# $\alpha$-ReQ : Assessing representation quality by measuring eigenspectrum decay

**Kumar Krishna Agrawal**[†]
UC Berkeley
CA, USA

**Arnab Kumar Mondal**[†]
Mila & McGill University
Montréal, QC, Canada

**Arna Ghosh**[†]
Mila & McGill University
Montréal, QC, Canada

**Blake A. Richards**
Mila, Montreal Neurological Institute & McGill University
Montréal, QC, Canada
Learning in Machines and Brains Program, CIFAR
Toronto, ON, Canada

## Abstract

Self-Supervised Learning (SSL) with large-scale unlabelled datasets enables learning useful representations for multiple downstream tasks. However, efficiently assessing the quality of such representations poses nontrivial challenges. Existing approaches train linear probes (with frozen features) to evaluate performance on a given task. This is expensive both *computationally*, since it requires retraining a new prediction head for each downstream task, and *statistically*, which requires task-specific labels for multiple tasks. This poses a natural question, *how do we efficiently determine the "goodness" of representations learned with SSL across a wide range of potential downstream tasks?* In particular, a task-agnostic statistical measure of representation quality that predicts generalization without explicit downstream task evaluation would be highly desirable.

In this work, we analyze characteristics of learned representations $\mathbf{f}_\theta$ in well-trained neural networks with canonical architectures & across SSL objectives. We observe that the eigenspectrum of the empirical feature covariance $\mathrm{Cov}(\mathbf{f}_\theta)$ can be well approximated with the family of a power-law distribution. We analytically and empirically (using multiple datasets, e.g. CIFAR, STL10, MIT67, ImageNet) demonstrate that the decay coefficient $\alpha$ serves as a measure of representation quality for tasks that are solvable with a linear readout, that is, there exist well-defined intervals for $\alpha$ where models exhibit excellent downstream generalization. Furthermore, our experiments suggest that key design parameters in SSL algorithms, such as BarlowTwins [1], implicitly modulate the decay coefficient of the eigenspectrum ($\alpha$). As $\alpha$ depends only on the features themselves, this measure for model selection with hyperparameter tuning for BarlowTwins enables the search with less compute.

## 1 Introduction

The recent success of self-supervised learning (SSL) has changed the landscape of deep learning significantly. With well-engineered architectures and training objectives, SSL models learn useful representations from large datasets without relying on any labels [1, 2, 3]. Despite this progress, quantifying the *representation quality* for models trained with SSL is still an open problem. The

[†] These authors contributed equally to this work

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

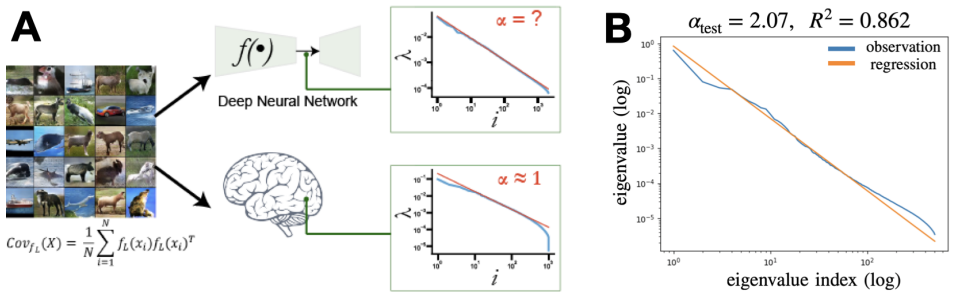



Figure 1: **(A)** One approach to evaluate representation quality is tracking the eigenspace of feature covariance matrix $\Sigma_n(\mathbf{f}) = 1/n \sum_{i=1}^{n} \mathbf{f}(x_i)\mathbf{f}(x_i)^\top$ . Analysing populations of neural activation [6, 7], suggests that the eigenspectrum for $\Sigma_n(\mathbf{f})$ can be approximated by power-law, where $\lambda_i \propto i^{-\alpha}$. **(B)** For a ResNet-50 model pretrained on ImageNet, we extract activations from an intermediate layer. We plot the eigenvalue spectrum $\{\lambda_1, ..., \lambda_d\}$ for covariance matrix $\Sigma_n$ (`resnet50(feats='block4')`) in `log-log` scale. Further, fitting a linear regressor we estimate the decay coefficient $\hat{\alpha}$.

most obvious (and common) solution is to assess performance of models using these representations on downstream tasks. However, if the goal is general representations that can be used across many domains, then this either requires a large investment of time and energy to be done well (in order to assess performance on many tasks and datasets). Alternatively, if models are assessed on only a small number of datasets and tasks, then it is hard to be confident in the assessment. Thus, we are left with a question: Can we assess the quality of learned representations without explicitly evaluating the performance on downstream tasks? To answer this question, we must formally define the "quality" of representations and subsequently examine statistical estimators that measure this property without needing downstream evaluation. Beyond theoretical interest, such a metric would be highly desirable for model selection and also useful for designing new SSL algorithms.

In search of such a metric, we turn our attention to one of the more efficient learning machines in existence – the mammalian brain. The hierarchical and distributed organization of neural circuits, especially in the cortex, provides neural representations that support a wide array of behaviours across many domains. For example, representations in primary visual cortex (V1) of mammalian brains are used by animals to support downstream behaviours ranging from object categorization to movement detection and motor control [4, 5]. This berth of downstream uses of V1 representations is desirable for artificial vision systems trained with SSL. Thus, understanding the properties of representations in V1 is a reasonable starting point for seeking a general metric of representation quality.

Recent breakthroughs in systems neuroscience enable large-scale recordings of neural activity. By recording and analyzing the response to visual stimuli, [6, 7] find that activations in the mouse and macaque monkey V1 exhibit a characteristic information geometric structure. In particular, these representations are high-dimensional, yet the amount of information encoded along the different principal directions varies significantly. Notably, this variance (computed by measuring the eigenspectrum of the empirical covariance matrix) is well-approximated by a power-law distribution with decay coefficient $\approx 1$, i.e., the $n^{th}$ eigenvalue of the covariance matrix scales as $1/n$.

Motivated by these results, we explore the use of the decay rate of the empirical eigenspectrum to characterize representation *quality* in neural networks trained with SSL (Fig. 1a). Across diverse model architectures, pretraining objectives, and downstream classification tasks, we empirically observe an eigenspectrum decay in the representation covariance matrix that roughly follows a power-law distribution (see e.g. Fig. 1b). We also find that the coefficient of this decay, denoted by $\alpha$, is informative of downstream generalization performance. Importantly, $\alpha$ can efficiently be calculated without any labels, and thereby, could be incorporated into existing SSL pipelines for efficient model selection on fixed compute budget. Our core contributions in this paper are:

1. $\alpha$ **as a potential metric** We observe that canonical pretrained architectures have representations which loosely conform to a power-law distribution in their covariance eigenspectrum. Under an assumption of a power-law distribution, we prove that the convergence rate and upper bound on generalization error are related to decay-coefficient ($\alpha$) of the corresponding power-law distribution, with best values of $\alpha$ being neither too large nor too small.

2. $\alpha$ **as a label-free measure for representation quality**: We empirically validate our theoretical results by demonstrating a relationship between $\alpha$ and downstream generalization. In particular, we find that either too high or too low an $\alpha$ value implies poor generalization, both in-distribution and out-of-distribution. Generally, the best representations are those where $\alpha$ is in a range that is close to 1, as observed in V1 of the real brain. Furthermore, these results hold irrespective of the choice of network architecture or pretraining objective.
3. $\alpha$ **for model selection in SSL**: We establish $\alpha$ as a reliable metric for model selection in a specific case of SSL, Barlow Twins [1]. Notably, we show that $\alpha$ allows us to identify model hyper-parameters that lead to representations that generalize well without any labels, more so than the actual loss function used to train the network.

Altogether, our results show that $\alpha$ is a promising task/architecture/data agnostic metric for assessing representation quality in SSL. We publicly release our results and code [†].

## 2 Theoretical Framework to Assess Representations in SSL

We are interested in evaluating the quality of high dimensional representations learned by neural networks. Formally, we consider the overparameterized setting, with datasets $\mathcal{X} \subset \mathbb{R}^d$, and learned mappings $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^D$ such that $\mathbf{f}(\mathbf{x})$ is a vector of $D$-dimensional features.

We consider DNNs as our function approximators, where each architecture implicitly defines a function class $\mathcal{F} = \{\mathbf{f}_\theta : \theta \in \Theta\}$ where $\Theta \subset \mathbb{R}^p$ is the feasible set of model parameters (e.g bounded $\Theta, [-B, B]^p$ for some $B \in \mathbb{R}$). The search for good representations poses the following optimization problem: with a dataset $\mathcal{D}_{\text{pretrain}}$ from some data distribution $\mathbb{P}_{\text{pretrain}}$ (potentially with labels), search for *optimal* parameters $\theta^*$ such that with pretrain objective $\mathcal{L}_{\text{pretrain}}(\mathbf{f}, \mathcal{D})$

$$\theta^* = \arg\min_{\mathbf{f}_\theta \in \mathcal{F}} \mathcal{L}_{pretrain}(\mathbf{f}_\theta, D_{\text{pretrain}}) \tag{1}$$

The above optimization problem is usually non-convex, and often uses gradient based optimizer to find an approximate solution $\hat{\theta}$. Typically, to measure the quality of $f_{\hat{\theta}}$, researchers evaluate the quality of representations on a downstream task $\mathcal{D}_{\text{downstream}}$ with a linear readout using some metric $\mathcal{R}(\texttt{linear}(\mathbf{f}_{\hat{\theta}}), D_{\text{downstream}})$. A concrete example is pretraining a VGG-16 model ($\mathcal{F}$) on ImageNet dataset ($D_{\text{pretrain}}$) and evaluating the learned representations using classification accuracy ($\mathcal{R}$) by learning a linear classifier on the MIT67 dataset ($D_{\text{downstream}}$). For the rest of the paper, we denote feature maps as $\mathbf{f}_\theta(x) \in \mathbb{R}^D$ where $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^D$, and the readout network as $\mathbf{g}_\phi : \mathbb{R}^D \to \mathbb{R}^k$, where $k$ is the target dimensionality. For simplicity of analysis, we consider linear readouts unless explicitly mentioned, i.e. $\mathbf{g}_\phi(x) = x^T\phi$.

### 2.1 Covariance estimation and eigenspectrum

For a parameterized function $\mathbf{f}_\theta : \mathcal{X} \to \mathbb{R}^D$ (assume centered), the ($n$-sample) empirical covariance matrix is defined as:

$$\Sigma_n(\mathbf{f}_\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_\theta(x_i)\mathbf{f}_\theta(x_i)^T$$

The eigenspectrum of $\Sigma_n(\mathbf{f}_\theta)$ informs us about the variance explained by each principal component of the space spanned by representations $\mathbf{f}_\theta(\mathbf{x})$. Using the spectral decomposition theorem on symmetric matrices, $\Sigma = U\Lambda U^T$, where $\Lambda$ is a diagonal matrix with nonnegative entries, and $U$ is a matrix whose columns are the eigenvectors of $\Sigma$. Without loss of generality we assume that $\lambda_1 \geq \lambda_2 ... \geq \lambda_m$, where $m = \min(n, D)$ is the rank of $\Sigma_n(\mathbf{f}_\theta)$.

### 2.2 Eigenspectrum Decay in Deep Representation Learning

Recent work in characterizing representation structure in canonical DNNs has demonstrated that the covariance eigenspectrum roughly follows a power-law [8, 9, 10]. Specifically, the eigenspectrum

[†] https://github.com/kumarkrishna/fastssl

of a covariance matrix follows a power-law distribution $PL(\alpha)$, or *zeta distribution* if for $\lambda_j \in [\lambda_{min}, \lambda_{max}]$, the eigenvalues $\lambda_j$ are all nonnegative, and

$$\lambda_j \propto j^{-\alpha}$$

for some $\alpha > 0$. Here, $\alpha$ is the *slope* of the power law, and is referred to as the *coefficient of decay* of the eigenspectrum. Intuitively, small $\alpha$ (typically $\alpha \leq 1$) suggests a dense encoding, while a high $\alpha$ (rapid decay) corresponds to a sparse encoding.

