# A fine-grained analysis of distribution shifts

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Robustness to distribution shifts is critical for deploying machine learning models in the real world. Despite this necessity, there has been little work in defining the underlying mechanisms that cause these shifts and evaluating the robustness of algorithms across multiple, different distribution shifts. To this end, we introduce a framework that enables fine-grained analysis of various distribution shifts. We provide a holistic analysis of current state-of-the-art methods by evaluating 19 distinct methods grouped into five categories across both synthetic and real-world datasets. Overall, we train more than 85K models. Our experimental framework can be easily extended to include new methods, shifts, and datasets. We find, unlike previous work [Gulrajani and Lopez-Paz, 2021], that progress has been made over a standard ERM baseline; in particular, pretraining and augmentations (learned or heuristic) offer large gains in many cases. However, the best methods are not consistent over different datasets and shifts.

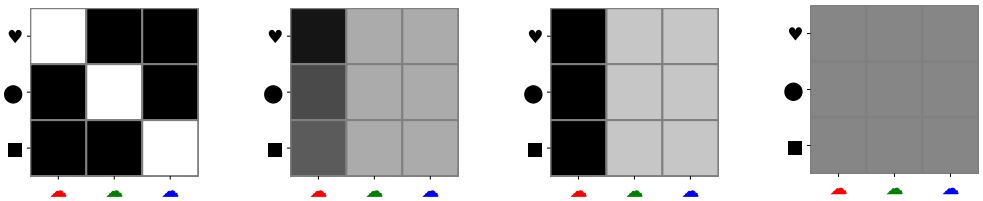
## 1 Introduction

If machine learning models are to be ubiquitous in critical applications such as driverless cars [Janai et al., 2020], medical imaging [Erickson et al., 2017], and science [Jumper et al., 2021], it is pivotal to build models that are robust to distribution shifts. Otherwise, models may fail surprisingly in ways that derail trust in the system. For example, Koh et al. [2020], Perone et al. [2019], AlBadawy et al. [2018], Heaven [2020], Castro et al. [2020] find that a model trained on one set of hospitals may not generalise to the imaging conditions of another; Alcorn et al. [2019], Dai and Van Gool [2018] find that a model for driverless cars may not generalise to new lighting conditions or object poses; and Buolamwini and Gebru [2018] find that a model may perform worse on subsets of the distribution, such as different ethnicities, if the training set has an imbalanced distribution. Thus, it is important to understand when we expect a model to generalise and when we do not. This would allow a practitioner to have confidence in the system (e.g. if a model is demonstrated to be robust to the imaging conditions of different hospitals, then it can be deployed in new hospitals with confidence).

While domain generalization is a well studied area, Gulrajani and Lopez-Paz [2021], Schott et al. [2021] have cast doubt on the efficacy of existing methods, raising the question: *has any progress been made in domain generalization over a standard expectation risk minimization (ERM) algorithm?* Despite these discouraging results, there are many examples that machine learning models *do* generalise across datasets with different distributions [Radford et al., 2021, Taori et al., 2020]. However, there is little understanding on *when* and *why* models generalise, especially in realistic settings inspired by real-world applications. To this end we introduce an evaluation framework to systematically evaluate the robustness of different methods under different distribution shifts.

Using this framework, we evaluate models across three distribution shifts: *spurious correlation*, *low data drift*, and *unseen data shift* (illustrated in figure 1). We choose these settings as they arise in the real world and harm generalization performance. The unique ability of our framework to evaluate

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) $p_{\text{train}}$: **SC.** (b) $p_{\text{train}}$: **LDD.** (c) $p_{\text{train}}$: **UDS.** (d) $p_{\text{test}}$: $y^l, y^a$ **are IID.**

Figure 1: Visualization of the joint distribution for the different shifts we consider on the DSPRITES example. The lighter the color, the more likely the given sample. figure 1a-1c visualise different shifts: *spurious correlation* (SC), *low data drift* (LDD), and *unseen data shift* (UDS). figure 1d visualises the test set, where the attributes are uniformly distributed.

fine-grained performance of models across different distribution shifts and under different conditions is of critical importance to analyze methods under a variety of real-world settings.

## 2 Models evaluated

We evaluate 19 algorithms to cover a broad range of approaches that can be used to improve model robustness to distribution shifts. These algorithms cover the following areas: architecture choice, data augmentation, domain adaptation, adaptive approaches and representation learning. Further discussion and how these models relate to our robustness framework is in appendix C.

**Architecture choice.** We evaluate the following standard vision models: ResNet18, ResNet50, ResNet101 [He et al., 2016], ViT [Dosovitskiy et al., 2021], and an MLP [Vapnik, 1992].

**Heuristic data augmentation.** We analyze the following augmentation methods: standard ImageNet augmentation [He et al., 2016], AugMix without JSD [Hendrycks et al., 2020], RandAugment [Cubuk et al., 2020], and AutoAugment [Cubuk et al., 2019].

**Learned data augmentation.** These approaches learn how to augment an image according to auxiliary information (e.g. given a blue shape and the color red, the learned augmentation will turn the shape red). We follow Goel et al. [2020], who use CYCLEGAN [Zhu et al., 2017], but we do not use their SGDRO objective in order to evaluate the performance of learned data augmentation alone.

**Domain generalization.** We evaluate IRM [Arjovsky et al., 2019], DeepCORAL [Sun and Saenko, 2016], domain MixUp [Gulrajani and Lopez-Paz, 2021], DANN [Ganin et al., 2016], and SagNet [Nam et al., 2021].

**Adaptive approaches.** We evaluate JTT [Liu et al., 2021] and BN-Adapt [Schneider et al., 2020].

**Representation learning.** We evaluate using a $\beta$-VAE [Higgins et al., 2017a] and pretraining on ImageNet [Deng et al., 2009].

## 3 Experiments

We evaluate the 19 different methods across six datasets and three distribution shifts. We plot aggregate results in figures 2-4 and complete results in the appendix in figures 7-9. We discuss the results by distilling them into six concrete takeaways in section 3.1 and four practical tips in section 3.2. Implementation details are given in appendix H.

**Datasets.** We use six vision, classification datasets – DSPRITES [Matthey et al., 2017], MPI3D [Gondal et al., 2019], SMALLNORB [LeCun et al., 2004], SHAPES3D [Burgess and Kim, 2018], CAMELYON17 [Koh et al., 2020, Bandi et al., 2018], and IWILDCAM [Koh et al., 2020, Beery et al., 2018]. We discuss precisely how we set up the shifts for these datasets in appendix G.2.

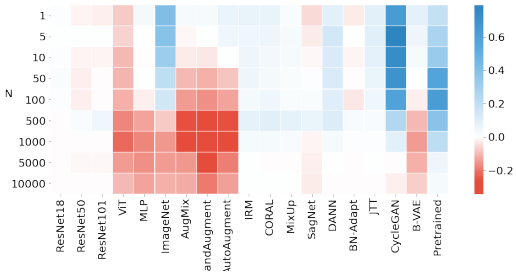



Figure 2: **Spurious Correlation.** We use all correlated samples and vary the number of samples $N$ from the true, uncorrelated distribution. We plot the percentage change over the baseline ResNet, averaged over all seeds and datasets. Blue is better, red worse. CycleGAN performs consistently best while ImageNet augmentation and pretraining on ImageNet also consistently boost performance.

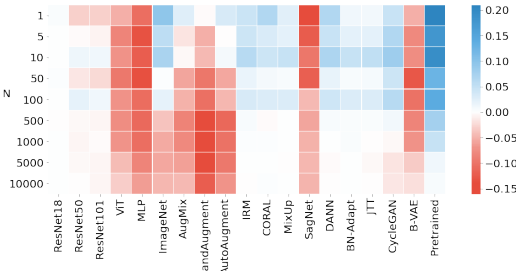



Figure 3: **Low data drift.** We use all samples from the high data regions and vary the number of samples $N$ from the low data region. We plot the percentage change over the baseline ResNet, averaged over all seeds and datasets. Blue is better, red worse. Pretraining on ImageNet performs consistently best, while CycleGAN, most domain adaptation methods and ImageNet augmentation also provide some boost in performance.

### 3.1 Takeaways

**Takeaway 1: While we can improve over ERM, no one method always performs best.** The relative performance between methods varies across datasets and shifts. Under *spurious correlation* (figure 2), CycleGAN consistently performs best but in figure 3, under *low-data drift*, pretraining consistently performs best. Under *unseen data shift* (figure 4), pretraining is again one of the best performing models. However, if we drill down on the results in figure 7, we can see pretraining performs best on the synthetic datasets, but not on Camelyon17 (where using augmentation or DANN is best) or iWildCam (where using ViT or an MLP is best).

**Takeaway 2: Pretraining is a powerful tool across different shifts and datasets.** While pretraining is not always helpful (e.g. on Camelyon17 in figures 7-9, iWildCam in figures 7-8), it often provides a strong boost in performance. This is presumably because the representation $z$ learned during pretraining is helpful for the downstream task. For example, the representation may have been trained to be invariant to certain useful properties (e.g. scale, shift, and color). If these properties are useful on the downstream tasks, then the learned representation should improve generalization.

**Takeaway 3: Heuristic augmentation improves generalization *if* the augmentation describes an attribute.** In all settings (figures 2-4), ImageNet augmentation generally improves performance. However, RandAugment, AugMix, and AutoAugment have more variable performance (as further shown in figures 7-9). These methods are compositions of different augmentations. We investigate each augmentation of RandAugment in appendix D.3 and find variable performance. Augmentations that approximate an existing property in the dataset (an attribute) lead to the best results; otherwise, the model may waste capacity. For example, on Camelyon17 (which consists of cell images), color jitter harms performance but on Shapes3D and MPI3D it is essential.

**Takeaway 4: Learned data augmentation is effective across different conditions and distribution shifts.** This approach is highly effective in the *spurious correlation* setting (figure 2). It can also help in the *low data* and *unseen data shift* settings (figure 3,4) (though the gains for these two shifts are not as large as for pretraining). This effectiveness can be explained by the fact that if the augmentations are learned perfectly, then augmented samples by design are from the true data distribution.

**Takeaway 5: Domain generalization algorithms offer limited performance improvement.** In some cases these methods (in particular DANN) do improve performance, most notably in the *low data drift* and *unseen data shift* settings (figures 3-4). However, this depends on the dataset (see figures 7-9) and performance is rarely much better than using heuristic augmentation.

**Takeaway 6: The precise attributes we consider directly impacts the results.** On dSprites, if we make color the label, we find that *all* methods generalise perfectly in the *unseen data shift* setting to unseen shapes (figure 6) unlike when shape is the label and color values unseen (figure 7).

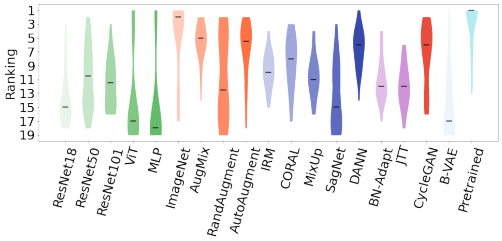



Figure 4: **Unseen data shift.** We rank the methods (where best is 1, worst 19) for each dataset and seed and plot the rankings, with the overall median rank as the black bar. Pretraining on ImageNet and ImageNet augmentation perform consistently best. DANN, CycleGAN and other heuristic augmentations perform consistently well.

### 3.2 Practical tips

**Tip 1: If heuristic augmentations approximate attributes in the dataset, use them.** Under this constraint, heuristic augmentations can significantly improve performance; this should be a first point of call. How to heuristically choose these augmentations without exhaustively trying all possible combinations is an open research question.

**Tip 2: If heuristic augmentations do not help, learn the augmentation.** If the attributes in the dataset cannot be readily approximated with heuristic techniques, but some subset of the attributes can be modelled by using a conditional generative model, this is a promising way to further improve performance. How to learn the true underlying generative model of the data directly and use this for augmentation is a promising area to explore more thoroughly.

**Tip 3: Use pretraining.** In general, pretraining was found to be a useful way to learn a robust representation. While this was not true for all datasets (e.g. CAMELYON17, IWILDCAM), performance could be dramatically improved by pretraining (DSPRITES, MPI3D, SMALLNORB, SHAPES3D). An area to be investigated is the utility of self-supervised pre-training.

**Tip 4: More complex approaches lead to limited improvements.** Domain generalization, adaptive approaches and disentangling lead to limited improvements, if any, across the different datasets and shifts. Of these approaches, DANN performs generally best. How to make these approaches generically useful for robustness is still an open research question.

## 4 Discussion

Our experiments demonstrate that no one method performs best over all shifts and that performance is dependent on the precise attribute being considered. This leads to the following considerations.

