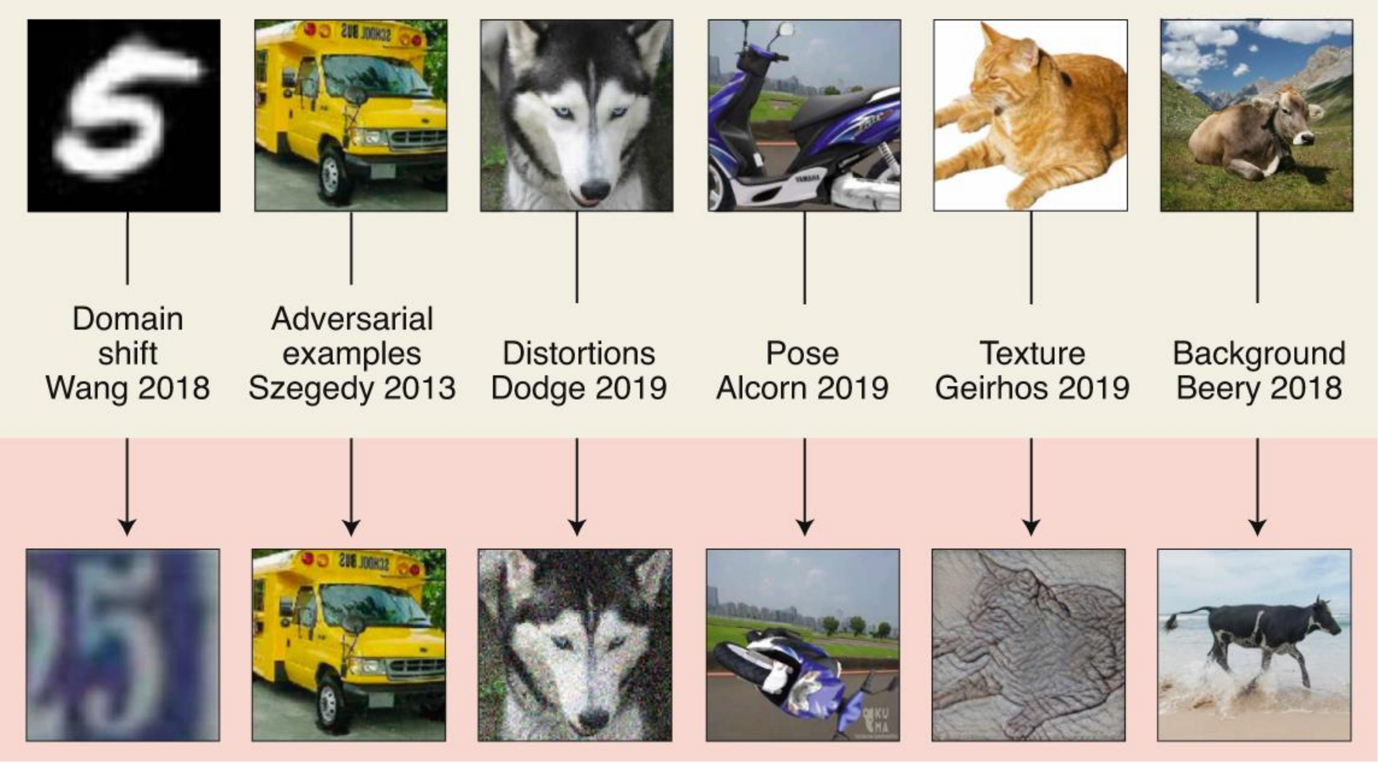
Selective Mixup Helps with Distribution Shifts, But Not (Only) because of Mixup

Damien Teney — Idiap Research Institute
Jindong Wang — Microsoft Research Asia
Ehsan Abbasnejad — University of Adelaide

We want ML models that generalize

Most models are not robust to distribution shifts.



