Evaluating Self-Supervised Learning via Risk Decomposition

Yann Dubois [1] **Tatsu Hashimoto** [1] **Percy Liang** [1]

Abstract

Self-supervised learning (SSL) pipelines differ in many design choices such as the architecture, augmentations, or pretraining data. Yet SSL is typically evaluated using a single metric: linear probing on ImageNet. This does not provide much insight into why or when a model is better, now how to improve it. To address this, we propose an SSL risk decomposition, which generalizes the classical supervised approximation-estimation decomposition by considering errors arising from the representation learning step. Our decomposition consists of four error components: approximation, representation usability, probe generalization, and encoder generalization. We provide efficient estimators for each component and use them to analyze the effect of 30 design choices on 169 SSL vision models evaluated on ImageNet. Our analysis gives valuable insights for designing and using SSL models. For example, it highlights the main sources of error and shows how to improve SSL in specific settings (full- vs few-shot) by trading off error components. All results and pretrained models are at `github.com/YannDubs/SSL-Risk-Decomposition`

1 Introduction

Self-supervised learning (SSL) is a popular approach for pretraining an encoder from minimal supervision, such that linear probes trained on the encoder's representation perform well on downstream tasks. SSL pipelines differ in many design choices, such as the objective (Chen et al., 2020a; He et al., 2022), architecture (Caron et al., 2021; Bardes et al., 2022b), augmentations (Tian et al., 2020a; Dubois et al., 2022) or pretraining data.Yet SSL models are typically evaluated using a single metric: linear probing on ImageNet. This is convenient for leaderboards but does not provide much insight into why or when a model is better, nor how to improve it. What are the major sources of errors in current SSL methods? Are there tradeoffs between SSl SSL models across different settings (e.g. full- vs few-shot probing)? How does each design choice affect the SSL model? Those are difficult to answer using a single metric.

In supervised learning, one can get more fine-grained insights using the estimation/approximation (or bias/variance) risk decomposition, which is estimated using the training and validation errors. For example, models with low training error and high generalization gap often perform better in large-data regimes and can be improved via regularization. In this paper, we generalize this classical decomposition to SSL. Our decomposition consists of four sources of errors:

1. **approximation** errors due to the encoder's architecture not having the capacity to perform the task;
2. **representation usability** errors due to using SSL followed by linear probing. Usability error is large if a given SSL algorithm fails to produce linearly separable representations that can be used to predict desired tasks;
3. **probe generalization** errors due to finite training data;
4. **encoder generalization** errors due to pretraining the encoder on finite data.

We further provide consistent and computationally efficient estimators for each risk component, akin to the training and validation errors in supervised learning. Using those estimators, we analyze 169 pretrained SSL models and the effect of 30 design choices. These results provide insights into the state of the field, help understand design choices, and suggest which SSL encoder to choose in various settings.

Our analysis highlights that the most important source of error used to be the representation usability but, since SimCLR, it is now the probe generalization. Furthermore, we show that some design choices (e.g. large projection heads, ViT encoders) improve all error components simultaneously. But others (e.g. representations' dimensionality or SSL objective) trade off components and thus only help in specific settings. For example, Fig. 1 shows that SwAV RN50w4 gives more usable representations (bottom left) than MSN ViT-L16 (Assran et al., 2022) but induces a worse probe generalization (bottom right). This results in the former being better in full-shot probing (76% vs 74% accuracy) but worse in 3-shot (37% vs 63%). In summary, we:

[1]Department of Computer Science, Stanford University. Correspondence to: Yann Dubois <yanndubs@cs.stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

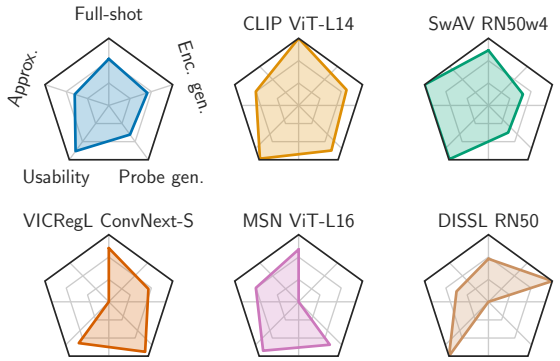


Figure 1: No model is uniformly better over risk components. "full-shot" axis shows linear probing on ImageNet. Other axes show normalized risk components. Higher is better. Top left (blue) shows average over all 169 models.

- provide an SSL risk decomposition with an efficient estimator for each error component;
- show that the main source of error for modern SSL is the generalization error of linear probes;
- highlight a tradeoff between usability and probe generalization, which leads to a few- vs full-shot tradeoff;
- analyze how 30 design choices affect the risk components and full-/few-shot performance of 169 SSL models.

2 Supervised risk decomposition

In supervised learning, one learns a predictor f_S from a hypothesis class $\mathcal{F}$ using a finite set of supervised samples S. The goal is for the predictor to achieve low population risk R_S, which can be evaluated using a test set. When designing models, it is nevertheless typical to consider both the training performance and the generalization gap (the difference between validation and training performance). This is useful to understand which component of the pipeline to improve (regularization, architecture, etc) and which model should be favored depending on the training size $|S|$.

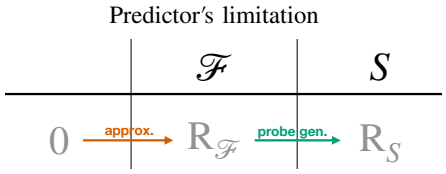


Figure 2: The risk decomposition is a path between settings of increasing expected risk for training the probe: $0 \to \mathrm{R}_{\mathcal{F}}$ (constrained family $\mathcal{F}$) $\to \mathrm{R}_S$ (finite supervised data).

The training performance and generalization gap are respectively estimators of the *approximation error* and the *estimation error* from the supervised risk decomposition (Barron, 1994; Shalev-Shwartz & Ben-David, 2014). [1] The approximation error $\mathrm{R}_{\mathcal{F}}$, is the error that a predictor $f_{\mathcal{F}}$ trained on infinite data incurs, i.e., the error due to the choice of a constrained family $\mathcal{F}$. The estimation error is the error due to training on finite samples, i.e., $\mathrm{R}_{\mathcal{F}} - \mathrm{R}_S$. As seen in Fig. 2, the decomposition arises by considering the difference of risk incurred in settings of increasing expected risk.

Formally, we learn a predictor $f_S := \mathrm{A}_{\mathcal{F}}(\hat{p}_S)$ from a family $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ using an algorithm $\mathrm{A}_{\mathcal{F}}$ (e.g. ERM) on an empirical distribution $\hat{p}_S$ induced by a training set $S \overset{\text{iid}}{\sim} p_{\text{sup}}(X, Y)$. Denote by $\mathrm{R}(f) := \mathbb{E}_{p_{\text{sup}}}[\ell(Y, f_S(X))]$ the risk w.r.t. a desired loss ℓ. To derive the decomposition we order the two risks $\mathrm{R}_S := \mathrm{R}(f_S)$, $\mathrm{R}_{\mathcal{F}} := \inf_{f \in \mathcal{F}} \mathrm{R}(f)$ and use a telescoping sum. Details at Appx. A.1.

3 SSL risk decomposition

Our goal is to derive a risk decomposition for representation learning that allows better development and understanding of SSL. SSL pipelines consist of two models: an encoder ϕ and a probe f. The probe is trained in a supervised fashion and, following Sec. 2, it is useful to consider the errors that arise from using a constrained family $\mathcal{F}$ and finite data S.

The difference with Sec. 2 is that the probe does not predict from inputs X but from their representations $\phi(X)$. As a result, errors also arise from the encoder $\phi \in \Phi$, which is pretrained from a family Φ using an SSL algorithm A_Φ and finite unsupervised data $U \overset{\text{iid}}{\sim} p_{\text{un}}$. The errors can thus come from each of the probe's limitations (constrained $\mathcal{F}$, finite S) as well as each of the encoder's limitations (constrained Φ, SSL algorithm A_Φ, finite U). We now give an overview of each error component, which we formalize later.

The **approximation error** measures errors due to the architecture of the encoder Φ (e.g. ResNet50) and probe $\mathcal{F}$ (e.g. linear) being too constrained to perform even the supervised task. Intuitively, it decreases with the capacity of $\Phi, \mathcal{F}$.

The **representation usability error** measures errors due to learning representations via an SSL pipeline $\mathrm{A}_\Phi, p_{\text{un}}$, rather than supervised learning. Intuitively, it is small if the SSL algorithm ensures that representations retain information that is usable by probes $\mathcal{F}$, e.g., linearly separable classes.

The **probe generalization error** measures the drop in performance due to training the probe on finite samples S instead of p_{sup}. Intuitively, it is small if: (i) the number of training samples $|S|$ is large, or (ii) representations ensure that downstream probes are sample efficient, e.g., by minimizing the margin between same-class examples.

[1] For conciseness, we assume in the main paper that the irreducible error is 0, as it is independent of any design choice. In appendices we instead decompose the excess risk.

$$\underbrace{\mathrm{R}_{U,S}}_{\text{Risk}} = \underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{A,S}}_{\text{encoder generalization}} + \underbrace{\mathrm{R}_{A,S} - \mathrm{R}_{A,\mathcal{F}}}_{\text{probe generalization}} + \underbrace{\mathrm{R}_{A,\mathcal{F}} - \mathrm{R}_{\Phi,\mathcal{F}}}_{\text{representation usability}} + \underbrace{\mathrm{R}_{\Phi,\mathcal{F}}}_{\text{approximation}} \tag{1}$$

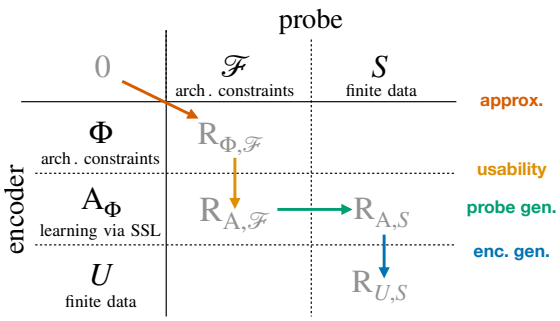


Figure 3: Our SSL decomposition is a path between settings of increasing expected risk. Columns show probe's limitations (constrained $\mathcal{F}$, finite supervised data S) as in Fig. 2. Rows show encoder's limitations (constrained Φ, SSL algorithm A_Φ, finite unlabeled data U). Risk components (colored) are the differences between risks in two settings.

The **encoder generalization error** measures the drop in performance due to pretraining the encoder on finite samples U compared to the population p_{un}. Intuitively, it is small if: (i) A_Φ makes pretraining sample efficient, or (ii) there are many pretraining examples $|U|$.

To derive those risk components we follow Sec. 2 and take the difference in risk between settings of increasing expected risk for the encoder $(\Phi, \mathrm{A}_\Phi, U)$ and probe $(\mathcal{F}, S)$. This gives our SSL risk decomposition Eq. (1), which we illustrate in Fig. 3 as a path through the matrix $(\Phi, \mathrm{A}_\Phi, U) \times (\mathcal{F}, S)$. Each cell corresponds to the risk incurred for a specific limitation for the encoder (row and 1st subscript) and the probe (column and 2nd subscript). Formally:

- $\mathbf{R}_{\boldsymbol{\Phi},\boldsymbol{\mathcal{F}}} := \inf_{f\in\mathcal{F}} \inf_{\phi\in\Phi} \mathrm{R}(f \circ \phi)$ is the risk incurred by the best possible risk for encoders in Φ and probes in $\mathcal{F}$.
- $\mathbf{R}_{\boldsymbol{A},\boldsymbol{\mathcal{F}}} := \inf_{f\in\mathcal{F}} \mathrm{R}(f \circ \phi_A)$ is the risk of the best probe in $\mathcal{F}$ and an encoder $\phi_A := \mathrm{A}_\Phi(p_{\text{un}}) \in \Phi$ pretrained using the desired SSL algorithm and the population distribution.
- $\mathbf{R}_{\boldsymbol{A},\boldsymbol{S}} := \mathrm{R}(f_{\phi_U(S)} \circ \phi_A)$ is the risk incurred by the same encoder but using a probe trained from finite samples $f_{\phi_A(S)} := \mathrm{A}_\mathcal{F}(\hat{p}_{\phi_A(S)})$, where $\phi_A(S) := \{(\phi_A(x), y) \mid (x, y) \in S\}$ is the represented training set.
- $\mathbf{R}_{\boldsymbol{U},\boldsymbol{S}} := \mathrm{R}(f_{\phi_U(S)} \circ \phi_U)$ is the risk when the probe and encoder are trained from finite samples $\phi_U := \mathrm{A}_\Phi(\hat{p}_U)$.

Our decomposition (Eq. (1)) corresponds to the specific path $0 \to \mathrm{R}_{\Phi,\mathcal{F}} \to \mathrm{R}_{A,\mathcal{F}} \to \mathrm{R}_{A,S} \to \mathrm{R}_{U,S}$ in Fig. 3. Considering different paths through matrix would give different decompositions. In Appx. A.2, we provide all other decompositions and show that those would be harder to estimate.

4 Estimating risk components for SSL

Our goal is to compare pretrained SSL models using our decomposition. We would thus like estimators of each risk component that are simple, computationally efficient, consistent, and applicable in the standard SSL ImageNet setting.

Compared to supervised learning, the main new challenge for estimating our risk components compared to supervised learning is that pretraining additional SSL encoders is computationally prohibitive, so we want each of our estimators to use the same SSL encoder. This is a challenge because our risk components are defined using three different encoders (ϕ, ϕ_A, ϕ_U). Our key insight is that we can estimate risk components by changing the training and evaluation set of the probe using the same pretrained SSL encoder.

In the following, we illustrate this for the standard ImageNet SSL setting where the metric comes from pretraining encoders and training probes on the *same* inputs S_{tr}, and evaluating them on $i.i.d.$ examples S_{te}. As a result, we can estimate risk components by training and evaluating probes on specific partitions of $S_{\text{tr}} \cup S_{\text{te}}$ as summarized in Table 1. We now provide the intuition behind each estimator. For formal derivations, properties, and pseudocode see Appx. B. As a reminder, the encoder is always pretrained on S_{tr}.

- $\hat{\mathbf{R}}_{\boldsymbol{U},\boldsymbol{S}}$: We need to estimate the risk when both the encoder and the probe are trained on finite data. They should thus both be evaluated on unseen data. We do so by training the probe on S_{tr} and evaluating it on S_{te}, i.e., we use the standard SSL metric. As S_{te} is disjoint from both the encoder's and probe's (pre)training set S_{tr}, this ensures that both models are evaluated on unseen data.
- $\hat{\mathbf{R}}_{\boldsymbol{A},\boldsymbol{S}}$: We need to estimate the risk when the probe is trained on finite samples but the encoder is pretrained on the population. To do so we use S_{tr} as a plug-in estimate for the population data, which we split into a training $S_{\text{sub}} \subset S_{\text{tr}}$ and testing set $S_{\text{tr}} \setminus S_{\text{sub}}$ for the probe. This ensures that the probe is evaluated on unseen data but not the encoder.
- $\hat{\mathbf{R}}_{\boldsymbol{A},\boldsymbol{\mathcal{F}}}$: We need to estimate the SSL risk when both the encoder and the probe are (pre)trained on the population distribution. We do so by using the *same* pretraining, training, and evaluating set S_{tr}, which ensures that the encoder and probe are evaluated on data they were trained on. $\hat{\mathrm{R}}_{A,\mathcal{F}}$ is thus the training error of the probe

used for standard evaluation.

- $\hat{\mathbf{R}}_{\Phi,\mathcal{F}}$: We need to estimate the risk of the best possible predictor in the composed family $\mathcal{F} \circ \Phi$, without considering SSL or finite samples. We do so using the *training* error of a supervised model with architecture $\mathcal{F} \circ \Phi$, e.g., a ResNet50 on ImageNet.[2]

Our estimators are simple and computationally efficient as they do not require retraining any other SSL encoder.Under mild assumptions, they are all consistent but can be very biased on small datasets. This is similar to how supervised training and testing errors coarsely estimate $\mathrm{R}_{\mathcal{F}}$ and R_{S}.

Table 1: We estimate risk components of an encoder $\phi_U \in \Phi$ pretrained on ImageNet's train set S_{tr}, by training and evaluating probes on different partitions of ImageNet's train S_{tr} and test set S_{te}. $S_{\text{sub}} \subset S_{\text{tr}}$ is a small training subset. $\phi_{\text{sup}} \in \Phi$ is a supervised encoder of the same family.

		Dataset		
Estimator	Encoder	Pretrain	Train	Eval
$\hat{\mathrm{R}}_{U,S}$	ϕ_U	S_{tr}	S_{tr}	S_{te}
$\hat{\mathrm{R}}_{A,S}$	ϕ_U	S_{tr}	$S_{\text{tr}} \setminus S_{\text{sub}}$	S_{sub}
$\hat{\mathrm{R}}_{A,\mathcal{F}}$	ϕ_U	S_{tr}	S_{tr}	S_{tr}
$\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$	ϕ_{sup}	S_{tr}	S_{tr}	S_{tr}

Table 2: Best performing models for ImageNet linear probing. The first 4 categories of rows show models pretrained on ImageNet-1K of various architectures (RN50, any CNN, ViT-S/16, any ViT). The last category allows any data and architecture. Underlined results are best in their category, bolded ones are best overall. Duplicate rows are removed.

			ImageNet probe acc.		
Obj.	Arch.	Param.	100%	1%	3-shot
MoCo-v3	RN50	24M	73.7	55.5	40.4
DINO	RN50	24M	74.2	52.9	35.9
SwAV	RN50w4	375M	76.2	56.2	36.9
VICRegL	CnvNxt-B	85M	74.8	64.3	56.3
MUGS	ViT-S16	22M	77.3	62.9	49.6
MSN	ViT-S16	22M	76.1	67.5	60.4
MSN	ViT-B4	86M	80.1	75.1	69.3
MUGS	ViT-L16	303M	80.9	74.0	68.5
MSN	ViT-L7	303M	79.9	74.9	**69.8**
CLIP	ViT-L14	304M	**85.0**	75.2	62.9
OpenCLIP	ViT-H14	632M	84.4	**75.8**	63.7

[2] $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$ requires training a supervised encoder $\phi \in \mathcal{F} \circ \Phi$, which can be inefficient. Thankfully, this can be reused for SSL models with the same architecture and can often be found online.

5 Experimental results

In the following, we use our risk decomposition to answer the three motivating questions from Sec. 1: What are the major sources of errors in current SSL? Are there tradeoffs affecting which models to prefer in certain settings? How does each design choice affect the SSL model?

To do so we analyze 169 SSL pretrained encoders, across 28 objectives, 20 architectures, and 7 years. For each model, we collected 30 design choices or hyperparameters, estimated our error components, and evaluated the ImageNet test performance of well-tuned linear probes trained in different subsets of ImageNet (100%, 30-shot, 1%, 5-shot, 3-shot).

In our pursuit of addressing our motivating questions, we thus provide the most comprehensive benchmarking of self-supervised learning models to date. We highlight the best-performing models in various settings in Table 2, which we will refer to throughout the section.

We also provide a simple `torch.hub` API at `github.com/YannDubs/SSL-Risk-Decomposition` to load all pretrained encoders, metadata, and results. For experimental details see Appx. C, for raw results see Appx. E, and for extended analysis see Apps. D and F.

5.1 Major sources of errors

In this section, we aim to understand the main sources of errors in current SSL, and how this might change over time. Identifying important sources of errors is potentially useful to understand what research to prioritize.

Fig. 4 shows how error components have changed over time. We now discuss each of them in detail.

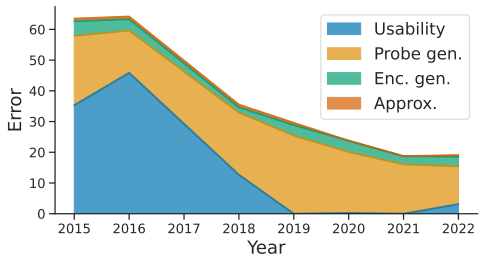


Figure 4: The major SSL improvements came from usability, but probe generalization is now the largest source of error. The plot shows risk components of the best ImageNet-pretrained model published in a given year. Lower is better. In Appx. F.3 we show similar trends for the average models.

Usability drove improvements. We see that usability used to be the largest source of error but it has improved steadily between 2016-2019. In Appx. F.3 we show that those improvements were mostly driven by the use and advances in contrastive learning.

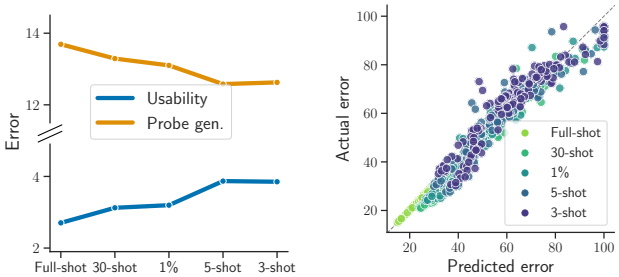
Probe generalization is now key. We see that probe generalization is now the largest source of error, which suggests that it should be prioritized. For example, since 2019, the field has been able to improve overall performance by improving significantly this source of error.

Encoder generalization is small and constant. We see that the encoder generalization has been relatively small over time but might become important in the near future.

The fact that the generalization error is smaller for the encoder than the probe is surprising. Indeed, they are both (pre)trained on the same data (ImageNet's training set) but the encoder is more "complex" than a regularized linear probe. This requires further analysis but could be due to overparametrization (Belkin et al., 2019; Yang et al., 2020).

Approximation error is negligible. Unsurprisingly, current encoders have the capacity to perform the desired task.

For the rest of the paper, we focus on the most common sources of errors: usability and probe generalization.



(a) Risk components (b) Scaling law prediction

Figure 5: Our estimated risk components are tightly related with performance in different settings. (a) Usability error of the best 20% of models increases as the training samples decreases, while probe generalization error decreases. (b) The performance predicted by our scaling law (x-axis) is close to the true performance (y-axis) for all data settings.

