Transfer Learning and looking for something in **common**

Multi-task learning

Invariant Risk Minimization

Same gradients

Antoine Cornuéjols

AgroParisTech – INRAé MIA Paris

EKINOCS research group

Outline

Notations

1. **Source** domain *S*
   - Source **training data** S_S
   - Source data **distribution** D_S
   - Source **hypothesis** h_S

2. **Target** domain *T*
   - Target **training data** S_T $(|S_T| << |S_S|)$
   - Target **data distribution** D_T
   - Target **hypothesis** h_T

What can we **transfer** from one task to another?

- In the following: a **strong assumption**

There is **something in common** between the source and the target

We will remove this assumption later on

What can we transfer

- What could be in **common**?

1. Looking for a universal **representation** (Multi-task learning)

2. Looking for common **causality relationships** (I.R.M.)

3. Looking for common **search behaviors**

4. Others

Outline

What is Multi-Task learning (MTL)?

- As soon you try to optimize more than one loss function

  - E.g. From someone’s picture, trying to guess both
    - The **gender**
    - The **age**
    - The **emotion**

Why Multi-Task learning (MTL)?

- (IF) The tasks at hand are **not unrelated**
  - E.g. From someone's picture, trying to guess both
    - The **gender**
    - The **age**
    - The **emotion**

- It may help to consider them all together:

  better performance with **less** computing resources
  - E.g. guessing the *gender* may help recognize the *emotion* and vice-versa

Rk: There are links with the LUPI framework

Assumption behind MTL

- The **combined learning** of multiple related tasks **can outperform learning each task in isolation**

- MTL allows for **common information** shared between the tasks to be used in the learning process, which leads to better generalization if the tasks are related

- E.g. Learning to **predict the ratings** for several **different critics** (in different countries) can lead to better performances for **each separate task** (predict the restaurant ratings for a specific critic)

- Learning to **recognize** a **face** and the **expression** (fear, disgust, anger, …)

- **Multi modality learning**: e.g. vision and proprioception

Possible **relations** between tasks

- All functions to be learn are **close** to each other **in some norm**
  - E.g. functions capturing preferences in users' modeling problems

- Tasks that share a **common underlying representation**
  - E.g. in *human vision*, all tasks use the **same set of features** learnt in the first stages of the visual system (e.g. local filters similar to wavelets)
  - Users may also *prefer* different types of things (e.g. books, movies, music) based on the **same set of features** or **score** functions

Question

How do we choose to

model the shared information between the tasks?

- Idea: Some shared underlying constraints
  - E.g. a **low dimensional representation** shared across multiple related tasks
    - By way of a **shared hidden layer** in a neural network
    - By explicitly constraining the **dimensionality of a shared representation**

An approach for the **linear** case: minimizing the distance with a shared weight vector

- T binary classification tasks defined over $\mathcal{X} \times \mathcal{Y}$

$$\mathcal{S} = \{\underbrace{\{(\mathbf{x}_{11}, y_{11}), (\mathbf{x}_{21}, y_{21}), \ldots, (\mathbf{x}_{m1}, y_{m1})\}}_{\text{Task 1}}, \ldots, \underbrace{\{(\mathbf{x}_{1T}, y_{1T}), (\mathbf{x}_{2T}, y_{2T}), \ldots, (\mathbf{x}_{mT}, y_{mT})\}}_{\text{Task T}}\}$$

For each task *j*: $h_j(\mathbf{x}) = \mathbf{w}_j \cdot \mathbf{x}$ **Linear** hypotheses

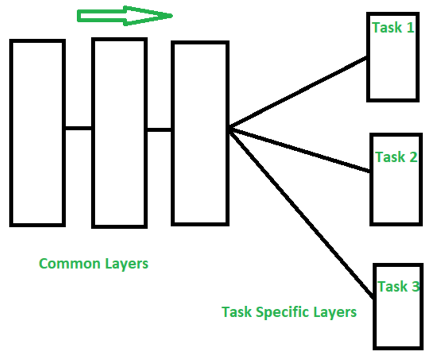
that share a weight vector $\mathbf{w}_j = \boxed{\mathbf{w}_0} + \mathbf{v}_j$

