ADADIM: DIMENSIONALITY ADAPATION FOR SSL REPRESENTATIONAL DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

A key factor in effective Self-Supervised learning (SSL) is preventing dimensional collapse, where higher-dimensional representation spaces (R) span a lower-dimensional subspace. Therefore, SSL optimization strategies involve guiding a model to produce R with a higher dimensionality ($H(R)$) through objectives that encourage decorrelation of features or sample uniformity in R. A higher $H(R)$ indicates that R has greater feature diversity which is useful for generalization to downstream tasks. Alongside dimensionality optimization, SSL algorithms also utilize a projection head that maps R into an embedding space Z. Recent work has characterized the projection head as a filter of noisy or irrelevant features from the SSL objective by reducing the mutual information $I(R; Z)$. Therefore, the current literature's view is that a good SSL representation space should have a high $H(R)$ and a low $I(R; Z)$. However, this view of SSL is lacking in terms of an understanding of the underlying training dynamics that influences the relationship between both terms. For this reason, we directly oppose the current literature's view of SSL representation spaces and instead assert that the best performing R is one arrives at an ideal balance between both $H(R)$ and $I(R; Z)$. Our findings reveal that increases in $H(R)$ due to feature decorrelation at the start of training lead to correspondingly higher $I(R; Z)$, while increases in $H(R)$ due to samples distributing uniformly in a high-dimensional space at the end of training cause $I(R; Z)$ to plateau or decrease. Furthermore, our analysis shows that the best performing SSL models do not have the highest $H(R)$ nor the lowest $I(R; Z)$, but effectively arrive at a balance between both. To take advantage of this analysis, we introduce AdaDim, a training strategy that leverages SSL training dynamics by adaptively balancing between increasing $H(R)$ through feature decorrelation and sample uniformity as well as gradual regularization of $I(R; Z)$ as training progresses. We show performance improvements of up to 3% over common SSL baselines despite our method not utilizing expensive techniques such as queues, clustering, predictor networks, or student-teacher architectures.

1 INTRODUCTION

Self-supervised learning (SSL) (44) algorithms approach or surpass fully supervised strategies on a wide variety of benchmark tasks (8; 7; 14; 50; 3; 9). SSL optimization generally involves an invariance loss that ensures the representations of similar samples align with each other and a mechanism to prevent dimensional collapse (21). Dimensional collapse refers to the phenomena where high dimensional representations span a lower-dimensional subspace. Therefore, to prevent dimensional collapse, a wide variety of works (17; 2; 41) suggest that good SSL representations (R) have a higher overall dimensionality. In this work, we analytically measure dimensionality of the representation $H(R)$ through the effective rank metric (35). Effective rank quantifies the distribution of singular values of R and provides a matrix approximation of dimensionality (46; 34). In practice, optimizing for higher dimensionality is either done through a dimension contrastive approach (18) that encourages feature decorrelation or through a sample-contrastive method that promotes a uniform spread of sample representations (47). Alongside a term to promote dimensionality, all SSL methods utilize a projection head that maps R into a lower dimensional embedding space Z where the SSL optimization objective is applied. Recent work (32) has characterized the purpose of the projection head as a filter that removes spurious features thus lowering the mutual information $I(R; Z)$. In

general, lower $I(R;Z)$ reflects representations varying only in feature directions that correspond well with task-relevant semantic concepts, while higher $H(R)$ corresponds to a greater degree of feature diversity. Together, these works imply that a good SSL representation space should have a high dimensionality $H(R)$ and low $I(R;Z)$.

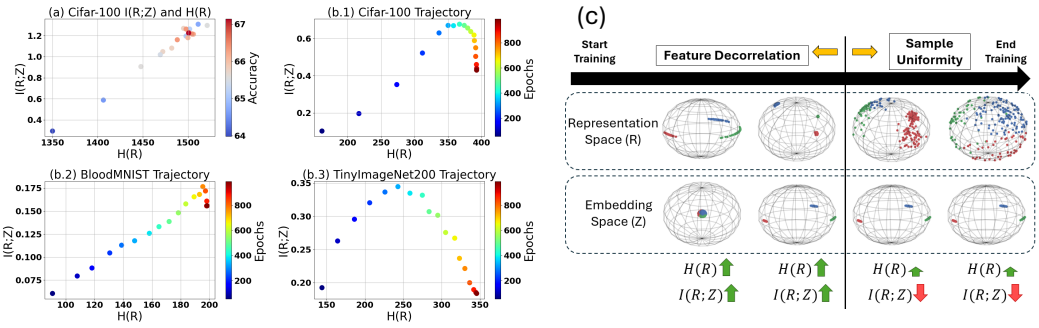

Figure 1: a) This figure shows how performance varies for 20 different pre-trained ResNet-50 models as a function of $H(R)$ and $I(R;Z)$. b) The first three figures show how $H(R)$ and $I(R;Z)$ vary across training of a ResNet-18 encoder with SimCLR (7) for 1000 epochs on three different datasets. c) This toy graphic shows how the representation space (R) and embedding space (Z) of a 3D dataset changes when following SSL training dynamics. We also demonstrate how these changes effect $H(R)$ and $I(R;Z)$.

However, this view of SSL is lacking in terms of an understanding of the underlying training dynamics that influences the relationship between both terms. For example, in part a) of Figure 1, we show how the final $H(R)$ and $I(R;Z)$ arrived at the end of training influences downstream performance. In this Figure, we train 20 different models with slightly different hyperparameters with a ResNet-50 (20) model for 400 epochs on Cifar-100. We find that the best performing models are not the ones with the highest $H(R)$ or lowest $I(R;Z)$, but instead approach a specific $H(R)$ and $I(R;Z)$ value where downstream performance is maximized. **Thus, our first claim is that the best performing SSL representations arrive at a balance between both $H(R)$ and $I(R;Z)$ such that there is enough feature diversity for the task of interest, but not so much that R contains irrelevant noise.** This claim directly opposes existing literature (17; 2; 41) that only considers the $H(R)$ value reached at the end of training as an indicator of downstream model performance.

In this work, we also analyze the representational dynamics that cause this behavior. In parts b.1) - b.3) of Figure 1 we show how $H(R)$ and $I(R;Z)$ evolve over the course of SimCLR (7) training on a ResNet-18 model for 1000 epochs across 3 distinct datasets. While $H(R)$ generally increases throughout training, as expected by the current literature, $I(R;Z)$ does not directly decrease and instead goes through distinct phases of increasing, plateauing, and decreasing. In part c), we show a toy example to visualize the dynamics causing this behavior. In this Figure, we have 200 samples distributed within a fictitious 3D spherical representation space. At the start of training, $H(R)$ increases by projecting R onto a higher dimensional space by mapping from a 2D plane to the surface of the sphere. Z correspondingly projects from a 1D to 2D space. This phase corresponds to feature decorrelation where both R and Z increase the number of dimensions in which they vary which causes $I(R;Z)$ to increase as both spaces are projecting to a higher dimension. However, later in training, $H(R)$ starts having fewer dimensions in which to project into and further increases in $H(R)$ are caused by samples distributing uniformly within the space that it arrives at. This change in sample spread is not reflected to the same degree in Z which causes $I(R;Z)$ to decrease. Thus, our **second claim is that feature decorrelation at the start of training leads to higher $I(R;Z)$, while samples uniformly spreading across higher dimensions at the end of training causes $I(R;Z)$ to plateau or decrease.**

Based on our first two claims, we propose an SSL training strategy called `AdaDim`. `AdaDim` takes advantage of the discussed training dynamics to adaptively balance increasing $H(R)$ through feature decorrelation and sample uniformity as well as gradual regularization of $I(R;Z)$ as training progresses. This adaptation is done in a manner that is specific to the dimensionality characteristics of the dataset of interest. This method implies our **third claim which is SSL optimization objectives**

should be constructed to allow adaptation to the evolving dynamics of their representation space.

1. We theoretically and empirically demonstrate that the relationship between $H(R)$ and $I(R;Z)$ can characterize SSL training dynamics through both a gaussian and information theoretic analysis.
2. We show that the best performing SSL models use the discussed dynamics to arrive at an ideal balance for both $H(R)$ and $I(R;Z)$ by the end of training and empirically demonstrate this behavior across a wide variety of data settings.
3. We develop a dimension adaptive (`AdaDim`) method that exploits our discovered training dynamics to arrive at a better balance between $H(R)$ and $I(R;Z)$. We demonstrate performance improvements across a wide variety of data settings and in comparison with state of the art methods without needing expensive training techniques such as queues, clustering, predictor networks, or student-teacher architectures.

2 Related Works

SSL Methods (18) categorizes SSL methods as dimension-contrastive or sample-contrastive. Sample contrastive methods involve enforcing sample uniformity by projecting sample augmentations (positives) closer to each other than that of other samples in a batch (negatives) (7). Other methods are derived from simple alterations to the definition of positive and negative sets. Research directions include using a momentum queue (8), using nearest neighbors as positives (14), enforcing cluster assignments (5), enforcing hierarchical structures (29; 26), and using label information (23). Dimension contrastive approaches enforce feature decorrelation through various methods. Examples of methods include regularizing the embedding covariance matrix (3; 50; 15) or introducing architectural constraints (9; 19; 6) that implicitly regularize the dimensions. Our method differs due to the introduction of an adaptive mechanism to interpolate between both sample and dimension contrastive approaches and $I(R;Z)$ based on SSL training dynamics.

Understanding SSL Training Dynamics A subset of works have also attempted to understand the training dynamics of SSL models. (21) analyzed the dimensional collapse phenomenon within contrastive learning settings. (38) explored the idea that SSL training dynamics involve learning one eigenvalue at a time. (42; 39) analyzed the learning dynamics of dimension contrastive methods in the context of simple linear networks. In general, there is a much more in depth literature for understanding training dynamics within supervised settings (1; 16; 37) while SSL understanding is relatively more limited. Our work attempts to understand SSL through the lens of training dynamics that influence the relationship between $I(R;Z)$ and $H(R)$.

3 Analysis of Training Dynamics

3.1 Simulated Training Dynamics

Through the analyses of this section, we find that increases in $H(R)$ due to feature decorrelation cause a corresponding increase in $I(R;Z)$ while increases in $H(R)$ due to sample uniformity cause $I(R;Z)$ to plateau or decrease. To investigate these dynamics between a representation space and its projection, we perform a simulation within a Gaussian setting. Assume that the Gaussian distributed data is represented by $R \sim \mathcal{N}(\mu_R, \Sigma_R)$ where $R \in \mathcal{R}^m$. Additionally, assume that there is some projection of R represented by $Z \sim \mathcal{N}(\mu_Z, \Sigma_Z)$ where $Z \in \mathcal{R}^n$ such that $n < m$. R and Z form a jointly multivariate normal distribution. Together, this distribution is defined by a block covariance matrix of the form $\Sigma = \begin{bmatrix} \Sigma_Z & \Sigma_{ZR} \\ \Sigma_{RZ} & \Sigma_R \end{bmatrix}$. In this setting, the closed form solution for $I(R;Z) = \frac{1}{2}(ln(|\Sigma_R|) + ln(|\Sigma_Z|) - ln(|\Sigma|))$. Applying Shur's complement to the block covariance matrix results in the following equation when all covariance matrices are invertible:

$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_Z|) - ln(|Var(Z|R)|)) = \frac{1}{2}(ln(|\Sigma_R|) - ln(|Var(R|Z)|)) \tag{1}$$

In equation 1, $Var(Z|R) = \Sigma_Z - \Sigma_{RZ}\Sigma_R^{-1}\Sigma_{ZR}$ and $Var(R|Z) = \Sigma_R - \Sigma_{ZR}\Sigma_Z^{-1}\Sigma_{RZ}$. The details of this derivation can be found in Section B.2. From this construction of the problem, several trends emerge. $I(R;Z)$ will increase or decrease depending on the relationship that the projection produces between R and Z. Specifically, $I(R;Z)$ will increase when the variance of the space of interest increases while its corresponding conditional variance remains relatively lower. These variance changes can occur through a larger number of features or through a more uniform spread of data samples. Figure 2 demonstrates a simulation of the effect of each by generating a synthetic gaussian dataset with 1000 samples, a defined variance for each of 5 generated clusters, and a defined number of features $m > 10$ to simulate R. This data is then projected with PCA to generate Z with either 2 components or 10 components. This design choice is to simulate the difference between early and late stage SSL training. Early in training, R and Z project closer to each other which is represented by the 10 component Z space while later in training R and Z diverge to a greater degree represented by the 2 component projection.