5.2 Tradeoffs affecting performance in various settings

In this section, we first show that our estimators of usability and probe generalization are useful to choose which models to prefer in full- or few-shot settings. We then highlight a tradeoff between those two components that directly translates to a tradeoff between full- and few-shot performance.

5.2.1 Predicting performance across settings

Our risk decomposition isolates generalization errors, and should by construction give insights into which models to favor in full- vs few-shot settings. Let us test whether this is also the case when using our simple estimators. As a reminder, error components are estimated on all of ImageNet but we analyze the performance of probes trained on varying number of train samples (100%, 1% and 30-, 5-, 3-shot).

Probe generalization signals sample efficiency. Intuitively, models with low probe generalization error perform better in few-shot settings (less variance) while those with low usability error perform better in full-shot settings (less bias). Fig. 5a shows that, indeed, the best encoders in few-shot regimes have smaller probe generalization errors. Can we use this relation to predict performance across settings?

Error components predict performance across settings. In Appx. F.4 we propose a simple 2-parameter scaling law that fits the performance of all 169 models as a function of estimated error components and the number of training samples $|S|$ (see Fig. 5b). We show that it performs significantly better than standard scaling laws (Kaplan et al., 2020; Rosenfeld, 2021) both in held-out settings (test $R^2 = 0.94$) and held-out encoders (test $R^2 = 0.96$ when holding out contrastive encoders). While the scaling law will not save much compute (probes are efficient to train), it is a useful validation of our risk decomposition and estimators.

5.2.2 Tradeoffs

One advantage of the supervised risk decomposition is that it highlights a tradeoff between approximation/estimation. Although this tradeoff does not always hold (Neal et al., 2018; Yang et al., 2020; Dar et al., 2021), it is a useful conceptual framework for developing models. For example, it suggests that high-capacity predictors perform better when there is plenty of training data and can benefit from regularization.

In Appx. A.5 we derive three corresponding tradeoffs in SSL. Two of those are not insightful as they depend on the negligible approximation error. More interestingly, we derive a usability/probe generalization (U/P) tradeoff. This corresponds to the standard approximation/estimation tradeoff but the gains in capacity come from changing the data (via encoding) rather than the predictor's family $\mathcal{F}$. As an illustration, constant representations lead to probes that perform badly on training (high usability error) but have zero generalization error. In contrast, if the representationa are one-hot encodings of inputs, then linear probes can achieve perfect training performance (usability) but will not generalize.

Usability/probe generalization tradeoff. Similarly to approximation/estimation, U/P is not an exact tradeoff but suggests that decreasing one tends to increase the other. This can be seen in Fig. 4: between 2016-2019 usability decreased at the expense of probe generalization, and vice-versa since 2019. This can also be seen in Fig. 6: at every point in time, the best models seem to form a tradeoff curve.

Table 3: Effect of design choices on error components and full-/3-shot. ↓↓: much better, ↓: better, ↑: worse, ↑↑: much worse.

	# dim. ↓	# views ↑	ViT	# param.↑	MLP proj.	generative SSL	# epoch ↑	Adam
Usability error	↑	↓		↓	↓	↑	↓	
Probe gen. error	↓	↓	↓		↓	↓	↓	↓
Full-shot error	↑	↓	↓	↓	↓	↑	↓	↓
3-shot error	↓	↓	↓	↓	↓	↓	↓	↓

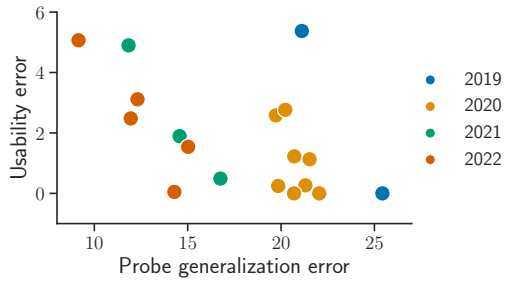


Figure 6: Usability vs probe generalization tradeoff for the best 20% of models for each year (color). Models differ in many design choices (e.g. objective, architecture, epochs).

Full-/few-shot tradeoff. Given the relation between usability/probe generalization and performance in different settings (Sec. 5.2.1), we expect the U/P tradeoff to translate in a full-/few-shot tradoff. Table 2 shows that, indeed, the best models in full-shot (100%) settings are never the best ones in 3-shot. This is true for the 5 considered categories. Fig. 5 suggests that this is indeed driven by the U/P tradeoff.

5.3 Analysing design choices

In this section, we analyze the impact of important SSL design choices on risk components and the performance in full- and 3-shot settings. Table 3 summarizes our findings. We use the following three methods to analyze our results:

- **Controlled analysis (CA).** Whenever possible we analyze the effect of a design choice while fixing others. To do so quantitatively, we fit a linear model from the current (possibly log-transformed) design choice to the metric: $\textit{metric} = \alpha \cdot \textit{hparameter} + \beta^T \mathbb{1}[\textit{model}]$, where $\mathbb{1}[\textit{model}]$ is a one-hot encoding of the value of all other design choices. The downside is that we can only apply CA if we have encoders that only differ in the desired design choice.
- **XGBoost+SHAP.** For each risk component and metric, we train one XGBoost model (Chen & Guestrin, 2016) using all design choices and potential confounders (e.g. year). We then perform feature selection to avoid feature redundancy. Finally, we analyze the SHAP value (Lundberg & Lee, 2017) of the desired design choice. The main disadvantage of XGBoost+SHAP is that there might be other confounders we did not consider.
- **Global linear analysis (GLA)** For each metric and design choice, we train a linear model from all metadata that we think are either important to predict the metric or may be confounders. The downsides of GLA are that it depends on our incomplete "expert knowledge" of how variables interact, and it makes a linearity assumption.

In the main paper, we focus on results from SHAP and qualitative CA, but write "(GLA p-value)" or "(CA p-value)" to show that the other analyses give consistent conclusions. Although different analyses with consistent conclusions mitigate issues with the overall analysis, they do not imply any causal conclusions. For more methodological details see Appx. C.4. For extended analysis of all results see Appx. D.

5.3.1 Dimensionality

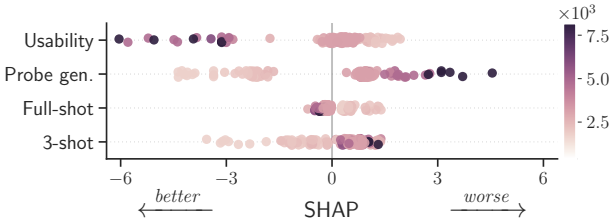


Figure 7: Impact of the representation's dimensionality (color) on the usability error, probe generalization error, and full-/3-shot linear probing. Impact is measured by SHAP values (x-axis). Lower is better as it decreases the risk.

Increasing dimensionality improves usability at the expense of probe generalization. Fig. 7 shows that increasing dimensionality improves usability but worsens probe generalization, which in turn worsens few-shot performance (Sec. 5.2.1). This is further supported by our linear model in the global and controlled setting (GLA/CA p-values <1e-9). In Appx. D.1 we show that what matters is the effective dimensionality (rank) of the representation.

The effect of dimensionality can be intuitively understood by the fact that the capacity of linear classifiers depends on the input dimension d (Vapnik & Chervonenkis, 1971), so increasing d may improve performance but cause overfitting. For a formal explanation see Dubois et al. (2022).

Moving along the U/P tradeoff without retraining. Appx. D.1 suggests that dimensionality might be a simple

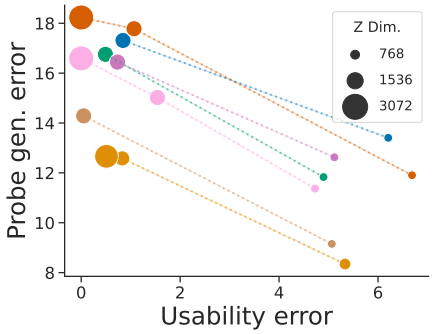


Figure 8: The representation's dimensionality trades off probe generalization and usability. Colors indicate representations from the same ViT. We concatenate CLS tokens from different blocks to vary the dimensionality (dot size).

way to move along the U/P tradeoff. To test this, we take ViT encoders and concatenate CLS tokens from different blocks to increase dimensionality. Fig. 8 shows that this method allows trading off usability and probe generalization.

Table 4: We improve few-shot performance by using representations from layers of smaller dimensionalities ("ours").

Ours	Obj.	ViT	Dim.	100%	1%	3-shot
✗	MUGS	S16	1536	**77.3**	62.9	49.6
✓	MUGS	S16	384	77.0	**66.6**	**57.9**
✗	OpenCLIP	H14	1280	**84.4**	75.8	63.7
✓	OpenCLIP	H14	1024	84.3	**76.5**	**65.5**

Improving performance without retraining. Fig. 8 and Sec. 5.2 suggest that we can extract representations of different dimensionalities from the same encoder to improve performance in desired settings. Indeed, Table 4 shows that we can improve few-shot performance by decreasing dimensionalities. Extracting smaller dimensional representations from the OpenCLIP model even achieves the best overall performance for 1% as seen in Tables 2 and 4. This explains why previous works, e.g. (Caron et al., 2021), showed full-shot improvement when concatenating outputs of ViT blocks, namely, they were increasing the dimensionality.

5.3.2 Data and augmentations

We now analyze the effect of the number of augmentations. We focus on multi-crops given that we have many pretrained models that only differ in this augmentation.

Augmentations improve usability and probe gen. A priori, one might think that using more augmentations improves generalization by acting as a regularizer. Fig. 9a shows that increasing the number of multi-crops actually mostly improves usability — although it can also help probe generalization. Fig. 9b shows similar results when controlling for confounders. Increasing the number of multi-crops thus overcomes the U-P tradeoff, which improves both full- and the few-shot performance (Fig. 9a). In Appx. D.2 we show similar results for other augmentations.

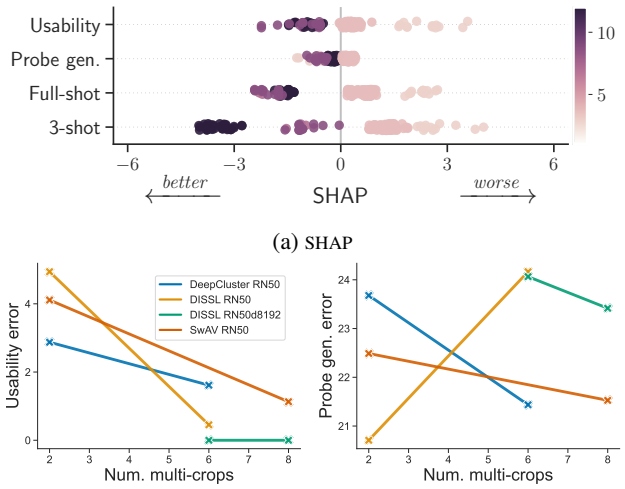


Figure 9: Effect of the number of multicrops on usability and probe generalization error, (a) when considering all models; and (b) when all other hyperparameters are constant.

Strengthening augmentations intuitively improves probe generalization by increasing the invariance of the SSL encoder, which will retain less information that probes can overfit to (Tsai et al., 2021; Tian et al., 2020b; Federici et al., 2020; Mitrovic et al., 2021; Wu et al., 2021; Ruan et al., 2022). The beneficial impact that augmentations have on usability is less obvious but has been suggested by Dubois et al. (2022). Specifically, they prove that stronger augmentations decrease the number of potential tasks and thus the required capacity of probes. Strengthening augmentations thus has a similar impact on usability as increasing the probe's capacity by increasing dimensionality (Fig. 7).

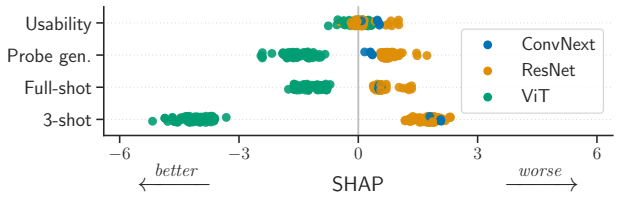
Additional pretraining data can worsen generalization. In Appx. D.2 we show that pretraining on ImageNet-22K, instead of its subset ImageNet-1K, worsens the encoder's and probe's generalization but can improve usability.

5.3.3 Architecture

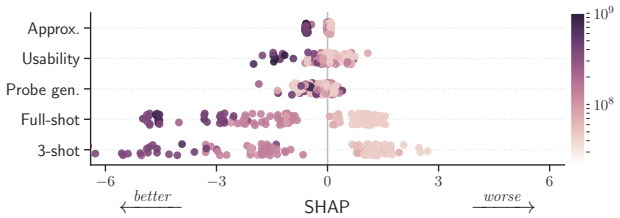
We now analyze the impact of the encoder's architecture.

ViTs improve probe generalization. Fig. 10a shows that ViTs are significantly better than ResNets for probe generalization (GLA p-value=9e-8) and do not worsen usability. This thus translates to few- and full-shot improvements.

Larger encoders improve usability and approximation. Fig. 10b shows that increasing the number of parameters improves the usability and approximation (GLA p-



(a) Achitecture's family



(b) Number of parameters

Figure 10: Impact of the (a) architecture's family, and (b) number of parameters (color) on risk components and aggregated full- or few-shot risk. Lower SHAP values (x-axis) are better as Y-axis are errors.

value=4e-17), without impacting generalization. Those gains improve full- and few-shot performance. In Appx. D.3 we show that smaller ViT patch sizes lead to similar gains.

Now let us analyze the impact of projection heads in SSL, which are known to improve overall full-shot performance (Bachman et al., 2019; Chen et al., 2020a;b).

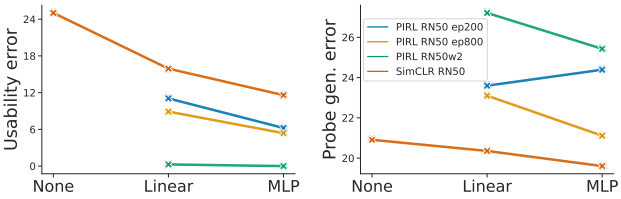


Figure 11: Effect of the projection head on usability and probe generalization error, when all other hyperparameters are kept the same. Each color shows a specific model.

Large projection heads improve usability. Fig. 11 shows that MLP projections improve usability (CA p-value=9e-12) and often also probe generalization. In Appx. D.3 we show that increasing the capacity (number of parameters) of an MLP projection head further improves usability.

Many works have tried to explain why projection heads improve SSL. For example, Jing et al. (2022) suggests that projections avoid dimensionality collapse. In Appx. D.3, we show that projection heads indeed improve effective dimensionality and thus usability (Sec. 5.3.1) but that the increase in effective dimensionality is not larger for non-linear projection heads. This suggests that we still do not completely understand the impact of non-linear projections.

5.3.4 Objective

We now analyze the effect that the objective has on the representation. To simplify the analysis we aggregate all (28) objectives into 6 types (x-axis of Fig. 12).

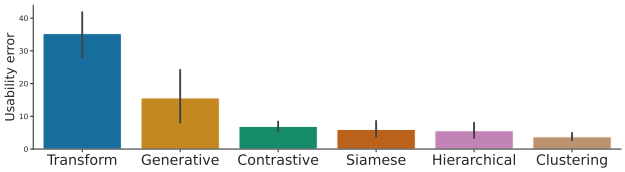


Figure 12: Impact of objective type on usability. Each bar shows the average usability error for all encoders pretrained with that type of SSL objective. Type details in Appx. C.4.

Generative and transformation-predicting objectives suffer from high usability error. Fig. 12 shows that representations learned using objectives that are generative (e.g. MAE or BEiT) or predict the data augmentation (e.g. RotNet or LocNet) are less usable (GLA p-value=3e-4). The other objectives give similar usability, with a slight edge for clustering objectives (e.g. DISSL, DINO, or SwAV).

The lack of usability explains why generative encoders such as MAE do not give a good linear probing performance, despite their strong fine-tuning performance (He et al., 2022). Intuitively, generative objectives preserve all information about the input but do not ensure that this information is usable by linear probes (Xu et al., 2020; Dubois et al., 2020). In comparison, contrastive objectives ensure linear usability because they maximize dot-product similarity (Saunshi et al., 2019; Tosh et al., 2021; HaoChen et al., 2021). More generally, Dubois et al. (2022) shows that many existing SSL losses explicitly optimize for usability.

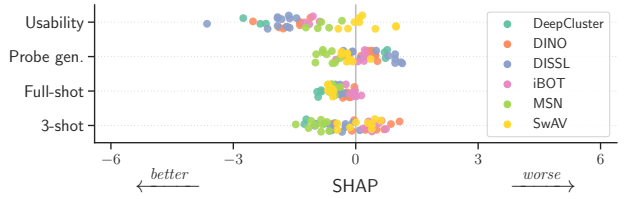


Figure 13: Comparison between clustering objectives.

The exact objective has little impact. Fig. 13 compares different clustering objectives and shows that the impact of the exact objective is relatively minor. For example, the impact on the aggregated risk is at most 1 percentage point. This suggests that one should choose a simple and easy-to-tune objective and focus on other components.

6 Related work

Risk decomposition. The estimation/approximation or the

bias/variance decomposition has been very useful for practitioners and theoreticians to focus on specific risk components (Kohavi & Wolpert, 1996; Domingos, 2000; Valentini & Dietterich, 2004). Such decomposition has nevertheless rarely been extended beyond classical supervised learning. Notable exceptions include (Wu et al., 2020) and (Zhou et al., 2022b) in the context of domain adaptation and federated learning respectively. To our knowledge, we are the first to provide an exact decomposition for SSL, but some theoretical works, e.g., Bansal et al. (2021), have decomposed bounds on the risk (rather than the risk).

Benchmarking SSL. One of our secondary contributions is a thorough benchmark of many SSL models (5 settings, 30 design choices, 28 objective, and 169 models). There have been previous SSL benchmarks but those are either much smaller or use a different evaluation pipeline for each model. For example, Goyal et al. (2019) provides a thorough but small benchmark (3 design choices and 2 objectives). While Goyal et al. (2021) and Contributors (2021) evaluate more models (66 and 22 respectively) but use different evaluation pipelines as their goal is to replicate previous work rather than to provide a fair benchmarking.

Understanding SSL. There is a growing literature of work that tries to explain the effect of specific SSL design choices, e.g. projections heads (Gupta et al., 2022; Appalaraju et al., 2020; Jing et al., 2022) or augmentations (Tsai et al., 2021; Tian et al., 2020b; Federici et al., 2020; Mitrovic et al., 2021; Wu et al., 2021; Dubois et al., 2021), or provide a conceptual framework to think about design choices (Dubois et al., 2022). Sometimes those explanations agree with one another but other times they are orthogonal or even in contradiction. Our work does not provide explanations but rather a new tool to empirically verify previous hypotheses and suggest new ones. For example, in Sec. 5.3 we highlight previous explanations that are supported by our empirical results.

7 Summary and outlook

We present an SSL risk decomposition to provide a fine-grained understanding of the type of errors made by a linear probe predicting from SSL representations. Our risk decomposition generalizes the supervised approximation/estimation decomposition by considering errors arising from the representation learning process. We provide consistent and computationally efficient estimators for each risk component, akin to the training and validation errors in supervised learning. Using those estimators, we analyze 169 pretrained SSL models and the effect of 30 design choices. Our findings suggest that the two primary sources of errors are the usability of the representation, resulting from linear separability issues, and the probe's generalization error, due to finite training data. Furthermore, we show that there is often a tradeoff between these two sources of errors, which translates into a performance tradeoff between few- and full-shot probing. Some design choices, such as the dimensionality of the representation and the SSL objective, can control this tradeoff and thus improve performance in certain settings at the expense of others. Meanwhile, other choices, such as the use of large projection heads and ViT encoders, overcome the tradeoff and thus improve performance in all settings.

Our risk decomposition and in particular our estimators have limitations that should be addressed to improve their applicability. Most notably, they require the probe's training data to be a subset of the encoder's pretraining data, limiting their application in common out-of-distribution settings. We hope that our findings will inspire further research in this direction, and, more generally, the use of risk decompositions for analyzing sources of errors in machine learning.

Acknowledgements

We thank Rohan Taori, Niladri Chatterji, Shibani Santurkar, Ananya Kumar for helpful feedback. YD is supported by a Knights-Hennessy Scholarship. The work is supported by an Open Philanthropy Project Award.

