

# Theoretical Foundations of Pre-trained Models

Qi Lei

Princeton University



# A.I. is Everywhere



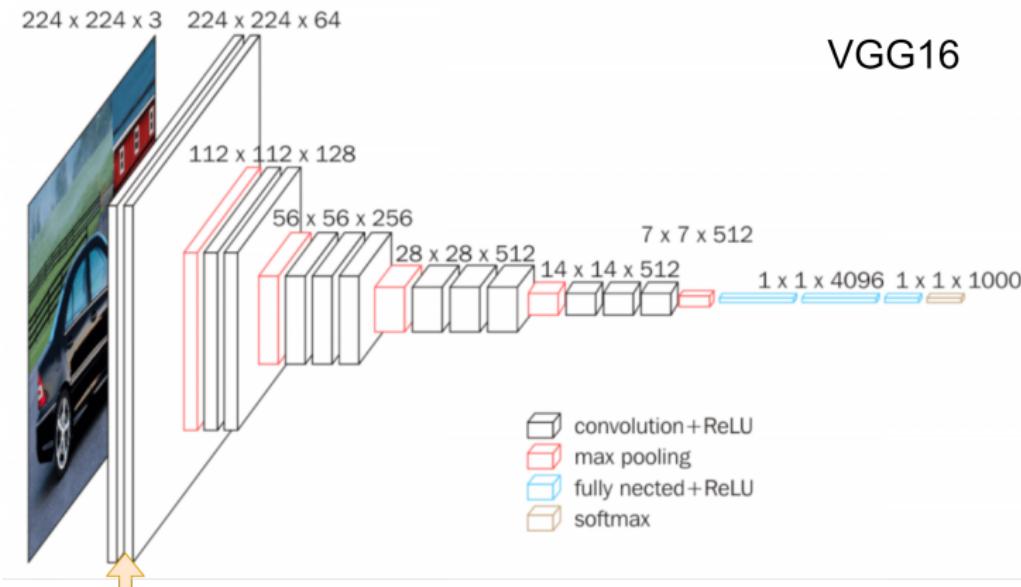
# A.I. is Everywhere



# A.I. is Everywhere



# Deep Learning



- Deep learning succeeds with abundant labeled data.

# Emerging Application Domains



- Fundamental areas to our society
- Labeled data is lacking: significant costs in money and time
- Demands comprehensive domain knowledge

# Emerging Application Domains

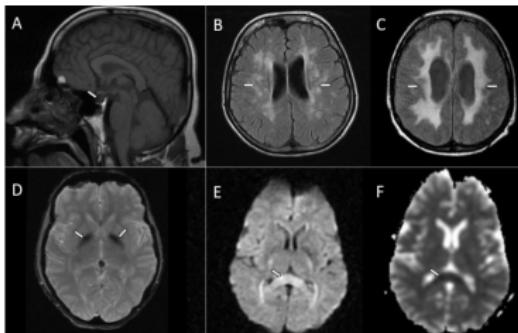


## Medical images

- X-ray



- MRI scan



# Pre-trained Models

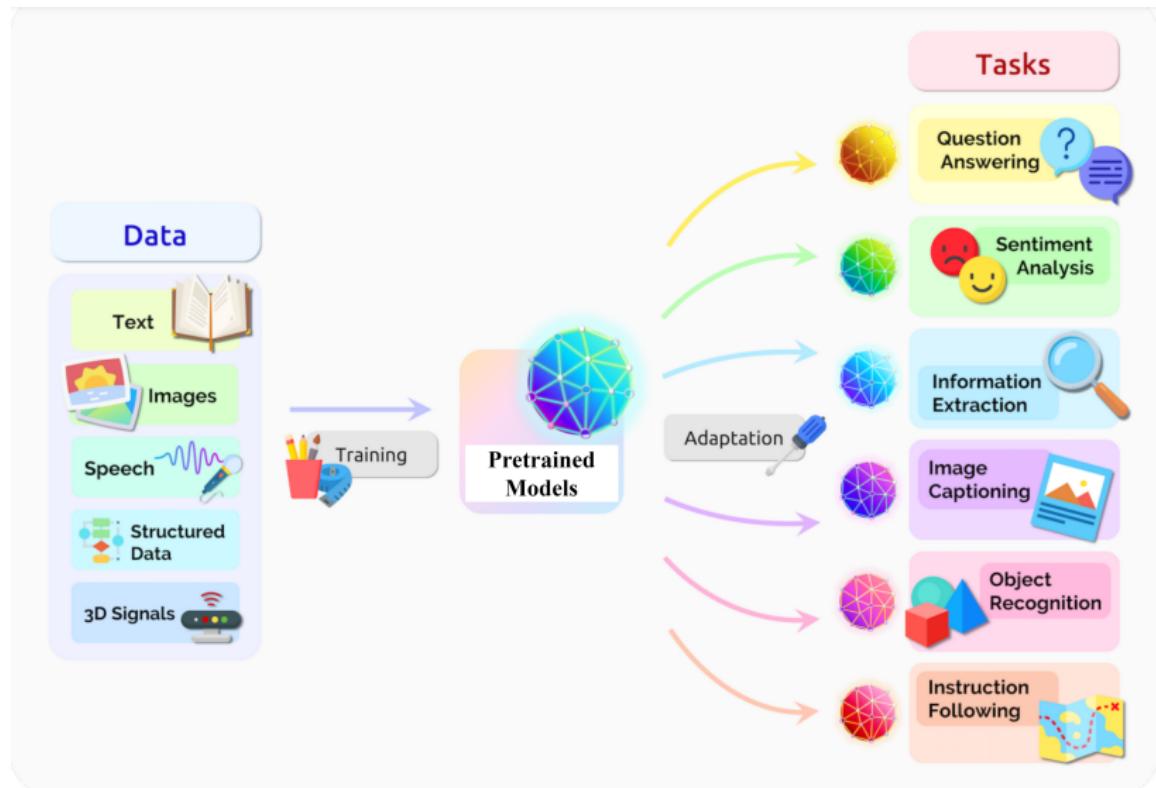
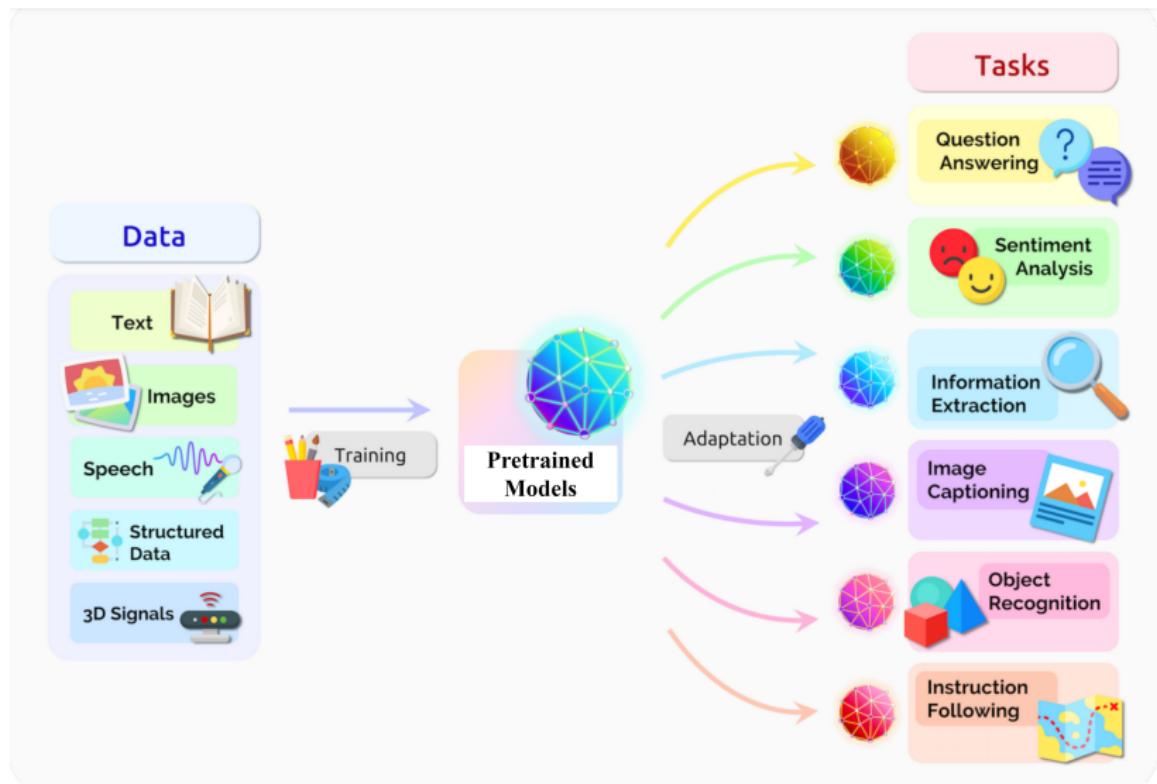


Figure credit to: "On the Opportunities and Risks of Foundation Models", CRFM, HAI, Stanford

# Pre-trained Models



BERT, GPT-3, Codex, AlphaCode, SimCLR, MAE, DALL-E, CLIP

# Example: DALL-E

DALL-E: create images from text captions

# Example: DALL-E

DALL-E: create images from text captions

a pentagonal green clock. a green clock in the shape of a pentagon.



a cube made of porcupine. a cube with the texture of a porcupine.



a collection of glasses is sitting on a table



Figures source: OpenAI DALL-E playground

7/40

# Example: CLIP

Zero shot image classifier

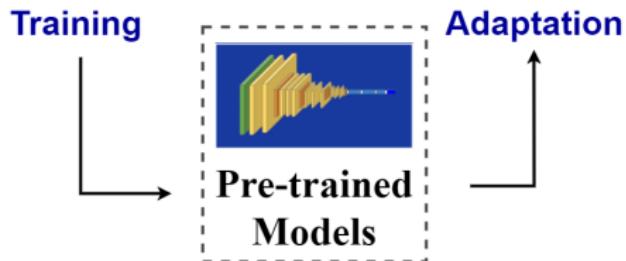
FOOD101

**guacamole** (90.1%) Ranked 1 out of 101 labels



- ✓ a photo of **guacamole**, a type of food.
- ✗ a photo of **ceviche**, a type of food.
- ✗ a photo of **edamame**, a type of food.
- ✗ a photo of **tuna tartare**, a type of food.
- ✗ a photo of **hummus**, a type of food.

Figure source: OpenAI, <https://clip.backprop.co/>



Theory's Role on  
Pre-trained models?



- What training tasks are useful for downstream tasks?
- What algorithm/architecture can identify the useful features?
- How many samples are required?

- guide technical decisions
- reduce trial and error
- forecast outcomes and risks
- inspire new methods

Current learning theory is relatively more mature in **supervised learning**.

- **Big labeled** dataset is necessary to fit deep learning.
- Training and testing data follow the **same distribution**.

# Challenge

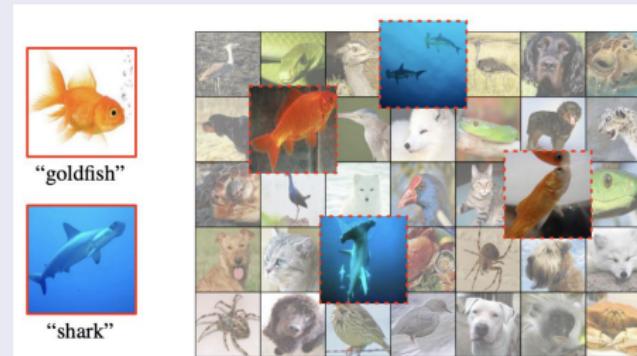
Current learning theory is relatively more mature in **supervised learning**.

- **Big labeled** dataset is necessary to fit deep learning.
- Training and testing data follow the **same distribution**.

## Our target

We want to understand how pre-trained model can

- adapt to new tasks **quickly**,



Current learning theory is relatively more mature in **supervised learning**.

- **Big labeled** dataset is necessary to fit deep learning.
- Training and testing data follow the **same distribution**.

## Our target

We want to understand how pre-trained model can

- adapt to new tasks **quickly**,
- be learned from **unlabeled samples**,



# Challenge

Current learning theory is relatively more mature in **supervised learning**.

- **Big labeled** dataset is necessary to fit deep learning.
- Training and testing data follow the **same distribution**.

