

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 $R^{n \times n}$. The entropy of matrix A can be represented as $H_\alpha(A) = \frac{1}{1-\alpha} \log[\text{tr}((\frac{A}{n})^\alpha)]$ where $\alpha=2$ 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 $ME(A) = -\text{tr}(A \log(A)) + \text{tr}(A) = -\sum_i \lambda_i \log(\lambda_i) + \sum_i \lambda_i$. 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

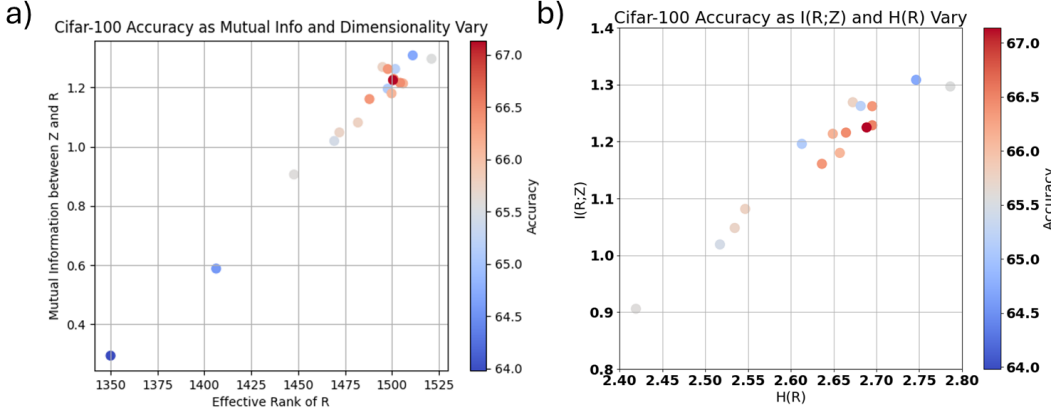


Figure 7: We show versions of the same opening Figure with $H(R)$ computed with a) the effective rank and b) an α -Renyi matrix approximator.

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	$\text{scale_loss} = 0.1$	LARS	256	0.3	$1e-4$
SimCLR	2048-2048-128	$\text{temperature} = 0.1$	LARS	256	0.4	$1e-4$
VICReg	2048 - 2048 - 2048	$\text{var_loss} = 25$, $\text{inv_loss} = 25$, $\text{cov_loss} = 1$	LARS	256	0.4	$1e-4$
BYOL	4096 - 4096 - 256	$\text{momentum} = 1.0$, $\text{base} = 0.99$	LARS	256	1.0	$1e-5$
NNCLR	2048 - 4096 - 256	$\text{queue} = 65536$, $\text{temperature} = 0.2$	LARS	256	0.4	$1e-5$
SimSiam	2048 - 2048 - 512	$\text{temperature} = 0.2$	LARS	256	0.5	$1e-5$
DeepCluster v2	2048 - 128	Prototypes = [3000, 3000, 3000]	LARS	256	0.6	$1e-5$
Moco v2	2048 - 256	$\text{temperature} = 0.2$, LARS, $\text{momentum} = [0.9, 0.99]$	SGD	256	0.3	$1e-4$
Moco v3	4096 - 4096 - 256	$\text{momentum} = [0.9, 0.99]$	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 $R = [r_1, r_2, \dots, r_n]$, $R' = [r'_1, r'_2, \dots, r'_n]$, $Z = [z_1, z_2, \dots, z_n]$, and $Z' = [z'_1, z'_2, \dots, z'_n]$. 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: $L_{NCE} = - \sum_{i \in I} \log \frac{\exp(\text{sim}(z_i, z'_i)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}[k \neq i] \exp(\text{sim}(z_i, z_k))}$ 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 $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_N]$ 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: $\frac{\sum_{i=1}^{p \cdot N} \lambda_i}{\sum_{i=1}^N \lambda_i}$. 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.