References

Appalaraju, S., Zhu, Y., Xie, Y., and Fehérvári, I. Towards good practices in self-supervised representation learning. *arXiv preprint arXiv:2012.00868*, 2020. (Cited on 9, 29)

Asano, Y. M., Rupprecht, C., and Vedaldi, A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations (ICLR)*, 2020. (Cited on 22)

Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., R., M., and Ballas, N. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision (ECCV)*, 2022. (Cited on 1, 22)

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (Cited on 8, 29)

Bansal, Y., Kaplun, G., and Barak, B. For self-supervised learning, rationality implies generalization, provably. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on 9)

Bao, H., Dong, L., Piao, S., and Wei, F. Beit: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022. (Cited on 22)

Bardes, A., Ponce, J., and LeCun, Y. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022a. (Cited on 22)

Bardes, A., Ponce, J., and LeCun, Y. VICRegl: Self-supervised learning of local visual features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b. (Cited on 1, 22)

Barron, A. R. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14:115–133, 1994. (Cited on 2, 14)

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116:15849–15854, 2019. (Cited on 5, 17, 39)

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011. (Cited on 23)

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007. (Cited on 17)

Bousquet, O. J., Daniely, A., Kaplan, H., Mansour, Y., Moran, S., and Stemmer, U. Monotone learning. In *Conference on Learning Theory (COLT)*, 2022. (Cited on 14)

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on 22)

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. (Cited on 22, 23, 28)

Caron, M., Touvron, H., Misra, I., J'egou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021. (Cited on 1, 7, 22, 23)

Chen, J., Gan, Z., Li, X., Guo, Q., Chen, L., Gao, S., Chung, T., Xu, Y., Zeng, B., Lu, W., Li, F., Carin, L., and Tao, C. Simpler, faster, stronger: Breaking the log-k curse on contrastive learners with flatnce. *arXiv preprint arXiv:2107.01152*, 2021a. (Cited on 22, 23, 29)

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *SIGKDD*, pp. 785–794, 2016. (Cited on 6, 24)

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020a. (Cited on 1, 8, 22, 23, 29)

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b. (Cited on 8, 29)

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c. (Cited on 22)

Chen, X., Xie, S., and He, K. An empirical study of training self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021b. (Cited on 22)

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible scaling laws for contrastive language-image learning. *arXiv preprint arXiv:2212.07143*, 2022. (Cited on 23)

Contributors, M. MMSelfSup: Openmmlab self-supervised learning toolbox and benchmark. `https://github.com/open-mmlab/mmselfsup`, 2021. (Cited on 9)

Dar, Y., Muthukumar, V., and Baraniuk, R. G. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. *arXiv preprint arXiv:2109.02355*, 2021. (Cited on 5, 17, 39)

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. (Cited on 22)

Doersch, C., Gupta, A., and Efros, A. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015. (Cited on 22)

Domingos, P. A unified bias-variance decomposition and its applications. In *International Conference on Machine Learning (ICML)*, 2000. (Cited on 9)

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on 22)

Dubois, Y., Kiela, D., Schwab, D. J., and Vedantam, R. Learning optimal representations with the decodable information bottleneck. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. (Cited on 8, 18)

Dubois, Y., Bloem-Reddy, B., Ullrich, K., and Maddison, C. J. Lossy compression for lossless prediction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on 9, 22, 23, 40)

Dubois, Y., Hashimoto, T., Ermon, S., and Liang, P. Improving self-supervised learning by characterizing idealized representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. (Cited on 1, 6, 7, 8, 9, 18, 22, 23, 27, 29, 30, 31, 40, 41)

Ericsson, L., Gouk, H., and Hospedales, T. M. Why do self-supervised models transfer? investigating the impact of invariance on downstream tasks. *arXiv preprint arXiv:abs/2111.11398*, 2021. (Cited on 40)

Federici, M., Dutta, A., Forré, P., Kushman, N., and Akata, Z. Learning robust representations via multi-view information bottleneck. In *International Conference on Learning Representations (ICLR)*, 2020. (Cited on 7, 9)

Foster, A., Pukdee, R., and Rainforth, T. Improving transformation invariance in contrastive representation learning. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on 40)

Geman, S., Bienenstock, E., and Doursat, R. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992. (Cited on 17)

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised visual representation learning by context prediction. In *International Conference on Learning Representations (ICLR)*, 2018. (Cited on 22)

Goyal, P., Mahajan, D., Gupta, A., and Misra, I. Scaling and benchmarking self-supervised visual representation learning. In *International Conference on Computer Vision (ICCV)*, 2019. (Cited on 9)

Goyal, P., Duval, Q., Reizenstein, J., Leavitt, M., Xu, M., Lefaudeux, B., Singh, M., Reis, V., Caron, M., Bojanowski, P., Joulin, A., and Misra, I. VISSL, 2021. (Cited on 9)

Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap Your Own Latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. (Cited on 22)

Gupta, K., Ajanthan, T., Hengel, A. v. d., and Gould, S. Understanding and improving the role of projection head in self-supervised learning. *arXiv preprint arXiv:2212.11491*, 2022. (Cited on 9, 29, 30)

HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. Provable guarantees for self-supervised deep learning with spectral contrastive loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on 8, 22, 40)

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on 22)

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on 22)

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on 1, 8, 22)

Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. On feature decorrelation in self-supervised learning. In *International Conference on Computer Vision (ICCV)*, 2021. (Cited on 40)

Jing, L., Vincent, P., LeCun, Y., and Tian, Y. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022. (Cited on 8, 9, 29, 40)

Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. (Cited on 5, 39)

Kohavi, R. and Wolpert, D. H. Bias plus variance decomposition for zero-one loss functions. In *International Conference on Machine Learning (ICML)*, 1996. (Cited on 9)

Lundberg, S. M. and Lee, S. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (Cited on 6, 24)

Miao, N., Mathieu, E., Dubois, Y., Rainforth, T., Teh, Y. W., Foster, A., and Kim, H. Instance-specific augmentation: Capturing local invariances. *arXiv preprint arXiv:2206.00051*, 2022. (Cited on 40)

Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on 22)

Mitrovic, J., McWilliams, B., Walker, J., Buesing, L., and Blundell, C. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on 7, 9, 40)

Mukherjee, S., Niyogi, P., Poggio, T., and Rifkin, R. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006. (Cited on 19)

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations (ICLR)*, 2020. (Cited on 17, 39)

Neal, B. On the bias-variance tradeoff: Textbooks need an update. *arXiv preprint arXiv:1912.08286*, 2019. (Cited on 17)

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018. (Cited on 5, 17, 39)

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *iclrworkshop*, 2015. (Cited on 17)

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on 22)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (Cited on 23)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12, 2011. (Cited on 24)

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. (Cited on 22)

Rosenfeld, J., Rosenfeld, A., and Belinkov, Y. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations (ICLR)*, 2020. (Cited on 39)

Rosenfeld, J. S. *Scaling laws for deep learning*. PhD thesis, Massachusetts Institute of Technology, 2021. (Cited on 5)

Ruan, Y., Dubois, Y., and Maddison, C. J. Optimal representations for covariate shift. In *International Conference on Learning Representations (ICLR)*, 2022. (Cited on 7, 40)

Santurkar, S., Dubois, Y., Taori, R., Liang, P., and Hashimoto, T. Is a caption worth a thousand images? a controlled study for representation learning. *arXiv preprint arXiv:2207.07635*, 2022. (Cited on 23)

Saunshi, N., Plevrakis, O., Arora, S., Khodak, M., and Khandeparkar, H. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning (ICML)*, 2019. (Cited on 8)

Saunshi, N., Ash, J. T., Goel, S., Misra, D., Zhang, C., Arora, S., Kakade, S. M., and Krishnamurthy, A. Understanding contrastive learning requires incorporating inductive biases. In *International Conference on Machine Learning (ICML)*, 2022. (Cited on 40)

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. (Cited on 2, 14, 17, 18)

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In *European Conference on Computer Vision (ECCV)*, 2020a. (Cited on 1)

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b. (Cited on 7, 9)

Tosh, C., Krishnamurthy, A., and Hsu, D. Contrastive learning, multi-view redundancy, and linear models. In *Conference on Algorithmic Learning Theory (ALT)*, 2021. (Cited on 8)

Tsai, Y. H., Wu, Y., Salakhutdinov, R. R., and Morency, L. Self-supervised learning from a multi-view perspective. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on 7, 9)

Valentini, G. and Dietterich, T. G. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research (JMLR)*, 5:725–775, 2004. (Cited on 9)

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:2110.02796*, 2019. (Cited on 22)

Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2000. (Cited on 19)

Vapnik, V. N. and Chervonenkis, A. Y. On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei i ee Primeneniya*, 16(2):264–279, 1971. (Cited on 6, 27)

Viering, T., Mey, A., and Loog, M. Open problem: Monotonicity of learning. In *Conference on Learning Theory (COLT)*, 2019. (Cited on 14)

Wang, F. and Liu, H. Understanding the behaviour of contrastive loss. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on 40)

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning (ICML)*, 2020. (Cited on 40, 41)

Wang, X., Zhang, R., Shen, C., Kong, T., and Li, L. Dense contrastive learning for self-supervised visual pretraining. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on 22)

Wang, Y., Zhang, Y., Wang, Y., Yang, J., and Lin, Z. Chaos is a ladder: A new theoretical understanding of contrastive learning via augmentation overlap. In *International Conference on Learning Representations (ICLR)*, 2022. (Cited on 40)

Wu, M., Zhuang, C., Mosse, M., Yamins, D. L. K., and Goodman, N. D. On mutual information in contrastive learning for visual representations. *arXiv preprint arXiv:2005.13149*, 2021. (Cited on 7, 9)

Wu, X., Guo, Y., Chen, J., Liang, Y., Jha, S., and Chalasani, P. Representation bayesian risk decompositions and multi-source domain adaptation. *arXiv preprint arXiv:2004.10390*, 2020. (Cited on 9)

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on 22)

Xu, Y., Zhao, S., Song, J., Stewart, R., and Ermon, S. A theory of usable information under computational constraints. In *International Conference on Learning Representations (ICLR)*, 2020. (Cited on 8)

Yan, X., Misra, I., Gupta, A., Ghadiyaram, D., and Mahajan, D. ClusterFit: Improving generalization of visual representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on 22)

Yang, Z., Yu, Y., You, C., Steinhardt, J., and Ma, Y. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning (ICML)*, 2020. (Cited on 5, 17, 39)

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow Twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, 2021. (Cited on 22)

Zhan, X., Xie, J., Liu, Z., Ong, Y.-S., and Loy, C. C. Online deep clustering for unsupervised representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on 22)

Zhiliang, P., Li, D., Bao, H., Ye, Q., and Wei, F. Beit v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv preprint arXiv:2208.06366*, 2022. (Cited on 22)

Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A. L., and Kong, T. ibot: Image BERT pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021. (Cited on 22)

Zhou, P., Zhou, Y., Si, C., Yu, W., Ng, T. K., and Yan, S. Mugs: A multi-granular self-supervised learning framework. *arXiv preprint arXiv:2203.14415*, 2022a. (Cited on 22)

Zhou, Y., Wu, J., Wang, H., and He, J. Adversarial robustness through bias variance decomposition: A new perspective for federated learning. In *Conference on Information and Knowledge Management (CIKM)*, 2022b. (Cited on 9)

A Risk decompositions

A.1 Supervised decomposition

The goal of supervised learning is to predict targets Y from inputs X sampled from a distribution $p_{\text{sup}}(X, Y)$. The predictor is selected from a desired functional family $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ by an algorithm $\mathrm{A}_{\mathcal{F}} : \mathcal{P}(\mathcal{X}, \mathcal{Y}) \to \mathcal{F}$. For example, empirical risk minimization (ERM) maps the empirical distribution $\hat{p}_S(X, Y)$ of a training set $S \overset{\text{iid}}{\sim} p_{\text{sup}}$ to risk minimizer $f_S := \mathrm{A}_{\mathcal{F}}(\hat{p}_S) \in \mathcal{F}$. The selected predictor f_S is then evaluated using the risk $\mathrm{R}(f_S) := \mathbb{E}_{p_{\text{sup}}}[\ell(Y, f_S(X))]$ with respect to a desired evaluation loss ℓ, e.g., 0-1 loss for classification error. Let us denote the best possible predictor in the desired functional family as $f_{\mathcal{F}} \in \arg\min_{f \in \mathcal{F}} \mathrm{R}(f)$, the Bayes (irreducible) risk by $\mathrm{R}_* := \min_{f:\mathcal{X}\to\mathcal{Y}} \mathrm{R}(f)$, and the $\hat{p}_S$-empirical risk of any predictor f by $\hat{\mathrm{R}}(f; \hat{p}_S)$.[3] For conciseness, we use subscripts to denote the risk $\mathrm{R}_{\mathcal{F}} := \mathrm{R}(f_{\mathcal{F}})$ and $\mathrm{R}_S := \mathrm{R}(f_S)$.

The risk R_S of the selected predictor is ultimately the value that we care about. But when designing, empirically evaluating, and theoretically analyzing a model, it is often helpful to understand the types of errors made by f_S. For example, it is useful to monitor both the generalization gap and the training error to know which pipeline component to improve (regularization, architecture, etc). This can be formalized by the standard excess risk decomposition (Barron, 1994):

$$\underbrace{\mathrm{R}_S - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_S - \mathrm{R}_{\mathcal{F}}}_{\text{estimation error}} + \underbrace{\mathrm{R}_{\mathcal{F}} - \mathrm{R}_*}_{\text{approximation error}}, \tag{2}$$

where the approximation error measures the error due to searching over a constrained family $\mathcal{F}$ and the estimation error quantifies the impact of using finite samples and a non-optimal learning algorithm. Typically, the algorithm is universally consistent so the estimation error does not depend on the algorithm because the predictor $f_{\mathcal{A}} = \mathrm{A}_{\mathcal{F}}(p_{\text{sup}})$ chosen on the population distribution is the best in the family $\mathrm{R}_{\mathcal{F}} = \mathrm{R}_{\mathcal{A}}$, where $\mathrm{R}_{\mathcal{A}} := \mathrm{R}(f_{\mathcal{A}})$. If this is not the case, one can further separate estimation error between generalization ($\mathrm{R}_S - \mathrm{R}_{\mathcal{A}}$) and algorithmic error ($\mathrm{R}_{\mathcal{A}} - \mathrm{R}_{\mathcal{F}}$).

$$\underbrace{\mathrm{R}_S - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_S - \mathrm{R}_{\mathcal{A}}}_{\text{generalization error}} + \underbrace{\mathrm{R}_{\mathcal{A}} - \mathrm{R}_{\mathcal{F}}}_{\text{algorithmic error}} + \underbrace{\mathrm{R}_{\mathcal{F}} - \mathrm{R}_*}_{\text{approximation error}}. \tag{3}$$

To derive the decomposition we order the expected risk of predictors $\mathbb{E}_S[\mathrm{R}_S] \geq \mathrm{R}_{\mathcal{A}} \geq \mathrm{R}_{\mathcal{F}} \geq \mathrm{R}_*$ and write the excess risk as a telescoping sum. By construction, the resulting error components are thus non-negative in expectation. The ordering holds if the algorithm trained on the population data learns a predictor that is at least as good than on any finite samples S, e.g., if the algorithm is a monotonic (Shalev-Shwartz & Ben-David, 2014; Viering et al., 2019; Bousquet et al., 2022). Note that the decomposition could be further expanded by considering other potential sources of errors such as optimization errors.

A.2 Alternative decompositions for representation learning

In the main paper, we saw one possible excess risk decomposition for representation learning. This decomposition is not unique, and we now briefly discuss other possible decompositions. To understand those, it is important to ask ourselves what are the properties of a good risk decomposition. We consider three specific properties, namely, each risk component should ideally: (i) be positive; (ii) highlight important representation learning errors; and (iii) have an efficient estimator.

For positivity to hold in expectation, one simply has to find a sequence of predictors that are ordered by expected risk and then write the final excess risk as a telescoping sum by adding and subtracting respective risks in order. For representation learning, we consider three potential sources of errors $(U, \mathrm{A}_\Phi, \Phi)$: the functional family Φ (e.g. ResNet50), the SSL algorithm A_Φ (e.g. SimCLR optimized with SGD), and the training set (e.g. ImageNet training). For the supervised probe, we essentially have the same choices $(S, \mathrm{A}_{\mathcal{F}}, \mathcal{F})$, but follow the standard supervised excess risk and remove the algorithm choice as it is typically universally consistent. Altogether we have 3 choices for the encoder and 2 for the probe, which can be represented as the matrix $(U, \mathrm{A}_\Phi, \Phi) \times (S, \mathcal{F})$. The question then becomes what ordered sequence to use, i.e., which

[3] For notational convenience, we assume throughout the paper that minimizers are achievable and algorithms are deterministic.

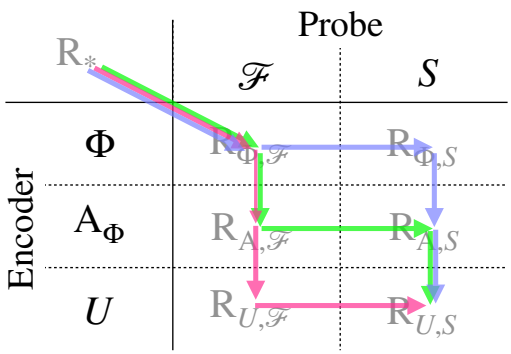


Figure 14: Illustration of the possible loss decompositions corresponding to different ways of traversing the encoder/probe training matrix. In green we see our proposed decomposition, in purple the generalization errors are switched, in pink the usability and probe's generalization are switched.

path to take to traverse the matrix as seen in Fig. 14. We thus have the three following possible (positive) decompositions.

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{A,S}}_{\text{encoder generalization}} + \underbrace{\mathrm{R}_{A,S} - \mathrm{R}_{A,\mathcal{F}}}_{\text{probe generalization}} + \underbrace{\mathrm{R}_{A,\mathcal{F}} - \mathrm{R}_{\Phi,\mathcal{F}}}_{\text{representation usability}} + \underbrace{\mathrm{R}_{\Phi,\mathcal{F}} - \mathrm{R}_*}_{\text{approximation}} \tag{4}$$

Our decomposition. First, there is Eq. (4) (green path in Fig. 14), which is the decomposition whose interpretation we discuss extensively in Sec. 3. The only difference here is that we use start the path from the Bayes Risk R_* instead of zero. We are thus decomposing the excess risk instead of the total risk, as is common in supervised learning (see Appx. A.1). As discussed in Sec. 4, each of our risk components admits practical estimators. Our risk decomposition thus satisfies our three desired properties (positivity, highlight representation learning errrors, and estimation).

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{U,\mathcal{F}}}_{①} + \underbrace{\mathrm{R}_{U,\mathcal{F}} - \mathrm{R}_{A,\mathcal{F}}}_{②} + \underbrace{\mathrm{R}_{A,\mathcal{F}} - \mathrm{R}_{\Phi,\mathcal{F}}}_{\text{representation usability}} + \underbrace{\mathrm{R}_{\Phi,\mathcal{F}} - \mathrm{R}_*}_{\text{approximation}} \tag{5}$$

Switching generalization errors. Another possible decomposition is Eq. (5) (purple path in Fig. 14), which replaces $\mathrm{R}_{A,S}$ with $\mathrm{R}_{U,\mathcal{F}}$. Looking more carefully at ① and ② we see that both risk components have a similar interpretation as in Eq. (4); they are generalization errors. The difference is that it first considers the generalization errors of the predictor ① and then that of the encoder ②. The choice is thus arbitrary in terms of highlighting important representation learning errors. The reason we favored the other decomposition (Eq. (4)) is due to estimation. Indeed, the natural estimator for $\mathrm{R}_{U,\mathcal{F}}$ would be to train and evaluate the probe on the test set S_{te} so that only the probe has to generalize, i.e., $\hat{\mathrm{R}}_{U,\mathcal{F}} := \min_{f\in\mathcal{F}} \hat{\mathrm{R}}(\phi_A \circ f; \hat{p}_{S_{\mathrm{te}}})$. The problem here is that S_{te} is relatively small (50K for ImageNet) and so the $\hat{\mathrm{R}}_{U,\mathcal{F}}$ would greatly underestimate $\mathrm{R}_{U,\mathcal{F}}$ as the probe can overfit S_{te}. In contrast, $\hat{\mathrm{R}}_{A,S}$ is a better estimator as it trains a probe on the much larger $S \setminus S_{\mathrm{sub}}$. In Appx. F.2 we use this second decomposition and show that it would make little impact on our experimental results, despite the worse estimator. This is reassuring as it suggests that our interpretation is robust to the choice of decomposition.

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{A,S}}_{\text{encoder generalization}} + \underbrace{\mathrm{R}_{A,S} - \mathrm{R}_{\Phi,S}}_{③} + \underbrace{\mathrm{R}_{\Phi,S} - \mathrm{R}_{\Phi,\mathcal{F}}}_{④} + \underbrace{\mathrm{R}_{\Phi,\mathcal{F}} - \mathrm{R}_*}_{\text{approximation}} \tag{6}$$

Switching representation usability and probe generalization error. The second possible decomposition is Eq. (6) (pink path in Fig. 14), which replaces $\mathrm{R}_{A,\mathcal{F}}$ with $\mathrm{R}_{\Phi,S}$. As a result, the representation usability③ would be considered before the probe generalization ④. The main downside is that the encoder generalization error does not depend on the pretraining algorithm A_Φ and so one would not be able to quantify how much the representation helps downstream sample efficiency. In

other words, given that we want to understand representation learning, we would like to have as many terms as possible that depend on the representations. Eq. (6) does not highlight/distinguish between important representation learning errors as the probe generalization error does not consider the effect of representations.

A.3 Alternative representation of our decomposition

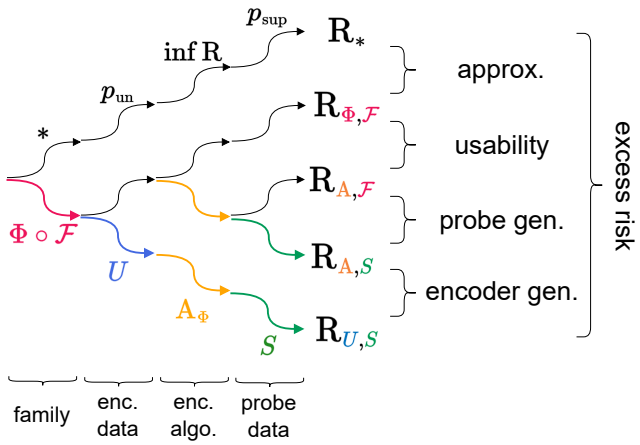


Figure 15: Our excess risk decomposition consists of the difference between risks in settings of increasing difficulty (going down). In particular, we consider 4 potential approximations: (i) constrained functional families $\Phi \circ \mathcal{F}$ instead of unconstrained $*$; (ii) finite pretraining data U instead of the population pretraining distribution p_{un}; (iii) non-optimal representation learning algorithm A_Φ instead of end-to-end risk minimization $\inf \mathrm{R}$; (iv) finite training data S instead of the population training distribution p_{sup}.

In the main paper and in Appx. A.2 we illustrate our risk decomposition as the path in the $(U, \mathrm{A}_\Phi, \Phi) \times (S, \mathcal{F})$ matrix. Another potentially useful illustration is Fig. 15, which shows that our excess risk decomposition consists of the difference between risks in settings of increasing difficulty (more approximation). Changing the order in which we consider different approximations give rise to alternative decompositions (Appx. A.2).

A.4 Relationship with the supervised decomposition

A natural question to ask is how does our risk decomposition for representation learning relates the standard supervised decomposition. The answer is that the former trivially generalizes the latter. In particular, if we define the family of predictors in a supervised setting as the family of composed encoders and probes $\Phi \circ \mathcal{F} := \{\phi \circ f \mid \phi \in \Phi, f \in \mathcal{F}\}$ and the new supervised algorithm $\mathrm{A}_{\Phi\circ\mathcal{F}}$ as a two step algorithm that first fits the encoder using A_Φ (after dropping labels) and then fits the probe with the desired supervised algorithm $\mathrm{A}_\mathcal{F}$, then we have the following equivalences between risk components. On the left we show representation learning components and on the right we show supervised learning components.