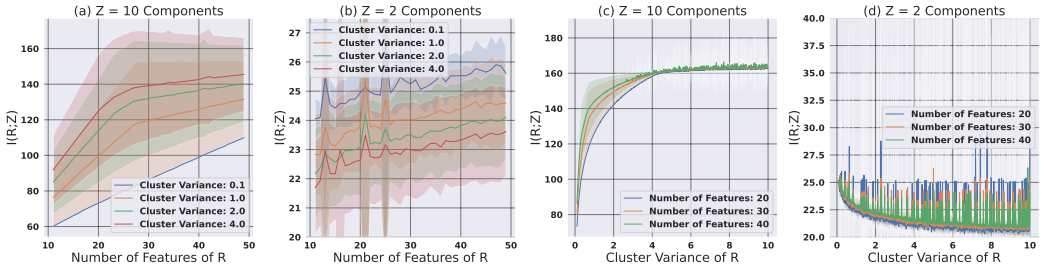

Figure 2: a) and b) show how $I(R;Z)$ in a gaussian setting changes as the number of features of R is increased. c) and d) show how $I(R;Z)$ varies as the sample cluster variance increases.

In Figure 2 a) and b), the $H(R)$ is increased through increasing the number of generated features while the cluster variance is kept constant. This corresponds to the feature decorrelation setting. The second experiment in Figure 2 c) and d) involves varying the sample variance while keeping the number of features fixed which corresponds to the setting where the sample uniformity changes between spaces. Note that within this Gaussian setting, PCA serves as a representative projection due to most of the information content of this data being represented by the variance parameter of an m-dimensional Gaussian. However, the same simulation is repeated in Section B.4 with the projector replaced with a small neural network. Further details of these experiments can be found in Section B.3. In parts a) and b), for different cluster variance values, increasing the number of features in R corresponds to an increase in $I(R;Z)$ regardless of the degree of projection. In parts c) and d), the behavior of $I(R;Z)$ varies significantly based on the degree of the projection. For the 10 component projection case, increasing the sample variance initially increases $I(R;Z)$, but it gradually plateaus as the sample variance increases further. This suggests that the projection cannot capture the variance along certain dimensions after a specific point. In part d), in the 2 component case, increasing the sample variance by any amount reduces $I(R;Z)$. **Overall, this Figure shows that $I(R;Z)$ increases with a greater number of decorrelated features in R regardless of the degree of the projection. In contrast, $I(R;Z)$ increases, plateaus, or decreases based on the degree of sample variance and projection from space m to n.** The exact choice of SSL optimization objective and training procedures will influence the degree to which $H(R)$ and $I(R;Z)$ increases or decreases, but the underlying representational dynamics will reflect our analysis.

Another important consideration is how the $H(R)$ and $I(R;Z)$ arrived at the end of training influences the subsequent performance of the model. To model this, the SSL information flow can be described by: $Y \to X \to R \to Z \to T$. Y represents the semantic concept associated with the data X. T represents the associated SSL task. The end goal of the SSL objective is to maximize $I(Y;R)$ which is the mutual information between the semantics of the data and the representation space. Recent work (32) showed that this information flow results in an upper bound on $I(Y;R)$:

$$I(Y;R) \leq I(Y;Z) - I(R;Z) + H(R) \tag{2}$$

Our objective is to show how this bound is effected by the training dynamics discussed in Section 3.1 and to show that simply reducing $I(R;Z)$ and increasing $H(R)$ to maximize this bound is not possible given these dynamics. Again it is assumed that R and Z are drawn from a joint multivariate Gaussian distribution. Furthermore, $I(Y;Z)$ is assumed to approach some constant G to isolate the analysis with respect to $I(R;Z)$ and $H(R)$. The justification for this term acting as a constant is from previous analyses (36) that assumed the information shared between semantic labels and the target SSL task can be regarded as a constant. Equation 2 can then be rewritten as:

$$I(Y;R) \leq G + \underbrace{\frac{1}{2}(ln(|\Sigma_R|) - ln(|\Sigma_Z|))}_{K(Both)} + \underbrace{\frac{1}{2}ln(|Var(Z|R)|)}_{V(I(R;Z))} + \underbrace{\frac{m}{2}(ln(2\pi)+1)}_{D(H(R))} \tag{3}$$

Equation 3 suggests that the bound on $I(Y;R)$ can be decomposed into three terms: a variance differential term K, a conditional variance term V, and a total dimension term D. The derivation of this bound is shown in Section B.5. Each term is labeled by its effect on $I(R;Z)$ or $H(R)$. Ideally, increasing each of these terms together would result in a higher overall bound on $I(Y;R)$. However, the SSL training dynamics discussed in Section 3.1 leads to the emergence of a dynamical system where increasing one of these terms can potentially limit the growth of others. For example, if $H(R)$ increases via feature decorrelation then D will increase due to a greater number of features m, but V will decrease due to corresponding feature decorrelation in the projection Z causing $I(R;Z)$ to correspondingly increase and limit the upper bound in equation 2. Additionally, K will be limited in this setting due to both of its terms increasing together. However, if $H(R)$ increases due to sample uniformity, then D is fixed in the number of dimensions which acts as a bound on how large $H(R)$ can grow. In contrast, K and V increase due to an increase in the variance of R without the projection Z having a corresponding increase in variance which lowers $I(R;Z)$. **This oscillatory behavior between each of these terms suggests that the downstream performance represented by $I(Y;R)$ cannot be maximized by optimizing for each term individually and requires a procedure that adaptively finds a balance between the two.**

3.2 Empirical Dynamics

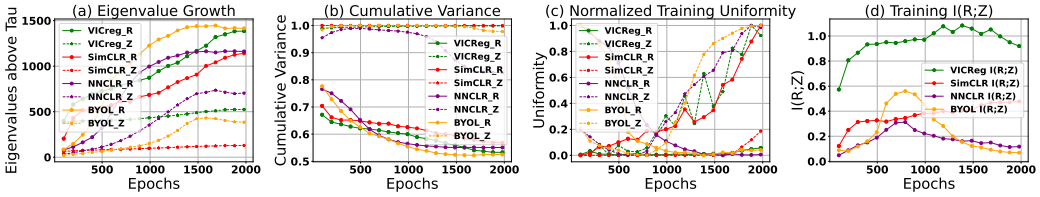

Figure 3: This is an analysis of 4 different SSL models R and Z space trained for 2000 epochs on Cifar-100 with ResNet-50. This analysis includes a) the number of eigenvalues above a threshold of $\tau = .01$, b) the cumulative explained variance ratio for top 30% of eigenvalues, c) the uniformity of each space, and d) $I(R;Z)$.

To verify the dynamics discussed theoretically in Section 3.1, an empirical analysis within a real SSL setting is shown in Figure 3. This experiment involves training a ResNet-50 model (20) with 4 different SSL methods for 2000 epochs on Cifar-100. The projector is designed such that R and Z both have 2048 features. In part a), we analyze the evolution of feature decorrelation for both the R and Z space across training by performing a count of the number of eigenvalues above a threshold $\tau = .01$. It is interesting to note that for the R space the number of eigenvalues consistently increases until late in training while the Z space has a more pronounced plateauing behavior earlier in training. This shows the behavior that the overall dimension of both spaces diverges from each other during training. In part b), we analyze the uniformity of eigenvalues by measuring what percentage of the variance in the space of interest is represented by the top 30% of eigenvalues. This is known as the cumulative explained variance ratio (22). We observe that the cumulative explained variance of R for all methods decreases during training which indicates that $H(R)$ is increasing due to a more uniform spread of eigenvalues and will gradually depend more on sample uniformity as training progresses. However, in Z, this metric is near 1.0 for all epochs of training which means that most

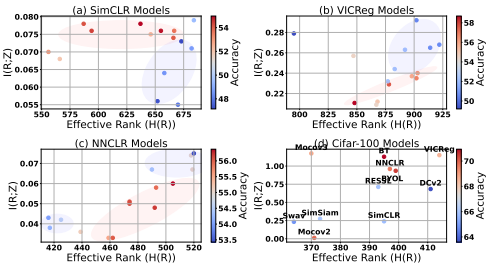

Figure 4: In Figures a), b), and c), the $H(R)$ and $I(R;Z)$ across 15 ResNet-50 models trained with randomized hyperparameters with 3 different SSL strategies are shown. In Figure d), we show the same plot across 11 different SSL methods trained on ResNet-18 for 1000 epochs.

Magnitude of Correlation with Performance Across Trained Models

Method	Dataset	Epochs	# of Models	ER (H(R))	I(R;Z)	Ratio
SimCLR	Cifar100	100	15	.082	.323	**.462**
VICReg	Cifar100	100	15	.013	**.772**	.751
NNCLR	Cifar100	100	15	.206	.229	**.337**
All-ResNet18	Cifar100	1000	11	.029	.372	**.375**
SimCLR	Cifar100	400	10	.557	.543	**.625**
VICReg	Cifar100	400	10	.351	.875	**.894**
SimCLR	TinyImageNet200	400	10	**.534**	.507	.521
SimCLR	Cinic-10	400	10	.029	.323	**.421**
SimCLR	Cifar-10	400	10	**.873**	.841	.833
SimCLR	OrganSMNIST	400	10	.0024	.435	**.442**

Table 1: This table shows the pearson correlation coefficient between the performance of a set of SSL models trained with different hyperparameters on a specific dataset and the effective rank ($H(R)$), $I(R;Z)$, and the ratio between them.

of the variance of Z is contained within only a small number of top eigenvalues. This suggests that samples in Z distribute uniformly along a restricted subset of dimensions which is in contrast to the behavior of space R that tries to distribute uniformly on as many dimensions as possible. This discrepancy in sample uniformity can also be visualized in part c) with the uniformity metric (47). We observe that for all SSL methods the uniformity between both spaces diverges from each other as training progresses. This divergent behavior is further confirmed in part d), where $I(R;Z)$ is measured with a matrix mutual information estimator (51) that increases at the start of training, but gradually decreases for every method later in training.

We also empirically verify how this relationship between $H(R)$ and $I(R;Z)$ impacts the downstream performance in Figure 4. In parts a), b), and c) we train 15 different models with randomized hyperparameters specific to 3 different SSL methods on Cifar-100 for 100 epochs each. We observe that for each method, the best performing models cluster around specific $H(R)$ and $I(R;Z)$ values. This trend also holds in part d), where every one of 11 models is trained with entirely different SSL approaches. In Table 1, we also compute the magnitude of the Pearson correlation coefficient between the performance of each of the generated models across different datasets and $H(R)$, $I(R;Z)$, and the ratio between both of them. We observe that generally the performance correlates more with the ratio, rather than either of the terms individually. Again, this result empirically shows the existence of an ideal balance between $H(R)$ and $I(R;Z)$ that will correspond to the best performing SSL model. This analysis suggests that SSL algorithms should have a mechanism to adaptively balance between both terms across training.

