
Welcome to DomainBed

DomainBed is a PyTorch suite containing benchmark datasets and algorithms for domain generalization, as introduced in In Search of Lost Domain Generalization.

Current results

Algorithm	CMNIST	RMNIST	VLCS	PACS	OfficeHome	TerraInc	DomainNet	Average
ERM	51.5 \pm 0.1	98.0 \pm 0.0	77.5 \pm 0.4	85.5 \pm 0.2	66.5 \pm 0.3	46.1 \pm 1.8	40.9 \pm 0.1	66.6
IRM	52.0 \pm 0.1	97.7 \pm 0.1	78.5 \pm 0.5	83.5 \pm 0.8	64.3 \pm 2.2	47.6 \pm 0.8	33.9 \pm 2.8	65.4
GroupDRO	52.1 \pm 0.0	98.0 \pm 0.0	76.7 \pm 0.6	84.4 \pm 0.8	66.0 \pm 0.7	43.2 \pm 1.1	33.3 \pm 0.2	64.8
Mixup	52.1 \pm 0.2	98.0 \pm 0.1	77.4 \pm 0.6	84.6 \pm 0.6	68.1 \pm 0.3	47.9 \pm 0.8	39.2 \pm 0.1	66.7
MLDG	51.5 \pm 0.1	97.9 \pm 0.0	77.2 \pm 0.4	84.9 \pm 1.0	66.8 \pm 0.6	47.7 \pm 0.9	41.2 \pm 0.1	66.7
CORAL	51.5 \pm 0.1	98.0 \pm 0.1	78.8 \pm 0.6	86.2 \pm 0.3	68.7 \pm 0.3	47.6 \pm 1.0	41.5 \pm 0.1	67.5
MMD	51.5 \pm 0.2	97.9 \pm 0.0	77.5 \pm 0.9	84.6 \pm 0.5	66.3 \pm 0.1	42.2 \pm 1.6	23.4 \pm 9.5	63.3
DANN	51.5 \pm 0.3	97.8 \pm 0.1	78.6 \pm 0.4	83.6 \pm 0.4	65.9 \pm 0.6	46.7 \pm 0.5	38.3 \pm 0.1	66.1
CDANN	51.7 \pm 0.1	97.9 \pm 0.1	77.5 \pm 0.1	82.6 \pm 0.9	65.8 \pm 1.3	45.8 \pm 1.6	38.3 \pm 0.3	65.6
MTL	51.4 \pm 0.1	97.9 \pm 0.0	77.2 \pm 0.4	84.6 \pm 0.5	66.4 \pm 0.5	45.6 \pm 1.2	40.6 \pm 0.1	66.2
SagNet	51.7 \pm 0.0	98.0 \pm 0.0	77.8 \pm 0.5	86.3 \pm 0.2	68.1 \pm 0.1	48.6 \pm 1.0	40.3 \pm 0.1	67.2
ARM	56.2 \pm 0.2	98.2 \pm 0.1	77.6 \pm 0.3	85.1 \pm 0.4	64.8 \pm 0.3	45.5 \pm 0.3	35.5 \pm 0.2	66.1
VREx	51.8 \pm 0.1	97.9 \pm 0.1	78.3 \pm 0.2	84.9 \pm 0.6	66.4 \pm 0.6	46.4 \pm 0.6	33.6 \pm 2.9	65.6
RSC	51.7 \pm 0.2	97.6 \pm 0.1	77.1 \pm 0.5	85.2 \pm 0.9	65.5 \pm 0.9	46.6 \pm 1.0	38.9 \pm 0.5	66.1

Model selection: training-domain validation set

Full results for commit 7df6f06 in LaTeX format available [here](#).

Available algorithms

The currently available algorithms are:

- Empirical Risk Minimization (ERM, Vapnik, 1998)
- Invariant Risk Minimization (IRM, Arjovsky et al., 2019)
- Group Distributionally Robust Optimization (GroupDRO, Sagawa et al., 2020)
- Interdomain Mixup (Mixup, Yan et al., 2020)
- Marginal Transfer Learning (MTL, Blanchard et al., 2011-2020)
- Meta Learning Domain Generalization (MLDG, Li et al., 2017)
- Maximum Mean Discrepancy (MMD, Li et al., 2018)
- Deep CORAL (CORAL, Sun and Saenko, 2016)
- Domain Adversarial Neural Network (DANN, Ganin et al., 2015)
- Conditional Domain Adversarial Neural Network (CDANN, Li et al., 2018)
- Style Agnostic Networks (SagNet, Nam et al., 2020)
- Adaptive Risk Minimization (ARM, Zhang et al., 2020), contributed by @zhangmarvin
- Variance Risk Extrapolation (VREx, Krueger et al., 2020), contributed by @zdhNarsil