## Our target

We want to understand how pre-trained model can

- adapt to new tasks **quickly**,
- be learned from **unlabeled samples**,
- handle **distributional shift** from training to adaptation



Autonomous driving  
Trained on sunny weather



Tested on rainy weather

## 1 Meta-learning

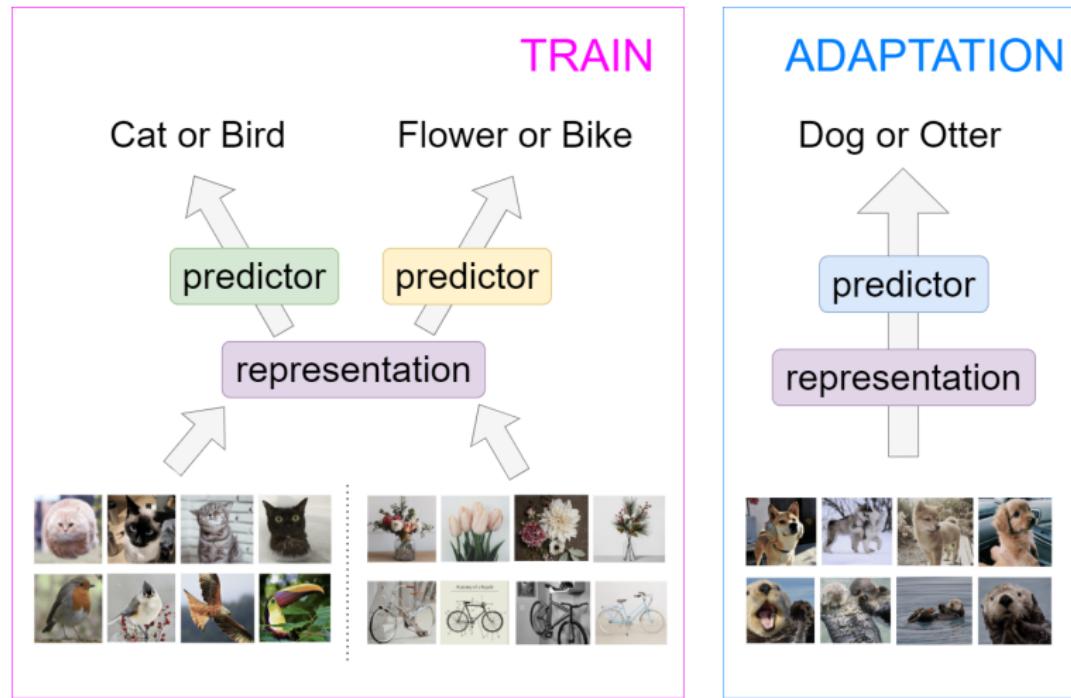
- Meta-Learning with Frozen Representation
- Meta-learning with Fine-tuned Representation

## 2 Self-Supervised Learning

## 3 Ongoing and Future Work

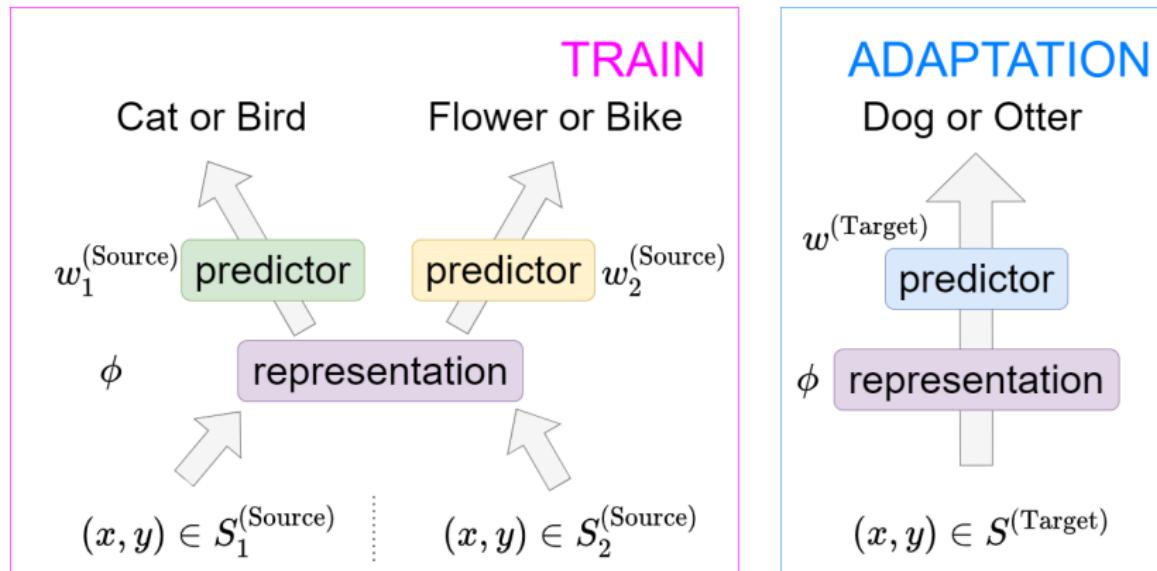
- Domain Adaptation
- Lifelong Learning
- Meta Reinforcement Learning

# Learning the Meta-Representation

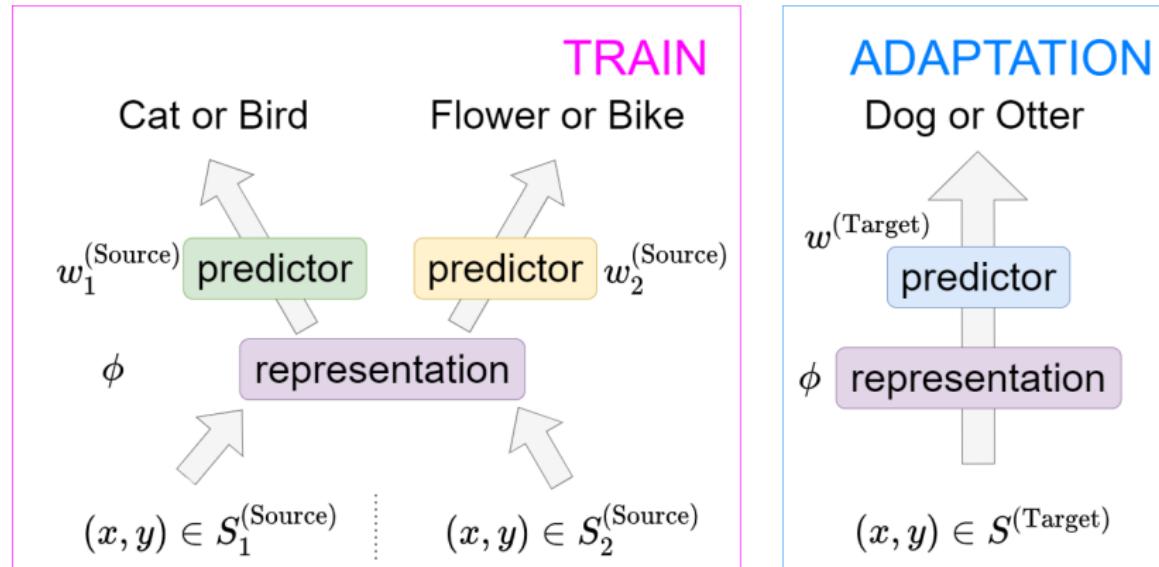


Prototypical network: (Snell et al. 2017), Meta-learning representation: (Javed and White, 2019)

# Algorithm



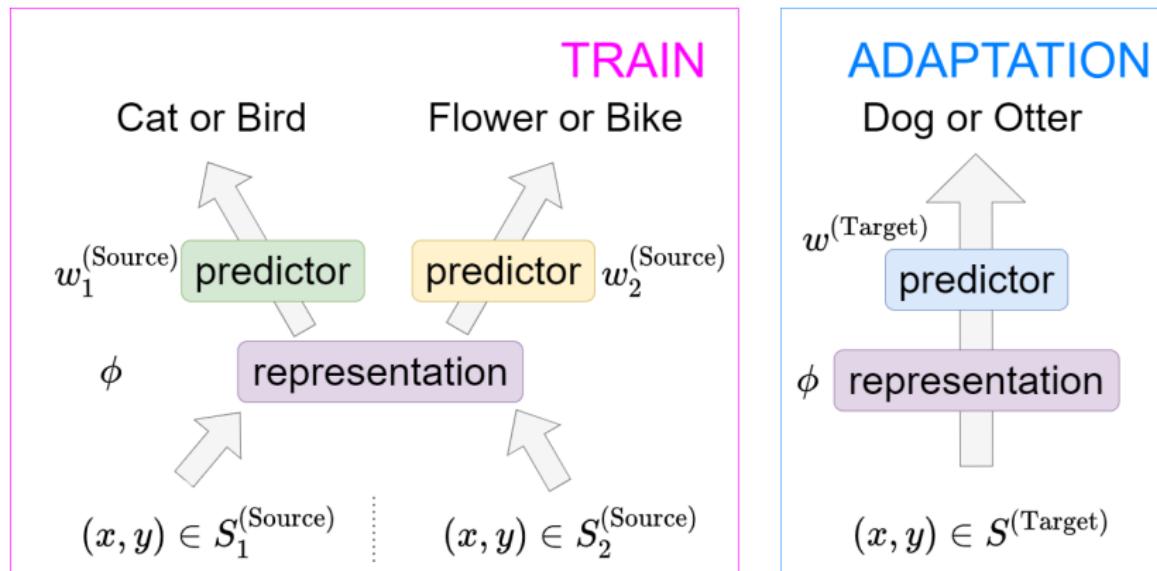
# Algorithm



On source tasks:

$$\hat{\phi} \leftarrow \arg \min_{\phi \in \Phi} \sum_{t=1}^{\#\text{source tasks}} \left\{ \min_{w_t} \sum_{(x,y) \in S_t^{(\text{Source})}} \text{loss}(y, w_t \circ \phi(x)) \right\}.$$

# Algorithm



On target task:

$$\hat{w}^{(\text{Target})} \leftarrow \arg \min_w \sum_{(x,y) \in S^{(\text{Target})}} \text{loss} \left( y, w \circ \hat{\phi}(x) \right).$$

# How to Quantify Task Similarities

Assumptions:

- ① Shared good representation across tasks:

There exist predictors  $w_t$ , representation function  $\phi$ ,  
 $y_t = (w_t)^\top \phi(x) + \text{noise}$ ,  $\phi \in \Phi$  for both source and target  
tasks.

- ② Is this enough?

# How to Quantify Task Similarities

Assumptions:

- ① Shared good representation across tasks:

There exist predictors  $w_t$ , representation function  $\phi$ ,  
 $y_t = (w_t)^\top \phi(x) + \text{noise}$ ,  $\phi \in \Phi$  for both source and target  
tasks.

- ② Is this enough?

Behind the scenes

- ① Shared representation encodes what transfers across the tasks.
- ② Need source tasks  $\{w_t^{(\text{Source})}\}$  diverse enough to “cover”  
 $w^{(\text{Target})}$ .

- ➊ Shared representation encodes what transfers across the tasks.
- ➋ Diversity of source tasks  $\{w_t^{(\text{Source})}\}$  (at least needs to “cover” the target task.)

## Task diversity

Source tasks:  
Classify types of dogs

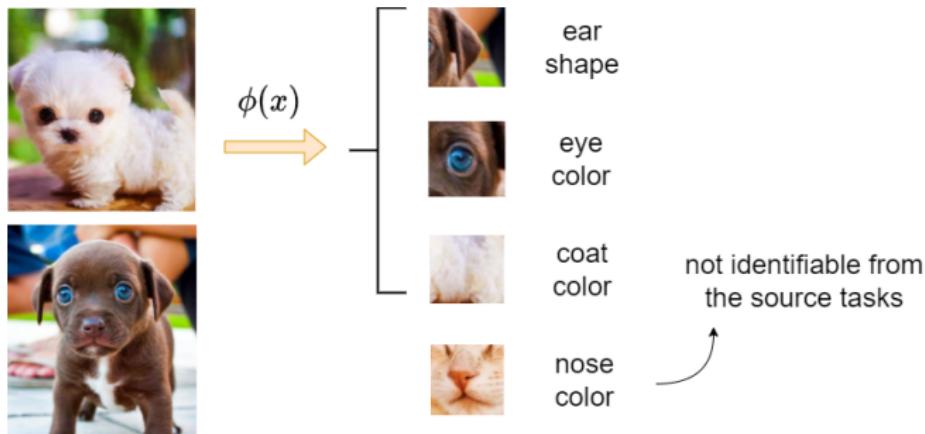


Target task:  
Cat or dog?