Search:

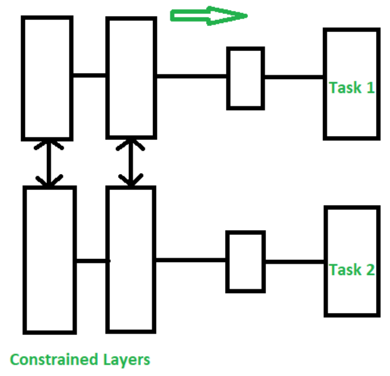
$$h_1^\star, \ldots, h_T^\star = \underset{\mathbf{w}_0, \mathbf{v}_j, \xi_{ij}}{\mathrm{Argmin}} \left\{ \sum_{j=1}^{T} \sum_{i=1}^{m} \xi_{ij} + \frac{\lambda_1}{T} \sum_{j=1}^{T} ||\mathbf{v}_j||^2 + \lambda_2 \, ||\mathbf{w}_0||^2 \right\}$$

MTL with deep neural networks

- Approaches
  1. Sharing **features** (first layers) and have multiple task-specific heads



  1. **Soft**-features or parameters sharing



- Multi-Task Learning induces a bias that prefers hypotheses that can “explain” all tasks
- Beware:
  - Can lead to **worse** performance if the tasks are **unrelated** or **adversarially** related
- Question: *how to measure the **relatedness** of learning tasks*?

...

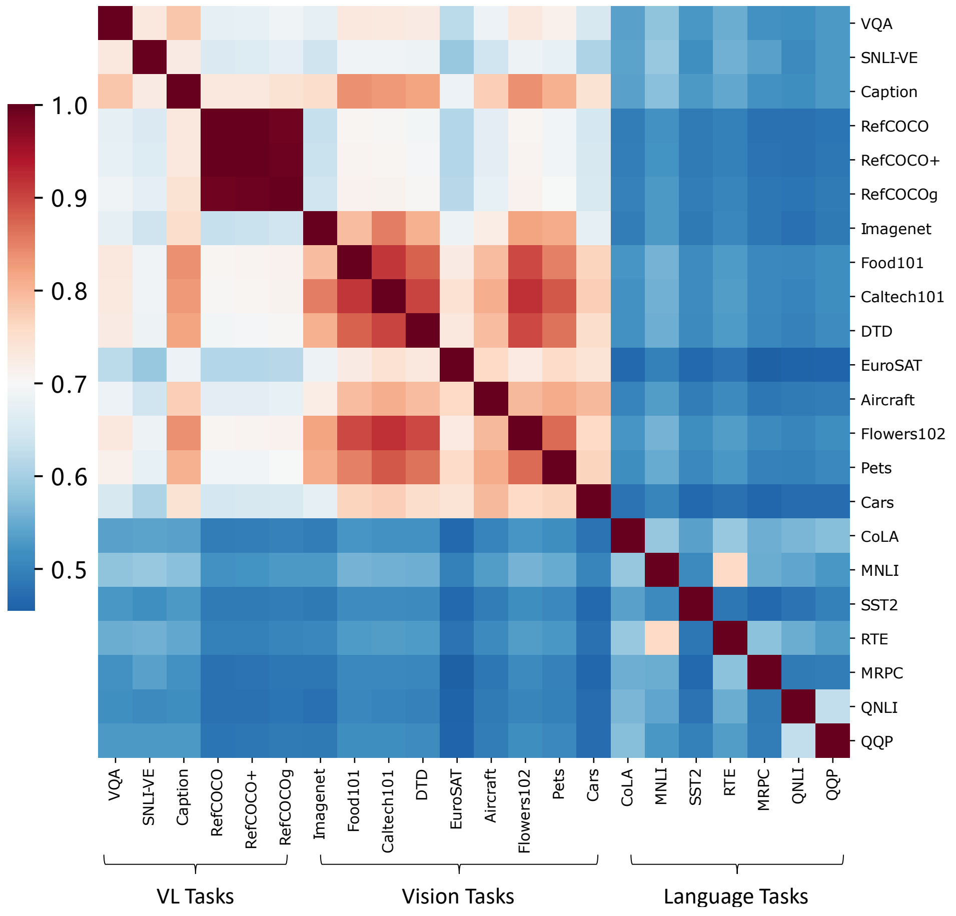
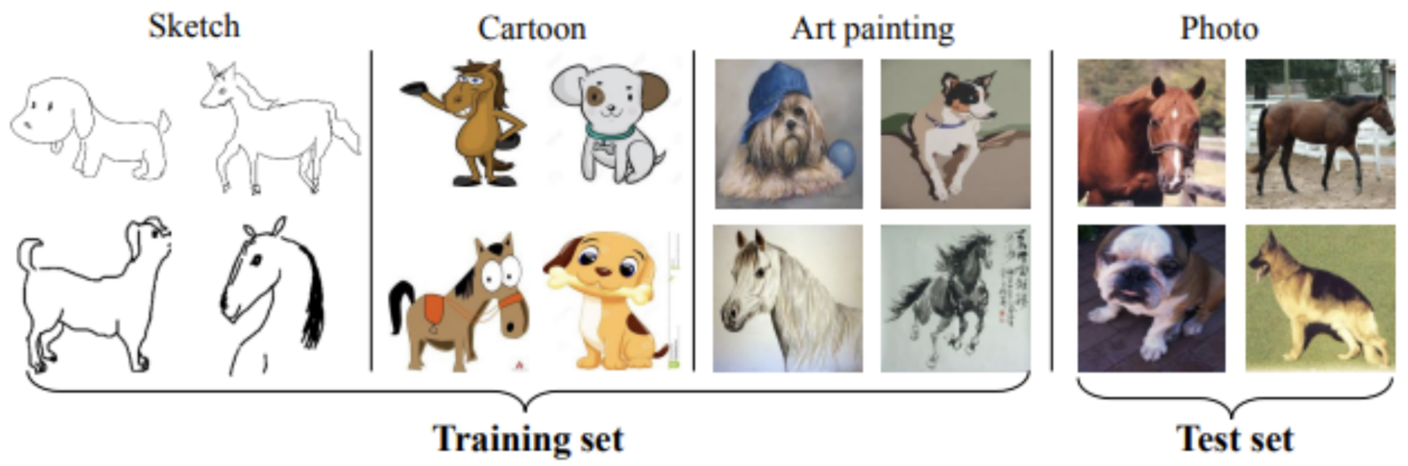


