CMPT 983 Fall 2022
Transfer Learning

Martin Ester

Simon Fraser University

Motivation

- Do not have enough data from the target domain to train a model.
- Have a machine learning model for a source domain.
- Applying it as is to the target domain does not work, if the IID (independent and identically distributed) assumption is violated.
- Transfer learning relaxes the assumption.
- Source and target domain need to have same input and output and need to be semantically related.

Motivation

- What to transfer?
  training examples,
  features,
  model parameters,
  feature extractor, . . .
- How to avoid negative transfer?
  Transfer that leads to worse performance than training a model on the target data from scratch.

Motivation

Example

source domain

Art painting

target domain

Photo

Definitions

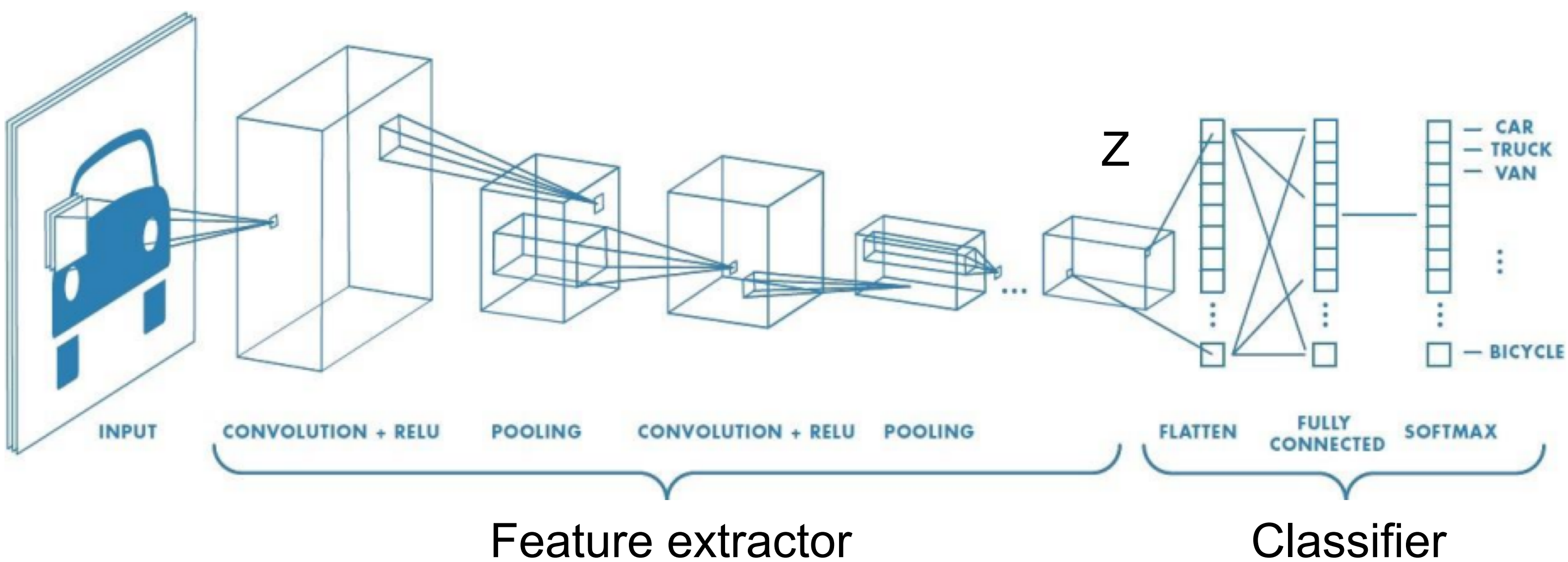
- X: input (covariates, independent variables)
- Y: output (dependent variable, outcome)
- Domain: examples assumed to have been generated from unknown probability distribution
- Source domain: $p(X)$, $p(Y|X)$
- Target domain $q(X)$, $q(Y|X)$
- Covariate shift: $q(x) <> p(x)$ and $q(y|x) = p(y|x)$
  Most common scenario
- But there may also be a shift in $p(y|x)$.

Definitions

- Three scenarios
- Some labeled target examples
  pre-training and fine-tuning
- No labeled target examples
  domain adaptation
- No target examples (but multiple source domains)
  domain generalization

Definitions

- We focus on transfer learning for DNNs.
- DNN learns latent representation Z, which is lower-dimensional, captures the essence of the input and is used to predict the class Y.



Pre-training and fine-tuning

- Train (pre-train) a source DNN and then copy its first n layers to the first n layers of a target DNN.
- The remaining layers of the target network are then randomly initialized and trained using the target data.
- Fine-tuning the copied layers:
  adjust their parameters performing some epochs of backpropagation using the target data.
- Frozen copied layers:
  their parameters do not change during training on the target task.

Pre-training and fine-tuning

- The choice between fine-tuning and freezing depends on the size of the target dataset and the number of parameters in the first n layers.
- If the target dataset is small and the number of parameters is large, fine-tuning may result in overfitting.
- If the target dataset is large or the number of parameters is small, then the source parameters can be fine-tuned to the new task to improve performance.

Pre-training and fine-tuning

- Want to transfer the first n layers of the feature extractor from source to target domain.
- How to determine the number n?
- Features of the first layers tend to be general, higher-layer features more specific.
- Transfer tends to work if the features are general, meaning suitable to both source and target tasks, instead of specific to the source task.

Pre-training and fine-tuning

[Yosinski et al 2014]

- Can we quantify the degree to which a particular layer is general or specific?
  Degree of generality of set of features learned on task A: the higher the lower the loss to which the features can be used for another task B.

→ This definition depends on the similarity between A and B.

- Does the transition occur suddenly at a single layer, or is it spread out over several layers?