Training data

Out-of-distribution test data

Selective mixup

- Popular class of methods
- Improve generalization with distribution shifts
  (consistent improvements on WILDS & Wild-Time benchmarks)

Yao et al., **Improving out-of-distribution robustness via selective augmentation (LISA)**, ICLR 2022
Hwang et al., **Selecmix: Debiased learning by contradicting-pair sampling**, NeurIPS 2022
Li et al. **Are data-driven explanations robust against out-of-distribution data?**, 2023
Lu et al. **Semantic discriminative mixup for generalizable sensor-based cross-domain activity recognition**, 2022
Palakkadavath et al., **Improving domain generalization with interpolation robustness**, NeurIPS DistShift 2022
Tian et al., **Cifair: Constructing continuous domains of invariant features for image fair classifications**. KBS, 2023
Xu et al., **Adversarial domain adaptation with domain mixup**, AAAI 2020

- It does work, but not (only) because of mixup!
- It implicitly resamples the training data

How? Why does it help?
Why did prior work miss it? How did we find out?

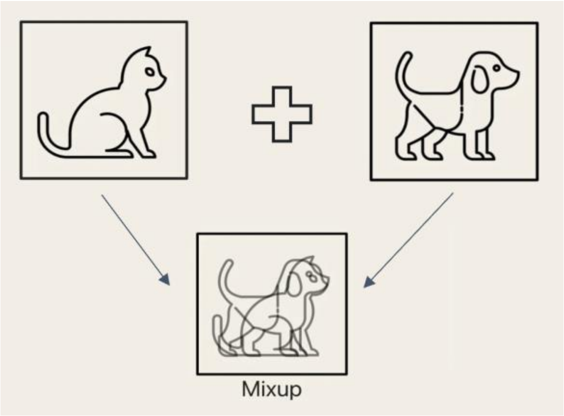
A bit of background: classical mixup

- Standard training: $\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})$

  Model f , training example x , label y , loss $\mathcal{L}$

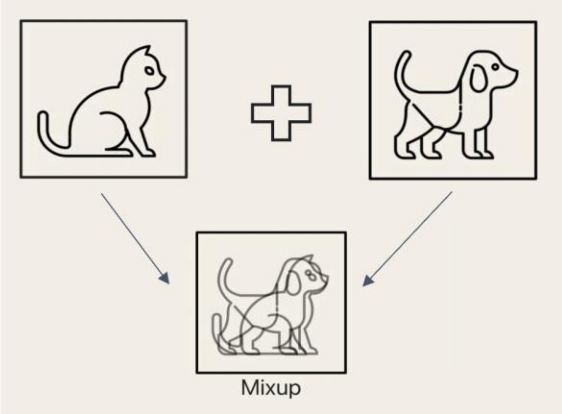
- Training with mixup: $\mathcal{L}(f(c\,\boldsymbol{x}+(1-c)\tilde{\boldsymbol{x}}), c\,\boldsymbol{y}+(1-c)\,\tilde{\boldsymbol{y}})$

  Mixing coefficient c (random or 0.5), paired examples $(\boldsymbol{x}, \boldsymbol{y})$ and $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ — Picked at random



Zhang et al., mixup: Beyond empirical risk minimization, 2017

A bit of background: classical mixup



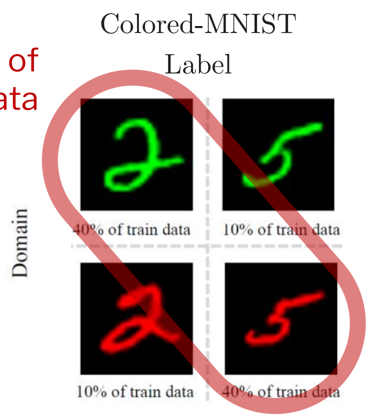
Improves generalization (even without distribution shifts)
Very often. Not always. It rarely hurts.

Why does it work? 🤔
Regularization, augmentation,
introduces label noise,
helps learn rare features, ...

