Learning useful representations for shifting tasks and distributions

Jianyu Zhang [1] **Léon Bottou** [2 1]

Abstract

Does the dominant approach to learn representations (as a side effect of optimizing an expected cost for a single training distribution) remain a good approach when we are dealing with multiple distributions? Our thesis is that *such scenarios are better served by representations that are richer than those obtained with a single optimization episode.* We support this thesis with simple theoretical arguments and with experiments utilizing an apparently naïve ensembling technique: concatenating the representations obtained from multiple training episodes using the same data, model, algorithm, and hyper-parameters, but different random seeds. These independently trained networks perform similarly. Yet, in a number of scenarios involving new distributions, the concatenated representation performs substantially better than an equivalently sized network trained with a single training run. This proves that the representations constructed by multiple training episodes are in fact different. Although their concatenation carries little additional information about the training task under the training distribution, it becomes substantially more informative when tasks or distributions change. Meanwhile, a single training episode is unlikely to yield such a redundant representation because the optimization process has no reason to accumulate features that do not incrementally improve the training performance.

1. Introduction

Although the importance of features in machine learning systems was already clear when the Perceptron was invented (Rosenblatt, 1957), learning features from examples was often considered a hopeless task (Minsky and Papert, 1969). Some researchers hoped that random features were good enough, as illustrated by the Perceptron. Other researchers preferred to manually design features using substantive knowledge of the problem (Simon, 1989). This changed when Rumelhart et al. (1986) showed the possibility of feature learning as a side effect of the risk optimization. Despite reasonable concerns about the optimization of nonconvex cost functions, feature discovery through optimization has driven the success of deep learning methods.

There are however many cues suggesting that learning features no longer can be solely understood through the optimization of the expected error for a single data distribution. First, adversarial examples (Szegedy et al., 2014) and shortcut learning (Geirhos et al., 2020) illustrate the need to make learning systems that are robust to certain changes of the underlying data distribution and therefore involve multiple expected errors. Second, the practice of transferring features across related tasks (Bottou, 2011; Collobert et al., 2011; Oquab et al., 2014) is now viewed as foundational (Bommasani et al., 2021) and intrinsically involves multiple data distributions and cost functions. It is therefore timely to question whether the optimization of a single cost function creates and accumulates features in ways that make the most sense in this broader context.

This contribution reports on experiments showing how the out-of-distribution performance of a deep learning model benefits from internal representations that are richer and more diverse than those computed with a single optimization episode. More precisely, *although optimization can produce diverse features, a single run is unable to collect them all into a rich representation that performs better when tasks or distributions change*. In a time where many organizations deploy considerable resources training huge foundational models, this observation should be sobering.

Organization of the manuscript Section 3 provides simple theoretical tools to discuss the value of features, discusses their consequences in-distribution and out-of-distribution, and approaches the notion of feature redundancy and feature richness. Sections 4, 5, 6, and 7, present experimental results pertaining respectively to supervised transfer learning, self-supervised transfer learning, meta-learning, and out-of-distribution learning.

[1]New York University, New York, NY, USA. [2]Facebook AI Research, New York, NY, USA.. Correspondence to: Jianyu Zhang <jianyu@nyu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Related work

Representations and optimization Papyan et al. (2020) shows how deep network representations collapse to a "simplex equiangular tight frame" when one trains for a very long time. Shwartz-Ziv and Tishby (2017) argue that the training process first develops representations in unsupervised mode, then prunes away the unnecessary features. Both papers associate this representation impoverishment with better generalization (in-distribution). We argue that it hurts performance when distributions change. Closer to our concerns, Pezeshki et al. (2021) describe the gradient starvation phenomenon which makes it hard to find the right features when a network already has spurious features. They do not however consider how to produce rich representations with multiple training episodes.

Ensembles (Dietterich, 2000) argues that model diversity is essential for the generalization performance of ensembles. Ganaie et al. (2021) reviews deep ensembles with the same conclusion. Our work focuses instead on scenarios involving multiple tasks or data distributions. We purposely refrain from engineering diversity, still observe substantial improvements, and draw conclusions about the undesirable properties of optimization.

Universal representations Several authors (Wang et al., 2022; Dvornik et al., 2020; Bilen and Vedaldi, 2017; Gontijo-Lopes et al., 2021; Li et al., 2021; 2022; Chowdhury et al., 2021) have recently proposed to collect features obtained with different tasks, datasets, network architectures, or hyper-parameters. The resulting so-"universal" representations can be helpful for a variety of tasks. This approach is certainly interesting for practical problems but would not have allowed us to draw our conclusions.

Model soups Another line of work uses weight averaging to combine the properties of diverse networks (Wortsman et al., 2022; Rame et al., 2022b), with an increasing focus on leveraging models trained on multiple tasks to achieve a high performance on a task of interest (Ilharco et al., 2022; Ramé et al., 2022). This engineered diversity provides for high performance, but does not allow the authors to draw conclusions about the optimization process itself.

Shortcut learning and mitigation Several authors (e.g. Huang et al., 2020; Teney et al., 2022) propose to work around the shortcut learning problem (Geirhos et al., 2020) by shaping the last-layer classifier or introducing penalty terms in a manner that favors richer representations. Zhang et al. (2022) argue that such additions make the optimization very challenging, but can be managed by initializing the networks with a rich representation constructed with an elaborate multi-step process. We show that rich representations can also be built by merely training the same network multiple times and combining their representations.

3. Features and representations

This section provides a conceptual framework for talking about richness and diversity of representations. Although it seems natural to compare representations using information theory concepts such as mutual information, this approach is fraught with problems. For instance, the simplest way to maximize the mutual information $M(\Phi(x), y)$ between the representation $\Phi(x)$ and the desired output y consists of making Φ equal to the identity. The information theoretic approach overlooks the main role of a feature extraction function, which is not filtering the information present in the inputs x, but formatting it in a manner exploitable by a simple learning system such as a linear classifier or a linear regression.[1] The following framework relies on the linear probing error instead.

Framework We call *feature* a function $x \mapsto \varphi(x) \in \mathbb{R}$, and we call *representation* a set Φ of features. We use the notation $\mathbf{w}^\top \Phi(x)$ to denote the dot product $\sum_{\varphi\in\Phi} w_\varphi\, \varphi(x)$ where the coefficients w_φ of vector $\mathbf{w}$ are indexed by the corresponding feature φ and are assumed zero if $\varphi \notin \Phi$.

We assume for simplicity that our representations are exploited with a linear classifier trained with a convex loss ℓ. The expected loss of classifier f is

$$C_P(f) = \mathbb{E}_{(x,y)\sim P}\left[\,\ell(f(x), y)\,\right]$$

and the optimal cost achievable with representation Φ

$$C^*_P(\Phi) = \min_{\mathbf{w}} \; C_P(f) \;\text{ with }\; f : x \mapsto \mathbf{w}^\top \Phi(x)\,. \qquad (1)$$

This construction ensures:

Proposition 1. $C^*_P(\Phi_1 \cup \Phi_2) \leq C^*_P(\Phi_2)$ *for all* Φ_1, Φ_2.

Intuitively, if the combined representation $\Phi_1 \cup \Phi_2$ performs better than Φ_2, then Φ_1 must contain something useful that Φ_2 does not. We formalize this using the word *information* to actually mean *linearly exploitable information about* y.

Definition 1. Φ_1 contains information not present in Φ_2 iff $C^*_P(\Phi_1 \cup \Phi_2) < C^*_P(\Phi_2)$.

Thanks to proposition 1, the opposite property becomes :

Definition 2. Φ_2 contains all the information present in Φ_1 iff $C^*_P(\Phi_1 \cup \Phi_2) = C^*_P(\Phi_2)$.

Finally we say that Φ_1 and Φ_2 carry equivalent information when Φ_2 contains all the information present in Φ_1, and Φ_1 contains all the information present in Φ_2 :

Definition 3. Φ_1 and Φ_2 carry equivalent information iff $C^*_P(\Phi_1) = C^*_P(\Phi_1 \cup \Phi_2) = C^*_P(\Phi_2)$.

[1] We choose linear classifiers as the "simple learning system" in our framework for the ease of theoretical analysis. This does not imply non-linear classifiers would behave differently. In fact, we empirically investigate another simple learning system, a cosine classifier, in the appendix Table 11.

This definition is stronger[2] than merely requiring equality $C^*_P(\Phi_1) = C^*_P(\Phi_2)$. In particular, we cannot improve the expected cost by constructing an ensemble :

Theorem 2. *Let representations Φ_1 and Φ_2 carry equivalent information. Let $f_i(x){=}\mathbf{w}_i^{*\top}\Phi_i(x)$, for $i{\in}\{1,2\}$, be corresponding optimal classifiers. Then, for all $0{\le}\lambda{\le}1$,*

$$C^*_P(\lambda f_1 + (1-\lambda) f_2) = C^*_P(f_1) = C^*_P(f_2).$$

Proof. Let $\Phi = \Phi_1 \cup \Phi_2$. Because the loss ℓ is assumed convex, the solutions of optimization problem (1) form a convex set S. Since $C^*_P(\Phi_1){=}C^*_P(\Phi_1 \cup \Phi_2){=}C^*_P(\Phi_2)$, set S contains w_1^* and w_2^*, as well as any mixture thereof. □

We now turn our attention to representations constructed by optimizing both the representation Φ and the weights $\mathbf{w}$:

$$\min_{\Phi} C^*_P(\Phi) = \min_{\Phi} \min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim P}[\ell(\mathbf{w}^\top \Phi(x), y)] \,. \quad (2)$$

This idealized formulation optimizes the expected error without constraints on the nature and number of features. All its solutions problem carry equivalent information :

Theorem 3. *Let Φ_1 and Φ_2 be two solutions of problem* (2). *Then Φ_1 and Φ_2 carry equivalent information.*

Proof. Proposition 1 implies $C^*_P(\Phi_1 \cup \Phi_2) \le C^*_P(\Phi_1)$. Since Φ_1 and Φ_2 are both solutions of problem 2, $C^*_P(\Phi_1) = C^*_P(\Phi_2) \le C^*_P(\Phi_1 \cup \Phi_2) \le C^*_P(\Phi_1)$. □

In-distribution viewpoint Consider a deep network that is sufficiently overparameterized to accommodate any useful representation in its penultimate layer. Assume that we are able to optimize its expected cost on the training distribution, that is, optimize its in-distribution generalization error. Although repeated optimization episodes need not return exactly the same representations, Theorem 3 tells us that these representations *carry equivalent information*; Definition 3 tells us that we cannot either improve the in-distribution test error by linear probing, that is, by training a linear layer on top of the concatenated representations; and Theorem 2 tells us that we cannot improve the test error with an ensemble of such networks. The performance of ensembles depends on the diversity of their components (Dietterich, 2000; Ganaie et al., 2021), and nothing has been done here to obtain diverse networks.

