Squeezing Water from a Stone: Improving Pre-Trained Self-Supervised Embeddings Through Effective Entropy Maximization

Deep Chakraborty
UMass Amherst

Tim G. J. Rudner
New York University

Erik Learned-Miller
UMass Amherst

Abstract

A number of different architectures and loss functions have been applied to the problem of self-supervised learning (SSL), with the goal of developing embeddings that provide the best possible pre-training for as-yet-unknown, lightly supervised downstream tasks. One of these SSL criteria is to maximize the entropy of a set of embeddings in some compact space. But the goal of maximizing the embedding entropy often depends—whether explicitly or implicitly—upon high dimensional entropy estimates, which typically perform poorly in more than a few dimensions. In this paper, we motivate a simple maximum-entropy criterion, defined in terms of easy-to-estimate, low-dimensional constraints, and demonstrate that using it to continue training an already-trained SSL model for only a handful of epochs leads to a consistent and, in some cases, significant improvement in performance.

1 Introduction

In this paper, we formulate a well-motivated, general-purpose criterion that allows *further improving* already-trained SSL embeddings with only a handful of epochs of *continued pre-training*.

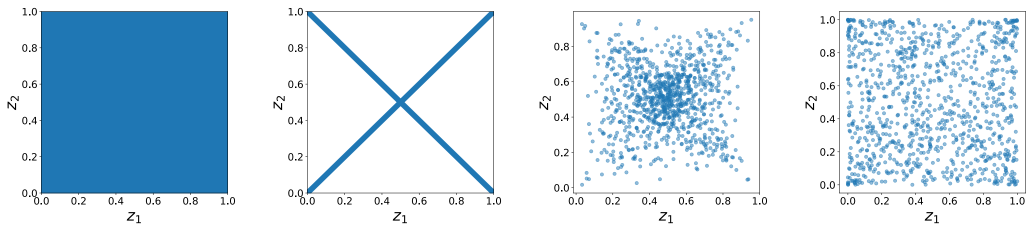
We start by revisiting the problem of how best to distribute SSL embeddings to maximize performance on downstream tasks. We take a novel approach to the idea of maximizing the entropy of embeddings in a compact embedding space: Rather than explicitly focusing on maximizing joint entropy or other properties of the full joint distribution of embeddings, we instead focus on the embedding marginals, in which we can tame the curse of dimensionality. To ensure that maximizing the marginal entropies results in useful embeddings, we simultaneously enforce simple, low-dimensional constraints that are necessary, but not sufficient, to guarantee joint uniformity over the entire embedding space. Surprisingly, we find that—without explicitly enforcing higher-dimensional constraints in our criterion—higher-order marginals of our embeddings naturally tend towards uniformity, resulting in practically useful embeddings.

The main contributions of this paper are as follows:

1. We consider the problem of further improving already-trained SSL embeddings.
2. We motivate a simple maximum-entropy optimization criterion grounded in information-theoretic principles and show that it can be used as an add-on criterion for popular SSL methods.
3. We perform an empirical evaluation and show that with only a handful of epochs of continued pre-training under the proposed criterion, we achieve consistent and, in some cases, significant improvements in downstream-task performance across a selection of computer vision tasks.

2 Related Work

Self-supervised learning (SSL) methods share a common objective of bringing two 'similar' input views (say, translated versions of the same image) closer together in the representation space, while



(a) (b) (c) (d)

Figure 1: **(a)**. A 2-d uniform distribution. **(b)** An "X" distribution. Both distributions have uniform (max-entropy) marginals and decorrelated components, and minimize our loss function. **(c)** Embedding distribution over a random pair from VICReg [1] (after transformation to compact space). **(d)** Our embeddings, where empirical results show distributions with uniform 2-d marginals despite the fact that this is not *explicitly* enforced by our loss.

spreading apart different images, either explicitly [4] or implicitly [9]. Most methods also regularize their representations to prevent trivial solutions, avoid information collapse, or impose specific topologies. For example, VICReg [1] minimizes the covariance between embedding dimensions to prevent encoding redundant information, in addition to preserving the variance of the components to prevent collapse. Other information maximization approaches [16, 12] have also been proposed to spread embeddings effectively in the representation space by maximizing approximate measures of entropy. More closely related to our work are NAT [2] and AUH [21] that operate on or mimic properties of samples of the joint uniform distribution.

Our maximum-entropy criterion also prevents information collapse in a similar way by mimicking certain properties of a fully uniform (and hence maximum entropy) distribution in a compact embedding space. The key difference between our method and all of the others is that we enforce properties of the 1- and 2-dimensional marginals of the distribution, rather than operating directly on properties of the joint distribution. For example, AUH evaluates the gaussian potentials between points from the high dimensional joint distribution over embeddings rather than of single-dimensional marginal distributions, which might make it less sample efficient compared to ours. In our experiments, we find that our embeddings are better suited to downstream tasks, and can be obtained quickly by adapting pre-trained SSL embeddings to get consistent improvements in performance.

3 A Maximum-Entropy Self-Supervised Learning Augmentation Criterion

In this section, we motivate a simple maximum-entropy SSL augmentation criterion that can be used to improve already-trained SSL embeddings with only a handful of epochs of continued pre-training.

We consider Siamese style neural networks f_θ (encoder) to compute representation vectors $Y = f_\theta(X)$ and $Y' = f_\theta(X')$, where X, X' are two input image views. These representation vectors are then further transformed by an MLP g_θ (projector) to produce the final embeddings $Z_\theta = g_\theta(Y)$ and $Z'_\theta = g_\theta(Y')$. Once training is complete, the projector is discarded, and the representation vector Y is used for downstream tasks. Our goal is to take any such pre-trained SSL model and update its representations by pre-training it for additional epochs using an effective maximum-entropy criterion.

Unlike all the other methods of which we are aware, we focus only on properties of the one- and two-dimensional marginal distributions, and speculate that by focusing on properties that are more reliably estimated with moderate sample sizes, we might be able to obtain a more useful criterion.

To motivate an effective maximum-entropy criterion, we start with an observation that the following facts about distributions over the unit cube are mathematically equivalent [6]:

1. The joint distribution has maximum joint entropy.
2. The joint distribution is uniform.
3. The one-dimensional marginal distributions are maximum entropy (i.e., uniform) *and* the components are mutually independent.

We use the third characterization to design our loss function. This characterization for formulating a self-supervised learning criterion requires (i) an effective approach to estimating one-dimensional marginal distributions and (ii) a method for encouraging mutual independence.