Figure 1. Heatmap of the predicted task similarities, composed of both unimodal and multimodal tasks. Vision-language tasks are more similar to vision tasks compared to language tasks.

WU, Chengyue, WANG, Teng, GE, Yixiao, et al. **$\pi $-Tuning: Transferring Multimodal Foundation Models with Optimal Multi-task Interpolation**. In *International Conf. on Machine Learning* (ICML). PMLR, 2023. p. 37713-37727.

Reminder of the problem of O.O.D. learning



- **distribution shifts** can stem from a variety of factors, including
  - **environmental** changes,
  - **sensor** noise,
  - and image **corruptions**

- O.O.D. learning approaches primarily focus on:
  - Extracting **domain-invariant features** (common representation space)
  - Enriching training data to **promote feature diversity**
    - Data augmentation
    - Adversarial learning
  - Developing specialized **optimization strategies**

Outline

Question

- How can we explicitly distinguish invariant features from spurious ones during training, and subsequently **suppress the influence of spurious features** so that the learned model **generalizes better** to new domains?

- A key aspect of domain generalization is to learn from **features** that remain *invariant* across **multiple domains**, while ignoring those that are *spuriously correlated* to label information

- A key aspect of domain generalization is to learn from **features** that remain *invariant* across **multiple domains**, while ignoring those that are *spuriously correlated* to label information
  - Consider, for example, a model that is built to distinguish between **cows** and **camels** using photos collected in nature under different climates. Since CNNs are known to have a bias towards texture (Geirhos et al., 2018, Brendel and Bethge, 2019), if we simply try to minimize the average loss across different domains, the classifier is prone to spuriously correlate "cow" with grass and "camels" with desert, and predict the species using only the background.
    Such a classifier can be rendered useless when the animals are placed indoors or in a zoo.
    However, if the model could recognize that while the landscapes change with climate, the biological **characteristics of the animals** (e.g. humps, neck lengths) remain invariant and use those features to determine the species, we have a much better chance at generalizing to unseen domains.