**There is no way to decide a-priori on the best method given only the dataset.** It would be helpful for practitioners to be able to select the best approaches without requiring comprehensive evaluations and comparisons. Moreover, it is unclear how to pinpoint the precise distribution shift (and thereby methods to explore) in a given application. This should be an important future area of investigation.

**We should focus on the cases where we have knowledge about the distribution shift.** We found that the ability of a given algorithm to generalize depends heavily on the attribute and dataset being considered. Instead of trying to make one algorithm for any possible shift, it makes sense to have adaptable algorithms which can use auxiliary information if given. Moreover, algorithms should be evaluated in the context for which we will use them.

**It is pivotal to evaluate methods in a variety of conditions.** Performance varies due to the number of examples from each part of the distribution. Thus it is important to perform comprehensive evaluations when comparing different methods, as in our framework. This gives others a more realistic view of different models' relative performance in practice.

## 5 Conclusion

This work has put forward a general, comprehensive framework to reason about distribution shifts. We analyzed 19 different methods, spanning a range of techniques, over three distribution shifts – *spurious correlation*, *low data drift*, and *unseen data shift*. We hope that our framework and comprehensive benchmark spurs research on in this area and provides a useful tool for practitioners to evaluate which methods work best under which shifts.

# References


Ehab A AlBadawy, Ashirbani Saha, and Maciej A Mazurowski. Deep learning for segmentation of brain tumors: Impact of cross-institutional training and testing. *Medical physics*, 2018.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the CAMELYON17 challenge. *IEEE Transactions on Medical Imaging*, 2018.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision*, 2018.

Sara Beery, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Neel Joshi, Markus Meister, and Pietro Perona. Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2020.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, 2018.

Chris Burgess and Hyunjik Kim. 3D shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $\beta$-VAE. In *Workshop on Learning Disentangled Representations at the 31st Conference on Neural Information Processing Systems*, 2017.

Fabio M Carlucci, Paolo Russo, Tatiana Tommasi, and Barbara Caputo. Hallucinating agnostic images to generalize across domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

Daniel C Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nature Communications*, 2020.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *International Conference on Intelligent Transportation Systems*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Andrea Dittadi, Frederik Träuble, Francesco Locatello, Manuel Wüthrich, Vaibhav Agrawal, Ole Winther, Stefan Bauer, and Bernhard Schölkopf. On the transfer of disentangled representations in realistic settings. In *Proceedings of the International Conference on Learning Representations*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, 2021.

Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of the International Conference on Learning Representations*, 2019.

Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation. *arXiv preprint arXiv:2008.06775*, 2020.

Muhammad Waleed Gondal, Manuel Wüthrich, Dorde Miladinović, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *arXiv preprint arXiv:1906.03292*, 2019.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Proceedings of the International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Will Douglas Heaven. Google's medical ai was super accurate in a lab. real life was a different story., 2020.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*, 2019.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the International Conference on Machine Learning*, 2019.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. In *Advances in Neural Information Processing Systems*, 2020.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *Proceedings of the International Conference on Computer Vision*, 2021.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. $\beta$-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations*, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017b.

Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 2020.

Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *The International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 2021.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the International Conference on Machine Learning*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the International Conference on Computer Vision*, 2017.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision*, pages 624–639, 2018.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just Train Twice: Improving group robustness without training group information. In *Proceedings of the International Conference on Machine Learning*, 2021.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the International Conference on Machine Learning*, 2019.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *Proceedings of the International Conference on Machine Learning*, 2020.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 2021.

Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. The role of disentanglement in generalisation. In *Proceedings of the International Conference on Learning Representations*, 2020.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, 2010.

Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the International Conference on Computer Vision*, 2019.

Christian S Perone, Pedro Ballester, Rodrigo C Barros, and Julien Cohen-Adad. Unsupervised domain adaptation for medical imaging segmentation with self-ensembling. *NeuroImage*, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the International Conference on Machine Learning*, 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, 2014.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *Proceedings of the International Conference on Learning Representations*, 2020.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *Proceedings of the International Conference on Learning Representations*, 2020.

Lukas Schott, Julius von Kügelgen, Frederik Träuble, Peter Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. In *Proceedings of the International Conference on Learning Representations*, 2021.

Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? *arXiv preprint arXiv:1906.02168*, 2019.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proceedings of the European Conference on Computer Vision*, 2016.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, 1992.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or Signal: The role of image backgrounds in object recognition. *ArXiv preprint arXiv:2006.09994*, 2020.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *Proceedings of the International Conference on Learning Representations*, 2020.

Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *AAAI Conference on Artificial Intelligence*, 2020.

Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. MixUp: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations*, 2018.

Ling Zhang, Xiaosong Wang, Dong Yang, Thomas Sanford, Stephanie Harmon, Baris Turkbey, Holger Roth, Andriy Myronenko, Daguang Xu, and Ziyue Xu. When unseen domain generalization is unnecessary? rethinking data augmentation. *arXiv preprint arXiv:1906.03347*, 2019.

Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, 2019.

Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *AAAI Conference on Artificial Intelligence*, 2020.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision*, 2017.

# A Overview

In the appendix we first motivate the types of distributions we considered by connecting them to ideas from disentanglement literature in appendix B. We then relate the 19 approaches we considered to this framework in appendix C. We then give the full results as well as additional results in appendix D; code samples from our framework showing how it can be extended in appendix E; a thorough literature review in appendix F; a description of the datasets and how we set up the distribution shifts on these datasets in appendix G; and finally implementation details in appendix H.

# B Robustness Framework

In this section we introduce our robustness framework for characterizing distribution shifts in a principled manner. We then relate three common, real world inspired distribution shifts.

## B.1 Latent Factorisation

We assume a joint distribution $p$ of inputs $\boldsymbol{x}$ and corresponding attributes $y^1, y^2, \ldots, y^K$ (denoted as $y^{1:K}$) with $y^k \in \mathbb{A}^k$ where $\mathbb{A}^k$ is a finite set. One of these $K$ attributes is a label of interest, denoted as $y^l$ (in a mammogram, the label could be cancer/benign and a nuisance attribute $y^i$ with $i \neq l$ could be the identity of the hospital where the mammogram was taken). Our aim is to build a classifier $f$ that minimizes the risk $R$. However, in real-world applications, we only have access to a finite set of inputs and attributes of size $n$. Hence, we minimize the empirical risk $\hat{R}$ instead:

$$R(f) = \mathbb{E}_{(\boldsymbol{x}, y^l)\sim p}\left[\mathcal{L}(y^l, f(\boldsymbol{x}))\right] \qquad \hat{R}(f; p) = \frac{1}{n} \sum_{\{(y_i^l, \boldsymbol{x}_i)\sim p\}_{i=1}^n} \mathcal{L}(y_i^l, f(\boldsymbol{x}_i)).$$

where $\mathcal{L}$ is a suitable loss function. Here, all nuisance attributes $y^k$ with $k \neq l$ are ignored and we work with samples obtained from the marginal $p(y^l, \boldsymbol{x})$. In practice, however, due to selection bias or other confounding factors in data collection, we are only able to train and test our models on data collected from two related but distinct distributions: $p_{\text{train}}, p_{\text{test}}$. For example, $p_{\text{train}}$ and $p_{\text{test}}$ may be concentrated on different subsets of hospitals and this discrepancy may result in a distribution shift; for example, hospitals may use different equipment, leading to different staining on their cell images. While we train $f$ on data from $p_{\text{train}}$ by minimizing $\hat{R}(f; p_{\text{train}})$, we aim to learn a model that generalises well to data from $p_{\text{test}}$; that is, it should achieve a small $\hat{R}(f; p_{\text{test}})$.

While generalization in the above sense is desirable for machine learning models, it is not clear *why* a model $f$ trained on data from $p_{\text{train}}$ *should* generalise to $p_{\text{test}}$. It is worth noting that while $p_{\text{train}}$ and $p_{\text{test}}$ can be different, they are both related to the true distribution $p$. As such, we take inspiration from disentanglement literature to express this relationship. In particular, that we can view data as being decomposed into an underlying set of factors of variations. We formalise various distribution shifts using a latent variable model for the true data generation process:

$$z \sim p(z) \qquad y^i \sim p(y^i|z) \quad i = 1 \ldots K \qquad \boldsymbol{x} \sim p(\boldsymbol{x}|z) \tag{1}$$

where $z$ denotes latent factors. By a simple refactorization, we can write

$$p(y^{1:K}, \boldsymbol{x}) = p(y^{1:K}) \int p(\boldsymbol{x}|z)p(z|y^{1:K})dz = p(y^{1:K})p(\boldsymbol{x}|y^{1:K}).$$

Thus, the true distribution can be expressed as the product of the marginal distribution of the attributes with a conditional generative model. We assume that distribution shifts arise when a new marginal distribution for the attributes is chosen, such as $p(y^{1:K}) \neq p_{\text{train}}(y^{1:K}) \neq p_{\text{test}}(y^{1:K})$, but otherwise the conditional generative model is shared across all distributions, i.e., we have $p_{\text{test}}(y^{1:K}, \boldsymbol{x}) = p_{\text{test}}(y^{1:K}) \int p(\boldsymbol{x}|z)p(z|y^{1:K})dz$, and similarly for $p_{\text{train}}$.

We focus on common cases of distribution shifts visualized in figure 1; we discuss these in more detail in section B.2. To provide more context, as a running example, we use the color DSPRITES dataset [Matthey et al., 2017]; where in our notation $y^1$ defines the color with $\mathbb{A}^1 = \{\text{red}, \text{green}, \text{blue}\}$, and $y^2$ defines the shape with $\mathbb{A}^2 = \{\text{ellipse}, \text{heart}, \text{square}\}$. We can imagine that a data collector (intentionally or implicitly) selects some marginal distribution over attributes $p_{\text{train}}(y^{1:K})$ when training; for example they select mostly blue ellipses and red hearts. This induces a new joint distribution

over latent factors and attributes: $p_{\text{train}}(z, y^{1:K}) = p(z|y^{1:K})p_{\text{train}}(y^{1:K})$. Consequently, during training, we get images with a different joint distribution $p_{\text{train}}(\boldsymbol{x}, y^{1:K}) = \int p(\boldsymbol{x}|z)p_{\text{train}}(z, y^{1:K})$. This similarly applies when collecting data for the test distribution.

The goal of enforcing robustness to distribution shifts is to maintain performance when the data generating distribution $p_{\text{train}}$ changes. In other words, we would like to minimize risk on $p, p_{\text{test}}$ given *only* access to $p_{\text{train}}$. We can achieve robustness in the following ways:

- **Weighted resampling.** We can resample the training set using importance weights $W(y^{1:K}) = p(y^{1:K})/p_{\text{train}}(y^{1:K})$. This means that given the attributes, the $i$-th data point $(y_i^{1:K}, \boldsymbol{x}_i)$ in the training set is used with probability $W(y_i^{1:K})/\sum_{i'=1}^{n} W(y_{i'}^{1:K})$ rather than $1/n$. We refer to this empirical distribution as $p_{\text{reweight}}$. This procedure requires access to the true distribution of attributes $p(y^{1:K})$, so to avoid bias and improve fairness, it is often assumed that all combinations of attributes happen uniformly at random.
- **Data Augmentation**: Alternatively, we can learn a generative model $\hat{p}(\boldsymbol{x}|y^{1:K})$ from the training data that aims to approximate $\int p(\boldsymbol{x}|z)p(z|y^{1:K})dz$, as the true conditional generator is by our assumption the same over all (e.g. train and test) distributions. If such a conditional generative model can be learned, we can sample new synthetic data at training time (e.g. according to the true distribution $p(y^{1:K})$) to correct for the distribution shift. More precisely, we can generate data from the augmented distribution $p_{\text{aug}} = (1 - \alpha)p_{\text{reweight}} + \alpha\hat{p}(\boldsymbol{x}|y^{1:K})p(y^{1:K})$ and train a supervised classifier on this augmented dataset. Here, $\alpha \in [0, 1]$ is the percentage of synthetic data used for training.
- **Representation Learning**: An alternative factorization of a data generating distribution (e.g. train) is $p_{\text{train}}(y^{1:K}, \boldsymbol{x}) = \int p(z|\boldsymbol{x})p_{\text{train}}(y^{1:K}|z)dz$. We can learn an unsupervised representation that approximates $p(z|\boldsymbol{x})$ using the training data only, and attach a classifier to learn a task specific head that approximates $p_{\text{train}}(y^l|z)$. Again, by our assumption $p(z|\boldsymbol{x}) \propto p(\boldsymbol{x}|z)p(z)$. Given a good guess of the true prior, the learned representation would not be impacted by the specific attribute distribution and so generalise to $p_{\text{test}}, p$.

### B.2 Distribution Shifts

While distribution shifts can happen in a continuum, we consider three types of shifts, inspired by real-world challenges.

