

□

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

Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long

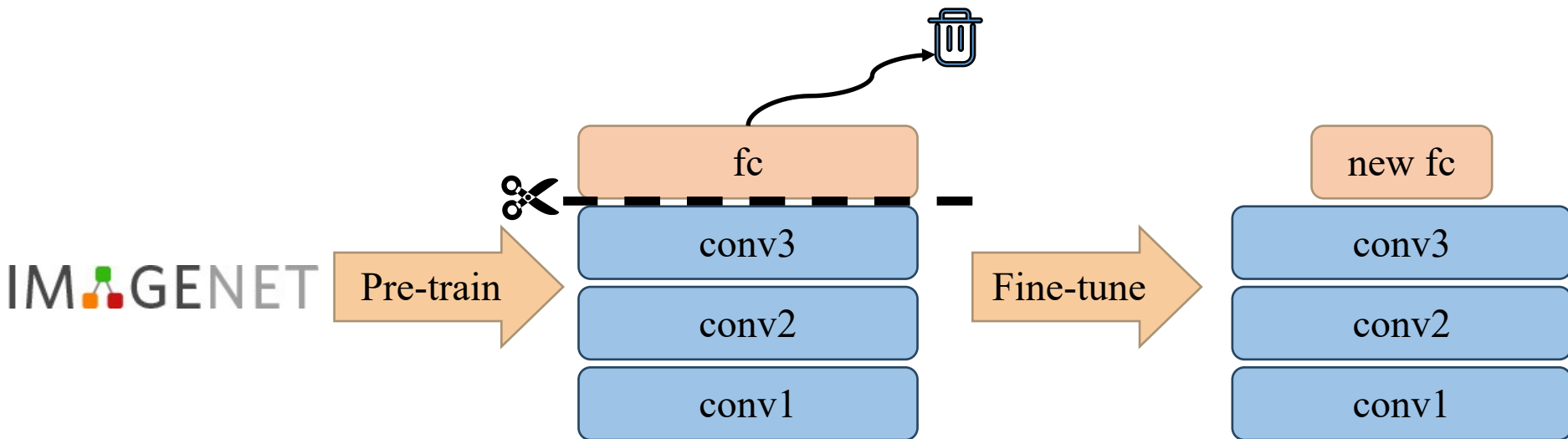
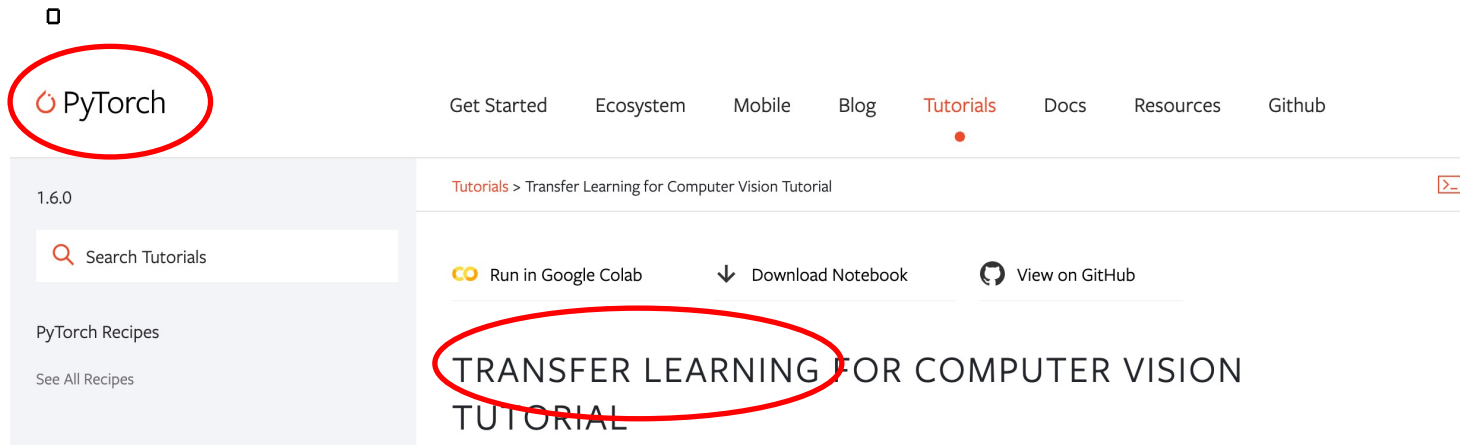
THUML Group, School of Software, Tsinghua University



Wechat Blogpost (Chinese)



# Transfer Learning Paradigm



- Improve transfer learning with a given pre-trained model
  - Co-Tuning for Transfer Learning, NeurIPS 2020
  - Stochastic Normalization, NeurIPS 2020



# Pre-trained Model Selection


## • Which pre-trained model to use?

All Audio Generative Nlp Scriptable Vision

Sort ▾


MiDaS 690

The MiDaS v2.1 model for computing relative depth from a single image.




ntsnet 10

classify birds using this fine-grained image classifier




Silero Speech-To-Text ... 512

A set of compact enterprise-grade pre-trained STT Models for multiple languages.




