Evading the Simplicity Bias: Training a Diverse Set of Models Discovers Solutions with Superior OOD Generalization

Supplementary material

A. Past reviews and FAQ

We include questions and comments received during the reviewing process along with our answers. We hope that they will be helpful to readers.

Q1: Where to split a model into "feature extractor" and "classifier" ? The separation between the feature extractor and the classifier is somewhat arbitrary. In our experiments, it was dictated by computational reasons. Using a pre-trained ResNet as the feature extractor was a natural choice. The subsequent classifiers need sufficient width/depth such that they can model different functions on top of the features. The choice of 2-hidden-layer MLP was a compromise between expressibility and computational cost. A possible extension of this work would be to evaluate deeper classifiers or complete replicated models.

The choice of the "extractor/classifier" separation also affects the dimension along which the gradients are compared. For example with ResNet, features undergo a global spatial pooling before being passed to the classifier. The gradients are thus compared only across channels, not spatial dimensions. In our experiments with collages, the feature extractor is the identity function and the classifier is a fully-connected network operating directly on the pixels. The gradients are thus compared across spatial dimensions

Q2: Why not design the diversity regularizer on the activations of the models rather than on the input gradients ? Because the activations in different models live in "different spaces" so they are not directly comparable. Consider any chosen layer: the activations do not tell much about the function implemented by the whole network. Whereas the input gradients (through the whole network) do.

The activations also cannot be used with a simple dot product to compare different models. For example, one can rearrange the weights of a model to permute the channels of the activations without actually changing the function implemented by the whole network.

Q3: Is the introduction of more diversity just a fancy random search ? The common training by SGD is already stochastic. It is not uncommon to restart training by SGD with different random seeds if training does not converge. But simply using different random seeds is not sufficient to evade the simplicity bias however, whereas the proposed method is.

The proposed method does not bring any additional source of randomness. It makes solutions found by SGD from different initializations more different from one another.

Q4: Why not use the Colored MNIST toy data ? A number of recent works have use "Colored MNIST" toy data introduced in [1] to evaluate OOD generalization. We believe it is an overly simplistic setting. The so-called "color" only means that the MNIST pixels are stored on one discrete channel or another. This is like operating on symbolic data, or on perfectly disentangled representations. And learning disentangled representations from high-dimensional data is itself a largely unsolved problem. Our collages achieves a similar objective (multiple predictive signals of different complexity) but it is more challenging and representative of real data.

Q5: Why not use dataset [X] ? We chose a few datasets representative of OOD challenges in computer vision (multiple predictive signals, biased data, generalization across visual styles). We had to choose datasets with an established OOD test set. For the reasons explained at length in the paper, no improvement is expected with identically-distributed training and test sets.

Q6: Why doesn't in-domain (ID) performance necessarily improve when OOD performance does ? Because the irreducible error with robust features can be larger than with spurious ones. This is not uncommon and it often makes improvements in OOD performance come at the expense of ID performance.

To do well ID, a model can exploit any predictive pattern in the training data, including spurious correlations. For example, birds may be reliably recognized in the training data by detecting a blue background. However, relying on blue backgrounds is not desirable for an OOD bird detector. A good OOD model would instead rely on shape, such that it can recognize birds in any scene. But shapes are more often ambiguous, so in ID scenes (where the blue color is a reliable indicator of birds) the OOD model will not be as accurate as a blue-background detector.

Q8: ***"In stage one, there is no guarantee that simple biases (i.e. spurious features) and complex patterns (i.e. robust features) can be disentangled. Could each model in the collection be a mixture of simple biases and complex patterns ?"*** It is correct that no semantically or causally-meaningful disentanglement can be guaranteed. What can be guaranteed is that the models trained in a large-enough set cannot be decomposed into "simpler", locally independent functions. Indeed, [65] showed that local independence enforced with orthogonal gradients leads to the recovery of sparse predictors. As one increases the number of predictors fitted in parallel, they approach a "maximal set" as defined in [65]. This implies functional simplicity, in the sense that each predictor in a maximal set cannot be fur-

ther decomposed into a combination of locally independent functions.