**Test distribution $p_{\text{test}}$.** We assume that the attributes are distributed uniformly: $p_{\text{test}}(y^{1:K}) = 1/\prod_i |\mathbb{A}^i|$. This is desirable, as all attributes are represented and a-priori independent.

**Shift 1: Spurious correlation – Attributes are correlated under $p_{\text{train}}$ but not $p_{\text{test}}$.** Spurious correlation arises in the wild for a number of reasons including capture bias, environmental factors, and geographical bias [Beery et al., 2018, Torralba and Efros, 2011]. These spurious correlations lead to surprising results and poor generalization. Therefore, it is important to be able to build models that are robust to such challenges. In our framework, spurious correlation arises when two attributes $y^a$, $y^b$ are correlated at training time, but this is not true of $p_{\text{test}}$, for which attributes are independent: $p_{\text{train}}(y^a|y^1 \dots y^b \dots y^K) > p_{\text{train}}(y^a|y^1 \dots y^{b-1}, y^{b+1} \dots y^K)$. This is especially problematic when one attribute $y^b$ is $y^l$, the label. Using the running DSPRITES example, shape and color may be correlated and the model may find it easier to predict color. If color is the label, the model will generalise well. However, if the aim is to predict shape, the model's reliance on color will lead to poor generalization.

**Shift 2: Low-data drift – Attribute values are unevenly distributed under $p_{\text{train}}$ but not under $p_{\text{test}}$.** Low-data drift arises in the wild (e.g. in [Buolamwini and Gebru, 2018] for different ethnicities) when data has not been collected uniformly across different attributes. When deploying models in the wild, it is important to be able to reason and have confidence that the final predictions will be consistent and fair across different attributes. In the framework above, low-data shifts arise when certain values in the set $\mathbb{A}^a$ of an attribute $y^a$ are sampled with a much smaller probability than in $p_{\text{test}}$: $p_{\text{train}}(y^a = v) \ll p_{\text{test}}(y^a = v)$. Using the DSPRITES example, only a handful of red shapes may be seen at training time, yet in $p_{\text{test}}$ all colors are sampled with equal probability.

**Shift 3: Unseen data shift – Some attribute values are unseen under $p_{\text{train}}$ but are under $p_{\text{test}}$.** This is a special case of *shift 2: low-data drift.* which we make explicit due to its important real world applications. Unseen data shift arises when a model, trained in one setting is expected to work in another, disjoint setting. For example: a model trained to classify animals on images at certain times of day should generalise to other times of day. In our framework, unseen data shift arises when some values in the set $\mathbb{A}^a$ of an attribute $y^a$ are unseen in $p_{\text{train}}$ but are in $p_{\text{test}}$:

$$p_{\text{train}}(y^a = v) = 0 \qquad p_{\text{test}}(y^a = v) > 0 \qquad |\{v | p_{\text{train}}(y^a = v)\}| > 1 \tag{2}$$

This is a stronger constraint than in standard out-of-distribution generalization (see appendix F), as multiple values for $\mathbb{A}^a$ must be seen under $p_{\text{train}}$, which allows the model to learn invariance to $y^a$. In the DSPRITES example, the color red may be unseen at train time but all colors are in $p_{\text{test}}$.

**Discussion.** We choose these sets of shifts as they are the building blocks of more complex distribution shifts. Consider the simplest case of two attributes: the label and a nuisance attribute. If we consider the marginal distribution of the label, it decomposes into two terms: the conditional probability and the probability of a given attribute value: $p(y^l) = \sum_{y^a} p(y^l|y^a)p(y^a)$. The three shifts we consider control these terms independently: *unseen data shift* and *low-data drift* control $p(y^a)$ whereas *spurious correlation* controls $p(y^l|y^a)$. The composition of these terms describes any distribution shift for these two variables.

# C Method

Here we give further description of the methods we implement and how they relate to our robustness framework. This allows us to obtain an intuition of what guarantees each method gives in this context and thereby under what circumstances they should promote generalization.

**Backbone architecture.** We investigate the performance of different standard vision models on the robustness task. The model is trained on $p_{\text{train}}$ to predict the true label, making no use of the additional attribute information. We use weighted resampling $p_{\text{reweight}}$ [Vapnik, 1992] to oversample from the parts of the distribution that have a lower probability. This is what we refer to as the *standard* setup.

**Heuristic data augmentation.** In this case, we use standard heuristic augmentation methods in order to augment the training samples in $p_{\text{train}}$. Instead of attempting to learn the conditional generative model, in this approach, we 'fake' the generative model by augmenting the images using a set of heuristic functions to create $p_{\text{aug}}, \alpha = 1$. However, we have to heuristically choose these functions, so the generative model they approximate may *not* correspond to the true underlying generative model $p(\boldsymbol{x}|y^{1:K})$. In practice, these methods make no use of the additional attribute information and are trained to predict the label, as in the standard setup.

**Learned data augmentation.** Again, we approximate the underlying generative model $p(\boldsymbol{x}|y^{1:K})$ by a set of augmentations. However, instead of heuristically choosing these augmentation functions, we learn them from data. We learn a function that, given an image $\boldsymbol{x}$ and attribute $y^a = v_i$, transforms $\boldsymbol{x}$ to have another attribute value $y^a = v_j$, while keeping all other attributes fixed. We can then use this function to generate new samples $p_{\text{aug}}, \alpha = 1$ with any distribution over $y^a$. In particular, we generate samples under the uniform distribution. In this case, the additional attribute information is used to generate new samples from a given image. However, the performance of this approach is highly dependent on the quality of the learned generative model. We follow Goel et al. [2020], who use CYCLEGAN [Zhu et al., 2017] to learn how to transform an image with one attribute value to have that of another. (We do not use their additional SGDRO objective as we want to study the impact of the data augmentation process alone.) In our case, there are more than two attribute values, so we use STARGAN [Choi et al., 2018] to learn a single model that, conditioned on an input image and desired attribute, transforms that image to have the new attribute.

**Domain generalization.** These works were devised to improve domain generalization. They can also be seen as a form of representation learning. The aim is to recover $p(z|\boldsymbol{x})$ such that the representation $z$ is independent of the domain (in our framework, the attribute $y^a$): $p(y^a, z) = p(y^a)p(z)$. If this is achieved, then the task specific classifier $p(y^l|z)$ will be independent of $y^a$ by definition. However, these approaches rely on the ability of the underlying method to learn invariance.

**Adaptive approaches.** These works modify the reweighting distribution in $p_{\text{reweight}}$ using multi stage training. The models are trained first as for the standard setup, giving a classifier $f_o$. JTT [Liu et al., 2021] then uses $f_o$ to approximate the difficulty of the sample. The more difficult samples are weighted higher in the second stage according to a factor $\lambda$. This is equivalent to sampling the more difficult sample $\lambda$ times in a batch (thereby learning a more complex function $W$). BN-Adapt [Schneider et al., 2020] learns $K$ models in the second stage by modifying the batch normalization parameters. For the $j$-th model, $W(y^j) = 1, W(y^i) = 0$ for $i \neq j$. However, neither of these methods give strong guarantees on the properties of the final model.

**Representation learning.** Finally, in representation learning, the aim is to learn an initial representation that has preferable properties to standard ERM training; the motivation for this approach is discussed in appendix B. To learn the prior, we can pretrain $f$ on large amounts of auxiliary data, such as on ImageNet [Russakovsky et al., 2015]. This has been demonstrated to improve model robustness and uncertainty between datasets [Hendrycks et al., 2019], but here we investigate its utility under different distribution shifts. Another approach is to attempt to learn a disentangled representation with a VAE, as in $\beta$-VAE [Higgins et al., 2017a], where $z$ would then describe the underlying factors of variation for the generative model. However, the robustness of these methods is dependent on the quality of the learned representation for the specific robustness task.

# D Results

We give the complete breakdown of results for all methods and shifts in appendix D.1. We give a detailed analysis on the impact of each augmentation type in RandAugment in appendix D.3. Finally, we evaluate the performance using a different attribute as the label in appendix D.3 and using the ID (as opposed to OOD) validation set on IWILDCAM and CAMELYON17 in appendix D.4.

## D.1 Complete results

Here, we give the complete results over each dataset for each shift (*spurious correlation*, *low-data drift*, and *unseen data shift*) in figures 7-9. In these plots, we plot the mean and standard deviation of each method on each dataset and sort them according to their performance.

## D.2 A detailed analysis on the impact of augmentation

We found that different methods of performing heuristic augmentation perform differently: some outperform the ERM baseline, some do not. Each of these methods are composed of individual augmentation techniques. Here, we investigate how each technique contributes to the end performance and thereby how the *choice* of augmentation function impacts the robustness of the models.

We take the augmentations used in RandAugment. Instead of sampling all augmentations randomly, for each augmentation, we randomly sample the given augmentation or the identity function. We plot the deviation from the mean in figure 5 for each dataset under *unseen data shift*. The results are surprising. No augmentation always leads to a strong boost in performance. For example, using *invert* improves performance on DSPRITES, SHAPES3D but harms performance on MPI3D, SMALLNORB, and IWILDCAM. Similarly, using *color* improves performance on most datasets but harms performance on CAMELYON17.

## D.3 Results with different attributes

Here we investigate if the results are dependent on the attributes being investigated. We investigate *unseen data shift* on DSPRITES, but instead of predicting shape, we predict color. We then leave out some shapes and test whether models can correctly predict color given the unseen shape. We find that *all* methods generalise to the OOD case with approximately perfect score, as shown in figure 6.

## D.4 Results using different, OOD validation sets

When evaluating on IWILDCAM and CAMELYON17, we explore whether using the out-of-distribution (OOD) or in-distribution (ID) validation sets are preferable for obtaining best performance. We find that neither the out-of-distribution (OOD) nor in-distribution (ID) validation sets perform best, but the

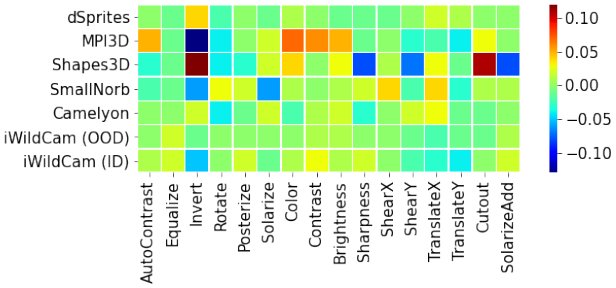



Figure 5: **RandAugment ablation.** Performance of each augmentation across the different datasets. As can be seen, no one augmentation always improves performance. An augmentation that may improve performance on one dataset (e.g. invert on SHAPES3D) hurts performance on another (e.g. invert on SMALLNORB).

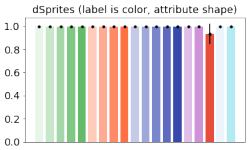
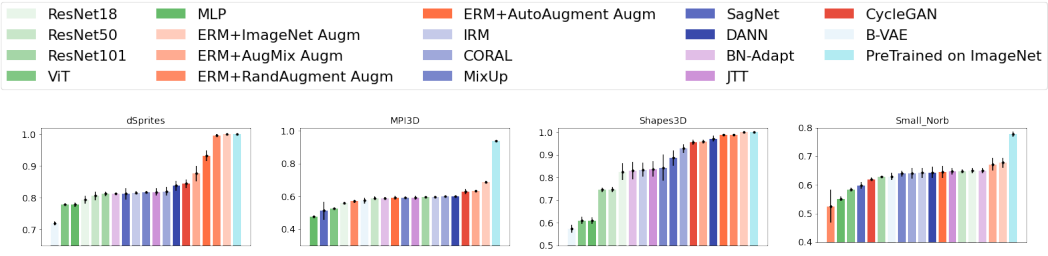



Figure 6: **Shift 3: Unseen data shift.** We plot the top-1 accuracy (y-axis) for the DSPRITES dataset. Unlike in the main paper, here the label is the color and the attribute the shape. Higher is better. All approaches solve the task.

relative performance remains similar The relative performance of different models in figure 7 when using the ID or OOD validation set is comparable on CAMELYON17 and IWILDCAM. However, the maximum performance is not consistently better using either the ID or OOD set (for CAMELYON17, OOD performs best but for IWILDCAM, ID performs best).

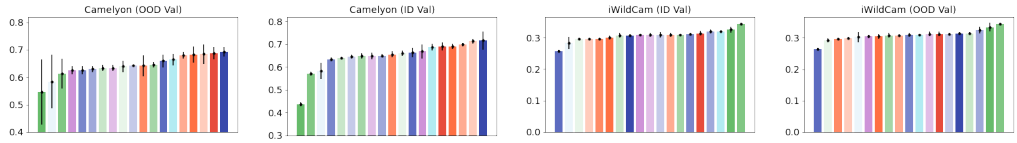



(a) DSPRITES (label = shape, attribute = color)

(b) MPI3D (label = shape, attribute = color)

(c) SHAPES3D (label = shape, attr = color)

(d) SMALLNORB (label = category, attr = azimuth)




(e) CAMELYON17. Using the ID val set.