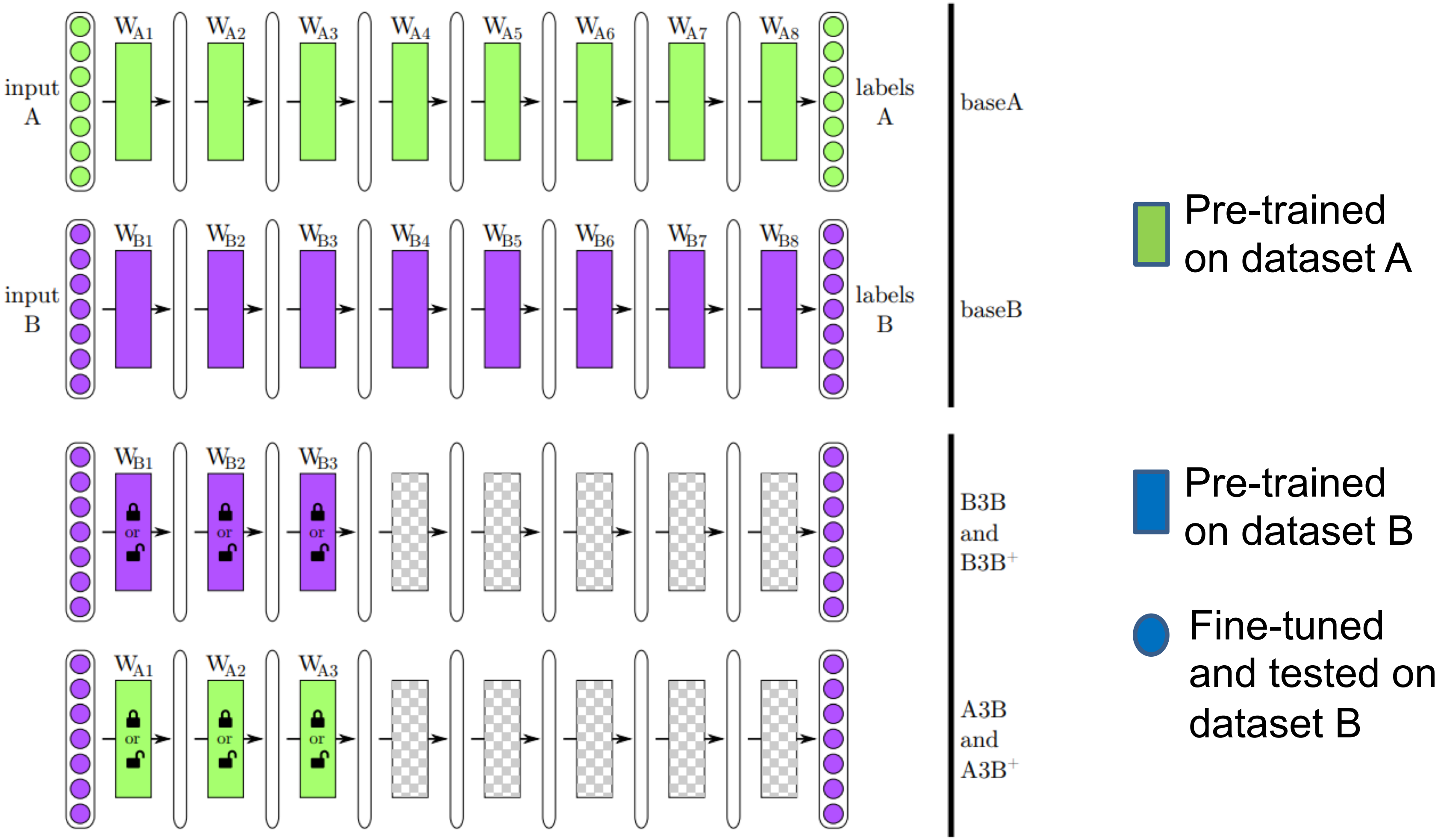
Experimental Design

- Simulating similar domains.
- Using the ImageNet dataset.
- Randomly assign half of the 1000 classes to A and half to B.
- ImageNet contains clusters of similar classes, particularly dogs and cats. Thus A and B are similar when created by randomly assigning classes to each.