Selective mixup

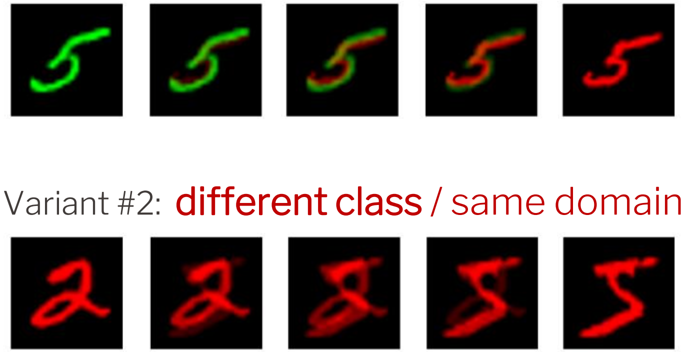
- Idea: applying mixup on selected pairs, according to some criterion
- Many variants! Focus on LISA
- For data with domain labels: collected in different places, periods of time, ...

Most of tr. data



Variant #1: same class / **different domain**

Variant #2: **different class** / same domain



Sometimes one works, sometimes the other 🤷

Yao et al., Improving out-of-distribution robustness via selective augmentation, ICLR 2022

Key insight:
Selective mixup implicitly resamples the data

With binary classification, it **perfectly** balances the classes!

Class A	Class B
75%	25%

1. Sample from original distribution	A A A B
2. Get pairs with “different class” criterion	B B B A

Identical proportions in aggregate!

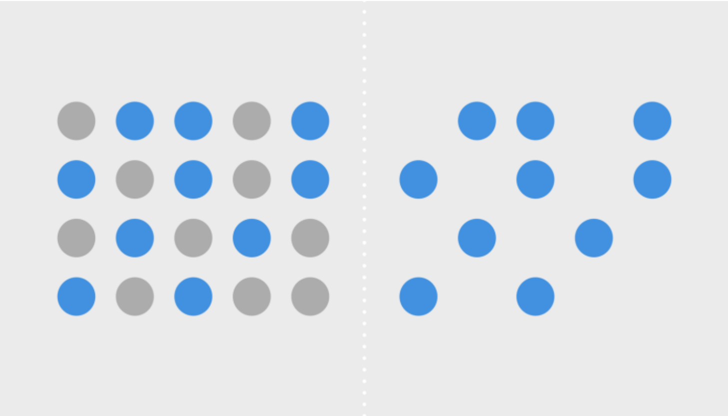
Key insight:
Selective mixup implicitly resamples the data

- In general: makes distributions of features/classes more uniform ("regression towards the mean")
- Resampling/reweighting is a known baseline for label shift / imbalanced data
- Two unrelated methods are actually doing the same thing!

Why was this missed in prior work?