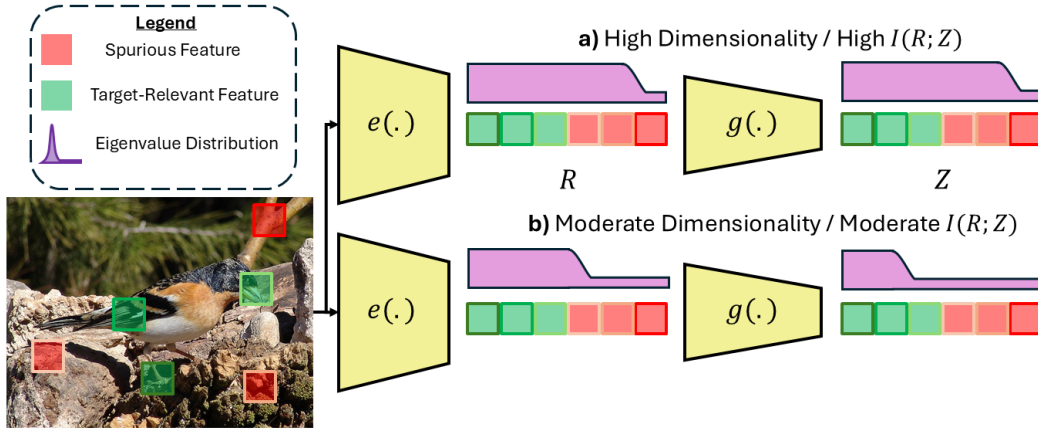


Figure 8: Assume there is an image with 3 task relevant features and 3 spurious features. The image is associated with a representation space R , projection space Z , and corresponding eigenvalue distributions for both. a) This is an example of R and Z with high dimensionality and high $I(Z; R)$. b)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

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

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:

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

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 $\ln(|\text{Var}(Z|R)|)$ increases as the variance increases. However, $\ln(|\text{Var}(Z|R)|)$ 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.

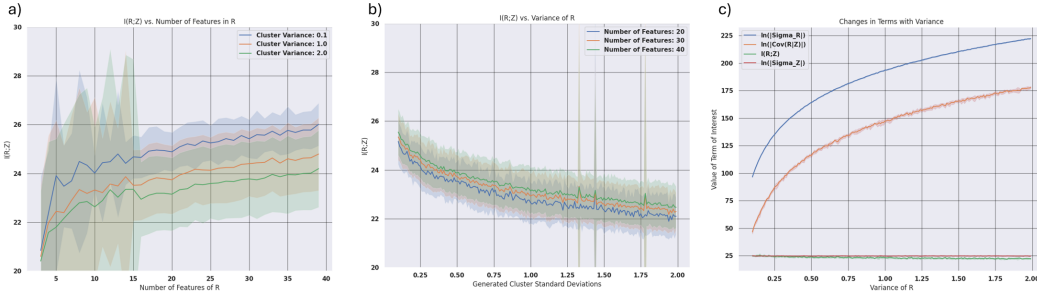


Figure 9: In a) and b), we again show the curves when Z is two components with experiments related to varying the number of features and the cluster variance. In c), we decompose each of the individual terms that make up $I(R; Z)$.

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 $n \ll m$. 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.

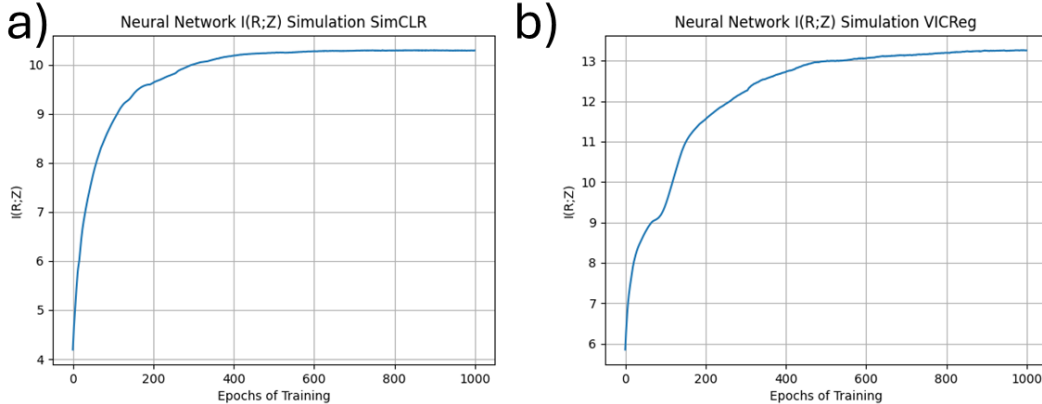


Figure 10: We show the $I(R; Z)$ curves across epochs of training for a gaussian dataset trained on a) SimCLR and b) VICReg.