(f) CAMELYON17. Using the OOD val set.

(g) IWILDCAM. Using the ID val set.

(h) IWILDCAM. Using the OOD val set.

Figure 7: **Shift 3: Unseen data.** We plot the top-1 accuracy (y-axis) over different datasets. For each value of $N$ we resort the results and show them in order. Higher is better.

# E Extending the evaluation framework

Our evaluation framework is easily extendable. Different methods, datasets, and distribution shifts can be easily added and evaluated. We include code in appendix E.1-E.3, demonstrating how this is done.

## E.1 Adding a new distribution shift

Changing the distribution shift simply amounts to updating four values. These values define (1) how to construct the distribution as a combination of other distributions; (2) the total number of samples from each distribution; (3) the probability of sampling from a given distribution; and (4) the probability that the label is corrupted at training time.

Assume a dataset that has the key *center* which takes values in the range [0, 4] and the key *label* which takes values in the set {0, 1}. The code in algorithm 1 defines the following dataset: with probability 0.1, we sample an example that is either from center 0 or center 1 from the complete dataset; with probability 0.4 we sample from a total of 100 samples that are from center 3 or 4 with label 0; and with probability 0.5, we sample from a total of 10K samples from centers 0, 1, 2. None of the labels are corrupted.

```
  """Given a string "dist1,dist2,...,distN", the string is first divided into the different
     distributions, separated by commas. Each distribution is defined by a string of "key11:val11:
     comp11^key12:val12:comp12|...^keyNk:valNk:compNk". This string is parsed as the OR of several AND
     statements. The ORs are at the top level (denoted by |), and divide into a set of AND statements.
     The AND values are denoted by ^ and operate at the bottom level.  For each "keyij:valij:compij"
     pairing, keyij is the key in the dataset, valij is the value the key is compared against and
     compij is the tensorflow comparison function (e.g. less, less_equal, equal, greater_equal, or
     greater)."""
config.filter_fn = "center:0:equal|center:1:equal,center:2:greater^label:0:equal,center:3:less_equal"

# num_samples defines the total number of samples from each distribution. 0 denotes that all samples
     are taken
config.num_samples = [0, 100, 10_000]

# weights defines the probability of sampling from each distribution.
config.weights = [0.1, 0.4, 0.5]

# Label_noise defines the probability that the label is corrupted (with uniform probability).
config.label_noise = [0.0]
```

Listing 1: Setting up a new distribution shift.

## E.2 Adding a new dataset

Adding a new data loader requires implementing three methods: (1) a preprocessing function for how to preprocess a batch; (2) a function to load the data in a tensorflow datasets format; (3) a function to batch and shuffle this tensorflow dataset. In algorithm 2, we demonstrate how this is done for the SHAPES3D dataset and in algorithm 3 how the config is updated to include a new dataset.

```
"""The preprocessing function to update the image and label key."""
def shapes3d_preprocess(mode = 'train'):
  del mode

  def _preprocess_fn(example):
    example['image'] = tf.image.convert_image_dtype(example['image'], dtype=tf.float32)
    example['label'] = example['label_shape']
    return example
  return _preprocess_fn


"""The tfds loading function."""
def unbatched_load_shapes3d(subset = 'train', valid_size = 10000, test_size = 10000):
  """Loads the 3D Shapes dataset in tfds format without batching."""
  if subset == 'train':
    ds = tfds.load(name='shapes3d', split=tfds.Split.TRAIN).skip(valid_size + test_size)
  elif subset == 'valid':
    ds = tfds.load(name='shapes3d', split=tfds.Split.TRAIN).skip(test_size).take(valid_size)
  elif subset == 'train_and_valid':
    ds = tfds.load(name='shapes3d', split=tfds.Split.TRAIN).skip(test_size)
  elif subset == 'test':
    ds = tfds.load(name='shapes3d', split=tfds.Split.TRAIN).take(test_size)
  else:
    raise ValueError('Unknown subset: "{}"'.format(subset))
  return ds
```

```python
def load_shapes3d(batch_sizes, subset = 'train', is_training = True, num_samples = None, preprocess_fn
      = None, transpose = False, valid_size = 10000, test_size = 10000, drop_remainder = True,
      local_cache = True):
  """Loads the 3D Shapes dataset.

  The 3D shapes dataset is available at https://github.com/deepmind/3d-shapes.
  It consists of 4 different shapes which vary along 5 different axes:
  - Floor hue: 10 colors
  - Wall hue: 10 colors
  - Object hue: 10 colors
  - Scale: How large the object is.
  - Shape: 4 values -- (cube, sphere, cylinder, and oblong).
  - Orientation: Rotates the object around the vertical axis.

  Args:
    batch_sizes: Specifies how to batch examples. I.e., if batch_sizes = [8, 4]
      then output images will have shapes (8, 4, height, width, 3).
    subset: Specifies which subset (train, valid, train_and_valid, or test) to use.
    is_training: Whether to infinitely repeat and shuffle examples (`True`) or
      not (`False`).
    num_samples: The number of samples to crop each individual dataset variant
      from the start, or `None` to use the full dataset.
    preprocess_fn: Function mapped onto each example for pre-processing.
    transpose: Whether to permute image dimensions NHWC -> HWCN (`True`) or not (`False`).
    valid_size: Size of the validation set to take from the training set.
    test_size: Size of the validation set to take from the training set.
    drop_remainder: Whether to drop the last batch(es) if they would not match
      the shapes specified by `batch_sizes`.
    local_cache: Whether to locally cache the dataset.

  Returns:
    ds: Fully configured dataset ready for training/evaluation.
  """
  if preprocess_fn is None:
    preprocess_fn = shapes3d_preprocess
  ds = unbatched_load_shapes3d(subset=subset, valid_size=valid_size, test_size=test_size)
  total_batch_size = np.prod(batch_sizes)
  if subset == 'valid' and valid_size < total_batch_size:
    ds = ds.repeat().take(total_batch_size)
  ds = batch_and_shuffle(ds, batch_sizes, is_training=is_training, transpose=transpose, num_samples=
      num_samples, preprocess_fn=preprocess_fn,  drop_remainder=drop_remainder, local_cache=local_cache)
  return ds
```

Listing 2: Setting up a new dataset.

```python
"""Update training data configuration."""
config.data.train_kwargs.load_kwargs = dict()
config.data.train_kwargs.load_kwargs['dataset_loader'] = unbatched_load_shapes3d
config.data.train_kwargs.load_kwargs['dataset_kwargs'] = dict(subset='train')  # Update with data
      specific values as appropriate.
config.data.train_kwargs.load_kwargs['preprocess_fn'] = shapes3d_preprocess

"""Update testing data configuration."""
config.data.test_kwargs = dict()
config.data.test_kwargs['loader'] = load_shapes3d
config.data.test_kwargs['load_kwargs'] = dict(subset='test')
```

Listing 3: Updating the config to use the new dataset.

### E.3 Adding a new method

**Adding a new loss function.** Adding a new loss function simply amounts to extending the abstract class in algorithm 4 and updating the config (see algorithm 5) to point to this loss function.

```python
class LearningAlgorithm(hk.Module):
  """Class to encapsulate a learning algorithm."""

  def __init__(self, loss_fn, name, **kwargs):
    """Initializes the algorithm with the given loss function."""
    super().__init__(name=name)
    self.loss_fn = loss_fn

  def __call__(self, logits, targets, reduction, attributes = None):
    """The loss function of the learning algorithm.

    Args:
      logits: The predicted logits input to the training algorithm.
      targets: The ground truth value to estimate.
      reduction: How to combine the loss for different samples (mean or sum).
      attributes: An optional set of properties of the input data.
    Returns:
      scalars: A dictionary of key and scalar estimates. The key 'loss' is the loss that should be
     minimized.
      preds: The raw softmax predictions.
    """
    pass
```

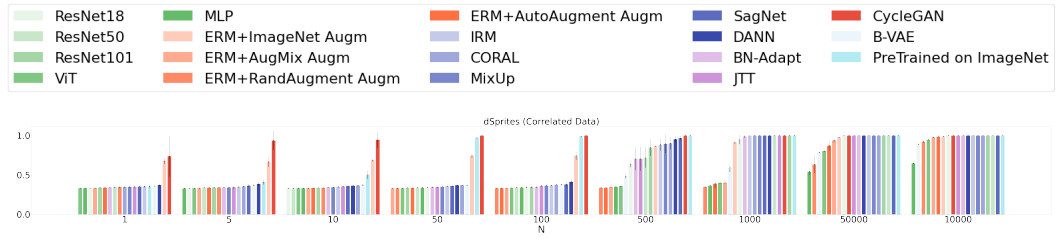
```python
  def adversary(self, logits, attributes, reduction = 'mean', targets = None):
    """The adversarial loss function.

    If la = LearningAlgorithm(), this function is applied in a min-max game with la(). The model is
     trained to minimize the loss arising from la(), while maximizing the loss from the adversary (la.
     adversary()). The adversarial part of the model tries to minimize this loss.

    Args:
      logits: The predicted value input to the training algorithm.
      attributes: A set of attributes of the input data.
      reduction: How to combine the loss for different samples ('mean' or 'sum').
      targets: The ground truth value to estimate (optional).
    Returns:
      scalars: A dictionary of key and scalar estimates. The key 'adv_loss' is the value that should be
       minimized (for the adversary) and maximized (for the model). If empty, this learning algorithm
     has no adversary.
    """
    # Do nothing.
    return {}


class ERM(base.LearningAlgorithm):
  """An example, demonstrating how to compute the empirical risk within our framework."""

  def __init__(self, loss_fn, name = 'empirical_risk'):
    super().__init__(loss_fn=loss_fn, name=name)

  def __call__(self, logits, targets, reduction = 'mean', **unused_kwargs):
    loss = self.loss_fn(logits, targets, reduction=reduction)
    return {'loss': loss}, logits
```

Listing 4: Setting up a new loss function.

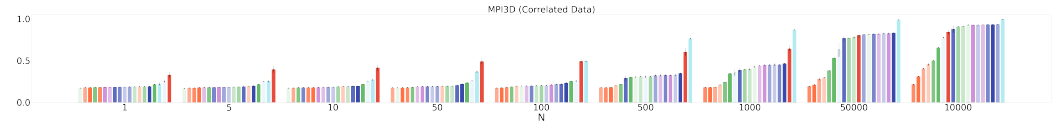
```python
"""Config for which learning algorithm to use."""
config.learner = dict()
config.learner['fn'] = ERM
config.learner['kwargs'] = dict()  # Update with algorithm specific parameters.
```

Listing 5: Updating the config for the new loss function.

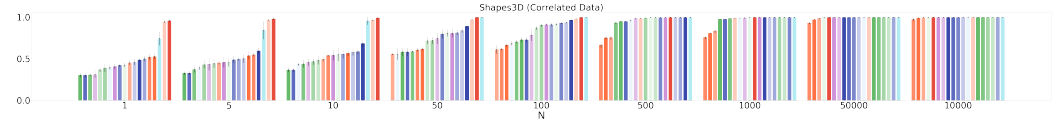
**Extending other parts of the framework.** We similarly define abstract classes, initialized in the config, for model selection, preprocessing and postprocessing a batch, and adapting model parameters at train and test time. These can all be extended in order to include new models, algorithms, preprocessing and postprocessing steps or adaptive approaches.



