2022/03/14

# Lab Seminar

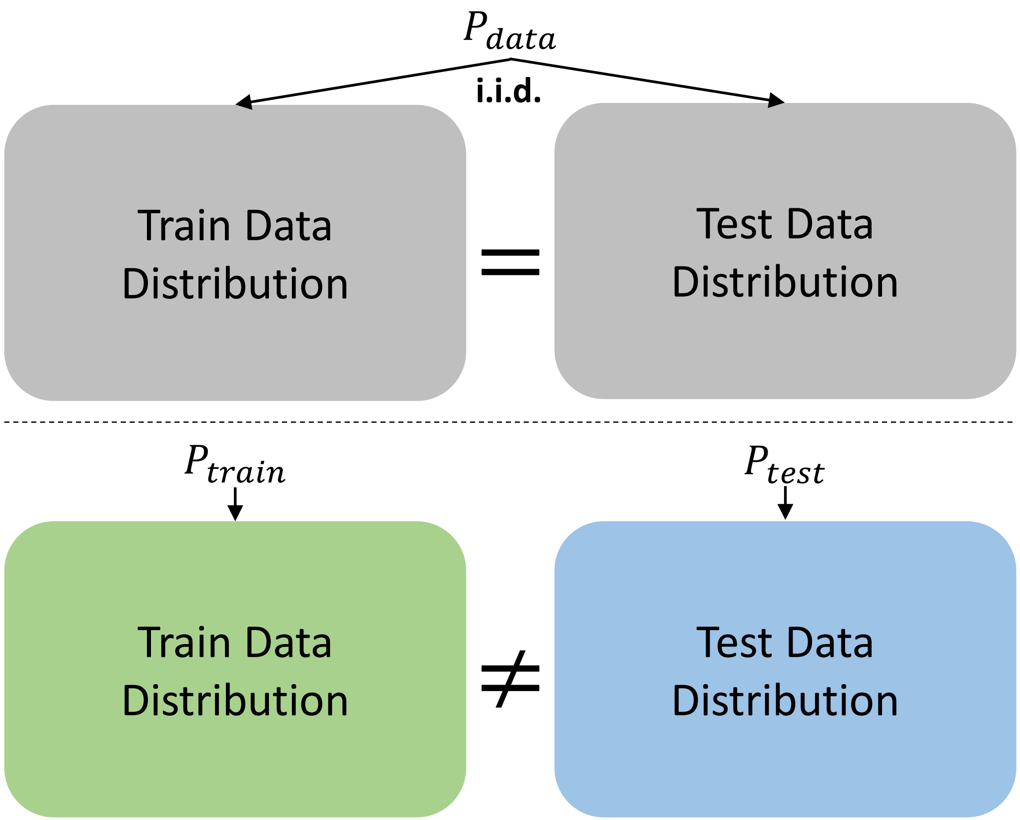
Domain Generalization
Gradient Matching for Domain Generalization

Taero Kim

MLAI, Univ. of Seoul

# Domain Generalization

- **Distribution Shift**




We take the assumption for granted that **train data and test data are sampled from same distribution.**

In reality, however, **distributions of train and test data are always different each other** unless the users adjust them carefully.
We call this situation **"Distribution Shift"**.

Inspired by Neurips 2021 Workshops Distribution shifts: connecting methods and applications (DistShift)

## Distribution Shift

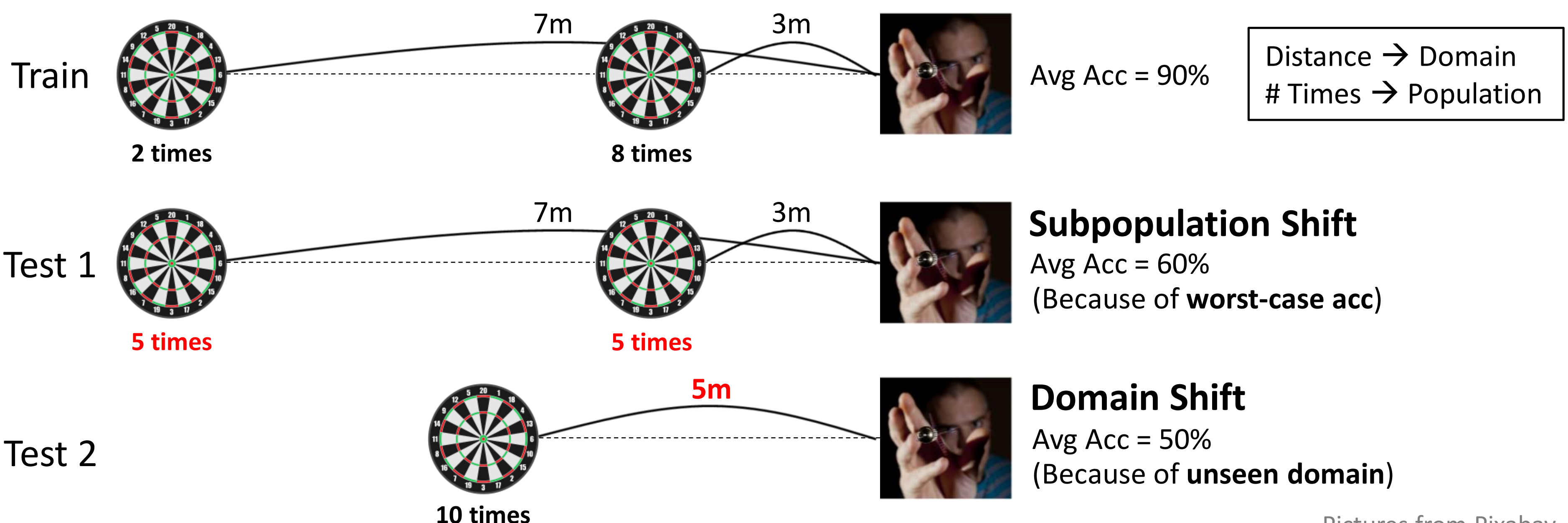
There are many types of Distribution Shift;

- Covariate Shift, **Domain Shift, Subpopulation Shift** , Continual Shift, etc.

In this seminar, we will consider only about domain shift and subpopulation shift.

To understand **the difference between domain shift and subpopulation shift,** we can introduce Darts game.