4 METHODOLOGY

Based on the analysis of the previous section, we introduce a method to balance the training trajectory of both $H(R)$ and $I(R;Z)$. Consider an image i drawn from a training pool $i \in I$. i is passed into two random transformations $a(i) = x_i$ and $a^{'}(i) = x_i^{'}$ where a and $a^{'}$ are drawn from the set of all random augmentations A. Both x_i and $x_i^{'}$ are passed into an encoder network $e(\cdot)$. This results in the representations $e(x) = r_i$ and $e(x^{'}) = r_i^{'}$. These representations are then passed into a projection head $g(\cdot)$ that produces the embeddings $g(x_i) = z_i$ and $g(x_i^{'}) = z_i^{'}$. The collection of all representations and embeddings within a batch of b samples can be represented by the R, $R^{'}$, Z, and $Z^{'}$ matrices. In this case, all matrices are composed of b vectors with F features. From this setup, we can compute L_{NCE} used in SimCLR (7) and the L_{VICReg} loss. The main details of each loss is provided in Section A.5. For the purposes of the `AdaDim` methodology, we highlight the sample uniformity term in L_{NCE} and the feature decorrelation term in L_{VICReg}:

$$L_{NCE} = \sum_{i \in I}(-z_i \cdot z_i^{'})/\tau + \underbrace{log(\sum_{k \in K(i)} exp(z_i \cdot z_k/\tau)))}_{uniformity} \quad L_{VICReg} = \lambda s(Z, Z^{'}) + \mu[v(Z) + v(Z^{'})] + \underbrace{\nu[c(Z) + c(Z^{'})]}_{decorrelation} \tag{4}$$

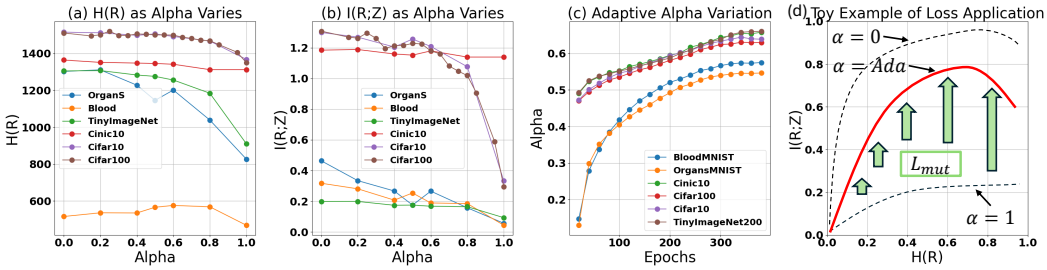

Figure 5: a) This figure shows the impact of manually varying alpha on $H(R)$. b) This figure shows the impact of manually varying alpha on $I(R;Z)$. c) This figure shows how the adaptive α parameter varies during training of a ResNet50 model for 400 epochs across a variety of datasets. d) This figure gives a toy example of the inution behind our loss that includes the adaptive α leading to an intermediate $H(R)$ and $I(R;Z)$ trajectory followed by gradual increases in L_{mut} regularization.

The second term in L_{NCE} is a sample uniformity loss as it distances the image of interest z_i away from all other samples in the batch of interest $k \in K(i)$. The final term in L_{VicReg} represents a decorrelation loss as it tries to drive the covariance matrix towards an identity matrix. It takes the form $c(Z) = \frac{1}{F}\sum_{i \neq j}[C(Z)]^2_{i,j}$ where $C(Z)$ is the covariance matrix of Z. We then compute the dimensionality of the current embedding space Z after every e_α epochs (20 in this paper) of training. This is done by computing the SVD of the representation space of 10 randomly chosen batches from the training set and then calculating the average effective rank across these batches $ER(Z)$ (35). We then scale $ER(Z)$ by the maximum possible dimensionality value which is $D = min(b, F)$ to produce the adaptive parameter $\alpha = \frac{ER(Z)}{D}$. α will gradually transition from 0 to 1 during training as the dimensionality of the space increases. Therefore, we can transition between optimizing between feature decorrelation and sample uniformity with the loss $(1-\alpha)L_{VICReg} + \alpha L_{NCE}$. However, we also want to gradually increase regularization on $I(R;Z)$. To do this, we compute an $I(R;Z)$ loss $L_{mut}(R,Z)$ that encourages lower $I(R;Z)$ with the α-Renyi entropy approximation technique (51; 32; 34). This loss first computes the entropy of a matrix with the formula $H(R) = -\frac{1}{2}log[tr(\frac{R}{b})^2]$. The mutual information can then be computed as $I(R;Z) = H(R) + H(Z) - H(R \odot Z)$. For purposes of numerical stability, $L_{mut} = I(\hat{R}\hat{R}^T; \hat{Z}\hat{Z}^T)$ where $\hat{R}$ and $\hat{Z}$ refer to the normalized version of each space. We scale its regularization through the term $\beta = \gamma * \alpha$ with γ. The final form of our loss is then:

$$L_{AdaDim} = (1-\beta)[(1-\alpha)L_{VICReg} + \alpha L_{NCE}] - \beta L_{mut} \tag{5}$$

Our goal is for the optimization objective to naturally lead to an ideal balance between $H(R)$ and $I(R;Z)$ by the end of training. In order to achieve this balance, the loss needs different components that both support and oppose the growth of $H(R)$ and $I(R;Z)$ at different points during the training process by exploiting the observed dynamics that we discuss in Section 3. The first set of components that are balanced with the α term are L_{NCE} and L_{VICReg}. In parts a) and b) of Figure 5, we show the impact on $H(R)$ and $I(R;Z)$ when manually varying α from 0 to 1 while fixing $\beta = 0$ across 6 different datasets. As a loss based on sample uniformity, L_{NCE} supports lower $H(R)$ and $I(R;Z)$ while a feature decorrelation based loss like L_{VICReg} supports higher $I(R;Z)$ and $H(R)$. This leads to the behavior of parts a) and b), where gradually varying the loss from 0 (L_{VICReg}) to 1 (L_{NCE}) consistently leads to both a lower $H(R)$ and $I(R;Z)$. In part c), we show that the adaptive α term grows from 0 to 1 in a manner that is specific to the unique dimensionality characteristics of each dataset. Therefore, the adaptive α term encourages an intermediate $H(R)$ and $I(R;Z)$ training trajectory when compared with $\alpha = 0$ or $\alpha = 1$ as shown in the toy intuition example of part d). However, at the end of training, both L_{VICReg} and L_{NCE} will demonstrate the SSL dynamic of lowering $I(R;Z)$ in late stage training. Therefore, to maintain balance in this dynamic system, we need an additional term that explicitly opposes the decrease in $I(R;Z)$ as shown by the magnitude of L_{mut} increasing as it scales with α in part d). In this way, all parts of this loss are designed to dynamically balance both $H(R)$ and $I(R;Z)$ across all stages of training.

AdaDimMut Parameter Variation Ablation			
α	γ	β	Accuracy
0	0	0	72.14
1	0	0	69.57
0.5	0	0	72.23
Ada	0	0	72.30
Cosine	0	0	71.40
Linear	0	0	71.59
Ada	1e-04	1	72.00
1	1e-04	1	68.81
0	1e-04	1	72.10
Ada	1e-04	Ada	**72.73**

Table 2: This table shows how performance varies on Cifar-100 as changes are made to the α and β parameters.

AdaDimMut Standardized Hyperparameter Ablation Study								
Method	α Type	Cifar100	Cifar10	TinyImageNet200	Cinic10	Blood	OrganS	iNat21
SimCLR (7)	N/A	64.00	88.59	44.78	78.54	92.54	77.67	23.96
VICReg (3)	N/A	64.70	90.02	45.54	78.25	92.48	76.50	24.24
SimCLR + λ (32)	N/A	64.37	88.00	45.54	76.96	92.86	77.98	23.51
VICReg + λ (32)	N/A	64.54	89.77	45.83	78.47	92.43	77.16	24.01
SimCLR +VICReg	$\alpha = 0.5$	66.53	90.43	46.26	79.35	92.86	78.50	24..56
SimCLR +VICReg	cosine	65.78	88.85	45.45	78.87	92.57	78.46	-
SimCLR +VICReg	linear	66.99	89.53	45.94	78.60	92.07	78.61	-
AdaDim (α = Ada)	Ours	66.90	90.72	47.81	78.55	93.10	78.55	-
AdaDim (α = Ada, β = Ada)	Ours	**67.15**	**90.81**	**48.24**	**79.53**	**93.24**	**79.19**	**24.81**

Table 3: This table shows an ablation study of performance across a variety of datasets when varying AdaDim parameters.

5 RESULTS

For the vast majority of experiments, a Resnet-50 (20) architecture is used in tandem with a simple 3-layer MLP projection head. All ablation study experiments utilize the same projection head with augmentation scheme for ease of comparison. For comparison with state of the art models, the parameters from the solo-learn (11) library or the original paper are used. Our AdaDim method is trained with a LARS optimizer, batch size of 256, a learning rate of 0.4, a weight decay of 1e-4, and $\gamma = 1e-4$. For all experiments, models are trained for 400 epochs. The exception to these conventions are comparisons with ImageNet-100 where a ResNet-18 model is used with $\gamma = -0.1$. An online linear evaluation setting is used for all experiments that has been shown to directly correlate with the offline setting and act as a standard benchmark (17; 18; 11). Further details are in Section A.4.

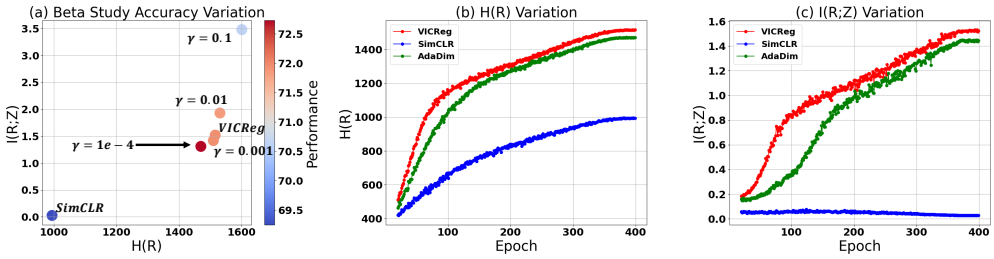

Figure 6: a) This figure shows how performance, $H(R)$, and $I(R;Z)$ change as γ is varied. b) This figure shows how the effective rank of AdaDim varies compared to baseline methods. c) This figure shows how the $I(R;Z)$ for AdaDim varies compared to baseline methods.

In Tables 2 and 3, we analyze the performance of `AdaDim` under a fixed setting where all comparisons use the same projector head, augmentations, and optimizer settings. In Table 2, we analyze the impact of different design choices of α, γ, and β on the downstream performance of out method. We compare against methods that make use of intutive α scaling methods such as cosine or linear growth between 0 and 1 over the course of training. However, these methods are lacking in terms of an ability to adapt the optimization based on the dimensional characteristics of the specific dataset and correspondingly perform worse than an adaptive α. Additionally, we compare against using a fixed $I(R;Z)$ term during training. We observe that this regularization causes a slight decrease in performance. This result suggests that $I(R;Z)$ regularization should be applied selectively at specific points in SSL training, rather than a constant term throughout. We also note the marked performance improvement when transitioning between training with α alone compared to β in tandem with α. In this case, the α term regularizes the transition from a feature decorrelation loss to a sample uniformity loss while β balances additional regularization on $I(R;Z)$ as training progresses. We confirm this ablation study across many different datasets in Table 3. Again, we observe that ad-hoc α scaling strategies such as linear and cosine scaling do not provide additional benefits. We also compare against the $I(R;Z)$ regularization strategy proposed by (32). This work argues that simply reducing $I(R;Z)$ during training without an adaptive mechanism can improve SSL representations. However, we find that using their suggested λ parameter results in little to no improvement in classification accuracy

ResNet-18 400 Epoch ImageNet100 Comparison	
Method	Accuracy
Barlow Twins (50)	80.38
BYOL (19)	80.16
DeepClusterv2 (4)	75.36
DINO (6)	74.84
Moco v2 (8)	78.20
Moco v3 (10)	80.36
NNCLR (14)	79.80
ReSSL (52)	76.92
SimCLR (7)	77.64
SimSiam (9)	74.54
SwAV (5)	74.04
VICReg (3)	79.22
AdaDim	**80.78**

Table 4: This table shows the performance of SOTA SSL methods under 400 epochs of training on ResNet-18. All results comparing with AdaDim are taken from the tables in the solo-learn library.