To obtain a good estimate of the marginal entropies, we leverage the m-spacings estimator, which is statistically efficient, easy to compute, and differentiable [20, 11]. Unfortunately, mutual independence of the components is a property of the joint distribution, and we believe that it is too high-dimensional to achieve directly. Instead, we consider a *necessary, but not sufficient, condition* for mutual independence: decorrelation of all pairs of embedding dimensions. Criteria that serve this purpose have been used in both VICReg and other SSL methods to attempt to move embeddings towards independent features [1, 13] but not, to our knowledge, in conjunction with the idea of maximizing marginal entropies.

Unfortunately, enforcing decorrelation of all pairs of embedding dimensions does not guarantee mutual statistical independence. Hence, maximizing marginal entropies while decorrelating embedding dimensions is not sufficient to guarantee maximum entropy of the joint distribution. We nevertheless press on and ask: What kinds of distributions have maximal marginal entropy *and* are decorrelated, but do not have maximum joint entropy? We approach this question using a visual example for exposition, shown in Figure 1. Part (a) shows a two-dimensional uniform distribution, which maximizes the joint entropy and minimizes our loss function. Part (b) is what we call the "X" distribution, which also has both uniform marginals and diagonal covariance (i.e., no correlations between components). In principle, either of these distributions could emerge under the criterion described above. Part (c) shows a 2-d marginal of VICReg, which is clearly non-uniform. Surprisingly, our maximum-entropy criterion, which enforces uniformity only of 1-d marginals, also produces nearly uniform 2-d marginals as shown in (d), instead of alternatives like the "X" distribution. One possible explanation could be that the inductive bias of such deep networks might make it difficult to produce non-smooth distributions like the "X" distribution.

3.1 Specifying a maximum-entropy augmentation criterion

To define a specific criterion from the discussion above, we first transform embedding samples $Z \in \mathbb{R}^d$ to lie in a compact space, and consider the transformed embedding random variable $\check{Z} \in [0,1]^d$ instead, for applying our criterion.[1] Finally, given an arbitrary SSL method pre-trained using loss function $\mathcal{L}^{\text{SSL}}(\theta)$, we define the constrained optimization problem

$$\min_\theta \mathcal{L}^{\text{SSL}}(\theta) \qquad \text{subject to} \qquad \mathcal{L}^{\text{Entropy}}(\theta) \geq C_1 \quad \text{and} \quad \mathcal{L}^{\text{Covariance}}(\theta) \leq C_2. \tag{1}$$

In practice, we express this objective equivalently as

$$\mathcal{L}(\theta) = \mathcal{L}^{\text{SSL}}(\theta) - \beta\mathcal{L}^{\text{Entropy}}(\theta) + \gamma\mathcal{L}^{\text{Covariance}}(\theta) \tag{2}$$

where $\beta, \gamma \in \mathbb{R}$ are hyperparameters. For transformed embeddings $\check{Z}_\theta$ and $\check{Z}'_\theta$ of views X and X', respectively, we have

$$\mathcal{L}^{\text{Entropy}}(\theta) = \frac{1}{d}\sum\nolimits_{j=1}^{d} \widehat{\mathcal{H}}_j(\check{Z}_\theta^1, ..., \check{Z}_\theta^n) + \widehat{\mathcal{H}}_j(\check{Z}_\theta'^1, ..., \check{Z}_\theta'^n). \tag{3}$$

And, letting $\check{\check{Z}}_\theta = \check{Z}_\theta - \frac{1}{n}\sum_{i=1}^n \check{Z}_\theta^i$ and $\check{\check{Z}}'_\theta = \check{Z}'_\theta - \frac{1}{n}\sum_{i=1}^n \check{Z}_\theta'^i$ for X and X',

$$\mathcal{L}^{\text{Covariance}}(\theta) = \frac{1}{nd}(\|(K_\theta - \text{diag}(K_\theta))\|_F^2 + \|(K'_\theta - \text{diag}(K'_\theta))\|_F^2), \tag{4}$$

where $||\cdot||_F$ is the Frobenius norm, and we defined $K_\theta = \check{\check{Z}}_\theta^\top \check{\check{Z}}_\theta$ and $K'_\theta = \check{\check{Z}}_\theta'^\top \check{\check{Z}}'_\theta$. We estimate the marginal entropies $\widehat{\mathcal{H}}_j$ using the m-spacings estimator (see Appendix A.3). Under this formulation, maximizing $\mathcal{L}^{\text{Entropy}}(\theta)$ maximizes the marginal entropies, and minimizing $\mathcal{L}^{\text{Covariance}}(\theta)$ corresponds to minimizing the squared off-diagonal entries of the sample covariance computed from the embedding. Complete implementation details can be found in Appendix B.

4 Empirical Evaluation

We select a base SSL method and do continued pre-training using our maximum-entropy criteria for exactly 10 epochs on top of a publicly available checkpoint, followed by evaluation of the *updated* representations on a downstream task. The full schematic is shown in Figure A.1 and pseudocode is in Algorithm 1. We focus on improving embeddings from three popular SSL methods: VICReg [1],

[1]For further details of this transformation, see Appendix A.2.

Table 1: **Evaluation of self-supervised embeddings.** We report Top-1-Accuracy (%) on ImageNet validation set using classifiers trained on SSL embeddings from various criteria under linear and semi-supervised evaluation paradigms. We also report transfer learning results from linear classifiers trained on iNat-18 and VOC07 datasets. Winning results in each category are shown in bold, along with standard errors computed over three random trials. † marks numbers taken from original papers, and * marks reproduced numbers that differ from original reported results despite best attempts.