Q8: ***"Is it possible to train all 96 MLP classifiers simultaneously (in parallel) on a single GPU ?"*** Indeed we had no issue training 96 MLPs on a laptop GPU, thanks to the sharing of mini-batches across models.

Q9: ***"In Table 1, can you explain the decrease in accuracy when increasing the number of models from 8 to 16 ?"*** We attribute this decrease in accuracy to evaluation noise since it is well within the standard deviation across runs.

Q10: ***"Can you elaborate on the connection between this paper's findings and the simplicity bias ?"*** The connection follows from the recent evidence [70] that the simplicity bias is an important source of poor OOD generalization (studied e.g. with MNIST/CIFAR collages in [70]). Our study is however not strictly tied to the simplicity bias: the proposed method improves the sampling of the hypothesis space, regardless of the default solution being explained by the simplicity bias or by other effects, such as neural anisotropies for example [52].

Q10: ***"Can the method apply to complete ResNet-scale models ?"*** The extension to full models involves no conceptual leap since the regularizer operates in the space of representations (input or latent), not of weights. The applicability of our findings to simple models with pretrained features is certainly useful in itself. We haven't addressed the tuning of full-scale models so far due to our limitations in computational resources.

Q11: ***"By relying on model selection to select the desired inductive biases, could the method be at greater risk of adaptive overfitting, since it requires more evaluations on the test set than other OOD approaches ?"*** Model selection is not limited to cross-validation, other choices include contrast sets [16], calibration-based methods [80], model explainability with expert knowledge, etc. If cross-validation is used in a real application, OOD **validation** data would be used, never the test set data itself obviously. Our results under "oracle selection" serve to provide an upper bound on achievable OOD performance (i.e. with perfect model selection).

Q12: Clarification of some definitions. Spurious correlations: statistical pattern in the data that does not reflect a causal relationship (*i.e.* not guaranteed to transfer OOD) but which results from confounding (*e.g.* selection bias). **Inductive biases**: assumptions made by a learning algorithm about the nature of the target function to enable finite-sample generalization.

B. OOD Generalization and causality

The ultimate goal of supervised machine learning is to learn a model that mimics a real-world process that jointly determines the values of an observed variable X and target variable Y. It makes sense to learn a task when the process relates these variables with causal ($X \rightarrow Y$) mechanisms, such that the conditional $P(Y|X)$ is a property of the task: it will always remain the same across training and (OOD) test conditions. The OOD setting used throughout this paper is also called *covariate shift*: $P(Y|X)$ is constant and only the marginal $P(X)$ varies across training and test sets. For example, X can represent images being drawn as photographs during training and paintings during testing.

To achieve OOD generalization across arbitrary covariate shifts, it is necessary that the causal mechanisms of inference (within the learned model) mirror the causal mechanisms of the data-generating process. In other words, features responsible for the prediction of a label by the model should be the same as those causally related to this label in the training data. However, causal properties of the real-world process are *not* properties a joint distribution over (X, Y) of training examples [68]. **The information necessary for OOD generalization is lost** by drawing i.i.d. training samples from the data-generating process. This is why optimizing a model for ERM cannot generally achieve the above condition. **Even infinite amounts of training data** cannot bring back this missing information.

Note that all of the above is true for arbitrary tasks and data distributions. If task- or dataset-specific knowledge is available, it sometimes sufficiently circumscribes the space of possible ground-truth data-generating processes to allow recovering some causal properties from observational data (see examples with images [40] and time series [33]). The method of this paper makes use of no such prior knowledge.

C. Project chronology and negative results

This section is an informal chronology of the developments that led to this paper, including things we tried that did not work.

Initial motivation. The initial motivation for this work was to learn patterns that a model would not learn by default because of the simplicity bias. The closest existing works were "debiasing methods" popular for visual question answering [7,10,31] and NLP [3,11,41,71,77]. They typically train one model that is biased by design (for example being fed a partial input) while a second model is subsequently trained to be different, hence more robust. Our first innovation was to enforce this difference in the space of input gradients of these models (rather than in the space of their activations, weights, etc.). This quickly appeared effective and easier to train than adversarial objectives of many debiasing methods. A later literature search showed that input gradients had previously been used in similar [32] and other applications [14].

