Universidade de São Paulo
Escola Politécnica - Engenharia de Computação e Sistemas Digitais

Transfer Learning and Knowledge Distillation

Prof. Artur Jordão

Transfer Learning

Introduction

Transfer Learning

- State-of-the-art deep learning models
  - Computational resources
  - Large models
  - Data hungry
  - Hyperparameters search

- Fortunately, the **internal representation** (pattern) learned for one particular task can also be useful for related tasks
  - Therefore, we can transfer the knowledge learned from the source to the target dataset

- The model can exploit commonalities shared by both source and target domains
  - For example, many image tasks require similar low-level features corresponding to the early layers of a deep neural network

Definitions

Transfer Learning

- Suppose two tasks A (source) and B (target)
  - Data for task $\boldsymbol{A}$ **is plentiful**
  - Data for task **B is very scarce**

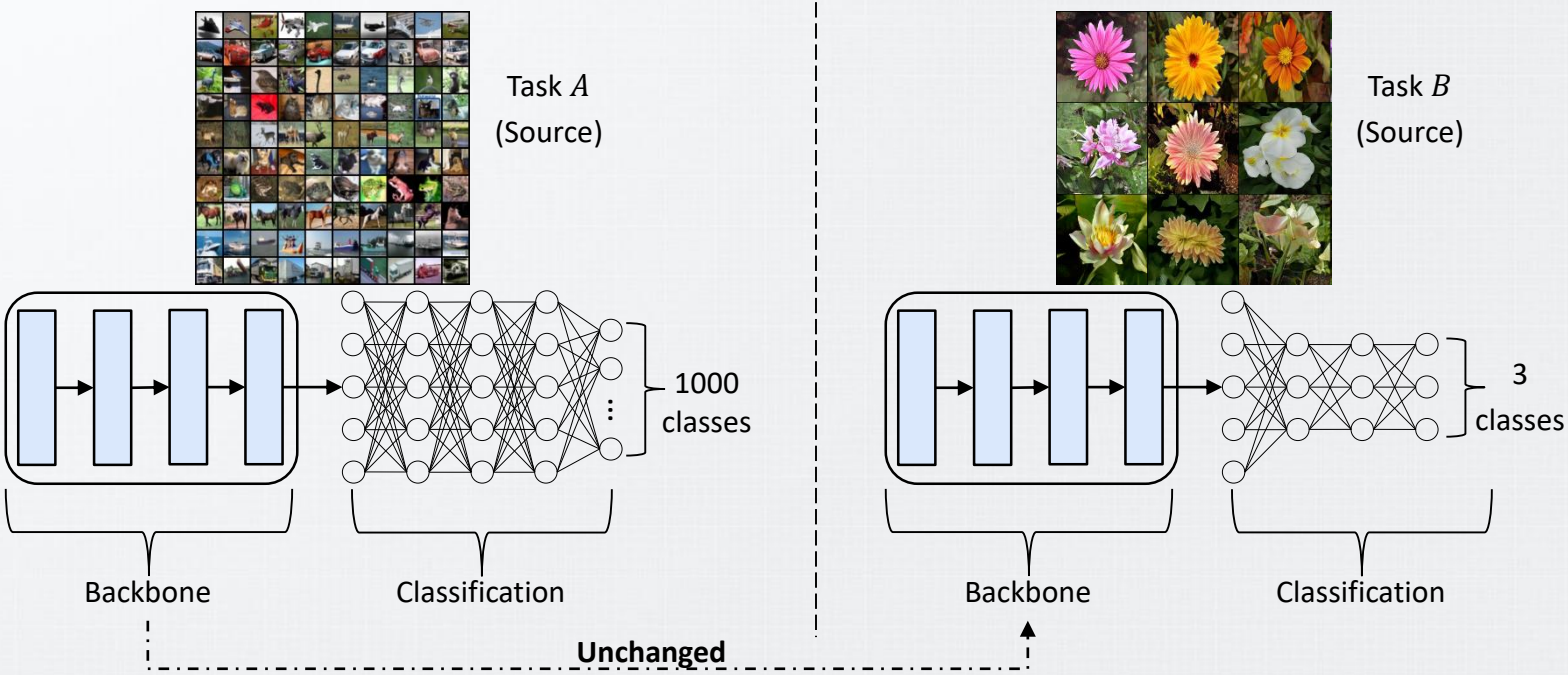
- Assume a model $\mathcal{F}(\cdot, \theta_A)$ trained on A for days using high-performance resources
  - θ_A means the parameters learned using A