Experimental Design

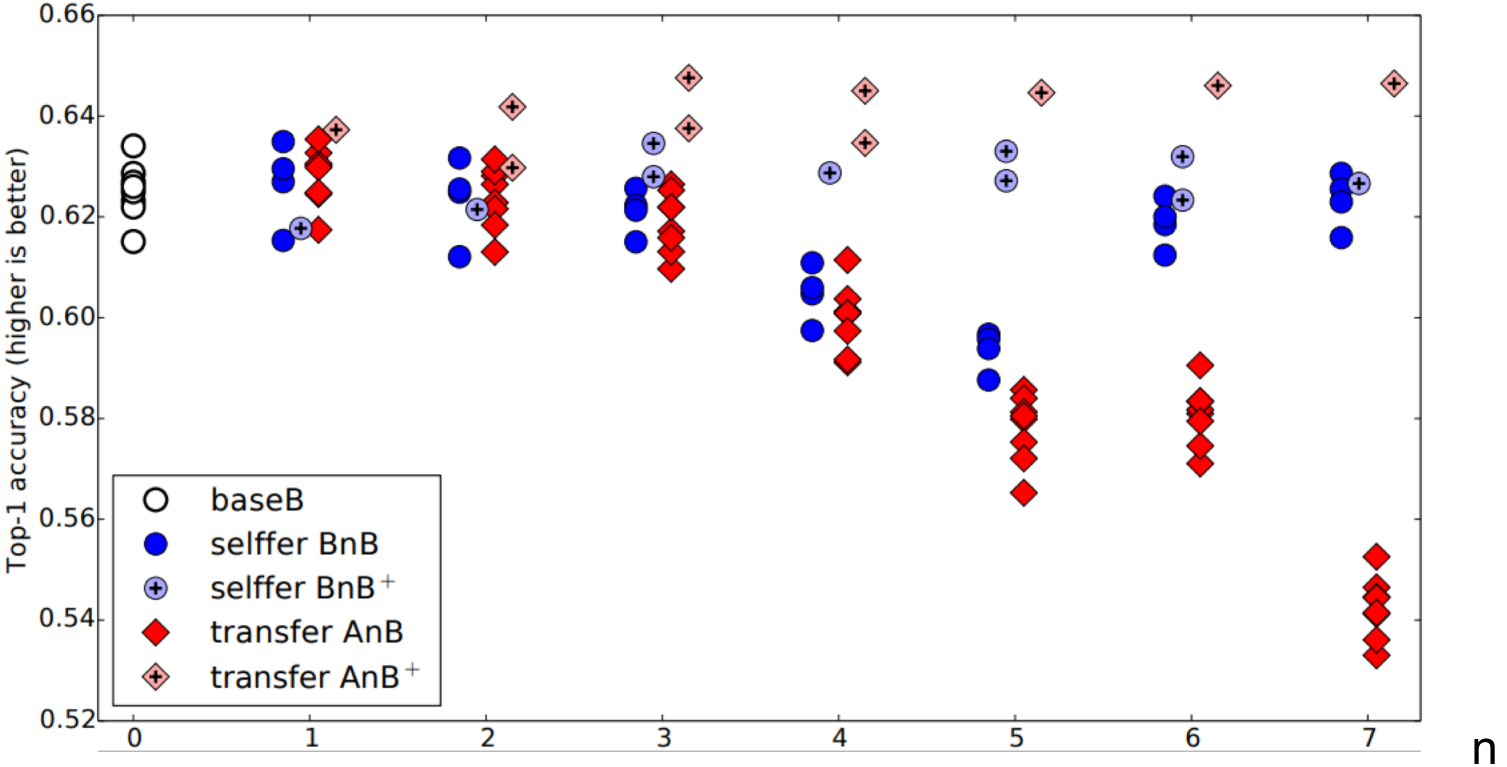
- Simulating different domains.
- Using again the ImageNet dataset.
- ImageNet comes with a hierarchy of classes.
- Create two datasets that are as dissimilar as possible dataset A containing only man-made entities and B containing natural entities.

Experimental Design



Experimental Results

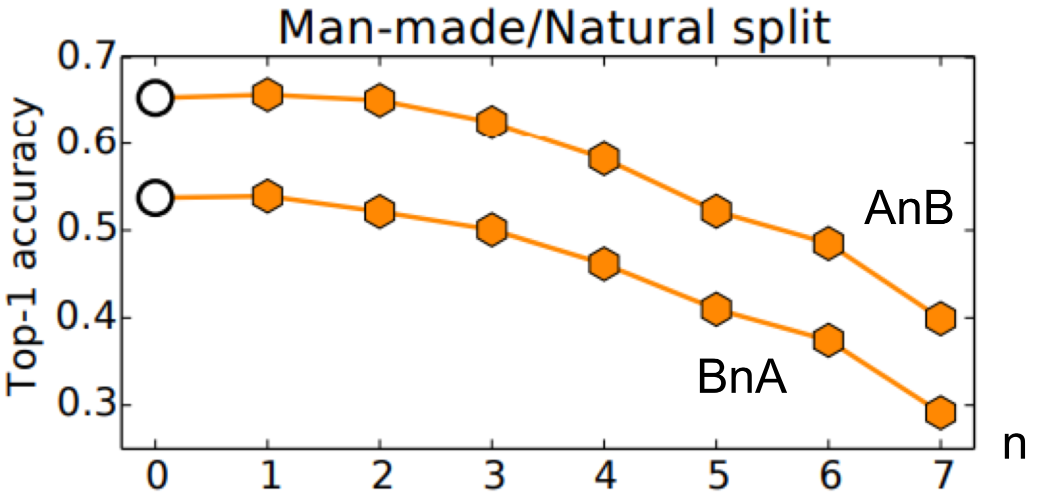
On similar datasets A and B



First layers are more general / transferrable
Selffer loses accuracy due to co-adapted layers

Experimental Results

On dissimilar datasets A and B



First layers are more general / transferrable
Transferring from A to B is easier than from B to A
Transferring to a dissimilar dataset is harder than transferring to a similar dataset

Domain Adaptation

[Ben-David et al 2010]

- Assume that we have plentiful labeled training data from a source distribution but no labeled training data drawn from the target distribution.
- Assume a binary classification problem.
- Under what conditions on the source and target distributions can we expect to learn well?
- Domain
  distribution D on inputs X and a labeling function (ground truth) f : X →{0, 1}
- source domain <DS,fS>, target domain <DT,fT>

Domain Adaptation

- Hypothesis (classifier) is a function h : X →{0, 1}
- Source classification error

$$\epsilon_S(h, f) = \mathrm{E}_{\mathbf{x}\sim\mathcal{D}_S}\left[|h(\mathbf{x}) - f(\mathbf{x})|\right]$$

- Theorem: Bound for the target classification error

  For any hypothesis h:

$$\epsilon_T(h) \leq \epsilon_S(h) + d_1(\mathcal{D}_S, \mathcal{D}_T)$$
$$+ \min\left\{\mathrm{E}_{\mathcal{D}_S}\left[|f_S(\mathbf{x}) - f_T(\mathbf{x})|\right], \mathrm{E}_{\mathcal{D}_T}\left[|f_S(\mathbf{x}) - f_T(\mathbf{x})|\right]\right\}.$$

Domain Adaptation

- The third term (difference of the labeling functions) is assumed to be small.
  P(Y|X) is the same in both domains
- $$d_1(\mathcal{D}, \mathcal{D}') = 2 \sup_{B \in \mathcal{B}} \left| \Pr_{\mathcal{D}}[B] - \Pr_{\mathcal{D}'}[B] \right|$$
  is the L1 (variation) divergence.
  $\mathcal{B}$ is the set of measurable subsets under D and D'.
- The L1 divergence cannot be accurately estimated from finite samples of arbitrary distributions.

