BDCR: DISTRIBUTIONAL CONTRASTIVE REGULARIZATION FOR SEPARATED REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Distinguishing semantically similar classes is challenging when class-conditional features exhibit substantial intra-class variation and inter-class overlap. We propose **BDCR (Bhattacharyya-based Distributional Contrastive Regularization)**, a distribution-level regularizer that represents each class by a Gaussian distribution characterized by its mean and variance. Rather than contrasting individual samples or point prototypes, BDCR contrasts class-conditional distributions using the Bhattacharyya distance, jointly capturing differences in feature location and variation. To focus regularization on ambiguous class relationships, BDCR maintains momentum-smoothed class statistics and dynamically selects hard negative classes, while treating the discrete selection process as non-differentiable. The resulting regularizer can be integrated into standard classification objectives without modifying the network architecture. Experiments across diverse architectures and datasets show that BDCR consistently improves classification performance, with distribution-level hard-negative regularization providing complementary gains over conventional sample- and prototype-based objectives.

1 INTRODUCTION

Deep neural networks have achieved strong performance in visual classification, yet distinguishing semantically similar classes remains challenging due to inter-class overlap and substantial intra-class variation. Standard classification objectives do not explicitly model these properties at the class level. Prior discriminative representation learning methods improve feature separation through metric-learning objectives (Chopra et al., 2005; Schroff et al., 2015) and center-based regularization (Wen et al., 2016), while prototype-, proxy-, and classifier-based approaches characterize classes through point representations or class-specific directions (Snell et al., 2017; Movshovitz-Attias et al., 2017; Kim et al., 2020; Deng et al., 2019). Although effective, such representations primarily characterize class location and do not explicitly model the variation of features within each class.

We argue that class ambiguity is more naturally characterized at the distribution level, building on the broader view that representations can be modeled as probability distributions rather than deterministic points (Vilnis & McCallum, 2014). In particular, the separation of two classes cannot be characterized solely by the distance between their centers. Two classes may have well-separated centers yet exhibit substantial feature overlap due to large intra-class variation, while classes with similar centers may nevertheless exhibit different patterns of feature variation that are not captured by point-based representations. This motivates treating the class-conditional feature distribution as the unit of supervised regularization, so that both feature location and variation contribute to the characterization of each class.

Based on this perspective, we introduce **BDCR** (Bhattacharyya-based Distributional Contrastive Regularization), a distributional contrastive regularizer that approximates the feature distribution of each semantic class with a Gaussian distribution characterized by a mean and diagonal covariance. Rather than contrasting individual samples as in supervised contrastive learning (Chen et al., 2020) or relying on point-based class representations, BDCR directly compares class-conditional distributions. Distributions estimated from different subsets of the same class are encouraged to remain consistent, while distributions from different classes are encouraged to separate. For Gaussian distributions, the Bhattacharyya distance (Bhattacharyya, 1943) provides a symmetric closed-form

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

measure that jointly captures differences in feature means and variances. Thus, the role of the Bhattacharyya distance in BDCR is to provide a tractable distributional comparison, while the overall framework uses these class-conditional comparisons to construct a supervised distribution-level contrastive objective.

Not all class relationships are equally informative for regularization. Many class pairs may already be well separated, whereas a smaller number of negative classes may exhibit high distributional similarity and therefore provide more informative training signals. Prior metric-learning studies have shown that focusing on informative or hard negative relationships can improve the effectiveness of discriminative learning (Wu et al., 2017). Motivated by this observation, BDCR maintains momentum-smoothed class statistics and dynamically selects hard-negative classes according to their distributional similarity. Unlike conventional hard-negative mining, which typically selects difficult individual samples, BDCR performs negative selection directly between class-conditional distributions. The resulting discrete selection defines the class pairs on which the distributional regularizer is applied, without differentiating through the selection process. This allows the optimization signal to focus on difficult class-level relationships while keeping the regularization objective compatible with standard gradient-based training.

BDCR requires no architectural modification and can be incorporated into standard classification objectives as an auxiliary regularizer. By using class-conditional distributions as the units of representation and comparison, and by performing hard-negative selection at the class level, BDCR explicitly models both feature location and intra-class variation while concentrating regularization on informative inter-class relationships. The resulting framework is architecture-agnostic and can be applied to diverse visual classification models.

Our contributions are summarized as follows:

- We formulate discriminative representation learning from a class-distribution perspective, treating an approximate class-conditional feature distribution as the fundamental unit for modeling intra-class variation and inter-class separation.
- We propose **BDCR**, a Bhattacharyya-distance-based distributional contrastive regularizer that encourages consistency among same-class distributions while promoting separation between different classes through their joint feature location and variation.
- We introduce a momentum-based class-level hard-negative selection strategy that identifies distributionally similar negative classes and concentrates regularization on these informative relationships without differentiating through the discrete selection process.
- We demonstrate the effectiveness of BDCR across diverse classification architectures and datasets.

2 Related Work

Metric and Contrastive Representation Learning. Metric and contrastive learning methods improve discriminative representations by shaping feature-space geometry, including contrastive and triplet losses (Hadsell et al., 2006; Schroff et al., 2015), N-pair and lifted structured losses (Sohn, 2016), center-based regularization (Wen et al., 2016), and angular-margin objectives (Liu et al., 2017; Wang et al., 2018; Deng et al., 2019). Supervised contrastive learning further exploits class labels to construct positive and negative sample relationships (Khosla et al., 2020). Prototype- and proxy-based methods instead represent classes using point embeddings (Snell et al., 2017; Movshovitz-Attias et al., 2017; Kim et al., 2020). These approaches primarily model individual features, pairwise relationships, or class locations, without explicitly incorporating intra-class variation into the class-level discriminative objective. BDCR instead uses class-conditional distributions as the unit of regularization, allowing both feature location and variation to contribute to class separation.

Distributional Representation Learning. Distributional representations model variation and uncertainty beyond deterministic point embeddings. Distributional distances such as MMD (Gretton et al., 2012; Sriperumbudur et al., 2012) and Wasserstein distance (Arjovsky et al., 2017), as well as probabilistic embeddings such as Word2Gauss (Vilnis & McCallum, 2014), provide general frameworks for representing or comparing distributions. In contrast, BDCR constructs a diagonal Gaus-

sian from the features of each semantic class and uses it as the unit of supervised discriminative regularization. The Bhattacharyya distance (Bhattacharyya, 1943) provides a symmetric, closed-form comparison that jointly captures differences in class means and variances, yielding an efficient objective compatible with standard classification training.

Hard-Negative Mining. Hard-negative mining focuses metric and contrastive learning on difficult negative relationships, from instance-level batch-hard mining (Hermans et al., 2017; Oh Song et al., 2016) to hardness-aware sampling and contrastive objectives (Zheng et al., 2019; Wang et al., 2019; Sun et al., 2020; Robinson et al., 2020; Xia et al., 2021). Class-aware methods can further identify difficult negative classes before selecting hard examples (Suh et al., 2019). Unlike these approaches, BDCR defines hardness directly between class-conditional distributions, using momentum-smoothed statistics to select the most confusable negative classes. The selected distribution pairs then receive the regularization signal without differentiating through the discrete selection process.

3 METHOD

3.1 OVERVIEW

We propose **BDCR**, a distributional contrastive regularizer for well-separated class representations. Let $f_\theta(\mathbf{x}) \in \mathbb{R}^d$ denote the representation produced by a neural network for an input $\mathbf{x}$. For each class c, we model its features as a Gaussian with mean and diagonal covariance, capturing location and uncertainty. Class overlap is measured by the Bhattacharyya distance. The regularizer enforces intra-class distribution consistency across subsets while increasing inter-class separation. A momentum-based hard-negative graph, built from an exponentially smoothed history of pairwise class distances, selects the most confusable classes and provides stabler geometry than per-batch distance estimates. This graph is updated dynamically and treated as non-differentiable; gradients flow only through the Bhattacharyya regularization on selected pairs.

BDCR consists of three components: (i) class-conditional distribution modeling, (ii) Bhattacharyya-based distributional contrastive regularization, and (iii) momentum-based hard-negative selection.

3.2 CLASS-CONDITIONAL DISTRIBUTION MODELING

The proposed regularizer does not act on individual feature vectors. Instead, it regularizes the class-conditional distribution of representations. For each class c, we model its representation distribution as a Gaussian,

$$p_c(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right), \tag{1}$$

where $\mathbf{z} = f_\theta(\mathbf{x}) \in \mathbb{R}^d$ is the network representation, $\boldsymbol{\mu}_c$ denotes the class center, and $\boldsymbol{\Sigma}_c$ captures the class-wise variation or uncertainty. Thus, each class is represented as a distribution rather than as a single prototype or a set of independent embeddings.

For efficiency and numerical stability, we use a diagonal covariance,

$$\boldsymbol{\Sigma}_c = \operatorname{diag}\left(\sigma_{c,1}^2, \ldots, \sigma_{c,d}^2\right). \tag{2}$$

This choice reduces the number of parameters from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ per class and makes inversion and determinant computations elementwise. It also avoids ill-conditioned covariance estimates in the small-sample regime typical of minibatch training. For numerical stability, all variance estimates are further bounded below by a small variance floor before computing the Bhattacharyya distance; the complete stabilization procedure is described in Appendix A.2.

