A PROOFS

A.1 COUNTEREXAMPLE AGAINST UNRESTRICTED q.

Pick some $x^* \sim p(x)$ and take f to be a continuous function whose range spans $[0, 1]$. For any $\epsilon > 0$, pick $q(x|x^*)$ to be a distribution such that for every $x \sim q(x|x^*)$ with non-zero probability, we have $f(x, x^*) < \epsilon$. Then, by varying ϵ closer to 0, we can bring our bound on mutual information to infinity, regardless of the true value, thus ceasing to be a bound. As such, we cannot use an unrestricted family of conditional distributions and preserve a bound.

A.2 PROOF OF THEOREM 3.2

Proof. We separately show the statements regarding bias and variance.

Fix some $x \sim p(x)$. Let $p(y_1, \dots, y_k) = \prod_{j=1}^{k} p(y_j)$. First, as $Z(y_1, \dots, y_k)$ with $y_1, \dots, y_k \sim p(y_1, \dots, y_k)$ iid (or NCE) is unbiased, and $\mathbf{E}_{q(y_1,\dots,y_k|x)}[Z]$ (or CNCE) lower bounds $\mathbf{E}_{p(y_1,\dots,y_k)}[Z]$ (using Thm. 3.1), the first statement follows immediately for any choice of k.

Second, by the law of total variance,

$$\mathbf{E}_p[\text{Var}_{p(y_1,\dots,y_k)}[Z|\mathbf{1}_{S_B}]] + \text{Var}_{p(y_1,\dots,y_k)}(\mathbf{E}_{p(y_1,\dots,y_k)}[Z|\mathbf{1}_{S_B}]) = \text{Var}_{p(y_1,\dots,y_k)}[Z]$$

Since both summands are non-negative and the variance on the right is the desired upper-bound, it suffices to show that

$$p(S_B) \cdot \text{Var}_{q(y_1,\dots,y_k|x)}[Z] \leq \mathbf{E}_{p(y_1,\dots,y_k)}[\text{Var}_{p(y_1,\dots,y_k)}[Z|\mathbf{1}_{S_B}]].$$

This follows immediately from the observation that by definition of $q(y_1, \dots, y_k|x)$ as the conditional distribution $p(y_1, \dots, y_k|S_B)$, the expectation on the right is precisely

$$p(S_B) \cdot \text{Var}_{q(y_1,\dots,y_k|x)}[Z] + (1 - p(S_B)) \cdot \text{Var}_{\tilde{q}(y_1,\dots,y_k|x)}[Z],$$

where $\tilde{q}(y_1, \dots, y_k|x)$ is the conditional distribution $p(y_1, \dots, y_k|S_B^c)$, and S_B^c represents the complement set of S_B. □

B A TOY EXAMPLE

Interestingly, Thm. 3.1 shows CNCE to lower bound NCE. To confirm this experimentally, we repurpose the toy setting from Tschannen et al. (2019). Pick two random variables Z and ϵ distributed such that $z_i \sim \mathcal{N}(0, \Sigma_Z)$ and $\epsilon_i \sim \mathcal{N}(0, \Sigma_\epsilon)$ where $\Sigma_Z = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$ and $\Sigma_\epsilon = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$ Then, let $(X, Y) = Z + \epsilon$. That is, let X be the first dimension of the sum and Y the second. The mutual information between X and Y can be analytically computed as $-\frac{1}{2}\log(1 - \frac{\Sigma[1,2]\Sigma[2,1]}{\Sigma[1,1]\Sigma[2,2]})$ since (X, Y) is jointly Gaussian with covariance $\Sigma = \Sigma_Z + \Sigma_\epsilon$. For this toy experiment, let w_ℓ

	True	NCE	CNCE					
ω			10	25	50	75	90	95
Mean	0.02041	0.01345	0.01241	0.00220	7.29e-5	1.67e-5	5.87e-6	1.97e-6
Stdev	–	0.001	3e-4	1e-4	9e-6	2e-6	1e-6	4e-6

Table 6: Looseness of CNCE as w_ℓ increases.

be a percentage from 0 to 100. Now, we define S_B as all examples whose dot product with the embedding of the current transformed instance is in the top w_ℓ percentage of all examples in the dataset. We can tractably compute this using a memory bank. As w_ℓ increases from 10 to 95, $q(x|t(x_i))$ has smaller support meaning that negative samples are more difficult to separate from the current instance x_i. Table 6 compares the estimated mutual information between X and Y from each estimator to the ground truth over 5 runs. The encoders are 5-layer MLPs with 10 hidden dimensions and ReLU nonlinearities. To build the dataset, we sample 2000 points and optimize the NCE objective with Adam with a learning rate of 0.03, batch size 128, and no weight decay for 100 epochs. Given a percentage for CNCE, we compute distances between all elements in the memory bank and the representation the current image — we only sample 100 negatives from the top p percent. We conduct the experiment with 5 different random seeds.