Domain Adaptation

- Use the H divergence instead

$$d_{\mathcal{H}}(\mathcal{D}_{\mathrm{S}}^{\mathcal{X}}, \mathcal{D}_{\mathrm{T}}^{\mathcal{X}}) \stackrel{\text{def}}{=} 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x}^s \sim \mathcal{D}_{\mathrm{S}}^{\mathcal{X}}} \left[\eta(\mathbf{x}^s) = 1\right] - \Pr_{\mathbf{x}^t \sim \mathcal{D}_{\mathrm{T}}^{\mathcal{X}}} \left[\eta(\mathbf{x}^t) = 1\right] \right|$$

- It measures the capacity of the hypothesis class H to distinguish between examples from the two different distributions.
- Approximate it through the empirical H divergence

$$\hat{d}_{\mathcal{H}}(S, T) \stackrel{\text{def}}{=} 2\left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^{m} I[\eta(\mathbf{x}_i^s) = 1] + \frac{1}{m'} \sum_{i=1}^{m'} I[\eta(\mathbf{x}_i^t) = 0]\right]\right)$$

where *I*[a] is the indicator function which is 1 if predicate a is true, and 0 otherwise.

Domain Adaptation

- In domain adaptation, need to minimize the first two terms, i.e. source classification error and difference between the distribution of the domains.
- Ds and Dt are given and assumed to be different, i.e. P(X) is different in both domains.
- Learn a latent representation Z of X that has similar distribution in both domains.
- Need a practical measure of the difference of two distributions.

Domain-Adversarial Neural Networks [Ajakan 2014]

- Empirical H divergence in the representation space

$$\hat{d}_{\mathcal{H}}\big(\mathbf{h}(S), \mathbf{h}(T)\big) \; = \\ 2\left(1 - \min_{\eta\in\mathcal{H}}\left[\frac{1}{m}\sum_{i=1}^{m} I\left[\eta(\mathbf{h}(\mathbf{x}_i^s)) = 1\right] + \frac{1}{m'}\sum_{i=1}^{m'} I\left[\eta(\mathbf{h}(\mathbf{x}_i^t)) = 0\right]\right]\right)$$

- Consider H as the class of hyperplanes in the representation space.
- Use logistic regression model:

$$o(\boldsymbol{\phi}) \stackrel{\text{def}}{=} \operatorname{sigm}(d + \mathbf{u}^\top \boldsymbol{\phi})$$

Domain-Adversarial Neural Networks

- We replace the minimization part of the empirical H divergence by the following:

$$\max_{\mathbf{u},d}\left(-\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^d\big(o(\mathbf{x}_i^s),1\big)-\frac{1}{m'}\sum_{i=1}^{m'}\mathcal{L}^d\big(o(\mathbf{x}_i^t),0\big)\right)$$

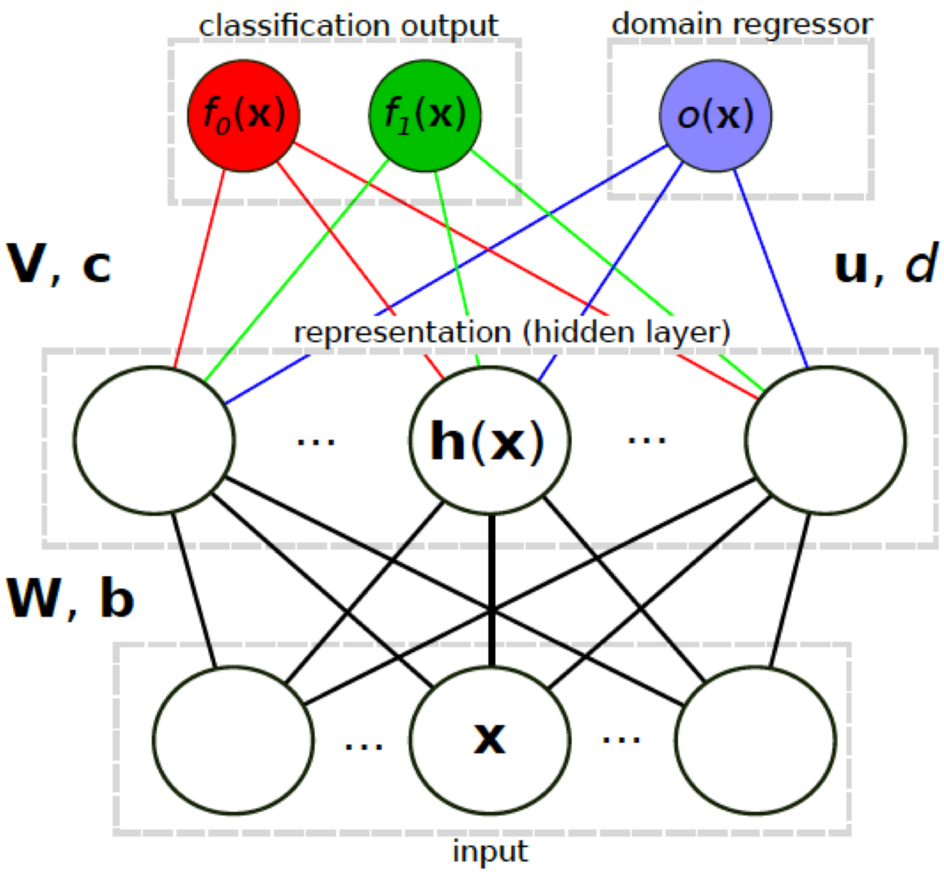
- We obtain the following overall loss function:

$$\min_{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}}\left[\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\big(\mathbf{f}(\mathbf{x}_i^s),y_i^s\big)\right.$$
$$\left.+\lambda\max_{\mathbf{u},d}\left(-\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^d\big(o(\mathbf{x}_i^s),1\big)-\frac{1}{m'}\sum_{i=1}^{m'}\mathcal{L}^d\big(o(\mathbf{x}_i^t),0\big)\right)\right]$$

Domain-Adversarial Neural Networks

SFU

DANN Architecture



Domain-Adversarial Neural Networks

- The neural network and the domain regressor are competing against each other, in an adversarial manner: the regressor wants to distinguish the domains, while the neural network wants to learn a representation that makes that hard.
- Optimization strategy 1: EM-style optimization Alternate between optimizing the adversarial parameters u; d and the other regular neural network parameters W;V; b; c.