In practice, we cannot truly optimize the expected error of an overparameterized network. The representations obtained with separate training episodes tend to carry equivalent information but will not do so exactly.[3] Although an ensemble of such identically trained networks can still improve both the training and testing errors, using such similarly trained networks remains a poor way to construct ensembles when one can instead vary the training data, the hyper-parameters, or vary the model structure (Ganaie et al., 2021). Engineering diversity escapes the setup of Theorem 3 because each component of the ensemble then solves a different problem. This is obviously better than relying on how the real world deviates from the asymptotic setup.

Out-of-distribution viewpoint Assume now that we train our network on a first data distribution $P(x, y)$, but plan to use these networks, or their representations, or their inner layers, with data that follow a different distribution $Q(x, y)$. Doing so also escapes the assumptions of our framework because the definition of representation carrying similar information (Definition 3) critically depends on the data distribution. Representations that carry equivalent information for the training distribution P need not carry equivalent information for a new distribution Q at all.[4]

Consider again representations obtained by performing multiple training episodes of the same network that only differ by their random seed.[5] These representations roughly carry equivalent information with respect to the training distribution, but, at the same time, may be very far from carrying equivalent information with respect to a new distribution.

If this is indeed the case, *constructing an ensemble of such similarly trained networks can have a far greater effect on out-of-distribution data than in-distribution.* Experimental results reported in the following sections will demonstrate this effect. In fact, since we cannot know which of these representations or features might prove more informative on the new distribution, it seems wise to keep them all. *Premature feature selection is not a smart way to prepare for distribution changes.*

Optimization dynamics There is growing evidence that implicit regularization in deep learning networks is related to various flavors of sparsity (e.g. Andriushchenko et al., 2022; Blanc et al., 2020). In an oversimplified account of this complex literature, the learning process explores the feature space more or less randomly; features that carry incrementally useful information stick more than those who do

[2]This is also weaker than using the quantity of information H : writing $H(\Phi_1){=}H(\Phi_1{\cup}\Phi_2){=}H(\Phi_2)$ would imply that Φ_1 and Φ_2 are equal up to a bijection. Theorems 2 and 3 are important because this is not the case here.

[3]Experience shows however that repeated trainings on large tasks, such as IMAGENET, yields networks with remarkably consistent training and testing performances.

[4]Information theoretical concepts are also tied to the assumed data distribution. For instance, whether two features have mutual information critically depends on the assumed data distribution.

[5]The random seed here may determine the initial weights, the composition of the mini-batches, or the data augmentations. It does not affect the data distribution, the model structure, or even the training algorithm hyper-parameters.

Table 1. Impact of L2 weight decay on supervised transfer learning between CIFAR10 and CIFAR100.

L2 weight decay	0	$5e-4$
CIFAR10	91.41±0.81	**94.89±0.23**
CIFAR10→CIFAR100	**49.68±0.72**	29.17±0.50
CIFAR100	70.37±1.49	**76.78±0.36**
CIFAR100→CIFAR10	**78.87±0.98**	75.92±0.54

not. Consider for instance a network with representation Φ_t at iteration t and a feature $\varphi \in \Phi_t$ whose information is already present in $\Phi_t \setminus \{\varphi\}$ in the sense of Definition 2. This feature does not incrementally improve the training distribution performance and therefore may not stick. Yet this feature might contain useful information when compared to a different representation, or when compared to $\Phi_t \setminus \{\varphi\}$ under a different distribution.

Explicit regularization in deep networks, such as the ubiquitous slight weight decay, also tends to destroy features that appear redundant. Papyan et al. (2020) describes how representations collapse when one trains a network for a very long time. Shwartz-Ziv and Tishby (2017) describe competing processes that create representations and prune representations in all layers at once.

Table 1 reports on a simple experiment to illustrate how capacity control with regularization can help in-distribution performance but hurt when the distribution changes. We pre-train a RESNET18 on the CIFAR10 task and transfer its learned representation to a CIFAR100 task by linear probing (see setups in appendix A). Although the best in-distribution performance, 94.9%, is achieved using a slight weight decay, the representation learned *without weight decay* transfers far better (49.7% versus 29.2%). The same observation holds when one reverses the role of the CIFAR10 and CIFAR100 datasets.

Next steps The remaining sections of this paper describe experiments that investigate the effect of concatenating representations obtained by multiple training episodes that only differ by their random seed.

Despite the *intentional lack of diversity* of these ensembles, the performance improvements observed on tasks involving distribution changes are far greater than the in-distribution performance improvements. This proves that representations constructed by multiple training episodes are indeed different. Even though their concatenation carries little additional information for in-distribution, these experiments show how they become substantially more informative when tasks or distributions change.

Meanwhile, we obtain worse performance (a) when we train a network whose size matches that of the ensemble from scratch, or (b) when we fine-tune the concatenated representations in a single additional run. We contend that this happens because optimization inherently impoverishes the representations in a manner that makes sense in-distribution but hurts out-of-distribution, and we propose *two-stage fine-tuning* (Figure 2) to correct this behavior.

Table 2. Supervised transfer learning from IMAGENET to INAT18, CIFAR100, and CIFAR10 using linear probing. The ERM (empirical risk minimization) rows provide baseline results. The CATn rows use the concatenated representations of n separately trained networks.

			ID	Linear Probing (OOD)		
method	architecture	params	IMAGENET	INAT18	CIFAR100	CIFAR10
ERM	RESNET50	23.5M	75.58	37.91	73.23	90.57
ERM	RESNET50W2	93.9M	77.58	37.34	72.65	90.86
ERM	RESNET50W4	375M	78.46	38.71	74.81	92.13
ERM	2×RESNET50	47M	75.03	39.34	74.36	90.94
ERM	4×RESNET50	94M	75.62	41.89	74.06	90.61
CAT2	2×RESNET50	47M	77.57	43.26	76.10	91.86
CAT4	4×RESNET50	94M	78.15	46.55	78.19	93.09
CAT10	10×RESNET50	235M	78.36	49.65	79.61	93.75

4. Supervised transfer learning

This section focuses on supervised transfer learning scenarios in which the representation learned using an auxiliary supervised task, such as the IMAGENET object recognition task (Deng et al., 2009), is then used for the target tasks, such as, for our purposes, the CIFAR10, CIFAR100, and INATURALIST18 (INAT18) object recognition tasks (Krizhevsky, 2009; Van Horn et al., 2018). We distinguish the *linear probing* scenario where the penultimate layer features of the pre-trained network are used as inputs for linear classifiers trained on the target tasks, and the *fine tuning* scenario which uses back-propagation to further update the transferred features using the target task training data.[6]

Linear probing The first three rows of Table 2, labeled ERM, provide baselines for the linear probing scenario, using respectively a RESNET50 network (He et al., 2016a), as well as larger variants RESNET50Wn with n times wider internal representations and roughly n^2 times more parameters. The following two rows provide additional baseline results using networks $n\times$RESNET50 composed of respectively n separate RESNET50 networks joined by concatenating their penultimate layers. Although these networks perform relatively poorly on the pre-training task IMAGENET, their linear probing performance is substantially better than that of the ordinary RESNETs.

The final three rows of Table 2, labeled CATn, are obtained by training n separate RESNET50 networks on IMAGENET with different random seeds, and using their concatenated

[6]Code is available at `https://github.com/TjuJianyu/RRL`

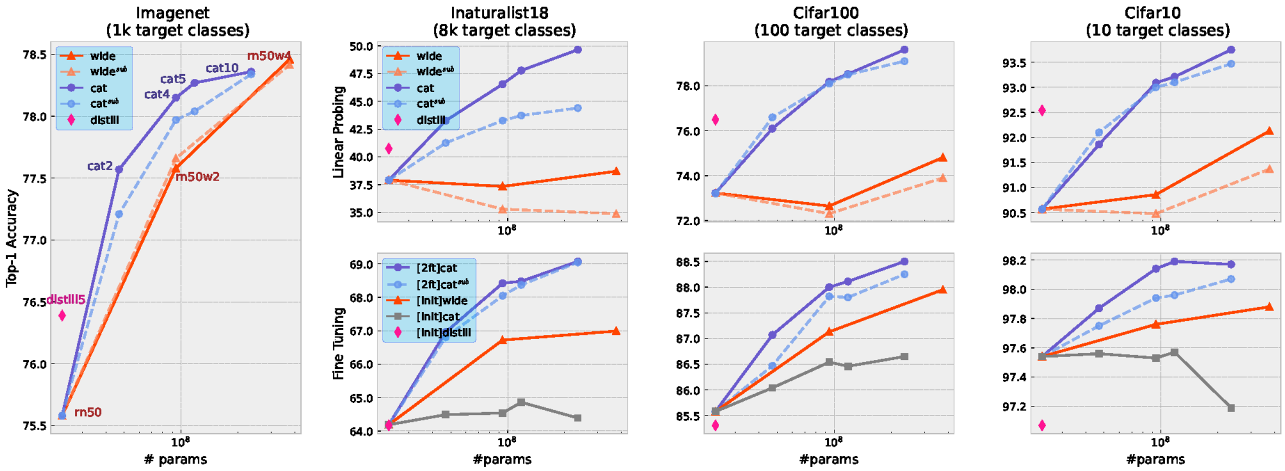

Figure 1. Supervised transfer learning from IMAGENET to INAT18, CIFAR100, and CIFAR10. The top row shows the superior linear probing performance of the CATn networks (blue, “cat”). The bottom row shows the performance of fine-tuned CATn, which is poor with normal fine-tuning (gray, “[init]cat”) and excellent for two-stage fine tuning (blue, “[2ft]cat”). DISTILLn (pink, “distill”) representation is obtained by distilling CATn into one RESNET50 (we omit DISTILL in this section due to the space limit. see details in the appendix B).

representations as inputs for a linear classifier trained on the target tasks. This approach yields linear probing performances that substantially exceed that of comparably sized baseline networks. Remarkably, CATn, with separately trained components, outperforms the architecturally similar $n\times$RESNET50 trained as a single network. See appendix B for experimental details.

