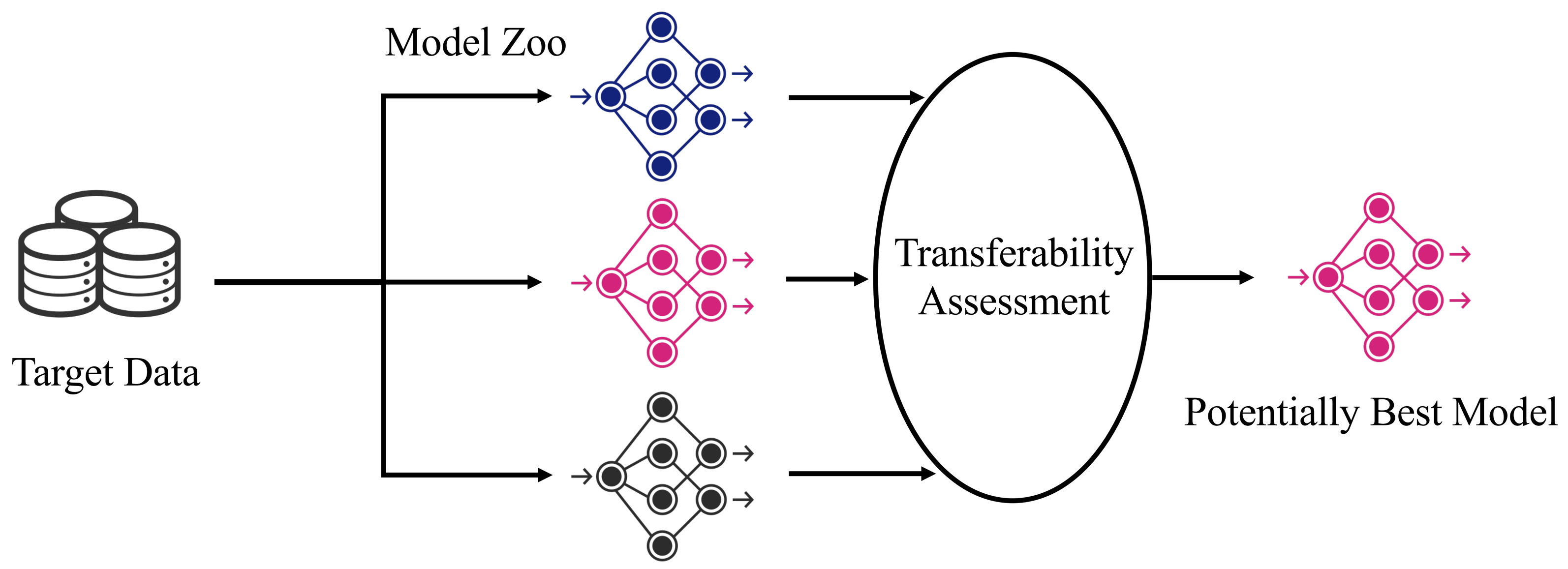
LogME: Practical Assessment of Pre-trained Models for Transfer Learning

Kaichao You, Yong Liu, Jianmin Wang, Mingsheng Long (School of Software, Tsinghua University)

ICML International Conference On Machine Learning

The Problem We Solve



How to select the best pre-trained model before fine-tuning!

Pre-trained Model Selection Problem

- Transfer learning is a widely-used paradigm for practitioners.
- There have been some papers working on improving transfer learning when a pre-trained model is given:
  - Co-Tuning for Transfer Learning, NeurIPS 2020
  - Stochastic Normalization, NeurIPS 2020
- But which pre-trained model should be used if we have a large model zoo?
  - TorchVision has over 100 pre-trained models.
  - HuggingFace has over 6000 pre-trained models.
- Design a general and fast algorithm for accurate selection!