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_*}_{\text{rep. excess risk}} = \mathrm{R}((\phi \circ f)_S) - \mathrm{R}_* = \underbrace{\mathrm{R}_S - \mathrm{R}_*}_{\text{sup. excess risk}} \tag{7}$$

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{A,S}}_{\text{encoder generalization}} + \underbrace{\mathrm{R}_{A,S} - \mathrm{R}_{A,A}}_{\text{probe generalization}} = \mathrm{R}_{U,S} - \mathrm{R}_{A,A} = \mathrm{R}((\phi \circ f)_S) - \mathrm{R}((\phi \circ f)_A) = \underbrace{\mathrm{R}_S - \mathrm{R}_A}_{\text{sup. generalization error}} \tag{8}$$

$$\underbrace{\mathrm{R}_{A,A} - \mathrm{R}_{A,\mathcal{F}}}_{\text{probe sup. algorithm}} + \underbrace{\mathrm{R}_{A,\mathcal{F}} - \mathrm{R}_{\Phi,\mathcal{F}}}_{\text{representation usability}} = \mathrm{R}_{A,A} - \mathrm{R}_{\Phi,\mathcal{F}} = \mathrm{R}((\phi \circ f)_A) - \mathrm{R}((\phi \circ f)_{\Phi\circ\mathcal{F}}) = \underbrace{\mathrm{R}_A - \mathrm{R}_{\Phi\circ\mathcal{F}}}_{\text{sup. algorithmic error}} \tag{9}$$

$$\underbrace{\mathrm{R}_{\Phi,\mathcal{F}} - \mathrm{R}_*}_{\text{approximation}} = \mathrm{R}((\phi \circ f)_{\Phi\circ\mathcal{F}}) - \mathrm{R}_* = \underbrace{\mathrm{R}_{\Phi\circ\mathcal{F}} - \mathrm{R}_*}_{\text{sup. approximation error}} \tag{10}$$

In Eq. (9), we introduced the probes supervised algorithmic error, which is natural when recovering the standard risk decomposition with an algorithmic error. As discussed in Appx. A.2 we typically drop this term as it is zero if the supervised algorithm is universally consistent (e.g. ERM) in which case $\mathrm{R}_{A,A} = \mathrm{R}_{A,\mathcal{F}}$ so the probe's generalization recovers in Eq. (8) recovers the definition from the main paper.

We thus see that our risk decomposition recovers the standard supervised decomposition and is a natural extension of it. Note that in the case when we use identity encoders Φ then the encoder's generalization and representation usability become zero. Then, as we would expect, the probe generalization, probe sup. algorithm, and approximation error respectively recover the sup. generalization, sup. algorithmic and sup. approximation error from Appx. A.1.

A.5 Tradeoffs

One of the advantages of using the standard supervised risk decomposition is that it highlights a potential tradeoff between estimation and approximation error (Bottou & Bousquet, 2007; Shalev-Shwartz & Ben-David, 2014). Such a conceptual tradeoff can be very useful to train and develop supervised models, e.g., when using larger models it is often useful to increase the training data or regularization. In the following we discuss three such tradeoffs in representation learning that directly arise from the standard estimation-approximation tradeoff. But first, let us briefly remind that the standard tradeoff (and by extension our tradeoffs) is a conceptual framework rather than a universal theorem.

The approximation-estimation and related tradeoffs are not universal. Although the approximation-estimation tradeoff (or the related bias-vias tradeoff) is typically stated as a universal fact that arises from the decomposition, this is not actually the case. There are usually three arguments given to support those intuitive tradeoffs. The first common argument is the risk decomposition. For example, Shalev-Shwartz & Ben-David (2014) state after providing the decomposition that "these two [approximation and estimation] terms imply a tradeoff between choosing a more complex [hypothesis class] $\mathcal{H}$". But this is only true assuming that the total aggregated risk is constant. An other common argument for the tradeoff is typically given by theoretical bounds on each term. The issue with those bounds is that they typically consider (upper bounds) on the worst-case scenario for constrained predictors rather than what actually happens in practice. In fact, recent theoretical work have argued that this tradeoff does not hold in the over-parameterized regime (Yang et al., 2020; Dar et al., 2021). Finally, the trade-off is often supported using empirical evidence. This is for example done by Geman et al. (1992), which is typically cited when discussing such tradeoff. But the empirical evidence does not universally support such tradeoff. In fact, there is growing empirical evidence that increasing the size of some models (e.g. neural networks) can improve both the approximation and the estimation error (Neyshabur et al., 2015; Belkin et al., 2019; Nakkiran et al., 2020). For a more detailed discussion about the non-universality of the approximation/estimation or bias/variance tradeoffs see Neal et al. (2018); Neal (2019).

Now that we have discussed what the standard approximation-estimation tradeoff is (not), let us see how it gives rise to the following three tradeoffs in our representation learning framework.

- Approximation vs probe generalization
- Approximation vs encoder generalization
- Usability vs probe generalization

Approximation vs probe generalization and **approximation vs encoder generalization.** The first two tradeoffs are direct consequences of the standard approximation-estimation tradeoff. Indeed, as discussed in Appx. A.4, representation learning with probing can be written as a standard supervised setting. In this case, the supervised approximation-estimation tradeoff becomes a tradeoff between the approximation error (Eq. (10)) and the sum of encoder and probe generalization (Eq. (8)). By fixing either the encoder or the probe we then directly get the first two tradeoffs. In the main paper, we do not discuss those two tradeoffs as they are relatively obvious and both contain the approximation error term, which is typically negligible in SSL Fig. 4.

Usability vs probe generalization. To understand the last tradeoff, consider the downstream probing task. For a given encoder, this corresponds to standard supervised learning and we thus know that there is an approximation vs estimation tradeoff. In standard supervised learning, one typically considers the underlying data distribution and the supervised learning algorithm fixed and so the only factor that affects the tradeoff is the predictive family.[4] Holding the data distribution fixed makes sense in standard supervised learning, but for the case of probes we can actually change this distribution by using a

[4] If we do the same in the probing case, then we recover the aforementioned approximation vs probe generalization tradeoff.

different encoder. Indeed, the inputs to the probes are the encoded examples and thus changing the encoder will change the underlying data distribution. The usability-probe generalization tradeoff then corresponds to the probe's supervised tradeoff if we keep the probing family fixed (e.g. linear probe) but modify the data distribution by changing the encoder. Changing the data distribution can indeed change the effective complexity of the probing family, which can be seen by standard data-dependent complexity measures such as the Rademacher Complexity (Shalev-Shwartz & Ben-David, 2014). We thus have a trade-off between the probe's training error and the generalization that is due solely to the pretraining algorithm A_{Φ} rather than the probing family $\mathcal{F}$. On the one hand, if the encoder does not allow the probe to extract any input information (e.g., the representation is a constant) then the representation is not usable (large probe's training error) but the probe generalizes . On the other hand, if the encoder allows the probe to extract all input information (e.g., the representation is a one-hot encoding of the input) then the representation is usable but the probe will overfit.

Given that all aforementioned tradeoffs are directly derived from the standard supervised tradeoffs they are also not universal tradeoffs. For example, it is possible to simultaneously achieve the minimal probe generalization and usability error (Dubois et al., 2020; 2022) despite the U-P tradeoff.

B Estimators

B.1 Supervised decompositon

First, let us review how risk components are estimated in practice when comparing and analyzing supervised learning models. To estimate Eq. (2) we need the following 3 estimators. The main challenge is that the risk components are defined using population risk, but we do not have access to the population distribution p_{sup}. The typical way to overcome this challenge is to use plug-in empirical estimators with the data we have S_{tr} and S_{te}.

$\hat{\mathbf{R}}_{\mathcal{S}}$**.** We want to estimate the risk when the predictor is trained on finite samples. Using the empirical distribution $\hat{p}_{S_{\text{te}}} \approx p_{\text{sup}}$, we get the plugin estimator corresponding to the standard evaluation loss: $\hat{\mathrm{R}}_{\mathcal{S}} := \hat{\mathrm{R}}(f_{\mathcal{S}}; \hat{p}_{S_{\text{te}}}) \approx \mathrm{R}(f_{\mathcal{S}}) =: \mathrm{R}_{\mathcal{S}}$. $\hat{\mathrm{R}}_{\mathcal{S}}$ is unbiased and consistent under standard technical assumptions (e.g. $S_{\text{te}}, S_{\text{tr}} \overset{\text{iid}}{\sim} p_{\text{sup}}$) by the law of large numbers.

$\hat{\mathbf{R}}_{\mathcal{F}}$**.** We want to estimate the risk on the population data. Using the empirical distribution $\hat{p}_{S_{\text{tr}}} \approx p_{\text{sup}}$, we get the plugin estimator corresponding to the training loss: $\hat{\mathrm{R}}_{\mathcal{F}} := \min_{f \in \mathcal{F}} \hat{\mathrm{R}}(f; \hat{p}_{S_{\text{tr}}}) \approx \mathrm{R}(f_{\mathcal{F}}) =: \mathrm{R}_{\mathcal{F}}$. It can be shown to be consistent under technical assumptions (Vapnik, 2000; Mukherjee et al., 2006) but it underestimates the true risk (biased).

$\hat{\mathbf{R}}_{*}$**.** Bayes risk is hard to estimate but is actually not necessary when comparing models as it is only a function of the task.

B.2 Decomposition for representation learning

Algorithm 1 Estimating risk components in the standard SSL setting

Require: Encoder family Φ, probe family $\mathcal{F}$, training S_{tr} and testing S_{te} sets, SSL algorithm A_{Φ}, evaluation loss ℓ.

function RISK($\mathcal{F}$, $\mathcal{D}_{tr}$, $\mathcal{D}_{te}$)
  $\hat{f} \leftarrow \inf_{f \in \mathcal{F}} \sum_{(x,y) \in \mathcal{D}_{tr}} \ell(y, f(x))$ ▷ Risk minimization
  return $\frac{1}{|\mathcal{D}_{te}|} \sum_{(x,y) \in \mathcal{D}_{te}} \ell(y, \hat{f}(x))$ ▷ Test risk
$\hat{\mathrm{R}}_{\Phi,\mathcal{F}} \leftarrow$ RISK($\Phi \circ \mathcal{F}, S_{\text{tr}}, S_{\text{tr}}$) ▷ Supervised train performance
$\phi \leftarrow \mathrm{A}_{\Phi}(\Phi, S_{\text{tr}})$ ▷ Pretrain SSL encoder
$S_{\text{tr}}^{\phi} \leftarrow$ [(ϕ(x),y) **for** x,y **in** S_{tr}] ▷ Featurize data
$S_{\text{te}}^{\phi} \leftarrow$ [(ϕ(x),y) **for** x,y **in** S_{te}]
$S_{\text{sub}}^{\phi} \leftarrow \text{subset}(S_{\text{tr}}^{\phi}, n = \text{len}(S_{\text{te}}))$
$\hat{\mathrm{R}}_{A,\mathcal{F}} \leftarrow$ RISK($\mathcal{F}, S_{\text{tr}}^{\phi}, S_{\text{tr}}^{\phi}$) ▷ Risk without generalization
$\hat{\mathrm{R}}_{A,S} \leftarrow$ RISK($\mathcal{F}, S_{\text{tr}}^{\phi} \setminus S_{\text{sub}}^{\phi}, S_{\text{sub}}^{\phi}$) ▷ Risk with only probe gen.
$\hat{\mathrm{R}}_{U,S} \leftarrow$ RISK($\mathcal{F}, S_{\text{tr}}^{\phi}, S_{\text{te}}^{\phi}$) ▷ Risk with enc. and probe gen.
approx_error $\leftarrow \hat{\mathrm{R}}_{\Phi,\mathcal{F}}$
usability_error $\leftarrow \hat{\mathrm{R}}_{A,\mathcal{F}} - \hat{\mathrm{R}}_{\Phi,\mathcal{F}}$
probe_gen $\leftarrow \hat{\mathrm{R}}_{A,S} - \hat{\mathrm{R}}_{A,\mathcal{F}}$
encoder_gen $\leftarrow \hat{\mathrm{R}}_{U,S} - \hat{\mathrm{R}}_{A,S}$
return approx_error, usability_error, probe_gen, encoder_gen

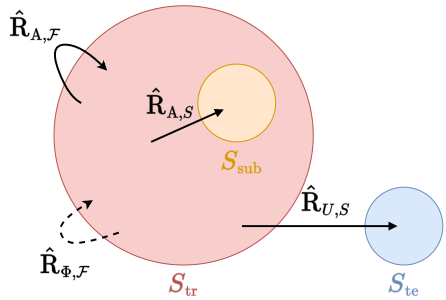


Figure 16: Estimators of our risk components in the standard SSL setting. (Top) Pseudocode. (Bottom) Illustration of the estimators as arrows from the probe's train set to the evaluation set. Full lines mean that we are only training the probe using supervised learning. Dashed line means that we are training both the encoder and the probe using supervised learning.

Fig. 16 provides an illustration and algorithm of the estimators we proposed in Sec. 4. Let us now discuss each estimator in more detail. As a reminder in the standard SSL setting we are in, the pretraining and training data distribution is the same (besides the labels) i.e., $p_{\text{un}} = p_{\text{sup}}$.

$\hat{\mathbf{R}}_{U,S}$. we want to estimate the risk when the families $\Phi, \mathcal{F}$ are constrained, the encoder is pretrained using the algorithm A_Φ, and both the probe and the encoder are trained on finite samples . Using the empirical distributions $\hat{p}_{S_{\text{te}}} \approx p_{\text{sup}}$ and $\hat{p}_{S_{\text{tr}}} \approx p_{\text{un}}$, we get the plugin estimator corresponding to the standard evaluation loss:

$$\begin{aligned}
\mathrm{R}_{U,S} &:= \mathrm{R}(f_S \circ \phi_U) && \text{where } \phi_U := \mathrm{A}_\Phi(\hat{p}_S) \text{ and } f_S := \mathrm{A}_\mathcal{F}(\hat{p}_S) \text{ and } S \overset{\text{iid}}{\sim} p_{\text{sup}} && (11)\\
&\approx \hat{\mathrm{R}}(\hat{f}_S \circ \hat{\phi}_A; \hat{p}_{S_{\text{te}}}) && \text{where } \hat{\phi}_A := \mathrm{A}_\Phi(\hat{p}_{S_{\text{te}}}) \text{ and } \hat{f}_S := \mathrm{A}_\mathcal{F}(\hat{p}_{S_{\text{te}}}) \qquad \hat{p}_{S_{\text{te}}} \approx p_{\text{sup}} && (12)\\
&=: \hat{\mathrm{R}}_{U,S} && (13)
\end{aligned}$$

Similarly to the supervised case (Appx. B.2), $\hat{\mathrm{R}}_{U,S}$ is unbiased and consistent under standard technical assumptions by the law of large numbers.

$\hat{\mathbf{R}}_{A,S}$. we want to estimate the risk when the families $\Phi, \mathcal{F}$ are constrained, the encoder is pretrained using the algorithm A_Φ on the population distribution, but the probe is now trained on finite samples $S \overset{\text{iid}}{\sim} p_{\text{sup}}$. We will again use the empirical distributions $\hat{p}_{S_{\text{tr}}} \approx p_{\text{sup}}$ as a plug in estimate for the population distribution. This means that the finite training data for the probes will need to be sampled from the empirical distribution $\hat{p}_{S_{\text{tr}}}$ to emulate the fact that the probe has to generalize to unseen data. To do so we partition the training data into a small subset S_{sub} on which we train the probe and its complement $S_{\text{tr}} \setminus S_{\text{sub}}$ for evaluation. The final estimator is:

$$\begin{aligned}
\mathrm{R}_{A,S} &:= \mathrm{R}(f_S \circ \phi_A) && \text{where } \phi_A := \mathrm{A}_\Phi(p_{\text{un}}) \text{ and } f_S := \mathrm{A}_\mathcal{F}(\hat{p}_S) \text{ and } S \overset{\text{iid}}{\sim} p_{\text{sup}} && (14)\\
&\approx \mathrm{R}(f_S \circ \hat{\phi}_A) && \text{where } \hat{\phi}_A := \mathrm{A}_\Phi(\hat{p}_{S_{\text{tr}}}) \text{ and } f_S := \mathrm{A}_\mathcal{F}(\hat{p}_S) \text{ and } S \overset{\text{iid}}{\sim} p_{\text{sup}} \qquad \hat{p}_{S_{\text{tr}}} \approx p_{\text{un}} && (15)\\
&\approx \hat{\mathrm{R}}(f_S \circ \hat{\phi}_A; \hat{p}_{S_{\text{sub}}}) && \text{where } \hat{\phi}_A := \mathrm{A}_\Phi(\hat{p}_{S_{\text{tr}}}) \text{ and } f_S := \mathrm{A}_\mathcal{F}(\hat{p}_S) \text{ and } S \overset{\text{iid}}{\sim} p_{\text{sup}} \qquad \hat{p}_{S_{\text{sub}}} \approx p_{\text{sup}} && (16)\\
&\approx \hat{\mathrm{R}}(\hat{f}_S \circ \hat{\phi}_A; \hat{p}_{S_{\text{sub}}}) && \text{where } \hat{\phi}_A := \mathrm{A}_\Phi(\hat{p}_{S_{\text{tr}}}) \text{ and } \hat{f}_S := \mathrm{A}_\mathcal{F}(\hat{p}_{S_{\text{tr}} \setminus S_{\text{sub}}}) \qquad S_{\text{tr}} \setminus S_{\text{sub}} \approx S && (17)\\
&=: \hat{\mathrm{R}}_{A,S} && (18)
\end{aligned}$$

The estimator can be shown to be consistent for the training set $S := S_{\text{tr}} \setminus S_{\text{sub}}$ in the case where $S_{\text{tr}} \setminus S_{\text{sub}}$ is fixed but $|S_{\text{tr}}|, |S_{\text{sub}}| \to \infty$. The estimator is generally biased. One other issue with the estimator is that it is consistent for the training set $S := S_{\text{tr}} \setminus S_{\text{sub}}$ instead of $S := S_{\text{tr}}$. In the case where $|S_{\text{tr}}| \gg |S_{\text{sub}}|$ this should be negligible as $\hat{p}_{S_{\text{tr}} \setminus S_{\text{sub}}}$ will be close to $\hat{p}_{S_{\text{tr}}}$. This is why in practice we use a very small S_{sub}. In particular, for ImageNet we have $|S_{\text{sub}}| = 5\text{e}4$ and $|S_{\text{tr}}| > 1\text{e}6$.

$\hat{\mathbf{R}}_{A,\mathcal{F}}$. we want to estimate the risk when the families $\Phi, \mathcal{F}$ are constrained, the encoder is pretrained using the algorithm A_Φ, but the probe and encoder are pretrained on the population distribution. The challenge is that we do not have access to the population distribution. Using the empirical distributions $\hat{p}_{S_{\text{tr}}} \approx p_{\text{sup}}$, we get a plugin estimator that corresponds to (pre)training the encoder and probe on the same distribution as they are being evaluate it on. This is the standard training error of the probe:

$$\begin{aligned}
\mathrm{R}_{A,\mathcal{F}} &:= \inf_{f \in \mathcal{F}} \mathrm{R}(f \circ \phi_A) && \text{where } \phi_A := \mathrm{A}_\Phi(p_{\text{un}}) && (19)\\
&\approx \inf_{f \in \mathcal{F}} \hat{\mathrm{R}}(f \circ \hat{\phi}_A; \hat{p}_{S_{\text{tr}}}) && \text{where } \hat{\phi}_A := \mathrm{A}_\Phi(\hat{p}_{S_{\text{tr}}}) \qquad \hat{p}_{S_{\text{tr}}} \approx p_{\text{sup}} = p_{\text{un}} && (20)\\
&=: \hat{\mathrm{R}}_{A,\mathcal{F}} && (21)
\end{aligned}$$

The estimator is similar to $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$ in that we use $\hat{p}_{S_{\text{tr}}}$ as a plug in estimate for the pretraining/training/evaluation set. $\hat{\mathrm{R}}_{A,\mathcal{F}}$ can thus also be shown to be consistent (as $|S_{\text{tr}}| \to \infty$) under the technical assumptions but it is biased (typically underestimates the true risk).

$\hat{\mathbf{R}}_{\Phi,\mathcal{F}}$. we want to estimate the best achievable risk for a given encoder and probe family $\Phi \circ \mathcal{F}$. The problem is that we do not have access to the population distribution. Using the empirical distributions $\hat{p}_{S_{\text{tr}}} \approx p_{\text{sup}}$, we get a plugin estimator that corresponds to the empirical risk minima (i.e. the training loss of a supervised model):

$$\mathrm{R}_{\Phi,\mathcal{F}} := \inf_{f \in \mathcal{F}} \inf_{\phi \in \Phi} \mathrm{R}(f \circ \phi) \tag{22}$$

$$\approx \inf_{f\in\mathcal{F}} \inf_{\phi\in\Phi} \hat{\mathrm{R}}(f\circ\phi; \hat{p}_{S_{\mathrm{tr}}}) \qquad \hat{p}_{S_{\mathrm{tr}}} \approx p_{\mathrm{sup}} = p_{\mathrm{un}} \tag{23}$$

$$=: \hat{\mathrm{R}}_{\Phi,\mathcal{F}} \tag{24}$$

Just as with the supervised case (Appx. B.2) it can be shown to be consistent (as $|S_{\mathrm{tr}}| \to \infty$) under the technical assumptions but it underestimates the true risk (biased). Indeed, this is the supervised empirical risk minima for predictors in $\Phi \circ \mathcal{F}$. Note that $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$ requires training a supervised model (empirical risk minimizer). This can be computationally prohibitive for large Φ, but is only required once per architecture and such pretrained model can often be found online. One issue with online models is that their empirical risk typically overestimate the desired minimal risk, as they are typically regularized.

$\hat{\mathbf{R}}_*$**.** Just as in the supervised case (Appx. B.2), the Bayes risk is unknown but it only depends on the task so we can disregard it is the same for all compared models.

