A Optimal Embeddings

Recall the setting of self supervised learning as described in Balestriero and LeCun (5): given a dataset $\boldsymbol{X'} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times D'}$ we construct a new dataset by creating k randomly augmented views of the original data, $\boldsymbol{X} = [\text{view}_1(\boldsymbol{X'}), ..., \text{view}_k(\boldsymbol{X'})] \in \mathbb{R}^{Nk \times D}$. The advantage of doing so is that we can now leverage the knowledge that different views of the same underlying datapoint are *semantically related*. We can express this notion of similarity in the symmetric matrix $\boldsymbol{G} \in \{0, 1\}^{Nk \times Nk}$ with $\boldsymbol{G}_{ij} = 1$ if augmented datapoints i and j are semantically related (and $\boldsymbol{G}_{ii} = 1$ as any datapoint is related to itself). We can normalize $\boldsymbol{G}$ such that its rows and columns sum to 1 (so rows of $\boldsymbol{G}$ are k-sparse with nonzero entries equal to $1/k$).

Now let $\boldsymbol{Z} \in \mathbb{R}^{Nk \times d}$ be an embedding of the augmented dataset. Then we have $\boldsymbol{GZ} = [\boldsymbol{C}, ..., \boldsymbol{C}]^T$ where $\boldsymbol{C}$ is the matrix of centroid vectors introduced above, and the number of repetitions of $\boldsymbol{C}$ is k. Then because $\boldsymbol{\sigma}([\boldsymbol{C}, ..., \boldsymbol{C}]) = \sqrt{k}\boldsymbol{\sigma}(\boldsymbol{C})$ we can write MMCR loss function as,

$$\begin{aligned} \mathcal{L} &= -||\boldsymbol{GZ}||_* \\ &= -||\boldsymbol{Q\Lambda Q}^T\boldsymbol{USV}^T||_* \\ &= -||\boldsymbol{\Lambda Q}^T\boldsymbol{US}||_* \end{aligned} \tag{6}$$

Where we have taken the eigendecomposition of $\boldsymbol{G}$ which is real and symmetric and the SVD of $\boldsymbol{Z}$, and then used the fact that the singular value spectrum is invariant under left or right orthogonal transformations. We now show that a global optima of this objective is achieved when the left singular vectors of $\boldsymbol{Z}$ are the eigenvectors of $\boldsymbol{G}$ and the singular values of $\boldsymbol{Z}$ are proportional to the eigenvalues of $\boldsymbol{G}$. Throughout we will assume that the size of the dataset is greater than the dimensionality of the embedddings, $N > d$, as is the case in practical applications. First we prove a simple lemma about the spectrum of matrices who are extended by zeros (i.e. embedded in a higher dimensional space).

Lemma A.1: For $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, $\boldsymbol{B} \in \mathbb{R}^{N \times d}$ with $d < N$, $||\boldsymbol{AB}||_* = ||\boldsymbol{A}\tilde{\boldsymbol{B}}||_*$ where $\tilde{\boldsymbol{B}} = [\boldsymbol{B}, \boldsymbol{0}] \in \mathbb{R}^{N \times N}$.

Proof: First note that $\boldsymbol{A}\tilde{\boldsymbol{B}} = [\boldsymbol{AB}, \boldsymbol{0}]$ so it suffices to show that for arbitrary $\boldsymbol{X}$ that $\boldsymbol{\sigma}(X) = \boldsymbol{\sigma}([\boldsymbol{X}, \boldsymbol{0}])$. Taking the SVD of $\boldsymbol{X}$,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{U} & \widetilde{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma} \\ \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}^T \end{bmatrix} = \boldsymbol{U\Sigma V}^T$$

Then a valid singular value decomposition for $\tilde{X}$ is

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{U} & \widetilde{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

Clearly then, $||\boldsymbol{X}||_* = ||\tilde{\boldsymbol{X}}||_*$

Theorem: The proposed loss achieves a global minimum when the left singular vectors of $\boldsymbol{Z}$ are the eigenvectors of $\boldsymbol{G}$, and the singular values of $\boldsymbol{Z}$ are proportional to the top d eigenvalues of $\boldsymbol{G}$.