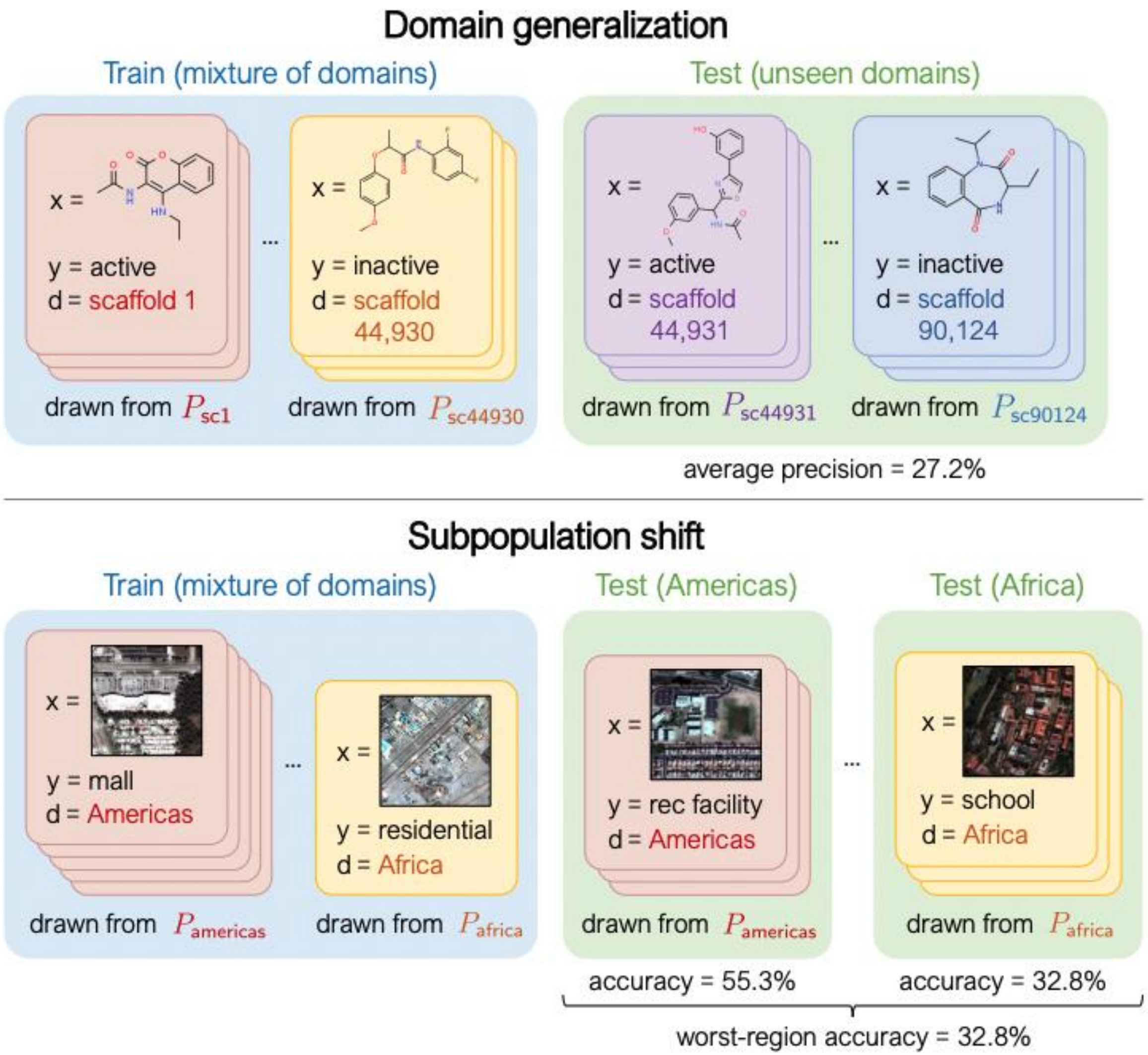
Let **d** be the distance between the player and the dart board.
Less than 10 points = label 0 / More than 10 points = label 1
The player practiced 8 times when d=3m and 2 times when d=7m.




Pictures from Pixabay

MLAI University of Seoul

- **Domain Generalization and Subpopulation Shift**
  **Wilds Dataset 1.2v**

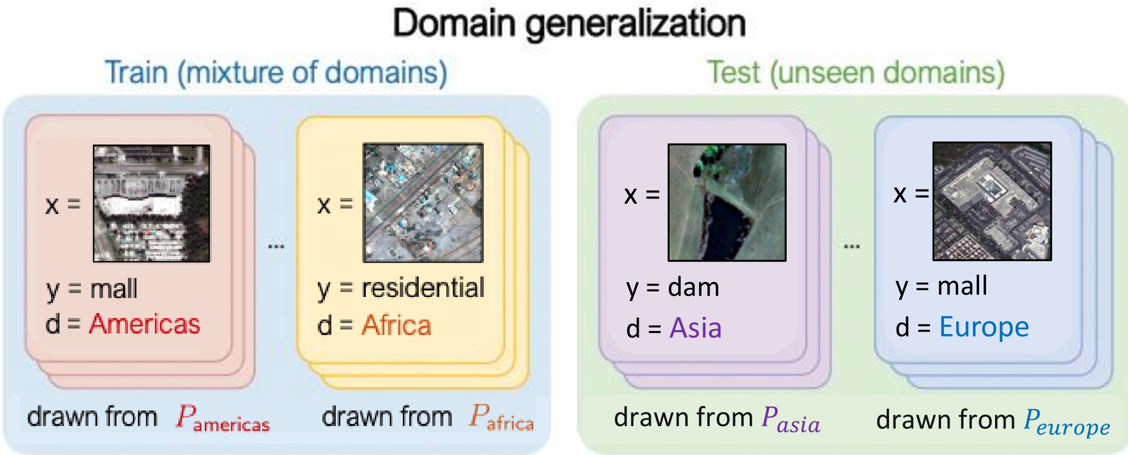



Koh et al., 2021

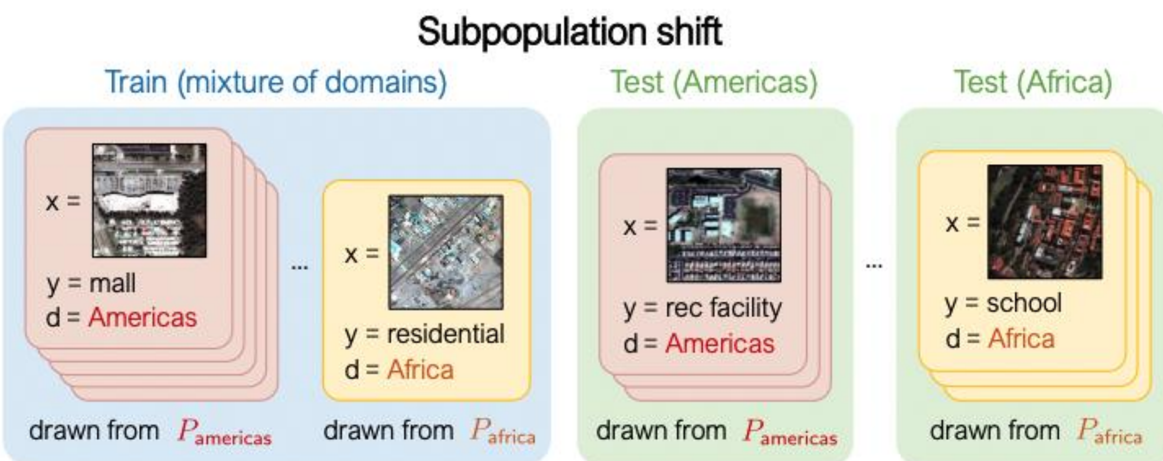
MLAI University of Seoul

- **Domain Generalization VS Subpopulation Shift**

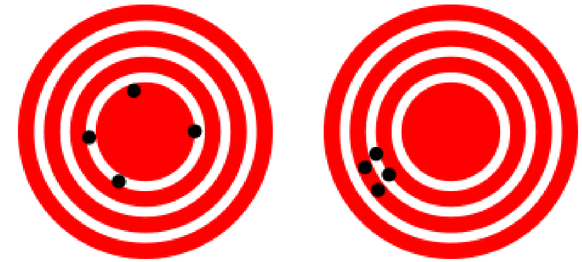
## fMoW Datasets in Wilds 1.2v




Avg Accuracy (std)
Precision




Worst-case Acc



**Low accuracy** due to **low precision**

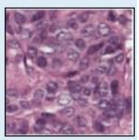
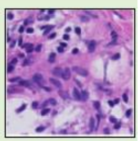
**Low accuracy** even with **high precision**

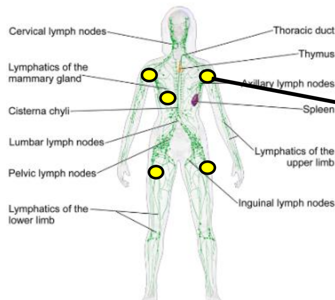
**Goal of DG is to get high performance about related but unseen domain.** The domain can vary depending on how we set it.

**Ensuring Subpopulation Shift is one way to improve DG.** Their methodologies are also similar. However, subpopulation shift alone cannot guarantee performance for unseen domain.