- Intuitions have already motivated several approaches that consider **learning "invariances"** accross domains as the **main challenge** of domain generalization.
  Most of these work focuses on **learning *invariant features***, for instance *domain adversarial neural networks* (Ganin et al., 2016), CORAL (Sun and Saenko, 2016) and *MMD for domain generalization* (Li et al., 2018b).

- Different from previous approaches, **invariant risk minimization** (Arjovsky et al., 2019) proposes to learn **intermediate** features such that we have *invariant predictor* (when optimal) across different domains.

ARJOVSKY, Martin, BOTTOU, Léon, GULRAJANI, Ishaan, *et al.* **Invariant risk minimization**. *arXiv preprint arXiv:1907.02893*, 2019.

Invariant Risk Minimization

- Discriminating **invariant** correlations from **spurious** ones

IRM: How to **recognize invariant correlations**?

$$\text{data representation} \quad \Phi : \mathcal{X} \to \mathcal{H}$$
$$\text{classifier} \quad w : \mathcal{H} \to \mathcal{Y}$$

- Hypothesis 1
  - **Learn** from **different environments** (e.g. cows with different backgrounds)
  - **To learn invariances** across environments, find a data representation Φ such that the optimal classifier *w* on top of that representation matches **for all environments** (i.e. is the same irrespective of the environment).

$$\left\{ \begin{array}{ll} \min\limits_{\substack{\Phi:\mathcal{X}\to\mathcal{H} \\ w:\mathcal{H}\to\mathcal{Y}}} & \sum\limits_{e\in\mathcal{E}_{\mathrm{tr}}} R^e(w \circ \Phi) \\ \text{subject to} & w \in \arg\min\limits_{\bar{w}:\mathcal{H}\to\mathcal{Y}} R^e(\bar{w} \circ \Phi), \ \text{for all } e \in \mathcal{E}_{\mathrm{tr}} \end{array} \right.$$

Find the best **invariant predictors**: data representation Φ

Same best solution *w* for all environments

IRM

But **why would that guarantee performance** across all possible "reasonable" environments?

- We have to make assumptions about what the "reasonable" environments are

- Hypothesis 2
  - All environments share the same underlying **Structural Equation Model** (i.e. causal model)

$$\begin{tikzcd}
 & H^e \arrow[dl] \arrow[d] \arrow[dr] & \\
Z_1^e \arrow[r] & Y^e \arrow[r] & Z_2^e
\end{tikzcd}$$

$$\begin{aligned}
H^e &\leftarrow \mathcal{N}(0, e^2) \\
Z_1^e &\leftarrow \mathcal{N}(0, e^2) + W_{h\to 1}H^e \\
Y^e &\leftarrow Z_1^e \cdot W_{1\to y} + \mathcal{N}(0, \sigma_y^2) + W_{h\to y}H^e \\
Z_2^e &\leftarrow W_{y\to 2}Y^e + \mathcal{N}(0, \sigma_2^2) + W_{h\to 2}H^e
\end{aligned}$$

Figure 3: In our synthetic experiments, the task is to predict Y^e from $X^e = S(Z_1^e, Z_2^e)$.

Experiment

- Colored MNIST
  - The goal is to predict the label y = 0 for digits 0-4, and y = 1 for digits 5-9
  - Protocol
    - Three **environments**: 2 training, 1 test. For all:
    - Flip the **label** *y* with probability 0.25 (thus best possible performance = 0.75)
    - Put **color** (red or green) using variable *z*.

      z is obtained by flipping *y* with probability *p*, where *p* = 0.2 in the first training set, 0.1 in the second and 0.9 in the test one (maximal flipping).

      Color the image red if *z* = 1 and green if *z* = 0

The **color** is strongly but spuriously correlated with the class **label**.