In practice, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ can be estimated from the samples of class c in the current minibatch, or maintained as running statistics. The detailed minibatch estimation procedure, including the minimum sample requirement and the construction of two independent subsets for positive pairs, is provided in Appendix A.1. For the contrastive formulation, two independent estimates can be constructed from two subsets of samples: $p_c^{(1)} = \mathcal{N}(\boldsymbol{\mu}_c^{(1)}, \boldsymbol{\Sigma}_c^{(1)})$ and $p_c^{(2)} = \mathcal{N}(\boldsymbol{\mu}_c^{(2)}, \boldsymbol{\Sigma}_c^{(2)})$. The Bhattacharyya distance is then applied directly to these class-conditional distributions. Its definition and closed-form expression for Gaussian distributions are given in the following section, where the distance serves as the basic measure of distributional separation and overlap between classes.

3.3 BHATTACHARYYA DISTANCE

To quantify the separation between class-conditional distributions, we use the Bhattacharyya distance. Compared with a distance defined only between class means, the Bhattacharyya distance jointly considers the displacement of the distribution centers and their associated uncertainties.

For two Gaussian distributions $p_i = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $p_j = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, the Bhattacharyya distance is given by

$$D_B(p_i, p_j) = \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_{ij}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\log \frac{\det(\boldsymbol{\Sigma}_{ij})}{\sqrt{\det(\boldsymbol{\Sigma}_i)\det(\boldsymbol{\Sigma}_j)}}, \tag{3}$$

where $\boldsymbol{\Sigma}_{ij} = \dfrac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2}$. The first term measures the discrepancy between the distribution means while accounting for their variances. The second term measures the discrepancy between the distribution shapes. Consequently, two classes can have a large Bhattacharyya distance either because their centers are far apart, because their variances differ substantially, or because both effects occur simultaneously.

With diagonal covariance matrices, the distance decomposes into a sum over feature dimensions. Let σ_{ik}^2 denote the variance of class i in dimension k, and define $s_{ijk}^2 = (\sigma_{ik}^2 + \sigma_{jk}^2)/2$. The Bhattacharyya distance is then evaluated as

$$D_B(p_i, p_j) = \frac{1}{d}\sum_{k=1}^{d}\left[\frac{(\mu_{ik} - \mu_{jk})^2}{8s_{ijk}^2} + \frac{1}{2}\log \frac{s_{ijk}^2}{\sqrt{\sigma_{ik}^2\sigma_{jk}^2}}\right]. \tag{4}$$

A small value of $D_B(p_i, p_j)$ indicates substantial overlap between the two class distributions, whereas a large value indicates that the distributions are well separated. This property makes the Bhattacharyya distance particularly suitable for identifying ambiguous class pairs and constructing a distribution-level contrastive objective.

3.4 BDCR: DISTRIBUTIONAL CONTRASTIVE REGULARIZATION

The Bhattacharyya distance provides a natural measure of distributional separation, but a distance alone does not specify how the representation space should be optimized. We therefore formulate **BDCR** as a distributional contrastive regularizer that explicitly encourages intra-class distributional consistency and inter-class separation.

Distributional positive pairs. For a class c, we randomly divide its available samples into two subsets and estimate two class-conditional distributions, denoted by $p_c^{(1)}$ and $p_c^{(2)}$ as defined in Section 3.2. These two distributions describe the same semantic class from different subsets of samples. Their Bhattacharyya distance is therefore used as the positive distance,

$$d_c^+ = D_B\left(p_c^{(1)}, p_c^{(2)}\right). \tag{5}$$

Minimizing this distance encourages the representation distribution of a class to be consistent across different subsets of its samples.

Distributional negative pairs. For a different class $j \neq c$, we define the negative distance as $d_{cj}^- = D_B(p_c^{(1)}, p_j^{(2)})$. Unlike the positive pair, the negative pair should have a large distributional distance. This formulation therefore operates on class-level distributions rather than individual feature pairs.

We convert the Bhattacharyya distance into a similarity score through $s_B(p_i, p_j) = -\kappa D_B(p_i, p_j)$, where $\kappa > 0$ controls the concentration of the contrastive objective.

Let $\mathcal{N}(c)$ denote the set of negative classes associated with class c. The distributional contrastive regularization loss is defined as

$$\mathcal{L}_{\mathrm{BDCR}} = -\frac{1}{|\mathcal{C}_B|}\sum_{c\in\mathcal{C}_B}\log \frac{\exp(-\kappa d_c^+)}{\exp(-\kappa d_c^+) + \sum_{j\in\mathcal{N}(c)}\exp(-\kappa d_{cj}^-)}, \tag{6}$$

where $\mathcal{C}_B$ denotes the set of classes for which the required distributional statistics are available.

Equivalently, Eq. (6) can be written as

$$\mathcal{L}_{\mathrm{BDCR}} = \frac{1}{|\mathcal{C}_B|} \sum_{c \in \mathcal{C}_B} \log \left[1 + \sum_{j \in \mathcal{N}(c)} \exp\left(-\kappa \left(d_{cj}^{-} - d_c^{+}\right)\right) \right]. \tag{7}$$

This formulation reveals the desired geometry directly. A negative class whose distribution is close to the anchor class produces a small d_{cj}^{-} and consequently a large contribution to the loss. In contrast, a sufficiently separated negative class contributes little to the objective. The regularizer therefore automatically emphasizes ambiguous class pairs.

The parameter κ controls the sharpness of this emphasis. In particular, for a collection of negative distances $\{D_B(p_c, p_j)\}$, the corresponding soft minimum is

$$\mathrm{softmin}_\kappa\{D_B(p_c, p_j)\} = -\frac{1}{\kappa} \log \sum_j \exp\left(-\kappa D_B(p_c, p_j)\right). \tag{8}$$

As κ increases, the objective increasingly concentrates on the smallest negative Bhattacharyya distances. Thus, the distributional contrastive objective naturally has a hard-negative interpretation, while remaining differentiable with respect to the selected distributional distances.

3.5 Momentum-Based Hard-Negative Graph

Although the distributional contrastive objective naturally assigns larger weights to nearby negative distributions, explicitly considering all class pairs is undesirable when the number of classes is large. Moreover, instantaneous minibatch statistics can be noisy and may lead to unstable hard-negative assignments.

To address these issues, we construct a dynamic hard-negative graph using momentum class statistics. The key distinction is that the momentum statistics are used to estimate stable distributional geometry and select hard negatives, while the current representations are used to compute the gradient-bearing regularization loss.

Momentum distributional geometry. For each class c, we maintain exponentially smoothed distributional statistics. The class mean is updated according to

$$\boldsymbol{\mu}_c^{(t)} = \beta \boldsymbol{\mu}_c^{(t-1)} + (1-\beta)\widehat{\boldsymbol{\mu}}_c^{(t)}, \tag{9}$$

while the class-wise variance is updated as

$$\boldsymbol{\sigma}_c^{2(t)} = \beta \boldsymbol{\sigma}_c^{2(t-1)} + (1-\beta)\widehat{\boldsymbol{\sigma}}_c^{2(t)}. \tag{10}$$

The resulting momentum distribution for class c is

$$\widetilde{p}_c^{(t)} = \mathcal{N}\left(\boldsymbol{\mu}_c^{(t)}, \mathrm{diag}\left(\boldsymbol{\sigma}_c^{2(t)}\right)\right). \tag{11}$$

These running distributions provide a temporally smoothed estimate of the global class geometry. Compared with statistics computed from a single minibatch, they are less sensitive to sampling fluctuations and are therefore better suited for determining which class pairs are currently ambiguous. The complete momentum update procedure and its use in hard-negative graph construction are detailed in Appendix A.3.

Dynamic hard-negative graph. At iteration t, we compute the distributional similarity between classes using the momentum distributions. Specifically, $\widetilde{D}_{cj}^{(t)} = D_B(\widetilde{p}_c^{(t)}, \widetilde{p}_j^{(t)})$ for $j \neq c$.

For each anchor class c, we select the K classes with the smallest Bhattacharyya distances:

$$\mathcal{N}_K^{(t)}(c) = \mathrm{TopK}_{j \neq c}^{\mathrm{smallest}}\left(\widetilde{D}_{cj}^{(t)}\right). \tag{12}$$

This defines a dynamic directed graph

$$\mathcal{G}^{(t)} = \left(\mathcal{V}, \mathcal{E}^{(t)}\right), \tag{13}$$

where each node corresponds to a semantic class and an edge satisfies $(c, j) \in \mathcal{E}^{(t)}$ if and only if $j \in \mathcal{N}_K^{(t)}(c)$.

The graph therefore concentrates the regularization budget on class pairs whose current distributions exhibit the largest overlap. Classes that are already well separated do not need to receive the same amount of regularization.

Graph-guided distributional contrastive regularization. The hard-negative graph is incorporated into BDCR by setting $\mathcal{N}(c) = \mathcal{N}_K^{(t)}(c)$. The resulting loss is

$$\mathcal{L}_{\mathrm{BDCR}}^{(t)} = -\frac{1}{|\mathcal{C}_B|} \sum_{c \in \mathcal{C}_B} \log \frac{\exp(-\kappa d_c^+)}{\exp(-\kappa d_c^+) + \sum_{j \in \mathcal{N}_K^{(t)}(c)} \exp(-\kappa d_{cj}^-)}. \tag{14}$$

The graph construction itself is discrete and is not differentiated through. In other words, the Top-K operation only determines which negative pairs participate in the regularizer. Once the graph is constructed, gradients are computed through the Bhattacharyya distances of the selected pairs using the current representations.

This separation gives the proposed method two complementary properties. First, momentum statistics provide stable hard-negative identification. Second, the current representations remain directly optimized by the distributional contrastive objective. The momentum mechanism therefore acts as a stable estimator of the evolving class geometry rather than as a separate representation branch.