Proof: Let $\tilde{\boldsymbol{Z}} = [\boldsymbol{Z}, \boldsymbol{0}] \in \mathbb{R}^{N \times N}$. By Lemma A.1 we have $||\boldsymbol{GZ}||_* = ||\boldsymbol{G}\tilde{\boldsymbol{Z}}||_*$. Von Neumann's trace inequality can be used to show $||\boldsymbol{G}\tilde{\boldsymbol{Z}}||_* \leq \sum_{i=1}^{Nk} \boldsymbol{\sigma}_i(\boldsymbol{G})\boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}})$ (see Marshall et al. (48) for proof). Examining (4) it is clear that this bound is achieved when $\boldsymbol{U} = \boldsymbol{Q}$. The problem can therefore be reduced to the constrained optimization problem,

$$\min_{\boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}})} \sum_{i=1}^{Nk} \boldsymbol{\sigma}_i(\boldsymbol{G})\boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}})$$

$$\text{subject to} \sum_{i=1}^{Nk} \boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}})^2 = Nk$$

where the constraint comes from the fact that columns of $\boldsymbol{Z}$ are unit vectors. Intuitively, we are maximizing the inner product between a fixed vector $\boldsymbol{\sigma}(\boldsymbol{G})$ and a vector with fixed L2 norm. The solution of course is to align the two vectors as closely as possible, i.e. when $\boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}}) \propto \boldsymbol{\sigma}_i(\boldsymbol{G})$ for $i = 1, ..., d$. It is worth noting that by construction $\boldsymbol{\sigma}_i(\tilde{\boldsymbol{Z}}) = 0$ for $i > d$ and the columns of $\boldsymbol{U}$ associated with these zero valued singular values are unconstrained.

B Pytorch Style Pseudocode for MMCR

```
# h: encoder
# g: projection head
# T: momentum temperature
# B: batch size
# K: number of augmentations
# D: projector output dimensionality
#
# lmbda: trade-off parameter

f_o, g_o = ResNet50(), MLP() # online networks

# initialize momentum network with identical params
f_m, g_m = f_o.copy, g_o.copy()

# momentum networks are not updated via gradient descent
f_m.requires_grad = False
g_m.requires_grad = False

for x in loader:
    # K randomly augmented views
    x = multi_augment(x) # B x K x H x W

    # push through encoder and projector
    z_o = g_o(h_o(x)) # B x K x D
    z_m = g_m(h_m(x)) # B x K x D
    z = concatenate(z_o, z_m, dim=1) # append outputs

    # project onto unit sphere
    z = normalize(z, dim=-1)

    # calculate centroids (mean over augmentation axis)
    c = z.mean(dim=1) # B x D

    # calculate singular values
    U_z, S_z, V_z = svd(z) # batch svd
    U_c, S_c, V_c = svd(c)

    # calculate loss
    loss = -1.0 * sum(S_c) + lmbda * sum(S_z) / B

    # backward pass and optimization step
    loss.backward()
    optim.step()

    # perform momentum update
    with torch.no_grad():
        f_m.parameters() = (1 - T) * f_o.parameters() + T * f_m.
    parameters()
        g_m.parameters() = (1 - T) * g_o.parameters() + T * g_m.
    parameters()
```

C Mean Field Theory Manifold Capacity Background Information

For completeness we summarize some of the central arguments from Chung et al. (16), which develops the general form of manifold capactiy theory.

Mean Field Theory Recall the problem setting for manifold capacity analysis: given a set of P manifolds embedded in a feature space of dimensionality D, each assigned a random binary class label (16). Manifold capacity theory is concerned with the question: what is the largest value of $\frac{P}{D}$ such that there exists (with high probability) a hyperplane separating the two classes? In the thermodynamic limit, where $P, D \to \infty$ but $\frac{P}{D}$ remains finite, the inverse capacity can be written exactly,

$$\alpha_M^{-1} = \mathbb{E}_{\vec{T}}[F(\vec{T})] \tag{7}$$

where, $F(\vec{T}) = \min_{\vec{V}} \left\{ \|\vec{V} - \vec{T}\|^2 \mid g_{\mathcal{S}}(\vec{V}) \geq 0 \right\}$, $\mathcal{S}$ is the set defining the manifold geometry (i.e. the set of vectors $\vec{S}$ that are points on an individual manifold), $\vec{T}$ are random vectors drawn from a white multivariate Gaussian distribution, and $g_{\mathcal{S}}(\vec{V}) = \min_{\vec{S}}\{\vec{V} \cdot \vec{S} \mid \vec{S} \in \mathcal{S}\}$, is the concave support function.