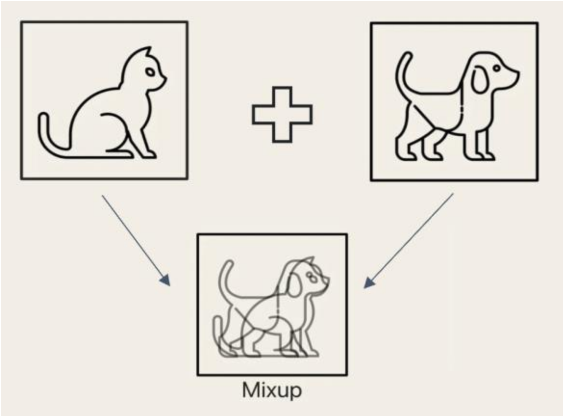
- The missing ablation

Step 1: select pairs



```
(x1, x1')
(x2, x2')
(x3, x3') …
```

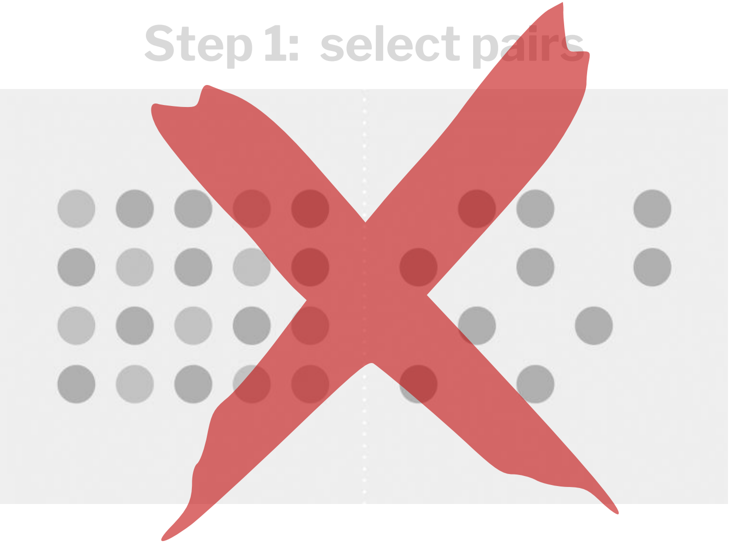
Step 2: mix them



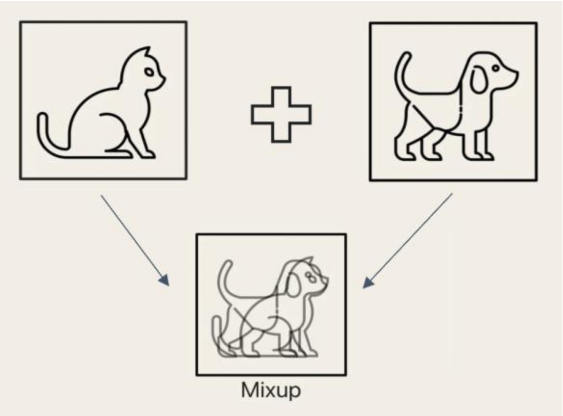
Training data = { mix(x1, x1'),
mix(x2, x2'),
mix(x3, x3'), … }

Why was this missed in prior work?

Step 1: select pairs



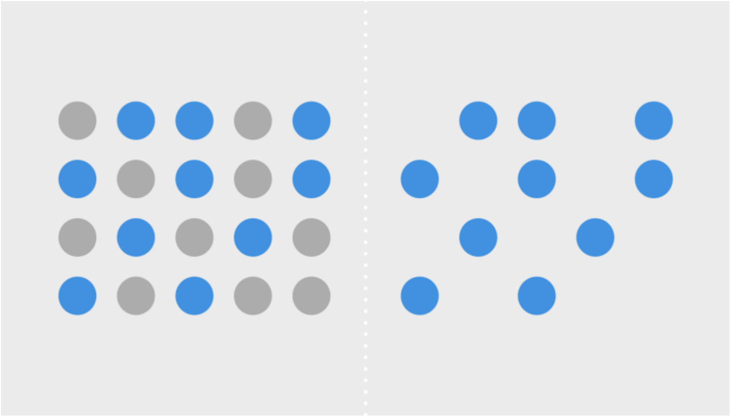
Step 2: mix them



Vanilla mixup

Why was this missed in prior work?