The properties of the estimators are summarized in Table 5

Table 5: Properties of each estimator.

estimator	consistent	unbiased	computationally efficient
$\hat{\mathbf{R}}_{U,S}$	✓	✓	✓
$\hat{\mathbf{R}}_{A,S}$	✓	✗	✓
$\hat{\mathbf{R}}_{A,\mathcal{F}}$	✓*	✗	✓
$\hat{\mathbf{R}}_{\Phi,\mathcal{F}}$	✓	✗	✗†

* The estimator is consistent for the training set $S = S_{\mathrm{tr}} \setminus S_{\mathrm{sub}}$ rather than $S = S_{\mathrm{tr}}$.

† The estimator requires training a supervised model of architecture $\Phi \circ \mathcal{F}$, which can be inefficient. This is only required once per architecture and thus becomes efficient when comparing multiple models of the same architecture. Furthermore, such supervised model can often be found online.

C Experimental details

C.1 Open source API

All the pretraining encoders, their associated metadata, and the results discussed below are available via a simple and unified API using respectively:

Models `torch.hub.load("YannDubs/SSL-Risk-Decomposition:main", encoder)` returns a pretrained pytorch `encoder` and the `preprocessing` pipeline. For all our models a `PIL` image `x` can be encoded using `encoder(preprocessing(x).unsqueeze(0))`. A list of available models can be found using `torch.hub.list("YannDubs/SSL-Risk-Decomposition:main")`. Each model's name is `<objective>_<architecture>_<other>`, where `other` is some compressed metadata that we use to distinguish models (it is the same as the "other" column in Table 7).

Metadata `torch.hub.load("YannDubs/SSL-Risk-Decomposition:main", "metadata_df")` returns a pandas dataframe of all metadata. For a nested dictionary use `"metadata_dict"` instead.

Results `torch.hub.load("YannDubs/SSL-Risk-Decomposition:main", "results")` returns a dataframe of all evaluated metrics (corresponding to Table 7).

More details and our evaluation code can also be found at `github.com/YannDubs/SSL-Risk-Decomposition`.

C.2 Pretrained models and metadata

Aside from the 14 SSL models we pretrained, all others were taken from: torch hub, torchvision, VISSL, timm, Hugging Face , MMSelfSup, PyContrast, or from the official GitHub repository of the considered model.

In total, we consider 169 pretrained encoders that we broadly categorize in the following categories:

Predicting transformations First, there are the encoders that are pretraiend by essentially predicting the augmented transformation. In particular, LocNet (Doersch et al., 2015), Jigsaw (Noroozi & Favaro, 2016), RotNet (Gidaris et al., 2018).

Contrastive We use contrastive to mean any methods that use some derivative of InfoNCE (van den Oord et al., 2019). Specifically, we consider NPID (Wu et al., 2018), NPID++ (Misra & van der Maaten, 2020), PIRL (Misra & van der Maaten, 2020), MoCo (He et al., 2020), MoCov2 (Chen et al., 2020c), MoCov3 (Chen et al., 2021b), SimCLR (Chen et al., 2020a), CLIP (Radford et al., 2021), Lossyless (Dubois et al., 2021), SpecCL (HaoChen et al., 2021).

Hierarchical We use hierarchical to mean methods that have a local and global component of the loss. Specifically, we consider DenseCL (Wang et al., 2021), MUGS (Zhou et al., 2022a), VICRegL (Bardes et al., 2022b). .

Clustering We use clustering to mean any method where representations are learned by predicting clusters of the data (e.g. via a clustering step or jointly learned by a teacher). Specifically, we consider DeepCluster (Caron et al., 2018), ClusterFit (Yan et al., 2020), SwAV (Caron et al., 2020), DeepClusterv2 (Caron et al., 2020), Selav2 (Asano et al., 2020; Caron et al., 2020), ODC (Zhan et al., 2020), iBOT (Zhou et al., 2021), DINO (Caron et al., 2021), DISSL (Dubois et al., 2022), MSN (Assran et al., 2022).

Siamese We call "siamese" models that do not nicely fall in the previous categories but still use siamese networks. This includes BYOL (Grill et al., 2020), SimSiam (Chen et al., 2021a), Barlow Twins (Zbontar et al., 2021), VICReg (Bardes et al., 2022a).

Generative We consider models that were pretrained with variants of Bert-style (Devlin et al., 2019) masking for vision. Specifically, we consider BEiT (Bao et al., 2022), BEiTv2 (Zhiliang et al., 2022), and MAE (He et al., 2022).

Supervised Finally, we also download and evaluate (with linear probing) pretrained supervised models. The reason is two-fold. First, supervised models of the same architecture are an important baseline to understand the performance of SSL encoders. Second, those models are used to estimate the approximation error as discussed in Sec. 4. In particular, we considered supervised ViTs (Dosovitskiy et al., 2021) and ResNets (He et al., 2016) of various architecture.

Note that for each of the SSL models we consider different hyperparameters, such as the encoder's architecture or the number of training epochs). For each of the pretrained model we also collected (to the best of our ability) metadata including information about the SSL objective, the architecture, the pretraining data, the representation, the pretraining optimization, and the compute budget. In particular, we collected the following information when applicable and available.

- SSL objective
- SSL category
- version of the objective
- number of negatives
- number of classes
- uses stop-gradients?
- uses EMA encoder?
- output dim. of proj.
- width of proj. head
- depth of proj. head
- architecture
- architecture family
- patch size
- architecture of proj. head 1
- architecture of proj. head 2
- weight tying between proj. head?
- # of parameters for encoder
- # of param. for proj.
- dim. of representation
- representation layer
- epochs
- batch size
- optimizer
- learning rate
- weight decay
- learning rate scheduler
- pretraining data
- finetuning data
- image size
- number of views
- invariant to aug?
- list of augmentations
- publication date
- license of weights
- official weights?
- model trained in industry?
- pretraining time
- type of pretraining machine
- number of pretraining machines

C.3 Evaluating all metrics

One of the contributions of our paper is to provide a thorough and fair linear probing evaluation of 169 pretrained models in 5 different label settings (100%, 30-shot, 1%, 5-shot, 3-shot). We now describe the evaluation pipeline for each of the models. The code is available online at `github.com/YannDubs/SSL-Risk-Decomposition`.

Featurization. For each pretrained model, we first featurize the entire ImageNet dataset (train and test) similarly to Cherti et al. (2022); Dubois et al. (2022; 2021); Santurkar et al. (2022). This differs from the standard SSL pipeline where images are featurized on-the-fly at every step (Caron et al., 2021; 2020; Chen et al., 2020a; 2021a). The advantage of prefeaturization is that training a probe becomes $\sim$1000$\times$ faster ($\sim$100 GPU hours $\rightarrow$ $\sim$10 min). The disadvantage is that we cannot use data augmentations to train the probe, which decreases accuracy by an average of 1 percent point.

For the following estimators, we essentially follow Algorithm 1.

Full-shot linear probing or $\hat{\mathrm{R}}_{U,S}$. To evaluate full-shot linear probing we use PyTorch (Paszke et al., 2019) and tune the following hyperparameters: lr, weight decay, batch size, is batchnorm, optimizer, scheduler. In particular, we see that the linear probe is potentially regularized. The hyperparameters are tuned using 30 steps of the Tree Parzen Estimator algorithm (TPE; (Bergstra et al., 2011)) to minimize a validation error. For computational efficiency, we only train the probe on 10% of ImageNet during tuning. Once the hyperaparameters are tuned we train the linear probe on all of ImageNet and return the test error. This corresponds to our desired full-shot metric as well as $\hat{\mathrm{R}}_{U,S}$.

Estimating $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$. To compute $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$ we need to train a supervised encoder of the desired architecture (Algorithm 1), which can be computationally prohibitive. As there are many online available supervised model, we, instead, download the model of the desired architecture (e.g. ResNet50) and evaluate its training performance. One issue with this strategy is that models available online are typically tuned to perform well on a validation set, rather than on a training set as desired. This means that we actually overestimate $\hat{\mathrm{R}}_{\Phi,\mathcal{F}}$ and thus the approximation error. This should not be a major issue given that our results show that the approximation error is actually very small (see Appx. F.3), e.g., for a ResNet50 we get $\hat{\mathrm{R}}_{\Phi,\mathcal{F}} = 0.84$ and so we don't overestimate the error by much.

Estimating $\hat{\mathrm{R}}_{A,S}, \hat{\mathrm{R}}_{A,\mathcal{F}}$. For $\hat{\mathrm{R}}_{A,S}, \hat{\mathrm{R}}_{A,\mathcal{F}}$ we follow the tuning pipeline used for $\hat{\mathrm{R}}_{U,S}$ (full-shot linear probing), the only difference being the train/validation/test data. Specifically, we always tune the probe on a dataset that mirrors the evaluation set. For example, for $\hat{\mathrm{R}}_{A,\mathcal{F}}$ the probe is trained and tested on ImageNet's train set (Algorithm 1), and so tuning is performed on the training set. For $\hat{\mathrm{R}}_{A,S}$ the probe is evaluated on S_{sub} (where $|S_{\text{sub}}| = 50K$) and evaluated on $S_{\text{tr}} \setminus S_{\text{sub}}$, for tuning we do the same but use a different S_{sub}.

Risk components. Once we have $\hat{\mathrm{R}}_{A,S}, \hat{\mathrm{R}}_{A,\mathcal{F}}, \hat{\mathrm{R}}_{\Phi,\mathcal{F}}, \hat{\mathrm{R}}_{U,S}$ we compute the risk components by using their definitions (see last lines of Algorithm 1)

Few-shot linear probing. To compute the few shot linear probes, we the same high-level pipeline as for the full-shot probing but now use sklearn's (Pedregosa et al., 2011) logistic regression with the lbfgs solver, which we found to be more efficient than PyTorch. We tune only the regularization parameter C using again 30 rounds of TPE.

C.4 Evaluating the impact of different hyperparameters

Given all the hyperparameters and metrics (performance in different settings and risk decomposition) that we have collected, we now want to evaluate the impact of each of the former on the latter. We do so using three different methods:

Controlled analysis (CA) and linear model . The most obvious way to analyze the impact of a hyperparameter on some metric is to consider models that differ only w.r.t. that hyperparameter. When such models are available, we train a linear model to predict the impact of that hyperparameter on the desired metric. Specifically, we train $f(\text{metric}) = \alpha \cdot f(\text{hyperparam}) + \boldsymbol{\beta} \cdot [\text{model}]$ where "metric"" denotes the metric we are predicting,$\alpha, \boldsymbol{\beta}$ are respectively a scalar and vector parameter fitted by least-squares, "[model]" is a one-hot encoding of the current model (models that differ in any other hyperparameter will have a different encoding), and $f()$ denotes either a log function or the identity whichever is best.

This controlled analysis has the advantage of removing the impact of any potential confounders. The disadvantage is that it only quantifies (potentially log) linear relationships, and there are not that many models that only differ in a single hyperparameter so there is a coverage and statistical power issue.

Table 6: Percentage of explained test variance (estimated by 30-fold cross-validation) for our XGBoost models before and after filtering. Each column corresponds to a different model predicting the given metric.

	Approx.	Usability	Probe gen.	Enc. gen.	Full-shot	3-shot
Pre-filtering	96.10	65.46	86.41	43.52	85.28	92.69
Post-filtering	89.59	68.17	87.26	41.35	86.05	92.48

XGBoost + SHAP values. We train one XGBoost model (Chen & Guestrin, 2016) for each metric that takes 51 available hyperparameters as inputs. We tune each of them separately using 50 runs of Bayesian hyperparameter tuning (Tree-structured Parzen Estimator) with 10-fold cross-validation. We then use the XGBoost models to give us the importance of each hyperparameter on a specific metric using SHAP values (Lundberg & Lee, 2017), which essentially estimates the impact of not using a certain hyperparameter for prediction. One issue with the above strategy is that when the hyperparameters are highly correlated it is hard to quantify the impact of those hyperparameters. To avoid such a problem, we filter features so as to decrease correlation without decreasing the cross-validation performance. This allows us to decrease the number of hyperparameters to 14 without decreasing the performance of the Xgboost model. The 14 hyperparameters that we retain are: `['objective', 'architecture', 'patch_size', 'epochs', 'pretraining_data', 'projection2_arch', 'nviews', 'z_dim', 'family', 'ssl_mode', 'n_parameters', 'n_augmentations', 'optimizer', 'projection_nparameters_hidden']`. When evaluating hyperparameters that are in that list, we use those models trained after feature selection and we use the full model otherwise.

Table 6 shows the percentage of test variance explained by the XGBoost model before and after features selection, we see that pos-filtering performs surprisingly well given that it needs to predict the performance on unseen models given only 14 hyperparameters and using less than 200 training examples. The model does nevertheless struggle for encoder generalization and to a lesser extent usability, which suggests that we might have failed to consider an important hyperparamater.

The main advantage of using XGBoost + SHAP values is that we can quantify non-linear relations and arbitrary interactions between hyperparameters, and that the output depends on all models (rather than only the ones that differ in a single hyperparameter). The disadvantage is that XGboost+SHAP values are harder to interpret and we cannot quantify statistical significance.

Global linear analysis (GLA). Finally, we also train a (potentially log) linear model to predict the metric using the desired hyperparameter while controlling for all other main hyperparameters that are not directly related to the desired hyperparameter. For example, when evaluating the impact of the "architecture" we do not condition on the "z_dim" or the

model "family" as those a directly related to the architecture. The advantage of this global linear model is that it does not suffer from the same coverage/statistical power issue than the controlled analysis. The issue is that the model is very simple (linearity without any interaction term) and we might not correctly control all confounders.

All of the above methods have some complementary advantages and disadvantages for interpreting the impact of a hyperparameter, which is why we consider the three simultaneously.

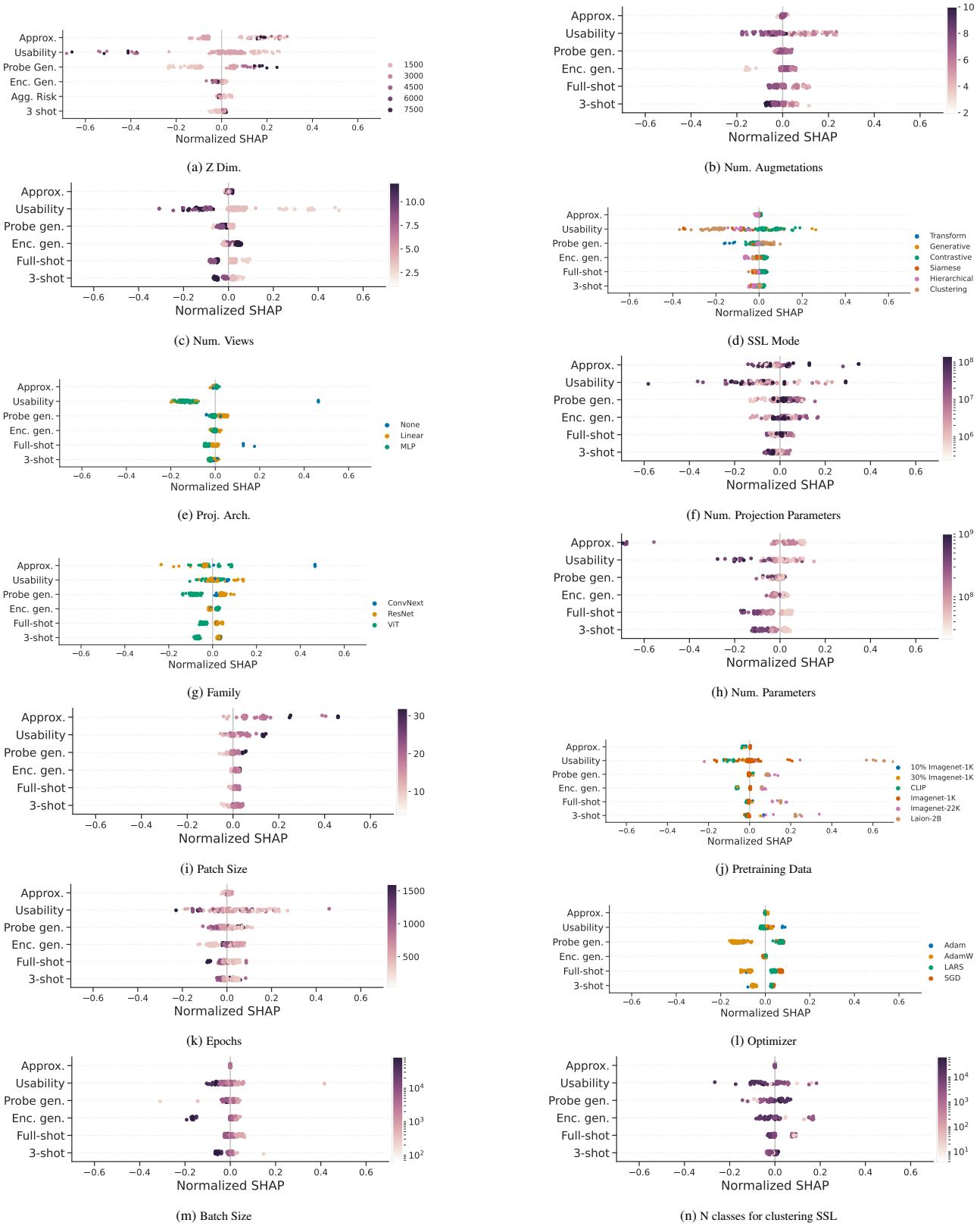


Figure 17: Impact of important hyperparameters. Each plot shows a hyperparameter. Each point shows a different model. The Y-axis shows the metric, either the risk component or the total risk in the full ("Agg. Risk') and few-shot regime ("3 shot"). The X-axis shows the normalized SHAP value. **Negative values mean that a hyperparameter is beneficial**: it decreases the risk. Axes cut to $[-0.7, 0.7]$.

D Impact of hyperparameters

Throughout this section, we will analyze the impact of different hyperparameters on the following metrics: every decomposed risk component (approximation error, usability error, probe generalization error, encoder generalization), the aggregated risk of a linear probe trained on all of ImageNet, and the aggregated risk of a linear probe trained in a 3-shot setting. We evaluate the importance of each hyparameter using XGBoost+SHAP, linear models in a controlled setting, and linear models in general settings as described in Appx. C.4.

Impact of each hyperparameter. A summary of how all hyperparameters impact each metric can be seen in Fig. 17. It shows, for each model (point in the scatter plot) how important the value of a certain hyperparameter (the color) is for each of the metrics (Y-axis) as measured by the SHAP value from the XGBoost model normalized by the average value of that metric (X-axis). Note that every metric is a risk measure, so a lower SHAP value is better. For the rest of the section, we will discuss the impact of key hyperparameters on usability and probe generalization.

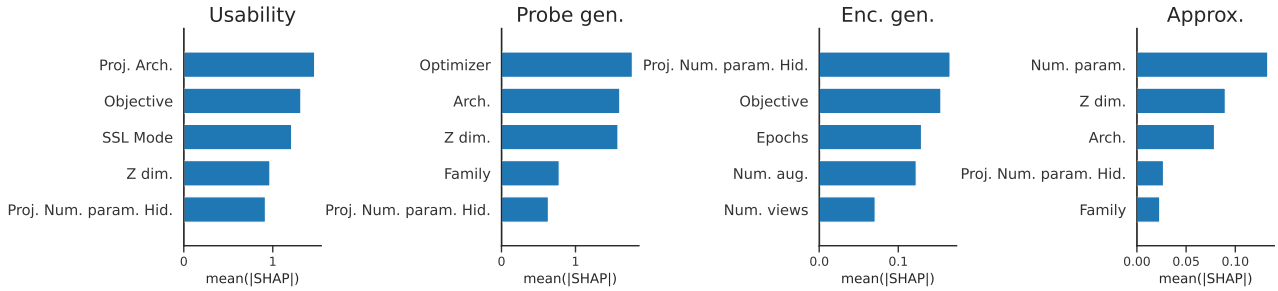


Figure 18: Most important parameters for each risk component as measured by the mean absolute SHAP value of an XGBoost model.

Most important hyperparameter for each metric. A summary of the most important hyperparameters for each metric can be seen in Fig. 18, which shows the average absolute SHAP value. We see that usability is mostly impacted by the dimensionality, the projection head ("Proj. Arch." and "Proj. #param"), and the objective ("objective" and "SSL Mode"). Probe generalization is mostly impacted by the dimensionality, the architecture ("Arch." and "Family"), and the optimizer. We will investigate each of those more carefully in the rest of the section. We see that the approximation error is mostly impacted by the architecture ("Num. param.", "Family", 'Z dim.', and "Arch.") as one would expect given that SSL hyperparameters should not impact this error. We also see that the encoder generalization depends on the augmentations ("augmentations" and "views"). Overall we see that the dimensionality and the projection head seem to be important design choices for all components.

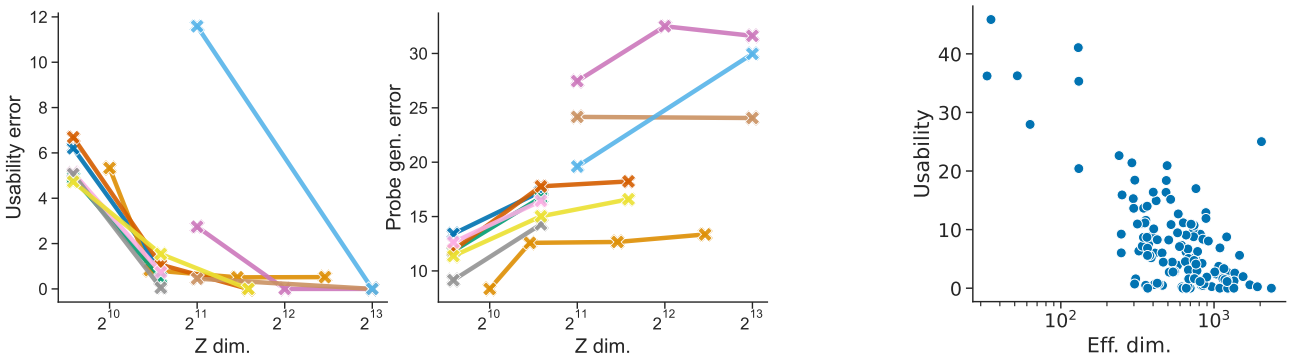
D.1 Dimensionality

Fig. 18 and Fig. 17a show that the dimensionality of the representation is a decisive hyperparameter for both the usability and the probe generalization error. Let us analyze this in more detail.