Experiments with colored MNIST

!!!

Algorithm	**Acc. train envs.**	**Acc. test env.**
ERM	87.4 ± 0.2	17.1 ± 0.6
IRM (ours)	70.8 ± 0.9	$\mathbf{66.9 \pm 2.5}$
Random guessing (hypothetical)	50	50
Optimal invariant model (hypothetical)	75	75
ERM, grayscale model (oracle)	73.5 ± 0.2	73.0 ± 0.4

Table 1: Accuracy (%) of different algorithms on the Colored MNIST synthetic task.

- Training with ERM returns a model with high accuracy in the training environments but below-chance accuracy in the test environment, since **the ERM model classifies mainly based on color**.
- Training with IRM results in a model that performs worse on the training environments, but relies less on the color and hence generalizes better to the test environments.
- An **oracle** that **ignores color information** by construction outperforms IRM only slightly.

What is the **difference** between

– Multi-Task Learning

– and Invariant Risk Minimization?

What is the **difference** between

- Multi-Task Learning
- and Invariant Risk Minimization?

- **Multi-task** learning
  - Share a **weight vector** in the (given) input representation

$$\mathbf{w}_j = \boxed{\mathbf{w}_0} + \mathbf{v}_j$$

- **I.R.M.**
  - Find a common **representation** (and more elaborate justification)

$$\text{data representation} \quad \Phi : \mathcal{X} \to \mathcal{H}$$

If the **search behavior** is the same

then the space is the same

How transferable are representations?

- The success of deep neural networks in supervised learning relies on the crucial assumption that the train and test data distributions are identical.
  - In particular, the tendency of networks to **rely on simple features** is generally a desirable behavior reflecting Occam's razor.
  - However, in case of distribution shift, this simplicity bias deteriorates performance when more complex features are needed

- To better generalize under distribution shifts, different **invariance criteria** across training domains have been proposed
  - Similar **feature distributions**
  - force the classifier to be **simultaneously optimal** across a **representation common to all domains** (IRM)

BUT none of these methods perform significantly better than the classical Empirical Risk Minimization (Gulrajani & Lopez-Paz, 2021; Ye et al., 2021)

Many Domain Generalization techniques yield limited improvements over empirical risk minimization due to reliance on spurious, domain-specific features.

Need for new ideas

Outline

- Natural idea: share the same **weights**
  - I.e. the **same function** for the source and target domains

- **New** proposal: share the gradients for each neuron
  - learning a model with *invariant gradient direction* for different domains (IDGM: Inter Domain Gradient Matching)
    - **Augment the loss** with an auxiliary term that **maximizes the gradient inner product between domains**, which encourages the alignment between the domain-specific gradients.
    - By simultaneously minimizing the loss and matching the gradients, IDGM **encourages the optimization paths** to be **the same for all domains,** favoring invariant predictions.

SHI, Yuge, SEELY, Jeffrey, TORR, Philip HS, *et al.* **Gradient matching for domain generalization**. *arXiv preprint arXiv:2104.09937*, 2021.