**Insights from High-Dimensional Linear Regression** : Recent work in theoretical machine learning has connected bounds on generalization error for a linear regression problem to the eigenspectrum of the feature covariance matrix. Specifically, in the infinite-dimensional setting with $D \to \infty$, [11] studied the linear regression setting with Gaussian features, and proved that if the eigenspectrum follows a power-law distribution (up to polylogarithm factors), the min-norm solution provides good generalization performance iff $\alpha = 1$. The asymptotic regime of infinite width is an excellent framework to study theoretical properties of DNN representations. However, despite having a large number of parameters, practical DNNs always possess finite dimensional representations, making it important to investigate the implications of such results in finite width models. In particular, for the finite dimensional setting we try to answer: *Does $\alpha$ sufficiently larger or smaller than 1, still allow efficient learning and strong generalizability?*

To answer this question, we narrow our focus to gradient-based optimization techniques usually used to train DNNs. In particular, from the optimization perspective, Advani *et al.* showed that for deep linear regression in high dimensions, the time required for training and the steady-state generalization error are both $\mathcal{O}(\frac{1}{\lambda_{min}})$ [12]. A key difference from our work is that they assume the inputs are drawn from an isotropic Gaussian distribution. Instead, we investigate the generalization of linear regression on $\mathbf{f}_\theta(x)$, which has a power law structure in its covariates. We show that the training convergence time grows exponentially with $\alpha$ using the following theorem:

**Theorem 2.1.** *Let $\hat{y} = \mathbf{f}_\theta(x)^T\psi$ be an overparameterized linear regression problem where $\psi$ is learned using gradient descent in order to optimize the training error, $\mathbb{E}_{x,y}[(y - \mathbf{f}_\theta(x)^T\psi)^2]$, where $(x, y) \sim \mathcal{D}_{train}$. If we assume a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, then the time required by gradient descent to minimize the training error, $T_{convergence} = \mathcal{O}(N^\alpha)$ where N is number of training samples.*

The outline of the proof builds on key results relating to gradient descent dynamics from [13]. We show that gradient descent updates, when $\psi$ is initialized to 0, yield a recursive relation for $\psi(k)$, i.e. $\psi$ after $k$ update steps. Plugging this relation in the gradient formulation, we show that the update step length along the $n^{th}$ principal direction of $\mathbf{f}_\theta(x)$ shrinks exponentially with a decay rate proportional to $\lambda_n$. Therefore, the time to convergence in training is controlled by the smallest eigenvalue which, by design, follows the power law. In sum, Theorem 2.1 provides an explanation against arbitrarily large values of $\alpha$.

Next we show that $\alpha$ can't be arbitrarily small as the upper bound on generalization error is higher for smaller values of $\alpha$.

**Theorem 2.2.** *Let $\hat{y} = \mathbf{f}_\theta(x)^T\psi$ be a linear regression problem as before. Let us further assume that $\mathbf{f}_\theta(x)\ \forall x \sim \mathcal{D}_{train}$ is a representative subset of the inputs from true data distribution: $(x, y) \sim \mathcal{D}$. Assuming a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, the generalization error after T weight update steps, $\mathcal{G}(T)$ is:*

$$\begin{aligned} \mathcal{G}(T) := \mathbb{E}_{x,y\sim\mathcal{D}}[(y - \mathbf{f}_\theta(x)^T\psi)^2] \quad &\leq \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)\right) \\ \textit{where} \qquad r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)) = \frac{\sum_{i=\hat{d}}^{m}\lambda_i}{\sum_{i=1}^{m}\lambda_i} &= \frac{\sum_{i=\hat{d}}^{m} i^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}} \end{aligned} \tag{2}$$

*Here, $m = min(n, D)$ is the rank of $\Sigma_n(\mathbf{f}_\theta)$.*

Notably, $r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))$ can be thought of as a measure of effective rank of the covariance matrix and grows with decreasing values of $\alpha$. Therefore, the upper bound for generalization error is higher for lower $\alpha$. Taken together, Theorems 2.1 and 2.2 suggest that $\alpha$ can neither be too high nor too low and there exists a tradeoff region which results in efficient learning and strong generalizability, where these representations form a good basis for gradient-based optimization on downstream task performance.

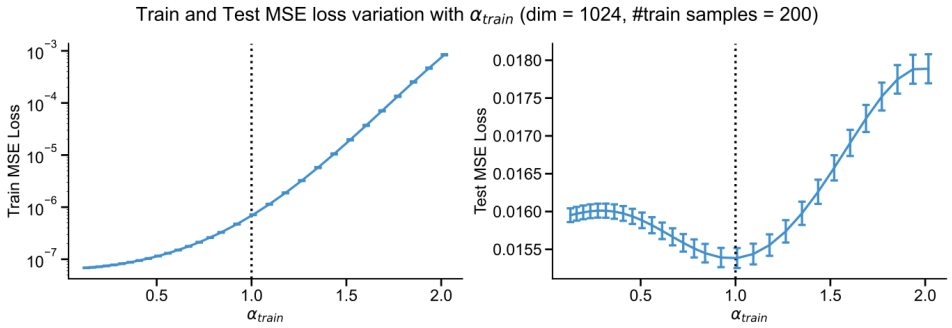



Figure 2: Overparameterized linear regression, with inputs drawn from Gaussian distribution with power-law in the covariance matrix. Models are trained with gradient descent with initialization $\psi_0 = \mathbf{0}$. Note that $\alpha > 1$ particularly suffers from high train, test MSE loss, with low generalization error with $\alpha \approx 1$.

**Another motivating example** Before exploring the link between $\alpha$ and generalization performance of deep networks, we validate our theorems in a simple linear regression setting.

We first consider its relationship to the finite dimensional regression setting as in [11]. Specifically, Theorem 6 of their paper states that the conditions for "benign overfitting", wherein a model can perfectly fit noisy training data without any subsequent loss of performance on testing data, may be looser in finite dimensions as opposed to the necessary and sufficient condition of $\alpha = 1$ in infinite dimensions. To empirically test this in the finite high dimensional setting, we examine linear least squares regression using different covariate structures for the input data. Formally, we consider covariates $\{x_i\}_{i=1}^N$, such that $x_i \in \mathbb{R}^d$ is sampled from a Gaussian distribution with covariance structure $\Sigma = \text{diag}\{\lambda_1, ...\lambda_d\}$ where $\lambda_j \sim PL(\alpha)$, i.e $\lambda_j \propto cj^{-\alpha}$. We assume access to the corresponding labels $\{y_i\}_{i=1}^N$ generated under a teacher function $\theta^*$, such that $y_i = x_i^T\theta^* + \epsilon_i$. We find that in this scenario there is a clear relationship between the proximity of $\alpha$ to 1 and the presence of benign overfitting. As shown in Fig. 2, when $\alpha$ is close to 1, the training loss is low, but the validation loss is also low. Thus, when $\alpha$ is close to 1, the generalization properties are at their best. This example thus provides another hint that $\alpha$ is a potential measure for how well a model will be able to generalize.

## 3 Experimental Setup

Our theoretical results suggest the following: for representations with power-law characteristics, the decay-coefficient $(\alpha)$ effects the *adaptivity*, where the convergence rate of linear regression with gradient descent scales exponentially $\mathcal{O}(n^\alpha)$ with $\alpha$. On the other hand, if trained sufficiently long, the upper bound for generalization error improves with larger $\alpha$. Together, these items imply that if one wants to use gradient descent to train a downstream task then $\alpha$ for the representations used

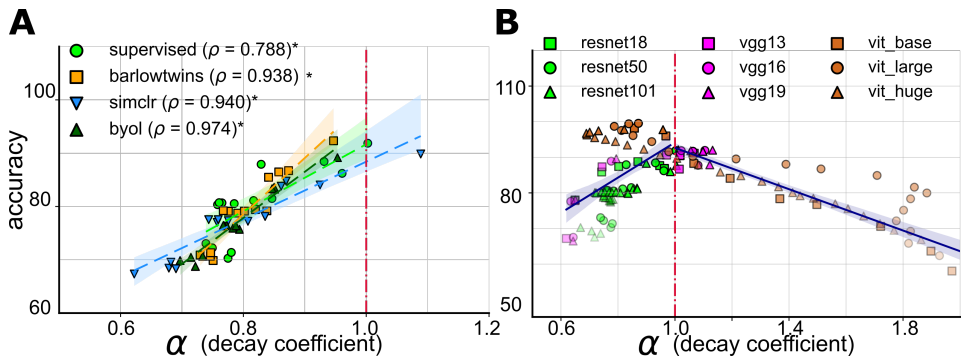



Figure 3: $\alpha$ **is predictive of out-of-distribution object recognition performance**. $\alpha$ is strongly correlated to object recognition performance on STL10 across different architectures and pretraining loss functions.

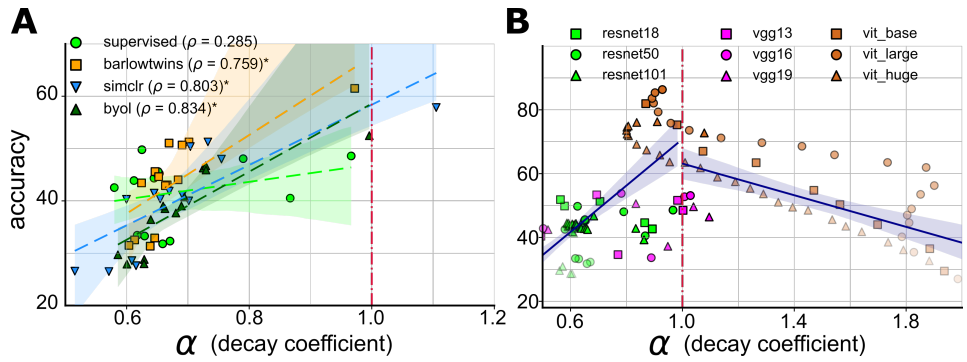



Figure 4: $\alpha$ **is predictive of out-of-distribution object recognition performance**. $\alpha$ is strongly correlated to object recognition performance on MIT67 across different architectures and pretraining loss functions.

should be neither too large nor too small, i.e. high quality representations are those with $\alpha$ in a "Goldilocks" zone. Building on these insights, we now design experiments to empirically study the potential for $\alpha$ to be used as a representation quality metric. We do so by measuring the relationship between $\alpha$ and the standard metric of representation quality, i.e. downstream task generalization performance. In this work, we restrict our scope to vision tasks, specifically object classification and scene recognition. In particular, our experiments:

- **measure the eigenspectrum decay & generalization** : We systematically evaluate *in-distribution* and *out-of-distribution* generalization, and simultaneously, we also measure $\alpha$, across diverse choices for network architectures, pretraining learning objectives, and across multiple datasets. We then look at whether $\alpha$ values near some neighbourhood around 1 are indeed predictive of representations that generalize well to downstream tasks.
- **explore $\alpha$ as a metric for model-selection in SSL:** Measuring $\alpha$ is label-free and computationally inexpensive, as it requires a single forward-pass on the downstream dataset. We thus ask, is $\alpha$ predictive of generalization capabilities for current SSL algorithms, and if so, can it be used for model selection? We run extensive ablations on design of the BarlowTwins [1] algorithm in an effort to answer this question.

### 3.1 Measuring eigenspectrum Decay & Generalization

**Evaluation Protocol:** To measure $\alpha$ for the learned representation $\mathbf{f}(x_i)$, we first extract features from intermediate layers of a DNN pretrained on $\mathcal{D}_{pretrain}$ and estimate the corresponding covariance matrix $\Sigma_n(\mathbf{f})$. Next, we compute the full set of numerical eigenvalues, and estimate $\alpha$ by a fitting a linear model on the eigenspectrum in `log-log` scale. Our pretrained models are taken from PyTorch Hub [14] and `timm` [15]. Given that we observed no significant difference between the observed $\alpha$ values in the train and test sets, we refer to this empirical estimate as the $\alpha$ for the dataset.

To estimate the capacity of intermediate representations in solving the downstream task, we train a linear readout layer, $\mathbf{g}(.)$, from representations to target logits. Intuitively, this comes down to establishing a relationship between the manifold geometry and linear separability of representations [16]. Thereafter, we observe the correlation between estimated $\alpha$ and the linear readout performance. Notably, we also tried non-linear $\mathbf{g}(.)$ and observed a similar trend in results (see Appendix B.1).

**Inductive bias of Model Architecture** In this section, we investigate the relationship between $\alpha$ of the representation covariance matrix and object/scene recognition performance in DNNs with different backbones and the role of depth. In order to do so, we examine varying depth configurations within network architectures across three generations of models on the STL10 and MIT67 dataset [17]. We choose both object and scene recognition task as they fundamentally differ in the nature of the features that must be extracted to perform well. For scene recognition, the model focuses on global features in the entire image, while for object recognition, the model focuses on local features of the image containing the object[18].

We examine deep Convolutional Neural Networks (CNNs) without any residual connection as our first family of models. Specifically, we choose three different configurations of VGG-Net [19], namely VGG-13, VGG-16 and VGG-19. We inspect representations that are input to the dropout and MaxPool layers during the forward pass of the network. Second, we consider Deep Residual Networks [20] which are widely used in computer vision. We inspect the representations that are input to each of the residual blocks as well as the Adaptive average pool in ResNet-13, ResNet-50 and ResNet-101 during their respective forward passes. Finally, owing to the recent success of transformers in object recognition tasks we consider Vision Transformers (ViT) [21] as the third family of models, namely ViT-Base/8 , ViT-Large/16 and ViT-Huge/14. Unlike VGG and ResNet, we only look at features in the intermediate layers because it summarizes the entire input image and is used in practice for class prediction. For all model architectures, we use the weights obtained from pretraining on ImageNet.