These results are succinctly[7] represented in the top row of Figure 1. For each target task INAT18, CIFAR100, and CIFAR10, the solid curves show the linear probing performance of the baseline RESNET50Wn (red, labeled “wide”) and of the CATn networks (blue, “cat”) as a function of the number of parameters of their inference architecture.

The left plot (double height) of Figure 1 provides the same information in-distribution, that is, using the pre-training task as target task. In-distribution, the advantage of CATn vanishes when the networks become larger, possibly large enough to approach the conditions of Theorem 3. The out-of-distribution curves (top row) are qualitatively different because they show improved performance all along.

An ensemble of n RESNET50 networks is architecturally similar to the CATn models. Instead of training a linear classifier on the concatenated features, the ensemble averages n classifiers independently trained on top of each network. Whether this is beneficial depends on the nature of the target task and its training data (dashed blue, labeled “cat$^{\text{sub}}$”). For completeness, we also present an ensemble baseline (dashed red plot, labeled “wide$^{\text{sub}}$”) averaging n

[7]In order to save space, all further results in the main text of this contribution are presented with such plots, with result tables provided in the appendix.

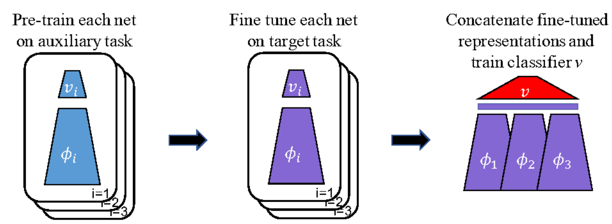

Figure 2. Two-stage fine-tuning consists of fine-tuning each network separately, then concatenating their feature extractors, now frozen, and training a final classifier.

linear classifiers trained on top of a random partition of the corresponding wide network features.

Fine-tuning Having established, in the linear probing case, that transferring concatenated representations CATn outperforms transferring the representation of an equivalently sized network, we turn our attention to fine-tuning.

Fine-tuning is usually achieved by setting up a linear classifier on top of the transferred feature and training it on the target task data while allowing back-propagation to update the transferred features as well. The bottom row of Figure 1 shows the performance of this approach using the baseline network representations (red curve, labeled “[init]wide”) and the concatenated representations (gray curve, labeled “[init]cat”), The latter perform very poorly.[8]

We posit that fine-tuning with a single training episode impoverishes the initially rich representation. Instead, we

[8]The poor performance of plain fine-tuning had already been pointed out by Kumar et al. (2022) and Kirichenko et al. (2022).

propose *two-stage fine tuning* which consists of separately training n networks on the pre-training task, separately fine-tuning them on the target task, and finally training a linear classifier on top of the concatenation of the n separately fine-tuned representations (Figure 2). The superior performance of two-stage fine-tuning is clear in the bottom row of Figure 1 (blue solid curve, labeled "[2ft]cat"). Ensembles of fine-tuned networks perform almost as well (blue dashed curve, labeled "[2ft]cat$^{\text{sub}}$").

The superior *two-stage fune-tuning* performance, compared with the *normal fine-tuning* (gray curve), may look counter-intuitive, since separately fine-tuning n sub-networks is also likely to reduce the richness of the representation due to the in-distribution equivalence of information (Theorem 2). A similar phenomenon also exists in IMAGENET pre-training in Table 2, where the ID (in-distribution) performance of CATn is substantially better than ERM on the same $n\times$RESNET50 architectures. We believe that the difference is with the dynamics of the optimization process. In appendix B.2, we show the accuracy of each leg of ERM pre-trained $n\times$RESNET50 are very disparate: one leg is doing all the work (The ID IMAGENET top-1 accuracy difference between legs is as large as 73%). This is not the case in CATn pretraining.

Vision transformers Figure 3 shows that transformer networks behave similarly. We carried out supervised transfer experiments using the original vision transformer, VIT, (Dosovitskiy et al., 2020), and using a more advanced version using carefully crafted data augmentations and regularization, VIT(AUGREG), (Steiner et al., 2021). We use two transformers of two different sizes, ViT-B/16 and ViT-L/16, pre-trained on IMAGENET21K.[9] Supervised transfer baselines (red, "wide&deep" or "[init]wide&deep") are obtained by linear-probing and by fine-tuning on IMAGENET(1K). These baselines are outperformed by respectively linear-probing and *two-stage fine tuning* on top of the concatenation of their final representations (CAT2).

An even larger transformer architecture, ViT-H/14, yields about the same IMAGENET1K fine-tuning performance as ViT-L/16, but lags 1% behind CAT2, despite having twice as many parameters (Dosovitskiy et al., 2020). Experiments with two-stage fine-tuned CAT2 in VIT(AUGREG) show even better results, possibly because changing the random seed does not just changes the initial weights and the mini-batch composition, but also affects the data augmentations of the VIT(AUGREG) networks.

5. Self-supervised transfer learning

In self-supervised transfer learning (SSL), transferable representations are no longer constructed using a supervised

[9]Checkpoints provided at `https://github.com/google-research/vision_transformer`.

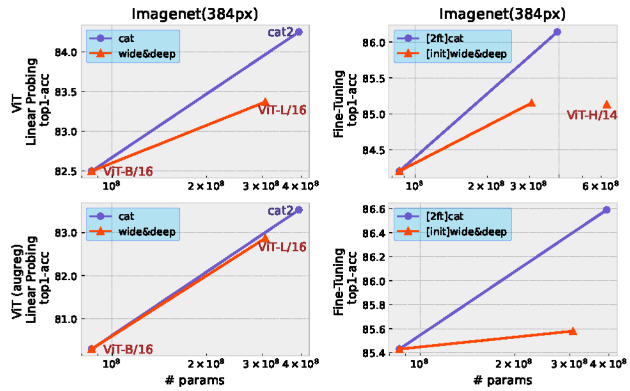

Figure 3. Supervised transfer learning from IMAGENET21K to IMAGENET on vision transformers.

auxiliary task, but using a training criterion that does not involve tedious manual labeling. We focus on schemes that rely on the knowledge of a set of acceptable pattern transformations. The training architecture then resembles a siamese network whose branches process different transformations of the same pattern. The SSL training objective must then balance two terms: on the one hand, the representations computed by each branch must be close or, at least, related; on the other hand, they should be prevented from collapsing partially (Jing et al., 2021) or catastrophically (Chen and He, 2020). Although this second term tends to fill the representation with useful features, what is necessary to balance the SSL training objective might still exclude potentially useful features for the target tasks.

This section presents results obtained using SWAV pre-training using 1.2 million IMAGENET images (Caron et al., 2020) and using SEER pre-training using 1 billion INSTAGRAM1B images (Goyal et al., 2022). These experiments leverage the pre-trained models made available by the authors: five RESNET50 (four from our reproduction), one RESNET50W2, one RESNET50W4 and one RESNET50W5 for the SWAV experiments;[10] one REGNET32GF, one REGNET64GF, one REGNET128GF, and one REGNET256GF (1.3B parameters) for the SEER experiments.[11]

The first four columns of Figure 4 present linear probing results for four target object recognition tasks: supervised IMAGENET, INATURALIST18, CIFAR100, and CIFAR10. The baseline curves (red, labeled "wide" or "wide&deep") plot the performance of linear classifiers trained on top of the pre-trained SSL representations. The solid CATn curves were obtained by training a linear classifier on top of the concatenated representations of the n smallest SSL pre-trained representations (solid blue, "cat"). The dash CATn

