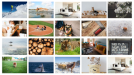
# Active Multi-task Representation Learning

Yifang Chen, Simon Du, Kevin Jamieson
University of Washington

# Background

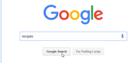
- Large-scale multi-task pretraining has become a standard approach in few-shot learning. (e.g. GPT-3, Clip, Bert)
- Multi-task can come from different sources, different domains, or even same sample with multi-labels

**Source tasks:**
For training the representation



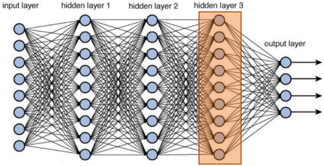
(Imagenet)

(wikipedia)




(various search terms)

......




**Target task** (VOC 07, 1-8 example per class): Few-shot learning with pretrained-representation


With representation learning, the accuracy improves **45% - 90%**

# Limitation of current methods and our solution

**BUT**, Using all feasible source tasks is computational costly

# Limitation of current methods and our solution

**BUT**, Using all feasible source tasks is computational costly

Also, Training on target-task-irrelevant tasks may hurt performance but due to the complexity of large-scale model, deciding the relevance in advance is difficult and problem-dependent.

# Limitation of current methods and our solution

**BUT**, Using all feasible source tasks is computational costly

Also, Training on target-task-irrelevant tasks may hurt performance but due to the complexity of large-scale model, deciding the relevance in advance is difficult and problem-dependent.

Solution:

- Learn target-source-task relevance by only using fewer samples from all source tasks
- Train the target model mostly on those relevant source samples.

# Formal problem setup

**Multi-task:** Given $M$ source tasks and one target task, denoted as task $M+1$, each task $m \in [M+1]$ is associated with a joint distribution $\mu_m$ over $\mathcal{X} \times \mathcal{Y}$.

# Formal problem setup

**Multi-task:** Given $M$ source tasks and one target task, denoted as task $M+1$, each task $m \in [M+1]$ is associated with a joint distribution $\mu_m$ over $\mathcal{X} \times \mathcal{Y}$.

**Multi-tasks representation learning:**

- Each i.i.d sample $(x, y)$ can be represented as $y = \phi^*(x)^\top w_m^* + \text{noise}$

# Formal problem setup

**Multi-task:** Given $M$ source tasks and one target task, denoted as task $M+1$, each task $m \in [M+1]$ is associated with a joint distribution $\mu_m$ over $\mathcal{X} \times \mathcal{Y}$.

**Multi-tasks representation learning:**

- Each i.i.d sample $(x, y)$ can be represented as $y = \phi^*(x)^\top w_m^* + \text{noise}$
- **Shared realizable representation function:** Given some function class $\Phi$, $\exists \phi^* : \mathcal{X} \to \mathcal{Z}$ that maps the input to some feature space $\mathcal{Z} \in \mathbb{R}^K$ where $K \ll d$.

# Formal problem setup

**Multi-task:** Given $M$ source tasks and one target task, denoted as task $M+1$, each task $m \in [M+1]$ is associated with a joint distribution $\mu_m$ over $\mathcal{X} \times \mathcal{Y}$.

**Multi-tasks representation learning:**

- Each i.i.d sample $(x, y)$ can be represented as $y = \phi^*(x)^\top w_m^* + \text{noise}$
- **Shared realizable representation function:** Given some function class $\Phi$, $\exists \phi^* : \mathcal{X} \to \mathcal{Z}$ that maps the input to some feature space $\mathcal{Z} \in \mathbb{R}^K$ where $K \ll d$.
- **Task specified linear predictor:** For each $m \in [M]$, $\exists w_m^* \in \mathbb{R}^k$ that maps feature space to output space.

# Formal problem setup

**Few-shot learning and large-scale pretraining**

- **Fixed and small amount of target samples:** We have only a small, fixed amount of data $\{x^i_{M+1}, y^i_{M+1}\}_{i \in [n_{M+1}]}$ drawn i.i.d. from the target task distribution $\mu_{M+1}$.

# Formal problem setup

**Few-shot learning and large-scale pretraining**

- **Fixed and small amount of target samples:** We have only a small, fixed amount of data $\{x^i_{M+1}, y^i_{M+1}\}_{i \in [n_{M+1}]}$ drawn i.i.d. from the target task distribution $\mu_{M+1}$.
- **Unlimited access to source samples:** At any point during learning we assume we can obtain an i.i.d. sample from any source task $m \in [M]$ without limit.