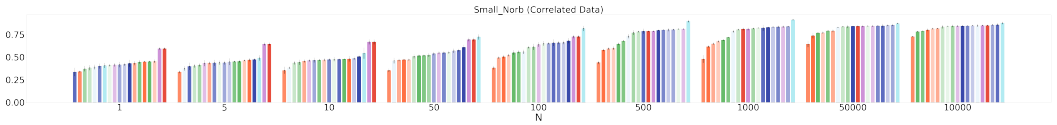

(a) DSPRITES (label = shape, attribute = color)



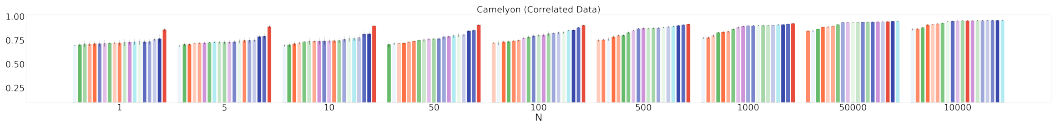

(b) MPI3D (label = shape, attribute = object color)



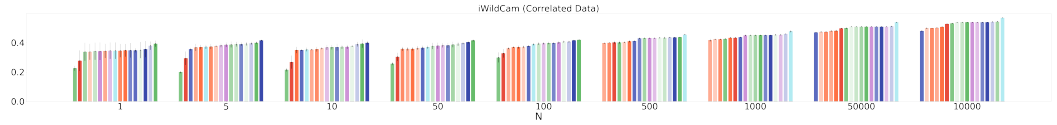

(c) SHAPES3D (label = shape, attribute = object color)




(d) SMALLNORB (label = category, attribute = lighting)

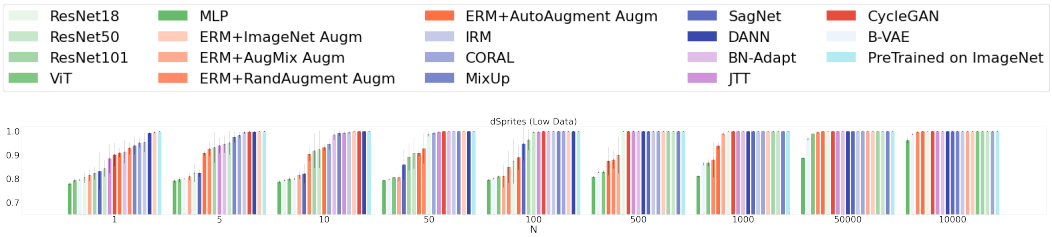



(e) CAMELYON17 (label = has tumor cells, attribute = hospital)

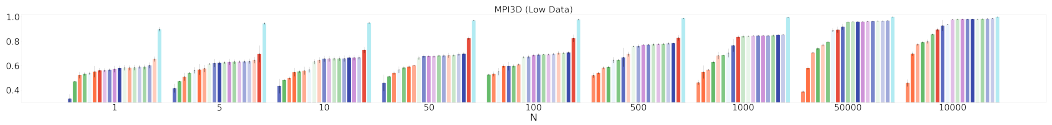



(f) IWILDCAM (label = animal, attribute = location)

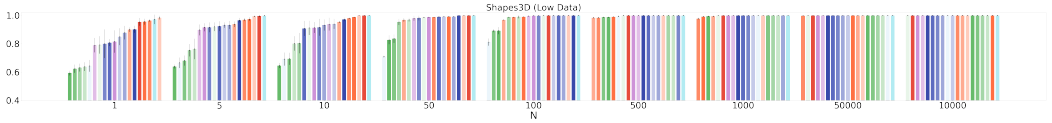
Figure 8: **Shift 1: Spurious correlation.** We plot the top-1 accuracy (y-axis) for different values of $N$ (the number of samples from the independent distribution, the x-axis) over different datasets. For each value of $N$ we resort the results and show them in order. Higher is better.



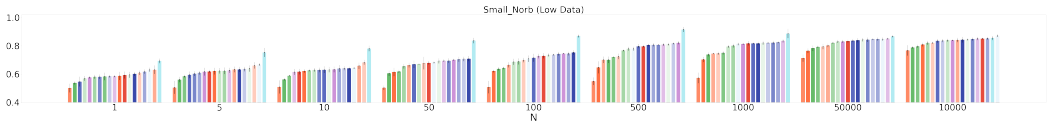

(a) DSPRITES (label = shape, attribute = color)



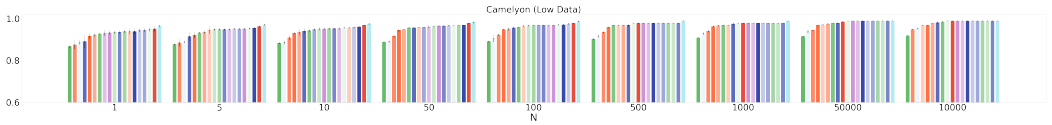

(b) MPI3D (label = shape, attribute = object color)



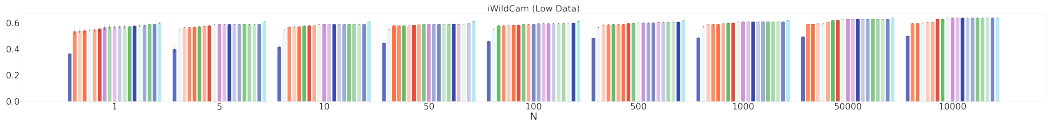

(c) SHAPES3D (label = shape, attribute = object color)




(d) SMALLNORB (label = category, attribute = azimuth)




(e) CAMELYON17 (label = has tumor cells, attribute = hospital)




(f) IWILDCAM (label = animal, attribute = location)

Figure 9: **Shift 2: Low data drift.** We plot the top-1 accuracy (y-axis) for different values of $N$ (the number of samples from the independent distribution, the x-axis) over different datasets. For each value of $N$ we resort the results and show them in order. Higher is better.

# F Literature review

Here we discuss in depth related literature. We first discuss datasets and benchmarks used to evaluate different distribution shifts. We then discuss methods related to the five common approaches we consider in the paper: architecture choice, data augmentation, domain generalization methods, adaptive approaches, and representation learning. We additionally discuss the hypotheses present in the current literature and whether our results corroborate or dispute those hypotheses.

**Benchmarking robustness to out of distribution (OOD) generalization.** While a multitude of methods exist that report improved OOD generalization, Gulrajani and Lopez-Paz [2021] found that in actuality no evaluated method performed significantly better than a strong ERM baseline on a variety of datasets. However, Hendrycks et al. [2021] found that, when we focus on better augmentation, larger models and pretraining, we can get a sizeable boost in performance. This can be seen on the Koh et al. [2020] benchmark (the largest boosts come from larger models and better augmentation). Our work is complementary to these methods, as we look at a range of approaches (pretraining, heuristic augmentation, learned augmentation, domain generalisation, adaptive, disentangled representations) on a range of both synthetic and real-world datasets. Moreover, we allow for a fine-grained analysis of methods over different distribution shifts.

**Benchmarking spurious correlation and low-data drift.** Studies on fairness and bias (surveyed by Mehrabi et al. [2021]) have demonstrated the pernicious impact of low-data in gender recognition [Buolamwini and Gebru, 2018], medical imaging [Castro et al., 2020], and conservation [Beery et al., 2018] and spurious correlation in classification [Geirhos et al., 2019] and conservation [Beery et al., 2020]. Arjovsky et al. [2019] hypothesized that spurious correlation may be the underlying reason for poor generalization of models to unseen data. To our knowledge, there has been no large scale work focused on understanding the benefits of different methods across these distribution shifts systematically across multiple datasets and with fine-grained control on the amount of shift. Here we introduce a framework for creating these shifts we seek to understand in a controllable way to allow such challenges to be investigated robustly.

**Benchmarking disentangled representations.** A related area, disentangled representation learning aims to learn a representation where the factors of variation in the data are separated. If this could be achieved, then models should be able to generalise effortlessly to unseen data as investigated in multiple settings such as reinforcement learning [Higgins et al., 2017b]. Despite many years of work in disentangled representations [Higgins et al., 2017a, Burgess et al., 2017, Kim and Mnih, 2018, Chen et al., 2018], a benchmark study by Locatello et al. [2019] found that, without supervision or implicit model or data assumptions, one can not reliably perform disentanglement; however, weak supervision appears sufficient to do so [Locatello et al., 2020]. Dittadi et al. [2021], Schott et al. [2021], Montero et al. [2020] further investigated whether representations (disentangled or not) can interpolate, extrapolate, or compose properties; they found that when considering complex combinations of properties and multiple datasets, representations do not do so reliably.

**Datasets to evaluate distribution shifts.** Obtaining datasets of real world distribution shifts is challenging and expensive. As a result, many works have focussed on using synthetic or existing datasets to build their benchmarks. One approach is to create synthetic shifts over ImageNet to study how models generalise in specific, synthetic conditions: ImageNet-C [Hendrycks and Dietterich, 2019] studies the impact of standard corruptions; Stylized ImageNet [Geirhos et al., 2019] studies the impact of texture; Waterbirds and the Backgrounds challenge [Sagawa et al., 2020, Xiao et al., 2020] study the impact of a fake background; and colored MNIST [Gulrajani and Lopez-Paz, 2021] studies impact of color on classification. Another approach is to consider how models trained on a given dataset generalise to other datasets of the same set of objects (e.g. cartoons to paintings in PACS or ImageNet to ImageNetV2) [Torralba and Efros, 2011, Li et al., 2017, Recht et al., 2019, Peng et al., 2019, Venkateswara et al., 2017] or over time [Shankar et al., 2019]. While these datasets give insight into the biases of a trained model, models that do well on them may not necessarily do well on real world tasks. As a result, the WILDS dataset [Koh et al., 2020] was created as a collection of real world distribution shifts where the OOD task is well defined (e.g. the model should generalise to unseen countries for satellite imagery or hospitals for tumour identification) in order to evaluate progress.

| | Controllable shifts? | Many methods? | Real world motivated? |
|---|---|---|---|
| Ours | ✓ | ✓ | ✓ |
| WILDS [Koh et al., 2020] | ✗ | ✗ | ✓ |
| Gulrajani and Lopez-Paz [2021] | ✗ | ✓ | ✗ |
| Hendrycks et al. [2021] | ✗ | ✓ | ✗ |

Table 1: Comparison of the three most similar benchmark works to ours. Unlike these works, our framework allows for controlling the distribution shift to be analysed. Moreover, we use real-world motivated datasets and evaluate methods across multiple approaches; we encompass more approaches than Gulrajani and Lopez-Paz [2021] and Hendrycks et al. [2021].

Our framework is complementary to these datasets and benchmarks (table 1). Given a dataset, our framework can be used to set up a desired distribution shift to be investigated with fine-grained control over the type of distribution shift and amount of shift. Moreover, our framework provides extendable classes for a wide range of approaches and datasets. Finally, we present a robustness framework inspired by disentangling literature hypothesizing *when* we expect models to be able to generalise across these shifts.

**Architecture choice.** Previous work has investigated the impact of model size on robustness to various distribution shifts, finding that larger models generally are more robust when considering common corruptions [Hendrycks et al., 2019] and adversarial training [Xie and Yuille, 2020]. However, [Schott et al., 2021] finds that larger models do not give representations that generalise better within the same domain. We find that there is no strict rule correlating model size or model capacity with performance. Sometimes deeper ResNets perform best, sometimes not. The ViT model (with the highest capacity) often performs worse, but on iWildCam it performs best.

**Data augmentation.** Data augmentation is often pivotal to achieve state of the art performance on machine learning benchmarks, leading to a wide range of methods to perform heuristic data augmentation [He et al., 2016, Hendrycks et al., 2020, Cubuk et al., 2019, 2020, Zhang et al., 2018]. When considering domain generalization, more specific techniques for augmentation have been devised to improve a model's ability to generalise using MixUp [Gulrajani and Lopez-Paz, 2021, Xu et al., 2020, Yan et al., 2020, Wang et al., 2020] or knowledge of the domains [Zhang et al., 2019].

Instead of using heuristic data augmentations which cannot be used to learn more complex transformations, the desired augmentation can be learnt. The training data can be augmented by transforming samples using a generative model to either come from another part of the domain conditioned on an attribute [Goel et al., 2020], another domain [Zhou et al., 2020], be domain agnostic [Carlucci et al., 2019], or to have a different style [Gowal et al., 2020, Geirhos et al., 2019]. These methods often build on work in image generation, such as CycleGAN [Zhu et al., 2017] and StyleGAN [Karras et al., 2019].

We find that these approaches can be effective to learn invariance to the heuristic or learned transformation. However, performance depends on whether the augmentation is a useful property to learn invariance against, else the model may waste capacity.

**Domain generalization.** While many works can be viewed as aiming to improve robustness of models in new domains, here we focus on those methods that aim to learn domain invariant features in stochastic trained machine learning models. One impactful approach to learning features invariant to the domain is DANN [Ganin et al., 2016] (domain adversarial neural networks). This work uses an adversarial network [Goodfellow et al., 2014] to enforce that the features cannot be predictive of the domain. Later work considered a number of ways to enforce invariance, such as the following: minimizing the maximum mean discrepancy [Long et al., 2015, 2017], invariance of the conditional distribution [Li et al., 2018], and invariance of the covariance matrix of the feature distribution [Sun and Saenko, 2016]. However, enforcing invariance is challenging to learn and often too strict a constraint [Johansson et al., 2019, Zhao et al., 2019]. As a result, IRM [Arjovsky et al., 2019] instead enforces that the optimal classifier for different domains is the same. We find that DANN performs consistently best but that performance varies over different datasets and distribution shifts.