[10]`https://github.com/facebookresearch/swav`

[11]`https://github.com/facebookresearch/vissl/tree/main/projects/SEER`

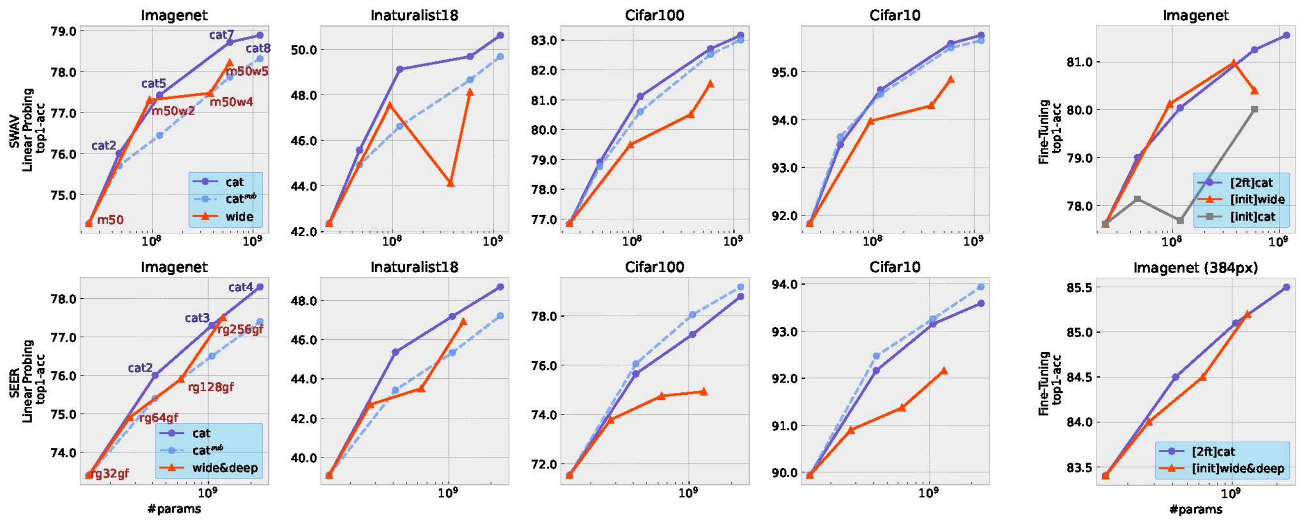

Figure 4. Self-supervised transfer learning with SWAV trained on unlabeled IMAGENET(1K) (*top row*) and with SEER on INSTAGRAM1B (*bottom row*). The constructed rich representation, CATn, yields the best linear probing performance ("cat" and "cat$^{\text{sub}}$") for supervised IMAGENET, INAT18, CIFAR100, and CIFAR10 target tasks. The two-stage fine-tuning ("[2ft]cat") matches equivalently sized baseline models ("[init]wide" and "[init]wide&deep"), but with much easier training. The sub-networks of CAT5 (and CAT2) in SWAV hold the same architecture. Due to the space limitation, we put other fine-tuning curves in appendix C.1.1.

curves train an ensemble of n small classifiers on subsets of the concatenated representation (dash blue, "cat$^{\text{sub}}$").[12] Overall, the CATn approach offers the best performance.

The last column of Figure 4 presents results with fine-tuning for the supervised IMAGENET task. Our *two-stage fine-tuning* approach (as Figure 2) matches the performance of equivalently sized baseline networks. In particular, the largest CAT4 model using SEER pre-training, with 2.3B parameters, achieves 85.5% correct classification rate, approaching the 85.8% rate of the largest baseline network in SEER (Goyal et al., 2022), REGNET10B with 10B parameters. Of course, separately training and fine-tuning the components of the CAT4 network is far easier than training a single REGNET10B network.

Additional results using SIMSIAM (Chen et al., 2020) and with distillation are provided in appendix C.3. Other experiment details are provided in appendix C.

6. Meta-learning & few-shots learning

Each target task in the few-shots learning scenario comes with only a few training examples. One must then consider a large collection of target tasks to obtain statistically meaningful results.

[12]Likewise the supervised transfer learning experiments, each small classifier learns on the representation of a sub-network (e.g. REGNET32GF, REGNET64GF). Now the representation subset cannot be treated as random subsets of the concatenated representation anymore, because the model architectures are not always the same. So we omit the ensemble classifiers for red curves.

We follow the setup of Chen et al. (2019)[13] in which the base task is an image classification task with a substantial number of classes and examples per class, and the target tasks are five-way classification problems involving novel classes that are distinct from the base classes and come with only a few examples. Such a problem is often cast as a *meta learning* problem in which the base data is used to learn how to solve a classification problem with only a few examples. Chen et al. (2019) find that excellent performance can be achieved using simple baseline algorithms such as supervised transfer learning with linear probing (BASELINE) or with a cosine-based final classifier (BASELINE++). These baselines match and sometimes exceed the performance of common few-shots algorithms such as MAML (Finn et al., 2017), RELATIONNET (Sung et al., 2018), MATCHINGNET (Vinyals et al., 2016), and PROTONET (Snell et al., 2017).

Figure 6 reports results obtained with a RESNET18 architecture on both the MINIIMAGENET (Vinyals et al., 2016) and CUB (Wah et al., 2011) five ways classification tasks with either one or five examples per class as set up by Chen et al. (2019). The MAML, RELATIONNET, MATCHINGNET, and PROTONET results (red bars) are copied verbatim from (Chen et al., 2019, table A5). The BASELINE and BASELINE++ results were further improved by a systematic L2 weight decay search procedure (see appendix D.2). All these results show substantial variations across runs, about

[13]We are aware of various existing few-shot benchmarks, such as MetaDataset (Triantafillou et al., 2019), that contain more datasets than Chen et al. (2020). We choose Chen et al. (2020), because it is enough to validate our ideas in section 3.

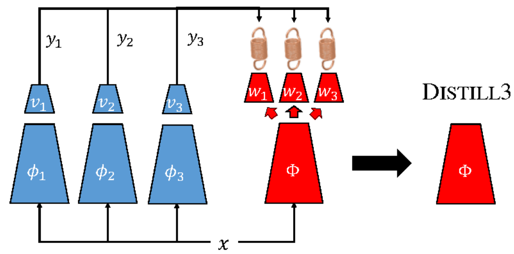

Figure 5. (DISTILLn) A multiple head network (red) trained to predict the outputs of the pre-trained networks $\Phi_1, \Phi_2, \cdots$ (blue) must develop a representation Φ that subsumes those of all the blue networks. The same distillation process is used by the BONSAI algorithm (Zhang et al., 2022) but after training the networks with adversarially re-weighted data.

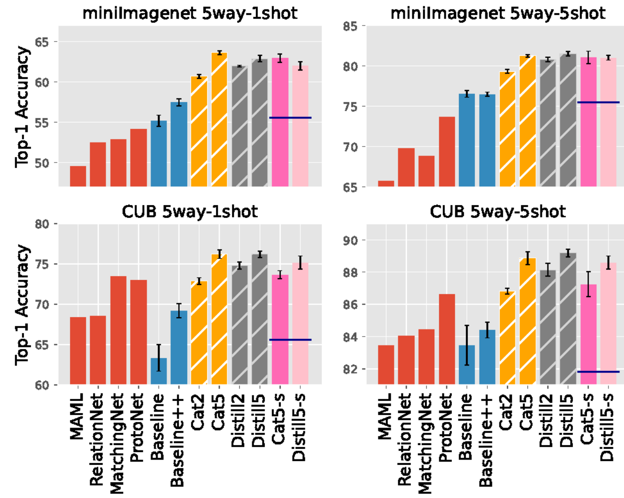

Figure 6. Few-shot learning performance on MINIIMAGENET and CUB. Four common few-shot learning algorithms are shown in red (results from Chen et al. (2019)). Two supervised transfer methods, with either a linear classifier (BASELINE) or cosine-based classifier (BASELINE++) are shown in blue. The DISTILL and CAT results, with a cosine-base classifier, are respectively shown in orange and gray. The CAT5-S and DISTILL5-S results were obtained using five snapshots taken during a single training episode with a relatively high step size. The dark blue line shows the best individual snapshot. Standard deviations over five repeats are reported.

4% for CUB and 2% for MINIIMAGENET.

The CATn and DISTILLn results were then obtained by first training n RESNET18 on the base data with different seeds, constructing a combined (rich) representation by either concatenation or distillation (as Figure 5), then, for each task, training a cosine distance classifier using the representation as input. Despite the high replication variance of the competing results, both DISTILL and CAT show very strong performance. Note that naively increasing model architecture, e.g. from RESNET18 to RESNET34, can only gain limited improvements ($\leq 1\%$, Chen et al. (2020), table A5) and is still lagging behind CAT and DISTILL.

The pink bars (CAT5-S and DISTILL5-S) in Figure 6, concatenate or distill five snapshots taken at regular intervals during a single training episode with a relatively high step size (0.8), achieve a similar few-shots learning performance as CAT5 and DISTILL5, perform substantially better than the best individual snapshot (dark blue line). *It implies that diverse features are discovered and then abandoned but not accumulated during the optimization process.* More results and details, as well as a comparison with conditional meta-learning algorithms (Wang et al., 2020; Denevi et al., 2022; Rusu et al., 2018), are shown in appendix D.

7. Out-of-distribution generalization

In the out-of-distribution generalization scenario, we seek a model that performs well on a family of data distributions, also called environments, on the basis of a finite number of training sets distributed according to some of these distributions. Arjovsky et al. (2020) propose an invariance principle to solve such problems and propose the IRMV1 algorithm which searches for a good predictor whose final linear layer is simultaneously optimal for all training distributions. Since then, a number of algorithms exploiting similar ideas have been proposed, such as VREX (Krueger et al., 2020), FISHR (Rame et al., 2022a), or CLOVE (Wald et al., 2021). Theoretical connections have been made with multi-calibration (Hebert-Johnson et al., 2018; Wah et al., 2011). Alas, the performance of these algorithms remains wanting (Gulrajani and Lopez-Paz, 2021). Zhang et al. (2022) attribute this poor performance to the numerical difficulty of optimizing the complicated objective associated with these algorithms. They propose to work around these optimization problems by providing initial weights that already extract a rich palette of potentially interesting features constructed using the BONSAI (Zhang et al., 2022) algorithm.

Following Zhang et al. (2022), we use the CAMELYON17 tumor classification dataset (Bandi et al., 2018) which contains medical images collected from five hospitals with potentially different devices and procedures. As suggested in Koh et al. (2021), we use the first three hospitals as training environments and the fifth hospital for testing. OOD-tuned results are obtained by using the fourth hospital to tune the various hyper-parameters. IID-tuned results only use the training distributions (see details in appendix E). The purpose of our experiments is to investigate whether initializing with the DISTILL or CAT algorithm provides a computationally attractive alternative to BONSAI.

Table 3 compares the test performance achieved by two

[14]We apply BONSAI algorithm with 2 discovery episodes. Check Zhang et al. (2022) for more details.