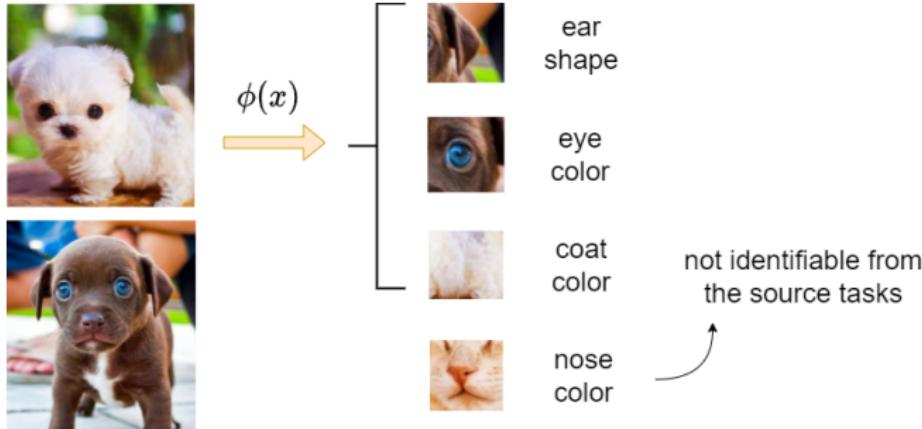
# Importance of Task Diversity

- ① Shared representation encodes what transfers across the tasks.
- ② Diversity of source tasks  $\{w_t^{(\text{Source})}\}$  (at least needs to “cover” the target task.)



# Importance of Task Diversity

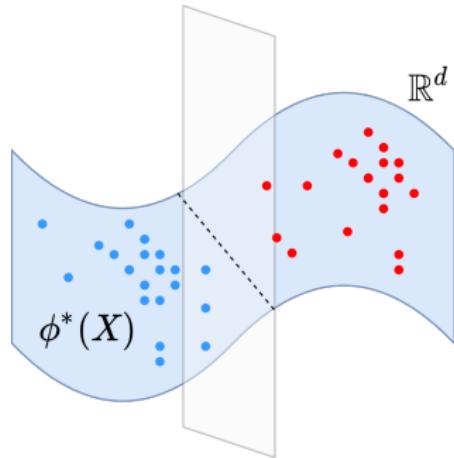
- ① Shared representation encodes what transfers across the tasks.
- ② Diversity of source tasks  $\{w_t^{(\text{Source})}\}$  (at least needs to “cover” the target task.)



Mathematically speaking,  $w^{(\text{Target})} \in \text{span}\{w_1^{(\text{Source})}, \dots, w_{ne}^{(\text{Source})}\}$ .

Setup:

- Shared representation:  
 $y_t = w_t^\top \phi(x_t) + \text{noise}$
- Representation layer is of dimension  $k$  (We assume  $k$  is small)



## Theorem 1 (Informal)

We only need  $O(k)$  labeled samples from target domain to get small **test error**.

In contrast, supervised learning requires samples up to the **complexity of the function class**. (500 vs.  $10^{13}$ .)

## Theorem 1

With shared representation and task diversity,

Test Error( $\hat{w}^{(\text{Target})} \circ \hat{\phi}$ )  $\leq$  Representation Error + Adaptation Error.

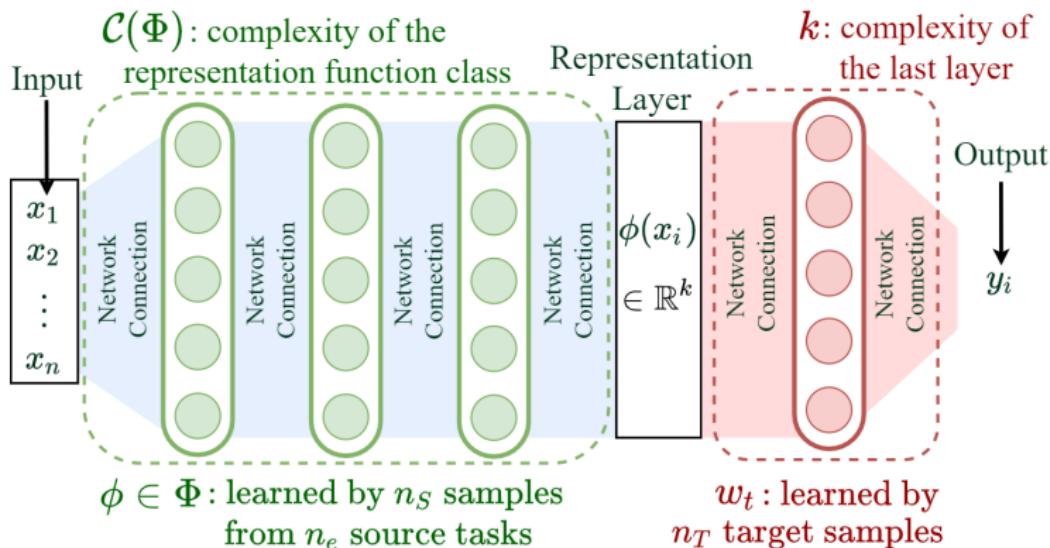
- Representation error: how well you learn representation layer  $\phi$
- Adaptation error: how well you learn target-task predictor  $w^{(\text{Target})}$

# Main Result on Meta Representation

## Theorem 1

With shared representation and task diversity,

$$\text{Test Error}(\hat{w}^{(\text{Target})} \circ \hat{\phi}) \lesssim \underbrace{\frac{\mathcal{C}(\Phi)}{n_S n_e}}_{\text{representation error}} + \underbrace{\frac{k}{n_T}}_{\text{adaptation error}}.$$



## Theorem 1

With shared representation and task diversity,

$$\text{Test Error}(\hat{w}^{(\text{Target})} \circ \hat{\phi}) \lesssim \underbrace{\frac{\mathcal{C}(\Phi)}{n_S n_e}}_{\text{representation error}} + \underbrace{\frac{k}{n_T}}_{\text{adaptation error}}.$$

Baselines:

- Supervised learning:

$$\text{Test error} \leq \frac{\mathcal{C}(w \circ \Phi)}{n_T}.$$

- Maurer et al. 2016:

$$\text{Test error} \leq \frac{\mathcal{C}(\Phi)}{\sqrt{n_e}} + \frac{k}{n_T}.$$

## Theorem 1

With shared representation and task diversity,

$$\text{Test Error}(\hat{w}^{(\text{Target})} \circ \hat{\phi}) \lesssim \underbrace{\frac{\mathcal{C}(\Phi)}{n_S n_e}}_{\text{representation error}} + \underbrace{\frac{k}{n_T}}_{\text{adaptation error}}.$$

Remark:

- Covariate shift is allowed: source and target data can come from different marginal distribution.
- Representation  $\phi$  selects  $k$  most important features from the data.

## Theorem 1

With shared representation and task diversity,

$$\text{Test Error}(\hat{w}^{(\text{Target})} \circ \hat{\phi}) \lesssim \underbrace{\frac{\mathcal{C}(\Phi)}{n_S n_e}}_{\text{representation error}} + \underbrace{\frac{k}{n_T}}_{\text{adaptation error}}.$$

Remark:

- Covariate shift is allowed: source and target data can come from different marginal distribution.
- Representation  $\phi$  selects  $k$  most important features from the data.

## 1 Meta-learning

- Meta-Learning with Frozen Representation
- Meta-learning with Fine-tuned Representation

## 2 Self-Supervised Learning

## 3 Ongoing and Future Work

- Domain Adaptation
- Lifelong Learning
- Meta Reinforcement Learning

What if there is misspecification in the representation?  
(Namely, the representation is only approximately shared.)

What if there is misspecification in the representation?  
(Namely, the representation is only approximately shared.)

- Previously:  $y_t = w_t^\top \phi(x) + \text{noise}$ .

What if there is misspecification in the representation?  
(Namely, the representation is only approximately shared.)

- Previously:  $y_t = w_t^\top \phi(x) + \text{noise}$ .
- Now:  $y_t = w_t^\top \phi_t(x) + \text{noise}$ ,  $\phi_t$  is  $\gamma$ -close to  $\phi$ .

What if there is misspecification in the representation?  
(Namely, the representation is only approximately shared.)

- Previously:  $y_t = w_t^\top \phi(x) + \text{noise}$ .
- Now:  $y_t = w_t^\top \phi_t(x) + \text{noise}$ ,  $\phi_t$  is  $\gamma$ -close to  $\phi$ .

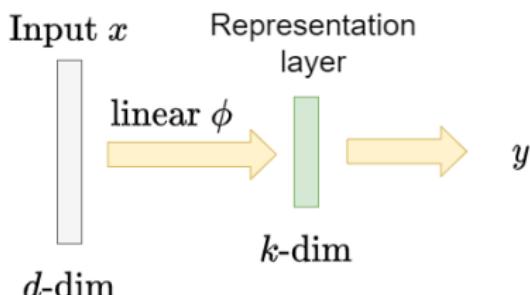
## Question

Does METAREP (previous algorithm) still work?

*If not, how should we modify the algorithm?*

# Failure with Previous Algorithm: METAREP

Instantiation in linear setting:



When  $\gamma = 0$  (no misspecification), METAREP requires at most  $O(k)$  samples on the target task.

Instantiation in linear setting:

When  $\gamma = 0$  (no misspecification), METAREP requires at most  $O(k)$  samples on the target task.

Theorem: However, when  $\gamma > 0$ , METAREP requires at least  $\Omega(d)$  samples on the target task

Remark: no improvement over traditional supervised learning that requires  $O(d)$  samples.

- ① Use source tasks to find  $\phi$  as an initialization.
- ② Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

- ① Use source tasks to find  $\phi$  as an initialization.
- ② Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

$$\min_{\phi} \min_{\|\phi_t - \phi\| \leq \gamma, w_t} \sum_{\text{Task } t} \text{loss}(w_t, \phi_t),$$

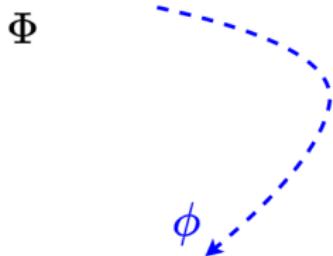
- ① Use source tasks to find  $\phi$  as an initialization.
- ② Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

$$\Phi$$

$$\min_{\phi} \min_{\|\phi_t - \phi\| \leq \gamma, w_t} \sum_{\text{Task } t} \text{loss}(w_t, \phi_t),$$

- ① Use source tasks to find  $\phi$  as an initialization.
- ② Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

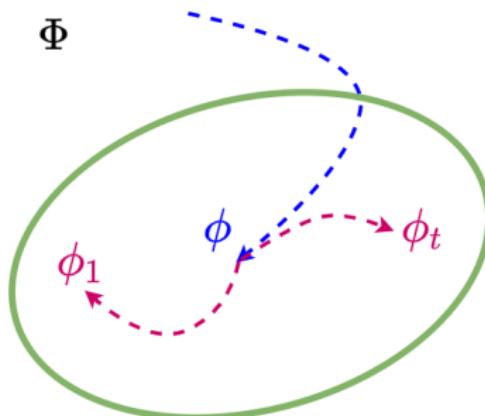
$$\min_{\phi} \min_{\|\phi_t - \phi\| \leq \gamma, w_t} \sum_{\text{Task } t} \text{loss}(w_t, \phi_t),$$



# Modified Algorithm: FINE-TUNEDREP

- ① Use source tasks to find  $\phi$  as an initialization.
- ② Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

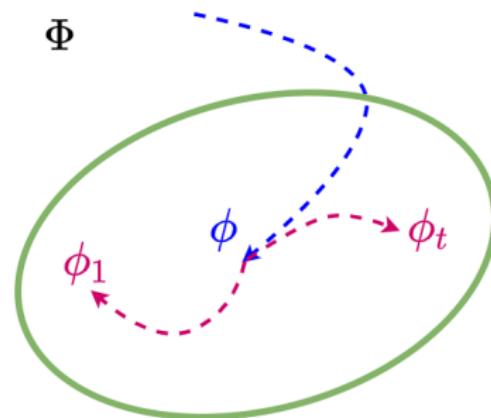
$$\min_{\phi} \min_{\|\phi_t - \phi\| \leq \gamma, w_t} \sum_{\text{Task } t} \text{loss}(w_t, \phi_t),$$



## Modified Algorithm: FINE-TUNEDREP

- 1 Use source tasks to find  $\phi$  as an initialization.
- 2 Fine-tune each representation  $\phi_t$  starting from  $\phi$  that tolerates mis-specification.

$$\min_{\phi} \min_{\|\phi_t - \phi\| \leq \gamma, w_t} \sum_{\text{Task } t} \text{loss}(w_t, \phi_t),$$



### Theorem 2 (Informal)

When adapting  $\phi$  to target task, it requires  $O(k) + O(\gamma^2)$  training samples from target domain.