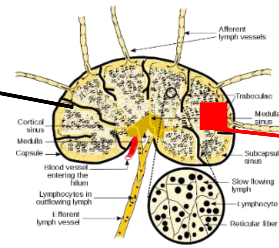
Inspired by Christie et al., 2018; Koh et al., 2021; Wikipedia Accuracy and Precision;

# Camelyon17 Dataset – Wilds 1.2

| Dataset | Camelyon17 |
|---|---|
| Input (x) | tissue slide |
| Prediction (y) | tumor |
| Domain (d) | hospital |
| # domains | 5 |
| # examples | 455,954 |
| Train example |  |
| Test example |  |
| Adapted from | Bandi et al. 2018 |

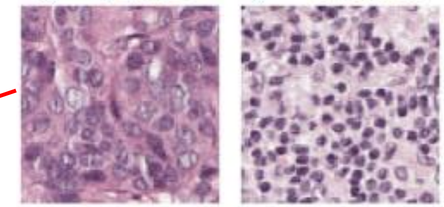



450 Lymph Node in human




Lymph Node


Slide




Tissue

Domain Generalization does not always guarantee worst-case accuracy.

| Center A | Slide 1 | … | Slide10 |
|---|---|---|---|
| Normal (0) | 97% | | 48% |
| Tumor (1) | 3% | | 52% |

Worst-case

```
====> Epoch: 019
=============== val ===============
Average acc: 0.838
  slide = 10  [n =   2078]: acc = 0.861
  slide = 11  [n =   5371]: acc = 0.490
  slide = 12  [n =   1705]: acc = 0.880
  slide = 13  [n =  10404]: acc = 0.948
  slide = 14  [n =    888]: acc = 0.957
  slide = 15  [n =   2966]: acc = 0.973
  slide = 16  [n =   4971]: acc = 0.869
  slide = 17  [n =   2659]: acc = 0.744
  slide = 18  [n =   1275]: acc = 0.932
  slide = 19  [n =   2587]: acc = 0.866
Worst-group acc: 0.490
```

Center A — Slides

```
=============== test ===============
Average acc: 0.802
  slide = 20  [n =   3810]: acc = 0.580
  slide = 21  [n =   3694]: acc = 0.469
  slide = 22  [n =   7210]: acc = 0.856
  slide = 23  [n =   5288]: acc = 0.840
  slide = 24  [n =   7727]: acc = 0.728
  slide = 25  [n =   4334]: acc = 0.770
  slide = 26  [n =   3815]: acc = 0.693
  slide = 27  [n =   4556]: acc = 0.536
  slide = 28  [n =  31878]: acc = 0.917
  slide = 29  [n =  12742]: acc = 0.813
Worst-group acc: 0.469
```

Center B — Slides

train/val/test
000/0/0

**10 slides per center**

Center (5) → Patient (43) → Node (5) → **slide (50)** → **normal (class 0)** / **tumor (class 1)**

**Domain** — **Group**

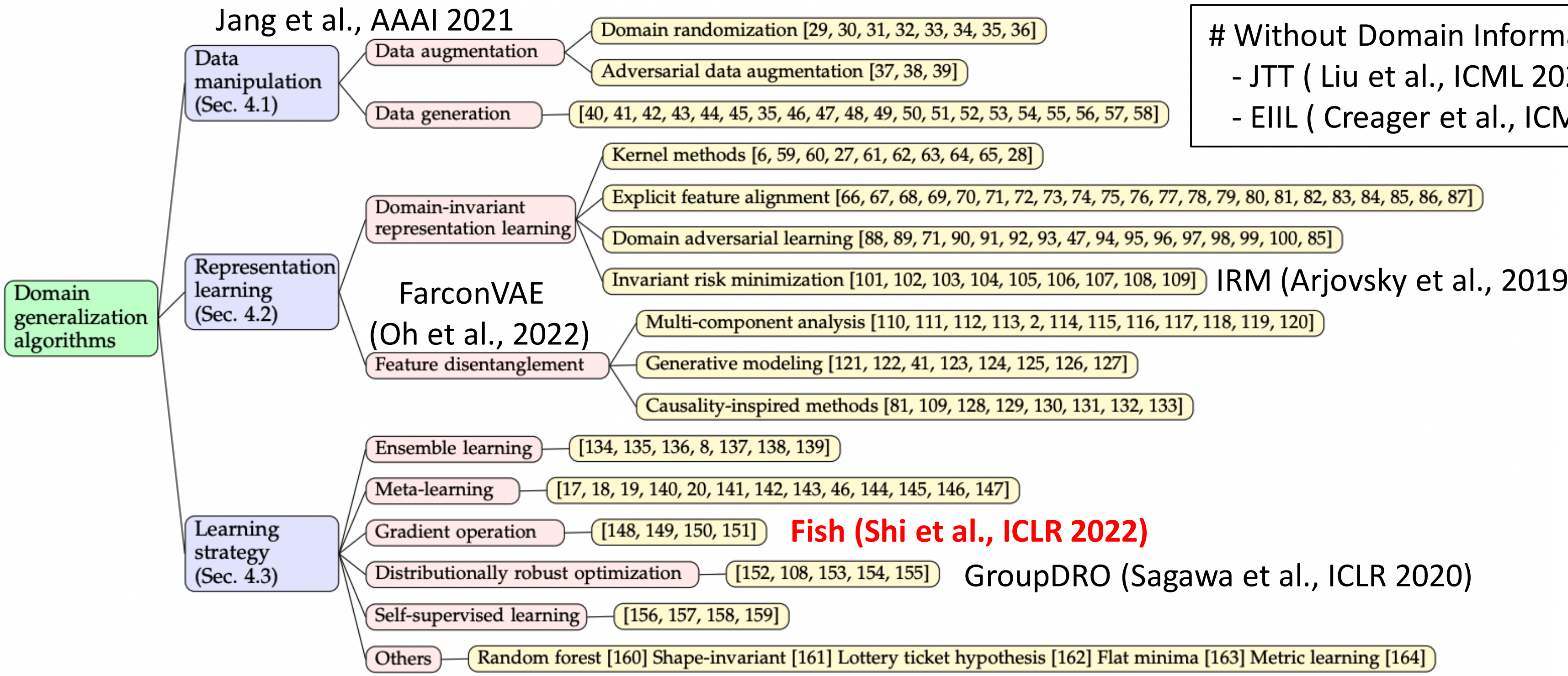
Koh et al., 2021

MLAI
University of Seoul

- **Algorithms Taxonomy for Domain Generalization**

Sorted by method

# Without Domain Information
- JTT ( Liu et al., ICML 2021)
- EIIL ( Creager et al., ICML 2021)

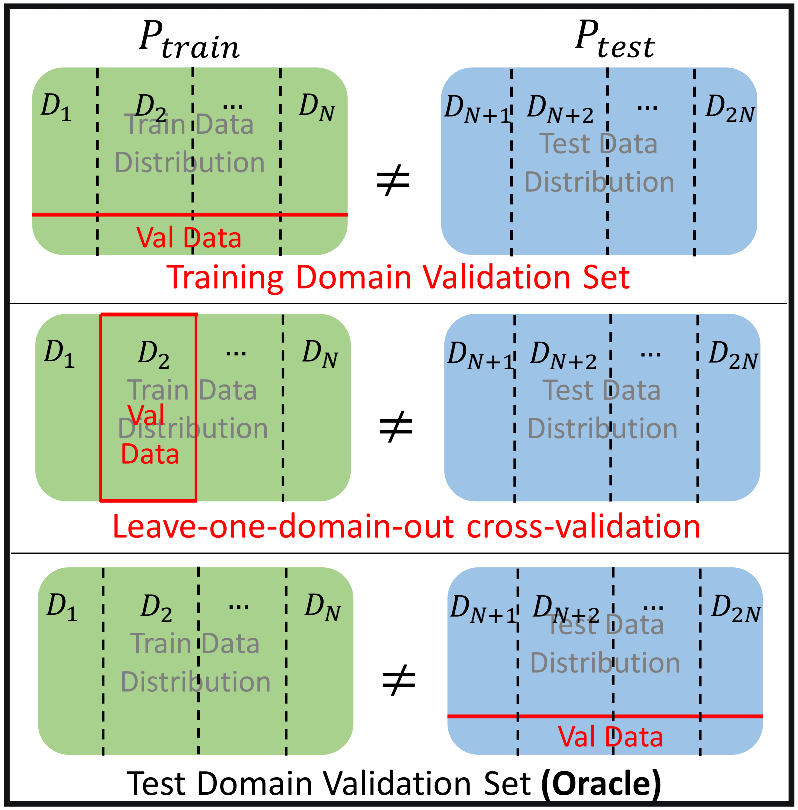



This picture shows the taxonomy of domain generalization algorithms. DG is a topic that has recently been in the spotlight, and it is not determined which methods works better.