Table 3. Test accuracy on the CAMELYON17 dataset with DENSENET121. We compare various initialization (ERM, CATn, DISTILLn, and BONSAI) for two algorithms vREX and ERM using either the IID or OOD hyperparameter tuning method. The standard deviations over 5 runs are reported.

	IID-Tune		OOD-Tune	
	vREX	ERM	vREX	ERM
ERM	69.6±10.5	66.6±9.8	70.6±10.0	70.2±8.7
CAT2	74.3±8.0	74.3±8.0	73.7±8.1	74.2±8.1
CAT5	75.2±2.9	75.0±2.7	74.9±3.3	75.1±2.8
CAT20	76.4±0.5	76.5±0.5	76.8±0.9	76.4±0.9
DISTILL2	67.1±4.7	66.9±4.8	67.4±4.3	66.7±4.2
DISTILL5	69.9±7.4	69.9±6.9	71.8±5.0	69.9±6.3
DISTILL20	73.3±2.5	73.2±2.3	74.8±3.2	73.1±2.7
BONSAI2[14]	77.9±2.7	78.2±2.6	79.5±2.7	78.6±2.6

algorithms, vREX and ERM, after initializing with ERM, CATn, DISTILLn, and BONSAI, in both the IID-tune and OOD-tune scenarios. The CAT and DISTILL initialization perform better than ERM but not as well as BONSAI. *This result clearly shows the need to research better ways to train networks in a manner that yields diverse representations.* Although this contribution shows that simply changing the seed (as in CAT and DISTILL) can achieve good results, the experience of deep ensembles (Gontijo-Lopes et al., 2022) suggests that more refined diversification methods might yield substantially better representations.

8. Conclusion

Using a simple theoretical framework and a broad range of experiments, we show that deep learning scenarios that involve changing tasks or distributions are *better served by representations that are richer than those obtained with a single optimization episode.* In a time where many organizations deploy considerable resources training huge foundational models, this conclusion should be sobering.

The simple multiple-training-episode approach CAT constructs such richer representation with excellent performances in various scenarios. The *two-stage fine tuning* method works around the poor performance of normal fine-tuning in various transfer scenarios.

More importantly, this work provides a lot of room for new representation learning algorithms that move away from relying solely on a single optimization episode.

Acknowlegments

The authors acknowledge stimulating discussions with Alexandre Ramé, Diane Bouchacourt and David Lopez-Paz. The authors also acknowledge support from the National Science Foundation (NSF Award 1922658) and from the Canadian Institute for Advanced Research (CIFAR).

References

Maksym Andriushchenko, Aditya Varre, Loucas Pillaud-Vivien, and Nicolas Flammarion. Sgd with large step sizes learns sparse features. *arXiv preprint arXiv:2210.05337*, 2022.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv*, 2020.

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.

Hakan Bilen and Andrea Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, al. Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL `https://arxiv.org/abs/2108.07258`.

Léon Bottou. From machine learning to machine reasoning. Technical report, arXiv:1102.1808, February 2011.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments.

Advances in Neural Information Processing Systems, 33: 9912–9924, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Arkabandhu Chowdhury, Mingchao Jiang, Swarat Chaudhuri, and Chris Jermaine. Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9445–9454, 2021.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, Aug 2011.

Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. Conditional meta-learning of linear representations. *Advances in Neural Information Processing Systems*, 35: 253–266, 2022.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

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. *arXiv preprint arXiv:2010.11929*, 2020.

Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *European Conference on Computer Vision*, pages 769–786. Springer, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Mudasir A Ganaie, Minghui Hu, et al. Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395*, 2021.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods. *arXiv preprint arXiv:2110.12899*, 2021.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin Dogus Cubuk. No one representation to rule them all: Overlapping features of training methods. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=BK-4qbGgIE3`.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Mannat Singh, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. Vision models are more robust and fair when pretrained on uncurated images without supervision. *arXiv preprint arXiv:2202.08360*, 2022.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=lQdXeXDoWtI`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016a.

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*, pages 770–778, 2016b.

Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (Computationally-identifiable) masses. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1939–1948. PMLR, 10–15 Jul 2018.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*, 2022. URL `https://openreview.net/forum?id=THOOBy1uWVH`.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). *arXiv*, 2020.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=UYneFzXSJWh`.

Wei-Hong Li, Xialei Liu, and Hakan Bilen. Universal representation learning from multiple domains for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9526–9535, 2021.

Wei-Hong Li, Xialei Liu, and Hakan Bilen. Universal representations: A unified look at multiple task and domain learning. *arXiv preprint arXiv:2204.02744*, 2022.

M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020. doi: 10.1073/pnas.2015509117. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2015509117`.

Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Recycling diverse models for out-of-distribution generalization. *arXiv preprint arXiv:2212.10445*, 2022.

Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 18347–18377. PMLR, 2022a.

Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *arXiv preprint arXiv:2205.09739*, 2022b.

F. Rosenblatt. The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Project PARA, Cornell Aeronautical Lab, 1957.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, volume I, pages 318–362. Bradford Books, Cambridge, MA, 1986.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

J.C. Simon. *From Pixels to Features*. North Holland, August 1989.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16761–16772, 2022.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In D Lee, M Sugiyama, U Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Yoav Wald, Amir Feder, Daniel Greenfeld, and Uri Shalit. On calibration and out-of-domain generalization. *arXiv preprint arXiv:2102.10395*, 2021.

Hongyu Wang, Eibe Frank, Bernhard Pfahringer, Michael Mayo, and Geoffrey Holmes. Cross-domain few-shot meta-learning using stacking. *arXiv preprint arXiv:2205.05831*, 2022.

Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. Structured prediction for conditional meta-learning. *Advances in Neural Information Processing Systems*, 33:2587–2598, 2020.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

Jianyu Zhang, David Lopez-Paz, and Leon Bottou. Rich feature construction for the optimization-generalization dilemma. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26397–26411. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhang22u.html.

Supplementary Material

A. CIFAR supervised transfer learning

CIFAR10 supervised transfer learning experiments train a RESNET18 network on the CIFAR10 dataset with/without L2 weight decay (4e-5) for 200 epochs. During training, we use a SGD optimizer (Bottou et al., 2018) with momentum=0.9, initial learning rate=0.1, cosine learning rate decay, and batch size=128. As to data augmentation, we use RANDOMRESIZEDCROP (crop scale in $[0.8, 1.0]$), aspect ratio in $[3/4, 4/3]$) and RANDOMHORIZONTALFLIP. During testing, the input images are resized to 36×36 by bicubic interpolation and CENTERCROPED to 32×32. All input images are normalized by $mean = (0.4914, 0.4822, 0.4465), std = (0.2023, 0.1994, 0.2010)$ at the end.

Then transfer the learned representation to CIFAR100 dataset by training a last-layer linear classifier (linear probing). The linear layer weights are initialized by Gaussian distribution $\mathcal{N}(0, 0.01)$. The linear probing process shares the same training hyper-parameters as the supervised training part except for a zero L2 weight decay in all cases.

The CIFAR100 supervised transfer learning experiments swap the order of CIFAR100 and CIFAR10.

B. IMAGENET supervised transfer learning

B.1. Experiment settings

Image Preprocessing: Following He et al. (2016b), we use RANDOMHORIZONTALFLIP and RANDOMRESIZEDCROP augmentations for all training tasks. For IMAGENET and INAT18, the input images are normalized by $mean = (0.485, 0.456, 0.406), std = (0.229, 0.224, 0.225)$. For CIFAR, we use the same setting as Appendix A.

IMAGENET Pretraining: The RESNETs are pre-trained on IMAGENET with the popular protocol of Goyal et al. (2017): a SGD optimizer with momentum=0.9, initial learning rate=0.1, batch size=256, L2 weight decay=1e-4, and 90 training epochs. The learning rate is multiplied by 0.1 every 30 epochs. By default, the optimizer in all experiments is SGD with momentum=0.9.

DISTILL: To distill the CATn representations $[\phi_1, \ldots \phi_n]$ ($n\times$RESNET50) into a smaller representation Φ (RESNET50), we use the multi-head architecture as Figure 5. Inspired by Hinton et al. (2015), we use the Kullback–Leibler divergence loss to learn Φ as:

$$\min_{\Phi, w_0, \ldots, w_n} \sum_{i=0}^{n} \sum_{x} \left[\tau^2 \mathcal{L}_{kl}\Big(s_\tau\big(v_i \circ \phi_i(x)\big) \;||\; w_i \circ \Phi(x) \Big) \right], \tag{3}$$

where $s_\tau(v)_i = \frac{e^{v_i/\tau}}{\sum_k e^{v_k/\tau}}$ is a softmax function with temperature τ, v_i is the learned last-layer classifier of i^{th} sub-network of CATn.

In the DISTILL experiments, we distill five separately trained RESNET50 into one RESNET50 according to Eq 3 with $\tau = 10$. We use a SGD optimizer with momentum=0.9, batch size=2048, and weight decay=0. The initial learning rate is 0.1 and warms up to 0.8 within the first 5 epochs. Then learning rate decays to 0.16 and 0.032 at 210^{th} and 240^{th} epochs, respectively. The total training epochs is 270.

Linear probing:

- **IMAGENET:** The IMAGENET linear probing experiments train a linear classifier with the same hyper-parameters as IMAGENET pretraining. By default, the last linear classifier in all linear probing experiments is initialized by $\mathcal{N}(0, 0.01)$.
- **INAT18, CIFAR100, CIFAR10:** Following the settings of Goyal et al. (2022), the linear probing experiments (on INAT18, CIFAR100, CIFAR10) adds a BATCHNORM layer before the linear classifier to reduce the hyper-parameter tuning difficulty. The learning rate is initialized to 0.01 and multiplied by 0.1 every 8 epochs. Then train these linear probing tasks for 28 epochs by SGD Nesterov optimizer with momentum=0.9, batch size 256. Note that BATCHNORM + a linear classifier is still a linear classifier during inference. We tune L2 weight decay from {1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2} for CIFAR100 and CIFAR10, {1e-6, 1e-5, 1e-4} for INAT18.