Brief Overview of LogME

- generally applicable to a broad range of tasks

Modality	Pre-train	Target	LEEP	NCE	LogME
vision	classification	classification	✓	✓	✓
	classification	regression	✗	✗	✓
	contrastive	classification	✗	✗	✓
	contrastive	regression	✗	✗	✓
language	LM	classification	✗	✗	✓

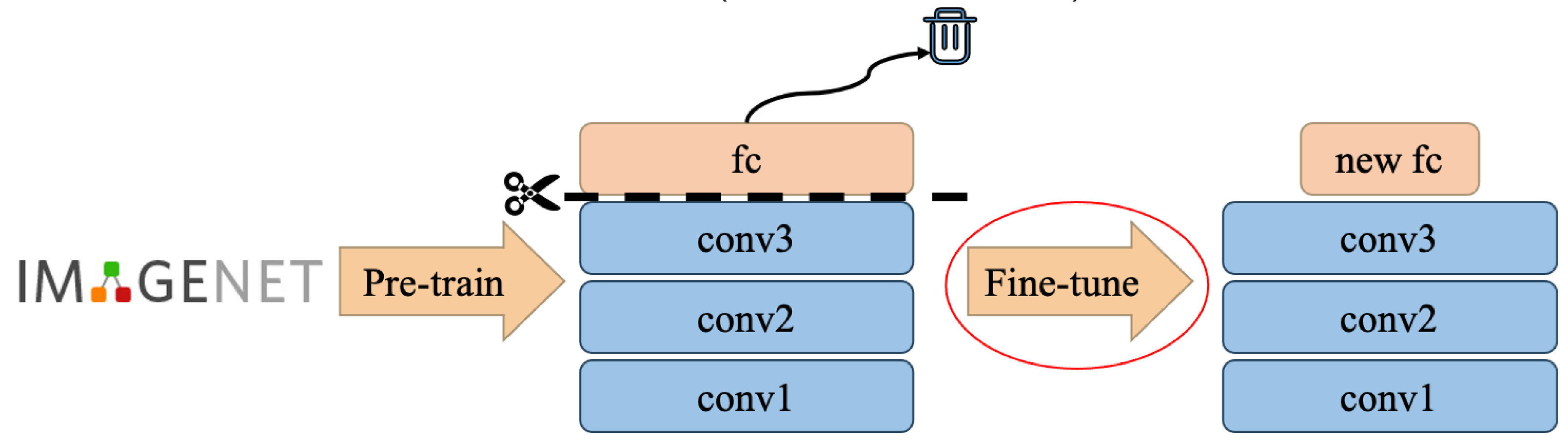
- fast and memory-efficient

	wall-clock time		memory footprint	
Computer Vision	fine-tune (upper bound)	161000s	fine-tune (upper bound)	6.3 GB
	extract feature (lower bound)	37s	extract feature (lower bound)	43 MB
	LogME	50s	LogME	53 MB
	benefit	3200 ↑	benefit	120 ↑
Natural Language Processing	fine-tune (upper bound)	100200s	fine-tune (upper bound)	88 GB
	extract feature (lower bound)	1130s	extract feature (lower bound)	1.2 GB
	LogME	1157s	LogME	1.2 GB
	benefit	86 ↑	benefit	73 ↑

- tested on 22 pre-trained models and 17 downstream tasks

Overall Idea

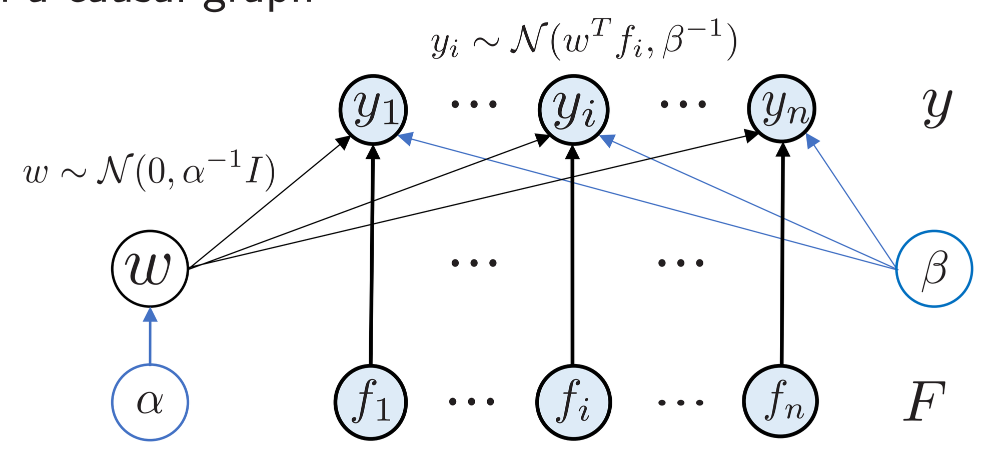
- Drawback of brute-force fine-tuning
  - hyper-parameter tuning, model training (both time-consuming)