## Theorem 2

For **general function classes**, under similar settings,

$$\text{Test Error} \lesssim \text{METAREP} \text{ (when } \gamma = 0) + O\left(\frac{\gamma}{\sqrt{n_T}}\right).$$

- $\gamma$  measures mis-specification in representation  $\phi$
- $n_T$ : number of samples from target training set

## Theorem 2

For **general function classes**, under similar settings,

$$\text{Test Error} \lesssim \text{METAREP} \text{ (when } \gamma = 0) + O\left(\frac{\gamma}{\sqrt{n_T}}\right).$$

- We need  $O(k) + O(\gamma^2)$  samples from target domain.

Baselines:

- Supervised learning and METAREP need  $\mathcal{C}(w \circ \Phi)$  samples from target domain.
- $\mathcal{C}(w \circ \Phi)$ : Complexity of the function class for the whole network.

## 1 Meta-learning

- Meta-Learning with Frozen Representation
- Meta-learning with Fine-tuned Representation

## 2 Self-Supervised Learning

## 3 Ongoing and Future Work

- Domain Adaptation
- Lifelong Learning
- Meta Reinforcement Learning

Create your own labels

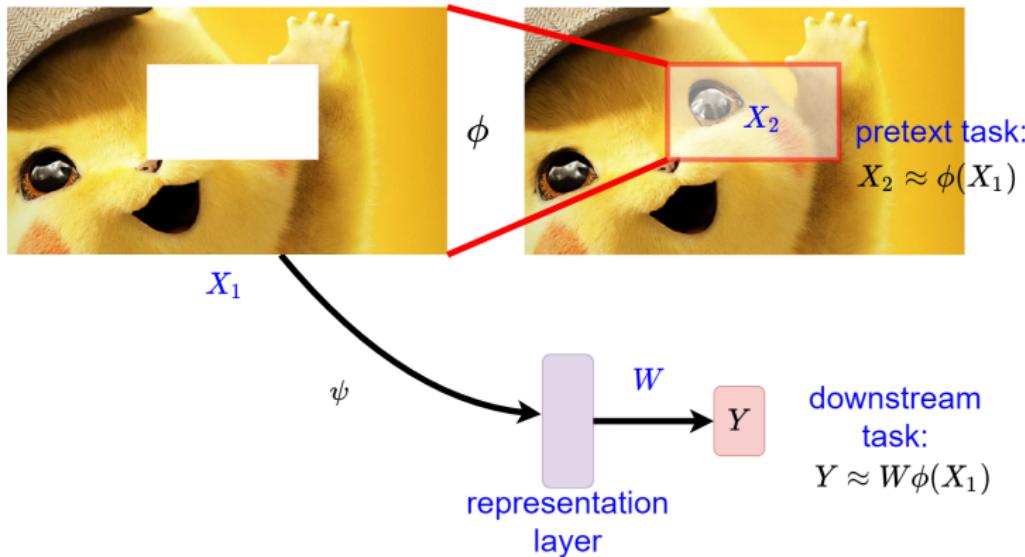
Supervised representation learning needs labels from related tasks.  
What if this isn't available?

**Create pseudo-labels from the input data.**

# Self-supervised Learning

## Type I: reconstruction-based SSL

Reconstructing part of the input from the other part



Context encoder: (Pathak et al. 2016)

Other examples: Masked Autoencoder: (He et al., 2021), Colorization: (Zhang et al., 2016)

## Type I: reconstruction-based SSL

Reconstructing part of the input from the other part

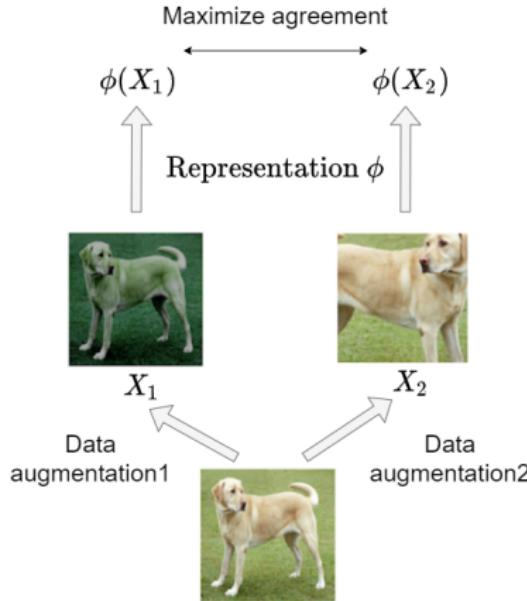


BERT: (Devlin et al., 2018)

Other examples: Masked Autoencoder: (He et al., 2021), Colorization: (Zhang et al., 2016)

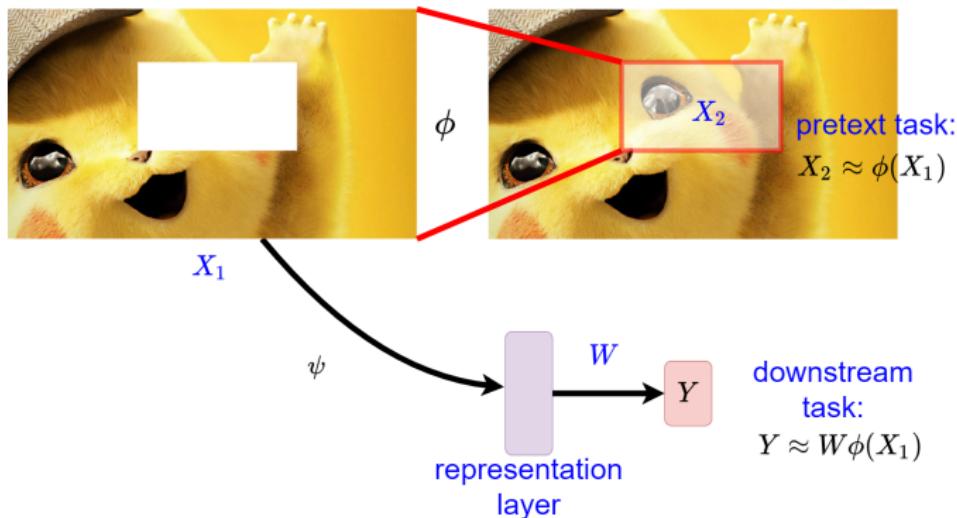
## Type II: similarity-based SSL

Enforcing two views of the same data to have similar representation

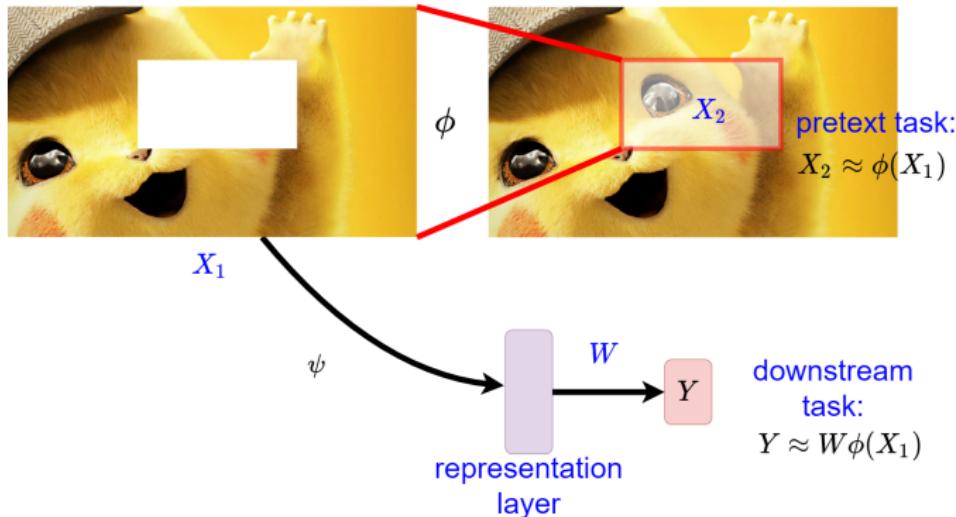


Examples: SimSiam: (Chen et al., 2021), CLIP: (Radford et al., 2021) ,  
SimCLR: (Chen et al., 2020)

- ① Label  $Y$  with  $k$  classes.
- ② Unmasked image  $X_1$  and masked image  $X_2$ .



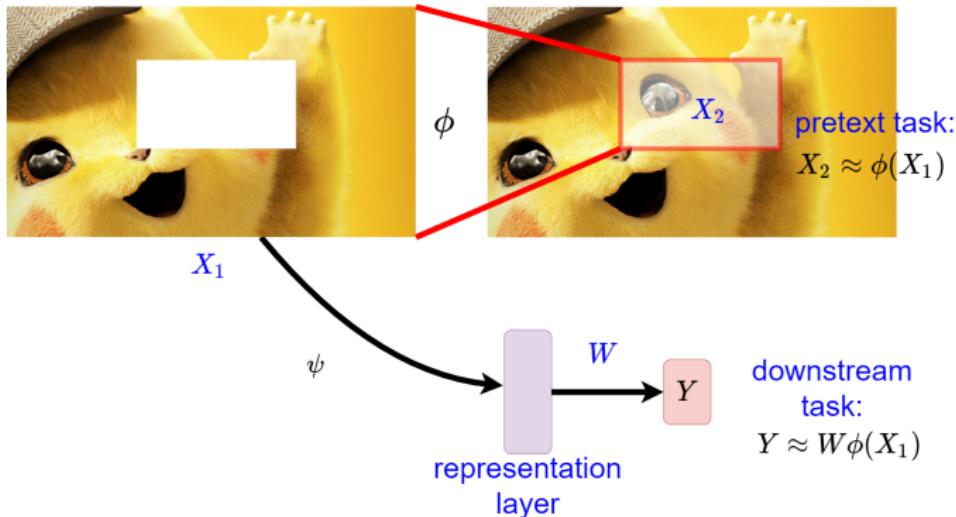
- ① Label  $Y$  with  $k$  classes.
- ② Unmasked image  $X_1$  and masked image  $X_2$ .



- ③ Key intuition: Pretext tasks should help us reduce irrelevant features/forget information that is not necessary to predict  $Y$

# Ideal Scenario

- ① Label  $Y$  with  $k$  classes.
- ② Unmasked image  $X_1$  and masked image  $X_2$ .



- ③ Ideal scenario:  $X_1 \rightarrow Y \rightarrow X_2$   
 $\iff X_1 \perp X_2 | Y$

Setting:

- ①  $k$ -class labels  $Y$ .
- ② Representation  $\phi$ , last layer  $W^*$ .

Compare this procedure to ground truth classifier  $f^*$ :

## Theorem 3

- ① (No representation error.) If  $X_1 \perp X_2 | Y$ ,

$$f^* = W^* \phi(X_1).$$

- ② Only need  $O(k)$  labeled samples.

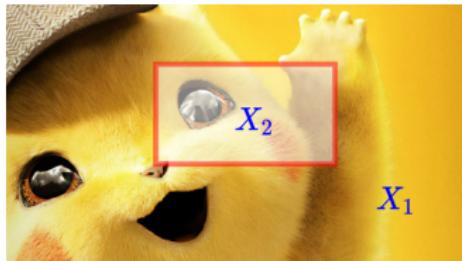
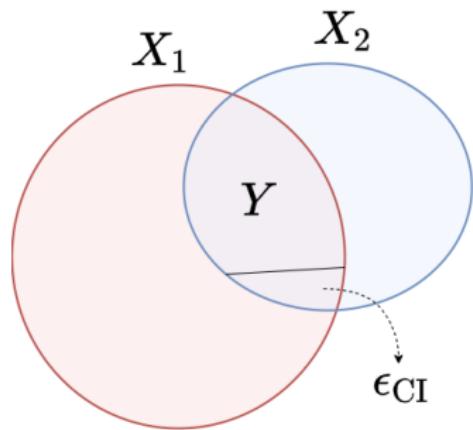
Remark: Only need  $k$  samples instead of Rademacher complexity of function class.

No representation error: if  $X_1 \rightarrow Y \rightarrow X_2$ , (i.e.,  $X_1 \perp X_2 | Y$ ),  
 then  $f^* = W^* \phi(X_1)$ .

$$\begin{aligned} \phi(\cdot) &:= \mathbb{E}[X_2|X_1] \quad \overbrace{\quad}^{\text{tower property}} \quad \mathbb{E}[\mathbb{E}[X_2|X_1, Y]|X_1] \overbrace{\quad}^{\text{CI}} \quad \mathbb{E}[\mathbb{E}[X_2|Y]|X_1] \\ &= \sum_{y=1}^k \mathbb{E}[X_2|Y=y] P(Y=y|X_1) =: \mathbf{A}^\top f(X_1), \end{aligned}$$

Here  $f(x_1)_y := P(Y=y|X_1=x_1), y=1 \cdots, k$   
 $\mathbf{A}$  satisfies  $\mathbf{A}_{y,:} = \mathbb{E}[X_2|Y=y]$ .