Fig. 3 and Fig. 4 illustrate the relation between performance and $\alpha$ for all the nine architectures across intermediate layer representations, as described above. We found that, while most intermediate representations in CNNs (with or without residual connections) exhibit $\alpha < 1$, representations in ViTs mostly exhibit $\alpha > 1$. Nevertheless, representations extracted from the deepest layers of all the models exhibit $\alpha$ close to 1, irrespective of the total depth of each model (see Appendix B). Furthermore, the performance on downstream task increases with depth. This is unsurprising because all networks were trained to perform object recognition on ImageNet [22] and thereby would have leveraged hierarchical processing to learn features that are tuned towards object recognition. Surprisingly enough, we observe a strong significant correlation between $\alpha$ and performance on the STL10 dataset, i.e. a different data distribution than the training dataset, across layers and model architectures ($\rho = -0.922$, $^*p < 0.05$ for representations exhibiting $\alpha > 1$ and $\rho = -0.922$, $^*p < 0.05$ for representations exhibiting $\alpha < 1$). It is worth noting here that the correlation was weaker for the earliest layers of each model. We believe that early layers learn more task invariant features that reflect the statistics of natural images [23, 24] and therefore lack task relevant information in their representations. Taken together, this observation confirms our hypothesis that $\alpha$ is a good indicator of out-of-distribution generalization performance when representations possess task relevant information. Thus, $\alpha$ has a *necessary but not sufficient* relationship with generalization (Appendix C).

**Learning objective** In this section, we first aim to understand how the $\alpha$ value changes across the layers of a fixed architecture DNN when trained with different learning objectives. We take a ResNet-50 model [20] pre-trained using three different SSL algorithms, namely SimCLR [2], BYOL [25] and Barlow Twins [1], and the supervised learning loss objectives on ImageNet-1k[22] dataset. We use a similar procedure as before to extract representations from the network and estimate $\alpha$.

Similar to results in the previous section, all networks irrespective of the pretraining loss function, exhibit $\alpha$ closer to 1 in the deeper layers in contrast to intermediate layers (see Appendix Fig. 9). This surprising result indicates that although the pre-training loss function was different, representations extracted from deepest layers are reflective of the object/scene semantics in natural images. Furthermore, Fig. 3 and Fig. 4 illustrate the strong correlation between $\alpha$ and generalization performance on STL10 and MIT67 across all pre-training loss functions. Together with results from the previous section, we validate our hypothesis that the representations that demonstrate good out-of-distribution generalization performance are characterized by $\alpha$ in the neighbourhood of 1. This suggests that high quality representations are indeed those with $\alpha$ in this region.

### 3.2 Label-agnostic metric for Model selection

With empirical evidence of $\alpha$ being a good measure for generalization, we now wish to study its suitability as a metric for identifying the best among models pretrained with SSL algorithms. A label-free metric like $\alpha$ could be beneficial when we don't have access to the downstream task annotations, and the SSL loss is not useful to distinguish models with good generalization performance (see Fig. 5). To investigate this, we exhaustively ablate the relationship between $\alpha$ and model performance across a wide range of hyper-parameters for a representative non-contrastive SSL algorithm.

Current SSL algorithms struggle with dimension collapse (characterized by large $\alpha$), due to pathologies in training dynamics. To study $\alpha$ across a wide-range of values, for our model selection experiments, we pick the *Barlow Twins*[1], a non-contrastive SSL algorithm that explicitly optimizes to avoid dimension collapse. In particular, the Barlow Twins learning objective ($\mathcal{L}_{\mathrm{BT}}$) proposes imposing a *soft-whitening* constraint :

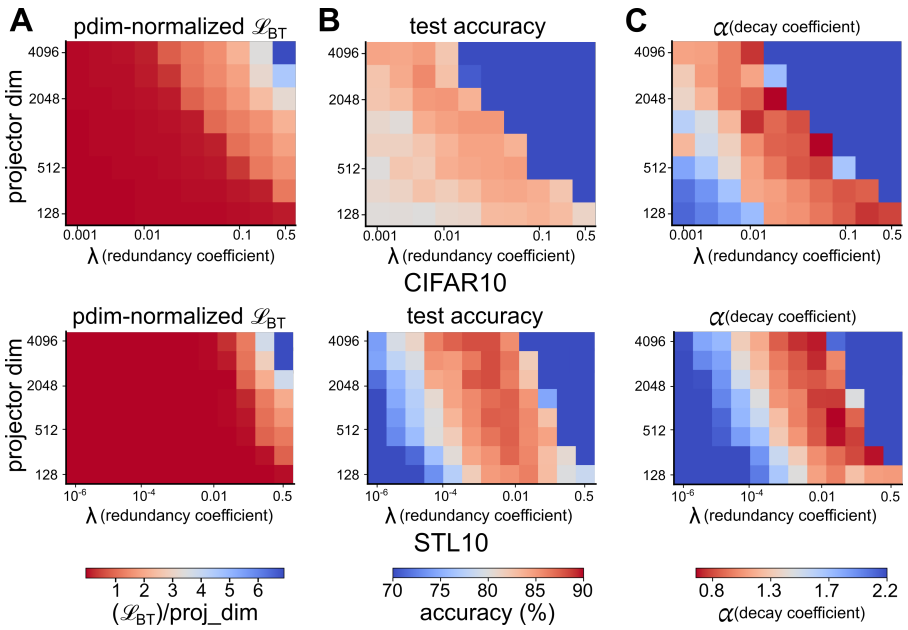



Figure 5: $\alpha$ **as a metric to inform model selection.** The SSL loss (training for same number of gradient steps) is no longer useful to distinguish models with superior downstream performance. However, decay coefficient $\alpha$ shows strong correspondence to downstream test accuracy over a large hyperparameter ranges. Measuring in-distribution generalization for (A-C) BarlowTwins trained and evaluated on CIFAR10. (D-F) BarlowTwins trained and evaluated on STL10.

$$\mathcal{L}_{\text{BT}} = \underbrace{\sum_{i=1}^{d}(1-\mathcal{C}(\mathbf{f}_\theta)_{ii})^2}_{\text{invariance}} + \lambda \underbrace{\sum_{i=1}^{d}\sum_{j\neq i}^{d}\mathcal{C}(\mathbf{f}_\theta)_{ij}^2}_{\text{redundancy-reduction}} \quad \text{s.t. } \mathcal{C}(\mathbf{f}_\theta)_{ij} = \frac{\sum_n \mathbf{f}_\theta(x^A)_i \mathbf{f}_\theta(x^B)_j}{\sqrt{\sum_n \mathbf{f}_\theta(x^A)_i^2 \sum_n \mathbf{f}_\theta(x^B)_j^2}} \tag{3}$$

Notably with sufficient large $\lambda$, the model would impose an $\alpha = 0$ constraint on the representations [26]. Despite the intuitive connection between $\alpha$ and $\lambda$, it is unclear whether this relationship holds across a wide-range of values for model hyperparameters. To empirically establish this connection, we vary $\lambda$ (redundancy coeffiecient), projection head dimensionality and learning rate, and train a ResNet50 encoder using *Barlow Twins* learning objective. We provide the results for CIFAR10 [27] and STL10 [28] in Fig. 5 demonstrating that $\alpha$ is a strong indicator of in-distribution generalization performance across a large range of hyperparameter ranges of $\mathcal{L}_{BT}$. We defer the reader to Appendix Fig. 14 & Fig. 15 for similar trends in optimization hyperparameters. Furthermore, $\alpha$ is predictive of out-of-distribution generalization performance in these settings (see Appendix B.3). The same relation also holds in contrastive SSL frameworks, specifically for SimCLR [2] (see Appendix B.4).

**Model Selection on a Compute Budget:** With a constrained compute budget, using $\alpha$ for model selection is significantly cheaper than evaluating the representations on a suite of downstream tasks. To illustrate this, consider the standard alternative of training a linear classifier on the representations and evaluating test accuracy. Compared to training a linear probe, which requires multiple epochs of forward & backward passes through the training dataset's features, to achieve reasonable estimates of downstream accuracy, computing $\alpha$ requires a single PCA step on the validation dataset's features. In Table 1, we contrast the compute times for $\alpha$ and evaluating a linear probe. A detailed algorithm to use $\alpha$ for model selection is provided in Algorithm 1 and its complexity analysis in Appendix B.6.

Table 1: We report the compute time for CIFAR10, STL10, and ImageNet. While CIFAR10 and STL10 are trained for 200 epochs for downstream classification ImageNet is trained for 100 epochs. (tested in 1 A100)

| Dataset | w/ eigenspectrum coefficient ($\alpha$) | w/ linear probe | perf. gains |
|---|---|---|---|
| CIFAR10 (s) | **3**$\pm$ **0.2** | 48$\pm$ 7 | $\sim 16\times$ |
| STL10 (s) | **5**$\pm$ **0.5** | 80$\pm$ 10 | $\sim 16\times$ |
| ImageNet (mins) | **4**$\pm$ **0.8** | 58$\pm$ 5 | $\sim 15\times$ |

**Algorithm 1** Model selection using $\alpha$

```
# M: Number of models that can be trained in parallel
# H: Number of sequential steps of model training
# K: Number of top models to log in memory for model selection

α_min = 0
α_max = ∞
models_dict = {}

for iter in range(H):
    # train M models in parallel, each with different hyperparam config
    trained_models = train_SSL_models(num_parallel_models=M)

    # evaluate α for the trained models
    alpha_trained_models = evaluate_alpha(trained_models)
    best_trained_models = {model: alpha for model, alpha in
        alpha_trained_models if alpha ∈ [α_min,α_max]}
    models_stat = {model: run_linear_eval(model) for model, alpha in
        best_trained_models.items()}

    # Update α_min and α_max
    α_min = max({α_m | α_m > α_j & acc_m ≥ acc_j ∀ m,j ∈ models_stat})
    α_max = min({α_m | α_m < α_j & acc_m ≥ acc_j ∀ m,j ∈ models_stat})

    # Trim models_stat to keep only top K models
    threshold = get_best_models(model_statistics, topk=K)
    models_stat = {model: acc for model, acc in models_stat.items() if acc >
        threshold}

# Return best model performance from the 'selected' models
best_model_acc = max({acc_m ∀ m ∈ models_stat})
return best_model_acc
```

## 4 Related Work

**Evaluating representations and model quality** We note a substantial body of work aiming to empirically characterize the structure of emergent representations in DNN without requiring labels [29, 30]. One such index that quantifies the similarity of representations across layers (of the same or different models) is Centered Kernel Alignment (CKA) [23]. While CKA does not provide explicit guidance for downstream performance, [31] shows that the in-distribution generalization gap can be predicted using a different index based on the model's parameters. In particular, they show that the Empirical Spectral Density (ESD) of weight matrices for many DNNs obey a power-law, with the decay coefficient being predictive of in-distribution performance. In the present work, we explore similar indices that potentially correlate with out-of-distribution generalization by examining the eigenspectrum of *activations*.

**Generalization in Overparameterized Models** Modern neural networks often have significantly more parameters than the number of training samples, challenging the classical understanding of the bias-variance tradeoff. Overparameterization permits neural networks to overfit to noise in training data without impairing their generalization to unseen data. In recent work, Bubeck *et al.* proved that overparameterization is a necessary condition for smooth interpolation in neural networks [32].

Furthermore, this *benign overfitting* phenomenon in an overparameterized linear regression problem has been linked to the power law coefficient of the input covariance matrix [11]. Specifically, Bartlett *et al.* showed that benign overfitting is possible for an infinite-dimensional linear regression problem iff the eigenspectrum satisfies a power law (up to polylog factors). More recently, Lee *et al.* found that the tail eigenvalues of infinite-width network kernels exhibit a power law decay [33]. Following this, Tripuraneni *et al.* explored high-dimensional random feature regression settings and analytically showed a dependence between the eigenspectrum decay rate of the feature covariance matrix and generalization error [34]. While these characterizations provide a theoretical understanding of generalization error in the asymptotic or random feature settings, corresponding questions in the finite-dimensional DNN trained with gradient descent are open problems.

For deep linear networks trained using gradient descent, the eigenvalues of input covariance determine the generalization error dynamics [12]. Advani *et al.* demonstrated that small eigenvalues determine the convergence of training dynamics as well as the overfitting error at convergence. For overpa-

rameterized 2-layer neural networks, Arora *et al.* provided a fine-grained analysis of generalization bounds [35]. In contrast, we study modern DNN architectures and explore the covariance structure of their learned features on visual recognition tasks.

## 5 Discussion

**Summary** Our experiments suggest a strong correlation between the decay coefficient for sample eigenspectrum of representations, $\alpha$, and both in-distribution and out-of-distribution generalization performance on tasks central to computer vision.