Wang et al., IJACAI 2021

- **DomainBed – Model Selection Method & DG Benchmark**

**Recommendation 1** *A domain generalization algorithm should be responsible for specifying a model selection method.*




Model selection method: training domain validation set

| Algorithm | CMNIST | RMNIST | VLCS | PACS | Office-Home | TerraInc | DomainNet | Avg |
|---|---|---|---|---|---|---|---|---|
| ERM | 52.0 ± 0.1 | 98.0 ± 0.0 | 77.4 ± 0.3 | 85.7 ± 0.5 | 67.5 ± 0.5 | 47.2 ± 0.4 | 41.2 ± 0.2 | 67.0 |
| IRM | 51.8 ± 0.1 | 97.9 ± 0.0 | 78.1 ± 0.0 | 84.4 ± 1.1 | 66.6 ± 1.0 | 47.9 ± 0.7 | 35.7 ± 1.9 | 66.0 |
| DRO | 52.0 ± 0.1 | 98.1 ± 0.0 | 77.2 ± 0.6 | 84.1 ± 0.4 | 66.9 ± 0.3 | 47.0 ± 0.3 | 33.7 ± 0.2 | 65.5 |
| Mixup | 51.9 ± 0.1 | 98.1 ± 0.0 | 77.7 ± 0.4 | 84.3 ± 0.5 | 69.0 ± 0.1 | 48.9 ± 0.8 | 39.6 ± 0.1 | 67.1 |
| MLDG | 51.6 ± 0.1 | 98.0 ± 0.0 | 77.1 ± 0.4 | 84.8 ± 0.6 | 68.2 ± 0.1 | 46.1 ± 0.8 | 41.8 ± 0.4 | 66.8 |
| CORAL | 51.7 ± 0.1 | 98.1 ± 0.1 | 77.7 ± 0.5 | 86.0 ± 0.2 | 68.6 ± 0.4 | 46.4 ± 0.8 | 41.8 ± 0.2 | 67.2 |
| MMD | 51.8 ± 0.1 | 98.1 ± 0.0 | 76.7 ± 0.9 | 85.0 ± 0.2 | 67.7 ± 0.1 | 49.3 ± 1.4 | 39.4 ± 0.8 | 66.8 |
| DANN | 51.5 ± 0.3 | 97.9 ± 0.1 | 78.7 ± 0.3 | 84.6 ± 1.1 | 65.4 ± 0.6 | 48.4 ± 0.5 | 38.4 ± 0.0 | 66.4 |
| C-DANN | 51.9 ± 0.1 | 98.0 ± 0.0 | 78.2 ± 0.4 | 82.8 ± 1.5 | 65.6 ± 0.5 | 47.6 ± 0.8 | 38.9 ± 0.1 | 66.1 |

Model selection method: Leave-one-domain-out cross-validation

| Algorithm | CMNIST | RMNIST | VLCS | PACS | Office-Home | TerraInc | DomainNet | Avg |
|---|---|---|---|---|---|---|---|---|
| ERM | 34.2 ± 1.2 | 98.0 ± 0.0 | 76.8 ± 1.0 | 83.3 ± 0.6 | 67.3 ± 0.3 | 46.2 ± 0.2 | 40.8 ± 0.2 | 63.8 |
| IRM | 36.3 ± 0.4 | 97.7 ± 0.1 | 77.2 ± 0.3 | 82.9 ± 0.6 | 66.7 ± 0.7 | 44.0 ± 0.7 | 35.3 ± 1.5 | 62.9 |
| DRO | 32.2 ± 3.7 | 97.9 ± 0.1 | 77.5 ± 0.1 | 83.1 ± 0.6 | 67.1 ± 0.3 | 42.5 ± 0.2 | 32.8 ± 0.2 | 61.8 |
| Mixup | 31.2 ± 2.1 | 98.1 ± 0.1 | 78.6 ± 0.2 | 83.7 ± 0.9 | 68.2 ± 0.3 | 46.1 ± 1.6 | 39.4 ± 0.3 | 63.6 |
| MLDG | 36.9 ± 0.2 | 98.0 ± 0.1 | 77.1 ± 0.6 | 82.4 ± 0.7 | 67.6 ± 0.3 | 45.8 ± 1.2 | 42.1 ± 0.1 | 64.2 |
| CORAL | 29.9 ± 2.5 | 98.1 ± 0.1 | 77.0 ± 0.5 | 83.6 ± 0.6 | 68.6 ± 0.2 | 48.1 ± 1.3 | 41.9 ± 0.2 | 63.9 |
| MMD | 42.6 ± 3.0 | 98.1 ± 0.1 | 76.7 ± 0.9 | 82.8 ± 0.3 | 67.1 ± 0.5 | 46.3 ± 0.5 | 39.3 ± 0.9 | 64.7 |
| DANN | 29.0 ± 7.7 | 89.1 ± 5.5 | 77.7 ± 0.3 | 84.0 ± 0.5 | 65.5 ± 0.1 | 45.7 ± 0.8 | 37.5 ± 0.2 | 61.2 |
| C-DANN | 31.1 ± 8.5 | 96.3 ± 1.0 | 74.0 ± 1.0 | 81.7 ± 1.4 | 64.7 ± 0.4 | 40.6 ± 1.8 | 38.7 ± 0.2 | 61.1 |

Model selection method: Test-domain validation set *(oracle)*

| Algorithm | CMNIST | RMNIST | VLCS | PACS | Office-Home | TerraInc | DomainNet | Avg |
|---|---|---|---|---|---|---|---|---|
| ERM | 58.5 ± 0.3 | 98.1 ± 0.1 | 77.8 ± 0.3 | 87.1 ± 0.3 | 67.1 ± 0.5 | 52.7 ± 0.2 | 41.6 ± 0.1 | 68.9 |
| IRM | 70.2 ± 0.2 | 97.9 ± 0.0 | 77.1 ± 0.2 | 84.6 ± 0.5 | 67.2 ± 0.8 | 50.9 ± 0.4 | 36.0 ± 1.6 | 69.2 |
| DRO | 61.2 ± 0.6 | 98.1 ± 0.0 | 77.4 ± 0.6 | 87.2 ± 0.4 | 67.7 ± 0.4 | 53.1 ± 0.5 | 34.0 ± 0.1 | 68.4 |
| Mixup | 58.4 ± 0.2 | 98.0 ± 0.0 | 78.7 ± 0.4 | 86.4 ± 0.2 | 68.5 ± 0.5 | 52.9 ± 0.3 | 40.3 ± 0.3 | 69.0 |
| MLDG | 58.4 ± 0.2 | 98.0 ± 0.1 | 77.8 ± 0.4 | 86.8 ± 0.2 | 67.4 ± 0.2 | 52.4 ± 0.3 | 42.5 ± 0.1 | 69.1 |
| CORAL | 57.6 ± 0.5 | 98.2 ± 0.0 | 77.8 ± 0.1 | 86.9 ± 0.2 | 68.6 ± 0.4 | 52.6 ± 0.6 | 42.1 ± 0.1 | 69.1 |
| MMD | 63.4 ± 0.7 | 97.9 ± 0.1 | 78.0 ± 0.4 | 87.1 ± 0.5 | 67.0 ± 0.2 | 52.7 ± 0.2 | 39.8 ± 0.7 | 69.4 |
| DANN | 58.3 ± 0.2 | 97.9 ± 0.0 | 80.1 ± 0.6 | 85.4 ± 0.7 | 65.6 ± 0.3 | 51.6 ± 0.6 | 38.3 ± 0.1 | 68.2 |
| C-DANN | 62.0 ± 1.1 | 97.8 ± 0.1 | 80.2 ± 0.1 | 85.7 ± 0.3 | 65.6 ± 0.3 | 51.0 ± 1.0 | 38.9 ± 0.1 | 68.7 |