	Linear Evaluation			Semi-supervised Learning		Transfer Learning	
Method (checkpoint)	1% labels	10% labels	100% labels	1% labels	10% labels	iNat18	VOC07
VICReg base [1] (1000 ep)	53.50 ±0.11	66.57 ±0.02	73.20†	54.53* ±0.12	67.97* ±0.03	47.00†	86.60†
VICReg continued (1010 ep)	53.51 ±0.07	66.57 ±0.06	73.16 ±0.02	–	–	–	–
VICReg **+ MaxEnt** (1010 ep)	**54.54** ±0.05	**66.82** ±0.05	**73.45** ±0.07	**55.05** ±0.08	**68.12** ±0.04	**47.18** ±0.11	**86.80**
SwAV base [3] (400 ep)	52.34 ±0.07	67.61 ±0.02	74.30†	52.57 ±0.15	69.25 ±0.05	46.00	**88.38**
SwAV continued (410 ep)	52.31 ±0.07	67.56 ±0.05	74.31 ±0.02	–	–	–	–
SwAV **+ MaxEnt** (410 ep)	**53.40** ±0.01	**67.73** ±0.03	**74.44** ±0.03	52.70 ±0.54	69.24 ±0.02	**46.71** ±0.17	88.24
SwAV base [3] (800 ep)	53.70 ±0.05	68.86 ±0.03	75.30†	53.89† ±0.13	70.22† ±0.05	49.08*	88.56*
SwAV continued (810 ep)	53.69 ±0.05	68.87 ±0.04	75.32 ±0.01	–	–	–	–
SwAV + AUH [21] (810 ep)	53.84 ±0.07	68.90 ±0.04	75.33 ±0.01	–	–	–	–
SwAV + VCReg [1] (810 ep)	54.02 ±0.05	68.88 ±0.03	75.36 ±0.02	–	–	–	–
SwAV **+ MaxEnt** (810 ep)	**55.27** ±0.07	**68.98** ±0.02	**75.41** ±0.02	53.94 ±0.30	70.32 ±0.05	**49.72** ±0.20	**88.69**
SimSiam base [5] (100 ep)	43.71 ±0.04	60.15 ±0.02	68.37*	–	–	38.75	**84.62**
SimSiam continued (110 ep)	43.78 ±0.05	60.23 ±0.08	68.45 ±0.08	–	–	–	–
SimSiam **+ MaxEnt** (110 ep)	43.78 ±0.06	60.23 ±0.07	**68.52** ±0.05	–	–	**38.99** ±0.20	84.54

SwAV [3], and SimSiam [5], chosen to cover a range of SSL techniques: feature decorrelation, online clustering and multi-crop, and feature distillation respectively. Moreover, these methods do not require negative examples in their loss, can work well with small batch sizes, and have official checkpoints and code available that can be easily modified to incorporate our criterion.

ImageNet evaluation. We evaluate methods on ImageNet [17] using the final representations (2048-d) from the ResNet-50 backbone [10] by (i) training a classifier on top of the frozen backbone (linear evaluation), and (ii) finetuning the whole backbone with a classifier on a subset of labels (semi-supervised evaluation). Table 1 shows the top-1 accuracy on the ImageNet validation set from classifiers trained on 1%, 10%, and (for linear evaluation) 100% of the available labels (using predefined splits from [4]). Our method "`[SSL]+MaxEnt`" surpasses almost all baselines from just 10 epochs of continued pre-training, with larger gains when training sets are smaller. For SwAV, the linear classifier trained on the *updated* 400-epoch model in the 1% setting is on par with the classifier trained on the base 800-epoch model (53.4% vs 53.7% respectively), which suggests our method has fast convergence. On SimSiam, our method shows little improvement over the baseline, which could be because the SimSiam checkpoint used is *less* trained compared to other methods (only 100 epochs), and gains from our method may only be visible for near-optimal pre-trained models.

Transfer learning. Following [14, 8], we show how our updated representations transfer to downstream tasks on other datasets. We show top-1 accuracy from linear classification on the challenging iNat18 dataset [19] (8000+ classes), and mAP for multi-label object classification on VOC07 [7]. We show consistent improvements for iNat18 and comparable performance to base models on VOC07.

Ablation study. To disambiguate performance gains from simply training longer, we train all base SSL methods for the same number of epochs as our method, but using only their original losses "`[SSL] continued`". No additional gains are seen in this setting. We also replace our criterion with other criteria such as the uniformity loss from AUH [21], or variance-covariance regularization from VICReg [1]. The gains from continued pre-training of the SwAV model with these other information criteria are inferior compared to ours. We hypothesize that VCReg is less effective at entropy maximization than us, and AUH is less sample efficient because it optimizes properties of the high dimensional joint distribution, instead of lower dimensional criteria such as ours. This makes our method more useful for quickly adapting existing SSL methods to produce better representations.

5 Conclusion

We proposed a simple add-on criterion for self-supervised learning, motivated by information-theoretic principles and applicable to any existing SSL method. We demonstrated empirically that the proposed criterion has desirable properties and that—with only a handful of epochs of continued pre-training—it is possible to achieve consistent and, in some cases, significant improvements in downstream-task performance across a selection of computer vision tasks.

Acknowledgments

The authors would like to thank Yann LeCun, Dmitry Petrov, Fabien Delattre, and Edmond Cunningham for helpful discussions and feedback on this work. Part of this work utilized resources from Unity, a collaborative, multi-institutional high-performance computing cluster managed by UMass Amherst Research Computing and Data.

References

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

[2] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *International Conference on Machine Learning*, pages 517–526. PMLR, 2017.

[3] 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.

[4] 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.

[5] 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.

[6] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[7] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[8] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on computer vision*, pages 6391–6400, 2019.

[9] 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.

[10] 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.

[11] Erik Learned-Miller and John W. Fisher III. ICA using spacings estimates of entropy. *The Journal of Machine Learning Research*, 4:1271–1295, 2003.

[12] Xin Liu, Zhongdao Wang, Ya-Li Li, and Shengjin Wang. Self-supervised learning via maximum entropy coding. *Advances in Neural Information Processing Systems*, 35:34091–34105, 2022.

[13] Grégoire Mialon, Randall Balestriero, and Yann LeCun. Variance covariance regularization enforces pairwise independence in self-supervised representations. *arXiv preprint arXiv:2209.14905*, 2022.

[14] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[15] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2(4):19–20, 1959.

[16] Serdar Ozsoy, Shadi Hamdan, Sercan Arik, Deniz Yuret, and Alper Erdogan. Self-supervised learning with an information maximization criterion. *Advances in Neural Information Processing Systems*, 35:35240–35253, 2022.

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[18] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. *arXiv preprint arXiv:1806.03198*, 2018.

[19] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2018.

[20] Oldrich Vasicek. A test for normality based on sample entropy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 38(1):54–59, 1976.

[21] 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.