**Representation Quality in High-Dimensional SSL** We demonstrate that assessing the quality of a pretrained model with our label-free metric ($\alpha$) is consistent with downstream generalization. While being computationally efficient, our procedure removes the dependence on labels, making the model selection more robust and amenable for privacy critical applications.

**Necessary, but not sufficient condition** Notably, a task-agnostic measure like $\alpha$ is a necessary but not sufficient condition for assessing good performance. To elaborate on this point, let us present a thought experiment. Suppose we have our ideal representation space that satisfies our claims of exhibiting alpha close to the Goldilocks zone. If we perform a set of permutation operations on individual datapoint representations, the structure of the representation space remains the same, but the mapping from representation to label is now destroyed. In doing this set of permutations, we have now arrived at a different set of representations that would exhibit the same (or similar) alpha but demonstrate a lower task performance when measured using a linear readout layer. A similar argument is presented as theorem C.1 in [36]. Extending the conclusions of this thought experiment to our observations, it is clear that we could have models with similar alpha values but distinctly different performances. But an alpha value that is not in the Goldilocks zone would be associated with inferior model performance. This property is the core of our claim and allows us to propose alpha as a measure for model selection in SSL pipelines.

**Redundancy in ViT representations** Unlike other models that we considered, representations from early layers of ViT had a rapid eigenspectrum decay with $\alpha >> 1$ (see Fig. 3 and Fig. 4). The transformer architecture has a notable difference by design, i.e. early layers possess global receptive field context via self-attention on patch embeddings. Raghu *et al.* found that early ViT layers incorporate both local and global information [30]. Based on these insights, one intuitive interpretation of our results is that the representations have a low effective rank and encode redundant information relevant across multiple scales.

**Limitations** While the role of $\alpha$ and its relationship to generalization performance is better understood in the asymptotic setting for linear regression, similar questions in finite-dimensional nonlinear models are unanswered. It is also worth noting that the empirical correlation was weaker for the earliest layers in each model. We believe that early layers learn more task invariant features such as corners and edges [23, 24], and thereby lack rich semantic information for the fine-grained downstream task. In this case, distinguishing between poorly-trained models may be inconsistent with $\alpha$.

**Future Directions** Learning efficiently at scale from unlabelled datasets poses an exciting open problem in deep representation learning. We hope this work encourages new perspectives into model selection for SSL pipelines and informs the design of learning objectives and model architectures to learn *task-agnostic, adaptive* features. Understanding the behaviour of $\alpha$ (notably, a scale-invariant metric) in high-dimensional representation learning might provide insight into developing a theoretically grounded understanding of generalization in deep neural networks.

## Acknowledgements

The authors would like to thank Zahraa Chorghay and Colleen Gillon for their aesthetic contribution to the manuscript and figures. This research was enabled in part by support provided by Mila and Compute Canada. This work was supported by Vanier Canada Graduate scholarship (AG); Healthy Brains, Healthy Lives (AG & BAR); NSERC, Grant No. RGPIN-2020-05105 and RGPAS-2020-00031 (BAR); and CIFAR, Canada AI Chair & Learning in Machines and Brains Fellowship (BAR).

## References


[1] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

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

[3] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning with dual deep networks. *arXiv preprint arXiv:2010.00578*, 2020.

[4] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991.

[5] Claus C Hilgetag and Alexandros Goulas. 'hierarchy'in the organization of brain networks. *Philosophical Transactions of the Royal Society B*, 375(1796):20190319, 2020.

[6] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[7] Nathan CL Kong, Eshed Margalit, Justin L Gardner, and Anthony M Norcia. Increasing neural network robustness improves match to macaque v1 eigenspectrum, spatial frequency preference and predictivity. *PLOS Computational Biology*, 18(1):e1009739, 2022.

[8] J Nassar, P Sokol, S Chang, and K Harris. On 1/n neural representation and robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Zijian Jiang, Jianwen Zhou, and Haiping Huang. Relationship between manifold smoothness and adversarial vulnerability in deep learning with local errors. *Chinese Physics B*, 30(4):048702, 2021.

[10] Zeke Xie, Qian-Yuan Tang, Yunfeng Cai, Mingming Sun, and Ping Li. On the power-law spectrum in deep learning: A bridge to protein science. *arXiv preprint arXiv:2201.13011*, 2022.

[11] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[12] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[13] Vatsal Shah, Anastasios Kyrillidis, and Sujay Sanghavi. Minimum norm solutions do not always generalize well for over-parameterized problems. *stat*, 1050:16, 2018.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[16] SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Classification and geometry of general perceptual manifolds. *Physical Review X*, 8(3):031003, 2018.

[17] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009.

[18] A Oliva and A Torralba. Chapter 2 building the gist of a scene: the role of global image features in recognition. *Progress in Brain Research*, pages 23–36, 2006.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[24] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[25] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[26] Bobby He and Mete Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In *International Conference on Machine Learning*, pages 8613–8634. PMLR, 2022.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

[28] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[29] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[30] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

[31] Charles H Martin, Tongsu Serena Peng, and Michael W Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):1–13, 2021.

[32] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *arXiv preprint arXiv:2105.12806*, 2021.

[33] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

[34] Nilesh Tripuraneni, Ben Adlam, and Jeffrey Pennington. Covariate shift in high-dimensional random feature regression. *arXiv preprint arXiv:2111.08234*, 2021.

[35] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[36] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19250–19286. PMLR, 17–23 Jul 2022.

# A Appendix

## A.1 Proofs

In this section, we present a formal proof of Theorem 2.1. In order to do so, we will use a lemma pertaining to iterative expression of the linear regression parameters over training epochs. This lemma is inspired by the results presented in [13]

**Lemma A.1.** *Let $\hat{y} = x^T w$ be a finite dimensional linear regression problem where $w$ is learned using gradient descent in order to optimize the training error,*

$$\mathcal{L} = \mathbb{E}_{x,y}[(y - \hat{y})^2] = \mathbb{E}_{x,y}[(y - x^T w)^2] \tag{4}$$

*where $(x, y) \sim \mathcal{D}_{train}$, i.e. the training dataset. Then $w_k$, i.e. $w$ after training for $k$ epochs can be written as*

$$w_k = X^T (XX^T)^{-1} \left[I - (I - \eta XX^T)^k\right] Y \tag{5}$$

*where $X, Y$ indicate the entire training dataset, i.e. $X \in \mathbb{R}^{N \times d}$ and $Y \in \mathbb{R}^N$*

*Proof.* We start with the gradient of the regression loss function for a defined training set denoted by $(X, Y)$ in a vectorized notation:

$$\begin{aligned} & F(w_k) = |Y - Xw_k|^2 \\ \implies & \nabla F(w_k) = X^T(Xw_k - Y) \end{aligned} \tag{6}$$

$$\tag{7}$$

We assume that the weights are initialized at 0, i.e. $w_0 = 0$. Using the gradient descent update:

$$\begin{aligned} w_{k+1} &= w_k - \eta \nabla F = w_k - \eta X^T(Xw_k - Y) = (I - \eta X^T X)w_k + \eta X^T Y \\ w_1 &= (I - \eta X^T X)w_o + \eta X^T Y = \eta X^T Y \\ \text{Let } w_k &= \eta X^T u_k Y \implies u_1 = I \\ w_2 &= \eta X^T u_2 Y \\ &= (I - \eta X^T X)\eta X^T Y + \eta X^T Y = \eta X^T[(I - \eta XX^T) + I]Y \\ \implies u_2 &= (I - \eta XX^T) + I \\ u_k &= (I - \eta XX^T)u_{k-1} + I = \sum_{i=0}^{k} (I - \eta XX^T)^{i-1} \\ &= (I - (I - \eta XX^T))^{-1}(I - (I - \eta XX^T)) \sum_{i=0}^{k} (I - \eta XX^T)^{i-1} \\ &= (\eta XX^T))^{-1}(I - (I - \eta XX^T)) \sum_{i=0}^{k} (I - \eta XX^T)^{i-1} \\ &= \frac{1}{\eta}(XX^T)^{-1} \sum_{i=0}^{k} [(I - \eta XX^T)^{i-1} - (I - \eta XX^T)^i] \\ &= \frac{1}{\eta}(XX^T)^{-1}[I - (I - \eta XX^T)^k] \\ \implies w_k &= \eta X^T u_k Y = X^T(XX^T)^{-1}[I - (I - \eta XX^T)^k]Y \end{aligned} \tag{8}$$

□

This proves Lemma A.1. We will now use this lemma to prove Theorem 2.1. From this result, we can also write:

$$\Delta w_k = \eta X^T (I - \eta XX^T)^k Y \tag{9}$$

We restate the theorem from the main text. Note that the notations are simplified here from the theorem statement to improve readability.

**Theorem A.2.** *Let $\hat{Y} = X^T w$ be a finite dimensional linear regression problem where $w$ is learned using gradient descent. If we assume power law distribution in eigenspectrum of $X$, i.e. $\lambda_n = \frac{c}{n^\alpha} \forall n \in \{1, 2...N\}$, then the time required by gradient descent to minimize the training error, $T_{convergence} = \mathcal{O}(N^\alpha)$*

*Proof.* Using result from Lemma A.1, it is clear that the gradient converges to 0 if $\lambda_1 < \frac{1}{\eta}$ where $\lambda_1$ is the leading eigenvalue of $XX^T$.

Thus, $\eta < \frac{1}{\lambda_1}$, i.e. small learning rate setting. So, we set $\eta = \frac{\hat{\eta}}{\lambda_1}$ where $\hat{\eta} < 1$. Plugging this in Eq. (9)

$$\Delta w_k = \frac{\hat{\eta}}{\lambda_1} X^T (I - \frac{\hat{\eta}}{\lambda_1} XX^T)^k Y \tag{10}$$

Let $X = U \wedge^{\frac{1}{2}} V^T$ denote the singular value decomposition (SVD), which implies $XX^T = U \wedge U^T$. Using the SVD, we get $(I - \frac{\hat{\eta}}{\lambda_1} XX^T)^k = (I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)^k$. It is worth noting that eigenvalues and eigenvectors of $(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)$ are related to that of $XX^T$ as shown below:

$$\begin{aligned}
(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T) u_i &= u_i - \frac{\hat{\eta}}{\lambda_1} (U \wedge U^T u_i) \\
&= u_i - \frac{\hat{\eta}}{\lambda_1} \lambda_i u_i \qquad \text{[Using } U^T U = I\text{]} \\
\implies (I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T) u_i &= (1 - \frac{\hat{\eta}}{\lambda_1} \lambda_i) u_i
\end{aligned} \tag{11}$$

Using Eq. (11), we can write $(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T)$ in the eigendecomposition form as $U \tilde{\wedge} U^T$ where $\tilde{\lambda}_i = 1 - \hat{\eta} \frac{\lambda_i}{\lambda_1}$. Thus, $\left(I - \frac{\hat{\eta}}{\lambda_1} U \wedge U^T\right)^k = U \tilde{\wedge}^k U^T$. Plugging this in Eq. (10), we get:

$$\Delta w_k = \frac{\hat{\eta}}{\lambda_1} V \wedge^{\frac{1}{2}} U^T U \tilde{\wedge}^k U^T Y = \frac{\hat{\eta}}{\lambda_1} V \wedge^{\frac{1}{2}} \tilde{\wedge}^k S \tag{12}$$

where $S = U^T Y \in \mathbb{R}^N$. For the $i^{th}$ element, we get $\Delta w_k^{(i)} = \frac{\hat{\eta}}{\lambda_1} \sum_j v_{i,j} \sqrt{\lambda_j} \tilde{\lambda_j}^k S_j = \frac{\hat{\eta}}{\lambda_1} \sum_j v_{i,j} \sqrt{\lambda_j} (1 - \hat{\eta} \frac{\lambda_j}{\lambda_1})^k S_j$. Since all other factors remain constant across training, i.e. do not change with $k$, the convergence of gradient descent depends on the factors $\left(1 - \hat{\eta} \frac{\lambda_j}{\lambda_1}\right)^k$. Note that we define gradient descent to converge when $\Delta w_k^{(i)} \approx 0 \; \forall \; i$. Therefore, the limiting factor that determines rate of convergence is $(1 - \hat{\eta} \frac{\lambda_j}{\lambda_1})^k$, which in turn is limited by the smallest eigenvalue factor: $\frac{\lambda_N}{\lambda_1}$.

Assuming $\frac{\lambda_N}{\lambda_1} \ll 1 \implies \hat{\eta} \frac{\lambda_N}{\lambda_1} \ll 1$ as $\hat{\eta} < 1 \implies (1 - \hat{\eta} \frac{\lambda_N}{\lambda_1})^k \approx 1 - k \hat{\eta} \frac{\lambda_N}{\lambda_1}$.