Increasing dimensionality improves usability. Fig. 17a shows that increasing dimensionality improves usability (decreases usability error). This is further supported by the controlled analysis plotted in Fig. 19a. The coefficient of log(dimensionality) for the controlled linear model is -3.9 (CA: pvalue=4e-9) for usability. The impact is also statistically significant for the global linear model. Although the ambient dimensionality is important, what really matters is actually the effective dimensionality of the representation as shown in Fig. 19b (CA: pvalue=6e-8).

The theory from Dubois et al. (2022) suggests why increasing (effective) dimensionality is necessary and sufficient for good usability. Namely, they prove that SSL clusters representations by the equivalence classes induced by the training augmentations. From those clusters, one can then linearly predict any downstream label that is invariant to the augmentations if and only if the effective dimensionality of the representation is at least the number of classes minus one. This is because predicting any downstream labels is equivalent to shattering the C clusters, which by standard statistical learning theory (Vapnik & Chervonenkis, 1971) is only possible by linear models iff $d = C - 1$. Intuitively, increasing the input dimension increases the capacity of a linear model.

Increasing dimensionality worsens probe generalization error. The SHAP+XGBoost analysis (Fig. 17a) and the controlled analysis (Fig. 19a) both show that increasing dimensionality leads to worse probe generalization error. In particular,



(a) Z dim. in the controlled setting

(b) Effective Z dim. vs usability

Figure 19: (a) Impact of Z dimensionality on usability and probe generalization error, when all other hyperparameters are kept the same. Each color shows a specific model and the effect that Z dimensionality has on that model. (b) Impact of the effective Z dimensionality (the rank of all the representations) on the usability error. Each point corresponds to a different model with different hyperparameters.

the coefficient of log(dimensionality) for the controlled linear model is 3.8 (CA: pvalue=2e-9) for probe generalization error. The impact is also statistically significant for the global linear model.

The negative effect that dimensionality error has on probe generalization can be understood in two different ways. First, by standard statistical learning theory, we expect a smaller dimensionality of the input data to lead to better generalization given that the model can overfit on fewer components. Second, due to the usability-probe generalization trade-off (Sec. 5.2.2) we expect dimensionality to have the opposite effect as it has on usability.

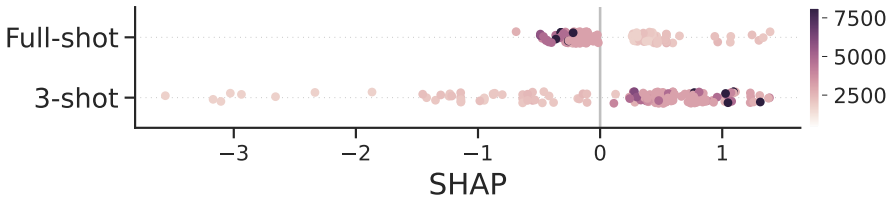


Figure 20: Z dimensionality has a significant impact on the performance in different settings. Every point corresponds to a model. The color shows the Z dimensionality. X-axis is the absolute SHAP value. Y-axis shows the performance in the full-shot ("Agg. Risk") and few-shot ("3-shot") setting.

Lower dimensional representations are better in few-shot settings. Given the important impact that dimensionality has on usability and probe generalization, we expect it to also have an important impact on the performance of the representations in different settings due to Sec. 5.2.1. In particular, we expect that lower dimensional representations will perform better in few-shot settings, while higher dimensional representations will perform better in full-shot settings. Fig. 20 shows that in the few-shot setting, using a low dimensionality can improve performance by up to 4 accuracy points, while it decreases full-shot performance by up to 1 accuracy point.

D.2 Data and Augmentations

Let us analyze the impact that the choice of augmentations has on each metric. One challenge is that there are many different augmentations and most models use the same ones, which makes it challenging to pin down the impact of a single augmentation. To avoid this issue, we focus on two specific hyperparameters that are related to augmentations. First, we consider the total number of augmentations used for training the model, which is coarser than the exact augmentations and thus easier to analyze. Second, we consider the number of views/multicrops (Caron et al., 2020) used to pretrain the model. The advantage of multicrops is that it is the only augmentations for which we have many models that only differ with respect to it.

In Sec. 5.3.2 we discuss the case of multicrops, here we focus on the total number of type of augmentations (e.g. rotation, flipping, cropping, ...)

Increasing the total number of augmentations likely improves usability. Fig. 17b suggests that increasing the number of augmentations might the usability of the representation. Using the global linear model for quantifying the importance of the log number of augmentations, we have that the coefficient of the log number of augmentations is -5.3 (CA: pvalue=4e-2). This high p-value compared to the effect of the number of views is likely due to the fact that increasing the number of augmentations does not monotonically decrease the number of equivalence classes because the augmentations are not comparable. For example, a model that uses only auto-augment and cropping would be counted as having only 2 augmentations but those are likely much stronger than using small x- and y-translations and rotations, which would be counted as 3 augmentations. We thus believe that the effect of increasing augmentation strength is similar to increasing the number of views, but that simply counting the number of augmentations is not an ideal way of quantifying the strength of augmentations.

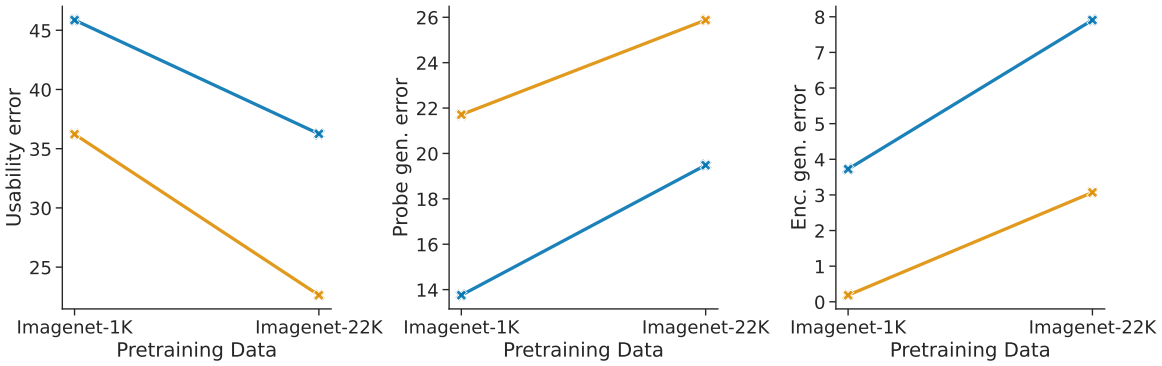


Figure 21: Effect of pretraining on ImageNet-22k on usability, probe generalization, and encoder generalization error. All other hyperparameters are kept the same. Each color shows a specific model.

Pretraining on ImageNet-22k worsens generalization. Fig. 17j shows that pretraining on ImageNet-22k worsens both the encoder and the probe generalization error. This can be seen also from the controlled setting in Fig. 21. This is interesting given that ImageNet-22k is a superset of the standard ImageNet-1k. This shows that pretraining on additional data can be detrimental to generalization.

D.3 Architecture

It is well known that using large non-linear projection heads helps (Bachman et al., 2019; Chen et al., 2020a;b), but it is not clear why it does work. To our knowledge there are four explanations that have been proposed in the literature for why using at least one non-linear head can help: (i) to avoid perfect invariance/alignment, which helps if the augmentations are stronger than desired (Chen et al., 2020a; Gupta et al., 2022; Appalaraju et al., 2020), (ii) to avoid dimensionality collapse (Jing et al., 2022), (iii) to be able to learn the optimal pseudo-label that should be predicted to ensure linearly predictability (Dubois et al., 2022), (iv) to avoid complete collapsing in non-contrastive learning (Chen et al., 2021a). All of those explanations suggest that adding a non-linear projection head would improve the usability of the representation.

Large projection heads improve usability. Fig. 18 shows that the size of the projection head is crucial for usability as expected (both the architecture and the number of parameters). Fig. 17e and Fig. 17f shows that using a large MLP projection head greatly improves usability. Quantitatively, we have that the global linear model predicts a coefficient of -8.6 ± 2.6 for using an MLP projection instead of no projection (GLA: p-value 1e-3) and a coefficient of -0.68 ± 0.28 for the log of the number of projection parameters (GLA: p-value 2e-2). The beneficial impact of using a larger projection head on usability is even more clear from the controlled setting seen in Fig. 11 (CA: p-value 9e-12).

This empirically support our hypotheses that a larger projection should improve usability as suggested by previous literature. This still does not explain which of the four previous explanations is (more) correct. As a partial answer to this question we consider the effect that projections heads have on effective dimensionality, and we have that using a linear projection head significantly improves effective dimensionality (GLA: pvalue=3e-9) but a non-linear projection head is not significantly different from the linear one. This suggests that Jing et al.'s (2022) hypothesis about dimensionality collapse explains some

of the performance gains but not all. Furthemore, we did not see any significant impact on alignment as suggested by (Gupta et al., 2022) or gains from using one-linear projection head as suggested by (Dubois et al., 2022). This shows that our understanding of the impact of non-linear projection heads is still lacking.

MLP projection improves probe generalization. Fig. 17e shows that using an MLP head is actually somewhat beneficial for **all** metrics. In particular, Fig. 11 shows that MLP projection heads also typically improve probe generalization (CA p-value 5e-3). This shows that using an MLP projection head is one effective way to overcome the usability-probe generalization tradeoff. The impact that a non-linear MLP projection head has on probe generalization cannot be predicted by the four previous hypotheses. This further suggests that we do not completely understand why large non-linear projection heads improve performance.

Fig. 18 shows that the architecture (family, number of parameters, and patch size) is really important for the probe generalization and approximation error.

Smaller patch sizes for ViTs is uniformly better. Fig. 17i shows that smaller patch sizes for ViT are uniformly better but is especially important for the approximation and usability error.

D.4 Objective

Let us analyze the impact that the choice of SSL objective has on each metric. One difficulty to do so is that there are many objectives and so (1) it is hard to analyze them simultaneously, and (2) there are only a few pretrained models for each objective. To avoid both of those problems we aggregate the SSL objectives into the 6 coarser clusters described in Appx. C.2 (transform, contrastive, clustering, siamese, generative, hierarchical).

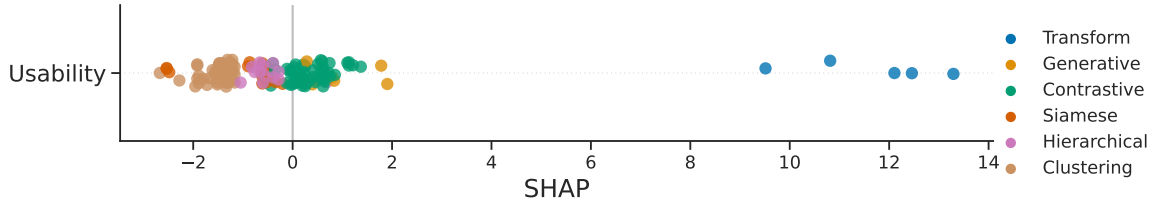


Figure 22: Effect measured by SHAP

Figure 23: SSL mode has an important impact on the usability error. (a) Average usability error for models of each SSL mode without considering potential confounders. (b) SHAP values of each model color coded by the SSL mode.

Objectives that are generative or predict the transformation worsen the usability. Fig. 17d shows that the SSL objective and the coarser SSL mode have an important impact on usability error. Fig. 22 shows more precisely the effect on usability. We see that the generative models and the ones that predict the transformation have much worse usability. The p-values as given by the global linear models are respectively 1e-4 and 1e-2.[5] In contrast, clustering objectives significantly improve usability.

Finer grain analysis of objectives. Fig. 24 shows the impact of the exact objective functions on each metric. To make sure that the results are meaningful, we only show objectives for which we have at least 7 models. We see that CLIP is particularly good for usability and full-shot risk, while MOCO is good in the few-shot regime. We also see that SimCLR a weak objective w.r.t. to few- and full-shot performance. This shows that the newer objective brings some meaningful improvement compared to SimCLR.

D.5 Optimization

Longer training improves usability and probe generalization. Fig. 17k suggests that increasing the number of epochs improves usability and probe generalization but might have a negative impact on encoder generalization. A similar trend can also be somewhat seen from the controlled setting in Fig. 25 for usability (CA p-value: 2e-3, coefficient: -1.37 ± 0.55) and to a lesser extent for probe generalization (CA coefficient: -0.58 ± 0.57, p-value: 0.3). We see that for the encoder

[5]The impact of having an objective that predicts the transformation is not as significant as what we would expect from Fig. 23 because it is highly correlated with the publication year which we have to control for.

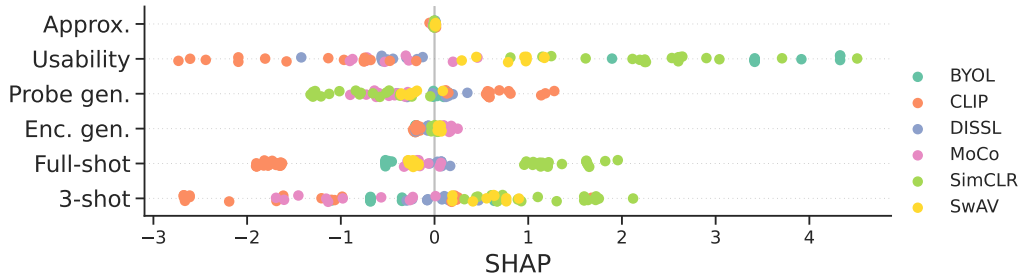


Figure 24: Effect of fine-grained objective functions on each risk component. We only show objectives for which there are at least 7 models, to avoid over interpreting the results. Every point corresponds to a model. The color shows the Z dimensionality. X-axis is the absolute SHAP value. The Y-axis shows the metric, either the risk component or the total risk in the full ("Agg. Risk') and few-shot regime ("3 shot").

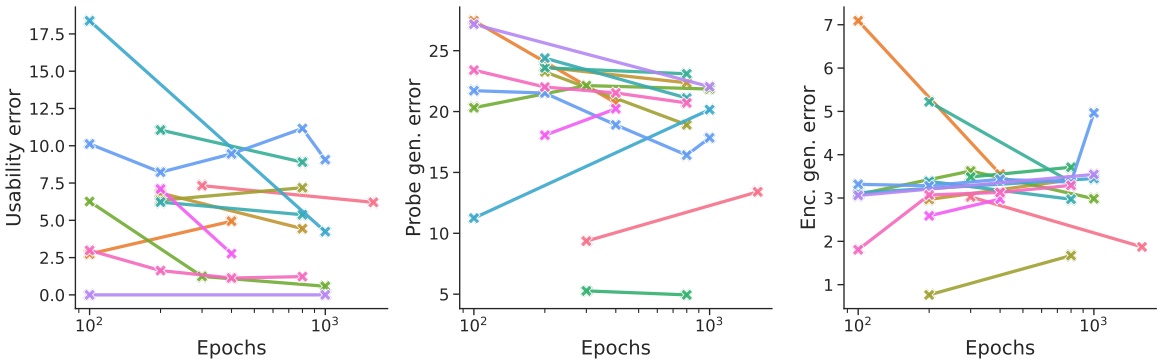


Figure 25: Effect of the projection head on usability, probe generalization, and encoder generalization error. All other hyperparameters are kept the same. Each color shows a specific model.

generalization, it is not very clear, for some models it improves, and for others, it makes it worse.

The improvements in usability and probe generalization can be partially understood from the fact that longer training with the proper SSL log loss with give rise to the collapse of equivalent representations (Dubois et al., 2022), which should improve downstream sample efficiency and linear predictability. The potential worsening or improvement of encoder generalization is likely due to the fact that at the beginning, training for longer allows you to better generalize but then the model starts overfitting given that you see multiple times the same examples.

Adam and AdamW improve probe generalization. Fig. 17l suggests that Adam and AdamW should be favored in both the full- and few-shot settings. Indeed, those optimizers seem to improve probe generalization.

Larger batch sizes can be beneficial for all components. Fig. 17m suggests that larger batch sizes can be beneficial but our global linear layer did not recognize the impact as being significant.

D.6 Other

The number of classes for clustering objectives can improve usability. Fig. 17n suggests that increasing the number of classes for clustering objectives (e.g. teacher's output in SwAV, DINO, or DISSL) can improve usability at the detriment of probe generalization but our GLA did not recognize the impact as significant. Both of those can be understood by Dubois et al.'s (2022) ISSL theory. First, fewer equivalences classes mean that you need to see fewer downstream samples. Second, if there are fewer teacher's classes than equivalence classes then the model might collapse examples that can differ in downstream labels, which will negatively impact usability.

E All raw results

E.1 Radar charts

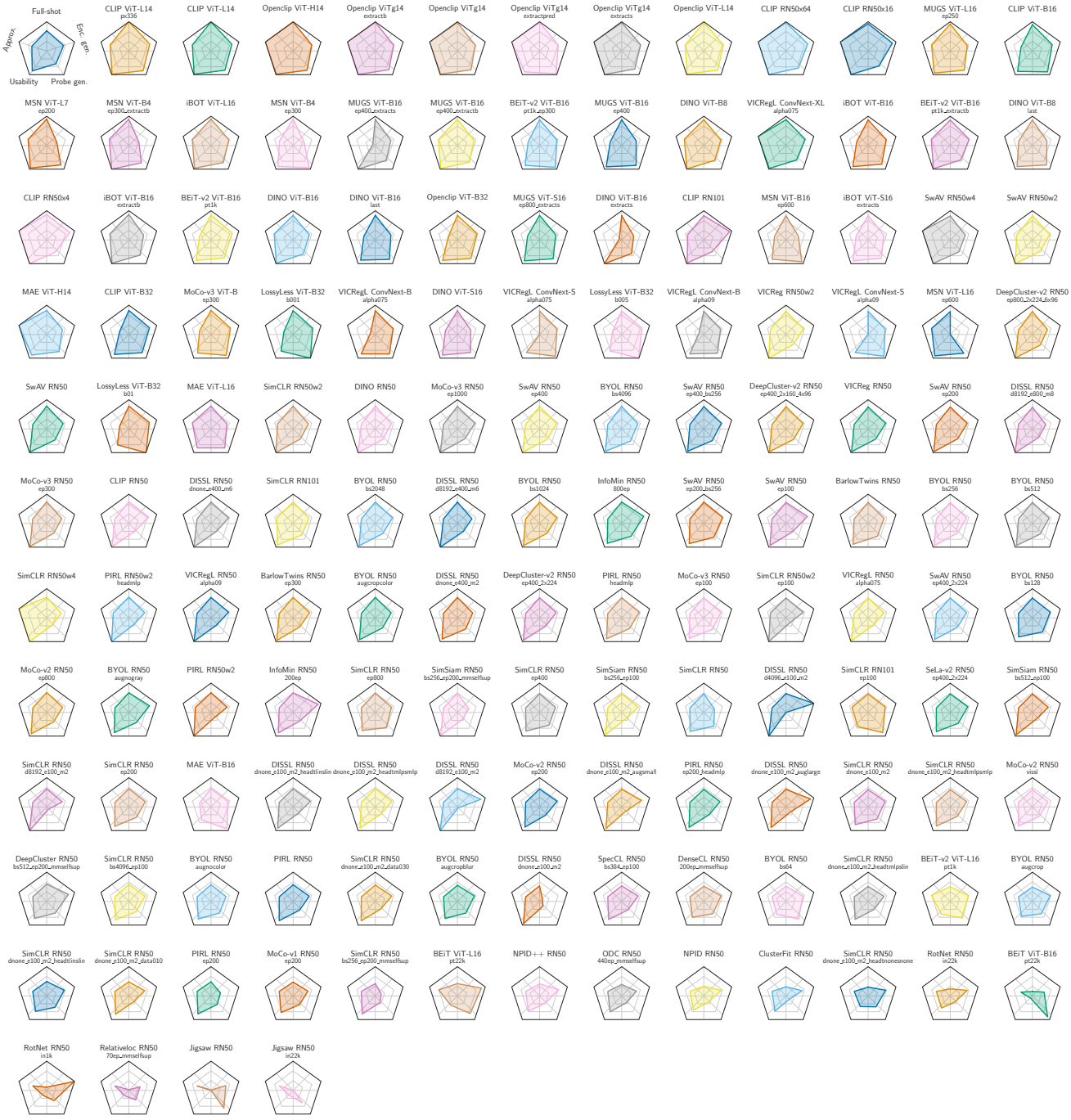


Figure 26: All risk components. Starting from the top, axes respectively show the standard linear probing ImageNet risk ("Agg. Risk"), encoder's generalization error ("Enc. Gen."), probe's generalization error ("Probe Gen."), representation usability error ("Usability"), and approximation error ("Approx."). Values are min-max scaled and substracted to 1 so that the worst model gets a 0 and the best gets a 1 (vertex). The top left plot shows the average over models, all other plots show a specific model described by its title (SSL objective and architecture) and subtitle (additional hyperparameters corresponding to "other" in Table 7). Colors are meaningless.

Fig. 26 shows the relative risk component of nearly every evaluated model. We do not show 2 models for which we did not find a supervised model with the same architecture, as we could not compute the approximation error for those models.

E.2 Table

The radar charts from Fig. 26 are useful to get a quick overview of each model but are not quantitative and do not allow comparison between risk components as each axes are normalized. Table 7 provides all the raw metrics.