**Recommendation 2** *Researchers should disclaim any oracle-selection results as such and specify policies to limit access to the test domain.*

ERM Loss $L_{ERM}(\mathcal{D}_{tr};\theta) = \mathbb{E}_{\mathcal{D}\sim\mathcal{D}_{tr}}\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell((x,y);\theta)]$

**ERM is powerful algorithms for Domain Generalization**

Gulrajani et al., ICLR 2021

# Gradient Matching for Domain Generalization

# Performances of Fish

**[Fish] Gradient Matching for Domain Generalization. (Shi et al., ICLR 2022)**

Table 4: Test accuracy (%) on DOMAINBED benchmark.

| | ERM | IRM | GroupDRO | Mixup | MLDG | Coral | MMD | DANN | CDANN | Fish (ours) |
|---|---|---|---|---|---|---|---|---|---|---|
| CMNIST | 52.0 (±0.1) | 51.8 (±0.1) | 52.0 (±0.1) | 51.9 (±0.1) | 51.6 (±0.1) | 51.7 (±0.1) | 51.8 (±0.1) | 51.5 (±0.3) | 51.9 (±0.1) | 51.6 (±0.1) |
| RMNIST | 98.0 (±0.0) | 97.9 (±0.0) | 98.1 (±0.0) | 98.1 (±0.0) | 98.0 (±0.0) | 98.1 (±0.1) | 98.1 (±0.0) | 97.9 (±0.1) | 98.0 (±0.0) | 98.0 (±0.0) |
| VLCS | 77.4 (±0.3) | 78.1 (±0.0) | 77.2 (±0.6) | 77.7 (±0.4) | 77.1 (±0.4) | 77.7 (±0.5) | 76.7 (±0.9) | 78.7 (±0.3) | 78.2 (±0.4) | 77.8 (±0.3) |
| PACS | 85.7 (±0.5) | 84.4 (±1.1) | 84.1 (±0.4) | 84.3 (±0.5) | 84.8 (±0.6) | 86.0 (±0.2) | 85.0 (±0.2) | 84.6 (±1.1) | 82.8 (±1.5) | 85.5 (±0.3) |
| OfficeHome | 67.5 (±0.5) | 66.6 (±1.0) | 66.9 (±0.3) | 69.0 (±0.1) | 68.2 (±0.1) | 68.6 (±0.4) | 67.7 (±0.1) | 65.4 (±0.6) | 65.6 (±0.5) | 68.6 (±0.4) |
| TerraInc | 47.2 (±0.4) | 47.9 (±0.7) | 47.0 (±0.3) | 48.9 (±0.8) | 46.1 (±0.8) | 46.4 (±0.8) | 49.3 (±1.4) | 48.7 (±0.5) | 47.6 (±0.8) | 45.1 (±1.3) |
| DomainNet | 41.2 (±0.2) | 35.7 (±1.9) | 33.7 (±0.2) | 39.6 (±0.1) | 41.8 (±0.4) | 41.8 (±0.2) | 39.4 (±0.8) | 38.4 (±0.0) | 38.9 (±0.1) | 42.7 (±0.2) |
| Average | 67.0 | 66.0 | 65.5 | 67.1 | 66.8 | 67.2 | 66.8 | 66.4 | 66.1 | 67.1 |

Model Selection Method : Train Domain Validation Set

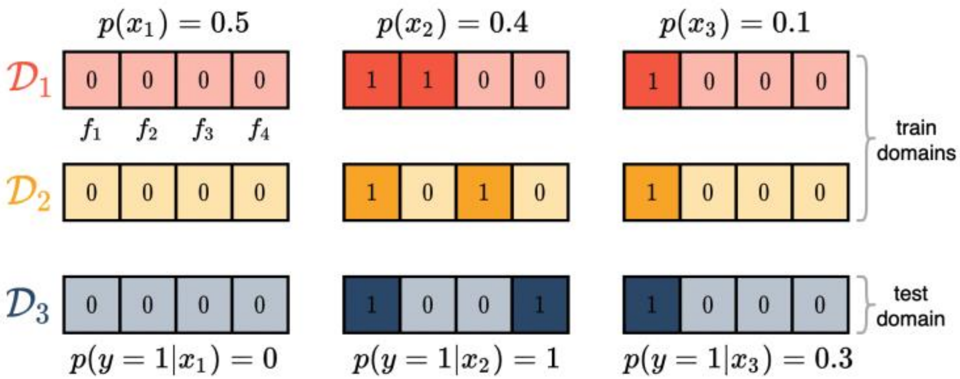
Table 3: Results on WILDS benchmark.

| | POVERTYMAP | CAMELYON17 | FMOW | CIVILCOMMENTS | IWILDCAM | AMAZON |
|---|---|---|---|---|---|---|
| | Pearson r | Avg. acc. (%) | Worst acc. (%) | Worst acc. (%) | Macro F1 | 10-th per. acc. (%) |
| **Fish** | **0.80** (±1e-2) | **74.7** (±7e-2) | **34.6** (±0.00) | **72.8** (±0.0) | 22.0 (±0.0) | **53.3** (±0.0) |
| IRM | 0.78 (±3e-2) | 64.2 (±8.1) | 33.5 (±1.35) | 66.3 (±2.1) | 15.1 (±4.9) | 52.4 (±0.8) |
| Coral | 0.77 (±5e-2) | 59.5 (±7.7) | 31.0 (±0.35) | 65.6 (±1.3) | **32.8**(±0.1) | 52.9 (±0.8) |
| Reweighted | - | - | - | 66.2 (±1.2) | - | 52.4 (±0.8) |
| GroupDRO | 0.78 (±5e-2) | 68.4 (±7.3) | 31.4 (±2.10) | 69.1 (±1.8) | 23.9 (±2.1) | 53.5 (±0.0) |
| ERM | 0.78 (±3e-2) | 70.3 (±6.4) | 32.8 (±0.45) | 56.0 (±3.6) | 31.0 (±1.3) | **53.8** (±0.8) |
| ERM (ours) | 0.77 (±5e-2) | 70.5 (±12.1) | 30.9 (±1.53) | 58.1 (±1.7) | 25.1 (±0.2) | **53.3** (±0.8) |

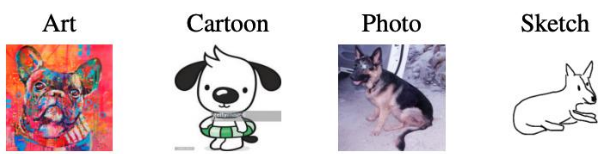
Shi et al., ICLR 2022

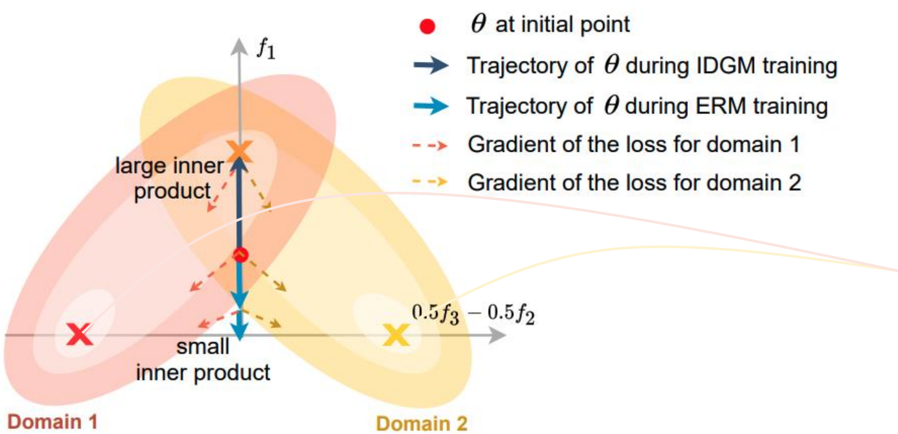
- **Inter Domain Gradient Matching - IDGM**

ERM Loss $L_{ERM}(\mathcal{D}_{tr};\theta) = \mathbb{E}_{\mathcal{D}\sim\mathcal{D}_{tr}}\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell((x,y);\theta)]$




Class: Dog




If there are only $x_1$ and $x_2$ types of data, but when $0 \le f_i \le 1$ *given* $\forall f_i \in \mathbb{R}$, will the model distinguish class well only using invariant feature $f_1$?