Hence the convergence time, $k^* = \mathcal{O}(\hat{\eta} \frac{\lambda_1}{\lambda_N}) = \mathcal{O}(\frac{\lambda_1}{\lambda_N})$

If $\lambda_i$ follows power law, i.e, $\lambda_i = c i^{-\alpha}$ and $\frac{\lambda_N}{\lambda_1} = N^{-\alpha}$ then $k^* = \mathcal{O}(N^\alpha)$ i.e. $k^*$ grows exponentially with $\alpha$. □

**Theorem A.3.** *Let $\hat{y} = \mathbf{f}_\theta(x)^T \psi$ be a linear regression problem as before. Let us further assume that $\mathbf{f}_\theta(x) \; \forall x \sim \mathcal{D}_{train}$ is a representative subset of the inputs from true data distribution: $(x, y) \sim \mathcal{D}$. Assuming a power-law distribution in the eigenspectrum of representations at $\mathbf{f}_\theta$, i.e. $\lambda_n = \frac{c}{n^\alpha} \quad \forall n \geq n^*$, where $n^* \in \{1, 2...N\}$, the generalization error after T weight update steps, $\mathcal{G}(T)$ is:*

$$\begin{aligned}
\mathcal{G}(T) := \mathbb{E}_{x,y \sim \mathcal{D}}[(y - \mathbf{f}_\theta(x)^T \psi)^2] \quad &\leq \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))\right) \\
\textit{where} \qquad r_{\hat{d}}(\Sigma(\mathbf{f}_\theta)) = \frac{\sum_{i=\hat{d}}^m \lambda_i}{\sum_{i=1}^m \lambda_i} &= \frac{\sum_{i=\hat{d}}^m i^{-\alpha}}{\sum_{i=1}^m i^{-\alpha}}
\end{aligned} \tag{13}$$

*Here, $m = min(N, d)$ is the rank of $\Sigma_N(\mathbf{f}_\theta)$.*

*Proof.* We start with the gradient of the regression loss function for a defined training set denoted by $(X_s, Y_s)$ in a vectorized notation where $X \in \mathbb{R}^{N \times d}$ and $Y \in \mathbb{R}^N$. For the brevity of notation let us

denote $\mathbf{f}_\theta(x)$ by $x$. The loss/error for the regression problem can be given by:

$$E = \frac{1}{2}(Y_s - X_s\phi)^T(Y_s - X_s\phi)$$

This gives gradient of weight vector as:

$$\implies \Delta\phi = -\eta\nabla_\phi E = \eta X^T Y - \eta X^T X\phi \tag{14}$$

where $\eta$ is the learning rate. Now, let the true labels be generated by $Y_s = X_s\phi^* + \epsilon$ where $\epsilon$ is a random variable denoting noise. Also, let $X_s = U_s \wedge^{\frac{1}{2}} V^T$ which gives $X_s^T X_s = V \wedge V^T$ and $X_s^T Y_s = V \wedge^{\frac{1}{2}} U_s^T Y = V \wedge^{\frac{1}{2}} \tilde{S}$ where $\tilde{S} = U_s^T Y = U_s^T(U_s \wedge^{\frac{1}{2}} V^T\phi^* + \epsilon) = \wedge^{\frac{1}{2}} Z^* + \tilde{\epsilon}$, $Z^* = V^T\phi^*$ and $\tilde{\epsilon} = U_s^T\epsilon$. Let $\phi = VZ$, i.e. expressing $\phi$ using the eigenbasis $V$ of $X^TX$ and $Z$ as the new parameters. This gives:

$$\Delta Z = V^T\Delta\phi = \eta V^T(V \wedge^{\frac{1}{2}} \tilde{S} + V \wedge V^T\phi) = \eta(\wedge^{\frac{1}{2}}\tilde{S} - \wedge Z) \tag{15}$$

$$\implies \Delta Z = \eta \wedge (Z^* - Z) + \eta \wedge^{\frac{1}{2}} \tilde{\epsilon} \tag{16}$$

This implies no mixing between the eigenmodes. Solving Eq. (16) we get:

$$Z_i^* - Z_i(t) = (Z_i^* - Z_i(0))e^{-\eta\lambda_i t} - \frac{\tilde{\epsilon_i}}{\sqrt{\lambda_i}}(1 - e^{-\eta\lambda_i t}) \tag{17}$$

Now, let $\phi^* - \phi = \delta$ and $Z^* - Z = \rho = V^T\delta$, which gives $\rho_i(t) = \rho_i(0)e^{-\eta\lambda_i t} - \frac{\tilde{\epsilon_i}}{\sqrt{\lambda_i}}(1 - e^{-\eta\lambda_i t})$. Then the generalization error is:

$$\mathcal{G} =< (\phi^* - \phi)^T X^T X(\phi^* - \phi) >=< \rho^T \wedge \rho >=< \sum_{i=1}^{m} \lambda_i\rho_i^2 >= \sum_{i=1}^{m} \lambda_i < \rho_i^2 > \tag{18}$$

where $X$ is the datapoints from the total dataset for which we care about the generalization error and $m$ is the rank of the covariance matrix, $X^TX$. Without loss of generality, we can assume $m = min(N, d)$. Note that $\langle . \rangle$ denotes the expectation over different network weight initialization and label noise. We define $X_s$ to be a reliable sample of dataset $X$ if both $X^TX$ and $X_s^TX_s$ have the same eigenvectors, i.e. $V$. Therefore, we could write $X^TX = X_s^TX_s$ in Eq. 18. Note that the reliability of the sample indicates the inherent structure of the data, i.e. the variance along the principal components in the data space have been captured by $X_s$. Subsequently, we get the expression for generalization error for a fixed training budget upto time/epoch (T):

$$\mathcal{G}(T) = \sum_{i=1}^{m} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 \tag{19}$$

where $\sigma_\rho$ indicates the noise due to weight initialization and $\sigma_\epsilon$ indicates the noise in labels. It is clear from Eq. (19) that directions corresponding to larger eigenvalues converge faster and contribute to the generalization error through second term. Let us, therefore, assume that the top $\tilde{d}$ eigenmodes have converged. Therefore, $e^{-\eta\lambda_i T} \leq c_0 \; \forall i \leq \tilde{d} < m$, where $c_0$ is some small number. Hence,

$$\begin{aligned} & e^{-\eta\lambda_i T} \leq c_0 \quad \forall i \leq \tilde{d} \quad \text{and} \quad e^{-\eta\lambda_i T} > c_0 \quad \forall i > \tilde{d} \\ \implies & e^{-2\eta\lambda_i T} \leq c_0^2 \quad \forall i \leq \tilde{d} \quad \text{and} \quad (1 - e^{-\eta\lambda_i T})^2 \leq (1 - c_0)^2 \quad \forall i > \tilde{d} \\ \text{Also} \quad & (1 - e^{-\eta\lambda_i T})^2 \leq 1 \quad \forall i \leq \tilde{d} \quad \text{and} \quad e^{-2\eta\lambda_i T} \leq 1 \quad \forall i > \tilde{d} \end{aligned} \tag{20}$$

Plugging inequalities from Eq. (20) in Eq. (19), we can upper bound the generalization error:

$$\begin{aligned} \sum_{i=1}^{\tilde{d}} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 &\leq \sigma_\rho^2 c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i + \sigma_\epsilon^2 \\ \sum_{i=\tilde{d}+1}^{m} \lambda_i\sigma_\rho^2 e^{-2\eta\lambda_i T} + \sigma_\epsilon^2(1 - e^{-\eta\lambda_i T})^2 &\leq \sigma_\rho^2 \sum_{i=\tilde{d}+1}^{m} \lambda_i + \sigma_\epsilon^2(1 - c_0)^2 \end{aligned} \tag{21}$$

Therefore, the generalization error can be upper bounded as follows:

$$\mathcal{G}(T) \leq \sigma_\rho^2 c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i + \sigma_\epsilon^2 + \sigma_\rho^2 \sum_{i=\tilde{d}+1}^{m} \lambda_i + \sigma_\epsilon^2(1 - c_0)^2 \tag{22}$$

Now, we can choose $c_0$ to be such that $c_0^2 \sum_{i=1}^{\tilde{d}} \lambda_i = \kappa_0$ where $\kappa_0$ is some constant independent of $T$. Furthermore, we impose the power-law assumption in the eigenvalues, i.e. $\lambda_i = ci^{-\alpha}$. With these assumptions, we can simplify Eq. (22) as follows:

$$\mathcal{G}(T) \leq \sigma_\rho^2 \kappa_0 + \sigma_\epsilon^2 \left(1 + (1 - c_0)^2\right) + \sigma_\rho^2 c \sum_{i=\tilde{d}+1}^{m} i^{-\alpha}$$

$$\implies \mathcal{G}(T) \leq \kappa + \sigma_\rho^2 c[var(X)] \frac{\sum_{i=\tilde{d}+1}^{m} i^{-\alpha}}{var(X)} = \kappa + \sigma_\rho^2 [var(X)] \frac{\sum_{i=\tilde{d}+1}^{m} i^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}} \tag{23}$$

where $\kappa$ is another constant (used for brevity), $\kappa = \sigma_\rho^2 \kappa_0 + \sigma_\epsilon^2 \left(1 + (1 - c_0)^2\right)$. Also, we used the relation: $var(X) = \sum_i \lambda_i = \sum_i ci^{-\alpha}$. Therefore, we can set $\hat{d} = \tilde{d} - 1$ and subsequently prove the statement of the theorem:

$$\mathcal{G}(T) \leq \kappa + \sigma_\rho^2 [var(X)] \frac{\sum_{i=\hat{d}}^{m} i^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}} = \mathcal{O}\left(r_{\hat{d}}(\Sigma(\mathbf{f}_\theta))\right) \tag{24}$$

□

## B Additional Experimental results

In this section we provide additional results, evidence to support our hypothesis of $\alpha$ being a useful measure of downstream generalization performance.

### B.1 Generalization & Eigenspectrum Decay on STL10 and ImageNet

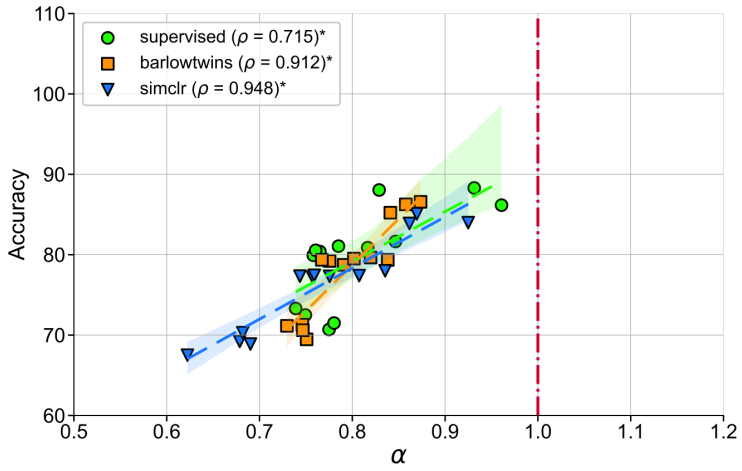



Figure 6: To measure downstream performance, one alternative is using non-linear readouts. With fixed feature-set, the performance of non-linear readout on STL-10 demonstrates correlation with $\alpha$ **(across different pretraining loss functions)** The trends are similar to the performance of a linear readout, as shown in Fig. 3. $^*p < 0.05$.

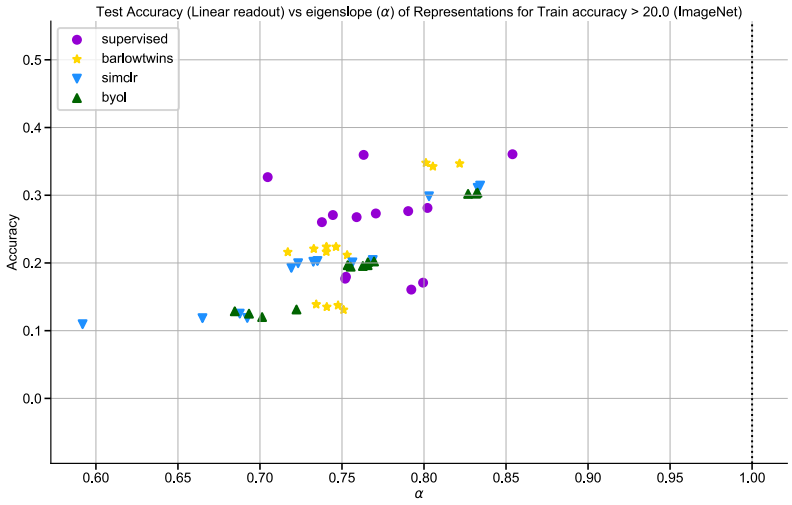



Figure 7: Unless specified explicitly, our models are pretrained on the ImageNet dataset. Here, we evaluate **in-distribution generalization** for ResNet-50 models pretrained with multiple learning objectives. We see $\alpha$ is correlated with the generalization on ImageNet. Note that the input images are downsampled for this experiment to get lower dimension features for intermediate layers, which facilitates the Eigendecomposition of the large covariance matrix for ImageNet. However, this is not a problem when we work with the features extracted from the final layers, which suggests that the model selection algorithm proposed in this paper also works for high-resolution images.