Silero Language Classi... 97

Pre-trained Spoken Language Classifier




Silero Number Detector 97

Pre-trained Spoken Number Detector



Silero Voice Activity ... 97

Pre-trained Voice Activity Detector



All Research Models (37) >

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNetV2
- MobileNetV3
- ResNeXt
- Wide ResNet
- MNASNet

- <https://pytorch.org/hub/>
- <https://pytorch.org/vision/stable/models.html>



# Pre-trained Model Selection

□

- Which pre-trained model to use? (6300 models)

Models 6300  Sort: Most Downloads

<b>distilbert-base-uncased</b> Fill-Mask • Updated Dec 11, 2020 • 14,306k	<b>bert-base-uncased</b> Fill-Mask • Updated Dec 11, 2020 • 14,266k
<b>cl-tohoku/bert-base-japanese-whole-word-masking</b> Fill-Mask • Updated Jan 25 • 4,013k	<b>jplu/tf-xlm-roberta-base</b> Fill-Mask • Updated Dec 11, 2020 • 3,236k
<b>xlm-roberta-base</b> Fill-Mask • Updated Dec 11, 2020 • 2,377k	<b>bert-large-uncased</b> Fill-Mask • Updated Jan 13 • 2,196k
<b>bert-base-cased</b> Fill-Mask • Updated Dec 15, 2020 • 1,998k	<b>bert-large-cased</b> Fill-Mask • Updated Jan 13 • 1,791k
<b>gpt2</b> Text Generation • Updated Dec 11, 2020 • 997k	<b>distilbert-base-uncased-finetuned-sst-2-english</b> Text Classification • Updated Feb 9 • 860k
<b>roberta-large</b> Fill-Mask • Updated Dec 11, 2020 • 854k	<b>valhalla/t5-small-qa-qg-hl</b> Text2Text Generation • Updated Dec 11, 2020 • 778k
<b>roberta-base</b> Fill-Mask • Updated Dec 11, 2020 • 772k	<b>facebook/bart-large-mnli</b> Zero-Shot Classification • Updated Dec 11, 2020 • 704k
<b>roberta-large-mnli</b> Text Classification • Updated Dec 11, 2020 • 634k	<b>t5-base</b> Translation • Updated Dec 11, 2020 • 618k
<b>sentence-transformers/distilbert-base-nli-stsb-m...</b> Updated Aug 31, 2020 • 597k	<b>microsoft/BiomedNLP-PubMedBERT-base-uncased-abst...</b> Updated Aug 8, 2020 • 566k