Computational consideration. Evaluating one Bhattacharyya distance between diagonal Gaussians costs $O(d)$. With the hard-negative graph, the BDCR loss over the selected K negative edges for all classes costs $O(CKd)$, where C is the number of classes and $K \ll C$. Exact Top-K construction requires $O(C^2 d)$ distance computations, which is separate from the gradient-bearing regularization cost; for large C, approximate retrieval can reduce this cost. See Appendix C for details.

3.6 Overall Objective and Optimization

Our regularizer complements the standard classification objective. For a training sample $(\mathbf{x}, y)$, let $\mathcal{L}_{\mathrm{CE}}$ denote the classification loss. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{BDCR}}, \tag{15}$$

where $\lambda \geq 0$ controls the strength of the distributional regularizer.

At each iteration, minibatch features yield class-conditional means and variances. Momentum class statistics are updated, and Bhattacharyya distances between momentum distributions form a dynamic Top-K hard-negative graph. The BDCR loss is then computed over the selected edges. The complete training procedure is summarized in Appendix A.4. Since Top-K selection is non-differentiable, gradients of $\mathcal{L}_{\mathrm{BDCR}}$ flow through the current class statistics to the feature extractor. This jointly preserves supervised discriminability and shapes the distribution-level geometry of the representation space, with $\mathcal{L}_{\mathrm{CE}}$ providing primary task supervision and $\mathcal{L}_{\mathrm{BDCR}}$ emphasizing hard-to-separate classes.

4 Experiments

4.1 Experimental Setup

4.1.1 Implementation Details

All experiments are conducted on NVIDIA GPUs using distributed data-parallel training. The primary system uses four V100-SXM2 16GB GPUs; a secondary system uses three RTX 3090 GPUs.

For all experiments, we initialize the backbone with pre-trained models and only train the parameters of the classification head from scratch, ensuring a fair comparison across different methods. For ResNet and MobileNet architectures, we use SGD with Nesterov momentum of 0.9 and weight decay of 1×10^{-4}. The learning rate is linearly scaled with the effective batch size as $0.1B/256$, followed by cosine decay with a 10-epoch warmup. For DeiT and other Transformer-based architectures, we use AdamW with a learning rate of 5×10^{-5} for the backbone and 5×10^{-4} for the classification head, with weight decay of 1×10^{-4}. ImageNet-1K experiments are trained for 100 epochs, while the smaller-dataset experiments use 50 epochs unless otherwise specified. The regularization coefficient is set to $\lambda = 0.2$ in all experiments.

4.1.2 DATASETS

We evaluate our method on five benchmark datasets covering coarse-grained, fine-grained, and texture classification tasks: ImageNet-1K, CIFAR-10, CIFAR-100, CUB-200-2011, and DTD.

ImageNet-1K Deng et al. (2009) contains 1,000 object classes with approximately 1.28M training images and 50K validation images. Images are resized to 224×224 for both training and evaluation.

CIFAR-10 and **CIFAR-100** Krizhevsky et al. (2009) consist of 60,000 32×32 color images each, with 50,000 training images and 10,000 test images. CIFAR-10 contains 10 classes, while CIFAR-100 contains 100 classes.

CUB-200-2011 Wah et al. (2011) is a fine-grained bird classification dataset with 200 species and 11,788 images, split into 5,994 training images and 5,794 testing images.

DTD Cimpoi et al. (2014) is a describable texture dataset containing 5,640 images from 47 texture categories. We follow the standard training/validation/test split.

These datasets provide a diverse evaluation of the proposed method under different levels of granularity and data regimes.

4.2 MAIN RESULTS

We evaluate BDCR as a distribution-level auxiliary regularizer across architectures and datasets, focusing on cross-architecture generalization, class-level negative selection, and momentum statistics. All variants share the same distributional contrastive objective and differ only in their negative-class selection strategy; details are provided in Appendix B. Further results, including additional ImageNet evaluations, CIFAR-10 ablations, and CIFAR-100 analyses, are reported in Appendix G–F.

Cross-architecture evaluation on ImageNet-1K. Table 1 reports ImageNet-1K top-1 accuracy using DeiT-tiny Touvron et al. (2021) (100 epochs) and ResNet18 He et al. (2016) under different training budgets and GPU settings. On DeiT-tiny, BDCR-Hard achieves 74.60%, improving over the 74.37% baseline, while BDCR-Uniform and BDCR-Logit reach 74.58% and BDCR-Momentum 74.50%.

On ResNet18, the best strategy varies with the training budget. At 150 epochs, BDCR-Uniform achieves 71.88% on the 4-GPU server, compared with 71.63% for the baseline, while BDCR-Momentum reaches 71.80% on the 3-GPU server versus 71.52%. At 250 epochs, all regularized configurations outperform their corresponding baselines, with BDCR-Hard achieving 72.28% on the 3-GPU server and BDCR-Momentum with κ annealing reaching 72.34% on the 4-GPU server. The advantage persists at longer budgets, with BDCR-Hard improving the 3-GPU result from 71.99% to 72.27% at 350 epochs and the 4-GPU result from 72.20% to 72.53% at 500 epochs.

Overall, BDCR improves both transformer and convolutional architectures, with gains generally becoming larger under longer training budgets. The strongest negative-selection strategy depends on the architecture and training setting, while Hard and Momentum variants consistently provide competitive improvements.

Smaller-data regimes. We further evaluate BDCR on CIFAR-10, DTD, CUB-200, and CIFAR-100 using ResNet18 with 50-epoch training. As shown in Table 2, BDCR improves the baseline in most settings. BDCR-Momentum is best on CIFAR-10 (96.34%) and DTD (65.90%), and reaches the best CIFAR-100 result together with BDCR-Hard (82.94%). On CUB-200, BDCR-

Table 1: Top-1 accuracy (%) on ImageNet-1K. DeiT-Tiny is trained for 100 epochs. ResNet-18 results are reported at 150, 250, 350, and 500 epochs on 3-GPU and 4-GPU configurations. "–" indicates that the corresponding configuration was not evaluated. Best results in each column are shown in bold.

Method	DeiT-Tiny 100 ep.	ResNet-18 150 ep.		ResNet-18 250 ep.		ResNet-18 350 ep.	ResNet-18 500 ep.
		3-GPU	4-GPU	3-GPU	4-GPU	3-GPU	4-GPU
Baseline	74.37	71.52	71.63	71.88	72.00	71.99	72.20
BDCR-Hard	**74.60**	71.74	71.74	**72.28**	72.23	**72.27**	**72.53**
BDCR-Momentum	74.50	**71.80**	71.85	72.13	72.24	72.19	–
BDCR-Momentum (anneal)	–	71.82	72.03	72.09	**72.34**	72.18	–
BDCR-Uniform	74.58	71.65	**71.88**	72.20	72.20	72.17	–

Table 2: Top-1 accuracy (%) on smaller datasets using ResNet18 trained for 50 epochs.

Method	CIFAR-10	DTD	CUB-200	CIFAR-100
Baseline	96.03	64.73	76.96	82.64
BDCR-Hard	96.22	65.05	**77.30**	**82.94**
BDCR-Momentum	**96.34**	**65.90**	77.11	**82.94**
BDCR-Uniform	96.18	64.47	77.25	82.84
BDCR-Logit	96.24	64.89	77.13	82.89

Hard achieves the best result at 77.30%. On CIFAR-100, every BDCR variant exceeds the baseline of 82.64%, with BDCR-Logit at 82.89% and BDCR-Uniform at 82.84%. On DTD, BDCR-Uniform is below the baseline (64.47% vs. 64.73%).

The different behaviors across datasets suggest that hard-negative selection and momentum statistics provide complementary benefits. The former focuses the regularization on difficult class pairs, while the latter stabilizes class-conditional distribution estimates. Additional tuned configurations reach 66.53% on DTD with momentum-based temperature annealing and 77.41% on CUB-200-2011 with $K = 20$ hard negatives.

4.3 Ablation Studies

Effect of training length. Table 3 compares 30- and 50-epoch CIFAR-100 training with ResNet18. Longer training improves every configuration. At 30 epochs, BDCR-Hard is best (82.62%), followed by BDCR-Momentum (82.57%) and BDCR-Uniform (82.48%); at 50 epochs, BDCR-Hard and BDCR-Momentum are tied for best at 82.94%. All BDCR variants exceed the baseline in both schedules. The gain is smallest at 30 epochs for BDCR-Uniform (82.48% vs. 82.25%) and largest at 50 epochs for BDCR-Hard and BDCR-Momentum (82.94% vs. 82.64%). The relative ordering of the sampling strategies is not identical across the two schedules, but the spread between the best and the weakest regularized configuration remains under 0.15 percentage points at both budgets, indicating that the choice of negative-selection mechanism is not the dominant factor at this scale.

Effect of κ and λ. Figure 1 evaluates the effect of the concentration parameter κ across $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ for different negative-sampling and graph-construction variants. The baseline is shown as a horizontal reference.

Across all regularized variants, accuracy remains relatively stable for $\kappa \leq 10$ but drops substantially at $\kappa = 100$. This indicates that excessively large κ makes the contrastive objective overly concentrated on the hardest negative distributions, potentially over-penalizing ambiguous class pairs and harming generalization. We therefore adopt $\kappa = 10$, which provides strong distributional contrast

Table 3: Effect of training length on CIFAR-100 top-1 accuracy (%) using ResNet18. "Best" values are the highest validation accuracy observed during training; the best result in each column is shown in bold.