**Adaptive approaches.** Instead of learning a single model and treating samples similarly, adaptive approaches can be used to either modify model parameters or dynamically modify training if there are multiple domains (or groups with different attributes) within the training set. Ways to adapt the model parameters to a new domain include meta learning (as in MAML [Finn et al., 2017]) and adapting the batch normalisation statistics based on the different domains [Schneider et al., 2020]. Instead of adapting the parameters, other approaches reweigh the importance of samples on which the model struggles. This, intuitively, should force the model to spend more capacity on more challenging parts of the domain (or groups with fewer samples). In GroupDRO [Sagawa et al., 2020], this amounts to putting more mass on samples from the more challenging domains at train time using the loss to determine the challenging domains. In JTT [Liu et al., 2021], this is done by two stage training. First, a classifier is trained in a standard manner. Then, a second classifier is trained by upweighting the samples with a high loss according to the first classifier. The authors posit that the most challenging samples will be those coming from groups with few samples (the low-data regions). We find that neither of these methods consistently performs better than the baselines.

**Representation learning.** Finally, instead of focusing on the downstream task, another approach is to learn a representation $p(z|\boldsymbol{I})$ that approximates the true prior over the latent factors. This can be done by pretraining on large amounts of data such as ImageNet [Russakovsky et al., 2015, Deng et al., 2009] as explored by [Hendrycks et al., 2021] for domain generalization, such that the learned representation is potentially more robust and general. Our results corroborate their findings: in many cases pretraining is helpful. However, this is not universally true. On CAMELYON17 and IWILDCAM, pretraining did not improve performance across all shifts. For example, pretraining was unhelpful under spurious correlation on CAMELYON17.

Another approach is to learn a representation subject to constraints. This is what the disentanglement literature aims to do: find a representation that is sparse and low dimensional, which hopefully will correspond to a higher level, disentangled representation. Disentanglement is a large research area, so we briefly mention some formative works, such as the $\beta$-VAE [Higgins et al., 2017a, Burgess et al., 2017] and FactorVAE [Kim and Mnih, 2018], which build on the original VAE [Kingma and Welling, 2013, Rezende et al., 2014] formulation. Other approaches build on GANs [Goodfellow et al., 2014], such as InfoGAN [Chen et al., 2016]. A recent study of these methods [Locatello et al., 2019] found that they did not reliably disentangle the representation into a semantically meaningful set of latent variables. Therefore, it is unclear how robust these methods are when used for distribution shifts, but our results imply that more research is needed to make these approaches practically useful.

# G Dataset

Here we give further details about the datasets in appendix G.1, how we set up the shifts and conditions for these datasets in appendix G.2 and finally we give samples from the different datasets and shifts in appendix G.3.

## G.1 Details

Here we give further detail about each of the datasets and how we set the two attributes: $y^l, y^a$.

**DSPRITES.** DSPRITES [Matthey et al., 2017] consists of shapes (heart, ellipse, and square), which we augment with three colors (red, green, and blue). The shapes vary in location, scale, and orientation. We make the shape the label, and the attribute the color (we consider other choices in the appendix).

**MPI3D.** MPI3D [Gondal et al., 2019] consists of real images of shapes on a robotic arm. There are six shapes and the images vary in terms of the object color, object size, camera height, background color and the rotation about the horizontal and vertical axis. We make the shape the label and color the other attribute.

**SHAPES3D.** SHAPES3D [Burgess and Kim, 2018] consists of images of shapes centered in a synthetic room. There are three shapes and the images vary in terms of the floor hue, object hue, orientation, scale, shape, and wall hue. We make the shape the label and object color the other attribute.

**SMALLNORB.** SMALLNORB [LeCun et al., 2004] consists of images of toys of five categories with varying lighting, elevation and azimuths. The aim is to create methods that generalize to unseen samples of the five categories. We make the object category the label and azimuth the other attribute for *unseen data shift, low-data drift*. We make the lighting the other attribute for *spurious correlation*.

**CAMELYON17.** CAMELYON17 [Koh et al., 2020, Bandi et al., 2018] contains tumour cell images coming from different hospitals. We follow Koh et al. [2020] and make the label the presence of tumor cells and the other attribute the hospital from which the image came from. In *spurious correlation, low-data drift*, we resplit the dataset randomly (there are no held out hospitals). In *spurious correlation*, we correlate the presence of tumor cells with the hospital (all images from a given hospital either do or do not have tumors). In *low-data drift*, we select the hospitals that are out of distribution for Koh et al. [2020] as the hospitals for which we only have $N$ samples. In *unseen data shift*, we use the split given by Koh et al. [2020] and optionally use either the in-distribution or out-of-distribution validation set for model selection.

**IWILDCAM.** IWILDCAM [Koh et al., 2020, Beery et al., 2018] contains camera trap imagery. There are a set of locations. The camera is kept in the same fixed spot in each location, and takes photos at different times. The task is to determine if there is an animal in the image and which animal is present. In *spurious correlation, low-data drift*, we resplit the dataset randomly (there are no held out locations). In *spurious correlation*, we correlate the presence of a given animal with a location (all images from a given location *only* show a given animal). In *low-data*, we select the locations that are out of distribution for Koh et al. [2020] as the locations for which we only have $N$ samples. In *unseen data shift*, we use the split given by Koh et al. [2020] and optionally use either the in-distribution or out-of-distribution validation set for model selection.

### G.2 Evaluated shifts

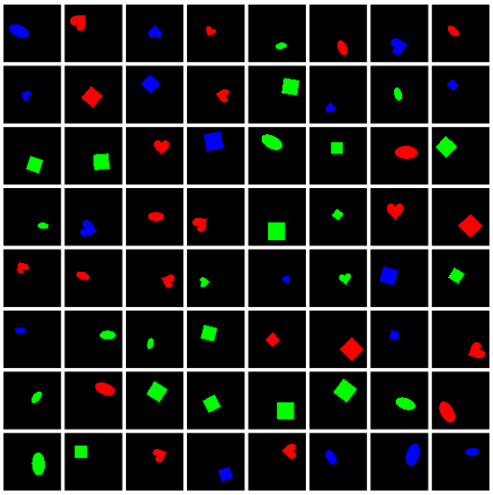
**Shift 1: Spurious correlation.** Under spurious correlation, we correlate $y^l, y^a$. At test time, these attributes are uncorrelated. We vary the amount of correlation by creating a new dataset with all samples from the correlated distribution in the dataset and $N$ samples from the uncorrelated distribution; this forms the training set. We set $N >= 1$ (as if $N = 0$, then the problem is ill defined as to what is the correct label). The test set is composed of sampling from the uncorrelated distribution and is disjoint from the training samples. Results are given in figure 2.

**Shift 2: Low-data drift.** Under low-data drift, we consider the set $\mathbb{A}^a$. For some subset $\mathbb{A}^a_c \subset \mathbb{A}^a$, we only see $N$ samples of those attributes. For all other values of $y^a$ ($\mathbb{A}^a / \mathbb{A}^a_c$), the model has access to all samples. Results are given in figure 3.

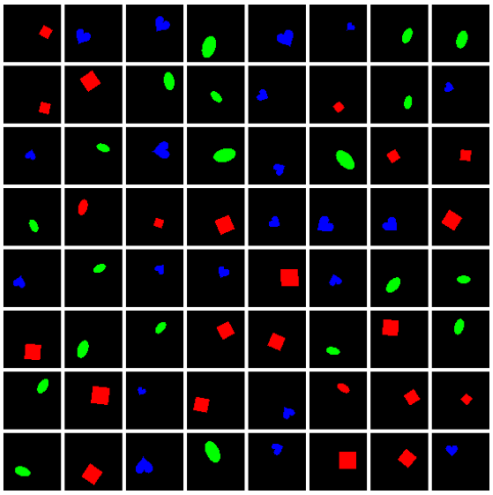
**Shift 3: Unseen data shift** This is a special case of *low-data drift*, where we set $N = 0$. Results are given in figure 4.

### G.3 Samples from the different distributions

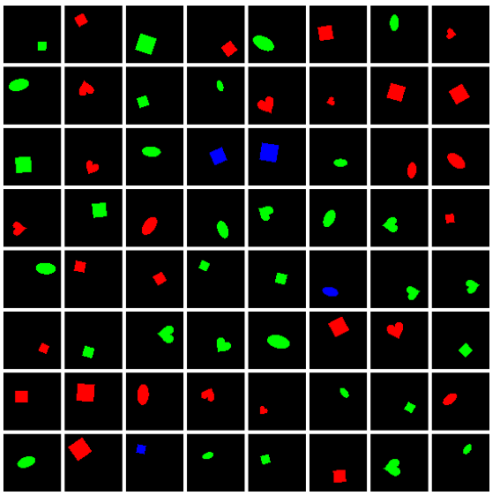
We include samples for each distribution for each dataset in figure 10-15. We describe each of the shifts in table 2. Note that the amount of correlation, samples from the low-data region, and the probability of sampling from the low-data or correlated distributions is controllable in our framework. Additionally, we can control the amount of label noise and total dataset size.



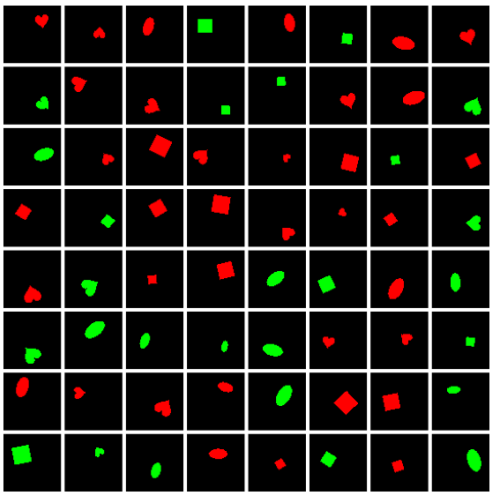
DSPRITES (IID test).



DSPRITES (**Shift 1: Spurious correlation.**)



DSPRITES (**Shift 2: Low-data drift.**)

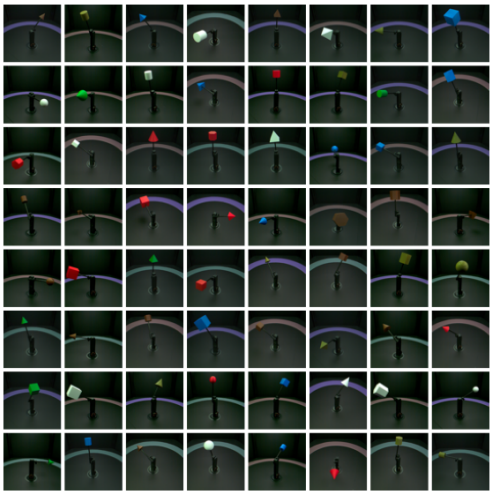


DSPRITES (**Shift 3: Unseen data shift.**)

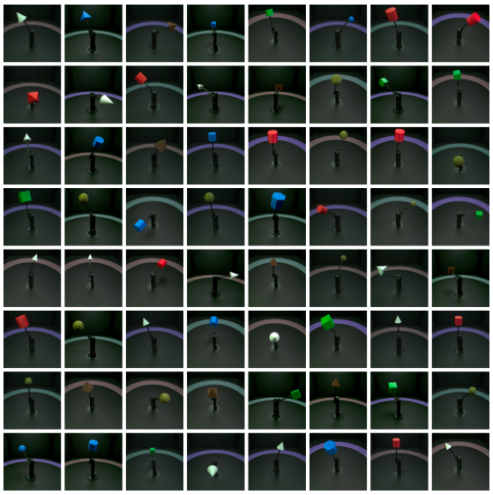
Figure 10: Sample distributions on DSPRITES.

| Dataset | label | nuisance attr | SC | LD / UDS ($\mathbb{A}_c^a$) |
|---|---|---|---|---|
| DSPRITES | $l$ = shape<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 3$ | $a$ = color<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 3$ | $v_i \sim u_i$ | {blue} |
| MPI3D | $l$ = shape<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 6$ | $a$ = color<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 6$ | $v_i \sim u_i$ | $\{u_i \mid i > 2\}$ |
| SHAPES3D | $l$ = shape<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 4$ | $a$ = obj. color<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 10$ | $v_i \sim u_i, i \leq 4$ | $\{u_i \mid i > 2\}$ |
| SMALLNORB | $l$ = category<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 5$ | $a$ = azimuth<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 18$<br><br>$b$ = lighting<br>$w_i \in \mathbb{A}^b$<br>$\lvert\mathbb{A}^b\rvert = 6$ | $v_i \sim w_i, i \leq 5$ | $\{u_i \mid i > 4\}$ |
| CAMELYON17 | $l$ = tumor<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 2$ | $a$ = hospital<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 5$ | $v_i = 0 \sim u_{i \in [1,2,3]}$<br>$v_i = 1 \sim u_{i \in [4,5]}$ | $\{u_i \mid i \in \{2, 3\}\}$ |
| IWILDCAM | $l$ = animal<br>$v_i \in \mathbb{A}^l$<br>$\lvert\mathbb{A}^l\rvert = 186$ | $a$ = location<br>$u_i \in \mathbb{A}^a$<br>$\lvert\mathbb{A}^a\rvert = 324$ | $v \sim u$ | $\lvert\mathbb{A}_c^a\rvert = 79$ |

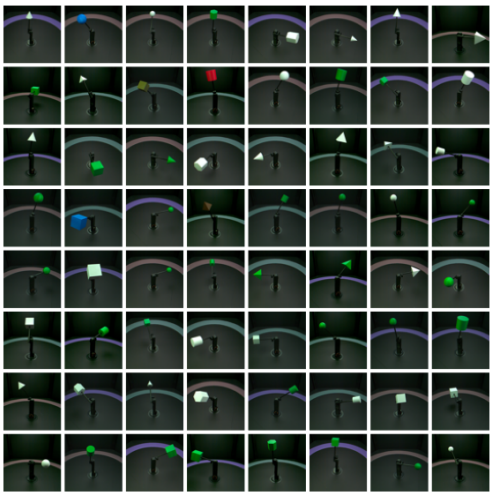
Table 2: The precise dataset shifts we use for each dataset. We describe the label and nuisance attribute for each dataset. We additionally describe the setup for *shift 1: strong correlation (SC)* and for *shifts 2/3: low-data drift (LD)/unseen data shift (UDS)*. $\sim$ denotes correlation. We denote each value in the attribute sets using 1-based indexing. For IWILDCAM, not all locations have all animals, so we find the most commonly occurring location for each animal and correlate that animal with that location. Additionally, for IWILDCAM and CAMELYON17, we use the OOD set from Koh et al. [2020] as $\mathbb{A}_c^a$. For IWILDCAM, there are 72 locations.