### B.2 Layerwise evaluation on STL 10 and MIT 67

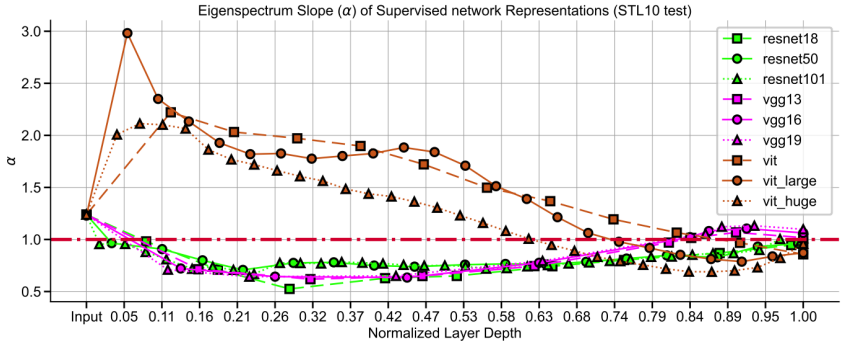



Figure 8: $\alpha$ for intermediate layer representations from different backbone architectures demonstrates the contrasting representations learned by CNNs and ViT.

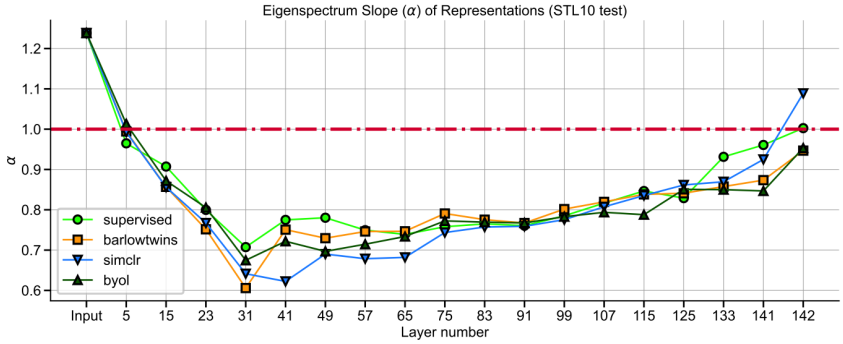



Figure 9: $\alpha$ for intermediate layer representations from networks trained using different loss functions show similar trends. Representations from deeper layers exhibit $\alpha$ closer to 1 as compared to middle layer representations.

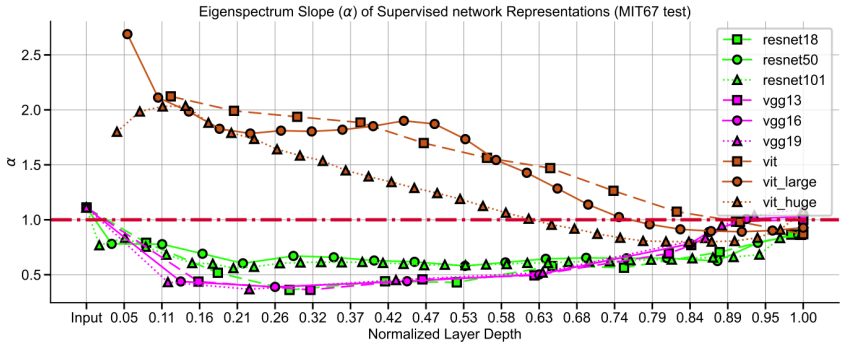



Figure 10: $\alpha$ for intermediate layer representations from different backbone architectures in MIT67. Representations learned by ViT is qualitatively different from those is CNNs both in object and scene recognition datasets.

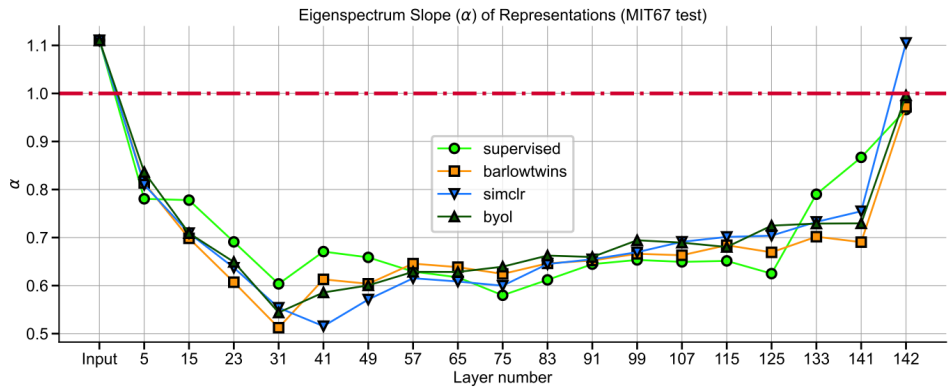



Figure 11: $\alpha$ for intermediate layer representations from networks trained using different loss functions show similar trends in MIT67. Empirical evaluations suggest that representations from deeper layers exhibit $\alpha$ closer to 1 as compared to intermediate layer representations.

### B.3 Visualizing design landscape for BarlowTwins

As a diagnostic metric for measuring representation quality, we chart the learning landscape for self-supervised learning algorithms (Barlow Twins here). To this effect, we empirically investigate the role of different critical hyperparameters in learnability and generalization. In particular, we ablate across projection-dimensionality, redundancy coefficient, weight-decay and learning rate.

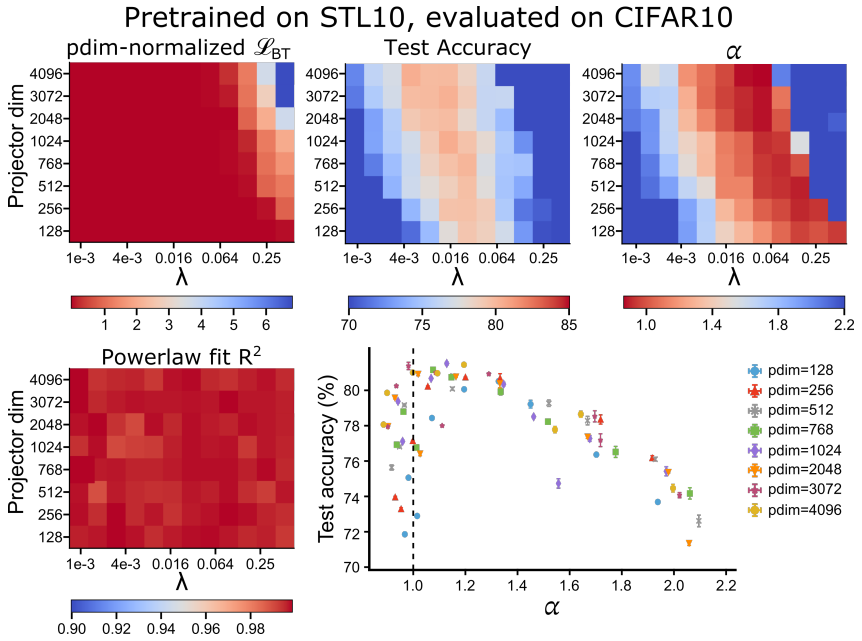



Figure 12: For ResNet-50 model architecture, we measure out-of-distribution generalization for BarlowTwins pretrained on STL10 and evaluated with linear-probes on CIFAR10.

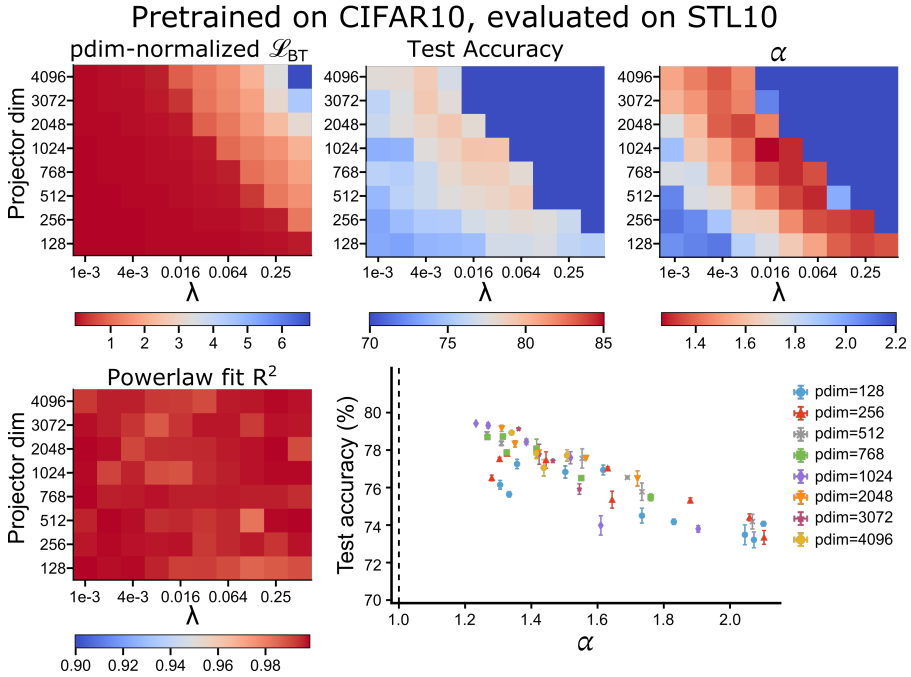



Figure 13: For ResNet-50 model architecture, we measure out-of-distribution generalization for BarlowTwins pretrained on CIFAR10 and evaluated with linear probes on STL10.

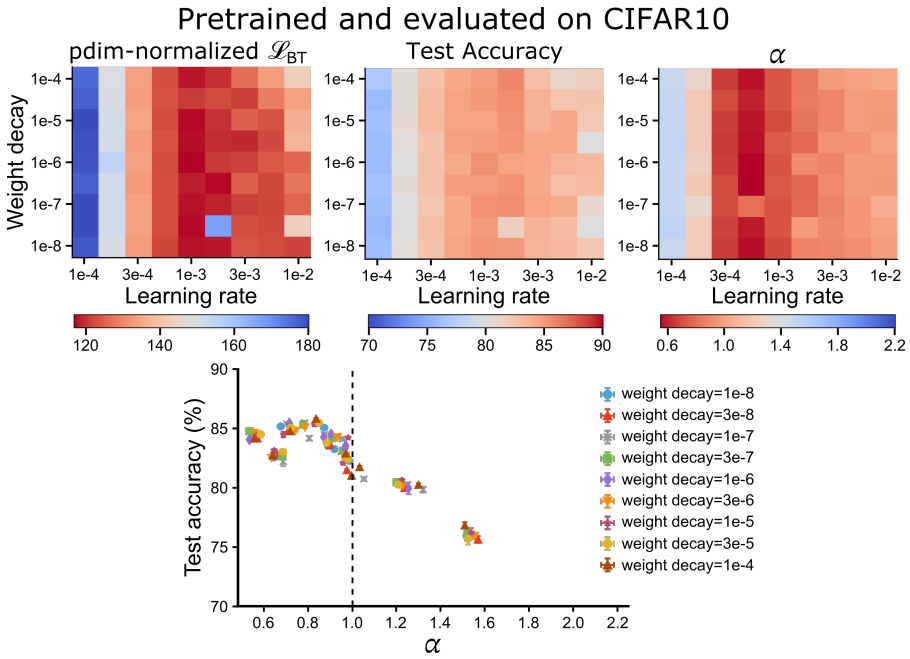



Figure 14: Measuring in-distribution generalization for BarlowTwins trained on CIFAR10 and evaluated on CIFAR10 when sweeping over different optimization hyperaprameters, namely learning rate and weight decay.

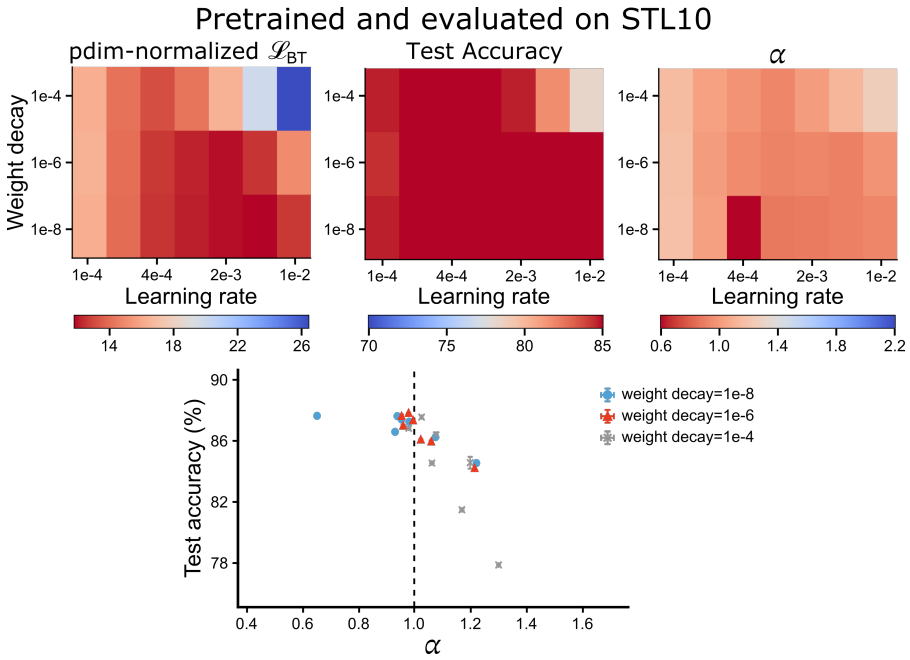



Figure 15: Measuring in-distribution generalization for BarlowTwins trained on STL10 and evaluated on STL10 when sweeping over different optimization hyperaprameters, namely learning rate and weight decay.