Mode	30 epochs	50 epochs
Baseline	82.25	82.64
BDCR-Hard	**82.62**	**82.94**
BDCR-Momentum	82.57	**82.94**
BDCR-Uniform	82.48	82.84
BDCR-Soft	82.51	82.79
BDCR-Logit	82.50	82.89

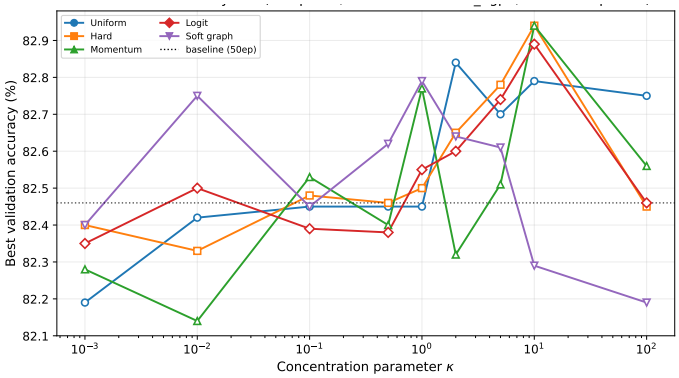

Figure 1: Sensitivity of different hard-negative strategies to the concentration parameter κ. We vary $\kappa \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ and compare Uniform, Hard, Momentum, Logit, and Soft graph regularization against the baseline. The baseline is independent of κ and serves as a reference. Accuracy is relatively stable for $\kappa \leq 10$, but drops sharply at $\kappa = 100$. Since larger κ increases the influence of the distributional contrastive loss, we set $\kappa = 10$ in all main experiments.

while remaining within the stable range. Accordingly, we set $\lambda = 0.2$ to balance the regularizer with the classification loss.

5 Discussion

Our results suggest that distribution-level regularization provides a complementary signal to conventional classification objectives. By modeling each class through both its center and variation, BDCR captures aspects of class overlap that are not represented by point-based prototypes. Its consistent gains across convolutional and Transformer architectures also suggest that the formulation is largely backbone-independent.

The results further highlight the importance of focusing regularization on ambiguous class relationships. Class-level hard-negative selection and momentum statistics generally provide complementary benefits, particularly in smaller-data regimes. BDCR has two main limitations: the diagonal covariance assumption does not capture correlations between feature dimensions, and exact class-level Top-K selection can become costly for large numbers of classes. Future work could explore richer covariance models and scalable class-level retrieval.

6 Conclusion

We presented BDCR, a distributional contrastive regularization framework for learning well-separated representations. BDCR represents semantic classes as Gaussian distributions and uses the Bhattacharyya distance to jointly model class location and variation. A momentum-based class-level hard-negative mechanism further focuses regularization on difficult class relationships without

differentiating through the discrete selection process. Experiments across multiple architectures and datasets demonstrate that BDCR provides a simple and architecture-agnostic complement to standard classification training. These results support class-conditional distribution geometry as a useful direction for discriminative representation learning.

AI USE STATEMENT

We used generative AI tools for English language polishing, grammar correction, restructuring and rewriting of manuscript text, and assistance with code development. We did not use generative AI tools to generate experimental results, fabricate data, or replace the authors' scientific judgment in the design, execution, or interpretation of the experiments. All AI-assisted text and code were reviewed by the authors, and AI-assisted code was manually checked and experimentally verified before use. The authors take full responsibility for the final content of this work, including all claims, experimental results, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, private data, or experiments requiring institutional review board (IRB) approval. All experiments are conducted on publicly available benchmark datasets, including ImageNet-1K, CIFAR, DTD, and CUB-200. We use these datasets solely for research and evaluation of representation learning and regularization methods. We are not aware of any additional ethical concerns specific to the methods or experiments presented in this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method and experimental settings in the main paper and appendix. The appendix includes additional implementation and experimental details needed to reproduce the reported results. The training and evaluation code will be provided as supplementary material, including the configurations used for the experiments. Training logs are also retained to facilitate verification of the reported results. All experiments use publicly available datasets, including ImageNet-1K, CIFAR, DTD, and CUB-200. Together, the paper, appendix, supplementary code, and training logs provide the necessary information and resources for reproducing our experiments.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223. PMLR, 2017.

Anil Kumar Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pp. 539–546. IEEE, 2005.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

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*, pp. 770–778, 2016.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person reidentification. *arXiv preprint arXiv:1703.07737*, 2017.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3235–3244. IEEE, 2020.

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

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 212–220, 2017.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9992–10002. Ieee, 2021.

Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 360–368. IEEE, 2017.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7244–7252. IEEE, 2019.

Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 6397–6406. IEEE, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5017–5025. IEEE, 2019.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pp. 499–515. Springer, 2016.

Chao-Yuan Wu, Raghavan Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2859–2867. IEEE, 2017.

Jun Xia, Lirong Wu, Ge Wang, Jintao Chen, and Stan Z Li. Progcl: Rethinking hard negative mining in graph contrastive learning. *arXiv preprint arXiv:2110.02027*, 2021.

Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 72–81, 2019.

A Additional Method Details

A.1 Class-wise Statistics Estimation

For each minibatch, we first group the feature representations according to their ground-truth classes. Let $\mathcal{B}_c$ denote the set of samples belonging to class c in the current minibatch, and let $\mathbf{z}_i = f_\theta(\mathbf{x}_i)$ denote the corresponding feature representation. The empirical class mean is computed as

$$\widehat{\boldsymbol{\mu}}_c = \frac{1}{|\mathcal{B}_c|} \sum_{i \in \mathcal{B}_c} \mathbf{z}_i, \tag{16}$$

while the diagonal variance is estimated dimension-wise as

$$\widehat{\boldsymbol{\sigma}}_c^2 = \frac{1}{|\mathcal{B}_c|} \sum_{i \in \mathcal{B}_c} \left(\mathbf{z}_i - \widehat{\boldsymbol{\mu}}_c\right)^2. \tag{17}$$

Only classes with a sufficient number of samples are used for distributional regularization. Specifically, we require

$$|\mathcal{B}_c| \geq n_{\min}, \tag{18}$$

where $n_{\min}$ is the minimum number of samples required to construct a reliable class distribution.

For the positive distributional pair, the valid samples of each class are randomly divided into two subsets. The corresponding statistics are used to construct $p_c^{(1)}$ and $p_c^{(2)}$. This procedure provides two independent estimates of the same class distribution and prevents the positive term from being computed from identical statistics.

A.2 Numerical Stabilization

The empirical variance of a class may become very small when only a limited number of samples are available or when the corresponding feature dimension has low variability. To avoid numerical instability in the Bhattacharyya distance, we impose a small variance floor:

$$\widetilde{\boldsymbol{\sigma}}_c^2 = \max\left(\widehat{\boldsymbol{\sigma}}_c^2, \epsilon\right), \tag{19}$$

where the maximum operation is applied element-wise and $\epsilon > 0$ is a small numerical constant.

The stabilized diagonal covariance is then given by

$$\widetilde{\boldsymbol{\Sigma}}_c = \operatorname{diag}\left(\widetilde{\boldsymbol{\sigma}}_c^2\right). \tag{20}$$

The same variance floor is applied to both minibatch and momentum statistics. This stabilization prevents near-zero variances from producing excessively large inverse-covariance terms or undefined logarithms in the Bhattacharyya distance.

A.3 Momentum Statistics Update

To obtain a temporally stable estimate of the class-wise distributional geometry, we maintain exponential moving averages of the class means and variances. Let $\widehat{\boldsymbol{\mu}}_c^{(t)}$ and $\widehat{\boldsymbol{\sigma}}_c^{2(t)}$ denote the statistics estimated from the current minibatch. The momentum statistics are updated as

$$\boldsymbol{\mu}_c^{(t)} = \beta \boldsymbol{\mu}_c^{(t-1)} + (1-\beta)\widehat{\boldsymbol{\mu}}_c^{(t)}, \tag{21}$$

and

$$\boldsymbol{\sigma}_c^{2(t)} = \beta \boldsymbol{\sigma}_c^{2(t-1)} + (1-\beta)\widehat{\boldsymbol{\sigma}}_c^{2(t)}. \tag{22}$$

The momentum distribution is consequently defined as

$$\widetilde{p}_c^{(t)} = \mathcal{N}\left(\boldsymbol{\mu}_c^{(t)}, \operatorname{diag}\left(\boldsymbol{\sigma}_c^{2(t)}\right)\right). \tag{23}$$

At each training iteration, the momentum statistics are used only for constructing the hard-negative graph. Specifically, we compute

$$\widetilde{D}_{cj}^{(t)} = D_B\left(\widetilde{p}_c^{(t)}, \widetilde{p}_j^{(t)}\right) \tag{24}$$

and select the K nearest classes for each anchor class.

The graph construction is treated as a non-differentiable operation. Therefore, gradients are not propagated through the momentum statistics or the Top-K selection. Once the graph is constructed, the gradient-bearing BDCR loss is computed using the current feature representations and their corresponding class statistics.

A.4 Training Procedure

The complete training procedure is summarized in Algorithm 1. At each iteration, the network first produces feature representations for the current minibatch. Class-wise statistics are then computed from the current features and used to update the momentum statistics. The updated momentum distributions determine the hard-negative graph through a non-differentiable Top-K operation. Finally, the distributional contrastive loss is evaluated on the selected graph edges and combined with the classification loss for parameter optimization.