				Risk Component				Aggregated Error				
Objective	**Arch.**	**Epochs**	**Other**	**Approx.**	**Usability**	**Probe gen.**	**Enc. gen.**	**100%**	**30 Shot**	**1%**	**5 Shot**	**3 Shot**
BEiT	ViT-B16	800	pt22k	1.00	41.06	10.53	4.61	57.19	94.35	89.67	94.35	95.92
	ViT-L16	800	pt22k	0.55	27.96	13.96	1.09	43.55	93.60	87.08	93.60	95.72
BEiT-v2	ViT-B16	300	pt1k_ep300	1.00	7.32	9.36	3.03	20.71	37.04	31.42	37.04	41.32
		1600	pt1k_extractb	0.54	0.84	17.31	2.54	21.23	41.93	34.93	41.93	47.53
			pt1k	1.00	6.21	13.41	1.87	22.48	41.53	35.42	41.53	46.43
	ViT-L16	1600	pt1k	0.55	21.40	15.03	2.78	39.77	72.85	63.25	72.85	77.95
BYOL	RN50	1000	augcropblur	0.85	12.90	20.26	3.13	37.14	71.59	63.67	71.59	75.81
			augcropcolor	0.85	3.74	21.98	3.46	30.02	60.70	52.50	60.70	66.12
			augcrop	0.85	17.00	19.54	2.86	40.25	75.64	67.86	75.64	79.66
			augnocolor	0.85	11.91	20.15	3.63	36.53	71.26	63.04	71.26	75.20
			augnogray	0.85	6.88	21.65	2.10	31.47	60.91	52.50	60.91	66.58
			bs1024	0.85	2.67	21.97	3.02	28.51	57.77	48.98	57.77	63.74
			bs128	0.85	9.22	17.70	2.86	30.63	59.80	51.73	59.80	65.09
			bs2048	0.85	2.72	21.82	3.07	28.45	57.82	49.05	57.82	63.64
			bs256	0.85	2.77	22.23	3.26	29.11	58.57	49.89	58.57	64.47
			bs4096	0.85	2.58	19.72	3.35	26.50	54.57	45.55	54.57	61.20
			bs512	0.85	3.34	21.31	3.19	28.68	57.53	49.22	57.53	63.31
			bs64	0.85	20.92	14.06	3.15	38.98	69.11	61.30	69.11	73.21
BarlowTwins	RN50	300	ep300	0.85	2.42	23.10	3.27	29.63	59.56	50.98	59.56	65.59
		1000		0.85	5.61	18.93	3.43	28.82	57.33	49.09	57.33	63.30
CLIP	RN101	32		0.71	1.60	20.33	0.57	23.21	51.35	41.62	51.35	58.25
	RN50	32		0.85	0.71	23.98	2.32	27.85	56.70	46.41	56.70	63.79
	RN50x16	32		0.00	0.62	16.62	1.06	18.30	41.21	32.64	41.21	48.25
	RN50x4	32		0.00	0.50	19.63	1.39	21.52	46.98	37.44	46.98	53.87
	RN50x64	32		0.00	0.49	14.49	1.73	16.72	35.74	28.16	35.74	42.28
	ViT-B16	32		1.00	6.30	10.51	2.28	20.08	40.65	32.79	40.65	46.88
	ViT-B32	32		1.13	7.53	13.13	2.11	23.90	47.55	39.03	47.55	53.62
	ViT-L14	32	px336_extractb	nan	nan	12.37	2.09	14.95	32.81	25.36	32.81	39.31
			px336_extractpredcls	nan	nan	11.52	2.14	14.95	30.61	24.51	30.61	36.44
			px336_extractpred	nan	nan	9.09	1.52	15.10	30.52	24.14	30.52	35.94
			px336_extracts	nan	nan	12.18	2.25	14.93	35.56	27.25	35.56	42.52
			px336	0.55	0.86	11.60	2.00	15.01	31.05	24.78	31.05	37.07
				0.55	0.77	12.12	2.02	15.46	32.08	25.53	32.08	37.96
ClusterFit	RN50	105		0.85	16.36	28.48	3.38	49.07	84.53	77.58	84.53	88.32
DINO	RN50	800		0.85	0.23	21.42	3.34	25.83	57.40	47.11	57.40	64.06
	ViT-B16	400	extracts	1.55	0.00	18.23	4.53	23.57	41.79	35.15	41.79	47.39
			last	1.00	6.69	11.91	3.51	23.10	37.44	32.55	37.44	41.68
				0.54	1.07	17.78	3.38	22.76	40.50	34.05	40.50	46.20
	ViT-B8	300	last	0.86	4.90	11.83	3.83	21.42	34.23	29.74	34.23	38.21
				0.51	0.49	16.75	3.13	20.88	36.78	30.66	36.78	41.74
	ViT-S16	800	extractb	nan	nan	10.44	4.10	25.11	44.17	37.22	44.17	50.39
			last	nan	nan	4.29	3.81	24.44	40.60	35.15	40.60	45.31
				0.96	6.00	13.48	4.16	24.60	46.43	39.13	46.43	52.87
	ViT-S8	800	last	nan	nan	4.45	3.82	21.79	34.26	29.57	34.26	38.05
DISSL	RN50	100	d4096_e100_m2	0.85	0.00	32.50	0.00	32.85	66.94	57.74	66.94	72.82
			d8192_e100_m2	0.85	0.00	31.62	1.30	33.58	66.41	57.16	66.41	72.34
			dnone_e100_m2_auglarge	0.85	6.01	26.74	0.99	34.59	70.10	60.75	70.10	75.77
			dnone_e100_m2_augsmall	0.85	3.91	27.56	2.25	34.57	69.29	59.76	69.29	75.23
			dnone_e100_m2_headtlinslin	0.85	4.45	25.52	3.02	33.84	68.67	58.96	68.67	74.55
			dnone_e100_m2_headtmlpsmlp	0.85	5.20	24.74	3.18	33.96	70.44	60.86	70.44	75.88
			dnone_e100_m2	0.85	2.74	27.46	7.09	38.14	68.82	59.30	68.82	74.63
		400	d8192_e400_m6	0.85	0.00	24.06	3.82	28.34	60.59	50.37	60.59	67.70
			d8192_e800_m8	0.85	0.00	23.42	4.12	28.00	61.12	50.86	61.12	68.26
			dnone_e400_m2	0.85	4.94	20.71	3.55	30.05	64.08	53.60	64.08	70.85
			dnone_e400_m6	0.85	0.45	24.17	2.91	28.38	64.15	53.24	64.15	71.58
DeepCluster	RN50	200	bs512_ep200_mmselfsup	0.85	12.67	20.27	1.72	35.51	71.36	62.02	71.36	76.80
DeepCluster-v2	RN50	400	ep400_2x160_4x96	0.85	1.61	21.44	3.09	26.99	56.57	47.37	56.57	62.77
			ep400_2x224	0.85	2.87	23.68	3.15	30.55	61.95	53.48	61.95	67.89
		800	ep800_2x224_6x96	0.85	0.27	21.30	3.63	26.05	55.37	45.39	55.37	62.51
DenseCL	RN50	200	200ep_mmselfsup	0.85	15.28	19.44	3.00	38.57	70.52	63.38	70.52	74.92
InfoMin	RN50	200	200ep	0.85	6.35	23.26	0.76	31.22	64.73	56.64	64.73	69.78
		800	800ep	0.85	7.19	18.91	1.67	28.62	55.97	49.43	55.97	60.02
Jigsaw	RN50	105	in22k	0.85	36.26	19.48	7.91	64.49	92.19	87.38	92.19	94.02
				0.85	45.86	13.76	3.72	64.19	94.43	90.80	94.43	95.85

Continued on next page

				Risk Component				Aggregated Error				
Objective	**Arch.**	**Epochs**	**Other**	**Approx.**	**Usability**	**Probe gen.**	**Enc. gen.**	**100%**	**30 Shot**	**1%**	**5 Shot**	**3 Shot**
LossyLess	ViT-B32	32	b001	1.13	13.65	7.43	2.38	24.59	46.72	38.44	46.72	52.79
			b005	1.13	13.99	7.53	2.21	24.86	47.04	38.92	47.04	53.18
			b01	1.13	14.93	7.55	2.18	25.80	47.83	39.47	47.83	53.90
MAE	ViT-B16	1600		1.00	20.42	9.46	3.12	34.00	72.67	62.68	72.67	78.27
	ViT-H14	1600		0.00	6.03	14.51	3.47	24.01	64.29	49.80	64.29	73.08
	ViT-L16	1600		0.55	9.23	12.45	3.42	25.65	61.72	49.59	61.72	69.43
MSN	ViT-B16	600	ep600	1.00	8.78	9.37	4.42	23.57	33.60	30.23	33.60	36.49
	ViT-B4	300	ep300_extractb	0.51	0.05	14.29	5.06	19.91	30.80	26.06	30.80	35.15
			ep300_extracts	nan	nan	14.20	5.58	20.67	34.57	28.91	34.57	39.62
			ep300	0.86	5.07	9.15	4.83	19.91	27.69	24.86	27.69	30.70
	ViT-L16	300	ep600	0.55	4.54	12.60	7.96	25.66	33.99	30.01	33.99	37.30
	ViT-L7	200	ep200_extractb	nan	nan	14.46	4.93	19.99	29.08	25.60	29.08	32.17
			ep200_extracts	nan	nan	14.34	6.59	21.79	28.29	25.66	28.29	30.89
			ep200	0.55	2.48	11.95	5.11	20.09	27.63	25.07	27.63	30.16
	ViT-S16	800	ep800	nan	nan	5.07	3.29	23.89	36.35	32.51	36.35	39.64
MUGS	ViT-B16	400	ep400_extractb	0.54	1.54	15.02	3.27	20.37	30.83	27.32	30.83	33.82
			ep400_extracts	1.55	0.00	16.59	3.52	20.70	35.13	29.79	35.13	39.91
			ep400	1.00	4.73	11.37	3.81	20.91	30.34	27.03	30.34	33.24
	ViT-L16	400	ep250_extractb	nan	nan	13.92	3.29	19.12	29.28	25.72	29.28	31.24
			ep250_extracts	nan	nan	10.17	3.65	19.69	30.89	27.01	30.89	33.87
			ep250	0.55	3.11	12.31	3.14	19.12	29.22	26.02	29.22	31.49
	ViT-S16	100	ep100	nan	nan	5.11	3.07	25.83	43.86	38.22	43.86	48.68
		300	ep300	nan	nan	5.27	3.49	23.37	39.33	34.22	39.33	43.88
		800	ep800_extracts	0.96	5.59	12.76	3.37	22.69	44.12	37.13	44.12	50.36
			ep800	nan	nan	4.94	3.71	23.01	37.91	33.40	37.91	42.11
MoCo-v1	RN50	200	ep200	0.85	13.64	23.16	3.74	41.39	79.35	70.51	79.35	83.94
MoCo-v2	RN50	200	ep200	0.85	6.83	23.64	2.97	34.28	68.86	61.07	68.86	74.17
			vissl	0.85	8.68	22.44	3.48	35.45	72.63	63.90	72.63	77.51
		800	ep800	0.85	4.44	22.36	3.42	31.07	60.46	53.30	60.46	64.75
MoCo-v3	RN50	100	ep100	0.85	6.26	20.32	3.08	30.51	63.99	54.58	63.99	69.19
		300	ep300	0.85	1.24	22.13	3.62	27.84	56.19	47.16	56.19	62.24
		1000	ep1000	0.85	0.58	21.85	2.98	26.26	53.00	44.46	53.00	59.58
	ViT-B	300	ep300	1.00	10.12	10.38	2.36	23.86	41.37	35.83	41.37	45.87
	ViT-S	300	ep300	nan	nan	5.42	3.85	27.94	46.05	40.41	46.05	50.40
NPID	RN50	200		0.85	18.44	26.01	3.02	48.32	86.15	78.70	86.15	89.85
NPID++	RN50	800		0.85	16.39	24.75	2.46	44.45	83.04	74.72	83.04	87.23
ODC	RN50	440	440ep_mmselfsup	0.85	15.16	25.51	3.78	45.29	80.63	73.01	80.63	84.24
Openclip	ViT-B32	32		1.13	6.51	12.93	2.34	22.91	43.83	35.83	43.83	50.02
	ViT-H14	32	extractb	nan	nan	12.27	2.94	15.73	30.52	24.48	30.52	36.09
			extractpred	nan	nan	9.36	2.45	15.73	29.24	23.54	29.24	34.55
			extracts	nan	nan	12.91	2.72	16.10	36.39	27.57	36.39	43.73
				0.00	0.80	12.13	2.66	15.59	30.63	24.23	30.63	36.30
	ViT-L14	32		0.55	1.43	12.26	2.41	16.65	32.16	25.81	32.16	37.35
	ViTg14	32	extractb	0.00	0.51	12.66	2.73	15.91	32.67	25.51	32.67	38.55
			extractpred	0.00	5.33	8.35	2.66	16.34	29.84	24.12	29.84	35.14
			extracts	0.00	0.52	13.36	2.52	16.40	34.59	26.94	34.59	40.90
				0.00	0.83	12.58	2.88	16.29	30.87	24.61	30.87	36.17
PIRL	RN50	200	ep200_headmlp	0.85	6.22	24.39	3.38	34.84	72.14	63.33	72.14	76.81
			ep200	0.85	11.06	23.60	5.22	40.72	78.59	69.32	78.59	83.22
		800	headmlp	0.85	5.37	21.11	2.97	30.30	62.93	55.12	62.93	67.61
				0.85	8.90	23.10	3.37	36.22	74.20	64.58	74.20	79.69
	RN50w2	400	headmlp	0.74	0.00	25.43	3.42	29.50	58.43	51.75	58.43	62.72
				0.74	0.27	27.21	3.35	31.58	68.32	57.96	68.32	73.60
Relativeloc	RN50	70	70ep_mmselfsup	0.85	35.32	22.59	4.75	63.51	94.14	90.05	94.14	95.61
RotNet	RN50	105	in1k	0.85	36.22	21.71	0.19	58.96	92.13	87.09	92.13	94.03
			in22k	0.85	22.63	25.88	3.07	52.42	88.48	82.62	88.48	91.17
SeLa-v2	RN50	400	ep400_2x224	0.85	8.75	20.99	2.93	33.51	62.60	55.56	62.60	67.36
SimCLR	RN101	100	ep100	0.71	18.38	11.25	3.12	33.46	68.80	60.14	68.80	74.06
		1000		0.71	4.24	20.16	3.45	28.56	60.89	51.17	60.89	67.13
	RN50	100	bs4096_ep100	0.85	10.12	21.71	3.32	36.00	72.82	63.95	72.82	77.94
			d8192_e100_m2	0.85	0.00	29.97	3.49	33.92	71.93	62.36	71.93	77.40
			dnone_e100_m2_headtlinslin	0.85	15.92	20.36	2.92	40.03	75.32	67.22	75.32	79.96
			dnone_e100_m2_headtmlpslin	0.85	11.14	24.06	3.43	39.47	74.33	66.02	74.33	78.81
			dnone_e100_m2_headtmlpsmlp	0.85	8.69	22.39	3.25	35.18	73.76	64.33	73.76	79.14
			dnone_e100_m2_headtnonesnone	0.85	25.02	20.91	2.94	49.71	77.20	70.38	77.20	81.31
			dnone_e100_m2	0.85	11.59	19.60	3.12	35.16	74.16	64.37	74.16	79.44
		200	bs256_ep200_mmselfsup	0.85	10.96	25.60	6.32	43.72	76.15	67.48	76.15	81.12
			ep200	0.85	8.23	21.52	3.28	33.87	70.62	61.39	70.62	76.19
		300	dnone_e100_m2_data030	0.85	8.27	24.49	3.30	36.90	74.81	65.71	74.81	80.12
		400	ep400	0.85	9.46	18.90	3.43	32.64	69.00	59.24	69.00	74.35
		800	ep800	0.85	11.16	16.42	3.37	31.79	67.28	57.91	67.28	73.00
		1000	dnone_e100_m2_data010	0.85	10.72	26.01	3.40	40.97	76.67	68.19	76.67	81.12
				0.85	9.07	17.84	4.97	32.72	66.67	57.26	66.67	72.77
	RN50w2	100	ep100	0.74	0.00	27.17	3.06	30.71	66.81	56.99	66.81	72.53
		1000		0.74	0.00	22.04	3.54	26.06	58.57	48.58	58.57	65.04

Continued on next page

				Risk Component				Aggregated Error				
Objective	**Arch.**	**Epochs**	**Other**	**Approx.**	**Usability**	**Probe gen.**	**Enc. gen.**	**100%**	**30 Shot**	**1%**	**5 Shot**	**3 Shot**
	RN50w4	1000		0.00	0.46	24.81	3.86	29.14	61.76	52.21	61.76	68.13
SimSiam	RN50	100	bs256_ep100	0.85	2.80	26.00	3.23	32.87	68.85	59.77	68.85	74.77
			bs512_ep100	0.85	2.87	26.08	3.40	33.19	69.07	59.56	69.07	74.87
		200	bs256_ep200_mmselfsup	0.85	1.59	25.17	4.73	32.32	65.91	56.45	65.91	72.10
SpecCL	RN50	100	bs384_ep100	0.85	10.86	23.38	3.25	38.34	73.91	65.82	73.91	78.39
SwAV	RN50	100	ep100	0.85	2.98	23.40	1.81	29.04	62.09	52.83	62.09	68.61
		200	ep200_bs256	0.85	7.10	18.06	2.59	28.59	60.69	51.45	60.69	67.14
			ep200	0.85	1.63	22.02	3.07	27.56	60.27	50.36	60.27	66.78
		400	ep400_2x224	0.85	4.11	22.49	3.30	30.75	62.22	53.41	62.22	67.97
			ep400_bs256	0.85	2.77	20.23	2.98	26.82	59.22	48.98	59.22	66.00
			ep400	0.85	1.13	21.53	3.13	26.63	58.80	48.99	58.80	65.83
		800		0.85	1.23	20.71	3.29	26.07	57.91	47.63	57.91	64.89
	RN50w2	400		0.74	0.00	20.70	3.04	23.98	56.33	45.79	56.33	64.08
	RN50w4	400		0.00	0.24	19.85	3.67	23.76	54.70	43.80	54.70	63.11
VICReg	RN50	1000		0.85	1.98	21.84	2.94	27.60	56.36	47.47	56.36	62.47
	RN50w2	1000		0.74	0.00	22.10	2.97	25.33	47.52	40.24	47.52	53.17
VICRegL	ConvNext-B	400	alpha075	1.41	8.36	11.85	2.88	24.49	41.08	36.91	41.08	44.92
			alpha09	1.41	8.05	12.59	3.20	25.24	39.78	35.66	39.78	43.71
	ConvNext-S	400	alpha075	1.69	10.94	9.43	3.05	25.11	41.38	37.47	41.38	44.96
			alpha09	1.69	10.92	9.41	3.14	25.17	41.05	36.98	41.05	44.82
	ConvNext-XL	150	alpha075	0.00	1.42	17.26	2.44	21.12	44.11	38.05	44.11	48.44
	RN50	300	alpha075	0.85	1.46	24.98	3.41	30.69	60.99	52.38	60.99	66.90
			alpha09	0.85	1.14	24.80	2.93	29.71	60.25	51.26	60.25	65.99
iBOT	ViT-B16	400	extractb	0.54	0.73	16.44	3.70	21.42	37.60	31.86	37.60	42.77
				1.00	5.12	12.63	2.77	21.51	34.60	30.14	34.60	38.08
	ViT-L16	250	extractb	nan	nan	16.05	2.66	19.57	34.54	28.96	34.54	39.05
			extracts	nan	nan	15.08	3.37	20.46	31.75	27.49	31.75	35.19
				0.55	1.89	14.57	2.90	19.91	31.72	27.38	31.72	35.51
	ViT-S16	800	extracts	0.96	5.58	13.24	3.51	23.29	45.35	37.91	45.35	51.57
				nan	nan	4.51	2.74	23.13	39.42	34.03	39.42	43.71

F Secondary results

F.1 Validating the metrics

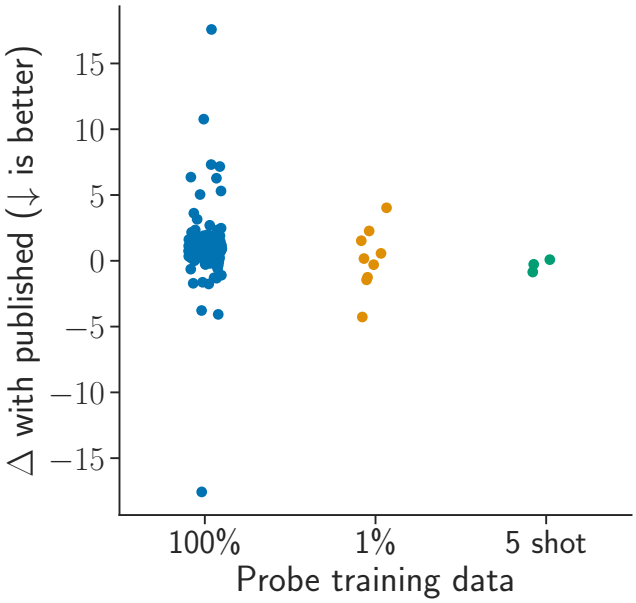


Figure 27: The difference between the standard metrics we found and the published values (when available). Negative means that the models performed worst than previously stated.

A secondary contribution of our paper is to evaluate 169 pretrained models, in a controlled and fair fashion. Appx. F.1 validates the metrics we computed by comparing it to previously published results (when available). We see that the values we found were generally close to the published results. But for the 100% the values can sometimes be very different. As a reminder (see Appx. C.3) the two main difference in this setting is that (i) we do not apply data augmentations when training the probe as it is more realistic and computationally efficient; and (ii) we perform extensive hyperparameter tuning on a validation set. The fact that we do not use data augmentations to train the probe, is likely why the values we found are worst (1.13 ± 3.01) than the published ones (p-value=1e-4 with a paired t-test). Our extensive hyperparameter tuning can likely explain why for some models we improve the probing results. For 1% and 5-shot, previous works also do not use data augmentations and should thus be more similar to our results. Indeed, we have that the respective aggregated values are 0.14 ± 2.39 and -0.34 ± 0.48 neither of which are statistically significant.