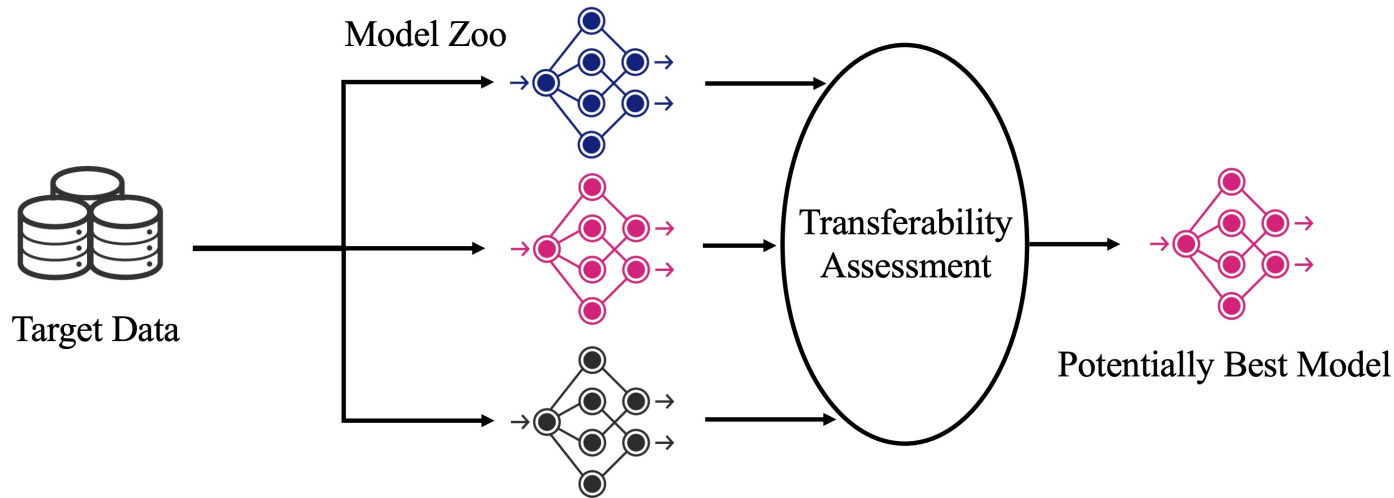
- <https://huggingface.co/models>



# Pre-trained Model Selection

□

## • Procedure of Pre-trained Model Selection



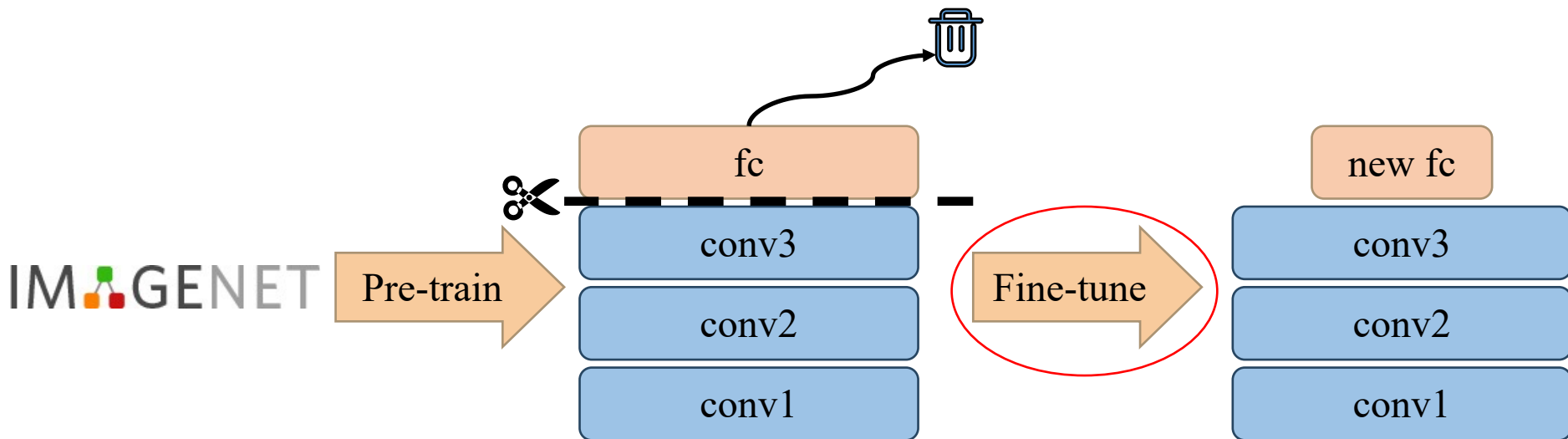
## • How to measure correlation

- M pre-trained models  $\{\phi_m\}_{m=1}^M$  with a dataset  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
- Ground truth transfer learning performance  $\{T_m\}_{m=1}^M$
- Assessment score  $\{S_m\}_{m=1}^M$
- weighted Kendall's Tau between  $\{T_m\}_{m=1}^M$  and  $\{S_m\}_{m=1}^M$