- Instead of training a novel model on B from scratch, we start the *training* from θ_A
  - **Transfer the learning from $\boldsymbol{A}$ to $\boldsymbol{B}$**
  - Adapt the parameters θ_A to the new domain B

Architectural Changes

Transfer Learning

- Often, to perform transfer learning, we need to **remove** the last layers (classification) and **add** layers that produce the output of the target task
  - The parameters from the new classification layer are randomly initialized



Fine-Tuning

Transfer Learning

- After performing the architectural changes, we need to adapt the parameters to the new task
- In transfer learning, the training phase is known as **Fine-tuning**
- Fine-tuning lies at the heart of transfer learning
- Recipes for fine-tuning
  - Employ small learning rates
  - (i) Freeze the backbone, (ii) train the classification layer for a few epochs; (iii) then, train the whole network
  - Freeze some layers (primarily the first layers)

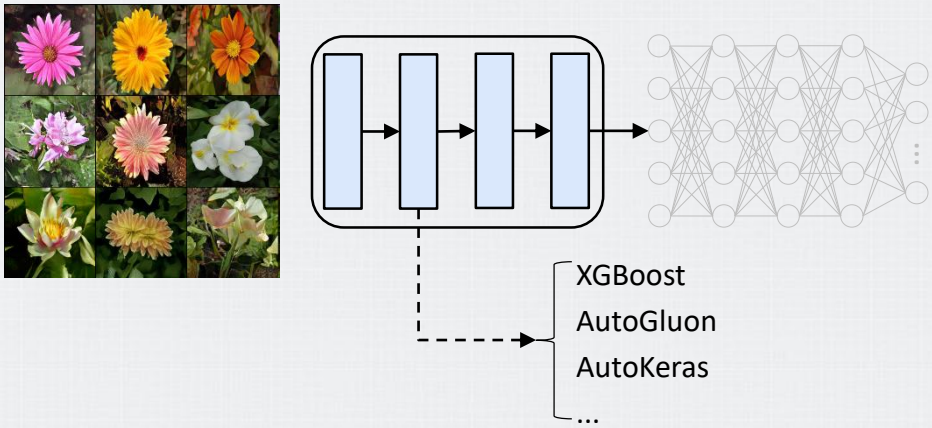
Feature Extraction

Transfer Learning

- A pattern learned in one domain can be sufficient to solve problems in another domain
  - A pattern is also known as **representation** (or internal representation) of the network

- Instead of rebuilding and fine-tuning a model, we can extract its features and feed them to train simple classifiers

- Recipes for feature extraction
  - Select layers carefully
  - Reduce the dimensionality of the data

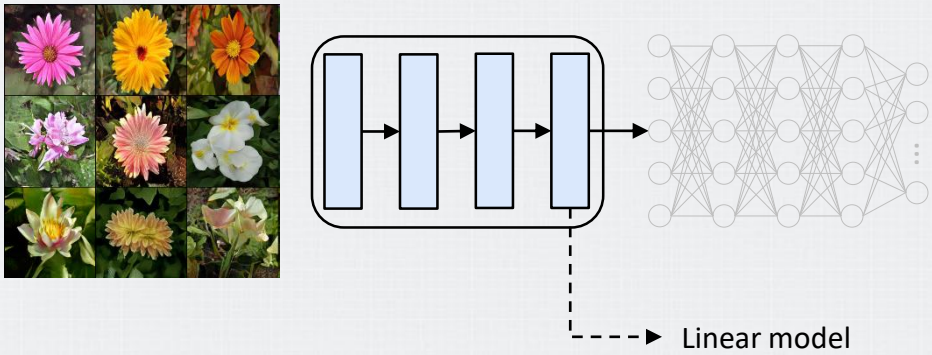


Linear Probing

Transfer Learning

- Linear probing involves **freezing the source model** and training a new classification head for the target domain
  - Often, we discard the classification head and train a new one from scratch