AdaDim Solo-Learn SOTA Comparison								
Method	Cifar100	TinyImageNet200	Cinic10	STL10	Blood	OrganA	OrganS	OrganC
SimCLR (7)	69.06	46.66	78.77	86.73	93.10	88.04	77.98	91.13
VICReg (3)	72.18	<u>48.47</u>	<u>82.70</u>	87.92	<u>93.77</u>	92.21	<u>80.37</u>	<u>91.84</u>
Moco v2 (8)	71.01	46.78	81.48	**92.41**	93.74	90.49	75.96	90.81
BYOL (19)	71.72	32.96	80.00	<u>89.96</u>	92.45	92.26	78.53	91.45
Barlow Twins (3)	70.84	46.73	81.5	88.45	89.91	91.69	78.69	89.77
NNCLR (14)	70.72	39.66	77.28	87.16	93.15	**92.93**	79.92	91.71
SimSiam (9)	65.52	31.35	79.97	89.45	91.78	91.91	78.31	90.79
Deepcluster v2 (4)	65.70	41.87	74.80	82.93	93.56	92.21	77.93	74.31
Moco v3 (6)	63.96	37.56	74.71	85.25	93.33	92.27	78.69	<u>91.84</u>
AdaDim (α =Ada)	<u>72.23</u>	47.87	82.38	88.11	93.74	<u>92.90</u>	80.19	**91.95**
AdaDim (α =Ada, $\beta = Ada$)	**72.73**	**48.76**	**82.77**	89.01	**94.24**	92.77	**80.80**	**91.95**

Table 5: This table compares `AdaDim` with other SSL methods across diverse data settings. All methods are trained with their best tuned parameters provided in the solo-learn library (11). Note that the `AdaDim` method for this comparison uses the stronger augmentation scheme provided by the solo learn library. We bold the best performing method and underline the second best.

across all datasets. The reason for this discrepancy in their results may be that in their original paper their method only showed improvements with 200 epochs of training. In this limited setting, fixed regularization may help as there isn't enough training time for the discussed dynamics to emerge. However, in the more robust 400 epoch baselines of our work, it is necessary to adapt $I(R;Z)$ to complement the SSL training dynamics. We also analyze the dynamics of our AdaDim method in Figure 6. In part a), we vary γ and plot the accuracy, $H(R)$, and $I(R;Z)$ of representations from Cifar-100. We find that lower γ expectedly leads to a corresponding increase in $H(R)$ and $I(R;Z)$ and gradually reduces performance. The best performing model reaches a balance between both at the $\gamma = 1e-4$ point. In parts b) and c), we compare the growth in $H(R)$ and $I(R;Z)$ with the representations from SimCLR and VICReg. We find that the adaptive regularization of our method leads to `AdaDim` reaching a trajectory that attempts to balance between both terms.

In Tables 5 and 4, we compare against state of the art SSL approaches in the solo-learn codebase setting where each method has its own specific tuned hyperparameters. In this setting, α by itself routinely underperforms relative to other methods across a diverse set of datasets. However, using both the α and β parameters together results in performance comparable to or exceeding all state of the art methods. Again, this highlights the importance of both terms scaling together during training. Additionally, we note that our method out performs or is comparable to strategies that require expensive training paradigms such as queues (14; 8), clustering strategies (5), student teacher networks (19), and additional prediction heads. The only additional overhead with our method is an SVD calculation on 10 batches every 20 epochs. Furthermore, we note that our method is able to consistently perform well both on common baselines such as ImageNet100 and Cifar100, but also on less commonly benchmarked medical datasets (48). The significance of this improvement is that our method scales the optimization objective by measuring the dimensional characteristics specific to each dataset during training. In this way, our method is better able to adapt to scenarios outside of the original natural image domain where previous methods were designed. Additionally, for all experiments we kept $\gamma = 1e-4$ mostly constant. However, further tuning of this term can potentially lead to further improvements on specific datasets that may benefit from more or less $I(R;Z)$ regularization. Overall, our results indicate the benefits of adapting to the specific representational dynamics of SSL training.

6 Conclusion

This paper demonstrates theoretically and empirically that the best performing SSL models arrive at a balance between the dimensionality $H(R)$ of the representation space and the mutual information between the representation and embedding spaces $I(R;Z)$. Specifically, these dynamics indicate that increases in $H(R)$ due to feature decorrelation are preserved between R and Z, but increases due to the samples spreading uniformly can cause $I(R;Z)$ to increase, plateau, or decrease depending on the stage of training of the SSL algorithm. We then introduce a training method called `AdaDim` based on an adaptive interpolation between dimension and sample contrastive approaches and gradual regularization of $I(R;Z)$. `AdaDim` results in improved performance over baseline strategies without requiring additional architectural overhead other than an intermittent SVD calculation.

REFERENCES

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856*, 2017.

[2] Kumar K Agrawal, Arnab Kumar Mondal, Arna Ghosh, and Blake Richards. alpha-req: Assessing representation quality in self-supervised learning by measuring eigenspectrum decay. *Advances in Neural Information Processing Systems*, 35:17626–17638, 2022.

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[7] 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.

[8] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9640–9649, 2021.

[11] Victor Guilherme Turrisi Da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

[12] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[13] 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. Ieee, 2009.

[14] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.

[15] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International conference on machine learning*, pages 3015–3024. PMLR, 2021.

[16] Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck. *arXiv preprint arXiv:2002.07017*, 2020.

[17] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *International conference on machine learning*, pages 10929–10974. PMLR, 2023.

[18] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

[19] 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.

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

[22] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[24] Jaeill Kim, Suhyun Kang, Duhun Hwang, Jungwook Shin, and Wonjong Rhee. Vne: An effective method for improving deep representation by manipulating eigenvalue distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3799–3810, 2023.

[25] Kiran Kokilepersaud, Stephanie Trejo Corona, Mohit Prabhushankar, Ghassan AlRegib, and Charles Wykoff. Clinically labeled contrastive learning for oct biomarker classification. *IEEE Journal of Biomedical and Health Informatics*, 27(9):4397–4408, 2023.

[26] Kiran Kokilepersaud, Seulgi Kim, Mohit Prabhushankar, and Ghassan AlRegib. Hex: Hierarchical emergence exploitation in self-supervised algorithms. *arXiv preprint arXiv:2410.23200*, 2024.

[27] Kiran Kokilepersaud, Mohit Prabhushankar, and Ghassan AlRegib. Volumetric supervised contrastive learning for seismic semantic segmentation. In *Second International Meeting for Applied Geoscience & Energy*, pages 1699–1703. Society of Exploration Geophysicists and American Association of Petroleum . . . , 2022.

[28] Kiran Kokilepersaud, Mohit Prabhushankar, Yavuz Yarici, Ghassan AlRegib, and Armin Parchami. Exploiting the distortion-semantic interaction in fisheye data. *IEEE Open Journal of Signal Processing*, 4:284–293, 2023.

[29] Kiran Kokilepersaud, Yavuz Yarici, Mohit Prabhushankar, and Ghassan AlRegib. Taxes are all you need: Integration of taxonomical hierarchy relationships into the contrastive loss. *arXiv preprint arXiv:2406.06848*, 2024.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[32] Zhuo Ouyang, Kaiwen Hu, Qi Zhang, Yifei Wang, and Yisen Wang. Projection head is secretly an information bottleneck. *arXiv preprint arXiv:2503.00507*, 2025.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[34] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, volume 4, pages 547–562. University of California Press, 1961.

[35] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

[36] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637. PMLR, 2019.

[37] Johannes Schneider and Mohit Prabhushankar. Understanding and leveraging the learning phases of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14886–14893, 2024.

[38] James B Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In *International Conference on Machine Learning*, pages 31852–31876. PMLR, 2023.

[39] Manu Srinath Halvagal, Axel Laborieux, and Friedemann Zenke. Implicit variance regularization in non-contrastive ssl. *Advances in Neural Information Processing Systems*, 36:63409–63436, 2023.

[40] Zhiquan Tan, Jingqin Yang, Weiran Huang, Yang Yuan, and Yifan Zhang. Information flow in self-supervised learning. *arXiv preprint arXiv:2309.17281*, 2023.

[41] Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M Susskind, and Etai Littwin. Lidar: Sensing linear probing performance in joint embedding ssl architectures. *arXiv preprint arXiv:2312.04000*, 2023.

[42] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

[43] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[44] Tobias Uelwer, Jan Robine, Stefan Sylvius Wagner, Marc Höftmann, Eric Upschulte, Sebastian Konietzny, Maike Behrendt, and Stefan Harmeling. A survey on self-supervised methods for visual representation learning. *Machine Learning*, 114(4):1–56, 2025.

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

[46] John Von Neumann. *Mathematical foundations of quantum mechanics: New edition*. Princeton university press, 2018.

[47] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pages 9929–9939. PMLR, 2020.

[48] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

[49] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2(5):8, 2015.

[50] 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.

[51] Yifan Zhang, Zhiquan Tan, Jingqin Yang, Weiran Huang, and Yang Yuan. Matrix information theory for self-supervised learning. *arXiv preprint arXiv:2305.17326*, 2023.

[52] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. *Advances in Neural Information Processing Systems*, 34:2543–2555, 2021.

A Appendix Experimental Details

Limitations and Broader Impact Our work focuses on finding an optimal point between $H(R)$ and $I(R; Z)$ and discussing the training dynamics that influences both terms. However, the notion of an "ideal" optimal point is difficult to prove with respect to a given data setting. Ideally, there should be a derived bound or ratio between $H(R)$ and $I(R; Z)$ that we can claim with high probability corresponds to a close to "ideal" optimal relationship between the two terms. Despite this limitation, the broader impact of this work is that it provides a general framework to develop SSL algorithms across diverse fields such as medicine (25), seismology (27), and autonomous driving (28). Therefore, it provides an avenue for potential growth of machine learning solutions in a wide variety of fields. We are not aware of any negative societal impacts directly caused by our work.

A.1 Codebase

We use the solo-learn codebase (11) for all experiments.

A.2 Datasets

We show explicit details of all datasets used in this paper in Table 6. The data sets were chosen on the basis of trying to achieve as much diversity across a wide variety of data settings to showcase the adaptability of our method. This includes medical and natural image datasets, datasets of varying sizes, datasets of varying class complexity, and datasets with varying class imbalances.

Dataset	Abbreviation & Link	Description	# of classes
CIFAR-100 (30)	cifar100	100 classes of 32x32 color images, including animals, vehicles, and various objects commonly found in the world.	100
CIFAR-10 (30)	cifar10	10 classes of 32x32 color images featuring everyday objects and scenes such as airplanes, cars, and animals.	10
Tiny ImageNet (49)	tinyimagenet200	200 classes of 64x64 images, a smaller version of the ImageNet dataset, used for object recognition and classification tasks.	200
BloodMNIST (48)	blood	8 classes of 28x28 images, designed for classification of diseases in red blood cells.	8
OrganSMNIST (48)	organs	11 classes of 28x28 images, designed for classifying various types of liver tumor problems.	11
OrganCMNIST (48)	organc	11 classes of 28x28 images, designed for classifying various types of liver tumor problems.	11
OrganAMNIST (48)	organa	11 classes of 28x28 images, designed for classifying various types of liver tumor problems.	11
STL10 (48)	stl10	10 classes of 96x96 images, designed for classifying various types of images.	10
Cinic-10 (12)	cinic10	10 classes of 96x96 images, designed for developing unsupervised feature learning, deep learning, and self-taught learning algorithms.	10
iNaturalist 2021 (45)	inat21	Large-scale dataset with over 10,000 species, collected from photographs of plants and animals in their natural environments for fine-grained classification.	10,000
ImageNet (13)	imagenet	Large dataset with over 1,000 classes, used for image classification and object detection, containing millions of images across a wide variety of categories.	1,000

Table 6: Overview of the datasets used in this paper.