C BIAS AND VARIANCE EXPERIMENT DETAILS

For IR, we explore $k = 16, 32, 64, 128, 256, 512, 1024, 4096$. For MoCo, we only evaluate $k = 256, 512, 1024, 4096$ as the queue cannot be smaller than the batch size. All hyperparameter choices are as detailed in the main experiments. To find the nearest neighbor of the training example, we store all embeddings in a memory bank (separate from the one possibly used in training).

D DETECTRON2 EXPERIMENTS

We make heavy usage of the Detectron2 code found at `https://github.com/facebookresearch/detectron2`. In particular, the script `https://github.com/facebookresearch/detectron2/blob/master/tools/convert-torchvision-to-d2.py` allows us to convert a trained ResNet18 model from `torchvision` to the format needed for Detectron2. The repository has default configuration files for all experiments. We change the following fields to support using a frozen ResNet18:

```
INPUT:
    FORMAT: RGB
MODEL:
    BACKBONE:
        FREEZE_AT: 5
    PIXEL_MEAN:
        - 123.675
        - 103.53
        - 116.28
    PIXEL_STD:
    - 58.395
    - 57.12
    - 57.375
    RESNETS:
        DEPTH: 18
        RES2_OUT_CHANNELS: 64
        STRIDE_IN_1X1: false
    WEIGHTS: <PATH_TO_CONVERTED_TORCHVISION_WEIGHTS>
```

We acknowledge that ResNet50 and larger are the commonly used backbones, so our results will not be state-of-the-art. However, the ordering in performance between algorithms is still meaningful and our primary interest. Future work can explore larger architectures.

E SPEECH EXPERIMENTS

We adapt the experimental setup from Tamkin et al. (2020): a contrastive representation is trained with the LibriSpeech 100 hour corpus (Panayotov et al., 2015) in which waveforms are truncated to 150,526 timesteps and processed to log Mel spectrograms with hop length 2,360 to output a matrix of size 64 by 64. Spectrograms are z-scored using training statistics. We use masking of time and frequency features as augmentations on training examples. Spectrograms are encoded with a ResNet18 to a 128-dim. embedding. In transfer, we also preprocess waveforms to a 64 by 64 matrix. Spectral augmentations are used in training the Logistic regression model but no augmentations are used when evaluating the test split. The input to the logistic regression model is the ResNet18 pre-pool features prior to the final linear layer.

F STATE-OF-THE-ART EXPERIMENTS

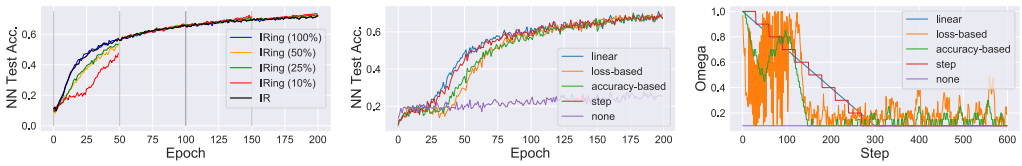
In Sec. 7, we reported results comparing MoCo-v2 and MoCoRing-v2 on CIFAR10, CIFAR100, and STL10 using state-of-the-art hyperparameters. We point out several differences from the training details presented in Sec. 5. First, for CIFAR10 and CIFAR100, we work directly 32x32 pixel images (instead of resizing to 224x224 pixels); for STL10, we work with the 96x96 pixel raw images.

Second, in ResNet encoder, we remove the max pooling layer and replace the first 7x7 convolutional layer with a 3x3 convolutional layer. Third, we used post-pooling features (instead of pre-pooling features) for linear evaluation. Fourth, we used a much larger learning rate in linear evaluation (0.1 vs 0.01). Fourth, for STL10, including the unlabeled data made a large impact in performance. All models were trained for 200 epochs (depite prior work optimizing for 800 epochs). The MoCo-v2 baseline implementation was adapted from the PyTorch Lightning Bolts repository: `https://github.com/PyTorchLightning/lightning-bolts`.

G Additional Experiments

We discuss a few observations surrounding Ring Discrimination and in particular, annealing.

Hard negative mining is not always productive. We can attribute this to the poor quality of embeddings early in training: using hard negatives can (1) simply be too difficult for the encoder to discriminate, or (2) focus the embedding on smaller, perhaps spurious, differences between the instance and the hard negatives, rather than prioritizing higher level semantic information (e.g. object identity). As a demonstration of this phenomena, Fig. 3a shows several training runs of IRing on CIFAR10 with varying thresholds ω_ℓ initialized at every 50 epochs of an IR model for a total of 200 epochs. In the legend, a smaller percentage indicates drawing negatives more similar to the embedding of the current instance as measured by dot products. (IRing (100%) and IR are identical.) The y-axis plots the accuracy of classification where for each test example, we predict the label of its L_2 nearest neighbor in the training split (Wu et al., 2018; Zhuang et al., 2019). Fig. 3 shows that (1) using smaller thresholds at the beginning of training results in lower test accuracies; (2) in the middle of training (epoch 50), the performance is equivalent for all models; and (3) in later training stages (epoch 100), using more difficult negatives is better. Notice the ordering of the lines in Fig. 3: $10\% < 25\% < 50\% < 100\%$ early in training while the inequalities are flipped at epoch 100.