The KKT equations for this convex optimization problem are:

$$\begin{aligned} \vec{V} - \vec{T} - \lambda \tilde{S}(\vec{T}) &= 0 \\ \lambda &\geq 0 \\ g_{\mathcal{S}}(\vec{V}) - \kappa &\geq 0 \\ \lambda \left[g_{\mathcal{S}}(\vec{V}) - \kappa \right] &= 0. \end{aligned} \tag{8}$$

, where $\tilde{S}(\vec{T})$ is a subgradient of the support function. When the support function is differentiable, the subgradient is unique and equal to the gradient,

$$\tilde{S}(\vec{T}) = \nabla g_{\mathcal{S}}(\vec{V}) = \arg\min_{\vec{S} \in \mathcal{S}} \vec{V} \cdot \vec{S} \tag{9}$$

$\tilde{S}(\vec{T})$ is the unique point in the convex hull of $\mathcal{S}$ that satisfies the first KKT equation, and is called the “anchor point" for $\mathcal{S}$ induced by the random vector $\vec{T}$.

Equivalent Interpretation of Anchor Points For a given dichotomy (random binary class labelling) the weight vector of the maximum margin separating hyperplane can be decomposed into a sum of at most P vectors, with each manifold contributing a single vector, which lies within the convex hull of the manifold. The position of said point point is a function of the manifolds position relative to all of the other manifolds in the space and depends on the particular set of random labels. Thus there exists a distribution of separating-hyperplane-determining-points for each individual manifold. Using the cavity method it can be shown that these points are none other than the anchor points that are involved in solving the optimization problem described above (31).

Numerical Solution To solve the mean field equations numerically, one samples several random Gaussian vectors $\vec{T}$, and then for each $\vec{T}$, $\vec{V}$ and $\vec{S}$ are determined by solving the quadratic programming program given above. The capacity is then estimated as the mean value of F or the samples $\vec{T}$.

Manifold Geometries The way the capacity varies in terms of the statistics of the anchor points can be simplified by introducing two key quantities, the manifold radius R_M and manifold dimensionality R_M:

$$\begin{aligned} R_M^2 &= \mathbb{E}_{\vec{T}}[||\tilde{S}(\vec{T})||^2] \\ D_M &= \mathbb{E}_{\vec{T}}[\vec{T} \cdot \hat{S}(\vec{T})] \end{aligned} \tag{10}$$

where $\hat{S}(\vec{T})$ is a unit-vector in the direction of the anchor point $\tilde{S}$. In particular as discussed in the main text, the manifold capacity can be approximated by $\phi(R_M \sqrt{D_M})$ where ϕ is a monotonically decreasing function.

Elliptical Geometries In the case where the manifolds exhibit elliptical symmetries, the manifold radius and dimensionality can be written in terms of the eigenvalues of the covariance matrix of the anchor points:

$$\begin{aligned} R_M^2 &= \sum_i \lambda_i^2 \\ D_M &= \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2} \end{aligned} \tag{11}$$

So, in this case R_M is the total variability of the anchor points, and D_M is a generalized participation ratio of the anchor point covariance, a well known soft measure of dimensionality.