A.3 METRIC ANALYSIS DETAILS

One possible mathematical description for the dimensionality of a representation space $H(R)$ is the von Neumann entropy of eigenvalues (46; 24) which takes the form $H(R) = -\sum_i \lambda_i log(\lambda_i)$ where each λ_i represents an eigenvalue of R. We note that many other entropy estimators take a similar form in Section A.3. To increase $H(R)$ in this setting, we can either increase the total number of non-zero eigenvalues or maintain the same number of eigenvalues, but make the eigenvalues more similar in value to each other (higher uniformity, lower variance). Increasing the total number of eigenvalues corresponds to feature decorrelation in which an SSL algorithm discovers a larger number of total dimensions along which R can vary. Decreasing the variance of eigenvalues within a fixed dimensional space corresponds to sample uniformity where representations spread more equally along all dimensions.

Throughout the paper, the dynamics between $H(R)$ and $I(R; Z)$ is discussed. However, this analysis requires a variety of metrics that were not fully detailed in the main paper. For our analytical experiments, the test set of interest is passed into the trained SSL model and its associated projection head. This results in a matrix for the representation space R and embedding space Z for the test set of size test set size $\times$ 2048. On top of these matrices, certain metrics for analysis are computed such as the effective rank discussed earlier (35). Additionally, $I(R; Z)$ is computed using the α-Renyi matrix mutual information approximation discussed in (40). To calculate this quantity, assume that normalized matrices A and B are both R^{nxn}. The entropy of matrix A can be represented as $H_\alpha(A) = \frac{1}{1-\alpha} log[tr((\frac{A}{n})^\alpha)]$ where α =2 for all experiments. This formulation results in a matrix mutual information estimator of the form $I(A; B) = H_\alpha(A) + H_\alpha(B) - H_\alpha(A \odot B)$ where $\odot$ is the hadamard product. This formulation only works for positive semi definite matrices so during our experiments the approximation of (40) is followed where the normalized covariance matrices RR^T and ZZ^T are used as inputs to calculate $I(R; Z)$.

Note that there are a variety of ways to approximate $H(R)$. In this paper, both $H_\alpha(R)$ and the effective rank are used at different points. The main reason for this choice is that the effective rank is normalized with respect to the eigenvalues of the current distribution. This means that the lowest possible value is 0 and the highest possible value is the dimension of the batch of interest. The advantage of the α-Renyi approximator is that the scale of the values will more closely match the values used to calculate $I(R; Z)$. This makes it more useful for visualization within a plot. However, both metrics result in the same trade-off behavior and are correlated with each other. This correlation is observed in Figure 7. In general, any computation of $H(R)$ can be thought of as an approximation of the dimensionality of the representation space. This is because higher dimensionality has been characterized in terms of eigenvalue distributions across a variety of works (18; 41; 2; 21). These metrics follow this trend as they are based on measuring the distribution of eigenvalues for a given matrix. For example, another possible entropy estimator is discussed in (51). This work states that for a positive semi definite (PSD) matrix A, matrix entropy (ME) can be defined as $ME(A) = -tr(Alog(A)) + tr(A) = -\sum_i \lambda_i log(\lambda_i) + \sum_i \lambda_i$. The first term will increase with the dimensionality of the representation space i.e. as the eigenvalues become more uniformly distributed. The second term will increase with more and larger eigenvalues i.e. as the dimensions of the space increases.

The uniformity metric (47) is also used as part of our analysis. This metric acts as a measurement of how uniformly distributed the points of a representation space are on a hypersphere. It takes the form of the pairwise gaussian potential kernel and can be expressed as $log(\mathbb{E}_{(x,y)\sim p_{data}}[e^{-2||e(x)-e(y)||_2^2}])$. In general, greater uniformity indicates a more negative value when this metric is computed empirically. We use the implementation from the original github of (47).

A.4 METHOD SPECIFIC TRAINING DETAILS

All essential hyperparameters for comparisons with state of the art methods are shown in Table 7. Note that we tried to use identical hyperparameters as much as possible for ease of comparison across experiments. In the case of method specific hyperparameters, we tried to use the parameters described in the solo-learn codebase as much as possible (11). We also compare with the explicit $I(R; Z)$ regularization. In these experiments, the experimental setup of (32) is used. This involves taking the matrices R and Z and computing the mutual information estimate based on the α-Renyi approximation discussed in Section A.3. This is added as a regularization term on top of the SSL

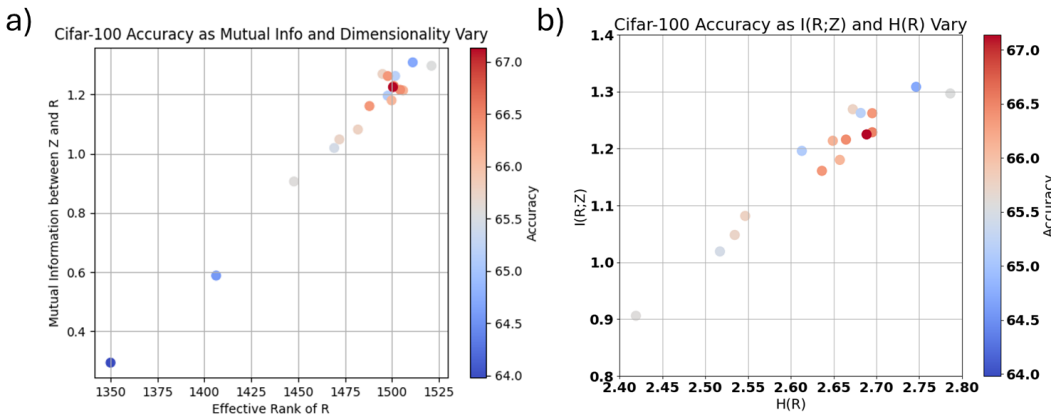

Figure 7: We show versions of the same opening Figure with $H(R)$ computed with a) the effective rank and b) an α-Renyi matrix approximator.

Method	Projection	Method-Specific Parameters	Optimizer	Batch Size	Learning Rate	Weight Decay
AdaDim - Baseline	2048-2048-2048	gamma = 1e-4, 20 Epochs SVD Calc, starting alpha = 0.1	LARS	256	0.4	1e-4
AdaDim - ImageNet100	2048-2048-2048	gamma = -1e-1, 20 Epochs SVD Calc, starting alpha = 0.1	LARS	256	0.4	1e-4
Barlow Twins	2048 - 2048 -2048	scale_loss = 0.1	LARS	256	0.3	1e-4
SimCLR	2048-2048-128	temperature = 0.1	LARS	256	0.4	1e-4
VICReg	2048 - 2048 -2048	var_loss = 25, inv_loss = 25, cov_loss = 1	LARS	256	0.4	1e-4
BYOL	4096 - 4096 - 256	momentum = 1.0, base = 0.99	LARS	256	1.0	1e-5
NNCLR	2048 - 4096 -256	queue = 65536, temperature = 0.2	LARS	256	0.4	1e-5
SimSiam	2048 - 2048 -512	temperature = 0.2	LARS	256	0.5	1e-5
DeepCluster v2	2048 - 128	Prototypes = [3000, 3000, 3000]	LARS	256	0.6	1e-5
Moco v2	2048 - 256	temperature = 0.2, LARS, momentum = [0.9,0.99]	SGD	256	0.3	1e-4
Moco v3	4096 - 4096 - 256	momentum = [0.9,0.99]	LARS	256	0.3	1e-6

Table 7: This table shows the parameters that were used to train every ssl comparison. These parameters were mostly taken from the solo-learn library with some exceptions in order to improve training.

method of interest in Table 7. This regularization term is scaled by a λ parameter that is set to .0001 for all experiments. This specific choice of λ is based on the best performing model in (32).

Additionally, due to the extensive nature of our experiments, an online linear evaluation setting is used where the classifier is trained alongside the backbone and projector. Representations are fed to a linear classifier while keeping the gradient of the classifier's cross entropy loss from flowing through the backbone. The performance of the online classifier correlates well with the offline setting, making it a reliable proxy as shown in (18; 7). In this setting, a single linear layer of size 2048 is used to match the feature size of ResNet-50 to perform this fine-tuning operation.

A.5 Complete SimCLR and VICReg Loss

In this section, we go into more depth regarding the L_{NCE} and L_{VICReg} losses. Suppose there is an image i drawn from a training pool $i \in I$. i is passed into two random transformations $t(i) = x$ and $t^{'}(i) = x^{'}$ where t and $t^{'}$ are drawn from the set of all random augmentations T. Both x and $x^{'}$ are passed into an encoder network $e(\cdot)$. This results in the representations $e(x) = r$ and $e(x^{'}) = r^{'}$. These representations are then passed into a projection head $g(\cdot)$ that produces the embeddings $g(x) = z$ and $g(x^{'}) = z^{'}$. The collection of all representations and embeddings within a batch of n samples can be represented by the R, $R^{'}$, Z, and $Z^{'}$ matrices. In this case, all matrices are composed of n vectors with dimension D. This can be written as $R = [r_1, r_2, ..., r_n]$, $R^{'} = [r_1^{'}, r_2^{'}, ..., r_n^{'}]$, $Z^{'} = [z_1^{'}, z_2^{'}, ..., z_n^{'}]$, and $Z = [z_1, z_2, ..., z_n]$. From this setup, the VICReg (3) and InfoNCE (31; 7) losses can be computed. In this case, VICReg corresponds to a feature decorrelation loss that is better at promoting higher $H(R)$ while InfoNCE

corresponds to a sample uniformity loss better at promoting lower $I(R;Z)$ at the end of training. The InfoNCE (L_{NCE}) loss is written as: $L_{NCE} = -\sum_{i \in I} log \frac{exp(sim(z_i, z_i^{'})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}[k \neq i] exp(sim(z_i, z_k))}$ where sim refers to the cosine similarity, τ represents a temperature parameter, and the summation in the denominator takes place over all samples from both transformations. The VICReg loss is written as: $L_{VICReg} = \lambda s(Z, Z^{'}) + \mu[v(Z) + v(Z^{'})] + \nu[c(Z) + c(Z^{'})]]$. The invariance term is $s(Z, Z^{'}) = \frac{1}{n} \sum_{i=1}^{N} ||z_i - z_i^{'}||_2^2$. The covariance term is $c(Z) = \frac{1}{D} \sum_{i \neq j} [C(Z)]_{i,j}^2$ where $C(Z)$ is the covariance matrix of Z. The variance term is $v(Z) = \frac{1}{d} \sum_{j=1}^{D} max(0, \gamma - S(z^j, \epsilon))$ where $S(x, \epsilon)$ is the regularized standard deviation, z^j represents the vector of each value at dimension j, and γ is a target value set to 1 for all experiments. For both L_{NCE} and L_{VICReg}, we use the conventions of the original papers which includes $\tau = 0.1$, $\lambda = \mu = 25$, and $\nu = 1$.

A.6 Compute Resources

Our resources included a personal PC with 8 Intel i7-6700K CPU Cores and 2 12 GB Nvidia GeForce GTX Titan X GPUs. We also used a lab work station server with 12 Intel i7-5930K CPU cores and 2 24GB Nvidia TITAN RTX GPUs. We also used a server with compute resources based on availability and priority queues. The vast majority of experiments run with these resources are shown in the main paper or the appendix. However, there may be early exploratory experiments in the development of our method that were not included.

A.7 Compute Discussion of our Method

Our method involves computing the distribution of the eigenvalues at different points in the training process. In general, computing eigenvalues is an expensive operation with order $O(n^3)$. However, the number of calculations is limited through a few mechanisms specific to `AdaDim`. This includes the usage of the E_α parameter. This parameter dictates how many epochs must pass before the α parameter is re computed. In Figure 16, performance improvements are maintained even when E_α is as much as 100 epochs. Additionally, for every E_α, the eigenvalues for only 10 training batches are computed. This is because we found empirically that most batches will have a similar effective rank as training progresses. This limits the need to compute the eigenvalues across all batches in an epoch. The averaging across 10 batches is done to ensure that the resulting α reflects the current dimensionality of the dataset. However, it may be possible to use even fewer batches in this computation.