Domain-Adversarial Neural Networks

- Optimization strategy 2:
  modified stochastic gradient descent
  Iterate until convergence:
  Sample a pair of source and target example (xsi; xtj) and apply a gradient step update of all parameters.
- The update of the regular parameters follows as usual the opposite direction of the gradient.
- But for the adversarial parameters u; d the step must follow the gradient's direction.

Domain-Adversarial Neural Networks

- Optimization strategy 2:
  modified stochastic gradient descent
  Iterate until convergence:
  Sample a pair of source and target example (xsi; xtj) and apply a gradient step update of all parameters.
- The update of the regular parameters follows as usual the opposite direction of the gradient.
- But for the adversarial parameters u; d the step must follow the gradient's direction.

Universal Domain Adaptation

[You et al. 2019]

- Most publications consider closed domain adaptation (DA), where the sets of classes in the source (Cs) and the target domain (Ct) are identical.
- Universal DA (UDA): assume shared classes as well as private source classes and private target classes.
- Challenge: align only the examples from shared classes, without knowing the classes in the target domain.
- *p*(.,.): probability density distribution of source data
  q(.,.): probability density distribution of target data

Universal Domain Adaptation

SFU

- Source classes $\mathcal{C}_s$
- Target classes $\mathcal{C}_t$
- Common classes $\mathcal{C} = \mathcal{C}_s \cap \mathcal{C}_t$
- Private source classes $\overline{\mathcal{C}}_s = \mathcal{C}_s \setminus \mathcal{C}$
- Private target classes $\overline{\mathcal{C}}_t = \mathcal{C}_t \setminus \mathcal{C}$
- Task of UDA

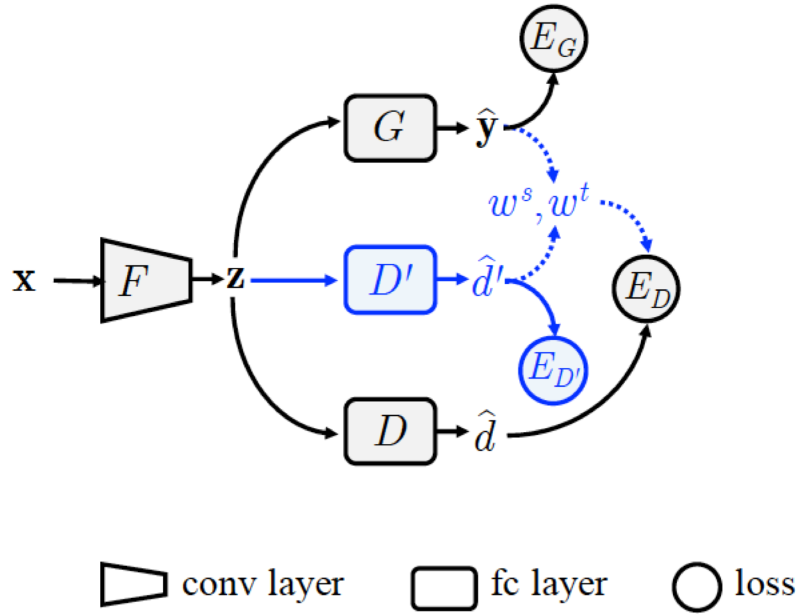
  1) distinguish between target data coming from *C* and target data coming from $\overline{\mathcal{C}}_t$

  2) classify target data from C, i.e. minimize

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim q_{\mathcal{C}}}\left[f(\mathbf{x}) \neq \mathbf{y}\right]$$

Universal Domain Adaptation

Training phase



F: feature extractor
G: classifier

D': non-adversarial domain discriminator
D: adversarial domain discriminator

Universal Domain Adaptation

- Feature extractor F:
  maps source and target examples x to a latent representation z.
- Classifier G
  estimates probability distribution ˆy = G(z) of x over the source classes Cs.
  Classification error $E_G = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} L\left(\mathbf{y}, G(F(\mathbf{x}))\right)$
- Non-adversarial domain discriminator D':
  computes the similarity d' = D'(z) of x to the source domain. Error $$E_{D'} = -\mathbb{E}_{\mathbf{x}\sim p} \log D'\left(F\left(\mathbf{x}\right)\right) - \mathbb{E}_{\mathbf{x}\sim q} \log\left(1 - D'\left(F\left(\mathbf{x}\right)\right)\right)$$

Universal Domain Adaptation

- Adversarial domain discriminator D:
  adversarially aligns the feature distributions of the source and target data from the common classes C.
  Error $$E_D = -\mathbb{E}_{\mathbf{x}\sim p} w^s(\mathbf{x}) \log D(F(\mathbf{x})) - \mathbb{E}_{\mathbf{x}\sim q} w^t(\mathbf{x}) \log (1 - D(F(\mathbf{x})))$$
  where $w^s(\mathbf{x})$ and $w^t(\mathbf{x})$ denote the probability of a source / target sample x belonging to the common label set C.
- The non-adversarial domain discriminator D’ is employed to compute $w^s(\mathbf{x})$ and $w^t(\mathbf{x})$.