D Additional Pre-training information

Settings for CIFAR/STL-10 We take the parameters of each augmentation directly from Zbontar et al. (63), but for these lower resolution images we omitted Gaussian blurring and solarization augmentations. All models were trained for 500 epochs using the Adam optimizer (40) with a learning rate of $1\text{e}-3$ and weight decay of $1\text{e}-6$. For all three methods we used a one hidden layer MLP with hidden dimension of 512 and output dimension of 128 for the projector head g. We swept batch size for each method and chose the one that resulted in the highest downstream task performance. For both SimCLR and Barlow Twins we found that a batch size of 128 was optimal (among 32, 64, 128, 256, and 512) for all 3 datasets. For MMCR there is a trade-off between batch size and the number of augmentations used, and the optimal value of that trade-off is highly dataset dependent. For CIFAR-10 and CIFAR-100 we used batch size of 32 and 40 views, and for STL-10 we used a batch of 64 with 20 views For Barlow Twins we used $\lambda = \frac{1}{128}$ which normalizes for the number of elements in the on-diagonal and off-diagonal terms in the loss. For SimCLR we used the recommended setting of $\tau = 0.5$. The overall performance of both baseline methods (and likely MMCR as well) could be increased with a more thorough hyperparameter search and by employing methodology that more closely matches the original works. For example, both methods would likely benefit from the combination of larger batch size, the use of the LARS optimizer (which is designed for large batch optimization), a learning rate scheduler consisting of linear warm-up followed by cosine annealing, longer training, and the use of more diverse augmentations (i.e. including solarization and gaussian blur). Additionally Barlow Twins reports that the representation can benefit from using a much larger projector network than we use. Because our goal was primarily to demonstrate that MMCR can produce representations that are comparable to these baselines rather than to produce state-of-the-art results on small scale datasets we opted for simplifications wherever possible (using off the shelf Adam for optimization with a fixed learning rate, and fixing architectural hyperparameters like the projector dimensionality).

Settings for ImageNet-100 For ImageNet we more closely match the pre-training procedures of previous works. We use a batch size of 2048 and a smaller number of views for MMCR (4), and also use the full suite of augmentations from Zbontar et al. (63). For the sake of efficiency we train for a reduced number of epochs (200). For MMCR and SimCLR we modified the projector hidden dimensionality to be 4096 for the projector head, following the original work (12). For Barlow Twins we used the recommended 2-layer MLP with hidden and output dimensions of 8192, and set $\lambda = 5e-3$, however these hyperparameters were optimal for the full ImageNet dataset, and not neccesarrily for ImageNet-100. We were unable to achieve better downstream performance using a ResNet-50 backbone than what has previously been reported in the literature for this dataset with a ResNet-18 backbone, therefore we report the ResNet-18 performance reported in (20). For SimCLR we use $\tau = 0.1$ which is the recommended setting for larger batch sizes.

Settings for ImageNet-1k: For ImageNet-1k we use mostly identical settings to ImageNet-100, but we increased the capacity of the projector network (using a 2 hidden layer MLP with hidden dimenisons of 8192 and output dimension of 512). We scaled the learning rate linearly with batch size: $\text{lr} = 0.6 \times \frac{\text{batch size}}{256}$. Additionally we reduce the number of pretraining epochs to 100. Finally for ImageNet-1k we found that employing a momentum encoder slightly boosted downstream performance (around +0.5% on ImageNet frozen-linear evaluation). Specifically, an identical encoding network and projector architecture is initialized with the same parameters as the initial "online" network, and during training the weights of this "momentum" network track a slowly moving average of the online network parameters (only the online network parameters are updated via gradient descent). Each augmented view is passed through both the online and momentum networks, and the resultant embeddings are all averaged to form the centroid vector for a particular image in the batch. We used a momentum coefficient of 0.99.

Pre-training on 16 A100 GPUs using 8 views (our most compute intensive setting) takes approximately 32 hours.

E Details of Representational Analyses

E.1 Manifold Capacity Analysis

For each pre-trained model, we extract layer activations across the ResNet hierarchy after a forward pass of a set of images. For class manifold analysis, the set of images contain 10 classes, where each class has 100 examples. Augmentation manifolds instead have 100 exemplars with 100 examples each. Following (18), we take activations from all convolutional layers in ResNet-50 after a ReLU non-linearity. The specific extracted layers highlighted in bold fonts are given by Table 3. The final analysis results are averaged over five data samplings with different random seeds and random projections of intermediate features to lower-dimension spaces (default 5000 dimensions).