Step 1: select pairs



(x1, x1')
(x2, x2')
(x3, x3') …

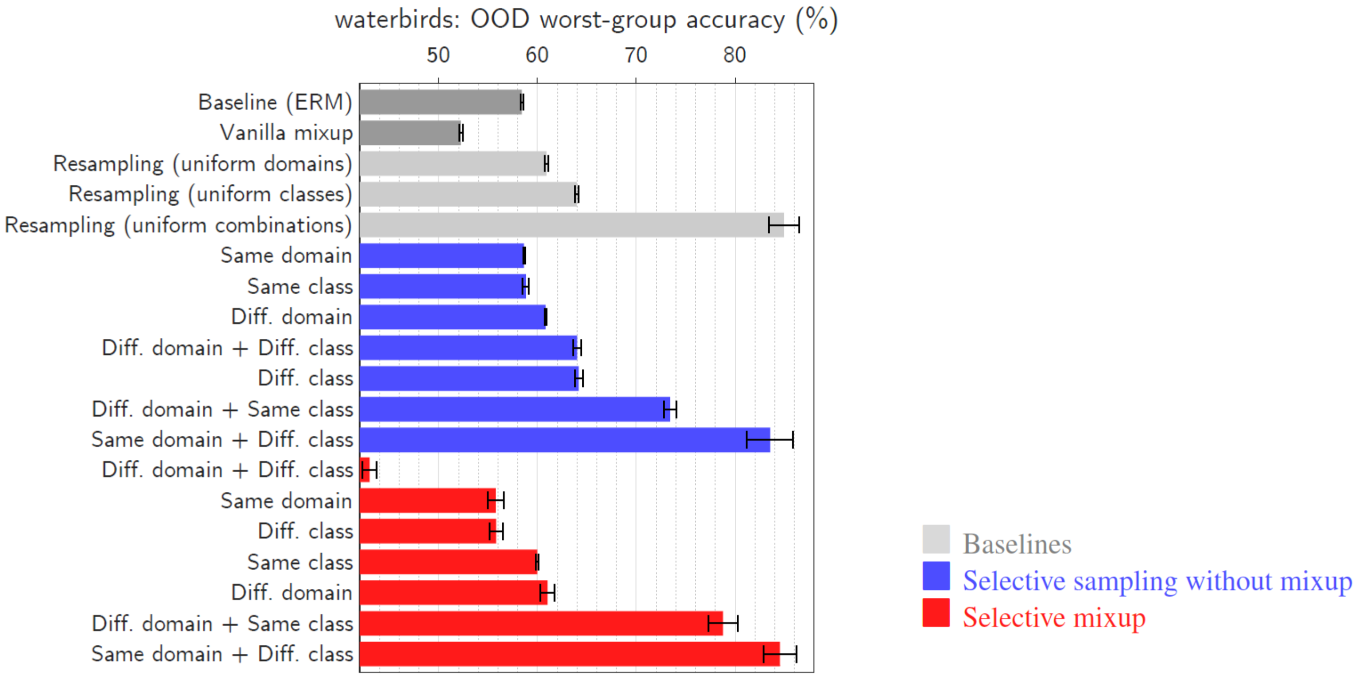
Step 2: mix them

{ x1, x1', x2, x2', x3, x3', … }

- Missing ablation: build mini-batches with the sampled pairs, but **no mixing**!

Empirical verification

- Needs experiments: we can't predict when the mixing helps
- Overall effects: sum of **vanilla mixup** + **resampling**
- Sometimes the mixing is detrimental: the resampling alone is better!



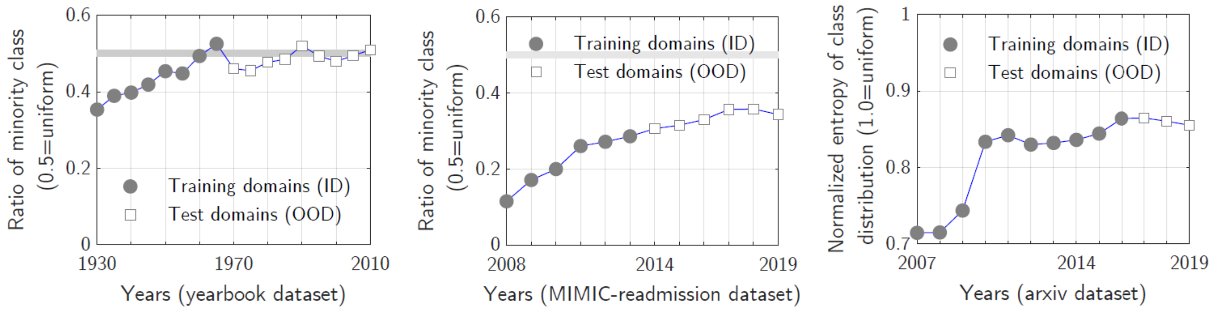
Empirical verification

- Needs experiments: we can’t predict when the mixing helps
- Overall effects: sum of **vanilla mixup** + **resampling**
- Sometimes the mixing is detrimental: the resampling alone is better!

- With most datasets, the story is not so clear (mixup <u>does help</u> sometimes!)