# Brute-force Fine-tuning

□

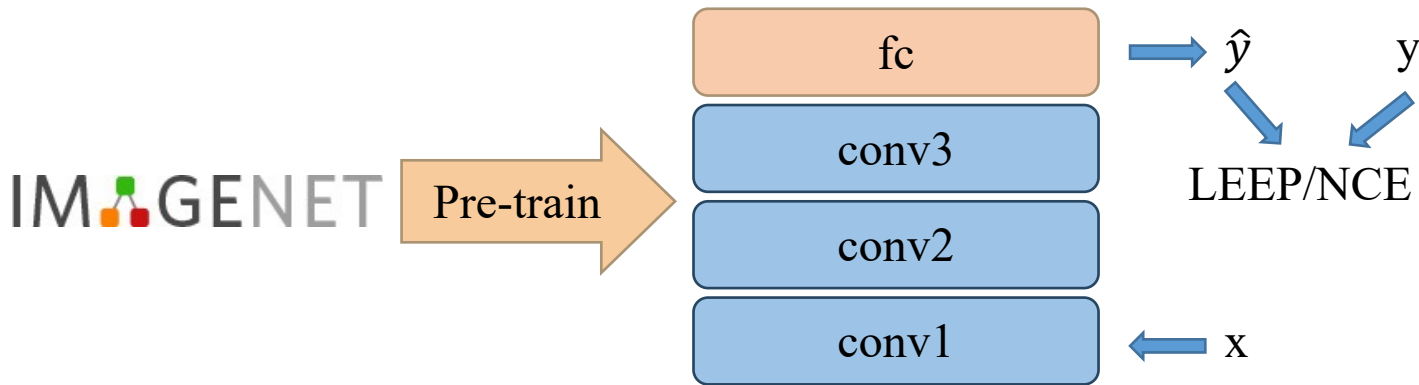


- complete fine-tuning of pre-trained models
  - hyper-parameter tuning, model training (☹️ costly)
  - yield ground-truth measure (☺️  $\tau_w = 1$ )



# LEEP/NCE

□

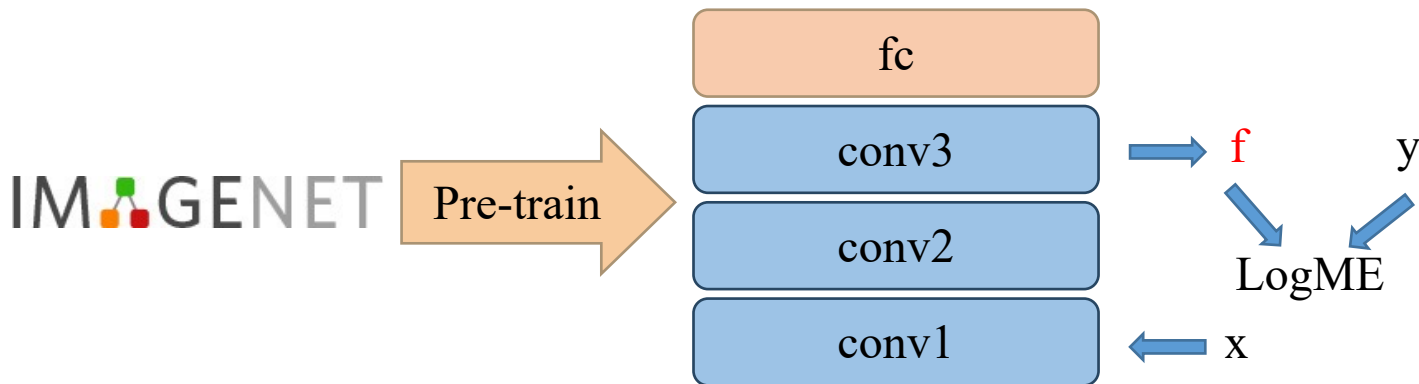


- no training (☺ fast )
- build on category relationship
  - not accurate (☹ small  $\tau_w$ )
  - limited applicability ☹
  - can only transfer supervised pre-trained models to classification



# LogME (proposed)

□



- fix pre-trained models  $\Rightarrow$  fast 😊
- treat pre-trained models as feature extractor
  - applicable to any pre-trained models 😊
- How to measure the compatibility between f and y?



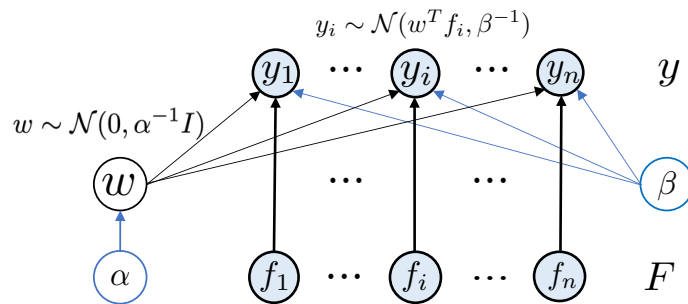


# LogME – unary output

□

- measure by  $p(y|f)$  with linear model  $y=w^Tf$
- point estimation ☹
  - train optimal  $w^*$ , compute  $p(y|f, w^*)$
  - prone to over-fitting
- distributional estimation (evidence) ☺
  - take expectation over all possible  $w$  with a causal graph