- trade-off between speed and accurate selection
  - freeze the feature extractor to avoid gradient update
  - leverage theoretical optimization to avoid hyper-parameter tuning
- setup a model to estimate the compatibility between features and labels

LogME – unary output

- measure $p(y|f)$ with linear model $y = w^T f$
- A naive solution (point estimation) of training optimal w^* and computing $p(y|f, w^*)$ is prone to over-fitting
- A better solution (distributional estimation) is to take expectation over all possible w with a causal graph



  - $p(y|F) = \int p(w)p(y|F,w)\mathrm{d}w$
  - $\mathcal{L}(\alpha,\beta) = \log p(y|F,\alpha,\beta) = \frac{n}{2}\log\beta + \frac{D}{2}\log\alpha - \frac{n}{2}\log 2\pi - \frac{\beta}{2}||Fm-y||_2^2 - \frac{\alpha}{2}m^T m - \frac{1}{2}\log|A|$ with $A = \alpha I + \beta F^T F, m = \beta A^{-1} F^T y$
  - $\mathcal{L}(\alpha,\beta)$ measures how likely labels are with respect to features.
- How to choose α, β?
  - alternative optimization (no grid search!)

Algorithm 1 LogME

1: **Input:** Pre-trained model ϕ
  Target dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
2: **Output:** logarithm of maximum evidence (LogME)
3: Extract features using pre-trained model ϕ:
  $F \in \mathbb{R}^{n\times D}, f_i = \phi(x_i), Y \in \mathbb{R}^{n\times K}$
4: Compute SVD $F^T F = V\mathrm{diag}\{\sigma\}V^T$
5: **for** $k = 1$ to K **do**
6:   Let $y = Y^{(k)} \in \mathbb{R}^n$, initialize $\alpha = 1, \beta = 1$
7:   **while** α, β not converge **do**
8:     Compute $\gamma = \sum_{i=1}^{D} \frac{\beta\sigma_i}{\alpha+\beta\sigma_i}, \Lambda = \mathrm{diag}\{(\alpha+\beta\sigma)\}$
9:     **Naïve:** $A = \alpha I + \beta F^T F, m = \beta A^{-1} F^T y$
10:     **Optimized:** $m = \beta(V(\Lambda^{-1}(V^T(F^T y))))$
11:     Update $\alpha \leftarrow \frac{\gamma}{m^T m}, \beta \leftarrow \frac{n-\gamma}{||Fm-y||_2^2}$
12:   **end while**
13:   Compute $\mathcal{L}_k = \frac{1}{n}\mathcal{L}(\alpha,\beta)$ using Eq. 2
14: **end for**
15: Return LogME $\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k$

- complexity $\mathcal{O}(KD^3 + nKD^2)$
- for common cases

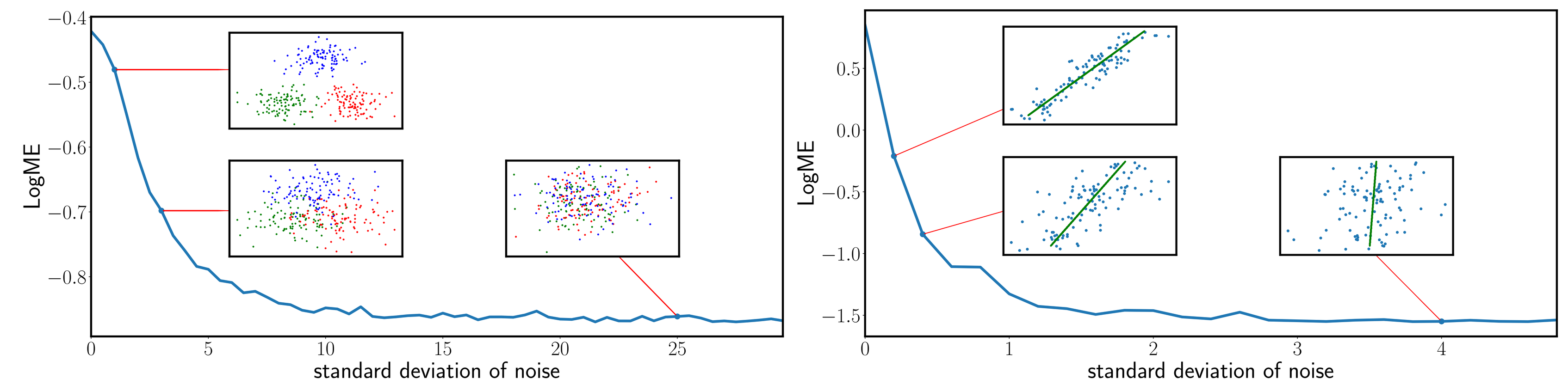
  $D \approx 10^3, n \approx 10^4, K \approx 10^3$