Appendix A   Maximum-Entropy Augmentation Criterion

A.1   Pseudocode for our max-entropy augmentation criterion

Algorithm 1 PyTorch pseudocode for our max-entropy augmentation criterion

```
# sample training loop
for x in loader:
    # two random views of x
    x_a, x_b = augment(x)

    # compute embeddings
    z_a = f(x_a) # N x D
    z_b = f(x_b) # N x D

    # base SSL loss
    base_loss = ssl_loss(z_a, z_b)

    # our criterion
    ent_z_a, cov_z_a = max_ent_criterion(z_a)
    ent_z_b, cov_z_b = max_ent_criterion(z_b)
    ent_loss = ent_z_a + ent_z_b
    cov_loss = cov_z_a + cov_z_b

    # final loss
    loss = base_loss - beta * ent_loss + gamma * cov_loss

    # optimization step
    loss.backward()
    optimizer.step()

def max_ent_criterion(x, type):
    if type == 'hypercube':
        # apply the sigmoid transformation
        x_hyper = torch.sigmoid(x)
    elif type == 'hypersphere':
        # apply the CDF of 0-mean, 1 variance gaussian
        x_hyper = 0.5 * (1 + torch.erf(x / math.sqrt(2)))
    ent_loss = m_spacings_estimator(x_hyper)
    cov_loss = sample_cov_estimator(x_hyper)
    return (ent_loss, cov_loss)

def m_spacings_estimator(x):
    n = x.shape[0] # batch size
    m = round(math.sqrt(n)) # window size
    eps = 1e-7 # small constant to avoid underflow
    x, _ = torch.sort(x, dim=0) # order statistics
    x = x[m:] - x[:n - m] # m-spaced differences
    x = x * (n + 1) / m
    marginal_ents = torch.log(x + eps).sum(dim=0) / (n - m)
    return marginal_ents.mean()

def sample_cov_estimator(x):
    n, d = x.shape
    x = x - x.mean(dim=0) # mean subtraction
    cov_x = (x.T @ x) / (n - 1) # sample covariance matrix
    cov_loss = off_diagonal(cov_x).pow(2).sum().div(d)
    return cov_loss
```

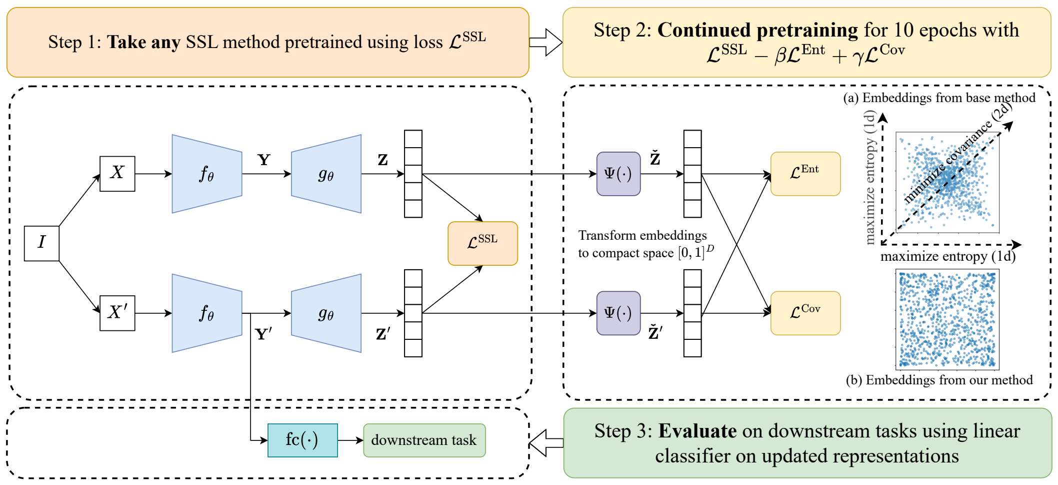

Step 1: **Take any** SSL method pretrained using loss $\mathcal{L}^{\text{SSL}}$
Step 2: **Continued pretraining** for 10 epochs with $\mathcal{L}^{\text{SSL}} - \beta\mathcal{L}^{\text{Ent}} + \gamma\mathcal{L}^{\text{Cov}}$
I
X
X'
f_θ
$\mathbf{Y}$
$\mathbf{Y}'$
g_θ
$\mathbf{Z}$
$\mathbf{Z}'$
$\mathcal{L}^{\text{SSL}}$
$\Psi(\cdot)$
$\check{\mathbf{Z}}$
$\check{\mathbf{Z}}'$
Transform embeddings to compact space $[0,1]^D$
$\mathcal{L}^{\text{Ent}}$
$\mathcal{L}^{\text{Cov}}$
(a) Embeddings from base method
maximize entropy (1d)
minimize covariance (2d)
maximize entropy (1d)
(b) Embeddings from our method
$\text{fc}(\cdot)$
downstream task
Step 3: **Evaluate** on downstream tasks using linear classifier on updated representations

Figure A.1: **Continued pre-training pipeline**.

A.2 Transformation to a compact space

Maximizing entropy on a non-compact space such as $\mathbb{R}^d$ is not meaningful, since the data can simply be spread out without bound. That is, our methods are meaningfully applied only on compact spaces. We discuss the maximization of entropy on two compact spaces: the unit hypercube and the surface of the unit hypersphere. We begin with the hypercube.

For SSL methods that produce embeddings in $\mathbb{R}^d$ and do not normalize their final embeddings (e.g., VICReg), we construct a transformation $\Psi : \mathbb{R} \to [0, 1]$, and apply it to every embedding component $\check{Z}_j = \Psi(Z_j)$, such that the transformed embedding $\check{Z} = [\check{Z}_1, \cdots, \check{Z}_d]$ lies in a unit hypercube of d dimensions, with an implicit joint distribution $p(\check{z}_1, ..., \check{z}_d)$ over the hypercube. We simply let Ψ be the sigmoid transformation, $\Psi(Z_j) = 1/(1 + \exp(-Z_j))$ and apply our loss function to this transformed embedding.

However, methods that do normalize their final embeddings to be on the hypersphere (e.g., SwAV, SimSiam, etc.) present a unique challenge. In particular, if we produce uniform marginal embeddings, *and then normalize*, the resulting distribution on the hypersphere will be far from uniform. In particular, mass will be much greater in directions corresponding to the corners of the hypercube, since the projections there will accumulate density along the longer diagonal directions of the hypercube. How then can we construct Ψ such that maximizing the entropy of the compact embeddings $\check{Z}$ also translates to a uniform (maximum entropy) distribution on the hypersphere when the original embeddings Z are normalized?