To improve over existing debiasing methods, we ex-

tended our approach to >2 models and removed the reliance on a "weak" first model by design. Instead, we used the same architecture for all models and realized the simplicity bias would make any model "weak" by default. The publication of multiple studies in late 2020 related to the simplicity bias encouraged pursuing with this approach.

Parallel training. Our first implementation used sequential training of multiple models, inspired by existing debiasing methods. We implemented a parallel version as a baseline, but it quickly proved more effective, to our surprise. Despite much effort, the sequential scheme could not equate the parallel version. We gave up the former after coming up with a satisfying intuitive explanation. The sequential training makes each model only marginally different from the previous one (for example, with the collages, every model would use another pixel of the MNIST digit, but would never focus on a completely different part of the images). This holds even after training a very large number of models, and whether the diversity regularizer is applied on the last two models, or on the whole collection of models trained so far. In contrast, the parallel training, using with pairwise constraints between all models simultaneously, could produce multiple potentially-good models at once.

Similarity of gradients. To achieve the goal of maximizing the diversity of the models, we designed many elaborate measures of similarity between the gradients. The intuitive goal was to "spread" the learned solutions evenly within the space of predictors. However, none of these alternative measures worked better than the sum of pairwise dot products described in this paper. Alternatives that we tested include cosine distances, the determinant of a matrix of dot products (similar to determinantal point processes or DPPs) or of other kernel function of the dot products, a soft approximation (logSumExp) of the maximum of the pairwise dot products (rather than the overall sum).

We also tried using the gradient w.r.t. the spatial input (all input pixels), or w.r.t. intermediate representations in CNNs. We also tried all of these as gradient-feature products (in the style of the Grad-CAM method). The bare gradients worked great with the collages, and the gradient-feature products worked clearly better with ResNet as the feature extractor. Our larger-scale experiments therefore all use the gradient-feature products.

We also tried applying various normalizations (L0, L1, L2, softmax: none worked) and rectifications to the gradients (absolute value, ReLU/positive part, negative part, square: the square did not work at all, but all other options performed similarly).

As described in the paper, we use the gradient of the top predicted score. Using the gradient of its corresponding logit (before sigmoid rectification) seems to work equally well. We tried alternatives: the gradient of the score predicted for the ground truth class, or the gradient of the classification loss. Both performed worse.

Datasets. We started with toy data (32-dimensional vectors of numeric values generated with known functions) in the style used in [55]. We then moved to colored MNIST digits. This dataset proved useless since the signal is perfectly disentangled across two input channels, which is ridiculously simple and unrealistic. We found the multi-dataset collages [70] the best compromise between toy and real data. We then moved to the real datasets PACS and BAR. Overall, most of our developments were done with the collages and PACS (using *art_painting* as the test style because it seemed to be the PACS setting with the clearest possible improvements, from results of other methods).

D. Related work

We provide below an extended literature review. Since this paper addresses a central problem in machine learning, it touches many well-established research areas.

Importance of OOD generalization. Failure to generalize OOD is the root cause of many limitations of machine learning: adversarial attacks [26], some model biases [49], failure to generalize across datasets [76], etc. Poor OOD generalization is only apparent and problematic with OOD test data. Academic benchmarks have traditionally been built with i.i.d. training and test samples. This rarely holds in the real world, and OOD is closer to the norm in real-life deployments of machine learning models.

Evaluation with i.i.d. training/test sets hide a model's limitations because spurious correlations and biases in the training data also exist in the test data. Thanks to the increasing awareness of issues of robustness with deep learning, many benchmarks now include OOD (a.k.a. "challenge") tests sets [16, 20]. OOD evaluation can also be done by cross-dataset evaluation without fine-tuning (*i.e.* zero-shot transfer) [60]. The assumptions then is that the spurious patterns in different datasets are uncorrelated.