Fine-tuning: As to the fine-tuning experiments (on CIFAR100, CIFAR10, and INAT18), we tune the initial learning rate from {0.005, 0.01, 0.05}, training epochs from {50, 100}. We further tune L2 weight decay from {0, 1e-5, 1e-4, 5e-4} for CIFAR100 and CIFAR10, {1e-6, 1e-5, 1e-4} for INAT18. A cosine learning rate scheduler is used in fine-tuning experiments. A 0.01 learning rate and 100 training epochs usually provide the best performance for these three datasets. So we fix these two hyperparameters in the following supervised learning two-stage fine-tuning experiments and self-supervised learning experiments.

Two-stage fine-tuning: For the two-stage fine-tuning experiments, we separately fine-tune each sub-network (i.e. RESNET50) of the CATn architecture by the same protocol as the normal fine-tuning above. Then train a last-layer linear classifier on top of the concatenated fine-tuned representation. The last-layer linear classifier training can be very efficient with a proper weights initialization strategy. In this work, we initialize the last-layer classifier w (including the bias term) by concatenating the last-layer classifier of each fine-tuned sub-network w_i, $w \leftarrow [w_0^\top, \dots, w_n^\top]^\top/n$. Then we only need to train the last-layer classifier w for 1 epoch with a learning rate = $1e-3$ for CIFAR and $1e-5$ for INAT18.

B.2. Performance difference between legs (subnetworks) in ERM pretrained $n\times$RESNET50

Table 4 showcases the performance difference between legs of ERM pretrained $n\times$RESNET50. In the $n\times$RESNET50, one leg is doing all the work. In the CATn pretrained $n\times$RESNET50, this is not the case. We believe the difference comes from optimization dynamics.

Table 4. Top-1 IMAGENET accuracy of each leg (RESNET50) of ERM pre-trained nRESNET50. To solely showcase the difference between the representation of legs, we report the training accuracy of fitting a linear classifier on top of the penultimate layer representation of each leg (subnetwork).

	subnetwork0	subnetwork1	subnetwork2	subnetwork3
2×RESNET50	73.94	18.05	-	-
4×RESNET50	9.25	74.33	0.40	0.96

B.3. Experiments on a deeper architecture: RESNET152

Similar to table 2 in section 4, table 5 provides similar experiments on a deeper architecture RESNET152. CATn exceeds ERM on IMAGENET, CIFAR10, CIFAR100 , and INAT18 linear probing tasks.

Table 5. Imagenet supervised transfer learning performance on a deep architecture RESNET152.

		ID	Linear Probing (OOD)		
method	architecture	IMAGENET	CIFAR10	CIFAR100	INAT18
ERM	RESNET152	77.89	92.50	76.23	39.70
CAT2	2×RESNET152	79.34	94.26	79.15	45.42
CAT5	5×RESNET152	80.14	94.91	81.35	50.32
CAT10	10×RESNET152	80.18	95.38	82.39	52.73

B.4. Fine-tuning experiments

For reference, table 6 provides numerical results for the fine-tuning experiments of Figure 1.

B.5. Vision transformer Experiment settings

For all vision transformer experiments, we keep the input image resolution at 384 × 384 and follow a similar protocol as appendix B.1. Specifically, we use a weight decay=5e-4 and a batch size=256 for linear probing, a weight decay=0 and a batch size=512 (following the Dosovitskiy et al. (2020) settings) for fine-tuning and two-stage fine-tuning. Following Dosovitskiy et al. (2020), all input images are normalized by $mean = (0.5, 0.5, 0.5)$, $std = (0.5, 0.5, 0.5)$.

Table 6. Supervised transfer learning by either normal fine-tuning or proposed two-stage fine-tuning. Various representations are pre-trained on IMAGENET and then fine-tuned or two-stage fine-tuned on CIFAR10, CIFAR100, INAT18 tasks.

			fine-tuning			two-stage fine-tuning		
method	architecture	params	CIFAR10	CIFAR100	INAT18	CIFAR10	CIFAR100	INAT18
ERM	RESNET50	23.5M	97.54	85.58	64.19	-	-	-
ERM	RESNET50W2	93.9M	97.76	87.13	66.72	-	-	-
ERM	RESNET50W4	375M	97.88	87.95	66.99	-	-	-
ERM	2×RESNET50	47M	97.39	85.77	62.57	-	-	-
ERM	4×RESNET50	94M	97.38	85.56	61.58	-	-	-
CAT2	2×RESNET50	47M	97.56	86.04	64.49	97.87	87.07	66.96
CAT4	4×RESNET50	94M	97.53	86.54	64.54	98.14	88.00	68.42
CAT5	5×RESNET50	118M	97.57	86.46	64.86	98.19	88.11	68.48
CAT10	10×RESNET50	235M	97.19	86.65	64.39	98.17	88.50	69.07
DISTILL5	RESNET50	23.5M	97.07	85.31	64.17	-	-	-

C. Self-supervised transfer learning

C.1. SWAV on IMAGENET

SWAV is a contrastive self-supervised learning algorithm proposed by Caron et al. (2020). We train RESNET50 on IMAGENET[15] by the SWAV algorithm four times, which gives us four pretrained RESNET50 models. As to the rest four SWAV pre-trained models in this work, we use the public available RESNET50[16], RESNET50W2[17], RESNET50W4[18], and RESNET50W5[19] checkpoints.

Linear probing: Following the settings in Goyal et al. (2022), the linear probing experiments (on IMAGENET, INAT18, CIFAR100, CIFAR10) add a BATCHNORM layer before the last-layer linear classifier to reduce the hyper-parameter tuning difficulty. The learning rate is initialized to 0.01 and multiplied by 0.1 every 8 epochs. Then train these linear probing tasks for 28 epochs by SGD Nesterov optimizer with momentum=0.9. We search L2 weight decay from $\{5e-4\}$, $\{5e-4, 1e-3, 5e-3, 1e-2\}$, and $\{1e-6, 1e-5, 1e-4\}$ for IMAGENET, CIFAR, and INAT18 tasks, respectively.

Fine-tuning:

- **IMAGENET**: Inspired by the semi-supervised IMAGENET fine-tuning settings in Caron et al. (2020), we attach a randomly initialized last-layer classifier on top of the SSL learned representation. Then fine-tune all parameters, using a SGD optimizer with momentum=0.9 and L2 weight decay=0. Low-layers representation and last-layer classifier use different initial learning rates of 0.01 and 0.2, respectively. The learning rate is multiplied by 0.2 at 12^{th} and 16^{th} epochs. We train 20 epochs for networks: RESNET50, RESNET50W2, RESNET50W4. We further search training epochs from $\{10, 20\}$ for the wide network (due to overfitting), RESNET50W5 and then select the best one with 10 training epochs.

- **CIFAR10, CIFAR100, INAT18**: Same as the fine-tuning settings in supervised transfer learning in Appendix B.1.

Two-stage fine-tuning:

- **IMAGENET**: Similar to the two-stage fine-tuning settings in supervised transfer learning, we initialize the last-layer classifier w by concatenation and then train 1 epoch with learning rate=0.001, L2 weight decay=0.

- **CIFAR10, CIFAR100, INAT18**: For CIFAR10, CIFAR100, we use same two-stage fine-tuning settings as in supervised transfer learning in Appendix B.1. For INAT18, we attach a BATCHNORM layer before the last-layer linear

[15] `https://github.com/facebookresearch/swav/blob/main/scripts/swav_400ep_pretrain.sh`
[16] `https://dl.fbaipublicfiles.com/deepcluster/swav_400ep_pretrain.pth.tar`
[17] `https://dl.fbaipublicfiles.com/deepcluster/swav_RN50w2_400ep_pretrain.pth.tar`
[18] `https://dl.fbaipublicfiles.com/deepcluster/swav_RN50w4_400ep_pretrain.pth.tar`
[19] `https://dl.fbaipublicfiles.com/deepcluster/swav_RN50w5_400ep_pretrain.pth.tar`

classifier to reduce the training difficulty. Note that BATCHNORM + a linear classifier is still a linear classifier during inference. Following the linear probing protocol, we train the BATCHNORM and linear layers by a SGD optimizer with momentum=0.9, initial learning rate=0.01, and a 0.2 learning rate decay at 12^{th} and 16^{th} epochs. As to L2 weight decay, we use the same searching space as in the fine-tuning.

C.1.1. ADDITIONAL RESULTS

Beside the SWAV IMAGENET fine-tuning experiments in Figure 4, Figure 7 provides additional SWAV fine-tuning / two-stage fine-tuning results on NATURALIST18, CIFAR100, and CIFAR10 tasks. We give a "[init]cat" curve on the IMAGENET task, but omit the curves on other tasks (NATURALIST18, CIFAR100, and CIFAR10) because they are computationally costly.