To answer this question, we use a simple result that is often used to draw samples uniformly from the surface of a hypersphere [15]: If we construct an embedding vector Z whose components Z_j are independent zero-mean, unit-variance Gaussians $Z_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$, then the normalized embedding vector $\tilde{Z} = Z/\left\|Z\right\|_2$ maps uniformly onto the surface of the unit hypersphere $\mathcal{S}^{d-1}$. In practice, we apply this result by letting Ψ be the cumulative density function (CDF) of the zero-mean, unit-variance Gaussian, $\Psi(Z_j) = 0.5(1 + \text{erf}(x/\sqrt{2}))$, and then applying our entropy maximization criterion to the transformed embeddings to produce a uniform distribution. This is possible due to the *probability integral transform*, in which a continuous random variable is mapped through its own CDF to become uniformly distributed. This implies that the distribution over transformed variables $p(\check{z}_j) \overset{\text{d}}{=} \mathcal{U}[0, 1]$ if and only if the distribution over original embeddings $p(z_j) \overset{\text{d}}{=} \mathcal{N}(0, 1)$. Our criterion thus ensures that the components of the embedding distribution prior to normalization are normal with zero mean and unit variance. In addition, our term to minimize correlation helps to minimize dependencies among the unit variance marginals.

Using these two methods of transforming to compact spaces, our criterion can be applied to SSL methods irrespective of their normalization strategy.

A.3 Additional details of components in the maximum-entropy augmentation criteria

A.3.1 Marginal entropy estimator

The m-spacings estimator of marginal (one-dimensional) entropy is defined as

$$\widehat{\mathcal{H}}_j(\check{Z}_\theta^1, ..., \check{Z}_\theta^n) = \frac{1}{(n-m)} \sum\nolimits_{i=1}^{n-m} \log\left(\frac{n+1}{m}\left(\check{Z}_{\theta j}^{(i+m)} - \check{Z}_{\theta j}^{(i)}\right)\right) \tag{A.1}$$

for the jth dimension of $\check{Z}_\theta$, where $\check{Z}_\theta \in [0,1]^D$ is the compact version of the embedding Z_θ. Parenthetical superscripts indicate the position in the ordering $\check{Z}_{\theta j}^{(1)} \leq \check{Z}_{\theta j}^{(2)} \leq ... \leq \check{Z}_{\theta j}^{(n)}$ for $j \in \{1, ..., d\}$., and $\check{Z}_{\theta j}^{(i+m)} - \check{Z}_{\theta j}^{(i)}$ is known as a spacing of order m (typically $m = \sqrt{n}$). We estimate the entropy $\mathcal{H}\left(\check{Z}_j\right)$ for each $j \in \{1, ..., d\}$ using this estimator and average them in the final loss.

$$\mathcal{L}^{\text{Entropy}}(\theta) = \frac{1}{d} \sum\nolimits_{j=1}^{d} \widehat{\mathcal{H}}_j(\check{Z}_\theta^1, ..., \check{Z}_\theta^n). \tag{A.2}$$

A.3.2 Sample Covariance

We estimate the squared off-diagonal covariance matrix entries using the sample covariance.

Let $\check{\check{Z}}_\theta = \check{Z}_\theta - \frac{1}{n}\sum_{i=1}^n \check{Z}_\theta^i$ for $\check{\check{Z}}_\theta, \check{Z}_\theta \in \mathbb{R}^{n \times d}$. Now, consider the sample covariance estimator

$$\widehat{\text{Cov}}_{jk}(\check{Z}_\theta^1, ..., \check{Z}_\theta^n) = \frac{1}{n-1} \sum\nolimits_{i=1}^{n} \check{\check{Z}}_{\theta j}^i \check{\check{Z}}_{\theta k}^i \tag{A.3}$$

for the jth and kth dimension of $\check{Z}_\theta$.

Letting $K_\theta = \check{\check{Z}}_\theta^\top \check{\check{Z}}_\theta$, we define

$$\mathcal{L}^{\text{Covariance}}(\theta) = \frac{1}{nd}\|(K_\theta - \text{diag}(K_\theta))\|_F^2 = \frac{1}{d}\sum\nolimits_{j=1,k=1}^{d} \mathbb{I}[k \neq j]\widehat{\text{Cov}}_{jk}(\check{Z}_\theta^1, ..., \check{Z}_\theta^n)^2, \tag{A.4}$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm.

Appendix B Full Implementation Details

Our method uses a three-stage approach (see Figure A.1 for an overview):

1. **Selecting** a base SSL method with publicly available checkpoints,
2. **Continued pre-training** on the base dataset using the base SSL method augmented with our criterion, and
3. **Evaluating** the representations learned by the backbone network using classifiers trained on downstream datasets.

While our approach can be used with *any* joint-embedding SSL method, we focus on three popular methods—VICReg [1], SwAV [3], and SimSiam [5]—to demonstrate the versatility of our criterion. We download the publicly available code and checkpoints for these methods and do continued pre-training on them. The checkpoints available are: 1000-epochs for VICReg, 400- and 800-epochs for SwAV, and 100-epochs for SimSiam.

In the continued pre-training stage, we train on the same dataset that was used for pre-training the base SSL method (ImageNet [17]), but with a fixed reduced learning rate (0.01 × base_lr) and batch size (512) than during original pre-training. We train for *exactly* ten additional epochs using the prescribed criterion, and report results using this updated model. All other training hyperparameters associated with the base SSL method including data augmentation strategies, optimizers, and loss coefficients, are kept identical. This allows us to treat the base SSL method as a black box, and leaves us with only a few hyperparameters to tune for our method, namely the coefficients β and γ for our proposed loss in Equation (2).

In the linear evaluation stage, the final representations from the ResNet-50 backbone from the updated SSL model (after 10 epochs of continued pre-training) are used to train classifiers for downstream tasks to evaluate any improvements in accuracy over the base method. We will now specify the exact hyperparameters used for each method and dataset used.

B.1 Continued Pre-training

Continued pre-training involves starting from a pre-trained checkpoint of a base SSL method and training for *exactly* 10 epochs with an additional criteria. This stage uses 2$\times$NVIDIA Quadro RTX8000 GPUs with 48GB VRAM for continued pre-training of each model. Training times for 10 epochs of continued pre-training are 10 hrs for VICReg, 13 hrs for SwAV, and 14 hrs for SimSiam. Due to the extremely limited number of these GPUs, we could not train models from scratch, do continued pre-training with bigger batch sizes or for longer, or run extensive grid searches for hyperparameters. We will now detail the different criteria used and their associated hyperparameters for various base SSL methods.