Improving OOD generalization. Given its central place, OOD generalization is addressed from multiple angles by multiple communities making different assumptions. **Domain generalization** methods [21] use multiple domains during training. **Domain adaptation** methods use unlabeled data from a second domain for rapid adaptation at test time. **Debiasing** methods use expert knowledge of the spurious correlations to prevent the model from using them. **Adversarial training** methods use expert knowledge of the type of statistical patterns that are undesirable to learn, or notions of smoothness and continuity that a model should exhibit. Other training objectives have also been proposed

to use **interventional data** such as counterfactual examples [25, 73] or non-i.i.d. datasets like non-stationary time series [23, 30, 58]. The common point to all approaches that improve OOD generalization is that extra knowledge is provided, either as expert task-specific knowledge, or as non-i.i.d. data. This corroborates the point made throughout this paper that i.i.d. data alone (even in infinite quantity) cannot improve OOD generalization.

Domain generalization. The goal of domain generalization (DG) is to learn models that generalize across visual domains such as photographs, sketches, paintings, etc. Images from multiple domains (a.k.a. training environments) are provided for training. The model is then evaluated on one held-out domain. The training environments can be formalized as different interventions on the data-generating process. This was shown to carry the kind of information required for OOD generalization [1, 56]. Intuitively, DG methods discover features of the input that are "common" and similarly-predictive across the environments. The principle is sound if a large number of training domains is available but this is not the case with existing datasets. PACS for example provides only three training domains. Although some of the information necessary for OOD generalization is theoretically there, the learning problem is still very ill-defined because of this large distribution shift between the training domains. Practically, this leaves much room to apply various inductive biases. This explains the plethora of methods already developed for this dataset. Because the problem is ill-defined, the effectiveness of any such method can only be assessed when confronted with the test domain (and this is what we also need to do after training a collection of models with our method).

Gulrajani and Lopez-Paz [21] discussed the practice of model selection using the test domain. They observed that no existing method performed better than ERM **when access to the test domain is restricted**. This is unsurprising to us: this follows from the fundamental need, to achieve generalization, of substantial knowledge about the relation between training and test distributions. And this information is unlikely to be available from a handful of disparate training domains.

Our method does no require the labels of training environments used by DG methods. Our setup is more similar "single-source" domain adaptation [59, 79]. These methods augment the training data using a generative model to expand the region of feature space in which the predictions of the model are "stable". Consequently, these methods cannot make the model use features of the data it was not using in the first place. Thus there is no hope to counter the simplicity bias.

Debiasing. Methods for debiasing are concerned with improving generalization of models against a precisely identified (undesirable) factor of variation in the data [3, 11, 41, 48, 71, 77]. In computer vision for example, this can be removing the bias towards texture in the ImageNet dataset [5, 18]. In NLP, this can be removing the "hypothesis-only bias" of entailment models, that make these models guess an answer without considering the whole input [11].

Debiasing is relevant to this paper because most methods rely on training multiple models that each use the input differently. The source of improvement is the explicit specification of the factor of variation to be be invariant to. Typically, debiasing methods train a pair of models to respectively focus or ignore it. The latter model is used at test time. For example in [3], a first CNN model is trained with an architecture providing a small receptive field, such that is focuses on local texture. A second CNN is then trained with a larger receptive field while a regularizer makes its activations uncorrelated with the first one's, such that it focuses on overall shape more than on texture. Variations of the method include the extension to more than two models [71]. In comparison to our work, debiasing methods require the explicit specification of a factor of variation to ignore and they require it to be easily disentangled from other features of the input.

Some debiasing methods claim to require **no explicit knowledge of the bias** [11, 67, 78]. They actually make this knowledge only less explicit: the authors design the architecture of the models such that the weak learner is forced to use the bias (through limited capacity, receptive field, etc.). Our method does not rely on heterogeneous architectures and is applied to many more models. We also found that parallel training of multiple models was much more effective than the sequential training used with most debiasing methods.