### B.4 Visualizing design landscape for SimCLR

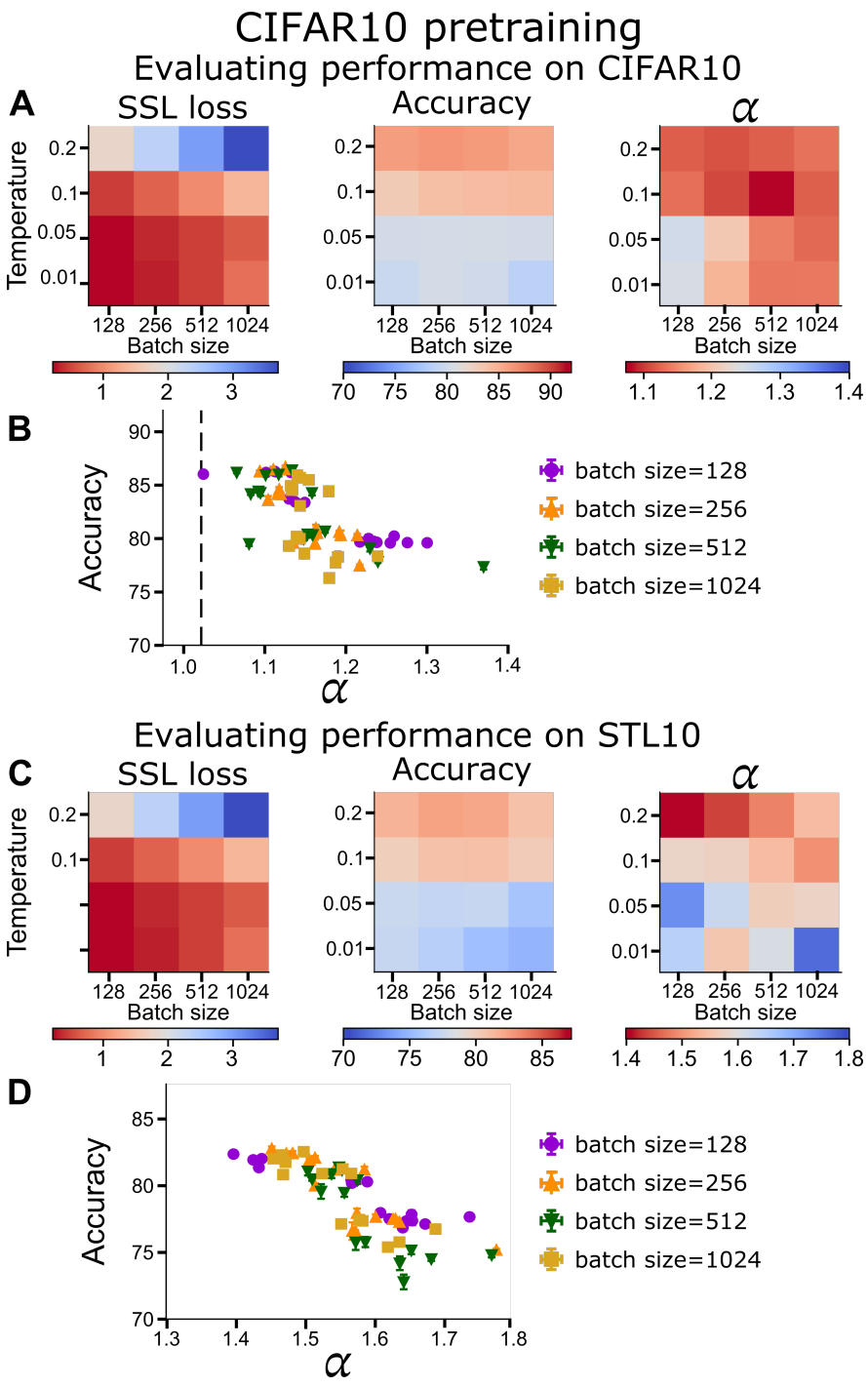



Figure 16: Hyperparameter sweep for SimCLR pretraining on CIFAR10: the correspondence between $\alpha$ and downstream accuracy holds for contrastive SSL methods as well. (A) SimCLR loss, (in-distribution) classification accuracy on CIFAR10 and corresponding $\alpha$ of learned representations when sweeping over different values of temperature and batch size, two key hyperparameters for SimCLR training. (B) Accuracy vs $\alpha$ plot for all models trained with different values of temperature, batch size and projector dimensionality. (C-D) Same as A and B for downstream (out-of-distribution) classification accuracy evaluated on STL10.

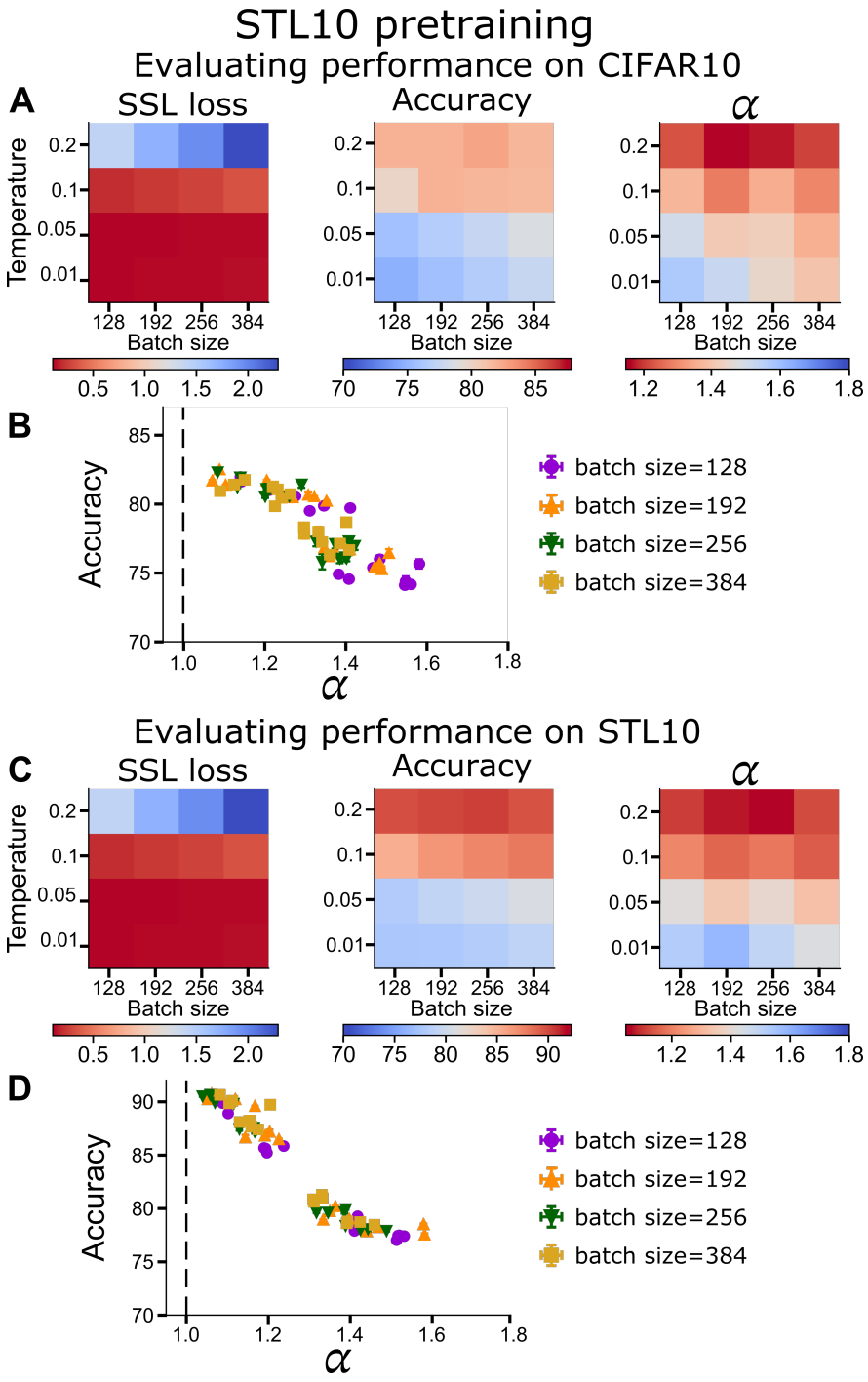



Figure 17: Hyperparameter sweep for SimCLR pretraining on STL10: the correspondence between $\alpha$ and downstream accuracy holds for contrastive SSL methods as well. (A) SimCLR loss, (out-of-distribution) classification accuracy on CIFAR10 and corresponding $\alpha$ of learned representations when sweeping over different values of temperature and batch size. (B) Accuracy vs $\alpha$ plot for all models trained with different values of temperature, batch size and projector dimensionality. (C-D) Same as A and B for downstream (in-distribution) classification accuracy evaluated on STL10.

### B.5 Model Selection on Compute Budget

**Algorithm 2** Model selection using $\alpha$

```
# M: Number of models that can be trained in parallel
# H: Number of sequential steps of model training
# K: Number of top models to log in memory for model selection

α_min = 0
α_max = ∞
models_dict = {}

for s in range(H):
    # train M models in parallel, each with different hyperparam config
    trained_models = train_SSL_models(num_parallel_models=M)

    # evaluate α for the trained models
    alpha_trained_models = evaluate_alpha(trained_models)

    # choose models based on alpha to run linear evaluation
    for model in trained_models:
        if alpha_trained_models[model] ∈ [α_min,α_max]:
            model_alpha = alpha_trained_models[model]
            model_performance = run_linear_eval(model)
            models_dict.insert({model:(model_alpha,model_performance)})
        else:
            pass

    # Update α_min and α_max
    α_min = max({α_m | α_m > α_j & performance_m ≥ performance_j ∀ m,j ∈ models_dict})
    α_max = min({α_m | α_m < α_j & performance_m ≥ performance_j ∀ m,j ∈ models_dict})

    # Trim models_dict to keep only top K models
    performance_thresh = get_top_K_model_performance(models_dict)
    for model in models_dict:
        if model.performance < performance_thresh:
            models_dict.pop(model)

# Return best model performance from the 'selected' models
best_model_performance = max({performance_m ∀ m ∈ models_dict})
return best_model_performance
```

### B.6 Average complexity analysis of Algorithm 1

Let us assume that the compute budget is characterized by M, i.e. the number of SSL models that can be trained parallely on the compute infrastructure. Let us assume H such parallel steps are run sequentially in order to train $M \times H$ different model configurations (e.g. sweeps over different hyperparameter configurations). Let us denote the probability of performing linear evaluation on a model trained in the $r^{th}$ sequential step as $\epsilon_r$. Therefore, expected number of linear evaluations in the $r^{th}$ step will be $M\epsilon_r$. Now, clearly $\epsilon_r \leq 1$, and hence:

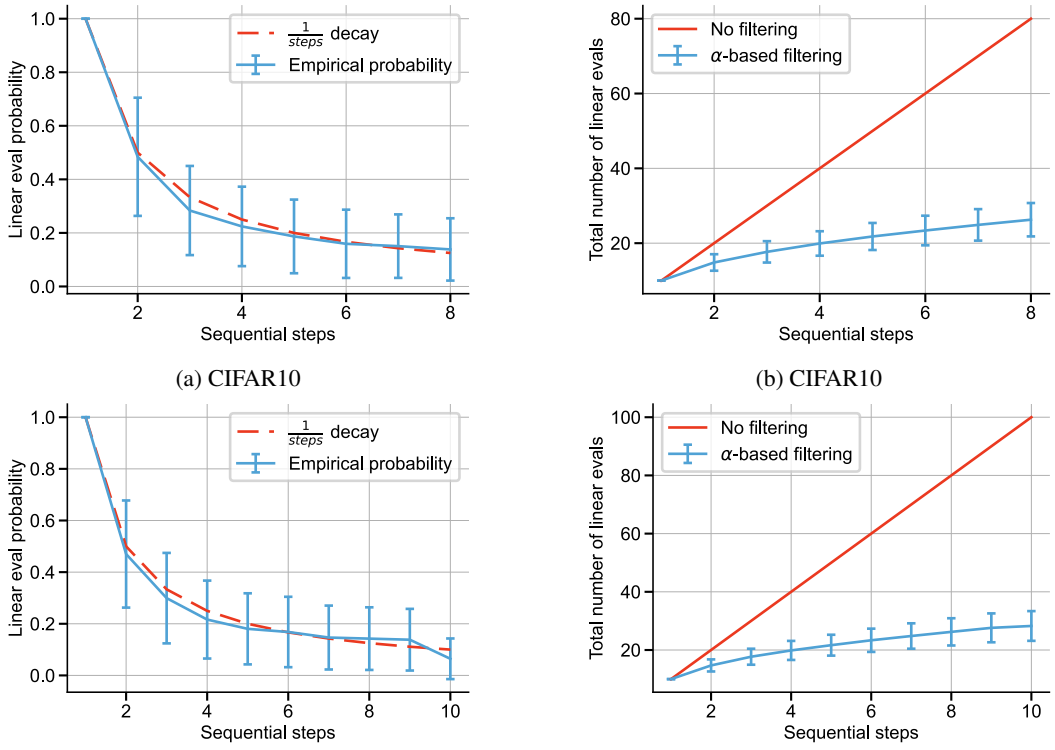
$$\mathbb{E}[\text{linear\_evals}] = \sum_{r=1}^{H} M\epsilon_r \leq \sum_{r=1}^{H} M = MH \tag{25}$$

This relationship provides the worst case complexity of Algorithm 2. We can further assume that the probability of linear evaluation decays with each sequential step. Therefore, the average number of linear evaluations would be less than $MH$. Assuming $\epsilon_r = \mathcal{O}\left(\frac{1}{r}\right)$ [†], we can write:

$$\mathbb{E}[\text{linear\_evals}] = \sum_{r=1}^{H} M\epsilon_r = \sum_{r=1}^{H} M\mathcal{O}\left(\frac{1}{r}\right) = \mathcal{O}\left(Mlog(H)\right) \tag{26}$$

Taken together, the above equation shows that the average case complexity of model evaluation using $\alpha$ is less than the standard complexity of model evaluation, i.e. $MH$ linear evaluations.

[†] Verified in Fig. 18




(a) CIFAR10

(b) CIFAR10

(c) STL10

(d) STL10

Figure 18: Our analysis suggests a range $[\alpha_{min}, \alpha_{max}]$ for $\alpha$ which indicates the best model in a set of pretrained models. If this range is known, using $\alpha$ is significantly more compute efficient that standard exhaustive search which requires multiple linear evaluations, as noted in 1. Furthermore, an effective search algorithm using offline-metric $\alpha$ allows us to reduce the number of linear evaluations from $\mathcal{O}(MH)$ to $\mathcal{O}(M \log H)$, assuming that the likelihood of linear evaluation for models decays as $\frac{1}{steps}$. This assumption is verified in (a,c)

## C On necessary and sufficiency conditions for predicting generalization

Understanding the correlation between $\alpha$ and *generalization* requires probing the natural question of whether $\alpha$ in a given range is sufficient for good downstream performance. To further elaborate on this question, consider the following: Let us suppose that we have our ideal representation space that exhibit alpha within the Goldilocks zone. If we perform a set of permutation operations on individual datapoint representations, the structure of the representation space remains the same but the mapping from representation to label is now destroyed. In doing this set of permutation operations, we have now arrived at a different set of representations that would exhibit the same (or similar) alpha but demonstrate a lower task performance when measured using a linear readout layer. Extending the conclusions of this thought experiment to our observations, it is clear that we could have models with similar alpha values but distinctly different performance. But an alpha value that is not in the Goldilocks zone would be associated with inferior model performance. This property is the core of our claim and allows us to propose alpha as a measure for model selection in SSL pipelines.

## D High-Resolution plots

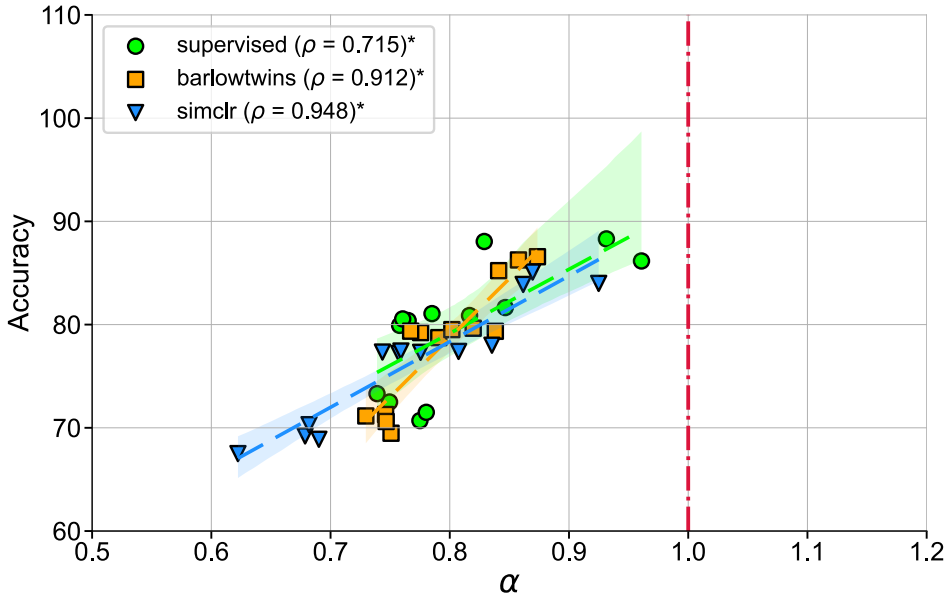



Figure 19: The x-axis represents estimated $\alpha$, y-axis is evaluating linear-probes on STL-10. Different models correspond to pretraining on ImageNet with different learning objectives (we fix the architecture to ResNet-50).

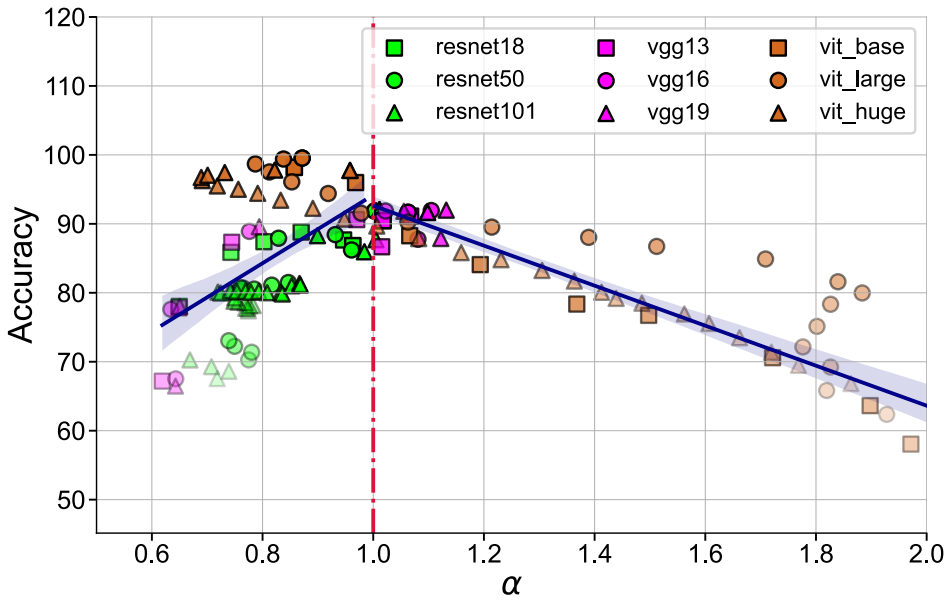



Figure 20: The x-axis represents estimated $\alpha$, y-axis is evaluating linear-probes across different layers across multiple architectures on STL10. All models are pretrained with supervised learning on ImageNet (we fix learning objective to be cross-entropy).

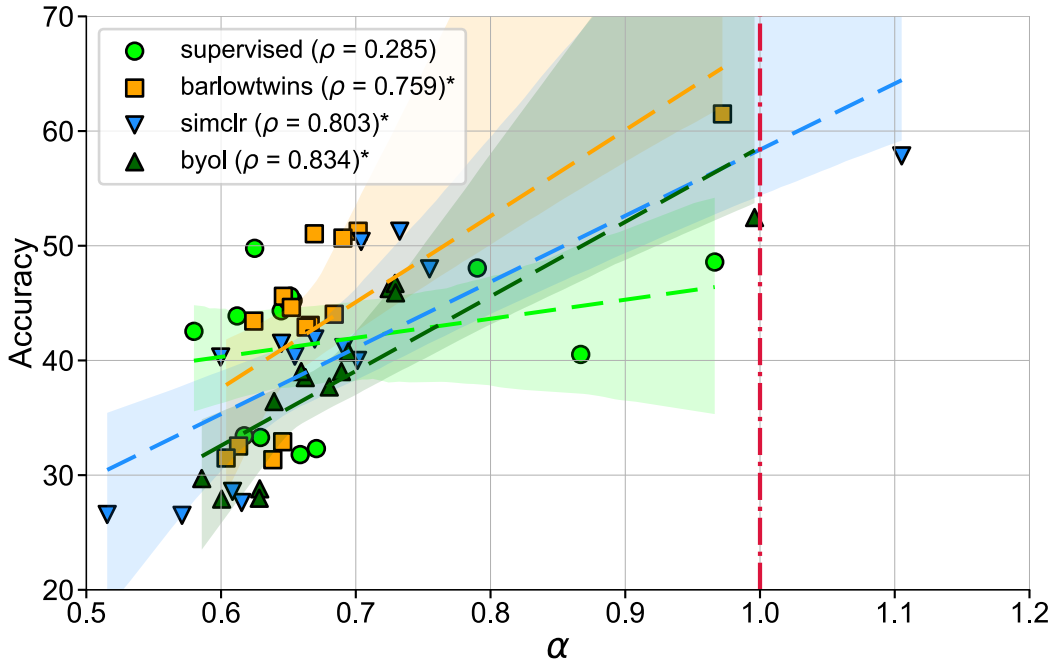



Figure 21: The x-axis represents estimated $\alpha$, y-axis is evaluating linear-probes on MIT-67 (scene recognition). Different models correspond to pretraining on ImageNet with different learning objectives (we fix the architecture to ResNet-50).

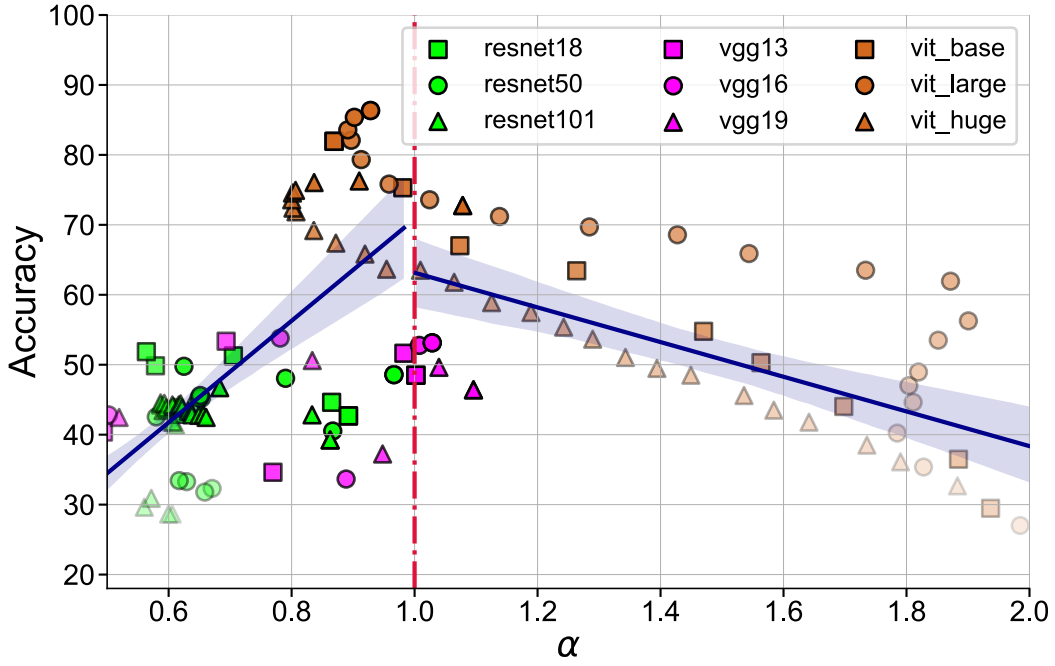



Figure 22: The x-axis represents estimated $\alpha$, y-axis is evaluating linear-probes across different layers across multiple architectures on MIT-67 (scene recognition). All models are pretrained with supervised learning on ImageNet (we fix learning objective to be cross-entropy).