B.1.1 VICReg

We start from the 1000-epoch checkpoint for VICReg [1] with ResNet-50 backbone, and 3-layer MLP (8192-8192-8192) as projector architecture. For continued pretraining, our criterion is applied to the final projector embeddings Z mapped through a sigmoid transformation. We use the default coefficients for the VICReg loss function $\lambda = \mu = 25,\ \nu = 1$, and coefficients used for our loss in Eqn. (2) are $\beta = 1000$, $\gamma = 100$. We do continued pre-training for 10 epochs, with a learning rate of 0.003 (i.e., $0.01\times$ the base learning rate used to train VICReg), batch size of 512, and all other hyperparameters left unchanged from the original method. The same settings are used for continued training ablation using the base loss only.

B.1.2 SwAV

We run experiments using both 400-epoch and 800-epoch checkpoints released for SwAV [3] with multicrop, using resnet50-backbone and 2-layer MLP (2048-128) as projector architecture. For continued pretraining, our criterion is applied to the projector embeddings Z before the cluster assignment layer and before normalization after mapping through the CDF function. We use default hyperparameters for SwAV loss namely $\tau = 0.1$, and apply our loss only to the embeddings of full-resolution crops (two views) and not the low resolution multiple crops for computational efficiency, with coefficients $\beta = 1$ and $\gamma = 25$. We do continued pre-training for 10 epochs, with a learning rate of 0.001 (i.e., $0.01\times$ the base learning rate used to train SwAV), batch size of 512, and all other hyperparameters left unchanged from the original method. The same settings are used for continued training ablation without our loss. We will now describe the ablation studies that train SwAV models further using an alternative criterion and associated hyperparameters.

Variance-Covariance Regularization (VCReg). We use the variance and covariance regularization losses from the VICReg [1] objective, to minimize the following loss

$$\mathcal{L}(\theta) = \mathcal{L}^{\mathrm{SSL}}(\theta) + \mu\mathcal{L}^{\mathrm{Variance}}(\theta) + \nu\mathcal{L}^{\mathrm{Covariance}}(\theta) \tag{B.5}$$

where μ and ν are the coefficients for the variance and covariance terms respectively, where

$$\mathcal{L}^{\mathrm{Variance}}(\theta) = \frac{1}{d}\,\mathrm{Tr}\Big(\max\Big(0, \eta - \sqrt{\mathrm{diag}(K_\theta) + \epsilon}\Big)\Big), \tag{B.6}$$

$$\mathcal{L}^{\mathrm{Covariance}}(\theta) = \frac{1}{nd}\left\|K_\theta - \mathrm{diag}(K_\theta)\right\|_F^2 \tag{B.7}$$

where $||\cdot||_F$ is the Frobenius norm, and we defined $K_\theta = \bar{Z}_\theta^\top \bar{Z}_\theta$, where $\bar{Z}_\theta = Z_\theta - \frac{1}{n}\sum_{i=1}^n Z_\theta^i$, and η is the target variance. We set $\mu = 0.1$ and $\nu = 0.001$ and this loss is applied only on the embeddings of the two full resolution crops to be consistent with our setting. These hyperparameters were determined experimentally by searching over $\mu = [0.01, 0.1, 1, 25]$ and $\nu = [0.001, 0.005, 0.01, 0.1, 1, 25]$ on the validation set using linear classifier trained on 1% ImageNet labels.

Alignment and Uniformity on the Hypersphere (AUH). We use the uniformity loss proposed in the AUH [21] objective, to minimize the following loss

$$\mathcal{L}(\theta) = \mathcal{L}^{\mathrm{SSL}}(\theta) + \lambda\mathcal{L}^{\mathrm{Uniform}}(\theta) \tag{B.8}$$

where,

$$\begin{aligned}\mathcal{L}^{\text{Uniform}}(\theta) \triangleq \log\ & \mathbb{E}_{[Z^p, Z^q \overset{\text{i.i.d.}}{\sim} p(z)]}\Big[G_t(Z_\theta^p, Z_\theta^q)\Big] \\ & + \log\ \mathbb{E}_{[Z'^p, Z'^q \overset{\text{i.i.d.}}{\sim} p(z')]}\Big[G_t(Z_\theta^{'p}, Z_\theta^{'q})\Big], \quad t > 0\end{aligned} \tag{B.9}$$

where $p, q \in \{1, \cdots, n\}$, and G_t is defined as the average pairwise gaussian potential between two embedding vectors:

$$G_t(Z_\theta^p, Z_\theta^q) = \exp\left(-t\|Z^p - Z^q\|_2^2\right) \tag{B.10}$$

In practice this is applied to all embedding pairs from each of the full resolution crops (each of the two views) to be consistent with our setting. The hyperparameters used are $\lambda = 0.5$ and $t = 2$, and continued pre-training is done for 10 epochs with the same training hyperparameters. The hyperparameters for this method are guided by the original paper and experimentally verified on 1%-ImageNet split as before.

B.1.3 SimSiam

We start from the 100-epoch checkpoints released for SimSiam [5], using resnet50-backbone and 3-layer MLP (2048-2048-2048) as projector, and 2-layer MLP (2048-512) as predictor architecture. For continued pretraining, our criterion is applied to the projector embeddings Z on both branches before the predictor, mapped through a CDF function. The projector embeddings thus serve as uniformly distributed points on the hypersphere that the predictor has to map to. We use default coefficients for SimSiam loss, and apply our loss with coefficients $\beta = 0.001$ and $\gamma = 0.01$. We do continued pre-training for 10 epochs, with a learning rate of 0.001 (i.e., $0.01\times$ the base learning rate used to train SimSiam), batch size of 512, and all other hyperparameters left unchanged from the original method. The same settings are used for continued training ablation without our loss.

B.2 Evaluation

In this stage, the final representations from ResNet-50 backbone are used to train classifiers on different datasets in order to evaluate the quality of the representations. Training hardware includes $4\times$NVIDIA RTX 2080TI GPUs with 11GB VRAM for each training.

B.2.1 Linear evaluation on ImageNet.

Following standard practice, we train linear classifiers using frozen ResNet-50 representations on 1% (12,811 images), 10% (128,117 images), and 100% (1,281,176 images) of ImageNet labels (using predefined splits from [4]) for 100 epochs, and report the top-1 accuracy on the validation set containing 50,000 images and 1,000 classes.