Encouraging diversity within one model. We can draw parallels between the method in this paper and existing methods that encourage a notion of diversity. Classical feature selection approaches [22] are related but they not suitable to deep learning model and high-dimensional representations. For example, SCOPS [29] performs self-supervised part discovery using an objective of orthogonality (akin to diversity) between parts. In comparison, we use diversity as an objective alongside predictive performance. Diversity was also used an objective during model compression, for fusing redundant neurons with similar activations [43]. Closer technically to our method, [14] uses the cosine similarity of gradients of multiple losses to measure their mutual correlation in the context of multitask learning. Previous works [32, 53] proposed to promote diversity in the space of learned representations within a model. Our approach is different in that we promote diversity across multiple in-

dependent models. These works focus on synthetic data and adversarial robustness while we show improvements on multiple benchmarks with real data.

Encouraging diversity within ensembles. Ensembling several models is a common technique to improve predictive performance over a single model. The diversity of the models in an ensemble is important [86] and usually promoted by training models with different hyperparameters or random seeds, or by enforcing diversity in the space of weights of the models [84]. In comparison, our diversity loss operates in the space of the gradients of the models. They need to be differentiable w.r.t. their input but their implementation as neural networks is irrelevant. For **adversarial robustness**, ensembles have shown benefits. Both [2] and [54] encourage diversity in the distributions of the models' logits. [32] minimizes the cosine similarity between gradients of the models. For **domain generalization**, [9] learns an ensemble of classifiers on CNN features. Each classifier is trained on a different visual domain and they promote diversity by minimizing the overlap between features used by each model, such that each specialize to one domain.

The multiverse loss [38, 42] improves transfer learning by duplicating a cross-entropy loss over multiple linear classifiers with an orthogonality constraint on their weights. It was shown to increase the number of distinct discriminative directions of the learned representation. It can be seen as a special case of our method.

In [64], the authors use input gradients to sequentially train multiple copies of a model to focus on different input features. The authors however mentioned in private communication that "sequential training doesn't work in most cases" which has also been our experience (see our negative results in the appendix). Their experiments are limited to toy datasets. After the writing of this paper, we realize that the same authors subsequently honed in on a parallel training scheme and diversity regularizer very similar to ours [63, 65], using a cosine similarity of gradients. Our motivation and experiments are very different and we invite the reader to consult those papers for a complementary view.

In [61], the authors train a generative model (HyperGAN) of parameters for a chosen network architecture. Their goal is to produce a diverse set of models, which they enforce and evaluate in parameter space. We think that our use of input gradients is more implementation-invariant and better capture the overall function implemented by deep models. We also believe that our evaluation with OOD tasks better captures the functional diversity of the learned models.

The earliest use of input gradients of neural networks was proposed by Drucker and LeCun [13] as "double back-propagation" to improve in-domain generalization. Almost identical formulations were described in many subsequent papers; see [28] and citations therein.

E. Experimental details

Collages dataset. We use images from MNIST, Fashion-MNIST, CIFAR-10, SVHN. The images are converted to grayscale. The images from MNIST and Fashion-MNIST are padded to 32×32 pixels. We pre-select two classes from each dataset to be respectively associated with the collages 0 and 1 labels. We follow [70] and choose 0/1 for MNIST, automobile/truck for CIFAR-10, and additionally choose 0/1 for SVHN and pullover/coat for Fashion-MNIST. We generate a training set of 51,200 collages, and multiple test sets of 10,240 collages each. Each collage is formed by tiling four blocks, each containing an image chosen at random from the corresponding source dataset. The images in our training/evaluation sets are selected respectively from the original training/test sets of the source datasets.

We propose two versions of the dataset. An *ordered* version, where the four blocks appear in constant order, and a *shuffled* version where the order is randomized in every collage. The shuffled version can be used to demonstrate that a given method does not rely on a known or constant image structure.

In the training set, the class in each block is perfectly correlated with collage label. In each of the four test sets, the class in only one block is correlated with the collage label. The other blocks are randomized to either of its two possible classes. We also generate four training sets in this manner, used solely to obtain upper bounds on the highest accuracy achievable on each block with a chosen architecture.