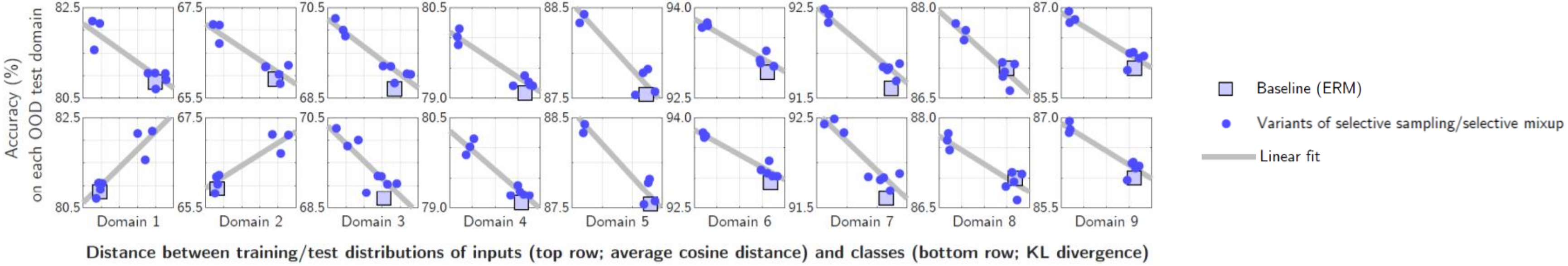
Testable predictions from the resampling effect

Resampling is beneficial when there is a “regression towards the mean”



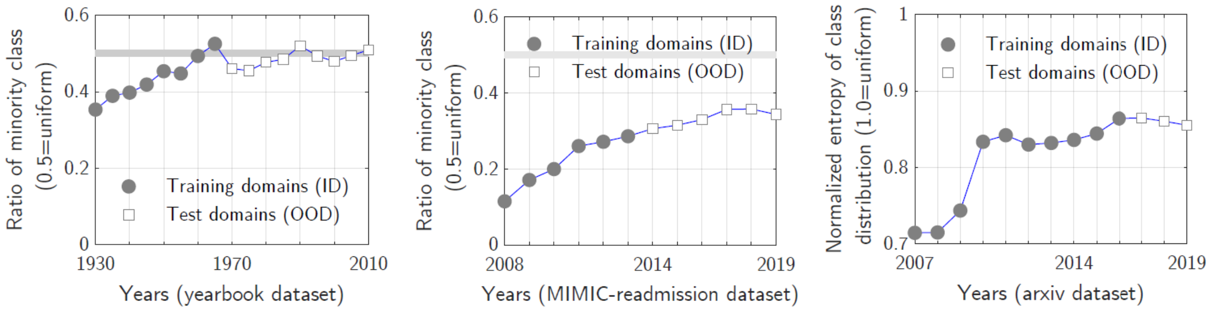
Class distribution trending towards uniformity (0.5) in the Wild-Time benchmark

Improvements correlate with the training/test distributions getting closer



Testable predictions from the resampling effect

Resampling is beneficial when there is a “regression towards the mean”



Class distribution trending towards uniformity (0.5) in the Wild-Time benchmark

Accidental property of existing datasets? Risk of overfitting to the benchmarks!

This predicts a new failure mode

- Detrimental effect if there's a "regression **away from** mean"
- Previously unknown limitation of selective mixup

- Verification: swapping training / test splits
- Indeed, good methods are now bad

Behind the paper

! **Accidental finding** from a different project
New method, meta learning mixup sampling/mixing coefficients
This finding was more interesting!

Performed on a single laptop: shallow MLPs, cached pretrained features

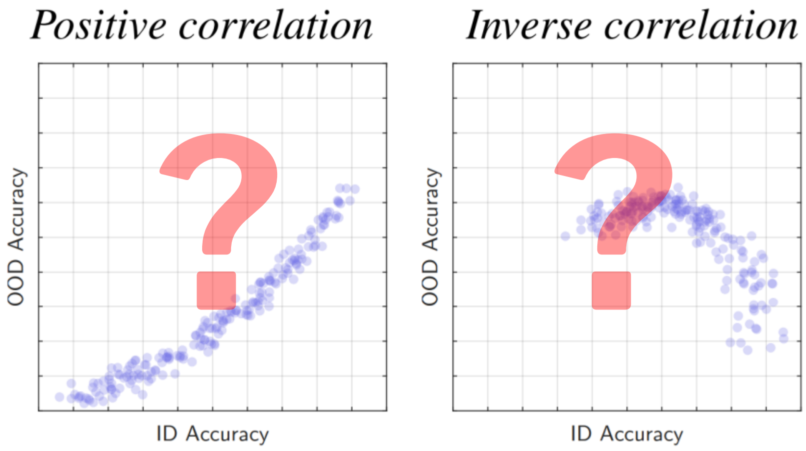
✕ **Rejected from NeurIPS** *"no new method", "only an 'insights' paper"*
Why is it interesting?