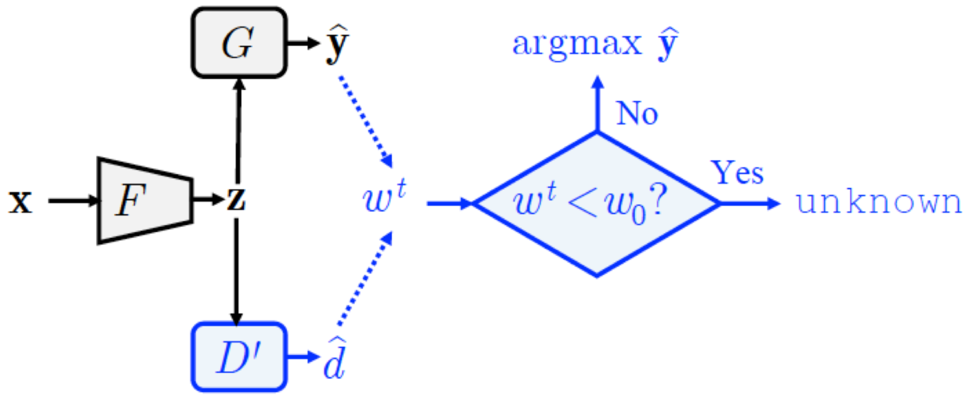
Universal Domain Adaptation

- Optimization through a minimax game

$$\max_{D} \min_{F,G} E_G - \lambda E_D$$

$$\min_{D'} E_{D'}$$

- Testing phase



Universal Domain Adaptation

- How to compute the sample-level transferability criterion $w^s(\mathbf{x})$ and $w^t(\mathbf{x})$?
- Assumptions on source domain similarity *d'* and prediction uncertainty H (entropy)

$$\mathbb{E}_{\mathbf{x}\sim p_{\overline{\mathcal{C}}_s}}\hat{d}' > \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{C}}}\hat{d}' > \mathbb{E}_{\mathbf{x}\sim q_{\mathcal{C}}}\hat{d}' > \mathbb{E}_{\mathbf{x}\sim q_{\overline{\mathcal{C}}_t}}\hat{d}'$$

$$\mathbb{E}_{\mathbf{x}\sim q_{\overline{\mathcal{C}}_t}}H(\hat{\mathbf{y}}) > \mathbb{E}_{\mathbf{x}\sim q_{\mathcal{C}}}H(\hat{\mathbf{y}}) > \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{C}}}H(\hat{\mathbf{y}}) > \mathbb{E}_{\mathbf{x}\sim p_{\overline{\mathcal{C}}_s}}H(\hat{\mathbf{y}})$$

- Definitions $$w^s(\mathbf{x}) = \frac{H(\hat{\mathbf{y}})}{\log|\mathcal{C}_s|} - \hat{d}'(\mathbf{x})$$

$$w^t(\mathbf{x}) = \hat{d}'(\mathbf{x}) - \frac{H(\hat{\mathbf{y}})}{\log|\mathcal{C}_s|}$$

Domain Generalization

- Compared to Domain Adaptation
  No target data
  Data from $K > 1$ (seen) source domains
  Goal is to classify examples from (unseen) target domain
- Basic assumption
  - there exists a feature space shared by all the
    seen source domains and the unseen target domain,
  - which captures information to discriminate the classes.