E.2 Gradient Coherence Analysis

In Fig. 3, for each of the classes of CIFAR-10, we generate 100 batches of 32 augmentation manifolds of samples from a specific class (with 40 augmentations each). We then measure the gradient of the loss function for each batch during different stages of training, and compute the cosine similarity between every pair of gradients. Across all stages of training the mean cosine similarity between gradients generated from batches of the same class is larger than those from distinct classes (left column). This observation remains true when isolating the gradients of parameters from different stages of in the resnet-50 hierarchy (center and right columns, respectively).

E.3 Manifold Subspace Alignment

For Fig. 4 we generated 100 samples from the augmentation manfiolds of 500 images in the CIFAR-10 dataset. We then measure the mean subspsace angle (left column), fraction of shared variance (middle column) and centroid cosine similarity between each pair of manifolds. The same procedure was used for generating the data for Fig. 5.

Subspace Angle. Besides measuring the size and dimensionality of individual object manifolds we also wish to characterize the degree of overlap between pairs of manifolds. For this, we measure the angle between their subspaces (41), which is a generalization of the notion of angles that applies to subspaces of arbitrary dimension.

Shared Variance. Object manifolds will generally have a lower intrinsic dimensionality then the space in which they are embedded. Therefore, the data will have low variance along several of the principal vectors used to calculate the set of subspace angles, and so many of the principal angles will have little meaning. To address this limitation we also compute the shared variance between the linear subspaces that contain object manifolds.

F Implicit MMCR Effectively Reduces Augmentation Manifold Nuclear Norm

To test whether or not implicit manifold compression actually reduces the mean augmentation manifold nuclear norm, we can vary the value of λ. Below we see the evolution of both terms of the loss for several different values of lambda during training on CIFAR-10. For these experiments the batch size was 64 and the number of augmentations per image was 4.0. As shown in Fig. 6, the level of compression of individual manifolds is nearly the same across all values of the parameter.

Table 3: A Total of 18 Extracted ResNet-50 Layers (in **Bold**) for MFTMA Analysis

Layer	Type	Conv2d Size (H × W × C)
pixel	**Input**	None
conv1	$\begin{bmatrix}[l]\text{Conv2d}\\ \text{BatchNorm}\\ \textbf{ReLU}\end{bmatrix}\times 1$	$[[l]7\times 7\times 64]\times 1$
conv2_x	$\begin{bmatrix}[l] & \begin{bmatrix}[l]\text{Conv2d}\\ \text{BatchNorm}\end{bmatrix}\times 3\\ & \textbf{ReLU}\end{bmatrix}\times 3$	$\begin{bmatrix}[l]1\times 1\times 64\\ 3\times 3\times 64\\ 1\times 1\times 256\end{bmatrix}\times 3$
conv3_x	$\begin{bmatrix}[l] & \begin{bmatrix}[l]\text{Conv2d}\\ \text{BatchNorm}\end{bmatrix}\times 3\\ & \textbf{ReLU}\end{bmatrix}\times 4$	$\begin{bmatrix}[l]1\times 1\times 128\\ 3\times 3\times 128\\ 1\times 1\times 512\end{bmatrix}\times 4$
conv4_x	$\begin{bmatrix}[l] & \begin{bmatrix}[l]\text{Conv2d}\\ \text{BatchNorm}\end{bmatrix}\times 3\\ & \textbf{ReLU}\end{bmatrix}\times 6$	$\begin{bmatrix}[l]1\times 1\times 256\\ 3\times 3\times 256\\ 1\times 1\times 1024\end{bmatrix}\times 6$
conv5_x	$\begin{bmatrix}[l] & \begin{bmatrix}[l]\text{Conv2d}\\ \text{BatchNorm}\end{bmatrix}\times 3\\ & \textbf{ReLU}\end{bmatrix}\times 3$	$\begin{bmatrix}[l]1\times 1\times 512\\ 3\times 3\times 512\\ 1\times 1\times 2048\end{bmatrix}\times 3$