# Formal problem setup

**Few-shot learning and large-scale pretraining**

- **Fixed and small amount of target samples:** We have only a small, fixed amount of data $\{x^i_{M+1}, y^i_{M+1}\}_{i\in[n_{M+1}]}$ drawn i.i.d. from the target task distribution $\mu_{M+1}$.
- **Unlimited access to source samples:** At any point during learning we assume we can obtain an i.i.d. sample from any source task $m \in [M]$ without limit.
- **Source tasks are diverse enough**

# Formal problem setup

**Our goal** is to use as few total samples from the source tasks ($N_{\text{total}}$) as possible to learn a representation and linear predictor $\phi, w_{M+1}$ that minimizes the excess risk on the target task defined as

$$\text{ER}_{M+1}(\phi, w) = L_{M+1}(\phi, w) - L_{M+1}(\phi^*, w^*_{M+1})$$

where $L_{M+1}(\phi, w) = \mathbb{E}_{(x,y)\sim\mu_{M+1}}\left[(\langle\phi(x), w\rangle - y)^2\right]$.

# Formal problem setup

**Our goal** is to use as few total samples from the source tasks ($N_{\text{total}}$) as possible to learn a representation and linear predictor $\phi, w_{M+1}$ that minimizes the excess risk on the target task defined as

$$\text{ER}_{M+1}(\phi, w) = L_{M+1}(\phi, w) - L_{M+1}(\phi^*, w^*_{M+1})$$

where $L_{M+1}(\phi, w) = \mathbb{E}_{(x,y)\sim\mu_{M+1}}\left[(\langle\phi(x), w\rangle - y)^2\right]$.

And of course, we want to keep same amount of target sample complexity.

# Summary of contributions

- We design active learning algorithm that iteratively samples from tasks to estimate the source-target-task-relevance and also simultaneously learn the target model

# Summary of contributions

- We design active learning algorithm that iteratively samples from tasks to estimate the source-target-task-relevance and also simultaneously learn the target model
- We prove that when the representation function class is linear, our algorithm never performs worse than uniform sampling, and can save up to a factor of $M$( the *number of source tasks*), compared with the naive uniform sampling from all source tasks.

# Summary of contributions

- We design active learning algorithm that iteratively samples from tasks to estimate the source-target-task-relevance and also simultaneously learn the target model
- We prove that when the representation function class is linear, our algorithm never performs worse than uniform sampling, and can save up to a factor of $M$( the *number of source tasks*), compared with the naive uniform sampling from all source tasks.
- We empirically demonstrate the effectiveness of our active learning algorithm by testing it on the corrupted MNIST dataset with both linear and convolutional neural network (CNN) representation function classes.

# Theoretical result

## Theorem (Informal)

*Suppose we know some* $\underline{\sigma} \geq \sigma_{\min}(W^*)$. *Under the benign low-dimension linear representation setting, with proper choice of* $\beta$, *we have* $\mathrm{ER}(\hat{B}, \hat{w}_{M+1}) \leq \varepsilon^2$ *with probability at least* $1-\delta$ *whenever*

$$N_{total} \gtrapprox \left(K(M+d) + \log\frac{1}{\delta}\right)\sigma^2 s^* \|\nu^*\|_2^2 \varepsilon^{-2} + \square\sigma\varepsilon^{-1}$$

*where* $\square = \left(MK^2 dR/\underline{\sigma}^3\right)\sqrt{s^*}$.

**Remarks:**

- $s^*$ is the approximate sparsity that we saved compared to passive learning (which is $M$)
- $\nu^*$ is the target-source relevance. How to estimate this relevance is the key to achieve this $s^*$-dependent result.

# Experiments - setup

**Design:**

- **Original dataset:** MNIST-C(orruption) proposed by Mu &Gilmer (2019), which consists of 16 different types of corruptions applied to the MNIST test set.
- **Multi-task data formulation:** We divide dataset into 160 tasks by applying one-hot encoding to 0-9 labels to each corruption type.
- **Models for representation function:** We test both linear and 2-layer ReLU convolutional neural net model and use $l_2$ loss.

# Experiments -results

**Performance between the adaptive (ada) and the non-adaptive (non-ada) algorithms.**

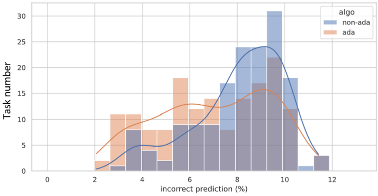



Figure: Linear model

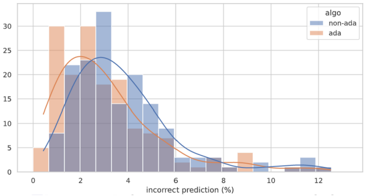



Figure: 2-layer convnet model

In linear model, the adaptive algorithm achieves 1.1% higher average accuracy than the non-adaptive one and results same or better accuracy in 136 out of 160 tasks. In convnet, it achieves 0.68% higher average accuracy than the non-adaptive one and results same or better accuracy in 133 out of 160 tasks.

# Thanks!

Hall E 1220, Poster session 2