- Linear probing employs the backbone of the network to obtain features and then learn a **linear classifier** (Evci et al., 2022; Kirichenko et al. 2023; Chen et al. 2024)
  - Cost-efficient strategy



Evci et al. *Head2Toe: Utilizing Intermediate Representations for Better Transfer Learning*. International Conference on Machine Learning (ICML), 2022
Chen et al. *Project and Probe: Sample-Efficient Adaptation by Interpolating Orthogonal Features.* International Conference on Learning Representations (ICLR), 2024
Kirichenko et al. *Last Layer Re-training is Sufficient for Robustness to Spurious Correlations*. International Conference on Learning Representations (ICLR), 2023

Linear Probing

Transfer Learning

- Linear probing enables us to assess the representations of the (pretrained) model by training a linear classifier on top of the fixed features extracted from the model (Xu et al., 2024)

- If the domain overlap is high, then features extracted for linear classification in the source domain should also be relevant for linear classification in the target domain (Evci et al., 2022)

- Kirichenko et al. (2023) mitigated the spurious by simply retraining a linear layer
  - Remember that spurious correlations are patterns that are predictive of the target in the train data, but that are irrelevant to the true labeling function

Xu et al., *Initializing Models with Larger Ones*. International Conference on Learning Representations (ICLR), 2024

Evci et al. *Head2Toe: Utilizing Intermediate Representations for Better Transfer Learning*. International Conference on Machine Learning (ICML), 2022

Kirichenko et al. *Last Layer Re-training is Sufficient for Robustness to Spurious Correlations*. International Conference on Learning Representations (ICLR), 2023

Parameter-Efficient Fine-Tuning (PEFT)

Transfer Learning

- Parameter-efficient fine-tuning (PEFT) techniques allow for the efficient adaptation of large models to different downstream applications without the need to fine-tune all of the model's parameters
  - Often pre-trained language models

- PEFT techniques reduce the memory consumption of model fine-tuning via updating a small number of parameters

- The PEFT zoo
  - Low-Rank Adaptation (LoRA) (Hu et al., 2022)
  - Gradient Low-rank Projection (GaLore) (Zhao et al., 2024)
  - ReLoRA (Lialin et al., 2024)

Hu et al. *LoRA: Low-rank Adaptation of Large Language Models*. International Conference on Learning Representations (ICLR), 2022

Zhao et al. *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*. International Conference on Machine Learning (ICML), 2024 (Oral)

Lialin et al. *ReloRA: High-rank training through low-rank updates*. International Conference on Machine Learning (ICML), 2024

Low-Rank Adaptation (LoRA)

Transfer Learning

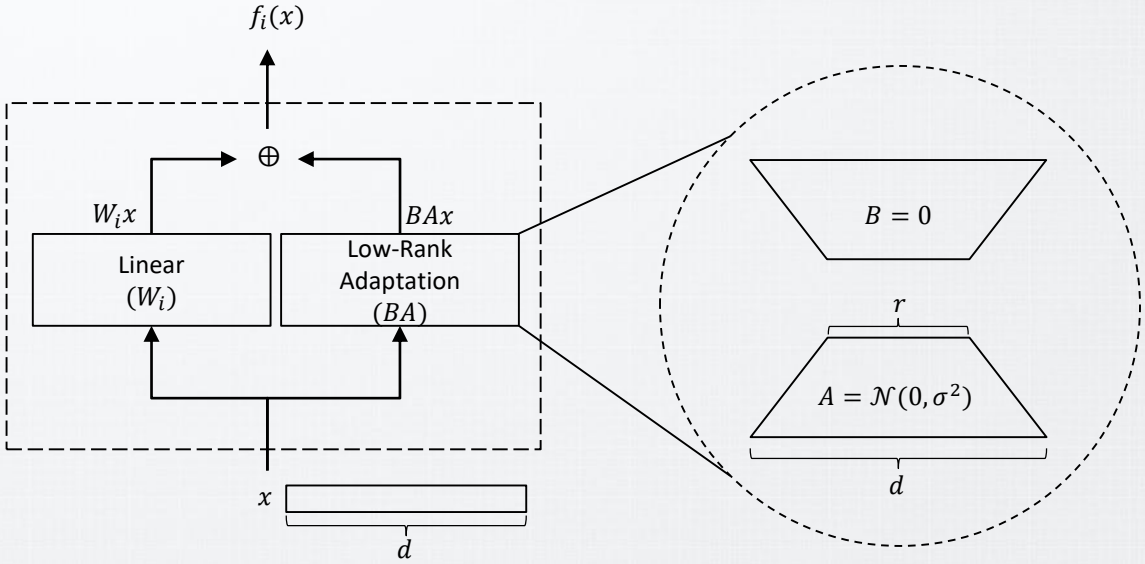
- Let $W \in \mathbb{R}^{m \times n}$ be the parameters within a given layer in a pre-trained model
- Let ΔW be a residual adjustment
  - LoRA accomplishes ΔW by reparameterizing it as the product of two matrices $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$, where r (a hyperparameter) indicates the rank and $r \ll \min(m, n)$ – low rank
  - $\Delta W = B \times A$
- The idea behind LoRA involves maintaining the original pre-trained parameters W constant (frozen) and learning ΔW to fine-tune the model for the new task
  - Weight change during fine-tuning is intrinsically low-rank
- The resulting updated layer parameters are then computed as $W + \Delta W$