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 $\frac{1}{2}(\ln(|\Sigma_Z|) - \ln(|\text{Var}(Z|R)|))$ 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:

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

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

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

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.

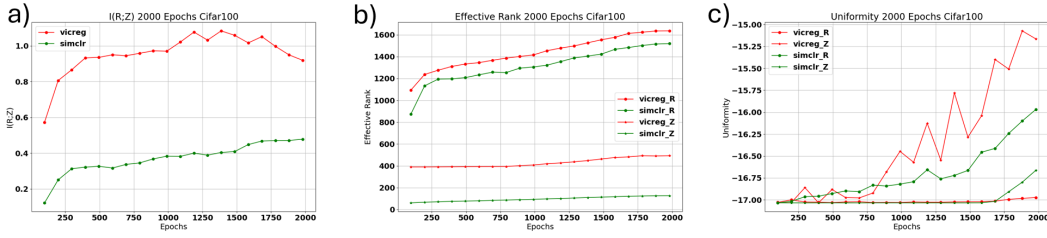


Figure 11: We train models with a two different SSL methods for 2000 epochs and then analyze changes in a) $I(R; Z)$, b) effective rank between R and Z , and c) uniformity between R and Z .

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.

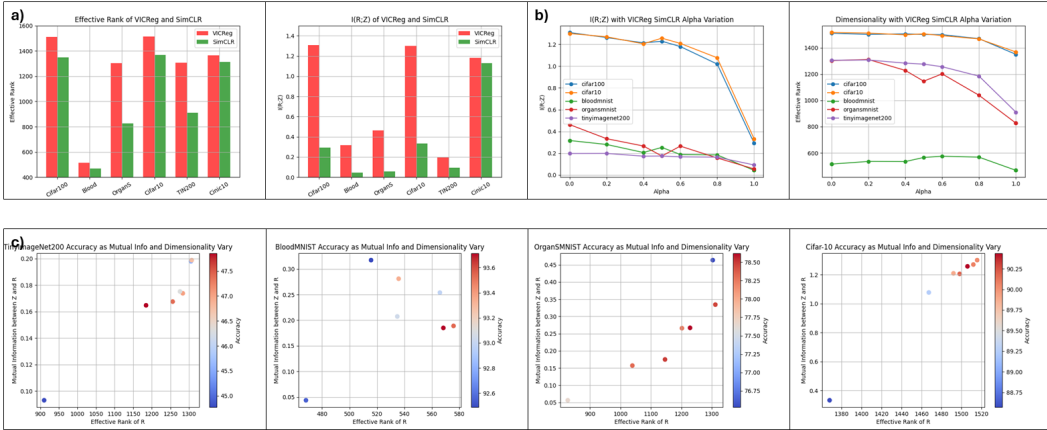


Figure 12: a) We compare the effective rank and $I(R; Z)$ between the representation of the test set for different datasets trained on VICReg and SimCLR. b) We show how the effective rank and $I(R; Z)$ vary after the introduction of the α parameter for each dataset. c) We show how the performance varies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

Method	Alpha	Dataset						
		Cifar100	Cifar10	TinyImageNet200	Cin10	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 outperforms 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

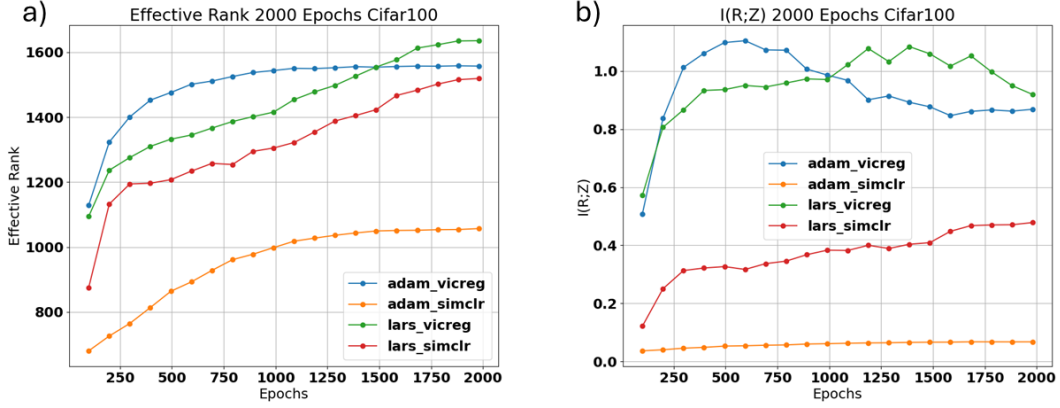


Figure 13: We show how the a) effective rank and b) $I(R; Z)$ vary for both SimCLR and VICReg under different optimization settings.