  10^{13} operations needs 10^4 seconds
  not fast enough ☹
- bottleneck
  matrix inverse and MatMul (line 9)
- Optimization (line 10):
  - leverage results from line 4
  - avoid matrix inverse
  - MatMul → MatVecMul
  - reduce from $O(n^4)$ to $O(n^3)$

	Complexity per for-loop	Overall complexity
naïve	$\mathcal{O}(D^3 + nD^2)$	$\mathcal{O}(KD^3 + nKD^2)$
optimized	$\mathcal{O}(D^2 + nD)$	$\mathcal{O}(KD^2 + nKD + D^3 + nD^2)$

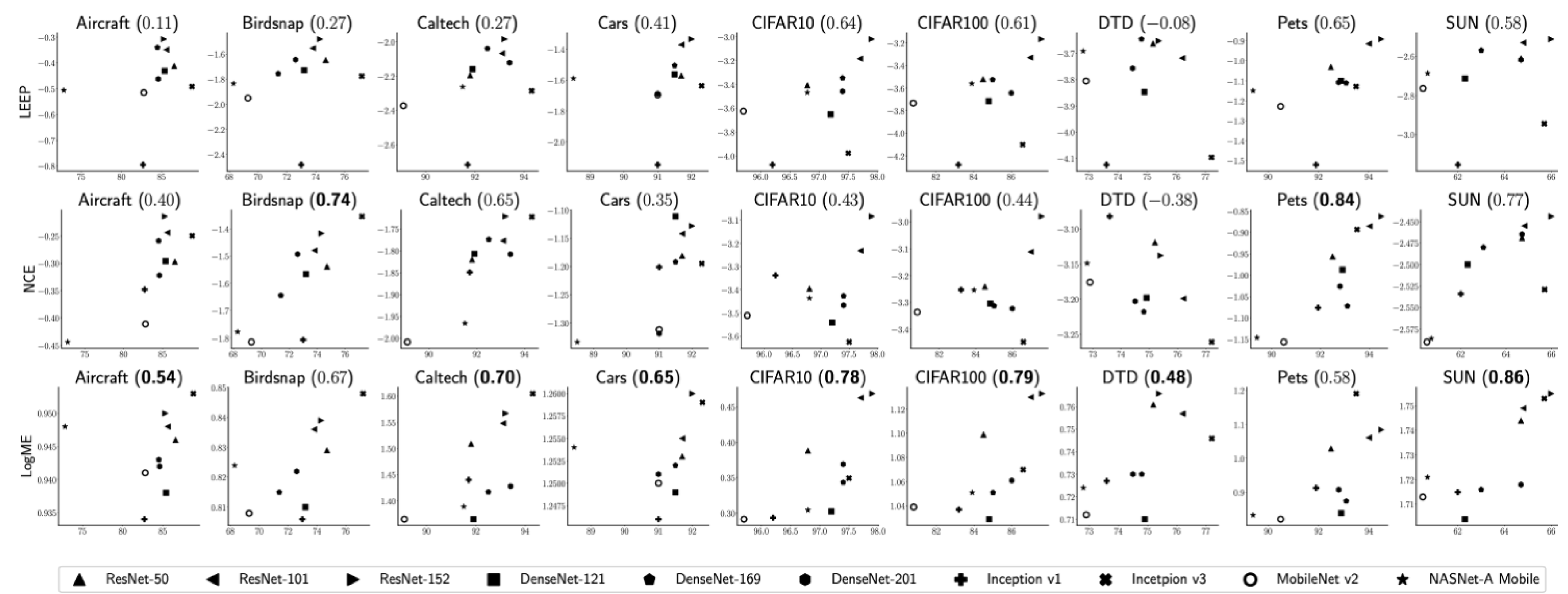
Experimental Results (toy data for intuitive explanation)

- Generated data with increasing noise (decreasing feature quality).
- LogME decreases as feature quality decreases.
- LogME can properly measure the quality of features!