Algorithm 1 Training with Momentum-Based Hard-Negative BDCR

Require: Network f_θ, minibatch $\mathcal{B}$, momentum coefficient β, number of negatives K, regularization weight λ

for each training iteration **do**
  $\mathbf{z}_i \leftarrow f_\theta(\mathbf{x}_i)$ for $(\mathbf{x}_i, y_i) \in \mathcal{B}$
  Compute current class statistics $\{\widehat{\boldsymbol{\mu}}_c, \widehat{\boldsymbol{\sigma}}_c^2\}$
  Update momentum statistics using β
  Compute pairwise Bhattacharyya distances $\widetilde{D}_{cj}$ between valid momentum distributions
  Select Top-K nearest classes for each anchor class to form the hard-negative graph $\mathcal{G}$
  Construct current positive distributions $p_c^{(1)}$ and $p_c^{(2)}$
  Compute $\mathcal{L}_{\mathrm{BDCR}}$ using $\mathcal{G}$ and current distributions
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{BDCR}}$
  Update θ by back-propagation
end for

B Alternative Hard-Negative Selection Strategies

To investigate the effect of negative-class selection, we compare the proposed momentum-based hard-negative graph with several alternative strategies. All variants use the same distributional contrastive objective in Eq. (6); they differ only in how negative classes are selected.

B.1 Uniform Negative Sampling

As the simplest baseline, we uniformly sample K negative classes for each anchor class:

$$\mathcal{N}_K^{\mathrm{uni}}(c) \sim \mathrm{Uniform}\left(\{1, \ldots, C\} \setminus \{c\}\right). \tag{25}$$

The selected classes are then used directly in the BDCR objective. This strategy does not require an explicit hard-negative graph and serves as a baseline for evaluating whether targeted negative selection provides additional benefit.

B.2 Batch Distributional Hard-Negative Selection

A stronger baseline selects hard negatives directly according to the Bhattacharyya distance computed from the current minibatch statistics. Specifically, for each class c, we compute

$$D_{cj}^{\mathrm{batch}} = D_B\left(\widehat{p}_c, \widehat{p}_j\right), \tag{26}$$

where $\widehat{p}_c$ denotes the Gaussian distribution estimated from the current minibatch.

The K nearest negative classes are selected according to

$$\mathcal{N}_K^{\mathrm{batch}}(c) = \mathrm{TopK}_{j \neq c}^{\mathrm{smallest}}\left(D_{cj}^{\mathrm{batch}}\right). \tag{27}$$

The selected graph is then used in the same BDCR objective as the proposed method. Unlike our momentum-based strategy, this variant relies entirely on instantaneous minibatch statistics and is therefore more sensitive to sampling noise.

B.3 Logit-Guided Hard-Negative Selection

We further consider a classifier-guided hard-negative strategy. For each class c, let $\mathcal{B}_c$ denote the samples of class c in the current minibatch and let $\mathbf{z}(x) \in \mathbb{R}^C$ denote the classifier logits. We compute the mean logit vector

$$\overline{\mathbf{z}}_c = \frac{1}{|\mathcal{B}_c|} \sum_{x \in \mathcal{B}_c} \mathbf{z}(x). \tag{28}$$

The K classes with the largest competing logits are selected as hard negatives:

$$\mathcal{N}_K^{\mathrm{logit}}(c) = \mathrm{TopK}_{j \neq c}^{\mathrm{largest}}\left(\overline{z}_{cj}\right). \tag{29}$$

The resulting class-level graph is used in the same BDCR objective. This baseline identifies hard negatives according to classifier confusion rather than distributional overlap, allowing us to assess whether the proposed distribution-based graph provides complementary information beyond the classifier logits.

C Computational Complexity

We analyze the computational cost of the proposed distributional regularization in terms of the number of classes C, the feature dimension d, and the number of selected hard negatives K.

For diagonal Gaussian distributions, the Bhattacharyya distance decomposes over feature dimensions. Therefore, evaluating one distributional distance requires

$$O(d) \tag{30}$$

operations.

If all class pairs are considered, the number of distance evaluations is

$$\binom{C}{2} = O(C^2), \tag{31}$$

resulting in an overall distance-computation complexity of

$$O(C^2 d). \tag{32}$$

With the proposed Top-K hard-negative graph, the gradient-bearing BDCR loss requires only K negative comparisons per anchor class. Its regularization cost is therefore

$$O(CKd), \tag{33}$$

where $K \ll C$ in our experiments.

It is important to distinguish this cost from the construction of an exact hard-negative graph. Exact nearest-neighbor selection requires evaluating all candidate class pairs and therefore still has

$$O(C^2 d) \tag{34}$$

complexity. The main computational benefit of the proposed formulation is that only the selected CK edges participate in the gradient-bearing regularization.

For larger-scale classification problems, the graph construction can be approximated using a candidate pool or efficient nearest-neighbor retrieval. If M candidate classes are examined for each anchor, where $K \ll M \ll C$, the corresponding distance computation can be reduced to approximately

$$O(CMd). \tag{35}$$

Thus, the framework provides a controllable trade-off between negative diversity, hardness, and computational cost.

D Experimental Details

D.1 Training Configuration

We provide the detailed training configurations used in all experiments. Unless otherwise specified, all models are trained using the same data augmentation, optimization strategy, learning-rate schedule, and number of training epochs as the corresponding baseline implementation.

The proposed regularizer is added to the standard classification objective according to

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{BDCR}}. \tag{36}$$

The number of selected hard negatives is denoted by K, the temperature-like coefficient by κ, and the momentum coefficient by β. Unless otherwise stated, these hyperparameters are fixed across

Table 4: Detailed training configurations.

Model	Epochs	Batch Size	Optimizer	LR	K	κ	λ
ResNet-18	150	256	SGD	0.01	15	10	0.2
DeiT-Tiny	100	256	AdamW	0.0005	15	10	0.2
Swin-Tiny	300	128	AdamW	0.0005	15	10	0.2

Table 5: Detailed results across different architectures.

Architecture	Baseline	BDCR
ResNet-18	71.63	71.88
DeiT-Tiny	74.37	74.60
Swin-Tiny	79.13	79.79

architectures and datasets. The following configurations are specifically used for ImageNet-1K. Detailed architecture-specific configurations are summarized in Table 4. Specifically, ResNet-18 is trained for 150 epochs with batch size 256 using SGD and a learning rate of 0.01; DeiT-Tiny is trained for 100 epochs with batch size 256 using AdamW and a learning rate of 0.0005; and Swin-Tiny is trained for 300 epochs with batch size 128 using AdamW and a learning rate of 0.0005. For all three architectures, we set $K = 15$, $\kappa = 10$, and $\lambda = 0.2$.

D.2 Hyperparameter Settings

The main hyperparameters of the proposed method are the number of hard negatives K, the concentration coefficient κ, the regularization weight λ, and the momentum coefficient β.

Unless otherwise specified, we use the same hyperparameter configuration across all models. The default values are selected using the validation performance on the corresponding training setting and are then fixed for the remaining experiments. For the ImageNet-1K experiments, the values of K, κ, and λ follow Table 4, i.e., $K = 15$, $\kappa = 10$, and $\lambda = 0.2$ for ResNet-18, DeiT-Tiny, and Swin-Tiny.

E Results across Architectures

To evaluate the generality of the proposed regularizer, we report detailed results across different network architectures. The experiments are conducted on ImageNet-1K. The evaluated models include both convolutional and transformer-based architectures, covering different feature dimensions and representational characteristics. As shown in As shown in Table 5, BDCR consistently improves over the baseline on all evaluated architectures. Specifically, ResNet-18 He et al. (2016) improves from 71.63% to 71.88%, DeiT-Tiny Touvron et al. (2021) improves from 74.37% to 74.60%, and Swin-Tiny Liu et al. (2021) improves from 79.13% to 79.79%. These results indicate that the proposed regularizer generalizes well across both CNN and transformer backbones, and that the improvement is more pronounced on Swin-Tiny, which achieves the highest absolute accuracy among the three architectures.

F Ablation Study

We examine the proposed regularizer through controlled experiments on CIFAR-10 and CIFAR-100. Both datasets use ResNet-18 and follow the same training configuration as the main experiments. Unless otherwise stated, the Bhattacharyya regularizer is combined with the cross-entropy objective using $\lambda_{\mathrm{reg}} = 0.2$, and we compare uniform, hard, momentum, and logit-guided negative selection together with the fully continuous soft-graph formulation. The experiments include both fixed-κ and annealed-κ configurations.

Table 6: Representative CIFAR-10 ablation results. "Best" denotes the highest validation Top-1 accuracy observed during training, while "Final" denotes the accuracy at the last epoch.

Method	κ / Schedule	**Epochs**	**Best (%)**	**Final (%)**	**Final CE**
Baseline	–	30	86.89	86.89	0.9018
Baseline	–	50	87.74	87.61	0.8502
Uniform	5	50	88.12	87.98	0.8548
Uniform	7	50	88.22	88.14	–
Hard	5	50	88.07	87.95	0.8498
Hard	10	50	87.92	87.87	–
Logit	5	50	88.27	88.14	0.8499
Momentum	1	50	88.22	88.10	0.8491
Soft graph	1	50	88.10	87.94	0.8511
Momentum	$7 \to 0.1$	50	**88.41**	**88.34**	0.8514

F.1 CIFAR-10

F.1.1 Overall Ablation

Table 6 summarizes the representative 50-epoch results, together with the 30-epoch baseline. The baseline reaches 86.89% after 30 epochs and 87.74% best / 87.61% final after 50 epochs. With longer training, several regularized configurations exceed 88% validation accuracy. Logit-guided sampling with $\kappa = 5$ reaches 88.27% best, uniform sampling with $\kappa = 7$ reaches 88.22%, and momentum sampling with a $7 \to 0.1$ annealing schedule reaches 88.41% best and 88.34% final accuracy.