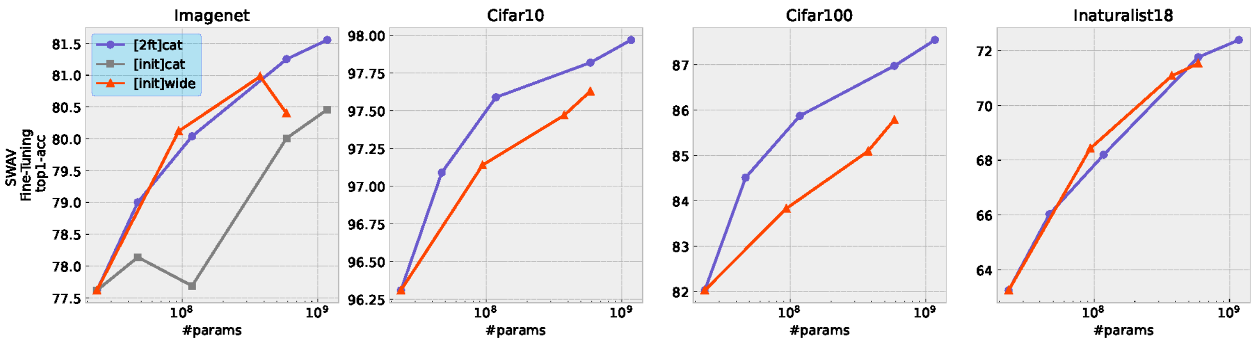

Figure 7. Fine-tuning performance of SWAV on IMAGENET, NATURALIST18, CIFAR100, and CIFAR10 tasks. SWAV is trained on unlabeled IMAGENET. "[2ft]cat" and "[init]cat" indicate our two-stage fine-tuning strategy and the normal fine-tuning strategy on n concatenated networks. "[init]wide" refers to the normal fine-tuning strategy on wide networks, i.e. RESNET50, RESNET50W2, RESNET50W4, and RESNET50W5.

C.2. SEER on INSTAGRAM1B

SEER (Goyal et al., 2022) trains large REGNET{32GF, 64GF, 128GF, 256GF, 10B} architectures on the INSTAGRAM1B dataset with 1 billion Instagram images, using the SWAV contrastive self-supervised learning algorithm.

Linear Probing: Same as the linear probing settings in SWAV.

Fine-tuning: We use SEER checkpoints[20] fine-tuned on IMAGENET with 384×384 resolutions. It is fine-tuned on IMAGENET for 15 epochs using SGD momentum 0.9, weight decay 1e-4, learning rate 0.04 and batch size 256. The learning rate is multiplied by 0.1 at 8^{th} and 12^{th} epochs.

Two-stage Fine-tuning: We keep L2 weight decay 1e-4 the same as fine-tuning. Then keep the other settings the same as in SWAV.

C.3. Additional experiment: SIMSIAM on CIFAR

SIMSIAM (Chen and He, 2020) is a non-contrastive self-supervised learning algorithm. In this section, we pre-train the networks using SIMSIAM on CIFAR10, then transfer the learned representation by linear probing to CIFAR10, CIFAR100, CIFAR10 with 1% training examples, and CIFAR100 with 10% training examples.

SIMSIAM pre-training Following Chen and He (2020) we pre-train RESNET18, RESNET18W2, RESNET18W4, 2RESNET18, and 4RESNET18 on CIFAR10 (32×32 resolution) by SIMSIAM for 800 epochs, using a SGD optimizer with momentum = 0.9, initial learning rate = 0.06, batch size = 512, L2 weight decay = $5e-4$, and cosine learning rate scheduler. The data augmentations include RANDOMRESIZEDCROP (crop scale in $[0.2, 1]$), RANDOMHORIZONTALFLIP, RANDOMGRAYSCALE ($p = 0.2$), and a random applied COLORJITTER $(0.4, 0.4, 0.4, 0.1)$ with probability 0.8. All images are normalized by $mean = (0.4914, 0.4822, 0.4465)$, $std = (0.2023, 0.1994, 0.2010)$ before training.

DISTILL Since self-supervised learning tasks don't contain target labels as supervised learning, we apply knowledge distillation on representation directly. Specifically, we set $v_1, \ldots v_n$ in Figure 5 as Identity matrices. Then we distill

[20] `https://github.com/facebookresearch/vissl/tree/main/projects/SEER`

$[\phi_1, \ldots, \phi_n]$ into Φ by use a cosine loss:

$$\min_{\Phi, w_0, \ldots, w_n} \sum_{i=0}^{n} \sum_{x} \left[1 - \cos\Big(\phi_i(x)\ ,\ w_i \circ \Phi(x)\Big) \right] \tag{4}$$

Linear Probing: Following again the settings of Goyal et al. (2022), the linear probing experiments (on CIFAR100, CIFAR10, CIFAR100(1%) with 10% training data, and CIFAR10(1%) with 1% training data) adds a BATCHNORM layer before the last-layer linear classifier to reduce the hyper-parameter tuning difficulty. We use batch size = 256 for CIFAR100 and CIFAR10, use batch size = 32 for corresponding sampled (10%/1%) version. Then we search initial learning rate from $\{0.1, 0.01\}$, L2 weight decay from {1e-4, 5e-4, 1e-3, 5e-3}. The learning rate is multiplied by 0.1 every 8 epochs during the total 28 training epochs. As to the optimizer, all experiments use a SGD Nesterov optimizer with momentum=0.9.

Results Table 7 shows the linear probing accuracy of SIMSIAM learned representation on various datasets and architectures. When linear probing on the same CIFAR10 dataset as training, the CATn method performs slightly better than width architectures (e.g. RESNET18W2 and RESNET18W4). When comparing them on the CIFAR100 dataset (OOD), however, CATn exceeds width architectures.

Table 7. Linear probing accuracy on CIFAR100, CIFAR10, CIFAR100(1%), and CIFAR10(10%) tasks. The representation is learned on CIFAR10 by SIMSIAM algorithm. CATn concatenates n learned representation before linear probing. DISTILLn distills n learned representation into RESNET18 before linear probing. RESNET18Wn contains around n^2 parameters as RESNET18.

method	architecture	Linear Probing (ID) CIFAR10	CIFAR10(1%)	Linear Probing (OOD) CIFAR100	CIFAR100(10%)
SIMSIAM	RESNET18	91.88	87.60	55.29	42.93
SIMSIAM	RESNET18W2	92.88	88.95	59.41	45.39
SIMSIAM	RESNET18W4	93.50	90.45	59.28	44.98
SIMSIAM	2RESNET18	91.62	87.14	55.67	43.07
SIMSIAM	4RESNET18	92.54	85.65	64.42	49.65
CAT2	2×RESNET18	92.94	88.32	59.40	46.06
CAT4	4×RESNET18	93.42	88.81	63.06	47.48
CAT5	5×RESNET18	93.67	88.78	63.71	48.31
CAT10	10×RESNET18	93.75	88.65	66.19	49.90
DISTILL2	2×RESNET18	93.04	88.59	59.65	45.10
DISTILL5	5×RESNET18	93.02	88.56	60.79	46.41
DISTILL10	10×RESNET18	93.11	88.72	61.35	46.75

C.4. Numerical results

For reference, Tables 8 and 9 provide the numerical results for the linear probing, fine-tuning, and two-stage fine-tuning plots of Figure 4.

D. meta-learning / few-shots learning

D.1. Datasets

CUB (Wah et al., 2011) dataset contains $11,788$ images of 200 birds classes, 100 classes ($5,994$ images) for training and 100 classes ($5,794$ images) for testing.

MINIIMAGENET (Vinyals et al., 2016) dataset contains $60,000$ images of 100 classes with 600 images per class, 64 classes for training, 36 classes for testing.

D.2. BASELINE and BASELINE++ experiment Settings

For BASELINE and BASELINE++ experiments, following Chen et al. (2019), we use RANDOMSIZEDCROP, IMAGEJITTER(0.4, 0.4, 0.4), and HORIZONTALFLIP augmentations, as well as a image normalization $mean = (0.485, 0.456, 0.406)$,

Table 8. Linear probing, fine-tuning, and two-stage fine-tuning performance of SWAV pre-trained representation and corresponding CATn representations.

method	architecture	params	linear-probing IMAGENET	CIFAR10	CIFAR100	INAT18	fine-tuning IMAGENET	two-stage ft IMAGENET
SWAV	RESNET50	23.5M	74.30	91.83	76.85	42.35	77.62	-
SWAV	RESNET50W2	93.9M	77.31	93.97	79.49	47.55	80.12	-
SWAV	RESNET50W4	375M	77.48	94.29	80.51	44.13	80.98	-
SWAV	RESNET50W5	586M	78.23	94.84	81.54	48.11	80.40	-
CAT2	-	47M	76.01	93.48	78.91	45.57	78.14	79.00
CAT5	-	118M	77.43	94.62	81.11	49.12	77.69	80.04
CAT7	-	587M	78.72	95.59	82.71	49.68	80.05	81.25
CAT9	-	1170M	78.89	95.76	83.16	50.61	80.46	81.55

Table 9. Linear probing, fine-tuning, and two-stage fine-tuning performance of SEER pre-trained representation and corresponding CATn representations.

method	architecture	params	linear-probing IMAGENET	CIFAR10	CIFAR100	INAT18	fine-tuning IMAGENET (384px)	two-stage ft IMAGENET (384px)
SEER	REGNET32GF	141M	73.4	89.94	71.53	39.10	83.4	-
SEER	REGNET64GF	276M	74.9	90.90	73.78	42.69	84.0	-
SEER	REGNET128GF	637M	75.9	91.37	74.75	43.51	84.5	-
SEER	REGNET256GF	1270M	77.5	92.16	74.93	46.91	85.2	-
CAT2	-	418M	76.0	92.16	75.65	45.36	-	84.5
CAT3	-	1060M	77.3	93.15	77.26	47.18	-	85.1
CAT4	-	2330M	78.3	93.59	78.80	48.68	-	85.5

$std = (0.229, 0.224, 0.225)$. Then use an ADAM optimizer with learning rate = 0.001, batch size = 16, input image size = 224×224. Finally, train RESNET18 on CUB and MINIIMAGENET datasets for 200 and 400 epochs, respectively. We further tune L2 weight decay from {0, 1e-5, 1e-4, 1e-3, 1e-2} and choose 1e-4 for CUB, 1e-5 for MINIIMAGENET experiments. Compared with the BASELINE and BASELINE++ performance reported by Chen et al. (2019) (table A5), this L2 weight decay tuning process provides $\sim 5\%$ and $\sim 1\%$ improvement on MINIIMAGENET 5way-1shot and 5way-5shot, respectively. In this work, we use this stronger setting in baseline methods.