-
- Representation Self-Challenging (RSC, Huang et al., 2020), contributed by @SirRob1997
 - Spectral Decoupling (SD, Pezeshki et al., 2020)
 - Learning Explanations that are Hard to Vary (AND-Mask, Parascandolo et al., 2020)
 - Out-of-Distribution Generalization with Maximal Invariant Predictor (IGA, Koyama et al., 2020)
 - Gradient Matching for Domain Generalization (Fish, Shi et al., 2021)
 - Self-supervised Contrastive Regularization (SelfReg, Kim et al., 2021)
 - Smoothed-AND mask (SAND-mask, Shahtalebi et al., 2021)
 - Invariant Gradient Variances for Out-of-distribution Generalization (Fishr, Rame et al., 2021)
 - Learning Representations that Support Robust Transfer of Predictors (TRM, Xu et al., 2021)
 - Invariance Principle Meets Information Bottleneck for Out-of-Distribution Generalization (IB-ERM, Ahuja et al., 2021)
 - Invariance Principle Meets Information Bottleneck for Out-of-Distribution Generalization (IB-IRM, Ahuja et al., 2021)
 - Optimal Representations for Covariate Shift (CAD & CondCAD, Ruan et al., 2022), contributed by @ryoungj
 - Quantifying and Improving Transferability in Domain Generalization (Transfer, Zhang et al., 2021), contributed by @Gordon-Guojun-Zhang
 - Invariant Causal Mechanisms through Distribution Matching (CausIRL with CORAL or MMD, Chevalley et al., 2022), contributed by @MathieuChevalley
 - Empirical Quantile Risk Minimization (EQRM, Eastwood et al., 2022), contributed by @cianeastwood

Send us a PR to add your algorithm! Our implementations use ResNet50 / ResNet18 networks (He et al., 2015) and the hyper-parameter grids described here.

Available datasets

The currently available datasets are:

- RotatedMNIST (Ghifary et al., 2015)
- ColoredMNIST (Arjovsky et al., 2019)
- VLCS (Fang et al., 2013)
- PACS (Li et al., 2017)
- Office-Home (Venkateswara et al., 2017)
- A TerraIncognita (Beery et al., 2018) subset
- DomainNet (Peng et al., 2019)
- A SVIRO (Dias Da Cruz et al., 2020) subset
- WILDS (Koh et al., 2020) FMoW (Christie et al., 2018) about satellite images
- WILDS (Koh et al., 2020) Camelyon17 (Bandi et al., 2019) about tumor detection in tissues

-
- Spawrious (Lynch et al., 2023)

Send us a PR to add your dataset! Any custom image dataset with folder structure `dataset/domain/class/image.xyz` is readily usable. While we include some datasets from the WILDS project, please use their official code if you wish to participate in their leaderboard.

Available model selection criteria

Model selection criteria differ in what data is used to choose the best hyper-parameters for a given model:

- `IIDAccuracySelectionMethod`: A random subset from the data of the training domains.
- `LeaveOneOutSelectionMethod`: A random subset from the data of a held-out (not training, not testing) domain.
- `OracleSelectionMethod`: A random subset from the data of the test domain.

Quick start

Download the datasets:

```
1 python3 -m domainbed.scripts.download \  
2     --data_dir=./domainbed/data
```

Train a model:

```
1 python3 -m domainbed.scripts.train\  
2     --data_dir=./domainbed/data/MNIST\  
3     --algorithm IGA\  
4     --dataset ColoredMNIST\  
5     --test_env 2
```

Launch a sweep:

```
1 python -m domainbed.scripts.sweep launch\  
2     --data_dir=/my/datasets/path\  
3     --output_dir=/my/sweep/output/path\  
4     --command_launcher MyLauncher
```

Here, `MyLauncher` is your cluster's command launcher, as implemented in `command_launchers.py`. At the time of writing, the entire sweep trains tens of thousands of models (all algorithms x all datasets x 3 independent trials x 20 random hyper-parameter choices). You can pass arguments to make the sweep smaller:

```
1 python -m domainbed.scripts.sweep launch\  
2     --data_dir=/my/datasets/path\  
3     --output_dir=/my/sweep/output/path\  
4     --command_launcher MyLauncher\  
5     --algorithms ERM DANN\  
6     --datasets RotatedMNIST VLCS\  
7     --n_hparams 5\  
8     --n_trials 1
```

After all jobs have either succeeded or failed, you can delete the data from failed jobs with `python -m domainbed.scripts.sweep delete_incomplete` and then re-launch them by running `python -m domainbed.scripts.sweep launch` again. Specify the same command-line arguments in all calls to `sweep` as you did the first time; this is how the sweep script knows which jobs were launched originally.

To view the results of your sweep:

```
1 python -m domainbed.scripts.collect_results\  
2     --input_dir=/my/sweep/output/path
```

Running unit tests

DomainBed includes some unit tests and end-to-end tests. While not exhaustive, but they are a good sanity-check. To run the tests:

```
1 python -m unittest discover
```

By default, this only runs tests which don't depend on a dataset directory. To run those tests as well:

```
1 DATA_DIR=/my/datasets/path python -m unittest discover
```

License

This source code is released under the MIT license, included [here](#).