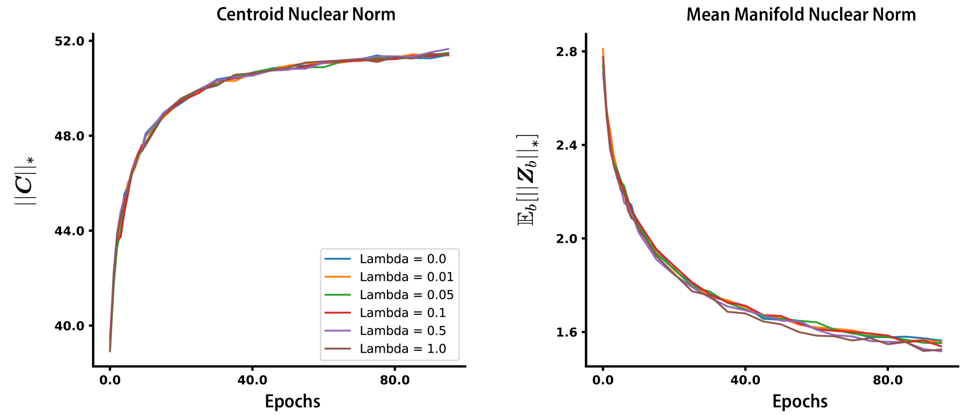

Figure 6: Validation loss values for different values of λ

G Classification Evaluation Procedure

CIFAR and STL-10: During pre-training all models were monitored with a k-nearest neighbor classifier (k=200) and checkpointed every 5 epochs. After pre-training, we trained linear classifiers on all checkpoints whose monitor accuracy was within 1% of the highest observed accuracy, and select the model that achieves the highest linear classification accuracy. Linear classifiers were trained using the Adam optimizer with batch size of 1024 and an initial learning rate of 0.1, which decayed according to a cosine scheduler over the course of 50 epochs. For the linear classifier training, at train time we use the same set of augmentations as during unsupervised pretraining, at test time we only use center cropping and random horizontal flipping.

ImageNet-1k/100: For ImageNet datasets we closely followed the most widely adopted evaluation procedure. Following pre-training we freeze the encoder weights and train a linear layer in a supervised fashion using SGD with a batch size of 256, learning rate of 0.3, and weight decay of 1e-6 for 100 epochs. During linear classifier training the only data augmentations are random cropping and random horizontal flips, and during evaluation inputs are center cropped.

Semi-Supervised: For semi-supervised evaluation we mostly follow the procedure outlined in Bardes et al. (6). We use the SGD optimizer with momentum of 0.9 and weight decay of 1e-6 and the standard cross entropy loss. The augmentation procedure was the same as described above. Because the linear classifier is being trained from scratch and the representation is being fine tuned the learning rate for the parameters of the backbone is scaled down by a factor of 10, and both learning rates (backbone and classifier) followed a cosine decay schedule for 20 epochs. We used a batch size of 256 and swept the learning rate over [0.1, 0.3, 1.0] for each model.

Other Downstream Classification Tasks: Classifiers on these datasets were trained in a similar fashion to those trained on the CIFAR and STL-10 datasets. The only difference was that we trained for only 20 epochs (which we found sufficient for convergence), the initial learning rate over [3e-2, 3e-3, 3e-4] for each model, and the augmentation procedure matched the standard setup for ImageNet training (only random cropping and horizontal flipping for training, resizing and center cropping for evaluation).

H Training Metrics

In the Fig. 7 below we monitor the evolution of both the objective (second panel), the mean augmentation manifold nuclear norm, the centroid norm, and the mean centroid similarity evaluated on the test set over the course of training.

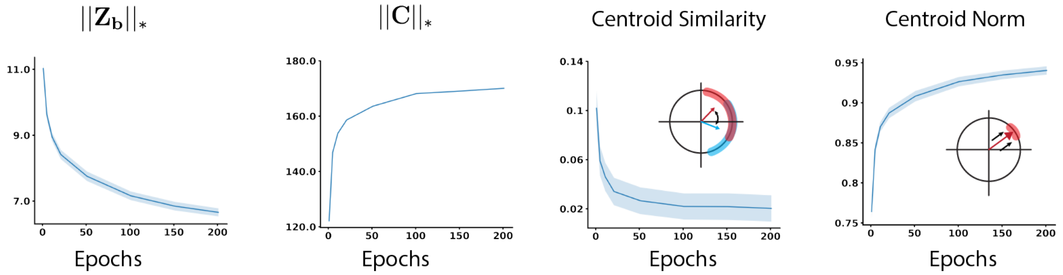

Figure 7: Evolution of various metrics during training. Geometric measures are evaluated on a set of 200 manifolds, each defined by an image drawn from the CIFAR-10 dataset, along with 16 augmentations. Shaded regions indicate a 95% confidence interval around the mean.