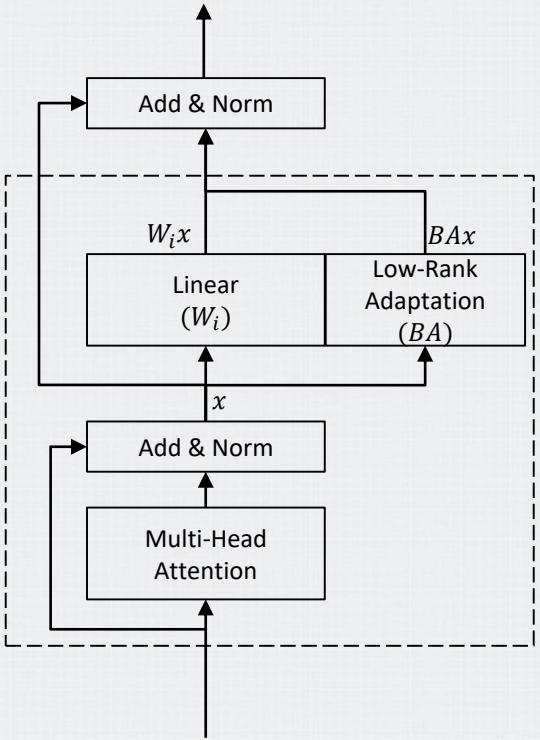
Hu et al. *LoRA: Low-rank Adaptation of Large Language Models*. International Conference on Learning Representations (ICLR), 2022

Low-Rank Adaptation (LoRA)

Transfer Learning

- $f_i(x) = W_i x + BAx \Rightarrow (W_i + BA)x$





Hu et al. *LoRA: Low-rank Adaptation of Large Language Models*. International Conference on Learning Representations (ICLR), 2022

LoRA as a Linear Combination of Random Basis

Transfer Learning

- Koohpayegani et al. (2024) proposed re-parameterizing the low-rank matrices in LoRA using linear combinations of randomly generated matrices (basis) and optimizing the linear mixture coefficients only

- Let $\{A_i \in \mathbb{R}^{r\times n}\}_{i=1}^{k}, \{B_j \in \mathbb{R}^{m\times r}\}_{j=1}^{l}$ be random matrices (basis) generated by a pseudo-random number generator with a **fixed seed**

- $A = \sum \alpha_i A_i, B = \sum \beta_j B_j$
  - α_i and β_j **are trainable parameters**
  - A and B are linear combinations of frozen random matrices

- $\Delta W = \sum \boldsymbol{\beta_j} B_j \times \sum \boldsymbol{\alpha_i} A_i$

Koohpayegani et al. *NOLA: Compressing LoRA using Linear Combination of Random Basis*. International Conference on Learning Representations (ICLR), 2024