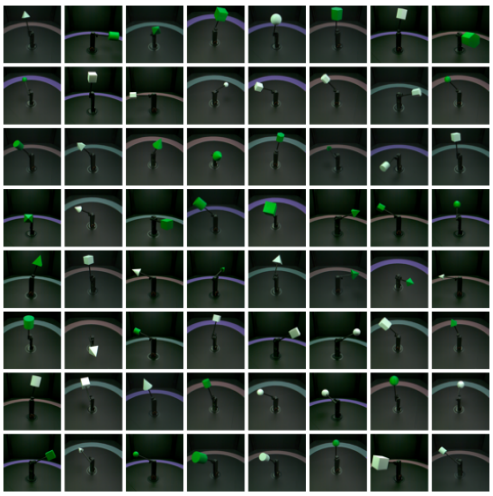
MPI3D (IID test).



MPI3D (**Shift 1: Spurious correlation.**)

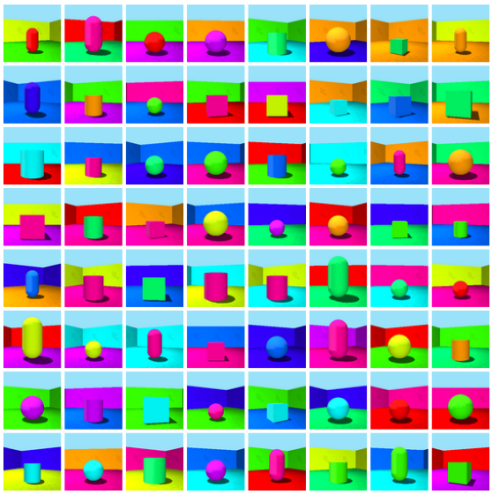


MPI3D (**Shift 2: Low-data drift.**)

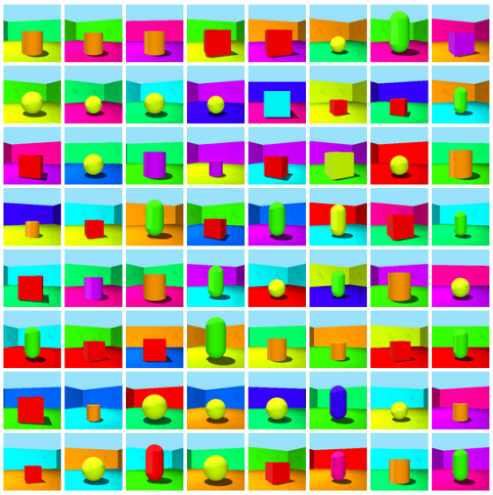


MPI3D (**Shift 3: Unseen data shift.**)

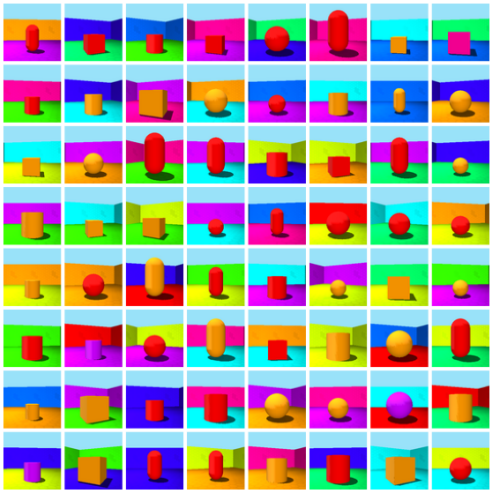
Figure 11: Sample distributions on MPI3D.



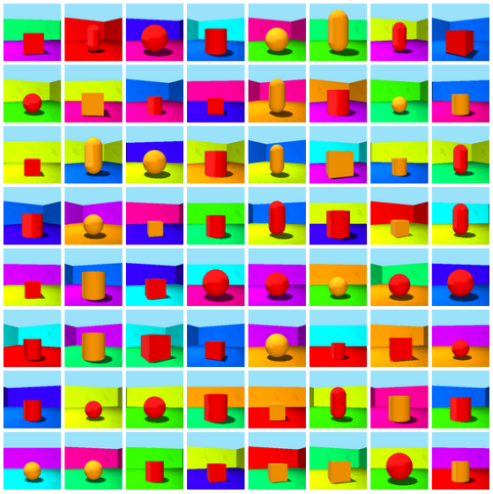
SHAPES3D (IID test).



SHAPES3D (**Shift 1: Spurious correlation.**)

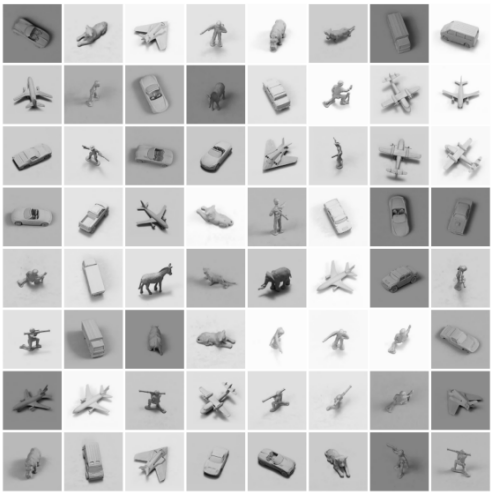


SHAPES3D (**Shift 2: Low-data drift.**)



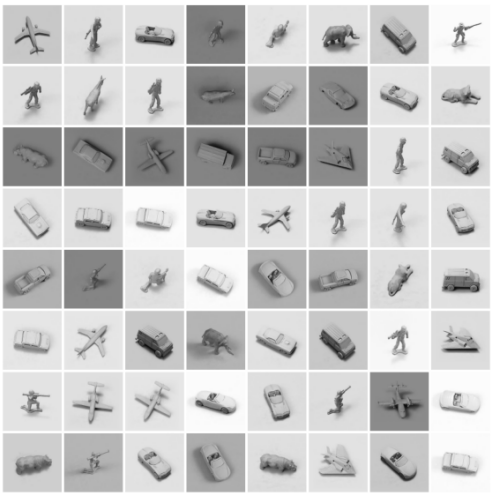
SHAPES3D (**Shift 3: Unseen data shift.**)

Figure 12: Sample distributions on SHAPES3D.

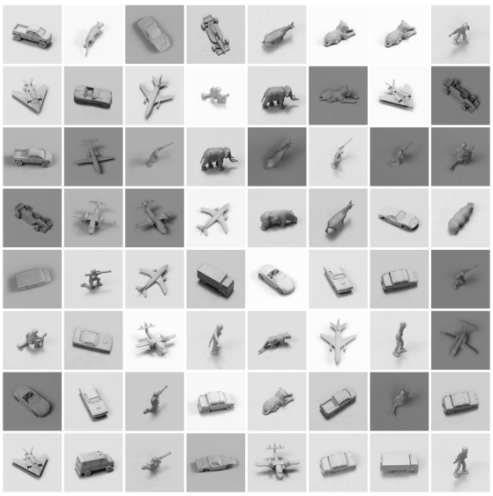


SMALLNORB (IID test).

SMALLNORB (**Shift 1: Spurious correlation.**)

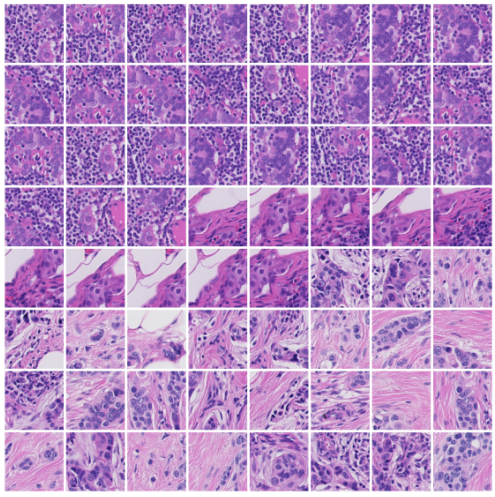


SMALLNORB (**Shift 2: Low-data drift.**)

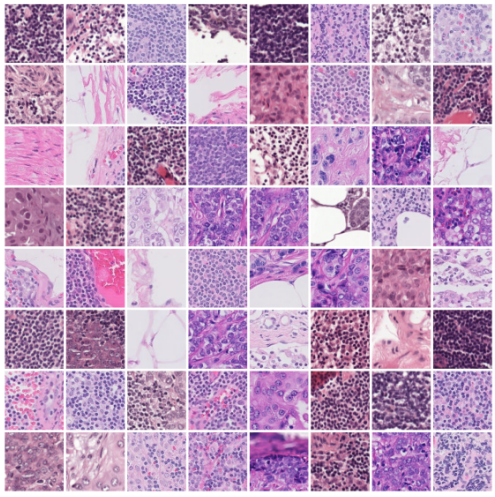


SMALLNORB (**Shift 3: Unseen data shift.**)

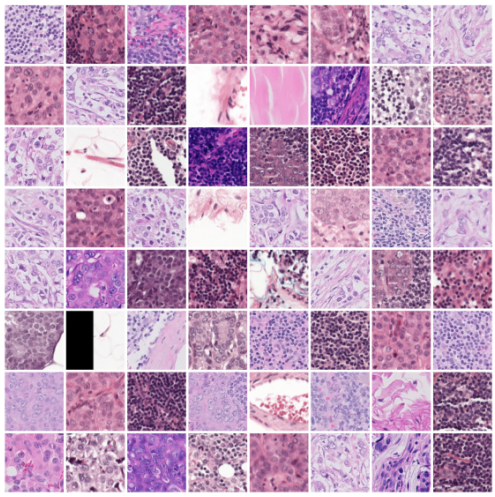
Figure 13: Sample distributions on SMALLNORB.



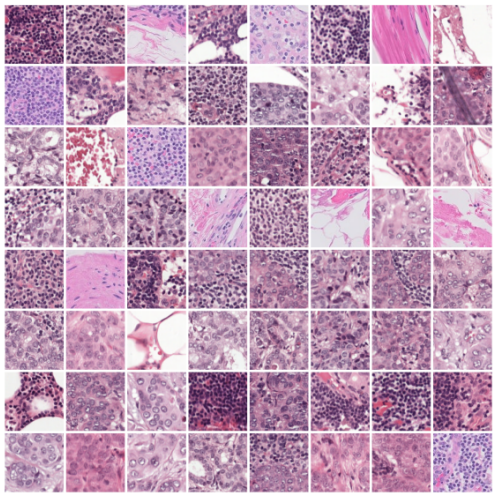
CAMELYON17 (OOD test).