$$p(y|F) = \int p(w)p(y|F, w)dw$$





# LogME – unary output

□

- Analytic form (  $A = \alpha I + \beta F^T F, m = \beta A^{-1} F^T y.$  )

$$\begin{aligned}\mathcal{L}(\alpha, \beta) &= \log p(y|F, \alpha, \beta) \\ &= \frac{n}{2} \log \beta + \frac{D}{2} \log \alpha - \frac{n}{2} \log 2\pi \\ &\quad - \frac{\beta}{2} \|Fm - y\|_2^2 - \frac{\alpha}{2} m^T m - \frac{1}{2} \log |A|\end{aligned}$$

- measures how likely labels are with respect to features.
- How to choose  $\alpha, \beta$  ?

- no grid search !
- maximize  $\mathcal{L}(\alpha, \beta)$  via alternative optimization

$$A = \alpha I + \beta F^T F, m = \beta A^{-1} F^T y, \gamma = \sum_{i=1}^D \frac{\beta \sigma_i}{\alpha + \beta \sigma_i}$$
$$\alpha \leftarrow \frac{\gamma}{m^T m}, \beta \leftarrow \frac{n - \gamma}{\|Fm - y\|_2^2}$$

- name converged value LogME (log maximum evidence)

$$(\alpha^*, \beta^*) = \arg \max_{\alpha, \beta} \mathcal{L}(\alpha, \beta)$$



# LogME – complex cases

□

- Multivariate output
  - average LogME over each dimension
- classification with  $K$  classes
  - no analytic form with softmax output
  - regress one-hot labels instead



# Complexity Analysis

□

## Algorithm 1 LogME

1: **Input:** Pre-trained model  $\phi$   
     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  $\phi$ :  
      $F \in \mathbb{R}^{n \times D}$ ,  $f_i = \phi(x_i)$ ,  $Y \in \mathbb{R}^{n \times K}$

4: Compute SVD  $F^T F = V \text{diag}\{\sigma\} V^T$

5: **for**  $k = 1$  to  $K$  **do**

6:   Let  $y = Y^{(k)} \in \mathbb{R}^n$ , initialize  $\alpha = 1, \beta = 1$

7:   **while**  $\alpha, \beta$  not converge **do**

8:     Compute  $\gamma = \sum_{i=1}^D \frac{\beta \sigma_i}{\alpha + \beta \sigma_i}$ ,  $\Lambda = \text{diag}\{(\alpha + \beta \sigma)\}$

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**

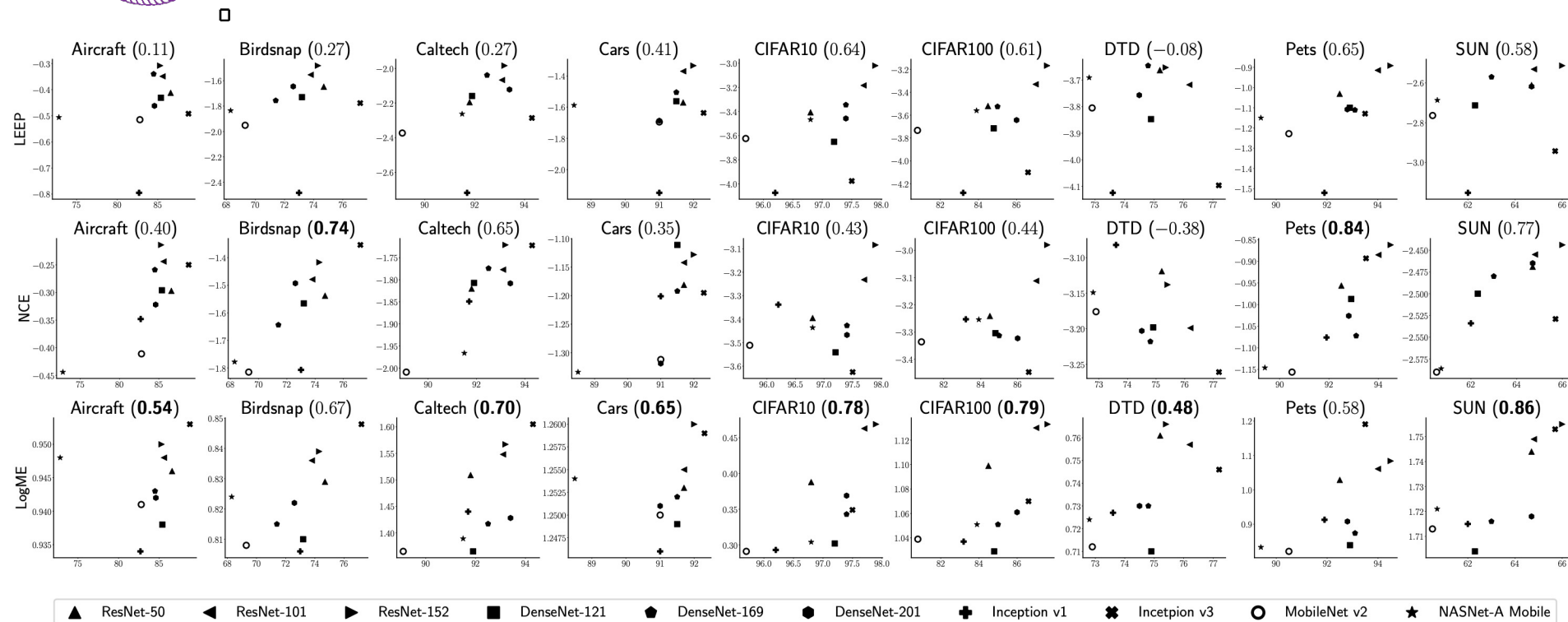
15: **Return** LogME  $\frac{1}{K} \sum_{k=1}^K \mathcal{L}_k$