→ **In reality, No**. It depends on **correlation** between class and each feature.

→ Many factors can contribute to $Corr(f_i)$, e.g. feature preference, population, etc.

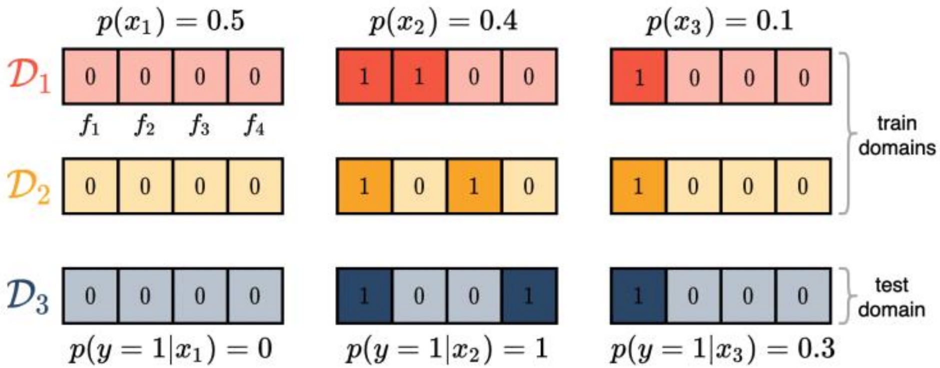
→ However, in this toy example, $f_i$ can have only binary value, 0 or 1. So, the model will be use the invariant feature $f_1$ **"mainly"**.

→ If we want model to use **spurious feature $f_2, f_3$**, rather than **invariant feature $f_1$**, we can inject some **noise** about $f_1$.

→ Finally, **we can understand that the role of $x_3$ is the noise for original correlation.**

Gulrajani et al., ICLR 2021

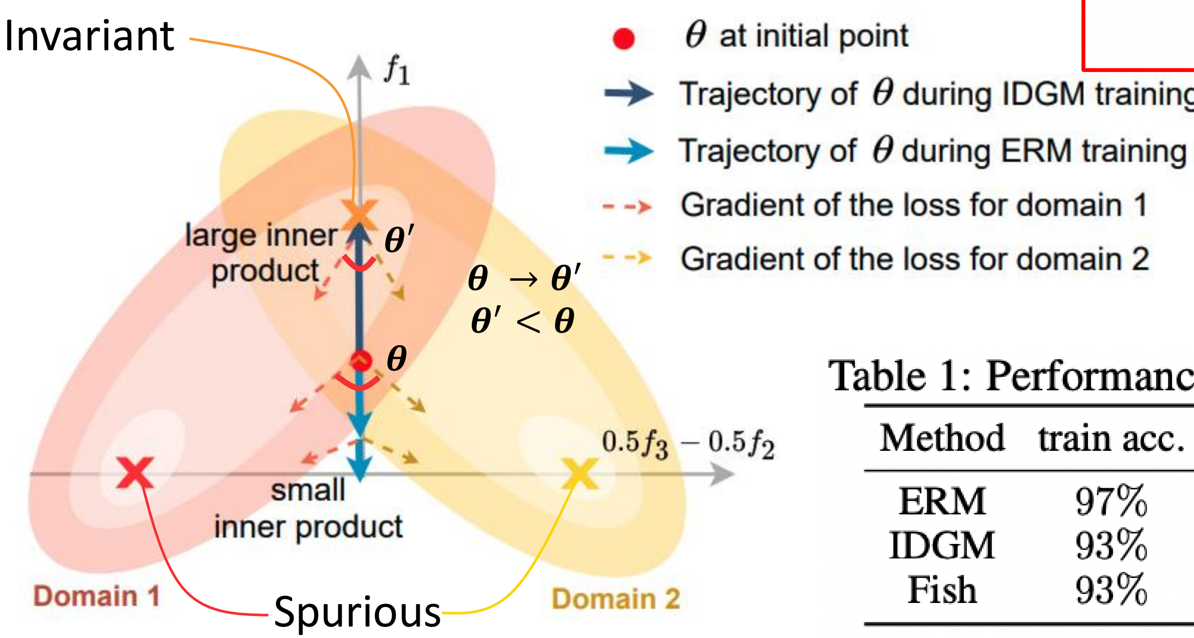
- **Inter Domain Gradient Matching - IDGM**




We can expect that the model can learn invariant feature across domain, if we maximize the inner product of gradient for each domain.

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

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




$G_i \cdot G_j > 0$

$G_i \cdot G_j < 0$

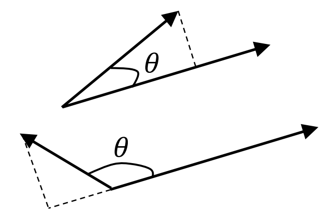
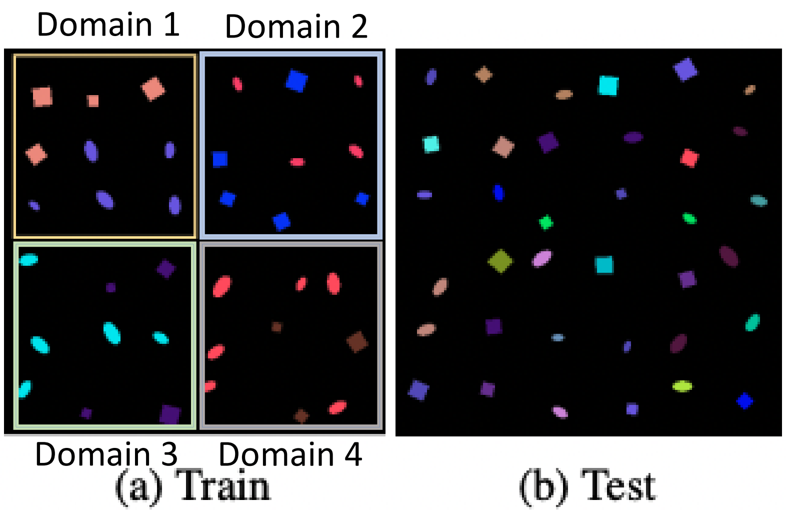



Table 1: Performance comparison on the linear dataset.