- It corrects previous (incomplete) explanations
- It connect two areas of the literature: selective mixup / resampling
- In some cases, we found better combinations of the two

ID vs. OOD performance

Testing models/methods in- & out-of-distribution (2 test sets)

Is ID performance a good proxy for OOD generalization?



Important for reliability & model selection

Purely an empirical question
(both can happen in principle)

ID & OOD performance always correlated?!

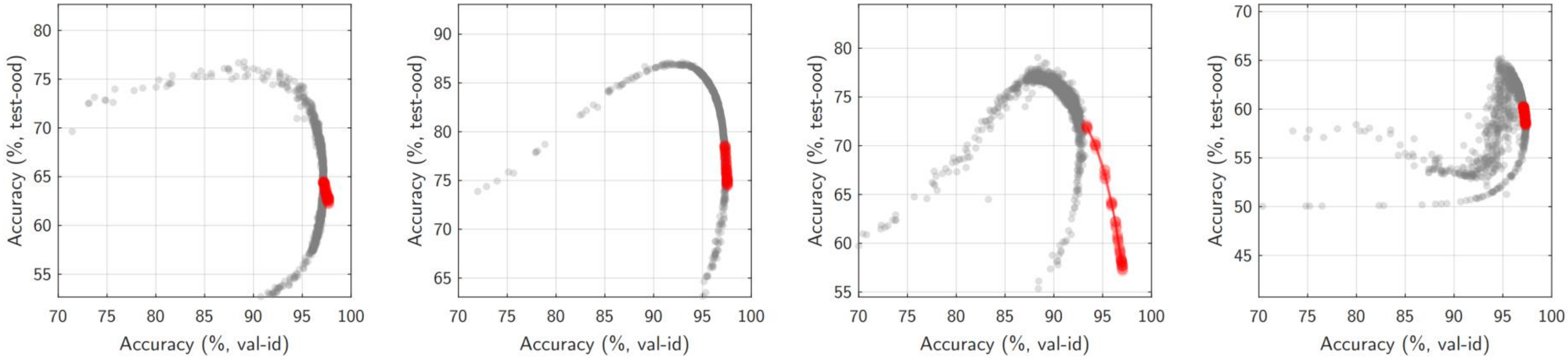
Common finding/claim in the literature

Miller et al., Accuracy on the line: on the strong correlation between OOD and ID generalization, ICML 2021

Wenzel et al., Assaying out-of-distribution generalization in transfer learning, NeurIPS 2022

Angarano et al., Back-to-bones: Rediscovering the role of backbones in domain generalization, 2022

But it doesn't match our observations!