Collages experiments. We use the *ordered* version of the dataset. This allows generating the visualizations of Figure 4 and the use of a simple fully-connected classifier. We initially downsample the images by a factor 4. In our models, the feature extractor is the identity function and the classifier is a fully-connected MLP with two hidden layers of size 16 with leaky ReLU activations (leak rate: 0.01). The classifier is followed by a sigmoid and trained to minimize a standard binary cross-entropy loss. Training is performed by SGD with Adam, a learning rate of 0.001, mini batches of size 256, for a fixed number of 65 epochs (13k iterations) with no early stopping. The diversity regularizer is implemented as described in the paper. The visualizations in Figure 4 are obtained with the same model trained on images downsampled by a factor 16. The input gradient is evaluated and averaged on randomly selected test images. They are then upsampled by bilinear interpolation to the original collage dimensions of 64×64 for visualization purposes (without upsampling, they obviously look more "blocky").

BAR Experiments. We follow [48] and use frozen features from a standard pretrained ResNet-50. We train a 2-hidden-layer MLP classifier on these features. We spent some effort optimizing this baseline to the point of almost equating the method proposed in [48] (they only used a linear classifier on ResNet features). We use Adam, a learning rate of 0.001, a batch size of 256, and hidden layer dimensions of 512. We applied our method on this strong baseline.

As usually done with ResNet, feature maps are globally pooled before the classifier. Thus, unlike the experiments on collages, the features $\boldsymbol{h}$ have no spatial dimensions. The input gradients are compared across channels instead. We implement this with a variant of Eq. (4) inspired by the Grad-CAM method [69], in which we multiply the gradients (with respect to the features) by the features themselves:

$$\delta_{g_{\phi_1}, g_{\phi_2}}(\boldsymbol{h}) \;=\; (\boldsymbol{h} \nabla_{\boldsymbol{h}} g^{\star}_{\phi_1}).(\boldsymbol{h} \nabla_{\boldsymbol{h}} g^{\star}_{\phi_2})\,. \qquad (7)$$

Each model is trained for 200 iterations with no early stopping. The optimal weight of the regularizer is found by selection on the OOD test set. For the reasons explained at length in the paper, we found no reliable strategy to tune it without access to the OOD test set. This strategy is used for both our method and all other regularizers, ensuring a fair comparison.

PACS Experiments. We use a standard ResNet-18 as the feature extractor like most current methods. [21] showed that a ResNet-50 could slightly improve performance but we did not have the computational resources to do so, and the results of most methods to compare ours with also use ResNet-18. We first fine-tuned a ResNet-18 in the standard "ERM" setup (aggregated data of three training domains, linear output layer on top of ResNet features, Adam optimizer, learning rate of $4e{-}5$, batch size of 32, with augmentations described in [21], and early stopping based on test set accuracy). We found that training with a sigmoid activation and binary cross-entropy loss was slightly better than the usual softmax. All our PACS models (baselines and others) therefore use a sigmoid output. All of these choices provided a very strong baseline on which to test our method. Our baseline is noticeably stronger than those in existing papers as noted in Table 4. Demonstrating an improvement over a strong baseline is obviously more challenging than over a weaker one.

Our method was with a two-hidden-layer MLP as the classifier, fed with frozen features from the fine-tuned ResNet-18 (hidden size of 512, leaky ReLUs of rate 0.01, Adam, learning rate $3e{-}5$, batch size 256). We use the input gradient over channels (not over spatial dimensions) described above (Eq. 7).

Existing regularizers. We describe below the existing regularizers reported in Tables 1, 2, and 3.