# Main Results on Reconstruction-based SSL



## Characterizing Approximate Conditional Independence

Define  $\epsilon_{CI} = \mathbb{E}_{X_1} \|\mathbb{E}[X_2|X_1] - \mathbb{E}_Y[\mathbb{E}[X_2|Y]|X_1]\|^2$ .

- This is 0 if  $X_1 \perp X_2 | Y$ .
- $\epsilon_{CI}$  can be viewed as quantification of extra shared features between  $X_1$  and  $X_2$  are not captured by  $Y$  (spurious feature not reducible by SSL)

## Characterizing Approximate Conditional Independence

Define  $\epsilon_{CI} = \mathbb{E}_{X_1} \|\mathbb{E}[X_2|X_1] - \mathbb{E}_Y[\mathbb{E}[X_2|Y]|X_1]\|^2$ .

- This is 0 if  $X_1 \perp X_2 | Y$ .
- $\epsilon_{CI}$  can be viewed as quantification of extra shared features between  $X_1$  and  $X_2$  are not captured by  $Y$  (spurious feature not reducible by SSL)

$$\text{Test Error} \lesssim \underbrace{\frac{k}{n_L}}_{\text{adaptation error}} + \underbrace{\epsilon_{CI}}_{\text{representation error}}$$

## Characterizing Approximate Conditional Independence

Define  $\epsilon_{CI} = \mathbb{E}_{X_1} \|\mathbb{E}[X_2|X_1] - \mathbb{E}_Y[\mathbb{E}[X_2|Y]|X_1]\|^2$ .

- This is 0 if  $X_1 \perp X_2 | Y$ .
- $\epsilon_{CI}$  can be viewed as quantification of extra shared features between  $X_1$  and  $X_2$  are not captured by  $Y$  (spurious feature not reducible by SSL)

$$\text{Test Error} \lesssim \underbrace{\frac{k}{n_L}}_{\text{adaptation error}} + \underbrace{\epsilon_{CI}}_{\text{representation error}}$$

- ERM would need  $n_L \asymp \text{Complexity of Function Class}$ .

## Characterizing Approximate Conditional Independence

Define  $\epsilon_{CI} = \mathbb{E}_{X_1} \|\mathbb{E}[X_2|X_1] - \mathbb{E}_Y[\mathbb{E}[X_2|Y]|X_1]\|^2$ .

- This is 0 if  $X_1 \perp X_2 | Y$ .
- $\epsilon_{CI}$  can be viewed as quantification of extra shared features between  $X_1$  and  $X_2$  are not captured by  $Y$  (spurious feature not reducible by SSL)

$$\text{Test Error} \lesssim \underbrace{\frac{k}{n_L}}_{\text{adaptation error}} + \underbrace{\epsilon_{CI}}_{\text{representation error}}$$

- ERM would need  $n_L \asymp \text{Complexity of Function Class}$ .
- Both terms are tight in  $\frac{k}{n_L} + \epsilon_{CI}$  in some scenarios

## Characterizing Approximate Conditional Independence

Define  $\epsilon_{CI} = \mathbb{E}_{X_1} \|\mathbb{E}[X_2|X_1] - \mathbb{E}_Y[\mathbb{E}[X_2|Y]|X_1]\|^2$ .

- This is 0 if  $X_1 \perp X_2 | Y$ .
- $\epsilon_{CI}$  can be viewed as quantification of extra shared features between  $X_1$  and  $X_2$  are not captured by  $Y$  (spurious feature not reducible by SSL)

$$\text{Test Error} \lesssim \underbrace{\frac{k}{n_L}}_{\text{adaptation error}} + \underbrace{\epsilon_{CI}}_{\text{representation error}}$$

- ERM would need  $n_L \asymp \text{Complexity of Function Class}$ .
- Both terms are tight in  $\frac{k}{n_L} + \epsilon_{CI}$  in some scenarios
- Also applies to similarity-based SSL

## Implications on pretext selection

- Design pretext tasks such that  $X_1$  and  $X_2$  have smaller dependence (given  $Y$ )

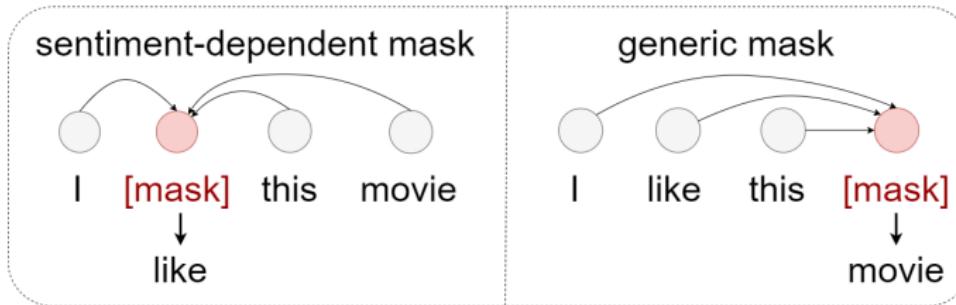
# Empirical Implications

## Implications on pretext selection

- Design pretext tasks such that  $X_1$  and  $X_2$  have smaller dependence (given  $Y$ )

## Applications:

- Image: image classification [He et al. 2021]
- Text: sentiment analysis [Zhang and Hashimoto, 2021]

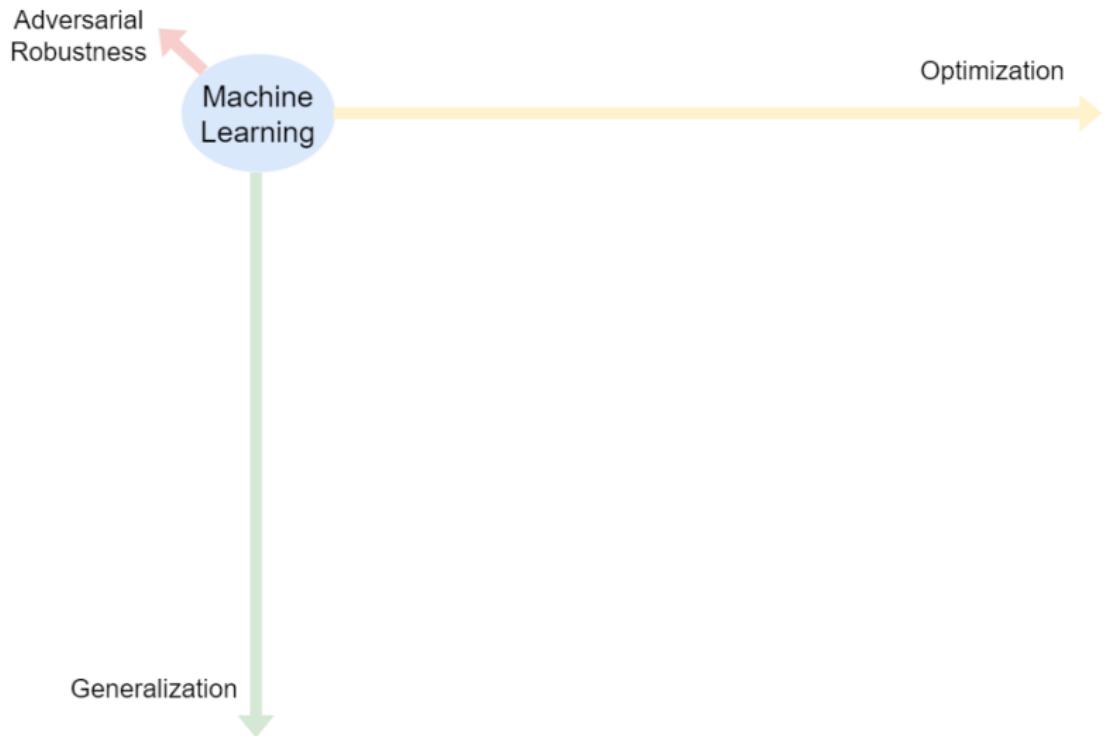


- Audio: speech recognition [Zaiem et al., 2021]

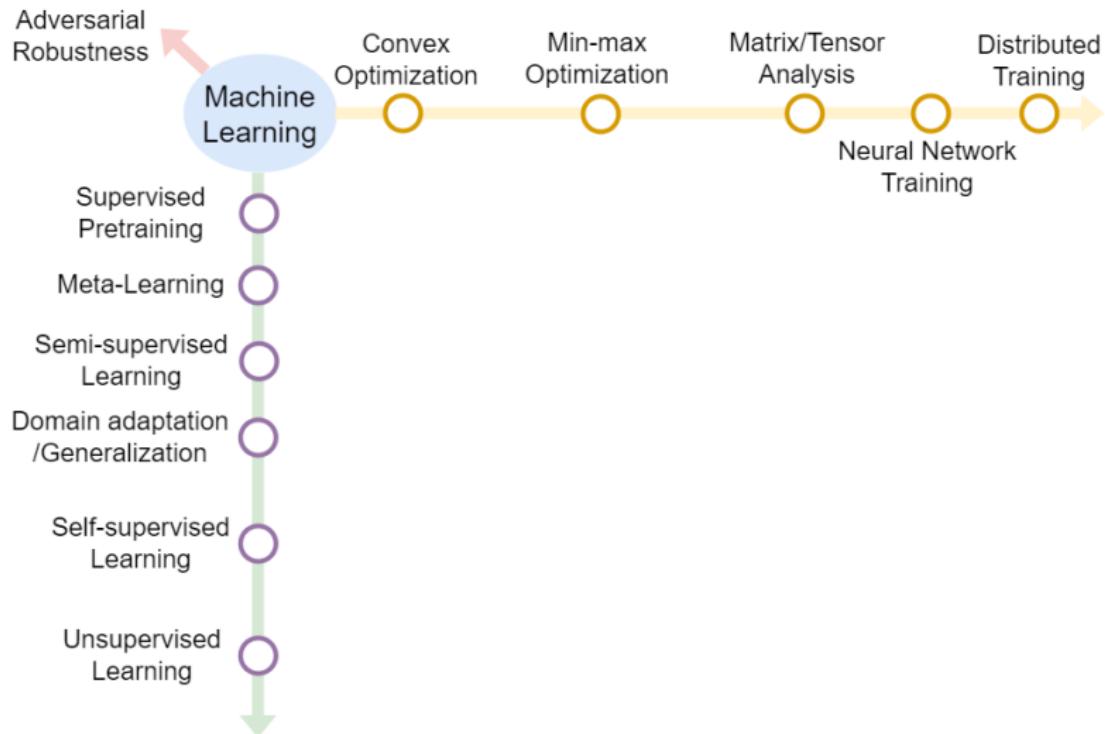
Predicting what you already know helps: Provable self-supervised learning,  
NeurIPS 2021