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

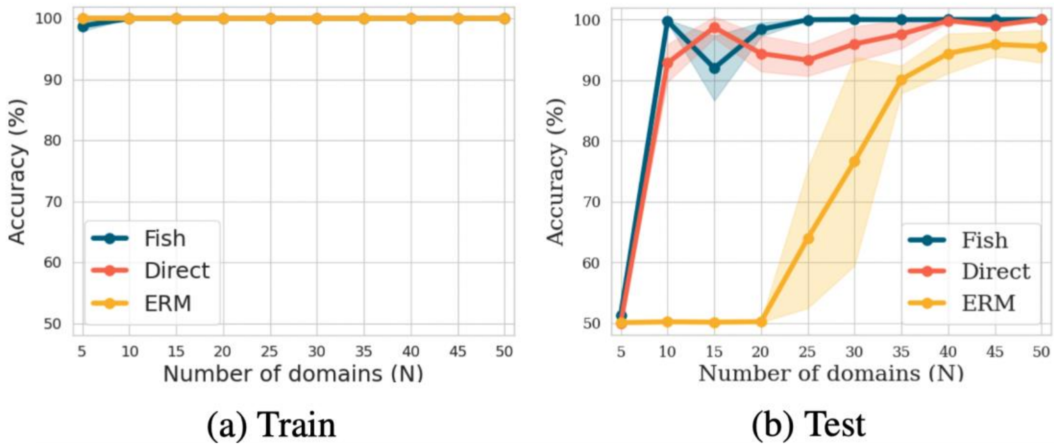
MLAI
University of Seoul

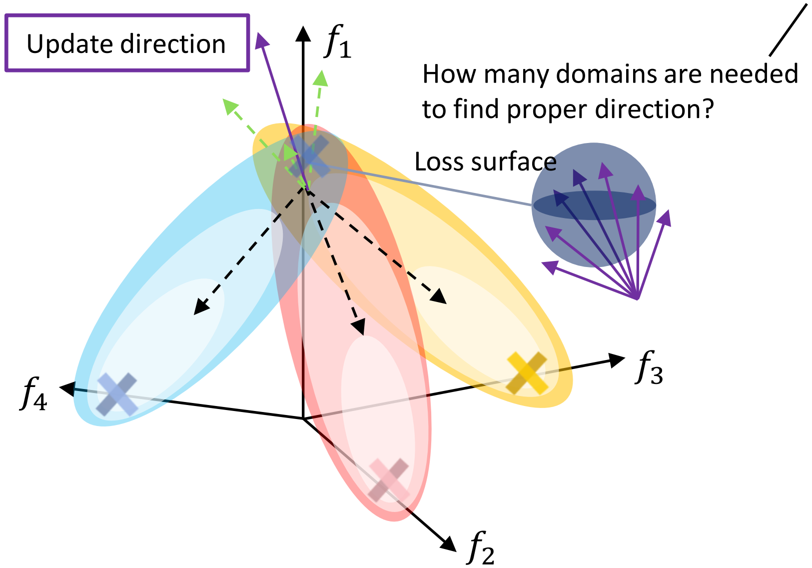
# • CDSprites-N Dataset




Label (Invariant) : Shape

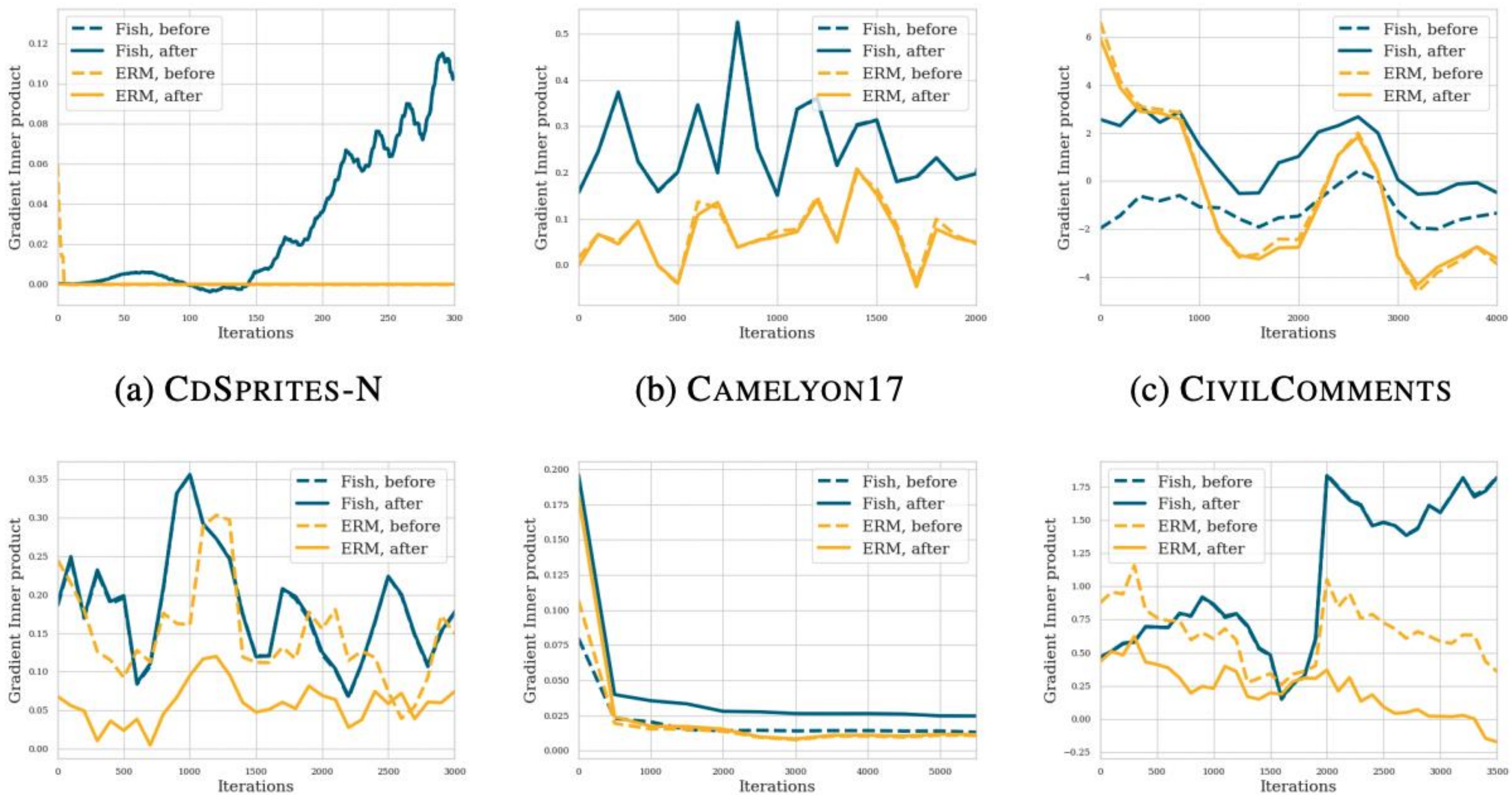
Attribute(Spurious) : Color




- When the number of domain is small, the model can easily remember the color, so it predict the result by using color instead of shape.
- But as the number of domain increase enough, IDGM, Fish and ERM can learn shape feature, rather than color.
- IDGM and Fish need the small number of domains than ERM. **So, we can say that IDGM and Fish have stronger generalization capabilities.**

- **Tracking Gradient Inner Product (GIP)**




(a) CdSprites-N (b) Camelyon17 (c) CivilComments

(d) FMoW (e) Poverty (f) iWildCam

(dot) Compute GIP before model update / (solid) Compute GIP after model update

Comparing ERM with Fish, ERM often results in the decrease of GIP,
**while for Fish it can either increase significantly or at least stay at same level.**

- **IDGM to Fish**

**Unfortunately, IDGM algorithms is too expensive to compute.**

**So, we will approximate hessian to first-order optimization. → Fish**

**Algorithm 1** Fish.

**for** iterations $= 1, 2, \cdots$ **do**
  $\boxed{\tilde{\theta} \leftarrow \theta}$
  **for** $\mathcal{D}_i \in \texttt{permute}(\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\})$ **do**
    Sample batch $d_i \sim \mathcal{D}_i$
    $\tilde{g}_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\tilde{\theta})}{\partial \tilde{\theta}}\right]$ //Grad wrt $\tilde{\theta}$
    Update $\tilde{\theta} \leftarrow \tilde{\theta} - \alpha \tilde{g}_i$
  **end for**

  $\boxed{\text{Update } \theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)}$
**end for**

$\approx$

**Algorithm 2** Direct optimization of IDGM.

**for** iterations $= 1, 2, \cdots$ **do**
  $\tilde{\theta} \leftarrow \theta$
  **for** $\mathcal{D}_i \in \texttt{permute}(\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\})$ **do**
    Sample batch $d_i \sim \mathcal{D}_i$
    $g_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\theta)}{\partial \theta}\right]$ //Grad wrt $\theta$

  **end for**
  $\bar{g} = \frac{1}{S}\sum_{s=1}^{S} g_s, \quad \hat{g} = \overbrace{\frac{2}{S(S-1)}\sum_{i,j\in S}^{i\neq j} g_i \cdot g_j}^{\text{GIP (batch)}}$
  Update $\theta \leftarrow \theta - \epsilon\left(\bar{g} - \gamma(\partial \hat{g}/\partial\theta)\right)$
**end for**

Conventionally, this type of algorithm is named by vertebrate name.
Fish algorithm is inspired by Reptile algorithm.