GaLore and ReLoRA

Transfer Learning

- Gradient Low-rank Projection (GaLore)
  - Since the gradient G may have a low-rank structure, if we can keep the gradient statistics of a small "core" of gradient G in optimizer states, rather than G itself, then the memory consumption can be reduced substantially

- ReLoRA
  - Periodically merges BA into W, and initializes new BA with a reset on optimizer states and learning rate

GaLore Algorithm, PyTorch-like

```
for weight in model.parameters():
        grad = weight.grad
        lor_grad = project(grad) ▷  Project original space onto a compact space
        lor_update = update(lor_grad) ▷ Update by Adam, Adafactor, etc.
        update = project_back(lor_update) ▷ Project the compact space back onto the original space
        weight.data += update
```

Zhao et al. *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*. International Conference on Machine Learning (ICML), 2024 (Oral)
Lialin et al. *ReloRA: High-rank training through low-rank updates*. International Conference on Machine Learning (ICML), 2024

Limitations and Open Questions of PEFT Techniques

Transfer Learning

- LoRA is not shown to reach a comparable performance as standard fine-tuning (Xia et al., 2024)
- PEFT might take extra time to converge (i.e., obtain the same accuracy as standard fine-tuning). For example, up to $21.37\times$ more fine-tuning steps (Zhao et al., 2024)
- It remains an open question whether the weight matrix should be parameterized as low-rank. For example, in linear regression $\hat{y} = xW^T$, if the optimal W^* is high-rank, then imposing a low-rank assumption on W never leads to the optimal solution
- PEFT models do not improve inference efficiency because the model size remains the same or **even increases** after fine-tuning

Xia et al. *Chain of LoRA: Efficient Fine-tuning of Language Models via Residual Learning*. International Conference on Machine Learning (ICML), 2024

Zhao et al. *APT: Adaptive Pruning and Tuning Pretrained Language Models for Efficient Training and Inference*. International Conference on Machine Learning (ICML), 2024

Knowledge Distillation

Introduction

Knowledge Distillation

- The success of deep neural networks (DNNs) generally depends on:
  - Elaborate design of architectures (over-parameterized – deep/wide – models)
  - Learning from large and high-quality datasets

- Unfortunately, over-parameterized models are computationally expensive

- Large and high-quality datasets are laborious and expensive to collect

- A popular approach for mitigating these issues is to **distillate the knowledge** from well-trained models to facilitate learning on a simple and small model

Introduction

Knowledge Distillation

- Knowledge Distillation is often characterized by the so-called *Student-Teacher* (S-T) learning framework
  - The model **providing knowledge** is called the **teacher**
  - The model **learning the knowledge** is called the **student**

- The idea is to replicate the predictive ability of a large model (teacher) to a small model (student)

- In this learning paradigm, the smaller student network is under the supervision of a larger teacher network