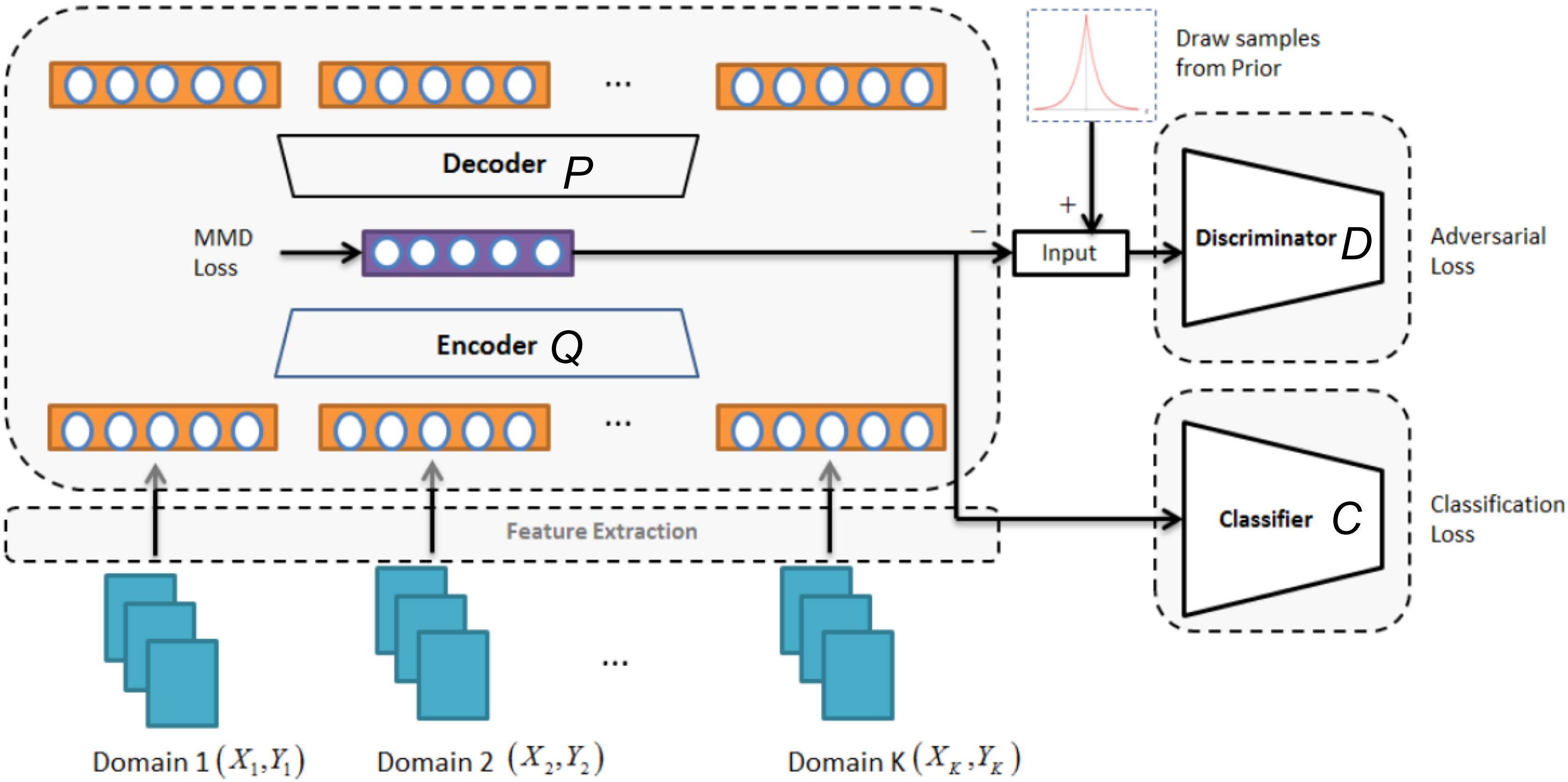
Domain Generalization

- Common approach
  learning a representation via minimizing the difference between the seen source domains
- Challenge
  the learned representation may overfit to the source domains and perform poorly on an unseen target domain.

Domain Generalization with Adversarial Feature Learning [Li et al 2018a]

- Learn a universal representation across domains by
  1) minimizing the MMD difference between the seen source domains, and
  2) matching the distribution of data in the representation space to a prior distribution.
- The method (called MMD-AAE) is based on an adversarial autoencoder (AAE) [23] extended to the multiple domain learning setting.

Domain Generalization with Adversarial Feature Learning



$$\min_{C,Q,P} \max_{D} \mathcal{L}_{\text{err}} + \lambda_0 \mathcal{L}_{\text{ae}} + \lambda_1 \mathcal{R}_{\text{mmd}} + \lambda_2 \mathcal{J}_{\text{gan}}$$

Domain Generalization with Adversarial Feature Learning

- Overall loss

$$\min_{C,Q,P} \max_{D} \mathcal{L}_{\text{err}} + \lambda_0 \mathcal{L}_{\text{ae}} + \lambda_1 \mathcal{R}_{\text{mmd}} + \lambda_2 \mathcal{J}_{\text{gan}}$$

- L_{err}: classification error of C
- L_{ae}: reconstruction error of $P(Q(.))$
- R_{mmd}: variance of the representations across the seen domains
- J_{gan}: discrimination error of D

Domain Generalization with Adversarial Feature Learning

- The encoder *Q* and decoder *P* are shared between all seen domains.
- To avoid overfitting to the seen domains, the representations *h* (= *Q*(*x*)) are matched to a prior distribution p(*h*), using the adversarial discriminator D.

$$\mathcal{J}_{\text{gan}} = \mathbb{E}_{\mathbf{h}\sim p(\mathbf{h})}[\log D(\mathbf{h})] + \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log(1 - D(Q(\mathbf{x})))]$$

- Any prior distribution could be assumed.
- In the experiments, a Laplace distribution was used.

Domain Generalization with Adversarial Feature Learning

- To make the latent representations invariant to the seen source domains, an MMD-based regularization term R_{mmd} is added.

$$\mathrm{MMD}(\mathbf{H}_l, \mathbf{H}_t)^2 = \left\| \frac{1}{n_l} \sum_{i=1}^{n_l} \phi(\mathbf{h}_{l_i}) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\mathbf{h}_{t_j}) \right\|$$

where Φ is a given kernel function and Hi is the distribution of representations in domain *i*

MMD = Maximum Mean Discrepancy

- In the experiments, a mixture kernel was used that averages the RBF kernels with bandwidths = 1, 5, 10.

Domain Generalization with Adversarial Feature Learning

SFU

- Theorem

  $\frac{1}{K^2} \sum_{1 \leq i,j \leq K} \mathrm{MMD}(\mathbf{H}_i, \mathbf{H}_j).$ is an upper bound of the variance of the representation distributions *Hi*

- Definition of R_{mmd}

$$\mathcal{R}_{\mathrm{mmd}}(\mathbf{H}_1, ..., \mathbf{H}_K) = \frac{1}{K^2} \sum_{1 \leq i,j \leq K} \mathrm{MMD}(\mathbf{H}_i, \mathbf{H}_j).$$

Meta-Learning for Domain Generalization [Li et al. 2018 b]

SFU

- Train a base learner on a set of source domains by synthesising virtual training and virtual testing domains within each minibatch.
- Minimize the loss on the training domains, while ensuring that the direction taken to achieve this also leads to an improvement of the (virtual) testing loss.
- Algorithm MLDG

Meta-Learning for Domain Generalization

SFU

- At each learning iteration the original S source domains S are split into S–V meta-train domains S¯ and V (virtual) meta-test domains S˘.
- Meta training

  The model parameters Θ are updated (obtaining Θ') on all the S –V meta-train domains S¯ in aggregate, using the loss function

$$\mathcal{F}(.) = \frac{1}{S-V}\sum_{i=1}^{S-V}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell_\Theta(\hat{y}_j^{(i)}, y_j^{(i)})$$

where *l*(.) is a classification loss

Meta-Learning for Domain Generalization

SFU

- Meta testing
  Simulates testing on new domains.
  The loss for the adapted parameters Θ' on the meta-test domains is defined as follows:

$$\mathcal{G}(\cdot) = \frac{1}{V}\sum_{i=1}^{V}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell_{\Theta'}(\hat{y}_j^{(i)}, y_j^{(i)})$$

- Overall loss

$$\underset{\Theta}{\operatorname{argmin}} \ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta - \alpha\mathcal{F}'(\Theta))$$

Meta-Learning for Domain Generalization

SFU

- The overall loss can be reformulated as

$$\underset{\Theta}{\operatorname{argmin}} \ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta) - \beta\alpha(\mathcal{G}'(\Theta) \cdot \mathcal{F}'(\Theta))$$

where $(a \cdot b)$ denotes the dot product of two vectors

- Optimize the parameters Θ so that
1) the loss in the meta-training and the meta-testing domains is minimized and that
2) the gradients of the loss in both sets of domains have a similar direction (maximize their dot product).