CAMELYON17 (**Shift 1: Spurious correlation.**)

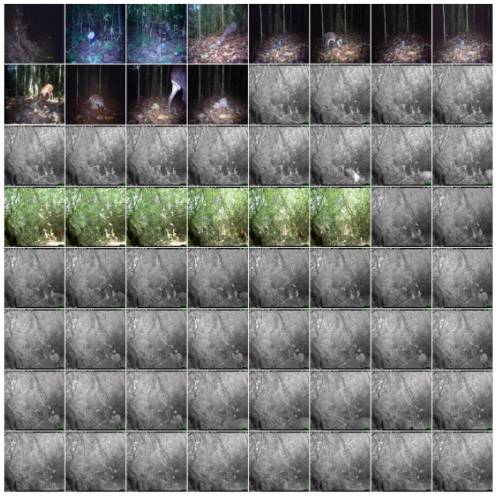


CAMELYON17 (**Shift 2: Low-data drift.**)



CAMELYON17 (**Shift 3: Unseen data shift.**)

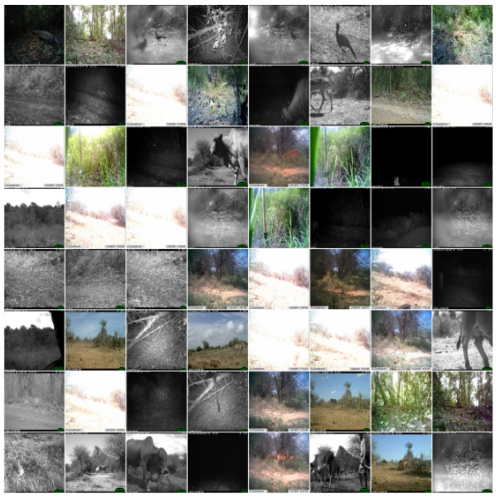
Figure 14: Sample distributions on CAMELYON17.



iWildCam (OOD test).

iWildCam (**Shift 1: Spurious correlation.**)

iWildCam (**Shift 2: Low-data drift.**)



iWildCam (**Shift 3: Unseen data shift.**)

Figure 15: Sample distributions on iWildCam.

| Model | # Parameters (M) |
|---|---|
| ResNet18 | 11.18 |
| ResNet50 | 23.51 |
| ResNet101 | 42.51 |
| MLP | 3.34 |
| ViT | 85.66 |

Table 3: The number of parameters for each model. While ViT has the largest capacity, it was the most brittle to train and only performed best on IWILDCAM. On the other datasets, the ResNets performed best.

# H Implementation

We first describe the experimental setup in appendix H.1 architectures and precise implementation of each approach in appendix H.2-H.7 and give training details in appendix H.8. We then give the sweeps over the hyperparameters considered in appendix H.9.

## H.1 Experimental setup

Here we describe the precise implementation choices. We do not claim that these are the best possible results obtainable with these methods (as performing sweeps over all possible settings would be computationally impractical) but that they are indicative of performance.

**Model selection.** When investigating heuristic data augmentation, domain generalization, learned augmentation, adaptive approaches, and representation learning, we use a ResNet18 for the simpler, synthetic datasets (DSPRITES, MPI3D, SHAPES3D, and SMALLNORB) but a ResNet50 for the more complex, real world ones (CAMELYON17 and IWILDCAM). To perform model selection, we choose the best model according to the in-distribution validation set ($\mathcal{D}_{\text{valid}} \subset \mathcal{D}_{\text{train}}$). In the *unseen data shift* setting for the CAMELYON17 and IWILDCAM, we use the given out-of-distribution validation set, which is a different, distinct set in $\mathcal{D}$ that is independent of $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$. (We consider using the in-distribution validation set in appendix D.4.)

**Hyperparameter choices.** We perform a sweep over the hyperparameters (the precise sweeps are given in appendix H.9). We run each set of hyperparameters for five seeds for each setting. To choose the best model, we perform model selection for each seed over *all* runs using the top-1 accuracy. In the low-data and correlated case, we choose a different set of samples from the low-data region with each seed. We report the mean and standard deviation over the five seeds.

## H.2 Base architectures

We train three types of models. These all have different capacities, which we report in table 3.

**ResNets.** We use the standard ResNet18, ResNet50, and ResNet101 setups [He et al., 2016].

**MLP.** For the MLP, we use a 4 layer MLP with 256 hidden units.

**ViT.** For the ViT [Dosovitskiy et al., 2021], we set the parameters as follows. For the smaller 64x64 images (DSPRITES, MPI3D, SHAPES3D), we use a patch size of 4 with a hidden size of 256. For the transformer, we use 512 for the width of the MLP, 8 heads and 8 layers. We use a dropout rate of 0.1. For the medium 96x96 images (CAMELYON17, SMALLNORB), we use a patch size of 12 and for the 256x256 images (IWILDCAM), a patch size of 16.

## H.3 Heuristic Augmentation

**ImageNet Augmentation.** ImageNet augmentation is composed of random crops and color jitter. We use the standard ratios as used in ImageNet training He et al. [2016]. We only apply the augmentation to the first three channels of a dataset (replicating across three channels if the data is grayscale).

**AugMix [Hendrycks et al., 2020].** AugMix composes multiple $k$ sequences of augmentations, randomly sampled from thirteen base augmentations. We use their default: $k = 3$.

**RandAugment [Cubuk et al., 2020].** RandAugment randomly samples $N$ augmentations from a set sixteen base augmentations with a severity $M$. For the augmentation parameters (cutout and translate) based on image size, we interpolate between the values used for ImageNet on 224 images and Cifar10 on 32 images. We set $N = 3, M = 5$.

**AutoAugment [Cubuk et al., 2019].** If the image size is less than 128 (e.g. all datasets but IWILDCAM), we use the cifar10 policy, else we use the ImageNet policy. Again, for the augmentation parameters (cutout and translate) based on image size, we interpolate between the values used for ImageNet on 224 images and Cifar10 on 32 images.

### H.4 Learned Augmentation

**CycleGAN [Goel et al., 2020, Choi et al., 2018, Zhu et al., 2017].** This approach proceeds in two stages. The first stage learns how to transform images of one attribute to have another attribute. In this stage, these models are trained to minimize three losses: a reconstruction loss $\mathcal{L}_r$, classifier loss $\mathcal{L}_c$, and adversarial loss $\mathcal{L}_a$. The final loss is a combination of these: $\mathcal{L} = \lambda_r \mathcal{L}_r + \lambda_c \mathcal{L}_c - \lambda_a \mathcal{L}_a$. We set $\lambda_c = 1, \lambda_r = 1$ and we sweep over $\lambda_a$. We train this model with a learning rate $\text{lr}_{\text{STARGAN}}$ (using the same learning rate for the generator and discriminator). We use the same ResNet style architecture as CYCLEGAN, except we append to the image a one hot encoding designating the original attribute, as described in STARGAN Choi et al. [2018]. We use five ResNet blocks for images of size 64 (DSPRITES, MPI3D, SHAPES3D), six ResNet blocks for images of size 96 (CAMELYON17, SMALLNORB) and nine ResNet blocks for images of size 256 (IWILDCAM).

### H.5 Domain Generalization

**IRM [Arjovsky et al., 2019].** IRM enforces that the optimal classifier for all domains is the same. This is done using two terms in the risk: a term to minimize the overall risk and a term to enforce the constraint, with a tradeoff $\lambda$.

**DeepCORAL [Sun and Saenko, 2016].** DeepCORAL enforces that the mean and covariance of the learned features in different domains is the same. This is achieved by two terms in the loss: a term to minimize the overall risk and a term to penalize differences in the mean and covariance across each pair of domains. These are weighted according to a tradeoff $\lambda$.

**Domain MixUp [Gulrajani and Lopez-Paz, 2021].** Domain MixUp enforces smoothness in the label prediction by interpolating between images from different domains and enforcing the prediction similarly interpolates between the labels. We set the interpolation parameter $\lambda = 0.2$ (as in MixUp Zhang et al. [2018]).

**DANN [Ganin et al., 2016].** DANN trains an adversary $g$ that attempts to predict the attribute from the learned representation and the model seeks to fool this adversary while also minimizing downstream accuracy: $\mathcal{L} = \mathcal{L}_c(l(z)) - \lambda_a \mathcal{L}_a(g(z))$. For the adversary, we use a two layer MLP with ReLU [Nair and Hinton, 2010] with hidden sizes of size 64.

**SagNet [Nam et al., 2021].** SagNet aims to learn a representation $z$ that is invariant to style by using an adversary. The classification loss $\mathcal{L}_c$ aims to use the content predictor $c$ to classify $z$. The adversary $\mathcal{L}_a$ aims to use the style predictor $s$ predict the class. This give a final loss: $\mathcal{L} = \mathcal{L}_c(c(z)) - \lambda_a \mathcal{L}_a(s(z))$. We use the feature extractor of a ResNet18 [He et al., 2016] (before the average pool) as the representation $z$. For the content $c$ and style predictor $s$, we use MLPs with three hidden dimensions of size 64 and ReLUs. We set $\lambda_a = 1$.

### H.6 Adaptive Approaches

**JTT [Liu et al., 2021].** JTT uses a pretrained classifier to find the most challenging samples. It then reweights these samples by $\lambda$. We set $\lambda = 20$ and train the first model for half the training time.

**BN-Adapt [Schneider et al., 2020].** BN-Adapt adapts the parameters of the original model according to the attribute label using a hyperparameter $r$ ($N/n$ in their paper). Intuitively, this controls the the strength of the original model $r$ and the new model 1. We set $r = 10$.

### H.7 Representation Learning

**$\beta$-VAE [Higgins et al., 2017a].** $\beta$-VAE poses the original VAE objective using constrained optimisation to obtain a new objective which have a trade-off $\beta$ between the log-likelihood objective and the KL objective. We resize images to 64x64 and use a convolutional architecture as in Higgins et al. [2017a]. The encoder consists of the following layers (with ReLUs) to obtain the final representation $z$ of size $L$: Conv 16x5x5 (stride 1, image size: 60), Conv 32x4x4 (stride 2, image size 29), Conv 64x3x3 (stride 1, image size 27), Conv 128x3x3 (stride 2, image size 13), Conv 256x4x4 (stride 1, image size 10), Conv 512x4x4 (stride 2, image size 4), Conv 512x4x4 (stride 1, image size 1), Linear $L$. The decoder has the reverse setup to the encoder with convolutional transpose layers.

There are two terms to trade-off: the strength of $\beta$ and the size of the latent representation $L$. However, a larger latent will need a stronger $\beta$. We use the $\beta_{\text{norm}} = \frac{\beta L}{I}$ which normalises these values by the data size $I$. We then sweep over $L$ and $\beta_{\text{norm}}$.

Finally, to train the downstream classifier, we fix the encoder of the $\beta$-VAE. We then train an MLP (the same as used in appendix H.2) from the representation of size $L$ to the labels.

**Pretrained on ImageNet.** We resize images to 224x224 and pass them to a model pretrained on ImageNet. We finetune the full network.

### H.8 Training.

We train the ResNets and MLP with the Adam optimizer for 100K steps on the synthetic datasets and 200K steps on CAMELYON17 and IWILDCAM with a batch size of 128. The ViT is trained with a batch size of 1024 (it did not converge for smaller batch sizes).

### H.9 Sweeps.

The sweeps are given in table 4.

| Method | Sweeps |
|---|---|
| ResNets | learning rate: [1e-2, 1e-3, 1e-4] |
| MLP | learning rate: [1e-2, 1e-3, 1e-4] |
| ViT | learning rate: [1e-2, 1e-3, 1e-4] |
| ImageNet Augm | learning rate: [1e-2, 1e-3, 1e-4] |
| AugMix | learning rate: [1e-2, 1e-3, 1e-4] |
| RandAugment | learning rate: [1e-2, 1e-3, 1e-4] |
| AutoAugment | learning rate: [1e-2, 1e-3, 1e-4] |
| CycleGAN | learning rate: [1e-4]<br>$\lambda_a$: [0.01, 0.1, 1, 10]<br>$\text{lr}_{\text{StarGAN}}$: [1e-3, 1e-4, 1e-5] |
| IRM | learning rate: [1e-2, 1e-3, 1e-4]<br>$\lambda$: [0.01, 0.1, 1, 10] |
| MixUp | learning rate: [1e-2, 1e-3, 1e-4] |
| CORAL | learning rate: [1e-2, 1e-3, 1e-4]<br>$\lambda$: [0.01, 0.1, 1, 10] |
| SagNet | learning rate: [1e-2, 1e-3, 1e-4] |
| DANN | learning rate: [1e-2, 1e-3, 1e-4]<br>$\lambda_a$: [0.01, 0.1, 1, 10] |
| JTT | learning rate: [1e-2, 1e-3, 1e-4] |
| BN-Adapt | learning rate: [1e-2, 1e-3, 1e-4] |
| $\beta$-VAE | learning rate: [1e-4]<br>$\log(\beta_{\text{norm}})$ : linspace(1e-4,10,4)<br>$L$ : linspace(10,210,4) |
| Pretrained on ImageNet | learning rate: [1e-2, 1e-3, 1e-4] |

Table 4: The sweeps over each method for each of the five seeds. The maximum total sweep size (due to capacity) is eight models per seed. For each seed, the best model (based on the ID or OOD validation set) is used for evaluation.