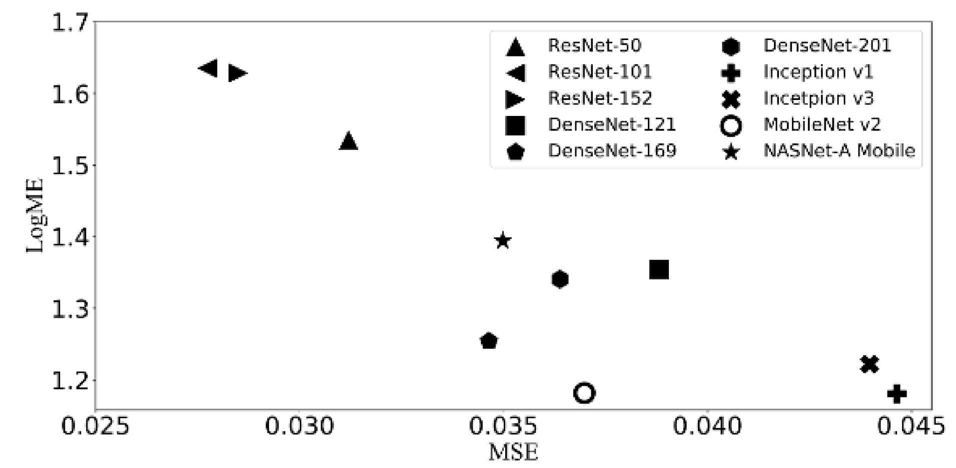
Experimental Results (compared with prior methods)

- 9 datasets, 10 pre-trained models; x-axis (accuray) vs. y-axis (assessment score)
- LogME has largest τ_w in most tasks



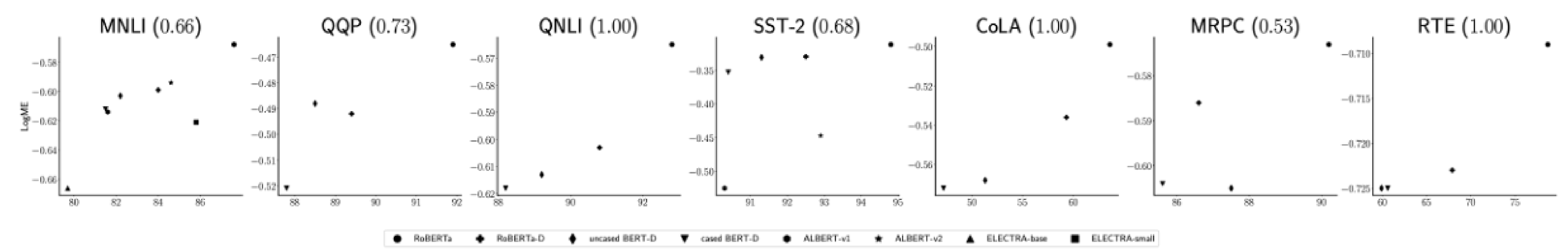
Experimental Results (LogME exclusive)

- regression tasks (larger LogME indicates better performance)
- contrastive pre-trained models (perfectly predict the order after fine-tuning)



Pre-trained Network	Aircraft		dSprites	
	Accuracy (%)	LogME	MSE	LogME
MoCo V1	81.68	0.934	0.069	1.52
MoCo V2	84.16	0.941	0.047	1.64
MoCo 800	86.99	0.946	0.050	1.58
SimCLR	88.10	0.950	-	-
	τ_w: 1.0		τ_w: 1.0	

- NLP tasks (7 GLUE tasks with 8 popular pre-trained language models)



Useful Links

- Code is available at https://github.com/thuml/LogME
  - Will be integrated into https://github.com/thuml/Transfer-Learning-Library in the future.

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