I Classification Performance on Smaller Datasets

In Table 4 below we report the performance of both our method as well as Barlow Twins and SimCLR when trained using a ResNet-50 backbone on smaller datasets.

J Batch Size Dependence

One of the most cited drawbacks of contrastive SSL methods has been that strong performance on downstream tasks requires training with large batch sizes, while non-contrastive methods (e.g., VICReg or Barlow Twins (63; 6)) that place constraints on the cross-correlation/covariance matrices of the embeddings are much more amenable to smaller batch training. It is also worth noting that the need for large batch sizes in contrastive methods can be alleviated in various ways, such as maintaining a memory bank (61) or employing a slowly updating momentum encoder (36). Given that our method is neither wholly contrastive nor non-contrastive (since it acts on the spectrum of the embedding matrix directly), we wondered howwould depend on training batch size. We pretrained on ImageNet-1k using batch sizes of $256, 512, 1024, 2048, 4096$ and evaluate the linear classification

Table 4: Top-1 classification accuracies of linear classifiers for representations trained with various datasets and objective functions. Note: for Barlow Twins on ImageNet-100 we report the result from da Costa et al. (20) which uses a ResNet-18 backbone, as we were unable to obtain better performance. For MMCR on ImageNet-100 we tested both 2 views (matched to baselines) and 4 views, results are formatted (2-view)/(4-view)

[t]

Method	CIFAR-10	CIFAR-100	STL-10	ImageNet-100
Barlow Twins (our repro.)	90.91	67.91	89.96	80.38*
SimCLR (our repro.)	92.22	70.04	91.11	79.64
MMCR ($\lambda = 0.0$)	93.53	69.87	90.62	81.52/82.88
MMCR ($\lambda = 0.01$)	93.39	70.94	90.77	81.28/82.56

accuracy for each. Encouragingly we observed only a modest decrease in performance for the smallest batch size tested. The results of this sweep, in comparison to Barlow Twins and SimCLR, is shown in in Fig. 8 Note that for these runs we used two views and the linear learning rate scaling as described in Appendix D. Future work should endeavor to better understand the impact of various hyperparameters on the quality of learned representations.

An important detail is that for this experiment we did not employ a momentum encoder when training MMCR. It is argued in He et al. (36) that the momentum encoder increases the effective minibatch size as the slowly moving weights encode information from preceding batches. However here we are interested explicitly interested in batch size dependence so we ablate this architectural confound (neither Zbontar et al. (63); Chen et al. (12) employ momentum encoders).

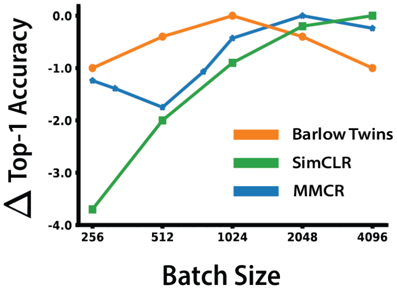

Figure 8: Drop in top-1 performance relative to that of the best setting for three methods. Data for both Barlow Twins and SimCLR are copied from Zbontar et al. (63).

K Additional Details on BrainScore

Brain-Score evaluates a model in terms of its ability to predict the measured responses of neurons to images. We provide a brief introduction to the metric here (see (54) for a complete description). Let $\boldsymbol{y} \in \mathbb{R}^N$ denote the average response of a single (biological) neuron to a set of N training images and $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ be the response of K model neurons to the same images. Brain Score first solves the linear regression problem $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$ for weights $\boldsymbol{w}$. The model then predicts responses $\boldsymbol{y}'$ to a set of held out images. Next the Pearson correlation coefficient between $\boldsymbol{y}'$ and y is calculated, and the score for a particular dataset is the median of the individual neuron predicitivities in said dataset. The mean of these scores is taken over different train-test splits of each dataset. The score for a total brain area (as shown in Table 2 is the mean over train-test splits and distinct datasets. The standard error of the means for each dataset (which are typically between 1e-3 and 1e-2) within an area are summed quadrature and divided by the number of datasets to produce the errors reported in table 2. In table 5 we split the brain area scores into individual datasets.