A.8 Empirical Eigenvalue Analysis Details

In Figure 3, a variety of analyses on the eigenvalue distribution of a model trained with the VICReg methodology is performed for 2000 epochs. In part b), all eigenvalues are normalized before counting the number of eigenvalues above a threshold τ that we set to .01 for all experiments. This normalization was performed by dividing all the eigenvalues by the l-1 norm of the complete eigenvalue distribution. This is similar to the normalization done in the computation of the effective rank. In part c), the cumulative explained variance ratio metric is computed. To compute this metric, assume that there is a set of eigenvalues $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]$ where the eigenvalues are ordered in the order of increasing magnitude. Assume that there is a percentage p of eigenvalues. This results in the explained variance metric: $\frac{\sum_{i=1}^{p*N} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$. This metric increases as the subset of eigenvalues that we sum over constitutes more of the overall variance of the data. However, it will decrease as the spread of this variance is distributed over eigenvalues outside of the percentage that the numerator is summed over.

A.9 Random Parameter Ablation Study

In Figure 4, we show how accuracy varies for a variety models with different hyperparameters. We generate 15 models for 3 different methods on Cifar-100 and display the exact parameters for each of these methods in Table 8.

Method	Dataset	Epochs	Parameters	Learning Rate	Temperature	Weight Decay	Effective Rank	Mutual Info	Accuracy
SimCLR	Cifar-100	100	d=2048	0.6	0.05	10-6	673	0.073	47.15
SimCLR	Cifar-100	100	d=2048	0.6	0.07	10-6	682	0.071	48.84
SimCLR	Cifar-100	100	d=2048	0.6	0.1	10-6	684	0.079	49.43
SimCLR	Cifar-100	100	d=2048	0.6	0.2	10-6	646	0.075	52.28
SimCLR	Cifar-100	100	d=2048	0.6	0.3	10-6	594	0.076	54.39
SimCLR	Cifar-100	100	d=2048	0.6	0.4	10-6	566	0.068	51.99
SimCLR	Cifar-100	100	d=2048	0.5	0.05	10-6	658	0.064	48.65
SimCLR	Cifar-100	100	d=2048	0.5	0.07	10-6	652	0.056	47.36
SimCLR	Cifar-100	100	d=2048	0.5	0.1	10-6	670	0.055	54.56
SimCLR	Cifar-100	100	d=2048	0.5	0.15	10-6	666	0.074	54.03
SimCLR	Cifar-100	100	d=2048	0.5	0.2	10-6	637	0.078	54.98
SimCLR	Cifar-100	100	d=2048	0.5	0.3	10-6	587	0.078	54.15
SimCLR	Cifar-100	100	d=2048	0.5	0.4	10-6	556	0.07	53.1
SimCLR	Cifar-100	100	d=2048	0.5	0.15	10-7	655	0.076	54.86
SimCLR	Cifar-100	100	d=2048	0.5	0.15	10-5	666.67	0.076	53.59
VICReg	Cifar-100	100	nu = 0.3	0.3	N/A	10-6	922	0.268	50.75
VICReg	Cifar-100	100	nu = 0.4	0.3	N/A	10-6	914	0.265	50.83
VICReg	Cifar-100	100	nu = 0.5	0.3	N/A	10-6	902	0.292	50.62
VICReg	Cifar-100	100	nu = 0.6	0.3	N/A	10-6	892	0.263	52.29
VICReg	Cifar-100	100	nu = 0.7	0.3	N/A	10-6	902	0.235	56.72
VICReg	Cifar-100	100	nu = 0.8	0.3	N/A	10-6	903	0.24	56.27
VICReg	Cifar-100	100	nu = 0.9	0.3	N/A	10-6	898	0.237	55.73
VICReg	Cifar-100	100	nu = 1.0	0.3	N/A	10-6	883	0.244	52.37
VICReg	Cifar-100	100	nu = 1.1	0.3	N/A	10-6	878	0.229	57.6
VICReg	Cifar-100	100	nu = 1.2	0.3	N/A	10-6	877	0.232	52.28
VICReg	Cifar-100	100	nu = 1.3	0.3	N/A	10-6	847	0.257	54.54
VICReg	Cifar-100	100	nu = 1.4	0.3	N/A	10-6	868	0.212	55.09
VICReg	Cifar-100	100	nu = 1.5	0.3	N/A	10-6	795	0.279	49.19
VICReg	Cifar-100	100	nu = 1.6	0.3	N/A	10-6	867	0.209	54.48
VICReg	Cifar-100	100	nu = 1.7	0.3	N/A	10-6	848	0.2107	58.69
NNCLR	Cifar-100	100	d=2048	0.6	0.05	10-6	416	0.043	54.09
NNCLR	Cifar-100	100	d=2048	0.6	0.07	10-6	417	0.038	54.27
NNCLR	Cifar-100	100	d=2048	0.6	0.1	10-6	459	0.033	55.47
NNCLR	Cifar-100	100	d=2048	0.6	0.2	10-6	493	0.058	55.9
NNCLR	Cifar-100	100	d=2048	0.6	0.3	10-6	490	0.067	54.43
NNCLR	Cifar-100	100	d=2048	0.6	0.4	10-6	519	0.067	55.07
NNCLR	Cifar-100	100	d=2048	0.5	0.05	10-6	425	0.042	54.59
NNCLR	Cifar-100	100	d=2048	0.5	0.07	10-6	439	0.036	55.16
NNCLR	Cifar-100	100	d=2048	0.5	0.1	10-6	462	0.033	56.01
NNCLR	Cifar-100	100	d=2048	0.5	0.15	10-6	474	0.05	56.09
NNCLR	Cifar-100	100	d=2048	0.5	0.2	10-6	505	0.06	56.37
NNCLR	Cifar-100	100	d=2048	0.5	0.3	10-6	518	0.074	55.02
NNCLR	Cifar-100	100	d=2048	0.5	0.4	10-6	520	0.075	53.43
NNCLR	Cifar-100	100	d=2048	0.5	0.15	10-7	492	0.048	56.19
NNCLR	Cifar-100	100	d=2048	0.5	0.15	10-6	474	0.051	56.09

Table 8: This table shows all the parameters, accuracies, rank scores, and mutual information values for the random parameter experiments shown in the main paper.

B Appendix Theoretical Details

B.1 High Level Intuition

Higher dimensionality in R is desirable because it counters the dimensional collapse effect discussed in (21) and encourages a more diverse feature space. Lower $I(R;Z)$ is also desirable because it implies that the projection head is effective in removing uninformative features from the representation space. However, we prove through information theoretic bounds that increasing the dimensionality of R causes a corresponding increase in $I(R;Z)$ thus necessitating a trade-off between the two for an ideal representation space. This trade-off is illustrated in Figure 1 where an image is passed through an encoder $e(\cdot)$ to produce a representation space R with 6 associated features. 3 features are target-relevant and 3 are uninformative. The feature space is associated with an eigenvalue distribution that indicates how relevant each feature is to the geometry of the representation space. Ideally, the eigenvalue distribution should capture just the target-relevant features; however, a higher dimensional space also captures uninformative features as shown in part a). To counter this, the projector should act as an information bottleneck (43) during training that projects the features into a lower dimensional space where only the target features are relevant. In part a), the distribution of eigenvalues remains the same after projection so the projection head does not remove spurious features from R which corresponds to a high $I(R;Z)$. Part b) represents an ideal case where R has sufficiently high dimensionality to capture mostly informative features while sufficiently low $I(R;Z)$ such that the projector guides the optimization process towards target-relevant features.

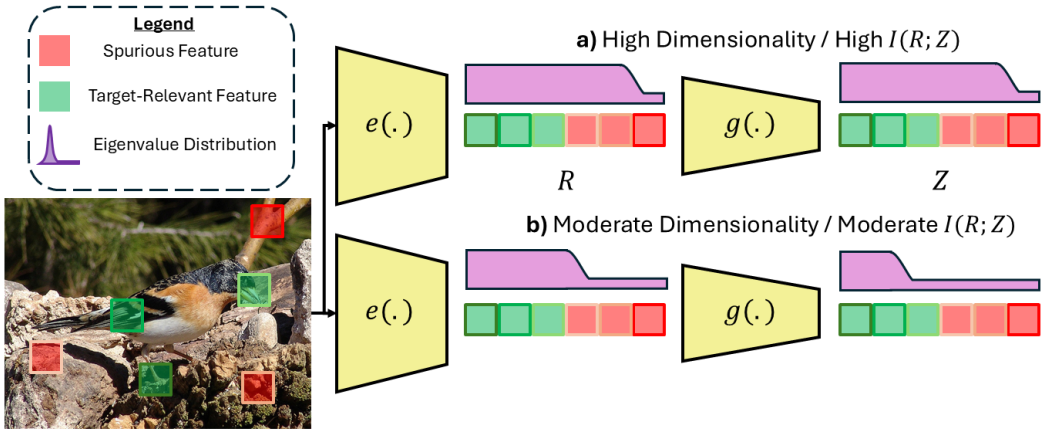

Figure 8: Assume there is an image with 3 task relevant features and 3 spurious features. The image is associated with a representation space R, projection space Z, and corresponding eigenvalue distributions for both. a) This is an example of R and Z with high dimensionality and high $I(Z;R)$. b) This is an example of R and Z that that has moderate dimensionality and moderate $I(R;Z)$.

B.2 Gaussian Mutual Info Derivation Details

We will follow from the assumptions found in Section 3 of the main paper. The following closed form equations are needed for this analysis:

$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_R|) + ln(|\Sigma_Z|) - ln(|\Sigma|))$$
$$H(R) = \frac{m}{2}ln(2\pi) + \frac{1}{2}ln(|\Sigma_R|) + \frac{m}{2}$$
$$ln(|\Sigma|) = ln(|\Sigma_Z||\Sigma_R - \Sigma_{RZ}\Sigma_Z^{-1}\Sigma_{ZR}|)$$
$$ln(|\Sigma|) = ln(|\Sigma_R||\Sigma_Z - \Sigma_{ZR}\Sigma_R^{-1}\Sigma_{RZ}|)$$

Note that the $ln(|\Sigma|)$ derivation arrives from Shur's formula that provides an equality for the determinant of a block covariance matrix.

In this setting, $I(R;Z)$ can be rewritten as:

$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_R|) + ln(|\Sigma_Z|) - ln(|\Sigma_R||\Sigma_Z - \Sigma_{ZR}\Sigma_R^{-1}\Sigma_{RZ}|)$$
$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_R|) + ln(|\Sigma_Z|) - ln(|\Sigma_Z||\Sigma_R - \Sigma_{RZ}\Sigma_Z^{-1}\Sigma_{ZR}|)$$

Using law of logarithms, we can simplify this equation into:

$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_Z|) - ln(|\Sigma_Z - \Sigma_{ZR}\Sigma_R^{-1}\Sigma_{RZ}|)$$
$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_R|) - ln(|\Sigma_R - \Sigma_{RZ}\Sigma_Z^{-1}\Sigma_{ZR}|)$$

This results in the form described in the main paper as:

$$I(R;Z) = \frac{1}{2}(ln(|\Sigma_Z|) - ln(|Var(Z|R)|)) = \frac{1}{2}(ln(|\Sigma_R|) - ln(|Var(R|Z)|))$$

We further analyze the specific terms that make up this equation in Figure 9. In parts a) and b) of this figure, the $I(R;Z)$ curves from the main paper are repeated. In part c), each of the terms that make up $I(R;Z)$ are analyzed as changes as the number of features is fixed and the sample variance increases. $ln(|\Sigma_R|)$ and $ln(|Var(Z|R)|)$ increases as the variance increases. However, $ln(|Var(Z|R)|)$ increases at a faster rate as the variance increases. This happens because $ln(|\Sigma_Z|)$ does not change in value. The end result is a reduction in mutual information which shows that Z is not able to preserve the variance in R under the conditions of its projection.