(a) Effect of Hard Negatives (b) Annealing Policies (c) Annealing Thresholds w_u

Figure 3: (a) Embedding quality as a function of how similar negative samples are to the current instance in Ring Discrimination (the percentage represents the percentile u). (b,c) An exploration of difficult four policies for annealing Ring percentiles u.

Exploring annealing policies. Given that our experiments show annealing is important, there is a question of "how to anneal". In our experiments, we opted for a simple linear policy: slowly reducing u from 100% to 10% in 100 epochs and maintaining it constant at 10% for the remaining epochs. Here, we briefly compare this to three other policies: a step function; an adaptive policy that lowers the percentile every epoch if the performance on a validation set increases, otherwise decreasing the percentile; and a similar adaptive policy that updates every step based on negative training loss. Fig. 3b compares the nearest neighbor test accuracies over 200 epochs of training IRing on CIFAR10 whereas Fig. 3b plots the percentile u. We find that all the policies converge to roughly the same test accuracy, although linear and step policies appear to converge more quickly. From Fig. 3c, we observe that the adaptive methods naturally push the percentile down to 10% (the lowest allowed percentile) around step 150, confirming our intuition that a smaller percentile later in training is desirable. Future work could explore more sophisticated policies.

Additional experiments measuring cost. Table 7 shows additional experiments comparing the computational cost of Ring and standard negative sampling on CIFAR100 and STL10. The costs remain negligible with Ring having a 1.0 to 1.4x cost on compute time.

Model	Cost (sec.)
IR	136.0 ± 4
IRing	141.1 ± 5 (**1.1x**)
MoCo	318.4 ± 16
MoCoRing	383.4 ± 12 (**1.2x**)

(a) CIFAR10

Model	Cost (sec.)
IR	105.2 ± 3
IRing	154.0 ± 3 (**1.4x**)
MoCo	346.6 ± 3
MoCoRing	371.8 ± 3 (**1.1x**)

(b) CIFAR100

Model	Cost (sec.)
IR	13.5 ± 1
IRing	14.2 ± 1 (**1.0x**)
MoCo	25.9 ± 3
MoCoRing	34.1 ± 4 (**1.3x**)

(c) STL10

Model	Cost (min.)	
IR	43.9 ± 1	
IRing	51.0 ± 1	(**1.2x**)
MoCo	61.1 ± 1	
MoCoRing	64.9 ± 1	(**1.1x**)

(d) ImageNet

Table 7: Cost of one training epoch in seconds, averaged over 200 epochs.

H Related Work: Ring and Local Aggregation

Of the many algorithms listed above, we focus on Local Aggregation (Zhuang et al., 2019), or LA, which we conjecture to already be (implicitly) mining hard negatives. While IR seeks to uniformly distribute embeddings, uniformity may not be optimal in all cases. For instance, images of the same class should intuitively be closer together than other images. The LA objective captures this intuition using a "close neighbor set" C_i and "background neighbor set" B_i conditioned on the current transformed instance $t(x_i)$. The background neighbor set contains the indices of elements in the dataset whose embeddings are closest to $g_\theta(t(x_i))$ in L_2 distance. The close neighbor set contains elements are same cluster as $t(x_i)$ using Kmeans assignments. Although not originally formulated in this manner, we can view the background neighbor set as being sampled from a CNCE distribution $q(B_i|t(x_i))$ with the lower percentile ℓ set to 0 i.e. the ring is fully enclosed. Now, writing LA in the notation of Eq. 2, its objective is

$$\mathcal{L}_{\text{LA}}(x_i; M) = \mathbf{E}_{t\sim p(t)}\mathbf{E}_{B_i\sim q(B_i|t(x_i))}\left[\log \frac{\frac{1}{|C_i|}\sum_{j\in C_i} e^{g_\theta(t(x_i))^T M[j]/\tau}}{\frac{1}{|B_i|}\sum_{j'\in B_i} e^{g_\theta(t(x_i))^T M[j']/\tau}}\right]. \tag{3}$$

Although Ring Discrimination and LA both mine hard negatives, LA additionally uses instances in the same KMeans cluster as positive views of x_i. Borrowing ideas from LA, we can explore several extensions of Ring Discrimination. First, by "Cave" Discrimination (including IRCave ad CMC-

Model	Top1
LA	83.9
IRCave	84.0
CMCCave	87.2
IRing (+C_i)	84.3
CMCRing (+C_i)	87.8

Table 8: Variants of Ring

Cave), we denote drawing negative samples from a CNCE distribution q with a support restricted to the examples in the same KMeans clustering as the current instance (Note that such a definition falls under Theorem 3.1 as a particular choice for the restricted set S_B). Second, Ring (+C_i) instead, includes members of the KMeans clustering as positive views of x_i, like in LA — here, negative samples are drawn as in regular Ring. Note that LA and IRing (+C_i) differ only by the lower percentile ℓ, which is zero in the former and nonzero in the latter. Table 8 shows promising results on CIFAR10 as these variations produce strong representations. This suggests that choosing good views and good negatives together can build even better contrastive algorithms.