For VICReg, we use the SGD optimizer with learning rate 0.02 and cosine decay, batch size of 256, and a weight decay of 10^{-4} for 1% and 10% splits, and 10^{-6} for 100% split respectively.

For SwAV, we use the SGD optimizer with learning rate 0.3 and cosine decay, batch size of 256, and a weight decay of 10^{-6} for all splits.

For Simsiam, we use the LARS optimizer with weight decay 0 and cosine decay for learning rate as follows. For the 1% split, we use learning rate 2.0 and batch size 256. For the 10% split, we use learning rate 0.2 and batch size 2048. For the 100% split, we use learning rate 0.1 and batch size 2048.

B.2.2 Semi-supervised learning on ImageNet.

We perform semi-supervised evaluation by finetuning the whole backbone with a classifier on a subset of available labels.

For VICReg, we use the SGD optimizer with batch size 256, cosine learning rate schedule, and no weight decay, and train for 20 epochs using learning rate 0.03 for the backbone and 0.08 for the linear classifier in the 1% labels setting, and learning rate 0.01 for the encoder and 0.1 for the linear classifier in the 10% labels setting. Unfortunately, we're unable to reproduce the numbers reported in the paper (54.8% and 69.5% respectively) exactly using these prescribed settings, and report our closest reproduced values in Table 1.

For SwAV, use the SGD optimizer with batch size 256, step decay of 0.2 at epochs 12 and 16 for a total of 20 epochs, and no weight decay using learning rate 0.02 for the backbone and 5 for the linear classifier in the 1% labels setting, and learning rate 0.01 for the encoder and 0.2 for the linear classifier in the 10% labels setting.

For SimSiam, the semi-supervised experiments were not conducted in the original paper, and therefore we skip this in our experiments.

B.2.3 Transfer learning performance on other datasets.

Following [14, 8], we show how representations updated using our method on ImageNet dataset generalize to downstream linear classification on other datasets such as iNaturalist 2018 [19] and Pascal VOC 2007 [7].

For iNat18 (437,513 images and 8,142 classes), we use `res5` features from the ResNet-50 backbone (before average pooling layer) subsampled to 8192-d using an average pooling layer of size (6, 6) and stride 1, followed by a batch normalization layer. A linear classifier is then trained on top of these representations using the SGD optimizer with batch size 256, weight decay 10^{-4}, momentum 0.9, and learning rate 0.01 reduced by a factor of 10 at epochs 24, 48, and 72, for a total of 84 epochs. These hyperparameters are used consistently across all methods, and we find that for SwAV, we obtain better performance (49.72) than reported in the original paper (48.6).

For VOC07 (5,011 images and 20 classes), we train linear SVMs on top of final average pooled representations (2048-d) from ResNet-50 backbone using the VISSL library [8] and report the mean Average Precision (mAP) of multi-label object classification on the validation set. In this setting, we were unable to reproduce numbers reported in the papers exactly due to missing hyperparameter details and default values in the library not working well. The numbers reported in the paper are using the following C values: $[0.000001, 0.000003, 0.00001, 0.00003, 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 2, 5, 10, 15, 20, 50, 100, 200, 500, 1000]$. We get close to the performance reported in original papers, and mark our results where there are significant differences, such as SwAV (88.56 vs. the reported 88.9).

Appendix C Additional Experiments

C.1 Ablation: Number of Continued Pre-training Epochs

In this experiment, we present results from intermediate epochs of continued pre-training for a total of 30 epochs, i.e., upto $3\times$ longer continued pre-training than the main reported results in Table 1. In Figure C.2, we report the Top-1-Accuracy of linear classifier trained on frozen ResNet-50 representations from intermediate epochs using 1% ImageNet labels. We show the performance of SwAV (800 epochs) and VICReg (1000 epochs) base models at epoch 0, and how they evolve during continued pre-training under our maximum-entropy criterion. It is evident that continued pre-training with our maximum-entropy criterion (MaxEnt) is better than simply training longer using the base criterion (continued). We also notice a rapid improvement in performance until epoch 10 after which the performance either improves marginally (e.g., for VICReg) or starts degrading (e.g., for SwAV). We therefore claim that 10 epochs of continued pre-training provides a good trade-off between performance and training time, and report results from this epoch in the main paper.

Appendix D Additional Analysis

D.1 Nearest Neighbor Distances

One of the advantages of a maximum-entropy embedding is that data is well separated in the embedding space thereby preserving discriminability for downstream tasks. Following [18], in Figure D.3, we show the histogram of distances to the nearest and 100th nearest neighbors for all points in the ImageNet validation set, computed using VICReg embeddings before (left) and after (right) continued pre-training using our maximum-entropy criterion. We note that for VICReg, the

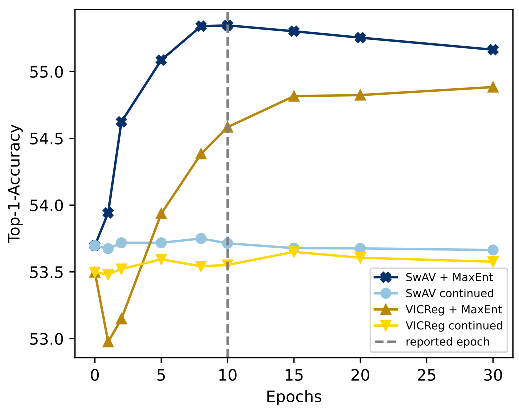


Figure C.2: Top-1-Accuracy of linear classifier trained on frozen representations using 1% ImageNet labels for intermediate epochs of continued pre-training. Continued pre-training with our criteria (MaxEnt) outperforms other baselines, and performance beyond the reported 10 epochs either improves marginally or degrades depending on the method.