# wrap-up

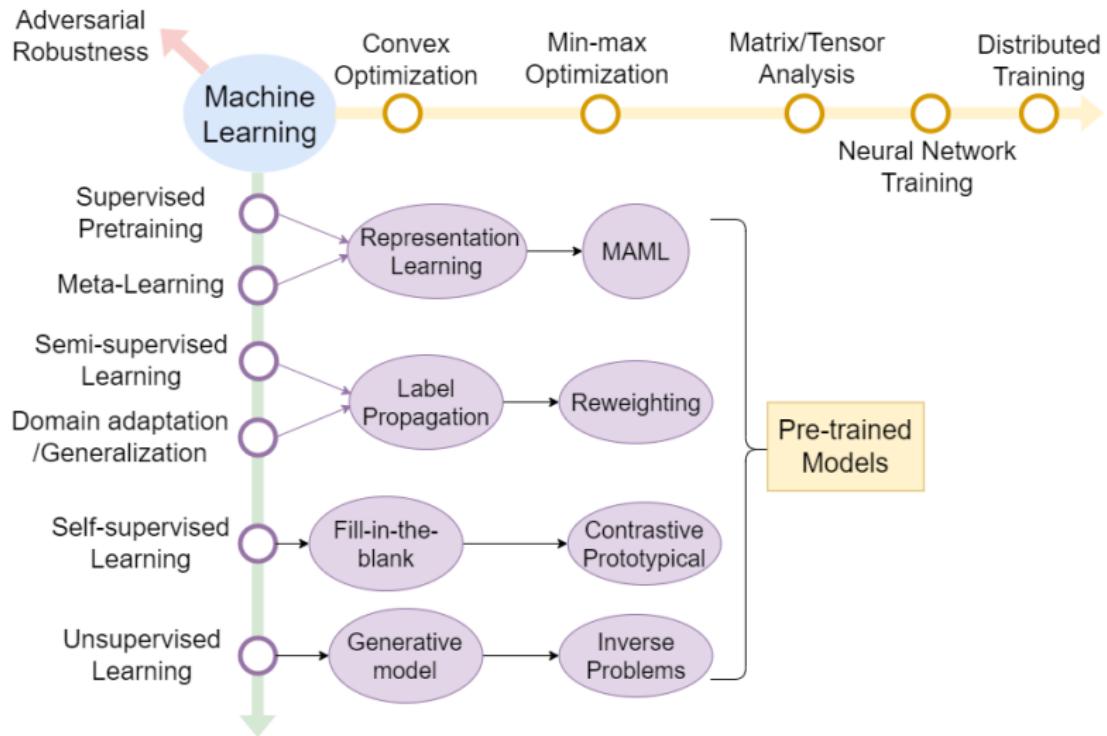
# Overview of My Research Contributions



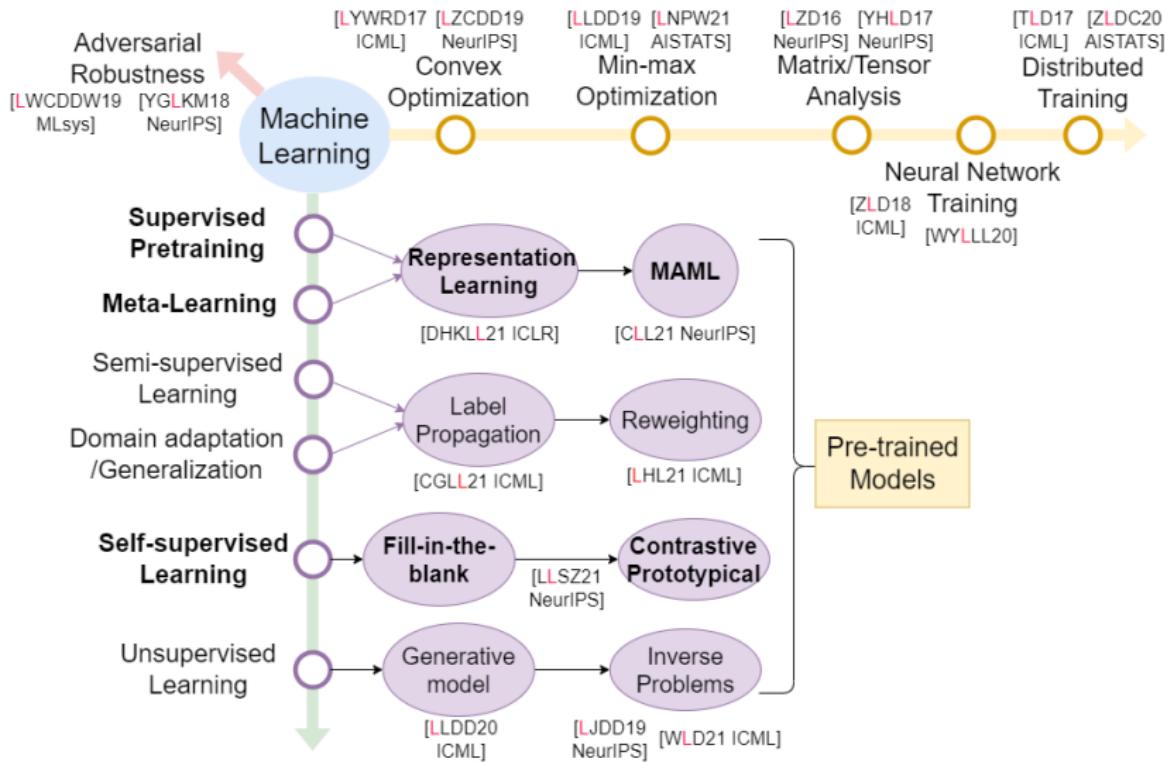
# Overview of My Research Contributions



# Overview of My Research Contributions



# Overview of My Research Contributions





- Our ultimate goal is to expand the capability of AI to broader disciplines
- Pre-trained model is a big step towards that direction
- I will work towards accelerating us to their “ImageNet” moment

# Remaining challenges:



- Understanding the optimization part
- From making predictions to making decisions
- Handle different types of distribution shift
- Training and testing time speed up
- More reliable: security, privacy, AI safety
- Fairness
- Interpretability
- How to evaluate the model without target data
- Other technical details

# Remaining challenges:



- Understanding the optimization part
- From making predictions to making decisions
- Handle different types of distribution shift
- Training and testing time speed up
- More reliable: security, privacy, AI safety
- Fairness
- Interpretability
- How to evaluate the model without target data
- Other technical details

## 1 Meta-learning

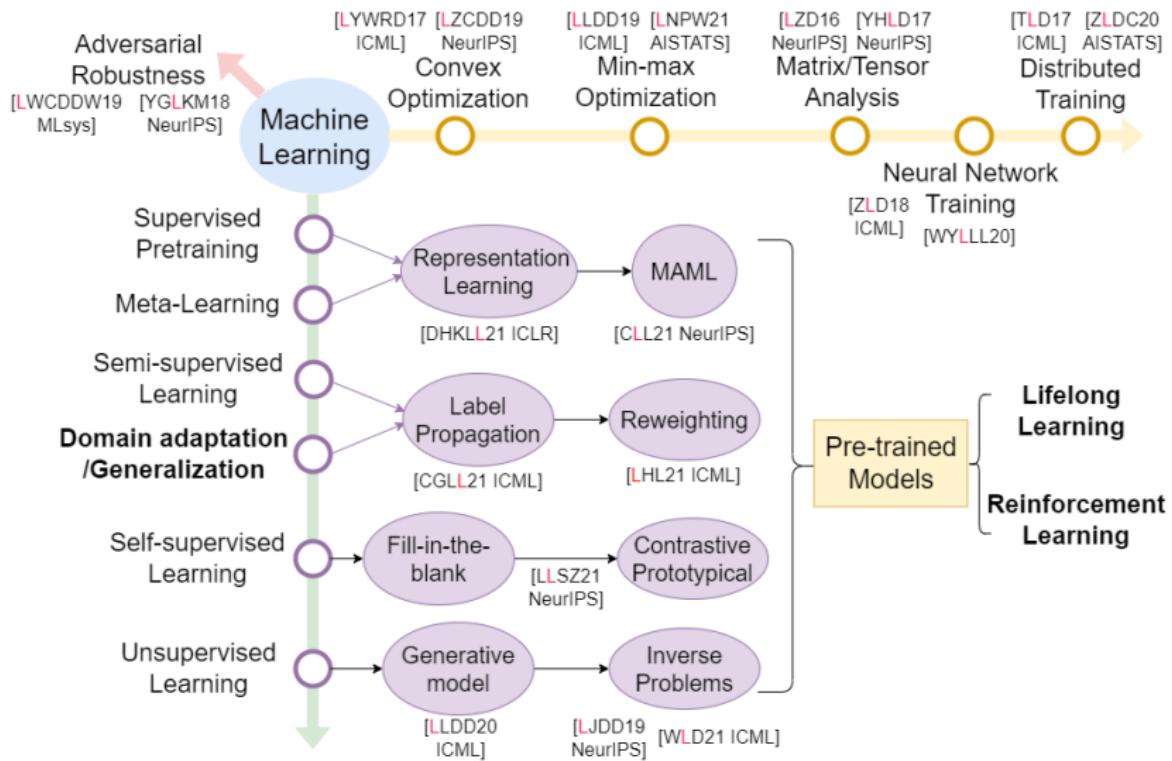
- Meta-Learning with Frozen Representation
- Meta-learning with Fine-tuned Representation

## 2 Self-Supervised Learning

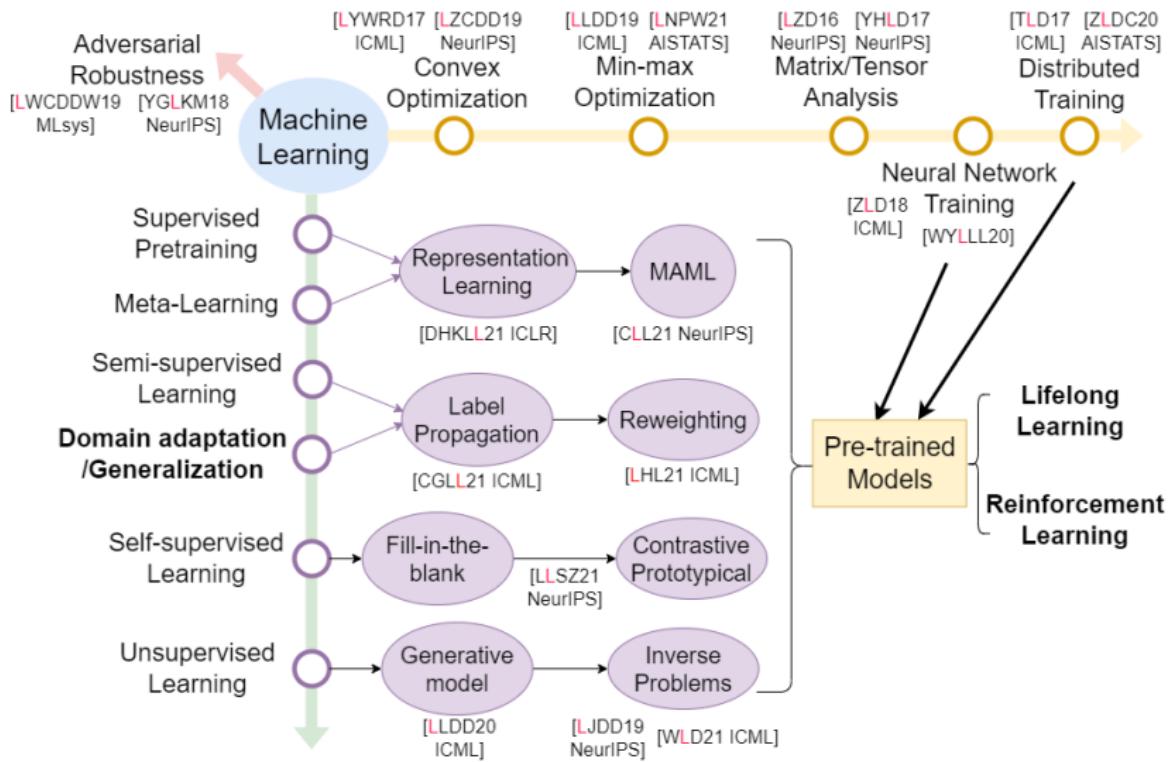
## 3 Ongoing and Future Work

- Domain Adaptation
- Lifelong Learning
- Meta Reinforcement Learning

# Future Work



# Future Work



We investigate sufficient conditions that guarantee:

- Deep networks **learn a good pre-trained model**.
- Good pre-trained models drastically **reduce sample complexity**.
- Pre-trained models can be **learned on unlabeled data**.
- Pre-trained models **transfer to other tasks/domains with covariate shift**.

We investigate sufficient conditions that guarantee:

- Deep networks **learn a good pre-trained model**.
- Good pre-trained models drastically **reduce sample complexity**.
- Pre-trained models can be **learned on unlabeled data**.
- Pre-trained models **transfer to other tasks/domains with covariate shift**.

In the future, I will explore other aspects of pre-trained model:

- How to handle different types of **distribution shift** for domain adaptation?
- How to utilize **latent structure** for meta reinforcement learning?
- How to improve the training algorithms?

# Thank you!

# Thank you!

---

Link back to:

METAREP

FINETUNEDREP

SSL

Back-up slides:

FINETUNEDREP

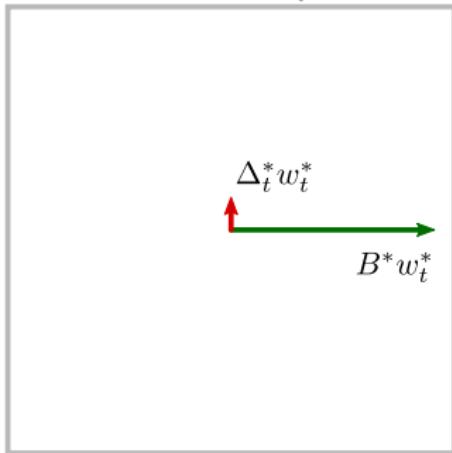
SSL Example

Experiments

METAREP (Fixed feature) has  $\Omega\left(\frac{d}{n_T}\right)$  minimax rate on  
the target task

METAREP chooses representation based on prediction space norm, not parameter space norm!