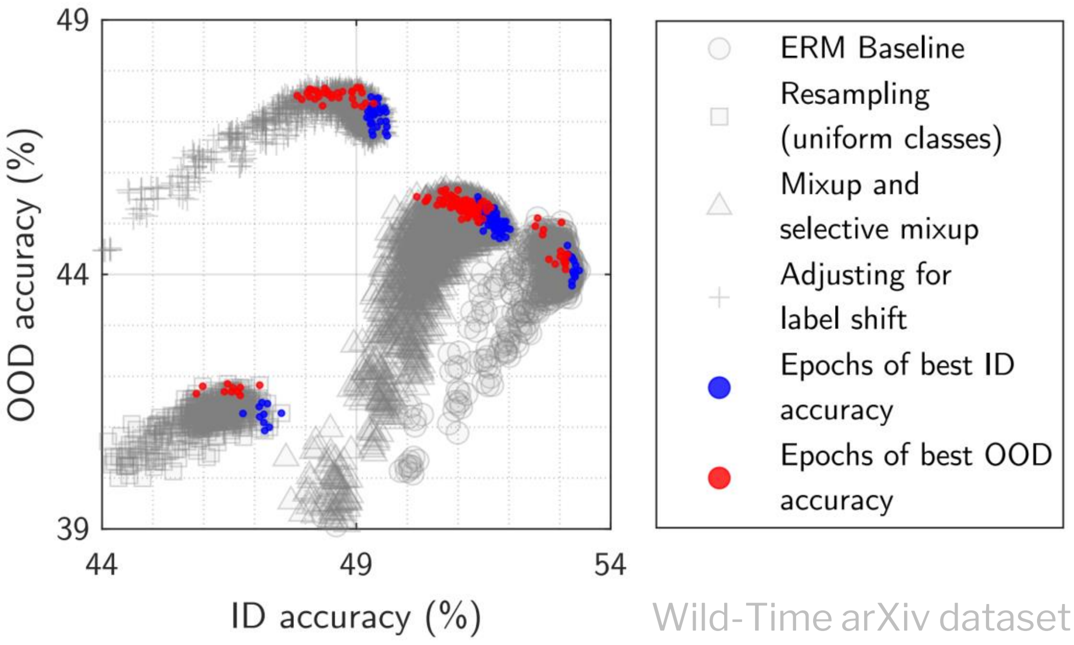


WILDS-camelyon dataset; 1 point = 1 model; various seeds & numbers of epochs; ●: trained with ERM; ●: trained with diversity regularizer

More funny ID/OOD correlations

Inverse correlations across methods

and within each method (different seeds, hyperparameters, number of epochs)



Wild-Time arXiv dataset

Why did prior work miss this?

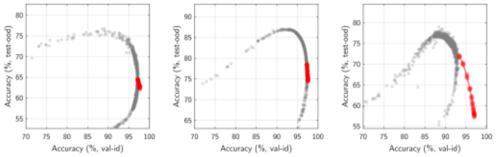
Methodology of most studies:

1. Train models
2. Early stopping/model selection for best ID perf.
3. Analyze only the selected models

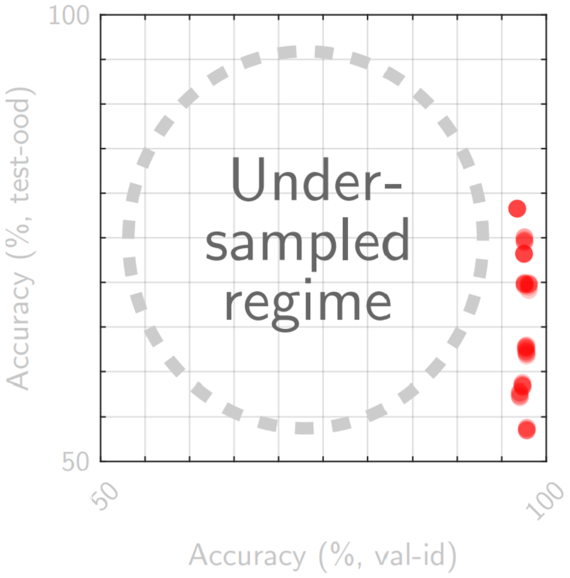
Valid when ID/OOD are correlated.
The very thing we want to check!

Excludes a lot of data!

Our observations:



What prior studies would have observed:



This completely misses the inverse correlations!

Teney et al., ID and OOD Performance Are Sometimes Inversely Correlated on Real-world Datasets, NeurIPS 2023

Implications for OOD generalization

- High OOD performance sometimes requires trading off ID performance.
- Improving ID perf. alone may produce diminishing/negative returns OOD.
- Model selection using ID performance will miss the best OOD models.
- Important to track multiple metrics (seems common now).

High-level take-aways

Plenty of room for **scientific inquiry** of existing methods

- Even established ones
- Even on a small scale
- No pressure to beat the SOTA

Methodological practices

- Question the assumptions
- Everyone does it $\not\Rightarrow$ It's the right thing to do
- Lookout for "overfitting to the benchmarks"

Teney et al., On the value of out-of-distribution testing: An example of Goodhart's law, NeurIPS 2020

↳ Normalized gaming of a benchmark for visual question answering