Limits of ERM for O.O.D. learning

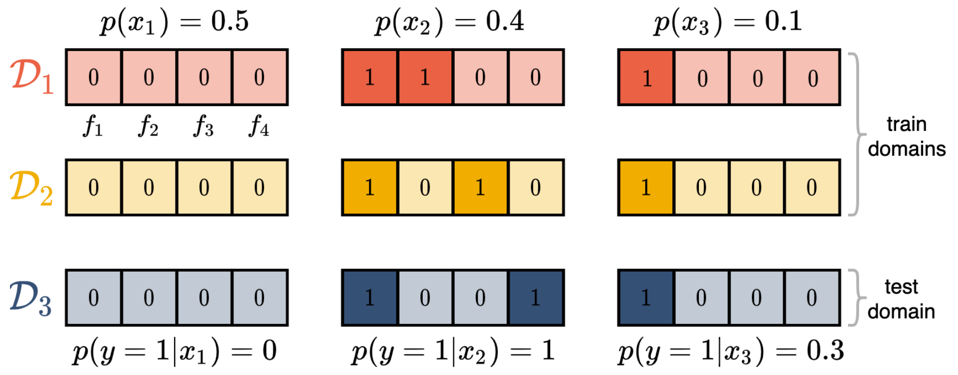


Figure 2: All domains contain 3 types of inputs x_1, x_2 and x_3, each depicted in one column. ***1st col.***: $x_1 = [0, 0, 0, 0]$, $y = 0$, makes up for 50% of each dataset; ***2nd col.***: x_2 changes for each domain, $y = 1$ always. 40% of each dataset; ***3rd col.***: $x_3 = [1, 0, 0, 0]$, 30% of $y = 1$ and 70% of $y = 0$. 10% of each dataset.

Table 1: Performance comparison on the linear dataset.

Method	train acc.	test acc.	W	b
ERM	97%	57%	$[2.8, 3.3, 3.3, 0.0]$	-2.7
IDGM	93%	93%	$[0.4, 0.2, 0.2, 0.0]$	-0.4

...

Inter **D**omain **G**radient **M**atching (IDGM)

- Suppose the train dataset consists of *S* = 2 domains $\mathcal{D}_{tr} = \{\mathcal{D}_1, \mathcal{D}_2\}$
- Given model θ and loss function *l*, the **expected gradients** for data in the two domains is expressed as

$$G_1 = \mathbb{E}_{\mathcal{D}_1} \frac{\partial l((x,y);\theta)}{\partial \theta}, \quad G_2 = \mathbb{E}_{\mathcal{D}_2} \frac{\partial l((x,y);\theta)}{\partial \theta}.$$

- If G_1 and G_2 point in a similar direction, i.e. $G_1 \cdot G_2 > 0$, taking a gradient step along G_1 or G_2 improves the model's performance on both domains, indicating that the features learned by either gradient step are invariant across $\{D_1, D_2\}$.

- Therefore, the new **loss function**:

$$\boxed{\mathcal{L}_{\text{idgm}} = \mathcal{L}_{\text{erm}}(\mathcal{D}_{tr};\theta) - \gamma \frac{2}{S(S-1)} \sum_{i,j \in S}^{i \neq j} G_i \cdot G_j,}$$

- The new **loss function**:

$$\boxed{\mathcal{L}_{\text{idgm}} = \mathcal{L}_{\text{erm}}(\mathcal{D}_{tr};\theta) - \gamma \frac{2}{S(S-1)} \sum_{i,j\in S}^{i\neq j} G_i \cdot G_j,}$$

is computationally expensive (requires computing the **second order derivative** of the model's parameters), thus [Shi et al. (2021)] propose a first-order simplification algorithm named Fish.

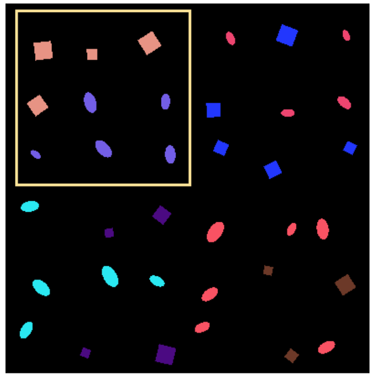
Algorithm 1 Fish.

```
1:  for iterations = $1, 2, \cdots$ do
2:      $\widetilde{\theta} \leftarrow \theta$
3:      for $\mathcal{D}_i \in$ permute($\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\}$) do
4:          Sample batch $d_i \sim \mathcal{D}_i$
5:          $\widetilde{g}_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\widetilde{\theta})}{\partial \widetilde{\theta}}\right]$   //Grad wrt $\widetilde{\theta}$
6:          Update $\widetilde{\theta} \leftarrow \widetilde{\theta} - \alpha \widetilde{g}_i$
7:      end for
8:
9:      Update $\theta \leftarrow \theta + \epsilon(\widetilde{\theta} - \theta)$
10: end for
```