Table 8 shows in more detail the values that are more than 3 accuracy points further than published results. We see that many of the highly positive values are older SSL models that predict the transformation (Jigsaw, RotNet, Relativeloc, RotNet) and are thus not invariant to data augmentations. We hypothesize that given that they are not invariant to the augmentations, training the probe with augmentations performs much better for them, which is why the differences are large. For the case of MUGS, we note that they do not specify the evaluation pipeline of their models. In particular, we do not know which block of the ViT they used as the representation. Table 8 suggests that we likely chose a different feature as the performance of the model is worst on 1% but better on 100%.

F.2 Alternative decomposition

In Appx. A.2 we have seen that our decomposition is not unique. There are two alternative decompositions, but only one of which would be useful for understanding the effect of representation learning. This decomposition would essentially switch the order of the two generalization errors and thus keep the same interpretation as our decomposition. As previously discussed, the estimates for this decomposition would likely be worse than for our decompositions. In the following, we compute those (worse) estimates of the generalization errors and compare them with the ones in the paper. The goal is to make sure that alternative decompositions and estimators do not change the main conclusions from our paper.

Fig. 28 shows that despite being different components and different estimators, the estimated probe and encoder generaliza-

Table 8: Models for which the evaluated metrics are more than 3 accuracy points further than published results.

Objective	Arch.	Epochs	Other	100%	1%	5 Shot
BYOL	RN50	1000	augnocolor	-4.07		
DISSL	RN50	100	dnone_e100_m2	5.04		
DeepCluster	RN50	200	bs512_ep200_mmselfsup	-17.57		
Jigsaw	RN50	105	in22k	17.58		
				10.77		
MSN	ViT-B16	600	ep600		-4.27	
	ViT-L16	300	ep600	6.36		
MUGS	ViT-S16	800	ep800_extracts	-1.71	4.03	
PIRL	RN50	200	ep200	3.62		
Relativeloc	RN50	70	70ep_mmselfsup	3.16		
RotNet	RN50	105	in1k	7.16		
			in22k	7.31		
SimCLR	RN101	100	ep100	-3.78		
	RN50	200	bs256_ep200_mmselfsup	6.28		
SpecCL	RN50	100	bs384_ep100	5.31		

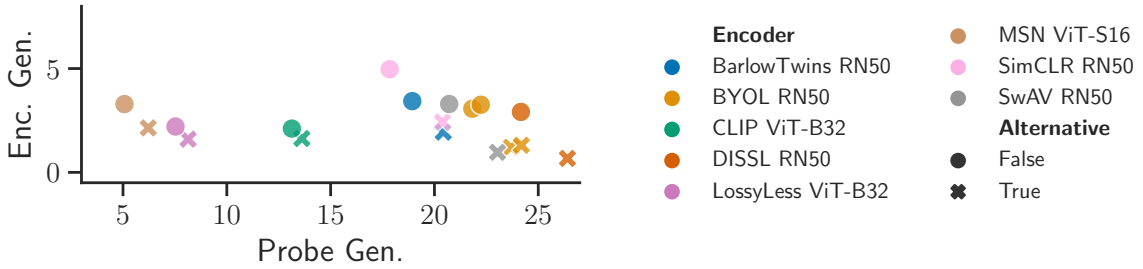


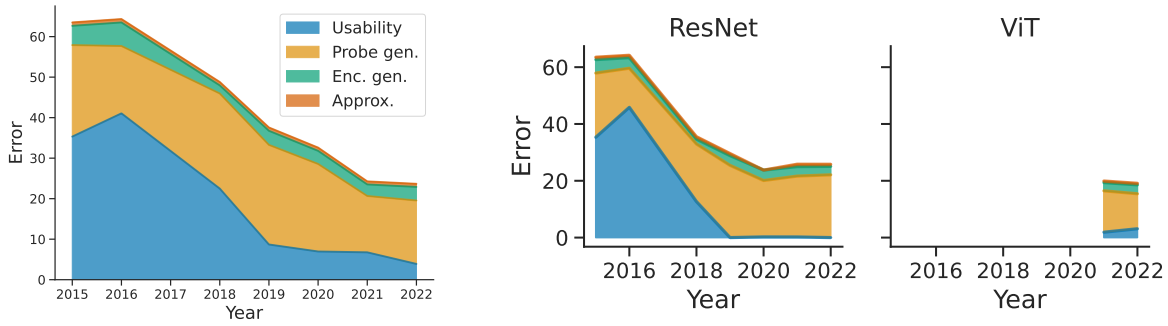
Figure 28: Our risk decomposition and the alternative Eq. (5), which switches the order of generalization errors, give similar results. Different colors show different encoder while the shape shows whether the generalization errors correspond to the alternative (cross) or our main decomposition (circles). We only show the generalizations as the other components are exactly the same. As discussed in Appx. A.2 the alternative risk components are likely worst estimates. Axes are on the same scale.

tion of the alternative and main decompositions are highly related. This is reassuring as it suggests that using a different decomposition would not change our interpretation of the results (encoder generalization still seems small in absolute terms and the relative ordering of models seems similar). Note that the plot is rectangular as the axes are on the same scale but encoder generalization is smaller than probe generalization.

F.3 Trends over time

In Fig. 4, we saw how risk components have been changing over time for the best ImageNet-pretrained models published in that year. Fig. 29a shows instead the average over all models published in that year (including those trained on the ImageNet-22K, LAION and CLIP dataset). We see that the global trend is essentially the same: usability has been driving improvements but probe generalization is now what matters.

We can also consider more fine-grained trends by looking at how risk components have been changing for a type of self-supervised learning method (Fig. 30) or encoder architecture (Fig. 29b). Fig. 30 shows that much of the improvements



(a) Average per year

(b) Best per year and neural family

Figure 29: Evolution of risk components over time. Lower is better. (a) the risk components are averaged over all models published in a given year; (b) the risk components are the best over models published in a given year and for a specific family of encoder's architecture (ResNet and ViT).

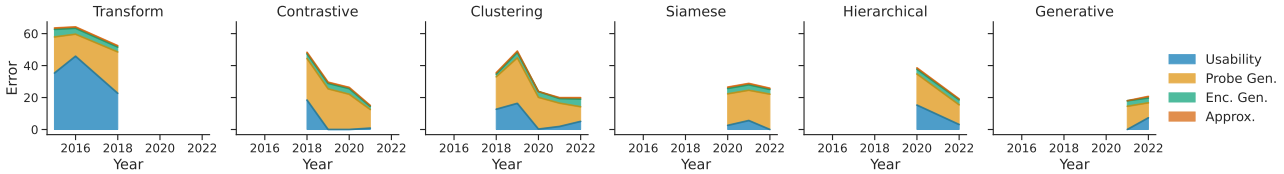


Figure 30: Evolution of average risk components over time. For a specific SSL mode.

in usability have been achieved under the contrastive learning framework. Using Fig. 29b we also see that the more recent improvements in probe generalization have mostly come from the clustering paradigm using ViTs.

F.4 Scaling laws

Table 9: Our scaling law predicts well performance. Numbers are R^2 on test data."Std" is a standard scaling law fitted on all encoders. "e=fam." fits separately ViT's and ResNet's, while "e=arch." and " e=obj." fit separate laws for each architecture and SSL objectives. Columns show held-out test sets. "IID" test on 3/5 settings for each encoder."Cntr", "Enc.", "2022", respectively tests on all encoders that are contrastive, ViTs, from last year. Missing values mean scaling laws cannot predict this test set.

		test set			
scaling law	param.	i.i.d.	2022	cntr.	ViT
Std	5	0.31	-0.12	0.46	-0.98
e=family	11	0.60	0.65	0.72	
e=arch.	41	0.63	0.66		
e=obj.	86	0.82			
Ours	2	**0.94**	**0.91**	**0.96**	**0.84**

We propose the following scaling law based on our decomposition:

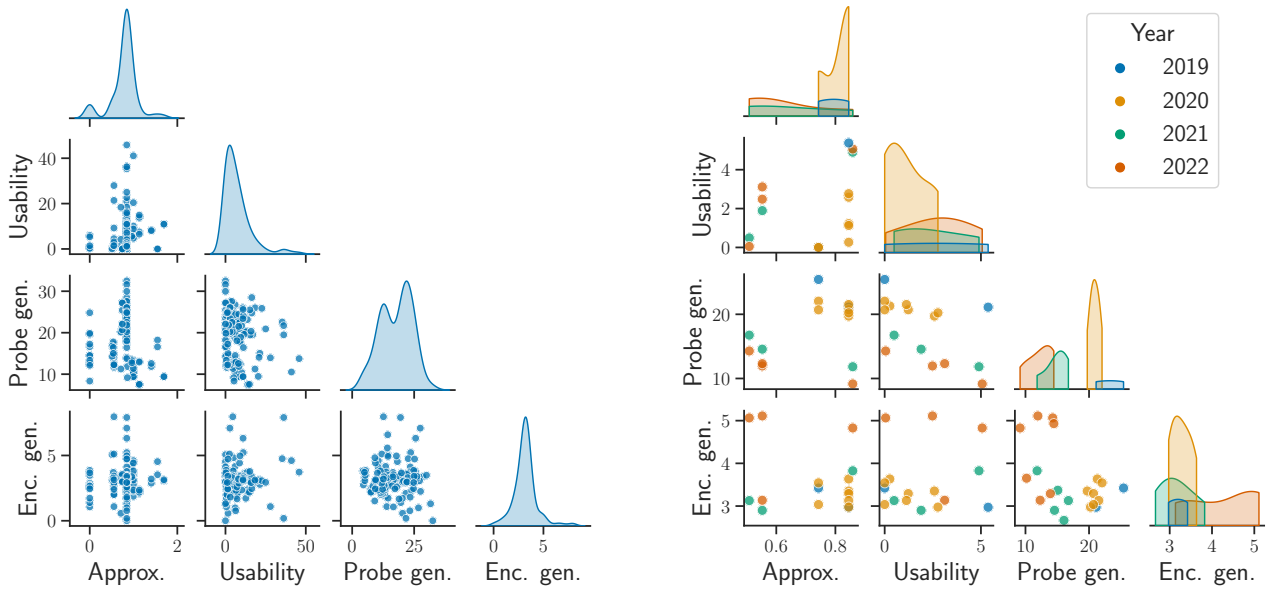
$$\mathrm{R}_{U,S}(n) \approx \mathcal{E}_{\text{app}} + \mathcal{E}_{\phi\text{gen}} + (1 - W)\mathcal{E}_{\text{use}} + (W\mathcal{E}_{\text{use}} + \mathcal{E}_{f\text{gen}})\left(\frac{N}{n}\right)^{\alpha} \tag{25}$$

where $\mathcal{E}_{\text{app}}, \mathcal{E}_{\phi\text{gen}}, \mathcal{E}_{\text{use}}, \mathcal{E}_{f\text{gen}}$ are respectively the risk components for the approximation, encoder generalzation, usability, and probe generalization. n is the number of samples used to train the probe, N is the number of samples used to estimate the decomposition, and α, W are fitted parameters quantifying sample efficiency and $\mathcal{E}_{\text{use}}$'s dependence on n.

Fig. 5b shows that Eq. (25) fits all results very well ($R^2 = 0.94$, $\alpha = 0.15$, $w = 0.51$). Table 9 shows that it predicts better the performance of held-out models compared to standard neural scaling laws (Kaplan et al., 2020; Rosenfeld et al., 2020) of the form $\mathrm{R}_{U,S}(n, p, e) \approx I_e + \frac{C_e}{n^{\alpha_e}} + \frac{K}{p^{\beta}}$ where p is the number of probe's parameters, e is a set of encoders for which we train the same scaling law (e.g. those with the same architecture), and $\{I_e\}_e, \{C_e\}_e, \{\alpha_e\}_e, K, \beta$ are fitted.

F.5 Trade-offs

As discussed in Appx. A.5 the standard approximation-estimation tradeoff implies three representation learning tradeoffs. Let us look at all possible tradeoffs empirically.



(a) All models

(b) Best 15% per year

Figure 31: All potential tradeoffs between our risk components when considering (a) all models pretrained on ImageNet. (b) the best 10% of model for recent years (since 2019).

Fig. 31 shows all possible pairwise tradeoffs between the risk components. We see that when considering all models in aggregation there seem to be no tradeoffs (Fig. 31a). Fig. 31b instead shows the best performing models for each (recent) year. Although there are not many points, we can see the usability-probe gen tradeoff and a glimpse of the approximation-probe gen tradeoff. But there does not seem to be any sign of any approximation-encoder generalization tradeoff. The fact that there seems to be a tradeoff for the probe but not the encoder, might be related to the fact that over-parametrized models seem to not follow the standard approximation-estimation tradeoff (Belkin et al., 2019; Yang et al., 2020; Nakkiran et al., 2020; Dar et al., 2021; Neal et al., 2018). This over-parametrization could potentially explain why the encoder generalization is smaller than the probe generalization. That being said we see in Fig. 31b that the approximation error is really small and so tradeoffs that depend on it are likely not important for practical SSL.

We emphasize that the tradeoff curve given by the top performing models (Fig. 6 and Fig. 31b) does not correspond to modifying a single hyperparameter on the best performing model, but those are instead models trained with different SSL objective, architectures, epochs, and many other hyperparameters. For example, the best-performing models for 2022 (in red in Fig. 6) include `msn_vitb4_ep300`, `msn_vitl7_ep200`, `mugs_vitl16_ep250`.

We have seen the U-P tradeoff for encoders that are pretrained with SSL. Our risk components and their tradeoffs are nevertheless not specific to SSL. A natural question is thus whether we see the same tradeoff for other representations. Fig. 32 provides evidence of such tradeoff more generally, by considering representations coming from untrained encoders.

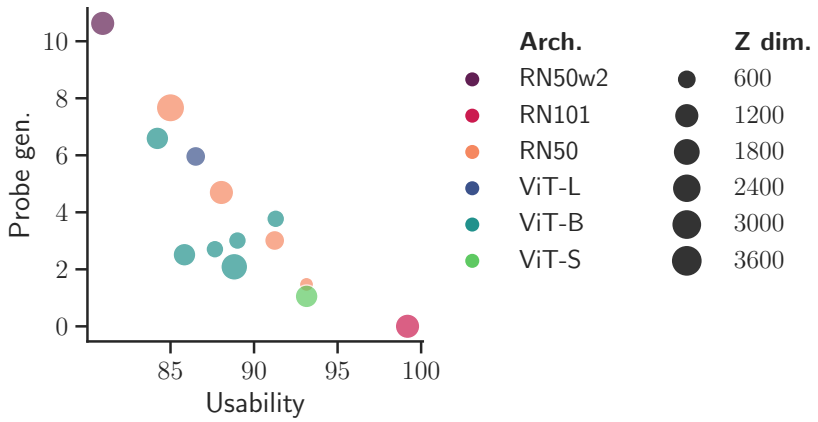


Figure 32: The tradeoff between probe generalization and usability also holds when the encoder is untrained. Each point shows the probe generalization (y-axis) and usability (x-axis) for representations that come from a different randomly initialized encoder.

F.6 Uniformity, alignment, and effective dimensionality

Many previous works have proposed different simple statistics to measure the quality of SSL representations. Three very common such statistics that are easy to compute are (we give code in PyTorch):

Effective dimensionality Dubois et al. (2022) recently proved that the effective dimensionality, i.e., the dimension of the space spanned by the representation's support, is a key property to ensure that downstream tasks with a few classes can be performed. The requirement for large effective dimensionality was also indirectly suggested by theoretical arguments of (Saunshi et al., 2022; HaoChen et al., 2021). For a fixed ambient dimensionality, the dimensionality collapse literature (Hua et al., 2021; Jing et al., 2022) also suggested that small effective dimensionality can be an issue. To compute the effective dimensionality we simply compute the rank (under some small tolerance) of Pearson's correlation coefficient matrix of the represented training set as follows:
`torch.linalg.matrix_rank(Z.T.corrcoef(), atol=1e-4, rtol=0.01).`

Uniformity Wang & Isola (2020) and follow-ups, e.g. (Wang & Liu, 2021), show that contrastive learning forces representations to be approximately uniformly distributed on a hypersphere, and they hypothesize based on empirical results that this is a desired property. But more recent theories (Dubois et al., 2022; Wang et al., 2022) suggest the opposite. We test the usefulness of uniformity using Wang & Isola (2020) original estimator:
`torch.pdist(F.normalize(Z, dim=-1), p=2).pow(2).mul(-2).exp().mean().log().`

Alignment Countless works (Ericsson et al., 2021; Dubois et al., 2022; Foster et al., 2021; Mitrovic et al., 2021; Dubois et al., 2021; Ruan et al., 2022; Foster et al., 2021; Miao et al., 2022) have proven or hypothesized that good encoders should be invariant to data augmentations. Although perfect invariance will not be achieved, it is natural to hypothesize that good encoders will map equivalent/augmented examples close together. Such property is called alignment (Wang & Isola, 2020), and can for example be quantified using the distance between augmented samples $z1, z2$:
`(z1 - z2).norm(dim=-1).pow(2).mean().`

In the following, we tested how well each of the previous statistics can predict the performance of a downstream model for the case of Resnet50s. Note that both uniformity and alignment were proposed by Wang & Isola (2020) in the case where the downstream representations are normalized before being probed. This is not the standard probing regime and we found that normalizing representations decreases downstream performance by 0.44 ± 0.28. We nevertheless compared the statistics to the performance in both the normalized and the unnormalized regimes for a subset of the models (19 models from VISSL) to compare in a setting more similar to (Wang & Isola, 2020). Fig. 33 shows qualitatively all the results. For quantitative results, we evaluated the model $\log(\text{agg_risk}) = \delta + \alpha \log(\text{eff_dim}) + \beta \text{uniformity} + \gamma \text{alignment}$ to test how

well each statistics (conditionally) correlates with the downstream performance.[6] The fitted model is

$$\text{agg_risk} = 93 - 9.5 \cdot \log(\text{eff_dim}) - 0.51 \cdot \text{uniformity} + 4.4 \cdot \text{alignment}, \tag{26}$$

it achieves an R^2 of 0.58.

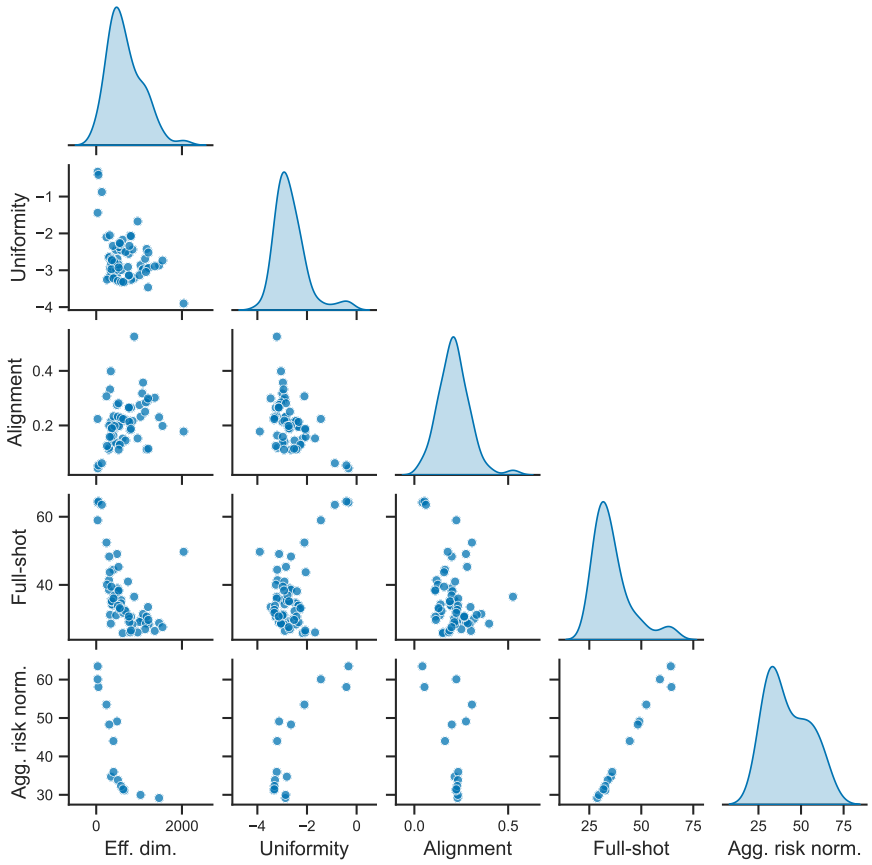


Figure 33: Relation between various statistics ("Uniformity", "Alignment", "Eff. Dim.") and probing performance from normalized ("Agg. Risk Norm.") and normalized representations ("Agg. Risk") for the case of ResNet50s. For normalized representations, we only evaluated the models from `VISSL`.

Effective dimensionality correlates with performance. Fig. 33 shows that higher effective dimensionality seems to improve downstream performance. Quantitatively, the effective dimensionality is statistically significant with a p-value of 4e-11 and the simple model Eq. (26) suggests that increasing the effective dimensionality by a factor of 3 improves the accuracy by 10 percentage points.

Uniformity is not predictive of performance. Looking at Fig. 33 it seems that uniformity is correlated with performance. But the quantitative results show that the p-value is 0.74, so the (conditional) correlation is not statistically significant. The difference between the quantitative and qualitative results comes from the fact that the estimated uniformity is actually highly correlated with effective dimensionality. For example, if we removed the effective dimensionality from Eq. (26) the coefficient of uniformity would increase to 6, and the p-value decrease to 0.001 (R^2 is only 0.19). This shows that although the estimator of uniformity does correlate with performance (as experimentally shown by (Wang & Isola, 2020)) it is only because it correlates with effective dimensionality. Beyond this, uniformity is not predictive of performance, which supports Dubois et al.'s (2022) theory.

[6]This was the best model for predicting the performance using a linear combination of effective dimensionality, uniformity, and alignment with potential log processing

Alignment does not correlate with performance. Fig. 33 shows that alignment does not seem to be correlated with (normalized or normalized) performance. This is further supported quantitatively by the fact that its impact is not statistically significant (p-value of 0.59), This is surprising given that previous theories and experiments have shown that alignment does predict performance. We do not have a good explanation of why this is the case but note that alignment for examples of the same class (e.g. alignment of 2 random dogs rather than the same dog with different augmentations) is highly correlated with performance (coefficient of 40.7 and p-value of 6e-7).