As to the few-shots learning evaluation, following Chen et al. (2019), we scale images by a factor of 1.15, CENTERCROP, and normalization. Then randomly sample 1 or 5 images from 5 random classes from the test set (5way-1shot and 5way-5shot). Finally, train a linear classifier on top of the learned representation with a SGD optimizer, momentum = 0.9, dampening = 0.9, learning rate = 0.1, L2 weight decay = 1e-3, batch size = 4, and epochs = 100. We take the average of 600 such evaluation processes as the test score.

The BASELINE and BASELINE++ results in Figure 6 report the mean of five runs with different training and evaluating seeds.

Implementation details of the cosine classifier Here we summarize the technical details of the cosine classifier implementation used in this work which follows Chen et al. (2019)[21].

Denote the representation vector as z. The cosine classifier calculates the i^{th} element of logits by:

$$h_i = g_i \frac{\langle u_i, z \rangle}{||u_i|| ||z||} \tag{5}$$

[21] `https://github.com/wyharveychen/CloserLookFewShot/blob/master/backbone.py#L22`

Table 10. Few-shots learning performance on CUB and MINIIMAGENET. The CAT5-S and DISTILL5-S results were obtained using five snapshots taken during a single training episode with a relatively high step size (0.8, SGD). The best snapshot performances are also reported. Standard deviations over five repeats are reported.

			CUB		MINIIMAGENET	
	architecture	classifier	5way 1shot	5way 5shot	5way 1shot	5way 5shot
best snapshot	RESNET18	linear	59.70±1.38	81.35±0.79	52.79±0.92	75.18±0.57
CAT5-S	5×RESNET18	linear	72.62±0.98	86.56±0.82	61.91±0.37	81.06±0.14
DISTILL5-S	RESNET18	linear	68.4±0.5	87.2±0.4	59.9±0.5	80.8±0.4
best snapshot	RESNET18	cosine	65.59±0.87	81.81±0.50	55.67±0.48	75.48±0.46
CAT5-S	5×RESNET18	cosine	73.66±0.82	87.25±0.77	62.94±0.51	81.05±0.16
DISTILL5-S	RESNET18	cosine	75.2±0.8	88.6±0.4	62.0±0.5	81.0±0.3

where u_i is a vector with the same dimension of z, g_i is a scalar, h_i is i^{th} element of logits h.

Then minimize the cross entropy loss between the target label y and softmax output $s(h)$ by updating w and g: $\min_{w,g} \mathcal{L}_{ce}(y, s(h))$.

D.3. CAT and DISTILL experiment settings

For CAT, we concatenate n representation separately trained by either BASELINE or BASELINE++ as the settings above. For DISTILL, we use the same multi-head architecture as figure 5 together with a cross-entropy loss function:

$$\min_{\Phi, w_0, \ldots, w_n} \sum_{i=0}^{n} \sum_{x} \left[(1-\alpha)\mathcal{L}_{ce}\Big(s\big(w_i \circ \Phi(x)\big), y \Big) + \alpha\tau^2 \mathcal{L}_{kl}\Big(s_\tau\big(v_i \circ \phi_i(x)\big) \;||\; w_i \circ \Phi(x) \Big) \right] \tag{6}$$

, where $\mathcal{L}_{ce}$ indicates a cross-entropy loss, α is a trade-off parameter between cross-entropy loss and kl-divergence loss. We set L2 weight decay = 0, $\tau = 10$, search $\alpha \in \{0.8, 0.9, 1\}$, and keep the other hyper-parameters as Appendix D.2. We find the impact of α is limited in both CUB ($\leq 1\%$) and MINIIMAGENET ($\leq 0.5\%$) tasks.

D.4. Snapshots experiment settings

In this section, we apply CAT and DISTILL on 5 snapshots sampled from one training episode (called CAT5-S and DISTILL5-S, respectively). We train CUB and MINIIMAGENET respectively for 1000 and 1200 epochs by naive SGD optimizer with a relevant large learning rate 0.8. Then we sample 5 snapshots, $\{200^{th}, 400^{th}, 600^{th}, 800^{th}, 1000^{th}\}$ and $\{400^{th}, 600^{th}, 800^{th}, 1000^{th}, 1200^{th}\}$, for CUB and MINIIMAGENET, respectively. The other hyper-parameters are the same as Appendix D.2.

D.5. More experimental results

Table 11 provides the exact number in Figure 6, as well as additional CATn and DISTILLn few-shots learning results with a linear classifier (The orange and gray bars in figure 6 report the few-shots learning performance with a cosine classifier).

Table 10 provides more CAT5-S and DISTILL5-S results with either a linear classifier or a cosine-based classifier.

D.6. Comparison with conditional Meta-learning approaches

In order to address heterogeneous distributions over tasks, the conditional meta-Learning approaches Wang et al. (2020); Denevi et al. (2022); Rusu et al. (2018) adapt a part of model parameters conditioning on the target task, while freeze the other model parameters that are pre-trained as a feature extractor.

The results presented in Wang et al. (2020); Denevi et al. (2022); Rusu et al. (2018) already allow us to make some elementary comparisons: Denevi et al. (2022) is derived from Wang et al. (2020). In practice, Wang et al. (2020) reuses the pre-trained frozen feature extractor (WRN-28-10) from Rusu et al. (2018). Table 12 below shows the performance of these conditional meta-learning methods and our DISTILL5 on the MINIIMAGENET few-shot learning task. The first 3 rows are copied from Wang et al. (2020) (marked by *). Despite the fact that the backbone in Wang et al. (2020); Rusu et al. (2018) (WRN-28-10) is wider and deeper than the backbone (RESNET18) used in our paper, DISTILL5 still outperforms both

Table 11. Few-shot learning performance on CUB and MINIIMAGENET dataset with either a linear classifier or cosine-distance based classifier. Standard deviations over five repeats are reported.

			CUB		MINIIMAGENET	
	architecture	classifier	5way 1shot	5way 5shot	5way 1shot	5way 5shot
supervised	RESNET18	linear	63.37±1.66	83.47±1.23	55.20±0.68	76.52±0.42
CAT2	2×RESNET18	linear	66.25±0.85	85.50±0.34	57.30±0.31	78.42±0.17
CAT5	5×RESNET18	linear	67.00±0.18	86.80±0.10	58.40±0.25	79.59±0.17
DISTILL2	RESNET18	linear	69.93±0.74	87.72±0.31	58.99±0.32	79.73±0.21
DISTILL5	RESNET18	linear	70.99±0.31	88.52±0.14	59.66±0.59	80.53±0.27
supervised	RESNET18	cosine	69.19±0.88	84.41±0.49	57.47±0.45	76.47±0.27
CAT2	2×RESNET18	cosine	72.87±0.43	86.82±0.17	60.69±0.24	79.29±0.23
CAT5	5×RESNET18	cosine	76.23±0.55	88.87±0.40	63.63±0.23	81.22±0.17
DISTILL2	RESNET18	cosine	74.81±0.45	88.14±0.40	61.95±0.11	80.79±0.26
DISTILL5	RESNET18	cosine	76.20±0.39	89.18±0.24	62.89±0.38	81.49±0.26

Wang et al. (2020) and Rusu et al. (2018). Other relevant details are summarized in table 13.

If our goal were to present state-of-the-art results exploiting diverse features, a more systematic comparison would be needed. however it is not clear that these results say a lot about how optimization constructs and (prematurely) prunes features. The conditional meta-learning addresses an orthogonal problem but does not seem to fix the premature feature pruning issue. Please not that the message of our paper is that a single optimization run — which is what most people are doing these days - prematurely prunes its representations, missing opportunities to produce the richer representations that benefit out-of-distribution scenarios.

	miniImageNet 5way-1shots	miniImageNet 5way-5shots
LEO (Rusu et al., 2018)	61.76±0.08*	77.59±0.12*
LEO(local) (Rusu et al., 2018)	60.37±0.74*	75.36±0.44*
TASML (Wang et al., 2020)	62.04±0.52*	78.22±0.47*
Distill5 (our)	62.89±0.38	81.49±0.26

Table 12. MINIIMAGENET few-shots learning comparison between DISTILL5 and conditional meta-learning approaches. The first three rows are copied from corresponding papers (marked by *).

	Our backbone	LEO backbone (Rusu et al., 2018; Wang et al., 2020)
Architecture	RESNET18	WRN-28-10
Parameters	11.4M	36.5M
L2 weight decay	✓	✓
Learning rate scheduler	×	✓
Data augmentation (color)	✓	✓
Data augmentation (scale)	✓	✓
Data augmentation (deformation)	×	✓

Table 13. Backbone pretraining details. Note that LEO only keeps the first 21 layers (21.7M parameters) after pretraining WRN-28-10 (Wide residual network). But it is still twice the time larger than RESNET18.

E. Out-of-distribution learning

Following Zhang et al. (2022), we use the CAMELYON17 (Koh et al., 2021) task to showcase the CAT and DISTILL constructed (rich) representation in out-of-distribution learning scenario. The first row of Table 3 is copied from Zhang et al. (2022). The rest results use a frozen pre-trained representation, either by concatenating n ERM pre-trained representations

(CATn), distilling of n ERM pre-trained representations (DISTILLn), or BONSAI constructed representations(BONSAI2). Then train a linear classifier on top of the representation by vREx or ERM algorithms.

For the vREx algorithm, we search the penalty weights from $\{0.5, 1, 5, 10, 50, 100\}$. For DISTILL$n$ representations in the CAMELYON17 task, we follow Algorithm 2 in Zhang et al. (2022), but use a slightly different dataset balance trick in the loss function (Zhang et al. (2022) Algorithm 2 line 13-14). We instead balance two kinds of examples: one shares the same predictions on all ERM pre-trained models, and one doesn't. We keep other settings to be the same as Zhang et al. (2022)[22].

[22] `https://github.com/TjuJianyu/RFC`