The regularizer is not dependent on a single negative-selection mechanism: uniform, hard, momentum, and logit-guided sampling all produce competitive validation accuracy, with their best fixed-κ configurations lying within a narrow range. The soft-graph formulation is also competitive despite using a different aggregation mechanism in which all valid class relationships contribute continuously rather than through an explicitly selected negative set. The comparison between 30 and 50 epochs additionally shows that the effect of the regularizer becomes clearer with continued optimization. At 30 epochs the regularized configurations are concentrated around approximately 87%; at 50 epochs several exceed 88%. Short runs are useful for identifying whether a configuration is viable, whereas longer runs provide a clearer comparison of the regularization mechanisms.

F.1.2 Effect of the Concentration Parameter κ

The concentration parameter controls the sharpness of the contrastive log-sum-exp over the positive and negative Bhattacharyya distances. For a positive distance D_c^+ and negative distances $\{D_{cj}^-\}$, the contrastive term has the form

$$\mathcal{L}_c = -\log \frac{\exp(-\kappa D_c^+)}{\exp(-\kappa D_c^+) + \sum_j \exp(-\kappa D_{cj}^-)}. \tag{37}$$

Increasing κ therefore makes the objective increasingly concentrated on the closest competing distributions.

The CIFAR-10 results exhibit a non-monotonic dependence on κ, particularly clear for uniform negative sampling (Table 7). Very small values provide a weakly concentrated contrastive signal, while excessively large values focus the objective too strongly on a small number of negative relationships.

The highest accuracy in this sweep is obtained at $\kappa = 7$, while both the smallest and the largest values perform less favorably. A similar tendency appears for other negative-selection mechanisms. Hard sampling reaches 87.95% at $\kappa = 5$, compared with 87.87% at $\kappa = 10$ and 87.66% at $\kappa = 100$. For logit-guided sampling, $\kappa = 5$ reaches 88.14%, whereas both $\kappa = 10$ and $\kappa = 100$ give 87.66%.

Table 7: CIFAR-10 sensitivity to the concentration parameter κ for uniform negative sampling at 50 epochs.

κ	**Final Top-1 Accuracy (%)**
0.001	87.81
0.01	87.49
0.5	87.76
2	87.76
5	87.98
7	**88.14**
10	87.64
100	87.48

F.1.3 Fixed Versus Annealed κ

The preceding experiment suggests that the appropriate concentration of the contrastive objective can change during optimization. A relatively concentrated signal can be useful while class distributions are still substantially overlapping, whereas a broader signal can remain useful once the representation has become more separated. We therefore investigate annealing κ rather than keeping it fixed throughout training.

The strongest effect is observed for momentum-based negative selection. With fixed $\kappa = 1$, momentum sampling reaches approximately 88.1% final accuracy; annealing from 7 to 0.1 produces 88.34% final and 88.41% best validation accuracy, while a $10 \to 0.1$ schedule reaches 88.12% final. The effect is not identical for every sampling mechanism. Uniform sampling with a $5 \to 0.1$ schedule reaches 87.98%, whereas the $10 \to 0.1$ schedule reaches 87.80%. For logit-guided sampling the corresponding final accuracies are 88.00% and 88.06%. Annealing is therefore not a universally beneficial replacement for fixed κ; the schedule can interact with the mechanism used to construct the negative set.

F.1.4 Effect of Negative-Sampling Strategy

We next compare the mechanisms used to construct the negative class set. Uniform sampling selects competing classes without using the current distributional geometry. Hard sampling selects the closest classes according to the current Bhattacharyya distances. Momentum sampling uses an EMA-smoothed pairwise graph to stabilize the hard-negative topology, while logit-guided sampling uses the classifier's mean logits to identify classes that are currently most confusable.

Despite these differences, the fixed-κ results remain relatively close. Uniform sampling with $\kappa = 7$ reaches 88.22% best, hard sampling with $\kappa = 5$ reaches 88.07%, momentum sampling with $\kappa = 1$ reaches 88.22%, and logit-guided sampling with $\kappa = 5$ reaches 88.27%. The soft-graph formulation reaches 88.10% best at $\kappa = 1$. Because the negative-selection mechanism determines where the regularization signal is concentrated but the separation objective remains the same Bhattacharyya contrastive loss, the similarity across selection rules indicates that the formulation is not tied to a single definition of a hard negative.

The momentum variant provides an additional form of temporal smoothing: its negative graph is constructed from an EMA of previously observed class-pair distances rather than from the noisy pairwise distances of a single minibatch. The annealed momentum experiment further shows that graph stability and concentration control can be combined.

F.1.5 Soft-Graph Regularization

The soft-graph formulation provides a complementary ablation because it removes the discrete negative-selection step. For each class, the Bhattacharyya distances to all other valid classes are combined through a temperature-controlled soft minimum,

$$\operatorname{softmin}_\kappa\{D_{cj}\} = -\frac{1}{\kappa}\log\left(\frac{1}{|\mathcal{N}_c|}\sum_{j\neq c}\exp(-\kappa D_{cj})\right). \tag{38}$$

As κ increases, this expression increasingly emphasizes the nearest competing distribution, while every valid pair continues to contribute to the objective. On CIFAR-10 the soft-graph formulation obtains 88.10% best and 87.94% final accuracy at $\kappa = 1$; at $\kappa = 0.75$ and $\kappa = 2$ the final accuracies are 87.74% and 87.93%. These results place the continuous formulation in the same general performance range as the sampled contrastive variants. The two objectives have different normalizations and aggregate class relationships differently, so the appropriate common comparison is classification performance and training behavior rather than the absolute magnitude of the regularization loss.

F.1.6 Training and Validation Behavior

The validation improvement is not explained by increased training accuracy. At 50 epochs the baseline reaches approximately 84.47% training accuracy and 87.61% validation accuracy. The momentum $7 \to 0.1$ configuration reaches approximately 84.40% training accuracy while obtaining 88.34% validation accuracy, and the logit-guided $\kappa = 5$ configuration reaches approximately 84.29% training accuracy with 88.14% validation accuracy. The regularized models therefore do not improve simply by fitting the training set more aggressively; in these experiments some regularized configurations have slightly lower training accuracy while obtaining higher validation accuracy. This is consistent with the intended role of the Bhattacharyya regularizer: the classification loss is optimized jointly with a constraint on the geometry of the class-conditional feature distributions.

The comparison also helps distinguish the regularization effect from an optimization-duration effect. The baseline itself improves from 86.89% at 30 epochs to 87.61% at 50 epochs, so the improvement of the regularized models cannot be attributed solely to training longer. Additional training instead reveals differences between configurations that are less pronounced at 30 epochs.

F.1.7 Summary

The CIFAR-10 ablation identifies three observations. First, the effect of Bhattacharyya regularization becomes more apparent with longer training. Second, the regularizer remains effective under different definitions of a negative class; uniform, hard, momentum, and logit-guided sampling all produce competitive results, while the soft-graph formulation provides a continuous alternative that avoids explicit negative selection. Third, the concentration parameter exhibits a non-monotonic effect, with moderate values providing a more favorable operating regime than either very small or very large values. Among the configurations evaluated, momentum sampling with κ annealed from 7 to 0.1 obtains the highest observed validation accuracy, reaching 88.34% final and 88.41% best. This result is the outcome of the evaluated ablation rather than evidence that momentum sampling is intrinsically superior to all other negative-selection mechanisms; the broader finding is that distribution-level Bhattacharyya regularization remains competitive across several selection strategies, while the concentration and its schedule have a measurable effect on the resulting representation.

F.2 CIFAR-100

CIFAR-100 provides a substantially more challenging setting than CIFAR-10, since the number of classes increases from 10 to 100. The larger class space produces a richer set of inter-class relationships and makes it useful to examine the robustness of the negative-selection mechanism and the sensitivity to the concentration parameter κ.

F.2.1 Overall Behavior

The strongest CIFAR-100 configurations reach approximately 82.94% best validation Top-1 accuracy. In particular, the fixed $\kappa = 10$ configurations using hard-negative and momentum sampling reach this value. Uniform sampling reaches 82.84% at $\kappa = 2$, the logit-guided configuration reaches 82.89% at $\kappa = 10$, and the soft-graph formulation reaches 82.79% at $\kappa = 1$. These results are summarized in Table 8.

The different negative-selection mechanisms remain close despite their substantially different construction rules. Hard sampling selects classes using the current Bhattacharyya geometry, momentum sampling stabilizes this topology with an EMA graph, uniform sampling does not use the geometry for selection, and logit sampling uses classifier confusion rather than Bhattacharyya distance.

Table 8: Representative CIFAR-100 results for the evaluated Bhattacharyya regularization configurations. Values correspond to the best validation Top-1 accuracy reported for each configuration.

Negative selection	κ / Schedule	**Best (%)**
Hard	10 fixed	82.94
Momentum	10 fixed	82.94
Uniform	2 fixed	82.84
Logit	10 fixed	82.89
Soft graph	1 fixed	82.79

The comparable validation accuracies indicate that the regularization effect is not restricted to one particular negative-selection mechanism.

F.2.2 Sensitivity to the Concentration Parameter

The CIFAR-100 experiments also reveal a clear dependence on κ. For small κ, the loss distributes its contribution relatively broadly over the available negative classes; increasing κ concentrates the contribution on the closest competing distributions. In the limiting case the negative term approaches a hard minimum,

$$-\frac{1}{\kappa}\log\left(\exp(-\kappa D_+)+\sum_j \exp(-\kappa D_j^-)\right) \longrightarrow \min\{D_+, D_1^-, D_2^-, \ldots\}, \tag{39}$$

up to the corresponding normalization and sign convention.