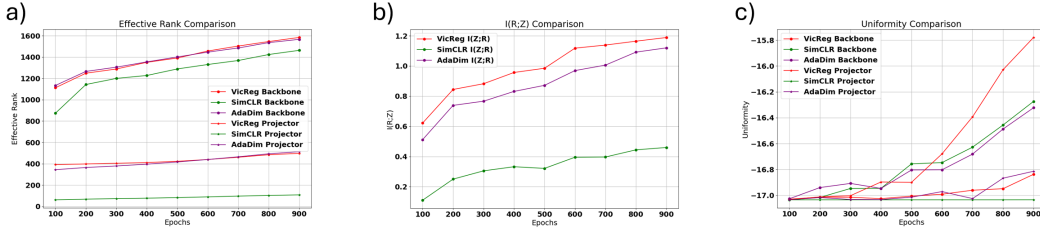


Figure 14: We show the impact of our method in comparison to SimCLR and VICReg over 1000 epochs of training for the a) effective rank metric, b) $I(R; Z)$, and c) uniformity.

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 outperforms 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 outperforms 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 outperform the baseline SimCLR model. In part c), we investigate how varying the E_α parameter effects

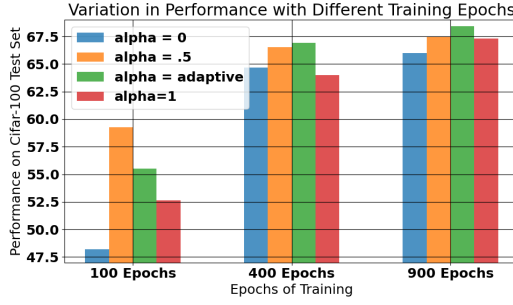


Figure 15: Comparison of AdaDim with different amounts of training time.

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.

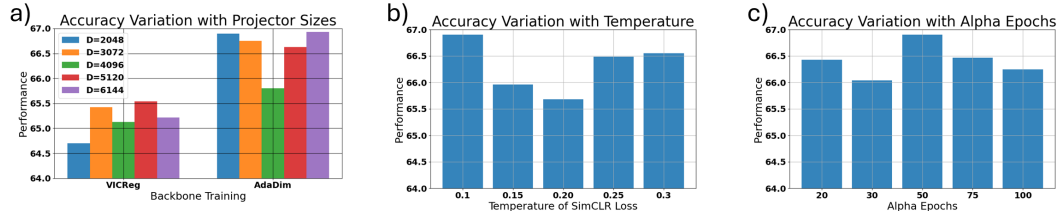


Figure 16: In this figure, we show how performance varies when we perturb key hyperparameters in AdaDim. a) We analyze the impact of different output projector sizes. b) We analyze the impact of varying the temperature parameter in the I_{NCE} loss. c) We analyze the impact of changing E_α .

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.

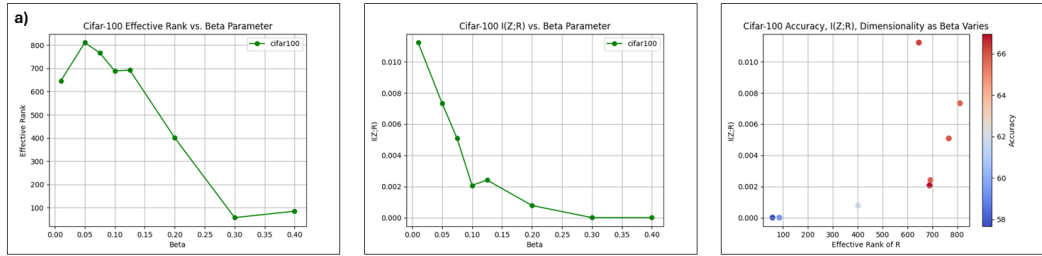


Figure 17: In this figure, we show how performance varies as we manually increase β with positive values. a) This shows how the effective rank varies. b) This figure shows how $I(R; Z)$ varies. c) This figure shows how the performance varies.