- 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  $\rightarrow$  MatVecMul
  - reduce from  $\mathcal{O}(n^4)$  to  $\mathcal{O}(n^3)$

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



# Classification with Supervised Pre-trained Models

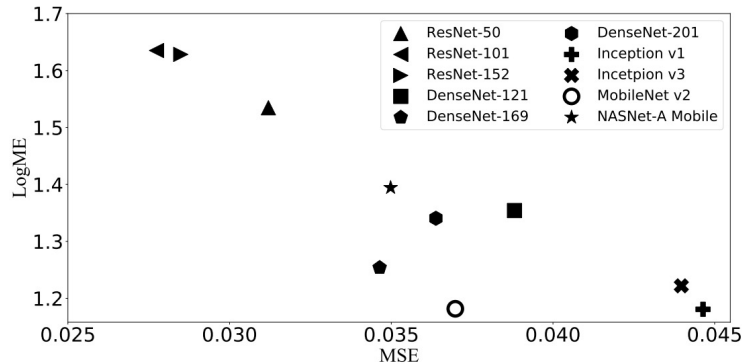


- 9 datasets, 10 pre-trained models
- x-axis (accuracy) vs. y-axis (assessment score)
- LogME has largest  $\tau_w$  in most tasks



# More experiments

- Regression

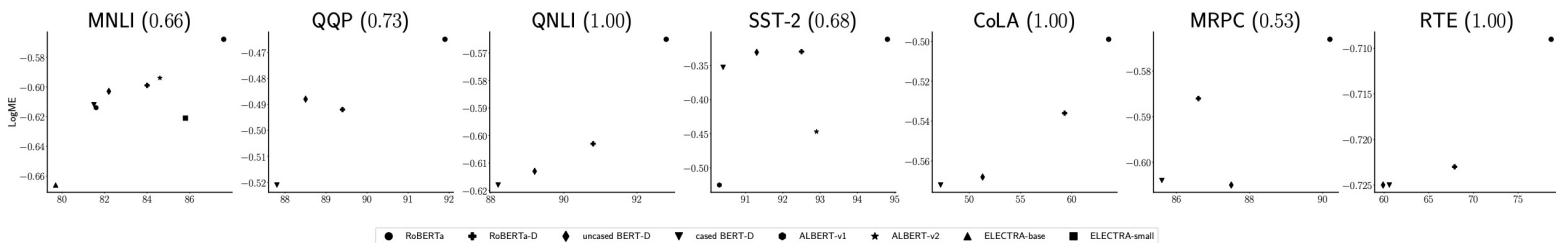


- Contrastive pre-trained models

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	-	-

$\tau_w: 1.0$                        $\tau_w: 1.0$

- NLP models



- Only LogME works, LEEP / NCE are not applicable



# Efficiency of LogME

	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 ↑

- Much more efficient than brute-force fine-tuning
  - at most 3000x speedup with only 1% memory
  - almost lower bound

# Thanks for Listening!

---

Code Available: <https://github.com/thuml/LogME>