The useful operating region is not located at the extreme end of this scale. The best uniform configuration occurs at $\kappa = 2$ (82.84%), and the best soft-graph configuration occurs at $\kappa = 1$ (82.79%), whereas the hard-negative and momentum variants remain effective at the larger value $\kappa = 10$, both reaching 82.94%. This difference between sampling mechanisms indicates that there is no single κ value that can be interpreted as universally optimal across all formulations. Instead, the effective concentration depends on how the negative set itself is constructed. Hard and momentum sampling already restrict the objective to difficult negative relationships, allowing a relatively large κ to further concentrate the signal. Uniform sampling exposes the objective to a broader negative set, for which a smaller κ provides a less aggressive concentration.

F.2.3 Negative-Sampling Mechanisms

The CIFAR-100 results provide a useful comparison of the four discrete negative-selection mechanisms. Uniform sampling provides a simple reference in which negative classes are selected without using either the classifier output or the distributional distance. Hard-negative sampling instead selects the closest classes according to the current Bhattacharyya geometry. Momentum sampling modifies the latter mechanism by maintaining an EMA-smoothed pairwise distance graph whose purpose is to reduce the batch-to-batch instability of exact hard-negative selection: a class pair that is close in one minibatch but temporarily moves farther away because of sampling noise does not immediately disappear from the negative graph, while the loss continues to use the current differentiable Bhattacharyya distances for gradient computation. The CIFAR-100 results show that momentum remains competitive with direct hard-negative selection; both reach 82.94% at $\kappa = 10$. This similarity is informative because the two mechanisms differ primarily in the stability of graph construction, so the benefit of distributional negative selection does not require the graph topology to be recomputed from scratch at every iteration.

Logit-guided sampling provides a different comparison. The classifier determines which classes are currently confused, while the Bhattacharyya distance determines the regularization geometry applied to those selected relationships. The CIFAR-100 logit configuration reaches 82.89% at $\kappa = 10$. Its performance being close to both hard and momentum sampling suggests that classifier confusion and distributional proximity identify partially overlapping sets of difficult class relationships.

F.2.4 SOFT-GRAPH FORMULATION

The soft-graph formulation provides a complementary experiment because it does not require an explicitly selected set of hard negatives. Instead, the pairwise distances are aggregated continuously through a temperature-controlled log-sum-exp operation. For a class c, a representative soft aggregation is

$$D_c^{\text{soft}} = -\frac{1}{\kappa} \log \left[\sum_{j \neq c} w_{cj} \exp(-\kappa D_{cj}) \right], \tag{40}$$

where w_{cj} denotes the corresponding normalization or weighting factor. As κ increases, the aggregation increasingly emphasizes the smallest Bhattacharyya distances. On CIFAR-100 the soft-graph configuration with $\kappa = 1$ reaches 82.79% best validation accuracy, close to the sampled contrastive configurations despite the absence of explicit hard-negative selection. The result indicates that the underlying distributional regularization signal remains effective when the discrete graph-selection step is replaced by a continuous aggregation. As on CIFAR-10, the soft-graph objective should be interpreted separately from the sampled contrastive configurations because the two have different normalizations and aggregate class relationships differently.

F.2.5 EFFECT OF ANNEALING

We additionally investigate schedules in which κ is progressively reduced during training, allowing the regularizer to change from a highly concentrated signal to a broader distributional constraint as optimization proceeds. The CIFAR-100 experiments do not indicate that annealing universally improves over fixed-κ training. Its effect depends on the negative-selection mechanism and the initial concentration. The fixed hard and momentum configurations at $\kappa = 10$ already reach 82.94%, the fixed uniform configuration reaches 82.84% at $\kappa = 2$, and the soft-graph configuration reaches 82.79% at $\kappa = 1$; in each case the fixed value already provides a strong result when it is appropriately matched to the mechanism used to construct the negative set. Annealing is therefore best regarded as a mechanism for controlling the temporal concentration of the regularizer rather than as an independent source of improvement.

F.2.6 COMPARISON WITH CIFAR-10

The CIFAR-100 experiments complement the CIFAR-10 ablation. On CIFAR-10 the strongest observed configuration uses momentum negative sampling with a $7 \to 0.1$ κ schedule, reaching 88.34% final and 88.41% best accuracy. On CIFAR-100, in contrast, strong fixed-κ configurations are already obtained: hard and momentum sampling at $\kappa = 10$ reach 82.94%, uniform sampling at $\kappa = 2$ reaches 82.84%, and logit sampling at $\kappa = 10$ reaches 82.89%. The difference is consistent with the broader class structure of CIFAR-100. With 100 classes, each minibatch contains a richer collection of potentially confusing class relationships, and the negative-selection mechanism becomes more consequential. At the same time, the method does not require a single particular construction of the negative graph: several distinct mechanisms reach closely clustered validation accuracies. Taken together, the CIFAR-10 and CIFAR-100 experiments support a general interpretation in which the central component is the distribution-level comparison of class-conditional feature statistics, while negative sampling and κ determine how this comparison is focused during optimization. The appropriate concentration can depend on the negative-selection mechanism and dataset, and the empirical results do not support treating one fixed κ or one graph-construction rule as universally optimal.

F.2.7 SUMMARY

The CIFAR-100 experiments provide three observations. First, the method remains effective across a substantially larger class space, with multiple configurations reaching approximately 82.8–82.9% best validation accuracy. Second, the negative-selection mechanism changes the useful concentration regime: uniform sampling performs strongly at $\kappa = 2$, whereas hard and momentum sampling perform strongly at $\kappa = 10$. Third, the soft-graph formulation remains competitive without explicit discrete negative selection. Taken together, these results suggest that the Bhattacharyya regularizer is best viewed as a general distributional constraint whose behavior is controlled jointly by the geom-

Table 9: Best validation Top-1 accuracy (%) at 250 epochs on ImageNet with ResNet-18, pretrained regime. "Fixed" denotes constant κ; "annealed" denotes a cosine schedule from the listed start value to 0.1. The best result in each column is shown in bold.

	3-GPU		4-GPU	
Configuration	Best	Final	Best	Final
Baseline	71.88	71.83	72.00	71.95
Uniform, $\kappa = 2$	72.20	72.13	72.20	72.13
Hard, $\kappa = 10$	**72.28**	**72.28**	72.20	72.16
Momentum, $\kappa = 10$	72.13	72.11	72.24	72.17
Momentum, $10 \to 0.1$	72.09	72.01	**72.34**	**72.30**

etry of the class statistics, the mechanism used to identify competing classes, and the concentration parameter κ.

G IMAGENET RESULTS

We evaluate the framework on ImageNet with ResNet-18. The experiments cover four negative-sampling strategies—uniform, hard, momentum, and logit—at a small number of fixed values of the concentration parameter κ, together with κ-annealing schedules. We report results in two training regimes: fine-tuning from a pretrained checkpoint and training from random initialization. To check reproducibility across hardware, the pretrained configurations were run on both a 3-GPU and a 4-GPU server. Unless stated otherwise, we report best validation Top-1 accuracy over the run.

G.1 PRETRAINED REGIME AT 250 EPOCHS

Table 9 reports the principal 250-epoch results. The baseline reaches 71.88% and 72.00% on the 3-GPU and 4-GPU servers, respectively. Regularized configurations reach 72.20–72.28% on the 3-GPU server and 72.20–72.34% on the 4-GPU server.

The best fixed-κ result is 72.28%, obtained by hard sampling with $\kappa = 10$ on the 3-GPU server. On the 4-GPU server the best fixed-κ result is 72.24% (momentum, $\kappa = 10$), and the best overall 250-epoch result is 72.34%, obtained by annealing κ from 10 to 0.1 with momentum sampling.

The absolute gains over the baseline are therefore small—between 0.2 and 0.5 percentage points—but they are consistent across the four sampling strategies and both servers. Every regularized configuration at 250 epochs exceeds its corresponding baseline.

G.2 EFFECT OF TRAINING DURATION

The regularization gain grows with training duration and stabilizes once training exceeds 150 epochs. Because the two servers were run at overlapping but not identical sets of durations (the 3-GPU server covers 150, 250, and 350 epochs; the 4-GPU server covers 150, 250, and 500 epochs), we report the duration dependence *separately* for each server: Table 10 for the 3-GPU server and Table 11 for the 4-GPU server. Mixing durations from the two servers in a single table would confound the effect of training length with the effect of hardware, so no row in either table uses data from the other server. Figure 2 shows the same data as a function of training budget.

Table 10 reports the 3-GPU results. The baseline reaches 71.52%, 71.88%, and 71.99% at 150, 250, and 350 epochs, while the best regularized configuration reaches 71.82%, 72.28%, and 72.27%, giving gains of 0.30, 0.40, and 0.28 percentage points. On this server the best configuration is hard sampling with $\kappa = 10$ at both 250 and 350 epochs, and momentum annealing at 150 epochs.

Table 11 reports the 4-GPU results. The baseline reaches 71.63%, 72.00%, and 72.20% at 150, 250, and 500 epochs, while the best regularized configuration reaches 72.03%, 72.34%, and 72.53%, giving gains of 0.40, 0.34, and 0.33 percentage points. On this server the best configuration is momentum annealing at 150 and 250 epochs, and hard sampling with $\kappa = 10$ at 500 epochs.

Table 10: Training-duration dependence on the **3-GPU** server. Best validation Top-1 accuracy (%) at each duration. All rows use data from the same server so the comparison across durations is not confounded with a hardware change.