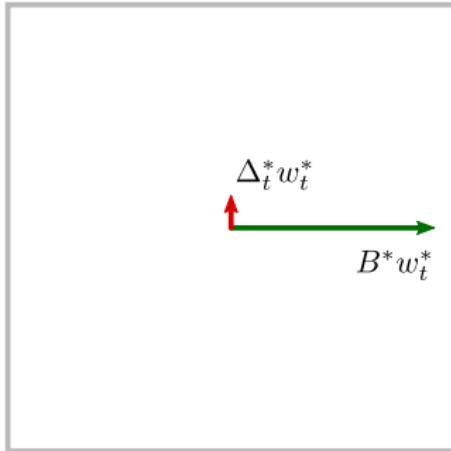
Parameter space ( $L_2$ -norm)



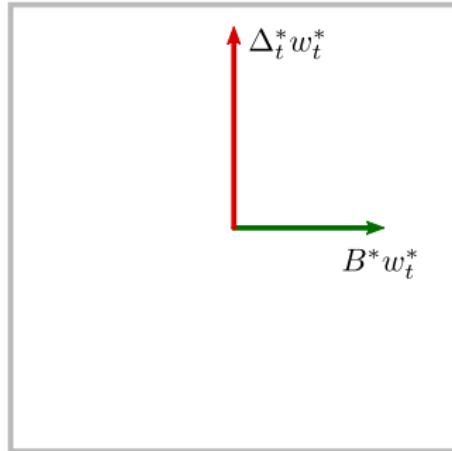
# Failure with Previous Algorithm: METAREP

METAREP chooses representation based on prediction space norm, not parameter space norm!

Parameter space ( $L_2$ -norm)



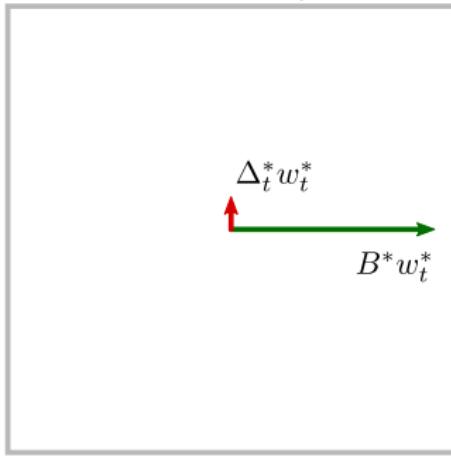
Prediction space ( $\Sigma$ -norm)



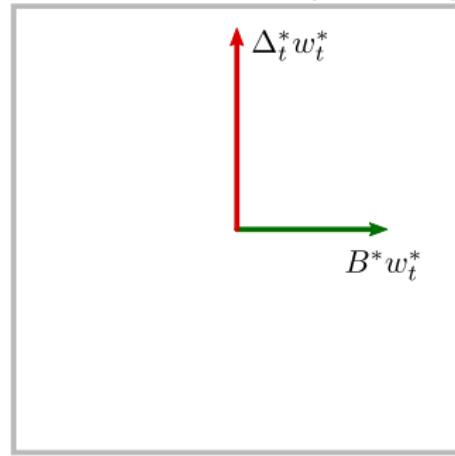
# Failure with Previous Algorithm: METAREP

METAREP *chooses representation based on prediction space norm, not parameter space norm!*

Parameter space ( $L_2$ -norm)



Prediction space ( $\Sigma$ -norm)



Intuition: Trick METAREP into learning subspace of  $\delta_t^*$ 's, and pay cost of having to fine-tune to learn large-norm  $B^*w_{test}^*$ .

Theoretical results:

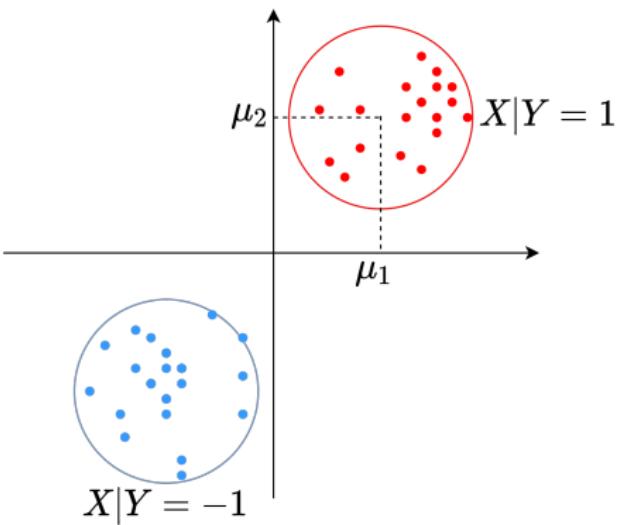
- ① If  $X_1 \perp X_2|Y$ ,

$$f^* = W^* \phi(X_1).$$

- ② Only need  $k$  (dimension of  $Y$ ) samples.

How can  $\phi(X_1)$  be better than  $X_2$  to predict  $Y$ ?

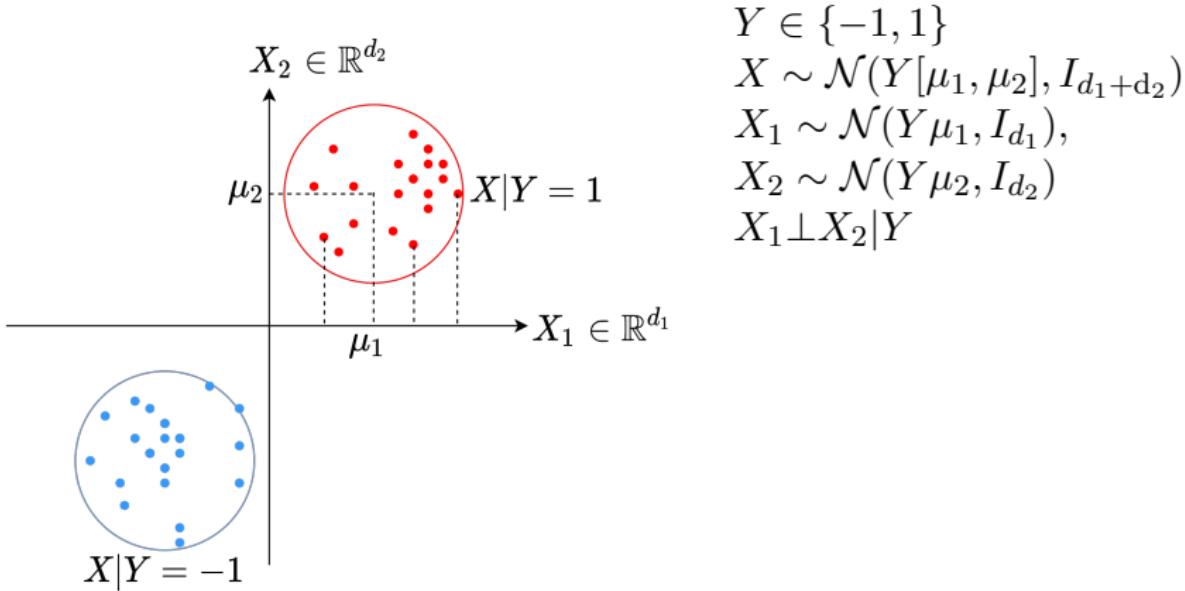
# Example: Gaussian Mixture



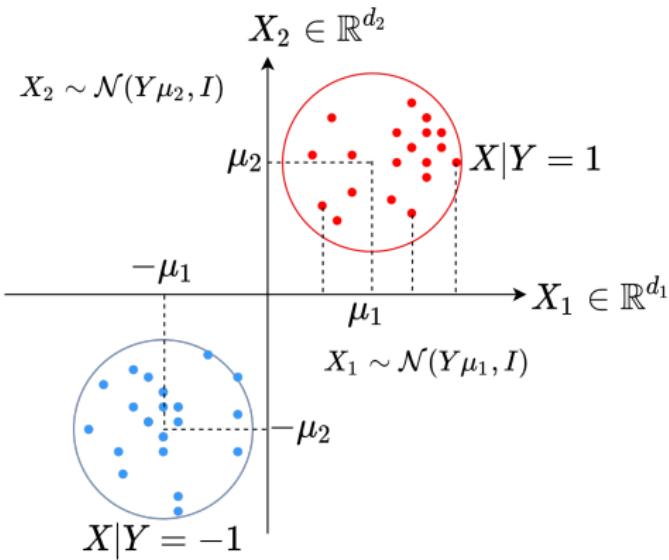
$$Y \in \{-1, 1\}$$

$$X \sim \mathcal{N}(Y[\mu_1, \mu_2], I_{d_1+d_2})$$

# Example: Gaussian Mixture

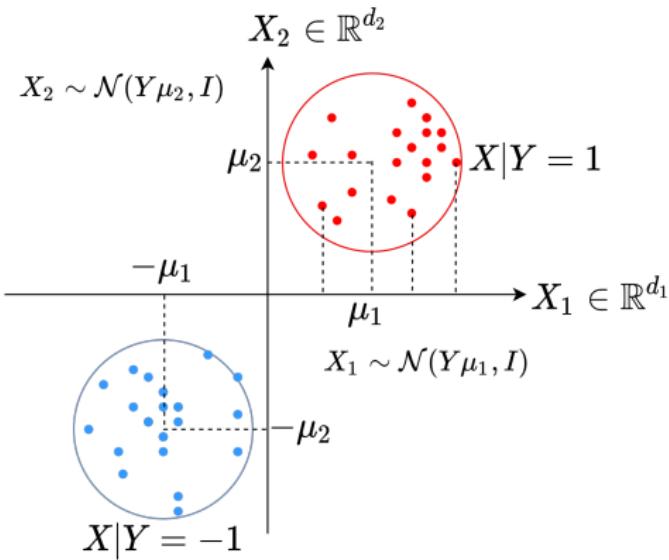


# Example: Gaussian Mixture



$Y \in \{-1, 1\}$   
 $X \sim \mathcal{N}(Y[\mu_1, \mu_2], I_{d_1+d_2})$   
 $X_1 \sim \mathcal{N}(Y\mu_1, I_{d_1})$ ,  
 $X_2 \sim \mathcal{N}(Y\mu_2, I_{d_2})$   
 $X_1 \perp X_2 | Y$   
 $\mathbb{E}[Y|X_2]$  is not linear, but  
 $\mathbb{E}[Y|\phi]$  is linear:

# Example: Gaussian Mixture



$Y \in \{-1, 1\}$   
 $X \sim \mathcal{N}(Y[\mu_1, \mu_2], I_{d_1+d_2})$   
 $X_1 \sim \mathcal{N}(Y\mu_1, I_{d_1}),$   
 $X_2 \sim \mathcal{N}(Y\mu_2, I_{d_2})$   
 $X_1 \perp X_2 | Y$   
 $\mathbb{E}[Y|X_2]$  is not linear, but  
 $\mathbb{E}[Y|\phi]$  is linear:

- $\phi(x_1) = \mathbb{E}[X_2 | X_1 = x_1] = p_1(x_1)\mu_2 + p_{-1}(x_1)(-\mu_2)$
- $\mathbb{E}[Y|X_1] = p_1(x_1) - p_{-1}(x_1) = \mu_2^\top \phi(X_1) / \|\mu_2\|^2.$
- $p_y(x_1) := P(Y = y | X_1 = x_1)$

# Connection to SimSiam method

- Before, we learn  $\phi(x_1) = \mathbb{E}[X_2|X_1 = x_1]$ , which naturally requires  $X_2$  and  $Y$  to be linearly correlated
- We can actually predict any  $g(X_2)|X_1$ , or even  $p(X_2|X_1)$

## ACE and nonlinear CCA

- Alternating conditional expectation (ACE):

$$\min_{\phi, \eta} L_{ACE}(\phi, \eta) = \mathbb{E}_{X_1, X_2} \left[ \|\phi(X_1) - \eta(X_2)\|^2 \right],$$

s.t.  $\Sigma_{\phi, \phi} = \Sigma_{\eta, \eta} = I_k.$

- This is equivalent to the following canonical correlation analysis (CCA):

$$\max_{\phi, \eta} L_{CCA}(\phi, \eta) = \mathbb{E}_{X_1, X_2} \left[ \phi(X_1)^\top \eta(X_2) \right],$$

s.t.  $\Sigma_{\phi, \phi} = \Sigma_{\eta, \eta} = I_k.$

Algorithm (SimSiam):

$$\max_{\phi, \eta, \text{normalized}} \mathbb{E}[\phi(X_1)^\top \eta(X_2)]$$

New measure of conditional independence:

$$\epsilon_{CI} := \max_{\|g\|_{L^2(X_2)} = 1} \mathbb{E}_{X_1} (\mathbb{E}[g(X_2)|X_1] - \mathbb{E}[\mathbb{E}[g(X_2)|Y]|X_1])^2.$$