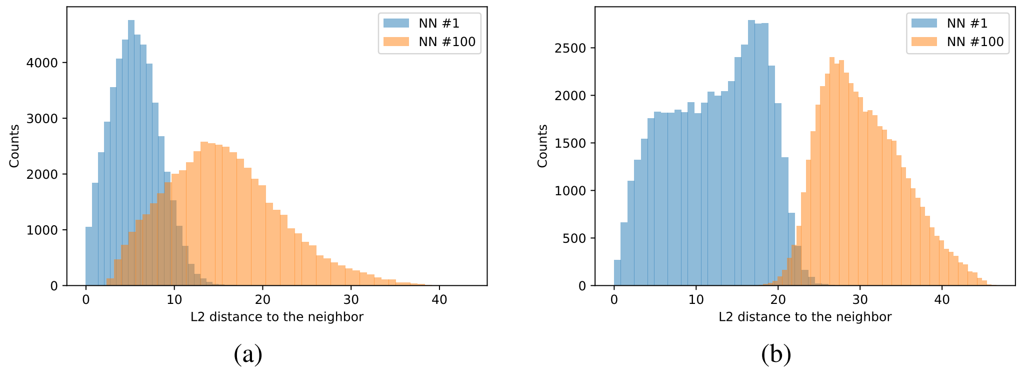
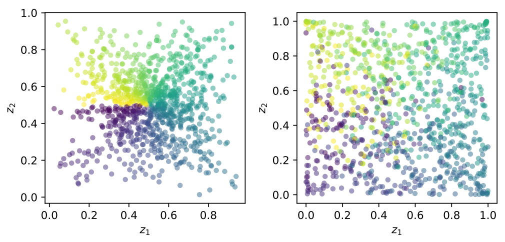


Figure D.3: Histogram of distances from a query point to its nearest neighbor (blue) and 100th nearest neighbor (orange) for (a) VICReg and (b) VICReg + MaxEnt. For VICReg, the histograms have signifcant overlap indicating less separability between points in embedding space. We show that continued pre-training with our criterion reduces the overlap between the two histograms ensuring greater separability between points for downstream tasks.

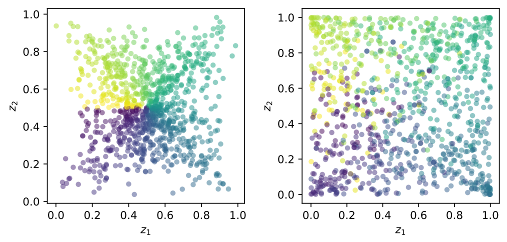
two histograms have significant overlap signifying that the 100th nearest neighbor for a point is often closer than the 1st nearest neighbor for another point, suggesting low separability that affects performance on downstream task. Continued pre-training using our criterion significantly reduces this overlap ensuring greater discriminability for downstream tasks.

D.2 More Sample Distributions over Two Dimensional Marginals

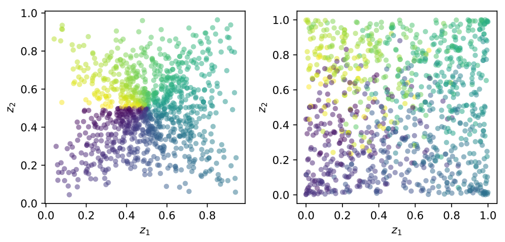
Figure D.4 shows more examples of sample distributions over a randomly selected pair of embedding dimensions (marginals) from VICReg *before* and *after* continued pre-training with our maximum-entropy criterion. Note how our method virtually always produces uniformly distributed marginals over any random pair even though this is not explicitly enforced by our loss. A fixed set of colors was assigned to the data points when plotting the “before" embeddings, and one can follow how the points were distributed by the application of our criterion by noting the relative distribution of colors in the “after" embeddings.



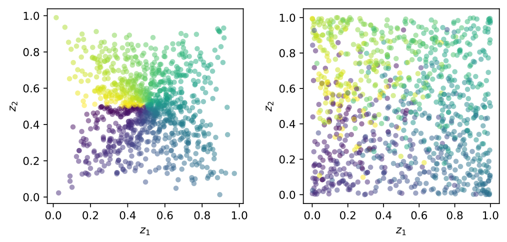
(a)



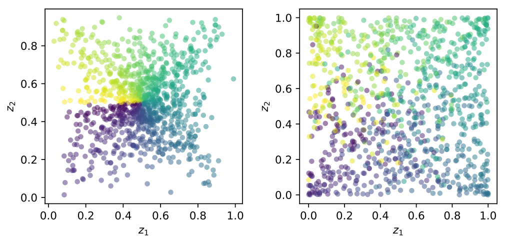
(b)



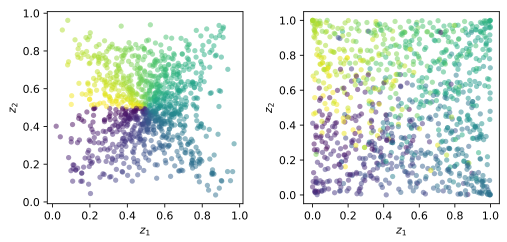
(c)



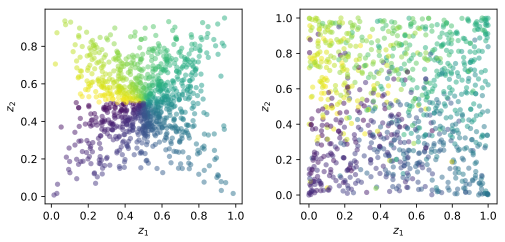
(d)



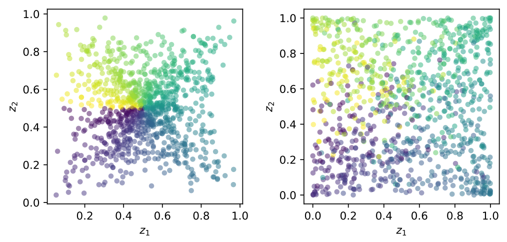
(e)



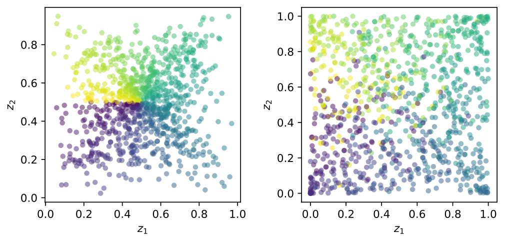
(f)



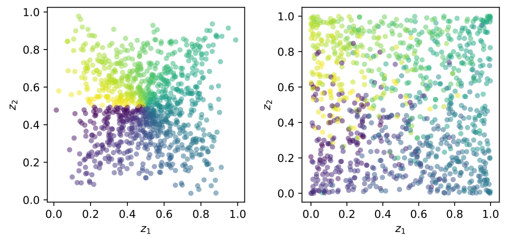
(g)



(h)



(i)



(j)

Figure D.4: (a)-(j) shows more sample distributions from VICReg (left) and VICReg + MaxEnt (right) over a random pair of compact embedding dimensions, for a fixed set of data points.