Model	V1.0	V1.1	V2.0	V4.0	V4.1	V4.2	V4.3	IT.0	IT.1	IT.2	IT.3
MMCR	0.270	0.718	0.311	0.577	0.627	0.492	0.226	0.554	0.558	0.545	0.424
SimCLR	0.224	0.776	0.288	0.576	0.626	0.48	0.224	0.552	0.545	0.518	0.456
BYOL	0.274	0.727	0.291	0.585	0.626	0.48	0.216	0.55	0.545	0.516	0.41
MoCo	0.273	.726	0.293	0.57	0.629	0.492	0.215	0.54	0.560	0.550	0.437
Barlow	0.276	.721	0.293	0.568	0.626	0.493	0.221	0.545	0.547	0.518	0.412
SwAV	0.252	.723	0.296	0.568	0.614	0.469	0.202	0.533	0.537	0.518	0.405

Table 5: Brain-score comparison of six self-supervised models, over 11 different electrophysiological data sets recorded from macaque monkeys (54). Datasets V1.0 and V2.0 are from Freeman et al. (26), V1.1 is from Marques et al. (47), and V4.0, V4.1, IT.0, and IT.1 are from Majaj et al. (45). (V4/IT).(2/3) are the SanghaviJozwik2020 and SanghaviMurty2020 datasets as denoted by brainscore.

L Additional Details on Spectral Properties

Participation Ratio We first extracted the 2048 dimensional feature vectors for each model in response to the images in the ImageNet validation set. Images were resized to 256×256 and then center cropped to 224×224 following the setting in which the classifiers are tested. We resampled the resultant feature matrices with replacement 10 times independently for each model For each resampled dataset (of features) we calculate the empirical covariance matrix, associated eigenspectra, and associated participation ratios (squared ratio of L_1 to L_2 norm of the eigenvcetors).

Decay Coefficient The spectra obtained using the procedure outlined above all decayed rapidly near the tails (the least significant eigenvalues). To avoid undo bias from these tails when estimating the decay coefficients we only considered the top 2000 eigenvalues. To estimate the decay coefficient we fit a regression line to the logarithm of the eigenvalues as a function of the logarithm of their indexes. This fitting procedure was repeated across the bootstrapped spectra for each model to obtain standard errors of the mean. For all models the linear regression produced a strong fit to the data, with the minimum observed R^2 value being 0.95.

M Object Detection

To ensure that the representations obtained with MMCR are not hyperspecialized to classification tasks we also evaluated our highest performing model on object detection. We follow (36; 63), fine tuning the representation network with a Faster R-CNN head and C-4 backbone on the VOC07+12 dataset (training with the train+val split and evaluating on the VOC07 test split). All settings except for the initial learning rate (which we set to 0.12) were identical to those from He et al. (36), and we similarly used the detectron2 library for this evaluation. A representation trained using MoCo v2 for 200 epochs achieves an AP50 (the most common evaluation metric for this dataset) of 82.4. MMCR with 8 views and 100 epochs of pretrainined produced an AP50 of 81.9 (this is the mean over three independent fine tunes, the standard deviation was 0.2), demonstrating that the representeations generated by our method are not limited to object recognition.

N Limitations

Time and compute limitations prevented us from conducting exhaustive optimization of all design choices. For example we do not explore effects of varying the projector network width and depth. We additionally restrict the model to a ResNet-50 encoding network pretrained on the ImageNet-1k dataset for 100 epochs. This choice allows us to make fair comparisons to several recently developed

SSL methods with a modest compute budget, but precludes answering questions about how the method scales up to larger problems.

We also note that although our objective function has favorable computational complexity compared to existing methods, the evaluation of our objective is not straightforward to distribute across machines. This is because we need to compute the SVD over the centroid matrix, which requires gathering the outputs of a distributed forward pass onto a single machine. Efficient methods for the distributed computation of the SVD could help alleviate this issue in the future.