Epochs	Baseline	Best fixed-κ	Best annealed	Best overall
150	71.52	71.80 (mom., κ=10)	71.82 (mom., $10 \to 0.1$)	71.82
250	71.88	72.28 (hard, κ=10)	72.09 (mom., $10 \to 0.1$)	**72.28**
350	71.99	72.27 (hard, κ=10)	72.18 (mom., $10 \to 0.1$)	72.27

Table 11: Training-duration dependence on the **4-GPU** server. Best validation Top-1 accuracy (%) at each duration. All rows use data from the same server so the comparison across durations is not confounded with a hardware change.

Epochs	Baseline	Best fixed-κ	Best annealed	Best overall
150	71.63	71.74 (hard, κ=10)	72.03 (mom., $10 \to 0.1$)	72.03
250	72.00	72.24 (mom., κ=10)	72.34 (mom., $10 \to 0.1$)	**72.34**
500	72.20	72.53 (hard, κ=10)	—	72.53

Across both servers the gain is in the range 0.28–0.40 percentage points once training exceeds 150 epochs. This is consistent with the intended role of the regularizer as a constraint on the learned representation: the constraint becomes useful as the representation matures, and it does not degrade at longer horizons.

G.3 Effect of Sampling Strategy and κ

For fixed κ, the useful configurations concentrate around moderate values of the concentration parameter: $\kappa = 2$ for uniform sampling and $\kappa = 10$ for hard and momentum sampling. At 250 epochs these configurations reach 72.20–72.28% on the 3-GPU server and 72.20–72.24% on the 4-GPU server.

Hard and momentum sampling perform comparably across all durations and both servers. Hard sampling obtains the best 3-GPU fixed-κ results at 250 and 350 epochs (72.28% and 72.27%) and the best 4-GPU fixed-κ result at 500 epochs (72.53%); momentum sampling obtains the best 4-GPU fixed-κ result at 250 epochs (72.24%). The two are within the run-to-run spread observed across the two servers. The improvement therefore does not depend on a unique negative-sampling mechanism: several mechanisms provide comparable gains when combined with the Bhattacharyya objective.

G.4 Effect of κ Annealing

Annealing κ from a large initial value to 0.1 provides a small additional improvement at intermediate horizons, but its advantage disappears at the longest ones.

On the 4-GPU server, momentum sampling with fixed $\kappa = 10$ reaches 71.85% at 150 epochs and 72.24% at 250 epochs, while the annealed $10 \to 0.1$ schedule reaches 72.03% and 72.34%, respectively. At 500 epochs on the same server, hard sampling with fixed $\kappa = 10$ reaches 72.53%, which exceeds any annealed configuration.

On the 3-GPU server the picture is more mixed. At 150 epochs fixed and annealed momentum differ by less than the run-to-run spread (71.80% versus 71.82%). At 250 and 350 epochs the best fixed-κ configuration (hard sampling, 72.28% and 72.27%) exceeds the best annealed configuration (momentum, 72.09% and 72.18%). On this server annealing does not improve on the best fixed-κ configuration at any of the durations we tested.

The benefit of annealing is therefore length- and server-dependent. It helps on the 4-GPU server at 150 and 250 epochs, is neutral on the 3-GPU server at 150 epochs, and is not the winning configuration at any long horizon on either server.

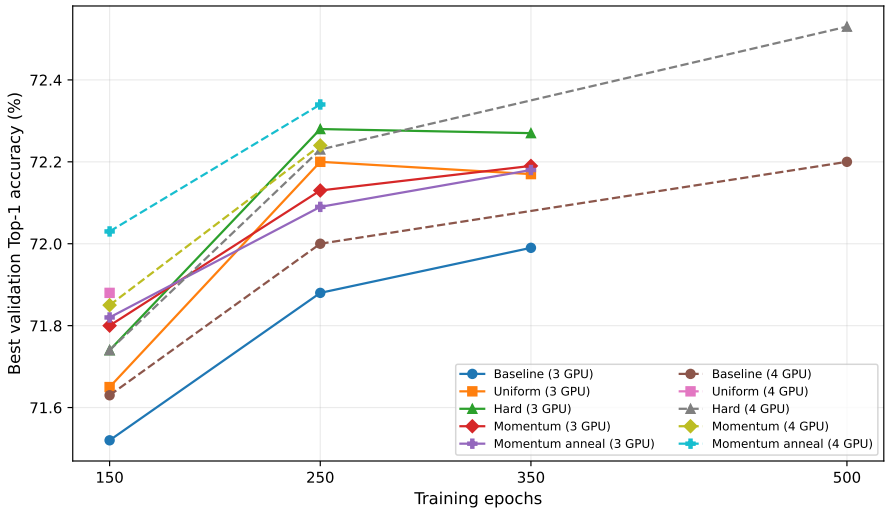

Figure 2: Best validation Top-1 accuracy as a function of training budget on ImageNet with ResNet-18, pretrained regime. Solid lines are the 3-GPU server; dashed lines are the 4-GPU server. Markers distinguish the sampling strategies. Each point is the best accuracy observed for that (server, budget, strategy) family; the exact winning configuration is given in Tables 10 and 11.

G.5 Cross-Server Consistency

The two servers provide a reproducibility check. At 250 epochs—the longest duration present on both servers—the baselines differ by 0.12 percentage points (71.88% on 3-GPU versus 72.00% on 4-GPU), and the regularized configurations remain within a similarly narrow band. For example, hard sampling with $\kappa = 10$ gives 72.28% on 3-GPU versus 72.20% on 4-GPU; momentum with $\kappa = 10$ gives 72.13% versus 72.24%; and annealed momentum gives 72.09% versus 72.34%. The two servers differ by less than the run-to-run variation observed within a single server at the configurations for which replicates are available, so the main conclusions are not tied to a particular GPU configuration.

G.6 Validation Accuracy versus Training Accuracy

The validation improvements are not explained by higher training accuracy. On the 4-GPU server at 250 epochs, the baseline reaches 74.48% final training accuracy while the best momentum-annealed run reaches 72.33%, yet the corresponding validation accuracies are 72.00% and 72.34%. The same pattern holds at 500 epochs on the same server: the baseline reaches 75.80% final training accuracy, while hard sampling with $\kappa = 10$ reaches 72.94% final training accuracy and yet obtains 72.53% best validation accuracy against the baseline's 72.20%.

The regularized models therefore achieve their validation performance while fitting the training set *less* well than the baseline, which is consistent with the intended role of the Bhattacharyya regularizer as a constraint on the representation rather than an additional fitting mechanism.

G.7 Training from Scratch

We additionally ran a set of configurations from random initialization to check whether the regularization gains persist when the representation must be learned from scratch. These runs use the same ResNet-18 backbone and the same optimization settings as the pretrained experiments and were run on a single 4-GPU server.

Table 12 reports the 150-epoch from-scratch results. The from-scratch baseline reaches 69.86%. Regularized configurations reach 69.94–70.22%, with momentum sampling at $\kappa = 10$ giving the best result (70.22%). The spread across regularized configurations is small (0.28 percentage points), but every contrastive configuration exceeds the baseline.

Table 12: Best validation Top-1 accuracy (%) at 150 epochs on ImageNet with ResNet-18, training from random initialization (4-GPU server). The best result is shown in bold.

Configuration	Best	Final
Baseline	69.86	69.82
Uniform, $\kappa = 2$	70.14	70.05
Hard, $\kappa = 5$	69.94	69.91
Hard, $\kappa = 10$	69.96	69.96
Hard, $\kappa = 20$	70.13	70.08
Momentum, $\kappa = 5$	70.09	70.01
Momentum, $\kappa = 10$	**70.22**	70.15
Momentum, $10 \rightarrow 0.1$	70.06	69.98
Logit, $\kappa = 10$	70.15	70.15

The regularization gain is comparable in the two regimes: 0.40 percentage points for the best pretrained configuration at 150 epochs and 0.36 for the best from-scratch configuration. The regularization therefore does not depend on the pretrained initialization for its effect.

One difference is worth noting. In the pretrained regime, the best 150-epoch configuration is momentum with κ annealing, which exceeds fixed momentum by 0.18 percentage points. In the from-scratch regime the ordering is reversed: fixed momentum at $\kappa = 10$ (70.22%) exceeds annealed momentum (70.06%). Given the run-to-run spread and the single seed per configuration, we do not treat this reversal as significant. It does, however, reinforce the observation from the pretrained experiments that the benefit of annealing is configuration-dependent rather than universal.

G.8 Summary

The ImageNet experiments support four observations. First, in the pretrained regime the proposed regularization produces small but consistent gains over the baseline at every training duration we study (150, 250, 350, and 500 epochs), on both hardware configurations; measured within each server, the gain is in the range 0.28–0.40 percentage points once training exceeds 150 epochs. Second, these gains are not specific to a single negative-sampling mechanism: uniform, hard, and momentum sampling all exceed their respective baselines, with hard and momentum sampling giving the strongest results. Third, the best configuration depends on training horizon and server: κ annealing is the winning choice on the 4-GPU server at 150 and 250 epochs, while fixed $\kappa = 10$ is the winning choice on the 3-GPU server at 250 and 350 epochs and on the 4-GPU server at 500 epochs, where hard sampling reaches 72.53% Top-1 accuracy. Fourth, the regularization gain persists when training from random initialization, with the best from-scratch configuration reaching 70.22% against a from-scratch baseline of 69.86%. The agreement between the 3-GPU and 4-GPU pretrained experiments at the durations they share supports the reproducibility of these observations.