Problem Definition

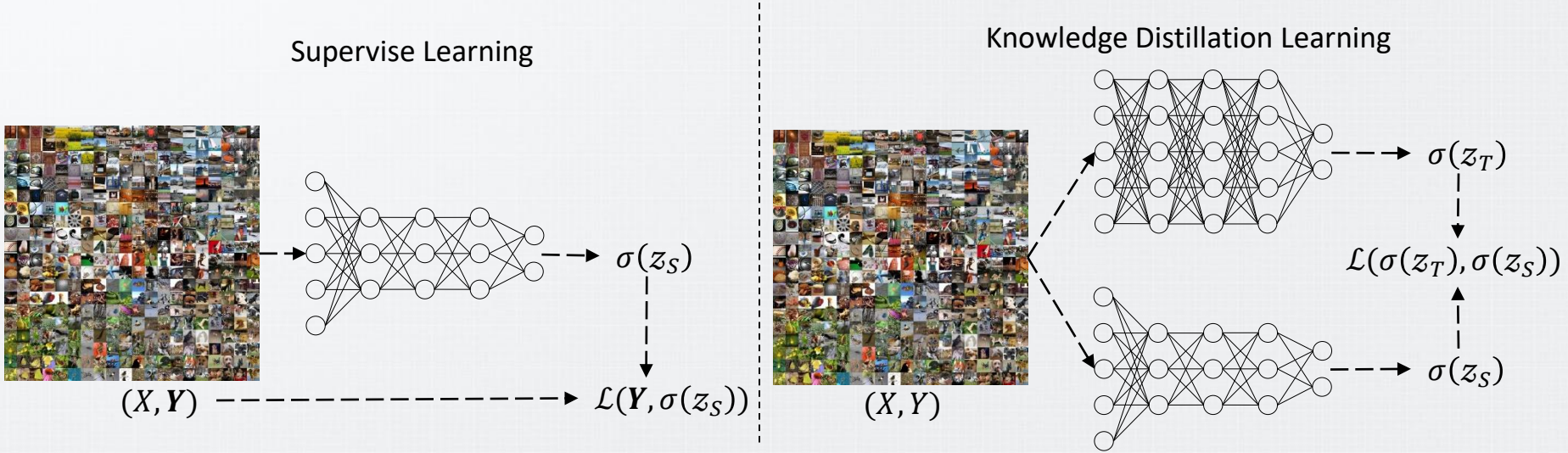
Knowledge Distillation

- Suppose $\mathcal{L}(\cdot,\cdot)$ be a loss function (i.e., categorical cross-entropy or ℓ_2)
- Assume $\sigma(\cdot)$ as a softmax function
- Let z_S and z_T be the logits of the student and teacher
  - Logits are the inputs to the final softmax activation

Problem Definition

Knowledge Distillation

- The knowledge is transferred from the teacher model to the student by minimizing the difference between the $\sigma(z_T)$ (logits from the teacher) model and $\sigma(z_T)$ (logits from the student): $\mathcal{L}(\sigma(z_T), \sigma(z_S))$
  - Remember that, in standard supervised learning, the loss is $\mathcal{L}(y, \sigma(z_S))$



Softmax Temperature

Knowledge Distillation

- The output of $\sigma(z_T)$ (teacher knowledge) has the correct class at a very high probability with all other class probabilities very close to zero
  - This does not provide much information beyond the ground truth labels (i.e., y) already provided in the dataset

- To address this problem, it is common to employ the ***softmax temperature***
  - $\sigma(z_i, \rho) = \frac{\exp\left(\frac{z_i}{\rho}\right)}{\sum_j \exp\left(\frac{z_i}{\rho}\right)}$
  - ρ is the temperature parameter, when $\rho = 1$ we get the standard softmax function
  - As ρ **increases**, the probability distribution produced by the softmax function **becomes softer**, providing more information as to which classes the teacher found more similar to the predicted class

Knowledge Distillation and Standard Loss

Knowledge Distillation

- The original Knowledge Distillation paper (Hitton et al., 2014) suggested that it is beneficial to train the student model together with the ground truth labels in addition to the teacher's soft labels

- Therefore, the overall Knowledge Distillation loss function is calculated in terms of
  - $\alpha \underbrace{\mathcal{L}\big(y, \sigma(z_S)\big)}_{\text{Student Loss (standard supervised)}} + \beta \underbrace{\mathcal{L}\big(\sigma(z_T, \rho), \sigma(z_S, \rho)\big)}_{\text{Distillation Loss (KD)}}$
  - α and β are coefficients that enable control the importance of each loss

Hinton et al. *Distilling the Knowledge in a Neural Network*. Neural Information Processing Systems (NeurIPS), Deep Learning Workshop. 2015

Bibliography

Bibliography

- Evci et al. *Head2Toe: Utilizing Intermediate Representations for Better Transfer Learning.* International Conference on Machine Learning (ICML), 2022

- Chen et al. *Project and Probe: Sample-Efficient Adaptation by Interpolating Orthogonal Features*. International Conference on Learning Representations (ICLR), 2024

ICLR
International Conference On
Learning Representations

ICML
International Conference
On Machine Learning

Bibliography

- Xu et al. *Initializing Models with Larger Ones*. International Conference on Learning Representations (ICLR), 2024

- Kirichenko et al. *Last Layer Re-training is Sufficient for Robustness to Spurious Correlations.* International Conference on Learning Representations (ICLR), 2023

- Howard et al. *Universal Language Model Fine-tuning for Text Classification*. Association for Computational Linguistics (ACL), 2018

ICLR
International Conference On Learning Representations

Association for Computational Linguistics