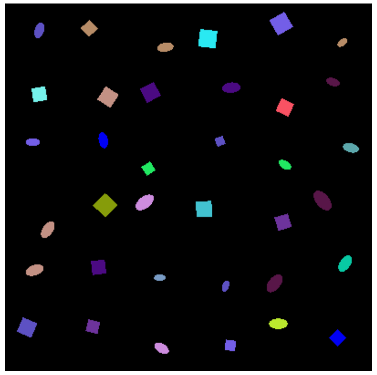
Algorithm 2 Direct optimization of IDGM.

```
1:  for iterations = $1, 2, \cdots$ do
2:      $\widetilde{\theta} \leftarrow \theta$
3:      for $\mathcal{D}_i \in$ permute($\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\}$) do
4:          Sample batch $d_i \sim \mathcal{D}_i$
5:          $g_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\theta)}{\partial \theta}\right]$   //Grad wrt $\theta$
6:
7:      end for
8:      $\bar{g} = \frac{1}{S}\sum_{s=1}^{S} g_s, \quad \overbrace{\widehat{g} = \frac{2}{S(S-1)} \sum_{i,j\in S}^{i\neq j} g_i \cdot g_j}^{\text{GIP (batch)}}$
9:      Update $\theta \leftarrow \theta - \epsilon\left(\bar{g} - \gamma(\partial \widehat{g}/\partial \theta)\right)$
10: end for
```

Controlled experiment

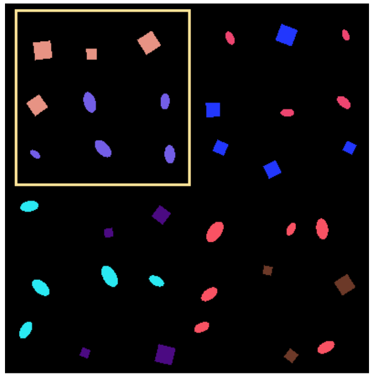

(a) Train

Four = 2^N (with N=2) **training domains**. In each, the **shape** is **correlated** with **color** (e.g. in top, left, pink → square, purple → oval)

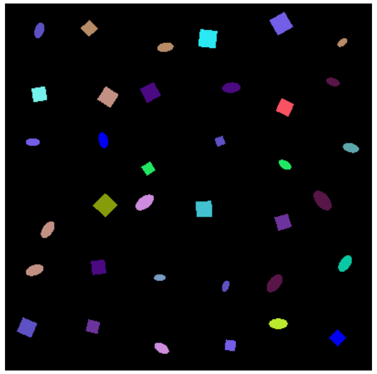

(b) Test

Test domain: 2^N (here N=2) colors are **randomly** associated with the **shapes**

Controlled experiment



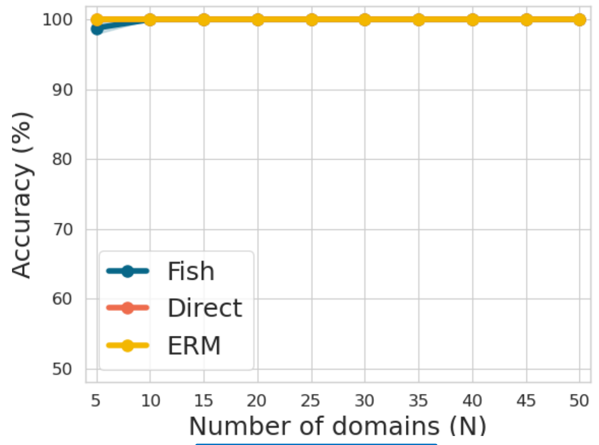
(a) Train



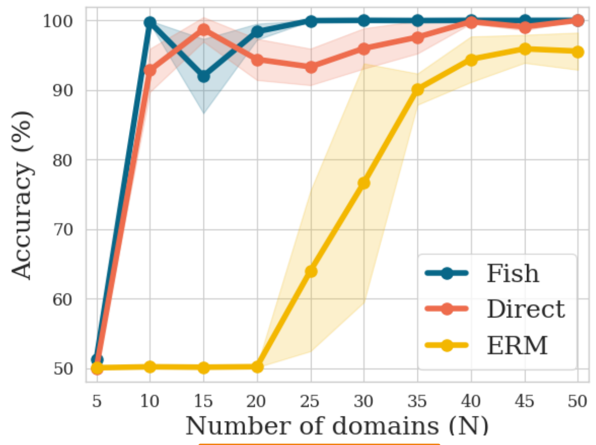
(b) Test

Four = 2^N (with N=2) **training domains**. In each, the **shape** is **correlated** with **color** (e.g. in top, left, pink → square, purple → oval)