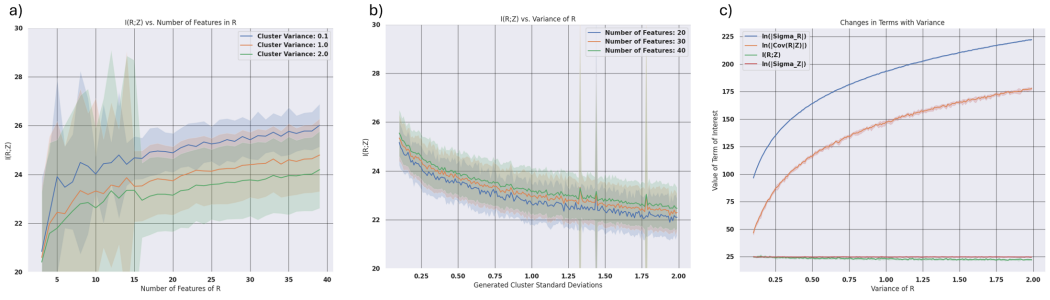

Figure 9: In a) and b), we again show the curves when Z is two components with experiments related to varying the number of features adn the cluster variance. In c), we decompose each of the individual terms that make up $I(R;Z)$.

B.3 Gaussian Simulation Details

In Figure 2, a detailed simulation on data generated from a Gaussian distribution is shown. The simulations discussed two settings between the R space and the projection space Z: $n < m$ and $n << m$. For each setting, $I(R;Z)$ varies when the number of dimensions is kept fixed while the variance of the data is perturbed as well as when the variance of the data is fixed and the number of dimensions is varied. To generate this data, the make blobs dataset from the sklearn library(33) is used. This library generates Gaussian isotropic clusters that are intended for clustering problems. However, for our purposes it acts as a reliable generator of Gaussian distributed data. The cluster labels of this dataset are not used in any capacity for our experiments to conform to the SSL setting. This dataset has the following parameters:

1. n_samples: We set this to 1000 for all experiments.
2. n_features: We set this based on the features required for the simulation of interest.
3. centers: This is set to 5 for all experiments. This describes the number of clusters to generate.
4. cluster_std: This is the parameter we vary to control the variance of the generated data.
5. random_state: This can set the initial random seed for the generation. We do not set this parameter so as to generate a slightly different version of the dataset after every simulation. We then take the average and standard deviation of 100 simulations for every set of parameters that we use in our experiments.

B.4 Neural Network Simulation

In the main paper, PCA is used as a general projection between R and Z for the purposes of modeling the interaction between a space and its projection without having to deal with the nuances of training neural networks. However, the projector can also be replaced with a neural network and either the SimCLR or VICReg loss and show that the same general trends hold.

For this experiment, synthetic gaussian data is generated in the manner described in Section B.3. However, this time a small MLP is used. It is composed of 5 layers and 20 hidden units per layer followed by a small projector with 2 layers and 5 hidden units per layer to output a dimension of size 5. The generated data has 25 features and a cluster standard deviation of .01. It is trained for 1000 epochs with either the SimCLR or VICReg loss. In this setting, augmentations were generated by adding randomly distributed Gaussian noise with a standard deviation of 0.5 to the generated data. During training, $I(R;Z)$ is measured for every epoch where R is the original generated data and Z is the output of the neural network. This value is computed using the closed form $I(R;Z)$ for gaussian distributed data. The Adam optimizer is used for these experiments with a learning rate of .0001 and a β of 0.9 to 0.999.

Figure 10 shows that the neural network simulation of our data exhibits the same trends both when trained on SimCLR or VICReg. At the start of training, $I(R;Z)$ increases and gradually plateaus by the end of training. Additionally, the dimension contrastive strategy VICReg approaches a higher $I(R;Z)$ than that of the sample contrastive strategy SimCLR.

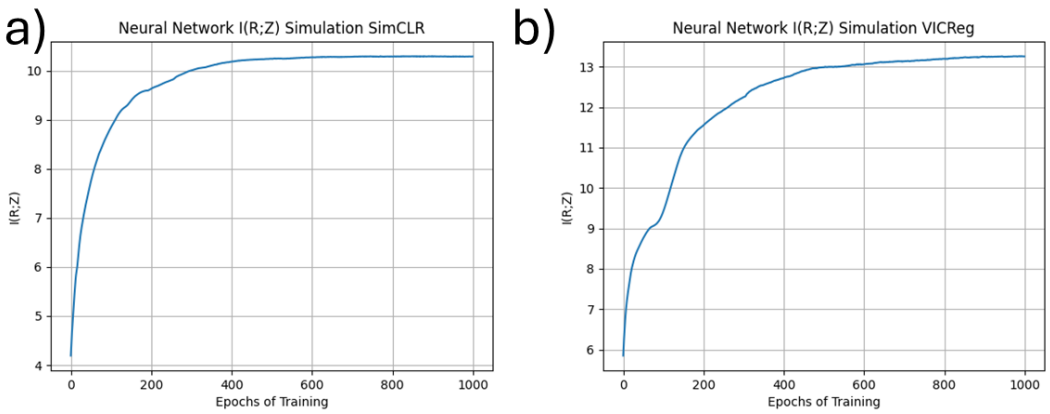

Figure 10: We show the $I(R;Z)$ curves across epochs of training for a gaussian dataset trained on a) SimCLR and b) VICReg.

B.5 Info Theoretic Bounds

The upper bound on $I(R;Y)$ described in the main text originated from a derivation performed in (32). The exact details of these bounds can be found in the original paper. Below the derivation of the bound described in equation 3 is shown completely. The original bound is described as:

$$I(Y;R) \leq I(Y;Z) - I(R;Z) + H(R)$$

The approximation $I(Y;Z) = G$ is used in the main paper which results in the following bound:

$$I(Y;R) \leq G - I(R;Z) + H(R)$$

We substitute in the equation $\frac{1}{2}(ln(|\Sigma_Z|) - ln(|Var(Z|R)|))$ for $I(R;Z)$ and $H(R) = \frac{m}{2}ln(2\pi) + \frac{1}{2}ln(|\Sigma_R|) + \frac{m}{2}$. This results in the bound:

$$I(Y;R) \leq G - \frac{1}{2}(ln(|\Sigma_Z|) - ln(|Var(Z|R)|)) + (\frac{m}{2}ln(2\pi) + \frac{1}{2}ln(|\Sigma_R|) + \frac{m}{2})$$

A simplification of terms results in the bound shown in the main paper as:

$$I(Y;R) \leq G + \underbrace{\frac{1}{2}(ln(|\Sigma_R|) - ln(|\Sigma_Z|))}_{K(Both)} + \underbrace{\frac{1}{2}ln(|Var(Z|R)|)}_{V(I(R;Z))} + \underbrace{\frac{m}{2}(ln(2\pi) + 1)}_{D(H(R))}$$

C Appendix Analytical Details

C.1 VICReg vs. SimCLR Comparison

The `AdaDim` methodology is based on the idea that VICReg better promotes higher $H(R)$ and SimCLR promotes lower $I(R;Z)$. This is based on our analysis that feature decorrelation leads to higher $I(R;Z)$ while sample uniformity leads to an $I(R;Z)$ behavior that depends on the stage of training. These same dynamics are observed in a real SSL setting in Figure 11 where a ResNet-50 model is trained for 2000 epochs on Cifar-100 (30) using the VICReg and SimCLR SSL methods. In part a), both methods have an increase in $I(R;Z)$, but it occurs at a slower rate for SimCLR. In part b), the overall dimensionality of the dataset increases across all training epochs for R, but begins to plateau at the end of training corresponding to the end of the feature decorrelation stage. Z exhibits this same behavior, but plateaus much more noticeably throughout training which may contribute partially to the plateauing effect of $I(R;Z)$. For both R and Z, the overall dimensionality is lower for SimCLR than for VICReg. In part c), R and Z have a similar uniformity for both methods at the start of training, but significantly diverge from each other by the end of training for both methods.

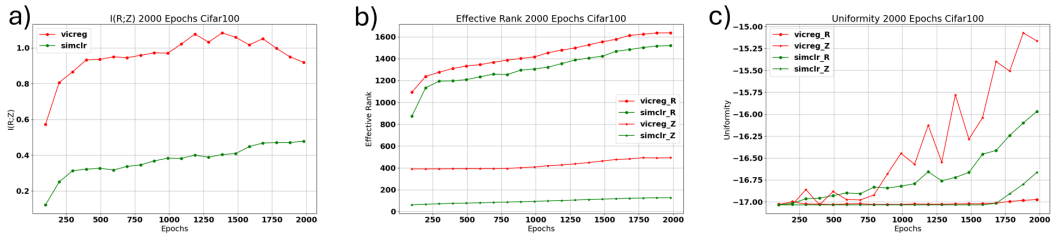

Figure 11: We train models with a two different SSL methods for 2000 epochs and then analyze changes in a) $I(R;Z)$, b) effective rank between R and Z, and c) uniformity between R and Z.

These observed trends for SimCLR and VICReg hold for a wide variety of datasets in parts a) and b) of Figure 12. In part a), at the end of training for 6 different datasets, the dimensionality and $I(R;Z)$ of VICReg is higher than that of SimCLR. In part b), these trends are analyzed over the course of manually setting the α parameter over the course of training from 0 to 1 in increments of 0.2. It is observed that as the optimization changes from VICReg ($\alpha = 0$) to SimCLR ($\alpha = 1$) the $I(R;Z)$ and the dimensionality for all data sets monotonically decreases.

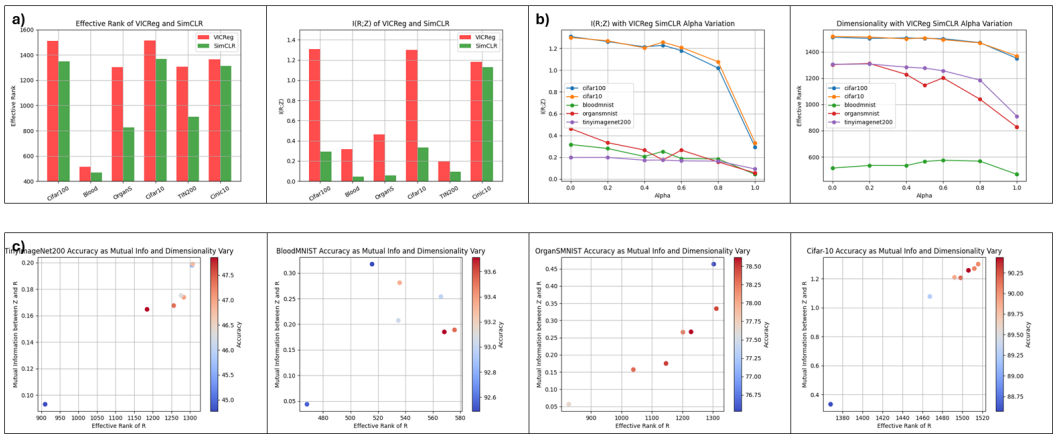

Figure 12: a) We compare the effective rank and $I(R;Z)$ between the representation of the test set for different datasets trained on VICReg and SimCLR. b) We show how the effective rank and $I(R;Z)$ vary after the introduction of the α parameter for each dataset. c) We show how the performance varies as a function of the $I(R;Z)$ and effective rank for a variety of datasets.

C.2 Generation of $H(R)$ vs $I(R;Z)$ Plots

In Figure 12, the empirical results that demonstrate the existence of an optimal point between high $H(R)$ and low $I(R;Z)$ are shown. In part a), the plots were generated by training on each respective dataset with manually chosen α parameters within the `AdaDim` framework. In this case, manual means that the adaptive α computation does not happen and a specific value from 0 to 1 is kept constant across the entire training time with γ set to 0. These α values are $\alpha = [0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0]$. The plot in part b) of Figure 1 was generated in a similar manner, with the exception of the size of the increments of α is reduced to 0.05.