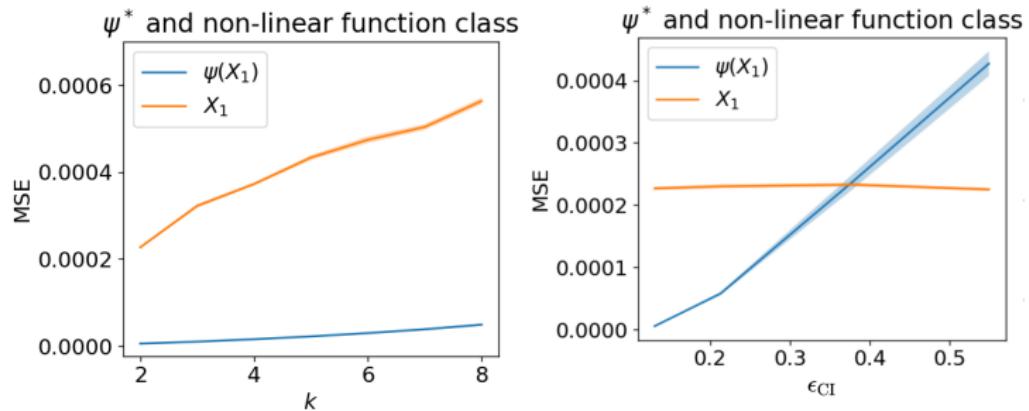
- Extension of previous result:

$$\text{test error} \lesssim \epsilon_{CI} + \frac{k}{n_L}.$$

- If both  $X_2$  and  $X_1$  can well predict  $Y$ , i.e.,  $P_{X_1,Y}(g^*(x_1) \neq y) \leq \alpha$  (same for  $X_2$ ), we have:

$$\text{test error} \lesssim \frac{\alpha}{1-\epsilon_{CI}} + \frac{k}{n_L}.$$

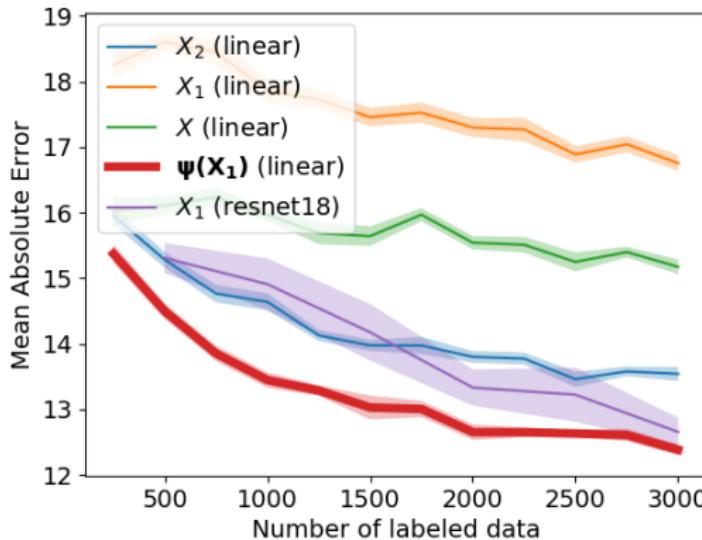
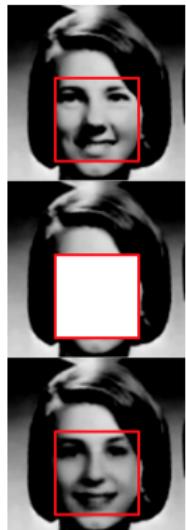
# Simulations: Both Terms Tight in $\frac{k}{n_L} + \epsilon_{\text{CI}}$



**Left:** Class Conditional Gaussian  $X \sim \mathcal{N}(\mu_Y, I)$ ,  $\mu_Y \in \mathbb{R}^{90}$ ,  $Y \in \{1, 2, \dots, k\}$ ,  $X_1 = X_{1:50}$ ,  $X_2 = X_{51:90}$ .  $X_1 \perp X_2 | Y$

**Right:** Similar mixture of Gaussian:  $X \sim \mathcal{N}(\mu_Y, \Sigma_{\epsilon_{\text{CI}}})$ ,  $\alpha \propto \epsilon_{\text{CI}}$  controls the dependence of  $X_1$  and  $X_2$ :  $\epsilon_{\text{CI}} = 0 \Rightarrow$  exact CI, and  $\epsilon_{\text{CI}} = 1 \Rightarrow X_2$  fully depends on  $X_1$ .

# Experiments



- Yearbook: portraits date from 1905 to 2013.

# Ongoing Work: Distribution Shift



Entity30 - Passerine

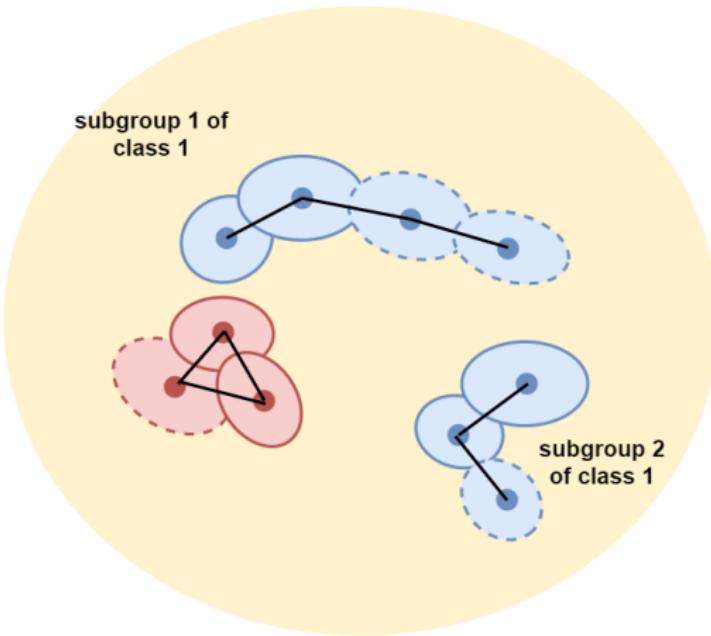
Entity30 - Tableware

# Our New Framework: Subpopulation Shift

	Class 1	Class -1
Source		
Target		



Components connected  
through data augmentation

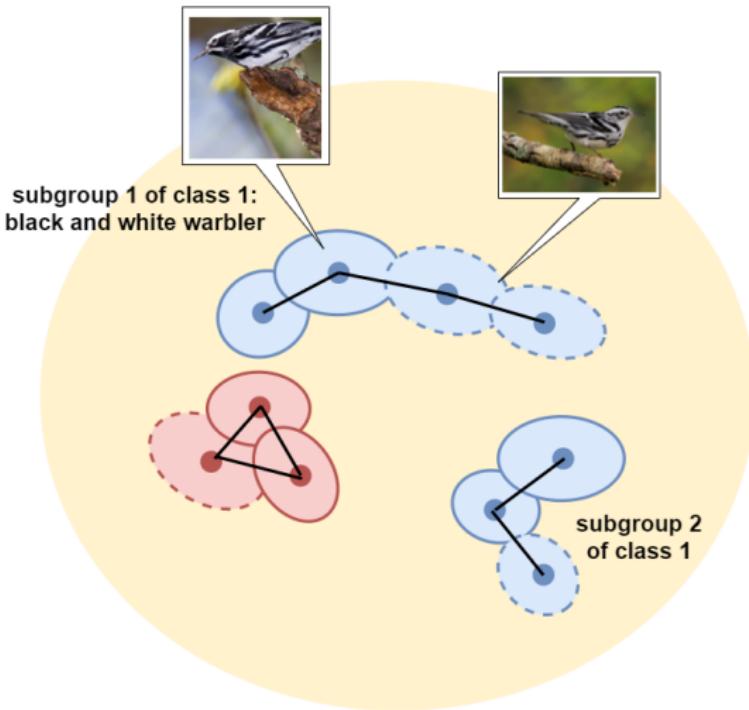


# Our New Framework: Subpopulation Shift

	Class 1	Class -1
Source		
Target		

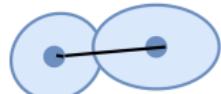


Components connected  
through data augmentation

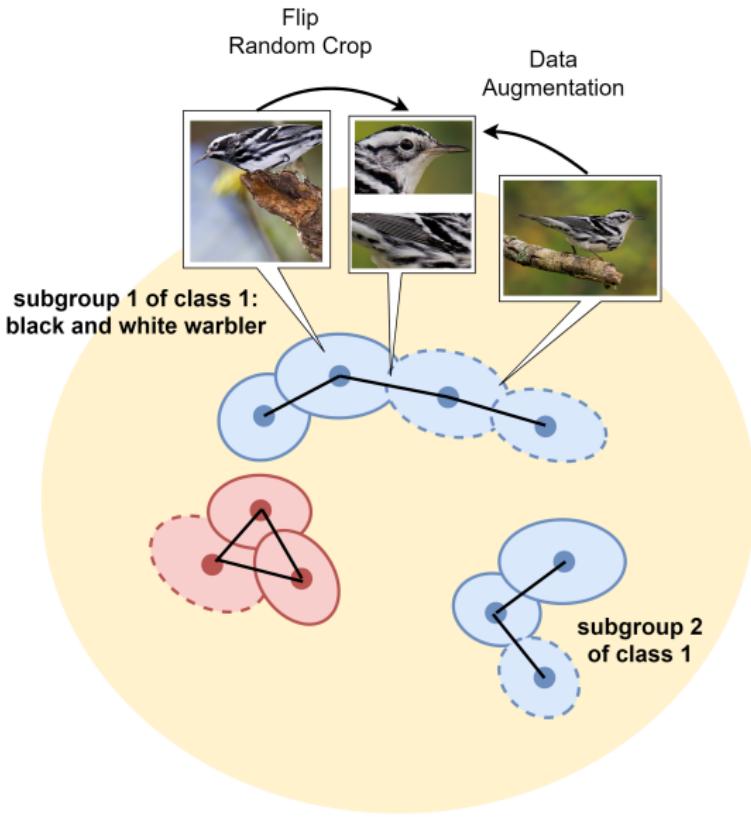


# Our New Framework: Subpopulation Shift

	Class 1	Class -1
Source		
Target		



Components connected  
through data augmentation



# Experiments

Method	A → W	D → W	W → D	A → D	D → A	W → A	Average
MDD	94.97±0.70	98.78±0.07	100±0	92.77±0.72	75.64±1.53	72.82±0.52	89.16
Ours	95.47±0.95	98.32±0.19	100±0	93.71±0.23	76.64±1.91	74.93±1.15	<b>89.84</b>

Performance of MDD<sup>1</sup> and our method on Office-31 dataset.

Method	Ar → Cl	Ar → Pr	Ar → Rw	Cl → Ar	Cl → Pr	Cl → Rw	Pr → Ar
MDD	54.9±0.7	74.0±0.3	77.7±0.3	60.6±0.4	70.9±0.7	72.1±0.6	60.7±0.8
Ours	55.1±0.9	74.7±0.8	78.7±0.5	63.2±1.3	74.1±1.8	75.3±0.1	63.0±0.6
Method	Pr → Cl	Pr → Rw	Rw → Ar	Rw → Cl	Rw → Pr	Average	
MDD	53.0±1.0	78.0±0.2	71.8±0.4	59.6±0.4	82.9±0.3	68.0	
Ours	53.0±0.6	80.8±0.4	73.4±0.1	59.4±0.7	84.0±0.5	<b>69.6</b>	

Performance of MDD and our method on Office-Home dataset.

<sup>1</sup>MDD: (Zhang et al. 2019)

# Experiments: Subpopulation Shift Dataset

- ENTITY-30 task from BREEDS tasks.
- We use FixMatch, an existing consistency regularization method. We also leverage SwAV, an existing unsupervised representation learned from ImageNet, where there can be a better structure of subpopulation shift. We compare with popular distribution matching methods like DANN and MDD.

Method	Source Acc	Target Acc
Train on Source	$91.91 \pm 0.23$	$56.73 \pm 0.32$
DANN (Ganin et al., 2016)	$92.81 \pm 0.50$	$61.03 \pm 4.63$
MDD (Zhang et al., 2019)	$92.67 \pm 0.54$	$63.95 \pm 0.28$
FixMatch (Sohn et al., 2020)	$90.87 \pm 0.15$	$72.60 \pm 0.51$