1. **Dropout.** Used on collages only. Dropout is applied on input images *i.e.* on pixels of the quarter-size images fed to the MLP. We tuned the dropout rate, hoping that very high dropout rates would force the model to learn different parts of the image, but it did not work.
2. **Penalty on L1 norm of gradients.** This adds the following term to the minimization objective: $||\nabla_{\boldsymbol{h}} g||_1$.
3. **Penalty on L1 norm of feature-gradient product.** Variant that uses the same product as our regularizer with BAR and PACS (Eq. 7): $||\boldsymbol{h} \nabla_{\boldsymbol{h}} g||_1$.
4. **Penalty on squared L2 norm of gradients.** Also known as Jacobian regularization [28], it adds the following term to the minimization objective: $||\nabla_{\boldsymbol{h}} g||_2^2$.
5. **Penalty on squared L2 norm of ReLU of gradient.** Variant that only uses the positive coordinates of the gradient: $||\,\mathrm{ReLU}(\nabla_{\boldsymbol{h}} g)||_2^2$. The rationale is that this variant is "half-way" like the feature-gradient product described below which also masks some coordinates of the gradient (the features come from a ReLU and have a number of coordinates equal to zero).
6. **Penalty on squared L2 norm of feature-gradient product.** Variant that uses the same product as our regularizer with BAR and PACS (Eq. 7): $||\boldsymbol{h} \nabla_{\boldsymbol{h}} g||_2^2$.
7. **Penalty on squared L2 norm of logits.** Also known as spectral decoupling [57], it adds the following term to the minimization objective: $||g||_2^2$.

All penalty terms are summed over all training examples.

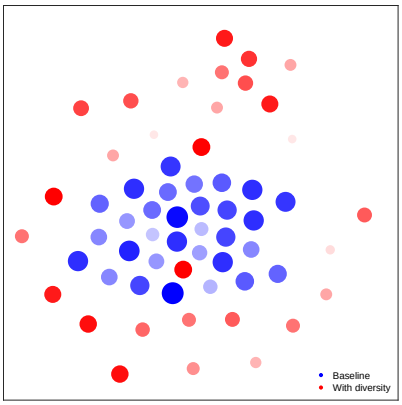

Figure 7. 2D Projection of the gradients $\{\nabla_{\boldsymbol{h}} g^{\star}_{\boldsymbol{\phi}_i}\}_{i=1}^{64}$ of a collection of models trained without and with our diversity regularizer (PACS dataset, "art" as test style). Each point represents one model. With the regularizer (in red), the gradients are clearly more spread out. The 2D projection is done with t-SNE using the inverse of the dot product as the distance function. The size and color saturation of each point are proportional to the accuracy of the corresponding model.

Figure 8. Examples from the biased activity recognition (BAR) dataset [48]. Each row shows a different class, and the upper/lower part of each row shows training/test images respectively (for example on the first row, rock climbing/ice climbing).

Collages dataset	Accuracy (%) on			
	MNIST	SVHN	Fashion-M.	CIFAR-10
Upper bounds: training data with all blocks but one randomized				
MNIST predictive only	**99.7** ±0.0	49.7 ±0.0	50.5 ±0.0	49.9 ±0.0
SVHN predictive only	50.2 ±0.2	**89.8** ±0.5	50.5 ±0.1	50.2 ±0.2
Fashion-M. predictive only	50.0 ±0.2	50.3 ±0.1	**77.3** ±0.4	49.0 ±0.4
CIFAR predictive only	49.9 ±0.2	50.1 ±0.3	50.2 ±0.4	**68.4** ±0.9
With proposed regularizer, 32 models				
Best model on MNIST	**95.4** ±0.4	49.6 ±0.1	50.6 ±0.3	49.8 ±0.1
Best model on SVHN	51.0 ±2.7	**79.3** ±3.1	52.2 ±1.2	51.4 ±1.1
Best model on Fashion-M.	50.6 ±1.6	50.4 ±0.3	**69.0** ±3.0	52.6 ±1.6
Best model on CIFAR	50.4 ±1.5	50.3 ±0.3	56.1 ±1.2	**59.6** ±0.5

Table 5. Detailed results on collages. We identify the best model on each test set as in Table 1. The difference is that we report the accuracy **on all four test sets** (in the main paper, these were summarized as a single row). Each model specializes and is good on a single test set at a time. This shows that the features used by each model do not overlap.