More plots are provided for more datasets in part c) of Figure 12. Note that potentially more models need to be generated in order for the balance trend to be more salient for specific datasets. However, for many of these datasets, such as TinyImageNet, OrganSMNIST, and Cifar-10, these trends of a performance balance between $H(R)$ and $I(R;Z)$ are clear.

One surprising observation from these results is that they contradict the a variety of recent works (2; 41; 18). In these papers, the authors try to argue that some measure of dimensionality can be used as an unsupervised surrogate of representational quality. In other words, higher dimensionality should correspond to the better performing model on potentially any downstream task. However, our work suggests that both dimensionality and $I(R;Z)$ should be considered for an unsupervised assessment of model quality. However, our result is not surprising when we consider how these works justify their conclusions. For example, (17) based their rank estimates off of pre-trained ImageNet models. However, in practice, this assumption may not hold and certain domains such as medicine may benefit more from an in distribution pre-training. (41) showed a wide range of coefficient correlation values (0.2 - 0.8) between different dimensionality based metrics and performance values derived from various sources. This suggests that in some settings dimensionality is a good surrogate for performance while in others $I(R;Z)$ needs to be considered. This corresponds to the dynamics discussed in this paper, where the best performing model is often not the one with the highest dimensionality. It is the one that reaches a suitable intermediate point between dimensionality and $I(R;Z)$. Our work suggests that future unsupervised estimators of representational quality should have some mechanism to detect this optimal balance between the two terms of interest.

C.3 Manual α Usage

		Dataset						
Method	Alpha	Cifar100	Cifar10	TinyImageNet200	Cinic10	Blood	OrganS	iNat21
SimCLR	N/A	64.00	88.59	44.78	78.54	92.54	77.67	23.96
VICReg	N/A	64.70	90.02	45.54	78,25	92.48	76.50	24.24
AdaDim	0.2	65.18	90.07	46.75	78.27	93.36	78.41	-
AdaDim	0.4	66.15	90.18	47.00	78.57	93.04	78.46	-
AdaDim	0.5	66.53	90.43	46.26	79.35	92.98	78.50	24.56
AdaDim	0.6	66.11	89.87	**48.06**	**79.58**	93.56	78.23	-
AdaDim	0.8	66.32	89.25	47.83	78.54	**93.71**	78.26	-
AdaDim	Ada	**66.90**	**90.72**	47.81	79.53	92.86	**78.55**	**24.81**

Table 9: This shows the performance of AdaDim under different α parameters on several different datasets.

In Table 9, an ablation study of the choice of α parameter when β is set to 0 is performed. We compare between the adaptive methodology of our main paper and a method based on setting a manual value that is consistent throughout training. We find that our adaptive methodology either out performs or is consistent with the best result that we get from manually choosing a hyperparameter for α. This highlights the importance of adaptively shifting between losses over the course of training to match the dynamics of SSL training.

C.4 Variation in Optimization Procedure

In Figure 13, the optimization setting is varied for several SSL methods. It is observed that the effective rank and $I(R;Z)$ curves have similar trends for both the adam and lars optimizers. However, the difference is that for the adam optimizer, the effective rank has a more pronounced upper limit on the values it can reach. Additionally, for $I(R;Z)$ the adam trained optimizer begins to decrease

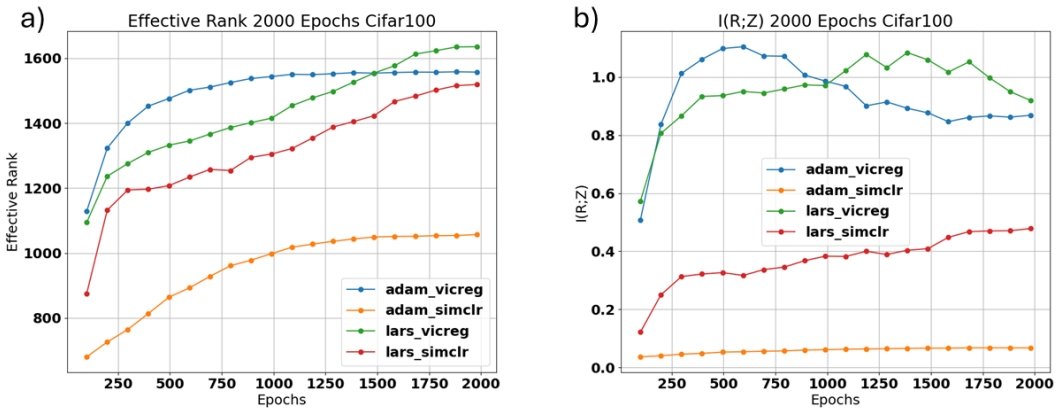

Figure 13: We show how the a) effective rank and b) $I(R;Z)$ vary for both SimCLR and VICReg under different optimization settings.

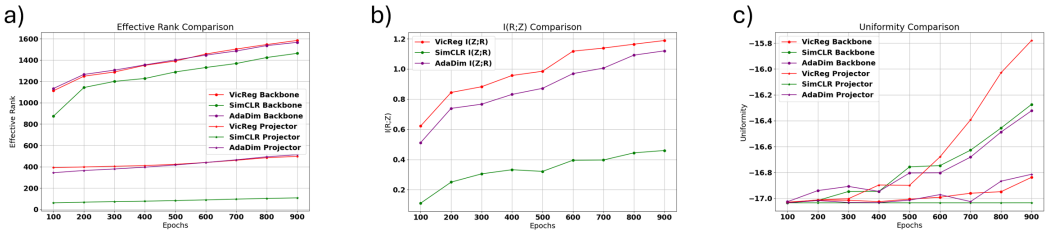

Figure 14: We show the impact of our method in comparison to SimCLR and VICReg over 1000 epochs of training for the a) effective rank metric, b) $I(R;Z)$, and c) uniformity.

or plateau quicker. This result highlights that the trends of this paper are general, but its exact manifestation across training will vary based on the setup of the experiment.

C.5 Training Dynamics of AdaDim

We also show how the training dynamics of our AdaDim approach compares to that of a fully dimension contrastive approach (VICReg) and fully sample contrastive approach (SimCLR) across 1000 epochs of training in Figure 14. In parts a) and b), the AdaDim approach arrives at an intermediate point between both methods in terms of dimensionality and in terms of $I(R;Z)$. Furthermore, in part c), the AdaDim methodology exhibits similar training dynamics in terms of a divergence between the uniformity of R and Z at the end of training. This result confirms our hypothesis that our method is able to find a better balance between $H(R)$ and $I(R;Z)$.

C.6 Hyperparameter Ablation Studies

In Table 10, AdaDim out performs or matches a wide variety of state of the art SSL approaches within the constrained hyperparameter setting that we use for our ablation studies on a diverse set of classification benchmarks. This is significant because AdaDim does not require any additional architectural nuances such as queues (8; 14), predictor architectures (19), or stop gradient calculations (9). It only requires optimization of the space after the projection head. However, an analysis of parameters that can potentially influence AdaDim are shown in Figure 16. In part a), the effect of the output projection size on the performance of AdaDim is shown. AdaDim out performs VICReg that has been previously shown to improve as the output dimension size increases. In part b), the temperature parameter in the I_{NCE} loss is varied. In this case, performance varies with respect to an appropriately chosen temperature parameter, but all temperature values still out perform the baseline SimCLR model. In part c), we investigate how varying the E_α parameter effects

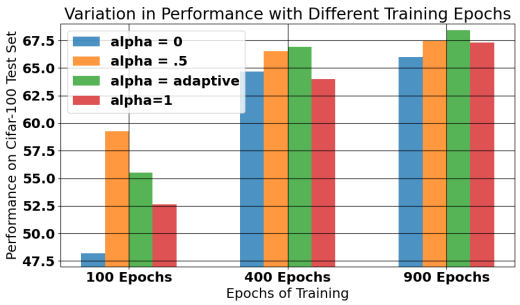

Figure 15: Comparison of AdaDim with different amounts of training time.

Method	Cifar100	TinyImageNet200	Cinic-10
SimCLR (7)	64.00	44.78	78.54
VICReg (3)	64.70	45.54	78.25
Moco v2 (8)	66.06	45.32	77.30
BYOL (19)	66.88	34.60	79.10
Barlow Twins (50)	63.58	44.29	75.98
NNCLR (14)	**67.15**	40.44	78.45
SimSiam (9)	62.61	27.20	78.72
AdaDim	66.90	**47.81**	**79.53**

Table 10: This table compares AdaDim with other SSL methods.

downstream performance. It is observed that any choice of this parameter still results in performance that significantly exceeds SimCLR and VICReg baselines on Cifar-100.

AdaDim is based on adapting to the dynamics of SSL representations. Therefore, it may benefit from a longer training time. This idea is illustrated in Figure 7 where AdaDim is compared against simply setting $\alpha = .5$ manually across all training epochs. Both choices for α out perform SimCLR($\alpha = 1$) and VICReg ($\alpha = 0$). However, the adaptive method significantly improves relative to the manual method as the amount of training time increases. This suggests that with less training time, the model is not able to undergo a complete transition between the feature decorrelation and sample uniformity stages.

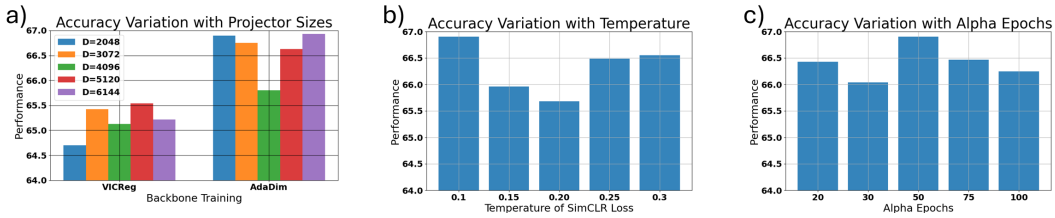

Figure 16: In this figure, we show how performance varies when we perturb key hyperparameters in AdaDim. a) We analyze the impact of different output projector sizes. b) We analyze the impact of varying the temperature parameter in the I_{NCE} loss. c) We analyze the impact of changing E_α.

C.7 Beta Analysis

In Figure 6, we performed an analysis of varying γ in a positive direction. We found that increases both the mutual information and effective rank of R. We also study the impact to R of varying γ in the negative direction. We find that both the mutual information and effective rank drop over the course of training as expected by the regularization on $I(R;Z)$. However, what is interesting is that the same trend of an optimal balance emerges despite the lower rank and $I(R;Z)$. This suggests the existence of multiple rank and $I(R;Z)$ regions where performance can be maximized.

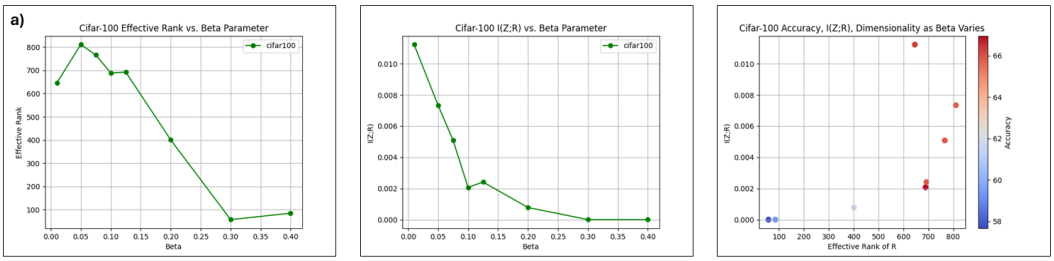

Figure 17: In this figure, we show how performance varies as we manually increase β with positive values. a) This shows how the effective rank varies. b) This figure shows how $I(R;Z)$ varies. c) This figure shows how the performance varies.