Test domain: 2^N (here N=2) colors are **randomly** associated with the **shapes**



(a) Train



(b) Test

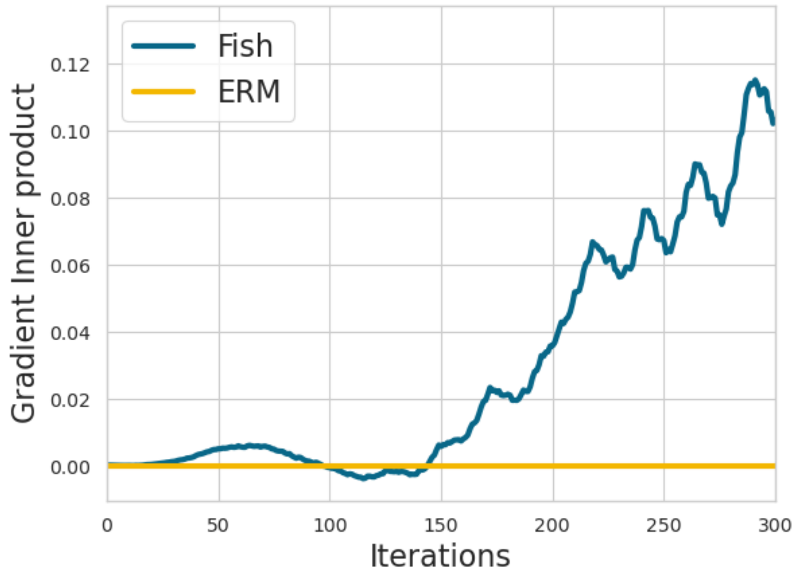
Figure 4: Performance on CDSPRITES-N, with $N \in [5, 50]$

If the value of N is **small enough**, the model can simply memorize the *N* colors that correspond to each shape, and make predictions solely based on colors (what **ERM** does in that case), and therefore be very poor on the test set.

Tracking the inter-domain gradient inner product

- Both Fish (blue) and ERM (yellow) until convergence while tracking the normalized gradient inner products between minibatches **from different domains** used in each inner-loop.
- **During training**, the normalized gradient inner product of Fish **increases**, while that for ERM **stays** at the same value.

Gradient inner product values during the training for CDSPRITES-N (N=15).



Follow up of this work

- RAME, Alexandre, DANCETTE, Corentin, et CORD, Matthieu. **Fishr: Invariant gradient variances for out-of-distribution generalization**. In: *International Conference on Machine Learning*. PMLR, 2022. p. 18347-18377.

Outline

Conclusion

- Many **types of invariances** between tasks and/or domains have been explored
- Surely, many more wait to be considered

- We have seen transfer learning by:
  - Sharing a **representation**
  - Identifying **common regularities**
  - Sharing the **search directions**

In the future, we will see a transfer of **decision function** without sharing, not even in the same space

Bibliography

- Ben-David, S., Lu, T., Luu, T., & Pál, D. (2010). Impossibility theorems for domain adaptation. In *International Conference on Artificial Intelligence and Statistics* (pp. 129-136).
- Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, *79*(1-2), 151-175.
- Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.
- Kuzborskij, I., & Orabona, F. (2013, February). Stability and hypothesis transfer learning. In *International Conference on Machine Learning* (pp. 942-950).
- Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*.
- Redko, I., Morvant, E., Habrard, A., Sebban, M., & Bennani, Y. (2019). *Advances in Domain Adaptation Theory*. Elsevier.
- H. Venkateswara, S. Chakraborty, and S. Panchanathan, "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 117–129, 2017.
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).
- Zhang, C., Zhang, L., & Ye, J. (2012). Generalization bounds for domain adaptation. In *Advances in neural information processing systems* (pp. 3320-3328).