Meta-Learning for Domain Generalization

SFU

Pseudo-Code

procedure MLDG
  Input: Domains $\mathcal{S}$
  Init: Model parameters Θ. Hyperparameters α, β, γ.
  for ite **in** iterations **do**
    Split: $\bar{\mathcal{S}}$ and $\breve{\mathcal{S}} \leftarrow \mathcal{S}$
    Meta-train: Gradients $\nabla_\Theta = \mathcal{F}'_\Theta(\bar{\mathcal{S}}; \Theta)$
    Updated parameters $\Theta' = \Theta - \alpha \nabla_\Theta$
    Meta-test: Loss is $\mathcal{G}(\breve{\mathcal{S}}; \Theta')$.
    Meta-optimization: Update Θ

$$\Theta = \Theta - \gamma \frac{\partial(\mathcal{F}(\bar{\mathcal{S}}; \Theta) + \beta \mathcal{G}(\breve{\mathcal{S}}; \Theta - \alpha \nabla_\Theta))}{\partial \Theta}$$

  end for
end procedure

Directions for Future Research

- Most domain adaptation methods address a scenario of closed sets of class labels.
- Universal domain adaptation (UDA) is a more practical scenario.
- Existing UDA methods classify all target examples that do not seem to belong to a shared class as “unknown”, but the target domain may have multiple private classes.
- How to discover all the private classes and label all classes accurately?

  [Brbic et al 2020] [Li et al 2021] [Tanwisuth et al 2020]

Directions for Future Research

- Classifiers should not only be accurate but also be calibrated, i.e. the confidence of their prediction should be similar to the accuracy.
- Calibration is crucial in mission-critical applications.
- Many DNNs are over-confident.
- In transfer learning there seems to be a trade-off: the more accurate the model on the target domain, the less calibrated.
- How to transfer such that the resulting model is both accurate and well-calibrated?

[Gong et al. 2021] [Park et al. 2020] [Wang et al. 2020] [Yu et al. 2022]

Directions for Future Research

SFU

- Most predictive models are based on correlations between X and Y.
- But correlations may be spurious and not transfer to a target domain.
- Causal models are believed to transfer better between domains since they model mechanisms of data generation.
- How to use structural causal models (SCMs) to improve domain adaptation and domain generalization?

[Javidian et al 2021] [Lv et al 2022] [Wang et al 2021] [Yang et al 2021]

References

- Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., & Marchand, M. (2014). Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*.
- Ben-David, Shai, et al. "A theory of learning from different domains." *Machine learning* 79.1 (2010): 151-175.
- Brbić, M., Zitnik, M., Wang, S., Pisco, A. O., Altman, R. B., Darmanis, S., & Leskovec, J. (2020). MARS: discovering novel cell types across heterogeneous single-cell experiments. *Nature methods*, *17*(12), 1200-1206.
- Gong, Y., Lin, X., Yao, Y., Dietterich, T. G., Divakaran, A., & Gervasio, M. (2021). Confidence calibration for domain generalization under covariate shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 8958-8967).
- Javidian, M. A., Pandey, O., & Jamshidi, P. (2021). Scalable Causal Domain Adaptation. *arXiv e-prints*, arXiv-2103.

References

- Li, H., Pan, S. J., Wang, S., & Kot, A. C. (2018). Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5400-5409).
- Li, D., Yang, Y., Song, Y. Z., & Hospedales, T. (2018, April). Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1).
- Li, G., Kang, G., Zhu, Y., Wei, Y., & Yang, Y. (2021). Domain consensus clustering for universal domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9757-9766).
- Lv, F., Liang, J., Li, S., Zang, B., Liu, C. H., Wang, Z., & Liu, D. (2022). Causality Inspired Representation Learning for Domain Generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8046-8056).

References

- Park, S., Bastani, O., Weimer, J., & Lee, I. (2020, June). Calibrated prediction with covariate shift via unsupervised domain adaptation. In *International Conference on Artificial Intelligence and Statistics* (pp. 3219-3229). PMLR.
- Tanwisuth, K., Fan, X., Zheng, H., Zhang, S., Zhang, H., Chen, B., & Zhou, M. (2021). A prototype-oriented framework for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, *34*, 17194-17208.
- Wang, X., Long, M., Wang, J., & Jordan, M. (2020). Transferable calibration with lower bias and variance in domain adaptation. *Advances in Neural Information Processing Systems*, *33*, 19212-19223.
- Wang, Y., Liu, F., Chen, Z., Lian, Q., Hu, S., Hao, J., & Wu, Y. C. (2021). Contrastive ACE: domain generalization through alignment of causal mechanisms. *arXiv preprint arXiv:2106.00925*.

References

- Yang, S., Yu, K., Cao, F., Liu, L., Wang, H., & Li, J. (2021). Learning causal representations for robust domain adaptation. *IEEE Transactions on Knowledge and Data Engineering*.
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *Advances in neural information processing systems*, *27*.
- You, K., Long, M., Cao, Z., Wang, J., & Jordan, M. I. (2019). Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2720-2729).
- Yu, Y., Bates, S., Ma, Y., & Jordan, M. I. (2022). Robust Calibration with Multi-domain Temperature Scaling. *arXiv preprint arXiv:2206.02757*.