**Loss of IDGM**

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

$$\boxed{\begin{matrix} f \to L \\ x_i \to \theta_i \end{matrix}}$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad \text{and} \quad \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

**Gradient** **Hessian**

http://www.cs.albany.edu/~lsw/MLmaterials/calculus1.

# Key-point in Fish Derivation

**Algorithm 1** Fish.

**for** iterations $= 1, 2, \cdots$ **do**
  $\tilde{\theta} \leftarrow \theta$
  **for** $\mathcal{D}_i \in \texttt{permute}(\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\})$ **do**
    Sample batch $d_i \sim \mathcal{D}_i$
    $\tilde{g}_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\tilde{\theta})}{\partial \tilde{\theta}}\right]$ //Grad wrt $\tilde{\theta}$
    Update $\tilde{\theta} \leftarrow \tilde{\theta} - \alpha \tilde{g}_i$
  **end for**

  $\boxed{\text{Update } \theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)}$
**end for**

If meta step $\epsilon = 1$, ERM?
Yes, but it not Fish
Fish Condition
1) Infinite Inner loop
2) Only in small $\alpha$
If even one condition is violated, approximation is not established.

**Algorithm 2** Direct optimization of IDGM.

**for** iterations $= 1, 2, \cdots$ **do**
  $\tilde{\theta} \leftarrow \theta$
  **for** $\mathcal{D}_i \in \texttt{permute}(\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_S\})$ **do**
    Sample batch $d_i \sim \mathcal{D}_i$
    $g_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\theta)}{\partial \theta}\right]$ //Grad wrt $\theta$

  **end for**
  $\bar{g} = \frac{1}{S}\sum_{s=1}^{S} g_s, \quad \hat{g} = \overbrace{\frac{2}{S(S-1)}\sum_{i,j\in S}^{i\neq j} g_i \cdot g_j}^{\text{GIP (batch)}}$
  Update $\theta \leftarrow \theta - \epsilon\,(\bar{g} - \gamma(\partial \hat{g}/\partial\theta))$
**end for**

$\{d_i\}_{i=1}^{S}$, where $d_i := \{x_b, y_b\}_{b=1}^{B}$ denotes a minibatch at step $i$ randomly drawn from one of the available domains in $\{\mathcal{D}_1, \cdots, \mathcal{D}_S\}$.

$\bar{G} = \frac{1}{S}\sum_{s=1}^{S} G_s$

$$G_f = \mathbb{E}[(\theta - \tilde{\theta})] - \alpha S \cdot \bar{G}, \quad \textit{Fish update - } \alpha S \cdot \textit{ERM grad}$$

$$G_g = -\partial \hat{G}/\partial\theta, \quad \textit{grad of } \max_{\theta}(\hat{G})$$

$$= -\frac{2}{S(S-1)}\sum_{i,j\in S}^{i\neq j}\frac{\partial}{\partial\theta} G_i \cdot G_j$$

$$\tilde{g}_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\theta_i)}{\partial \theta_i}\right] \quad \text{(gradient at step } i\text{, wrt } \theta_i)$$

$$\theta_{i+1} = \theta_i - \alpha \tilde{g}_i \quad \text{(sequence of parameters)}$$

$$g_i = \mathbb{E}_{d_i}\left[\frac{\partial l((x,y);\theta_1)}{\partial \theta_1}\right] \quad \text{(gradient at step } i\text{, wrt } \theta_1)$$

$$H_i = \mathbb{E}_{d_i}\left[\frac{\partial^2 l((x,y);\theta_1)}{\partial \theta_1^2}\right] \quad \text{(Hessian at } \boxed{\text{initial point}})$$

Perform **Taylor Approximation** for $\tilde{g}_i$ near $\theta_1$.

$$\tilde{g}_i = l_i'(\theta_i)$$

$$= l_i'(\theta_1) + l_i''(\theta_1)(\theta_i - \theta_1) + \boxed{\underbrace{\mathcal{O}(\|\theta_i - \theta_1\|^2)}_{=\mathcal{O}(\alpha^2)}}$$

$$= g_i + H_i(\theta_i - \theta_1) + \mathcal{O}(\alpha^2)$$

$$= g_i - \alpha H_i \sum_{j=1}^{i-1}\tilde{g}_j + \mathcal{O}(\alpha^2).$$ Ignore from **second-order**

$$\tilde{g}_j = g_j + \mathcal{O}(\alpha)$$ Ignore from **first-order**

let us consider performing two steps in inner-loop updates, i.e. $S = 2$.

$$\theta - \tilde{\theta} = \alpha(\tilde{g}_1 + \tilde{g}_2)$$

$$= \alpha\underbrace{(g_1 + g_2)}_{①} - \alpha^2 \underbrace{H_2 g_1}_{②} + \mathcal{O}(\alpha^3)$$

$$① = \mathbb{E}_{1,2}\left[g_1 + g_2\right] = G_1 + G_2$$

$$② = \mathbb{E}_{1,2}\left[H_2 g_1\right] = \mathbb{E}_{1,2}\left[H_1 g_2\right] \quad \text{(interchanging indices)}$$

$$= \frac{1}{2}\mathbb{E}_{1,2}\left[H_2 g_1 + H_1 g_2\right] \quad \text{(averaging last two eqs)}$$

$$= \frac{1}{2}\mathbb{E}_{1,2}\left[\frac{\partial(g_1 \cdot g_2)}{\partial\theta_1}\right]$$

$$= \frac{1}{2}\cdot\frac{\partial(G_1 \cdot G_2)}{\partial\theta_1}$$

$$\mathbb{E}[\theta - \tilde{\theta}] = \alpha(G_1 + G_2) + \frac{\alpha^2}{2}\cdot\frac{\partial(G_1 \cdot G_2)}{\partial\theta_1} + \mathcal{O}(\alpha^3)$$

We can also expand this to the general case where $S \geq 2$:

$$\mathbb{E}[\theta - \tilde{\theta}]$$

$$= \alpha\sum_{s=1}^{S} G_s - \frac{\alpha^2}{S(S-1)}\sum_{i,j\in S}^{i\neq j}\frac{\partial(G_i \cdot G_j)}{\partial\theta_1} + \mathcal{O}(\alpha^3).$$

$$\bar{G} = \frac{1}{S}\sum_{s=1}^{S} G_s.$$

$$G_f = \mathbb{E}[\theta - \tilde{\theta}] - \alpha S \bar{G}$$

$$= -\frac{\alpha^2}{S(S-1)}\sum_{i,j\in S}^{i\neq j}\frac{\partial}{\partial\theta_1} G_i \cdot G_j$$

$$\boxed{\lim_{\alpha\to 0}\frac{G_f \cdot G_g}{\|G_f\| \cdot \|G_g\|} = 1.}$$ **Cauchy-Schwartz Inequality**

$$\boxed{\therefore G_f \approx k G_g}$$ **Almost Same Direction**

# Summary and Recommend References

- Domain Generalization
  - Domain Generalization VS Subpopulation Shift
  - Wilds 1.2v Benchmark
  - Algorithms for Domain Generalization
  - DomainBed Benchmark, Model Selection Method
- Gradient Matching for Domain Generalization
  - IDGM – Inter Domain Gradient Matching
  - IDGM and Fish Algorithms
  - Derivation IDGM → Fish